Chapter 8

System Integration

8.1 Introduction

The results presented in Chapter 6 and Chapter 7 show the applicability of the Incremental Evolution (IE) technique to robotic control systems. It has been shown that the method allows the robot’s body plan and the controlling neural network to build from a simple to a complex form. The technique has been successfully used to evolve neural control systems up to the level of those required for quadruped robots. Other applications of the technique have also been discussed in the latter part of Chapter 6. In this Chapter, the experiments will concentrate on incorporating the technique into a more advanced robot with a vision system. Later, the technique is used to grow and incorporate both locomotion and vision into the same structure to form a system.

8.2 Vision System

Since the discussion in Chapter 6 and 7 was based on networks which mainly control outputs (producing walking patterns), it was also decided to build networks for a vision system using a similar method. This provides a contrast since such networks are involved in processing inputs. In particular, to provide a difficult but realistic task, the network was configured to mimic a toad’s behavior as reported by Ewert [Ewert 1985, 1987] developed by Arbib [Arbib 1995] and implemented by Reddipogu [Reddipogu 2002] (see Section 2.4 of Chapter 2).

Before proceeding further, consider the development of the human sensory system. There are limits to body plan evolution as far as actuators and sensors are concerned. For example, in the case of a robotic man, when all the joints are in place and able to be well controlled (the robotic equivalent of an australopithecus), then only the “mind” neural network will continue to evolve with a more complex environment.
The same idea applies to sensors – for example, Sight, Hearing, Smell, Taste and Touch. It is likely that these will be evolved along with or after basic locomotion (going up the ANS model starting at the bottom).

We can assume that all such systems start with the simplest possible arrangement (just a single sensor - the equivalent of “one degree of freedom”) and become more complex incrementally [Ewert 1985]. Let us consider sight as an example. This would start in nature as just a light sensitive spot on the skin of the animal and develop eventually into an organ capable of forming an image. Figure 8-1 shows the development of the vision system from a single pixel.

![Figure 8-1 Vision system](image)

To do this, the input sensor and the range of patterns to which it is exposed are allowed to grow in a similar way to that previously explained for the body plan. The pixels on the grid can be in two different states, either ‘ON’ (black pixels) or ‘OFF’(white pixels). There are three different stages involved in the evolution of the vision system explored here. It starts as a single pixel in Stage 1. Then a 3 x 3 sensor block was added to vision system in Stage 2. Finally, a 5 x 5 block was added. Figure 8-1 illustrates the evolution at different stages. Appropriate leg patterns have to be produced on the 4 output neurons. Figure 8-2 shows the progression in sensor complexity with the desired leg patterns for different inputs. The repertoire of patterns available ranges from simple fight or flight responses to the identification of obstacles in the field of view.
Figure 8-2 Evolution of vision sensor complexity
The discussion below is based on the Stage 1 evolution of the vision system but is applicable to the other stages as well.

The leg pattern indicates which output gait should be triggered for an input. Firstly, a module with 4 neurons was trained to produce the initial leg pattern (retreat). The network was awarded a score of 10 for successfully producing the correct output pattern. Then, the connection weights and neuron parameters of the current module were frozen. Secondly, a new module was added to the previous network in order to train both the patterns (retreat and walk). The network was awarded 20 points if it managed to reproduce the correct output pattern for both these inputs. The vision sensors (pixels) are fully connected to the first module and connections to other modules are determined by the EA. The outputs were always taken from the first module.

The reason for connecting all the sensory inputs to the first module was to make sure that, at least at one stage, the sensory inputs are relayed to all the neurons. Figure 8-3 illustrates the above explanation.

![Diagram](image_url)

**Figure 8-3 First stage evolution**
The Spike Accumulation and Delta Modulation [Shigematsu 1996] neuron model described in Section 5.2 of Chapter 5 was used to evolve the modules. The duration for all the vision experiments is 1 timestep.

A module with 4 neurons was trained successfully to produce the retreat response. Then a new module with 2 neurons was added to produce both (retreat and walk) leg patterns. The explanation on different numbers of neurons required in the newly added modules has been given in Chapter 6. Figure 8-4 shows the output of the leg patterns for different inputs and the fitness improvement as new modules were added to the previous modules.

