Computational modeling of online reaching

Emmanouil Hourdakis[1] and Panos Trahanias[1]

[1]Foundation for Research and Technology
Institute of Computer Science
ehourdak@ics.forth.gr, trahania@ics.forth.gr

Abstract

Humans are able to perform an unlimited repertoire of reaching movements with high accuracy. The skillfulness with which we carry out a giving reaching task suggests that there are fundamental control policies that allow us to move our body. In the current paper we examine how an adaptive reach policy can be established, using biologically inspired techniques. The developed model, after an initial imitation phase, can replicate any given trajectory with very good performance.

Introduction

Reaching is a demanding task, due to the difficulty that lies in the coordination and control of the high-dimensional kinematics of the arm. Despite this fact, primates are able to perform it quite effortlessly. To interface between the symbolic level, that everyday tasks are described, and the low level of motor coordination, the brain uses several intermediate stages of processing (Atkeson, 1989). The richness of human motor abilities suggests that these stages allow the adaptive control of our body, by generalizing motor knowledge to other tasks and directions of movement. Much of this ability to reach is acquired during the early developmental stages of imitation (Piaget, 1962), where infants learn to regulate and control their complex musculoskeletal system.

Reaching motions have widely been studied in order to understand the brain structures that facilitate motor control. Research has revealed that the cerebral cortex uses several different cognitive processes to accomplish this goal, including kinematic (Atkeson, 1989) and dynamic (Soechting and Flanders, 1992) representations of movement, combined with forward and inverse models (Wolpert, 1997). To reduce the complexity of regulating all these processes, the brain makes use of modular structures (Ballard, 1986). Modularity is realized in various levels of the cognitive processing hierarchy and serves to hide the low level spinal system from the higher control centers of the cortex, allowing proper reuse of the motor knowledge.

At the spinal level, converging evidence suggests that modularity is implemented by a pre-coded set of control modules known as primitives (Degallier and Ijspeert, 2010). This concept has received considerable attention in the field of engineering. From a mathematical perspective the method of primitives, or basis functions, is an attractive way to solve the complex nonlinear dynamic equations that are required for motor control. For this reason several models have been proposed, including the VITE model that describes a way to regulate sets of agonist and antagonist muscles to move the limb to a desired state or the FLETE model that consists of a fixed parameterized system of differential equations that produce basis motor commands (see Degallier and Ijspeert, 2010 for a review). More recent studies in vertebrates suggest a force dependent encoding of motor primitives. For example experiments in paralyzed frogs revealed that limb postures are stored as convergent force fields (Bizzi et al, 1991). In (Gizster et al., 1993) the authors describe how such elementary basis fields can be used to replicate the motor control patterns of a given trajectory.

To be able to reach adaptively, the agent must learn to manipulate its primitives using control policies that generalize across different behaviors. In the cerebral cortex one of the dominant themes used for learning is by receiving rewards from the environment. This paradigm, known as reinforcement learning in engineering, does not require an exact learning signal of the error but rather a scalar, temporally delayed, reward function (Barto, 1995). It is more consistent with the type of feedback provided to humans during learning, where exact information on the error is usually not available. An agent that learns based on reinforcement learning tries to find a policy that will maximize the probability of receiving immediate or future rewards.

In the current paper we investigate how a simulated agent can learn an adaptive reaching policy using methods inspired from biological systems. To accomplish the low level motor control we employ the notion of force fields to design higher order primitives, i.e. motor programs that facilitate the synergetic control of multiple joints. Learning is implemented by modeling the circuitry of the dopaminergic neurons that are responsible for the perception of rewards in the cerebral cortex, and using it to form an adaptive control policy for reaching.

The proposed model consists of several interconnected regions. The roles of these regions are derived based on evidence from imaging and lesion studies that describe their cognitive functions. To implement the modularity at the cortical level we break down the whole system into pathways, i.e. sets of inter-dependent regions that carry out a specific process (Hourdakis and Trahanias, 2009). The computational areas in each pathway are modeled using liquid state machines (LSMs, Maass et al, 2002). LSMs consist an alternative to the traditional finite-state machine methods for brain modeling. Their difference lies in that they do not require any convergence to attractor states. Moreover, they are consistent with the homogenity inherent in the cortical regions where different processing functions are carried out by similar structures (Mountcastle, 1978). To accomplish this dynamic form of processing, LSMs perturb neuronal populations using

continuous or discrete input signals. A large variety of functions can be learned from this perturbation using readout neurons, i.e. neurons implemented with traditionally supervised learning methods. Recently it has been shown that LSMs can carry out any computation with fading memory, provided that the properties of separation and approximation are fulfilled (Mass et al 2002; Hourdakis and Trahanias, 2011).

In the following sections we describe the development and evaluation of a reaching model that is inspired by the aforementioned cortical processes. We begin by examining the biological evidence that underpins the model, and continue to describe the implementation of the model and its evaluation.

