Kinematic Motor Learning

Wolfram Schenck

Computer Engineering Group
Faculty of Technology
Bielefeld University
Universitätsstr. 25, D-33615 Bielefeld, Germany

phone: +49-521-106-6162
fax: +49-521-106-6440
mail: wschenck@ti.uni-bielefeld.de

Abstract

This paper focuses on adaptive motor control in the kinematic domain. Several motor learning strategies from the literature are adopted to kinematic problems: “feedback-error learning”, “distal supervised learning”, and “direct inverse modeling”. One of these learning strategies, direct inverse modeling, is significantly enhanced by combining it with abstract recurrent neural networks. Moreover, a newly developed learning strategy (“learning by averaging”) is presented in detail. The performance of these learning strategies is compared on different learning tasks on two simulated robot setups (a robot-camera-head and a planar arm). The results indicate a general superiority of direct inverse modeling if combined with abstract recurrent neural networks. Learning by averaging is especially successful if the motor task is constrained by special requirements.

KEYWORDS: Motor Learning, Internal Models, Neural Networks, Kinematics, Robotics

PREPRINT

— TO APPEAR IN THE JOURNAL “CONNECTION SCIENCE” —
1 Introduction

Motor control is a general requirement for any agent which is equipped with some kind of actuators. These have to be moved in a way that the needs and goals of the agent get fulfilled. Motor control serves to generate the corresponding movement commands, both in biological organisms and in artificial agents like robots. In the framework of internal models for sensorimotor processing (Kawato, 1999; Lalazar and Vaadia, 2008; Miall and Wolpert, 1996; Wolpert et al., 1998), so-called “inverse models” (IM) serve as motor controllers. They generate the motor commands which are necessary to minimize the difference between the current and a desired sensory state. In addition, predictive “forward models” (FM) belong to this framework. They anticipate future sensory states as the consequence of the current sensory state and a motor command of the agent.

Whenever internal models are accepted as building blocks of sensorimotor processing, the question arises how they are acquired. For biological organisms, internal models represent certain information processing capabilities of the central nervous system (CNS) (Schaal and Schweighofer, 2005; Shadmehr and Wise, 2005). Such capabilities can either be inherited or learned during the lifetime of the organism. But even inheritance involves learning by evolutionary forces in the process of phylogenesis. In robotics, a classical engineering approach would try to describe the “plant” (which comprises the robot and its environment) analytically as precise as necessary. Knowing the plant, one already knows the FM, and IMs (motor controllers) can be designed using the tools of control theory (Dorf and Bishop, 2004). But such an approach is infeasible for autonomous robots in complex and changing environments (where the “body” of the robot might change as well due to wear and tear). Thus, not knowing the plant beforehand, the robot has to adapt its internal models to the environment in which it is moving and acting — it has to learn. In summary, the learning of internal models is an integral part of sensorimotor processing, both for biological and artificial agents.

This paper focuses on learning strategies for inverse models of kinematic relationships. Several motor learning strategies from the literature are adopted to kinematic problems: “feedback-error learning” (FEL) (Kawato, 1990), “distal supervised learning” (DSL) (Jordan and Rumelhart, 1992), and “direct inverse modeling” (DIM) (Kuperstein, 1988). One of these learning strategies, DIM, is significantly enhanced by combining it with abstract recurrent neural networks (Möller and Hoffmann, 2004). Moreover, a newly developed learning strategy (“staged learning by averaging”; SLbA) is presented in detail. The performance of these learning strategies is compared on different learning tasks on two simulated robot setups (a robot-camera-head and a planar arm). The results of these experimental studies contribute to the fields of biological cybernetics and adaptive robotics.

Section 2 starts with a formal description of internal models. It continues with an overview of the general problems of kinematic motor learning and with a description of all tested learning strategies. In Sect. 3, the comparison study on motor learning for the robot-camera-head is presented, in Sect. 4 the study for the planar arm. The results are discussed in Sect. 5.
Figure 1 — The plant as abstract representation of the physical system. The box labeled $D$ indicates a delay by one time step.

Figure 2 — Left: Forward model (the output is either a prediction of the system state $\hat{x}_{t+1}$ or the sensory output $\hat{y}_{t+1}$ of the system in the next time step). Right: Inverse model.

2 Adaptive Motor Control

2.1 The framework of internal models

The following description of internal models relies on the formalism of state-space control theory for dynamical systems (Dorf and Bishop, 2004). This general approach will later be transferred to the kinematic domain. At a given time step $t$, the physical system consisting of the agent and its environment is in a certain state $x_t$. The external input of the system in each time step is the vector $u_t$. In motor control, it is the motor command which is generated and executed by the agent. The output of the system is denoted by the vector $y_t$. In the domain of sensorimotor processing, $y_t$ is usually identical with the sensor readings of the agent (or with a more abstract sensory state description derived from these readings). Generally, the output $y_t$ is a function of the current system state $x_t$: $y_t = h(x_t)$. The so-called “plant” serves as an abstract representation of the physical system. Figure 1 shows a block diagram of the important relationships. The system itself is represented by the plant $P'$:

$$x_{t+1} = P'(x_t, u_t)$$

The measurement process yields

$$y_{t+1} = h(x_{t+1}) = h(P'(x_t, u_t)) .$$

For convenience, the plant $P$ is defined which combines $P'$ and the measurement process:

$$y_{t+1} = P(x_t, u_t)$$

(1)

As already stated, there are basically two classes of internal models: forward models (FM) and inverse models (IM). FMs mimic the behavior of the plant and generate a prediction of $x_{t+1}$ or
Figure 3 — Simplified motor control scheme (the internal state loop of the plant as in Fig. 1 is omitted for simplicity; instead, the state $x_t$ is depicted as input of the plant).

$y_{t+1}$ (see Fig. 2, left):

$$\hat{x}_{t+1} = FM(x_t, u_t) \quad \text{or} \quad \hat{y}_{t+1} = FM(x_t, u_t)$$

IMs represent the inverse relationship. While an FM is an approximation of the plant, an IM is an approximation of the inverse of the plant and acts as motor controller (see Fig. 2, right):

$$u_t = IM(x_t, y^*) \quad \text{or} \quad u_t = IM(y_t, y^*)$$

In these equations, $u_t$ is an estimate of the motor command which would best approach the desired plant’s response $y^*$ of the next time step.

### 2.2 Kinematic motor control

In a very general sense, the term “kinematics” refers to a transformation between coordinate systems (Jordan, 1996). A classical example from engineering is the transformation between joint angles $\theta$ of a robot arm and the position and orientation of its gripper tip in the world coordinate system. This relationship is often referred to as “forward kinematics” while the inverse relationship is designated as “inverse kinematics” (Spong and Vidyasagar, 1989).

For motor control, the inverse kinematics is required to determine the correct posture for a given desired limb position and limb orientation. For example, in a reaching task, the target is first represented in retinal (eye-centered) coordinates. These coordinates have to be transformed first into a head-centered and finally into a body-centered representation (Battaglia-Mayer et al., 2003; Buneo et al., 2002; Carrozzo et al., 1999; Snyder, 2000). From the body-centered coordinates, the final reaching posture of the arm (the joint angles) can be generated by an inverse kinematics model. Actually, there is a large amount of experimental evidence from neurophysiological and psychophysical studies on humans and primates that these coordinate transformations take place in the CNS (for a review, see Battaglia-Mayer et al., 2003). Very often, the input-output relationship of the inverse kinematics is a one-to-many mapping. For example, to grasp for a cup on the desk before them, humans can use a large variety of different final arm postures. Nevertheless, many human movements and final postures are rather stereotype, thus the CNS seems to prefer certain solutions of the inverse kinematics (Cruse et al., 1990; Grea et al., 2000; Rosenbaum et al., 2001).

In the robot arm example, the kinematic relationship is a direct transformation between the joint angles $u$ and the workspace coordinates $y$. There is no system state $x$ which evolves over time;
thus, the “plant” has the simplified form $y = P(u)$. This implies that any desired sensory state $y^*$ can either be achieved by a single motor command (in contrast to a series of motor commands) or not at all. In the following, this property is applied to plants of the more general form $y_{t+1} = P(x_t, u_t)$ from the previous section. Let $Y$ be the image of the plant function $P$ and $X$ the set of all possible system states. A plant is defined as a kinematic one if for any pair $(y_{t+1}, x_t) \in Y \times X$ a vector $u_t$ exists such that $y_{t+1} = P(x_t, u_t)$. For a kinematic plant defined in the time-discrete domain in this way, a first-order dead-beat controller exists whose motor output $u_t$ can drive the output $y_{t+1}$ of the plant within one time step to the desired value $y^*$ irrespective of the state $x_t$.

Figure 3 shows a slightly simplified control scheme combining an inverse kinematics model with a kinematic plant. A perfectly adapted inverse kinematics model would operate like the aforementioned dead-beat controller. Since the correct motor output $u_t$ directly depends on $y^*$ and $x_t$, inverse kinematics models can be acquired by supervised learning techniques, notwithstanding specific problems which are discussed in the next section. In contrast, non-kinematic plants may require a series of motor commands $\{u_t\}$ to get from a state $x_0$ to the desired output $y^*$. The learning of the corresponding dynamic inverse models belongs to the field of optimal control for which techniques from the field of reinforcement learning are well suited (Doya, 2000; Wolpert and Flanagan, 2003).

Since this paper addresses rather abstract plants above the level of physical state variables, in the following $y$ is interpreted as a set of variables which is used to define the desired plant output and $x$ as (sensory) context information.

### 2.3 Kinematic motor learning

#### 2.3.1 Problems of kinematic motor learning

By motor learning, adaptive motor controllers are acquired. Motor learning is goal-oriented and aims on a certain effect in the external world or on the agent itself. This effect is first measured and specified in the sensory domain. For example, visual and tactile information indicate the failure or success of a human grasping movement. However, despite this sensory feedback the correct motor command remains unknown. To specify this problem in a more formal way, the starting point is the simplified motor control scheme in Fig. 3. Instead of the term “inverse kinematics model”, the shorter term “controller” is used in the following (abbrev.: $C$). The controller equation is $u_t = C(x_t, y^*)$. Ideally, the controller is the inverse of the plant (with regard to $u$ and $y$). For an untrained adaptive controller, there remains a residual $\Delta y = y^* - y_{t+1}$ after a movement. While this sensory error is measurable, the corresponding $\Delta u$ for which $\Delta y = 0$ is not directly accessible. For this reason, an important challenge of motor learning is the transfer of the sensory error $\Delta y$ to an error signal $\Delta u$ in the motor domain. This challenge is called here the “problem of the missing teacher signal”. A very simple solution would be the random exploration of sensorimotor space in the search for motor commands which accomplish the task at hand. Unfortunately, through the high dimensionality of sensorimotor space in natural tasks such an approach is infeasible in practice.

A second problem arises for motor learning if the inverse plant is a one-to-many mapping. This is illustrated in Fig. 4a for a very simple plant mapping from a one-dimensional motor
space $u$ to a one-dimensional sensory space $y$ (omitting the state $x$). To reach $y^*$, two different motor commands $u^*_1$ and $u^*_2$ are suitable. This causes an ambiguity in the motor error. If one considers an untrained controller which produced $u_C = C(y^*)$ leading to $y_C \neq y^*$, two different corrections of its motor output, $\Delta u_1$ and $\Delta u_2$, are possible (see Fig. 4b). Thus, a motor learning procedure either has to guarantee that the controller output converges to one of the possible solutions, or the controller architecture itself has to be capable of storing multiple outputs for one and the same input. The multi-layer perceptron (MLP; Rumelhart et al., 1986) and other function approximators converge instead to the average of $u^*_1$ and $u^*_2$. The resulting motor command has obviously not the desired sensory effect $y^*$. Only if the solution sets $\{u | P(u) = y^*\}$ are convex, function approximators can deal with one-to-many mappings. When I speak in the following of the “problem of one-to-many mappings”, I usually address one-to-many mappings with non-convex solution sets.

In the literature regarding internal models and adaptive controller learning, several approaches have been proposed to circumvent these problems. The most popular are “feedback-error learning” (FEL) (Kawato, 1990), “distal supervised learning” (DSL) (Jordan and Rumelhart, 1992), and “direct inverse modeling” (DIM) (Kuperstein, 1988). In the following, I provide a review of these approaches to motor learning in the context of kinematic control and show the relationship between them. Moreover, I present a novel learning procedure called “staged learning by averaging” (SLbA) (Schenck and Möller, 2004, 2006). In Sects. 3 and 4, the performance of FEL, DSL, DIM, and SLbA is compared on two different learning tasks, concerning active vision and the control of a planar robot arm.

FEL and DSL are closely related; they use a local linear approximation for the unknown mapping from sensory error to motor error space and solve the one-to-many problem by converging to one of the possible solutions. DIM relies on a reformulation of the learning problem. It offers no genuine solution for the one-to-many problem, but the usage of abstract recurrent neural networks as adaptive controller (Hoffmann and Möller, 2003; Möller and Hoffmann, 2004) can overcome this shortcoming. SLbA employs an MLP as adaptive controller and actually exploits its averaging capabilities in the learning process. Learning examples are generated by a heuris-
tic search process. SLbA can only cope with the one-to-many problem as long as additional learning constraints are defined.

To reduce the complexity of figures and equations, the time step indices $t$ and $t + 1$ are omitted in the following for the most part as long as it is possible without any loss in generality or clarity.

### 2.3.2 Feedback-error learning

FEL has been developed in the context of dynamic motor control (Gomi and Kawato, 1993; Jo, 2008; Kambara et al., 2009; Kawato et al., 1987; Kawato, 1990; Nakanishi and Schaal, 2004), but it can also be adopted in modified form to kinematic problems. The control scheme is shown in Fig. 5. In addition to the controller $C$ and the plant $P$, a feedback controller $F$ is displayed. As input, it gets the sensory error $\Delta y$. The feedback controller has the task to convert $\Delta y$ into the motor error $\Delta u$ which is used as error signal for the adaptation of $C$. In contrast to the dynamic version, $u$ and $x$ are additional inputs to $F$ that provide the necessary context information.\(^1\) Usually, $F$ is a linear function:

$$\Delta u = F(\Delta y, u, x) = G_{u,x} \Delta y$$  \hspace{1cm} (2)

$G_{u,x}$ is a gain matrix depending on $u$ and $x$. There are several ways to determine this matrix. If the plant characteristics are only known qualitatively, one has to “guess” the gain factors, especially their sign. For example, in the univariate case a positive value of $\Delta y$ should require either a positive or a negative correction of the motor command $u$, thus $g_{11}$ (the single element of $G_{u,x}$ in this case) has to be chosen correspondingly.

If the plant is an analytically known continuously differentiable function, more sophisticated methods are available. They are related to closed-loop inverse kinematics schemes (CLIK)

---

\(^1\) Furthermore, in contrast to the usual presentation of FEL, the feedback controller $F$ in Fig. 5 only provides the teaching signal for the adaptive controller and does not contribute to the finally executed motor command; this simplification is motivated by the focus on the generation of controller teaching signals in this paper.
(Chiacchio et al., 1991; Siciliano, 1990). In a CLIK scheme, Eqs. (1) and (2) are applied iteratively to minimize $\Delta y$ via $u \leftarrow u + \Delta u$. In this way, a solution for the inverse kinematics is determined. Common CLIK schemes rely on the Jacobian $J_{u,x}$ of the plant at the position $(u, x)$ in motor and state space: $\dot{y} = J_{u,x} u$ (note that $J_{u,x}$ is not the full Jacobian here; it only contains the columns concerning motor space). The time-discrete approximation conforms to:

$$\Delta y = J_{u,x} \Delta u$$  \hspace{1cm} (3)

By solving this equation for $\Delta u$, one obtains the pseudoinverse $J_{u,x}^+ \Delta y$ as gain matrix $G_{u,x}$:

$$\Delta u := \eta J_{u,x}^+ \Delta y$$  \hspace{1cm} (4)

$\eta$ is the approximation rate. Notwithstanding this factor, $\Delta u$ is either the least squares solution to Eqn. (3) (if the problem is overdetermined) or the minimum length solution (if the problem is underdetermined). Equation (4) can also be interpreted as application of the Newton method to find the zeros of the function $f(u; x, y^*) = y^* - P(x, u)$ by varying $u$.

Alternatively, the computationally less expensive transpose $J_{u,x}^t$ can be used as gain matrix:

$$\Delta u := \eta J_{u,x}^t \Delta y$$  \hspace{1cm} (5)

This equation can be derived from gradient descent on a quadratic error function as shown in the next section on DSL. Using an appropriate value for $\eta$, both CLIK schemes are known to converge so that $\Delta y$ finally amounts to zero as long as the iterative approximation does not get stuck in a kinematic singularity (Chiacchio et al., 1991; Slotine and Yoerger, 1987). In the case of a one-to-many mapping between sensory and motor space, a successful CLIK application arrives at one of the distinct solutions of the inverse kinematics. For this reason, CLIK schemes are recommended for redundant manipulators.

