Fully Self-Supervised Learning of an Arm Model

Martin V. Butz and **Armin Gufler** and **Konstantin Schmid** and **Fabian Schrodt**
Cognitive Modeling, Department of Computer Science,
Faculty of Science, Eberhard Karls University of Tübingen,
Sand 14, 72076 Tübingen, Germany
martin.butz@uni-tuebingen.de; armin.gufler@student.uni-tuebingen.de;
konstantin.schmid@tum.de; FSchrodt@gmx.de

Abstract

Performing a mere infinite number of different movements in our everyday life happens mostly in an automatic and unconscious way, requiring hardly any attention. One necessary requirement for generating different movements dependent on the current circumstances is knowledge about redundant behavioral alternatives and the capability to flexibly choose the current best one. In this paper, we evaluate an architecture that learns to represent such behavioral alternatives in the form of a modular body model from scratch. Moreover, the architecture is able to selectively choose between the behavioral alternatives, yielding kinematic control commands. The proposed architecture combines (temporal) Hebbian learning mechanisms for learning the body model with model-based reinforcement learning techniques for controlling the body. We evaluate the current system capabilities, comparing several configurations and parameter dependencies. Our results show that the architecture can robustly learn a highly flexible arm control system.

1 Introduction

Nicolai Bernstein has called behavior that is extremely flexible and adaptive *dexterous behavior* [Bernstein, 1967]. A key ingredient to succeed in the generation of dexterous behavior is (i) knowledge about the redundant behavioral interaction alternatives and (ii) the capability to flexibly choose between these alternatives task-dependently on the fly.

Few learning models so far have focused on utilizing redundant behavioral capabilities for generating flexible behavior. On the other hand, a huge number of learning architectures exist that learn to coordinate an arm without exploiting redundancy. Traditional Reinforcement learning (RL) algorithms [Berthier *et al.*, 2005] and various forms of policy gradients have been used [Peters and Schaal, 2008; Sigaud and Peters, 2010]. Also other direct learning methods have succeeded in controlling robot arms, such as direct inverse modeling approaches [Kuperstein, 1988] or resolved motion rate control and distal supervised learning [Whitney, 1969; Jordan and Rumelhart, 1992], each of which resolves redundancy during learning.

Here we focus on the SURE_REACH model [Butz *et al.*, 2007; Herbort and Butz, 2007], which is a sensorimotor unsupervised learning model that learns about the redundancy of a body and resolves this redundancy on the fly once the current goal and constraints are given. The model represents the end-effector location (i.e. location of the hand) in a *task space* and, additionally, the arm constellation (i.e. joint angles) in a *posture space*. Previously, hard-encoded population codes were used to cover both spaces by means of uniform neural grids. Hebbian and temporal Hebbian learning mechanisms were used to learn the (redundant) inverse kinematic mappings from task space to posture space. Model-based RL within posture space was used to execute dexterous, goal-directed behavior. Both, location goals or posture goals can be pursued, while potentially avoiding obstacles and considering additional task constraints. It was also shown that the model is able to anticipate subsequent task goals and consequently to incorporate those into the current behavior optimization [Herbort and Butz, 2007]. Moreover, SURE_REACH is able to incorporate uncertainties in its goal choice [Herbort *et al.*, 2007]. In sum, it has been shown that SURE_REACH is able to generate highly dexterous behavior.

Although SURE_REACH proved to be a useful model of dexterous human arm control, two challenges remained. First, the neural representation in joint space scales hyper-exponentially with the degrees of freedom controlled, so that a seven degree of freedom arm cannot be modeled with sufficient accuracy. Second, the neural population codes were pre-wired and not learned. While the first issue has been addressed by modularizing SURE_REACH, separating the posture space into individual joint spaces [Ehrenfeld and Butz, 2013], the second issue remained open. Thus, we developed a model that also learns the neural population codes. In the following we explain and evaluate this system, which is able to learn a model of an arm with three degrees of freedom (3-DOF) in two-dimensional (2D) space from scratch.

2 Arm Model Overview

Using kinematic motor commands a 3-DOF 2D arm is simulated. The simulation provides angles and end-effector location signals to the learner and is able to execute kinematic motor commands (small angular changes), respecting joint and torque constraints and adding some noise. Fig. 1 gives an overview over the implemented architecture.

