Development of Sensory Data Processing Capability in Artificial Organisms *

Brad Fullmer
Department of Computer Sciences
The University of Texas at Austin, Austin, TX 78712-1188
e-mail fullmer@cs.utexas.edu

Abstract

The ability of natural organisms to process sensory data arose in response to evolutionary pressures. Simply put, those organisms having a better knowledge of their environment can react to external changes more rapidly and with greater precision, thereby gaining an advantage over less capable organisms. This project attempts to apply suitable evolutionary pressures to effect the development of sensory processing capabilities in artificial organisms. In pursuit of this goal, key biological concepts are used to guide the simulation. A genetic algorithm is used as the mechanism for development, neural networks are used to implement processing function in the artificial organisms, and sensory input is provided in a raw, unprocessed form. Our experiments show that by offering a survivability advantage to those organisms which are best able to use their sensory input, neural networks quickly evolve with the capacity to process sensory (visual) input effectively. The development of the complex neural network structures needed for this task is facilitated by a new gene-to-neural-network mapping which allows networks of arbitrary complexity to evolve.

1 Introduction

Providing computers with mechanisms to interpret the physical world is a difficult process. Gathering the real world data is simple enough, involving, for instance, converting analog visual or acoustic signals into digital form. The difficulty, however, lies in extracting meaningful information from this sea of raw data. The strategy used in this paper attempts to develop sensory processing function by traversing the same developmental path which has led to our own considerable sensory capabilities. Since visual data contains the most useful overall assessment of an environment, we will focus on the development of visual processing function. Before sophisticated high level visual capabilities can evolve, simple visual schemas must develop which process the raw input with varying levels of abstraction. Effecting the development of a specific visual schema using simulated evolution requires the formulation of a task whose solution mandates the use of that particular schema. If the task is properly constructed, the evolution of an organism adept at performing the task will indicate the development of the desired visual skill. In this manner, increasingly complex visual processing capabilities can be developed by increasing the complexity of the task.

*Thanks to Risto Miikkulainen for his assistance throughout the course of this research.
While one cannot argue with the effectiveness of evolution for producing advanced visual processing capability, one could question the practicality of such an approach. Even the simplest improvements in functionality take an inconceivable amount of time to occur in nature. Fortunately there are key areas where we can bypass much of the overhead associated with the emergence of functional advancements. The visual processing capability of any organism is limited by the bandwidth of the visual data received by its central nervous system. In order for our own sophisticated visual processing skills to emerge, the evolution of complex physical structures, such as the lens and retina, was a necessity. The relative stability of these visual organs from reptiles to humans indicates the difficulty of developing new physical structures through the evolutionary process. By contrast, the evolution of advanced neural capacity to make use of these new structures seems comparatively simple. Since the basic physical structure already exists (the neuron), evolving the desired functionality involves, basically, finding the right combination of these units. With the simulated evolution approach we can easily change the bandwidth of the input provided to the neural networks, leaving the evolutionary process the single task of finding a neural network that can use it effectively. Considering the additional advantage of being able to evolve generations of organisms in milliseconds rather than years, it is quite feasible to expect reasonably sophisticated visual processing capability to emerge in a timely manner.

Where do we begin the process of evolving artificial visual processing? What are the fundamental visual primatives needed for the development of useful visual function? The first step in the development of visual skills involves an organism's understanding of the relationship between its own movements and changes in its visual input. Without this understanding, an organism could not distinguish between visual changes resulting from its own actions and visual changes caused by new environmental information. The experiments described in this paper are geared toward evolving this basic schema. A broader goal of the experiments is to formulate a general strategy for allowing visual processing capabilities to develop, independent of the degree of complexity involved. The bandwidth of the visual data for these experiments may seem trivial, but the effectiveness with which the neural networks exploit this visual input is the critical factor and should be carefully noted.