Kinematic FEL transfers the idea of CLIK to the adaptation of motor controllers. Instead of directly applying $\Delta u$, the motor error is used to adapt the controller output. Incorporating $\eta$ as learning rate, gain matrices for Eqn. (2) are either $G_{u,x} = \eta J_{u,x}^+ \Delta y$ or $G_{u,x} = \eta J_{u,x}^t$. The appropriate value range for $\eta$ for proper convergence depends on the motor task. Usually, a function approximator like an MLP is used as adaptive controller, and $\eta$ implicitly adjusts the learning rate.

Thus, FEL does not provide a perfect mapping from sensory error to motor error space. Instead, the feedback controller provides a coarse local linear approximation. By adapting along these small local corrections during the course of learning, the MLP output converges to one of the possible solutions for $u$ (in case of a one-to-many problem). Because of this stepwise linear adaptation, FEL is basically an online training procedure. It is not possible to collect a set of training examples for batch learning.\(^2\)

FEL for kinematic problems is rarely used in the literature. Exemplary applications are from the field of biologically inspired active vision systems (Bruske et al., 1997; Dean et al., 1991) where FEL is used for adaptive oculomotor control. If the plant is too complex to define a gain matrix heuristically, FEL is no longer a really adaptive learning scheme since one needs to know

---

\(^2\) Online learning: adaptation of the free parameters of a learning system after the presentation of each single training example. Batch learning: adaptation of the free parameters of a learning system after the presentation of all training examples from a training set.
the plant analytically to compute $G_{u,x}$. Thus, for the biological modeling of kinematic control, FEL is only attractive for rather simple plants like the ones found in oculomotor control. Prewired feedback controllers for such simple plants could have been acquired during evolutionary development.

### 2.3.3 Distal supervised learning

DSL was introduced by Jordan and Rumelhart (1992) for the kinematic control of a planar arm (for subsequent studies employing DSL see Howard and Huckvale, 2005; Jordan et al., 1994; Jordan, 1996; Martin and Millan, 2000). The controller learning scheme of DSL is very similar to FEL as shown in Fig. 6a. The feedback controller $F$ is replaced by a forward model FM. The FM itself approximates the plant (see Fig. 6b): $FM = \hat{P}$. It receives a motor command $u$ and state $x$ and predicts the sensory outcome $y$. For DSL, the FM has to be implemented by an MLP. Training data for the FM is generated by collecting the plant’s response to random motor commands.

For controller training, DSL uses the trained FM in reverse direction (Fig. 6a): The sensory error $\Delta y$ becomes the input, the motor error $\Delta u$ becomes the output. This conversion of error signals is possible by error backpropagation (Rumelhart et al., 1986) without changing the weights of the MLP network for the FM. Backpropagation implements gradient descent on MLPs to minimize the error $E$ of the activation of the units of the output layer. For the sensory output $y$, the error is defined as $E = \frac{1}{2} \| \Delta \hat{y} \|^2$ with $\Delta \hat{y} = y^* - \hat{y}$. $\hat{y}$ is the output of the FM (used in its normal direction). In the following, $N_{out}$ denotes the dimension of the output space (with index $j$). The backpropagated error signal $\delta_i$ for each motor unit $i$ of the input layer is computed as follows:

$$\delta_i = -\frac{\partial E}{\partial u_i}$$
According to the chain rule:

$$\frac{\partial E}{\partial u_i} = \sum_{j=1}^{N_{out}} \frac{\partial E}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial u_i}$$

Substituting for the partial derivatives one obtains:

$$\frac{\partial E}{\partial \hat{y}_j} = - \left( y_j^* - \hat{y}_j \right)$$

$$\frac{\partial \hat{y}_j}{\partial u_i} = \hat{J}_{u,x(ji)} = \hat{J}^t_{u,x(ij)}$$

$$\Rightarrow \frac{\partial E}{\partial u_i} = - \sum_{j=1}^{N_{out}} \left( \hat{J}^t_{u,x(ij)} \cdot (y_j^* - \hat{y}_j) \right)$$

\(\hat{J}_{u,x}\) is the motor part of the Jacobian of the FM (which approximates the plant). Thus, gradient descent in motor space with a step size \(\eta\) based on the backpropagated error signal \(\delta_i\) yields:

$$\Delta u_i = \eta \delta_i = -\eta \frac{\partial E}{\partial u_i}$$

$$= \eta \sum_{j=1}^{N_{out}} \left( \hat{J}^t_{u,x(ij)} \cdot (y_j^* - \hat{y}_j) \right)$$

$$\Leftrightarrow \Delta u = \eta \hat{J}^t_{u,x} \Delta \hat{y} \quad (6)$$

Equation (6) shows that backpropagation through the FM results in a local linear approximation of the motor error with \(\hat{J}^t_{u,x}\) as gain matrix. However, there is still one downside: The learning algorithm would try to change the motor commands \(u\) in a way that the output of the FM finally equals the desired sensory output. But one is actually interested in a close match with the sensory output of the plant, which might differ from the output of the FM. To overcome this problem, Jordan and Rumelhart (1992) replaced \(\Delta \hat{y}\) by \(\Delta y\), the difference between the desired output and the real plant output. Thus, Eqn. (6) changes to

$$\Delta u = \eta \hat{J}^t_{u,x} \Delta y \quad (7)$$

Considering Eqns. (5) and (7), DSL is equivalent to FEL with a gain matrix \(G_{u,x} = \eta \hat{J}^t_{u,x}\). The adaptive motor controller (also an MLP) is trained online by small approximated motor error signals. For one-to-many problems, the controller output converges to one of the possible solutions (Jordan and Rumelhart, 1992).

DSL is truly adaptive because no analytical knowledge about the plant is needed beforehand. Moreover, even with a rather imprecise FM successful controller learning is possible as Jordan and Rumelhart (1992) emphasize. Since the FM is usually trained before the controller, the overall DSL learning scheme corresponds to findings that the ability to predict precedes the ability to control the motor apparatus in human subjects (Flanagan et al., 2003; Sailer et al., 2005). However, the main drawback of DSL for biological modeling is the requirement of backpropagation which is in itself not a biologically plausible neural learning mechanism (Arbib, 1998; Kawato and Gomi, 1992).
Figure 7 — Direct inverse modeling: Random motor commands $u^*$ are generated for controller training (for details see text).

2.3.4 Direct inverse modeling

DIM is based on a reformulation of the motor learning problem (Fig. 7). Instead of searching for the right motor command $u$ for a certain desired sensory output $y^*$ and state $x$, random motor commands $u^*$ are generated. For each $u^*$, the resulting sensory output $y^*$ of the plant is recorded. Afterwards, $y^*$ is interpreted as desired output for which $u^*$ is actually a perfect motor response. The combination $[x, y^* \rightarrow u^*]$ forms a perfect learning example for the training of the motor controller. Due to this characteristic, DIM works both for online and batch learning.

DIM-related learning strategies have been used in various applications, e.g. for visually guided kinematic robot arm control (Kuperstein, 1987, 1988; Martinetz and Schulten, 1993; Mel, 1991) or for the learning of robot arm dynamics (Miller et al., 1990; Vijayakumar and Schaal, 2000). Kuperstein motivated DIM by the “circular reaction” Piaget (1952) observed in children during their development: Children carry out explorative actions in a rather random fashion and observe the sensory effects. In this way, they find out which actions are best suited to obtain these effects. This corresponds very closely to the DIM learning strategy.

Although this psychological motivation seems to be plausible, DIM has three main shortcomings. First, DIM cannot cope with the one-to-many problem as long as the controller is implemented by a function approximator. This was illustrated by Jordan and Rumelhart (1992) for a planar arm (see also Sect. 4.1.1). To overcome this shortcoming, Hoffmann and Möller (2003) used abstract recurrent networks as controller. This type of network approximates high-dimensional sensorimotor data manifolds by a mixture of local principal component analysis (PCA) units (see Sect. A on NGPCA). Each PCA unit can be interpreted in geometrical terms as a hyperellipsoid which covers part of the data manifold. Using such networks, the controller is actually able to reproduce all motor outputs of the one-to-many mapping as long as an appropriate recall procedure is used (Hoffmann, 2004; Hoffmann et al., 2005; Schenck et al., 2003). The learning strategy which combines DIM with NGPCA is called DIM-NGPCA in the following, and the combination of DIM with an MLP as adaptive controller is called DIM-MLP. Whenever the term DIM is used, it addresses both DIM-NGPCA and DIM-MLP.

The second shortcoming is that DIM is not goal-oriented. Random sampling in motor space may elicit various sensory effects, but may rarely hit the ones which are later used as desired sensory outcomes. In this case, the resulting controller has to extrapolate the motor output in the region of sensory space containing the desired outcomes and will most likely exhibit bad
performance.

Third, on the neural implementation level, DIM seems to be biologically rather implausible at the first glance. During learning, the input units of the controller receive the real sensory signals, later during the usage of the controller, these units receive the desired sensory outcome. It is argued that such a switch requires a “rewiring” of the connections to the input units which does not seem to take place in the CNS (Kawato, 1990). I suggest in Sect. 5.4 a way to overcome this shortcoming.

2.3.5 Staged learning by averaging

The development of “staged learning by averaging” (SLbA) started with the simple observation that unfavorable results in motor performance are often scattered around the desired outcome in the sensory domain. E.g., in throwing a ball to a certain target, the throw might be too close or too far. In this task, the sensory outcome depends heavily on the force generated by the muscles, thus it seems to be a good guess to apply the average force of a throw which is too close and of a throw which is too far. If a throw is way off (e.g., accidentally backwards), it should be discarded completely and not considered at all for learning.

This illustrates the basic idea of SLbA: to collect learning examples which are neither too bad nor necessarily perfect, and to adapt the motor controller to their average motor output during training. In this way, the problem of the missing teacher signal is solved: Instead of directly converting the sensory into the motor error, averaging over non-perfect learning examples takes place. To accomplish this, one needs a controller architecture capable of averaging over learning examples. For example, the MLP fulfills this requirement.

However, pure averaging alone is not sufficient to train precise motor controllers. For this reason, controller performance has to be further enhanced. This is achieved by incrementally improving the quality of the learning examples used for controller training. Such improved learning examples are obtained by searching in motor space in the region around the motor output of the already trained but still imprecise controller for even better motor output. This search can be accelerated by an evolutionary optimization method. In the following, learning by averaging is presented in a staged version for batch learning.

SLbA works by repeatedly generating a set of learning examples and subsequently training a controller with this set (see Fig. 8). Learning examples are included in the training set only if they exceed a certain quality threshold $\tilde{Q}$. The quality $Q$ of a learning example $[x, y^* \rightarrow u]$ or of a controller output $u$ in response to the input $(x, y^*)$ depends on the desired output $y^*$ and the resulting plant output $y = P(x, u)$. It is computed by a function $Q : (y, y^*) \rightarrow Q(y, y^*)$. This function has the following property: the smaller the deviation between $y$ and $y^*$, the larger $Q$. Here, it is assumed that the maximum of $Q$ is 1.0. The deviation between $y$ and $y^*$ can be expressed by their Euclidean distance, but alternative distance measures which are meaningful for the motor learning task at hand are usable as well. Moreover, the quality function $Q$ can be used to incorporate additional constraints in the learning task as it will be shown for the planar arm (see Sect. 4).

In stage $k$, a single learning example $[x, y^* \rightarrow u]$ is generated by the following steps:

1. $x$ and $y^*$ are created at random within their operating range.
2. In the first stage \((k = 1)\), without an existing controller, a random vector \(u_0\) is drawn. In later stages \((k > 1)\), \(u_0 = C_{k-1}(x, y^*)\); \(C_{k-1}\) is the controller trained in the preceding stage.

3. The quality threshold \(\tilde{Q}_k\) is determined: In the first stage \((k = 1)\), \(\tilde{Q}_k\) may be a constant or may depend on \(x, y^*, \) and \(u_0\). In later stages \((k > 1)\), \(\tilde{Q}_k\) may depend on the overall quality of the preceding controller \(C_{k-1}\) as well.

4. \(u\) is repeatedly computed as random variation of \(u_0\) until \(Q(P(x, u), y^*) > \tilde{Q}_k\). In its simplest form, the random variation is realized by adding noise to \(u_0\). Alternatively, one can apply an evolutionary optimization process instead (see Sect. 4.3.1 for the planar arm). This step relies on the assumption that it is always possible to return to the initial state \(x\) to try the next variation of \(u_0\).

The generated learning examples are accumulated in a training set of a predefined size. This set is used for the training of the controller \(C_k\) afterwards. Ideally, with a controller implementation which is capable of averaging, two learning examples \([x, y^* \rightarrow u_1]\) (with quality \(q_1\)) and \([x, y^* \rightarrow u_2]\) (with quality \(q_2\)) result in a controller response \(u_C = \frac{u_1 + u_2}{2}\) to the input \((x, y^*)\). Successful learning only takes place when the controller response is at least better than the inferior learning example of the two. Thus, SLbA requires as necessary precondition that \(Q(P(x, \frac{u_1 + u_2}{2}), y^*) > \min(q_1, q_2)\) for \(u_1 \neq u_2\). This is illustrated in Fig. 9: For a certain \(x\) and \(y^*\), two different combinations of \(P\) and \(Q\) result in different equi-quality curves in a two-dimensional motor space. In Fig. 9a, these curves enclose convex subsets in motor space; for this reason, the SLbA precondition is fulfilled. On the contrary, in Fig. 9b, the quality curves enclose non-convex subsets; in this setting, it is easily possible that the average of two motor commands \(u_1\) and \(u_2\) has lower quality than each of the commands alone (as shown in the figure).
Figure 9 — For a fixed state $x$ and a fixed desired sensory outcome $y^*$, the quality of motor commands $u$ in a two-dimensional motor space is shown as contour plot. Maximum quality is achieved at the point marked $Q_{\text{max}}$. A. Convex equi-quality curves. B. Non-convex equi-quality curves. C. Illustration of SLbA. For further explanation see text.

Figure 9c illustrates the functioning of the staged procedure: in this example, $\tilde{Q}_{k+1} = Q(P(x, u_{C_k}); y^*)$ (marked by the bold dashed curve). Random learning examples are created by adding uniform noise with radius $r$ to $u_{C_k}$. Only the learning examples whose quality exceeds $\tilde{Q}_{k+1}$ are included in the training set of stage $k + 1$ (gray area). Training controller $C_{k+1}$ with this set results in a controller output $u_{C_{k+1}}$ with better quality than $u_{C_k}$.

Figure 10 depicts the core information flow of SLbA as if it was an online learning algorithm. This information flow is similar to FEL and DSL. Instead of a feedback controller or reversed FM, a “quality enhancer” serves to generate an improved motor command $u_{QE}$ by which the motor error $\Delta u$ is computed. But in contrast to FEL and DSL, this motor error is not the result of a local linear approximation, but of a heuristic search in motor space. The generated motor errors average out in controller training to guide learning in the right direction.

However, SLbA does not offer a straightforward solution for the one-to-many problem. Usually, a one-to-many mapping implies a multi-modal quality function in motor space for fixed values of $x$ and $y^*$ (like in Fig. 9a, but with multiple peaks equal to $Q_{\text{max}}$ for the multiple solutions of the inverse kinematics and “valleys” between these peaks). Accordingly, the equi-quality curves no longer enclose convex subsets of the motor space, and it is no longer guaranteed that
averaging over learning examples results in improved motor output. Whenever a one-to-many mapping spoils learning by averaging in this way, one has to include additional constraints in the quality function $Q$ to disambiguate the learning task. SLbA for the planar arm in Sect. 4 relies on such constraints. In previous studies, SLbA has been applied to saccade control with a robot camera head (Hoffmann et al., 2005; Schenck et al., 2003; Schenck and Möller, 2004); in addition, an online version of learning by averaging without distinct stages has been used to train the saccade controller in the study by Schenck et al. (2009).

\[ \Delta u = u_{OE} - u \]

Figure 10 — Learning by averaging: A quality enhancing mechanism serves to generate motor error signals $\Delta u$ (for details see text).

