

Swapping Evolved Behaviours in a Subsumption Architecture Robot

Mike Blow
Candidate Number 80761

ABSTRACT

In recent robotics research two methodologies have become prevalent; the subsumption architecture which uses several 'hand-coded' behaviours running in parallel, and evolutionary robotics which aims to evolve a single holistic controller. Both of these approaches suffer from a certain lack of scalability and it is proposed that a hybrid approach might enable that boundary to be pushed a little further. This paper describes an experiment which uses evolved behaviours in a subsumption-style architecture. Furthermore it demonstrates that a single neural network can be used for all behaviours by simply swapping the relevant network values, a departure from normal subsumption practise. Lastly it discusses some of the problems and issues raised by this approach.

1.0 BACKGROUND

1.1 Subsumption Architecture

In the 1960's representational artificial intelligence made some impressive advances and extravagant claims were made of robots that would replace human beings in the near future. However, progress proved slow and by the end of the 1970's robotics research had bogged down as the scientists realised the world was far more complicated than they gave it credit for. In the 80's Brooks suggested that autonomous behaviour should be approached in a 'ground-up' rather than 'top-down' fashion. He proposed a parallel architecture made up of behavioural layers, each of which could be used when relevant and 'subsumed' by the layer above it if necessary [2,3]. This

paradigm also stressed that the robot should have little or no representational information about the world in which it lived, in stark contrast to the 'sense-plan-act' world model strategy of previous robots like 'shakey' [11]. Subsumption robots were strikingly successful given their simplicity.

1.2 Evolutionary Robotics

A second popular approach to robotics research has been to evolve controllers using genetic algorithms [4,5,8,9,10]. Typically a population of robots, each with a neural network 'brain', is evaluated for a particular task and the best-performing individuals form the basis for the next generation. Using crossover and mutation the new generation is bred from the old and the new robots are assessed. Over many generations the fitness increases until the robot can perform the required task. Evolutionary robotics can produce surprisingly effective results from minimal functional elements.

1.3 Problems

However both of these approaches suffer from a scaling problem. Complexities of behaviour definition and layer interaction limit the subsumption approach. Likewise difficulty in defining the fitness function and the exponential increase in the search space with extra parameters cause problems in the evolutionary case. Evolutionary robotics also has the drawback that an evolved neural net can be hard to analyse and predict, a problem which will only increase with the complexity of controller and which limits its industrial acceptance.

2.0 PROPOSAL

2.1 Evolved Layers

Regarding the problems above it seems worth investigating a hybrid approach whereby various behaviours are evolved and used as layers in a subsumption style robot. The premise is that a number of small evolved networks will prove efficient and easier to evolve and analyse than a single large controller.

2.2 Swappable Behaviours

Normal subsumption robots literally have a series of separate behavioural layers. However the approach proposed here gives us the chance to investigate whether a single neural network could be used in all circumstances. A single network is capable of producing a variety of behaviours. Each behaviour is defined by the weights and biases evolved in the network. Therefore it should be possible to swap the weights and biases while the robot is running in order to change its behaviour. This further improves the efficiency of the evolved approach, as only one network is required.

3.0 THE EXPERIMENT

3.1 The Simulator

The experiment used a 2D evolutionary robot sim that I had written for a previous assignment based on Jakobi's minimal Khepera simulation [6,7], with various modifications (please see appendix 1 for a list of original additions made in this project). The basic sim comprises three screens: an 'arena' in which the robot moves on an unbounded 2D plane, a 'nodeview' which displays the state of the neural network in graphical form and a 'statsview' which displays the network values in numerical form (*fig. 1*).



fig. 1 the full simulator screen. The arena is on the left, the nodeview top right and the statsview bottom right. Note the lights the robot seeks (round, yellow) and avoids (square, red). Each light is surrounded by a ring denoting the scoring or avoidance trigger zone. In subsequent pictures only the Arena will be shown for clarity.

The experiment comprised two stages. Stage one was the evolution, in isolation, of several distinct behaviours using the same neural net architecture. Stage two involved combining these behaviours into subsumption style layers with the appropriate switching, and testing whether this produced acceptable results.

