Psychological Review
2007, Vol. 114, No. 4, 1015–1046

0033-295X/07/$12.00 DOI: 10.1037/0033-295X.114.4.1015

Exploiting Redundancy for Flexible Behavior: Unsupervised Learning in a Modular Sensorimotor Control Architecture

Martin V. Butz, Oliver Herbort, and Joachim Hoffmann
University of Würzburg

Autonomously developing organisms face several challenges when learning reaching movements. First, motor control is learned unsupervised or self-supervised. Second, knowledge of sensorimotor contingencies is acquired in contexts in which action consequences unfold in time. Third, motor redundancies must be resolved. To solve all 3 of these problems, the authors propose a sensorimotor, unsupervised, redundancy-resolving control architecture (SURE_REACH), based on the ideomotor principle. Given a 3-degrees-of-freedom arm in a 2-dimensional environment, SURE_REACH encodes 2 spatial arm representations with neural population codes: a hand end-point coordinate space and an angular arm posture space. A posture memory solves the inverse kinematics problem by associating hand end-point neurons with neurons in posture space. An inverse sensorimotor model associates posture neurons with each other action-dependently. Together, population encoding, redundant posture memory, and the inverse sensorimotor model enable SURE_REACH to learn and represent sensorimotor grounded distance measures and to use dynamic programming to reach goals efficiently. The architecture not only solves the redundancy problem but also increases goal reaching flexibility, accounting for additional task constraints or realizing obstacle avoidance. While the spatial population codes resemble neurophysiological structures, the simulations confirm the flexibility and plausibility of the model by mimicking previously published data in arm-reaching tasks.

Keywords: unsupervised motor learning, sensorimotor control, population encoding, redundancy resolution, computational model

Controlling one's body is the first prerequisite to successful interaction with the environment. This ability, however, is mostly not innate. Even seemingly simple goal-directed reaching movements are learned during infancy (Konczak, Borutta, & Dichgans, 1997; Konczak & Dichgans, 1997; von Hofsten, 2003). More complex behavior—like walking, skiing, or playing an instrument—requires significantly more, and often continuous, learning and training. To enable the acquisition as well as the flexible and continuous adaptation of behavior to changing environmental and bodily constraints, highly modular hierarchical control architectures appear necessary (Poggio & Bizzi, 2004).

Recent findings suggest that goals in goal-directed behavior are encoded in terms of desired perceptual states (Elsner & Hommel, 2001; Hoffmann, 1993; Hoffmann, Stöcker, & Kunde, 2004; Kunde, Koch, & Hoffmann, 2004; Prinz, 1997). To trigger goal-directed behavior, these desired perceptual states have to be linked to efferent signals, or actions, whose application is expected to result in perceiving what is desired. These perception–action links can be encoded only self-supervised from interactions with the environment.

One way to encode such links is to learn sensorimotor contingencies, that is, action-dependent sensory correlations. Such contingencies were recently acknowledged to be highly important in cognitive systems for the realization of forward sensory emulation and inverse goal-directed motor control (for reviews, see Grush, 2004; O'Regan & Noë, 2001). We encode sensorimotor contingencies as action-dependent situation-effect links.

The contingencies are learned and continuously adapted. To enable goal-directed behavior, the sensorimotor knowledge is integrated into inverse models, which evoke suitable actions to reach desired goal states (e.g., goal postures or locations) dependent on the current state of the body (e.g., current arm posture; Flash & Sejnowski, 2001; Grush, 2004; Kawato, 1999; Wolpert & Ghahramani, 2000).

Three Challenges for a Self-Developing, Adaptive Controller

We are interested in learning a flexible, adaptive, goal-directed motor control system from simple interactions of body and environment. This is by no means trivial: (a) Learning needs to be unsupervised, or self-supervised, because no teaching signals are available; (b) action sequences have to be triggered to reach distant goals; (c) redundancies have to be resolved for action execution because desired effects can usually be produced by multiple, often infinite, action possibilities.

Martin V. Butz, Oliver Herbort, and Joachim Hoffmann, Department of Psychology, University of Würzburg, Röntgenring, Würzburg, Germany.

This work was supported by the European Commission within the MindRACES project, Contract FP6-511931. We thank our colleagues at the Department of Cognitive Psychology III at the University of Würzburg for the support and fruitful discussions as well as Marjorie Kinney for her proofreading effort.

Correspondence concerning this article should be addressed to Martin V. Butz, Department of Cognitive Psychology III, University of Würzburg, Röntgenring 11, 97070 Würzburg, Germany. E-mail: butz@psychologie.uni-wuerzburg.de

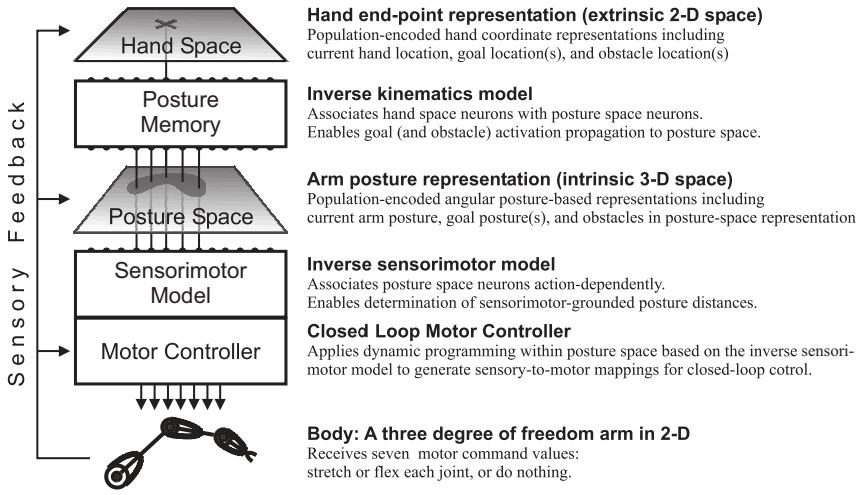

Figure 1. The SURE_REACH implementation in this article consists of five basic modules: two population-encoded spatial representations of hand end-point location and arm posture: a posture memory (an inverse kinematics model) that associated hand end-point locations with (redundant) corresponding arm postures; a motor controller that contains an inverse sensorimotor model, which associates postures action-dependently; a body, which is a three-degrees-of-freedom arm in this article.

This article addresses these three challenges proposing a *sensorimotor, unsupervised, redundancy-resolving control architecture* (*SURE_REACH*). SURE_REACH is implemented as a neural network model and is tested on the control of a three-degrees-of-freedom arm in a two-dimensional (2-D) environment.[1] Figure 1 shows the basic components of the SURE_REACH architecture and notes the representations and capabilities of each component with respect to the investigated arm control task. During movement production, a desired hand location in external hand space is mapped onto a set of suitable arm postures by a posture memory. The redundant set of suitable arm postures is diffused in the population-encoded posture space by means of dynamic programming based on a learned inverse sensorimotor model. The result is a *sensory-to-motor mapping*, which encodes the closest paths to the desired goal postures from any starting posture. The mapping is similar to field-based trajectory representations that encode whole families of solutions in a holographic memory (Morasso, Sanguineti, & Spada, 1997). The mapping is used by a closed-loop motor controller to generate the motor commands that move the arm toward the most suitable arm posture and hence the hand toward the desired hand location. Both the posture memory and inverse sensorimotor model encode redundant solutions. Whereas the posture memory expands a single hand position to all associated arm postures, the inverse sensorimotor model encodes redundant motor commands to reach goal postures. In the next paragraphs, we further specify how SURE_REACH solves the three challenges listed above.

The Unsupervised, or Self-Supervised, Learning Challenge

A powerful principle that proposes a solution to the challenge of unsupervised, or self-supervised, learning during development is the *ideomotor principle,* which has been proposed in psychology for over a century (Herbart, 1825; James, 1890; cf. Hoffmann et al., 2004). The ideomotor principle suggests that an organism learns an inverse model for action selection and control based on the sensorimotor contingencies it perceives during initially random *motor babbling.* SURE_REACH adheres to this principle. It learns unsupervised both (a) an inverse kinematics model, which maps hand positions in extrinsic hand space to (redundant) arm postures in intrinsic posture space, and (b) an inverse sensorimotor model, which associates contiguous arm postures with each other action-dependently. The inverse kinematics model is stored in the posture memory. The inverse sensorimotor model enables the motor controller to determine sensorimotor-grounded distances to goals. It is used to accomplish efficient goal-directed motor control. The approach is generally similar to early self-supervised control approaches (Kuperstein, 1988; Mel, 1991) but extends them to control redundant bodies more flexibly and efficiently.

The Action Sequence Generation Challenge

Desired goal states can often be reached only after the execution of extended action sequences. To control highly complex plants, in which the consequences of actions unfold in time, complex action sequences may need to be executed to reach a particular destination. SURE_REACH proposes a solution to this problem by learning an inverse sensorimotor model, which is embedded in posture space. Action trajectories are generated dynamically during movement preparation by means of dynamic programming, which propagates activity by means of the inverse sensorimotor model within posture space. The result is a sensory-to-motor mapping, which enables the fast and flexible closed-loop control of complex action sequences to distant goal states.

[1] The source code, an executable demo program, and a program manual are available at http://www.psychologie.uni-wuerzburg.de/i3pages/SURE_REACH/

The Redundancy Resolution Challenge

The final problem addressed is the resolution of redundancies (Bernstein, 1967). Almost any goal can be reached in multiple ways, but only one way can be applied at any given time. For example, when reaching for a glass, the left or right hand may be used, the final arm posture can vary in an infinite number of ways (because of the redundant degrees of freedom of the arm), and one movement trajectory must be selected from infinite possibilities. Figure 2 illustrates two types of redundancy problems: In Figure 2A, multiple goal postures realize a single hand location; in Figures 2B and 2C, multiple trajectories are possible to reach the same goal posture. Thus, for even relatively simple tasks, an abundance of possible means are available.

Despite this abundance, human arm movements are usually rather stereotypic, suggesting that many constraints bias the action selection mechanism (Flash & Hogan, 1985). These additional constraints, which reduce redundancy, are called *optimality principles.* They enable the selection of an optimal solution among all possible solutions on the basis of constraints such as movement smoothness (Flash & Hogan, 1985) or end-point accuracy and comfort (Harris & Wolpert, 1998; Weigelt, Kunde, & Prinz, 2006; for reviews on optimality principles, see Engelbrecht, 2001; Todorov, 2004).

To resolve these redundancies, the motor controller of SURE_REACH applies dynamic programming to propagate goal-originating activity within the population-encoded posture space on the basis of its inverse sensorimotor model. The result is a sensory-to-motor mapping in posture space that is used to execute motor commands by means of closed-loop control. Goal distances are consequently implicitly encoded in the generated sensory-to-motor mapping, which is grounded on the experienced sensorimotor contingencies. The mapping implicitly encodes movement trajectories that directly lead to the goal posture that is closest to the actual posture according to the inverse sensorimotor model.

The redundancy resolution mechanism in SURE_REACH builds on the *posture-based motion planning theory* (PB theory), which also addresses the problem of redundancy resolution and trajectory generation (Rosenbaum, Engelbrecht, Bushe, & Loukopoulos, 1993; Rosenbaum, Loukopoulos, Meulenbroek, Vaughan, & Engelbrecht, 1995; Rosenbaum, Meulenbroek, Vaughan, & Jansen, 2001). The ability to apply different optimality criteria from one movement to the next has enabled the PB theory to account for a wide range of empirical findings. SURE_REACH additionally grounds the required optimality criteria on the experienced self-generated sensorimotor contingencies. Further differences and similarities between both models are scrutinized in the Discussion section.

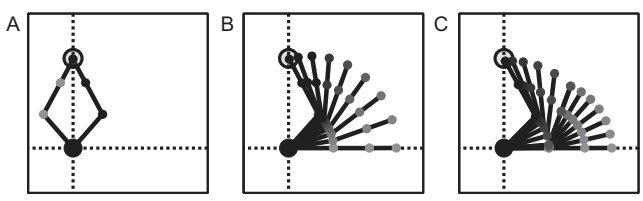

Figure 2. Motor redundancy has to be resolved on different levels, like end postures (A) or trajectories (B and C).

Overview

The remainder of this work is structured as follows: First, we review other computational models of motor learning and control, focusing on their capabilities of learning sensorimotor contingencies and resolving redundancies. Second, the proposed SURE_REACH neural network model is introduced and evaluated, confirming efficient and flexible initialization and control of reaching movements. Moreover, we show how the architecture can flexibly adapt behavior to additional task constraints. Comparisons with various behavioral data from the literature confirm the plausibility of the model. The article ends with a discussion on the novelty of the approach, the relation to neuroscience, future challenges for SURE_REACH, and final conclusions.

Current Computational Models

In this section, we investigate various available computational models of motor learning and control that focus on unsupervised or self-supervised learning. Models that explicitly investigate supervised learning are not discussed (e.g., Ito, Noda, Hoshino, & Tani, 2006; Stringer, Rolls, Trappenberg, & de Araujo, 2003). For each investigated model, we provide a short introduction and then assess the model's contribution to the solution of the three problems discussed above. Focusing on the redundancy problem, we divide the computational models into two groups. In the first group, redundancy is resolved before or during learning by adding additional optimality constraints to the learning task. In the second group, redundant solutions are encoded and resolved only immediately before movement production. SURE_REACH belongs to the latter group of models.

When redundancy is resolved before or during learning (see Figure 3, top), the inverse model does not store all possible solutions for an inverse problem, but rather it stores one single solution, which optimizes a given optimality criterion. This makes action selection straightforward. For any given start and goal, the inverse model learns exactly one solution so that no redundancy needs to be resolved when acting goal-directedly. The drawback of this approach is that it lacks flexibility. If a behavior that was previously optimal is now suboptimal or impossible because of, for example, injury, the appearance of obstacles, or other task-related constraints, then the organism is unable to flexibly switch to an alternative, previously suboptimal behavior.

When redundancy is resolved only before movement production, the architecture has to store many redundant behaviors and then select one of the available ones to pursue its current goal (see Figure 3, bottom). Thus, the inverse model may provide multiple or even an infinite variety of possible solutions. The approach is computationally more demanding because the inverse model needs to store multiple solutions for a problem and because redundancy has to be resolved on the fly during behavior selection and control. The advantage, however, is a system that is able to quickly react to novel or changing task-dependent optimality criteria.

We now compare several relevant models available in the literature, focusing on their redundancy-resolving approaches, on their capability of triggering complex motor sequences to reach distant goals, and on their learning approaches.

Resolving Redundancy Before or During Learning

Models of motor learning that resolve redundancy before or during learning usually use error-based learning mechanisms that strive to learn the optimal solution to the inverse problem. Three common goal-directed approaches to motor learning are *direct inverse modeling* (DIM; Jordan & Rumelhart, 1992), *feedback error learning* (FEL; Kawato, Furukawa, & Suzuki, 1987), and *resolved motion rate control* (RMRC; Jordan & Rumelhart, 1992; Whitney, 1969). Additionally, direct reinforcement learning methods have shown some success in this context (Berthier, 1996; Berthier, Rosenstein, & Barto, 2005).

DIM Approaches

DIM has been applied to model motor learning within a range of computational models (Baraduc, Guigon, & Burnod, 1999, 2001; Bullock, Grossberg, & Guenther, 1993; Kuperstein, 1988, 1991; Ognibene, Rega, & Baldassarre, 2006). DIM learns an inverse model by observing action effects. Regardless of what the current movement goal is or if such a goal exists, given a current sensory effect, DIM computes the action that may have caused the sensation. If a difference between computed and actually executed action is detected, DIM adjusts its inverse model accordingly. As soon as the inverse model is sufficiently accurate, it can be used to issue motor commands for desired goal states.

A big drawback of DIM is that it is not guaranteed to converge if the controlled plant is redundant. In fact, it is destined to fail if the set of goal states is *nonconvex* (Jordan & Rumelhart, 1992).[2] Consider the task of determining a joint configuration that locates the hand of a two-joint planar arm at a desired position. During learning, each hand location may be realized by different joint angle configurations, for example, one with the elbow oriented clockwise and one with the elbow oriented counterclockwise. Hence, no consistent bias toward one of the joint configurations exists, and the inverse model is likely to evoke a mixture of both postures, consequently missing the target (cf. Figure 4).

Some DIM approaches use arm models with only two joints and a limited set of possible joint angles, avoiding the necessity of handling redundancy (e.g., Baraduc et al., 1999, 2001; Ognibene et al., 2006). Others are able to control plants with redundant degrees of freedom by imposing additional constraints during learning that eliminate the redundancy (Bullock et al., 1993).

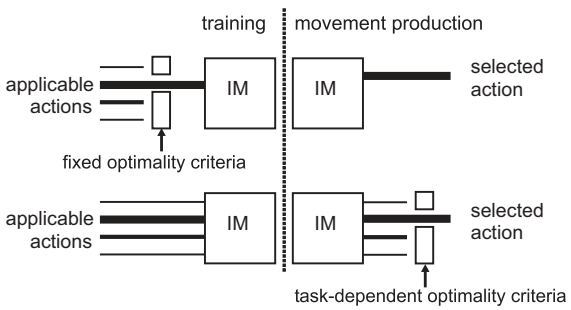

Figure 3. Motor redundancy may be resolved before the inverse model is learned (top) or afterward during movement production (bottom).

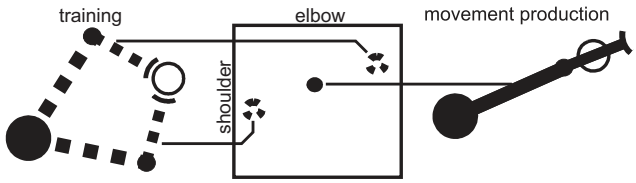

Figure 4. If the goal set is nonconvex (e.g., two separate points in joint angle space), direct inverse modeling might store a nonsolution for the inverse problem (solid circle inside box).

Additionally, DIM is only applicable in kinematic learning contexts that yield a clear temporal relationship between action and effect. For example, DIM can be applied if movements in the same direction require the same motor commands, independently of the desired movement amplitude (Bullock et al., 1993). However, DIM can only be applied with additional enhancements to problems in which a sequence of different motor commands needs to be issued to reach a goal effectively.

With respect to learning, DIM is a self-supervised learning approach. No training signals need to be provided from the outside. DIM implementations learn according to the ideomotor principle, initially executing random actions and learning from the observation of the consequently observed sensorimotor contingencies.

To summarize, albeit DIM is a self-supervised learning approach, complex body interactions that unfold in time can be learned only with additional system enhancements. Most important, redundancy is resolved before learning, for example, by constraining the learning experience or the plant characteristics.

FEL Approaches

Another error-based learning scheme for training inverse models is FEL (Kawato et al., 1987). In this case, the inverse model is learned dependent on a preexisting feedback controller. Initially, the inverse model does not contribute to the formation of useful actions. When a difference between desired and perceived states arises, the "innate" feedback controller triggers suitable actions for adjustment. These adjustments are used as training signals for the inverse model, shifting the output of the inverse model in the direction of the adjustment signal of the feedback controller. This levels out deficiencies of the feedback controller. Eventually, the inverse model is able to control the plant on its own.

This principle was mainly applied to model cerebellar motor learning (Barto, Fagg, Sitkoff, & Houk, 1999; Berthier, Singh, Barto, & Houk, 1992, 1993; Haruno, Wolpert, & Kawato, 2001; Kawato et al., 1987; Kawato & Gomi, 1992; Schweighofer, Arbib, & Kawato, 1998; Schweighofer, Spoelstra, Arbib, & Kawato, 1998; Wolpert & Kawato, 1998) but also to motor learning in general (Karniel & Inbar, 1997). FEL models analyze the role of the cerebellum as a side loop of the motor system, which is trained by corrective cerebral motor commands, and which is responsible for smoothing movements and enhancing control efficiency.

[2] A set of states is nonconvex if there exists at least one state not in the set that lies between two states that are in the set.

Despite their compelling hierarchical control structures and progressively smoother and more efficient behavior control, FEL approaches contribute little to the question of how motor redundancy might be resolved. The redundancy-resolving problem persists as in the DIM approaches above. If different actions are applicable to reach a goal, and the feedback controller randomly chooses between those actions, then the learning mechanism is destined to fail if the set of alternatives is nonconvex (Jordan & Rumelhart, 1992). The feedback controller has to be consistent in its action choice to ensure successful learning. Hence, it is the preexisting feedback controller that implicitly solves the redundancy problem in FEL.

With respect to the unsupervised learning challenge, FEL somewhat sidesteps this problem by assuming the existence of a simple closed-loop controller, which exerts corrective motor commands that are based on position error. The controller is used to transform errors in task space to errors in action space (Kawato, 1990), providing the missing teaching signal. Hence, FEL is not interested in learning a basic controller, but rather in improving the performance of the controller by higher level control structures.

