Technical Report OUCS-2014-01

A neural network model of causative actions

Authors:

Jeremy Lee-Hand, Alistair Knott
Department of Computer Science, University of Otago, New Zealand

A neural network model of causative actions

Jeremy Lee-Hand, Alistair Knott
Department of Computer Science, University of Otago, Dunedin, New Zealand
∗ E-mail: Corresponding alik@cs.otago.ac.nz

Abstract

In this paper we present a neural network model of motor learning and motor control that learns a class of actions termed *causative actions*. A causative action is an action that brings about a specified effect or movement in a target object: for instance the action of causing a lever to bend, or of causing a door to open. The network comprises a layered set of motor learning circuits that associate motor actions with their sensory effects, as proposed by Hommel *et al.* [1]. The circuit for learning simple manual actions is trained by sensory representations in the haptic modality that function as rewards, as previously proposed by Oztop and Arbib [2]. We propose that the circuit responsible for learning causative actions makes similar use of sensory representations as reward signals. The key novel idea is that the sensory representations in this case come not from the tactile system, but from a high-level perceptual module that registers arbitrary movements taking place in external objects in the world.

Introduction

An agent executing a motor action produces bodily movements. In most cases, these movements of the body are not ends in themselves; instead they bring about particular effects that are desired by the agent. One interesting class of actions are those whose effect is to bring about movements *in other objects*. For instance, an agent opening a door performs an action that causes *the door to open*; an agent bending a lever performs an action that causes *the lever to bend*. We will term these actions 'causative actions'. They are interesting because they require a distinction between the agent executing the action and the object undergoing the action. In simple actions, there is no distinction: for instance, when a man *grasps* a cup, the action takes place in the man, not the cup. But when a man *bends* a lever, it is the lever that bends, not the man. Causative actions have been intensely studied by linguists (see Schäfer [3] for a review). However they have not received much attention from motor neuroscientists: we do not know much about how causative actions are learned, or represented, in the motor system. In this paper we present a model of the learning and representation of causative actions, expressed within a neural network model of motor control and motor learning.

The starting point for our model is the well known proposal that an agent's motor programs are encoded in a way that makes reference to the sensory effects they bring about. This idea is particularly associated with Prinz's [4] theory of 'common coding' and Hommel *et al.*'s theory of of 'event codes' [1]. The key idea in these theories is that motor programs are not defined purely within the motor domain: in addition their neural representation includes a representation of the effects they are expected to have on the world, as apprehended by the perceptual system. Before we introduce our model of causative actions, we will outline some arguments for this view of action representations.

Theoretically, a model in which action representations make reference to their perceptual effects can be motivated from considerations about how actions are learned. Motor programs are learned through reinforcement. A reinforcing signal is ultimately a sensory signal of some kind. When an agent executes a motor program and receives a rewarding signal, the agent learns an association between the sensory signal and that particular program. After training in a range of contexts, the rewarding sensory signal will become associated with a range of related motor movements, which bring it about in different ways or under different circumstances, perhaps in ways which are parameterised or organised by features of the sensory stimulus. If the learned associations are bidirectional, then after training the agent can activate the sensory signal as a goal, and trigger a motor movement that actually brings about this signal (or at least, that brought it about during training). Importantly, this manner of activating motor progammes groups them according to their associations with reafferent perceptual stimuli, rather than according to their properties as physical movements. Two movements which are physically similar but are associated with different sensory effects will be classified as different categories of movement. In summary, we expect action categories learned through reinforcement to be organised around sensory representations.

Experimentally, the idea that action representations make reference to their sensory effects has been supported in several ways. One set of experiments relate to the well-known stimulus-response compatibility effect (Simon [5]). In the classical experiment, subjects responding to a stimulus with a hand movement are slower to respond if the spatial location of the stimulus is incompatible with the responding hand, even when the location of the stimulus plays no role in the task. In Hommel's [6] variant of this experiment, the tone of an auditory stimulus indicated whether subjects should press a button with their left or right hand. The stimulus was presented to the left or right, as a distracting factor, as in the classical study. In addition, button presses generated a reafferent visual stimulus (a light), appearing ipsilaterally or contralaterally to the hand pressing the button, to explore whether the compatibility effect operates in the domain of motor movements or that of their sensory consequences. Hommel found that the stimulus-response compatibility effect depended on compatibility with the perceptual effects of button-presses, rather than on the hand which was used. This shows that the way subjects encode actions makes some reference to their sensory consequences—at least enough to interfere with stimulus-response mappings.

Neurophysiological studies also provide evidence for effect-based representations of motor actions. For instance, Umiltà *et al.* [7] observed the activity of neurons in premotor area F5 of monkeys performing grasp actions with specially constructed tongs. F5 neurons respond to a range of grasp movements made by the hand (see e.g. Rizzolatti *et al.* [8]). But under normal circumstances, executed grasp movements correlate strongly with visual signals. The experiment was designed to decouple motor movements from their observed effects. In one condition monkeys used a regular pair of pincers, which closed when the monkey squeezed. In another, they used reversed pincers, which opened when the monkey squeezed, and closed when they relaxed their grip. Most F5 neurons responded as a function of the movement of the pliers rather than the movement of monkeys' hands. This is evidence that many neurons in this grasp-planning area encode the effects of motor movements, rather than their properties as motor movements. Further evidence for effect-based neural representations of motor actions comes from studies by Matsumoto *et al.* [9] and Moore and Obhi [10].

In summary, there is good evidence that both human and nonhuman agents' representations of motor actions make reference to their perceptual effects. But there are many open questions about the nature of these effect-based action representations. How does the brain represent the relation that links a motor action with an effect? It is presumably a learned association of some kind, as just discussed, but we do not yet know much about the nature of the association. The question is particularly pressing for causative actions. In this case, the motor system must keep two distinct actions separate: a motor action of the agent, and a perceived action of the target object. It must also represent the relation between these actions, which seems complex and action-dependent. In some cases the duration of the causative action is identical to that of the caused action (as in the action of crumpling paper); in others the caused action can continue after the causative action has ceased (as in the action of rolling a ball across a room). In some cases the manner in which the hand approaches the target is important (for instance to break a target, the hand may have to approach with a certain speed); in other cases it is less important. In our neural network model, we will propose an account of the learning and representation of causative actions that addresses these questions.

Methods

Our simulations were conducted using the GraspProject environment produced by Tim Neumegen [11]. The enviroment and the rigid-body arm that was used in our simulations are shown in Figure 10. There

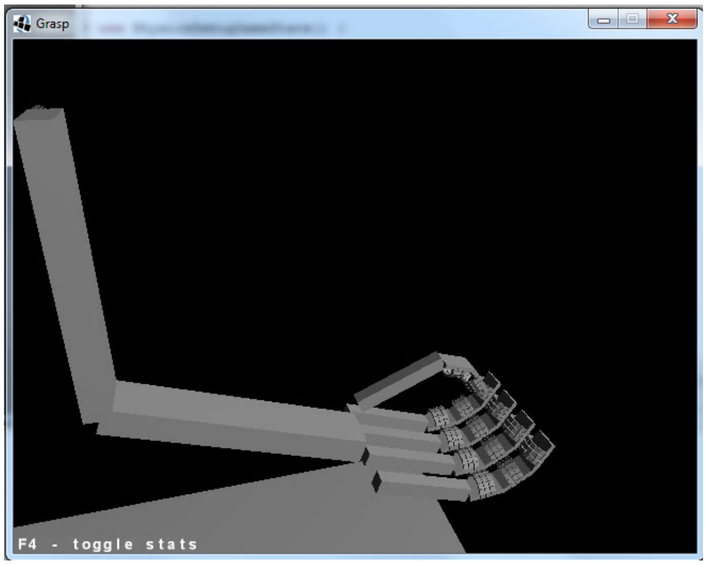

Figure 1. The GraspProject environment. Shown is the arm model that is used in our simulations.

are three degrees of freedom in the arm (two at the shoulder and one at the elbow) and one in the hand (controlling grip aperture). The simulation includes a novel implementation of ‘soft fingers’, which are modelled as deformable grids of plates connected by springs. High-resolution information about light touches is provided by collision detectors on each plate, and information about stronger touches which deform the surface of finger pads is read from the joint angles between adjacent plates. There are also sensors on the back of the hand and fingers, that deliver much lower-resolution information, and are less sensitive.

