Representing reach-to-grasp trajectories using perturbed goal motor states

Jeremy Lee-Hand, Tim Neumegen, Alistair Knott

Dept of Computer Science, University of Otago, New Zealand

Abstract. In the biological system which controls movements of the hand and arm, there is no clear distinction between movement planning and movement execution: the details of the hand's trajectory towards a target are computed 'online', while the movement is under way. At the same time, human agents can reach for a target object in several discretely different ways, which have their own distinctive trajectories. In this paper we present a method for representing different reach movements to a target without reference to full trajectories: movements are defined through learned perturbations of the hand's ultimate goal motor state, creating distinctive deviations in the hand's trajectory when the movement is under way. We implement the method in a newly developed computational platform for simulating hand/arm actions.
Keywords: reach/grasp actions, touch receptors, reinforcement learning

1 Introduction: biological models of reaching, and the problem of action representation

In this paper, we consider how human infants learn to reach and grasp target objects in their immediate environment. Performing this task is complex, as the human hand/arm system has many degrees of freedom. For any given peripersonal target, the infant must issue a sequence of commands to the muscle groups of the shoulder, elbow, wrist and fingers which result in her hand achieving a stable grasp on the target. Theorists term the function which computes a sequence of commands the **motor controller**. The research question for us is how infants learn a motor controller which lets them reach and grasp target objects.

Traditionally, control theorists have construed the motor controller as two separate functions: one which plans a trajectory along which the effector must move, and one which computes the appropriate motor commands to issue at each point to cause it to describe this trajectory (see e.g. Jordan and Wolpert, 2000). However, there is a growing consensus that the biological motor controller does not work in this way; in the biological system, it appears that a detailed trajectory is only computed while the movement is actually under way (Cisek, 2005). Before the movement is initiated, the agent only computes a rough trajectory representation, which specifies little more than the basic direction the movement will be in. The main evidence for this suggestion is that the regions of the brain in which reach-to-grasp movements are prepared overlap extensively with those

involved in actually executing movements. The preparation of a visually guided reach-to-grasp movement involves activity in a neural pathway running from primary visual cortex and somatosensory cortex through posterior parietal cortex to premotor cortex (see e.g. Burnod *et al.*, 1999). When a prepared action is executed, these same pathways are involved in delivering a real-time signal to primary motor cortex, which sends torque commands to individual joints (see e.g. Cisek *et al.*, 2003; Cisek, 2005), bringing about motor movements. These movements create new reafferent sensory states, which provide updated inputs to the pathway, which in turn result in new motor commands, and so on, in a loop. In this conception of motor control, movements of the arm/hand are described by a complex dynamical system whose components are partly physical and partly neural: the task of the neural motor controller is to set the parameters of this system so that the attractor state it moves towards is one in which the agent's hand achieves the intended goal state. This model of motor control was first posited by Bullock and Grossberg (1988), and is now quite well established (see e.g. Hersch and Billard, 2006).

While there is some consensus that a detailed hand trajectory is not precomputed by the agent, at some level of abstraction human agents are clearly able to represent a range of different trajectories of the hand onto a target. For instance, when confronted with a target object, we can choose different ways to grasp it, which require the hand to approach from different angles. We can also choose to perform actions other than reaching-to-grasp, which have their own idiosyncratic trajectories. For instance, to 'squash' the target our hand must approach it from above, while 'slapping' the target requires an approach from the side; 'snatch', 'punch' and 'stroke' likewise have distinctive trajectories. If we do not compute detailed hand trajectories, how do we represent the discrete alternative grasps afforded by target objects, and how do we represent discretely different transitive actions like 'squash' and 'slap'? In this paper we propose an answer to this question, and investigate its feasibility in some initial experiments.

2 Representing reach-to-grasp actions using perturbed goal motor states

Any model of reaching-to-grasp needs to make reference to the concept of a **goal motor state**: the state in which the agent has a stable grasp on the target object, and which must function as an attractor for the hand-arm system as a whole. We first envisage the existence of a simple feedback controller, which works to minimise the difference between the current and goal motor states of the hand and arm by generating a force vector in the direction of the difference. This controller defines a 'default' trajectory of the hand towards the target, which will be suboptimal in many ways, but which provides a framework for specifying more sophisticated controllers.

