Noise and The Reality Gap: The Use of Simulation in Evolutionary Robotics

Nick Jakobi and Phil Husbands and Inman Harvey
School of Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QH, England
email: nickja or philh or inmanh@cogs.susx.ac.uk

Abstract

The pitfalls of naive robot simulations have been recognised for areas such as evolutionary robotics. It has been suggested that carefully validated simulations with a proper treatment of noise may overcome these problems. This paper reports the results of experiments intended to test some of these claims. A simulation was constructed of a two-wheeled Khepera robot with IR and ambient light sensors. This included detailed mathematical models of the robot-environment interaction dynamics with empirically determined parameters. Artificial evolution was used to develop recurrent dynamical network controllers for the simulated robot, for obstacle-avoidance and light-seeking tasks, using different levels of noise in the simulation. The evolved controllers were down-loaded onto the real robot and the correspondence between behaviour in simulation and in reality was tested. The level of correspondence varied according to how much noise was used in the simulation, with very good results achieved when realistic quantities were applied. It has been demonstrated that *it is* possible to develop successful robot controllers in simulation that generate almost identical behaviours in reality, at least for a particular class of robot-environment interaction dynamics.

Keywords: Evolutionary Robotics, Noise, High Fidelity Simulations, Artificial Evolution.

1 Introduction

A number of New-Wave roboticists have consistently warned of the dangers of working with over-simple unvalidated robot simulations [2, 1, 16]. Indeed, as Smithers has pointed out [16], the word simulation has been somewhat debased in the fields of AI, robotics, and animat research. Many so-called simulations are abstract computer models of imaginary robot-like entities, not carefully

constructed models of *real* robots. Whereas these abstract models can be very useful in exploring some aspects of the problem of control in autonomous agents, great care must be taken in using them to draw conclusions about behaviour in the real world. Unless their limitations are recognised, they can lead to both the study of problems that do not exist in the real world, and the ignoring of problems that do [1]. Behaviours developed in a simulation worthy of the name must correspond closely to those achieved when the control system is down-loaded onto the real robot.

One area of New-Wave robotics where these issues may be particularly pertinent is evolutionary robotics [8]. This is the development of control systems (and potentially morphologies) for autonomous robots through the use of artificial evolution. Populations of robots evolve over many generations in an open-ended way under the influence of behaviour-based selection pressures. Two of the earliest papers on this topic both stressed the likelyhood of having to work largely in simulation to overcome the time consuming nature of doing all the evaluations in the real world [3, 8]. However, both discussed the potential problems with simulations and remarked on the great care that would have to be taken. In [3], Brooks was highly sceptical[1]:

> There is a real danger (in fact, a near certainty) that programs which work well on simulated robots will completely fail on real robots because of the differences in real world sensing and actuation – it is very hard to simulate the actual dynamics of the real world.

and later,

> ...[sensors]...simply do not return clean accurate readings. At best they deliver a fuzzy approximation to what they are apparently measuring, and often they return something completely different.

But since the aim of evolutionary robotics is to produce working real robots, if simulations are to be used, these problems must be faced. The question is how. In [8] (with further elaborations in [9]) it is argued that:

- The simulation should be based on large quantities of carefully collected empirical data, and should be regularly validated.
- Appropriately profiled noise should be taken into account at all levels.
- The use of networks of adaptive noise tolerant units as the key elements of the control systems will help to 'soak up' discrepancies between the simulation and the real world.

[1] It is likely that these comments were influenced by experiences with devices rather different to the robot used in the experiments described later. This issue is returned to in the Conclusions.

- Noise added *in addition to* the empirically determined stochastic properties of the robot may help to cope with the inevitable deficiencies of the simulation by blurring them. A control system robust enough to cope with such an envelope-of-noise may handle the transfer from simulation to reality better than one that cannot deal with uncertainty over and above that inherent in the underlying simulation model.

This paper reports the results of experiments that were intended to explore the validity of some of these assertions and claims. Network-based control systems for generating simple behaviours were evolved in simulations of different levels of fidelity and then down-loaded onto the real robot. Comparisons were made between behaviours in the simulations and in the real world.

In [8] it was argued that as the robot's sensory coupling with its environment becomes more complex, simulations would become extremely difficult to construct and would be slower than real time unless highly specialised hardware were available. This problem resulted in the development of the Sussex gantry-robot which allows evolution in the real world [7]. This issue is revisited in the present paper in the light of the experiments outlined above.