We built several objects in the simulation environment which could serve as targets for hand actions. One is a simple rigid object (a cylinder) which serves as the target for three simple motor actions: grasping, punching and slapping. Three others are articulated objects which can undergo various changes in interal configuration. One is a lever which can pivot around a joint, and can be bent; one is a hinged door in a plane, which can be pushed open; one is a pair of horizontal plates connected by a spring, which can be ‘squashed’ by pushing down on the top plate. These objects are illustrated in Figure 11.

Architecture of the motor control network

Our model of the motor system is a neural network for learning hand actions directed at target objects. It provides a simple model of some aspects of infant motor development. The general architecture of the network is shown in Figure 12. It consists of three sub-networks arranged in sequence. These are assumed to be trained at three successive developmental stages, by reward signals of different degrees of complexity. In this scheme, the system is initially rewarded by very simple sensory signals, which train a simple motor circuit, but as learning takes place in this circuit, more complex reward signals become available, which in turn train higher-order circuits. In this section we will introduce each network in turn; their training and evaluation are described in more detail in Training and Results.

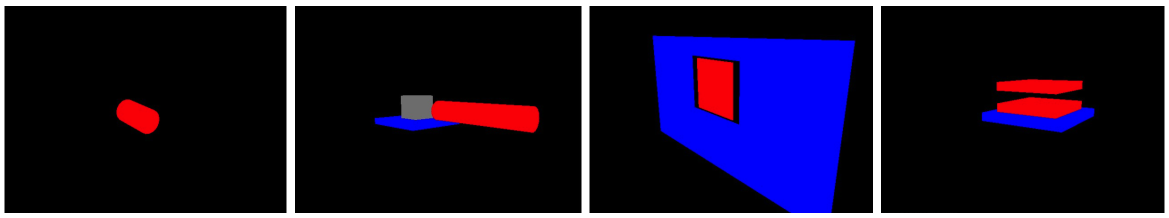

Figure 2. Objects created for the simulations. From left: a cylinder (for grasping, punching and slapping); a lever (for bending); a door (for opening); a compressable object (for squashing).

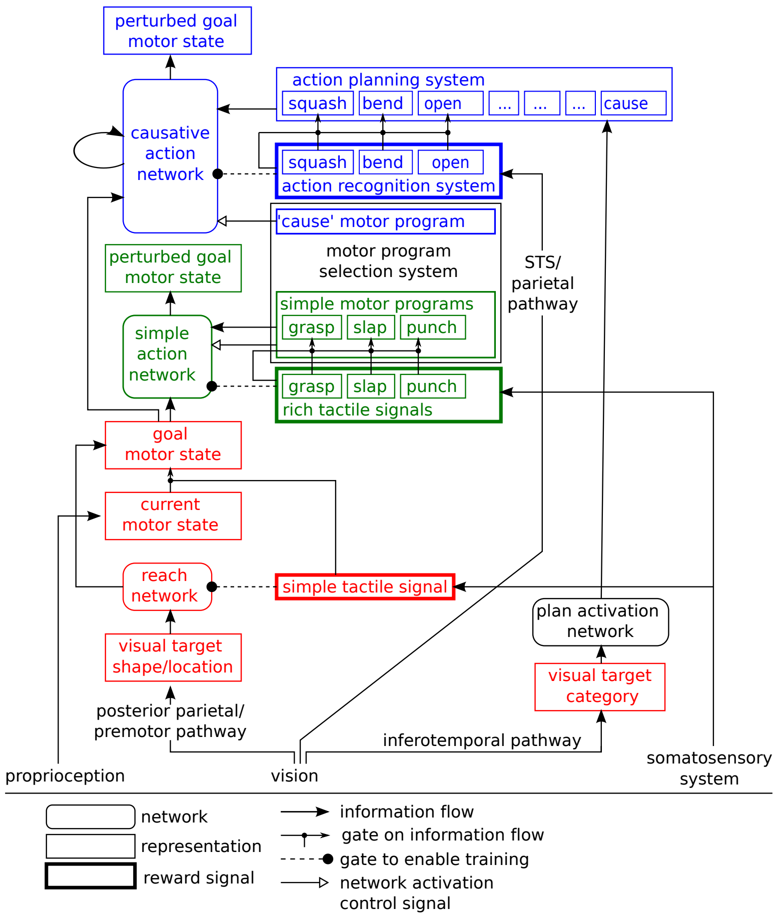

Figure 3. Architecture of the motor control network.

The first network to be trained is called the **reach network** (see the red part of Figure 12, and Figure 13). This network learns a function which maps a visual representation of the location of the target object onto a goal motor state of the hand and arm. The visual representation is a point in a three-dimensional visual coordinate system: two dimensions come from the visual projection of the object; depth is computed directly from the physics engine. The goal motor state is a point in three-dimensional

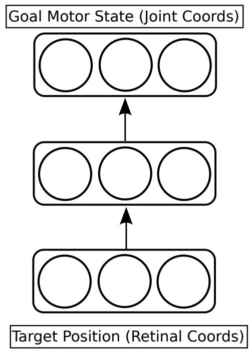
Goal Motor State (Joint Coords)
Target Position (Retinal Coords)

Figure 4. Detailed architecture of the reach network

joint space: each dimension corresponds to one of the degrees of freedom of the arm.[1]

In each training run a target object (the cylinder) is placed at a random position, and the agent executes a hand/arm movement towards a goal motor state selected by the reach network. The network's outputs are annealed with noise, that is gradually reduced to zero during training, so initially these movements are essentially to random locations. Sometimes a movement results in the hand touching the target, evoking a touch signal (the simple touch signal). This signal is intrinsically rewarding (as in Oztop *et al.* [2]). The touch signal has two functions. First, it causes a proprioceptive representation of the agent's current motor state to be copied into the medium holding its goal motor state (see the gating link terminating on the connection between the current and goal motor states in Figure 12). Second, it causes the reach network to be trained (see the gating link terminating on the reach network), so that the current visual representation of the target object is mapped onto this newly specified goal motor state. This training encourages the network to generate an approporiate motor goal when a target appears in the same position in the future.

This simple circuit implements a particular version of Hommel *et al.*'s model of event codes. Learning in the circuit creates what can be thought of as a single simple action category, associated with the sensory representation of a touch to the hand: after training, when the reach network is presented with the visual location of a target object, it will activate a motor goal which when achieved will reliably elicit this sensory representation. Motor goals in the circuit are associated with sensory stimuli in three ways. Any representation in the motor goal medium is implicitly associated with one particular reward stimulus (a simple touch sensation). Specific motor goals are associated axiomatically with specific motor states (sensed proprioceptively) when the reward stimulus is evoked. And specific motor goals are also associated through learning with arbitrary sensory stimuli (in this case visual), which carry information about the motor states associated with reward signals. Again this happens at the time the reward stimulus is evoked. The key devices in the circuit are reward-gated copy and learning operations. These devices are replicated in the other two networks.

The reach network generates a motor goal—but of course there must also be a mechanism which achieves this goal. In our current model, we assume this mechanism is a simple feedback motor controller. This controller takes the current motor state and the goal motor state and generates a motor signal proportional to the difference between them, in a direction which reduces this difference. (The controller is not shown in Figure 12.) A feedback controller does not need to be trained; it is a simple circuit, which is present innately in many motor systems (see e.g. Kawato *et al.* [13]). (We use a PID controller; see

[1]The hand joint controlling grip aperture is controlled directly by tactile stimulation: a touch on the inner surface of the fingers causes the hand to close reflexively, mimicking the palmar grasp reflex found in infants (see e.g. Schott and Rossor [12]).

e.g. Araki [14]). However, mature motor control involves a mixture of hardwired feedback control and learned *feedforward* control (see again Kawato *et al.* [13]). Feedforward control exploits learning about the properties of the agent's motor system to optimise action trajectories. If we think of the feedforward controller in sufficiently general terms, we can say that it is through learning in this controller that an agent can acquire a repertoire of different action categories. Different actions (like grabbing or punching or slapping) have different characteristic trajectories of the hand and fingers; the feedforward control system somehow learns about the distinct effects of particular trajectories and creates action categories associated with each. However, it is not clear how different trajectories are represented in the biological motor control system. There is good evidence that agents do not compute detailed trajectories in advance; these are only generated 'on the fly', as an action is actually underway (see e.g. Cisek [15]). Our network implements a particular idea about how trajectories are represented. We assume that the agent evoking a goal motor state can generate learned *perturbations* of this goal state as an action is under way, which deviate the hand from the normal course it would take under simple feedback control. For instance, to generate a trajectory bringing the hand onto the target from above, the goal state could be temporarily perturbed to a point above the target, so the hand initially moves higher than it would normally do. This idea is discussed in more detail and evaluated in Lee-Hand *et al.* [16]. The circuit that learns how to apply perturbations to the goal motor state is the second network in our model, the **simple action network** (see the green part of Figure 12, and Figure 14).