3.2 The Robot

The robot was modelled as a simple two wheeled differential-steer platform with two forward-facing light sensors. Each sensor had an angular range of 80 degrees, 60 one side of straight ahead and 20 the other. This gave a total visual field of 120 degrees with an overlap of 40 degrees in the centre where both sensors were active (*fig. 2*). Conditioning of the sensor inputs was not modelled; instead the luminance of the light source was varied. In addition the robot had knowledge of its position relative to the light sources in the arena so it had a form of distance sensing.

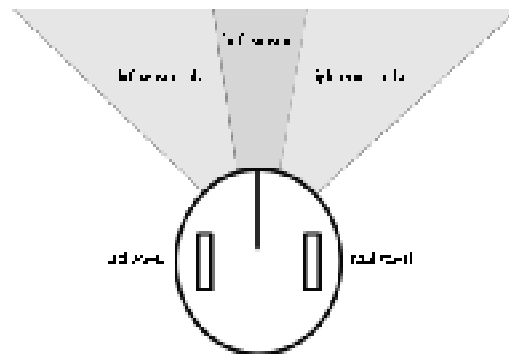


fig.2 the robot morphology

The controller was modelled as a fully-interconnected 5 neuron Continuous Time Recurrent Neural Network (CTRNN), a network model commonly used in these situations due to its fairly high biological plausibility and 'memory' capability. More information can be found in [1]. The latter means that the network's current state can be affected both by current sensory input and by past network states, taking it beyond the purely reactionary and enabling some complex and interesting behaviours. The left light sensor was attached to neuron 0, the right light sensor to neuron 1, the left motor to neuron 3 and the right motor to neuron 4. The motor outputs were scaled using a

sigmoid function to between +5 (forward) and -5 (backwards). 10% random noise was added to sensor inputs and motor outputs. The genome for a controller consisted of 5 node descriptor objects, each defining seven parameters (*fig. 3*):

- tau: learning rate of this node, 10 raised to the power of value in the range -1 to +4.
- bias: bias of this node, in the range -4 to +4.
- weight 0: connection weight from node 0, in the range -4 to +4.
- weight 1: connection weight from node 1, in the range -4 to +4.
- weight 2: connection weight from node 2, in the range -4 to +4.
- weight 3: connection weight from node 3, in the range -4 to +4.
- weight 4: connection weight from node 4, in the range -4 to +4.

Node	tau	bias	w0	w1	w2	w3	w4
0	1.2	0.5	0.1	0.2	0.3	0.4	0.5
1	2.1	1.2	0.2	0.3	0.4	0.5	0.6
2	3.0	2.1	0.3	0.4	0.5	0.6	0.7
3	4.0	3.0	0.4	0.5	0.6	0.7	0.8
4	5.0	4.0	0.5	0.6	0.7	0.8	0.9

fig. 3 graphical view of the complete genome

3.3 Stage 1: Evolution of Behaviours

In most cases a population of 200 individuals was evolved for 1000 generations, each individual having 5 trials of 5 simulated seconds. For simpler fitness functions the number of generations and trials was reduced. Each new generation was bred from the entire old population using fitness-proportional roulette wheel selection. Child genes had an equal chance of being from either parent and 100% chance of being mutated, using real values derived from a Gaussian distribution with a mean of 0 and a standard deviation of 0.3. It was found that larger values than this were detrimental to the evolution; indeed, an attempt to implement a kind of simulated annealing where the SD decreased as the generations increased failed despite its promise of wide search-space coverage. Mutations were limited to a range of +/-4, except for the tau value which was limited in the negative to -1 so as not to be

smaller than the integration step used in the simulator. The ‘bounce back’ technique was used at the range limits so, for example, a value of 3.95 plus a mutation of 0.12 would become 3.93.

