Evolutionary Robotics: the Sussex Approach

I. Harvey, P. Husbands, D. Cliff, A. Thompson and N. Jakobi [1]

School of Cognitive and Computing Sciences
University of Sussex, Brighton BN1 9QH, UK

We give an overview of evolutionary robotics research at Sussex over the last 5 years. We explain and justify our distinctive approaches to (artificial) evolution, and to the nature of robot control systems that are evolved. Results are presented from research with evolved controllers for autonomous mobile robots; simulated robots, coevolved animats, real robots with software controllers, and a real robot with a controller directly evolved in hardware.

1 Why Evolutionary Robotics?

Humans are naturally evolved creatures, and the selection criteria under which our ancestors were judged did not include the ability to design complex systems — in fact, we are not very good at it. A common and useful trick to overcome our shortcomings is that of *Divide and Conquer*; a complex problem is decomposed into separate, less daunting, sub-problems.

However, the interactions between such sub-problems must be few in number, so that the human designers can temporarily ignore them while solving one sub-problem at a time. When it comes to designing such complex systems as a cognitive control system for a robot, there are at least three major problems.

- It is not clear *how* a robot control system should be decomposed.
- Interactions between separate sub-systems are not limited to directly visible connecting links between them, but also include interactions mediated *via the environment.*
- As system complexity grows, the number of potential interactions between sub-parts of the system grows *exponentially.*

[1] Email: inmanh, philh, davec, adrianth, nickja@cogs.susx.ac.uk

Classical approaches to robotics have often assumed a primary decomposition into Perception, Planning and Action modules. Many people now see this as a basic error [5]. Brooks acknowledges the latter two problems above in his subsumption architecture approach. This advocates slow and careful building up of a robot control system layer by layer. Each layer is responsible for a new robot behaviour, and is implemented by mechanisms (hardware or software) linking sensors to motors. Interactions between a new layer and its predecessors are limited (at least in theory) to simple inhibition, suppression or message passing from a 'higher' layer to a 'lower' one. This restriction, even if sometimes breached in practice, is a design heuristic intended to minimise unintended interactions.

Brooks' subsumption approach is explicitly claimed to be inspired by natural evolution. Initially simple behaviours are 'wired into' a robot, and thoroughly debugged, before adding the next behaviour. This incremental approach echoes the phylogenetic history of complex cognitive creatures, including humans, some of whose behaviours we are trying to emulate in robots. Nevertheless, each new layer of behaviour is wired in by hand design; and despite the heuristics used to minimise interactions between layers, it seems that unpredictable interactions become insuperable when the number of layers gets much beyond single figures — published architectures usually have less than 10 layers.

So an obvious alternative approach is to explicitly use evolutionary techniques to incrementally evolve increasingly complex robot control systems, rather than attempt to figure out each evolutionary step by hand design. Unanticipated and elusive interactions between sub-systems, though tricky or perhaps impenetrable for human designers, need not directly bother an evolutionary process where the only benchmark is the behaviour of the whole system. This is the approach taken in the Evolutionary Robotics research at Sussex, within the Evolutionary and Adaptive Systems group.

Other individuals and groups have taken a somewhat similar evolutionary approach, and the next section briefly surveys their work. Thereafter this paper is primarily an overview of our work at Sussex. We discuss what artificial evolutionary techniques are appropriate, followed by discussion of which *classes* of robot control systems are appropriate for evolutionary design. Relationships between simulations of a robot and the real thing are covered, as is the problem of evaluation within a noisy and uncertain environment. Then a number of Sussex projects in this area are described, with both simulations and real robots, illuminating further methodological issues.

2 Other related work

Important work on an evolutionary approach to agent control using neural networks has been done by Beer and Gallagher [3]. They explore the evolution of continuous-time recurrent neural networks as a mechanism for adaptive agent control, using as example tasks chemotaxis, and locomotion-control for a six-legged insect-like agent. The networks are based on the continuous Hopfield model [22], but allow arbitrary recurrent connections. They used a standard genetic algorithm (GA) to determine neuron time constants and thresholds, and connection weights. A fixed number of network parameters are encoded in a straightforward way on bitstring 'genotypes'. They report success in their objectives; in the case of locomotion control, controllers were evolved that in practice generated a tripod gait (front and back legs on one side in phase with the middle leg on the opposite side). This was achieved both with and without the use of sensors which measured the angular position of each leg.

Beer [4] develops a dynamical systems perspective on control systems for autonomous agents, influenced by early work in Cybernetics [2]. In further developments of their evolutionary approach, Yamauchi and Beer [38] evolve networks which can control autonomous agents in tasks requiring sequential and learning behaviour. The prime focus of Beer's use of artificial evolution is as a means of developing models of simple nervous systems in order to test theories of how real nervous systems may work.

Colombetti and Dorigo [10] use Classifier Systems (CSs) for robot control. In this work the ALECSYS implementation is used to build a hierarchical architecture of CSs — one for each desired behaviour, plus a coordinating CS. Results are reported which have been generated in simulations, and then transferred to a real robot.

Floreano and Mondada [14], were able to run a GA on a real robot in real time, rather than a simulation; they used the *Khepera* robot developed at Lausanne. The GA set the weights and thresholds in a simple recurrent network where every sensory input was connected to both motor outputs. The task was to traverse a circular corridor while avoiding obstacles, and this work demonstrates that with well-designed equipment it is possible to avoid the problems associated with simulations.

In related work with Nolfi [30] similar experiments were performed to compare evolving in simulation with using a real *Khepera* robot. The problem of transferring control systems evolved in simulation over to the real robot was discussed. One approach advocated was that of continuing the evolution for some time further on the real robot, to compensate for any inadequacies in the simulation. In contrast to Beer's primarily scientific motivation, this work

primarily emphasises the practical engineering problems.

Koza used the technique of Genetic Programming to develop subsumption architectures [5] for simulated robots engaged in wall-following and box moving tasks [27]. Craig Reynolds [31] also uses Genetic Programming to create control programs which enable a simple simulated moving vehicle to avoid collisions. He comments that these solutions are *brittle*, vulnerable to any slight changes or to noise. In further work where the fitness-testing includes noise, he reports that the brittleness problem is overcome, and only compact robust solutions survive [32].

3 Artificial Evolution for Robots

Genetic Algorithms (GAs) are the most common form of algorithm which uses evolutionary ideas for search, optimisation and machine learning — the fields covered in [16]. However, recently concerns have been voiced [12] to the effect that GAs, when originally proposed by Holland [21], were intended as algorithms for complex *adaptive* systems, and their use for function optimisation is perhaps not best suited to their strengths. Evolutionary Robotics typically needs adaptive improvement techniques, rather than optimisation techniques, and this critical but little-understood distinction needs to be made clear.

The majority of published GA work, both applications and theoretical analysis, refers to optimisation problems which can be seen as search problems in some high-dimensional search space, of known (but usually enormous) size. Each dimension typically corresponds to some parameter which needs to be set, which is coded for on a small section of the genotype, a 'gene'. What such optimisation problems share is the well-defined finite nature of the search space. This allows the choice of some genotype coding, such that a genotype, often binary, of fixed length can encode any potential solution within the space of possibilities. In robotics, a genotype specifies the characteristics of a control system.

The GA works with a population of such genotypes, each of which is evaluated in terms of how good is the potential solution that it encodes. Genotypes which happen to be fitter in the current population (which initially may be generated at random) are preferentially selected to be parents of the next generation. Offspring inherit genetic material from their parents; usually inheriting part of this from each of two parents. A small number of random mutations are applied to the genotypes of the offspring. This cycle of selection, reproduction with inheritance of genetic material, and variation, is repeated over many generations, with the population remaining the same size as old members are replaced by new ones, typically the offspring of those members demonstrated

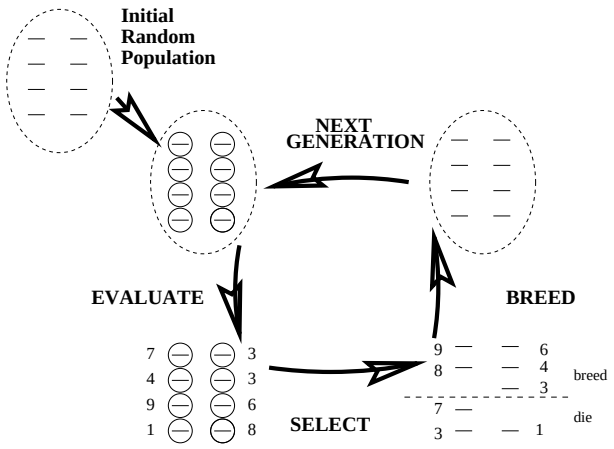

Fig. 1. *The Genetic Algorithm Cycle.*

to be fitter.

A GA optimisation problem has typically been seen as starting with a population of random points effectively spanning and crudely sampling the whole search space. Successive rounds of selection, reproduction and mutation are intended to focus the population of sample points towards fitter regions of the space, homing in on an optimum or near-optimal region. Theorems such as the Schema Theorem [16], intended to show the circumstances under which GAs can be expected to produce the desired results, rely on these assumptions. One consequence of this approach has been the primary reliance on recombination as the genetic operator, which mixes and matches information from different samples in order to move towards regions of expected higher fitness; mutation is typically relegated to the rôle of a background genetic operator.