Computational Model

Cortical model

The central nervous system performs reaching by transforming sequential target locations into muscle commands that move the hand to a desired state (Soechting and Flanders, 1992). To relate the intrinsic proprioceptive state of the agent to extrinsic behavioral goals, such as the points in a trajectory, a forward transformation must be learned (Wolpert, 1997). Anatomical evidence from imaging studies suggest that the cerebral cortex learns such transformations using supervised learning (Doya, 1999). The forward model is implemented in the connections of the primary somatosensory cortex, where the proprioceptive state of the arm is encoded (Sergio and Kalaska, 2003), to the parietal lobe, which is responsible for state estimation. After the behavioral goals have been established, using the forward model and perception, reaching can be accomplished by the adaptive control of primitives. Data from animal lesions and human studies (Sakai et al, 1998) suggest that the basal ganglia are one of the main regions involved in learning sequential movements. This is accomplished by processing the rewards of the environment, which in the brain are evident from the secretion of dopamine, in order to gate motor programs (Thach et al, 2000). Learning of new motor policies is implemented in the projections of the basal ganglia with regions of the prefrontal cortex, where segments of motor acts are encoded (Jeannerod et al., 1995), and primary motor cortex, where the neurons' activity is strongly correlated to the level of activation of individual muscles (Todorov, 2000). Finally, lower level control is mediated by the connections of the primary motor cortex to the spinal cord (Dum and Strick, 1991).

To model the interaction between the sensori and motor systems in the cerebral cortex we use the notion of pathways (Hourdakis and Trahanias, 2009). Each pathway implements a distinct cognitive function and is defined by two factors: *(i)* the regions that participate in its processing and *(ii)* the directionality of the information as it progresses the levels of the cognitive hierarchy. This type of abstraction helps to identify and describe at a computational level how cognitive functions are carried out neuraly. As a result development of the model becomes a two stage process; first individual cognitive functions are designed, and then integrated together in order to achieve the required behavioral tasks. The modularity induced by pathways allows us to overcome traditional problems with large scale distributed architectures, such as cross talk. This type of modular approach provides important benefits in computational modeling since it allows identifying how the complex processes that exist in biological systems can be modeled with computational principles.

To design a reaching model, we identify three different pathways: *(i)* motor control, *(ii)* reward assignment and *(iii)* forward model. These are displayed in the following figure, where each pathway is marked with a different color.

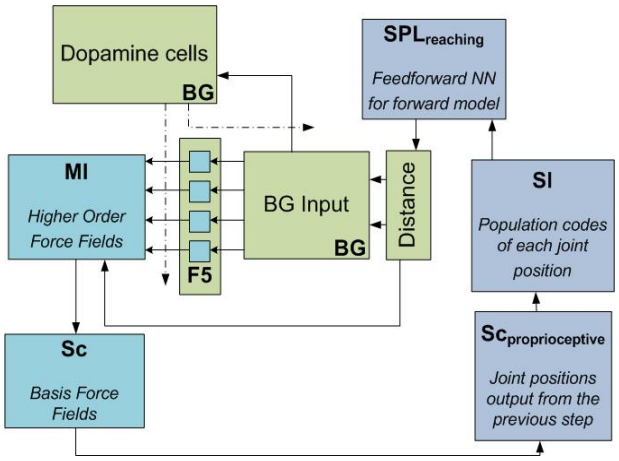

Fig. 1. The complete layout of our model with the three pathways, motor (blue), reward assignment (green) and forward model (purple).

Motor control (marked in blue) is responsible for the encoding of the primitive model. It includes regions Sc, where a set of basis primitives are hardwired, MI, where the basis modules are combined into higher order control modules and F5, where the higher order control modules are synthesized based on an adaptive reaching control policy. The latter is learned implicitly through the reward assignment pathway (marked in green). Finally the forward transformation of the body-centered state of the agent is accomplished in the forward (marked in blue) pathway. In the proposed model there is also an additional visual perception pathway that handles the perception of the trajectory. However, due to space constraints, in the current paper we assume that the trajectory is given to the agent as a series of consecutive points. In the following sections we describe the mathematical framework that underpins our model, as well as the implementation of each pathway.

Arm control

To model the effect that the torques have on the joints of the robot we use established laws from control theory (Paul, 1981). The second order kinematics of the robot hand are modeled using the following equation:

$$D(q, \dot{q}, \ddot{q}) = H(q)\ddot{q} + C(q, \dot{q})\dot{q} \quad (1)$$

where D is the controller that produces the torques that must be applied on the joints of the robot given its state q, and its first and second order derivatives, $\dot{q}$ and $\ddot{q}$ respectively. H is the joint-space inertia matrix and C describes the Coriolis and centripetal effects from the joint movement. Eq. 1 can be extended with additional terms such as the viscocity of the joints or the gravity loading of the plant. In the current paper, we applied the model on a simulated frictionless two-link plant, and therefore we didn't include these parameters.

The aim of the computational model is to derive the appropriate local control laws that will allow the plant to reach towards any location. In practice we look for a control policy that will map the state vector of the robot to a control vector from the computational model in a way that minimizes the error of reaching. Degallier and Ijspeert (Degallier and Ijspeert, 2010), suggested that such a control policy π can be defined as:

$$v = \pi(q, t, a) \quad (2)$$

where v are the joint torques that will be applied to the robot, q is the state space vector, t stands for time and a is the parameterization of the computational model.