Our main proposal is that on top of the basic feedback controller there is a more abstract motor controller which is able to generate arbitrary *perturbations* of the goal motor state provided to the feedback controller. A given perturbation,

applied for a specified amount of time, will cause a characteristic deviation in the hand's trajectory to the target. For instance, if the higher-level controller initially perturbs the goal motor state some distance to the right, the feedback controller will generate a hand trajectory which is deviated to the right.

Of course, appropriate perturbations, or perturbation sequences, must be learned. We suggest that infants learn useful trajectories bringing the hand into contact with a target object by exploring the space of possible perturbations of the goal hand state in which the hand is touching the object, and reinforcing those perturbation sequences which have successful consequences. The model we propose is a learning model, intended to simulate certain aspects of the motor learning done by infants.

Our perturbation model echoes a number of existing proposals. Oztop *et al.* (2004) present a model of infant reach-to-grasp learning in which infants learn 'via-points' for the hand to pass through on its way to the target. Their scheme conforms to the classical motor control model in which trajectories are first planned and then executed; they use a biologically plausible neural network to learn suitable via-points for reaches to a range of object locations, and then use techniques from robotics to compute the kinematics of a reach which brings the hand first to the appropriate via-point and then to the target. Via-points are specified in a motor coordinate system centred on the goal motor state, as our perturbed goal states are. However, in Oztop *et al.*'s model, the agent's hand actually reaches the learned intermediate state on its way to the target; in our model, the learned perturbation pulls the hand's trajectory in a particular direction, but the hand would not typically reach the perturbed goal state. Our model allows less fine-grained control over hand trajectory than Oztop *et al.*'s—of necessity, because it does not allow the precomputation of detailed trajectories.

Another notion of intermediate goal states is implemented in Fagg and Arbib's (1998) model of reaching-to-grasp. This model focusses on the grasp component of the reach, which maps a visually derived representation of the shape of the target onto a goal configuration of the fingers. It is important that the fingers are initially opened wider than this goal configuration, so that the hand can move to its own goal position close to the target. Fagg and Arbib model a circuit within the grasp pathway which operates simultaneously with the reach movement, which first drives the fingers to their maximum aperture, and then brings them into their goal configuration. In our model, we can think of the larger grasp aperture achieved during the reach movement as generated by a learned perturbation of the goal state of the fingers, rather than through a specialised circuit.

The idea that biological motor control involves a combination of a simple feedback controller and a more complex learned controller is fairly uncontroversial (see e.g. Kawato *et al.*, 1987). The learned controller is often modelled as a feedforward controller, which takes the current motor state and the goal motor state at the next moment and returns the command which produces this state. But this way of modelling it assumes a fully precomputed trajectory: since this assumption cannot be sustained, we have to find another way of representing

the learned aspect of motor control. Our model provides a way of describing how learning supervenes on basic feedback control which does not depend on the assumption of a precomputed trajectory.

3 A developmental methodology

It remains to be seen whether learned perturbations to goal motor states provide a rich and accurate enough way of representing motor movements. To answer this question, we need to propose mechanisms which learn perturbations, and examine their effectiveness. Our approach is to model an agent learning simple perturbations, which support the action of reaching-to-grasp. The agent is assumed to be an infant, learning its earliest reach-to-touch and reach-to-grasp actions. We assume a reinforcement learning paradigm: the infant learns motor behaviours which maximise a reward term, which is derived from tactile sensations (what Oztop *et al.*, 2004 call 'the joy of grasping'). The assumption is that early in infant development, touch sensations are intrinsically rewarding, so that the infant's behaviours are geared towards achieving particular kinds of touch.

Roughly speaking, infants first learn actions which achieve simple touches on target objects, and later learn actions more reliably achieve stable grasps. Before five months of age, infants' fingers do not move as they reach for objects, and hand trajectories (in Cartesian space) are relatively convoluted; at around five months, reaches reliably touch their targets, but grasps are only reliably achieved from around nine months (see e.g. Gordon, 1994; Konczak and Dichgans, 1997).