However, some domains — including much of evolutionary robotics — do not always fall into this convenient picture of a fixed-dimensional search space. Standard GA theory does not necessarily then apply.

In evolutionary robotics a genotype will specify the control system (possibly more, see below) of a robot which is expected to produce appropriate behaviours when tested in its environment. However the evaluation of fitness is in terms of the robot's *behaviour*; for all except toy problems there is unlikely to be any obvious way to predict in advance the necessary complexity of control system for a given behaviour. Hence it is often appropriate to choose a genetic encoding which allows for, and encodes the characteristics of, a variable number of components. This has the added benefit of making incremental evolution possible: initially simple robots are evolved under a selection criterion based on simple tasks, and then the same robot population is allowed to increase in complexity in response to a gradual and continuing increase in task complexity. Such incremental evolution calls for *GAs as adaptive improvers* rather than *GAs as optimisers*.

The litmus test for making this distinction is: *can a fitness function to be maximised (or a set of them) be fully and unchangeably defined* **before** *one starts from scratch?* If the answer is yes, then optimisation techniques are called for. In the case of natural evolution, of course, the answer is no — though *a posteriori* one can always posit fitness functions (in terms of what traits were conducive to survival and reproduction) specifying what was selected for at different periods of the evolutionary history of some species. To give an analogy, over the long term aircraft design has been an evolutionary process in this sense: there is a line of descent from the Wright brothers to the Airbus, and in all eras of flight design the designers were optimising for their own short-term criteria — but the Wrights were in no position to foresee the requirements of plane travel in the 1990s.

Of course, any specific robot task can be posed in the form of an optimisation problem. However, most of the Sussex work has been done with the intention, sooner or later, of incremental open-ended evolution. GAs when applied to search spaces of varying dimensionality need a different framework from those used for standard optimisation problems. Species Adaptation Genetic Algorithms (SAGA) were developed as this framework, specifically for the class of open-ended problems where the task-specification will inevitably be changed (by the experimenters, by circumstances ...) in unforeseen ways *after* the starting point.

4 SAGA

The conceptual framework of SAGA was introduced by Harvey in 1991 in order to try to understand the dynamics of a GA when genotype lengths are allowed to increase [19]. It was shown, using concepts of epistasis and fitness landscapes drawn from theoretical biology [26], that progress through such a genotype space will only be feasible through relatively gradual increases in genotype length. A general trend towards increase in length is associated with the evolution of a *species* rather than global search — the population will be largely genetically converged.

Evolutionary search can be thought of as searching around the current focus of a species for neighbouring regions which are fitter (or in the case of neutral drift, not less fit) while being careful not to lose gains that were made in achieving the current *status quo*. The population can be visualised as moving around on a mountainous fitness landscape, where the altitude represents fitness, and movements measured in horizontal directions loosely represent movements in genotype space; the closer two points are on the landscape, the more similar are their genotypes. As generation succeeds generation, selection is a force which tends to move a population up hills, and keep them centred around a

local optimum; whereas mutation produces offspring exploring outwards from the current population.

To increase exploration, mutation rates should be increased; but if they become too high then the population will disperse completely, losing the current local optimum or hill-top, and the search will become random. For any given selection pressure, there is a maximum rate of mutation which simultaneously allows the population to retain a hold on its current hill-top, while maximising search along relatively high ridges in the landscape, potentially towards higher peaks [13]. To maintain a selection/mutation balance requires maintaining a constant selective pressure, which normal fitness-proportionate selection does not provide. Hence in SAGA rank-based or tournament selection is used to achieve this (the expected number of offspring of any member of the population should depend on its current ranking within the population, rather than the ratio of its fitness to the average fitness); and mutation rates should be maintained at a rate of about 1 mutation per genotype [18].

5 What building blocks for a control system?

We are relying on evolution for the design of a control system, but we must choose appropriate building blocks for it to work with. Some have advocated production rules (Classifier Systems are the GA version of these [16]). Some propose LISP-like programming languages [27]. Brooks [6] has proposed using Koza's ideas applied to a high-level language GEN which can be compiled into BL (Behavior Language, a LISP-based language for programming autonomous mobile robots). Beer [3] has used dynamical neural networks. It is only this last approach that is advocated here, as the other approaches rely on relatively high-level languages.

There is good reason to suggest that the primitives manipulated by the evolutionary process should be at the lowest level possible. Any high level semantic groupings inevitably incorporate the human designer's prejudices. Primitives that are equivalent to those of a programming language give rise to a rugged fitness landscape with steep precipices. A program taken as a linear string of characters *can* be treated as a genotype, but typically a single mutation in a working program is fatal — a 'precipice'. Genetic programming [27] relies on recombination rather than mutation and hence sidesteps this problem, but the typical reliance on large populations for a relatively small number of generations is not compatible with the longterm incremental evolution approach.

At Sussex we largely fall into the same camp as Brooks [5] in dismissing the classical Perception, Planning, Action decomposition of robot control systems. Instead we see the robot as a whole — body, sensors, motors and control sys-

tem or 'nervous system' — as a dynamical system coupled (via the sensors and motors) with a dynamic environment [3]. This coupled interaction generates the robot behaviour which is to be evaluated. The control system should itself be a dynamical system, and hence the genetic specification of this should be at the level of the primitives of a dynamical system.

One convenient form of dynamical system is an (artificial) neural net (NN). If this takes the form of a feedforward net from sensors, perhaps via hidden nodes, to motors, then such a control system would have no internal state, and be capable only of generating reactive behaviour. However if a recurrent net is used, with temporal specifications to determine the timescales on which internal feedback is propagated, then non-reactive behaviour is also possible. There are two distinct questions that might be asked at this point. The first is whether *in principle* this class of networks is able to produce some desired behaviour. Continuous time recurrent neural networks can be shown to be a class of dynamical systems capable *in principle* of replicating to an arbitrary degree of accuracy the dynamical behaviour of any other dynamical system with a finite number of components [15]. In our work we sometimes use Dynamical Recurrent Neural Networks (DRNNs) where instead of the continuous model there are discrete events, namely activation changes at nodes; the repercussions of such events at connected nodes occur after time-delays associated with each link. It can similarly be shown that this class of DRNNs with time-delays can *in principle* replicate any other dynamical system to arbitrary accuracy [2]. Such a proof of principle, once accepted, has zero relevance to the second important question of how easy or difficult it is to find through evolution a network giving the desired behaviour. It is this second question which is the focus of our research.

Our DRNNs are equivalent (only simple transformations are needed) to the class of networks using Augmented Finite State Machines (AFSMs) that occur in Brook's subsumption architectures [3]. One significant difference from subsumption architecture is that we deliberately introduce internal noise at the nodes of DRNNs. This has two effects. First, it makes possible new types of feedback dynamics, such as self-bootstrapping feedback loops and oscillator

[2] A sketch of a proof: McCulloch and Pitts neurons [28] can be used to make a Universal Turing machine, and hence implement a program. The computation of Turing machines without a real clock cannot act as a dynamical system, but the arbitrary time-delays of DRNNs can be used to make a clock of any precision. Hence a network of such neurons can replicate in real time the temporal behaviour of any described finite dynamical system.

[3] Both take the form of a network of nodes whose non-temporal properties can be translated between DRNN and AFSM form. The crucial temporal properties of a DRNN are derived from time-delays on links between nodes; but these delays could be 'moved within the nodes' and hence play the same role as the timers within ASFMs [5].

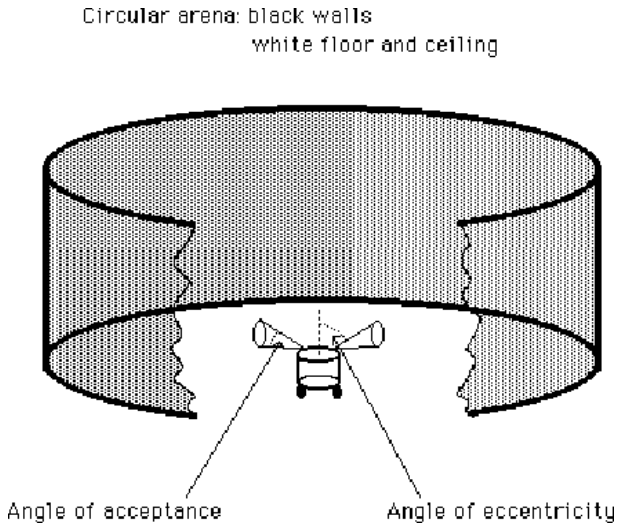

Fig. 2. *The arena for a simulated robot.*

loops, which would not initiate themselves without the noise. Second, it helps to make more smooth the fitness landscape on which the GA is operating; a mutation which deletes a link or a node is comparable to a lot of noise, and hence the behaviour of the system before and after such a mutation is more closely correlated in the presence of noise than it would have been without the noise.

With this in mind, our genotypes need to specify a finite number of nodes, together with their thresholds or other details of a non-linear activation function transforming summed node inputs into node outputs; and connections between nodes, specifying weights and time-delays on the links. This can be generalised to include weight-changing rules, although to date our experiments have been with fixed weights. A specified subset of the nodes are designated as input nodes, receiving sensory inputs; similarly there is a set of output or motor nodes. Other nodes ('hidden') can be arbitrary in number, and genetically specified links are not necessarily restricted to feedforward ones.