The output of our model is the signal produced by the spinal cord circuit. In a biological agent the torques produced would be applied to the hand and result in movement. However since we use a simulated agent we find the second order kinematics of the hand by integrating eq. 1 and solving against the acceleration:

$$\ddot{q} = H(q)^{-1}\{\tau_p - C(q, \dot{q})\dot{q}\} \quad (3)$$

The next configuration state of the robot is calculated using the acceleration $\ddot{q}$ from the equation above, were H, C, q and $\dot{q}$ are as in eq. 1. The goal of the computational model is to produce the appropriate τ_p vector of joint torques that will enable the agent to perform reaching.

To evaluate the proposed model we use a simulated two-link planar arm. Control is accomplished by applying torques to the elbow and shoulder joints respectively. Therefore in the presented simulations the τ_p vector is two dimensional.

Forward model pathway

One of the main transformations that takes place during reaching is the cognitive implementation of a forward model (Wolpert, 1997). In the current paper, the forward model is implemented in the regions of the somatosensory and parietal lobe, and allows the agent to approximate the end point position of its hand using the proprioceptive input from the spinal cord.

To accomplish this we have designed the SI network to encode the proprioceptive state of the agent using population codes. This is inspired from the local receptive fields that exist in this region and the somatotopic organization of the SI (Kaas et al., 1979). Population codes assume a fixed tuning profile of the neuron, and therefore can provide a consistent representation of the encoded variable. To learn the forward transformation we train a feedforward neural network in the SPL region that learns to transform the state of the plant to a Cartesian x, y coordinate.

Motor pathway

Due to the high nonlinearity and dimensionality that is inherent in controlling the arm, devising an appropriate policy for learning to reach can be quite demanding. In the current paper this policy is established upon a few higher order primitives, i.e. self-organized spinal circuits that coordinate elementary motor behaviors. It turns out that, in the adopted planar arm, in order to perform any reaching behavior, only four higher order primitives are required namely up, down, left and right (Fig. 2). In humans such modules are formed during the first stages of the vertebrate motor development.

In order to make the agent generalize motor knowledge to different domains, the primitive model must be consistent with two properties: *(i)* superposition, i.e. the ability to combine different basis modules together and *(ii)* invariance, so that it can be scaled appropriately. Primitives based on force fields satisfy these properties (Gizster et al, 1993). As a result by weighting and summing the four higher order primitives shown in Fig. 2 we can produce any motor pattern required.

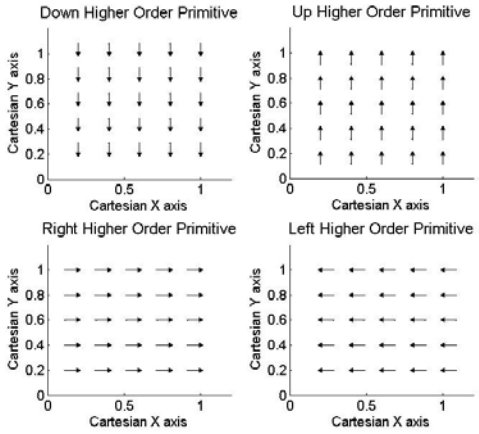

Fig. 2. The higher order primitive model proposed. The four plots show the force map of the primitive, i.e. the forces that are applied to the end position of the limb when the corresponding primitive is active. In the current model we use four different modules, namely up, down, left and right.

The higher order primitives are composed from a set of basis torque fields, implemented in the Sc module. By deriving the force fields using basis torque fields, the primitive model creates a direct mapping between the state space of the robot (i.e. joint values and torques) and the Cartesian space that the trajectory must be planned in (i.e. forces and Cartesian positions), resembling the way motions are processed by humans. We first define each torque field in the workspace of the robot, and then transform it to its corresponding force field. Each torque field is described by a Gaussian multivariate potential function:

$$G(q, q_0^i) = -e^{\left(\frac{(q-q_0^i)^T K^i (q-q_0^i)}{2}\right)} \quad (4)$$

where q_0^i is the equilibrium configuration of each torque field, q is the robot's angle and K^i a stiffness matrix. The torque

applied by the field is derived using the gradient of the potential function:

$$\tau^{\iota}(q) = \nabla G\left(q, q_0^i\right) = K^i\left(q - q_0^i\right) G\left(q, q_0^i\right) \quad (5)$$

Previous research has indicated that in order to achieve stability, two types of primitives must be defined: discrete and rotational (Degallier and Ijspeert, 2010). The rotational primitives are harmonic oscillators associated with a joint. The discrete ones apply a force on the hand based on a shaped valley with different equilibrium points. To ensure good convergence properties we have used 9 discrete and 9 rotational basis torque fields, spread throughout different locations of the robot's workspace (Fig. 3). These are generated from eq. 5 using different stiffness matrices. To generate the discrete torque fields (left block in Fig. 3) we use a semi-definite skew symmetric matrix K_{disc}, while to generate the rotational fields we use a rotation matrix, K_{rot}.