2.4 Introduction to the experimental studies

In the next two sections, the performance of the presented learning strategies is compared in different (simulated) motor control tasks: first, for saccade-like fixation movements with a robot camera head, and second, for the kinematic control of a planar arm. Both motor control tasks are explored for various different task conditions. The performance measure is always the number of exploration trials needed to achieve a certain controller quality $Q^*$. The term “exploration trial” is defined as motor command carried out to collect the plant’s response. Because exploration trials are the most expensive part of learning, their number is used as performance measure: The fewer trials are needed the better. The quality level $Q^*$ is determined by training a controller network with a set of perfect learning examples and assessing its quality afterwards. Important research questions cover the following topics:

- How do the learning strategies perform for plants with approximately linear input-output relationships (saccade learning) compared to plants with highly non-linear input-output relationships (planar arm)?
- How well do the learning strategies deal with one-to-many mappings in practice (planar arm)?
- How robust are the learning strategies in the presence of noise (saccade learning and planar arm)?
- How well do the learning strategies cope with additional constraints for the motor task (planar arm)?
3 Experimental Comparison Study on Saccade Control

Saccades are fast fixation movements of the eyes. Their purpose is to center interesting targets detected in the visual surroundings on both foveas. With the exception of special experimental settings, saccades are generally assumed to be “ballistic” open-loop movements: Once started, their course stays generally unchanged. The research on the mechanisms and neural underpinnings of saccade control in humans and primates has gained a lot of interest from psychology, biology, and neurophysiology (for a comprehensive overview, see Leigh and Zee, 1999). In computer science, especially robotics, the field of “active vision” deals with artificial saccades of robot camera heads. While this research is centered to a large extent on the development of technical solutions (e.g., Klarquist and Bovik, 1998), several studies propose models of saccade generation which are closely related to neurophysiological findings (Dean et al., 1994; Gancarz and Grossberg, 1999). In this area between robotics and biology, methods of adaptive saccade learning are of special interest (Bruske et al., 1997; Pagel et al., 1998).

3.1 Saccade controller

3.1.1 Setup

Due to the large overall amount of needed exploration trials for the present study, the results were generated with a simulated geometrical model of a real-world robot camera head which is shown in Fig. 11 (right). The following description applies both to the real-world setup and the geometrical model. Each camera is equipped with a 1/3” CCD (4.8 × 3.6 mm). Only a central quadratic region with a size of 3.2 × 3.2 mm was used for target identification (in the following, the term “camera image” refers to this cropped region). The lenses have a focal
length of 4 mm, resulting in a diagonal angle of view of 59 degrees. Each camera is mounted on
its own pan-tilt unit, providing two degrees of freedom (horizontal pan, vertical tilt). The pan
angle could vary between −60.4° and 23.8°, the tilt angle between −42.9° and 21.4°. Fixation
targets were located in a cube with a size of 550 × 550 × 550 mm (defined by the corners
(275, −925, −275) and (825, −375, 275) (in mm) in the world coordinate system). This cube
extends above and below the surface of the white table in Fig. 11 (left). Further specifications
and a precise definition of the geometrical model are provided in App. B of Schenck (2008).

3.1.2 Control scheme

The sensory context x the controller receives as input (see Fig. 3) consists of a kinesthetic and a
visual part. The kinesthetic input comprises the current position of the cameras represented by
a conjoint pan-tilt direction and a horizontal and a vertical vergence value: \( pan, tilt, verg_{\text{hor}}, \)
\( verg_{\text{vert}} \). These values are scaled to the range \([-1; +1]\). The corresponding pan and tilt angles
of each camera (\( \text{pan}_{\text{left}}, tilt_{\text{left}}, \text{pan}_{\text{right}}, \) and \( tilt_{\text{right}} \)) are computed by the following equations:

\[
\begin{align*}
\text{pan}_{\text{right/}}_{\text{left}} &= \frac{\text{pan} \pm \lambda_{\text{hor}} \left( \frac{1}{2} verg_{\text{hor}} + \frac{1}{2} \right)}{1 + \lambda_{\text{hor}}} \\
\text{tilt}_{\text{right/}}_{\text{left}} &= \frac{\text{tilt} \pm \lambda_{\text{vert}} verg_{\text{vert}}}{1 + \lambda_{\text{vert}}}
\end{align*}
\]

\( \lambda_{\text{hor}} \) and \( \lambda_{\text{vert}} \) are the horizontal and vertical vergence factor with values of 0.5 and 0.2, re-
respectively. They restrict the vergence operating range to avoid postures where both cameras
point into completely different directions without any overlapping part of their fields of view.
Moreover, the term \( \frac{1}{2} verg_{\text{hor}} + \frac{1}{2} \) in the first equation ensures that the optical axes of the cameras
never point horizontally into diverging directions. After this conversion, \( \text{pan}_{\text{left}}, tilt_{\text{left}}, \text{pan}_{\text{right}}, \)
and \( tilt_{\text{right}} \) are scaled back to the selected operating range of the pan-tilt units (in degrees). The
operating range is chosen in a way that at any \( (pan, tilt) \) setting with zero vergence at least a
small part of the table surface is visible in at least one camera image.

The visual part of the controller input x represents the target position in the left and right camera
image: \( x_{\text{left}}, y_{\text{left}}, x_{\text{right}}, y_{\text{right}} \). These image coordinates are scaled to the range \([-1; +1]\) as well,
the image center is at the origin. The desired sensory state \( y^* \) as additional controller input is
also defined via the sensory variables \( x_{\text{left}}, y_{\text{left}}, x_{\text{right}} \) and \( y_{\text{right}} \). Successful fixation implies
that all of these variables amount to zero. Thus, \( y^* \) is constant for the saccade control task; for
this reason, it can be omitted as controller input. Only DIM-MLP and DIM-NGPCA require
that \( y^* = (x^*_\text{left}, y^*_\text{left}, x^*_\text{right}, y^*_\text{right})^T \) is provided as controller input because \( y^* \) is an essential
element of these learning strategies (see Sect. 2.3.4).

The motor output u of the controller is defined as change of the motor position. It consists of
four values: \( \Delta \text{pan}, \Delta \text{tilt}, \Delta \text{verg}_{\text{hor}}, \) and \( \Delta \text{verg}_{\text{vert}} \) (range: \([-2; +2]\)). The plant adds the delta
values to the current motor position to arrive at the new position. Whenever the valid range
\([-1; +1]\) for the new position is exceeded, it is corrected so that no range transgression takes
place. Moreover, the plant generates the new target position in the left and right camera image
after the camera movement. Within the required working range for the fixation task, the saccade
plant is close to linear. Figure 12 summarizes the controller input and output.

---

3 Horizontal vergence indicates the difference between the individual pan directions of each camera, vertical
vergence between the individual tilt directions.
Figure 12 — Input and output of the saccade controller. The desired visual input is only used for DIM-MLP and DIM-NGPCA.

3.2 Experimental procedure

3.2.1 Task conditions

There were two different task conditions, one without and one with retinal noise. In the task condition with noise, the controller input from the visual modality \((x_{\text{left}}, y_{\text{left}}, x_{\text{right}}, y_{\text{right}})\) was disturbed by Gaussian noise with a standard deviation of 0.015 (approx. 1% of the camera images’ diagonal).

3.2.2 Quality measure

The different learning strategies are compared with regard to the number of exploration trials \(N_{\text{EX}}\) which are required for the controller quality \(Q_C\) to exceed a specific quality level \(Q^*\). The quality \(Q_C\) of a controller is computed as average quality of 250 motor outputs \(u\) to random inputs \(x\). The quality \(Q\) of a single motor output is computed by the following quality function which is also used for SLbA (with \(y^* = 0\); \(Q(y, y^*) = Q(y, 0) = Q(y)\)):

\[
Q(y) = 1 - \frac{r_{L/R}}{2} = 1 - \frac{r_L + r_R}{2} \quad \text{(8)}
\]

\[
r_{L/R} = \frac{1}{\sqrt{2}} \sqrt{x_{\text{left/right}}^2 + y_{\text{left/right}}^2}
\]

\(r_L\) and \(r_R\) represent the left and right radial target distance: the Euclidean distance between the image center and the current image coordinates of the fixation target in the left and right camera, respectively. If the target object is not visible in both camera images simultaneously, \(Q\) is set to \(-1\).

The quality level \(Q^*\) is determined by training the standard controller network (see Fig. 13, left) with a set of 500 perfect learning examples over 500 epochs and assessing its quality afterwards. These examples were obtained by the optimization method “differential evolution” (DE; Storn and Price, 1997) without applying retinal noise. The optimization goal was to find parameters \(u\) for which \(Q(y) = Q(P(x, u))\) was as close to 1.0 as possible; each learning example

\[\text{4 Whenever random sensory input is generated for the saccade task, first a random position for the target object within the predefined cube is determined. Afterwards, the initial motor position of the cameras is repeatedly generated at random until the target object is “visible” in both camera images.}\]
Table 1 — Results of the DE controller networks for the saccade control task. $Q^{DE}$ is the average controller quality for the respective task condition, $Q^*$ the desired quality level.

<table>
<thead>
<tr>
<th>Condition</th>
<th>$Q^{DE}$</th>
<th>$Q^*$</th>
<th>Exploration trials (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without noise</td>
<td>0.985</td>
<td>0.98</td>
<td>516721 (79917)</td>
</tr>
<tr>
<td>With noise</td>
<td>0.977</td>
<td>0.972</td>
<td></td>
</tr>
</tbody>
</table>

was created for a different random context $x$ (for a detailed technical description of DE and its application see Schenck, 2008). This procedure — generation of a perfect training set and subsequent controller training — was repeated 20 times. The value for $Q^*$ for each task condition was chosen slightly below the average quality of the resulting condition-specific controller networks. For the task condition with noise, retinal noise was applied during controller testing.

The results of the DE controller networks are shown in Table 1. For each task condition, two quality values are given. The left one is the average controller quality $Q^{DE}$, the right one the value chosen for $Q^*$ for the respective task condition for the subsequent studies. The average quality of these controller networks (trained with virtually perfect learning examples) marks the upper performance limit of these networks for the saccade control task. For this reason, $Q^*$ is chosen just slightly below their average quality. Moreover, the average number of exploration trials required to collect the training set by DE is reported in Table 1 as well. It is rather large in comparison to the results obtained with the different learning strategies (see Table 3).

### 3.2.3 Parameter variation

All learning strategies differ in the way they can be adjusted by implementing certain constraints or by setting specific parameters. To arrive at a fair comparison of their performance, less important parameters were carefully set to assure a good basic performance of each learning strategy (“fixed parameters” in Table 13 in App. B). Afterwards, one to three parameters with significant impact on the performance were varied systematically for each strategy (“variable parameters” in Table 13). For each parameter combination, 20 learning passes were carried out. $Q_C$ was measured during the course of learning. Whenever $Q_C$ equaled or exceeded $Q^*$, the learning pass was halted. The needed number of exploration trials for a certain parameter combination is the average of all 20 learning passes. Furthermore, a learning pass was stopped without success whenever the number of learning cycles became larger than an upper limit $T_{\text{max}}$ (for the online methods FEL and DSL, a learning cycle is identical to the generation of a single learning example; for the batch methods DIM-MLP and DIM-NGPCA, a learning cycle is identical to a training epoch (DIM-MLP) or a single training step (DIM-NGPCA); for SLbA, a learning cycle is identical to a stage; these different definitions result from the different structure of the compared learning strategies). Values for $T_{\text{max}}$ are also reported in Table 13. A combination of variable parameter values is only designated as “successful” if all 20 learning passes succeeded. For clarity, Table 2 provides an explanation of the terms “exploration trial”, “learning pass”, “learning cycle”, “training step”, and “training epoch”.
<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploration trial</td>
<td>Execution of a single motor command during the course of learning</td>
</tr>
<tr>
<td>Learning pass</td>
<td>A single attempt to acquire the desired quality level through learning with a</td>
</tr>
<tr>
<td></td>
<td>specific setting of the variable parameters</td>
</tr>
<tr>
<td>Learning cycle</td>
<td>Each learning pass consists of a sequence of learning cycles; the exact</td>
</tr>
<tr>
<td></td>
<td>meaning varies between learning strategies</td>
</tr>
<tr>
<td>Training step</td>
<td>An update of the network parameters by a single learning example</td>
</tr>
<tr>
<td>Training epoch</td>
<td>An update of the network parameters by an iteration through all of the</td>
</tr>
<tr>
<td></td>
<td>learning examples in the training set</td>
</tr>
</tbody>
</table>

Table 2 — Definition of important terms.

### 3.3 Learning strategies

#### 3.3.1 Networks

Except for DIM-NGPCA, the adaptive controllers were implemented by multi-layer perceptrons (MLP; see Rumelhart et al., 1986). Figure 13 (left) shows the general controller network with linear input units, four hidden sigmoid units (hyperbolic tangent as activation function), and linear output units. A shortcut connection projects directly from the input to the output layer. This facilitates learning of the linear part of the inverse plant. Only the network for DIM-MLP (Fig. 13, right) is larger (with six hidden sigmoid units) because of the additional input for $y^*$. Figure 14 depicts the combined network used for DSL consisting of the forward model (FM) and the controller. As pointed out in section 2.3.3, first the network weights belonging to the FM are learned. Afterwards, these connections are frozen and the controller part of the network is trained. For all learning strategies, stochastic gradient descent (online backpropagation; see Rumelhart et al., 1986) was applied for network training. To keep things as straightforward as possible, the standard backpropagation algorithm was not modified. The weights of the MLP networks were initialized to random values from the range $[-0.1; 0.1]$. Thus, regarding network training, there is only one free parameter, the learning rate $\eta$.

The NGPCA network for DIM-NGPCA is specified according to the description in Sect. A. The learning parameters were $T = 100000$ (alias $T_{\max}$), $T_{ortho} = 10000$, $\epsilon(0) = 0.5$, $\epsilon(T) = 0.05$, $\rho(0) = 1.0$, $\rho(T) = 0.01$, $\sigma^2(0) = 0.0$, and $\lambda(0) = 10.0$. The number of PCA units $N$ and the number of eigenvectors $m$ were varied.

#### 3.3.2 Parameter settings

**Parameters for FEL**  The only free parameter for FEL is the gain factor $\eta$. It is equivalent to the learning rate of stochastic gradient descent. The feedback controller equation $\Delta u =$
\( G_{u,x} \Delta y \) takes the following form for the saccade control task:

\[
\begin{array}{c}
\Delta_{\text{pan}} \\
\Delta_{\text{tilt}} \\
\Delta_{\text{verg}_{\text{hor}}} \\
\Delta_{\text{verg}_{\text{vert}}}
\end{array}
= \begin{pmatrix}
\frac{1}{2} & 0 & \frac{1}{2} & 0 \\
0 & \frac{1}{2} & 0 & \frac{1}{2} \\
-\frac{1}{2} & 0 & \frac{1}{2} & 0 \\
0 & -\frac{1}{2} & 0 & \frac{1}{2}
\end{pmatrix}
\begin{pmatrix}
-x_{\text{left}} \\
-y_{\text{left}} \\
-x_{\text{right}} \\
-y_{\text{right}}
\end{pmatrix}
\]

This is a heuristic form which does not take the kinesthetic input into account. As shown by Schenck (2008), the kinesthetic input plays only a subordinate role for the characteristics of the corresponding plant. Therefore, this omission is justified.

**Parameters for DSL**  DSL has four parameters: the number of learning examples for the FM \( N_{\text{FM}} \), the number of epochs used to train the FM, and the learning rates \( \eta_{FM} \) for FM training and \( \eta \) for controller training. The number of epochs used to train the forward model was set equal to \( N_{\text{FM}} \). This proved to result in proper learning of the FM without overfitting. \( \eta_{FM} \) was set to a fixed value, \( N_{\text{FM}} \) and \( \eta \) were varied systematically.

**Parameters for DIM-MLP**  The most important parameter for DIM-MLP is the number of learning examples \( N_{\text{CON}} \) in the training set. The learning rate \( \eta \) for the MLP was set to a fixed value.

**Parameters for DIM-NGPCA**  In addition to the number of PCA units \( N \) and the number of eigenvectors \( m \), a third variable parameter for DIM-NGPCA is the number of learning examples \( N_{\text{CON}} \) in the training set.