6 Evolutionary Robotics in Simulation

Initial experiments done at Sussex [7] used simulations of a round, two-wheeled, mobile robot with touch sensors and just two visual inputs — simulated photoreceptors, with (genetically specified) angles of acceptance, and of eccentricity relative to the frontal direction of the robot. One task to be achieved was to evolve control systems (and visual morphologies) which allowed the robot to reach the centre of a simulated circular arena, with white walls and black floor and ceiling (Fig. 2). The visual input to each (simulated) photoreceptor was calculated on the basis of a square cross-section to its visual field.

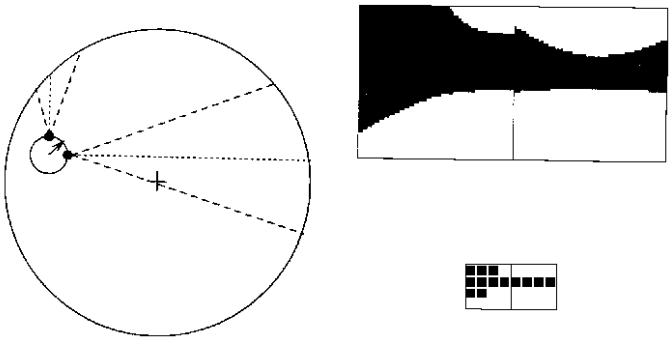

Fig. 3. *Illustration of the ray-tracing system. The left-hand figure shows the robot's position and orientation within the cylinder, which has black walls and white floor and ceiling. At upper right is a pair of relatively high-resolution images, traced from the robot's position inside the cylinder. The lower-right figure shows the two* 4×4 *images traced prior to averaging. The final two photoreceptor brightness levels are derived by averaging the* 4×4 *images.*

If only a single ray was traced for each pixel aliasing problems would arise; hence sixteen rays (in a regular 4×4 grid) are traced for each pixel, the result interpreted as giving a grey-level in the range $[0, 16]$; see Fig. 3.

The motion of this simulated robot was calculated to a considerable degree of verisimilitude, based on measurements taken from a real robot. Collisions with walls were modelled, as were noisy motor properties.

The networks used had a fixed number of input nodes, one for each sensor, and a fixed number of output nodes, namely two attached to the left and two to the right motors. All the nodes are noisy linear threshold devices with outputs in the range $[0.0, 1.0]$, except for the two units for the motors that give a signal in the range $[-1.0, 1.0]$. This range of signal arriving at a motor is divided into 5 segments, giving 5 possible speeds for the wheel: full ahead, half ahead, stop, half-speed reverse, full-speed reverse.

In addition to the fixed number of input and output nodes, there is an arbitrary number of internal or 'hidden' nodes. Within the networks there are two separate types of connection: normal (or excitatory), and veto (or inhibitory), with separate threshold values for each. A normal connection is a weighted link joining the output of one unit to the input of another. A veto connection is a special infinitely inhibitory link, or gating link, between two units.

Each of the following experiments was run for 50 to 100 generations, and with a population size of 40 or 60. The crossover rate was set at 1.0, while the mutation rate was of the order of 1 bit per genotype. The task is set implicitly by the evaluation function, and the robots are rated on the basis of how much time they spent at or near the centre of the arena. This was done by measuring the distance d of the robot from the centre, and weighting this distance by a

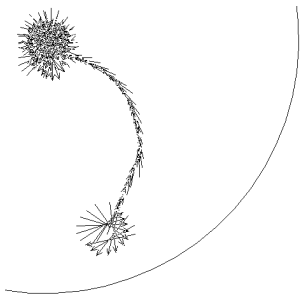

Fig. 4. *Typical path of a successfully evolved robot, which heads fairly directly for the centre of the room and circles there, using input from 2 photoreceptors. An arrow shows the diameter of the robot and the direction it is facing at a single time step; the succession of superimposed arrows indicates the path.*

Gaussian $\mathcal{G}$ of the form:

$$\mathcal{G} = \exp(-d^2/c)$$

for some constant c, which ensures that $\mathcal{G} \approx 0$ for positions of the robot near the perimeter of the arena. Over 100 timesteps the value of this Gaussian was summed, to give the final score. The robot was always started near the perimeter, facing in a random direction.

Two different successful evolved control systems, termed C1 and C2, have been described elsewhere [9]. In both cases, the robots make a smooth approach towards the centre of the arena, and then circle there, either on the spot or in a minimum radius circle. Before analysing these results fully, it had been speculated that the control system might be recognising that the centre of the arena had been reached by using the absolute level of the light input at the centre; there it is at a maximum, as far more of the white wall and floor are within the angle of acceptance of each photoreceptor than at any other position nearer the wall. In these first experiments the wall had been fixed at a height of 15 units, the diameter of the arena being 40.

With this in mind, a more difficult task was set up for the next set of experiments, where the height of the wall could vary over one order of magnitude, from 4 to 40 units; the diameter of the arena remained at 40. The full range of possible heights was divided into 10 equal sections, and each robot was given 10 trials, with a height of wall chosen at random from each in turn of the 10 sections. In this way it could be ensured that it was tested across the full range. Thus no use could be made of absolute light values; as is usual in these experiments, the evaluation of the robot was based on the *worst* score it obtained across its trials. Success was similarly achieved in this more difficult navigation task.

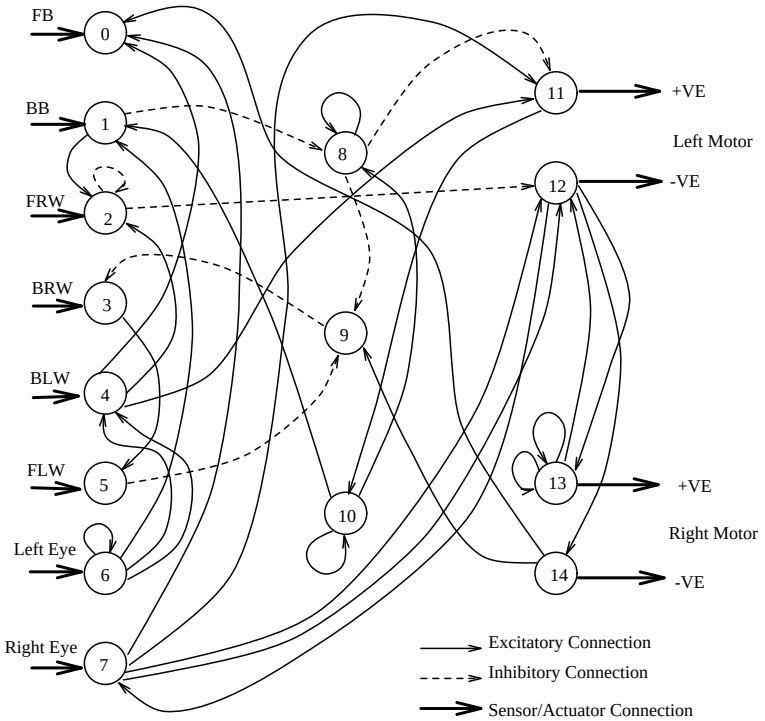

Fig. 5. *C1 control network. The left-hand column are units originally designated as input units: FB=Front Bumper; BB=Back Bumper; FRW=Front Right Whisker; BRW=Back Right Whisker; BLW=Back Left Whisker; FLW=Front Left Whisker. Right-hand column shows output units to the motor units for left and right wheels. Centre column shows 'hidden units'.*

Typical behaviour for a particular network C1 is shown in Fig. 4. The robot starts at the edge, moves to the centre, and then stays there, spinning on the spot; the evaluation function does not penalise continued movement. The vision chromosome had evolved so as to specify that the two photoreceptors have 45^o acceptance angles, and face just 6^o each side of the straight-forward direction. The network for C1 is shown in Fig. 5.

After a network such as C1 has evolved, we can analyse it by first identifying any redundant units and connections (e.g. unit 0 has no outputs and can be disregarded). Identification of particular sensory-motor pathways which mediate identifiable patterns of behaviour is aided by studying records, recorded over a trial, of inputs, outputs, activity levels of nodes, and such robot variables as speed, orientation and distance from centre of the arena. This allows identification of those nodes which are largely inactive; we can then produce a simplified diagram of the network, easier to interpret, in which such nodes are eliminated. The result of eliminating redundant nodes is shown in Fig. 6.