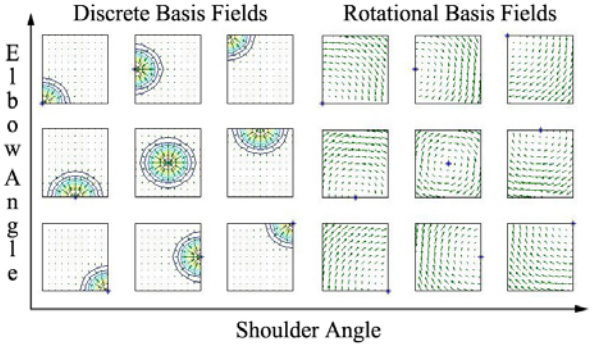

Fig. 3. Nine basis discrete (left block) and rotational fields (right block) scattered along the $-\pi..\pi$ configuration space of the robot. On each subplot the x axis represents the elbow angle of the robot while the y axis represents the shoulder angle. The two stiffness matrices used to generate the fields are $K_{disc} = \begin{bmatrix} -0.672 & 0 \\ 0 & -0.908 \end{bmatrix}$ and $K_{rot} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$.

Each plot in Fig. 3 shows the gradient of each torque field. The axes correspond to the $q1, q2$ joint values of the robot's hand. Since we want the model of higher order primitives to be based on the forces that act on the end point of the limb, we need to derive the appropriate torque to force transformation. To accomplish this we convert a torque field to its corresponding force field using the following equation:

$$\varphi = J^T * \tau \quad (6)$$

In eq. 6, τ is the torque produced by a torque field while φ is the corresponding force that will be acted to the end point of the plant if the torques are applied. J is the robot's Jacobian. In the current implementation where the plant is located in a 2 dimensional workspace, the 6x3 Jacobian matrix can be constrained to a 2x2 matrix as:

$$J = \begin{bmatrix} -l_1 * \sin(q_1) + l_2 * \sin(q_1 + q_2) & -l_2 * \sin(q_1 + q_2) \\ l_1 * \cos(q_1) + l_2 * \cos(q_1 + q_2) & l_2 * \cos(q_1 + q_2) \end{bmatrix} \quad (7)$$

Each higher order force field from Fig. 2 is composed by summing and weighting the basis force fields from eq. 6. To find the weight coefficients, we form a system of N linear equations by sampling M vectors P from the robot's operational space, for all B basis force fields.

$$\begin{bmatrix} \varphi_1^1(x^1) & \cdots & \varphi_1^B(x^1) \\ \vdots & \ddots & \vdots \\ \varphi_1^1(x^M) & \cdots & \varphi_1^B(x^M) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} P_1^1 \\ \vdots \\ P_2^M \end{bmatrix} \quad (8)$$

Each higher order force field is formed by summing and scaling the basis order force fields with the weight coefficients a. The vector a is obtained from the least squares solution to the problem:

$$\Phi * \alpha = P \quad (9)$$

In the results section we show the force fields that are produced by solving the system in eq. 9, as well as how the plant moves in response to a higher order force field.

Reward assignment pathway

One of the main methods of primate learning is by obtaining rewards from the environment. In the cerebral cortex, reward is associated with the secretion of dopamine, where approximately 80% of the dopaminergic neurons exist in the basal ganglia. One of the properties of these neurons is that they start firing when a reward is first presented to the primate, but suppress their response with repeated presentations of the same reward stimulus. At this convergent phase, the neurons start responding to stimuli that predicts a reinforcement, i.e. events in the near past that have occurred before the presentation of the reward.

In the early nineties, Barto (Barto, 1995) suggested an actor-critic architecture that was able to facilitate learning based on the properties of the basal ganglia. This architecture gave inspiration to several models that focused on replicating the properties of the dopamine neurons (see Joel et al., 2002 for a review). In the current paper we propose an implementation based on liquid state machines, and demonstrate how the interactions of this region with other neural networks of the brain can be modeled. The proposed implementation follows the actor-critic architecture and is shown in Fig. 4.

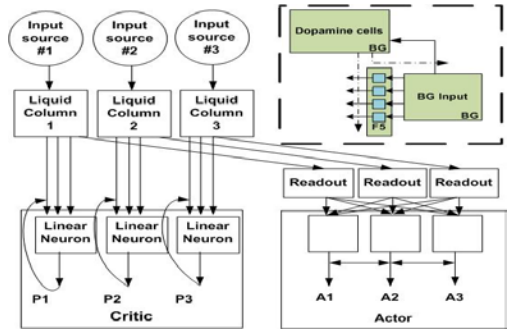

Fig. 4. The liquid state machine implementation of the actor-critic architecture. Each liquid column is implemented using a liquid state machine with feedforward delayed synapses. The critics are linear neurons, while the readouts are implemented using linear regression. On the top right of the figure (colored with green), we show how the actor-critic architecture is mapped on the model of Fig. 1.

The Critic neurons ($P1, P2, P3$) model the dopamine neurons in the basal ganglia. Their role is to learn to predict the reward that will be delivered to the agent in the near future. The $A1, A2, A3$ neurons learn based on the signal emitted by the Critic neurons. To model them in the current implementation we use a set of linear neurons. The liquid columns in Fig. 4 encode the input to the basal ganglia circuit.

To implement the neurons in each liquid column we use the leaky integrate and fire neuron model:

$$\tau_m \frac{dV_m}{dt} = -(V_m - V_{rest}) + R_m * \left(I_{syn}(t) + I_{inject} + I_{noise}\right) \quad (10)$$

where V_m is the membrane voltage, $\tau_m = C_m * R_m$ is the membrane time constant, R_m is the membrane resistance, C_m is the resistor capacitance, I_{inject} is a constant current injected to the neuron and I_{noise} a Gaussian random variable with zero mean and a small variance noise. After the emission of a spike, the membrane potential is reset to its resting value V_{rest}. $I_{syn}(t)$ is the incoming current from the presynaptic neurons.