FEL architectures mainly tackle the problem of controlling highly complex plants, whose bodily interactions unfold in time. During learning, goals are explicitly represented over longer periods of time, enabling the learning of far-reaching sensorimotor contingencies. The approach stresses the formation of associations between potential goals and actions that have to be carried out for a considerable duration before the goal is actually reached. As a consequence, hierarchical control structures develop that improve behavior efficiency and smoothness (Haruno et al., 2001; Kawato et al., 1987; Wolpert & Kawato, 1998).

To summarize, FEL does not explicitly tackle the redundancy problem. Rather, FEL resolves redundancy by means of a preexisting feedback controller. In this sense, FEL addresses the problem of acquiring a sophisticated inverse model based on a basic feedback controller. Thus, FEL significantly refines a simple control strategy in order to be able to execute complex behavioral control strategies that efficiently unfold in time.

RMRC

RMRC (Whitney, 1969) learns the forward kinematics of a plant by determining the Jacobian of the plant.[3] The learned Jacobian in an arm usually relates displacements in joint angle space to displacements in end-effector space (Craig, 2005). In effect, the Jacobian can be used to determine end-effector position changes given joint angle changes.

To solve the inverse kinematics, the inverse of the Jacobian must be determined or approximated. In redundant control problems, however, RMRC also faces the one-to-many mapping problem and needs to resolve redundancy to be able to generate a suitable inverse of the Jacobian. For example, additional optimality criteria may be imposed, such as a preferred basic body posture (D'Souza, Vijayakumar, & Schaal, 2001).

Another instantiation of RMRC is the distal supervised learning approach (Jordan & Rumelhart, 1992), which was introduced to solve the nonconvexity problem in DIM. In this case, first a forward model is trained to predict action consequences. Then an inverse model is learned, which may be initialized by an inverse model learned by DIM. The inverse model is learned by back propagating the error in end-point space through the forward model. This gives an error signal for the motor controller and allows the learning of a unique inverse model. The necessary constraints to resolve redundancy are implicit in the error backpropagation approach coupled with the sampled training set.

As in FEL and DIM, though, distal supervised learning and RMRC in general learn one solution to a problem, forgetting about all alternative solutions. In distal supervised learning, this solution is the one that generates the shortest paths averaged over the training set data. In RMRC, in general, other constraints can be imposed to enforce optimal paths. Thus, RMRC cannot flexibly account for novel task constraints, previously unseen goal combinations, or obstacle avoidance on the fly (D'Souza et al., 2001).

Direct Reinforcement Learning (RL) Approaches

Unlike the above error-based learning schemes, direct RL approaches learn behavioral policies solely on the basis of reward-based feedback.[4] The reward is usually back propagated, approximating a value function that indicates the value of a state or state–action pair for the task at hand. Usually, a particular form of the Bellman (1957) equation for optimal control is learned. Dependent on the representation, the policy can be either directly deduced from the value function or improved progressively respecting the value function, which is done, for example, in Q-learning or in actor–critic methods, respectively (Sutton & Barto, 1998). In previous arm control applications of direct RL, either the action that yielded the highest reward was learned with actor–critic methods (Berthier et al., 2005; Kositsky & Barto, 2002), or a state–action value function was learned with Q-learning (Berthier, 1996).

In these direct RL approaches, redundancy is also resolved before learning by the imposition of a particular reward scheme. Direct RL approaches learn the optimal policy for one particular reward scheme, such as a particular goal location in a reaching task. Thus, depending on the chosen reward scheme and the representation used, direct RL is more or less inflexible in accounting for novel task constraints or goal constellations.

The compelling advantage of RL is that it does not require the existence of a training signal, such as provided by the feedback controller in FEL, which explicitly encodes the correct action for a given goal. Reward propagation alone suffices to develop an optimal behavioral policy for a particular task at hand. Because the learning approach is consequently more general than FEL, DIM, and RMRC, learning usually takes considerably longer.

An additional positive feature of RL approaches is that they are able to generate complex action sequences that unfold in time.

---

[3] The Jacobian is a matrix of all first-order partial derivatives.

[4] *Direct RL* approaches are those RL approaches that learn optimal behavioral policies directly without the use of a model of their environment. Therefore, direct RL is often also referred to as *model-free RL*. In contrast to direct RL, *indirect RL* is model-based. That is, indirect RL uses a potentially learned model of the environment to learn or generate its current behavioral policy. Dynamic programming may be considered the most fundamental form of model-based RL (Sutton & Barto, 1998). Many hybrid systems exist that combine indirect RL with direct RL techniques, as exemplified in the Dyna architecture (Sutton, 1990) or in real-time dynamic programming (Barto et al., 1995).

Because of the reward back propagation, RL can acquire optimal policies in plants with delayed action effects. Thus, tasks can be solved that require the execution of long and complex action sequences.

To summarize, direct RL methods provide an interesting alternative to the approaches mentioned above. Unlike DIM, they are also applicable in redundant plants and for learning complex action sequences. Unlike FEL, they do not require a preexisting feedback controller that provides consistent learning signals. Unlike RMRC, no forward model is necessary to be able to learn a behavioral policy. The largest drawback of direct RL approaches, however, is their behavioral inflexibility: Direct RL approaches learn one particular policy that optimally solves one particular task (with the encountered reward scheme). If the task changes, such as reward signals, discount factors, plant properties, or desired goal constellations, then the optimal policy changes and direct RL needs to relearn (often from scratch). Relative goal representations can partially alleviate this problem, but they only work well in plants in which the control strategy is sufficiently independent of the current situation or in which the input space is small enough to be able to account for situation dependencies. Alternatively, different goals may be represented explicitly, which, however, significantly decreases learning speed. Thus, although direct RL approaches are very elegant behavioral policy learners, they are not very well suited to flexibly adjust their behavioral policies to novel task constraints or reward distributions.

Short Summary

This section shows how different learning schemes solve the redundancy problem before or during learning. DIM requires a nonredundant plant with a one-to-one mapping between actions and effects, or it requires the imposition of additional mechanisms that resolve redundancy before learning. FEL requires the preexistence of a feedback controller that is used to provide the required teaching signals for supervised learning. RMRC uses a forward model to convert error signals from sensory to motor encodings, training the inverse model controller in this way. Finally, direct RL methods rely on appropriate reward signals and suitable representations. They are able to learn control policies in more complex environments, but they are rather inflexible because their behavioral policies cannot be easily adjusted to novel goal constellations or task constraints.

In all of the discussed learning schemes, only one solution is stored for a particular control problem. Although this solves the redundancy problem, the approaches lack the flexibility to quickly adapt to novel task constraints that may change the available set of actions, action preferences, and spatial or other constraints. For example, if the usually chosen action is not available any longer because of, for example, obstacles or a broken limb, then the discussed approaches do not offer alternative behaviors. Similarly, if a usually chosen joint movement or joint position is suddenly highly uncomfortable because of injury, the models are not able to rapidly compensate.

Instead of storing only a single control policy solution, it can be advantageous to represent all possible solutions in a control structure and delay action choice until it is really necessary. If obstacles or new optimality criteria reduce the appropriateness of formerly suitable actions, then alternatives would be immediately available. In the following section, we review computational models that store multiple solutions for each possible task and resolve redundancy only during action preparation and execution.

Storing Redundancy

The second approach to modeling motor learning is to compactly store many, or even all, possible solutions for an inverse kinematics problem, such as all redundant arm postures that coincide with a hand end-point location. The currently most suitable posture may then be immediately selected before action execution. Similarly, a matching trajectory needs to be chosen that resolves the redundant possibilities in movement trajectory. These selection processes should be guided by task-dependent constraints.

Storing redundancy has several appealing features. First, no teaching or reward signals are necessary, which enable the model to store one specific solution for the inverse kinematics problem. Essentially, unsupervised learning is possible. Second, because all solutions for a problem are stored, the organism is still able to act goal-directedly if the preferred solution is not applicable. If optimality criteria or reward values change, a complete remapping of the learned model is not necessary, rather only constraints need to be adjusted. Third, current findings on biological motor systems show that biological control systems encode and exploit motor redundancy to improve control flexibility and efficiency, following a minimal intervention principle (Latash, Scholz, & Schöner, 2002; Todorov & Jordan, 2002).

In the following sections, we review two rather different approaches, the *mean of multiple computations* (MMC) network (Cruse & Steinkühler, 1993) and the PB theory (Rosenbaum et al., 1993). Both approaches resolve redundancies for inverse kinematics problems.

The MMC Network

The MMC architecture models the complete kinematics of the plant in question (Cruse & Steinkühler, 1993; Cruse, Steinkühler, & Burkamp, 1998). It solves the inverse kinematics problem (as well as the forward kinematics problem or any mixed problem) for arbitrary combinations of desired output values by a very compact neural network representation.

The neural network dynamics inherent in MMC resolve redundancy iteratively. By fixing some of the network inputs to currently desired values, the network dynamics cause the network to shift into one of the states that represent a solution to the posed inverse problem. Thus, MMC implicitly resolves redundancy by means of its inherent system dynamics.

Although the structure allows the imposition of novel constraints on the controlled arm, such as a desired arm angle or a desired position, MMC does not necessarily provide the shortest path, or the least costly path, to a goal state. Rather, the shortest path is determined by the internal dynamics of the model. Consequently, the exhibited dynamics in MMC are not grounded in the environment or the body so that the extent to which the inherent system dynamics reflect body constraints depends on the designer. Currently, no learning mechanism is available that is based on the interaction of the MMC and the associated body.

The PB Theory

On the action selection side, the PB theory (Rosenbaum et al., 1993, 1995, 2001) is the model closest to SURE_REACH. It solves the inverse kinematics problem of selecting an appropriate set of joint angles (i.e., a posture) to move the hand to a desired location. In contrast to the approaches that resolve redundancy before learning, the model stores a set of encountered arm postures during learning. Once a goal is triggered, all postures are evaluated with a weighting scheme (Rosenbaum et al., 1995) or a constraint hierarchy (Rosenbaum et al., 2001), which selects the goal posture on the basis of the *elimination by aspects principle* (Tversky, 1972).

Constraints, such as collision avoidance, movement accuracy, or the minimization of travel costs, are ranked hierarchically according to the requirements of the current task. The posture that best satisfies the constraint hierarchy is chosen as the movement end posture. The chosen posture then triggers the execution of a controller, which moves the hand toward the chosen posture by obeying the invoked constraints. For example, if the task requires a maximally accurate goal position, the posture with the hand closest to the target is selected, where the Euclidean distance is then used to measure goal distance. On the other hand, if movement velocity was critical, a posture may be chosen that may be less accurate but that may specify a less costly angular transition. Absolute angular displacement in conjunction with optimal transition times are used to determine travel costs (Rosenbaum et al., 1995). Additional constraints may be added, for example, to prefer postures with specific joint angles (Rosenbaum et al., 2001). Recently, the PB theory was also enhanced to 3-D workspaces by using a quaternion-based representation to account for (single-axis) rotations (and their noncommutativity) and the geodesic to determine the shortest path between two postures (Vaughan, Rosenbaum, & Meulenbroek, 2006).

To accomplish obstacle avoidance, the full specifications of arm lengths and angles are used to execute internal forward simulations to check for obstacle collisions (Rosenbaum et al., 2001). If an obstacle collision is detected, the trajectory is altered by superimposing a back-and-forth movement on the movement to the goal. Once a suitable superimposed movement is found, the combined movement is executed, effectively deflecting the arm around the obstacle.

The PB theory can also generate realistic, bell-shaped velocity curves, imposing sinusoidal movement speeds on the action execution scheme of goal approaching and obstacle avoidance (Meulenbroek, Rosenbaum, Jansen, Vaughan, & Vogt, 2001; Rosenbaum et al., 1995, 2001). Obstacle avoidance was also successfully applied to the challenge of grasping objects (Meulenbroek et al., 2001) integrated in the arm approaching movement. These challenges are not directly addressed in this article but are kept for future work on the proposed SURE_REACH architecture.

The PB theory emphasizes the resolution of redundancy but not the motor learning problem. No properties of the stored postures, which are relevant for the evaluation process, such as the coinciding hand and arm location in space or the transition times from one posture to another, are learned. What is learned is a set of postures, which serve as starting points for the search of a posture that fulfills the current constraints and optimizes current optimality criteria.

Short Summary

Both of the models discussed in the last section can account for the resolution of motor redundancies. The MMC allows the invocation of posture and end-point constraints, which are resolved by the inherent network dynamics. Currently, however, MMC cannot be trained unsupervised. The PB theory stores a set of experienced postures and then chooses the currently optimal goal posture on the basis of weighing or prioritizing constraints. It learns a set of possible goal postures but not the motor controller or the distance measures used to resolve redundancies. Besides the incorporation of additional task constraints, it has been shown that the PB theory can avoid obstacles by internal trajectory simulation and superimposed back-and-forth movements (Meulenbroek et al., 2001; Rosenbaum et al., 2001). The SURE_REACH architecture introduced in the next section builds on the high flexibility gained by the PB theory.

SURE_REACH

The survey of related approaches in the previous section shows that learning architectures resolve motor redundancy either (a) before or during learning or (b) after learning during action preparation and execution. SURE_REACH belongs to the latter class. Compared with the MMC model and the PB theory, which also resolve redundancy after learning, SURE_REACH adds a neural-based, unsupervised learning architecture that grounds distance measures in experienced sensorimotor contingencies. The architecture is implemented as a biologically plausible neural network model. It extends our previous computational models, which accounted for nonredundant single (Herbort, Butz, & Hoffmann, 2005a, 2005b) or multijoint (Herbort, 2005) arm movements.

SURE_REACH Overview

SURE_REACH is hierarchically structured (see Figure 1 and Figure 5). The knowledge of SURE_REACH about its body and environment consists of two population-encoded spatial body representations and two associative structures. An extrinsic hand space encodes hand locations (x–y coordinates) with a uniformly distributed, partially overlapping 2-D array of neurons. An intrinsic posture space similarly encodes arm postures with a uniformly distributed, partially overlapping 3-D array of neurons (shoulder, elbow, and wrist angles). A posture memory associates hand with posture space neurons, encoding an inverse kinematics model. A sensorimotor model associates postures action-dependently with each other. It essentially encodes a posture-based body model that is able to predict which posture is reached given a current posture and chosen motor command. More important for the purpose of this study, though, is its inverse capability: The model is also able to deduce the posture that preceded a given posture given some action was executed. We refer to this aspect of the model as the *inverse sensorimotor model.*

Posture memory and the sensorimotor model are acquired in an initial motor babbling phase, during which random movements are executed. During this phase, the posture memory acquires its kinematic mapping, and the sensorimotor model encodes the experienced sensorimotor contingencies. We should emphasize that infant motor behavior is certainly not based on merely random

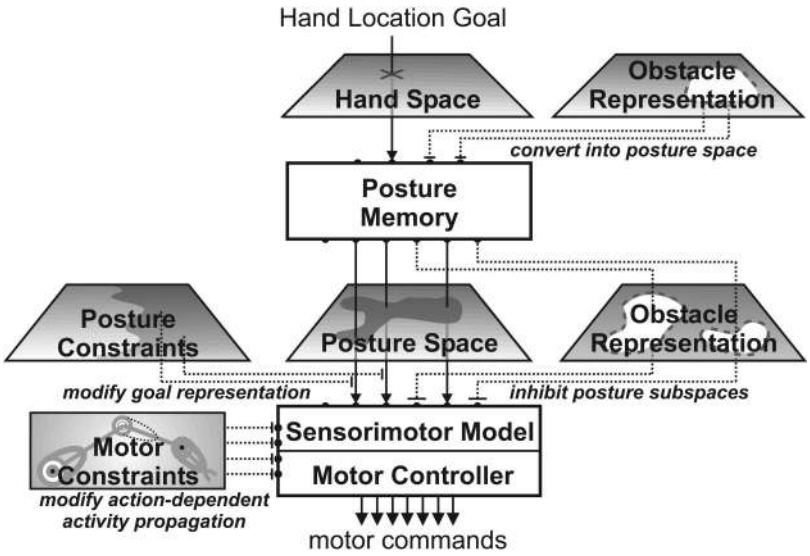

Figure 5. The SURE_REACH architecture is hierarchically structured, associating intrinsic hand space with intrinsic posture space by an associative posture memory. Both spaces are population encoded. The motor controller processes arbitrary goal activations in posture space by means of an inverse sensorimotor model to invoke motor commands, depending on the actual arm state. Additional constraints can be imposed on the goal representations in posture space, posture subspaces, or the movement preparation process realized in the motor controller.

motor neuron excitations. The ideomotor learning approach, which underlies the model, does not require that movements are intentionally directed during learning. More elaborate learning strategies, however, might be used that could account for the goal-directed reaching and exploration strategies observed in infants (e.g., von Hofsten, 2004).

During goal-directed movement production, a goal location may be activated in hand space. The resulting neural activity is propagated into posture space by means of the posture memory, resulting in a redundant goal representation in posture space. The resulting neural activity in posture space is further propagated within posture space by means of dynamic programming that is based on the inverse sensorimotor model. The results are posture-based, action-dependent activation maps that cover the complete posture space. The activation maps essentially encode the suitability of each available motor action at each location in posture space for reaching the closest activated goal posture. Taken together, they encode a sensory-to-motor mapping in posture space. Given the current arm posture, the activation maps provide the motor actions that lead to the closest goal posture. The trajectory to the goal posture unfolds in time because of the sensory-to-motor mapping in combination with a closed-loop control process. Which goal posture is finally reached consequently depends on two criteria: (a) The posture memory needs to encode that the goal posture coincides with an activated hand goal location, and (b) the inverse sensorimotor model needs to encode that the goal posture lies closest to the current arm location.

The modular, hierarchical encoding enables the flexible imposition of additional constraints on hand space, posture space, and on the activity propagation that is based on the sensorimotor model (cf. Figure 5): (a) The chosen population encoding enables the activation of complex hand location goals; (b) obstacle representations can inhibit parts of the hand space, causing the arm to avoid these regions and generate alternative movement trajectories when the inhibition is propagated through to posture space (see Figure 5, right-hand side); (c) activated goal-posture representations can be modified by posture constraints, adding, for example, a preference for comfortable end postures or accounting for additional joint angle constraints (see Figure 5, left-hand side); (d) finally, motor constraints can be imposed onto the activity propagation, which is mediated by the sensorimotor model, accounting for, for example, injured joints (see Figure 5, bottom left-hand side).

In the following subsections, the arm and the actual neural implementation of SURE_REACH are described in detail. First, the arm that is used to evaluate the model is specified. Next, the implementation is detailed. Finally, we exemplify how the activation of goal(s) and potentially additional constraints trigger appropriate motor activity. In subsequent sections, we rigorously evaluate our implementation of SURE_REACH and show that the architecture accounts for various behavioral empirical data patterns.

A Three-Joint Planar Arm

The body controlled by the current SURE_REACH implementation is a three-joint planar arm (see Figure 6). The three limbs have the lengths $l_1 = 1.0$, $l_2 = 0.8$, and $l_3 = 0.6$. The shoulder and elbow joints are allowed to rotate within $\pm 180°$, whereas the wrist joint is restricted to values between 0° and 180°. Note that shoulder and elbow joints, however, cannot circle, that is, no jumps from $-180°$ to 180° are possible.

Each joint is controlled by two antagonistic actuators, which cause either a clockwise or a counterclockwise rotation. An array of motor neurons $\tilde{y}$ of size y activates the actuators. The activity of

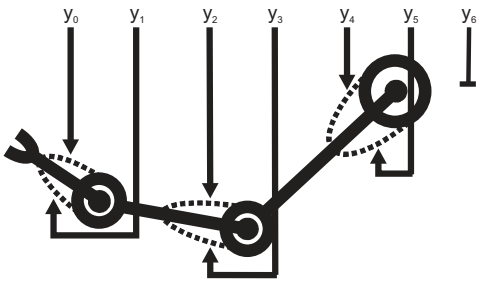

Figure 6. The three-joint arm is controlled by three pairs of antagonistic actuators. The actuators are gradually controlled by motor neurons y_i. Additionally, a seventh action exists that has no effect on the arm.

each motor neuron y_i lies between zero and one. The actual joint displacement per time step during the simulation is calculated by subtracting the activation level of the two antagonistic actuators and then by multiplying the normalized result with a gain factor g. The gain factor determines the maximal angular displacement of a joint in one iteration. If not stated differently, the gain factor is set to $g = 15°$. The total number of encoded actuators in the arm is $y = 7$: two for each of the three joints and one null actuator. Any weighted combination of actuators is possible. The null actuator enables the arm to stay in place.