Our agent learns in two phases, which roughly replicate these two developmental stages. During Stage 1, the agent learns a simple function in the reach sensorimotor pathway, which transforms a retinal representation of target location into a goal motor state. This function is trained whenever the agent achieves a touch sensation anywhere on the hand. In this stage, the agent's movements are generated by a simple feedback controller, with no perturbations; when learning at this stage is complete, the agent can generate an accurate goal motor state for targets presented at a range of retinal locations, and can reliably reach to touch these objects (through somewhat suboptimal trajectories). During Stage 2, the agent learns a function which perturbs the goal motor state generated from vision during the early reach. This function is trained whenever the agent achieves a particular kind of touch—an 'opposition touch' where contact is felt simultaneously on the inner surfaces of the thumb and opposing fingers, or (better still) a stable grasp.

One point to mention about infant motor development is that infants' reach-to-grasp actions become *reliable* some time before they become *straight*. A hallmark of adult point-to-point hand movements is that they are straight in Cartesian space (see e.g. Morasso, 1996). While infant grasps are reliable from around 9 months, the hand trajectories produced during grasping only become straight at around 15 months (see again Konczak and Dichgans, 1997). Until recently, the straightness of adult reach trajectories was taken as evidence that agents explicitly precompute trajectories—and moreover, that they do so using Carte-

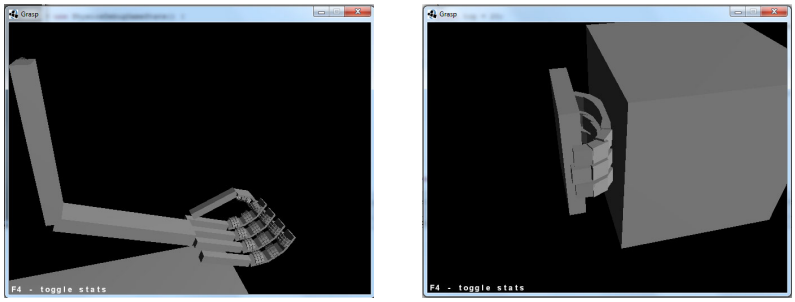
sian (or at least, retinal) coordinates. However, there are ways of generating straight Cartesian trajectories without precomputing them: as Todorov and Jordan (2002) have shown, if reach movements are optimised to minimise error, and movement errors are represented in visual coordinates, this suffices to produce straight reach trajectories. In our current simulations, we want to model the developmental stage when infants reliably reach, but have not yet begun to optimise their reaches using visual error criteria. We will briefly discuss how this optimisation might happen in Section 7.

4 A platform for modelling reach-to-grasp actions

We developed a new suite of tools for simulating hand/arm actions to investigate the model of reaching-to-grasp outlined in Section 2. There were two motivations for this. One was a desire to use a general-purpose physics engine to model the physics of the hand/arm and its interactions with the target. Grasp simulation packages often implement special-purpose definitions of 'a stable grasp'; we wanted to define a concept of stable grasp using pre-existing routines for collision detection and force calculation, to ensure that no unrealistic assumptions are built in. The second motivation was a desire to model the tactile system in more detail than most simulation environments allow. Our learning methodology assumes a gradation of tactile reward signals: we wanted to ensure that the signals we use correspond to signals which are obtained by the human touch system. These two motivations are in fact linked; the notion of a stable grasp should make reference to general-purpose physics routines, but must also make reference to the tactile system, because the agent's perception that a stable grasp is achieved occurs mainly through tactile representations.

4.1 Software components

Our simulation package makes use of a Java game engine called JMonkey (jme2, http://jmonkeyengine.com). JMonkey combines a physics engine called Bullet with the OpenGL graphics environment. The basic unit of representation in JMonkey is the **node**: a rigid body with a defined shape and mass; JMonkey calculates the forces on nodes, and OpenGL renders them graphically. The user can connect nodes together using various types of joint; Bullet uses rigid body dynamics to compute their movements, interpreting joints as constraints on movement. On top of this, our own simulation defines routines for constructing a simple hand/arm system, a simple model of the visual field from a fixed point above the hand/arm, a basic feedback controller which moves the hand/arm from its current state to an arbitrary goal state, a manual controller which allows a user to select goal states, and a collection of neural networks which support motor learning. All these will be described in more detail in the remainder of the paper. The software implementing the physics, graphics and motor controller is available at `http://graspproject.wikispaces.com`.



