Learning to Select Targets within Targets in Reaching Tasks

Oliver Herbort[1], Dimitri Ognibene[2], Martin V. Butz[1], Gianluca Baldassarre[2]
[1]*University of Würzburg, Roentgenring 11, 97070 Würzburg, Germany*
[2]*Istituto di Scienze e Tecnologie della Cognizione - CNR, Via S.M. della Battaglia 44, 00185 Roma, Italy*
[oliver.herbort, butz]@psychologie.uni-wuerzburg.de, [dimitri.ognibene, gianluca.baldassarre]@istc.cnr.it

***Abstract*— We present a developmental neural network model of motor learning and control, called RL_SURE_REACH. In a childhood phase, a motor controller for goal directed reaching movements with a redundant arm develops unsupervised. In subsequent task-specific learning phases, the neural network acquires goal-modulation skills. These skills enable RL_SURE_REACH to master a task that was used in a psychological experiment by Trommershäuser, Maloney, and Landy (2003). This task required participants to select aimpoints within targets that maximize the likelihood of hitting a rewarded target and minimizes the likelihood of accidentally hitting an adjacent penalty area. The neural network acquires the necessary skills by means of a reinforcement learning based modulation of the mapping from visual representations to the target representation of the motor controller. This mechanism enables the model to closely replicate the data from the original experiment. In conclusion, the effectiveness of learned actions can be significantly enhanced by fine-tuning action selection based on the combination of information about the statistical properties of the motor system with different environmental payoff scenarios.**

***Index Terms*— Motor Learning, Motor Control, Noise, Redundancy, Optimal Control**

I. Introduction

Our world confronts us with an abundance of potential targets that we can reach or interact with in some way. Much research has been conducted to understand the neural mechanisms that enable the selection of single objects and suitable actions to manipulate them [1]. This research mainly focuses on the selection of a discrete target among few visually distinguishable ones and on the precision of the execution of the consequent action (cf. [2]).

However, after a target has been selected, there are redundant ways to implement the related reaching action in that the system has still to establish (1) where exactly and (2) how exactly to approach the target. The former problem may be termed a *location redundancy problem*: almost all targets can be reached assuming different final hand-contact points. For example, movements to grasp a pen can terminate at different locations along the pen's long axis without substantially different outcomes. The latter problem is often referred to as the *motor redundancy problem*. It arises, for example, because each hand position in extrinsic space can be realized by different arm postures and one can move to each arm posture with an abundance of different arm trajectories. Since the motor system can only realize one possibility at a time, it has to decide between seemingly equivalent alternatives. Regardless of which type of redundancy applies, a fundamental challenge for a motor control system is to choose those means that accomplish a task most reliably and efficiently.

In the motor control literature *optimal control* describes ways to cope with redundancy [3]. If redundant possibilities are at hand to accomplish a task, additional criteria may be considered to choose the currently optimal actions. Recently, optimality criteria have been proposed that lead the motor system to maximize movement accuracy by reducing the impact of motor noise [4], [5]. Most models focus on the resolution of motor redundancy. However, also the resolution of location redundancy is important to optimize action outcomes. Recent psychological experiments revealed that humans consider motor noise and different penalty situations when choosing movement aimpoints within target areas (cf. [6]). In these experiments, a target area was touched at locations shifted away from an adjacent area if hitting the adjacent area was both likely and strongly penalized.

Here, we present a developmental computational model of unsupervised motor learning and the acquisition of task-specific reaching skills in an experimental context. The development of the motor control system is modeled by the *Sensorimotor Unsupervised Redundancy Resolving Architecture (SURE_REACH)* [7], [8]. This neural network architecture is capable of solving the inverse problem of generating a sequence of motor commands to move a redundant arm to goal locations encoded in an extrinsic coordinate frame. SURE_REACH is enhanced by a neural population-coded reinforcement-learning model, which generates optimal target representations for maintaining a high level of performance despite system-inherent neural and motor noise [9]. We refer to the enhanced architecture as RL_SURE_REACH. We used RL_SURE_REACH to simulated the above mentioned reaching task [6]. As shown in detail below, the architecture successfully reproduces the tendency of humans to adjust movement endpoints dependent on the distance and severity of a penalty area. In the remainder, Section II and III describe the computational model, the original experiments and its simulation, Section IV presents the simulation results, and Section V draws conclusions.