Of course, the arm lacks the complexity of a human arm, but it captures two important features. First, the arm has redundant degrees of freedom so that each desired hand location can be realized by multiple arm postures, and each arm posture can be reached by many different action sequences. Second, temporal relationships have to be considered to control the arm. Most goal states cannot be reached in one time step by one action command but require the execution of an action sequence. Thus, although joint torques and velocities are not modeled, the arm captures the challenging property that, in general, goals cannot be reached by the definition of a single set of input values.

Hand and Posture Space

Hand end-point space and arm posture space are represented by population codes, which cover the spaces with uniformly distributed, partially overlapping receptive fields. Because of the overlap, a particular position in hand or posture space is encoded by a unique activity distribution of those neurons whose receptive fields overlap with the position.

Hand space is encoded by a neuronal array $\vec{h}$ of size $h = 21 \times 21 = 441$, with local activation patterns for each neuron (Atkeson, Moore, & Schaal, 1997). Each neuron h_i of $\vec{h}$ fires if the current hand coordinates (x_{hand}, y_{hand}) are sufficiently close to the neuron's center (h_i^x,h_i^y):

$$h_i = \max\left(1.0 - \frac{|x_{hand} - h_i^x|}{.24};0\right) \times \max\left(1.0 - \frac{|y_{hand} - h_i^y|}{.24};0\right). \quad (1)$$

This sets the activation of a neuron to 1.0 if the invoked stimulus coincides with its center.

The centers of the neurons are distributed equidistantly in the coordinate space. The covered squared area is centered on the shoulder joint and has a side length of twice the length of the extended arm (cf. Figure 7). The receptive fields of adjacent neurons overlap in such a way that the activity of a neuron reaches zero at the centers of the eight neighboring neurons. The resulting distance between adjacent neural centers is 10% of the length of the stretched arm (i.e., .24 arm-length units) because the centers of the border neurons lie directly on the boundaries of the covered area (i.e., 2.4 arm-length units from the center in the x or y direction). Consequently, any location of the hand is uniquely encoded by the four closest neurons surrounding the location.

The posture space is represented likewise by a neuronal array $\vec{p}$ of size $p = 9 \times 9 \times 5 = 405$. The neural receptive fields cover the entire posture space (360° × 360° × 180°). The consequent distance between adjacent centers is 45° (again, the centers of the border neurons lie directly on the boundaries). Dependent on the current arm posture (ϕ_0, ϕ_1, ϕ_2), each neuron has the following activity:

$$p_i = \prod_{j=0}^{2} \max\left(1.0 - \frac{|\phi_j - p_i^{\phi_j}|}{45°};0\right), \quad (2)$$

where $p_i^{\phi_j}$ are the preferred joint angles of each neuron p_i. Thus, all neurons are broadly tuned to 90° wide, overlapping receptive fields in each dimension. Posture space encodes any possible posture uniquely by the activities of the eight surrounding neurons in the 3-D spatial encoding.

Posture Memory

The posture memory encodes the inverse kinematics of the arm. That is, it learns to associate hand end-point activations with corresponding, redundant posture space activations. The posture memory is implemented as a fully connected single layer neural network, with input arrays of neurons that encode the hand space ($\vec{h}$) and the posture space ($\vec{p}$). The network weights encode the

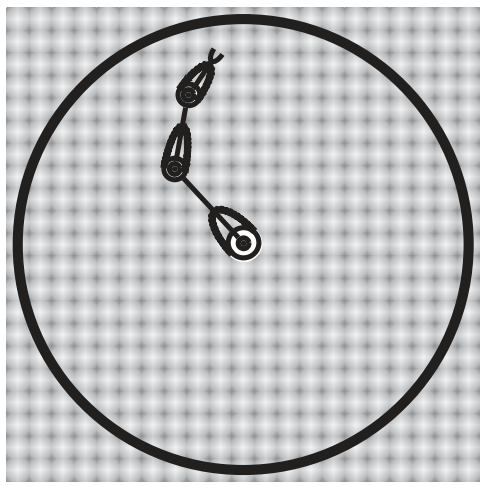

Figure 7. The arm is embedded in a 21 × 21 population-encoded hand end-point coordinate space. The single neurons cover overlapping rectanglar areas with pyramidal activity profile.

degree of correlation between postures and hand positions. The weights are stored in a $p \times h$ matrix, W_{PM}. To compute the appropriate posture-goal activity $\vec{p}_g$ for a desired hand goal $\vec{h}_g$, the activation of the hand position neurons is propagated through the neural network as follows:

$$\vec{p}_g = W_{PM} \times \vec{h}_g. \tag{3}$$

To train the neural network, a Hebbian learning rule (Hebb, 1949) is applied:

$$W_{PM}(t) = W_{PM}(t-1) + \varepsilon \vec{p} \vec{h}^T, \tag{4}$$

where $\vec{h}^T$ denotes the transpose of column vector $\vec{h}$. The learning rate parameter ε modulates the speed of learning. The unsupervised learning rule increases the synaptic weights between neurons that are active at the same time, associating hand locations with all corresponding posture encodings.

Note that because hand space and posture space are population encoded, the joint angles of different postures that match one hand location are not intermixed. Moreover, the overlapping receptive fields of the population codes result in an implicit generalization in posture memory. During training, the neural encoding of any hand coordinate is associated with the corresponding neural encoding in posture space. Because of the redundancy of the arm and the associative learning rule, hand coordinate encodings will be associated with progressively larger subspaces in posture space, comprising the experienced coinciding arm posture encodings. The associated posture subspaces may not necessarily be convex or fully connected.

Sensorimotor Model

The sensorimotor model encodes action-grounded distances in posture space. It consists of $y = 7$ action columns. Each action column is associated with one of the seven motor commands and consists of a synaptic weight matrix, in which the experienced sensorimotor contingencies are encoded. During model learning, an action column learns to associate those postures with one another that were experienced in (not necessarily immediate) succession when the encoded action was executed. Thus, the synaptic weights of each action column correlate contiguous posture encodings with each other action-dependently.

Each action column i of the sensorimotor model represents those correlations in a neural matrix W_i of size $p \times p$ ($405 \times 405 = 164{,}025$), dependent on the amount of executing action y_i. That is, matrix W_i associates successively experienced posture encodings $\vec{p}$, dependent on the executed actions. To be able to associate more remote postures with each other, leaky integrator neurons $\vec{r}_i$ (of size $r_i = 405$) are implemented in each column i. These neurons encode traces of previous posture activations. At time t, given that posture encoding $\vec{p}(t-1)$ was perceived last and $\vec{y}(t-1)$ was executed, the leaky integrator neural activity $\vec{r}_i(t)$ is updated as follows:

$$\vec{r}_i(t) = y_i(t-1)\vec{p}(t-1) + \rho \vec{r}_i(t-1), \tag{5}$$

where ρ is the activity decay coefficient of the leaky integrator neurons. The equation causes the columnar leaky integrator neurons $\vec{r}_i$ to encode a perceptual trace of posture encodings $\vec{p}$, where the trace depends on the respectively executed motor actions y_i.

The associative matrices W_i are again learned by a Hebbian learning mechanism, which depends on the currently perceived posture $\vec{p}(t)$ and the perceptual trace $\vec{r}_i(t)$. Figure 8 illustrates the circuitry used for motor learning in one action column i. Each weight w_i^{jk} of matrix W_i is updated as follows:

$$w_i^{jk}(t) = w_i^{jk}(t-1) + \delta r_i^j(t) \mathrm{p}^{\mathrm{k}}(t)(\theta - w_i^{jk}(t-1)), \tag{6}$$

where lowercase letters with upper indices indicate particular values in the weight matrix W_i, the columnar leaky integrator neurons $\vec{r}_i$, and the current posture encoding $\vec{p}(t)$. The degree of weight update is controlled by learning rate δ. To prevent the network weights from growing infinitely high after long training phases, the upper threshold θ is used. The settings of the parameters are specified in the Appendix.

Because of the action-dependent perceptual traces in the columnar leaky integrator neurons $\vec{r}_i$, action-dependent sensory associations develop in each action column. The more remote a posture is from another posture, the weaker the association in matrix W_i will be due to the discounting factor ρ of the leaky integrator neurons. Moreover, matrices W_i will form stronger associations between posture encodings between which action y_i contributes to reach the one posture from the other posture. Overall, a sensorimotor model develops that encodes the experienced sensorimotor contingencies in the matrices W_i.

During learning, motor command $\vec{y}$ is purely determined by an endogenous excitation generator that causes random motor commands and hence random movements. When goal-directed behavior is executed, $\vec{y}$ is determined by an action selection process, which is explained in the following sections.

Motor Controller

The motor controller relies on the sensorimotor model to generate motor activity. Given a current posture goal activation $\vec{p}_g$, the controller generates a sensory-to-motor mapping in posture space by means of dynamic programming dependent on the sensorimotor model. That is, the controller encodes the action that is most suitable to reach the closest activated goal posture from each

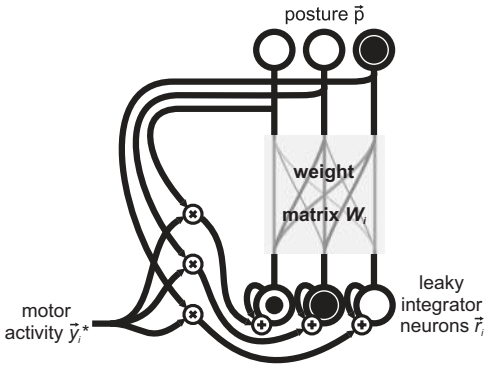

Figure 8. During learning, each action column develops a unique weight matrix (W_i) between successively firing state neurons. The graph visualizes the taken inverse model learning approach. The columnar leaky integrator neurons $\vec{r}_i$ remember an action-dependent, exponentially decreasing trace of recent posture activities enabling the action-dependent association of recent postures with the current posture.

possible posture. Given the sensory-to-motor mapping, the motor controller operates the arm by means of a closed-loop control process. Figure 9 shows the neural circuitry for the generation of a sensory-to-motor mapping and goal-directed action execution in one action column.

Generation of sensory-to-motor mapping. The sensory-to-motor mapping is represented by seven columnar posture-encoded activation maps $\vec{a}_i$ (of size p). During the generation of the sensory-to-motor mapping, the activation maps are initialized by the current posture-encoded goal activity $\vec{p}_g$. The initial activity in the activation maps is then distributed throughout the maps by means of dynamic programming on the basis of the matrices of the sensorimotor model W_i. The activation map of the ith action column $\vec{a}_i$ is updated by the following equations:

$$\vec{a}_i^* \leftarrow max\left\{\beta\left(\gamma\frac{\Sigma_{j\neq i}\vec{a}_j}{y-1} + (1-\gamma)\vec{a}_i\right), \vec{p}_g\right\}, \tag{7}$$

$$\vec{a}_i \leftarrow \vec{a}_i^* + W_i \times \vec{a}_i^*, \tag{8}$$

where $y = 7$ is the number of action columns, *max* is an operator that computes the entry-wise maximum of two vectors, and β and γ are scalar decay factors. The settings of these parameters are specified in the Appendix. Both the goal-activation pattern $\vec{p}_g$ and the action columnar-activation vectors $\vec{a}_i$ are normalized in each iteration so that the component values in each vector add up to 1.0.

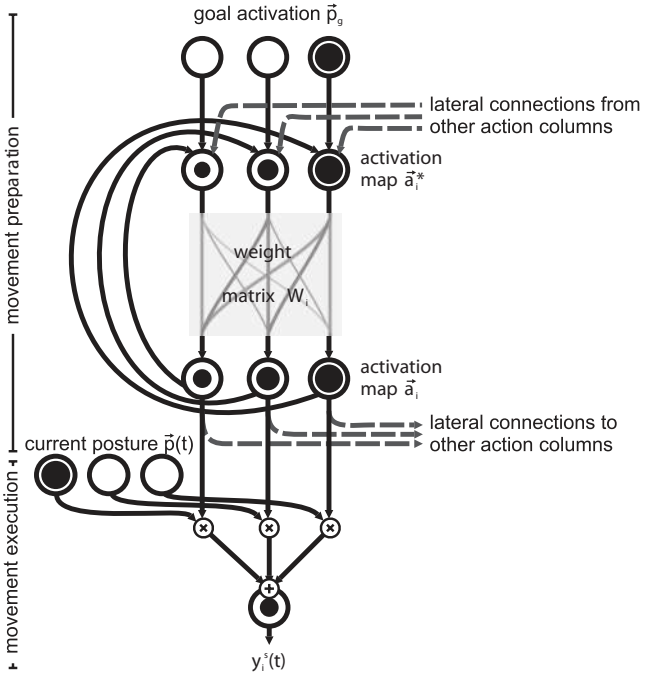

Figure 9. During movement preparation, an action column propagates goal activation $\vec{p}_g$ with its learned weight matrix W_i, combining the activity with activities from other action columns and sending the propagated activity to other action columns. The resulting activation map $\vec{a}_i$ is read out during movement execution by multiplicative units that determine the suitability of the motor command that is associated with each action column dependent on the current posture state. The solid circles inside the represented neurons indicate an exemplar activation propagation.

The two equations iteratively propagate activity inversely within the columnar posture space encodings starting from the goal-activation pattern ($\vec{p}_g$). Parameter γ balances the activation propagation within one action column ($\vec{a}_i$), with the activation propagation stemming from other action columns ($\vec{a}_{j\neq i}$). Parameter β discounts activation propagation and ensures that activation ceases when no goal activation is applied. In effect, the activation pattern in each action column encodes a distance measure in posture space to the currently activated goal(s), given action i is executed first. Larger values indicate smaller distances.

The activation propagation process in one action column is shown in Figure 9. The illustrated goal is to move to the posture represented by the rightmost goal neuron (degree of activation is indicated by the radius of the inner solid circle). Let us assume that the action column is associated to a motor command that usually moves the arm from the posture encoded by the leftmost neuron to the posture encoded by the rightmost neuron. In this case, synaptic weights will form during learning that connect the right neurons to neighbors on the left. During activity propagation, the goal activation consequently spreads to the postures on the left, as is indicated in the activation map $\vec{a}_i$ in Figure 9. Because the synaptic weights are not symmetrical and differ in the action columns, a unique activation pattern emerges in each action column. The propagation of activity between action columns (cf. Equation 7) ensures that postures also receive activations that can only be reached by an extended sequence of different motor commands. Together, the activation maps of the seven action columns encode the sensory-to-motor mapping, which is used for action execution.

Action execution. Given the seven activation maps, actions can be selected for execution by comparing the current activities in the activation maps $\vec{a}_i(t)$ with the current arm posture encoding $\vec{p}(t)$ (see Figure 9, bottom). The execution of the chosen action then results in a change of the current arm posture and the subsequent generation of the next motor command.

To assess the suitability of the different actions y_i given a particular current posture $\vec{p}(t)$, we multiply the current activity of the activation maps $\vec{a}_i(t)$ with the current posture encoding:

$$y_i^S(t) = \vec{p}(t)^T\vec{a}_i(t). \tag{9}$$

Figure 9 (bottom) illustrates the inner product computation dependent on the state of the current posture in one action column. In the figure, the leftmost neuron of the encoding of the current posture is active. Thus, only the activity in the leftmost neuron of the activation map is used to compute action suitability.

The action suitability output is then squared and normalized to emphasize the suitability of the best current action command:

$$y_i^{net}(t) = \frac{y_i^S(t)^2}{\Sigma_{j=0}^{6} y_j^S(t)^2}. \tag{10}$$

The activations of opposing actuators (those that cause antagonistic movements) then cancel each other out. That is,

$$y_i^{net}*(t) = \begin{cases} y_i^{net}(t) - y_j^{net}(t), & \text{if } y_i^{net}(t) > y_j^{net}(t); \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

given i and j are antagonistic motor action pairs. Finally, the resulting absolute activities are normalized to 1.0 and multiplied by gain factor g:

$$y_i(t) = g\frac{y_i^{net}*(t)}{\Sigma_{j=0}^{6} y_j^{net}*(t)}. \quad (12)$$

In this way, movement velocity becomes independent from absolute activation levels. The applied normalization procedure worked well in all experiments conducted for this article. However, in principle, other or further modifications can be made to generate the motor commands.

Goal-Directed Motor Control

Figure 10 shows a complete goal-directed movement preparation and execution process in the implemented SURE_REACH architecture. The initial goal-activation pattern necessary for the generation of the sensory-to-motor mapping usually stems from a hand space goal-activation pattern $\vec{h}_g$. This pattern may encode one particular desired hand location, but it may also encode a whole, possibly weighted, set of hand locations. The goal-activation pattern is then propagated through to posture space, determining the posture goal activity $\vec{p}_g$ by the posture memory matrix W_{PM} (cf. Equation 3). Alternatively, though, the posture goal activation may also be set directly by initializing $\vec{p}_g$ to certain values. Moreover, the activity of $\vec{p}_g$ may be modified further by other posture constraints.

Given a goal-activation pattern in posture space $\vec{p}_g$, the pattern is diffused by means of the synaptic weights of the sensorimotor model with dynamic programming, as specified in Equations 7 and 8 above. This results in the sensory-to-motor mapping, encoded within the seven action maps $\vec{a}_i$, necessary for the closed-loop control of the arm. Finally, arm execution starts by determining an action command $\vec{y}$, given the current arm posture $\vec{p}$, as described in Equations 9, 10, 11, and 12. The result is a closed-loop control process that directs the arm to the closest activated goal posture, which is determined by the dynamic programming process based on the encoded sensorimotor model.

As discussed above, to be able to flexibly account for various task and goal constraints, goal-directed motor control systems need to store redundant solutions and resolve redundancy on the fly before and during action execution. SURE_REACH encodes two types of motor redundancies. First, the posture memory encodes kinematic redundancies, consequently activating all known arm postures that coincide with a given goal location. Second, the inverse sensorimotor model encodes sensorimotor redundancies. In effect, the model implicitly encodes all possible trajectories to reach a goal posture from any other posture. Thus, SURE_REACH knows in principle about all alternative goal postures, given a goal location, and also how all these alternative goal postures may be reached.

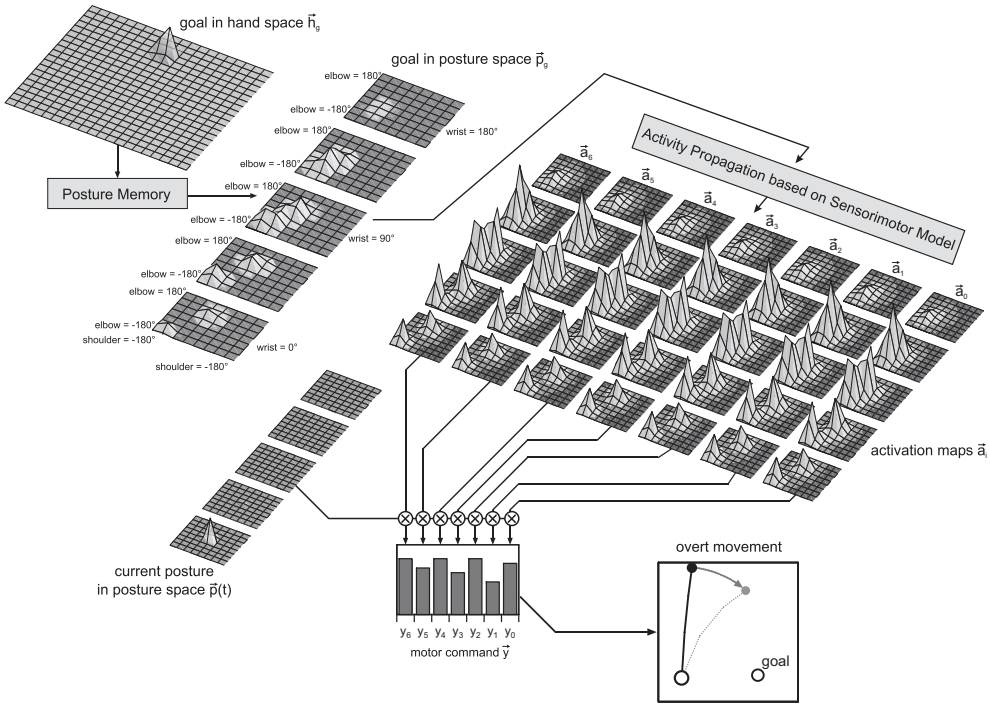

Figure 10. Goal-directed behavior in SURE_REACH usually begins with a goal activity induction $\vec{h}_g$ in hand space (left top). The hand space activity is then transformed into a posture-based goal representation $\vec{p}_g$ with the posture memory matrix W_{PM} (inverse kinematics). The goal activity $\vec{p}_g$ then induces the dynamic programming process within posture space, which results in the generation of columnar activation maps $\vec{a}_i$ for each possible action i. To control a movement, the activities in the activation maps are read out, combining them with the current posture representation $\vec{p}$. The result is a motor command $\vec{y}$, which causes the movement toward the activated goal.