One should note that early on in the evolutionary process strategies evolved such that the robots avoided the walls through use of visual signals; hence during later generations none of the touch-sensors (whiskers and bumpers) were used for detecting collisions. From Fig. 6 it can be seen that evolution

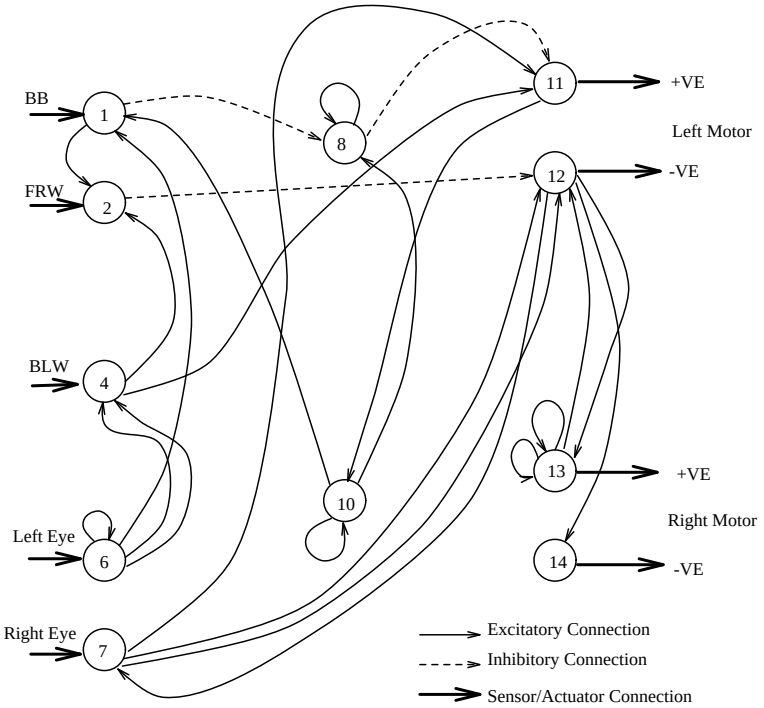

Fig. 6. *The same C1 network as in the previous figure, but with redundant and non-visual units deleted.*

has opportunistically taken over some of the otherwise-redundant tactile input units (nodes 1, 2 and 4) to act as 'hidden' units, or 'interneurons', used in sensory-motor pathways between visual inputs and the motor outputs.

In [23] a successfully evolved network from this experiment was analsyed in depth from a dynamical systems perspective. The analysis is in terms of an appropriate state space, and it is shown that within the arena environment the entire state space gives a single basin of attraction (or under some conditions two basins) for a point attractor which corresponds to the desired behaviour. This is robust in the face of noise. In other words, the evolved robot is guaranteed to succeed at its task — the visuo-motor couplings, via the network dynamics and the visual structure of the robot's world, are in perfect harmony relative to the evaluation task.

7 The Gantry

Even with the basic circular arena, ray-tracing in simulation was computationally expensive. For dynamic real-world domains with noisy lighting conditions it was necessary to investigate performing the whole evolutionary process with a real robot, moving in the real world. Artificial evolution requires the evaluation of large numbers of robot control systems, so it is advisable to automate

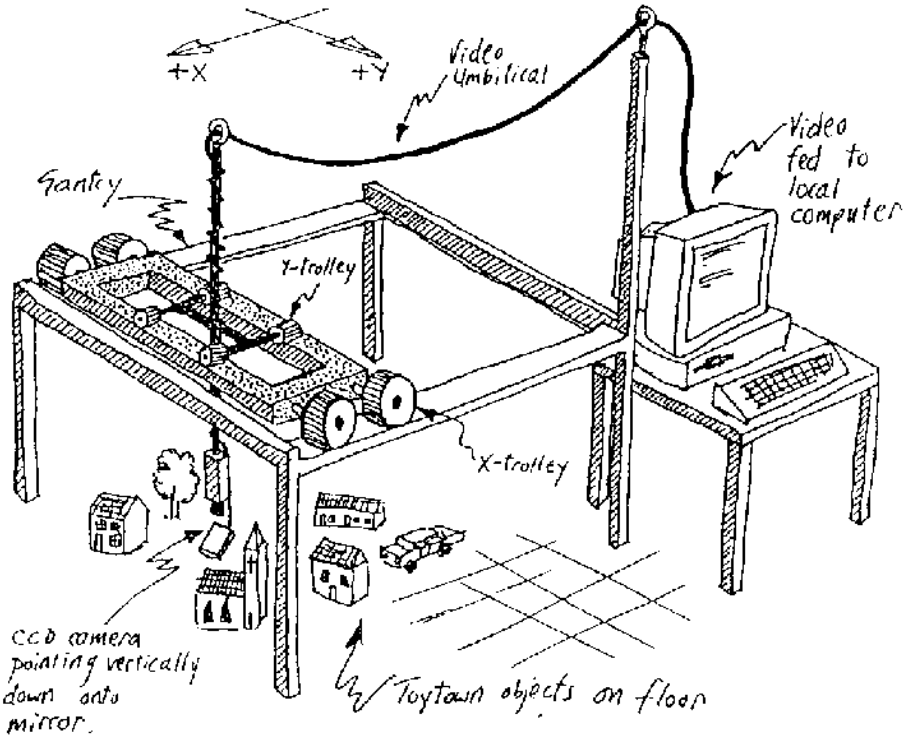

Fig. 7. *The Gantry.*

evaluations. With navigation tasks, the absolute position of the robot during its trials should be available to an overseeing program which evaluates the robots for the genetic algorithm — while of course this information should not be available in any way to the individual robot control systems. Automatic re-positioning of the robot at fixed or random positions for the start of each trial is also necessary. We have developed a specialised piece of visuo-robotic equipment fulfilling these requirements — the gantry-robot.

The gantry-robot occupies a position conceptually partway between a physical mobile robot with two wheels and low-bandwidth vision, and the simulation thereof within a simulated environment. The robot is cylindrical, some 150mm in diameter, and moves in a real environment — the term 'robot' is here used to refer to that part which moves around and has the sensors mounted on it. Instead of two wheels, however, the robot is suspended from the gantry-frame with stepper motors that allow translational movement in the X and Y directions, relative to a co-ordinate frame fixed to the gantry (see Fig. 7). Such movement, together with appropriate rotation of the sensory apparatus, can be thought of as corresponding to that which would be produced by left and right wheels. The visual sensory apparatus consists of a CCD camera pointing down at a mirror inclined at 45° to the vertical (see Fig. 8). The mirror can be rotated about a vertical axis so that its orientation always corresponds to the direction the 'robot' is facing. The visual inputs undergo some transformations en route to the control system, so that the CCD image is subsampled into 3

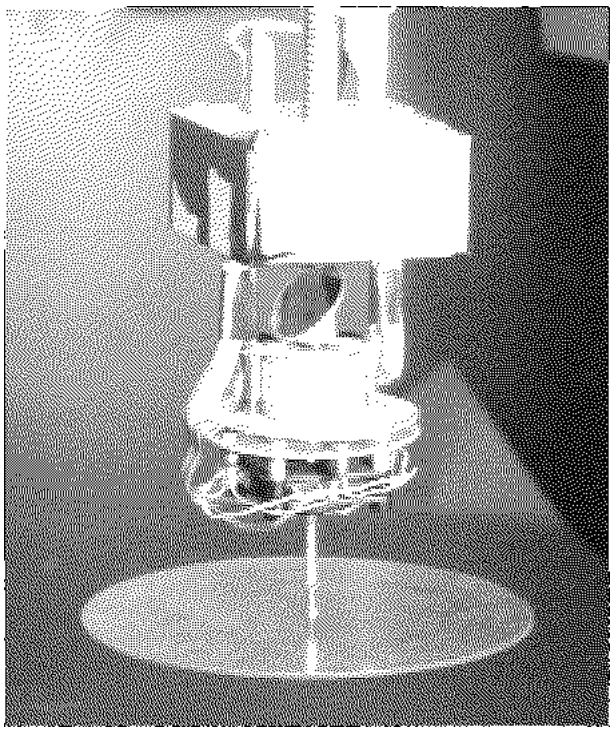

Fig. 8. *The gantry-robot. The camera inside the top box points down at the inclined mirror, which can be turned by the stepper-motor beneath. The lower plastic disk is suspended from a joystick, to sense bumps.*

or more (genetically specified) virtual photoreceptors, or receptive fields. Each such receptive field is a circle, whose centre and radius are genetically specified, within the field of view of the CCD image; the subsampling routine returns a single scalar value for the average intensity within this circle. The hardware is designed so that these transformations are done completely externally to the processing of the control system.

We used the same networks and genetic encoding schemes as in the simulation work (for full details see [7]). This was mainly because we have a detailed understanding of their properties and wanted to see how well they transferred to real world tasks. We used a genetic algorithm acting on pairs of 'chromosomes' encoding the network and visual morphology of a robot control system. The network chromosome is of variable length, allowing (in principle) networks of arbitrary complexity to evolve. The visual chromosome is in these experiments a fixed length bit string encoding the position and size of three visual receptive fields as described above; this can be extended to allowing a variable number of receptive fields.

With the walls and floor of the gantry environment predominantly dark, initial tasks were navigating towards white paper targets. In keeping with the incremental evolutionary methodology, deliberately simple visual environments were used initially, as a basis to moving on to more complex ones. The follow-

ing sequence of tasks of increasing difficulty were used:

(i) Forward movement.
(ii) Movement towards a large target.
(iii) Movement towards a small target.
(iv) Distinguishing a triangle from a square.

Each of these tasks can be seen as an individual *optimisation* problem — though differing from conventional optimisation problem-solving in that here the starting point for a further task is a population that achieves the previous task. The succession of tasks has here been deliberately designed by the experimenters to be in some order of increasing difficulty (the possibility of taking humans out of the loop is touched on in the next section on co-evolution). One should expect (evolutionarily) later competencies to be built on top of earlier (sub-)competencies, but this should not be confused with the classical assumption (which we rejected earlier) that such sub-competences will necessarily be implemented by individual modules within the control system.

For the exploratory experiments reported here, an initial randomly generated population of size 30 was judged by eye on the intuitive criterion of 'interesting' behaviour — a form of 'breeder selection'. Two members displayed forward-moving behaviour, which altered in character when the white target was within view of the visual system, and one of these two was selected. The informal criterion of 'interestingness' allowed a clear choice, whereas the 'official' evaluation function used thereafter did not give clear preferences on this initial random population, as the scores it gave there were dominated by noise.