The basic task the robot was set was to move towards a yellow light while avoiding red ones. Two approaches were envisaged; one using a collection of simple actions for red-light avoidance (such as ‘back’, turn left’ etc.) and one using a more sophisticated ‘light circling’ behaviour that, when run for a short time, should enable to robot to skirt around a red light more efficiently. The two are compared and discussed later in this report. To this end six basic behaviours were evolved:

Behaviour 1: ‘Seeking’

This behaviour caused positive phototaxis towards the centre of the yellow light (*fig. 4*). After [5], the fitness function was defined as (1-(final distance to the light/initial distance to the light)) + the proportion of time spent within 5 body radii of the light.

Network Analysis: In use, the left motor (LM) is on full permanently. The right motor (RM) is fully positive if the right sensor (RS) is activated, and goes fully negative if not. This results in the robot spinning in place to find the light, and then quick approach and tightly circling it.

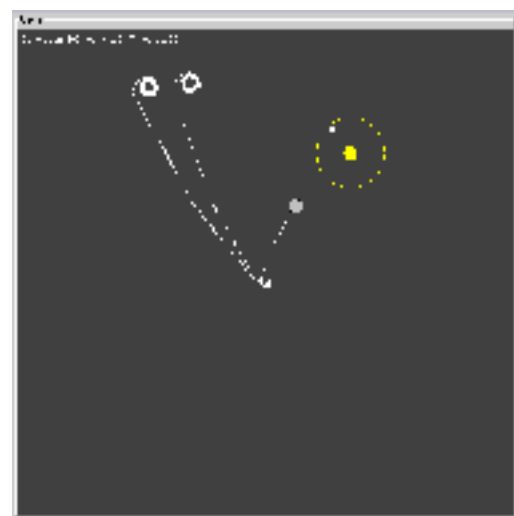


fig. 4 light seeking behaviour. Paths from two previous runs are also shown.

Behaviour 2: ‘Circling’

This behaviour caused positive phototaxis towards a red light and then circling around it at a constant distance (fig. 5). The fitness function for this behaviour was defined as $(1 - (\text{final distance to the light} / \text{initial distance to the light})) + \text{the proportion of time spent within a ring around the light between 5 and 7 body radii wide}$.

Network Analysis: When the LS is active the LM is fully on causing the robot to approach the light in an elongated curve. Once near the light the robot keeps it just out of its field of view, and with no sensor input the RM is on full and the LM on partially resulting in a circle of the required radius.

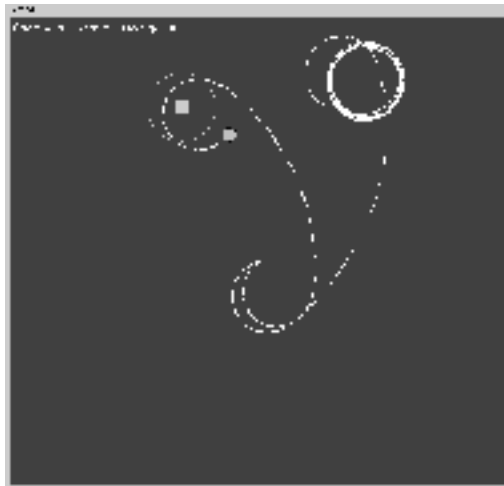


fig. 5 light circling behaviour

Evolved variations on circling using the same fitness function:

'Triangling'

In this behaviour the robot approaches the light in a more angular way and circles it in a triangular pattern (fig. 6).

Network Analysis: This behaviour displays the memory effect of the network as it depends on the decay time of node 0 to define the size of the triangle around the light. The RM is always active. Activation of the LS drives the LM causing the robot to approach the light. When it becomes inactive the robot continues for a short while in a straight line until the node 0 activation decays to about 0. At this point the LM becomes inactive, but as the RM is still active this causes the robot to make a sharp turn

towards the light, activating the LS momentarily causing a small forward movement. As this cycle repeats the robot describes a triangle around the light.