II. The Model

Fig. 1 shows the four main components of the model. (1) A model of the human motor apparatus and the experimental setup receives motor commands and provides the proprioception of current joint angles, visual information about the hand position and target locations in extrinsic space, and overall reward values to the neural controller. (2) A *motor controller (MC)* generates step-by-step motor commands to move the arm toward target postures ($\mathbf{p}_{target}$). MC is trained by unsupervised associative learning in an initial "childhood" learning phase, during which random motor commands are executed and their effect on joint postures are encoded. Later on, this information is used in an inverse fashion to map from (desired) arm postures to motor commands. (3) As the task requires movements to targets represented in an extrinsic coordinate frame, a *posture memory (PM)* converts an extrinsically encoded hand target ($\mathbf{h}_{target}$) into a representation of the redundant arm postures that correspond to it. The output of PM is used as the target representation for MC. Like MC, PM develops in an unsupervised fashion during the childhood learning phase. (4) Finally, an actor-critic *reinforcement learning (RL)* mechanism [10] modulates the retinal input ($\mathbf{i}$) before it is used as target representation for MC. During task specific learning phases in the simulation of the experiment, RL explores the consequences of the selection of varying target representations. It crystallizes on a mapping of retinal to target representations that, given the configuration of the reward/penalty areas as well as neural and motor noise, maximizes the overall payoff.

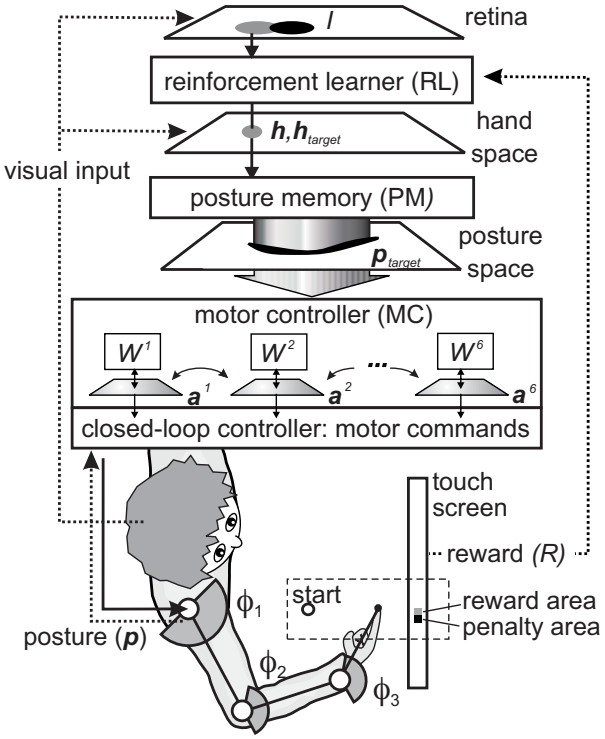

Fig. 1. The simulated experimental task requires fast arm movements from a start location to a rewarded target on the screen while avoiding an adjacent penalty area. The retinal representation of the visual configuration of the stimuli ($\mathbf{i}$) is used by a reinforcement-learning component (RL) to select a hand target ($\mathbf{h}_{target}$) that is then passed to a posture memory (PM). PM expands the hand target representation to a representation of the redundant associated arm postures ($\mathbf{p}_{target}$), which is used by the motor controller (MC) to generate motor commands. PM and MC are acquired unsupervised in a childhood phase, whereas RL learns to fine-tune motor control in task-specific experimental learning phases.