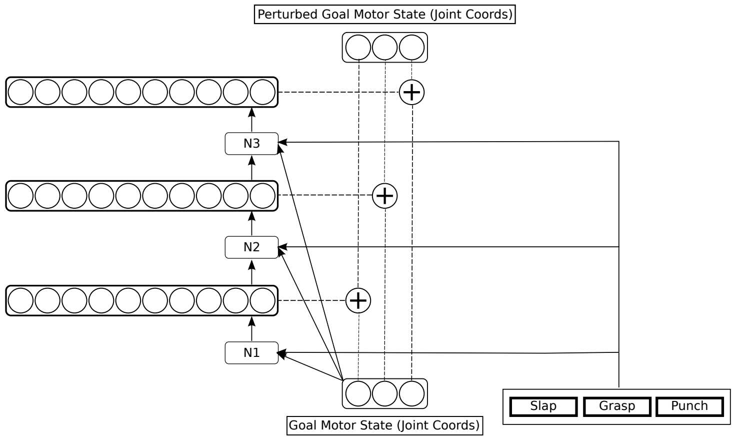
Perturbed Goal Motor State (Joint Coords)
N3
N2
N1
Goal Motor State (Joint Coords)
Slap
Grasp
Punch

Figure 5. Detailed architecture of the simple action network

Training in the simple action network begins when the reach network reliably generates actions that lead to simple touch signals. The simple action network learns to generate perturbations to the goal motor state computed by the reach network, which cause the hand to follow particular trajectories onto the target. These trajectories are associated with different simple action categories, like grasping, punching and slapping. As input the network takes a motor program specifying a particular action category, and also the goal motor state itself (as the perturbation required may be subtly different depending on the location of the target). The perturbation computed by the network is applied at the start of a reach action and removed when the hand is at a specified distance from the target.

While the simple action network is trained by *any* tactile signals, the reach network is trained by specific *rich* tactile signals. Each action category is associated with a particular type of rich tactile stimulus: the stimulus produced by a grasp is different from that produced by a slap, and different again from that produced by a punch. The tactile sensors on our simulated hand can detect these differences. Following Hommel *et al.*, the way our network learns is by associating these distinct patterns of tactile feedback with distinct action categories.

To train the simple action network, a simple rigid object (a cylinder, illustrated in Figure 11) is placed in a random location, oriented horizontally, generating a visual representation for the system. The reach network computes a goal motor state from this visual representation; this is passed as input to the simple action network, which produces a perturbation of this goal state. The perturbation is annealed with noise, which is again progressively reduced to zero during training. The feedback controller moves the hand towards the perturbed goal state; when it attains a certain distance from the target, the perturbation is removed, and the hand approaches the actual goal state.

Learning opportunities occur when, from time to time, the perturbation applied results in richer tactile reward signals than those used to train the reach network. Different rich tactile signals result from particular perturbations. Some perturbations result in a grasp or near-grasp, which generates one class of tactile stimulus. Others result in slapping movements, which generate another, different, class of tactile stimuli, or in punching movements, which generate another distinct class of tactile stimuli. (These rich stimuli are almost never generated through pure feedback control, because they result from special trajectories.) When a rich tactile stimulus is generated, copy and learning operations take place in the simple action network which are analogous to those in the reach network. First, the tactile stimulus is copied to an area holding ‘motor programs’. Second, the simple action network is trained to map the current goal motor state, *plus the currently active motor program*, onto the perturbation which resulted in the reward. After this learning, activating a specific motor program will generate an action with a characteristic trajectory, resulting in a characteristic rich tactile stimulus. We envisage motor programs competing with one another, with a single winner being selected.

In the simple action network, the three motor components of a perturbation are computed one by one, in the three networks labelled N1, N2 and N3 in Figure 14. This is because there are typically several possible perturbations which result in any given tactile reward signal: the network needs to select one of these, and selection of the different components of a perturbation cannot be performed independently. So the network N1 computes the alternative possible values for the first component of the perturbation, then selects one of these, and passes the selected value to the network N2 as input, and N2 performs a similar operation. In each network, ten output neurons are used to encode this distributed coding with each representing a particular discrete value in the range of possible joint angles. The neuron with the highest activation is chosen to be the ‘best’ motor program and the activation of the nearby neurons is used to alter the value slightly. All of the neurons outside of this region are inhibited and are not considered when decoding this perturbation component.

Note that the simple action network must execute in parallel with the simple reach network. It modulates the behavior of the simple network, in a manner reminiscent of Brooks’ [17] subsumption architecture. In order to execute a simple motor program, it is important that the whole simple action circuit is enabled, or turned on. Accordingly, while different motor programs provide different input to the simple action network, they also uniformly generate a control signal to enable the network they provide input to. This control signal is shown in Figure 12 by the unfilled arrow leading from the simple motor program medium to the simple action network.

The final network to be trained is the **causative action network** (see the blue part of Figure 12, and Figure 15). This network implements our model of causative actions. Our key proposal is that above the simple action network there is a higher-level network trained from still more sophisticated sensory signals, which derive not from the tactile system, but from a high-level perceptual module which can classify arbitrary actions taking place in the external world, relying on vision rather than touch. There is a well-studied perceptual module of this kind in the brain, implemented in a pathway from sensory cortices (in particular visual cortex) through the superior temporal sulcus (STS) and inferior parietal cortex to the premotor cortex (see e.g. Keysers and Perrett [18]; Iacoboni *et al.* [19]). When an agent allocates attention to an external object, representations in this pathway encode the actions *of this object*, in increasingly complex ways. This action classification pathway is normally thought of as being engaged when an agent is passively observing the external world. But consider what happens when the agent

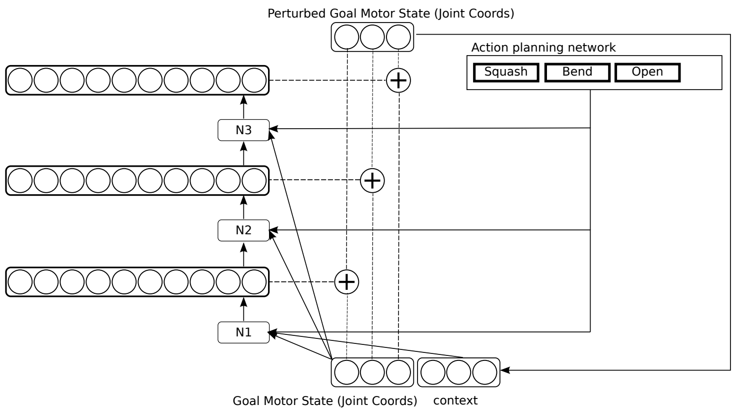
Perturbed Goal Motor State (Joint Coords)
Action planning network
Squash
Bend
Open
N3
N2
N1
Goal Motor State (Joint Coords)
context

Figure 6. Detailed architecture of the causative action network

is attending to an external object *as a target*, while directing the hand towards it along a particular trajectory. Any actions regularly evoked in the action recognition pathway in this scenario are likely to be actions *caused by the hand's movement.* We propose that during action execution, action signals evoked in the action recognition pathway are hardwired to function as reward signals, which train the causative action network to bring about particular distal actions in the world.

Training in this higher-level motor circuit involves presenting different articulated objects to the system, which can undergo various different changes in configuration. Objects are of three types: One is a lever that can pivot around a joint, and can be bent; one is a hinged door in a plane, which can be pushed open; one is a pair of horizontal plates connected by a spring, which can be 'squashed' by pushing down on the top plate. These objects are illustrated in Figure 11.

Training the causative action network again proceeds by random generation of perturbations to the goal motor state delivered by the reach network. In this circuit, *sequences* of perturbations are applied, to generate still more complex trajectories. (This is depicted in Figure 15 by a recurrent input, though in our implementation we 'unroll' this recurrence and generate exactly two perturbations.) Some of these sequences cause particular patterns of movement in the target object, which are interpreted as external actions by the action recognition system. Activation of an action representation in the action recognition system when performing an action on a target object is hard-wired to generate a reward signal. This signal has two effects. First, the observed action is copied to a medium in which action plans are held (the action planning system). Second, the causative action network is trained to map the basic goal motor state delivered by the reach network onto the sequence of perturbations which led to reward. Note that the network also takes representations in the action planning system as input. After training, the causative action network can take a simple goal motor state, plus an action representation in the action planning system, and generate a sequence of perturbations which will lead to observation of the planned action on the attended target.

