Chapter 8: Grasp Configuration Learning

In this chapter, we present a model that learns the transformation from visual and task parameters to an appropriate grasp. The transformation is acquired through a process of trial and error: the simulated robot makes successive attempts at grasping objects and observes the outcome of the attempts. The degree of success of each attempt is scored by a teacher, which is then used by the *reinforcement learning algorithm* to adjust the visual/motor mapping. After repetitively experiencing several different situations, the network learns to perform the appropriate transformations. In addition, individual neurons in the network either align themselves with one of the two grasps (precision pinch or power grasp), or come to represent the aperture of the grasp that is executed.

Introduction

The grasping model presented in previous chapters of this thesis has relied on a heuristically wired network for the transformation from visual and task information into a grasp description. The wiring algorithm requires that the gross behavior of each F5 and AIP module be specified a priori by the network "programmer." In this chapter, we demonstrate that such a transformation may be automatically acquired by a network through the use of a learning algorithm. Rather than specifying the intermediate representation of the mapping (behavior of individual neurons), the *teacher* provides only information about the input/output pairs of the desired transformation. The learning algorithm makes use of the teaching information in adjusting the behavior of individual neurons such that the desired transformation is ultimately achieved.

In the experiment presented below, it is assumed that the teacher actually does not know the exact form of the transformation. In other words, the teacher cannot say what the exact motor commands are that should be sent to the fingers of our simulated robot given any particular situation. However, the teacher is capable of evaluating the performance of a grasp that is executed by the robot. The high-level performance measure of *how well* the robot did, referred to as the *reinforcement* feedback signal, is given in the form of a scalar score. In the case of the grasping task, this score measures two elements: the success and the efficiency of the executed grasp (described below in more detail). Based upon this feedback information, the reinforcement learning algorithm (Barto, Sutton, & Anderson, 1983; Sutton, 1988) adjusts the connection strengths of the artificial neural network so as to maximize the success and efficiency of the grasps that are executed.

Neural Dynamics

The neural architecture (as shown in Figure 8.1), which is significantly simplified from that presented in earlier chapters, has been adapted from our work on primate visual-motor conditional learning (Fagg & Arbib, 1992) and work on reinforcement learning for reactive control of a mobile vehicle (Fagg, Lotspeich, Hoff, & Bekey, 1994). The visual and task information that is input into the model is represented as an activity pattern across a set neurons (V). A total of 11 neurons are used in this layer: three represent object type (cylinder, cube, and plate), three represent length (short, medium, long; note that this is a discrete representation), three for diameter (narrow, medium, wide), and two for task requirements (manipulability and stability). In the monkey, this visual information is provided by a sub-region of the posterior parietal cortex (specifically PIP); task information can be derived from a number of different regions including prefrontal cortex (including area 46), and preSMA (F6).