The next section discusses related work. Following that is a description of the robot simulation and then an outline of the experimental setup. After detailing the evolutionary techniques used, experimental results are presented and discussed. Finally conclusions are drawn.

2 Related Work

Recently there have been a number of reports on experiences with transferring control systems from simulation to reality. These have met with varying degrees of success. Mondada and Verschure [14] describe the development, through the use of a learning algorithm, of a network-based control system both in simulation and in reality. They used the Khepera robot, the same device involved in the study described later in this paper (see Section 3). Qualitatively similar behaviours were observed in simulation and in reality. However, behaviour on the real robot was significantly less robust and a far greater number of learning steps were needed to achieve reasonable results. This appears to be partly due to the fact that noise was not modelled in the simulation. Miglino, Nafasi and Taylor [13] evolved recurrent network controllers for a very crude computer model of a simple Lego robot. Not surprisingly the evolved controllers generated significantly different behaviours in the real robot. Nolfi, Miglino and Parisi evolved network-based controllers for a simulation of the Khepera robot (described in [15]). Behaviours developed in simulation did not transfer at all well to the robot, but if the GA run was continued for a few generations in the real world (using techniques similar to those described in [5]) successful robust

controllers were obtained. This was probably due to the fact that the simulation was based on empirically sampled sensor readings. Sampling appears to have been too coarse, and possibly not enough readings were taken at each point to accurately determine the statistical properties of the sensors. We feel a better approach is to use large amounts of empirical data to set parameters and mappings in a continuous mathematical model of the robot-environment interactions. This technique seems to be vindicated by the results presented later in this paper. In [18] Yamauchi and Beer describe an experiment in which dynamical neural networks were evolved, in a manufacturer supplied simulation of a Nomad 200 robot, to solve a landmark recognition task using sonar. When the evolved controllers were used on the real robots, behaviours were very similar, although not quite as successful, as in the simulation. The simulator was probably very accurate and the highly dynamical networks used are likely to be good at 'soaking up' discrepancies between simulation and reality. Thompson had significant success (although there were some discrepancies) in transferring evolved hardware controllers developed in a semi-simulation (the robot's sonar-environment interaction was simulated, the real hardware and actuators were used) [17]. This was despite the fact that the robot involved is much more cumbersome and less 'clean' than devices such as Khepera, and the sonar simulation was rather crude (although noise was added to the underlying model).

3 The Robot Simulation

3.1 Khepera

The robot used in this project was the Khepera robot developed at E.P.F.L. in Lausanne, Switzerland. It has been specifically designed as a research tool allowing users to run their own programs and control algorithms on the powerful 36 MHz Motorola 68331 chip carried on board (see [11]). The robot is cylindrical with a diameter of 5.8 cm and a height of 3.0 cm. It has eight active IR proximity sensors mounted six on the front and two on the back. The receiver part of these sensors may also be used in a different mode to passively measure surrounding ambient light. The wheels are driven by extremely accurate stepper motors under P.I.D. control. Each motor incorporates a position counter and a speed of rotation sensor. It should be noted that it is probably far easier to build an accurate simulation of this sort of robot than it would be for many others.

3.2 The Simulation

The simulation of the Khepera was built using empirical information obtained from various experiments. The simulation is based on a spatially continuous,

two dimensional model of the underlying real world physics and not on a look-up tabl approach as in [15]. 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.

3.2.1 Modelling Methodology

The decision on which level of detail to pitch the simulation was arrived at intuitively. The idea was to build into the simulation all the important features of the Khepera's interaction with its environment without going into so much detail that computational requirements became excessive. After initial experimentation, it became clear which features were important to model (as fields in a state vector) and which could be left out. For example the distance from an IR sensor to the nearest object is an important feature model whereas the height of objects is not.

An idealised mathematical model was constructed by applying elementary physics and basic control theory to the interactions between these environmental variables. More specifically, generalised equations (with unassigned constants) were derived capable of producing values proportional to the ambient light intensities, the reflected infra red intensities and the wheel speeds.

After developing these general equations, several sets of experiments were performed to find actual sensor values and noise levels for specific settings of environmental variables. Using curve-fitting techniques it was then possible to produce mappings from the predictions of light intensities, wheel speeds and so on produced by the model to the actual sensor values observed during experimentation. As part of the same process values were also attributed to all unassigned constants to produce the set of equations (given below) that are used by the simulation to calculate specific values for the IR sensors, ambient light sensors and wheel speeds.