2 The Experiment

The goal of the experiment is to construct a test in which the development of basic visual schemas could be observed. In order to encourage the emergence of basic visual skills, this task requires the development of higher level search and recognition skills given only an unprocessed 'digitized' representation of its visual field as input. The hypothesis is that without an effective low level interpretation of the visual data, such higher level processes would be impossible.

2.1 Test Scenario

The test is designed to produce specific visual processing abilities in artificial creatures. Thirty such creatures are used as the subjects of the testing process. The basic test involves placing a creature and a certain object together in a simulated world and allowing the creature to operate. The simulated world is a five by five grid with the property that the edges wrap around, so the square immediately to the right of the rightmost square on the
grid is the leftmost square on the same row. The objects used in the experiment vary in size and shape but generally occupy seven to nine squares and are placed in the center of the grid. A creature occupies one square and can 'see' five squares around it: the square directly in front of it, directly to its left and right and diagonally in front of it to its left and right. If the creature (represented by an 'A') is facing towards the top of the page it sees the following five squares:

```
123
4A5
```

A creature can perform the following operations:

1. Turn left
2. Turn right
3. Move forward
4. Do nothing

The fourth option is included so that a creature can change the internal state of its neural net without changing its position.

A creature is initially placed in a random square (one not occupied by the object) and faces a random direction. A creature's lifespan is 35 cycles long. Each cycle consists of evaluating a creature's neural network (given its visual field as input) and using the output of the network to adjust the creature's physical position (if necessary). When the experiment is configured, one of the objects is said to be 'good' and the other is 'bad'. Each lifetime can end in one of two results depending on the property of the object.

1. Creature survives if:
   - The object was 'good' and the creature hit it, that is moved onto a square occupied by the object -or-
   - The object was 'bad' and the creature avoided it for all 35 cycles.

2. Creature dies if:
   - The object was 'good' and the creature does not hit it in 35 cycles -or-
   - The object was 'bad' and the creature hit it.

If the creature dies, its run is finished. If a creature survives, however, it continues to be tested. Each time a creature survives, its neural network is reset to its initial state, it is randomly placed within a world containing a randomly selected object, and it is given a new lifespan of 35 cycles. A creature will continue in this manner until it performs incorrectly and dies. If a creature performs correctly for sufficiently many runs, it is terminated and said to have perfected the task. A creature's measure of success is based on a score tabulated by adding up the number of consecutive times that the creature performs correctly. After each creature has been tested and scored in this manner, a genetic algorithm (described fully in section 4) is used to manipulate the creatures' genes in order to improve overall performance.
3 Gene to Neural Network Mapping Algorithm

3.1 Background

When using genetic algorithms, the primary unit of operation is a set of homogeneous values called a gene. The final form of the entities being evolved is unimportant at the level of the genetic operations, only the gene is manipulated. Therefore when selecting a structure as the subject of evolution, one must also construct a way to convert the values of a gene into the final form of the structure in question. Since this project is concerned with evolving neural networks, a method of converting a sequence of integers (the gene) into a neural network was required.

In order to find the most efficient solution to a problem, it is essential that the search space of the genetic algorithm be as large as possible. With neural nets the goal becomes to allow as many different (valid) neural nets as possible to evolve, placing minimal restrictions on the characteristics of the network. Since the gene must contain all the data which is subject to evolution, the key to opening up the search space is to map as many net characteristics as possible onto the gene. The mapping scheme constructed for this project allows every aspect of a network's topology to fall under evolutionary control.

3.2 Biological Motivation

The basic concepts of genetic algorithms are biologically motivated, with the 'gene' of the genetic algorithm being analogous to the DNA of nature. Taking this analogy a step further, the gene-to-net mapping algorithm used for this project is loosely based on the method by which a cell's DNA is mapped to its production of proteins.