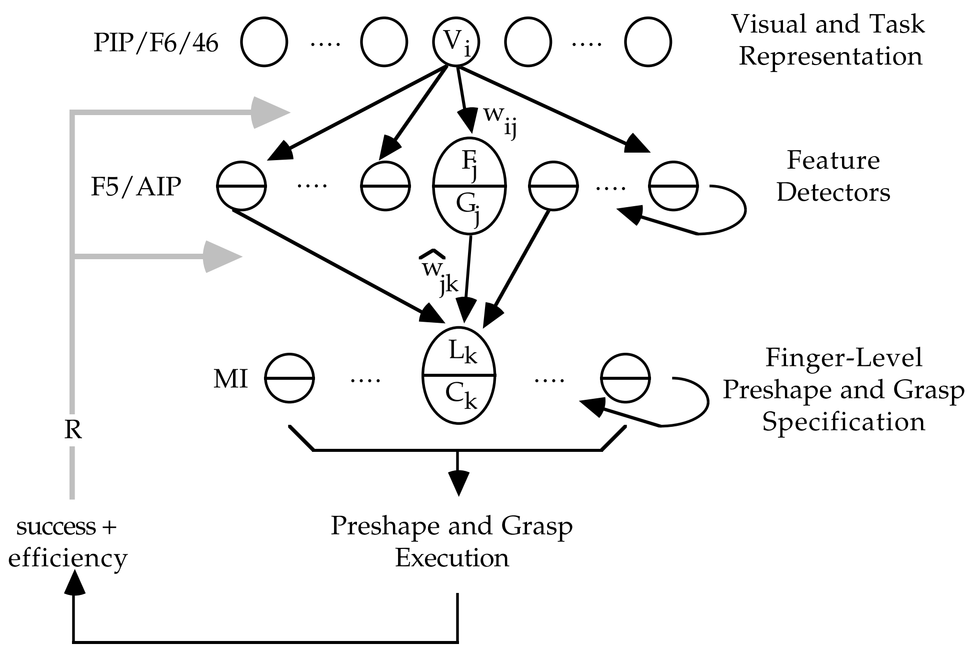

Figure 8.1: Schematic view of the architecture for the grasp configuration learning model. Visual parameters (from PIP) and task requirements (from F6 and area 46) are combined at the feature detector layer (representing the AIP/F5 complex). The activated feature detectors in turn select a hand configuration by specifying how individual fingers will behave during the preshape and grasp (primary motor cortex; MI). This program is executed and then evaluated by a teacher. The evaluation (R) is used to update the inter-layer connection strengths.

This information (V) is projected through a set of synapses (W) to a feature detector layer (F/G), in which each neuron represents some higher-order feature of the original description. For example, a feature detector representing *cylinder* ***and*** *manipulability* would receive connections from the corresponding input neurons. The activity level of the feature detector unit (F) is computed according to the following equation:

$$F_j = \sum_i \left(V_i * w_{ij}\right) + Noise_j \tag{8.1}$$

where:

F_j is the activity level of feature detector unit j.

V_i is the activity level of input unit i.

w_{ij} is the strength of connection from input unit i to feature detector j.

$Noise_j$ is a random signal that is injected into feature detector j.

These feature detector units then interact through a local competition mechanism to contrast-enhance the incoming activity pattern. This is implemented is a one-pass local-maximum operation: a neuron produces a non-zero output if and only if it is the most active neuron within a small neighborhood. The output of the feature detector units (G) is computed as follows:

$$G_j = \begin{cases} F_j & if\ F_j = \underset{j-N \le l \le j+N}{Max}\{F_l\} \\ 0 & otherwise \end{cases} \tag{8.2}$$

where:

G_j is the output of unit j.

N defines the size of the region of competition. Note that we have implicitly assumed that the neurons are arranged in a circular array.

The active feature detectors then vote for the configuration to be imposed upon the hand by passing activation to the output units (L):

$$L_k = \sum_j \left(G_j * \hat{w}_{jk}\right) + Noise_k \tag{8.3}$$

where:

L_k is the activity level of output unit k.

$\hat{w}_{jk}$ is the strength of connection from feature detector unit j to output unit k.

$Noise_k$ is a random signal that is injected into output unit k.

The configuration specifies which fingers will be actively participating in the grasp and how the fingers should be positioned during the actual execution of the preshape and grasp. As this model is designed to drive the Belgrade/USC Hand (Bekey, Tomovic, & Zeljkovic, 1990), the index/middle and the ring/little finger pairs are considered single entities to be controlled. The output (C) consists of seven separate sub-vectors, each specifying a different detail of the grasp configuration. Three of the sub-vectors (each consisting of two units) specify the participation of the thumb, I/M fingers, and R/L fingers in the grasp, respectively. One sub-vector (also consisting of two units) determines whether the thumb will be abducted or not. The three remaining sub-vectors (each consisting of three units) determine the degree of flexion (small, medium, and large) during preshape of the thumb and of the I/M finger and R/L finger pairs, respectively.

For each sub-vector, a winner-take-all circuit computes the single most active unit of the set:

$$C_k = \begin{cases} 1 & if\ L_k = \underset{m \in S(k)}{Max}\{L_m\} \\ 0 & otherwise \end{cases} \tag{8.4}$$

where:

$S(k)$ is the set of units that are in the same sub-vector as unit k.

C_k indicates whether configuration bit k is a winner.

It is this resulting pattern of activity (C) that is used by the non-neural execution system to generate the motor commands that result in the preshape and enclose movements of the hand.