Redundancy is then resolved by means of the dynamic programming process. Essentially, dynamic programming determines all possible trajectories to the activated goal postures in parallel within the emerging activation maps $\vec{a}_i$. The resulting sensory-to-motor mapping encodes all shortest trajectories to the closest goal postures from all possible postures. Through the use of closed-loop control, the trajectory from the current arm posture is executed.

The population encoding of posture space plus the embedded sensorimotor model enable the efficient generation of these sensory-to-motor mappings. In the current iterative implementation, the effort to generate a mapping from the current state to the goal is linear in sensorimotor distance to the goal (as encoded in the matrices M_i) times the (constant) size ($p \times p$) of the matrices M_i used during activity propagation (see Equation 8). In a possible parallel implementation of the dynamic programming process, effort would be linear times negligible constant effort.

The redundancy resolution process is immensely flexible, enabling the imposition of at least four further task constraints, as is illustrated in Figure 5. First, any type of hand goal-activation pattern $\vec{h}_g$ can be imposed, in principle. The dynamic programming process simply causes movement to that hand goal, which caused the strongest posture goal activation. If there are multiple similarly strongly activated goal postures available, the arm will approach the closest posture. Second, obstacle representations can inhibit the activation of hand space or posture space subspaces. In the case of a hand space inhibition, this inhibition has to be propagated through to posture space, for example, by using the posture memory. The inhibition can prevent the arm from attempting to reach certain posture subspaces by setting the activity maps $\vec{a}_i$ in these subspaces to zero, causing the dynamic programming process to generate alternative trajectories around the inhibited subspaces (possibly reaching an alternative goal posture). Third, additional posture constraints can be imposed, modifying the goal-activation pattern $\vec{p}_g$. In this way, for example, certain arm postures or joint angles can be preferred over other postures that coincide with them. Fourth, additional motor constraints can be imposed. In this case, the contribution of each action during the dynamic programming-based activation propagation process (see Equation 7) may be modified, causing SURE_REACH to prefer the execution of particular actions over alternative ones.

In the following sections, we evaluate the SURE_REACH model confirming the four types of behavioral flexibility. Moreover, we show that the learning process is robust to parameter modifications, and, most important, we compare the behavior of SURE_REACH to available behavioral data from the psychological literature. Before we start with the rigorous model evaluation, however, we give some illustrative examples of trajectory generation and motor control in the current implementation of SURE_REACH.

Three Illustrative Examples

The following examples illustrate the motor activation process in SURE_REACH. We show the activation propagation within the actual simulation but focus on movements that do not require wrist motions for visualization purposes. The evaluations in the subsequent sections are all conducted on the full three-degrees-of-freedom arm.

Example: Approaching a Particular Posture

Figure 11A shows a typical activity propagation process in the SURE_REACH implementation within the described arm, given one particular goal posture is activated. However, in this illustrative example, the wrist angle does not change from 0° to enable the 2-D representation of the posture space. Figure 11A shows cross sections (wrist angle is 0°) of activation maps $\vec{a}_i$. Columns show different activation maps. Rows show the maps at different moments during their generation. White areas are not activated at all; dark areas are highly activated. The data stems from a well-trained (unconstrained) model.

Initially, only one particular goal posture is activated in this example. Next, the activity is propagated. This can be clearly seen in row $t = 1$ where, for example, a counterclockwise movement of the shoulder (left column) coactivates the clockwise space more strongly, whereas a clockwise movement (second left column) coactivates the counterclockwise space more strongly. Because of the leaky-integrator property of neurons $\vec{r}_i$ during learning, not only the immediate surrounding is activated after one activation propagation iteration but also more distant posture subspaces (row $t = 1$ in Figure 11A). These far-reaching connections facilitate the movement preparation process and enable faster movement onset.

The motor commands, which are ultimately sent to the actuators, depend on the activation of the neurons representing the current state in the different activation maps. Figure 11B shows the corresponding sensory-to-motor mappings, which are generated from the activation maps in the seven action columns by applying the normalization procedure described in the previous section (see Equations 9, 10, 11, 12). The action columns of the motor controller trigger action activity for many locations in posture space. However, between some state neurons no synaptic connectivity may have emerged because they represent very distinct postures, and the transition time between those postures is high. Thus, the activation maps for row $t = 1$ do not nearly cover the entire posture space. To establish connections between highly remote postures, activation needs to be propagated further. The activation maps for $t = 2$ cover a substantially larger part of the posture space. After 10 iterations, suitable sets of motor commands are available for the entire posture space.

Movement can be initialized as soon as the activation pattern reaches the current arm posture. As the arm posture changes, the closed-loop control process reads out the activation of the state neurons and issues motor commands accordingly. Figures 11C and 11D show the resulting movement trajectories in hand space and posture space, respectively.

Example: Activation Maps With Multiple Goals

So far, we discussed the activation of a single posture. The activation propagation process can also be initialized with more than one goal neuron activated. In this case, the sensory-to-motor mapping does not lead toward a single posture but toward a set of goal postures. Thereby, each of the postures in the goal set represents an acceptable final state. In Figure 12A, the goal set is the solution of the inverse kinematics problem of moving the hand to a specific location and thereby maintaining a stretched wrist joint

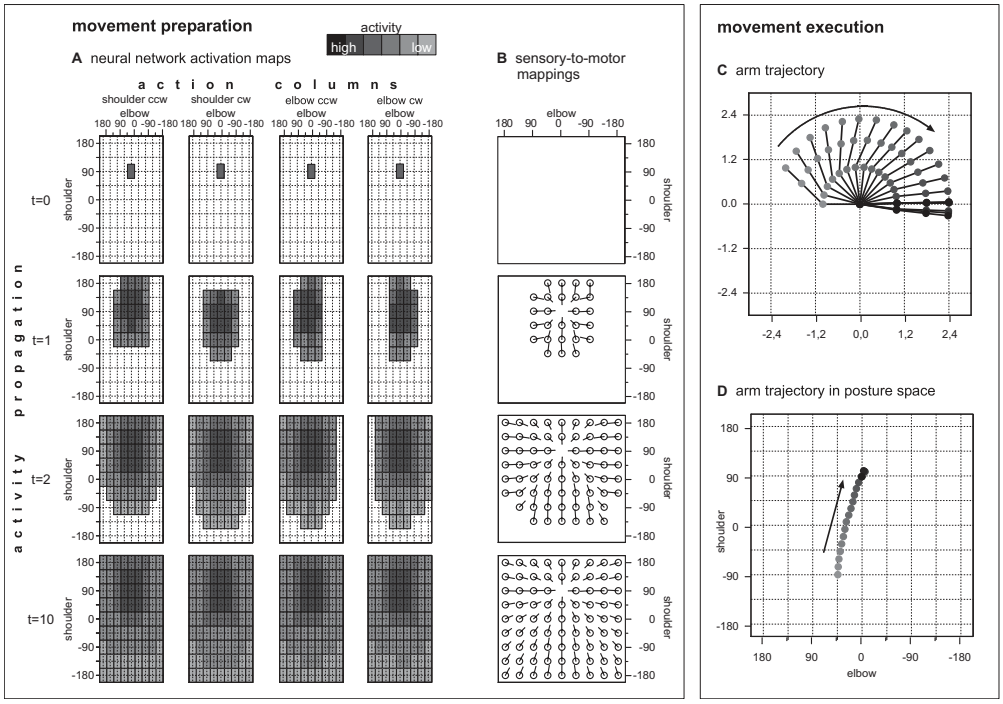

Figure 11. A: The activation maps show the activity of the neurons that encode postures with $\phi_{wrist} = 0°$, in different action columns at activity propagation iteration steps $t = 0$, $t = 1$, $t = 2$, $t = 10$ (rows). B: The sensory-to-motor mappings are derived from weighting the activation levels in the different neurons in the action columns. They show the direction and strength of a motor command executed by the motor controller, dependent on the actual arm posture. C: The movement that the motor controller exerts once a sensory-to-motor mapping is available for the initial joint configuration. D: The resulting movement in posture space ($\phi_{wrist} = 0°$). CW = clockwise; CCW = counterclockwise.

(see Figure 2A). The sensory-to-motor mapping directs the shortest path to the goal for all possible arm postures. Furthermore, goal sets with an infinite number of solutions are possible. For example, Figure 12B charts the sensory-to-motor mapping to any posture with stretched wrist and shoulder joints.

Example: Subspace Avoidance

SURE_REACH also makes it possible to avoid (generally) arbitrary subspaces, consequently realizing, for example, obstacle avoidance. In the following example, we inhibit neural activity for

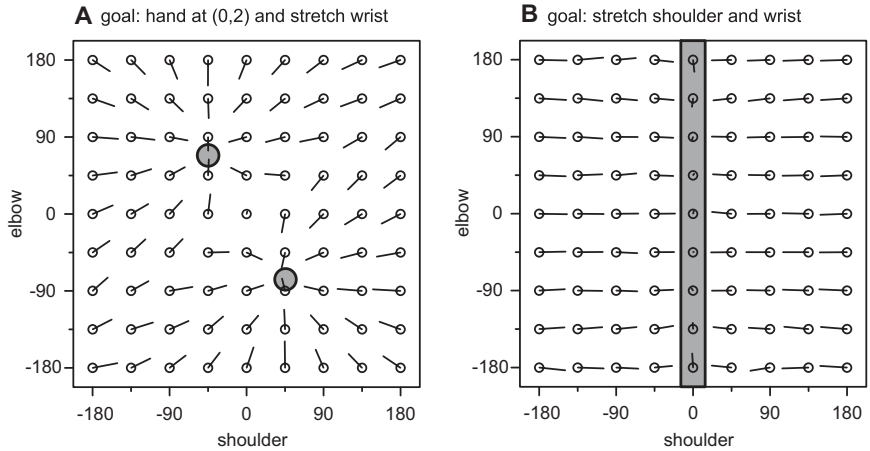

Figure 12. The charts show intersections (for $\phi_{wrist} = 0$) of sensory-to-motor mappings for different goal sets (in gray scale). A: The goal consists of two different joint angle configurations. B: The goal is the infinite set of postures with a stretched shoulder and wrist.

certain postures, effectively disabling the arm to enter some areas in posture space. In the subsequent evaluations, we also show that subspaces in end-point coordinate space can be avoided by inhibiting neurons in coordinate space and propagating this inhibition through to posture space by using the posture memory (encoded in matrix W_{PM}). Note, however, that currently the arm only encodes its end-point in coordinate space so that it is only possible to prevent end-point collisions.

Figure 13A shows an example in which the posture space is constrained, disallowing postures in which the elbow is flexed when the shoulder is in an angular area between $-90°$ and $+90°$. The movement starts with the upper limb pointing downward ($\phi_{shoulder} = 135°$) and a 90°-flexed elbow. The goal posture requires a counterclockwise movement of the shoulder until the upper limb points straight down. The elbow and wrist angles in start and goal posture are identical, but the additional constraint requires an extension of the elbow during the movement.

The constraint is realized by inhibiting all postures that collide with the virtual obstacle in the activation maps $\tilde{a}_i$. The constraint requires the activation to spread from neurons in the clockwise elbow rotation action column to neurons in the counterclockwise shoulder rotation column, as can be inferred from the maps in Figure 13A. The resulting sensory-to-motor mapping (see Figure 13B) causes substantial changes to the motor commands during action execution. Figures 13C and 13D show the changes in the elbow motor commands to avoid the obstacle, first stretching it and then flexing it again.

Evaluation and Application

After this exemplification of SURE_REACH's capabilities, we now systematically evaluate learning and behavior in the architecture on the full three-degrees-of-freedom arm. The evaluation starts with an assessment of the general learning accuracy of the model and its parameter dependency. Next, the behavior of the model is characterized and compared with various behavioral findings in humans. First, the effects of extensive training on reaction and movement times are considered. Next, we show that the dynamics in SURE_REACH exhibit priming effects that are comparable to experimental data from psychological experiments (Schmidt, 2002). Finally, it is confirmed that SURE_REACH benefits from the representation of motor redundancies on the end-posture level as well as on the trajectory level, exhibiting behavior in accordance with experimental data (Bock & Arnold, 1992; Cruse & Steinkühler, 1993; Erlhagen & Schöner, 2002; Fischer, Rosenbau, & Vaughan, 1997; Jaric, Corcos, & Latash, 1992; Soechting, Buneo, Herrmann, & Flanders, 1995).

General Properties

A model that accounts for motor learning and control should improve during learning and should acquire at least some dexterity. To show this, we evaluate the overall improvement of movement accuracy during learning in SURE_REACH. First, the sensorimotor model and the motor controller alone are evaluated for

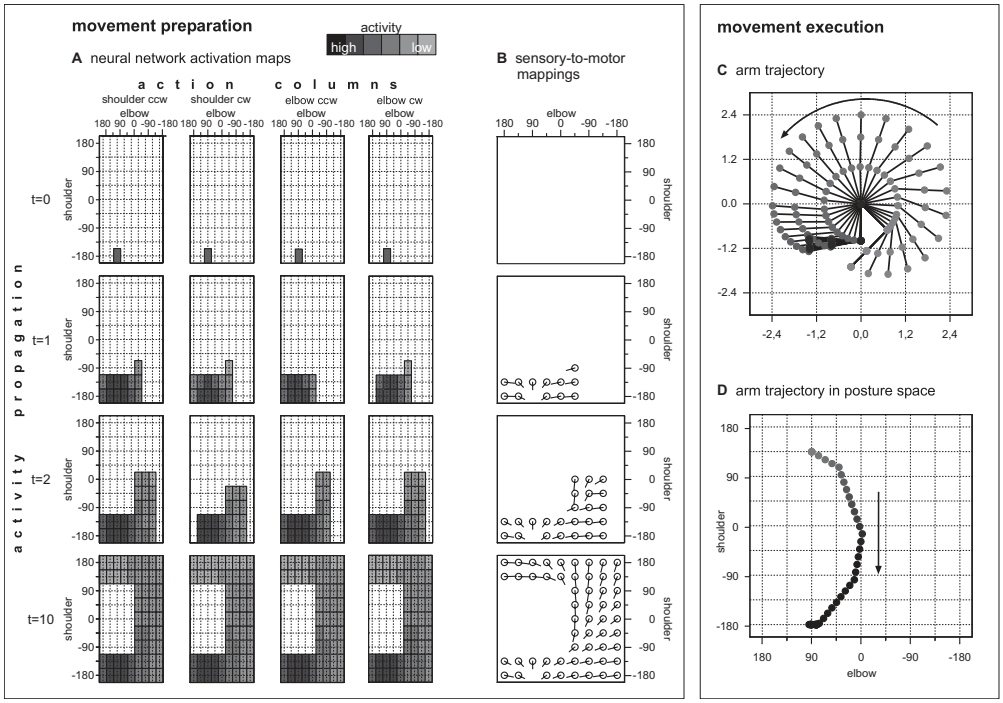

Figure 13. A: The maps show the activation levels of neurons that encode postures with $\phi_{wrist} = 0°$ in different action columns and at different activity propagation iteration steps (rows). State neurons that overlap with the simulated obstacle are inhibited (resulting white rectangle in row $t = 10$). B: The emergent sensory-to-motor mapping shows trajectories that avoid the obstacle. C: The movement exerted by the motor controller bypasses the obstacle. D: The movement in posture space. CW = clockwise; CCW = counterclockwise.

movements toward specific arm postures. Second, the complete SURE_REACH model, including its posture memory, is evaluated for movements toward specific hand targets. Finally, to ensure that the model is robust and does not rely on a fine-tuned set of parameters, the sensitivity of the model's performance is tested with respect to its most important parameters.

Motor Controller

To evaluate the overall properties of the model, we trained 10 controllers individually for 1,000,000 time steps each. The 10 controllers differed in two properties. First, their learning experiences differed because each controller executed a different sequence of random motor commands during motor learning. Second, the goals generated during evaluation of each controller differed. During performance tests, each controller had to perform a unique set of test movements, starting from certain randomly selected postures, pursuing particular goals. During motor learning, though, each controller had to perform the identical test movements after various iterations of training. Thus, variances in the performances of the controllers reflect both variances that were due to different learning experiences and variances that were due to different test movements.

Training began by setting the arm to a random posture. Then, a random motor command set was generated and changed every one to four time steps. Each motor neuron was set to 1.0, with a probability of 0.3 and was otherwise 0.0. If none of the motor neurons were activated, this procedure was repeated. In each time step, the neural network weights of the posture memory and the sensorimotor model were updated. To evaluate learning progress, we froze the neural network weights and tested the controller. After a test trial, training was continued from a new random posture.

To assess the accuracy of the sensorimotor model and the capabilities of the motor controller alone, we tested each individual controller in 16 trials to move from randomly selected starting postures to randomly selected target postures (ϕ_0, $\phi_1 \in [-135°; 135°]$, $\phi_2 \in [45°; 135°]$). Target posture activity ($\vec{p}_g$) was directly generated, bypassing the posture memory. The controller was allowed to take up to 80 time steps to reach the activated goal posture.

The average error of the last 10 time steps of each movement was used to determine the accuracy of a single movement. The error was computed by averaging the absolute differences between the three target and actual joint angles. To assess the accuracy of a controller, we computed the average movement accuracy and the worst movement accuracy during a test phase. Figure 14A shows the average and worst case accuracy of the 10 individual controllers. Movement error before training was on average 82.1° (SD = 5.84°). After 1,000,000 steps of training, the average error dropped to 3.52° (SD = .114°), and the average error of the least accurate movement of each controller dropped to 4.43° (SD = .314°). The results confirm that the architecture is capable of reliably moving the arm with reasonable accuracy to desired postures, considering the sparse distribution of the receptive fields (45° between adjacent centers of receptive fields).

Posture Memory

Similar to the evaluation of reaching goal postures, the accuracy of the whole architecture was evaluated. The error is now defined by the Euclidean distance between the goal hand position and the final hand position in percentage of the workspace size (twice the length of the arm). Each controller had to perform 16 movements from random start postures to random hand targets. Hand targets ($\vec{h}_g$) were generated by computing the hand position of random arm postures to ensure that the target was within reach of the arm. Figure 14B depicts the average and maximal error during learning. The average error dropped from 36.5% (SD = 7.07%) to 4.73% (SD = .715%) after 1,000,000 time steps of learning. The average error of the least accurate movements of each controller was 9.32% (SD = 2.70%).

The results confirm that the SURE_REACH implementation can account for accurate goal-directed hand movements. As in the case of the posture space evaluation above, the remaining error in final hand location is due to the low resolution of posture and hand end-point space. Potential solutions to this problem are discussed below.

Parameter and Training Sensitivity

To ensure that the performance is not bound to a specific set of parameters or a specific training setup, we systematically varied

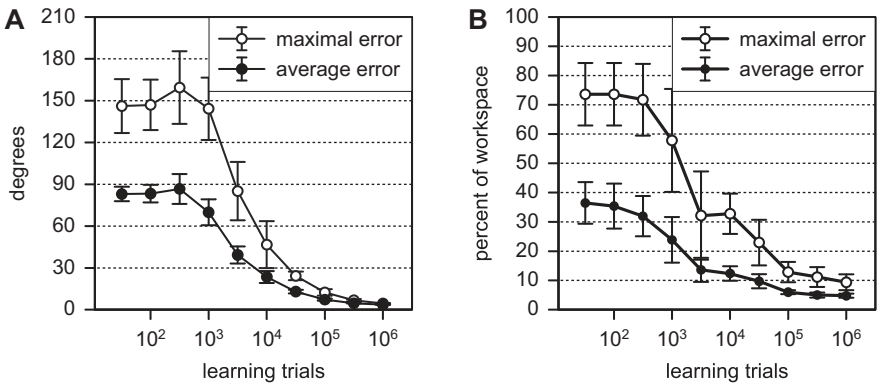

Figure 14. A: The average and worst end-posture error in mean absolute degrees continuously improves during training. B: The average and maximal Euclidean distance from the desired goal location continuously decreases during learning and consistently reaches a low level. Error bars show standard deviations.

three important factors. First, the random activation generation procedure applied in the training phase was varied. Either exactly one random motor neuron was activated or each motor neuron was activated with a probability of .3 or .5 (repeating the action activation process, if no action was activated). Second, the joint angle gain g was set to 11.25°, 15.0°, or 22.5° during learning. Third, parameter ρ, which modulates the leak of the leaky integrator neurons, was set to 0.00, 0.10, or 0.50. Ten individual controllers were trained for each combination of values, and the performance after 1,000,000 time steps of learning was assessed as described above except that only the final arm state of a movement was used to calculate error values.