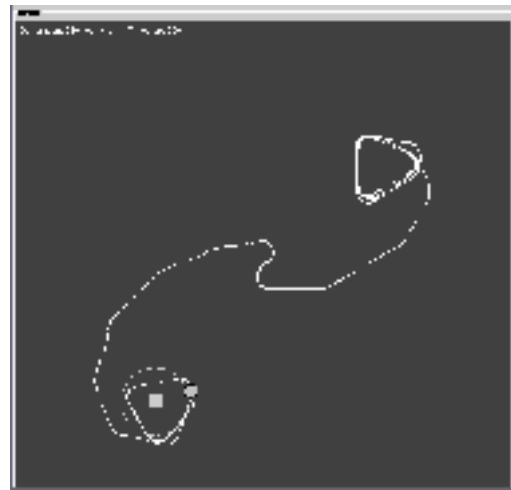


fig. 6 'triangling' behaviour

'Backing Up'

In this interesting behaviour the robot approaches the light with an angular path and circles it using a series of small forward jumps separated by short 'turning and backing up' movements (fig. 7).

Network Analysis: This behaviour is essentially a more extreme version of triangling. Node 0 decays a little faster and the LM briefly goes into reverse, causing the robot to turn and back up slightly. Once again the decay of node 0 defines the robot's path around the light.

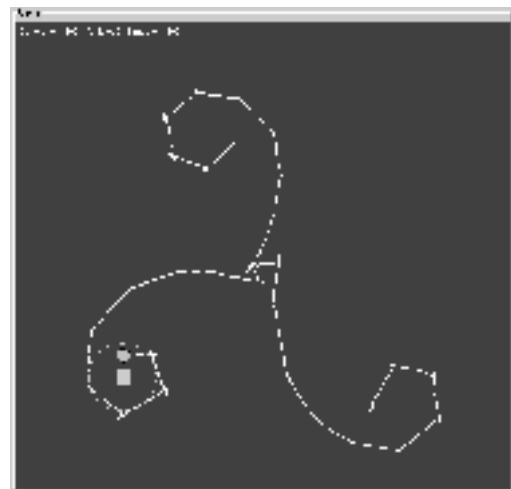


fig. 7 'backing up' behaviour

Behaviour 3: 'Backward'

Fast backward motion in a straight line. The fitness function was $LM < 0$ and bonus awarded if $RM=LM$.

Network Analysis: The sensors were not used in this behaviour although in certain evolved versions there was a small turning effect if the robot saw the light. Both LM and RM were fully negative.

Behaviour 4: 'Forward'

Fast forward motion in a straight line. The fitness function was $LM > 0$ and bonus awarded if $RM=LM$.

Network Analysis: The sensors were not used in this behaviour. Both LM and RM were fully positive.

Behaviour 5: 'Turn Anti-Clockwise'

In this behaviour the robot span in place in an anticlockwise direction. The fitness function was $RM - LM$, encouraging the maximum difference between the two motors.

Network Analysis: The sensors were not used in this behaviour. The LM was fully negative and the RM fully positive.

Behaviour 6: 'Turn Clockwise'

Spinning in place in a clockwise direction. The fitness function was $LM - RM$, encouraging the maximum difference between the two motors.

Network Analysis: The sensors were not used in this behaviour. The RM was fully negative and the LM fully positive.

3.4 Stage 2: Creation of Subsumption Architectures

3.4.1 Using Basic Behaviours

Two behaviours were used as the minimum necessary to prove the concept. Classic subsumption architecture (and other versions of evolved SA that I have seen) would have two distinct networks, one for each layer, but in this experiment only one network was used and 'soft layers' were created by substituting new weights and biases therein.

Layer 1: Seek. Using evolved seeking behaviour to search for the yellow light.

Layer 2: Avoid. When within 5 body radii of a red light this layer, a composite behaviour made of several simple evolved behaviours, would be triggered:

- ◆ Backwards for 10 time steps
- ◆ Turn (either always in one direction or randomly) for 5 time steps
- ◆ Forwards for 10 time steps
- ◆ Return to seeking behaviour

Note that in this behaviour the robot was not using the sensors while it was avoiding. Nevertheless the layer provided fairly robust avoidance of the red lights.

3.4.2 Using More Complex Behaviours

Layer 1: Seek. Using evolved seeking behaviour to search for the yellow light.