Learning Dynamics

After execution of the specified grasp, a teacher evaluates the performance of the system. There are two elements to this evaluation: success and efficiency. *Success* tells us whether or not the grasping movement was able to pick up the object. If the robot is unable to accomplish this, then a reinforcement signal of R=-0.1 is given by the teacher. If the grasp is successful, then a positive reinforcement signal (R=1) is given, but discounted if the grasp is *inefficient*. A grasp is considered inefficient if the fingers preshape to a larger extension than is necessary for the presented object (this is related to Hoff's notion of *hand comfort* (Hoff & Arbib, 1993)). The discount factor is set such that if the system produces a preshape of maximum possible aperture for a narrow cylinder, a reinforcement of 0.4 is given (assuming success).

The reinforcement signal is used by the learning algorithm to update the connection strengths in the projections from the visual/task representation to the feature detectors (W), and from the feature detectors to the action units ($\hat{W}$), with the goal of ultimately maximizing the level of reward that is received. The adjustments to the connection strengths are done using a Hebbian/Anti-Hebbian learning algorithm, as follows:

The system is presented with a certain input, for which a grasp plan is computed by the network and then executed. Suppose that the teacher gives a positive reinforcement signal for the system's performance on the trial. In this case, we would like to ensure that the next time the system is presented with the same input, the same plan is output. This is accomplished by:

1) Making sure that the same set of feature detectors is activated by increasing the connection strength from active input units (V) to active feature detectors (G) (thereby increasing their response level the next time).
2) Increasing the active feature detectors' support of the selected grasp plan by increasing the connection strengths from the active feature detectors (G) to the active output units (C).

On the other hand, suppose negative reinforcement is received. This could be due to the fact that either the wrong set of feature detectors was selected, or just that the specified grasp plan was incorrect. Since we do not know which is the case, both are assumed:

1) The connection strengths from the active input units (V) to the active feature detectors (G) are reduced, thereby giving other feature detectors an opportunity to become active next time (implementing a search for the correct set of feature detectors).
2) The support of the active feature detectors for the specified configuration is also reduced. This allows other configurations to be tried.

These rules are captured in the following connection strength update equations:

$$
\begin{aligned}
\Delta w_{ij} &= \alpha R V_i G_j w_{ij} \\
\Delta \hat{w}_{jk} &= \alpha R G_j C_k \hat{w}_{jk}
\end{aligned}
\tag{8.5}
$$

where:

Δw_{ij} and $\Delta \hat{w}_{jk}$ are the changes to the connection strengths.

α is a learning rate parameter.

R is the teacher's reinforcement signal.

In a final step, the weight updates are integrated into the weight matrices W and $\hat{W}$ according to:

$$
w_{ij} \leftarrow \frac{w_{ij} + \Delta w_{ij}}{\sum_i \left(w_{ij} + \Delta w_{ij} \right)}
\tag{8.6}
$$

$$
\hat{w}_{jk} \leftarrow \frac{\hat{w}_{jk} + \Delta \hat{w}_{jk}}{\sum_k \left(\hat{w}_{jk} + \Delta \hat{w}_{jk} \right)}
\tag{8.7}
$$

The combination weight update and normalization not only maintains individual weights within a limited range, but also implements a form of competition between the different connections. Relative to a single feature detector unit, when a set of input connections are incremented (in response to positive reinforcement), the remaining connections are subsequently decremented (equation 8.6). This encourages the development of feature detector units that are responsive to a small number of situations. On the other hand, when a set of weights are decremented (negative reinforcement), the remaining connection strengths are increased, making the feature detector responsive to a larger number of situations (forcing the system to continue searching for the appropriate combination of feature detector units).

The set of outputs leading from an individual feature detector unit also compete with one-another (equation 8.7). This can be interpreted as an individual feature detector unit having a limited number of votes that can be distributed across all possible configurations.

Reinforcement Prediction

As presented, the reinforcement learning algorithm has a tendency to converge on the first successful solution that it discovers for each situation. This is the case because the reception of positive reinforcement always results in an increase of some set of connection strengths, even after the network reliably produces the successful solution. The result is that the activity level of the feature detector and output units is not significantly affected by the injected noise (equations 8.1 and 8.3), and hence the system does not continue the search process, even in the vicinity of the successful solution. Here, we outline a modification to the learning algorithm that attempts to focus learning to situations in which there is an increase or decrease in performance, rather than being sensitive to the absolute performance of the network (more details of this formulation can be found in (Fagg, Lotspeich, Hoff, & Bekey, 1996)).