2.1 Learning of a Kinematic Arm Representation

In contrast to SURE_REACH, the arm space representations are learned using growing, self-organizing neural network techniques. In particular, we use the Time Growing Neural Gas (TGNG) algorithm [Butz *et al.*, 2010]. TGNG

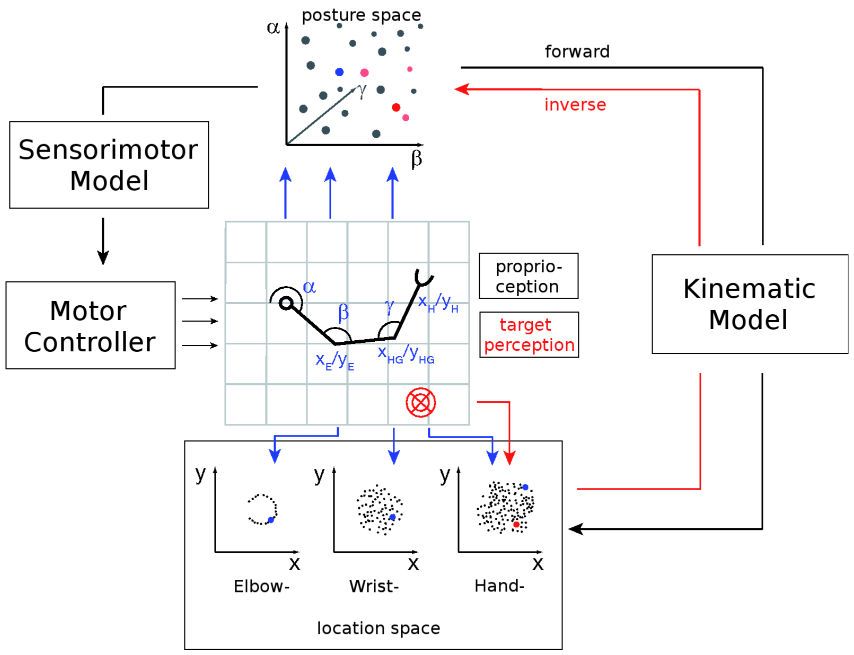


Figure 1: Overview of the arm model

grows neurons on demand, given the current sensory input differs more from the closest neuron than a threshold, specified by parameter θ. Moreover, TGNG grows neural connections between neurons that were the closest neurons in temporal succession. The connections are associated with a motor code that approximates the average motor command executed when traversing this connection. TGNG is used to learn neural representations of the task space given (x,y) locations of the end-effector, elbow, and wrist, and of the posture space given 3D angular vectors. Concretely, learning is achieved by a random walk, executing random arm movements and learning from the consequences. In this way, all accessible locations in the posture and task spaces will be observed, so that neural population codes can be distributed across the respective spaces by means of TGNG.

The second key component of the architecture is the mapping between task and posture spaces. Similar to SURE_REACH, we use a Hebbian learning mechanism [Carpenter and Grossberg, 1991] for learning the inverse kinematic mappings. More details about this learning process are provided below.

2.2 Goal Directed Behavior

The final central feature of the system is its ability to behave in a goal directed manner. Either a target in location space can be specified or a target in posture space. Pursuing a particular posture is rather easy: in this case, first, a posture goal activates the closest neurons in posture space. Next, this goal activity is propagated backwards throughout the posture space by means of model-based RL [Sutton and Barto, 1998]. Finally, control is invoked by deducing the neuron in posture space closest to the current arm posture, extracting the connection to the most strongly activated neighboring neuron, and executing the motor code that is associated with the neural connection. Given a well-connected network of neurons, behavior is guaranteed to reach the goal-activated neuron. When pursuing a goal location, however, this goal location first activates the closest neurons in the corresponding task space. Next, this activation is projected into the posture space by the inverse kinematic connection matrix (established by the Hebbian learning mechanism) between task space and posture space. The resulting goal manifold in posture space is then propagated throughout posture space and control is invoked as before.

To get more specific, we use the following notation: nodes of a neural network are denoted by small letters, $\mathcal{P}$ denotes the set of all posture space neurons, and $\mathcal{L}$ the set of all hand location space neurons. Furthermore, the terms "node" and "neuron" will be used in an interchangeable fashion, as nodes are part of neuronal networks. The activity of node n is denoted by a_n.

Location Goal Reaching in Detail

For a given target $X \in \mathbb{R}^2$, the neuron $l^* \in \mathcal{L}$ being nearest to X is determined. Then, the correlations (weights from the learned inverse kinematic) are used to activate corresponding posture nodes. As the mapping encodes full redundancy, there may be many posture neurons having significant correlations. Our reference approach considers all posture neurons having a correlation weight greater or equal to the correlation threshold τ ($\tau = 0.15$ has shown to be a good choice) and induces external activity a_p^{ext} to posture neuron $p \in \mathcal{P}$ by

$$a_p^{ext} = w_{p,l^*} \cdot \xi + 1, \quad \text{if } w_{p,l^*} \geq \tau, \tag{1}$$

with $w_{p,l} \in [0,1]$ being the learned correlation weight (mapping location- with posture-space neurons) between p and l. Thus, a good portion of neurons is ignored and a manifold of interesting postures is activated. The constant factor ξ applied to the weight is set to 0.3.