A. Arm Model

The model of a three joint planar arm roughly approximates the kinematic features of a human arm that is restricted to move on the horizontal plane. The lengths of the upper arm, forearm, and hand are $l_1 = 30cm$, $l_2 = 25cm$, and $l_3 = 20cm$, respectively. The shoulder, elbow, and wrist joints are allowed to move within $-45° \leq \phi_1 \leq 135°$, $-140° \leq \phi_2 \leq 0°$, and $-70° \leq \phi_3 \leq 70°$, respectively. Each limb is actuated by two antagonistic "muscles" each of which is activated by motor commands ranging between $0.0 \leq mc_i \leq 1.0$. The final movement of a joint ϕ_i is determined by subtracting antagonistic motor commands and scaling the result by a gain factor $g = \sigma 2.25°$, where σ is a Gaussian distributed random value ($m = 1, SD = 0.05$):

$$\phi_i(t+1) = \phi_i(t) + g(mc_{2i-1} - mc_{2i}), \quad i = 1,2,3 \quad (1)$$

B. Space Representation

In the architecture, extrinsic visual space (2D), extrinsic hand location space (2D), and intrinsic arm posture space (3D) are represented. Visual stimuli are realized by colored dots (red, green or blue). The dots are used to activate the system's three color retinas, each formed by 20×20 neurons, similarly to the encoding of hand coordinates illustrated below (the activation of the three retinas is denoted by a vector $\mathbf{i}$ with $20 \times 20 \times 3$ elements). Hand coordinates are encoded in a population of neurons $\mathbf{h}$. Each neuron h_i of $\mathbf{h}$ fires, if the actual hand coordinates (x, y) are close enough to the neuron's preferred hand location (h_i^x,h_i^y):

$$h_i = \max(1.0 - \frac{|x - h_i^x|}{3.0}; 0) \cdot \max(1.0 - \frac{|y - h_i^y|}{3.0}; 0) \quad (2)$$

The preferred hand locations covered a task relevant $40cm \times 15cm$ rectangle (top left at $25cm, -7.5cm$ relative to shoulder, Fig. 1) forming a $20 \times 20 = 400$ node grid. Likewise, arm postures are encoded in a population of neurons $\mathbf{p}$, where each neuron p_i is activated according to:

$$p_i = \prod_{j=1}^{3} \max(1.0 - \frac{|\phi_k - p_j^k|}{23°}; 0) \quad (3)$$

where p_j^k are the preferred joint angles of each neuron, which cover the entire posture space with a $9 \times 7 \times 7 = 441$ node

grid, and ϕ_k is the k-th angle of the current posture.

C. Motor Controller

The motor controller (MC) develops six internal motor-command-specific arm models during the childhood learning phase. Later on, the MC can use these models to issue commands to the arm to reach desired postures, on the basis of neurally implemented dynamic programming.

1) Motor Learning: In the initial motor learning phase MC and PM are trained for 1,000,000 time steps. Initially, a random set of motor commands is generated by setting each motor command (mc'^k in Equation 9) to 1.0 with a probability of $p_{mc} = 0.3$ and to 0.0 otherwise. This procedure is repeated until at least one motor command is set to 1.0. A new random set of motor commands is generated in random intervals of 1 to 8 time steps, resulting in random arm movements. During this training, the touch screen used in the experimental setup was not present and no reward was provided to the system.

The postural transitions caused by each motor command are encoded in recurrently interconnected neural networks resulting in $n = 6$ motor-command-specific internal models. Each of the six neural networks consists of a single layer of 441 interconnected neurons whose activation is denoted by the vector $\mathbf{a}^k$, isomorphic to the neural population code of the posture ($\mathbf{p}$). Each neuron in a layer is connected to itself and all other neurons of the same layer by a 441×441 synaptic weight matrix $\mathbf{W}^k$. During learning, the neurons' activation vector $\mathbf{a}^k$ has the following dynamics:

$$\mathbf{a}^k(t) = \rho \mathbf{a}^k(t-1) + mc'^k(t-1)\mathbf{p}(t-1) \tag{4}$$

where ρ is a decay coefficient that enables the learning of temporally far reaching posture transitions in that it maintains a trace of past posture representations, and mc'^k is the activation of the k-th motor command. Neural network weights are updated on the basis of a Hebbian learning rule that associates the current posture $\mathbf{p}$ to the preceding, action dependently encoded postures in $\mathbf{a}^k$, thus linking to each potential target posture those postures from which the target can be reached if the k-th motor command is executed:

$$w^k_{ji}(t) = w^k_{ji}(t-1) + \delta a^k_j(t) p_i(t)(\theta - w^k_{ji}(t-1)) \tag{5}$$

where w^k_{ji} are single values of the weight matrix W^k, a^k_j and p_i are single values of the neuron vectors $\mathbf{a}^k$ and $\mathbf{p}$, δ is the learning rate, which decreases exponentially from $\delta_0 = 0.1$ to $\delta_{1{,}000{,}000} = 0.01$ during learning, and $\theta = 0.1$ is a ceiling value that prevents weights from increasing infinitely.

2) Dynamic Programming: Unlike many other motor control models (e.g., [11]), SURE_REACH does not acquire a specific inverse sensorimotor model that maps perceived and desired arm postures directly to motor commands during learning but generates such a mapping on the fly for each goal-directed movement. As soon as a hand target representation is provided, dynamic programming builds a mapping from postures to motor commands that are well-suited to reach the target from the respective posture. This target-specific inverse model is then used to direct the arm to the goal by closed-loop control. This dynamic process enhances the model's flexibility and enables it to incorporate novel task constraints without relearning [7], [8]. The dynamic programming process is based on the connectivity between the different neurons of each neural network ($\mathbf{W}^k$) and also on constant interconnections assumed to exist between neurons with identical receptive fields in different neural networks. In particular, at each time step the activity level $\mathbf{a}^k$ of neurons is updated as follows:

$$\mathbf{a}'^k \leftarrow \max\left[\beta\left(\gamma\frac{\sum_l^{l\neq k}\mathbf{a}^l}{n-1} + (1-\gamma)\mathbf{a}^k\right), \mathbf{p}_{target}\right] \tag{6}$$

$$\mathbf{a}^k \leftarrow \mathbf{a}'^k + \mathbf{W}^k \times \mathbf{a}'^k \tag{7}$$

where max returns the entry-wise maxima of two vectors, $\beta = 0.17$ regulates overall neural activity, $\gamma = 0.43$ regulates the intensity of crosstalk between networks, and $\mathbf{p}_{target}$ is the representation of suitable target postures, which is normalized so that single values add up to 1.0. The rationale of this formula is that once a target posture representation is provided to MC, a target activation is injected into all six neural networks. The activation $\mathbf{a}^k$ then spreads to the neighboring neurons through the lateral connections ($\mathbf{W}^k$). The activations $\mathbf{a}^k$ spread in the opposite direction of the successive activations experienced during the childhood learning phase. This activation diffusion results in different activity patterns in different networks. Activities $\mathbf{a}^k$ propagate action-dependently in the networks to those neurons that represent postures from which the target postures can be reached using the associated action. The activities also partially diffuse to corresponding loci of other networks through inter-network connections assumed to have constant weights equal to $\gamma/(n-1)$.

Given these network activations, motor commands are generated by comparing the activation levels of the neurons in the different networks that encode the current posture $\mathbf{p}$:

$$mc'^k = \mathbf{p}^T\mathbf{a}^k \tag{8}$$

$$mc^k = \frac{\max[mc'^k - mc'^{anta(k)}; 0]}{\sum_{l=1}^{6}\max[mc'^l - mc'^{anta(l)}; 0]} \tag{9}$$

where $mc^{anta(k)}$ is the antagonistic motor command to mc^k. This results in a normalized set of motor commands that moves the arm 2.25° in posture space (1-norm). By iteratively determining motor commands and executing them the current posture gradually changes and the goal is pursued smoothly.