First we assume that for any situation that is presented to the network (situation being object and task), we have an estimate of the network's average performance, referred to as $\hat{R}$. We can then define *internal reinforcement* as the deviation of the network's performance (on a single trial) from its average performance:

$$R' = R - \hat{R} \tag{8.8}$$

and replace the weight update equations (8.5) with:

$$\begin{aligned} \Delta w_{ij} &= \alpha\, R'\, V_i\, G_j\, w_{ij} \\ \Delta \hat{w}_{jk} &= \alpha\, R'\, G_j\, C_k\, \hat{w}_{jk} \end{aligned} \tag{8.9}$$

The effect of this algorithmic change is that when the network performs better (or worse) than expected, a positive (negative) internal reinforcement signal is generated. In addition, when the network performs as expected, no learning occurs.

The average performance of the network is estimated as a linear function of the input and feature detector states:

$$\hat{R}(V,G) = M^T \bullet \begin{bmatrix} V \\ G \end{bmatrix} \tag{8.10}$$

where:

M is a column vector of parameters.

After observation of the outcome of a grasp, the estimate of average performance is updated according to the incremental LMS algorithm (Widrow, 1962):

$$M \leftarrow M + \hat{\alpha}\Delta M \tag{8.11}$$

$$\begin{aligned} \Delta M &= \left(R - \hat{R}(V,G)\right)\nabla_M \hat{R}(V,G) \\ &= \left(R - \hat{R}(V,G)\right)\begin{bmatrix} V \\ G \end{bmatrix} \end{aligned} \tag{8.12}$$

where:

$\hat{\alpha}$ is the learning rate.

The effect of this reinforcement prediction algorithm is that when the system discovers a new (and better) solution to a certain situation, the network will receive positive internal reinforcement for several trials. This positive reinforcement will cause the network to favor the new solution over the old. However, the reinforcement predictor will respond (with a lag) by increasing the estimate of average performance. This means that after the first few trials, the new solution will result in a near-zero internal reinforcement signal, and therefore learning is stopped unless an even better solution is discovered. This allows the search process to continue (due to the injected noise) around the vicinity of the currently favored solution.

Note that this process of first estimating and then climbing the reinforcement gradient is a related to Sutton's temporal difference algorithm (Sutton, 1988), and its application to learning of optimal controllers (Barto & Bradtke, 1991).

Experimental Results

During the training process, the system was presented with a sequence of situations selected from six possibilities. The presented object was a cylinder of one of three widths. In addition, the task requirement could be either manipulability or stability. The experiments were performed in simulation. Evaluation of the grasps was accomplished by checking that the fingers were opened wide enough to clear the object and that the task requirements were suitably satisfied. If manipulability was a requirement, the system was expected to generate a pad opposition grasp configuration (with the thumb opposing the finger tips). If stability was required, then the system had to have produced a palm opposition in order to be considered successful. The protocol randomly presented the system with one of the six possible situations. However, if the system failed to grasp the object, then the same situation was presented until a successful grasp was obtained.

For the results presented below, the network required 2500 trials before its performance peaked. After training, all grasps were successful but not all were completely efficient. Of the 80 feature detector units, only 18 achieved very significant response levels. Figure 8.2 shows the post-learning response curves of five of these units as a function of the situation that is presented. Units A-D are all selective for a pad opposition, although C and D are also selective to some degree for the width of the object. Unit E is selective for palm opposition. These grasp specific neurons, similar to those reported by Rizzolatti (Rizzolatti, et al., 1988) in area F5, resulted from the learning process and the system's interaction with its environment, even though the architecture did not inherently contain these concepts.

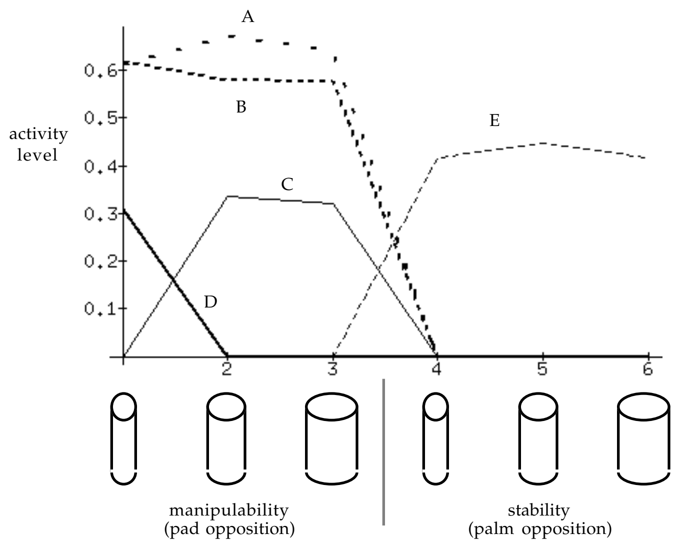

Figure 8.2: Responses of several feature detector units (A-E) given the six different situations. All units are selective for either pad or palm opposition.