When the sensorimotor model plus motor controller were tested alone by activating random, single goal postures, we found little impact from the parameter variations. A 3 × 3 × 3 analysis of variance (ANOVA) revealed only a significant influence from the random activation generation procedure, yielding better results if more than one action was active at a time, $F(2, 243) = 7.46$, $p < .01$. However, the variation had little impact on absolute performance: All average end-point errors ranged between 3.25° ($SD = .370°$) and 4.05° ($SD = .199°$).

When the complete architecture was tested by the activation of goal coordinates and the successive posture space activation by the posture memory, a 3 × 3 × 3 ANOVA also revealed a significant influence from the random activation generation procedure, $F(2, 243) = 18.5$, $p < .01$. In this case, better results were obtained when only one action was applied at a time. For the coactivation of multiple goal postures, the results indicate that it is slightly better to learn the sensorimotor model by executing single actions because the inverse model becomes more uniquely dependent on each action. Nonetheless, the results ranged between 3.85% ($SD = .686\%$) and 5.10% ($SD = .727\%$) so that the absolute difference in error remained small.

In conclusion, the small performance differences suggest that parameter and training variations only slightly affect model performance. Thus, the architecture does not need a fine-tuned set of parameters to work, rather it is generally robust.

Effects of Extensive Training

It is a common finding that training not only affects the accuracy of biological movements, but also that movement times and reaction times decrease (Flament, Shapiro, Kempf, & Corcos, 1999; Gottlieb, Corcos, Jaric, & Agarwal, 1988; Lavrysen et al., 2003; Ludwig, 1982). Our model contributes to both findings. First, in SURE_REACH, the time it takes to initiate a movement decreases by encoding temporally far-reaching sensorimotor contingencies. The further reaching the contingencies, though, the longer it takes to encode them during training. Second, the movement time decreases because the sensorimotor model and posture memory encode progressively more alternative movement trajectories and goal postures, respectively.

Reduced Reaction Times

For movements to remote goals, the dynamic programming process requires some time to prepare a movement because movement onset relies on a sufficient spread of activation through the activation maps ($\vec{a}_i$). The time it takes from presenting a target to SURE_REACH until the activation is spread far enough to initiate a movement can be considered the *latency,* or reaction time, of SURE_REACH. This time decreases during motor learning.

The leaky integrator neurons enable the controller to establish direct connections between remote situations and goals, thus being able to replace, or at least enhance, the activation propagation process. If the parameter ρ, which specifies the leak of the leaky integrators, is set to zero, only associations of state neurons that are activated in subsequent time steps are learned. In this case, it is impossible to learn far-reaching connections. The higher ρ, the more far-reaching connections will be learned and the faster action execution is initiated.

To confirm this, we trained three groups of 10 individual controllers for 1,000,000 time steps, with the leaky integrator set to $\rho = 0.00$, $\rho = 0.50$, and $\rho = 0.80$, respectively. The test procedure for movements to different postures was applied. Only the final arm position was used to compute the error value. After 10,000 time steps of training, all controllers were trained well enough to be able to initialize movements to all given goals.

Figure 15A shows that a high value of ρ causes on average a significantly decreased movement latency after 1,000,000 time steps of learning (for $\rho = 0.00$: $M = .663$, $SD = .145$; for $\rho = 0.50$: $M = .356$, $SD = .179$; for $\rho = 0.80$: $M = .113$, $SD = .0922$), $F(2, 27) = 36.9$, $p < .01$. Furthermore, if ρ is set to 0.50 or 0.80, the movement latency progressively decreases during learning. Pairwise t tests revealed that a significant decrease is even detectable between the final two test periods, that is, after 10^5 and 10^6 time steps of learning (for $\rho = 0.50$: $M = .0438$, $SD = .0593$, $t(9) = 2.33$, $p < .05$; for $\rho = 0.80$: $M = .0250$, $SD = .0323$, $t(9) = 2.45$, $p < .05$). On the other hand, if ρ is set to zero, the decrease of movement onset time ceases completely after 100,000 time steps of learning.

Despite the variations of parameter ρ and the consequent differences in the connectivity within the inverse models, no significant impact on the average end-posture accuracy was detectable (for $\rho = 0.00$: $M = 3.71°$, $SD = .417°$; for $\rho = 0.50$: $M = 3.54°$, $SD = .227°$; for $\rho = 0.80$: $M = 3.85°$, $SD = .471°$), $F(2, 27) = 1.64$, $p > .05$. The results confirm that the leaky integrator neurons enable the inverse model to associate far-reaching arm postures. Movement latency is significantly reduced with extensive training without affecting the accuracy of the reaching movements. Hence, SURE_REACH accounts for the empiric finding that movement latencies or reaction times decrease during motor learning.

Improved Movement Times

Besides a reduction of the reaction time, training also reduces the time needed to move to a goal. In the model, two factors are responsible for this effect. First, during training, the inverse kinematics model learns more and more postures that coincide with particular hand locations. A well-trained posture memory provides a broader goal-posture activity ($\vec{p}_g$). It is likely that at least some of the postures activated in the broader goal activity are closer to the starting position than any posture in a smaller subset. Hence, movement transitions get faster on average.

Second, the representation of sensorimotor contingencies gets more reliable during training and covers bigger parts of posture space. Because the determination of the motor commands depends

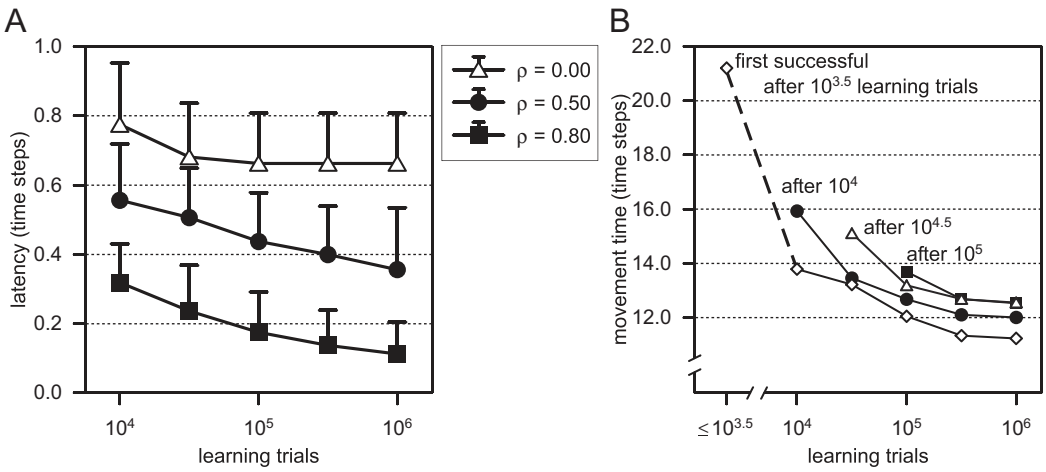

Figure 15. A: Movement latencies are lower and continue to decrease during extensive training if the controller is able to learn far-reaching sensorimotor associations in the sensorimotor model ($\rho = 0.8$ and $\rho = 0.5$). The decrease ceases if the leaky integrator neurons do not carry the activation of past states ($\rho = 0$). B: The movement times for movements to goals that could first be reached after $10^{3.5}$, after 10^4, after $10^{4.5}$, or after 10^5 training iterations. Movement times decrease monotonously. Error bars show standard deviations.

on the learned inverse sensorimotor model, more suitable motor commands are generated after longer learning periods.

To show the influence of training on movement time, we reanalyzed the data of the previous section. Each controller had to perform identical movement tasks after varying amounts of learning. Only movement times from movements that did arrive at the target (distance from hand to target lower than 15% of the workspace size) were compared. For each run, the targets were divided into four groups: those targets that were reached after $10^{3.5}$ learning iterations (16.3% movements total), those that were reached after 10^4 (19.2% movements total), those that were reached after $10^{4.5}$ (29.6% movements total), and those that were reached after 10^5 learning iterations (29.4% movements total). The remaining 5.6% movements required more than 10^5 learning iterations and were not included in the analysis. Because parameter ρ had no significant impact on movement times, all controllers were evaluated independent of this factor. Figure 15B depicts the development of average movement times for the four groups of movements. On average, when a movement is successful for the first time, it takes longer ($M = 15.7$, $SD = 2.21$) than it does after complete training ($M = 12.4$, $SD = 1.50$), $t(29) = 10.8$, $p < .001$. Both the development of posture memory and the sensorimotor model contribute to the decrease of movement time. On the one hand, the average distances of the movement in joint space (d2 norm) decrease from 186° ($SD = 19.2$ to 180°, $SD = 20.6$), $t(29) = 8.06$, $p < .001$. This shows that the posture memory activates end postures that are closer to the goal after complete training compared with those of not fully trained controllers. On the other hand, a similar analysis of the movement for the sensorimotor model alone reveals that movement times also decrease, comparing first successful movements ($M = 11.0$, $SD = 0.868$) with completely trained movements ($M = 10.0$, $SD = 0.752$), $t(9) = 6.11$, $p < .001$. This shows that because of the increase of both, the accuracy of posture memory and the sensorimotor model, movement times decrease during training.

Priming Effects on Movement Execution

Movement preparation and execution often depend on environmental stimuli. Reactions to external stimuli can be facilitated to some degree if primes precede the stimulus that affords a reaction. The influence of these primes on action execution has been extensively studied, mainly with choice reaction time tasks (Dehaene et al., 1998; Kunde, Kiesel, & Hoffmann, 2003; Vorberg, Mattler, Heinecke, Schmidt, & Schwarzbach, 2003).

Continuous movements also have been studied in this context. Participants in an experiment by Schmidt (2002) were shown a red and a green target at opposing directions from their resting index finger. As soon as these targets appeared, they had to point as quickly as possible to the target with a specific color. For example, a participant could be instructed to point to the red target. However, 10 ms to 60 ms before the actual targets appeared either congruent or incongruent primes were displayed for 10 ms. In the congruent case, the primes appeared at the same locations and had the same color as the actual targets. In the incongruent case, the primes appeared at the same locations as the actual targets but with switched colors.

The movement trajectories of incongruently primed targets clearly showed a short motion in the wrong (i.e. primed) direction before this error was corrected and the trajectories approached the actual target. The extent of the motion in the wrong direction was significantly larger if the incongruent primes were shown longer before the targets appeared (longer stimulus onset asynchrony [SOA]). The primes were replaced by the targets for at least approximately 200 ms before movement onset. The results were explained by assuming that, as soon as the primes were visible, they contributed to the generation of a response until the target appeared, thus causing initially misguided movements in the incongruent case.

SURE_REACH accounts for these effects. We simulated priming of different durations by preactivating the activation

maps for one to five time steps with a congruent or incongruent target posture, without actually executing the movement (targets: stretched arm, $\phi_{elbow,wrist} = 0$, with shoulder either $\phi_{shoulder} = -90°$ or $\phi_{shoulder} = 90°$, movements started from $\phi_{shoulder,elbow,wrist} = 0°$). Then the primes were replaced by the targets and the movement was initialized. To measure the extent of movements in the wrong direction, the maximal Euclidean distances between the hand coordinates associated with the goal posture ($\pm 2.4, 0$), and the hand coordinates during each movement were assessed. The movements were executed by 10 controllers that were individually trained for 1,000,000 time steps. To enhance the effect, we increased the inertia of the dynamic generation of the sensory-to-motor mapping by slightly increasing parameter β to .48 (see Equation 7) and seting gain factor g to 6°.

Figure 16A shows the distance between the hand and the actual target during the movement. Shown are average performances of 10 controllers in both movement directions. If the prime is congruent, the arm moves monotonously toward the target. However, if the prime is incongruent, an initial movement in the wrong direction is made. As in Schmidt's (2002) experiment, the extent of this movement depends on the duration between prime onset and the onset of the actual target (SOA). Figure 16B plots the average maximal distance of the movements against different SOAs for the incongruent case. A one-way ANOVA revealed a significant main effect for the SOA, $F(4, 45) = 42.2$, $p < .01$.

To summarize, the population-encoded space representation is suitable to model preparatory effects of movements. The model accounts for the preparation of movements, once a goal is selected. Moreover, priming effects that alter the encoded goal representations can be simulated because of the neural network dynamics.

Benefits of Memorizing Kinematic Redundancy

One of the claimed advantages of storing multiple solutions for a single goal is enhanced behavioral flexibility. To test this flexibility, we now constrain the joint angle space to examine whether SURE_REACH benefits from storing redundancy at the kinematic level. We show that, as is observable in humans, the final posture of a movement depends on the starting posture.

In human hand movements, the final arm posture that places the hand at the target location is dependent on the starting posture (Cruse, Brüwer, & Dean, 1993; Fischer et al., 1997; Jaric et al., 1992; Soechting et al., 1995). This exploitation of kinematic redundancies seems to minimize movement costs (Fischer et al., 1997).

The posture memory activates sets of postures that all realize a desired hand position. If this redundancy is beneficial, the controller should exert the shortest possible path in joint angle space to reach a goal location. Less direct but still accurate movements should be triggered if additional constraints apply.

We trained 10 unconstrained individual controllers with the procedure described above and tested each controller in five different scenarios. In the first one, there were no constraints except for the targeted hand location. In the second and third scenarios, besides the targeted hand location, the desired shoulder angle was set to 0° and 45°, respectively. In the fourth and the fifth scenarios, besides a desired hand position, the desired elbow angle was set to 0° and 45°, respectively. During each test phase, 16 movements were made toward random goals, but only movements toward goals that could be theoretically reached with the given constraints were included in the evaluation. The additional constraints were imposed by inhibiting all neurons in posture memory output ($\vec{p}_g$) that did not satisfy the joint angle constraints.

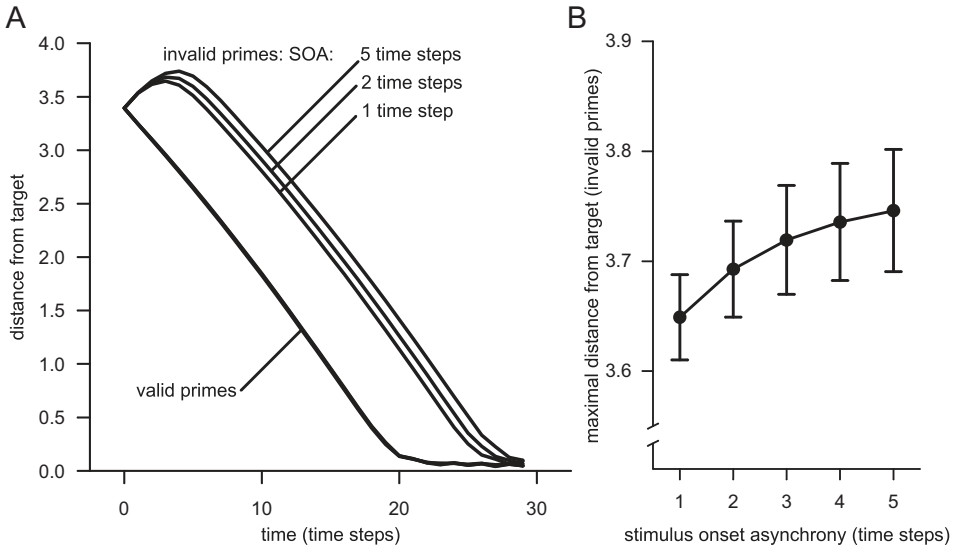

Figure 16. A: The average distance to the actual target of 10 different controllers after a congruent (lower line) or an incongruent (top three lines) prime. The effect of the incongruent prime is altered by the duration between the onsets of the prime and the actual target (stimulus onset asynchrony; SOA). B: The maximal Euclidean distances from the target for movements with incongruent primes increases with the SOA. Error bars show standard deviations.

Figure 17 shows the distribution of shoulder, elbow, and wrist angles when the goal was reached. The desired joint angle constraint is met selectively for each condition (with a standard deviation of ± 45° for each constraint). Furthermore, the controller uses a broad variety of final joint angles in the unconstrained joints to reach the activated goal location, indicating that the architecture effectively exploits redundancy.

To determine whether fulfilling constraints comes at a price, we also compared the hand position accuracy and movement duration for constrained and unconstrained movements. Figure 18 shows the development of movement duration and error after various amounts of learning. Movement duration is only charted from 100,000 trials of learning because this is the first time that all controllers were able to finish all required movements. After 1,000,000 time steps, there was no difference in average hand position accuracy between constrained (M = 4.77% of the workspace size, SD = .835%) and unconstrained (M = 4.56%, SD = .791%) movements, $t(18) = .591$, $p > .05$. On the other hand, movement durations differed significantly. The movement duration is measured as the number of time steps used from the first change of the posture until the hand moves closer than 15% of the workspace size to the goal. After 1,000,000 time steps of learning, the average movement time of an unconstrained movement was 6.44 steps (SD = 1.79). Constrained movements were significantly slower (M = 16.6, SD = 6.51), $t(10.4) = 4.75$, $p < .01$.

Each hand target in the previous evaluation was approached from two different random starting postures in order to allow an analysis of start-posture dependency. For movements with the same hand target but different starting postures, we compared the final postures of the movement (d2 metric). On average, the end postures of unconstrained movements differed by 111° (SD = 56.3°). For the constrained movements, the average posture difference was 70.9°(SD = 49.2°). Constrained movements could still express a high amount of start-posture dependency because some targets could be realized by moving one joint to one extreme or the other and because not all constraints were exactly met. Both values differ significantly from zero and from each other (from zero for unconstrained movements, $t(9) = 6.22$, $p < .001$; from zero for constrained movements, $t(9) = 4.56$, $p < .001$; from each other [pairwise t test], $t(9) = 3.32$, $p < .01$).

The data confirm two important claims. First, the capability to store and process many possible arm postures for single goal coordinates enables SURE_REACH to flexibly incorporate new task-dependent constraints. For example, a goal that requires the hand to reach a certain position while maintaining a specific elbow angle can be easily pursued by our model—even if this task has never been explicitly trained. Second, the significant difference in movement times reveals that the sensory-to-motor mapping induces more efficient movements if it is activated by a larger set of acceptable goal postures (encoded in $\vec{p}_g$). SURE_REACH exploits the redundancy provided by a larger goal set, reaching the desired goal locations faster because the posture within the goal set that is closest to the starting posture is approached. These findings fit well with current behavioral data from human participants (Cruse et al., 1993; Fischer et al., 1997; Jaric et al., 1992; Soechting et al., 1995).

Benefits of Memorizing Sensorimotor Redundancy

In the last section, we demonstrated that the representation of kinematic redundancy significantly enhances the flexibility of the controller. In this section, sensorimotor redundancy is exploited to adapt to obstacles, different movement costs, and immobilized joints.

Obstacle Avoidance

The previous section shows that the end posture of a movement can be influenced by additional constraints. It illustrates that the

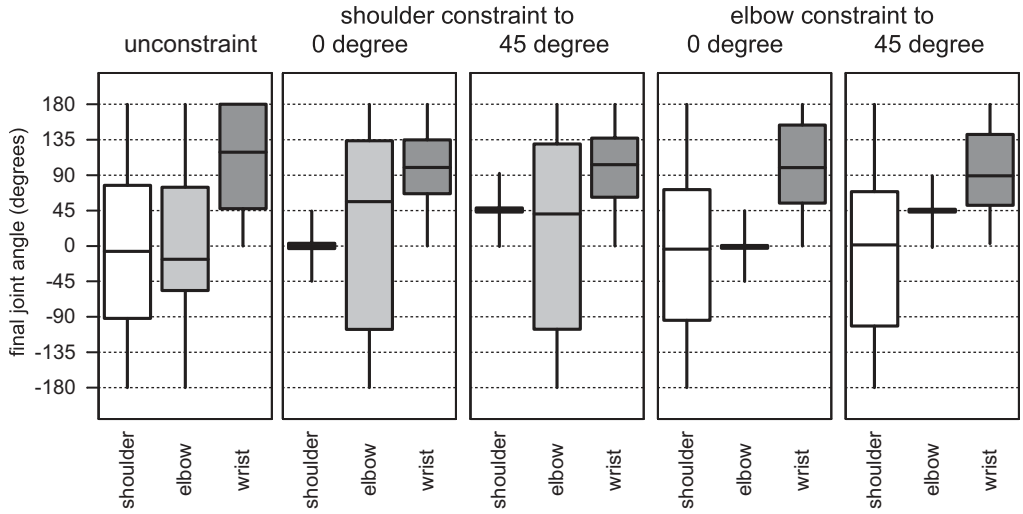

Figure 17. Box plots for the distribution of the final joint angles in five constraint conditions. The different constraints for elbow and shoulder joint are met selectively, albeit not completely accurately, because of the broad tuning of the receptive fields in joint space (90°). A box ranges from the lower to the upper quartile, the lines in the boxes indicate medians, and the whiskers show the range from the smallest to the largest observed joint angle.