**Parameters for SLbA**  For SLbA, one needs a strategy how to increase the number of learning examples and training epochs in each stage. It proved to be a favorable approach to increase the number of learning examples from stage to stage, starting with 10 examples in stage 1 and increasing this number by 10 every stage. This corresponds to learning the coarse structure of
the problem first with only a small number of learning examples and refining it in later stages. For proper learning without overfitting, the number of training epochs in each stage was chosen to equal the number of learning examples. The applied quality function $Q$ is stated in Eqn. (8). The quality threshold for the generation of a single learning example with random sensory context input $x$ is computed as $\tilde{Q}_k = Q(P(x, u_0))$. In the first stage ($k = 1$), $u_0$ is a random motor command. For $k > 1$, $u_0 = C_{k-1}(x)$. The noise which is added to $u_0$ in the search for a better motor output is drawn from a multivariate Gaussian distribution with zero mean and standard deviation $\sigma = \sigma_0 [1 - Q(P(x, u_0))]$ for all dimensions. Computing $\sigma$ this way ensures that the better the saccade $u_0$, the smaller its variation. This is reasonable since large variations of a good saccade are more likely to result in worse than better fixation. The parameter $\sigma_0$ was varied systematically. It has a significant impact on the performance of SLbA.

The parameter settings for all learning strategies and task conditions are reported in Table 13 in App. B.

### 3.4 Results

**General remarks** The results for the saccade learning tasks are presented in Table 3. The number of required exploration trials $N_{\text{EX}}$ for the best successful combination of variable parameter values and the settings of these parameters are reported there (for DSL, $N_{\text{EX}}$ is computed as the sum of the number of exploration trials for the generation of the training set of the FM $N_{\text{FM}}^{\text{EX}}$ and for the subsequent controller training $N_{\text{CON}}^{\text{EX}}$).

For DIM-MLP and DIM-NGPCA, the ratio $N_{\text{EX}}/N_{\text{CON}}$ is larger than 1.0 since a single random saccade is not necessarily usable as learning example. Instead, the saccadic target is most often not longer visible in both camera images after a random saccade. Learning examples like this
Saccade learning task without retinal noise

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>410 (57)</td>
<td>$N_{CON} = 40, N = 1, m = 16$</td>
</tr>
<tr>
<td>DIM-MLP</td>
<td>538 (63)</td>
<td>$N_{CON} = 50$</td>
</tr>
<tr>
<td>FEL</td>
<td>1904 (322)</td>
<td>$\eta = 0.26$</td>
</tr>
<tr>
<td>DSL</td>
<td>4137 (849)</td>
<td>$N_{FM} = 20, \eta = 0.18$</td>
</tr>
<tr>
<td>SLbA</td>
<td>19005 (1069)</td>
<td>$\sigma_0 = 0.8, k_{max} = 22.9 (0.74)$</td>
</tr>
</tbody>
</table>

Saccade learning task with retinal noise

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>648 (71)</td>
<td>$N_{CON} = 60, N = 1, m = 16$</td>
</tr>
<tr>
<td>DIM-MLP</td>
<td>652 (64)</td>
<td>$N_{CON} = 60$</td>
</tr>
<tr>
<td>FEL</td>
<td>1680 (366)</td>
<td>$\eta = 0.2$</td>
</tr>
<tr>
<td>DSL</td>
<td>3197 (499)</td>
<td>$N_{FM} = 20, \eta = 0.11$</td>
</tr>
<tr>
<td>SLbA</td>
<td>26498 (5308)</td>
<td>$\sigma_0 = 1.4, k_{max} = 16.6 (1.35)$</td>
</tr>
</tbody>
</table>

Table 3 — Results for the saccade learning tasks. Learning strategies are sorted in ascending order with regard to the required number of exploration trials. The corresponding best settings for the variable parameters are shown in the right column (for SLbA, the average required number of stages $k_{max}$ and its standard deviation (in brackets) are shown as well).

are useless. For the same reason, the collection of a single learning example for the training set of the FM in DSL requires multiple exploration trials. The ratios $N_{EX}/N_{CON}$ (for DIM) and $N_{FM}^{EX}/N_{FM}$ (for DSL) amount roughly to 10 for all task conditions.

In addition, for DIM-NGPCA a special difficulty arises: The number of exploration trials depends only on the size of the training set $N_{CON}$. Thus, a multitude of combinations of the number of PCA units $N$ and the number of eigenvectors $m$ can be successful for a certain value of $N_{CON}$. In Table 3, only one of these combinations is reported.

Figure 15 illustrates exemplary for SLbA for the saccade learning task without noise that the range of variable parameter values was carefully chosen. This histogram plot shows the number of exploration trials $N_{EX}$ depending on the noise factor $\sigma_0$. The optimum choice for $\sigma_0$ indicates the minimum of an approximately u-shaped distribution.

**General performance** Without retinal noise, DIM-NGPCA is clearly the fastest learning strategy with only 410 required exploration trials ($N_{EX} = 410, N_{CON} = 40$), closely followed by DIM-MLP ($N_{EX} = 538, N_{CON} = 50$). The remaining results are: FEL ($N_{EX} = 1904$), DSL ($N_{EX} = 4137$), SLbA ($N_{EX} = 19005$). Nearly the same ranking order results from the saccade learning task with noise, also the number of explorations trials stays at a similar level for all strategies. However, with noise DIM-NGPCA and DIM-MLP share the first place, both with the same size of the training set ($N_{CON} = 60$).
Statistical tests Since this study was started without any in-advance hypotheses, the only purpose of statistical tests is to support the post-hoc analysis of the data by indicating how likely measured differences are caused by random variations or by real differences between the underlying populations. The statistical analysis was restricted to the most important comparison, namely between learning strategies within each task condition. For this comparison, every learning strategy was matched with every other strategy. The compared measure was the mean number of required exploration trials (as reported in Table 3). The computed pairwise t-tests (two-sided, for independent samples) yielded highly significant results \((p < 0.001)\) in all pairwise comparisons except for the ones in which DIM-MLP and DIM-NGPCA were compared with \(N_{\text{CON}}\) being equal (saccade learning task with noise). Since the process of training data generation is identical for DIM-MLP and DIM-NGPCA, the latter result is inevitable. Overall, these statistical results support the reliability of the reported data.

Additional observations In examining the results for DIM-NGPCA for all variable parameter combinations, two interesting observations were made which partly reveal the structure of the underlying learning task. First, the lower bound for the number of eigenvectors \(m\) amounts to 11 for successful parameter combinations. Second, in every task condition, a single PCA unit is sufficient for successful learning as long as \(N_{\text{CON}}\) is large enough. The combination of one PCA unit with 16 eigenvectors belongs even to the parameter combinations which succeed with the smallest number of training examples \(N_{\text{CON}}\). Taken together, these findings support the initial claim that the sensorimotor data manifold in the saccade learning task is not curved but mainly linear, and moreover that the intrinsic dimensionality of this manifold amounts to 11.

With regard to SLbA, the required number of stages \(k_{\text{max}}\) is of special interest. It amounts to 22.9 (without noise) or 16.6 (with noise). The standard deviations are close to one, indicating small variability. Fig. 16 illustrates the result of a single learning pass with optimum parameter settings for the saccade learning task without noise with \(k_{\text{max}} = 22\). For the first and last stage
and a stage in between, a fourfold panel is shown, depicting the coordinate space of the left and right camera image, the \((\text{pan, tilt})\) subspace, and the \((\text{verg}_\text{hor}, \text{verg}_\text{vert})\) subspace. The sensory consequences \(\mathbf{x}_{t+1}\) of 1000 controller outputs \(\mathbf{u}_t\) in response to random controller inputs \(\mathbf{x}_t\) are marked with black dots in the different panels. After the final stage, the controller is a very good performer; this can be concluded from the small scatter around the image centers, indicating precise fixation. In contrast, after the first stage this scatter is still very large. Furthermore, the \((\text{verg}_\text{hor}, \text{verg}_\text{vert})\) panel of the last stage indicates how small the variation of these variables is for proper fixation movements.

Table 4 demonstrates that the averaging during controller training actually increases the quality. This table presents for the saccade learning task without noise the average quality \(Q_{PS}\) of the learning examples in the training set side by side with the average controller quality \(Q_C\) on a test set for the different stages of a single learning pass. During the first stages, \(Q_C\) is often larger than \(Q_{PS}\). This shows that learning by averaging mostly works as expected. During the last stages shortly before final convergence, the positive effect of averaging is lost. Nevertheless, controller quality still increases from stage to stage because the underlying training set continues to improve.

### 3.5 Discussion

The saccade learning task requires that the controller adapts to an approximately linear input-output relationship. Learning strategies which work by local linear approximation like FEL and DSL, or which use linear models to represent the training data manifold like DIM-NGPCA, seem to profit from this task characteristic. In contrast, SLbA cannot exploit the linear task characteristics as efficient as the other learning strategies. The influence of noise on the overall ranking order is basically non-existent. Since the performance of FEL is considerably better than the performance of DSL, it is fair to say that the heuristically determined gain matrix for FEL is better suitable for the local linear adjustment than the estimate \(\hat{\mathbf{J}}_{u,x}\) which is provided by the FM in DSL.
In conclusion, for mainly linear tasks like saccade learning the learning strategies FEL, DIM-NGPCA, and DIM-MLP seem to be best suited. Only if the ratio $N_{EX}/N_{CON}$ gets too large because of the task characteristics, this recommendation has to be restricted to FEL. However, every learning strategy is the superior choice for saccade learning compared to optimization methods like DE which require a huge number of exploration trials (see Table 1 vs. Table 3). The rather good performance of FEL supports biologically inspired models of saccade learning which rely on FEL. In these models, the feedback controller is mostly identified as (inherited and pre-wired) “brainstem mechanism” (Dean et al., 1994; Gancarz and Grossberg, 1999).

<table>
<thead>
<tr>
<th>Stage $k$</th>
<th>Without noise</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$Q_{PS}$ 0.615</td>
<td>$Q_{C}$ 0.624</td>
</tr>
<tr>
<td>2</td>
<td>$Q_{PS}$ 0.764</td>
<td>$Q_{C}$ 0.771</td>
</tr>
<tr>
<td>3</td>
<td>$Q_{PS}$ 0.842</td>
<td>$Q_{C}$ 0.834</td>
</tr>
<tr>
<td>4</td>
<td>$Q_{PS}$ 0.868</td>
<td>$Q_{C}$ 0.865</td>
</tr>
<tr>
<td>5</td>
<td>$Q_{PS}$ 0.880</td>
<td>$Q_{C}$ 0.895</td>
</tr>
<tr>
<td>6</td>
<td>$Q_{PS}$ 0.903</td>
<td>$Q_{C}$ 0.919</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>$Q_{PS}$ 0.980</td>
<td>$Q_{C}$ 0.977</td>
</tr>
<tr>
<td>21</td>
<td>$Q_{PS}$ 0.980</td>
<td>$Q_{C}$ 0.978</td>
</tr>
<tr>
<td>22</td>
<td>$Q_{PS}$ 0.982</td>
<td>$Q_{C}$ 0.980</td>
</tr>
</tbody>
</table>

Table 4 — Comparing the average quality $Q_{PS}$ of the learning examples in the training set with the average controller quality $Q_{C}$ on a test set for the different stages of a single learning pass of SLbA for the saccade learning task without retinal noise.
4 Experimental Comparison Study on the Control of a Planar Arm

In this section, the performance of the different learning strategies is compared for the kinematic control of a simulated planar arm. This task poses several special difficulties. First, the plant is strongly non-linear. Second, it implies a one-to-many mapping. Since some of the explored learning strategies are not suited for one-to-many mappings, different constraints are imposed on the learning task.

Three different planar arms are explored in this study. They differ with regard to the number of links $L$ which varies between two and four. For the 3- and 4-link arm, two different learning constraints are explored in addition to the no-constraint condition. Moreover, for each planar arm, there is an additional experimental condition with sensory noise. Altogether, this yields ten different comparisons. Not all learning strategies are suited for each combination of planar arm and constraint, thus the list of learning strategies in each comparison varies (see Table 6).

4.1 Arm controller

4.1.1 Setup

The planar arm consists of $L$ links, each of them with unit length (see Fig. 17). The basis joint is at the origin of the two-dimensional coordinate system of the working space. The joint angles of the planar arm form the vector $\theta = (\theta_1, ..., \theta_L)'$. The relevant sensory information in this system is the position of the tip of the last link $y = (y_1, y_2)'$. It is used to define the desired sensory state $y^*$ as well. The plant $P$ is defined by the following equations:

$$
\begin{pmatrix}
  y_1 \\
  y_2
\end{pmatrix} = P(\theta) = 
\begin{pmatrix}
  \sum_{i=1}^{L} \cos \left( \sum_{j=1}^{i} \theta_j \right) \\
  \sum_{i=1}^{L} \sin \left( \sum_{j=1}^{i} \theta_j \right)
\end{pmatrix}
$$

The operating range in which the desired sensory states $y^*$ are positioned is restricted to an area with $y_1^* \in [-L/\sqrt{2}; L/\sqrt{2}]$ and $y_2^* \in [0.2; L/\sqrt{2}]$. The task of the controller is to position the planar arm by a motor command $u = \theta$ resulting in $y^* = y$. In contrast to the saccade controller of Sect. 3, $y^*$ is variable and there is no sensory context input $x$.

Except for the marginal case of a completely outstretched arm, the inverse kinematics of such a planar arm yields two solutions (for $L = 2$) or infinitely many solutions (for $L \geq 3$). Figure 18 illustrates for a 3-link arm that the set of solutions is non-convex: The average of two solutions is no solution in itself. This is clearly visible, when an elbow-down and an elbow-up solution are combined (Fig. 18, left), but holds as well for two solutions from the elbow-down class (Fig. 18, right). Thus, the planar arm suffers from the one-to-many problem of motor learning (see Sect. 2.3.1 and also Jordan and Rumelhart, 1992).

Since SLbA works also on the basis of function approximation, the mixture of elbow-down and elbow-up postures during training resulted in a severe performance drop for this learning strategy during initial preliminary tests. For this reason, the learning task was slightly facilitated by restricting the initialization range of the joint angles for the random generation of motor
Figure 17 — Planar arm with $L$ links. In this illustration, the tip position $y$ deviates from the desired tip position $y^*$. 

Figure 18 — These figures illustrate the non-convexity of the sets of solutions for the inverse kinematics of the 3-link planar arm. Left: The average of an elbow-down and an elbow-up solution deviates considerably from the desired tip position. Right: Even within the set of elbow-down solutions, averaging is not possible.

commands (whenever required in the course of each learning strategy): The angle $\theta_1$ of the basic joint was drawn from the range between $-90^\circ$ and $180^\circ$, all other angles from the range between $0^\circ$ and $180^\circ$. This prevented the mixture of elbow-down and elbow-up solutions during the random generation of motor commands while preserving part of the one-to-many relationship. Beyond this, the operating range of the joint angles was not restricted at all; depending on the learning strategy, new motor commands outside the initialization range could arise.

4.1.2 Control scheme

The task of the controller is to generate motor output $u = \theta$ such that the tip of the last link is placed at the desired position in the working space. As controller input, there is no sensory context $x$ (see Fig. 3) in this control task, only the desired tip position $y^*$. The basic quality $Q$
of an arm posture $\theta$ in conjunction with a desired tip position $y^*$ is computed as:

$$Q_{\text{basic}}(P(\theta), y^*) = Q_{\text{basic}}(y, y^*) = 1 - \|y - y^*\|$$

As mentioned before, there are different constraints imposed on the learning task. These constraints are reflected by different quality functions for the evaluation of the arm controllers (all computations involving $\theta$ angles are in radiant):

- **No additional constraint:** $Q_0(y, y^*) = Q_{\text{basic}}(y, y^*)$

- **First constraint (“maximum symmetry”):** All joint angles $\theta_i$ with $i > 1$ should be equal.

$$Q_1(y, y^*) = \frac{1}{2}Q_{\text{basic}}(y, y^*) + \frac{1}{2} \left(1 - \sqrt{\frac{1}{L-1} \sum_{i=2}^{L} (\bar{\theta} - \theta_i)^2}\right)$$

with $\bar{\theta} = \frac{1}{L-1} \sum_{i=2}^{L} \theta_i$

Whenever the arm collides with itself, $Q_1(y)$ is set to a penalty value of $-100$.

- **Second constraint (“minimum energy”):** The arm should move as little as possible from the zero position (where all joint angles amount to zero).

$$Q_2(y, y^*) = \frac{2}{3}Q_{\text{basic}}(y, y^*) + \frac{1}{3} \left(1 - \frac{1}{2L} \sum_{i=1}^{L} \theta_i^2\right)$$

The first constraint completely disambiguates the learning problem. To achieve maximum quality $Q_1$, only one elbow-down and one elbow-up solution are applicable. Together with the restricted initialization range, controller learning is reduced to a functional relationship. It was expected in advance that such a constraint would work in favor of SLbA. On the contrary, the second constraint poses additional difficulties to SLbA since the maximum achievable quality $Q_2$ varies depending on the desired tip position. This requires changes of the SLbA learning strategy which are described later on in Sect. 4.3.1.