Layer 2: Avoid. When within 5 body radii of a red light this layer would be triggered:

- ◆ Backwards for 10 time steps.
- ◆ Light circling behaviour for 30 time steps (several durations were tried).
- ◆ Return to seeking behaviour

In this behaviour the robot needed to use the sensors during the light circling behaviour and initially the fact that there were multiple lights caused problems. This was overcome by giving the robot a simple form of attention, whereby it would only concentrate (i.e. receive input from) the most appropriate light. In seeking mode this was the yellow light and in avoid mode it was deemed the light closest to the robot. This problem was symptomatic of the fact that the evolutionary scenario differed from the test scenario and is discussed in more depth later in the report.

3.5 Results

In a simple test arena comprising four red lights in a square between the robot and the yellow light both subsumption architectures worked. The robot would correctly avoid the red lights and find the yellow light (*basic behaviours figs. 8 and 9, circling behaviour figs. 10 and 11*). Further tests were done with 10 randomly placed red lights and a yellow light that would jump to a new location once the robot reached it and these too were largely successful. Occasionally the robot would overrun a red light because the sensors were not checked during some of the 'avoid' behaviours but this could be easily rectified. However the robot would always find the yellow light and managed to

navigate successfully out of some tight corners.

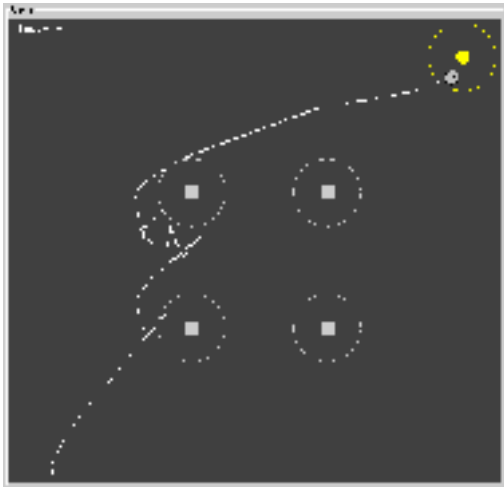


fig. 8 basic avoidance behaviour using AVOID layer: backwards for 10 time steps, turn ACW for 5, forwards for 10

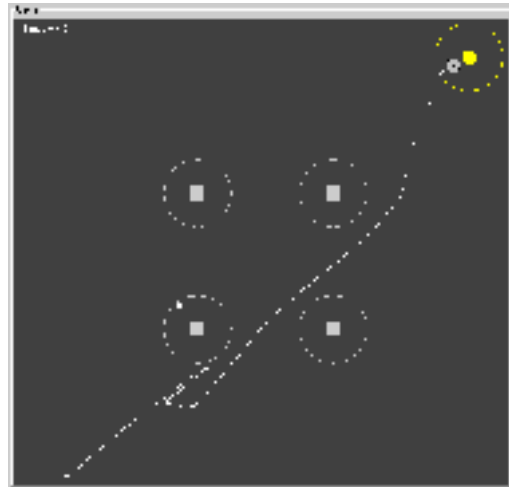


fig. 10 circling avoidance behaviour using AVOID layer: backwards for 10 time steps, circle light for 30. This time is so short the circling is not particularly apparent.

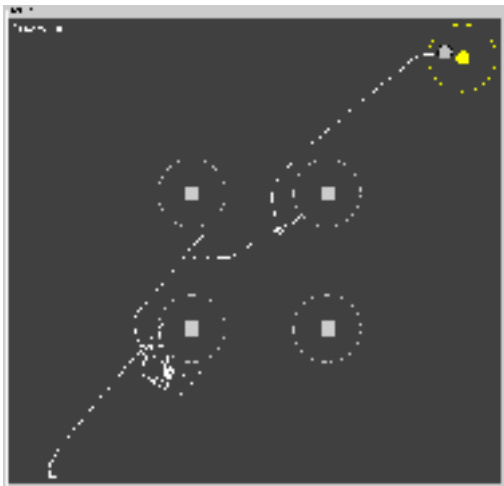


fig. 9 basic avoidance behaviour using AVOID layer: backwards for 10 time steps, turn randomly CW or ACW for 5, forwards for 10