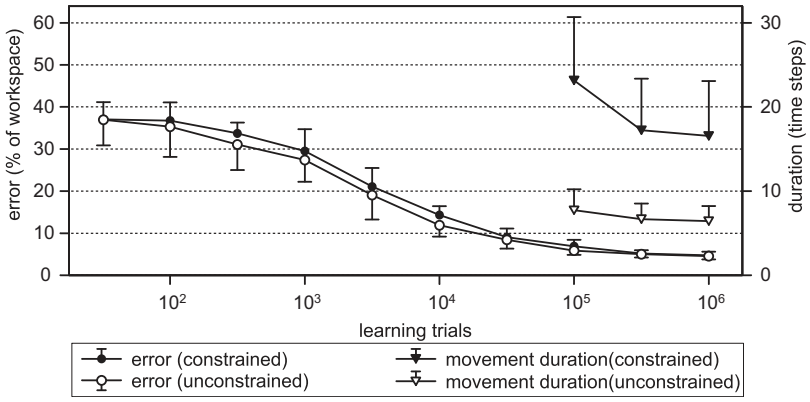

Figure 18. Although the average hand position errors of constrained and unconstrained movements do not differ significantly, unconstrained movements are much faster than constrained movements. Error bars show standard deviations.

redundancy of postures associated to a hand position can be exploited to choose from among alternative goal postures, considering additional constraints.

Another form of motor redundancy resolution lies in the trajectory generation by means of dynamic programming. If an obstacle blocks a certain area in posture space, the activity in the associated neural subspace may be inhibited consecutively. Thus, the activation diffusion by means of dynamic programming generates an alternative trajectory that circumvents the obstacle. However, obstacles are usually seen before movement onset and are represented in an extrinsic coordinate frame. In SURE_REACH, an extrinsically represented obstacle can be transformed into an obstacle representation in posture space by the posture memory (see the right-hand side of Figure 5). Because the posture memory activates all arm postures that realize certain hand positions, feeding the complete extrinsic obstacle representation into the posture memory results in a representation of all those postures for which the hand would collide with the obstacle. For now, the posture memory only activates those postures that coincide with certain hand locations so that it is only possible to avoid hand collisions but not collisions of other body parts. A more general posture memory that not only maps from certain hand locations to postures but also determines arm postures that coincide with any other point on the arm could be used to extend obstacle avoidance to the entire arm.

To evaluate the current obstacle avoidance capability, we trained 10 individual controllers for 1,000,000 time steps and tested each of them in two different tasks (see Figure 19). In each task, obstacles had to be avoided. Obstacles were defined in hand space. A hand space obstacle representation was generated by inhibiting neurons whose preferred values were within the obstacle. This inhibition was passed through to posture space by means of the posture memory, consequently inhibiting those neurons in posture space that collide with the obstacle. We chose to inhibit all neurons (set to zero activity) in the activation maps ($\vec{a}_i$) that had an activation level of at least .01, stemming from the inhibition passed through posture memory.

In the first task, the arm had to move the hand to the lowest position in the workspace (open circle in Figures 19A and 19B) from an upward pointing posture. This goal can be pursued by two different movements. The arm can be rotated either clockwise or counterclockwise. In two different settings, either the clockwise or counterclockwise movement was blocked by a square obstacle placed next to the shoulder. Figures 19A and 19B show the movement trajectories of 10 individual controllers. Each controller avoided the obstacle by rotating the shoulder joint in the unblocked direction.

In the second task, the controllers had to move the arm to a stretched posture with a shoulder angle of −225°. The initial posture was a stretched arm, however with a shoulder angle of 225°. If no obstacle was in the way, the controller rotated only the shoulder joint (see Figure 19C). In a second condition, a ceiling obstacle was introduced that would cause a collision if the hand location was too close to the ceiling. In this case, the controller bent the other joints in order to reduce the height of the extended arm and thus forced the hand to move beneath the obstacle (see Figure 19D). After the obstacle was passed, the arm restretched the joints to reach the desired end-point position. The highest hand

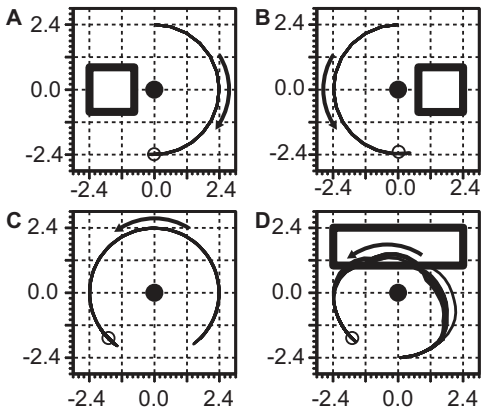

Figure 19. The hand paths of 10 movements differ significantly, if an obstacle (rectangle) is presented to the left (A) or to the right (B). All movements bypass the obstacle successfully. C: Movement of a stretched arm without any obstacles. D: The same movement if a ceiling obstacle constrains possible hand locations. The open circles indicate desired hand targets, and the solid circles indicate the position of the shoulder joint.

locations of the trajectories ($M = 1.40$, $SD = .0563$) were significantly lower when an obstacle was present compared with those with unconstrained movements ($M = 2.39$, $SD = .00626$), $t(9) = -58.7$, $p < .01$. The figure shows that all trajectories slightly moved through areas of hand space that were part of the obstacle. These collisions result from the broad tuning of the state neurons' receptive fields and could be avoided by a larger set of neurons covering the hand and posture spaces or by a further reaching inhibition, which would trigger even stronger obstacle avoidance.

To summarize, obstacle avoidance behavior can be incorporated into SURE_REACH by the inhibition of neurons. This capability is due to the fact that the inverse sensorimotor model of the motor controller implicitly stores all possible trajectories to a particular goal position and that the posture memory can be readily used to convert an obstacle representation from a hand-based to a posture-based representation. Without any obstacle-based constraints, the sensorimotor model triggers the most direct trajectory in posture space. If such a trajectory is blocked because of, for example, obstacle-originated inhibitions, supplementary links induce effective, obstacle-avoiding control. As obstacle avoidance is based on the sparsely encoded posture representation, the current model can only account for the avoidance of larger obstacles and goals that are sufficiently distant from obstacles. However, a more fine-grained neural encoding might also enable the avoidance of smaller objects and closer goal and obstacle locations.

Reduced Joint Mobility

During life, the costs of moving certain limbs might suddenly change. For example, arthralgic patients suffer severe pain from moving a specific joint and hence have to achieve certain behavioral goals while trying to move one joint or the other as little as possible. If a joint is in a cast, some motions not only might be costly but also might be suddenly impossible. Despite this impairment, patients can usually control their arm accurately and effectively with the remaining mobility. SURE_REACH can account for this flexibility, limiting the extent to which some actions are applied by reducing the impact of those actions on the activation propagation process. This adjustment might be considered a neural implementation of the adjustment of movement cost functions proposed by Rosenbaum et al. (1995). It was simulated by enhancing Equation 7 as follows:

$$\vec{a}_i^* \leftarrow v_i max\left\{\beta\left(\gamma\frac{\Sigma_{j\neq i}\vec{a}_j}{y-1} + (1-\gamma)\vec{a}_i\right), \vec{p}_g\right\}. \qquad (13)$$

The difference between Equations 7 and 13 is that in the former, all actions contribute equally to the activation propagation process, whereas in the latter, the contribution of each action is weighted according to a weighting coefficient v_i. By adjusting coefficient v_i, the extent to which certain actions and hence certain joint motions are executed during a movement can be regulated. A larger coefficient v_i means relying more on the associated action.

An Arthralgic Joint

Figure 20 shows examples of movements that result from the modified activation propagation process. Whereas Figure 20A displays a normal unconstrained movement, Figures 20B and 20C show movements in which the contribution of the actions affecting the elbow and wrist joint have been reduced to 1% of the contribution of the remaining actions. It is apparent that all joints are used in the unconstrained movement but that the motions of the elbow or wrist joint, respectively, are highly reduced in the other examples.

To determine whether this process also holds in the general case, we trained 10 controllers individually for 1,000,000 steps. Each controller had to perform movements from 16 different start postures to different goals provided in hand space. For each of the 16 start–goal pairs, one normal movement and three movements with a reduced desired contribution of the shoulder, elbow, or wrist were executed. For normal movements, all actions contributed equally to the activation propagation process. For movements in which a reduced motion of a specific joint was desired, the v_is for actions associated with that joint were set to 1% of the v_is of the remaining actions. For example, if the wrist joint should move as little as possible $v_0 = v_1 = 0.01$ and $v_2 = \ldots = v_6 = 1.00$, the contribution of each joint to a movement was operationalized as the absolute difference between its start angle and final angle. The contributions were averaged for each controller, each of the four movement conditions, and each joint. Start–goal pairs yielding movements that did not move as close as 15% of the workspace size to the goal within 160 time steps in at least one of the four conditions were removed from the evaluation (48.8%). Figure 20 shows the results. It can be seen that the contribution of each joint angle is reduced selectively for the desired joint. Pairwise t tests confirm this finding (normal arm: $M_{shoulder} = 69.1°$, $SD_{shoulder} = 49.1°$; $M_{elbow} = 66.5°$, $SD_{elbow} = 48.7°$; $M_{wrist} = 60.2°$, $SD_{wrist} = 44.5°$; shoulder: $M = 32.5°$, $SD = 28.2°$, $t(9) = 15.7$, $p < .01$; elbow: $M = 26.5°$, $SD = 25.2°$, $t(9) = 17.2$, $p < .01$; wrist: $M = 24.0°$, $SD = 22.5°$, $t(9) = 16.4$, $p < .01$).

Thereby, the final position error is only slightly larger for movements with reduced joint motions for normal movements ($M = 4.00\%$ of the workspace size, $SD = 0.427$; for constraint movements, $M = 4.67\%$, $SD = 0.443$, $t(9) = 5.216$, $p < .01$). Motion of the impaired joint is not reduced to zero because many movements require a transition of all joints in order to move to the goal. Note that the reduction of the contribution of an action to the action propagation process does not inhibit the action per se. It is only when other actions can be used to fulfill the given goal that the inhibited action will not be executed.

A Broken Arm

Sometimes it might be necessary to move the arm without the capability of relying on the complete action repertoire that was available during motor learning. This may be the case if the arm is broken and in a cast. Usually used action sequences will then be fruitless if they rely on actions that are now impossible. Experiments with human participants show that constraining one limb hardly affects the capability for accurate movements and does not require exhaustive relearning (Robertson & Miall, 1997).

To verify that the reduction of the contribution of formerly performable actions to the activation process can be used to control an arm with a joint in a cast, we trained 10 controllers individually for 1,000,000 steps to perform reaching movements with either normal mobility or with the shoulder, elbow, or wrist joint angle set to $\phi_i = 0$, regardless of the actions that were executed. The v_i

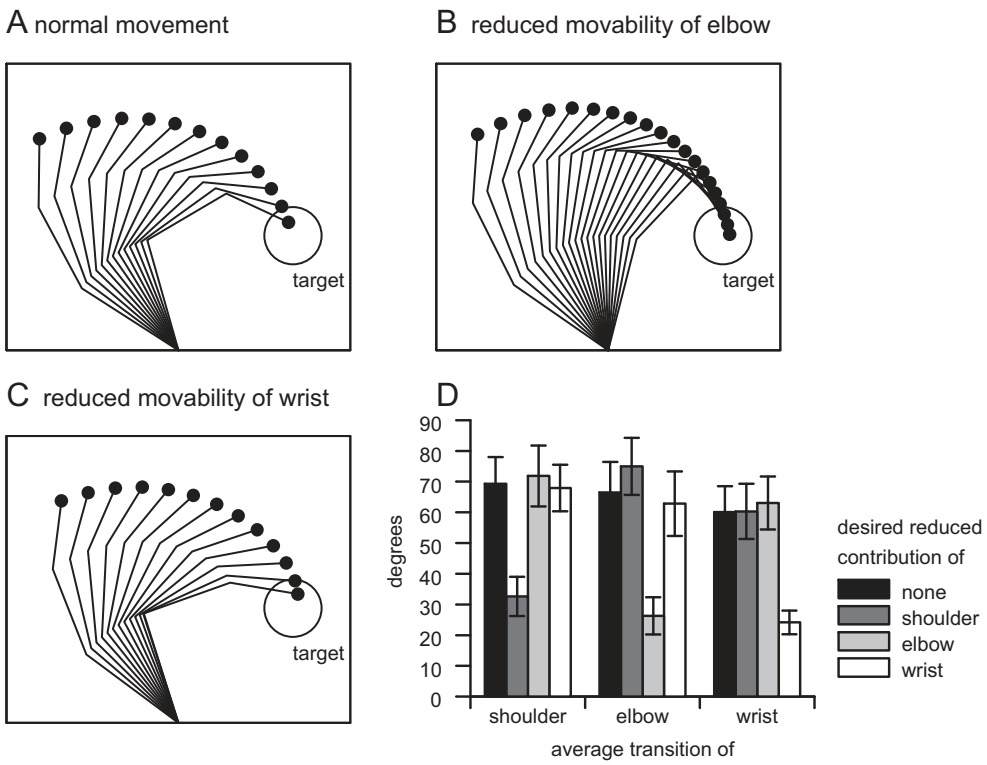

Figure 20. A: An example of an unconstrained movement. B: Movements with reduced rotation of the elbow joint. C: Movements with reduced rotation of the wrist joint. D: A systematic evaluation reveals that the contribution of each joint to a movement (operationalized as the change in joint angle between movement onset and the final position) can be selectively reduced. Error bars show standard deviations.

values for actions associated with joints in a cast were set to $v_{joint\ in\ cast} = 0$, whereas all others were set to $v_{free} = 1.0$. To compare the different conditions, we averaged and compared the end-point error of all reaching movements with random goals that could be theoretically reached in all movement conditions (on average 12.6 per controller) by pairwise t tests. The end-point error for free movements was ($M = 3.54\%$ of the workspace size, $SD = 0.659$). Movements with the shoulder or wrist in a cast were only somewhat less accurate (for shoulder: $M = 8.08\%$, $SD = 2.40$, $t(9) = 5.72$, $p < .01$; for wrist: $M = 6.70\%$, $SD = 0.861$, $t(9) = 11.22$, $p < .01$). Fixing the elbow did not affect average accuracy ($M = 3.24\%$, $SD = 0.724$), $t(9) = -1.118$, $p > .05$.

To summarize, the activation propagation process can be modified by weighting each joint movement. On the one hand, joint movements that cause pain or are otherwise costly can be replaced by movements of the other joints (as long as this is possible). On the other hand, the modification can be used to control the arm if certain joint movements are suddenly impossible. In either case, it is not necessary to relearn the sensorimotor mapping of the arm.

Summary of Results

The evaluations of our particular implementation of SURE_REACH confirmed several interesting model features. The overall reliability and accuracy of the model was high despite the rather sparse representation of postures and end-point coordinates. Model learning was stable under several different learning and parameter conditions.

As a model for motor learning and control, SURE_REACH revealed several properties. First, during human motor learning, accuracy increases and movement times and reaction times decrease. The model not only accounts for increasingly accurate movements, but also exhibits that training decreases movement preparation and movement execution times. Second, representing goals by population codes is in line not only with neurophysiological data but also with psychological findings and theories (Erlhagen & Schöner, 2002; Flash & Sejnowski, 2001). In contrast to many other models of motor learning and control, which can only process a single target posture or hand location, SURE_REACH can account for more complex target representations, such as the target activation of multiple alternative postures. Likewise, human participants and primates are able to partially prepare movements toward subsets of movement directions or distances (Bastian, Schöner, & Riehle, 2003; Bock & Arnold, 1992). Third, a priming experiment was replicated by means of the space representation and network dynamics. Fourth, it was demonstrated that representing kinematic redundancy enables the simulation of some features of the flexibility of human motor control. In humans, the final arm posture of a movement depends on the starting posture (Cruse et al., 1993; Fischer et al., 1997; Jaric et al., 1992; Soechting et al., 1995). Also in SURE_REACH, movements

to the same hand position differ, depending on the starting posture. Finally, because of the encoded sensorimotor redundancy in the sensorimotor model, SURE_REACH is able to adapt quickly to novel task constraints. The architecture is able to avoid obstacles and recruit alternative actions, given that previously optimal actions are suddenly costly or even impossible. Together, the evaluations show that the encoding of motor redundancies on many levels enables the simulation of flexible human motor behavior that cannot be accounted for by models that strive to resolve redundancy before learning.

Discussion

Our implementation of SURE_REACH has shown interesting learning capabilities and robust performance. Learning started from scratch without any prior information, except for the general modular architecture and the uniform population-encoded representation of hand and posture space. SURE_REACH extends or complements the capabilities of previous learning models. We showed that SURE_REACH can solve the redundancy problem flexibly online and can easily incorporate additional task constraints. In the following discussion, we first reevaluate SURE_REACH's adaptive solution of the redundancy problem, comparing it with the available literature. Next, we put SURE_REACH into a broader perspective, suggesting that similar architectures and representations are used in primates and humans. We support this claim with neurophysiological findings. We end the discussion with the current limitations of the model and possible extensions.

Relations to Previous Models

Having described and evaluated our model, we now pinpoint the contributions of SURE_REACH with respect to other adaptive motor control models.

FEL

SURE_REACH learns motor control bottom-up, starting with the formation of basic sensorimotor contingencies. Hence, our model addresses a different topic than does the cerebellar models of FEL (Berthier et al., 1993; Kawato et al., 1987). These models address how the cerebellum acquires a fine-tuned, fast, inverse sensorimotor model for reaching movements by learning and finally substituting cerebral motor commands. In this perspective, SURE_REACH might be a model of the cerebral source of the motor signals in FEL, or it might be complemented by FEL to smooth behavior execution with a modular architecture similar to hierarchical MOSAIC (Haruno et al., 2001; Haruno, Wolpert, & Kawato, 2003).

DIM and RMRC

SURE_REACH extends models that implement DIM and RMRC techniques in two ways. First, the temporally weighted association mechanism in the action columns and the dynamic-programming-based activation propagation process results in effective associative representations. It enables the linkage of potential goal states to temporally remote actions and initial condition. Second, and more important, the unsupervised learning method enables the representation and flexible online resolution of motor redundancy. Hence, unlike DIM and RMRC approaches, SURE_REACH is able to exert effective flexible control in redundant contexts, offering a new level of behavioral flexibility.

PB Theory

SURE_REACH is able to resolve redundancy online and under varying constraints, similar to the PB theory (Rosenbaum et al., 1993, 1995, 2001). In direct comparison to the PB theory, it adds four novel features. First, no distance measures need to be provided to enable goal-posture selection in SURE_REACH. Distance measures are learned from experience within the inverse sensorimotor model. They are sensorimotor grounded. Second, SURE_REACH enables online redundancy resolution, as does the PB theory. However, SURE_REACH can incorporate additional task constraints such as blocked paths that are due to obstacles immediately during redundancy resolution. Such task constraints can be incorporated by both multiplicative inhibition or excitation of both neural activity or activity propagation. Third, SURE_REACH is implemented in a biologically plausible neural network. Finally, SURE_REACH not only resolves redundancy in posture and trajectory selection, but also interactively executes trajectories by closed-loop control. This enables SURE_REACH to account for novel task constraints (such as the occurrence of a new goal) during movement production.

Nonetheless, the PB theory still has much to offer SURE_REACH. First, the PB theory also accounts for typical velocity profiles in reaching arm movements. By the imposition of a sine function for movement execution, realistic, bell-shaped velocity profiles are generated. They are also used in obstacle avoidance consequently generating typical, bell-shaped avoidance patterns (Rosenbaum et al., 1995). Such sinusoidal activation patterns may also be added to the SURE_REACH control process. This might be possible by modifying the gain factor g during motor activity generation (see Equation 12), similar to the force modulation proposed in Bullock, Cisek, and Grossberg (1998), to the neural time-base generator proposed in Tanaka, Tsuji, Sanguineti, and Morasso (2005), or to the PB theory approach. The addition of smoothness constraints on motor activity execution may result in more realistic movement patterns (Flash & Hogan, 1985; Todorov, 2004).

Moreover, additional challenges such as grasping (Meulenbroek et al., 2001) or tool use were not further investigated. Meulenbroek et al. (2001) implemented grasping mechanisms concurrently to arm approaching on the basis of the PB theory. An integration of this grasping mechanism into SURE_REACH with appropriate adjustments seems possible. In this respect, the addition of a population-encoded hand-surround space may enable the desired obstacle avoidance behavior for grasping. Tool use has been shown to actually result in a temporally expanded body space encoding of hand end-points (Maravita, Spence, & Driver, 2003), so a similar encoding within the SURE_REACH approach could trigger comparable behavioral patterns.