This network enables a rich repertoire of actions to be learned. It preserves Hommel *et al.*'s idea that action representations are organised around their perceptual effects. But since the action recognition network generates rich, high-level perceptual signals, a correspondingly rich set of motor programs can be established. At the same time, the basic mechanisms through which learning happens are the same as in much simpler motor learning systems.

Part of the design of the causative action circuit is that 'cause' is a motor program in its own right, which competes within the motor program selection system against regular motor programs like 'grasp' and 'slap'. One important difference is that the 'cause' action enables the causative actions network rather than the simple action network, but other than that it counts as a regular motor program. This raises some important questions about how causative actions are planned and executed. When an agent decides to perform a causative action, presumably he has some particular caused action in mind. But at the time of planning, this caused action is in the future: minimally, the agent must bring his hand into contact with the target object before he can cause it to move in any way. In order to cause a particular action in a target object, the trajectory of the hand towards the object must often be biased from the very start: for instance, to cause an object to squash, the hand must approach the target from a particular direction, and with particular force. So the movements which bring about the caused action must be initiated some time before the action is perceived.

Our way of addressing this issue in the network is to activate the motor correlates of perceived actions in the medium holding *planned actions*, rather than in the medium of regular motor programs like 'grasp' and 'slap'. An underlying assumption in our model is that an agent brings about actions through planned sequences of sensory or motor operations (for details see Knott [20]). We also assume that planned sequences are selected as wholes, and that the component actions in a planned action sequence are active in parallel in the working memory medium where actions are planned. (This assumption is well supported by single-cell recordings in monkeys; see e.g. Averbeck *et al.* [21].) When the causative actions network is exploring causative actions, it will activate the 'cause' motor program experimentally, and choose a random sequence of perturbations. In some cases, this results *some time later* in activation of an action in the action recognition system: say 'squash'. This observed action activates a corresponding planned action. Additionally the **plan activation network** (see the bottom right of Figure 12) learns that the sequence 'cause', 'squash' is a good one to execute on the category of object currently present, so that when a similar object is presented in future, it will tend to activate this planned sequence. Now consider what happens when the planned sequence is executed. The agent first executes the motor program 'cause'. This enables the causative action network, which generates a sequence of perturbations. Crucially, the causative action network also takes input from the planning medium in which the caused action ('squash') is active as part of the planned sequence. So as soon as it is initiated, the network is configured to generate the perturbation sequence which led to the caused action, even before this action actually occurs.

The key mechanism enabling causative actions to be executed is one which activates a sensory representation (the squash action) *as a goal* some time before it is evoked as a sensory stimulus. Note that something very similar happens in the other networks; for instance in the reach network the actual motor state where the touch sensation occurs is activated as a goal motor state. In the simple network this activation is possible because visual perception provides information about reward-associated motor states. In the higher-level causative actions network, the advance notification of reward comes from the working memory system which stores prepared actions. But the effect is much the same.

Training

The first network that is trained is the reach network. A single object (the cylinder) is presented in each training trial in any location within the space reachable by the arm. The retinotopic location of this object is computed, and provided as input to the reach network, which generates an output goal motor state. Initially this output is annealed with noise, so the goal motor state is essentially random. A feedback controller then brings the hand towards the goal motor state. If the hand happens to make contact with the object (i.e. if a tactile signal is received), the current motor state is logged as training data for the reach network, paired with the retinotopic location of the object. After each trial, the reach network is trained on all the training data logged so far. During learning, the noise applied to the network's output is progressively reduced to zero. The training algorithm is displayed in more detail in Algorithm 1.

There is one further point to make about the training of the reach network (and those that follow). It is of course unrealistic to assume a storage medium where a large number of training items can be logged. In a more plausible online learning scheme, the network being trained would interleave self-generated pseudo-training items with new training data arriving sequentially (see e.g. Robins [22]). To

Algorithm 1: Learning Goal Arm states for reach actions

```
Data: $o = (o_x, o_y, o_z)$ (object centroid in retinal coordinates),
      $\theta = (\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles)
begin
    $o$ input into reach network to produce $\theta_g = (\theta_{g1}, \theta_{g2}, \theta_{g3})$ (goal
    motor state);
    Random noise added to $\theta_g$;
    while Maximum time not exceeded do
        Calculate force applied to arm using PID controller (a function
        of $\theta - \theta_g$);
        if Tactile feedback occurs then
            Store touch score paired with $\theta$;
    if Touch data recorded then
        Log the maximum touch score and corresponding $\theta$, paired
        with $o_r$, as a new training item for the reach network;
        Maximum possible random noise reduced;
        if number of training items exceeds 200 then
            Discard oldest training item;
        Reach network trained on training data;
```

approximate an online scheme, we do however impose various constraints to ensure that the quality of stored training data improves during the course of training. Firstly, the learning rate used by the network for any given logged training item is a function of a **score** associated with the haptic feedback pattern that the hand received (recall some patterns are better than others). This means that trajectories which result in higher scores influence the behavior of the network more than trajectories producing lower scores. In addition to this, there is a minimum threshold score needed in order for an action to be logged as training data, which is increased as learning proceeds. Finally, we only retain the most recent 200 logged training items to use for training.

The next network to be trained is the simple action network. In each training trial a single object (the cylinder) is presented in one of a small number of training positions, as shown in Figure 16(a). The trained reach network generates a goal motor state as usual; this is passed as input to the simple action network, along with a randomly selected simple action category (grasp, slap or punch). The output of the network is a perturbation, which is applied to the goal motor state. This output is again annealed with noise, which is reduced to zero as training progresses. The motor contoller then moves the hand in the direction of the perturbed goal motor state, and when the perturbation is removed, in the direction of the actual goal motor state. If the resulting movement activates one of the predefined classes of rich tactile signal, a training item is logged, mapping the actual goal motor state providing input to the network, plus the simple action category corresponding to the activated tactile signal, onto the perturbation that was applied. After each trial, the network is trained on all the logged training items. This training algorithm is shown in Algorithm 2.

Finally the causative action network is trained. In each training trial either the squashable, bendable or openable object is presented in one of the locations given in Figure 16(a). The plan activation network maps the object's category onto a planned sequence of two actions in the action planning system. The causative action network maps the the current goal motor state and the planned action sequence onto

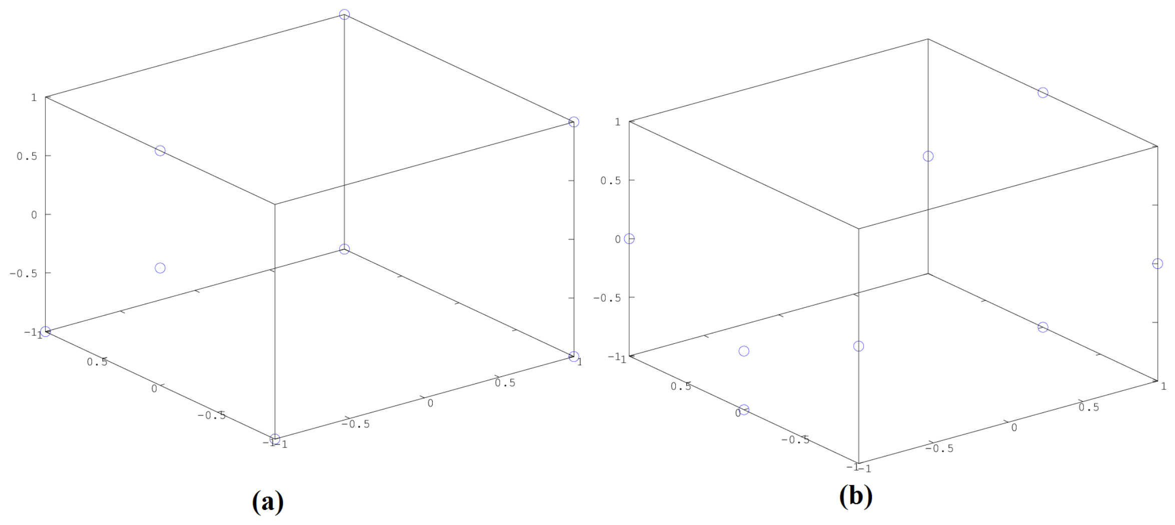

Figure 7. Possible locations of target objects (a) during training; (b) during testing

a sequence of two perturbations. (These outputs are again annealed with noise, which is reduced to zero during training.) The two perturbations drive the hand/arm along a particular trajectory. If this trajectory happens to result in the target object undergoing an action (bend, open or squash), the action recognition system will activate the relevant action category, and a training item is logged, mapping the current goal motor state, plus a unit in the action planning system corresponding to the recognised action category, onto the perturbation sequence. At the same time, the plan activation network learns to map the category of the target object onto the sequence 'cause', followed by the perceived action. This training regime is detailed in Algorithm 3.