![Figure 8-4 Vision output for stage 1](image)

Next, a $3 \times 3$ sensor block was added to the vision system as shown previously in Figure 8-2 (b). A new controlling network was evolved at each stage. Connections were not allowed between the different stages (although there is no specific reason for doing so). Two modules, each with 4 and 3 neurons have been evolved to produce the
“Go Right” and “Go Left” responses. Later, new modules with 2, 3, 4, and 5 neurons were added but these modules failed to produce the third leg pattern (“Go Forward”). Figure 8-5 shows the fitness (score) improvement for the second stage of the vision system. It can be seen from the graph that the fitness levels off at 20 with an increasing number of modules thereafter.

![Figure 8-5 Vision output for stage 2](image)

It seems that the network has problems producing 3 or more different leg patterns. It is very likely that the neurons have difficulty dividing the solution space into different domains. Another experiment (equivalent to Figure 6-15, Section 6.4 of Chapter 6) was conducted where the outputs were taken from the newly added module. Figure 8-6 illustrates the concept.

![Figure 8-6 Adding new module in front](image)
Even with this technique the network failed to produce all the required patterns. At this stage it was thought that the neuron functionality might be causing the problems. Similar problems were faced when the MMM neuron model was used for the evolution of the bipedal locomotion at the beginning of the research (refer to Section 6.2). It was thought a simplified neuron model might perform better.

The most common type of artificial neuron model was used and is shown in Figure 8-7. This is the modified standard “McCulloch-Pitts” or “Perceptron” type neuron [McCulloch 1943] with a threshold function. The operation of this neuron model can be summarized as follows: The weights of the connections \((w_n)\) represent the strength of the synapse in a biological neuron. The total input to the neuron is calculated as the weighted sum of all inputs. The weighted sum is normalized using a function, commonly the sigmoid function. The sigmoid function produces an output in the range 0 to 1. The threshold is fixed at 0.5. If the output of the sigmoid function is greater than the threshold, then the neuron fires and produces a pulse (an output value of 1), vice versa no pulse (an output value of -1). Only the connection weights are trained when this type of neuron model is used.

\[
y = f \left( \sum_{j=0}^{n} i_j w_j \right)
\]

![Figure 8-7 Modified Standard McCulloch-Pitts neuron with threshold function](image)

The initial experiment concentrated on evolving a network to produce all the four different leg patterns in Stage 2. This is because previously we had difficulties in evolving a network to integrate the different leg patterns at this stage. The same technique illustrated in Figure 8-3 was initially used for this experiment. A network with two modules each with 4 and 3 neurons has been used to master the first two patterns. The network failed to produce the third pattern when a new module was added. It was very difficult to predict what was causing the problems. The technique illustrated in Figure 8-6 showed successful results when it was used for evolving
locomotive networks (see Section 6.4). This technique was then considered together with the neuron model shown in Figure 8-7.

A network with 4 modules, each with 4 neurons, was successfully evolved to produce all the 4 patterns. There were 4 neurons in each new module because 4 output neurons are required for each pattern. Figure 8-8 (a-d) shows the output leg patterns for the respective inputs for stage 2. Figure 8-8 (e) shows the fitness improvement as new modules are added to the previous modules. These results show that the neuron functionality is very important to network success.
Figure 8-8 Output leg patterns for respective inputs for stage 2

Figure 8-9 shows the output leg pattern for stages 1 and 3. Networks have been grown in the sequence shown in Figure 8-2 to successfully integrate all the patterns presented. These results show that the technique of adding new modules in front of the previously evolved modules is very useful when the traditional approach fails.

**Stage 1 Leg Pattern**

![Stage 1 Leg Pattern Graphs]

**Stage 3 Leg Pattern**

![Stage 3 Leg Pattern Graphs]

Figure 8-9 Output leg patterns for respective inputs for stage 1 and 3
[Reddipogu 2002] used a fixed neural network topology to mimic toad’s vision system. The connection weights were trained until the network successfully learnt all the different input patterns. Reddipogu used Evolutionary Algorithms for Reinforcement Learning (EARL) to train the network. The learning algorithm took more than 13000 generations to master all the different visual patterns. It is hoped that this new evolutionary technique will be able to evolve a network with superior performance with lesser number of generations.