### 4.2 Experimental procedure

#### 4.2.1 Quality measure

Like in the saccade control task, the different learning strategies are compared with regard to the number of exploration trials which are required to arrive at a certain controller quality $Q_C > Q^*$. The quality $Q_C$ of a controller is computed as average quality of 250 motor outputs $u$ in response to random desired sensory states $y^*$. The quality $Q$ of a single motor output is computed by the quality function which corresponds to the current learning constraint.

The quality level $Q^*$ is determined by training the standard controller network (see Fig. 19, left) with a set of 1000/1500 perfect learning examples over 2000/3500 epochs (first value for the
Table 5 — Results of the DE controller networks for the planar arm control task, evaluated with the different quality functions. $Q^{\text{DE}}$ values are the average controller qualities for the respective task conditions, $Q^*$ values are the corresponding desired quality levels. $Q^{\text{DE}0N}$ and $Q^*_{0N}$ indicate the values for the task condition with additional noise and no constraint.

<table>
<thead>
<tr>
<th>Links</th>
<th>$Q^{\text{DE}}_0$</th>
<th>$Q^*_0$</th>
<th>$Q^{\text{DE}0N}$</th>
<th>$Q^*_{0N}$</th>
<th>$Q^{\text{DE}}_1$</th>
<th>$Q^*_1$</th>
<th>$Q^{\text{DE}}_2$</th>
<th>$Q^*_2$</th>
<th>Explor. trials (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.979</td>
<td>0.97</td>
<td>0.957</td>
<td>0.945</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>763413 (61724)</td>
</tr>
<tr>
<td>3</td>
<td>0.962</td>
<td>0.96</td>
<td>0.926</td>
<td>0.93</td>
<td>0.981</td>
<td>0.97</td>
<td>0.867</td>
<td>0.86</td>
<td>5508549 (310954)</td>
</tr>
<tr>
<td>4</td>
<td>0.947</td>
<td>0.94</td>
<td>0.904</td>
<td>0.9</td>
<td>0.973</td>
<td>0.97</td>
<td>0.889</td>
<td>0.86</td>
<td>10533129 (485243)</td>
</tr>
</tbody>
</table>

The results of the DE controller networks are shown in Table 5. The average quality of these controller networks (trained with virtually perfect learning examples) marks the upper performance limit for the planar arm task. For this reason, $Q^*$ is generally chosen close to $Q^{\text{DE}}$ with the exception of the 4-link arm combined with $Q^*_2$. Because this task condition proved to be very hard and time-consuming, $Q^*$ was chosen considerably lower than $Q^{\text{DE}}$ in this case. Moreover, the average number of exploration trials required to collect the training set by DE is reported in Table 5 as well. It increases considerably with the number of links and demonstrates that motor learning by DE is not well suited for real world applications.

4.2.2 Parameter variation

The variation of the parameters of each learning strategy was carried out as in the saccade learning task (for a description see Sect. 3.2.3). The “fixed” and “variable parameters” are reported in Tables 14 to 16.

4.2.3 Task conditions

The task variation for the arm control task has two components as explained before. First, the number of links is varied between two and four, and second, two different constraints are applied to the learning task (in addition to the standard no-constraint condition). Moreover, for each arm, there is one condition with additional sensory noise, added to the measurement of the tip position $y$. This influences both the accuracy of the sensory error signal and the quality measurement. The noise is generated from a Gaussian distribution with variance $\sigma_{\text{noise}} = $
### Table 6 — Overview of the experimental conditions. Combinations of learning strategy and constraint without tick are excluded from the study.

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEL-$J^+$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FEL-$J^t$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DIM-MLP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DSL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SLbA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

0.015$L$ (1.5% of the length of the fully stretched arm). Altogether, this yields ten task conditions as depicted in Table 6, in which the learning constraints are designated by their respective quality function: $Q_0 \rightarrow$ no constraint, $Q_{ON} \rightarrow$ no constraint with noise, $Q_1 \rightarrow$ ”maximum symmetry” constraint, $Q_2 \rightarrow$ ”minimum energy” constraint.

### 4.3 Learning strategies

#### 4.3.1 Special considerations for the planar arm task

In the following, it is explained which learning strategies are employed in the comparison study on the planar arm in which task conditions (see Table 6) and which particular modifications are applied.

**FEL** As outlined in Sect. 2.3.2, the gain matrix $G_u$ can be defined in different ways. In this study, FEL is tested with $G_u = J_{u,x}^+$ and with $G_u = J_{u,x}^t$ (called FEL-$J^+$ and FEL-$J^t$ in the following). The second variation is of special interest since it allows to compare the performance of FEL with a precise gain matrix $J_{u,x}^t$ with the performance of DSL which relies on approximating $J_{u,x}^t$ (see Sect. 2.3.3).

FEL is only incorporated into the comparisons without additional constraint. Although it is possible to define a plant whose output reflects how well the constraints are fulfilled (see DSL), the analytical identification of the matrices $J_{u,x}^+$ and $J_{u,x}^t$ for this extended plant involves so many operations which are not plausible for a truly adaptive system that FEL was only included in the comparisons without learning constraint.

**DSL** DSL is included in every task condition. In the conditions with constraint, the FM of the DSL learning scheme has to learn the output of an extended plant which not only provides the tip position as output but also information relevant for the fulfillment of the constraint. For the first constraint (“maximum symmetry”), the following $L - 1$ additional outputs are provided by
the plant:
\[ y_{i+2} = \theta_{i+1} - \theta_{i+2} \quad \text{with} \quad i = 1, \ldots, L - 2 \quad \text{for} \quad L \geq 3 \]
Moreover, \( y_{L+1} \) is set to 1 for a posture with collision, and to -1 for a collision-free posture. The desired outputs \( y^*_i \) amount to 0 for \( i = 3, \ldots, L \), and to -1 for \( i = L + 1 \).

For the second constraint (“minimum energy”), the following additional output is provided by the plant:
\[ y_3 = \frac{1}{2L} \sqrt{\sum_{i=1}^{L} \theta_i^2} \]
The corresponding desired output \( y^*_3 \) amounts to 0.

**DIM-MLP** Since DIM-MLP on the basis of function approximation cannot work for one-to-many problems (Jordan and Rumelhart, 1992), it is only included in the comparisons with the 2-link arm. During the search for learning examples for the training set of DIM-MLP (and also of DIM-NGPCA), only motor commands are included which result in sensory states \( y \) such that \( y_1 \) and \( y_2 \) are positioned in the operating range of the desired sensory states \( y^* \). For this reason, the number of exploration trials for DIM-MLP and DIM-NGPCA is larger than the number of collected learning examples. In this way, the lacking goal-directedness of DIM is directly reflected by the number of exploration trials.

**DIM-NGPCA** The combination of DIM with NGPCA should be able to tackle one-to-many problems. For this reason, DIM-NGPCA is included in every comparison. In the conditions with constraint, the desired sensory state as controller input and the output of the plant are extended in the same way as described for DSL. This leads to the additional difficulty that the portion of randomly generated learning examples in the training set which are useful for the control task is considerably reduced (only the ones close to fulfilling the respective constraint by chance). Moreover, the second constraint, which enforces minimum deviation from the zero position of the arm, implies a tradeoff between reaching precision and constraint fulfillment. During controller usage after training, the desired sensory state \( y^*_3 \) as controller input for this constraint is specified as 0 to indicate that it should be as small as possible. However, none of the learning examples contains an input \( y^*_3 = 0 \) since \( y_3 \) is always larger. Thus, the NGPCA network has to extrapolate while generating motor commands during controller usage. The results will show how well DIM-NGPCA can cope with these added difficulties.

**SLbA** The simple standard strategy of learning by averaging as employed for the saccade learning task did not yield satisfying results for the strongly non-linear plant of the planar arm which includes non-convex solution sets for the controller output. Therefore, three different enhancements are implemented, all of them concerning the generation of a single learning example \([x, y^* \rightarrow u]\). They are explained in the following with the notation and on the basis of Sect 2.3.5, thus including the sensory context \( x \).

First, a simple evolutionary strategy is used in the search for better learning examples. The new motor command \( u \) is not only determined by the random variation of \( u_0 \) until \( Q(P(x, u), y^*) > \tilde{Q}_k \). Instead, for each generated motor command \( u \), an additional check is performed: Whenever \( Q(P(x, u), y^*) > Q(P(x, u_0), y^*) \), \( u_0 \) is substituted by \( u \): \( u_0 := u \). Afterwards, the
new \( u_0 \) is the basis for the generation of motor commands \( u \). This process is repeated until \( Q(P(x, u), y^*) > \tilde{Q}_k \) (as usual). This evolutionary strategy speeds up the search for better learning examples.

Second, during each search for a better learning example, the quality threshold \( \tilde{Q}_k \) is lowered with every non-successful attempt to find a motor command \( u \) which exceeds this threshold. Let \( \tilde{Q}_k^{\text{init}} \) be the initial quality threshold, \( u_0^{\text{init}} \) the initial \( u_0 \) before the evolutionary process kicks in, and \( t \) the number of attempts to find a suitable \( u \) so far. \( \tilde{Q}_k \) is determined as

\[
\tilde{Q}_k = (1 - \lambda_{SLbA}^*) Q(P(x, u_0^{\text{init}}), y^*) + \lambda_{SLbA}^* \tilde{Q}_k^{\text{init}}
\]

\[
\lambda_{SLbA}^* = e^{-\lambda_{SLbA} t}
\]

with \( \lambda_{SLbA} \in ]0; 1] \). To interpret these equations: \( \tilde{Q}_k \) decreases exponentially towards the quality of \( u_0^{\text{init}} \), starting from \( \tilde{Q}_k^{\text{init}} \). \( \lambda_{SLbA} \) determines the speed of decay and has to be set by the user as free learning parameter. This enhancement helps to avoid that SLbA gets stuck with the generation of better motor commands in a region where this is very difficult. The initial quality threshold \( \tilde{Q}_k^{\text{init}} \) is determined in the following way: In the first stage \((k = 1)\), it is set to 0.3. For \( k > 1 \), \( \tilde{Q}_k^{\text{init}} \) is computed by

\[
\tilde{Q}_k^{\text{init}} = \frac{1}{2}(1 - Q(P(x, u_0^{\text{init}}), y^*)) + Q(P(x, u_0^{\text{init}}), y^*) .
\]  

Thus, the initial quality threshold depends solely on the quality of the controller output \( u_0^{\text{init}} \) in the specific sensory context \( x \).

Third, an additional learning parameter \( \lambda_\sigma \in ]0; 1] \) is introduced for the random variation of \( u_0 \). The noise which is added to \( u_0 \) in the search for a better motor command \( u \) is drawn from a multivariate Gaussian distribution with zero mean and standard deviation \( \sigma = \sigma_0 \lambda_\sigma [1 - Q(P(x, u_0))] \) for all dimensions. \( t \) is again the number of attempts to find a suitable \( u \) so far, \( \sigma_0 \) is another free learning parameter also found in the saccade learning task. By this means, the search region for motor commands is reduced step by step, with the expectation that it is easier to find better motor commands in the close vicinity of \( u_0 \).

4.3.2 Networks

Like in the saccade control task, the adaptive controllers were implemented by MLPs and trained by stochastic gradient descent (except for DIM-NGPCA). Figure 19 (left) shows the general controller network with linear input units, ten hidden sigmoid units (hyperbolic tangent as activation function), and linear output units. The only network input is the desired tip position, no sensory context is provided. Figure 19 (right) depicts the combined network used for DSL consisting of the FM and the controller. The hidden layer of the FM has 30 sigmoid units. This rather large number was chosen to ensure that the network is complex enough to acquire a precise FM even for the 4-link arm.

The learning parameters of the NGPCA network for DIM-NGPCA were \( T_{\text{ortho}} = 10000 \), \( \varepsilon(0) = 0.5 \), \( \varepsilon(T) = 0.05 \), \( \rho(0) = 1.0 \), \( \rho(T) = 0.01 \), \( \sigma^2(0) = 0.0 \), and \( \lambda(0) = 10.0 \) (see Table 12). The maximum number of training steps \( T_{\text{max}} \) differed between task conditions, the number of PCA units \( N \) and the number of eigenvectors \( m \) were variable parameters.
Figure 19 — Left: General controller network for the planar arm control task. The motor output consists of two to four joint angles, depending on the number of links of the planar arm. Right: Combined network for DSL for the planar arm control task, consisting of the controller (top three layers) and the forward model (bottom three layers). The output of the forward model contains both the tip position and (optionally) additional constraint units.

4.3.3 Parameter settings

The parameter settings for all learning strategies and task conditions are reported in Tables 14 to 16. For FEL, DSL, DIM-MLP, and DIM-NGPCA, the free parameters of each learning strategy were handled in the same way as for the saccade learning task (see Sect. 3.3.2). There is only one exception: For DSL, the number of epochs used to train the FM was set to a quarter of $N_{FM}$.

For SLbA, one needs a strategy how to increase the number of learning examples and training epochs in each stage. This strategy was varied depending on the number of links and the selected constraint. It is reported in Tables 14 to 16 in the format $LE: a-b-c / EP: a-b-c$ with $a$ being the start value, $b$ the increase from stage to stage, and $c$ the maximum value. $LE$ indicates the number of learning examples, $EP$ the number of epochs. The other learning parameters are $\lambda_{SLbA}$ (fixed), $\lambda_\sigma$ (fixed), and $\sigma_0$ (varied systematically) as described in Sect. 4.3.1. The quality function is either $Q_0$, $Q_1$, or $Q_2$, depending on the applied constraint.

4.4 Results

General remarks The results for the 2-link arm are presented in Table 7, for the 3-link arm in Table 8, and for the 4-link arm in Table 9. The number of required exploration trials $N_{EX}$ for the
Table 7 — Results for the 2-link arm. Learning strategies are sorted in ascending order with regard to the required number of exploration trials. The corresponding best settings for the variable parameters are shown in the right column (for SLbA, the average required number of stages $k_{\text{max}}$ and its standard deviation (in brackets) are shown as well). For learning strategies which never succeeded in all learning passes the parameter combination with the maximally achieved average final controller quality $\bar{Q}_C$ and the percentage of failed passes with these settings is reported. For DIM-NGPCA, only one of the successful combinations of the number of PCA units $N$ and the number of eigenvectors $m$ for a certain value of $N_{\text{CON}}$ is given.

best successful combination of variable parameter values and the settings of these parameters are reported there (for DSL, $N_{\text{EX}}$ is computed as the sum of the number of exploration trials for the generation of the training set of the FM $N_{\text{FM}}^{\text{EX}}$ and for the subsequent controller training $N_{\text{EX}}^{\text{CON}}$; $N_{\text{FM}}^{\text{EX}} = N_{\text{FM}}$ for the planar arm task).

**General performance** Instead of reiterating the numbers given in Tables 7 to 9, I will only point out the most interesting results here. Moreover, the task conditions are abbreviated in the following: “L$x$Q$y$” is the task condition with $x$ links and quality function $Q_y$. The task conditions with noise are L2Q0N, L3Q0N, and L4Q0N. A learning strategy is designated as “successful” in a certain task condition if it is able to exceed the desired quality level $Q^*$ in all learning passes at least with one parameter combination; otherwise, it has failed in this task condition.