Results

After training, the reach network could reliably generate reach-to-touch movements to targets at a range of locations, including unseen locations; a detailed evaluation of this network's performance is given in Lee-Hand *et al.* [16]. In this section we describe the performance of the simple action and causative action networks.

The trained simple action network was tested by presenting a cylinder at a number of selected locations within the system's peripersonal space, activating a simple motor program at random (grasp, slap or punch), and observing how often the tactile stimulus associated with this motor program was produced. Results from these tests are summarised in Figure 17(a). The causative action network was tested by presenting one of the articulated objects at one of a number of locations, and observing how often the network generated a series of perturbations that led to the action perception system registering the action appropriate for the object. Results of these tests are presented in Figure 17(b).

In general, the system was quite successful in producing motor actions with the expected perceptual consequences. The motor program network produced actions resulting in the expected tactile stimuli for an average of 86.25% of seen target locations and an average of 81.2% of unseen locations; the causative action network produced actions resulting in the target undergoing the expected action in an average of 86.7% of seen locations and an average of 83.8% of unseen locations. The differences between the performances of actions are related to the relative complexity of the action and the way that a successful

Algorithm 2: Learning simple motor programs

```
Data: $o = (o_x, o_y, o_z)$ (object center in retinal coordinates),
      $\theta_c(\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles), $d$ (perturbation
      removal distance)
begin
    Reach algorithm and network used to determine $\theta_g = (\theta_{g1}, \theta_{g2}, \theta_{g3})$
    (goal motor state);
    while Maximum time not exceeded do
        $\theta_g$ input into reach network to produce $\Delta = (\Delta_1, \Delta_2, \Delta_3)$
        (perturbation of motor state);
        $\theta_p = (\theta_{p1}, \theta_{p2}, \theta_{p3}) \leftarrow \theta_g + \Delta$ (perturbed goal motor state);
        Random noise added to $\theta_p$;
        if distance to object $< d$ then
            $\theta_p \leftarrow \theta_g$;
        Calculate force applied to arm using PID controller (a function
        of $\theta_p - \theta_c$);
        if correct tactile feedback occurs then
            Store touch score, paired with $\theta_p$;
    if Touch data recorded then
        Log the maximum touch score and corresponding $\theta_p$, paired
        with $o_r$, as a new training item for the simple action network;
        Maximum possible random noise reduced;
        if number of training items exceeds 200 then
            Discard oldest training item;
        Simple action network trained on training data;
```

action is recorded. Grasping actions had the lowest success rate; this is largely due to the mechanical precision required to attain a stable grasp on an object. By comparison, the slap action requires only that the agent makes contact with an object with part of its palm. Illustrations of representative successful actions of each type are shown in Figure 18. The cases where actions do not result in the expected sensory consequences can be accounted for by two main factors. Most failures are due to the simplicity of the feedback motor controller that moves the hand towards a goal state. The hand/arm system is subject to complex Coriolis forces when in motion, and there are limits to how precisely it can be controlled by a simple feedback controller. A few failures result from difficulties generalising from training locations to unseen locations, but in general the networks do this quite well.

Discussion

While there are many computational models of learning to grasp in the field of robotics, there are comparatively few models of how hand/arm actions are learned in infants. By far the most relevant is the neural network model of Oztop *et al.* [2]. This model focusses on infant learning of grasp actions directed at rigid target objects. One point of similarity already mentioned is that this model considers touch sensations to be intrinsically rewarding. Our model adopts this idea of ‘the joy of grasping’, but extends it in various ways, by envisaging a progression of increasingly elaborate sensory stimuli functioning as rewards in a succession of increasingly high-level motor networks. Another point of similarity is in the way

Algorithm 3: Learning causative actions

```
Data: $o = (o_x, o_y, o_z)$ (object center in retinal coordinates),
      $c \in \{lever, door, squashable\_object\}$ (object category),
      $\theta = (\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles), $d$ (perturbation
      removal distance)
begin
    Reach network maps $o$ onto $\theta_g = (\theta_{g1}\theta_{g2}, \theta_{g3})$ (goal motor state);
    Plan activation network maps $c$ onto action plan $P = [P_1, P_2]$
    ($P_1, P_2 \in \{squash, bend, open, cause\}$);
    Causative action network maps $\theta_g$, $P$ onto $\Delta = (\Delta_1, \Delta_2, \Delta_3)$
    (perturbation of motor state);
    $\theta_p = (\theta_{p1}, \theta_{p2}, \theta_{p3}) \leftarrow \theta_g + \Delta$ (perturbed goal motor state);
    Random noise added to $\theta_p$;
    while $\theta \neq \theta_p$ do
        Calculate force applied to arm using PID controller (a function
        of $\theta_p - \theta_c$);
    Store $\theta_p$ as $p_1$;
    $\theta_g$ input into causative action network to produce $\Delta$;
    $\theta_p \leftarrow \theta_g + \Delta$;
    Random noise added to $\theta_p$;
    while Maximum time not exceeded do
        if distance to object $< d$ then
            $\theta_p \leftarrow \theta_g$;
        Calculate force applied to arm using PID controller (a function
        of $\theta_p - \theta_c$);
        if Action recognition system detects an action $a$ then
            Store action score with $(\theta_{p1}, \theta_{p2}, \theta_{p3})$ as $p2$ paired with $p1$;
    if An action $a$ was recognised then
        Identify unit in action planning system $a'$ corresponding to $a$;
        Retrieve maximum stored action score $s_{max}$;
        Log a training item mapping $a'$, $\theta$ onto the perturbation
        sequence $(p_1, p_2)$ with learning constant $s_{max}$;
        Reduce maximum possible random noise;
        if Number of Training items $> 200$ then
            Discard oldest training item;
        Causative action network trained on training data;
        Plan activation network trained to map $c$ onto the planned
        sequence $[cause, a']$;
```

hand trajectories are defined. Oztop *et al.*'s trajectories are defined by 'via-points' through which the hand must pass on the way to the target: their network produces via-points in the way our network produces perturbations. Via-points are represented in a target-centred coordinate system, and so are perturbations, which are specified as deviations from a motor-centred representation of the target location. But there is an important difference: Oztop *et al.*'s model computes trajectories through via-point *in advance*, while our perturbations create trajectories 'on-the-fly'. Our model avoids precomputation of trajectories, as there is no evidence that this is done by humans or monkeys (see again Cisek [15]). But there is a tradeoff: using perturbations to generate trajectories is much less precise than using via-points. How to produce precise trajectories without precomputing them is still an open question.

The most novel aspect of our model is its account of causative motor actions. The closest relevant work in this regard is the model of Arbib *et al.* [23], which explores how an agent learns to interact

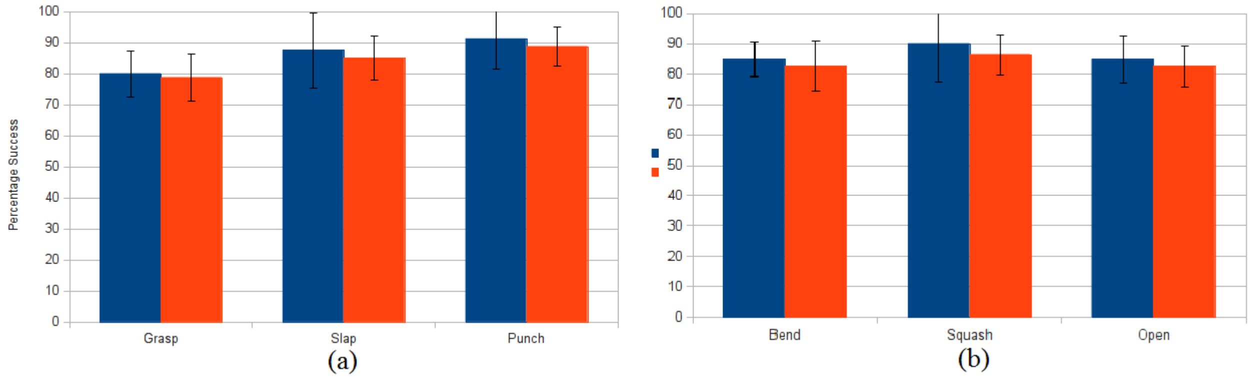

Figure 8. (a) Results from testing the simple action network. (b) Results from testing the causative action network. Error bars show standard deviation for seen and unseen locations across 10 trials of 8 object locations.