(a) (b)

Fig. 1: (a) The hand/arm system. (b) detail of a single finger pad

The hand/arm system we describe in this paper is illustrated in Figure 1a. It has five degrees of freedom: the shoulder can rotate in two dimensions, the elbow can bend, the forearm can twist, and the fingers of the hand can open and close. (A single parameter controls the angles of both thumb joints and all finger joints, so that fingers and thumb are straight when open and curled when closed.)

4.2 A model of soft fingers

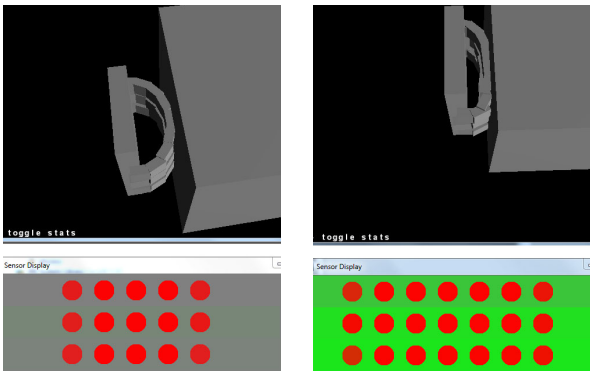
An important physical property of fingers is that they are not rigid: they deform if they make (forceful) contact with an object. There are many computational simulations of 'soft fingers'. One common solution is to model fingertips as spheres, and compute the deformations on these spheres which contacting objects would produce, using the result to calculate forces at the fingertips (see e.g. Barbagli *et al.*, 2004). But we implemented an alternative strategy inspired by a model of deformable objects from computer graphics, which represents a deformable object as a lattice of rigid cubes connected by springs (Rivers and James, 2007). This model allows us to obtain fine-grained tactile information from the fingertips: each object in the lattice delivers information about the forces applied to it. The structure of a single finger pad is shown in Figure 1b.

4.3 The touch system

The inputs to the human touch system come from neurons called **mechanoreceptors** in the top two layers of the skin (the epidermis and the dermis). There are several types of mechanoreceptor: the most relevant ones for grasping are **Meissner's corpuscles** which sense light touches which do not deform the skin, **Pacinian corpuscles** which sense firmertouches which deform the skin surface, **Merkel's cells** which detect the texture of objects slipping over the skin, and **Ruffini endings** which detect stretches in the skin. The latter two detectors combine to provide information about the slippage of objects the agent

is attempting to grasp: roughly speaking, a stable grasp is one where no slippage is detected at the points the hand is contacting the object, even when the arm is moved.

Our current implementation of touch sensors simulates Meissner's corpuscles by reading information about contacting objects from each link in each finger pad into an array of real-valued units (see Fig. 2a) and Pacinian corpuscles by reading information about the deviation of each link from its resting position (Fig. 2b). We simulate slip sensors by consulting the physics engine, and computing the



(a) (b)

Fig. 2: Sensor outputs for a single fingerpad light touch (a) and firm touch (b). Strength of contacting force for links in the pad is shown by the shade of circles; degree of deformation of a link is shown by the shade of a circle's background.

relative motion of each fingerpad link with any object contacting it.

4.4 A basic feedback motor controller

We assume a hardwired feedback motor controller. The one we implement is a proportional-integral-derivative (**PID**) controller (see Araki, 2006), which computes an angular force vector $u(t)$ based not only on the current difference $e(t)$ between the actual and goal motor states (given in joint angles) but also on the sum of this difference over time, and on its current rate of change.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

The integral term is to ensure the hand does not stop short of the goal state. The derivative term is to slow the hand's acceleration as it approaches the goal state, to minimise overshoot and help suppress oscillations around the goal state. The parameters K_p, K_i and K_d were optimised for a combined measure of reach speed and accuracy on a range of randomly selected target locations.