The connections between the neurons in the liquid are implemented using a model of dynamic synapses (Markram et al., 1998). The post-synaptic potential (PSP) of each neuron is transferred to its efferent based on the following equations:

$$PSP_n = L * R_n * u_n \quad (11)$$

$$u_{n+1} = u_n * e^{\left(-\frac{\Delta t}{\tau_{facil}}\right)} + U * \left(1 - u_n * e^{\left(-\frac{\Delta t}{\tau_{facil}}\right)}\right) \quad (12)$$

$$R_{n+1} = R_n(1 - u_{n+1}) * e^{\left(-\frac{\Delta t}{\tau_{rec}}\right)} + 1 - e^{\left(-\frac{\Delta t}{\tau_{rec}}\right)} \quad (13)$$

The maximum output of the synapse is governed by the absolute synaptic efficacy L. The change of the efficacy is determined using the variables u_n and R_n, which are calculated according to eqs. 12 and 13. u_n defines the utilization of the synaptic efficacy which decays exponentially based on the τ_{facil} parameter to its resting value U. R_n is the fraction of available synaptic efficacy and defines the strength of the PSP_n at a given spike. It reduces due to the arrival of new spikes and recovers exponentially according to the τ_{rec} parameter.

The actors, i.e. the cortical region that learns based on the predicted rewards of the critics is implemented using a set of linear regression readouts that are trained to output a firing rate proportional to the sum of firing rates of each liquid column. Input from different sources is modeled as a set of rate code neurons that each projects to a separate liquid column using linear synapses with zero delay.

To implement the synapses between the liquid columns and the P, A neurons, we use the imminence weighting scheme (Barto, 1995). In this setup, the critic must learn to predict the reward of the environment using the weighted sum of past rewards:

$$P_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^t r_1 \quad (14)$$

where the factor γ represents the weight importance of predictions in the past and r_t is the reward received from the environment at time *t*. To teach the critics to output the prediction of eq. 14 we update their weights using gradient learning, by incorporating the prediction from the previous step:

$$v_t^c = v_{t-1}^c + n[r_t + \gamma P_t - P_{t-1}]x_{t-1}^c \quad (15)$$

where v_t^c is the weight of the Critic at time t, n is the learning rate and x_t^c is the activation of the critic at time *t*. The parameters γ, P and r are as in eq. 14. The weights of the actor are updated according to prediction signal emitted by the critic:

$$v_t^a = v_{t-1}^a + n[r_t - P_{t-1}]x_{t-1}^a \quad (16)$$

where v_t^a is the weight of the Actor at time t, n is the learning rate and x_{t-1}^a is the activation of the actor at time *t-1*. In the results section we demonstrate how the output of the Critic neurons approximates the response properties of the dopamine cells discussed above, as well as how the actor neurons learn to control the higher order primitive model.

Policy learning

Based on the higher order primitives and reward subsystems described above, the problem of reaching can be solved by searching for a policy that will produce the appropriate joint torques to reduce the error:

$$q_e = \hat{q} - q \quad (17)$$

where $\hat{q}$ is the desired state of the plant and q is its current state. In practice we do not know the exact value of this error since the agent has only information regarding the end point position of its hand and the trajectory that it must follow in Cartesian coordinates. However because our higher order primitive model is defined in Cartesian space, minimizing this error is equivalent to minimizing the distance of the plant's end point location with the nearest point in the trajectory:

$$d_e = |l - t| \quad (18)$$

where l and t are the Cartesian coordinates of the hand and point in the trajectory, respectively. The transformation from eq. 17 to eq. 18 is inherently encoded in the higher order primitives discussed before.

From the output of the forward model we obtain the end point Cartesian location of the hand, while from the demonstrator we obtain the point in the trajectory that must be reached. These are injected as rate codes into a liquid state machine, where a readout neuron is taught to estimate the subtraction of the two input rates using a feedforward neural network.

The policy is learned based on two elements: *(i)* decide which higher order primitive force fields will be activated, and *(ii)* determine each one's weight. The output of the actor neurons described in the previous section implement the activation of the canonical neurons in the F5 premotor cortex which are responsible for gating the higher order primitives. Due to the binary output of the actor neurons, when a certain actor is not firing then its corresponding force field will not be activated. In contrast when an actor is firing, its associated force field is scaled using the output of the subtraction readouts, mentioned above, and added to compose the final movement.

To teach the actors the local control law, we use a square trajectory shown in Fig. 5, which consists of eight consecutive points $p_1..p_8$. The agent is taught the trajectory backwards, i.e. starting from the final location (p_8) in four blocks. Each block contains the whole repertoire of movements up to that point. Therefore in the first block the actor learns to perform the left motion. Whenever it finishes a trial successfully, the actor is delivered a binary reward, and moves to the next phase which includes the movement it just learned and a new behavior.