From a population formed of mutated clones of the selected one, the evolutionary process was run using an evaluation function which favoured movement towards one long wall of the environment which was covered with white paper. The remainder of the environment was dark, and each trial started with the robot at a far corner, with various different starting orientations. High scores were achieved after about 10 generations.

7.1 Small Target

The experiment continued from the stage already reached, but now using a much narrower target (22cm) placed about 2/3 of the way along the same wall the large target had been on, and away from the robot's starting corner, with evaluation $\mathcal{E}_2$:

$$\mathcal{E}_2 = \sum_{i=1}^{i=20} (-d_i) \qquad (1)$$

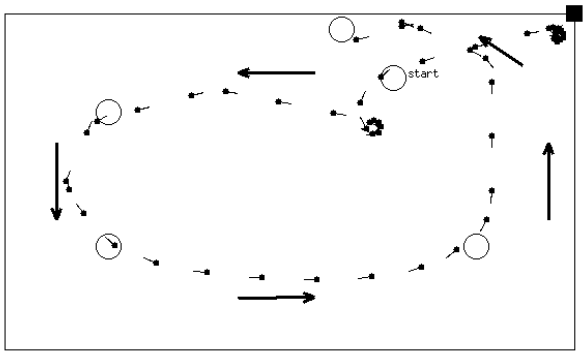

Fig. 9. *Tracking behaviour of the evolved control system. The unfilled circles show the position of the target at a number of points on its path (starting position indicated). The arrows roughly indicate the path of the target.*

where d_i is the distance of the robot from the centre of the target at one of the sampled instances during an evaluation run; the sampling was at 1 second intervals over a 20-second trial. Again, the fitness of an individual was set to the worst evaluation score from four runs with starting conditions as in the first experiment. The first experiment with the large target had run for 12 generations, and this 12th generation was used as the initial population for the small target (i.e. there was incremental evolution on top of the existing behaviours). Interestingly, individuals that had performed best at the previous task did very poorly on the new task.

Within six generations a network architecture/visual morphology had evolved displaying robust behaviour. This control system was tested from widely varying random starting positions and orientations, with the target in different places, and with smaller and differently shaped targets. Its behaviour was able to cope with all these conditions for which it had not explicitly been evolved. Indeed, it was capable of following a moving target, which can be thought of as a generalised version of static target approaching — see Fig. 9.

7.2 Rectangles and Triangles

Now, using the population that had become competent with the small target, the task was made significantly more difficult. Two white paper targets were fixed to one of the gantry walls; one was a rectangle 21cm wide and 29.5cm high, the other was an isosceles triangle 21cm wide at the base and 29.5cm high to the apex. The robots were started at four position and orientations near the opposite wall such that they were not biased towards either of the two targets. The evaluation function was changed so as to add bonus points for getting close to the triangle, but subtract penalty points for nearing the rectangle — see Fig. 10.

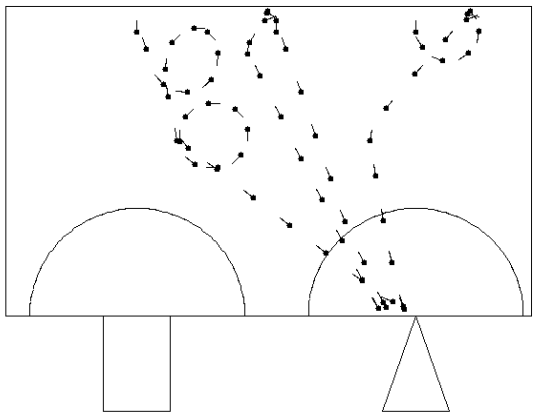

Fig. 10. *Distinguishing triangular target from rectangular target. In this view from above the rectangular arena, the positions of the rectangle and triangle are loosely indicated (in reality they are on the walls of the arena, in a plane perpendicular to the figure). The semi-circles indicate the regions where penalty and bonus scores are accumulated. Trajectories of 4 trials, from a variety of starting positions, are shown.*

As before, we started evolution for the new task with a population comprised of the last generation of the previous task, the small target experiment. After some 10 further generations, fit individuals emerged capable of approaching the triangle, but not the rectangle, from each of the four widely spaced starting positions and orientations. The successful networks were of a similar complexity to those shown earlier. Whereas the fit control systems for the previous experiments only made use of one visual receptive field at a time (out of the three available), those successful at this new task were seen (with *a posteriori* analysis) to have made use of two simultaneously. The control systems, both visual morphology and networks, evolved such that robots rotated on the spot when both visual inputs were low; moved in a straight line when only one visual input was high; and rotated when both inputs were high. The two relevant receptive fields were so arranged such that when the robot was turning, and its visual field sweeping across the scene horizontally, they would cross a vertical dark/light boundary nearly simultaneously; whereas when crossing a dark/light boundary which slanted from bottom-left to top-right, there was a significant period when one receptive field was in the dark and the other in the light. This was sufficient for the network to switch the motors from rotating the robot to propelling it in a straight line. It would often initially fixate on the edge of the rectangle but as it moved towards it both visual signals would go high, resulting in a rotation towards the triangle.

Hence if one were to test the robot in more varied environments it would

perhaps then be more useful to describe the control system as an 'oblique dark/light boundary detector'. Nevertheless, within the environment in which it was evolved, it did perform the required task of detecting the triangle, and rejecting the square; within this context it is indeed a 'triangle detector'. This experiment illustrates that tasks such as these can be achieved with extremely minimal vision systems and very small networks.

The sequence of tasks performed illustrate the distinction between evolution and optimisation that was drawn earlier. Each individual task was an optimisation problem with a specific fitness function, but in practice successive tasks were not all predefined at the start. Such incremental evolution raises two separate questions of interest. The first is: given the scenario of an evolved design capable of task n, and a new more complex but related task $(n+1)$, is it more efficient to evolve by adaptation from what one has already achieved, or by starting from scratch? SAGA was developed for this incremental scenario, and the answer is in general that incremental adaptation must be faster — subject only to some qualifications on the relatedness between tasks n and $(n+1)$. The long term development of design in many fields, not just robotics, may be expected to have this nature.

There is a very different second question: if one is starting from scratch, and wants to achieve through evolution designs capable of task n, will evolution be speeded up if intermediate tasks are set, tasks 1, 2 ... $(n-1)$? One might expect the answer to be yes, but then the problem of choosing the appropriate intermediate tasks must be faced. It is a legitimate worry that this choice, by humans, of suitable stepping stones may in some circumstances be comparable in complexity to the very design problem which evolution is intended to automate. This is currently an open research topic.

8 Co-Evolution

In the incremental evolutionary experiments reported above, the task complexity was altered by the experimenters after success at each level had been achieved. Co-evolution is one possible way to take the humans out of this loop. This is where the fitness landscape[4] for a population is dependent (at least in part) on the distribution of genotypes in either the same population or other populations.

[4] A fitness landscape is a way of picturing all possible genotypes as points in a notional space, where the distance 'horizontally' between any two points relates to Hamming distance between corresponding genotypes, and differences in 'height' relate to differences in fitness.

One of the first demonstrations of artificial co-evolution was by Hillis [20], where sorting networks co-evolved against their test-sets. In the biology literature, some authors have suggested that the 'Red Queen effect' arising from co-evolutionary arms races has been a prime source of evolutionary innovations[37,33].

At Sussex, Cliff and Miller have been exploring the dynamics of co-evolution in the context of pursuit-evasion contests, using simulated robot-like autonomous virtual pursuers which chase autonomous virtual evaders around a 2-dimensional (2-D) space, generating pursuit and/or evasion strategies on the basis of simulated visual input [29,8]. The fitness landscape of one population is affected by the current strategies of any opponent populations; and the movements of one population over a fitness landscape can significantly alter the fitness landscapes of the other populations.

The simulation uses a conventional generational GA. There are two separate populations which compete and co-evolve against each other: one undergoes selection for pursuit behaviors, the other for evasion. Each population is spatially distributed with local mating and local replacement. That is, each individual in the population is assigned a spatial location on a 2-D grid (with toroidal wrap-around at the edges). When a new generation is bred, each individual is only allowed to breed with other individuals from nearby grid locations, and the offspring is also placed in a nearby grid location. In principle, this spatial structuring of the population should allow for the emergence and maintenance of somewhat distinct *subpopulations*, as was demonstrated in [20].

Reproductive success is determined by fitness, and fitness is evaluated for each individual by taking the mean score of a number of noisy trials with differing initial conditions (i.e. individual positions and orientations). In each trial a pursuer and an evader are given fixed amounts of energy which is expended in movement. They compete for a fixed length of time, terminating if there is a collision or both run out of energy. Significant noise affects the simulated sensors and effectors, and the activities of the artificial neural units. For efficiency, we use the same technique as Sims [34], where each individual's fitness is evaluated only in trials against the elite (i.e. highest-scoring) individual from the previous generation of the opponent population. At the end of each trial, the individual under evaluation is given a score. The score for evaders is the amount of simulated time before the trial ended; the score for pursuers is a temporal integral of the instantaneous rate of approach (which encourages the pursuer to approach the evader), plus a 'bonus' reward awarded if a collision occurs.