3.2.2 Modelling the Khepera's Movement, Motors and PID controllers

All experiments performed on the Khepera's motors, PID controllers and general movement were carried out with the aid of the in built position and speed sensors. By connecting the Khepera to a host computer using the supplied serial cable, accurate statistics on the Khepera'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 and mapped onto the model given below.

When one of the motors on the Khepera is set to run at a certain target speed V its actual speed, U, will in fact oscillate around this figure due to axle noise, irregularities on the ground surface and so on. Khepera uses a PID control algorithm [12] that ensures that U never varies significantly from V. It also has the important consequence that each wheel, over time, travels approximately the correct distance.

The PID algorithm changes the motor torque T according to the equation:

$$T \propto K_p(V-U) + K_i \int V - U dt + K_d \frac{d(V-U)}{dt}$$

where K_p K_i and K_d are the proportional, integral and derivative constants.

In the simulation, the integral and derivative terms could only be approximated due to the relatively coarse time slicing involved. The proportional error, P, at time t is calculated as $P_t = (V-U)/0.1$ since the simulation is updated every 0.1 seconds. The integral term, I_t, is the sum of the proportional terms over the last five time steps. The derivative term, D_t, is proportional to the force applied to the wheels on the last time step. It is calculated by dividing the change to the wheel speeds on the last time step, δv_{t-1}, by an empirically found constant of 50000, equivalent to the robot's mass. Once these terms have been calculated then the change to the wheel speed at time t, δv_t is calculated as:

$$\delta v_t = K_p \times P_t + K_i \times I_t + K_d \times D_t$$

The simulated robot's movement was found empirically to best match that of the Khepera when K_p K_i and K_d were set at 3800, 800 and 100 respectively. These values are, in fact, the same as those used by the PID controller on the Khepera (see [11]).

Static friction (the force that has to be overcome to start an object moving) is modelled by noisily thresholding the integral term. The path taken by each wheel during a simulation update is modelled as an arc of a circle.

3.2.3 Modelling the Khepera's Infra Red Sensors

Ray tracing techniques are used to calculate values for the IR sensors (see [6]). Ten rays are used for each sensor arranged in an arc spanning 180°. If the distance from a particular sensor along a ray i to an object is d_i, then the sensor value is calculated as

$$I = \sum_{i=1}^{10} \cos \beta_i (a/d_i^2 + b)$$

where β_i is the angle at which the ray i leaves the sensor and a and b were set empirically (see above) to 3515.0 and -91.4 respectively.

3.3 The Ambient Light Sensors

In reality there are many factors that have a measurable effect on the ambient light sensor values. The model used in the simulation is, therefore, correspondingly complicated. Ray tracing to a depth of two rays (the IR sensors are calculated from a depth of one) the intensity of the ambient light at a sensor is calculated as a sum of direct illumination and reflection. Experiments using the ambient light sensors were carried out in an environment with one major light source (see Section 4). The light source (in reality a 60W desk lamp) is modelled by five point sources. For each point at which one of the rays leaving the sensor hits the wall, five lamp-rays are calculated between this point and the five point sources approximating the lamp. The number of unobstructed lamp-rays is used to calculate the brightness of the reflection. Similarly direct illumination of a sensor is calculated from the number of unobstructed lamp-rays between that sensor and the five point sources.

Because the light source is brighter in the middle than at the edges, lamp-rays originating from the centre play a greater role in determining sensor illumination than those at the edges. The direct illumination D of a sensor is calculated as:

$$D = (-4.15 + \sum_{i=1}^{5} L_i \times k_i)/d^2$$

where d is the average of the distances from the sensor to each point source. L_i is 0 if the lamp-ray from the sensor to point source number i is occluded and 1 otherwise. k_i are the empirically derived weights for each point source: 4.19 for the point source on each edge, 4.24 for the pair inside each of these and 8.24 for the point source at the centre.

If D_i is the direct illumination of ambient sensor i and D_{ij} is the direct illumination of the point at which ray j of length d_{ij} from sensor i hits an object, then the total illumination A_i of sensor i is calculated as:

$$A_i = D_i + \sum_{j=1}^{10} (D_{ij} \times \cos^2 \beta_{ij} \times 1.5)/d_{ij}^2$$

where β_{ij} is the angle at which ray j leaves sensor i.

Finally the simulated sensor value V is calculated from the total illumination A according to the empirically determined mapping:

$$V = 55.0 + 1/\sqrt{A}$$

4 The Experimental Setup

In order to test the assertions outlined in the introduction, two sets of evolutionary runs were carried out, each involving the evolution of a different behaviour.