A strand of DNA is comprised of a sequence of nucleotide triplets. These triplets are used to specify strings of amino-acids which make up a protein. In a single typical strand of DNA there are multiple proteins specified in this fashion. To separate the specification of different proteins, certain nucleotide triplets are used as markers rather than for the purpose of representing amino-acids. There is a triplet for a start marker and one for an end marker with the triplets in between specifying the amino acids used to construct the protein [1]. This concept of using marker values to section off the working areas of the genetic material is a key feature of our new gene-to-net mapping algorithm. Rather than specifying proteins, as in DNA, the marker values found in our genes are used to specify neurons.

3.3 Implementation

The gene used for this project consists of 800 integers, where each integer is a value between -100 and 100. Certain values are designated as start and end markers. Each artificial neuron (subsequently referred to as a node) is defined by the following sequence of gene values:

\[</start> <key> <initial value> <k1> <w1> <k2> <w2> . . . . . <kn> <wn> </end>\]

\[
\begin{align*}
\text{start} & \quad - \text{Start marker.}
\end{align*}
\]
key - Value used to identify this node in connections (see below).

initial value - Output value of node prior to first evaluation.

<ki> <wi> - Each pair specifies one connection used as input to the node. ki specifies the source node of the connection, with the node whose key (see above) is closest to this value being chosen. wi specifies the weight of the connection.

<end> - End marker.

Only the start and end markers are identified by their gene value. All the other characteristics are defined by their positional relationship to these markers. Determining which gene values are start and end markers is done using the following rules:

IF gene_value MOD 15 = 1 THEN gene_value is a start marker

IF gene_value MOD 15 = 2 THEN gene_value is an end marker

The gene is treated as a continuous circular entity, with the start marker of the first node serving as the end-marker for the last node if there is an 'open' node at that point.

Input is provided to the network in the form of five binary values. Each value represents the contents (empty or occupied) of a world-square with a certain relationship to the creature. These values are referenced by nodes as connection sources in the same way that other nodes are: by specifying a particular key value. If a connection is specified where the magnitude of the key value is less than 20, it is treated as a reference to an input. The exact input is taken as the modulus of the key value and the number of inputs.

3.4 Additional Information

When the network is evaluated, each the output value of each node is determined as follows:

Node definition:

<start> <key> <initial value> <k1> <w1> <k2> <w2> .... <kn> <wn> <end>

\[
\text{sum} = (\text{output value of node referenced by k1} \times w1) + \\
(\text{output value of node referenced by k2} \times w2) + \\
\vdots \\
(\text{output value of node referenced by kn} \times wn)
\]

IF sum >= 0 THEN output_value = 1
ELSE output_value = 0
The nodes are evaluated in the order which they are read off the gene. Before each node's initial evaluation, its output value is set to the modulus of its initial value (specified in the gene) and two. The output of the network is taken as the output values of the last two nodes read off the gene. If a net contains fewer than two nodes it is not evaluated.

3.5 Advantages of the New Mapping Scheme

Whereas most gene to net mapping strategies fix each position in the gene to a particular net characteristic, the algorithm presented here allows each gene position to be used in the way which produces the maximum benefit for the creature. In some cases many nodes with a small number of connections may be ideal, in other cases fewer nodes with a larger number of connections could be required. If a small, efficient network topology is desired, net size or execution speed could be incorporated as an evolutionary constraint.

An interesting phenomenon which consistently emerges when using the mapping is the occurrence of 'constant nodes', nodes with no input connections. Since these nodes have no input, their output value will never deviate from its initial value. This property allows other nodes to reference these constant nodes as inputs in order to effect a threshold other than zero. Since the standard threshold is zero, the effective threshold of a node connected to the constant node becomes:

\[ 0 - \text{constant_node's\_initial\_value} \times \text{weight\_of\_connection} \]