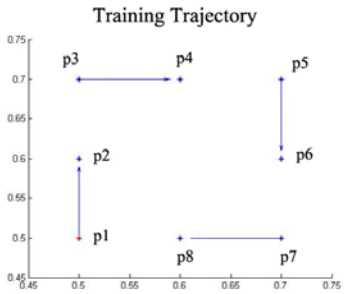

Fig. 5. The initial trajectory used to train the robot. It consists of 8 points that form 4 perpendicular vectors in four different directions (up, right, down, left).

Reward is delivered only when all movements in a block have been executed successfully. Therefore, the agent learns to activate the correct force field primitives using the prediction signal from the Critic neurons in Fig. 4. The final torque that is applied on each joint is the linear summation of the scaled higher order primitives:

$$\begin{aligned}\tau_p &= \left[x_{e,1} * (J^{-1})^T * \varphi_{up}\right]_{act1} \\ &+ \left[x_{e,2} * (J^{-1})^T * \varphi_{down}\right]_{act2} \\ &+ \left[x_{e,3} * (J^{-1})^T * \varphi_{right}\right]_{act3} \\ &+ \left[x_{e,4} * (J^{-1})^T * \varphi_{left}\right]_{act4} \quad (19)\end{aligned}$$

where $x_{e,i}$ is the output from the neural network distance readout, while $[\ ]_{act}$ is an operator that includes each force field in eq. 19 only if the corresponding actor from the basal ganglia module is active. φ is obtained from eq. 6 for each higher order force field, and J from eq. 7.

Results

In the current section we present the results of the proposed model. We focus on the training of each pathway, as well as the model's ability to follow various different trajectories.

The first result we consider is the convergence of the least squares solution for the system of linear equations in eq. 9. Figure 6 presents the solution for the "up" higher order primitive, where it is evident that the least squares algorithm has converged to a good result. The three subplots at the bottom illustrate how the hand moves towards the "up" direction when this force field is active. Similar solutions were obtained for the other three primitives, where the least squares solution converged to 7 (left), 2 (right) and 5 (down) errors (the error represents the average deviation of the vectors in a field from the correct direction of the force).

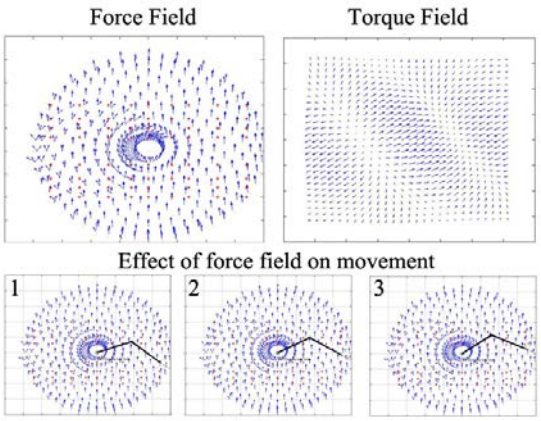

Fig. 6. The force field (upper left subplot) and torque field (upper right subplot) as converged by the least squares solution for the "up" primitive. The three subplots at the bottom show how the hand moves when the primitive is active.

The policy for reaching was learned during the initial imitation phase described previously. During this phase the robot performed the training trajectory, and was delivered a binary reinforcement signal upon successful completion of a whole trial.

Since the reward signal was only delivered at the end of the trial, the agent relied on the prediction of the reward signal elicited by the critic. In the following we look more thoroughly on the response properties of the simulated dopaminergic critic neurons and how the actors learned to activate each force field accordingly based on this signal.

Figure 7 illustrates how the critic neurons of the model learned to predict the forthcoming of a reward during training. In the first subplot (first successful trial) when reward is delivered at *t=4*, the prediction of the 1st critic is high, to indicate the presence of the reward at that time step. After the first 10 successful trials (Fig. 7, subplot 2), events that precede the presentation of the reward *(t=3)* start eliciting some small prediction signal. This effect is more evident in the third and fourth subplots where the prediction signal is even higher at *t=3* and starts responding at *t=2* as well.

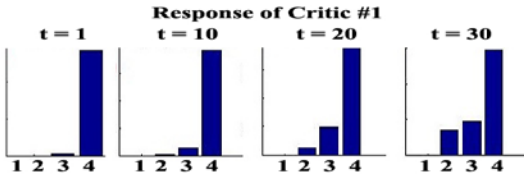

Fig. 7. The prediction signal emitted by the critic component of the model during the initial stages of the training (subplot 1), after 10 trials (subplot 2), after 20 trials (subplot 3) and after 30 trials (subplot 4).

The effects of this association are more evident in Fig. 8, where it is shown that, after training, even though rewards are not available in the environment, the neurons start firing because they predict the presence of a reward in the subsequent steps. Using the output of this prediction signal,

the actor, i.e. in the case of the model the F5 premotor neurons that activate the force fields in the MI, forms its weights in order to perform the required reaching actions.