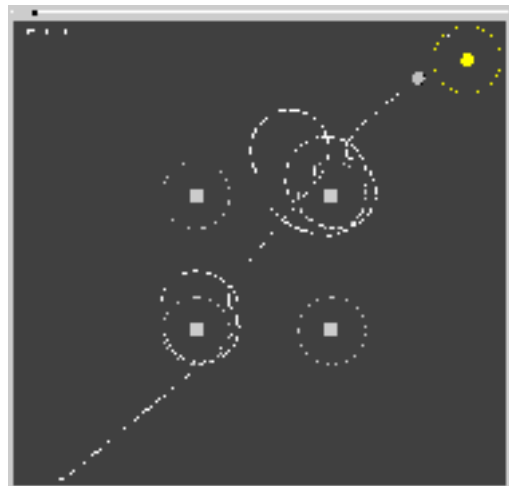


fig. 11 circling avoidance behaviour using AVOID layer: circle light for 150 time steps (no backwards behaviour). This time the circling is very apparent! Note the consistent turn direction of the robot esp. at light 2.

4.0 DISCUSSION AND CRITISICISM

4.1 Summing Up

The experiment can be viewed as a proof of concept and a success as far as it went. However the ‘avoid’ behaviour timings rely on the red lights being a consistent size and more complex sensing (at least) would be required to transfer this idea into the real world.

4.2 Is it worth it?

In terms of criticisms, the main question to ask here is: ‘Is it worth evolving these behaviours?’. After all most of them are so simple it would be quicker to program them by hand. I feel that in this simple example that may be so but as tasks get more complex time could be saved by using a handful of evolved behaviours rather than an involved web of hand-coded interactions. The important decision would be how to separate the required behaviours into efficient, easily evolvable parts. Subsumption architecture clearly has an advantage when it comes to complex sequential tasks, which are very difficult to define in terms of a fitness function. Using the architecture outlined in this paper it would be a simple matter to create a robot that would collect yellow lights while avoiding red ones, and when, say, five had been collected, dump them in a storage area and spin round manically on the spot for a few seconds in a completist dance of joy. Defining a fitness function to evolve this sequence would take an enormous amount of trial and error if it were possible at all. Indeed during this investigation I attempted to evolve a robot to perform the simple holistic task of collecting the yellow light while avoiding the red ones with the intention of comparing the two approaches but without success.

4.3 Memory Effect

The fact that only one network is used does negate the ‘memory effect’. For example, one behaviour may rely on a node with a long decay time and in another behaviour that node may be given a short decay time, meaning the robot would have to ‘start over’ when returned to the first behaviour. Possibly node activations could be stored, although when they were reset the robot would likely be

in a different place and they may no longer be valid. Certainly in this experiment no negative effects were observed.

4.4 Problems with Evolution

Evolution is notorious for finding unexpected answers and for being difficult to define in terms of fitness functions. Appendix 1 lists various evolutionary approaches that were tried in this experiment. I came up against two main problems. The first was that the robot would evolve to only circle the light in one direction, which meant that in the finished architecture if the robot approached the light from one side all was well but if from the other side it would inappropriately head off at right angles to it (see *fig. 11*). A useful further project would be to evolve a behaviour that would cause the robot to circle the light by the shortest path whichever side that might be. The second problem was the fact that the behaviours were evolved in isolation (i.e. with only one light) and when placed in an arena the robot would become confused. In fact, it would successfully circle a group of red lights at the correct distance but it would get stuck in this pattern and not approach the yellow light.

4.5 Adding the Concept of Attention

When confronted with the problem described above I considered what I would do in this situation and realised that my attention would likely be solely on a red light while I was avoiding it, and then switch back to the yellow light when the obstacle was passed. I decided to give the robot a simple form of attention. Attention in animals serves to filter out an enormous amount of unwanted visual data that would otherwise cause sensory overload. To this end the robot only received sensory information from the most relevant light in its current situation, namely the yellow light in ‘seek’ mode or the closest red light in ‘avoid’ mode. Although obviously a truly simple model this approach worked, a reminder that cognitive strategies employed by animals should not be forgotten when undertaking robotics research.