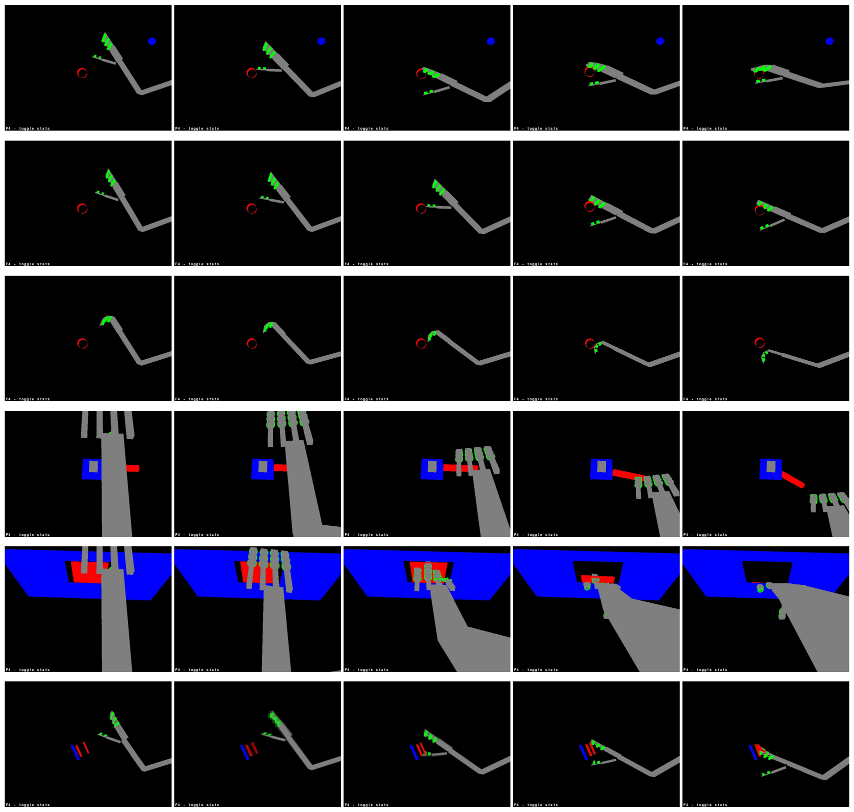

Figure 9. Learned actions. From top: grasping, slapping and punching a cylinder (seen in cross-section); bending a lever, opening a door and squashing a sprung plate. These sequences are taken from the latter stages of each action, when the hand makes contact with the target.

with objects as tools. When an agent holds a tool, representations in the motor system often encode the actions *of the tool* rather than of the hand (see again Umiltà *et al.* [7]): this model aims to explain this phenomenon, which is closely related to the phenomenon we study in the current paper. The main difference in our model is that it does not only consider the actions brought about on an object when the hand is in contact with it; it also addresses how to configure the approach trajectory of the hand so as to achieve a particular action, and it can learn movements producing actions that occur after the hand loses contact with the target. Arbib *et al.*'s model focusses on how a tool becomes incorporated into an agent's motor representation of his hand as an effector, while ours focusses on how an effector can bring about movements in objects even without achieving a stable grasp on them. How to create a model that addresses both issues is again an open question.

Our model makes various novel predictions which could be tested experimentally. For one thing, it predicts that the goal motor state of an effector can be perturbed while it is moving towards a target. We know something about the neural areas where motor goals are evoked (see again Cisek [15]); it would be interesting to examine whether changes in these representations during action execution influence the trajectory of the effector. Another interesting prediction is that the neural circuit controlling causative actions like bending and squashing is partly separate from, and more complex than, the circuit controlling simple actions like grasping and slapping. A specific prediction is that the former circuit involves action-recognition machinery in the superior temporal sulcus in a way that the latter does not. This prediction could readily be tested in an imaging experiment comparing brain activity during execution of simple and causative actions.

Acknowledgments

We would like to thank Tim Neumegen for developing the GraspProject software. We are grateful to the Marsden fund of New Zealand for Grant 13-UOO-048.

References

1. Hommel B, Müsseler J, Aschersleben G, Prinz W (2001) The theory of event coding (TEC): A framework for perception and action learning. Behavioral and Brain Sciences 24: 849–878.

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

3. Schäfer F (2009) The causative alternation. Language and Linguistics Compass 3: 641–681.

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

5. Simon J (1969) Reactions towards the source of stimulation. Journal of Experimental Psychology 81: 174–176.

6. Hommel B (1993) Inverting the Simon effect by intention. Psychological Research 55: 270–279.

7. Umiltà M, Escola L, Intskirveli I, Grammont F, Rochat M, et al. (2008) When pliers become fingers in the monkey motor system. PNAS 105: 2209–2213.

8. Rizzolatti G, Fogassi L, Gallese V (2000) Cortical mechanisms subserving object grasping and action recognition: A new view on the cortical motor functions. In: Gazzaniga M, editor, The New Cognitive Neurosciences, MIT Press. pp. 539–552.

9. Matsumoto K, Suzuki W, Tanaka K (2003) Neuronal correlates of goal-based motor selection in the prefrontal cortex. Science 301: 229–232.

10. Moore J, Obhi S (2012) Intentional binding and the sense of agency: A review. Consciousness and Cognition 21: 546–561.

11. Neumegen T (2013). A computational platform for simulating reach-to-grasp actions: modelling physics, touch receptors and motor control mechanisms. MSc thesis, Dept of Computer Science, University of Otago.

12. Schott J, Rossor M (2003) The grasp and other primitive reflexes. Journal of Neurology, Neurosurgery and Psychiatry 74: 558–560.

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

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

15. Cisek P, Kalaska J (2005) Neural correlates of reaching decisions in dorsal premotor cortex: Specification of multiple direction choices and final selection of action. Neuron 45: 801–814.

16. Lee-Hand J, Neumegen T, Knott A (2012) Representing reach-to-grasp trajectories using perturbed goal motor states. In: Proceedings of the Pacific Rim Conference on Artificial Intelligence (PRICAI). pp. 250–261.

17. Brooks R (1991) Intelligence without representation. Artificial Intelligence 47: 139–159.

18. Keysers C, Perrett D (2004) Demystifying social cognition: A Hebbian perspective. Trends in Cognitive Sciences 8: 501–507.

19. Iacoboni M, Molnar-Szakacs I, Gallese V, Buccino G, Mazziotta J, et al. (2005) Grasping the intentions of others with one's own mirror neuron system. PLoS Biology 3: e79.

20. Knott A (2012) Sensorimotor Cognition and Natural Language Syntax. Cambridge, MA: MIT Press.

21. Averbeck B, Chafee M, Crowe D, Georgopoulos A (2002) Parallel processing of serial movements in prefrontal cortex. PNAS 99: 13172–13177.

22. Robins A (1995) Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science 7: 301–329.

23. Arbib M, Bonaiuto J, Jacobs S, Frey S (2009) Tool use and the distalization of the end-effector. Psychological Research 73: 441–462.

Figure Legends

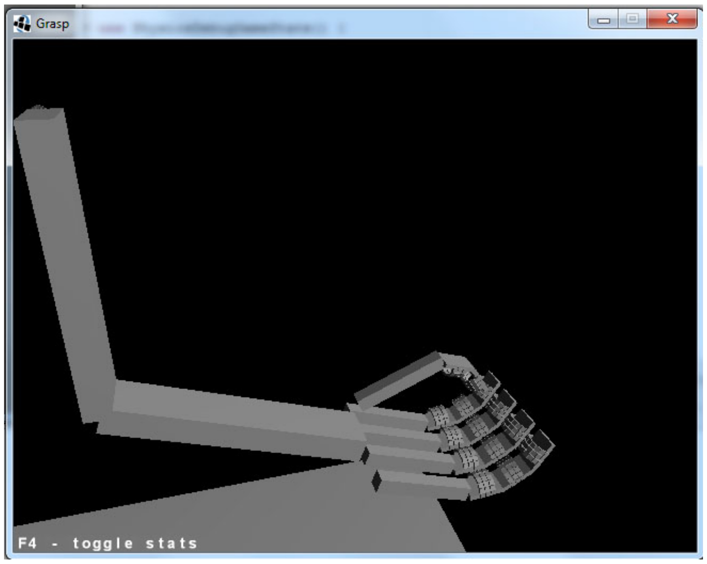

Figure 10. The GraspProject environment. Shown is the arm model that is used in our simulations.