The induced activity is then propagated through the network using the following model-based state value learning rule (according to [Butz *et al.*, 2010])

$$a_p \leftarrow \max(a_p^{ext}; \gamma \cdot \max_{q \in N(p)}(a_q)), \tag{2}$$

where $N(p)$ denotes the set of neurons being neighbors of neuron p. The learning rule thus encodes either the own ex-

Table 1: Performance of location goal reaching (702 start-target combinations) using the setups STD (standard setting), *1ACT* (activating only the best node in location space), *ALLACT* (activating all nodes in location space), *NNEM* (check for new node after every movement). See text for details about the configurations.

Measurement	*STD*	*1ACT*	*ALLACT*	*NNEM*
Successfully reached	**676.7**	597.5	679.0	628.1
Quality of path (QoP)	**2.46**	4.09	10.50	3.10
Average #steps needed	**44.7**	65.8	246.2	56.6
Median #steps	**36.5**	61.8	126.0	38.3
Standard deviation #steps	**52.1**	43.8	336.8	128.3

ternal activity a_p^{ext} or the activity of the most active neighboring neuron. The parameter γ is a constant discount factor, set to 0.7 to accord with ξ, which is set to 0.3 in (1). In this way, propagated activity will always be smaller than externally induced activity from task space.

The reaching process uses the proprioception of the arm to determine which posture neuron is currently the closest. This is denoted as the arms' current node. Given the arm's current node, the system considers all neighboring nodes and attempts to move in the direction of the most active neighbor node. This is done until the arm has reached the actual target.

Inhibition Once a node has been visited by the arm, it gets inhibited, in order to make it less interesting for the arm to aim at the same posture again. Hence, loops where the arm moves back and forth are avoided effectively. Also in general it seems reasonable to avoid the reaching of the same postures with the arm within one goal-directed movement. A similar mechanism was also used in TGNG [Butz *et al.*, 2010].

Transitioning to neighbor nodes At every time step the arm is at a specific node and tries to move to the best neighbor node. An important concept is how to actually transition the state of the arm to the next node. The most straight forward approach is to check after each movement which node in the posture space network is nearest to the arms' new proprioception. Analyzing the behavior showed that in many cases the arm does not reach the neighbor node it originally aimed at. Often a node in between is encountered and becomes the new current node. In consequence, the arm has to do new planning again and the trajectories in turn become more turbulent. Such nodes in between are not necessarily connected to the node the arm came from and the neighbor it wants to move to, as connections are established depending on the movements made during the learning process. In effect, undesired erroneous behavior may occur.

To avoid this behavior, we estimate the approximate distance d the arm has to travel when starting at node v and aiming at neighbor node w. Searching for a new current node is now only done if the distance moved since starting at v is $\geq d$. The benefit of this approach is that movements are becoming much smoother overall and the arm jumps less between near nodes. Additionally, lots of searches for the current nearest node are omitted, which is particularly useful when the posture network is very dense.

3 Evaluation of the Current Performance

To evaluate the current performance we focus on reaching hand location goals. 702 different start-goal combinations were chosen randomly to test the behavior of the arm. Before trying tor reach targets, the system executes the self-supervised learning process for $T = 100,000$ time steps. Within each step one smooth movement of the arm is executed, the space representations (neural networks) are updated, and forward and inverse kinematics are learned. No obstacles were placed within the environment for the main parameter evaluations. In order to decrease the influence of random learning movements, if not reported differently, all results presented are average values from 100 runs, each of which was tested on the identical 20 sets of 702 start-goal combinations.