D. Posture Memory

The PM is modeled by a fully connected single layer neural network which maps from extrinsic hand space to

intrinsic arm posture space by a 400×441 weight matrix $\mathbf{W}^{PM}$. At each time step during childhood motor learning, the current hand position (vector $\mathbf{h}$) and corresponding arm posture (vector $\mathbf{p}$) are associated with a Hebbian learning rule:

$$\mathbf{W}^{PM}(t) = \mathbf{W}^{PM}(t-1) + \epsilon \mathbf{p}\mathbf{h}^T, \quad (10)$$

where $\epsilon = 0.001$ is a learning rate. This procedure results in a neural network that connects each reachable hand location to *all* the redundant arm postures that correspond to it. A representation of redundant arm postures ($\mathbf{p}_{target}$) from a given hand target ($\mathbf{h}_{target}$) is retrieved by feeding $\mathbf{h}_{target}$ into the network:

$$\mathbf{p}_{target} = \mathbf{W}^{PM} \times \mathbf{h}_{target} \quad (11)$$

The representation $\mathbf{p}_{target}$ of redundant postures so obtained is then sent as input to MC where it is used to generate motor commands, as described above.

E. Reinforcement Learner

The RL component is a neural implementation of the actor-critic model [10], modified to take into account the population code of actions (see [9] for details). The actor is a two-layer feed-forward neural network that takes as input the retinas' activations $\mathbf{i}$ and has as output a layer of 20×20 sigmoid "vote" neurons (vector $\mathbf{v}$). These neurons have topological one-to-one connections with a layer of 20×20 leaky neurons, $\mathbf{l}$ having lateral excitatory connections with neighboring neurons and inhibitory connections with distant neurons. This connectivity and the neurons' leak imply that they engage in a many-winner-take-all competition based on the votes $\mathbf{v}$. This leads to the emergence of a unique "hill" of active neurons within $\mathbf{l}$. When any leaky neuron reaches an activation threshold of 2.0 the activation of the whole map $\mathbf{l}$ is passed to PM as input (details on the dynamics and parameters of the leaky neurons can be found in [9]). The critic network has the same input as the actor and a linear output unit. This unit assigns scalar evaluations $e(t)$ to perceived states $\mathbf{i}$ and uses pairs of successive evaluations together with the overall reward $r(t)$ to compute the *surprise* $s(t)$ (cf. [10]; ω is a discount factor set to 0.3):

$$s(t) = (r(t) + \omega e(t)) - e(t-1). \quad (12)$$

The surprise is used to train both the actor and evaluator each time the system accomplishes a reaching movement. The evaluator's weight vector $\mathbf{w}^e$ is trained on the basis of the temporal difference learning rule [10] (η is a learning rate set to 0.6):

$$\mathbf{w}^e(t) = \mathbf{w}^e(t-1) + \eta s(t)\mathbf{i}(t). \quad (13)$$

The actor's weight vector $\mathbf{w}^a$ is trained with the following supervised learning rule:

$$\mathbf{w}^a(t) = \mathbf{w}^a(t-1) + \eta \; s \; (\mathbf{l} \; .* \; (\mathbf{v} \; .* \; (1 - \mathbf{v})))\mathbf{i}^T \quad (14)$$

where .* is the vector point product and $\mathbf{v} .* (1 - \mathbf{v})$ is the vector of derivatives of the vote sigmoidal neurons. This formula implies that, in correspondence to active neurons $\mathbf{l}$, votes $\mathbf{v}$ are increased or decreased when surprise is respectively positive or negative (see [9] for details). To give RL a bias to select targets corresponding to visible objects, the weights of connections between the three color retinas neurons and the topologically corresponding vote neurons were initially set to 1.0 whereas those of non-topologically corresponding neurons were set to 0.0.

III. Experimental Setup

The following section describes the original experiment by Julia Trommershäuser and her colleagues [6] and its simulation with RL_SURE_REACH.

A. Original Experiment

In the original experiment, participants had to quickly touch a green circular target area ($1.8cm$ in diameter) on a touch screen monitor. A hit of the target was rewarded with 100 points. A red circular penalty area ($1.8cm$ in diameter) was displayed adjacent to or partially overlapping with the target. Hitting the red area was penalized by a loss of a certain amount of points. Two crucial parameters in the experimental setup were varied. First, the distance between the centers of the target and the penalty area could be $\pm 1.8cm$, $\pm 1.35cm$, or $\pm 0.9cm$, Second, the penalty for hitting the red area was either nonexistent (0 points), low (-100 points), or high (-500 points). Rewards and penalties where summed up in case of hits on overlapping areas.