### 8.3 Integration of Locomotive with Vision Networks

As explained in Section 7.2 of Chapter 7, if the robot is to become smarter, it must be introduced to an environment to which it can adapt. However, there seems little point in starting with a full scale (unconstrained) environment. There are simply too many (potentially conflicting) possibilities for it to contend with. The environment must be allowed to evolve along with the robot as previously described (that is, “deconstraining” the environment, an equivalent term to the process of “sensor and leg joint deconstraint” in the body plan).

An analysis of the sort of tasks of different complexities that simple animals can undertake indicates a possible forward direction. Table 1 below lists all the objects used to illustrate the progression.

<table>
<thead>
<tr>
<th>Objects</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Light source" /></td>
<td>Light source</td>
</tr>
<tr>
<td><img src="image" alt="Simplest Animals" /></td>
<td>Simplest Animals</td>
</tr>
<tr>
<td><img src="image" alt="Simplest Invertebrates" /></td>
<td>Simplest Invertebrates</td>
</tr>
<tr>
<td><img src="image" alt="More complex invertebrates" /></td>
<td>More complex invertebrates</td>
</tr>
<tr>
<td><img src="image" alt="Path" /></td>
<td>Path</td>
</tr>
<tr>
<td><img src="image" alt="Obstacles" /></td>
<td>Obstacles</td>
</tr>
<tr>
<td><img src="image" alt="Mates" /></td>
<td>Mates</td>
</tr>
<tr>
<td><img src="image" alt="Food" /></td>
<td>Food</td>
</tr>
<tr>
<td><img src="image" alt="Predator" /></td>
<td>Predator</td>
</tr>
</tbody>
</table>

**Table 1 Objects and its representation**
1) Simplest animals – Go towards light

2) Simplest invertebrates – Recognise and avoid obstacles (plus skills of stage 1)

3) More complex invertebrates – Recognise food and mates (plus skills of stage 1 and 2)

4) Flatworms type animals – Recognise and flee from predators (plus skills of stage 1, 2 and 3)

5) Fish type animals – Path finding and learning (mission skills) (plus skills of stage 1, 2, 3 and 4)
6) Reptile / Bird Skills – Manipulation e.g. Object which must be removed, etc
7) Higher Skills in mammals – tool skills, etc.

These different degrees of environmental interaction must be added one at a time in a thoughtful way to the robot. This may be accomplished through the addition of changing targets to the system in the changing environment or alternatively, by making the fitness function of the robot gradually more complex as it develops. The neural networks required to control the robot would be grown in similar ways to those previously described (see Section 5.6 of Chapter 5). It is clear with this technique that the neural network that has been evolved to interact with a particular environment will still be present even after a new network has been grown for another environment. This is useful because the previously evolved network could be re-used when the same environment re-occurs.

Returning to our previous work, separate networks exist for the locomotion and vision systems. The next stage was to grow networks to interface the first stage of the vision system to the previously evolved single degree of freedom bipedal walking and jumping gait. This problem is somewhat similar to the environment number 3 (recognise food and mates) illustrated above since there are two different possible conflicts to deal with. The other stages (Stage 2 and 3) of the vision system are not considered in the discussion since the interest is in proving that the technique can be used to integrate multiple different networks to form a system. The growth algorithm was unchanged from that described in Section 5.6. The network allowed different locomotive gaits to be triggered when different visual patterns were input. In this case, the bipedal walking gait will be triggered when the walking leg pattern is present at the vision system and vice-versa for gallop. The interface network can be said to be a $2 \times 1$ multiplexer because one from the two different input channels (bipedal walking and jumping gait network) will be selected to be the output depending on the input selection (coming from the vision system) at any time. Figure 8-10 shows the system configuration for the above case.
The ES is not allowed to form connections from the new module to the previously evolved networks since this could modify the original behaviour of the networks. The Spike Accumulation and Delta Modulation neuron described in Chapter 5 was used to evolve the interface network. The fitness function for the interfacing network is a measure of the number of leg positions successfully relayed from the locomotive network to the output module for a triggering input. Each time an output neuron relayed the correct output to the actuator, the network was awarded a score of 1. Since there were 2 neurons in the output module, a maximum score of 2 can be awarded for a single timestep. A total score of 1000 could be achieved for simulation of 500 timesteps. In this case, the maximum fitness was 2000 since there were 2 locomotive gaits (walking and jumping).