4 The Genetic Algorithm

4.1 Overall Strategy

The goal of this project is to replicate certain features which have arisen naturally through evolution. This necessitates the use of a genetic algorithm to simulate the effects of evolutionary pressure on the population of artificial creatures. The fundamentals of the algorithm used in this project are fairly standard [2]. After all the creatures have been tested and assigned a score, the genetic algorithm is applied to the population. This involves first sorting the creatures by score and using the best creatures for recombination. Each of these elite creatures is paired with a mate to produce two new offspring (section 4.2.1). A mate is assigned to each of the elite creatures by randomly selecting another creature whose score is \textit{at least as good} as the first's. This property ensures that the top scoring creature is always duplicated and offers increased chances of propagation to the higher scoring creatures. The new creatures produced by the elite are used to replace the non-elite creatures. The original elite creatures remain in the population. Finally, every creature except the top scoring creature undergo mutation (section 4.2.2). See table 1 for a summary of the entire process.

A population size of 50 creatures was used in all the experiments.
4.2 Genetic Operations

Genetic operations are the mechanisms by which genes change over the course of evolutionary cycles. The two operations described here, crossover and mutation, are standard genetic operations [2]. Both of these operations have analogous counterparts in biological systems.

It is common in genetic algorithm experiments to allow the genetic operators to manipulate the gene at the bit level [2]. The scheme adopted for this project, however, treats the integer as the basic genetic unit. This approach was adopted mainly to reduce processing overhead thereby allowing larger gene sizes to be used. The impact of this integer gene representation with regard to the individual genetic operations is discussed in each section.

4.2.1 Crossover

When two creatures are combined to form offspring, the standard two-point crossover [2] approach is used. With this method, two random points on the gene are chosen by which to partition it. Since the gene is treated as a circular entity, this effectively breaks the gene into two continuous chunks. Each new gene is constructed by taking one chunk from each parent and combining them to form an offspring. Either parent may supply either chunk resulting in two possible combinations, thus two offspring are formed. The idea is illustrated below:

```
Parent 1:
AAAAAAABBBBAAA
Crossover points: 1 2
Offspring:
AAAAABBBBAAA

Parent 2:
BBBBBBBBBBBBBB
AAAAABBBBAAA

Offspring:
BBBBBAAAAAAABBB
```

Using the crossover operation is motivated by the idea that certain sections of a given gene will evolve to effect beneficial characteristics in the entity specified by the gene. By combining different areas of different genes the hope is that different beneficial areas, previously on different parent genes, will end up on the same offspring gene, yielding the beneficial traits of both parents.

The crossover process has seems especially well suited for use with the gene-to-net mapping used for this project (section 3). Since some of the space on the gene will likely go unused (the space between the end of one node and the start of the next). The crossover points have a chance of falling in these unused sections thereby transporting intact node definitions to the offspring. The key based referencing scheme is also advantageous, allowing a group of nodes can be transported to the offspring with all of its connections intact, hopefully preserving its function.

If this operation was performed at the bit level, it would be possible that certain gene values could change if a split broke up the bits of an integer. This would have the advantage of introducing new variability in the gene pool, but at the cost of the possible disruption of beneficial gene values.
4.2.2 Mutation

Typically, the standard mutation operation works by flipping a bit in a gene [2]. To simulate the natural variability of this scheme at the integer level, the following approach is used: Individual integer elements are mutated by randomly selecting a delta value within the range allowed for gene values and adding this value to the existing value of the gene. If the new value falls outside of the allowable range, it will ‘wrap-around’ in either the positive or negative direction. Each element in the gene has a 0.4 percent chance of undergoing mutation during each evolutionary cycle. The top scoring creature does not undergo the mutation process.

The function of the mutation operation is simply to introduce new variability into the gene pool.

5 The Experiments

In the tests, three different data sets of varying complexity are used. Each data set consists of a pair of objects, one of which is deemed ‘good’ and the other ‘bad’. The creatures must evolve the capacity to avoid the bad object and seek out the good object.