During the tests, the coordinates of the hits on the touch screen were recorded to evaluate if the participants adjusted their movement strategy to the different pay-off scenarios. The results clearly show that average final movement positions were only close to the center of the target area if the penalty area was either distant or had no effect (0 points). Otherwise, the average final movement position was shifted away from the penalty area. The strongest shift occurs if both areas were highly overlapping and if the loss associated with the penalty area was high (-500 points). The authors concluded that the participants took into account knowledge of motor variability to select aimpoints within the target area that reduced the probability of accidentally hitting the penalty area even if this somewhat reduced the probability of receiving a reward. In doing so, the participants were able to maximize their overall reward.

B. Simulated Experiment

These experiments were simulated with the setup depicted in the lower part of Fig. 1. The simulated touch screen was placed $55cm$ in front of the simulated participant's shoulder. The red and green areas were displayed slightly behind the screen ($2.5cm$) to ensure that most movements actually hit the screen. Both targets were realized as $1.8cm$ long rows

of five equidistant red or green dots. The distances between targets were equal to those of the original experiments. The red dots were activated at 5% of the intensity of the green dots (activated with 1) to incorporate in RL the initial participants' knowledge that only green targets had to be hit. The rewards and penalties equal to $\{100, 0, -100, -500\}$ points were normalized to $\{0.2, 0.0, -0.2, -1.0\}$ before being sent to the system.

A trial began with the presentation of a white dot at the starting position, located at ($30cm$,$0cm$) to the right of the shoulder joint, making RL_SURE_REACH move the hand there. As soon as a movement ended within $5cm$ of the starting location, the target area and the penalty area were displayed. RL_SURE_REACH processed the visual information and executed a movement toward the screen. If the movement ended on the screen the overall reward was calculated and provided to RL. A penalty of -1 incurred if the movement failed to reach the screen. After reinforcement the next trial began with the display of the starting location. A movement was considered completed as soon as the hand touched the screen or did not move out of a $3cm \times 3cm$ area for 50 time steps. Nine independent runs with different target locations (y-coordinate: $-0.88cm$, $-0.66cm$, ..., $0.88cm$ relative to the shoulder joint) were simulated for the three penalty times six distance conditions, summing up to 162 total runs. Each run consisted of 500 movements to the screen.

IV. Results

To compare the simulation data with the original results, the distance of the hand positions relative to the center of the target (relative endpoint) was recorded. In particular, each simulated run was split into 10 blocks of 50 movements and the relative endpoint was averaged for each block. Movements that did not touch the screen were not included in the analysis (1.3%). In 5 of the 162 runs the reinforcement learner was unable to modulate the visual target representation in a way that ensured an average positive reward in the final 50 movements (4 runs with a penalty of -1 and a distance of $-0.9cm$ and one run for a penalty of -1 and a distance of $0.9cm$). These runs were also excluded from the analysis. Fig. 2 shows a summary of the average relative endpoints in the different conditions. The results clearly replicate those of Trommershäuser et al. [6]. In particular, they show that if the distance between the penalty area and the target area is small, the average end position is shifted away from the penalty area. This effect is stronger for the high penalty condition and absent if no penalty is delivered.