5 Stage 1: reaching-to-touch

The first developmental stage in our simulation involves learning a function which maps a retinal representation of target location onto a goal hand state. In this stage, the fingers of the hand are not moved; a 'goal hand state' is any state in which a touch sensation is generated anywhere on the hand.

The model proceeds through a series of trial reaches, each from the same starting position. In each trial, a horizontally oriented cylindrical target object is presented at a random location in reachable space (on average 40cm away from the hand), and its projection onto a simulated retina is calculated. The centroid of this projection is computed, to provide approximate x and y retinal coordinates of the object, and the physics engine is consulted to provide a retina-centred z (depth) coordinate (which in a real agent would be computed from depth cues, in particular stereopsis). These three coordinates are provided as input to a **reach network**, whose structure and training are described below.

Network structure The reach network is a feedforward neural network which takes a 3D retina-centred location as input and produces a 3D goal motor state as output. The network's output layer has three units, which represent goal angles for the two shoulder joints and the elbow joint, encoded using a localist scheme. Its structure is shown in Figure 3a.

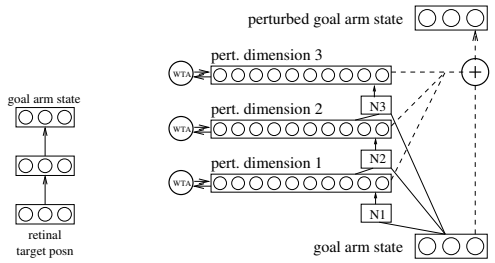


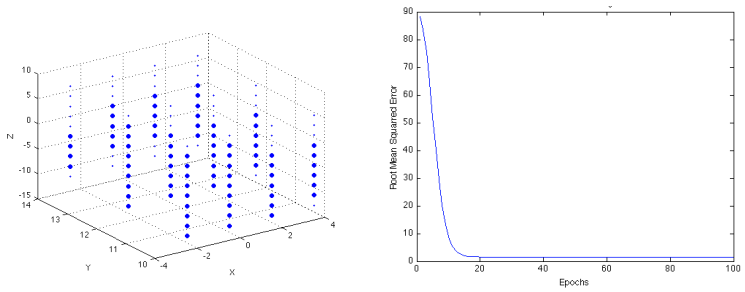
Fig. 3: (b) The reach network. (b) The arm perturbation network.

Training In each trial, the goal arm state computed by the reach network is combined with a noise term, and the resulting vector is given to the feedback controller, which executes a reach to this state. The noise term is initialised to a high value, and gradually annealed over trials, so the arm begins by reaching to random locations, and over time comes to reach accurately to the location computed by the network.

The reach network is trained every time a touch sensation is generated. During each frame of each trial, a touch-based reward value is calculated, which

sums touch inputs across all parts of the hand, including the back of the fingers and thumb as well as the pads. Each hand part contributes a touch value, and these values are summed to generate the reward value. The frame with the highest (non-zero) reward value is used to log a training item, pairing the visually-derived location of the target with the current motor state (the joint angles of the elbow and shoulder in that particular frame). The most recently logged training items are retained in a buffer (of size 175 in our experiments); the network is trained on all the items in this buffer after each trial in which a positive reward value is generated. The training algorithm is back-propagation (Rumelhart *et al.*, 1986). Each training item is also tagged with the magnitude of the reward, which is used to set the learning constant for that item, so that more learning happens for higher rewards.

Results After training for 300 trials using the above scheme, the agent learns to reach for objects at any location quite reliably. It achieves a successful touch on objects at unseen locations in 76% of tests, and in the remainder of tests, gets on average within 3.5cm of the target (SD=1.3cm). A plot showing the areas of motor space where successful touches are achieved is shown in Figure 4a, and a learning curve showing error performance is shown in Figure 4b.



(a) (b)

Fig. 4: (a) Coverage of the trained reach network, in motor coordinates. (Large dots denote touches, small dots denote misses.) (b) Root mean squared error (in cm) on unseen objects during training.