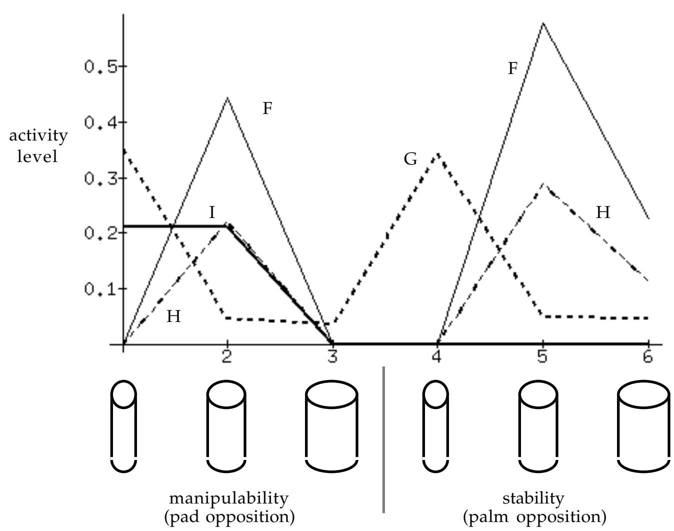

Figure 8.3: Responses of four additional feature detector units (F-I) from the same experiment. Units F, G, and H are selective for the width of the cylinder. Unit I is selective for small & medium cylinders for pad opposition.

Finally, it is important to note that the firing pattern of some cells did not follow a simple symbolic rule such as *fire during execution of the pad opposition*. Rather, these cells responded to a combination of several different input dimensions. Despite this apparent difficulty of placing symbolic labels on these cells, their interaction with other cells in the feature detector layer was able to yield a correct and consistent grasp plan.

In addition to grasp selective cells, other types of units were also seen in this experiment. Figure 8.3 demonstrates several such cells. Cells F and H are selective for objects of medium width regardless of the grasp type that is made. Cell G is selective for small objects, but does show some activity for the other situations. In all, nine units demonstrated purely opposition-related activity, where only four showed object size selectivity. The remaining five either showed selectivity to exactly one situation or to an arbitrary combination of multiple situations. Although size-specific responses have not been observed in F5 (due to experimental design), Sakata has observed such responses in AIP (Sakata, 1994).

Summary

In this chapter, we have demonstrated that a transformation from visual and task parameters into a motor plan can be acquired automatically. The network programmer (or teacher) need not specify the computational processes behind the transformation (i.e., the behavior of individual neurons), but rather may specify the expected results of the computation. However, unlike algorithms that learn to mimic human grasp plans (Iberall, 1987; Iberall, 1988; Uno, Fukumura, Suzuki, & Kawato, 1993), the planner actually learns by observing the results of executing its own plans. Note that there is still a teacher involved in the process - instead of telling the robot exactly *what* to do, the teacher only tells the robot *how well* it did in performing its task. In this way, it is possible to construct plans that are best oriented towards the actuation and sensing capabilities of the robot. This is a property that is not necessarily achievable when only mimicking human performance, since underlying the human behavior may be a program that relies on sensory feedback that is not available to the robot. When this happens, the robot is unable to distinguish situations in which different motor decisions must be made.

It is important to note, however, that the network architecture presented in this chapter is significantly simplified from that used in the main model (Chapters 4, 5, and 6). First of all, the mapping from visual and task parameters into motor outputs is implemented as a one-pass neural algorithm; any recurrent interactions are simulated through the use of winner-take-all mechanisms. Second, the output of the network is a static representation of the grasp. The actual execution of the preshape and grasp are assumed to be implemented by a separate, non-neural module. Although these simplifications are justified from a computational efficiency argument, it is also not immediately clear how the temporal of aspects of the neural firing patterns (in the original model) can be shaped by an associative algorithm. This is the subject of ongoing research.