For the statistical analysis, we combined the data of runs with the same absolute distance between the target and penalty area by inverting the signs of distances and relative endpoints for movements with distances of $-1.8cm$, $-1.35cm$, or $-0.9cm$. For the relative endpoint of the last block of 50 movements, an analysis of variance revealed main effects for absolute distance, penalty, as well as a significant interaction between them: absolute distance, $F(2, 148) = 57.0$, $p < 0.001$; penalty, $F(2, 148) = 64.6$, $p < 0.001$; interaction, $F(4, 148) = 30.0$, $p < 0.001$. Post-hoc t-tests (Table I) confirm that the relative endpoint is only shifted if hitting the red area is both likely and associated to an actual penalty. To verify that the observed behavioral adjustment results in an increase of reward, the development of the average reward and the relative endpoint during the ten blocks of movements was analyzed (Fig. 3). The initial movement strategy only remains unchanged if hitting the red area is either unlikely (distance = $1.8cm$) or not penalized because a near optimal reward is ensured from the beginning of the simulated experiment. In the other conditions, the

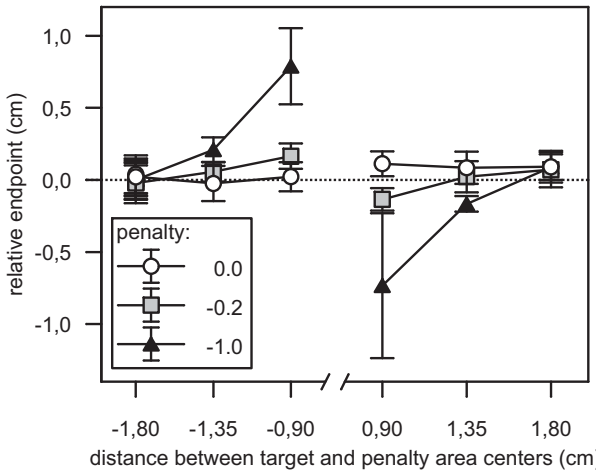

Fig. 2. Average relative endpoints of the final 50 movements of the simulated runs dependent on distances between target and penalty area and penalty. Error bars show standard deviations.

TABLE I
POSTHOC T-TESTS FOR RELATIVE ENDPOINT

	penalty: 0 vs. -0.2		penalty: -0.2 vs. -1	
abs. distance	T(34)	p	T(34)	p
0.90	5.860	< 0.001	6.070[a]	< 0.001
1.35	2.000	0.053	6.000	< 0.001
1.80	-0.404	> 0.500	0.049	> 0.500

	abs. dist.: 0.90 vs.1.35		abs. dist.: 1.35 vs. 1.80	
penalty	T(34)	p	T(34)	p
0.0	0.241	> 0.300	-0.663	> 0.500
-0.2	4.480	< 0.001	1.630	0.102
-1.0	4.880	< 0.001	7.260[b]	< 0.001

[a] t-test with $T(12.7)$ due to $n = 31$ and inhomogenity of variance
[b] t-test with $T(12.5)$ due to $n = 31$ and inhomogenity of variance

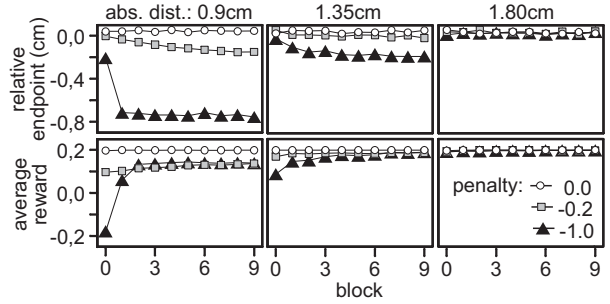

Fig. 3. Relative endpoints and rewards of movements, dependent on distance between reward and penalty area, penalty, and block (50 movements).

endpoint shifts away from the penalty area. In parallel, the average reward increases in value (Fig. 3). To quantify the impact of the various conditions on the target selection, the average reward per movement in the first 50 movements was compared with the average reward of the final 50 movements. Pairwise t-tests revealed significant improvements for the conditions with small absolute distances and non-zero penalty (Table II). The conditions with an absolute distance of $1.80cm$ also showed a significant but tiny improvement of the average reward (< 0.009 points), but no changes in the relative endpoint. All other conditions showed no significant changes (Fig. 3). In summary, these results confirm that the reinforcement learning based modulation of the target representations increases average payoffs.