6 Stage 2: reaching-to-grasp

During the second developmental stage, the agent learns to generate a single perturbation to the visually computed goal motor state, which helps the hand achieve a more grasp-like touch on the target, and eventually a reliable stable grasp. The perturbation is computed by a second network, the **arm perturbation network**; it is applied at the start of the reach, and is removed when the hand reaches a certain threshold distance from the target.

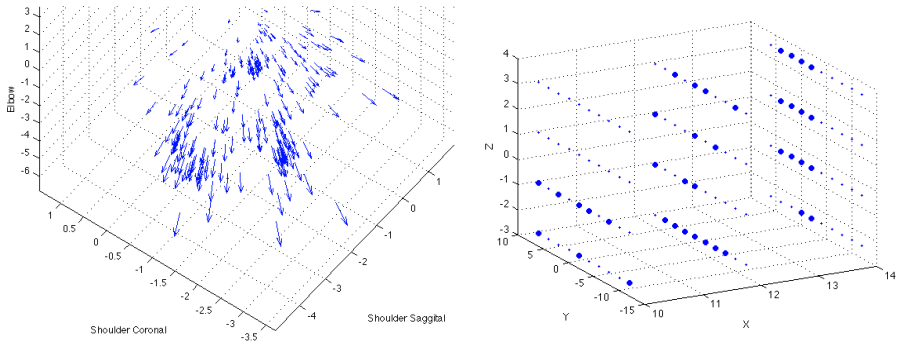
Network structure The arm perturbation network takes a goal arm state as input, and computes a perturbation of this state as output. Its structure is shown in Figure 3b. To allow the representation of multiple alternative perturbations (and therefore multiple alternative trajectories), the network uses population coding to represent perturbations. Each of the three motor dimensions in which perturbations are applied is represented by a bank of units, with Gaussian response profiles tuned to different values in the range of possible perturbations. The network can generate more than one value for a given dimension; in order to return a single value, each bank of units also supports a winner-take-all (WTA) operation, which chooses the most active population in the layer.

As shown in Figure 3b, the arm perturbation network comprises three separate subnetworks, N1–N3, which compute the three dimensions of the perturbation separately, in series. Each subnetwork takes as input the goal motor state, plus the dimensions of the perturbation which have been computed so far. Subnetworks apply the WTA operation before passing their results on, to ensure that decisions in subsequent networks reflect a single selected value. When all three dimensions have been computed, they can be added to the goal arm state. (Note that although perturbations are defined in relation to the goal arm state, it is still important to provide this goal state as input to the network. This is a matter of fine-tuning: the perturbations which should be applied when grasping an object at different points in space will be similar, but not identical.)

Training regime As in Stage 1, training happens in a series of reach trials to targets at different locations. In each trial, a target is presented and the reach network generates a goal arm state from visual information; the arm perturbation network then uses this goal state to compute a perturbation. The arm perturbation network's outputs are annealed with noise over the course of Stage 2, so that early trials explore the space of possible perturbations, and later trials exploit learning in the network.

As before, training data for the network is only logged to the training buffer when a tactile reward is obtained. But now rewards have to be more grasp-like touches: either opposition touches, with contact on both the thumb and opposing fingers, or (better still) a fully stable grasp, detected through the hand's slip sensors. Tactile rewards again vary on a continuum, and training items are tagged with their associated reward, so that the learning constant used by the training algorithm (again backprop) can be adjusted to reflect reward magnitudes.

Results During Stage 2 training, the arm perturbation network learns to produce trajectories which bring the target object into the open hand, since these states are those with the highest tactile rewards. The fingers of the hand are hardwired to close when contact is felt on the fingerpads. After training, the network was presented with objects at a grid of locations unseen during training. Figure 5a shows the perturbations applied for objects at each point in the grid, in motor coordinates. As can be seen, they vary continuously over the reachable space. Figure 5b shows the coverage of the network. There are regions of space where grasps are reliably achieved, but also regions where grasps are not achieved. Note



(a) (b)

Fig. 5: (a) Perturbations for each point in reachable space (in motor coordinates). (b) Coverage of the trained perturbation network, in motor coordinates. (Large dots denote successful grasps, small dots denote misses.)

that the reach network trained in Stage 1 continues to learn during Stage 2, so that by the end of this stage the goal arm state computed from vision is a state achieving a stable grasp on the target, rather than just a touch. As the goal arm state changes, so do the optimal perturbations which must be applied to it, so learning happens in parallel in the two networks during Stage 2.