A total of 5 modules with 2, 4, 3, 5 and 2 neurons was required to integrate the vision and the locomotive networks (refer to Chapter 6 for more explanation on the requirement for variable number of modules and neurons). Figure 8-11 shows the fitness improvement as each new module is added. Figure 8-12 shows the individual fitness for each gait as new modules are added. Table 2 gives a more detailed breakdown of the fitness in both Figure 8-11 and Figure 8-12 above.
From Table 2 it can be seen that there is a gradual increment in fitness for the bipedal jumping gait. The fitness dropped by 7 to 903 (Bipedal walking) and increased by 60 to 900 (Bipedal jumping) when the third module was introduced to the network. The probable reason for the decrement in the fitness that is, the ES could not manage to
evolve a set of weights and neuron parameters for both the gaits. There is also no requirement in the fitness function to make sure an increment in distance moved is achieved when a new module is introduced to the network. The gain entirely depends on previously evolved modules and the ES. It also can be seen that there is symmetry in the increment of individual fitness scores from the third module onwards. The number of generations required to achieve the fitness level was 1500 for each module. The number of generations needed is relatively large compared to the number of generations required for much simpler tasks presented in Chapter 6 and Chapter 7. One possible reason could be the neuron functionality. The interface network (as mentioned before could be a $2 \times 1$ multiplexer) has to integrate all three different networks. In electronics a multiplexer can be built using logic gates. If neurons in the network have to function like any of those logic gates, without any doubt the number of generation required to evolve a network would be fewer. Also the network was evolved to integrate several different objective functions. Evolving networks for multiple objective functions has proved a problem in past work [Lund 1994]. Modules with 2, 3, 4 and 5 neurons were trained for 5000 generations but the fitness level was not as good as that listed in the table. This shows that there is a minimum number of neurons required in order for the system to successfully evolve. Networks could also be grown to integrate Stages 2 and 3 of the vision system. In another experiment, the growth technique failed to evolve a network to control the robot’s actuator and the vision system at the same time. This shows the success of the incremental growth technique in dealing with a complex problem incrementally.

8.4 Discussion

The system described above holds promise as a solution to the problem of the open ended evolution and development of neural networks and hence to the creation of large and complex multi-functional neural systems. Since the technique adopts a systems approach to the problem, it is particularly useful in robotics and similar problems where various unrelated subsystems need to be developed and integrated in an intelligent way.

Two important findings from the research were: That the neuron used should be as flexible as possible, as it is necessary to perform many difficult mappings in both the amplitude and time domains, especially when interfacing different modules of
previously grown networks and, secondly, that the evolutionary algorithms must be able to choose the network’s connections as well as their weights.

The need for a flexible neuron with evolvable functionality has led the group to consider “universal” neuron models which can potentially evolve any continuous response [Capanni 2003]. This work is at an early stage but moves away from the idea of biologically feasible models and towards evolvable processors.

One possible disadvantage of the system is that, unlike a network designed by an optimal method, these networks may be wasteful of computing resources, in that they are potentially larger, although the current simulations do not show this with small networks. Another limitation, although, again, this has not been experienced in the simulations, may be apparent in systems where evolution or growth cannot go through an obvious sequence from simple to complex as part of its development. A related problem occurs in evolutionary timetabling and scheduling systems, in which a particular module must be placed early in the sequence to avoid a “bottle neck” occurring later – that is, a particular evolutionary path may preclude certain later developments.

It can be envisaged that, as systems become more complex, there will be a need to engineer changes (deconstraint) in the Fitness Function as development proceeds, choosing carefully the required steps to allow the system to evolve in the required manner. In the end, this process would stop body plan change, once full motor control had been achieved, and allow only the evolution of behaviour, in much the same way as the human brain continued to evolve in our early ancestors, even after our body plan was essentially fixed. The issue described above is a subject for future work. The final issue is whether some flexibility in previously evolved modules would make the evolution of later modules easier.

It is hoped that, once these issues have been resolved and integrated into the framework, new and interesting intelligent behaviours will emerge out of larger and more systems-orientated networks.