We evaluated the system with the aim of revealing parameter dependencies and robustness. Thus, we first vary several crucial parameter settings revealing the respective parameter influences. After that, we illustrate the capability of the arm to avoid obstacles. Finally, we show one slightly more involved run, in which case more than 99% of all location goals are reached successfully.[1]

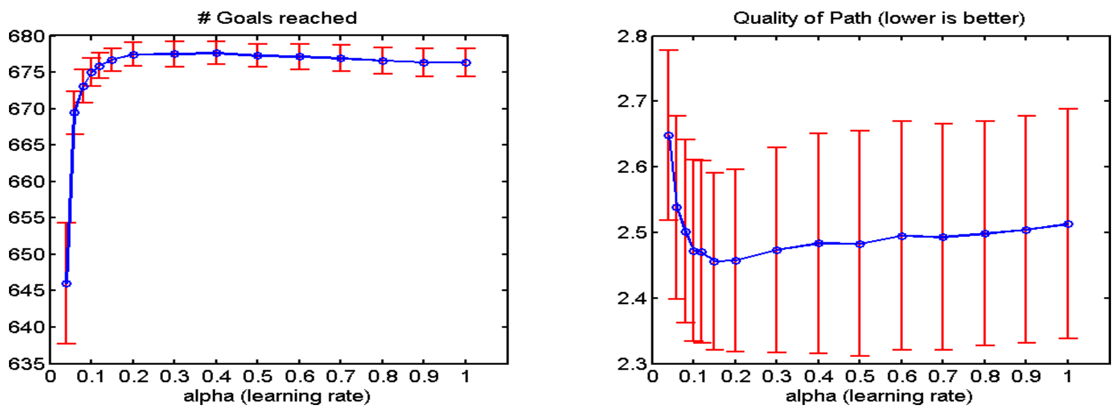
3.1 Goal-based Neural Activation

Table 1 shows the performance of goal directed behavior of the arm. We present the average number of successfully reached targets (of 702 in total) along with measurements regarding the quality and length of the path taken. The latter values are only considered for successful movements. The quality of path (QoP) is defined as the ratio of the actual distance moved to the optimal distance, with the optimal distance being the Euclidean distance from start to target location.

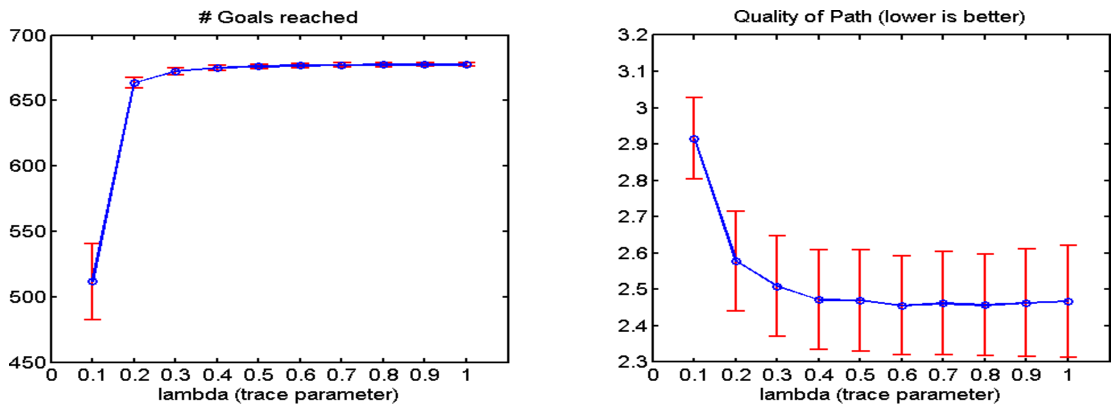
The reference approach (STD) induces activity to a manifold of suitable posture neurons, activating each of them with an external activity close to one (slightly graded dependent on the strength of the learned connection weights). Note that the system does not reach all targeted locations because we stop the goal reaching and count the trial as a failure when the highest activated node in posture space is reached but this node is not close-enough to the targeted goal. If the trial was not stopped in this case, the system typically reaches all 702 nodes eventually (due to the neural inhibition mechanism), but the quality of the path degrades strongly.

The baseline approach (1ACT) considers the results when only the strongest connected node in posture space is activated by the task space goal node. In this case, the redundant mapping is fully disregarded; the arm simply attempts to reach the neuron with the highest connection