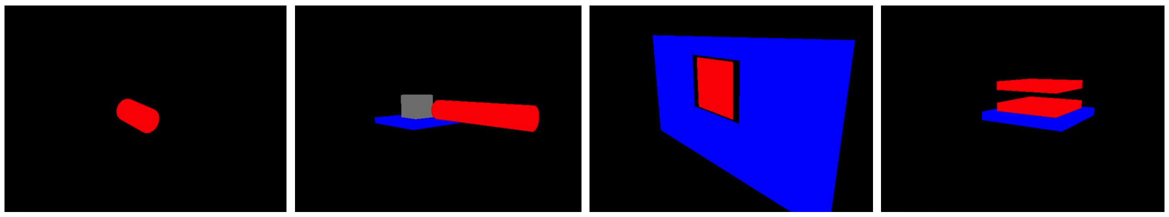

Figure 11. Objects created for the simulations. From left: a cylinder (for grasping, punching and slapping); a lever (for bending); a door (for opening); a compressable object (for squashing).

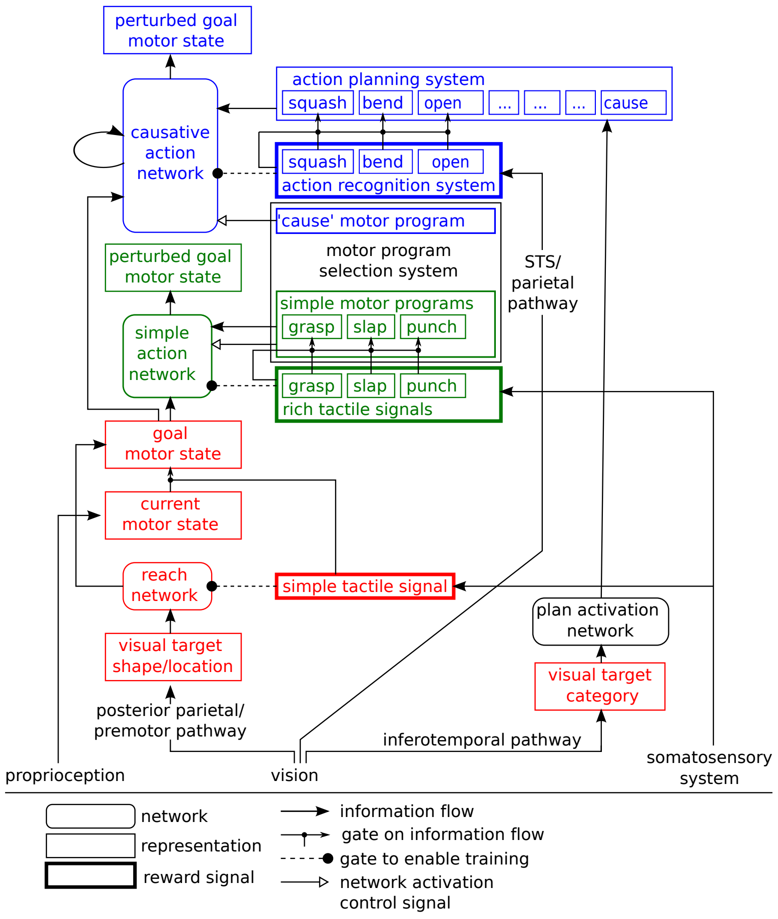

Figure 12. Architecture of the motor control network.

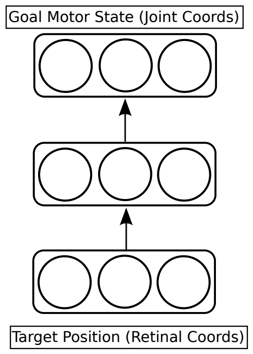
Goal Motor State (Joint Coords)
Target Position (Retinal Coords)

Figure 13. Detailed architecture of the reach network

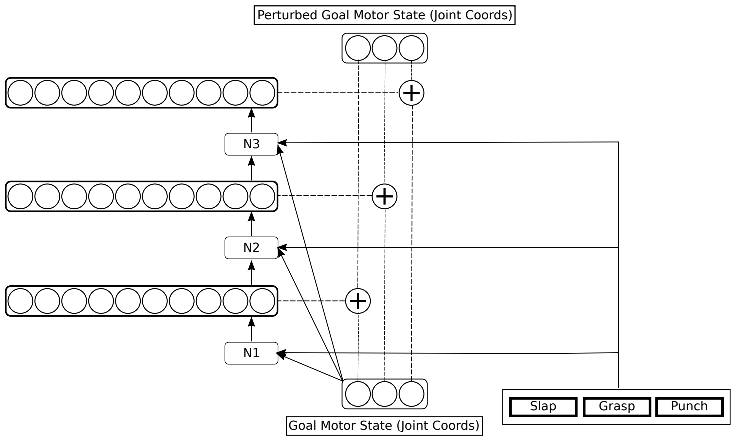
Perturbed Goal Motor State (Joint Coords)
N3
N2
N1
Goal Motor State (Joint Coords)
Slap
Grasp
Punch

Figure 14. Detailed architecture of the simple action network

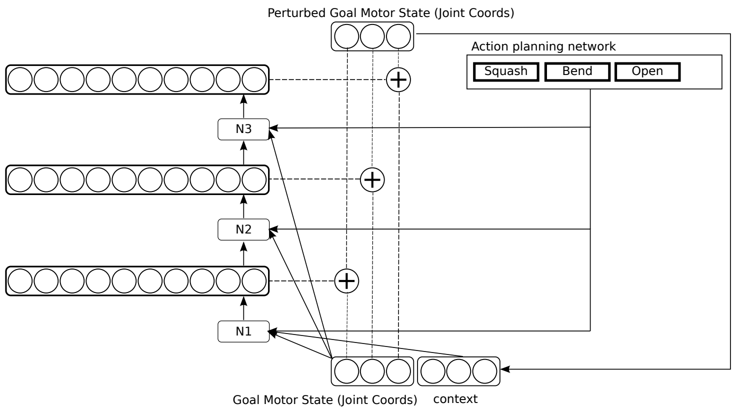
Perturbed Goal Motor State (Joint Coords)
Action planning network
Squash
Bend
Open
N3
N2
N1
Goal Motor State (Joint Coords)
context

Figure 15. Detailed architecture of the causative action network

Algorithm 1: Learning Goal Arm states for reach actions

```
Data: $o = (o_x, o_y, o_z)$ (object centroid in retinal coordinates),
      $\theta = (\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles)
begin
    $o$ input into reach network to produce $\theta_g = (\theta_{g1}, \theta_{g2}, \theta_{g3})$ (goal
    motor state);
    Random noise added to $\theta_g$;
    while Maximum time not exceeded do
        Calculate force applied to arm using PID controller (a function
        of $\theta - \theta_g$);
        if Tactile feedback occurs then
            Store touch score paired with $\theta$;
    if Touch data recorded then
        Log the maximum touch score and corresponding $\theta$, paired
        with $o_r$, as a new training item for the reach network;
        Maximum possible random noise reduced;
        if number of training items exceeds 200 then
            Discard oldest training item;
        Reach network trained on training data;
```

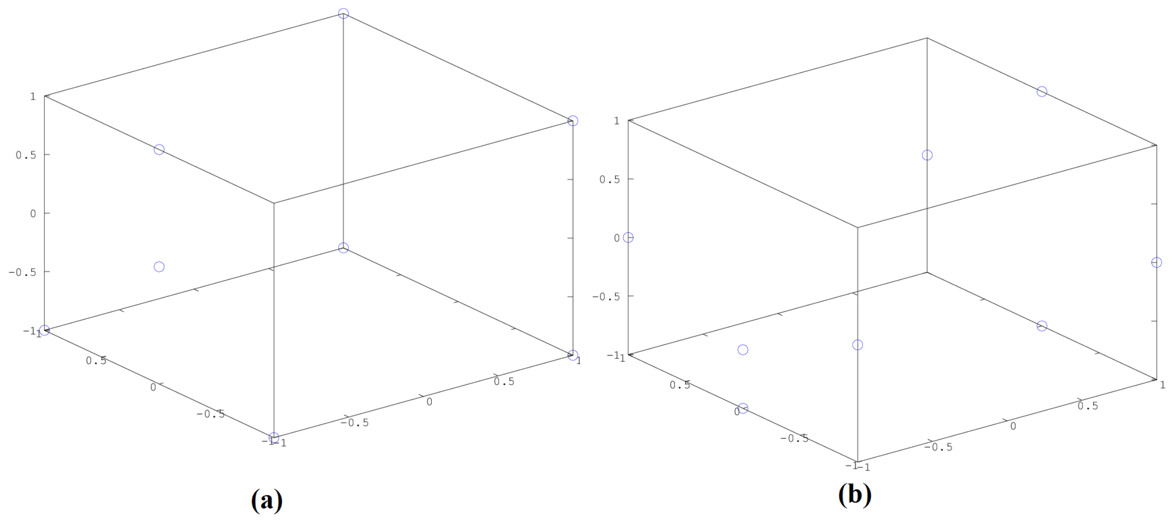

Figure 16. Possible locations of target objects (a) during training; (b) during testing