To guard against aberrations (either beneficial or detrimental) the tests were run for each data set five different times using five different seeds for the random number generator. The same five seeds were used for testing each data set. The tests were performed by evolving the population until a creature completely mastering the task had evolved. Recall that in order to master a task a creature must be able to perform correctly from an arbitrary starting position and orientation resulting in a completely general solution.

The initial gene values for all the creatures are randomly generated, with gene values always kept between -100 and 100 (before and during evolution). The gene size used for the tests is 800 integers.
<table>
<thead>
<tr>
<th>Seed</th>
<th># Generations to Solve</th>
<th># of Nodes</th>
<th>Avg. # connections per node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>304</td>
<td>15</td>
<td>11.20</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>11</td>
<td>9.00</td>
</tr>
<tr>
<td>3</td>
<td>77</td>
<td>16</td>
<td>7.56</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>15</td>
<td>10.07</td>
</tr>
<tr>
<td>5</td>
<td>296</td>
<td>15</td>
<td>9.73</td>
</tr>
</tbody>
</table>

Table 2: Test results for data set one.

5.1 Data Set One

5.1.1 Description

The first (and simplest) data set consists of the following objects:

```plaintext
***  *
*** ***
***  *
bad  good
```

This problem is simple because each object has an identifying characteristic which falls entirely within a creature's field of view. The creature need only roam around until it finds one of these distinguishing views and then act appropriately. For example, if the creature sees '***' directly in front of it, it can simply stop and wait for its lifetime to expire, since that pattern cannot be found in the good object. Likewise, if the creature sees a '*' directly in front of it and a '*' directly to one side it need only move forward, since this pattern only appears in the good object.

5.1.2 Results

This task was mastered in an average of 138 generations (each generation represents one iteration of the genetic algorithm as shown in table 1). Two types of behavior evolved. In one case the creatures would search for a distinguishing characteristic of the shape and once found, go into a new wait pattern if the object was bad, or hit the object if it was good. The other type of behavior (developed by only one creature) had the creature circle the bad object indefinitely and quickly hit the good object. It should be noted that the behavior of the third creature is identical to that of two of the creatures which evolved in data set two (section 5.2.2). In fact when this creature was run on data set two, it achieved a perfect score. See table 2 for full test data.
<table>
<thead>
<tr>
<th>Seed</th>
<th># Generations to Solve</th>
<th># of Nodes</th>
<th>Avg. # connections per node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>116</td>
<td>17</td>
<td>6.94</td>
</tr>
<tr>
<td>2</td>
<td>211</td>
<td>15</td>
<td>10.80</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>16</td>
<td>9.25</td>
</tr>
<tr>
<td>4</td>
<td>414</td>
<td>14</td>
<td>8.29</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>9</td>
<td>20.11</td>
</tr>
</tbody>
</table>

Table 3: Test results for data set two.

5.2 Data Set Two

5.2.1 Description

The second data set consists of the following objects:

```
***   *
***   ***
***   ***
```

bad    good

These objects present a greater challenge as there is no longer any unique view which identifies the bad object. When encountering a bad object, the creature must survey different areas of the object. This process is required since every field of view found with the bad object is also possible with the good object.

5.2.2 Results

The average number of generations taken to evolve a perfect creature in this test was 158. The behavior of all the creatures followed the same basic pattern: circle the bad object indefinitely and hit the object if a missing corner is seen.

An interesting phenomena was observed in the circling creatures. They seem to have evolved an initialization phase, whereby upon first starting a new life, they will usually move forward one square and rotate 360 degrees before they start circling. Occasionally, though, they will forgo this phase and start immediately into their pattern. Apparently for certain initial positions and orientations, they can't tell exactly where they are from the visual field and must look around to get their bearings. Other starting positions, however, leave no doubt as they have a unique visual field associated with them. See table 3 for full test data.
<table>
<thead>
<tr>
<th>Seed</th>
<th># Generations to Solve</th>
<th># of Nodes</th>
<th>Avg. # connections per node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>820</td>
<td>13</td>
<td>10.30</td>
</tr>
<tr>
<td>2</td>
<td>1302</td>
<td>13</td>
<td>5.92</td>
</tr>
<tr>
<td>3</td>
<td>1643</td>
<td>16</td>
<td>5.94</td>
</tr>
<tr>
<td>4</td>
<td>832</td>
<td>14</td>
<td>10.93</td>
</tr>
<tr>
<td>5</td>
<td>978</td>
<td>11</td>
<td>13.00</td>
</tr>
</tbody>
</table>