Generally, DIM-NGPCA is the fastest learning strategy. The only exceptions are conditions L3Q2 (DSL is best) and L4Q1 (SLbA is best). In condition L2Q0, DIM-MLP and DIM-NGPCA
3-link arm with $Q_0$ ($Q_0^* = 0.96$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>2696 (70)</td>
<td>$N_{CON} = 1000$, $N = 100$, $m = 4$</td>
</tr>
<tr>
<td>SLbA</td>
<td>62614 (34174)</td>
<td>$\sigma_0 = 0.4$, $k_{max} = 6$ (2)</td>
</tr>
<tr>
<td>FEL-J$^\dagger$</td>
<td>100615 (36808)</td>
<td>$\eta = 0.03$</td>
</tr>
<tr>
<td>DSL</td>
<td>106590 (43683)</td>
<td>$N_{FM} = 3000$, $\eta = 0.03$</td>
</tr>
<tr>
<td>FEL-J$^+$</td>
<td>10% failed ($Q_C = 0.959$)</td>
<td>$\eta = 0.035$</td>
</tr>
</tbody>
</table>

3-link arm with $Q_0$ and additional sensory noise ($Q_{ON}^* = 0.93$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>5439 (52)</td>
<td>$N_{CON} = 2000$, $N = 40$, $m = 4$</td>
</tr>
<tr>
<td>FEL-J$^\dagger$</td>
<td>90775 (29422)</td>
<td>$\eta = 0.02$</td>
</tr>
<tr>
<td>DSL</td>
<td>102530 (41040)</td>
<td>$N_{FM} = 5000$, $\eta = 0.03$</td>
</tr>
<tr>
<td>SLbA</td>
<td>232908 (168604)</td>
<td>$\sigma_0 = 0.6$, $k_{max} = 8$ (4)</td>
</tr>
<tr>
<td>FEL-J$^+$</td>
<td>15% failed ($Q_C = 0.929$)</td>
<td>$\eta = 0.04$</td>
</tr>
</tbody>
</table>

3-link arm with $Q_1$ ($Q_1^* = 0.97$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>8116 (123)</td>
<td>$N_{CON} = 3000$, $N = 60$, $m = 3$</td>
</tr>
<tr>
<td>SLbA</td>
<td>36325 (5284)</td>
<td>$\sigma_0 = 0.9$, $k_{max} = 3$ (0)</td>
</tr>
<tr>
<td>DSL</td>
<td>82765 (30371)</td>
<td>$N_{FM} = 7000$, $\eta = 0.03$</td>
</tr>
</tbody>
</table>

3-link arm with $Q_2$ ($Q_2^* = 0.86$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSL</td>
<td>53950 (13972)</td>
<td>$N_{FM} = 7000$, $\eta = 0.03$</td>
</tr>
<tr>
<td>SLbA</td>
<td>70691 (30483)</td>
<td>$\sigma_0 = 0.3$, $k_{max} = 4$ (0)</td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>100% failed ($Q_C = 0.80$)</td>
<td>$N_{CON} = 41000$, $N = 120$, $m = 1$</td>
</tr>
</tbody>
</table>

Table 8 — Results for the 3-link arm. See caption of Table 7 for further explanation. $k_{max}$ and its standard deviation are rounded down to integer values.
### 4-link arm with $Q_0$ ($Q_0^* = 0.94$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>18171 (175)</td>
<td>$N_{CON} = 7500$, $N = 200$, $m = 6$</td>
</tr>
<tr>
<td>FEL-J$^c$</td>
<td>48650 (15288)</td>
<td>$\eta = 0.015$</td>
</tr>
<tr>
<td>DSL</td>
<td>56100 (19144)</td>
<td>$N_{FM} = 3000$, $\eta = 0.01$</td>
</tr>
<tr>
<td>FEL-J$^+$</td>
<td>158050 (59411)</td>
<td>$\eta = 0.02$</td>
</tr>
<tr>
<td>SLbA</td>
<td>40% failed ($Q_C = 0.93$)</td>
<td>$\sigma_0 = 0.55$, $k_{max} = 38 (11)$</td>
</tr>
</tbody>
</table>

### 4-link arm with $Q_0$ and additional sensory noise ($Q_{0N}^* = 0.9$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>18141 (188)</td>
<td>$N_{CON} = 7500$, $N = 140$, $m = 4$</td>
</tr>
<tr>
<td>FEL-J$^c$</td>
<td>52500 (23972)</td>
<td>$\eta = 0.0125$</td>
</tr>
<tr>
<td>DSL</td>
<td>63850 (22477)</td>
<td>$N_{FM} = 5000$, $\eta = 0.01$</td>
</tr>
<tr>
<td>FEL-J$^+$</td>
<td>142850 (82594)</td>
<td>$\eta = 0.02$</td>
</tr>
<tr>
<td>SLbA</td>
<td>20% failed ($Q_C = 0.90$)</td>
<td>$\sigma_0 = 0.3$, $k_{max} = 36 (8)$</td>
</tr>
</tbody>
</table>

### 4-link arm with $Q_1$ ($Q_1^* = 0.97$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLbA</td>
<td>199904 (45781)</td>
<td>$\sigma_0 = 0.7$, $k_{max} = 6 (1)$</td>
</tr>
<tr>
<td>DSL</td>
<td>20% failed ($Q_C = 0.97$)</td>
<td>$N_{FM} = 15000$, $\eta = 0.01$</td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>40% failed ($Q_C = 0.968$)</td>
<td>$N_{CON} = 90000$, $N = 360$, $m = 4$</td>
</tr>
</tbody>
</table>

### 4-link arm with $Q_2$ ($Q_2^* = 0.86$)

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Exploration trials (SD)</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM-NGPCA</td>
<td>16976 (133)</td>
<td>$N_{CON} = 7000$, $N = 820$, $m = 1$</td>
</tr>
<tr>
<td>DSL</td>
<td>18750 (2826)</td>
<td>$N_{FM} = 5000$, $\eta = 0.015$</td>
</tr>
<tr>
<td>SLbA</td>
<td>376095 (247611)</td>
<td>$\sigma_0 = 0.5$, $k_{max} = 19 (11)$</td>
</tr>
</tbody>
</table>

Table 9 — Results for the 4-link arm. See caption of Table 7 for further explanation. $k_{max}$ and its standard deviation are rounded down to integer values.
share the first place with \( N_{\text{CON}} = 125 \). Under the application of sensory noise (L2Q0N), DIM-MLP performs worse (but still comes in second). For the 2-link and the 3-link arm, the winning margin of DIM-NGPCA compared to the second-best is large (for L2Q0, \( N_{\text{EX}} = 552 \) for DIM-NGPCA and \( N_{\text{EX}} = 23559 \) for SLbA; for L3Q0, \( N_{\text{EX}} = 2696 \) for DIM-NGPCA and \( N_{\text{EX}} = 62614 \) for SLbA). For the 4-link arm, this distance is smaller (for L4Q0, \( N_{\text{EX}} = 18171 \) for DIM-NGPCA and \( N_{\text{EX}} = 48650 \) for FEL-\( J^4 \)). In task condition L3Q2, DIM-NGPCA fails completely by a considerable margin (\( \bar{Q}_C = 0.80 \) compared to \( Q^*_2 = 0.86 \)), in condition L4Q1 only by a small amount. In the latter task condition, a slight decrease of the desired quality level would likely have resulted in a success of DIM-NGPCA with at least a few parameter combinations.

The performance of SLbA is mixed. SLbA gets the first place in task condition L4Q1 and the second place in the task conditions L2Q0, L3Q0, L3Q1, and L3Q2. In the conditions L4Q0 and L4Q0N, none of the parameter combinations for SLbA is successful.

Comparing DSL and FEL-\( J^4 \), the required number of exploration trials \( N_{\text{EX}} \) is in the same order of magnitude. In condition L2Q0N, DSL is faster, in conditions L3Q0, L3Q0N, L4Q0, and L4Q0N, FEL-\( J^4 \) is faster (even if only the number of exploration trials for controller training \( N_{\text{CON}}^{\text{EX}} \) in DSL is considered: \( N_{\text{CON}}^{\text{EX}} = N_{\text{EX}} - N_{\text{FM}} \) for the planar arm task). First and second places are reached in the following task conditions: L3Q2 (first place for DSL); L3Q0N, L4Q0, and L4Q0N (second place for FEL-\( J^4 \)); L4Q2 (second place for DSL). FEL-\( J^+ \) is generally slower or worse than FEL-\( J^4 \). Only in task conditions L4Q0 and L4Q0N some of the parameter combinations are successful for FEL-\( J^+ \); nevertheless, FEL-\( J^+ \) still requires around three times as many exploration trials \( N_{\text{EX}} \) as FEL-\( J^4 \).

For DIM-NGPCA, the number of required exploration trials \( N_{\text{EX}} \) increases with the number of links. Comparing for example the conditions L2Q0, L3Q0, and L4Q0, DIM-NGPCA requires \( N_{\text{EX}} = 552 \) for the 2-link arm, \( N_{\text{EX}} = 2696 \) for the 3-link arm, and \( N_{\text{EX}} = 18171 \) for the 4-link arm. Thus, the increased complexity and dimensionality of the sensorimotor space for larger link numbers has a direct impact on the required learning effort for DIM-NGPCA; a similar picture emerges for SLbA. On the contrary, FEL-\( J^4 \) and DSL seem to perform the better the more links are involved. The results for FEL-\( J^4 \) are: no success at all in condition L2Q0, \( N_{\text{EX}} = 100615 \) in condition L3Q0, and \( N_{\text{EX}} = 48650 \) in condition L4Q0. Moreover, FEL-\( J^+ \) in only successful for the 4-link arm and not for smaller link numbers.

The application of sensory noise has only a small impact on the performance of the different learning strategies. For the 2-link arm, the ranking order does not change between conditions L2Q0 and L2Q0N; without noise, DSL and FEL-\( J^4 \) are not successful at all, with noise, at least some parameter combinations allow successful learning for these strategies. Comparing conditions L3Q0 and L3Q0N for the 3-link arm, DSL and FEL-\( J^4 \) profit again from the application of noise which causes a slight reduction in the number of required exploration trials \( N_{\text{EX}} \). On the contrary, SLbA suffers from the noise through a quadruplication of \( N_{\text{EX}} \); this changes the ranking order as well. For the 4-link arm, the application of noise neither changes the ranking order nor the number of required exploration trials considerably between conditions L4Q0 and L4Q0N.

The influence of the constraints on the learning performance is rather inconsistent. Generally, SLbA profits from the first constraint in conditions L3Q1 and L4Q1. For SLbA, \( N_{\text{EX}} = 62614 \) in condition L3Q0 compared to \( N_{\text{EX}} = 36325 \) in condition L3Q1. A similar relationship holds for the 4-link arm: no success in condition L4Q0 vs. \( N_{\text{EX}} = 199904 \) in condition L4Q1. In
the latter condition, SLbA is the only successful learning strategy although DIM-NGPCA and DSL only fail by a small margin. A slight decrease of the desired quality threshold $Q^*$ would likely have changed the picture. In condition L3Q1, DIM-NGPCA has still the lead, but with a triplication of $N_{\text{EX}}$ compared to condition L3Q0. For the second constraint, the results are very different between the 3-link and the 4-link arm. Comparing conditions L3Q0 and L3Q2, the ranking order of learning strategies reverses. SLbA stays on the same performance level, while DSL improves from $N_{\text{EX}} = 106590$ (L3Q0) to $N_{\text{EX}} = 53950$ (L3Q2) and DIM-NGPCA fails completely in condition L3Q2 by a considerable margin. For the 4-link arm, the ranking order does not change between conditions L4Q0 and L4Q2. DIM-NGPCA stays first with roughly the same number of required exploration trials. DSL stays second but with a much better performance ($N_{\text{EX}} = 18750$ for L4Q2 compared to $N_{\text{EX}} = 56100$ for L4Q0); SLbA improves as well from no success at all (L4Q0) to $N_{\text{EX}} = 376095$ (L4Q2).

**Statistical tests**  
The post-hoc statistical analysis is restricted to the comparison between learning strategies within each task condition: Every learning strategy was matched with every other strategy (excluding the failed ones). The compared measure was the mean number of required exploration trials (see Tables 7 to 9). The computed pairwise t-tests (two-sided, for independent samples) yielded significant results ($p < 0.05$) in all pairwise comparisons except for DIM-MLP vs. DIM-NGPCA in condition L2Q0 (with $N_{\text{CON}}$ being equal, thus inevitable result) and FEL-$J^t$ vs. DSL in conditions L2Q0N, L3Q0, L3Q0N, L4Q0, L4Q0N. These results suggest that it is not possible to draw any firm conclusions from the direct comparison between FEL-$J^t$ and DSL. Otherwise, the statistical analysis supports the reliability of the reported data.

**Additional observations**  
In examining the results for DIM-NGPCA for all variable parameter combinations, interesting observations were made regarding the interaction of the task characteristics and the behavior of NGPCA. First of all, for the conditions without constraint (L2Q0, L2Q0N, L3Q0, L3Q0N, L4Q0, and L4Q0N), the lower limit for the number of eigenvectors $m$ in successful learning passes seems to be equal to $L$, the number links. Only if a huge number of PCA units $N$ is available to the network (in conditions L4Q0 and L4Q0N for $N > 200$), parameter combinations with $m < L$ succeed consistently. The upper limit for $m$ for successful performance seems to be $L + 2$, the overall number of dimensions of the sensorimotor space. Only if additional noise is applied and $N$ is large in relation to the size of the training set $N_{\text{CON}}$ (as in conditions L3Q0N and L4Q0N), parameter combinations with $m > L$ can be unfavorable. These results show that there is at least a partial tradeoff between $m$ and $N$, and that network performance can drop if the data manifold is too crowded with too many PCA units with too many eigenvectors.

Applying the second constraint, DIM-NGPCA fails completely in condition L3Q2. In condition L4Q2, success is only possible for $m \leq 2$ eigenvectors. The smallest number of learning examples with $N_{\text{CON}} = 7000$ is required for the combination of $m = 1$ and $N = 820$. For $m = 2$, at least $N_{\text{CON}} = 13000$ learning examples are necessary (with $N = 520$). Overall, at least $N = 420$ PCA units are necessary (starting with $N_{\text{CON}} = 22000$ and $m = 2$). This strange behavior of NGPCA can be explained by the special characteristic of the second constraint which has already been discussed before in Sect. 4.3.1 for DIM-NGPCA. Basically, the second constraint forces the NGPCA network to extrapolate for the generation of motor commands. Obviously, this extrapolation only yields the desired results if the training data manifold is represented by a densely packed large number of units with only one or two directions for
Figure 20 — Distribution of 1000 learning examples for DIM-NGPCA in the \((y_1, y_2)\) working space for the different planar arms. Only the position of the tip \(y\) is shown as black dot.

linear interpolation. This network structure reminds of a lookup table where interpolation plays a subordinate role. Thus, no far-reaching extrapolation along the direction of the eigenvectors takes place, but instead the best-fitting motor output is recalled from the PCA unit closest to the desired sensory state.

Figure 20 shows the distribution of 1000 learning examples in the working space which are collected for DIM-NGPCA for the different planar arms (the tip position \(y\) is shown as black dot). These learning examples are collected by generating random motor commands \(u = \theta\) and assessing the plant output \(y = P(\theta)\) afterwards. Only if \(y\) is within the operating range of the desired sensory states \(y^*\), the learning example \([y \rightarrow u]\) is added to the training set. Therefore, the ratio between the number of required exploration trials \(N_{\text{EX}}\) and the size of the training set \(N_{\text{CON}}\) for DIM-MLP/DIM-NGPCA is larger than one. It amounts to 4.5 for the 2-link arm, to 2.7 for the 3-link arm, and to 2.4 for the 4-link arm. Although this ratio works in favor for DIM-NGPCA for larger link numbers, Fig. 20 shows that larger link numbers are actually worse: The distribution of learning examples becomes more and more unbalanced, lumping around the origin of the working space while the outer corners of the operating range are only sparsely populated with learning examples. Therefore, the overall number of learning examples has to increase to guarantee that controller training is successful in the periphery as well. For even longer planar arms, this effect will work strongly against DIM-NGPCA while the ratio \(N_{\text{EX}}/N_{\text{CON}}\) will converge to a value around 2 (since the operating range takes roughly half of the working space in which the main part of the learning examples is generated). This uneven distribution of learning examples in the training set for DIM-NGPCA is an indirect consequence of the lacking goal-directedness of DIM.

With regard to SLbA it is interesting to note that the number of required stages increases strongly with the number of links. For example, in condition L2Q0 \(k_{\text{max}}\) amounts to 3.4, in condition L3Q0 to 6, and in condition L4Q0 to 30 (only considering the 60% of successful learning passes in this condition). With noise, \(k_{\text{max}}\) gets slightly larger, whereas both constraints help to reduce \(k_{\text{max}}\) considerably (e.g., \(k_{\text{max}} = 6\) in condition L4Q1). As examples for the course of learning with SLbA, Table 10 reports the average quality \(Q_{\text{PS}}\) of the training set side by side with the average controller quality \(Q_C\) on a test set for the different stages of a single learning pass for the task conditions L3Q0, L3Q0N, L3Q1, and L3Q2. The quality values in this table demonstrate that averaging has here the same positive effect (as theoretically supposed) as in the saccade learning task: At least in the first and second stage, the controller quality is larger than the quality of its learning examples. In later stages, especially in condition L3Q0N with
Figure 21 — Graphical results of SLbA for $Q_0$ for the 3-link arm, depicted in the $(y_1, y_2)$ working space. The controller performance after different stages is illustrated by black bars which indicate the distance between the desired tip position $y^*$ and the tip position $y$ resulting from the arm posture which is generated by the controller output $u$. These error bars are shown for a regularly spaced grid of desired tip positions covering the whole operating range. Moreover, for 14 desired tip positions at the outer border of the operating range the corresponding controller-generated arm posture is shown in gray color.

sensory noise, learning slows down and relies mainly on the improvement of the training set from stage to stage.