7 Conclusions and further work

From the experiments reported above, there is some indication that learned goal state perturbations can support the learning of useful hand trajectories. But these results are quite preliminary; there are many questions which remain to be explored. For one thing, our current model does not simulate the grasp component of reaching-to-grasp in any detail. The human grasp visuomotor pathway maps the visually perceived shape of an attended object onto a goal hand motor state; we need to add functionality to Stage 2 to implement this. For another thing, there are gaps in our current network's coverage of reachable space, especially for the Stage 2 network. We need to investigate the interpolation and generalisation potential of the combined networks, to determine if the perturbation model is a practical possibility. We know that population codes are helpful in this regard, but we do not have a good idea of the magnitude of the learning task. Finally, as mentioned at the outset, our main goal is to represent the characteristic trajectories associated with high-level hand/arm motor programmes such as 'squash', 'slap' and 'snatch'. It remains to be seen whether perturbations can be learned which implement high-level motor actions like these.

References

Araki, M. (2006). PID control. In H. Unbehauen, editor, *Control Systems, Robotics and Automation Vol. II*.

Barbagli, F., Frisoli, A., Salisbury, K., and Bergamasco, M. (2004). Simulating human fingers: A soft finger proxy model and algorithm. In *Proceedings of the International Symposium on Haptic Interfaces*, pages 9–17.

Bullock, D. and Grossberg, S. (1988). Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review*, **95**(1), 49–90.

Burnod, Y., Baraduc, P., Battaglia-Mayer, A., Guigon, E., Koechlin, E., Ferraina, S., Laquaniti, F., and Caminiti, R. (1999). Parieto-frontal coding of reaching. *Experimental Brain Research*, **129**, 325–346.

Cisek, P. (2005). Neural representations of motor plans, desired trajectories, and controlled objects. *Cognitive Processes*, **6**, 15–24.

Cisek, P. and Kalaska, J. (2005). Neural correlates of reaching decisions in dorsal premotor cortex. *Neuron*, **45**, 801–814.

Cisek, P., Cramond, D., and Kalaska, J. (2003). Neural activity in primary motor and dorsal premotor cortex in reaching tasks with the contralateral versus ipsilateral arm. *Journal of Neurophysiology*, **89**, 922–942.

Fagg, A. and Arbib, M. (1998). Modeling parietal-premotor interactions in primate control of grasping. *Neural Networks*, **11**(7/8), 1277–1303.

Gordon, A. (1994). Development of the reach to grasp movement. In K. Bennett and U. Castiello, editors, *Insights into the reach to grasp movement*, pages 37–58. Elsevier, Amsterdam, New York.

Hersch, M. and Billard, A. (2006). A biologically-inspired controller for reaching movements. In *Proceedings of the 1st IEEE/RAS-EMBS Intl. Conference on Biomedical Robotics and Biomechatronics*, pages 1067–1072, Pisa.

Jordan, M. and Wolpert, D. (2000). Computational motor control. In M. Gazzaniga, editor, *The New Cognitive Neurosciences*, pages 71–118. MIT Press.

Kawato, M., Furawaka, K., and Suzuki, R. (1987). A hierarchical neural network model for the control and learning of voluntary movements. *Biological Cybernetics*, **56**, 1–17.

Konczak, J. and Dichgans, J. (1997). The development toward stereotypic arm kinematics during reaching in the first 3 years of life. *Experimental Brain Research*, **117**(2), 346–354.

Morasso, P. (1996). Spatial control of arm movements. *Experimental Brain Research*, **42**, 223–227.

Oztop, E., Bradley, N., and Arbib, M. (2004). Infant grasp learning: a computational model. *Experimental Brain Research*, **158**, 480–503.

Rivers, A. and James, D. (2007). FastLSM: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graphics*, **26**(3), Article 82.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *PDP Vol. 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, USA.

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