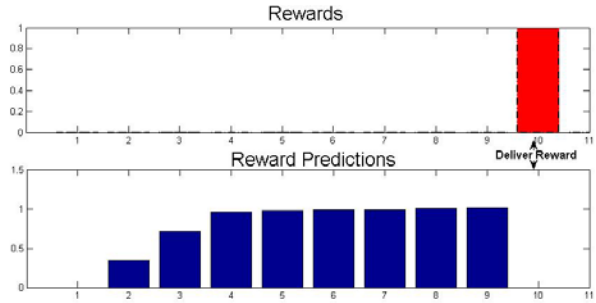

Fig. 8. The actual reward signal given to the robot at the end of a successful trial (upper subplot), and the reward predicted by the critic component after training (bottom subplot). The x-axis represents the 100ms time blocks of the simulation while the y-axis the values of the reward and prediction signals respectively.

The second part of the policy is for the model to learn to derive the distance of the end effector location from the current point in the trajectory. This is accomplished by projecting the output from the forward model and perception pathways in an LSM and using a readout neuron to calculate their subtraction. Having run several different simulations we found that to shape the liquid dynamics and learn this transformation the dynamic synapses must have delays of approximately 10ms. Since our model resolution was set to 100ms, we averaged the output of the readout neuron over the 10 steps of the simulation. In Fig. 9, we illustrate two sample signals as input to the liquid (top subplot), the output of the readout neuron in the 10ms resolution (middle subplot) and the averaged over the 100ms of simulation time output of the readout neuron (bottom subplot).

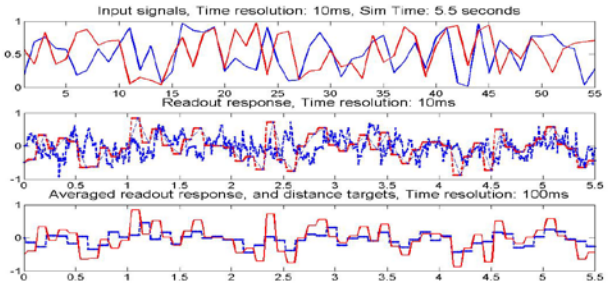

Fig. 9. The output of the distance LSM after training. The top plot illustrates two sample input signals of 5.5 seconds duration. The bottom two plots show the output of the neural network readout used to learn the subtraction function from the liquid (middle plot), and how this output is averaged using a 100ms window (bottom plot).

The whole simulation trial lasted 5.5 seconds. As the results show the liquid was able to extract the distance information with a good accuracy. Due to the local control laws used to implement the reaching policy, any small errors in the computation of distance are actually compensated in later steps.

Having established that the individual pathways/components of the proposed model operate successfully, we now turn our attention to the performance of the model in various reaching tasks. We note here that the model wasn't trained to perform any of the given reaching tasks, apart from the initial training/imitation period at the beginning of the experiments, shown in Fig. 5. After this stage the model was only given a set of points in a trajectory and followed them with very good performance. The first three trajectories we tested were variations of a straight line motion.

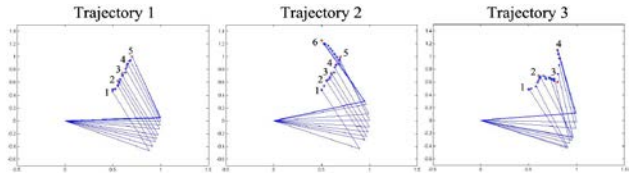

Fig. 10. Three trajectories shown to the robot (red points) and the trajectory produced by the robot (blue points). Numbers mark the sequence with which the points were presented.

As Fig. 10 shows the agent was able to follow all three trajectories quite precisely. The average normalized deviation of the agent's position from the points of the trajectory was 0.03 which shows that the resulting performance was satisfactory.

In order to evaluate further the performance of the model we used two more complex trajectories. The first required the robot to reach towards various random locations spread in the robot's workspace (Fig. 11, Trajectory 1) while the second complex trajectory required the robot to perform a circular motion in a cone shaped trajectory (Fig. 11, Trajectory 2). Figure 11 illustrates how the aforementioned trajectories were followed by the robot.

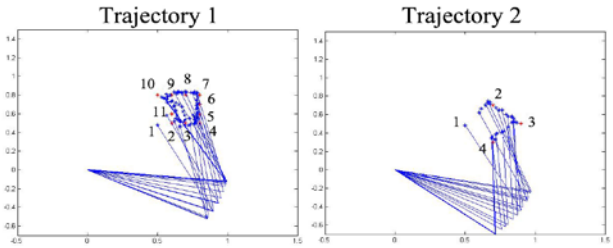

Fig. 11. Two complex trajectories shown to the robot (red points) and the trajectories produced by the robot (blue points). Numbers mark the sequence with which the points were presented.

To evaluate the performance of the model on any given path we created 100 random trajectories and tested whether the agent was able to follow them. Each of these random movements was generated by first creating a straight line trajectory (Fig. 12, left plot) and then randomizing the location of 2, 3 or 4 of its points; an example is illustrated in Fig. 12, right plot. The error was calculated by summing the overall deviation of the agent's movement from the points in the trajectory for all the entries in the dataset. The results indicate that the agent was able to follow all trajectories with an average error of 2%. This suggests that the selected model can confront with high accuracy any reaching task.