Figure 21 further illustrates the course of learning for SLbA for the 3-link arm with quality function $Q_0$ (condition L3Q0). The controller performance after all stages from the very first to the very last is indicated with black error bars (difference between desired tip position $y^*$ and the tip position $y$ resulting from the controller output $u = C_k(y^*)$). Moreover, the corresponding arm posture $u$ is depicted as well for 14 positions $y^*$ at the border of the operating range. The presented learning pass has been carried out with the optimal parameter settings from Table 8. Learning progresses noticeably from stage to stage with shorter and shorter error bars. Depending on the region within the operating range, learning takes place with different speed, the outer corners being the most difficult part.

Figure 22 shows the final results for all learning strategies in condition L4Q0; controllers are trained with the optimal parameter settings from Table 9. It is noticeable that the learning strategies which rely on local linear approximation by $J^i$ or its approximation (FEL-$J^i$ and
Figure 22 — Comparison of various learning strategies for $Q_0$ for the 4-link arm. For further explanation see the caption of Fig. 21. Since SLbA is not capable to reach the desired quality level in this task condition, the error bars for SLbA are longer on average than for the other learning strategies.
Table 10 — Comparing the average quality $Q_{PS}$ of the learning examples in the training set with the average controller quality $Q_C$ on a test set for the different stages of a single learning pass of SLbA for the task conditions L3Q0, L3Q0N, L3Q1, and L3Q2.

<table>
<thead>
<tr>
<th>Stage $k$</th>
<th>L3Q0</th>
<th>L3Q0N</th>
<th>L3Q1</th>
<th>L3Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Q_{PS}$</td>
<td>$Q_C$</td>
<td>$Q_{PS}$</td>
<td>$Q_C$</td>
</tr>
<tr>
<td>1</td>
<td>0.53 &lt; 0.59</td>
<td>0.52 &lt; 0.57</td>
<td>0.46 &lt; 0.85</td>
<td>0.49 &lt; 0.60</td>
</tr>
<tr>
<td>2</td>
<td>0.84 &lt; 0.90</td>
<td>0.84 &lt; 0.88</td>
<td>0.94 &lt; 0.96</td>
<td>0.76 ≈ 0.76</td>
</tr>
<tr>
<td>3</td>
<td>0.96 &gt; 0.94</td>
<td>0.95 &gt; 0.92</td>
<td>0.98 &gt; 0.97</td>
<td>0.84 ≈ 0.84</td>
</tr>
<tr>
<td>4</td>
<td>0.98 &gt; 0.96</td>
<td>0.97 &gt; 0.92</td>
<td>0.88 &gt; 0.87</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.97 &gt; 0.93</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Fig. 23, the influence of the different constraints on the final controller output is compared for SLbA for 4-link arm. For $Q_1$ (the “maximum symmetry” constraint), it is clearly visible from the controller-generated arm postures that this symmetry has been successfully achieved. Without any constraint ($Q_0$), the arm postures show barely any symmetry, with constraint $Q_2$, the results are in-between.

4.5 Discussion

In contrast to the saccade learning task, the plant in the planar arm task is non-linear. As result, DIM-NGPCA is the clear performance leader while the local linear approximation techniques

...
like FEL and DSL perform less well. The strength of DIM-NGPCA is that it combines the simple learning strategy of DIM with the capability of NGPCA to learn one-to-many mappings. DIM-NGPCA offers a unique approach to the one-to-many problem by storing multiple solutions of the inverse kinematics simultaneously if encountered during learning. For the recall, an algorithm is used which generates one of these solutions as network output (see Sect. A.2). This solves one of the two main computational problems of DIM. The second problem, its lacking goal-directedness, remains but is not that important for planar arms with up to 4 links. However, with an increasing number of links the distribution of learning examples in the operating range becomes more and more unfavorable as shown in the results section. Thus, at some point the missing goal-directedness might become an issue. Furthermore, one would expect that the application of the first constraint ($Q_1$) would exacerbate this problem since only the learning examples in a small subset (the ones close to “maximum symmetry”) represent the desired sensory outcome in this case. In practice, the first constraint has only a mild negative impact on the performance of DIM-NGPCA (at least for the 3-link arm).

Overall the performance of DIM-NGPCA becomes less reliable when learning constraints are enforced by additional units for the desired sensory input. Especially for the “energy minimization” constraint ($Q_2$), the successful NGPCA networks require at least 420 PCA units with exactly 2 eigenvectors and remind of a lookup table. Such an NGPCA network has 10080 parameters compared to the 74 weights of the MLP controller networks which perform successfully with DSL or SLbA. One can state in favor of DIM-NGPCA that it even manages to learn a task which requires extrapolation, but on the downside this is very expensive in terms of network complexity, storage requirements, and computational effort during recall. Sensory noise has a negative impact on DIM-NGPCA as well, but not up to a point where the performance leadership of DIM-NGPCA is endangered. To learn noisy data, larger training sets are required. In the 2-link task, where no one-to-many mapping is involved, DIM-MLP performs just as good as DIM-NGPCA in the condition without noise, but suffers more from noise.

Surprisingly, the performance of the local linear approximation techniques (FEL and DSL) increases for larger link numbers. These learning strategies might surpass DIM-NGPCA at some point. However, this has not been tested in the present study. FEL-$J^I$ shows the best performance, closely followed by DSL and with a larger margin by FEL-$J^+$. Thus, the best gain matrix is the exact transpose of the Jacobian $J_{u,x}^t$. The estimate $J_{u,x}^t$ which is generated by the FM in DSL is slightly worse (but these are not firm conclusions; see the statistical analysis). The pseudoinverse $J_{u,x}^+$ is the worst choice as gain matrix; for the planar arm, one would clearly prefer $J_{u,x}^t$. However, based on the available data it is not clear to what extent this ranking order of gain matrices generalizes to other learning tasks. From the local linear approximation techniques, DSL was the only one tested with constraints. These were imposed on the learning task by adding sensory output units to the FM. Each of these output units had a fixed desired value signaling perfect constraint fulfillment. Generally, DSL experienced a speedup of learning through the constraints, only for the 4-link arm the first constraint (“maximum symmetry”) caused DSL to fail by a very small margin. Additional sensory noise has only a small impact on FEL and DSL: for the 2-link arm, it even helped DSL and FEL-$J^I$ to get into the set of successful learning strategies. In addition, FEL and DSL cope very well with the one-to-many mapping of the planar arm task by converging to distinct solutions which vary between learning passes due to the stochastic nature of network initialization and training pattern generation.

The performance of SLbA decreases with an increasing number of links in the no-constraint conditions. The most likely reason is that the one-to-many nature of the learning task becomes
the more dominant the more links are involved. For the 2-link arm, there is no ambiguity at all; here, SLbA gets the second place while FEL and DSL fail completely (at least without noise). For the 3-link arm, SLbA still comes in second although the one-to-many problem is present and — even worse — the non-convexity of the solution sets (see Sect. 4.1.1) violates one precondition of learning by averaging: It is not guaranteed that the average of two arm postures has a quality which is larger than the quality of the worse of these two postures. Although this precondition is violated, SLbA is fairly successful for the 3-link arm, and averaging over learning examples has the expected positive effect for subsequent controller performance (see Table 10). But for the 4-link arm, SLbA finally fails because it cannot cope with the increased ambiguity. Nevertheless, as soon as the ambiguity is removed by the first constraint (there is only one posture for each tip position with “maximum symmetry”), SLbA even becomes the best learning strategy while DSL and DIM-NGPCA fail (at least for the 4-link arm). The second constraint (“minimum energy”) does not help to reduce the ambiguity completely, instead it introduces a tradeoff between reaching precision and the distance of the arm posture to the resting position in joint space. However, this constraint helps SLbA as well to become successful for the 4-link arm although only with a huge number of required exploration trials compared to the other learning strategies. The impact of sensory noise on SLbA is rather negative for the 2- and the 3-link arm (duplication respectively triplication of required exploration trials), but for the 4-link arm additional noise lifts SLbA nearly into the group of successful learning strategies: With noise, SLbA fails only by a very tight margin.

In conclusion, for non-linear and redundant task domains like the planar arm, the application of DIM-NGPCA can be recommended as long as the negative impact of the lacking goal-directedness of DIM is not too strong, for example due to an uneven distribution of learning examples in the operating range. Otherwise, DIM-NGPCA is fairly efficient and resistant to noise. If additional constraints have to be incorporated into the learning task, SLbA is a strong contender since the constraints can be specified in a straightforward fashion in the quality function which is used during learning. Moreover, SLbA proved to be the overall most reliable learning strategy if constraints are applied. On the downside, SLbA slows down if sensory noise is present and does not cope well with one-to-many mappings and non-convex solution sets. The local linear approximation techniques (FEL and DSL) show no distinct advantage other than their resistance to noise. They are not that fast and also not that reliable with constraints (speaking of DSL; for FEL, constraints get even more complicated since one needs to determine the Jacobian of the plant which is extended by the constraint output units analytically). However, the available data shows the trend that DIM-NGPCA slows down with an increasing number of links while DSL and FEL speed up. Thus, for motor tasks with high-dimensional sensorimotor spaces DSL and FEL may be faster than DIM-NGPCA since the latter is hampered by the lacking goal-directedness in the generation of learning examples.
5 General Discussion

5.1 Evaluation of the experimental findings

In summary, the initial research questions have been answered by the results of the two learning tasks in the following way: For linear plants, local linear approximation techniques like FEL and DSL work very well while DIM-NGPCA is a good allround performer for both linear and non-linear plants. DIM-NGPCA deals well with one-to-many mappings as do FEL and DSL while SLbA fails if the redundancy of the learning task is too large. With regard to sensory noise, the most considerable performance drop can be observed for SLbA and DIM. If additional learning constraints are imposed, the most reliable learning strategy is SLbA (closely followed by DSL).

However, it has to be emphasized that the performance of DIM-related techniques like DIM-NGPCA depends heavily on two task characteristics: First, how much sampling effort is required to find learning examples which are close to the desired operating range of the controller in sensory space, and second, how much sampling effort is required to fill all areas within the operating range with at least the minimum of learning examples needed for good interpolation during controller learning. Although these task characteristics are not directly linked to the dimensionality of the combined input/output space of the controller, one can expect that a larger number of dimensions often implies enlarged sampling effort (as it does for the planar arm).

Additional learning constraints have only been applied in the planar arm task. Depending on the learning strategy, they were implemented differently. SLbA adjusts itself easily to the constraints since they are directly encoded in the quality function. For DIM-NGPCA and DSL, constraints were imposed by additional desired sensory input units of the controller (DIM-NGPCA) or by additional sensory output units of the FM (DSL). For DIM-NGPCA and DSL, this constraint implementation caused less reliable learning success, while SLbA partly even relied on additional constraints for successful learning to reduce the ambiguity of the one-to-many mappings. For learning tasks with constraints SLbA is the most direct approach without the need to modify the controller input or to specify additional plant outputs, and moreover the most reliable approach throughout the studies in this paper.

5.2 Discussion of the experimental methods

The comparisons in these studies rely on the number of required exploration trials. This performance indicator works in favor of learning strategies which are based on batch learning, this means collecting first a set of learning examples and using it afterwards multiple times for controller adaptation. In contrast, learning strategies like FEL and DSL cannot reuse learning examples, thus every cycle of controller adaptation requires additional exploration trials to generate a new learning example. If one dismisses batch learning, the number of required exploration trials for DIM-MLP/DIM-NGPCA and SLbA (and also for DSL due to FM training) gets much larger. For example, the DIM-NGPCA networks with optimal parameter settings in condition L3Q0 for the planar arm (3-link arm without constraint and without noise) would require around seven times as many exploration trials if they were not allowed to reuse the collected learning examples during controller adaptation. However, in this task condition DIM-NGPCA would remain fastest even with this additional burden.
Other performance indicators besides the required number of exploration trials could be the number of required network adaptation cycles (as suggested in the previous paragraph), the overall computational effort, the minimum required number of adjustable network parameters, etc. In these studies, I referred to the number of exploration trials since these are connected to “real” movements of the agent. I judge these real movements to be more relevant than indicators of computational effort because the latter are not only linked to the motor learning strategy but also to the applied neural network algorithm. For the same reason, I evaluated batch learning strategies just by the number of required exploration trials without any attempt to make this number equivalent to the online learning strategies as discussed in the previous paragraph. Future neural network algorithms (e.g., an advanced version of NGPCA) may allow much faster learning so that a single cycle through the training set would be sufficient for successful controller learning. The absolute number of learning examples in the training set, on the other hand, has a lower limit to allow precise interpolation between the provided data points for any neural network algorithm, and this number is directly linked to the required number of exploration trials.

5.3 Theoretical summary of the learning strategies

Table 11 summarizes and compares the most important properties of FEL, DSL, DIM, and SLbA. FEL and DSL are pure online learning strategies, SLbA is restricted to batch learning, and DIM can be used in both modes. With regard to one-to-many mappings, theoretical considerations and experimental results are mostly congruent. FEL and DSL converge to one of the possible solutions, and DIM in combination with abstract recurrent networks of the NGPCA type copes also well with one-to-many problems. An NGPCA network even stores all learned outputs for a specific input simultaneously. However, the recall algorithm used in this study (see App. A) only generates one of the many outputs (which one depends on quasi-random fluctuations in the overall configuration of the local PCA units). SLbA even slightly exceeded the theoretical expectations since it could handle at least one one-to-many problem: the 3-link arm without constraints. However, as soon as the ambiguity in the learning task gets more prevalent, additional learning constraints are required for SLbA to eliminate the one-to-many characteristic of the learning task.

FEL and SLbA are the only strategies which are clearly goal-directed towards the desired sensory outcomes during the learning process. DIM is not goal-directed at all, and DSL suffers from limitations: While the controller learning in DSL is goal-directed, the preceding or accompanying adaptation of the FM is not. In practice, the lacking goal-directedness of DIM did not stop it from being the overall most efficient learning strategy (at least in its DIM-NGPCA variant). Nearly all of the learning strategies are fully adaptive insofar as no analytical knowledge about the plant is required (except for FEL for complex plants), thus they are suited for adaptive motor control in the kinematic domain. Being fully adaptive is also a precondition for biological plausibility. This aspect is discussed in more depth in the next subsection.

5.4 Biological plausibility

FEL is biologically plausible for simple kinematic plants, for example for oculomotor control (Dean et al., 1994; Gancarz and Grossberg, 1999). Here, no precise analytical knowledge of the
<table>
<thead>
<tr>
<th>Learning Strategy</th>
<th>FEL</th>
<th>DSL</th>
<th>DIM</th>
<th>SLbA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch learning</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Online learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Usable for one-to-many</td>
<td>Yes, converges to one solution</td>
<td>Yes, converges to one solution</td>
<td>Only with abstract recurrent networks</td>
<td>Only with additional constraints</td>
</tr>
<tr>
<td>problems</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal-directed</td>
<td>Yes</td>
<td>(Yes)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Fully adaptive</td>
<td>Only for simple plants*</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Biologically plausible</td>
<td>Only for simple plants*</td>
<td>Questionable</td>
<td>(Yes)</td>
<td>Questionable</td>
</tr>
</tbody>
</table>

* complex plants require analytical knowledge

Table 11 — Comparison of learning strategies. A “yes” in round brackets indicates a restricted affirmation.

 plant is required. The same applies to dynamic motor control where the necessary torques for a desired trajectory in joint angle space have to be determined. For this application, simple pre-wired feedback controllers are sufficient, and a large body of research has been dedicated to develop models of the underlying neural brain mechanisms (Magescas et al., 2009; Schweighofer et al., 1998; Wolpert et al., 1998).

DSL is in principle biologically even more plausible than FEL since the pre-wired feedback controller is replaced by an adaptive forward model which is acquired for the sensorimotor domain of the specific motor task. However, DSL requires that this forward model is used in reverse direction for motor learning, and while this is an elegant computational solution, the backpropagation of the sensory error through a multi-layer network is biologically not plausible (Arbib, 1998). Future research might alleviate this issue by offering more realistic alternatives to classical backpropagation (e.g., see Körding and König, 2001).

Kawato (1990) states as argument against the biological plausibility of DIM the supposed need for neural rewiring: During learning, the real sensory state after the movement is fed as input to the inverse model, while during the later application of the inverse model the same input lines are used to transmit the desired sensory state. To overcome this issue, I propose a slightly more complex model in which these input lines are connected to a set of neurons which form an associative memory for sensory states. These neurons learn to encode the sensory states the organism encounters, and during learning of the inverse model the activation of these neurons represents the current sensory state. Later on these neurons are activated by other brain areas to represent imagined desired sensory states, and in this way the inverse model can retrieve real and desired sensory states via the same input lines. Thus, I argue here that DIM is in fact biologically plausible. Actually, DIM corresponds nicely to the circular reactions described by Piaget (1952): Children elicit random sensory effects by random exploratory movements; afterwards, movements with desirable sensory outcome are repeated (see also Kuperstein, 1988).