Algorithm 2: Learning simple motor programs

```
Data: $o = (o_x, o_y, o_z)$ (object center in retinal coordinates),
      $\theta_c(\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles), $d$ (perturbation
      removal distance)
begin
    Reach algorithm and network used to determine $\theta_g = (\theta_{g1}, \theta_{g2}, \theta_{g3})$
    (goal motor state);
    while Maximum time not exceeded do
        $\theta_g$ input into reach network to produce $\Delta = (\Delta_1, \Delta_2, \Delta_3)$
        (perturbation of motor state);
        $\theta_p = (\theta_{p1}, \theta_{p2}, \theta_{p3}) \leftarrow \theta_g + \Delta$ (perturbed goal motor state);
        Random noise added to $\theta_p$;
        if distance to object $< d$ then
            $\theta_p \leftarrow \theta_g$;
        Calculate force applied to arm using PID controller (a function
        of $\theta_p - \theta_c$);
        if correct tactile feedback occurs then
            Store touch score, paired with $\theta_p$;
    if Touch data recorded then
        Log the maximum touch score and corresponding $\theta_p$, paired
        with $o_r$, as a new training item for the simple action network;
        Maximum possible random noise reduced;
        if number of training items exceeds 200 then
            Discard oldest training item;
        Simple action network trained on training data;
```

Algorithm 3: Learning causative actions

```
Data: $o = (o_x, o_y, o_z)$ (object center in retinal coordinates),
      $c \in \{lever, door, squashable\_object\}$ (object category),
      $\theta = (\theta_{c1}, \theta_{c2}, \theta_{c3})$ (current arm joint angles), $d$ (perturbation
      removal distance)
begin
    Reach network maps $o$ onto $\theta_g = (\theta_{g1}\theta_{g2}, \theta_{g3})$ (goal motor state);
    Plan activation network maps $c$ onto action plan $P = [P_1, P_2]$
    ($P_1, P_2 \in \{squash, bend, open, cause\}$);
    Causative action network maps $\theta_g$, $P$ onto $\Delta = (\Delta_1, \Delta_2, \Delta_3)$
    (perturbation of motor state);
    $\theta_p = (\theta_{p1}, \theta_{p2}, \theta_{p3}) \leftarrow \theta_g + \Delta$ (perturbed goal motor state);
    Random noise added to $\theta_p$;
    while $\theta \neq \theta_p$ do
        Calculate force applied to arm using PID controller (a function
        of $\theta_p - \theta_c$);
    Store $\theta_p$ as $p_1$;
    $\theta_g$ input into causative action network to produce $\Delta$;
    $\theta_p \leftarrow \theta_g + \Delta$;
    Random noise added to $\theta_p$;
    while Maximum time not exceeded do
        if distance to object $< d$ then
            $\theta_p \leftarrow \theta_g$;
        Calculate force applied to arm using PID controller (a function
        of $\theta_p - \theta_c$);
        if Action recognition system detects an action $a$ then
            Store action score with $(\theta_{p1}, \theta_{p2}, \theta_{p3})$ as $p2$ paired with $p1$;
    if An action $a$ was recognised then
        Identify unit in action planning system $a'$ corresponding to $a$;
        Retrieve maximum stored action score $s_{max}$;
        Log a training item mapping $a'$, $\theta$ onto the perturbation
        sequence $(p_1, p_2)$ with learning constant $s_{max}$;
        Reduce maximum possible random noise;
        if Number of Training items $> 200$ then
            Discard oldest training item;
        Causative action network trained on training data;
        Plan activation network trained to map $c$ onto the planned
        sequence $[cause, a']$;
```

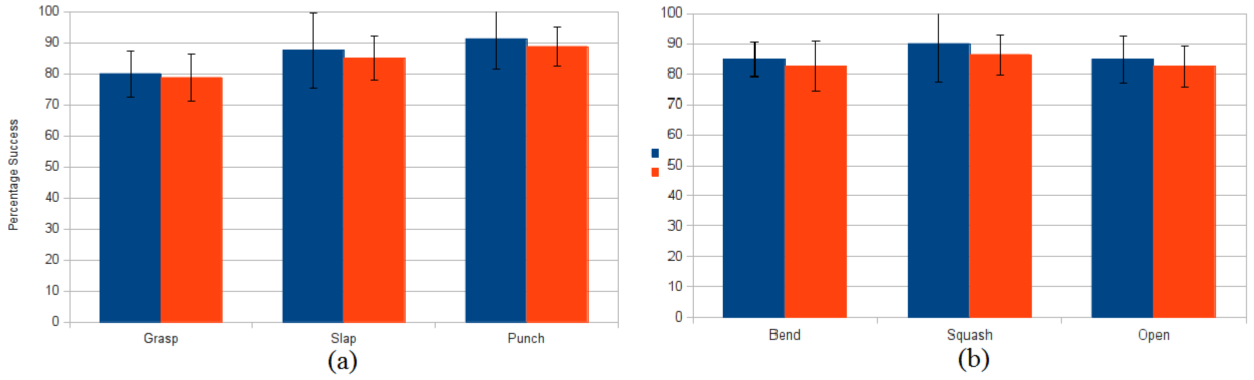

Figure 17. (a) Results from testing the simple action network.(b) Results from testing the causative action network. Error bars show standard deviation for seen and unseen locations across 10 trials of 8 object locations.

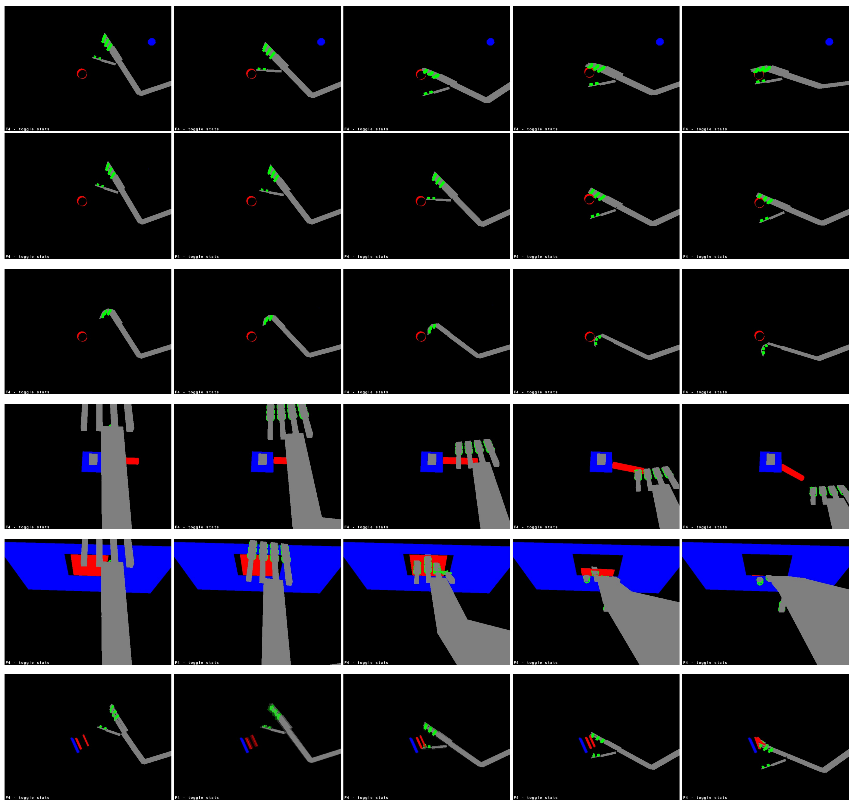

Figure 18. Learned actions. From top: grasping, slapping and punching a cylinder (seen in cross-section); bending a lever, opening a door and squashing a sprung plate. These sequences are taken from the latter stages of each action, when the hand makes contact with the target.