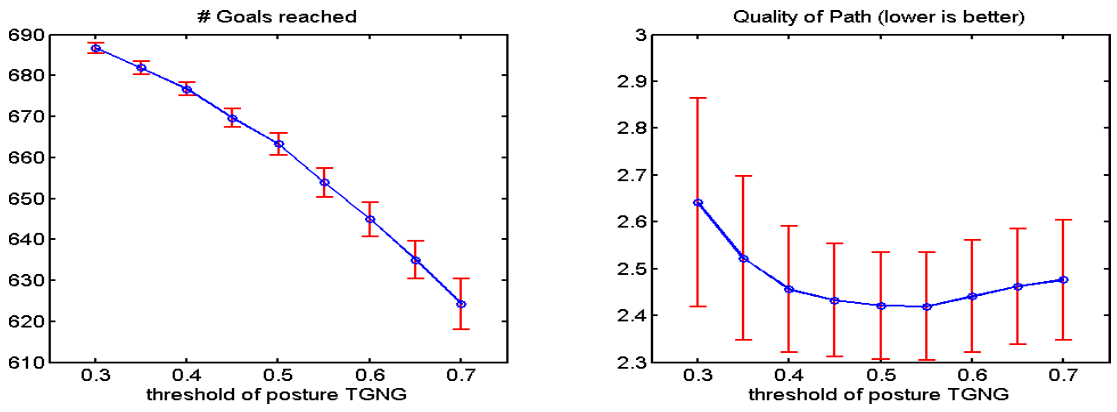
[1]If not stated differently, the parameters of the system were set as follows: TGNG node creation thresholds: $\theta_P = .4$; $\theta_L = .2$; TGNG node parameter adaptation value: $\epsilon_P = .075$; $\epsilon_L = .2$. Hebbian mapping learning parameters: learning rate $\alpha = .15$, trace value $\lambda = .6$, decelerating learning rate .99.



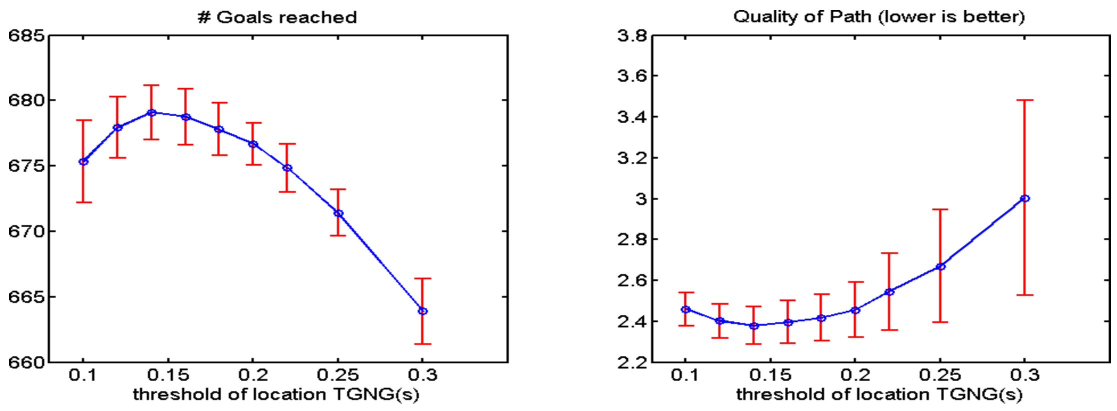
(a) Influence of learning rate parameter α.



(b) Influence of activity trace parameter λ



(c) Influence of posture space TGNG threshold parameter θ_P.



(d) Influence of location space TGNG threshold parameter θ_L.

Figure 2: Parameter influences on goal-reaching and quality of path performance.

weight from task space. The evaluation results show, that less goals are reached and the average distance moved to reach goals is larger using this configuration. This clearly shows, that the information encoded in the mapping is indeed useful. It activates a whole manifold of posture nodes and nicely handles redundancy.

In the all activated (ALLACT) case, all posture nodes that are connected to the goal location neuron are activated. The evaluation demonstrates (see Table 1) that the goal reaching performance of the arm gets worse using this configuration. Even though most goals are finally reached, the quality of the reaching movement is really bad compared to the other configurations. This is because neurons in joint space are activated that are hardly anywhere close to the goal location – thus the arm moves through this broad goal manifold in posture space rather randomly, yielding a much worse QoP.

As explained in Sec. 2.2 our reference model does not search for a new current posture node after each executed movement step. *NNEM* denotes the alternative approach estimating a new nearest posture node after each step (as was done previously in TGNG [Butz *et al.*, 2010]). The evaluations of Table 1 clearly show that this approach is leading to worse results. All quality measures get worse, particularly the much higher variance in the steps needed suggests that there are cases where the arm is getting lost in some loops or unsuitable detours. Note however, that the approximation of the distance to travel before making a transition may in other representation spaces not be that easy to estimate. For example many obstacles or frequently changing environments can be a problem.

3.2 Parameter Influence

Besides these goal-directed neural activations, we also pursued a more involved study of parameter learning influences. In the following, we explore influences of the learning rate and the trace parameter of the Hebbian learning mechanism, as well as of the threshold parameter of TGNG for learning the neural location and posture populations.