The basic idea of learning by averaging — to learn the average of several failed but not completely erratic motor commands — seems to be a viable approach for biological organisms.
Unfortunately there is also one downside here: SLbA in its current form requires the storage of a set of learning examples before the actual training takes place. Batch learning like this is biologically not plausible because it seems to be rather unlikely that the CNS is capable of storing thousands of learning examples with the necessary precision over a long period of time. In an extended version of the present study an attempt was made to overcome this problem by combining learning by averaging with online learning. This attempt was only successful for the saccadic plant; for the non-linear arm kinematics it failed (Schenck, 2008). However, alternative neural network algorithms instead of the MLP might perform better in this respect, thus the current need for batch learning does not completely rule out the biological plausibility of SLbA. Even on the contrary, SLbA is compatible with approaches in which the kinematic control of the human arm is modeled on the basis of cost functions to explain how the one-to-many problem in this area is solved by the brain (Cruse et al., 1990; Grea et al., 2000). These cost functions could be used as constraints for SLbA, and this combination might result in a plausible model for the human learning of kinematic arm control.

In summary, none of the learning strategies can be designated as biologically plausible without mentioning restrictions or the need for modifications or future developments in the area of neural learning algorithms.

5.5 Recommendations for robotic applications

The following recommendations can be derived from the results and the theoretical considerations:

- For simple linear plants, DIM-MLP and FEL are well suited. Both are simple to implement and quite efficient. FEL has the additional advantage that it can deal with one-to-many mappings if required.

- For more complex and non-linear plants, DIM-NGPCA clearly proved to be the most efficient and reliable learning strategy and is thus strongly recommended. If the lacking goal-directedness of DIM-NGPCA is a problem for the learning task, DSL is a practical alternative since it is also fully adaptive and can as well deal with one-to-many mappings.

- If the motor task involves cost functions to favor certain types of movements, SLbA offers a straightforward way to incorporate these constraints into the learning process. In this case, this advantage of SLbA might outweigh its complexity and its comparatively slow learning speed.

5.6 Final conclusions

In conclusion, DIM in combination with NGPCA is the overall winner of these comparison studies. This contradicts the view that DIM is the least favorable approach among DIM, FEL, and DSL although this view is often expressed in the literature (e.g., Jordan, 1996; Kawato, 1990). It could be shown that the first argument against DIM, its inability to deal with one-to-many mappings, can be overcome by abstract recurrent neural networks. Moreover, the impact
of its second weakness, the lacking goal-directedness, depends heavily on the task characteristics, thus one can expect DIM to be fairly efficient for many kinematic control tasks like in the presented studies. The third criticism (with regard to biological plausibility), the hypothesized need for neural rewiring, can be counteracted by neural architectures where the input layer of the DIM controller is a neural map which can be activated both from sensory afferences and from memory (to represent desired sensory states). To resolve the remaining second weakness of DIM more convincingly, future research might explore goal-directed search strategies in motor space to replace the random sampling for the generation of learning examples.
A Neural Gas Principal Component Analysis (NGPCA)

NGPCA combines the vector quantization method “neural gas” (NG) (Martinetz et al., 1993) with local principal component analysis (PCA) (Hoffmann and Möller, 2003; Möller and Hoffmann, 2004). The goal is to represent a high-dimensional data distribution by a model with only a small number of parameters (compared to the amount of training data).

PCA is a method of dimension reduction in multivariate data (Diamantaras and Kung, 1996). The high-dimensional pattern space (with \( n \) dimensions) is approximated by a subspace which is spanned by the first \( m \) principal eigenvectors of the data covariance matrix (with \( m \leq n \)). In the following, \( W \) denotes the matrix of estimated eigenvectors \( w_j, j = 1, ..., m \) (one vector per column). \( W \) is of size \( n \times m \). The eigenvalue \( \lambda_j \) of \( w_j \) is equal to the variance of the data distribution in this eigendirection. The \( m \times m \) matrix \( \Lambda \) is a diagonal matrix containing the values \( \lambda_j \). \( c \) is the mean vector of the data distribution and the center of the PCA. Altogether, the multivariate data is just represented by the matrices \( W \) and \( \Lambda \) and the vector \( c \).

A complete NGPCA model consists of \( N \) local PCAs, each described by the tuple \( \{ c_i, W_i, \Lambda_i, \lambda_i^* \} \), with \( i = 1, ..., N \). \( \lambda_i^* \) represents an estimate of the eigenvalue in each of the remaining \( n - m \) minor eigendirections. In geometrical terms, each unit \( i \) is a hyperellipsoid centered at \( c_i \) (see Fig. 24). In the \( n - m \) minor eigendirections, it has the form of a hypersphere. Each local PCA unit adapts to the structure of the data distribution in the neighborhood of \( c_i \).

A.1 Learning

During the learning process the units’ parameters are updated such that the overall NGPCA network approximates the training data distribution \( \{ x_t \} \) afterwards (\( t \) is the training step index which runs from 1 to a predefined maximum \( T \)). In each training step, the adaptation of the centers \( \{ c_i \} \) is carried out in the same way as in the NG algorithm, only the distance measure for the ranking of the units is modified and computed as the sum of a volume–normalized Mahalanobis distance plus reconstruction error (Hoffmann, 2004; Möller and Hoffmann, 2004):

\[
\begin{align*}
  d_i(x_t) &= \left( \xi_i^T W_i \Lambda_i^{-1} W_i^T \xi_i + \frac{1}{\lambda_i^*} (\xi_i^T \xi_i - \xi_i^T W_i W_i^T \xi_i) \right) V^{2/n} \\
  &\quad \text{for } \lambda_i^* \text{ determined as } \lambda_i^* = \frac{\sigma_i^2}{n - m} .
\end{align*}
\]  

(10)

\( \xi_i \) is the difference between the unit center \( c_i \) and the presented data point \( x_t; \xi_i = x_t - c_i \). \( V \) is proportional to the volume of the hyperellipsoid and computed as \( V = \sqrt{|\Lambda| \lambda_i^{n-m}} \).

\( \lambda_i^* \) in Eqn. (10) is determined as

\[
\lambda_i^* = \frac{\sigma_i^2}{n - m} .
\]

\( \sigma_i^2 \) is the residual variance of unit \( i \). It is computed by an iterative update:

\[
\sigma_i^2 \leftarrow \sigma_i^2 + \alpha_i \cdot (\xi_i^T \xi_i - \xi_i^T W_i W_i^T \xi_i - \sigma_i^2)
\]

\( \alpha_i \) is a unit-specific learning rate: \( \alpha_i = \epsilon \cdot h_{\rho}[r_i(d)] \). As in the NG method, the neighborhood range \( \rho \) and the learning rate \( \epsilon \) decrease exponentially. \( h_{\rho} \) is the neighborhood function, a Gaussian, and \( r_i \) is the rank of unit \( i \) for the presented data point \( x_t \).
In addition to the centers \( \{ c_i \} \), the matrices \( \{ W_i \} \) and \( \{ \Lambda_i \} \) have to be updated with each pattern presentation \( x_t \). For this purpose, an online PCA method is applied:

\[
W_i, \Lambda_i \leftarrow \text{PCA}\{ W_i, \Lambda_i, \xi_i, \alpha_i \}
\]

Throughout the reported studies, the online PCA method suggested by Möller (2002) is used. It works by the interlocking of RRLSA (Ouyang et al., 2000), a neural method for PCA based on the recursive least squares method, and the Gram-Schmidt method for orthonormalization (Golub and van Loan, 1996). Since this interlocked online PCA method does not guarantee perfect orthogonality, after every \( T_{\text{ortho}} \) learning steps an explicit Gram-Schmidt orthonormalization is carried out for all units.

In the beginning of the learning process, the centers \( \{ c_i \} \) are set to randomly chosen data vectors from the training set \( \{ x_t \} \). The matrices \( \{ W_i \} \) are initialized to random orthonormal systems, the eigenvalues in \( \{ \Lambda_i \} \) to a constant \( \lambda(0) \).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>Number of training steps</td>
</tr>
<tr>
<td>( T_{\text{ortho}} )</td>
<td>Orthogonalization enforcement cycles</td>
</tr>
<tr>
<td>( \epsilon(0) )</td>
<td>Initial value for parameter ( \epsilon ) (learning rate of NG)</td>
</tr>
<tr>
<td>( \epsilon(T) )</td>
<td>Final value for parameter ( \epsilon )</td>
</tr>
<tr>
<td>( \rho(0) )</td>
<td>Initial value for parameter ( \rho ) (neighborhood range of NG)</td>
</tr>
<tr>
<td>( \rho(T) )</td>
<td>Final value for parameter ( \rho )</td>
</tr>
<tr>
<td>( \sigma^2(0) )</td>
<td>Iteration start value for the residual variance</td>
</tr>
<tr>
<td>( \lambda(0) )</td>
<td>Initial eigenvalues</td>
</tr>
</tbody>
</table>

Table 12 — Learning parameters for NGPCA.

### A.2 Recall

After training, the tuple \( \{ c_i, W_i, \Lambda_i, \lambda^*_i \} \) has been adapted for every unit \( i \) such that the overall model represents the data distribution from which the training vectors \( \{ x_t \} \) have been drawn. To use this NGPCA model for pattern recall in a feedforward fashion, some dimensions of the data space have to be defined as input and the others as output. Hoffmann and Möller (2003) presented a method for pattern recall with arbitrary input-output assignments which generates a unique output pattern for each input pattern via the NGPCA model. This method guarantees that one arrives at the globally optimal output. For this reason, this recall method is used throughout the studies in this paper.
Figure 24 — NGPCA model with four local PCA units, depicted as ellipsoids in a two-dimensional data space. The arrows within each ellipsoid indicate the principal components. The data points $\{x_i\}$ belonging to the data distribution are shown as gray dots.
## B Experimental Settings

### 13. Saccade learning task

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>without retinal noise</th>
<th>with retinal noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fixed parameters</td>
<td>Variable parameters</td>
</tr>
<tr>
<td>FEL</td>
<td>$T_{\text{max}} = 5000$</td>
<td>$\eta = 0.04 : 0.02 : 0.4$</td>
</tr>
<tr>
<td>DSL</td>
<td>$T_{\text{max}} = 10000$</td>
<td>$N_{\text{FM}} = 10 : 10 : 150$</td>
</tr>
<tr>
<td></td>
<td>$\eta_{\text{FM}} = 0.025$</td>
<td>$\eta = 0.05 : 0.01 : 0.2$</td>
</tr>
<tr>
<td>DIM-MLP</td>
<td>$T_{\text{max}} = 10000$</td>
<td>$N_{\text{CON}} = 10 : 10 : 150$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.01$</td>
<td>$\eta = 0.01$</td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>$T_{\text{max}} = 100000$</td>
<td>$N_{\text{CON}} = 20 : 10 : 150$</td>
</tr>
<tr>
<td></td>
<td>$N = 1 : 1 : 4$</td>
<td>$m = 4 : 1 : 16$</td>
</tr>
<tr>
<td>SLbA</td>
<td>$T_{\text{max}} = 100$</td>
<td>$\sigma_0 = 0.2 : 0.2 : 3.0$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.01$</td>
<td>$\eta = 0.01$</td>
</tr>
</tbody>
</table>

Table 13 — Parameter values for the saccade learning tasks.

### 14. Learning tasks with the 2-link planar arm

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>with quality function $Q_0$</th>
<th>with quality function $Q_0$ and sensory noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fixed parameters</td>
<td>Variable parameters</td>
</tr>
<tr>
<td>FEL-J$^6$, FEL-J$^7$</td>
<td>$T_{\text{max}} = 2000000$</td>
<td>$\eta = 0.025 : 0.0125 : 0.125$</td>
</tr>
<tr>
<td>DSL</td>
<td>$T_{\text{max}} = 2000000$</td>
<td>$N_{\text{FM}} = 3000 : 3000 : 21000$</td>
</tr>
<tr>
<td></td>
<td>$\eta_{\text{FM}} = 0.005$</td>
<td>$\eta = 0.04 : 0.02 : 0.2$</td>
</tr>
<tr>
<td>DIM-MLP</td>
<td>$T_{\text{max}} = 100000$</td>
<td>$N_{\text{CON}} = 25 : 25 : 300$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.05$</td>
<td>$\eta = 0.05$</td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>$T_{\text{max}} = 100000$</td>
<td>$N_{\text{CON}} = 25 : 25 : 300$</td>
</tr>
<tr>
<td></td>
<td>$N = 7 : 3 : 22$</td>
<td>$m = 1 : 1 : 4$</td>
</tr>
<tr>
<td>SLbA$^7$</td>
<td>$T_{\text{max}} = 50$</td>
<td>$\sigma_0 = 0.3 : 0.2 : 1.9$</td>
</tr>
<tr>
<td></td>
<td>$\lambda_{\text{SLbA}} = 0.005$</td>
<td>$\lambda = 1.0$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.05$</td>
<td>$\eta = 0.05$</td>
</tr>
</tbody>
</table>

Table 14 — Parameter values for the learning tasks with the 2-link planar arm.

---

$^6$ In all tables, variable parameters are given in MATLAB vector notation: start value : step size : final value.

$^7$ The LE and EP values for SLbA define the strategies for increasing the number of learning examples (LE) and training epochs (EP) in each stage; they are denoted in the format LE: a-b-c / EP: a-b-c with a being the start value, b the increase from stage to stage, and c the maximum value.
<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3-link arm</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>with quality function $Q_1$</td>
<td></td>
<td>with quality function $Q_1$ and sensory noise</td>
<td></td>
</tr>
<tr>
<td><strong>Table 15</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>$T_{\text{MAX}} = 100000$</td>
<td>$N_{\text{CON}} = 500 : 250 : 1500$</td>
<td>$N_{\text{CON}} = 5000 : 2500 : 1250$</td>
<td>$N_{\text{CON}} = 5000 : 2500 : 1250$</td>
</tr>
<tr>
<td></td>
<td>$N = 40 : 20 : 100$</td>
<td>$m = 2 : 1 : 5$</td>
<td>$N = 40 : 20 : 100$</td>
<td>$m = 2 : 1 : 5$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
</tr>
<tr>
<td><strong>Table 16</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>$T_{\text{MAX}} = 300000$</td>
<td>$N_{\text{CON}} = 2500 : 5000 : 17500$</td>
<td>$N_{\text{CON}} = 2500 : 5000 : 17500$</td>
<td>$N_{\text{CON}} = 2500 : 5000 : 17500$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
</tr>
</tbody>
</table>

**Table 16 — Parameter values for the learning tasks with the 4-link planar arm.**

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4-link arm</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>with quality function $Q_1$</td>
<td></td>
<td>with quality function $Q_1$ and sensory noise</td>
<td></td>
</tr>
<tr>
<td><strong>Table 15</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>$T_{\text{MAX}} = 300000$</td>
<td>$N_{\text{CON}} = 30000 : 150000$</td>
<td>$N_{\text{CON}} = 30000 : 150000$</td>
<td>$N_{\text{CON}} = 30000 : 150000$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
</tr>
<tr>
<td><strong>Table 16</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIM-NGPCA</td>
<td>$T_{\text{MAX}} = 300000$</td>
<td>$N_{\text{CON}} = 30000 : 150000$</td>
<td>$N_{\text{CON}} = 30000 : 150000$</td>
<td>$N_{\text{CON}} = 30000 : 150000$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 15</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL</td>
<td>$T_{\text{MAX}} = 400000$</td>
<td>$N_{\text{FM}} = 5000 : 10000 : 55000$</td>
<td>$N_{\text{FM}} = 5000 : 10000 : 55000$</td>
<td>$N_{\text{FM}} = 5000 : 10000 : 55000$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning strategy</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
<th>Fixed parameters</th>
<th>Variable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 16</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL</td>
<td>$T_{\text{MAX}} = 400000$</td>
<td>$N_{\text{FM}} = 5000 : 10000 : 55000$</td>
<td>$N_{\text{FM}} = 5000 : 10000 : 55000$</td>
<td>$N_{\text{FM}} = 5000 : 10000 : 55000$</td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
<td>$\lambda_S = 1.0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
<td>$\eta = 0.005$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
<td>$\sigma = 1$</td>
<td></td>
</tr>
</tbody>
</table>

Table 15 — Parameter values for the learning tasks with the 3-link planar arm.

Table 16 — Parameter values for the learning tasks with the 4-link planar arm.
References