In the first set of experiments, obstacle avoiding behaviours were sought. The task here was to move around the environment covering as much ground as possible without crashing into objects. The environment, shown in figure 1, consisted of a square arena with sides of length 50cm constructed from yellow painted wood and four grey cardboard cylinders with a radius of 4cm.

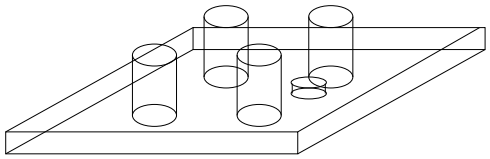

Figure 1: The environment used in obstacle avoiding experiments

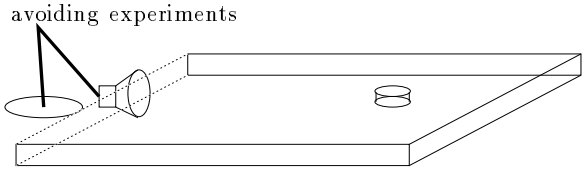

Figure 2: The environment used in light-seeking experiments

In the second set of experiments an ordinary sixty watt desk lamp was placed at one end of a 110cm by 70cm arena again made from yellow painted wood (Figure 2). A light seeking behaviour was sought which would enable the robot to move towards the lamp when started from random orientations and positions at the other end of the arena.

Before describing the experiments in detail, the evolutionary techniques will be explained.

5 Evolutionary Machinery

5.1 The Genetic Algorithm

In order to try and avoid some of the problems of premature convergence associated with more traditional genetic algorithms, a distributed GA was used [4]. The population was distributed over a two dimensional grid, and local selection was employed. Each member of the population breeds asynchronously with a mate chosen from its eight immediate neighbours on the grid. The mate is selected probabilistically using a linear fitness-rank-based distribution. The offspring replaces a member of the neighbourhood (this could potentially be either of its two parents) according to a linear inverse fitness-rank-based probability distribution. The genetic operators employed were mutation and crossover at

rates of 0.05 (mutations per piece of information stored on the genotype) and 0.8 respectively. Neurons and links were also added and/or deleted from the offspring genotype (see below) on each breeding with probabilities of 0.05 and 0.1 respectively.

5.2 The Encoding Scheme

The encoding scheme is the way in which phenotypes (in this case neural nets) are encoded by genotypes (the structure on which the genetic algorithm operates). The most commonly used encoding schemes involve direct one to one mapping between genotype and phenotype. The genotype consists of a series of fields expressed in bits, real numbers or characters. Separate fields specify the characteristics of each neuron and the connections associated with it. Networks encoded using a direct scheme can suffer gross distortions when their genotypes are allowed to grow or shrink.

The encoding scheme used in this research was designed to resolve some of these problems. The size of the phenotype is under genetic control but the addition or deletion of neurons and connections has minimal effect on the structure of the original network. The main difference between the encoding scheme used here and more normal direct encoding methods is that while phenotype size is under evolutionary control genotype size stays fixed. Instead of using a series of fields that specify neuronal characteristics, each genotype is made up of a series of 'slots', each one of which *may or may not* define a particular neuron on the phenotype and the links associated with that neuron. Connections address neurons by the absolute address of the slot they are associated with. Provided the genotype does not run out of spare slots to store new neurons in, any addition or deletion of neurons has minimal consequences on the rest of the network. In the runs described below there were 30 slots per genotype, of which typically 10–12 were used.Full details of the scheme can be found in [10].

5.3 The Neural Networks

A form of arbitrarily recurrent dynamical network was used in this research. The activation function of each neuron is defined as a simple linear threshold unit with fixed slope and a genetically determined lower threshold. Connections had genetically set time delays and weights.

In general, the smaller the number of parameters there are that need to be set in order to define a particular neural network, the quicker most learning algorithms or search techniques will be in finding suitable values for solving a particular problem. For this reason network parameters were restricted to a small number of integer values. Connection weights and delays were restricted to the interval ± 4, where the unit of time delay is ten milliseconds. Activation thresholds were restricted to the interval ± 10 and neuron output values to the

interval ± 10. All inputs and outputs of the network were scaled to the interval ± 10 in order to maximise network response and were updated every hundred milliseconds.

6 Experimental Results

Once a good underlying simulation model had been developed, it was found that in general, a neural network evolved in simulation evoked qualitatively similar behaviour on the real robot. During the entire period of this research there was never a negative instance in the sense of no similarity at all. The correspondence between simulated and situated behaviour turns out to be 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.