RL

The generation of sensory-to-motor mappings by dynamic programming in SURE_REACH is similar to indirect (model-based)

RL approaches (Sutton, 1990; Sutton & Barto, 1998). From the RL perspective, posture goals in SURE_REACH are similar to a particular reward scheme imposition in RL. The sensorimotor model corresponds to the model of the environment, which is necessary to enable indirect RL. The generation of the sensory-to-motor mappings based on a given posture is accomplished by synchronous dynamic programming (cf. Barto, Bradtke, & Singh, 1995). Thus, SURE_REACH learns optimal behavioral policies immediately before action execution by the application of dynamic programming. The application of dynamic programming concurrently to action execution was successfully applied in the simulation of the priming experiment shown above. Further advanced concurrent dynamic programming techniques are certainly imaginable. For example, real-time dynamic programming techniques (Barto et al., 1995) or prioritized sweeping (Moore & Atkeson, 1993) could be applied, which focus their internal value updates on the currently most important subspaces.

However, SURE_REACH differs rather strongly from the direct RL approaches for arm control discussed above (Berthier, 1996; Berthier et al., 2005). These approaches learn a behavioral policy directly without learning a model of the environment. Generally, direct RL approaches may be superior policy learners if no model is available or if learning a complete model requires too much effort. If a model can be learned and represented efficiently, though, dynamic programming and other indirect RL approaches are usually more flexible than are direct RL approaches. This is because the model can be used to adjust behavioral policies on the fly to novel task constraints, such as changing goals and goal priorities, path blockages, or trajectory constraints, without the need to execute actual actions in the environment. Although more suitable representations and extensive training may also make direct RL behaviorally more flexible, the consequently blown-up policy representations and the additionally necessary training effort can be expected to yield a mediocre learning system though most likely still not reaching the flexibility of indirect RL approaches.

Population Encodings

An early self-supervised control approach can be found in Mel (1991), who also tackled the problem of redundant robot arm control with population encodings. However, because each dimension of joint angles was encoded separately, dynamic programming was not applicable and heuristic search had to be used to avoid obstacles. Besides the posture state representation, an inverse differential kinematics representation was also used to encode joint movements toward any direction in external space posture-dependently. Such a representation might be very well added to the current SURE_REACH architecture so as to activate direction goals besides the currently possible hand location or posture goals.

Other approaches that implement motion planning in joint space and project obstacle representations into that space can be traced back to Lozano-Perez (1981, 1987). However, Lozano-Perez (1987) had to use A* search[5] to find an optimal path through the joint space (termed *configuration space*) while avoiding obstacles. Another related approach that does not require A* search uses harmonic functions to generate suitable trajectories through a state space for robot control (Connolly & Grupen, 1993).[5] In addition, the harmonic function approach allows the inhibition of problem subspaces and the emergent avoidance of obstacles, as applied, for example, for collision avoidance in the interaction of two robots (Souccar & Grupen, 1996). The dynamic fields approach for redundant arm control, introduced in Morasso et al. (1997), builds representations that are comparable to harmonic functions. Both representations have similarities with the sensory-to-motor mappings generated in SURE_REACH. However, though both resolve redundancy within their respective applications, to our knowledge, neither approach addresses the problem of learning sensorimotor-grounded distance representations in joint space (the learned sensorimotor model in SURE_REACH). Rather, both approaches assume that the nearest neighbors are connected with Euclidean distance measures.

Toussaint (2006) recently published an adaptive behavior approach that learns sensorimotor-grounded distance measures within local receptive fields and that uses these to generate movement trajectories. During the learning phase, the architecture distributes a growing neural network grid in a maze environment, randomly exploring the environment. The grid is used to represent the space, and the action-dependent connections between grid nodes represent the sensorimotor contingencies inside that space. The representation was successfully applied to generate goal-approaching behavior. However, Toussaint addressed neither the redundancy resolution problem (no activation of multiple goals) nor the representation of correlated spaces (no inverse kinematics model).

Relations to Neurophysiology

Although we did not focus on anatomical accuracy, our model is in line with current neurophysiological and neuroanatomical findings. From a macroscopic perspective, the model fits into current theories that locate associative unsupervised learning mechanisms in the cerebral cortex and that emphasize the importance of recurrent neural connections in these areas (Doya, 1999, 2000).

More important, recent neurophysiological and psychological studies suggest that population-encoded, isomorphic representations of body postures, but also peripersonal and extrapersonal surround spaces, can be found in various forms in various integrative cortical sensory, motor, and sensorimotor areas (Holmes & Spence, 2004; Maravita et al., 2003; Paninski, Shoham, Fellows, Hatsopoulos, & Donoghue, 2004; Rizzolatti, Fadiga, Fogassi, & Gallese, 1997; Schwartz, Moran, & Reina, 2004). In fact, Schwartz et al. (2004) showed that it is possible to measure distinct representations of arm movements in a monkey when an artificial mismatch is created by a visor that displays skewed actual arm movements from the monkey. Integrating the activities of redundant population encodings, neural activity in the ventral premotor cortex did not match the encodings in the primary motor cortex, representing the visually perceived arm movements and the proprioceptively perceived arm movements, respectively (Schwartz et al., 2004). This shows that the brain encodes different body spaces, whose activities depend predominantly on different sensory infor-

[5] A Euclidean distance prioritized search through the space (cf. Russel & Norvig, 1995).

mation. The identified encodings are somewhat similar to the posture space and hand space encodings in the SURE_REACH architecture, in which posture space relies on proprioceptive joint angle information and in which hand space relies on visual arm location information.

Moreover, Graziano (2006) suggested that the primary motor cortex represents the arm as posture-based, as does the posture representation in SURE_REACH. It was shown that neural activity and artificial neural stimulation of a neuron correlate most significantly with particular postures, either holding the arm in that posture or causing the arm to move to that posture, respectively. Although these cortical representations are certainly much more robust, redundant, and self-structuring, the population-encoded posture space in SURE_REACH mimics such population codes.

On a synaptic level, our model relies entirely on simple Hebbian learning, which is principally known to modulate synaptic connectivity in the brain (Abbott & Nelson, 2000; Jackson, Mavoori, & Fetz, 2006). The associative and recurrent connections were structured via Hebbian learning in an unsupervised (or self-supervised) manner. Thus, the learning mechanisms in SURE_REACH are also biologically plausible.

Shortcomings and Extensions

Despite the correlations to known neuronal representations and particular cortical brain structures, as well as to basic behavioral data in humans, SURE_REACH currently mimics only a small and highly abstract part of a biological cognitive system. Nonetheless, the results are promising and the current architecture is modular enough to enable improvements and enhancements to face current system challenges and shortcomings. In the following sections, we touch on some of the forthcoming challenges.

Representation of Internal Spaces

Although SURE_REACH realizes many aspects previously hardwired in cognitive system architectures, the architecture is still hardwired in multiple aspects. First and foremost, the neuronal populations covering posture space and hand space are uniformly distributed and hardwired. In future work on SURE_REACH, it would be desirable to add growing, self-organizing maps (Fritzke, 1995; Haykin, 1999; Martinetz, Berkovitsch, & Schulten, 1993), which are able to develop a space-covering neural population from scratch. Dependent on the imposed biases on the growth mechanism, such structures might be more robust and flexible than might be the uniformly distributed structures used in the current implementation. Self-structuring mechanisms are currently under further development, such as the development of hyper-ellipsoidal activation patterns (Butz, 2005; Butz, Lanzi, & Wilson, 2006) for accurate function approximations; vision-based predictive field representations (Olsson, 2006); or the coverage of arbitrary, maze-like environments (Toussaint, 2006). It is interesting to note that Aflalo and Graziano (2006) have shown that a simple self-organizing map can develop an artificial motor cortex array that resembles many properties observed in monkeys, such as maps of hand locations, simply by sampling the behavior repertoire typical for a monkey.

For the development of population encodings that cover higher dimensional spaces, it seems necessary that the space coverage will be more modular with receptive fields of varying sizes that cover overlapping subspaces in the high-dimensional space. Recent advances in manifold learning show that higher dimensional spaces can indeed be efficiently represented by observing local neighborhood structures that derive lower dimensional, neighborhood-preserving structures from that data (Roweis & Saul, 2000; Tenenbaum, de Silva, & Langford, 2000).

Another approach to higher dimensionality spaces is that of modular encodings. Various neurophysiological findings suggest that population encodings are highly modular, encoding, for example, hand and finger postures independently of arm postures (Graziano, 2006; Rizzolatti et al., 1988). A similar approach is taken in robotics path planning in artificial intelligence, in which road maps can partition high-dimensional spaces into multiple, lower dimensional spaces (Russel & Norvig, 1995). Thus, a modular encoding of higher dimensional spaces to overcome the curse of dimensionality appears plausible.

Another challenge for internal spatial representations is the ability to reach locations in space that have not been reached before. With the current approach and encoding, locations outside of the population-encoded space cannot be reached. Action-dependent directional encodings may alleviate this problem, as already proposed in Mel (1991). That is, besides locations, directions may be encoded and linked to action codes that lead in a specific direction. Such a representation may compete with the currently used location-dependent encoding. Future research is necessary to investigate the plausibility of such an approach and the possible interaction between the two encodings.

Besides the challenge of reaching locations outside of the population-encoded space, end-point accuracy of SURE_REACH should be further improved. The most straightforward improvement would be to learn more finely grained spatial encodings. However, other less computationally intensive methods are imaginable. Somewhat similar to the PB theory approach, which samples end postures that surround an approximately suitable stored posture (Rosenbaum et al., 2001), SURE_REACH may be endowed with a local adjustment mechanism. Local target adjustments could be applied, as in the PB theory, but direction-based adjustments are also imaginable with the direction-based encodings suggested above.

Another concern is that of obstacle avoidance with parts of the arm other than the hand. Currently, only the hand location is processed in posture memory so that obstacle avoidance is possible only for the hand. Additional associative maps may be learned, which could also relate elbow and wrist locations in space with corresponding postures. Extensions to linear inhibitions that extend in coordinate space to the next joint may be possible to avoid not only joint collision but complete arm collision.

Action Space Encoding

The action space is currently hardwired and discrete. This representational choice was made because of spatial and computational time constraints. It is clearly imaginable that the action space representation may also be structured with population codes, as has been observed in biological neural codes (Flash & Sejnowski, 2001; Georgopoulus, 1995; Georgopoulos, Caminiti, Kalaska, & Massey, 1983; Poggio & Bizzi, 2004). In this case, the recurrent connections between different action columns would have to be more convoluted, conditioned on different associated action sub-

spaces. It might be interesting to enable the activation of more elaborate action sequences, instead of simple, stepwise actions, as is suggested within the concept of motor synergies (Bernstein, 1967; Poggio & Bizzi, 2004). A visionary approach in this direction was taken in Kuperstein (1991), who interleaved an action target map for which each target position encoded the saccadic eye movement necessary to move to that target. Such encodings suggest a correspondence to cortical columns that represent mixtures of experts confined to specific receptive fields (Hubel, 1993; Mountcastle, 1978). Action-mediated, competitive neural structures may enable a similar constraint-recurrent neural activity propagation with much more intricate, action-dependent activation patterns.

System Plasticity

Another challenge is the improvement of the plasticity of the learning algorithms currently implemented in SURE_REACH. The usage of a simple Hebbian learning mechanism showed that no complex learning mechanism is necessary to solve the targeted problem. It should be obvious, though, that more elaborate learning mechanisms can replace the current one. Recent observations have suggested that the plasticity of actual motor behavior adaptation works along multiple timescales (Krakauer & Shadmehr, 2006). These adaptation processes enable (a) temporary, fast adaptation to temporally changed dynamics (e.g., carrying a heavy object); (b) slower adaptation to temporally extended changes (e.g., muscle fatigue); (c) even slower adaptation to general bodily constraints and strength (e.g., general muscular capabilities or body flexibility); and (d) very slow adaptation to general kinematic and dynamic body constraints (e.g., body size). How this is realized in the brain remains an open discussion. In a population-encoding approach, overlapping neurons could have different learning rates to simulate the different speeds of adaptation. Further improvements and advancements in the available learning algorithms and representations are necessary to account for these capabilities.

Improved Exploration

Besides such architectural and learning constraints, there are also multiple imaginable behavioral extensions. Currently, behavior during motor learning in the SURE_REACH implementation is completely randomized and not at all goal-driven. However, it is known that motor behavior becomes goal-oriented very early in infancy, even if it is not yet well controlled (von Hofsten, 2004). Information seeking exploration strategies especially may actually speed up model learning (Butz, 2002). Also in SURE_REACH, motor activity might be goal-oriented earlier during learning. To trigger goals in this way, though, the associative networks may need to contain many more coarsely grained neurons to enable action activations despite a highly incomplete posture memory and inverse sensorimotor model.

Body-Related Grounding and Improvements

In its current implementation, SURE_REACH interacts with a body that has hardly any morphological features. It becomes increasingly clear, though, that biological bodies exhibit a multitude of morphological or embodied intelligence (Pfeifer & Gomez, 2004). These bodily constraints enable a safer exploration of the environment and more energy efficient behavior execution by the exploitation of inherent bodily dynamics. These constraints must also shape the development of internal motor control representations.

Although the spatial distributions of the population codes in SURE_REACH need to be adjusted to more elaborate bodies, and consequently more elaborate sensory information, the architecture seems ready to integrate such enhancements. Besides the more complex body constraints, the morphology might also enable faster learning of simple movement patterns as a result of self-stabilization effects that cause the natural, morphologically grounded selection of preferred postures and movement executions. It seems an interesting research challenge to investigate how such beneficial morphological constraints can shape internally developing posture spaces, hand spaces, or inverse kinematics and sensorimotor models.

Integration of Multiple Information Sources

Currently, the internal spaces of SURE_REACH were activated by simple activation functions that assumed a uniform distribution of receptive fields, providing exact sensory feedback of limb posture and hand positions. Moreover, the motor controller relied on immediately available sensory feedback to control the arm effectively. In the real world, though, these information sources are often highly noisy and delayed in time. To be able to control a more complex body in real time, future SURE_REACH implementations will need to integrate (a) sensory information from multiple modalities and also (b) emulated forward model feedback information.

Multi-modal sensory integration. It becomes increasingly clear that internal body spaces gather information from multiple sources, integrating the information in various, internal body spaces (Holmes & Spence, 2004; Maravita et al., 2003). Essentially, such internal representations are estimates of environmental, bodily, and behaviorally relevant properties. The integration of multiple sensory information sources into internal body space representations is certainly an interesting and highly important challenge.

Psychological research suggests that the brain integrates multiple sources of information, dependent on the significance and current reliability of each source. One approach to realize such sensory integration processes is Bayesian information processing with population codes (Knill & Pouget, 2004; Deneve & Pouget, 2004). In Bayesian information processing, a priori knowledge is combined with sensory information accounting for information reliability. For example, it was shown that human participants tend to rely more on knowledge of generally likely finger positions or perturbations than on visual feedback, if the visual feedback is blurry or inaccurate (Körding, Ku, & Wolpert, 2004; Körding & Wolpert, 2004).

SURE_REACH is ready for such Bayesian information processing mechanisms. The current arm posture is represented by a number of different neurons forming a population code. In situations of uncertainty, many neurons may be activated, forming a broader, more blurry representation of the current posture, whereas only few neurons may be active if the posture is well known to the

controller. Because of the associative learning scheme, frequently observed sensorimotor contingencies will be represented by stronger synaptic connections than will others. Hence, if the arm state is well defined, a very narrow set of sensorimotor contingencies is included in the motor command generation process, resulting in precise movements. On the other hand, if the arm state is uncertain and the current posture is consequently represented by many neurons, many sensorimotor contingencies compete for their contribution to the motor commands. Neurons that represent frequently observed—and thus very likely—arm states will be more active and will contribute more strongly to movement execution. Thus, a priori knowledge will be encoded by the strength of the synaptic connections and their interaction with neural activity.

Forward model incorporation. Although SURE_REACH learns sensorimotor contingencies, its learned sensorimotor model has hitherto been used only as an inverse sensorimotor model for the motor controller. The learned sensorimotor contingencies, however, can generally be used bidirectionally and thus also as a forward model. Such forward models may be used essentially as an additional source for state estimations, as is done in Kalman filtering approaches (Haykin, 2002). Moreover, the forward model may be used during fast action execution as a replacement or as compensation for delays in actual sensory feedback in order to avoid behavioral instabilities, which are sometimes otherwise unpreventable (Mehta & Schaal, 2002). An efferent copy of recent motor activity could be propagated through the forward model and may then be used to compensate for sensory delays during closed-loop control (Flash & Sejnowski, 2001; Todorov, 2004). In this way, the sensorimotor model in SURE_REACH could be used not only inversely for the generation of motor commands, but also as an emulator to enhance sensory processing by the generation of sensory predictions, which can be used to substitute delayed sensory information, and, consequently, to improve and stabilize motor control during action execution (Cruse, 2003; Grush, 2004; Kawato, 1999; Wolpert & Ghahramani, 2000).

Summary and Conclusions

The biologically plausible SURE_REACH architecture overcomes deficiencies of previous learning models of motor behavior. Previous error-based methods require the resolution of redundancy before learning and are consequently inflexible when additional task constraints arise. Previous unsupervised learning models have been applied mainly for the resolution of inverse kinematics with limited, often hard-coded capabilities of trajectory generation. SURE_REACH is an unsupervised learning architecture that is capable of controlling a redundant plant. The current implementation of SURE_REACH confirms robust learning as well as flexible adaptation to novel contexts and task constraints. Reaching goal locations is learned quickly, accurately, and reliably, relying only on learning signals that are self-generated and sensorimotor grounded. Various behavioral findings were replicated, including improvements in reaching accuracy and efficiency during learning, priming effects, and the start-posture dependency of the movement outcome. Moreover, the model accounts for the remarkable flexibility that humans exhibit when confronted with novel task constraints, such as obstacles or impaired joints.

SURE_REACH is able to solve the redundancy problem for end-posture redundancy, as well as for trajectory redundancy, immediately before action onset, thus keeping movement control flexible. This flexibility enables the immediate adaptation to novel task-dependent constraints. The redundant encodings in posture memory and the sensorimotor model enable the generation of motor commands that avoid obstacles or account for additional end-posture constraints, which had not been encountered during learning. Thus, SURE_REACH proposes an expandable theoretical model for understanding motor learning and flexible, task-dependent motor behavior.

Future research may tackle the control of more elaborate bodies by SURE_REACH and the consequently necessary development of other, advanced spatial and bodily representations. Also in the current arm model of SURE_REACH, modified and additional sensory, motor, and sensorimotor representations may further improve motor control accuracy and flexibility. Moreover, the currently involved learning mechanisms in SURE_REACH may be enhanced or modified to tackle other challenges, such as noisy or delayed sensory feedback or changing arm properties. Also, action execution in SURE_REACH could be controlled or modified further in order to yield more plausible velocity profiles for reaching movements. Finally, a motivational module may be designed to trigger positive and negative rewards internally, causing either the invocation of additional constraints, such as the simulated neural inhibitions in the case of an arthralgic joint or a broken arm, or the internal activation of hand or posture goals.

With these enhancements, systems based on SURE_REACH will become progressively independent, developmental, cognitive architectures that autonomously start interacting with their body and environment. Ultimately, the system enhancements may lead to the creation of self-sustaining, autonomous, artificial cognitive systems. Although the design of such systems appears to be an interesting research challenge in its own right, it can be expected that this research endeavor will also reveal many more useful insights into the functionality of biological motor control and cognition in general. We hope that SURE_REACH provides an inspiring step in the right direction.

References

Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity: Taming the beast. *Nature Neuroscience, 3,* 1178–1183.

Aflalo, T. N., & Graziano, M. S. A. (2006). Possible origins of the complex topographic organization of motor cortex: Reduction of a multidimensional space onto a two-dimensional array. *Journal of Neuroscience, 26,* 6288–6297.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review, 11,* 11–73.

Baraduc, P., Guigon, E., & Burnod, Y. (1999). Where does the population vector of motor cortical cells point during reaching movements? In M. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems* (Vol. 11, pp. 83–89). Cambridge, MA: MIT Press.

Baraduc, P., Guigon, E., & Burnod, Y. (2001). Recoding arm position to learn visuomotor transformation. *Cerebral Cortex, 11,* 906–917.

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence, 72,* 81–138.

Barto, A. G., Fagg, A. H., Sitkoff, N., & Houk, J. C. (1999). A cerebellar model of timing and prediction in the control of reaching. *Neural Computation, 11,* 565–594.