Table 4: Test results for data set three.

5.3 Data Set Three

5.3.1 Description

The third data set consists of the following objects:

```
*     ***
***   ***
***   ***
bad    good
```

It will be noted that these are the same two objects found in the previous data set. The difficulty of the problem is increased, however, by making the object with the distinguishing characteristic bad. Now, in order to act correctly in the presence of the good object, the creature must survey the entire object, determine that it does not have the undesirable characteristic and then hit it. The important feature being that the creature must internally 'remember' what it has seen previously and use that data along with the current input to determine the appropriate action. This is precisely because there is no single view that indicates that an object should be hit.

5.3.2 Results

The average number of generations taken to evolve a perfect creature for this data set was 1115. The behavior of the five creatures was very similar. When encountering a good object, they would survey two sides and if both were found to be intact, hit the object. If a bad object was recognized, the creatures would go into a holding pattern of a few cyclic moves until the lifetime expired. See table 4 for full test results.

5.4 Notes on the Experiments

The behavior of the various creatures is quite interesting and could have been described in greater detail. As previously mentioned though, the aim of this project is geared toward demonstrating the emergence of lower level skills. Behavior is important, but only to the extent that it indicates other capabilities.
6 Discussion

The goal of these tests was to encourage the evolution of low level visual processing capabilities in artificial creatures. Specifically, we sought to evolve a creature with the ability to 'ground' external objects relative to its position regardless of the its current perspective. Examining the test results we see the emergence of sophisticated high level behavior. The question now becomes: Can we infer the presence of the low level capabilities based only upon the existence of high level behavior?

Let us examine the question from a creature's perspective. Every time a creature moves, certain predictable changes occur. For example if a creature sees a '*' directly in front of it and executes a right turn, the creature will see the '*' to its left. Always. In the case of right turns, the visual input representing the left view will never contain any new information. Consider now the effects of a left turn on a creature's visual field. Suddenly the left view reveals previously unseen information, possibly indicating the identity of an object. The same visual input which was previously meaningless, now contains valuable data. For each movement that the creature makes, there is a predictable change which will occur in the creature's visual field. Without the ability to relate a particular movement to a predictable change, the visual input would make absolutely no sense. It therefore seems reasonable to assume that in order for a creature to perform complicated recognition tasks, it must first develop a functional understanding of what effects its own movements have on its visual inputs. It must then use this understanding to modify the way it interprets visual data in accordance with its movements.

Interesting questions about the fundamental concepts used in this project remain. What would happen if, rather than feeding a five element visual field to a creature, a (much larger) creature was supplied with a 10,000 element digitized three-dimensional image? Could it use this data in a meaningful way? Performance considerations currently limit the bandwidth of visual data that can be processed using the techniques presented here. However, the ease with which the creatures in these tests were able to evolve useful functionality, suggests that increased visual processing capabilities could be attained by increasing the sophistication of the test scenarios (using higher resolution visual inputs for instance). The free-form gene encoding scheme introduced in this paper is well suited for more complicated applications since it can develop nets of arbitrary complexity and capacity (providing a large enough gene size is used).

7 Conclusion

This project has clearly shown the potential of using biological concepts to develop rudimentary sensory processing function. The observed development of simple visual schemas in response to evolutionary pressures strongly indicates that this approach could be used for the development of more sophisticated visual processing capabilities.

References