TABLE II

PAIRWISE T-TESTS FOR REL. ENDPOINT AND REWARD CHANGE

		penalty: -1		penalty: -0.2	
abs. distance		T(17)	p	T(17)	p
0.90	reward	9.51[a]	< 0.001	4.50	< 0.001
	endpoint	4.50[a]	< 0.01	4.34	< 0.001
1.35	reward	7.20	< 0.001	5.71	< 0.001
	endpoint	5.10	< 0.001	2.98	< 0.01

[a] t-test with $T(12)$ due to $n = 13$ for abs. distance = 0.9

V. Conclusions

This paper presented the RL_SURE_REACH architecture, which was used to model the acquisition of basic motor skills with unsupervised learning during a childhood phase and their use for the acquisition of task-specific skills with reinforcement-learning in a later phase. The model was validated by reproducing data obtained in a psychological experiment, in which participants had to hit a rewarded area on a touch screen while avoiding to touch penalty areas, facing various cost and position configurations. In these tests, similarly to humans, the model exhibited a remarkable capability of shifting movement endpoints *within the target area*, taking into account the possibility of hitting the penalty areas due to motor and neural noise.

Most neural-network models of motor learning and control proposed so far focus on the extraction of compact representations of sensory-to-motor mappings. In this respect, the experiments presented here show that adding reinforcement-learning components to such models enables a sensorimotor control loop to take into account the statistical properties of the motor system. This can be very important to effectively solve the location-redundancy problem and thus increase behavioral performance. Neural population codes, as used in RL_SURE_REACH, are well-suited to encode knowledge about the statistical properties of tasks and our sensorimotor systems [12], [13]. The results reported here show that this knowledge is necessary to achieve one's goals optimally despite sensorimotor uncertainty.

Acknowledgment

This research was supported by the EU Projects *ICEA*, contract no. FP6-IST-027819-IP, and *MindRACES*, contract no. FP6-511931-STREP.

References

[1] G. Rizzolatti and G. Luppino, "The cortical motor system." *Neuron*, vol. 31, no. 6, pp. 889–901, 2001.

[2] P. Cisek and J. F. Kalaska, "Neural correlates of reaching decisions in dorsal premotor cortex: Specification of multiple direction choices and final selection of action," *Neuron*, vol. 45, no. 5, pp. 801–814, 2005.

[3] E. Todorov, "Optimality principles in sensorimotor control," *Nature Review Neuroscience*, vol. 7, no. 9, pp. 907–915, 2004.

[4] C. M. Harris and D. M. Wolpert, "Signal-dependent noise determines motor planning," *Nature*, vol. 394, pp. 780–784, 1998.

[5] E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature Neuroscience*, vol. 5, no. 11, pp. 1226–1235, 2002.

[6] J. Trommershäuser, L. T. Maloney, and M. S. Landy, "Statistical decision theory and trade-offs in the control of motor response," *Spatial Vision*, vol. 16, no. 3-4, pp. 255–275, 2003.

[7] M. V. Butz, O. Herbort, and J. Hoffmann, "Exploiting redundancy for flexible behavior: Unsupervised learning in a modular sensorimotor control architecture," *Psychological Review*, in press.

[8] O. Herbort and M. V. Butz, "Encoding complete body models enables task dependent optimal control," in press.

[9] D. Ognibene, A. Rega, and G. Baldassarre, "A model of reaching integrating continuous reinforcement learning, accumulator models, and direct inverse modelling," in *From Animals to Animats 9: Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior (SAB-2006)*, S. Nolfi, G. Baldassarre, R. Calabretta, J. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, and D. Parisi, Eds. Berlin: Springer Verlag, 2006, pp. 381–393.

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

[11] M. Haruno, D. M. Wolpert, and M. Kawato, "Mosaic model for sensorimotor learning and control," *Neural Computation*, vol. 13, no. 10, pp. 2201–2220, 2001.

[12] K. P. Körding and D. M. Wolpert, "Bayesian integration in sensorimotor learning," *Nature*, vol. 427, pp. 244–247, 2004.

[13] A. Pouget, T. Dyan, and R. Zemel, "Information processing with population codes," *Nature Reviews Neuroscience*, vol. 1, pp. 125–132, 2000.