Bastian, A., Schöner, G., & Riehle, A. (2003). Preshaping and continuous evolution of motor cortical representations during movement preparation. *European Journal of Neuroscience, 18,* 2047–2058.

Bellman, R. (1957). *Dynamic programming.* Princeton, NJ: Princeton University Press.

Bernstein, N. A. (1967). *The co-ordination and regulation of movements.* Oxford, England: Pergamon Press.

Berthier, N. E. (1996). Learning to reach: A mathematical model. *Developmental Psychology, 32,* 811–823.

Berthier, N. E., Rosenstein, M. T., & Barto, A. G. (2005). Approximate optimal control as a model for motor learning. *Psychological Review, 112,* 329–346.

Berthier, N. E., Singh, S. P., Barto, A. G., & Houk, J. C. (1992). A cortico-cerebellar model that learns to generate distributed motor commands to control a kinematic arm. *Advances in Neural Information Processing Systems, 4,* 611–618.

Berthier, N. E., Singh, S. P., Barto, A. G., & Houk, J. C. (1993). Distributed representation of limb motor programs in arrays of adjustable pattern generators. *Journal of Cognitive Neuroscience, 5,* 56–78.

Bock, O., & Arnold, K. (1992). Motor control prior to movement onset: Preparatory mechanisms for pointing at visual targets. *Experimental Brain Research, 90,* 209–216.

Bullock, D., Cisek, P., & Grossberg, S. (1998). Cortical networks for control of voluntary arm movements under variable force conditions. *Cerebral Cortex, 8,* 48–62.

Bullock, D., Grossberg, S., & Guenther, F. H. (1993). A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. *Journal of Cognitive Neuroscience, 5,* 408–435.

Butz, M. V. (2002). Biasing exploration in an anticipatory learning classifier system. In P. L. Lanzi, W. Stolzmann, & S. W. Wilson (Eds.), *Advances in learning classifier systems: Fourth International Workshop, 2001* (pp. 3–22). Berlin, Germany: Springer-Verlag.

Butz, M. V. (2005). Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. *GECCO 2005: Genetic and Evolutionary Computation Conference,* 1835–1842.

Butz, M. V., Lanzi, P. L., & Wilson, S. W. (2006). Hyper-ellipsoidal conditions in XCS: Rotation, linear approximation, and solution structure. *GECCO 2006: Genetic and Evolutionary Computation Conference,* 1457–1464.

Connolly, C. I., & Grupen, R. A. (1993). The applications of harmonic functions to robotics. *Journal of Robotic Systems, 10,* 931–946.

Craig, J. J. (2005). *Introduction to robotics: Mechanics and control* (3rd ed.). Upper Saddle River, NJ: Pearson Education.

Cruse, H. (2003). The evolution of cognition—A hypothesis. *Cognitve Science, 27,* 135–155.

Cruse, H., Brüwer, M., & Dean, J. (1993). Control of three- and four-joint arm movement: Strategies for a manipulator with redundant degrees of freedom. *Journal of Motor Behavior, 25,* 131–139.

Cruse, H., & Steinkühler, U. (1993). Solution of the direct and inverse kinematic problems by a common algorithm based on the mean of multiple computations. *Biological Cybernetics, 69,* 341–351.

Cruse, H., Steinkühler, U., & Burkamp, C. (1998). MMC—A recurrent neural network which can be used as manipulable body model. *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior,* 381–389.

Dehaene, S., Naccache, L., Le Clec'H, G., Koechlin, E., Mueller, M., Dehaene-Lambertz, G., et al. (1998, October 8). Imaging unconscious semantic priming. *Nature, 395,* 597–600.

Deneve, S., & Pouget, A. (2004). Bayesian multisensory integration and cross-modal spatial links. *Journal of Physiology—Paris, 98,* 249–258.

Doya, K. (1999). What are the computations of the cerebellum, the basal ganglia, and the cerebral cortex? *Neural Networks, 12*(7–8), 961–974.

Doya, K. (2000). Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current Opinion in Neurobiology, 10,* 732–739.

D'Souza, A., Vijayakumar, S., & Schaal, S. (2001). Learning inverse kinematics. *IEEE International Conference on Intelligent Robots and Systems (IROS 2001), 1,* 298–303.

Elsner, B., & Hommel, B. (2001). Effect anticipation and action control. *Journal of Experimental Psychology: Human Perception and Performance, 27,* 229–240.

Engelbrecht, S. E. (2001). Minimum principles in motor control. *Journal of Mathematical Psychology, 45,* 497–542.

Erlhagen, W., & Schöner, G. (2002). Dynamic field theory of movement preparation. *Psychological Review, 109,* 545–572.

Fischer, M. H., Rosenbau, D. A., & Vaughan, J. (1997). Speed and sequential effects in reaching. *Journal of Experimental Psychology: Human Perception and Performance, 23,* 404–428.

Flament, D., Shapiro, M. B., Kempf, T., & Corcos, D. M. (1999). Time course and temporal order of changes in movement kinematics during learning of fast and accurate elbow flexions. *Experimental Brain Research, 129,* 441–450.

Flash, T., & Hogan, N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience, 5,* 1688–1703.

Flash, T., & Sejnowski, T. J. (2001). Computational approaches to motor control. *Current Opinion in Neurobiology, 11,* 655–662.

Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in neural information processing systems 7* (pp. 625–632). Cambridge, MA: MIT Press.

Georgopoulus, A. P. (1995). Current issues in directional motor control. *Trends in Neuroscience, 18,* 506–510.

Georgopoulos, A. P., Caminiti, R., Kalaska, J. F., & Massey, J. T. (1983). Spatial coding of movement: A hypothesis concerning the coding of movement direction by motor cortical populations. *Experimental Brain Research, 49,* 327–336.

Gottlieb, G. L., Corcos, D. M., Jaric, S., & Agarwal, G. C. (1988). Practice improves even the simplest movements. *Experimental Brain Research, 73,* 436–440.

Graziano, M. (2006). The organization of behavioral repertoire in motor cortex. *Annual Review Neuroscience, 29,* 105–134.

Grush, R. (2004). The emulation theory of representation: Motor control, imagery, and perception. *Behavioral and Brain Sciences, 27,* 377–396.

Harris, C. M., & Wolpert, D. M. (1998, August 20). Signal-dependent noise determines motor planning. *Nature, 394,* 780–784.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Computation, 13,* 2201–2220.

Haruno, M., Wolpert, D. M., & Kawato, M. (2003). Hierarchical MOSAIC for movement generation. In T. Ono, G. Matsumoto, R. R. Llinas, A. Berthoz, R. Norgren, H. Nishijo, & R. Tamura (Eds.), *Excepta Medica International Congress Series* (Vol. 1250, pp. 575–590). Amsterdam: Elsevier.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Haykin, S. (2002). *Adaptive filter theory* (4th ed.). Upper Saddle River, NJ: Prentice Hall.

Hebb, D. O. (1949). *The organization of behavior.* New York: Wiley.

Herbart, J. F. (1825). *Psychologie als Wissenschaft neu gegründet auf Erfahrung, Metaphysik und Mathematik. Zweiter, analytischer Teil* [Psychology as a newly founded science based on research, metaphysics, and mathematics. Second, analytical part]. Königsberg, Germany: August Wilhem Unzer.

Herbort, O. (2005). *A computational model of hierarchical anticipatory control for reaching.* Unpublished Diplomarbeit thesis, University of Würzburg.

Herbort, O., Butz, M. V., & Hoffmann, J. (2005a). Towards an adaptive hierarchical anticipatory behavioral control system. In C. Castelfranchi, C. Balkenius, M. V. Butz, & A. Ortony (Eds.), *From reactive to*

anticipatory cognitive embodied systems: Papers from the AAAI fall symposium (pp. 83–90). Menlo Park, CA: AAAI Press.

Herbort, O., Butz, M. V., & Hoffmann, J. (2005b). Towards the advantages of hierarchical anticipatory behavioral control. In K. Opwis & I.-K. Penner (Eds.), *Proceedings of the KogWis05. The German Cognitive Science Conference* (pp. 77–82). Basel, Germany: Schwabe.

Hoffmann, J. (1993). *Vorhersage und Erkenntnis: Die Funktion von Antizipationen in der menschlichen Verhaltenssteuerung und Wahrnehmung.* [Anticipation and cognition: The function of anticipations in human behavioral control and perception.]. Göttingen, Germany: Hogrefe.

Hoffmann, J., Stöcker, C., & Kunde, W. (2004). Anticipatory control of actions. *International Journal of Sport and Exercise Psychology, 2,* 346–361.

Holmes, N. P., & Spence, C. (2004). The body schema and multisensory representation(s) of peripersonal space. *Cognitive Processing, 5,* 94–105.

Hubel, D. H. (1993). Evolution of ideas on the primary visual cortex, 1955–1978: A biased historical account. In J. Lindsten (Ed.), *Nobel lectures in physiology or medicine, 1981–1990* (pp. 24–56). Singapore: World Scientific Publishing.

Ito, M., Noda, K., Hoshino, Y., & Tani, J. (2006). Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks, 19,* 323–337.

Jackson, A., Mavoori, J., & Fetz, E. E. (2006, November 2). Long-term motor cortex plasticity induced by an electronic neural implant. *Nature, 444,* 56–60.

James, W. (1890). *The principles of psychology.* New York: Holt.

Jaric, S., Corcos, D. M., & Latash, M. L. (1992). Effects of practice on final position reproduction. *Experimental Brain Research, 91,* 129–134.

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

Körding, K. P., Ku, S. P., & Wolpert, D. M. (2004). Bayesian integration in force estimation. *Journal of Neurophysiology, 92,* 3161–3165.

Körding, K. P., & Wolpert, D. M. (2004, January 15). Bayesian integration in sensorimotor learning. *Nature, 427,* 244–247.

Karniel, A., & Inbar, G. F. (1997). A model for learning human reaching movements. *Biological Cybernetics, 77,* 173–183.

Kawato, M. (1990). Feedback-error-learning neural network for supervised learning. In R. Eckmiller (Ed.), *Advanced neural computers* (pp. 365–372). Amsterdam: North-Holland.

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

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

Kawato, M., & Gomi, H. (1992). A computational model of four regions of the cerebellum based on feedback-error learning. *Biological Cybernetics, 68,* 95–103.

Knill, D. C., & Pouget, A. (2004). The Bayesian brain: The role of uncertainty in neural coding and computation. *Trends in Neurosciences, 27,* 712–719.

Konczak, J., Borutta, M., & Dichgans, J. (1997). The development of goal-directed reaching in infants. II. Learning to produce task-adequate patterns of joint torque. *Experimental Brain Research, 113,* 465–474.

Konczak, J., & Dichgans, J. (1997). The development toward stereotypic arm kinematics during reaching in the first 3 years of life. *Experimental Brain Research, 117,* 346–354.

Kositsky, M., & Barto, A. G. (2002). The emergence of movement units through learning with noisy efferent signals and delayed sensory feedback. *Neurocomputing, 44–*46, 889–895.

Krakauer, J. W., & Shadmehr, R. (2006). Consolidation of motor memory. *Trends in Neurosciences, 29,* 58–64.

Kunde, W., Kiesel, A., & Hoffmann, J. (2003). Conscious control over the content of unconscious cognition. *Cognition, 88,* 223–242.

Kunde, W., Koch, I., & Hoffmann, J. (2004). Anticipated action effects affect the selection, initiation, and execution of actions. *The Quarterly Journal of Experimental Psychology: Human Experimental Psychology, 57*(A), 87–106.

Kuperstein, M. (1988, March 11). Neural model of adaptive hand-eye coordination for single postures. *Science, 239,* 1308–1311.

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

Latash, M. L., Scholz, J. P., & Schöner, G. (2002). Motor control strategies revealed in the structure of motor variability. *Exercise and Sport Sciences Reviews, 30,* 26–31.

Lavrysen, A., Helsen, W. F., Tremblay, L., Elliott, D., Adam, J. J., Feys, P., et al. (2003). The control of sequential aiming movements: The influence of practice and manual asymmetries on the one-target advantage. *Cortex, 39,* 307–325.

Lozano-Perez, T. (1981). Automatic planing of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics, 11,* 681–698.

Lozano-Perez, T. (1987). A simple motion planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation, RA-3,* 224–238.

Ludwig, D. A. (1982). EMG changes during the acquisition of a motor skill. *American Journal of Physical Medicine, 61*(5), 229–243.

Maravita, A., Spence, C., & Driver, J. (2003). Multisensory integration and the body schema: Close to hand and within reach. *Current Biology, 13,* 531–539.

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

Mehta, B., & Schaal, S. (2002). Forward models in visuomotor control. *Journal of Neurophysiology, 88,* 942–953.

Mel, B. W. (1991). A connectionist model may shed light on neural mechanisms for visually guided reaching. *Journal of Cognitive Neuroscience, 3,* 273–292.

Meulenbroek, R. G. J., Rosenbaum, D. A., Jansen, C., Vaughan, J., & Vogt, S. (2001). Multijoint grasping movements. Simulated and observed effects of object location, object size, and initial aperture. *Experimental Brain Research, 138,* 219–234.

Moore, A. W., & Atkeson, C. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning, 13,* 103–130.

Morasso, P., Sanguineti, V., & Spada, G. (1997). A computational theory of targeting movements based on force fields and topology representing networks. *Neurocomputing, 15,* 411–434.

Mountcastle, V. B. (1978). An organizing principle for cerebral function: The unit model and the distributed system. In G. M. Edelman & V. V. Mountcastle (Eds.), *The mindful brain* (pp. 7–50). Cambridge, MA: MIT Press.

Ognibene, D., Rega, A., & Baldassarre, G. (2006). A model of reaching integrating continuous reinforcement learning, accumulator models, and direct inverse modeling. *From Animals to Animats 9: Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior (SAB-2006),* 381–393.

Olsson, L. A. (2006). *Information self-structuring for developmental robotics: Organization, adaptation, and integration.* Unpublished doctoral dissertation, School of Computer Science, Faculty of Engineering and Information Sciences, University of Hertfordshire.

O'Regan, J. K., & Noë, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences, 24,* 939–1031.

Paninski, L., Shoham, S., Fellows, M. R., Hatsopoulos, N. G., & Donoghue, J. P. (2004). Superlinear population encoding of dynamic hand trajectory in primary motor cortex. *Journal of Neuroscience, 24,* 8551–8561.

Pfeifer, R., & Gomez, G. (2004). Interacting with the real world: Design principles for intelligent systems. *Artificial Life and Robotics, 9,* 1–6.

Poggio, T., & Bizzi, E. (2004, October 14). Generalization in vision and motor control. *Nature, 431,* 768–774.

Prinz, W. (1997). Perception and action planning. *European Journal of Cognitive Psychology, 9,* 129–154.

Rizzolatti, G., Camarda, R., Fogassi, L., Gentilucci, M., Luppino, G., & Matelli, M. (1988). Functional organization of inferior area 6 in the macaque monkey. *Experimental Brain Research, 71,* 491–507.

Rizzolatti, G., Fadiga, L., Fogassi, L., & Gallese, V. (1997, July 11). Enhanced: The space around us. *Science, 277,* 190–191.

Robertson, E. M., & Miall, R. C. (1997). Multi-joint limbs permit a flexible response to unpredictable events. *Experimental Brain Research, 117,* 148–152.

Rosenbaum, D. A., Engelbrecht, S. E., Bushe, M. M., & Loukopoulos, L. D. (1993). Knowledge model for selecting and producing reaching movements. *Journal of Motor Behavior, 25,* 217–227.

Rosenbaum, D. A., Loukopoulos, L. D., Meulenbroek, R. G. J., Vaughan, J., & Engelbrecht, S. E. (1995). Planning reaches by evaluating stored postures. *Psychological Review, 102,* 28–67.

Rosenbaum, D. A., Meulenbroek, R. G. J., Vaughan, J., & Jansen, C. (2001). Posture-based motion planning: Applications to grasping. *Psychological Review, 108,* 709–734.

Roweis, S., & Saul, L. (2000, December 22). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290,* 2323–2326.

Russel, S., & Norvig, P. (1995). *Artificial intelligence—A modern approach.* London: Prentice Hall.

Schmidt, T. (2002). The finger in flight: Real-time motor control by visually masked color stimuli. *Psychological Science, 13,* 112–118.

Schwartz, A. B., Moran, D. W., & Reina, G. A. (2004, January 16). Differential representation of perception and action in the frontal cortex. *Science, 303,* 380–383.

Schweighofer, N., Arbib, M. A., & Kawato, M. (1998). Role of the cerebellum in reaching movements in humans. I. Distributed inverse dynamics control. *European Journal of Neuroscience, 10,* 86–94.

Schweighofer, N., Spoelstra, J., Arbib, M. A., & Kawato, M. (1998). Role of the cerebellum in reaching movements in humans. II. A neural model of the intermediate cerebellum. *European Journal of Neuroscience, 10,* 95–105.

Soechting, J. F., Buneo, C. A., Herrmann, U., & Flanders, M. (1995). Moving effortlessly in three dimensions: Does Donders' law apply to arm movement? *Journal of Neuroscience, 15,* 6271–6280.

Souccar, K., & Grupen, R. A. (1996). Distributed motion control for multiple robotic manipulators. *Proceedings of the 1996 IEEE Conference on Robotics and Automation,* 3434–3439.

Stringer, S. M., Rolls, E. T., Trappenberg, T. P., & de Araujo, I. E. T. (2003). Self-organizing continuous attractor networks and motor function. *Neural Networks, 16,* 161–182.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning,* 216–224.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press.

Tanaka, Y., Tsuji, T., Sanguineti, V., & Morasso, P. G. (2005). Biomimetic trajectory generation using a neural time-base generator. *Journal of Robotic Systems, 22,* 625–637.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000, December 22). A global geometric framework for nonlinear dimensionality reduction. *Science, 290,* 2319–2323.

Todorov, E. (2004). Optimality principles in sensorimotor control. *Nature Reviews Neuroscience, 7,* 907–915.

Todorov, E., & Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature Neuroscience, 5,* 1226–1235.

Toussaint, M. (2006). A sensorimotor map: Modulating lateral interactions for anticipation and planning. *Neural Computation, 18,* 1132–1155.

Tversky, A. (1972). Elimination by aspects: A theory of choice. *Psychological Review, 79,* 281–299.

Vaughan, J., Rosenbaum, D. A., & Meulenbroek, R. G. J. (2006, June 3). Modeling reaching and manipulating in 2- and 3-D workspaces: The posture-based model. *Proceedings of the Fifth International Conference on Development and Learning* [CD-ROM]. Bloomington, IN.

von Hofsten, C. (2003). On the development of perception and action. In J. Valsiner & K. J. Connolly (Eds.), *Handbook of developmental psychology* (pp. 114–140). London: Sage.

von Hofsten, C. (2004). An action perspective on motor development. *Trends in Cognitive Science, 8,* 266–272.

Vorberg, D., Mattler, U., Heinecke, A., Schmidt, T., & Schwarzbach, J. (2003). Different time courses for visual perception and action priming. *Proceedings of the National Academy of Sciences, USA, 100,* 6275–6280.

Weigelt, M., Kunde, W., & Prinz, W. (2006). End-state comfort in bimanual object manipulation. *Experimental Psychology, 53,* 143–148.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems, 10,* 47–53.

Wolpert, D. M., & Ghahramani, Z. (2000). Computational principles of movement neuroscience. *Nature Neuroscience, 3,* 1212–1217.

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks, 11,* 1317–1329.

(*Appendix follows*)

Appendix

Parameter Specifications

The following parameters were used to evaluate the model, if not mentioned otherwise in the text: $\theta = 0.10$; $p = 0.10$. In the motor controller, δ decayed exponentially from 0.1 in the first time step to 0.01 in the 1,000,000th time step. The learning rate in posture memory was set to $\varepsilon = 0.001$. For the generation of the sensory-to-motor mappings, the following factors were used: $\gamma = .434$, $\beta = .172$.

In the first obstacle avoidance task, neurons with preferred values within the rectangle with top left corner at $(-2.4, 0.8)$ and bottom right corner at $(-0.8, -0.8)$ were considered part of the left obstacle and activations were set to 1.0. The corners of the rectangle representing the right obstacle were (0.8, 0.8) and bottom right corner at $(2.4, -0.8)$. In the second task, state neurons with preferred values that had a y component of at least 1.0 were considered part of the ceiling obstacle.

The model is implemented in Java 1.6. The experiments were run on a standard office PC (Pentium 4 HT, 3.2 GHz). Training and evaluating a single controller took approximately 25 min.

Received September 15, 2006
Revision received May 10, 2007
Accepted May 11, 2007 ■