Hebbian Learning Parameters

The learning rate parameter α determines how fast and aggressive the mapping weights between task and posture space are adjusted. Learning the inverse mapping is done via the Hebbian feedback learning rule [Carpenter and Grossberg, 1991]:

$$\Delta_w = \alpha \cdot \overline{a_p} \cdot (a_l - w) \tag{3}$$

where w is the current weight between node $p \in \mathcal{P}$ and location node $l \in \mathcal{L}$ and Δ_w is the change of the weight in the current learning step. $\overline{a_p}$ denotes the activity trace of p and a_l the activity of l.

Varying the parameter between 0.04 (very slow learning process) and 1.0 (immediate weight changes) shows that a rather low learning rate between 0.1 and 0.3 is well-suited (cf. Fig. 2(a)). Higher values do not necessarily lead to much less goals reached, but the quality of path gets worse. Overall, however, the learning rate has a rather small influence on the whole system, if within a reasonable, moderate range.

The parameter λ defines the length of the activity trace, when learning the weights of a mapping between posture and location space. The rule to compute the activity trace $\overline{a_n}$ of node n is defined as

$$\overline{a_n} = (1 - \lambda) \cdot a_n^{old} + \lambda \cdot a_n^{new}, \tag{4}$$

where a_n^{old} is the activation of n in the previous step and a_n^{new} the newly induced activation in the current time step. The activity trace aims at encoding movement trajectories. Thus, nodes visited previously can also be correlated to current nodes (from other representation spaces). Setting λ to 1 means that there is no activation trace at all.

The results shown in Fig. 2(b) suggest, that a medium lambda parameter ≥ 0.4 is suitable, in order to have a good QoP. Having a really long trace (low λ), the resulting mappings are lacking quality. Having no trace on the other hand seems to be a reasonable option. At least the variables analyzed here are not suffering from the absence of the activity traces.

Thresholds of TGNG Networks

The threshold parameter(s) of the Time Growing Neural Gas networks (TGNGs) [Butz *et al.*, 2010] have a huge influence on the general performance. A lower (error)-threshold means that the network will have higher density because new nodes are inserted more frequently.

Fig. 2(c) shows the influence of the posture threshold θ_P and the location threshold θ_L. Generally, with a lower threshold parameter θ_P more goals can be reached successfully. But having $\theta_P < 0.4$ leads to a significantly worse average quality of path. Further evaluation showed, that for $\theta_P < 0.4$ the standard deviation of the average number of steps needed to reach targets also increases significantly. With more nodes in posture space the arm has more movement possibilities and the probability that it looses the optimal direction for some steps increases. Moreover, the number of updates per connection decreases when a denser network is grown. Another important consideration at this point is the proper choice of θ_L depending on θ_P and vice-versa. The densities of the networks should be somewhat comparable to ensure proper mappings between them. While this is a rather experimental task at this point, it appears that the densities can be increased significantly for both networks without getting bad behavioral results. However, the computation-wise efficiency worsens with decreasing θ_P as the number of nodes increases quite rapidly (approximately 4000 nodes for a threshold of 0.3 compared to 2000 nodes with the threshold set to 0.4). Moreover, the number of learning iterations necessary to develop a proper kinematic mapping and to associate sufficiently accurate motor codes increases with increasing density of the networks.

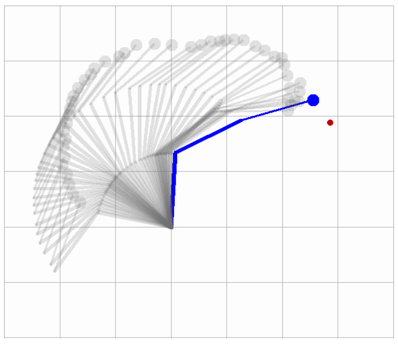
The location TGNG threshold parameter θ_L has a similar influence. With increasing θ_L, less goals can be reached. The quality of path gets worse with increasing θ_L too. But for lower values like 0.15 and 0.20 the difference is not really significant.

Obstacles

The system is also able to handle obstacles within the environment. To avoid crashing into them, neurons near and within obstacles are simply inhibited. Moreover, the inverse kinematics model provides information about postures that are possibly causing crashes, which are also inhibited so that no reward can be propagated through obstacle-based inhibitions. Fig. 3 shows typical trajectories of the arm when reaching a target, pointing out different trajectories chosen due to an obstacle.