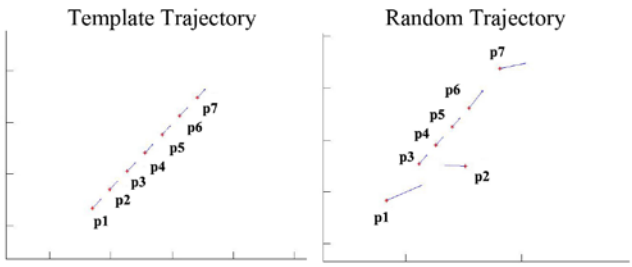

Fig. 12. The template used to generate the random test set of 100 trajectories (left plot) and a random trajectory generated from this template (right plot).

Conclusion

One of the important aspects of human skills is the ability to generalize knowledge to different domains and tasks. Using modularity and principles from neuroscience, in the current paper we investigated how adaptive learning skills can be acquired in a simulated agent that performs reaching tasks. One of the extensions that we plan for the presented model is to investigate how the primitive model can be designed to be adaptive, i.e. allow the agent to match the control of primitives to the properties of its body. In addition we will extend the current 2D plant model to its 3D equivalent by adjusting the equations of the Jacobian and primitive model. Moreover, we plan to investigate the role of the cerebellum in reaching movements, and its involvement in providing corrective feedback in respect to the global error of movement. Finally one of the important additions that we plan to investigate is how the agent developed in the current paper can be used during observational learning, i.e. improve its performance in reaching tasks without using its body. This extended model will be used to evaluate certain hypotheses regarding whether learning can be implemented in primates during observation.

References

Atkeson (1989). Learning arm movement kinematics and dynamics, *Annual Review of Neuroscience*, 12(1):157-183

Ballard D.H. (1986). Cortical connections and parallel processing: Structure and function, Behavioral and Brain Sciences, 9(1):67-90

Barto, A.G. (1995). Adaptive critics and the basal ganglia, Models of information processing in the basal ganglia, MIT Press, Cambridge.

Bizzi E., Mussa-Ivaldi, F.A. and Gizster, S. (1991). Computations underlying the execution of movement. A novel biological perspective, Science, 253(5017):287

Degallier, S. and Ijspeert, A. (2010). Modeling discrete and rhythmic movements through motor primitives: a review, Bio. Cyb. , pp.1-20

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

Dum, R.P. and Strick, P.L. (1991). The origin of corticospinal projections from the premotor areas in the frontal lobe, Journal of Neuroscience, 11(3):667

Giszter, S.F. and Mussa-Ivaldi, F.A. and Bizzi, E. (1993). Convergent force fields organized in the frog's spinal cord, The Journal of Neuroscience, 13(2):467

Hourdakis E. and Trahanias, P. (2011). Improving the performance of liquid state machines based on the separation property, submitted to the Engineering Applications of Neural Networks, EANN11, Corfu.

Hourdakis E. and Trahanias, P. (2009). A framework for automating the construction of computational models, CEC2009, Norway

Jeannerod, M., Arbib, M.A., Rizzolatti, G. and Sakata, H. (1995). Grasping objects, the cortical mechanisms of visuomotor transformation, Trends in Neurosciences, 18(7):314:320.

Joel, D., Niv, Y. and Ruppin, E. (2002). Actor-critic models of the basal ganglia: a new anatomical and computational perspectives, Neural networks, 16(4):535-547.

Kaas, J.H., R.J. Nelson, M. Sur, C.S. Lin, and M. M. Merzenich. (1979). “Multiple representations of the body within the primary somatosensory cortex of primates”. Science 204(4392): 521-523.

Maass, W. , Natschlager, T. and Markram, H. (2002). Real Time computation without stable states: A new framework for neural computation based on perturbations, Neural Computation, 14(11): 2531-2560

Markram H., Wang, Y. and Tsodyks, M. (1998). Differential signalizing viaa the same axon of neocortical pyramidal neurons, Proceedings of the National Academy of Sciences, 95(9):5323

Mountcastle,V.B.(1978). An organizing principle for cerebral function: The unit module, The mindful brain, MIT Press, Cambridge

Paul R. P. (1981). Robot Manipulators: Mathematics, Programming, and Control. Cambridge, Massachusetts: MIT Press

Piaget (1962). Dreams and imitation in childhood, Norton, New York

Plaut, D.C. and Hinton, G.E. (1987) Learning sets of filters using backpropagation, Computer Speech and Language, 2(1):35-61

Sakai, K., Hikosaka, O., Miyauchi, S., Takino, R., Sasaki, Y. and Putz, B. (1998). Transition of brain activation from frontal to parietal areas in visuomotor sequence learning, Journal of Neuroscience, 18(5):1827.

Sergio, L.E. and Kalaska, J.F. (2003). Systematic changes in motor cortex cell activity with arm posture during directional isometric force generation, Journal of Neurophsysiology, 89(1):212.

Soechting, J.F. and Flanders, M. (1992). Moving in three dimensional space: frames of reference, vectors and coordinate systems, Annual Review of Neuroscience, 15(1):167-191

Thach, W.T., Mink, J.W., Goodkin, H.P. and Keating, J.G. (2000). Combining versus gating motor programs: Differential Roles for the cerebellum and basal ganglia?, Cog. Neuros.: A reader, pp.366-375

Todorov, E. (2000). Direct control of muscle activation in voluntary arm movements: A model, Nature Neuroscience, 3(4):391-398.

Wolpert, D.M. (1997). Computational approaches to motor control, Trends in cognitive sciences, 1(6):209-216.