The differences in the scoring techniques mean that the contests are not zero-sum. Under this experimental regime, noisy continuous-time recurrent neural networks, similar to the DRNNs discussed earlier, can evolve to produce ef-

fective pursuit and evasion strategies. In [8] the difficulties associated with coevolutionary scenarios are discussed; in the absence of a fixed fitness landscape measurement of 'progress' is problematic, and a number of monitoring techniques are proposed.

9 Evolvable Hardware

The work discussed so far has generally used a genetically specified dynamical system as the control system for a simulated or real robot. But this dynamical system, for instance when conceptualised as a DRNN, has in practice been implemented on a computer. There is a related approach of evolving control systems directly onto hardware, which has been taken within our group by Thompson [35,36]. Using silicon chips such as FPGAs (Field Programmable Gate Arrays) it is possible for a designer (or in this case a GA) to reconfigure a real physical circuit embedded in silicon.

This work is *intrinsic* hardware evolution, in that for each genetically specified piece of hardware, the actual hardware is tested *in situ*; as contrasted with extrinsic hardware evolution, where simulations of the hardware are evaluated during evolution. The actual low-level physics of the hardware can be utilised, and the dynamics operate in real time at their proper timescales. Our notion of the nature of electronic systems is heavily biased by our design methodologies and the constraints applied to facilitate their abstractions, so evolvable hardware demands a radical rethink of what electronic circuits can be. Both the spatial structure (modularity) and the temporal structure (synchronisation and the role of phase in general) need to be considered.

With digital design by conventional methods, care must be taken to prevent switching transients (a feature absent from the designer's model) from affecting the system's overall behaviour. Usually, this means that the circuit is broken into modules, the internal transient dynamics of which are hidden from each other. Real physical electronic circuits are continuous-time dynamical systems. They can display a broad range of dynamical behaviour, of which discrete-time systems, digital systems and even computational systems are but subsets. These subsets are much more amenable to design techniques than dynamical electronic systems in general, because the restrictions to the dynamics that each subset brings support design abstractions. Intrinsic hardware evolution does not require abstract models, so there is no need to constrain artificially the dynamics of the reconfigurable hardware being used.

In particular, there no longer needs to be an *enforced* method of controlling the phase (temporal co-ordination) in reconfigurable hardware originally intended to implement digital designs. The phase of the system does not have to be

advanced in lock-step by a global clock, nor even the local phase-controlling mechanisms of asynchronous digital design methodologies imposed.

In one simulation experiment [35], Thompson demonstrated that a network of high-speed logic gates could be evolved to oscillate at a much slower timescale. At the start of the experiment, each of 100 logic nodes was assigned a real-valued propagation delay, selected uniformly randomly from the range 1.0 to 5.0 nanoseconds. The genotype specified the boolean function performed at each node, and the connectivity between nodes. Evolution was performed on a population of such genotypes, which were evaluated on the basis of the average period between logic transitions at one specified node: the closer to a square wave oscillation of 1kHz, the fitter. After 40 generations under this selection pressure, the output of the best individual was approximately $4\frac{1}{2}$ thousand times slower than the best of the random initial population, and was six orders of magnitude slower than the propagation delays of the nodes. Fitness was still rising at generation 40 when the simulation was terminated.

Thompson then used artificial evolution to design a real hardware circuit as an on-board controller for a two-wheeled autonomous mobile robot (diameter of 46cm, a height of 63cm) required to display simple wall-avoidance behaviour in an empty 2.9m$\times$4.2m rectangular arena. For this scenario, the d.c. motors driving the wheels were not allowed to run in reverse and the robot's only sensors were a pair of time-of-flight sonars rigidly mounted on the robot, pointing left and right. The sonars fire simultaneously five times a second; when a sonar fires, its output changes from logic **0** to logic **1** and stays there until the first echo is sensed at its transducer, at which time its output returns to **0**.

Conventional design methods would preprocess the sonar output pulses to give indications of the range to the nearest objects. From these, a central controller would be a hardware implementation of a finite-state machine (FSM), with the next-state and output functions designed so as to compute motor speeds for each wheel. From these speeds, an appropriate way of pulsing the motors would be then be calculated.

It would be possible to evolve the central controller FSM through intrinsic hardware evolution by implementing the next-state and output functions as look-up tables held in an off-the-shelf random access memory (RAM) chip. [5] The FSM would then be specified by the bits held in the RAM, which could be reconfigured under the control of each individual's genotype in turn. There would be no benefit in evolving this architecture as hardware, however, because the electronics is constrained to behave in accordance with the FSM design abstraction: all of the signals are synchronised to a global clock to give clean, deterministic state-transition behaviour as predicted by the model.

[5] This is the well known 'Direct Addressed ROM' implementation of an FSM [11].

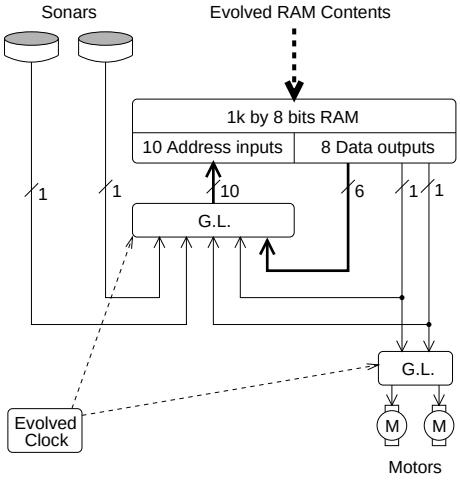

Fig. 11. *A characterisation of the hardware implementation (which is real, not simulated) of the evolvable DSM. 'G.L.' stands for a bank of genetic latches: it is under genetic control whether each signal is passed straight through asynchronously, or whether it is latched according to the global clock of evolved frequency.*

Consequently, the hardware would behave identically to a software implementation of the same FSM.

The alternative approach taken here is to relax the constraint of synchronisation of all signals, and place it under evolutionary control. The global clock frequency is under genetic control, as also is the choice of whether each signal is synchronised (latched) by the clock or asynchronous. This new architecture is termed a *Dynamic State Machine* (DSM). It is not a finite-state machine because a description of its state must include the temporal relationship between the asynchronous signals, which is a real-valued analogue quantity. In the conventionally designed control system there was a clear sensory/control/motor decomposition (timers/controller/pulse-width-modulators), communicating in atemporal binary representations which hid the real-time dynamics of the sensorimotor systems, and the environment linking them, from the central controller. Now, the evolving DSM is intimately coupled to the real-time dynamics of its sensorimotor environment, so that real-valued time can play an important role throughout the system. The evolving DSM can explore special-purpose tight sensorimotor couplings because the temporal signals can quickly flow through the system being influenced by, and in turn perturbing, the DSM on their way.

For convenience, evolution took place with the robot in a kind of 'virtual reality.' The real evolving hardware controlled the real motors, but the wheels were just spinning in the air. The wheels' angular velocities were measured, and used by a real time simulation of the motor characteristics and robot dynamics to calculate how the robot would move. The sonar echo signals were then artificially synthesised and supplied in real time to the hardware DSM. Realistic levels of noise were included in the sensor and motor models, both

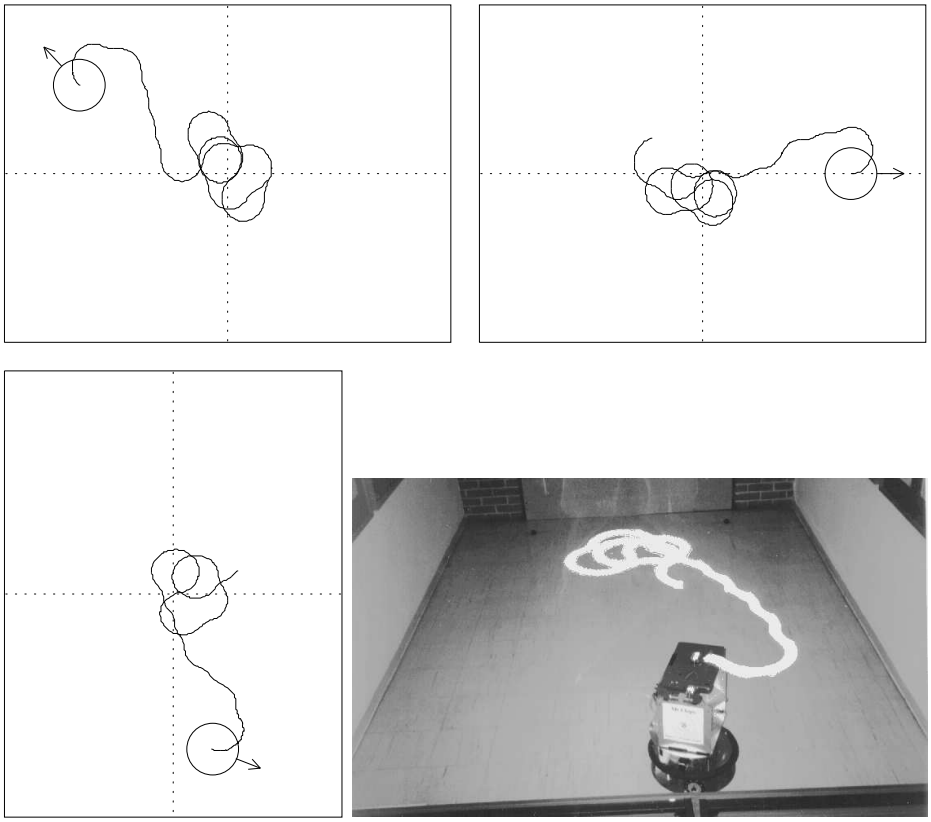

Fig. 12. *Wall avoidance in virtual reality and (bottom right) in the real world, after 35 generations. The top pictures are of 90 seconds of behaviour, the bottom ones of 60.*

of which were constructed by fitting curves to experimental measurements, including a probabilistic model for specular sonar reflections.