Reliable Goal Location Reaching

Seeing that the presented results so far have not reached all goal locations robustly, we ran slightly more involved runs with the TGNG threshold parameters set to $\theta_P = .2$;


(a) Without obstacles

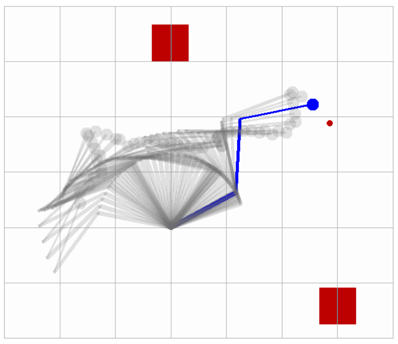
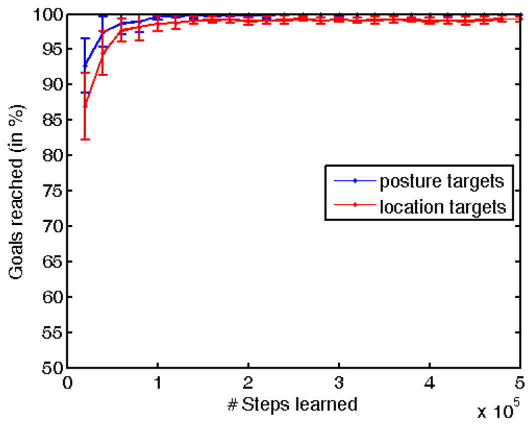

(b) With two obstacles

Figure 3: Two reaching movements, (a) without obstacles and (b) with two obstacles (red squares). The arm has to move its end effector to the location goal target (small red circle). The trajectory taken by the arm is visualized in light gray color. The resulting posture when the target was (nearly) reached is shown in blue. Due to the obstacles in (b) the resulting trajectory differs, effectively avoiding crashing into the upper obstacle.



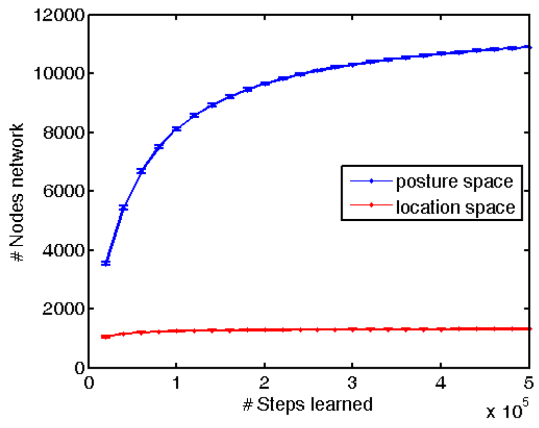


Figure 4: With even lower TGNG thresholds, the networks grow bigger and the target reaching performance also reaches close-to 100% even when reaching for location targets.

$\theta_L = .05$ for $500,000$ learning iterations. All other parameters were set to the standard settings. Fig. 4 shows that with these settings nearly all location goals are reached. The results also confirm that posture goals are easier to reach and are in all cases reached reliably. Moreover, the results also show that the network sizes grow significantly. Clearly the posture network is much larger due to the 3D angular space being covered in contrast to the 2D location space. When considering the path qualities achieved, Fig. 5 shows that the path does not become fully straight. The performance with respect to posture space goals indicates that the motor encodings in the neural connections are not perfect. Moreover, due to the focus on the next best node, the path cannot be fully straight. Low-pass filters in the motion generation may alleviate this problem. The location space performance is even worse. In this case, though, the system attempts to generate a straight path in posture space, not in location space. Thus, the measure confirms progressive learning but it is not an absolute performance measure.

In sum, the results show that the overall system is able to reach all goals in posture space as well as in location space, even though all representations and associations between these representations were learned from scratch learning from uncontrolled, random motions. While the path optimality may be improved further, it has to be kept in mind that the system currently blindly executes each movement

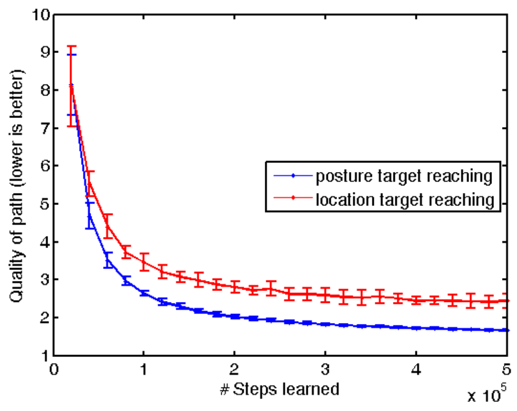


Figure 5: Also the path quality improves when a larger network is learned. Note that the system always attempts to move straight in posture space - thus the path quality for location goals has to be taken with a grain of salt. Nonetheless, there is definite room for improvement when considering the path quality results.

without considerations of the previous one. Thus, integrating successive motion vectors may be able to solve the challenge of generating more smooth and straight paths to goals. Nonetheless, the results concerning obstacle avoidance have confirmed that the system is indeed able to generate dexterous behavior, having learned its body model fully from scratch.