REFERENCES

- [1] **Beer, R.** (1995). "On the dynamics of small continuous-time recurrent neural networks". *Adaptive Behaviour* 3(4):469-509.
- [2] **Brooks, R. & Flynn, A.** (1989), 'Fast, Cheap, and Out of Control; A Robot Invasion of the Solar System', *Journal of the British Interplanetary Society* 42:10, 478-485.
- [3] **Brooks R.A.** (1991) "Intelligence without reason". In Mylopoulos, J & Reiter, R (Eds.), *Proceedings of 12th International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann.
- [4] **Cliff, D, Miller, G** (1996), "Co-Evolution of Pursuit and Evasion II: Simulation Methods and Results". In Maes, P. et al. (Eds.), *From Animals to Animats IV*, Procs. of the Fourth Int. Conf. on Simulation of Adaptive Behaviour, pages 506-515, MIT Press.
- [5] **diPaulo, E.** (2000). "Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions". In Meyer, J-A, Berthoz, A, Floreano, D, Roitblatt, H and Wilson, S (Eds.), *From animals to animats VI: proceedings of the 6th International Conference on Simulation of Adaptive Behaviour*, pages 440-449. Cambridge, MA: MIT Press.
- [6] **Jakobi, N.** (1998) "Minimal Simulations for Evolutionary Robotics". PhD thesis, School of Cognitive and Computing Sciences, University of Sussex.
- [7] **Jakobi, N.** (1997) "Half-baked, ad hoc, and noisy: minimal simulations for evolutionary robotics". In Husbands, P and Harvey, I, (Eds), *Fourth European Conference on Artificial Life*, pages 348-357. MIT Press.
- [8] **Sims, K.** (1994). "Evolving 3D Morphology and Behavior by Competition". In *Artificial Life IV*, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, pages 28-39, MIT Press.
- [9] **Tuci, E, Harvey, I and Quinn, M.** (2002). "Evolving integrated controllers for autonomous learning robots using dynamic neural networks". In Hallam, B, Floreano, D, Hallem, J, Hayes, G and Meyer, J-A, editors, *From Animals to Animats VII* Cambridge, MA, 4-9 August 2002. MIT press.
- [10] **I. Harvey, I., Husbands, P., Cliff, D., Thompson, A. and Jakobi, N.** (1997). "Evolutionary robotics: the Sussex approach". *Journal of Robotics and Autonomous Systems* v. 20 pp. 205-224.
- [11] **Nilson, N.J.** (1984). "Shakey the robot." SRI A.I. Center Technical Note 323.

APPENDICES

Appendix 1

A list of original elements in this project

Non-original elements:

- ◆ bare bones of sim
- ◆ robot and network architecture
- ◆ positive phototaxis fitness function
- ◆ figs 2 and 3 in the report

Original elements:

- ◆ gaussian mutation values
- ◆ 100% mutation
- ◆ bounceback at mutation limits
- ◆ larger mutation limits
- ◆ no mutation limits tried
- ◆ large network weights and biases tried
- ◆ evolution using only mutation tried (no crossover)
- ◆ toroidally wrapped arena tried
- ◆ 'simulated annealing' style evolution tried
- ◆ all fitness functions except positive phototaxis
- ◆ attempted to evolve holistic collect and avoid behaviour
- ◆ enhanced symmetry code to allow symmetrical network mutations, tried but eventually not used.
- ◆ swapping network values
- ◆ subsumption architecture version of sim
- ◆ defining and controlling subsumption layers
- ◆ programmed attention into robot
- ◆ all report except figs 2 and 3.

Appendix 2

The code
Java 1.4.2

Note: Many versions of this code were used in the experiments described. The one presented here is from the final subsumption version using light circling. Changes and additions to the original code are denoted therein by the word NEW in the documentation 'RobotSim' and 'Vehicle' have changed considerably and can be considered re-written for the purposes of this experiment.