Fig. 12 shows the excellent performance which was attained after 35 generations, with a good transfer from the virtual environment to the real world. The robot is drawn to scale at its starting position, with its initial heading indicated by the arrow; thereafter only the trajectory of the centre of the robot is drawn. The bottom-right picture is a photograph of behaviour in the real world, taken by double-exposing a picture of the robot at its starting position, with a long exposure of a light fixed on top of the robot, moving in the darkened arena. If started repeatedly from the same position in the real world, the robot follows a different trajectory each time (occasionally *very* different), because of real-world noise. The robot displays the same qualitative range of behaviours in the virtual world, and the bottom pictures of Fig. 12 were deliberately chosen to illustrate this. One of the evolved wall-avoiding DSMs was analysed, and found to be going from sonar echo signals to motor pulses using only 32 bits of RAM and 3 flip-flops (excluding clock generation): highly efficient use of hardware resources, made possible by the absence of design constraints.

Further experiments are demonstrating that evolution may be an effective method of producing hardware tolerant to single-stuck-at (SSA) faults in the RAM's memory array. Evolutionary search with a population, tends to seek high local *areas* in the fitness landscape, rather than single high *points*. A local area is here defined as nearby points in genotype space: points reachable from each other by a single mutation, or a very small number. Experiments reported in [36] show, by emulating the effects of adverse SSA faults, that fault-tolerance does indeed develop under evolution (under circumstances where a mutation in genetic specification has similar consequences to a fault).

10 Evolution with the *Khepera*

The *Khepera* robot, from EPFL, Lausanne in Switzerland, is a popular robot for experimental work because of its size (6cm diameter) and convenience. It has been used in several laboratories for evolutionary experiments [14,30]. Evolution can be carried out directly on a real *Khepera*, or alternatively with a simulation with only the better solutions downloaded and tested on the actual robot. When using simulations it is an important question to decide just how realistic the model should be, and how noise should be handled.

With these questions in mind Jakobi [25] built a simulator, *Khepsim*, and conducted a number of experiments. The simulation is based on a spatially continuous, two dimensional model of the underlying real world physics and not on a look-up table approach as in [30]; parameters were set using empirical information from a *Khepera* robot. This affords greater generality with respect to new environments and unmodelled situations although at some computational expense. The simulation is updated once every 100 simulated milliseconds: the rate at which the inputs and outputs of the neural network control architectures are processed. This results in relatively coarse time slicing, some of the effects of which may be moderated by noise.

The empirical data used to set the simulation parameters for the *Khepera* robot's motors, PID controllers and general movement (in free space and in collisions) were collected out with the aid of the built-in position and speed sensors. By connecting the *Khepera* to a host computer using the supplied serial cable, accurate statistics on the robot's current speed and position could be gathered while the robot was moving. In this way a profile of the *Khepera*'s response to motor signals was calculated.

Ray tracing techniques are used to calculate values for the IR sensors, ten rays being used for each one. For the ambient light sensors ray tracing included the effects of single reflections. A light source (in reality a 60W desk lamp) was modelled as five point sources, and ambient light sensor values calculated as

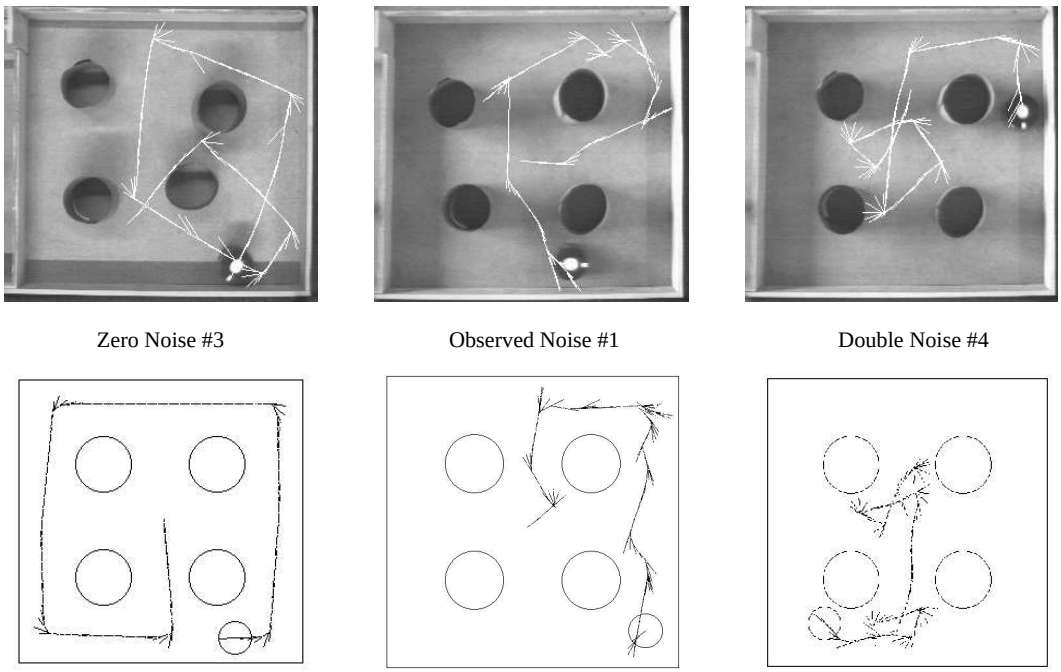

Fig. 13. *Obstacle avoidance: from simulation to reality. These six pictures display the situated and simulated behaviours of three different neural network controllers. The numbers #3, #1 and #4 refer to tables of behaviour descriptions given in [25], but not discussed further here.*

the sum of direct illumination and reflection.

Neural networks evolved in simulation evoked qualitatively similar behaviour on the real robot, the correspondence being a matter of degree rather than binary valued. The following experiments were designed to inspect two factors that affect this correspondence: the nature of the behaviour itself and the level of noise present in the simulation.

For each of two behaviours, obstacle avoiding and light seeking, three sets of five evolutionary runs were performed, one set for each of three different noise levels. These three noise levels were set at zero noise, observed noise and double observed noise. Observed noise (on sensors, motors etc.) refers to a roughly Gaussian distribution with standard deviation equal to that empirically derived from experiments. Double observed noise refers to the same distribution with double the standard deviation. As expected, it was found that in general, networks evolved in an environment that is less noisy than the real world will behave more noisily when downloaded onto the *Khepera* and, conversely, networks evolved in an environment that is noisier than the real world will behave less noisily when downloaded. Simulation to situation correspondence (as measured qualitatively and subjectively by the experimenter) seems to be maximised when the noise levels of the simulation have similar amplitudes to those observed in reality. The behaviours shown in Figs. 13 and 14 graphically illustrate this.

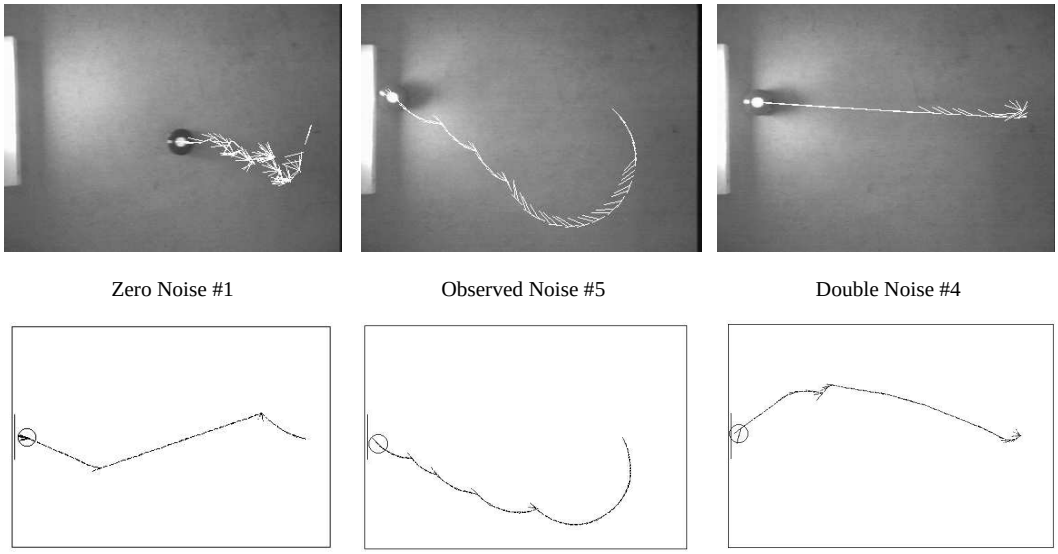

Fig. 14. *Light seeking: from simulation to reality. These six pictures display the situated and simulated behaviours of three different neural network controllers, one taken from each noise class.*

For both obstacle avoidance and light-seeking, the set of experiments running under observed noise obtained the highest average behaviour score. In a zero noise environment, brittle 'hit or miss' strategies (as on the lower left of Fig. 14) tend to evolve which either score incredibly well or incredibly badly on each fitness trial, depending on their initial random starting positions. Although noise, in general, blurs the fitness landscape, reducing the possibility of 'hit or miss' strategies evolving (since they are far more likely to 'miss' rather than 'hit'), too much randomness in the environment, as in the double noise case, ensures that the same genotypes may again achieve very different scores on two otherwise identical fitness evaluations. A balance between these two cases seems to be achieved at the observed noise level.