4 Conclusion

The results presented in this paper confirm that the combination of TGNG with Hebbian learning and model-based RL works rather effectively. However, clearly learning takes a rather long time and the final path quality is not optimal. Note, however, that learning was based on completely random movements. Others have shown that goal-babbling from early on can improve the speed of learning significantly [Rolf *et al.*, 2011]. Moreover, active information seeking, that is curious behavior [Oudeyer *et al.*, 2007], which was included in the TGNG algorithm [Butz and Reif, 2010], may be included in the current system to speed-up learning even further by essentially acting information-oriented. In this study, however, we refrained from utilizing such techniques to reveal a baseline system performance.

Besides curiosity, other motivations may be included to explore the external environment further once the arm model is sufficiently accurate. Forward kinematic mappings can also be learned along similar lines, allowing the anticipation of action consequences in posture space as well as in task space. Such anticipatory capabilities may be used for filtering incoming sensory information, acting based on internal expectations, as well as for forward planning. Finally, we intend to use this learning approach to learn the population encodings and mappings in the modular modality frame (MMF) model [Ehrenfeld and Butz, 2013]. MMF modularizes the SURE_REACH approach yielding a body model with maximally three dimensional spaces. However, currently no structural learning takes place in MMF. Due to this dimensional restriction in MMF, the combination of the utilized learning techniques with MMF promises to yield a system that is able to learn a full seven degree of freedom human arm model or even a full human body model in 3D space. Future research will need to investigate the capabilities of generating dexterous behavior within such a learned, distributed body model.

Acknowledgments

Funding from the Emmy Noether program (German Research Foundation, DFG, BU1335/3-1) is acknowledged. Moreover, the authors like to thank the Cognitive Modeling team for their support.

References

[Bernstein, 1967] N A Bernstein. *The co-ordination and regulation of movements*. Pergamon Press, Oxford, 1967.

[Berthier *et al.*, 2005] N. E. Berthier, M. T. Rosenstein, and A. G. Barto. Approximate optimal control as a model for motor learning. *Psychological Review*, 112:329–346, 2005.

[Butz and Reif, 2010] M. V. Butz and K. L. Reif. Motivated TGNG: Algorithm and performance evaluations. Technical Report CoboslabY2010N001, COBOSLAB, Department of Psychology, University of Würzburg, Würzburg, Germany, 2010. http://www.coboslab.psychologie.uni-wuerzburg.de.

[Butz *et al.*, 2007] M. V. Butz, O. Herbort, and J. Hoffmann. Exploiting redundancy for flexible behavior: Unsupervised learning in a modular sensorimotor control architecture. *Psychological Review*, 114:1015–1046, 2007.

[Butz *et al.*, 2010] M. V. Butz, E. Shirinov, and K. L. Reif. Self-organizing sensorimotor maps plus internal motivations yield animal-like behavior. *Adaptive Behavior*, 18(3-4):315–337, 2010.

[Carpenter and Grossberg, 1991] G. A. Carpenter and S. Grossberg. *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, Cambridge, MA, 1991.

[Ehrenfeld and Butz, 2013] S. Ehrenfeld and M. V. Butz. The modular modality frame model: Continuous body state estimation and plausibility-weighted information fusion. *Biological Cybernetics*, 107:61–82, 2013.

[Herbort and Butz, 2007] O. Herbort and M. V. Butz. Encoding complete body models enables task dependent optimal behavior. *Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007*, pages 1424–1429, 2007.

[Herbort *et al.*, 2007] O. Herbort, D. Ognibene, M. V. Butz, and G. Baldassarre. Learning to select targets within targets in reaching tasks. *6th IEEE International Conference on Development and Learning*, ICDL 2007:7 – 12, 2007.

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

[Kuperstein, 1988] M. Kuperstein. Neural model of adaptive hand-eye coordination for single postures. *Science*, 239:1308–1311, 1988.

[Oudeyer *et al.*, 2007] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11:265–286, 2007.

[Peters and Schaal, 2008] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21:682–697, 2008.

[Rolf *et al.*, 2011] M. Rolf, J. J. Steil, and M. Gienger. Online goal babbling for rapid bootstrapping of inverse models in high dimensions. *IEEE Int. Conf. on Development and Learning and on Epigenetic Robotics*, pages 1–8, 2011.

[Sigaud and Peters, 2010] O. Sigaud and J. Peters, editors. *From Motor Learning to Interaction Learning in Robots*. Springer, 2010.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 1998.

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