One conclusion from these experiments is that simulations can *in some circumstances* be good enough to be used for artificial evolution, with the resulting designs successfully downloaded onto a real *Khepera*. It seems likely that there are strong limitations on how far it is realistic to extend this approach.

A second conclusion is that the noise used in such simulations should be at a level similar to that observed experimentally. If there is a significant difference in noise levels, then whole different classes of behaviours become available which, while acquiring high fitness scores in simulation, necessarily fail to work in reality. This is true both for too little noise, and for too much noise.

11 Morphogenesis

In most of the experiments discussed above, the genotype contained a fairly direct description of the phenotype, the robot control system and the sensory morphology. Hence there would be a direct correspondence between the number of components in the phenotype (typically nodes or connections within a network) and the length of the genotype, even if there were regularities and repetitions in the phenotype.

Under many circumstances one would expect regularities and repetitions in the phenotype to be useful; for instance, reflecting bilateral symmetry in a robot. Yet for bilateral symmetry to emerge in a directly encoded phenotype, the specific characteristics of each half need to be specified separately on the genotype; and the specific values determined (through evolution) twice over. This basic inefficiency of a direct scheme makes it advisable to seek an indirect form of encoding from genotype to phenotype, such that the genotype specifies a morphogenetic process which constructs the phenotype; any regularities and repetitions to emerge naturally during development.

Morphogenetic schemes are an active research area at Sussex, by the present authors [24] and also by Gruau who has joined our group [17]. One scheme was used in the co-evolutionary experiments cited above, and a variety of different approaches are being investigated. Currently it seems likely that one major hurdle to be crossed if artificial evolution is to progress beyond relatively simple problems is that of morphogenesis.

12 Design of Fitness Functions

A further hurdle to be tackled is that of the appropriate design of fitness functions under which robot architectures are to be evaluated. Artificial evolution is not magic — useful phenotypes are only created at the expense of testing and rejecting large numbers of genotypes; since simulations are likely to be limited, these tests must in general be in the real world, in real time.

One way to minimise the number of evaluations is to organise the GA efficiently, in the context of noisy evaluations [1]. When incremental evolution is being used, with the SAGA methodology, then a sequence of tasks must be designed, 'shaping' the evolution towards desired end-goals. It is a non-trivial problem to design such a sequence, and to date it has been done largely by trial and error. There is of course the notorious tendency of GAs to find solutions which 'cheat', by complying with the letter of an evaluation function without meeting the intentions which lay behind it.

A principled approach to this problem has yet to be found; co-evolution is one possible approach.

13 Conclusion

Evolutionary Robotics is a research area in its infancy, but the test for all newborn AI philosophies is whether they can grow up into the real world, and whether they scale up with increasing complexity.

In the evolutionary experiments performed at Sussex we have moved on from unvalidated simulations and started to explore the possibilities with real world robots. The SAGA methodology has been designed specifically to adapt GAs to the incremental evolution of such complex systems, and arguments have been presented for DRNNs, or equivalent dynamical systems, as an appropriate class of robot control system for artificial evolution.

Up until now, all robot behaviours that have been evolved could have been designed and hand-coded with considerably less effort and time. Interesting lessons have been learnt, but in the long run the only engineering justification for pursuing an evolutionary approach would be demonstration of cases where evolution is more efficient than the opposition — human designers. Can the evolutionary tortoise keep plodding on after the hare of human design has been stopped by a complexity barrier?

The evolutionary approach advocated here is a bet on the tortoise, but there is much work yet to be done. Several major hurdles have been identified: morphogenesis, the design of fitness functions, and the role of simulations. The race is still in progress.

References

[1] A. N. Aizawa and B. W. Wah. Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation*, 2(2):97–122, 1994.

[2] W. Ross Ashby. *Design for a Brain.* Chapman, 1960.

[3] R. D. Beer and J. C. Gallagher. Evolving dynamic neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.

[4] R.D. Beer. A dynamical systems perspective on autonomous agents. Technical Report CES-92-11, Case Western Reserve University, Cleveland, Ohio, 1992.

[5] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Rob. Autom.*, 2:14–23, 1986.

[6] Rodney A. Brooks. Artificial life and real robots. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 3–10. MIT Press/Bradford Books, Cambridge, MA, 1992.

[7] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2(1):71–104, 1993.

[8] D. Cliff and G. F. Miller. Co-evolution of pursuit and evasion II: Simulation methods and results. Technical Report CSRP377, COGS, University of Sussex, 1995.

[9] D. T. Cliff, I. Harvey, and P. Husbands. Incremental evolution of neural network architectures for adaptive behaviour. In M. Verseylen, editor, *Proceedings of the First European Symposium on Artificial Neural Networks, ESANN'93*, pages 39–44. D facto Publishing, Brussels, 1993.

[10] M. Colombetti and M. Dorigo. Learning to control an autonomous robot by distributed genetic algorithms. In J.-A. Meyer, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, pages 305–312. MIT Press/Bradford Books, Cambridge, MA, 1993.

[11] D. J. Comer. *Digital Logic and State Machine Design.* Holt, Rinehart and Winston, 1984.

[12] K. A. De Jong. Are genetic algorithms function optimizers? In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 3–13. North-Holland, 1992.

[13] M. Eigen, J. McCaskill, and P. Schuster. Molecular quasi-species. *Journal of Physical Chemistry*, 92:6881–6891, 1988.

[14] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3, Proc. of 3rd Intl. Conf. on Simulation of Adaptive Behavior, SAB'94.* MIT Press/Bradford Books, 1994.

[15] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:1–64, 1993.

[16] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading MA, 1989.

[17] F.C. Gruau. Cellular encoding of genetic neural networks. Technical Report 92-21, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, May 1992.

[18] I. Harvey. Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In C. Langton, editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVI*, pages 299–326. Addison Wesley, 1993.

[19] Inman Harvey. Species adaptation genetic algorithms: The basis for a continuing SAGA. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354. MIT Press/Bradford Books, Cambridge, MA, 1992.

[20] W. D. Hillis. Co-evolving parasites improves simulated evolution as an optimization technique. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 313–384. Addison-Wesley, 1991.

[21] John Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA, 1975.

[22] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[23] P. Husbands, I. Harvey, and D. Cliff. Circle in the round: State space attractors for evolved sighted robots. *Journal of Robotics and Autonomous Systems. Special Issue on "The Biology and Technology of Intelligent Autonomous Agents".*, 15:83–106, 1995.

[24] P. Husbands, I. Harvey, D. Cliff, and G. Miller. The use of genetic algorithms for the development of sensorimotor control systems. In P. Gaussier and J.-D. Nicoud, editors, *From Perception to Action*, pages 110–121, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[25] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In F. Moran, A. Moreno, J. Merelo, and P. Chacon, editors, *Advances in Artificial Life: Proc. 3rd European Conference on Artificial Life*, pages 704–720. Springer-Verlag, Lecture Notes in Artificial Intelligence 929, 1995.

[26] Stuart Kauffman. Adaptation on rugged fitness landscapes. In Daniel L. Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison Wesley: Santa Fe Institute Studies in the Sciences of Complexity, 1989.

[27] J. R. Koza. *Genetic Programming.* MIT Press/Bradford Books, Cambridge MA, 1992.

[28] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[29] G. F. Miller and D. Cliff. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB94)*, pages 411–420. MIT Press, Bradford Books, Cambridge MA, 1994.

[30] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 190–197. MIT Press/Bradford Books, 1994.

[31] C. Reynolds. An evolved, vision-based model of obstacle avoidance behavior. In C. Langton, editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVI.* Addison Wesley., 1993.

[32] C.W. Reynolds. Evolution of corridor following in a noisy world. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB94)*, pages 402–410. MIT Press/Bradford Books, Cambridge MA, 1994.

[33] M. Ridley. *The Red Queen: Sex and the Evolution of Human Nature.* Viking, London, 1993.

[34] K. Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39. MIT Press, Bradford Books, 1994.

[35] A. Thompson. Evolving electronic robot controllers that exploit hardware resources. In F. Moran, A. Moreno, J. Merelo, and P. Chacon, editors, *Advances in Artificial Life: Proc. 3rd European Conference on Artificial Life*, pages 640–656. Springer-Verlag, Lecture Notes in Artificial Intelligence 929, 1995.

[36] A. Thompson, I. Harvey, and P. Husbands. Unconstrained evolution and hard consequences. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware.* Springer-Verlag Notes in Computer Science, 1996.

[37] L. Van Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, 1973.

[38] B. Yamauchi and R. Beer. Integrating reactive, sequential, and learning behavior using dynamical neural networks. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB94)*, pages 382–391. MIT Press/Bradford Books, Cambridge MA, 1994.