Because of the stochastic nature of the evolutionary process thirty runs were performed in total in order to acquire some statistical support for the conclusions that may be drawn from them. After the runs were complete, the evolved behaviours were subjectively marked by the authors on their optimality (how close they came to the ideal strategy/behaviour) and the correspondence between behaviours in simulation and reality[2]. The results for both behaviours are displayed in Tables 1 and 2.

Figure 3 and 5 both contain pictures showing paths taken by the Khepera in the real world. These were made by applying image processing techniques to short films of the Khepera, moving around its environment, with a specially constructed black and white disk placed on its uppermost face. Each frame underwent convolution operations using D.O.G. [3] center-surround masks specifically designed to respond maximally to the white patches at the centre and front of the disk placed on the Khepera. The positions of the peaks in the resultant intensity arrays were then used to pinpoint precisely the position and orientation of the Khepera in each frame. After processing an entire sequence,

[2]It is possible that some objective scoring system could be devised based on statistics of agent-environment interactions, but because of the nature of the problem it is not clear what this system would look like.

[3]A three dimensional mask constructed from the rotation of the difference between two Gaussian curves, of appropriate widths, around the vertical axis

the lines, one per frame, joining the centre of the Khepera to its leading edge, were overlaid on the final frame to produce an image of the Khepera with a white 'tail' behind it. The pictures of paths taken by the simulated robot contain a 'tail' of the same form. These were constructed by failing to erase a line, plotted on each previous time step, also joining its centre to its leading edge.

6.1 Obstacle Avoidance

Nolfi et al. [15] were the first to try evolving behaviours in simulation for the Khepera robot. In their work, outlined in Section 2, they were attempting to evolve obstacle avoiding behaviours. As already discussed, they did not achieve close correspondence between simulation and reality.

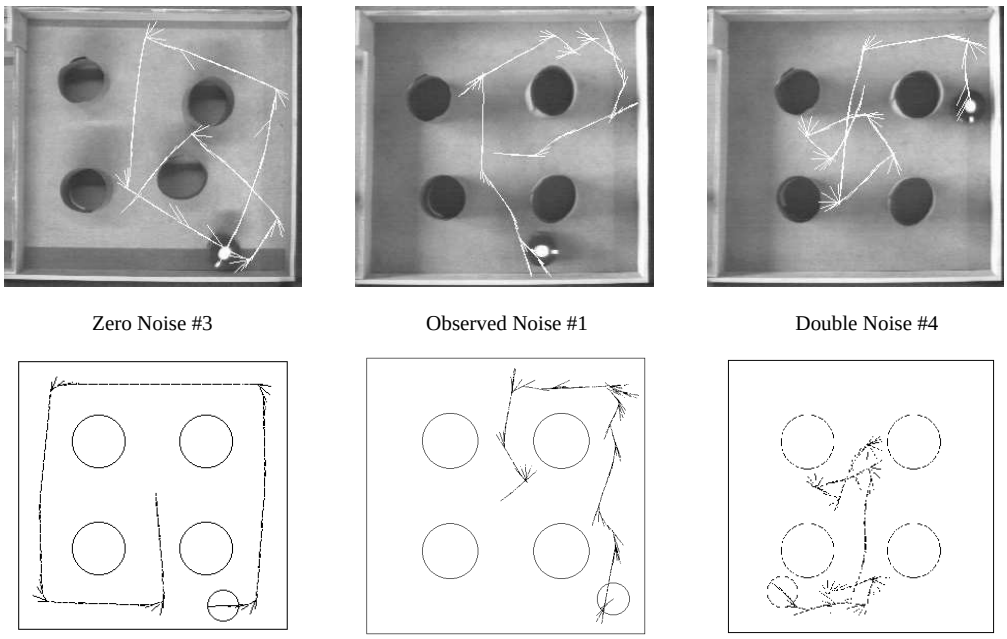

Figure 3: Obstacle avoidance: from simulation to reality. These six pictures display the situated and simulated behaviours of three different neural network controllers, one taken from each noise class. The #s refer to Table 1.

They identified three distinct components to such behaviours: moving forwards as fast as possible, moving in as straight a line as possible and keeping as far away from objects as possible. The fitness function they employed reflects this. During a trial three normalised sums are calculated: V, the sum of the wheel speeds at each time step, D, the signed sum of the absolute differences between the speeds of the two wheels at each time step and I the sum of the largest of the eight sensor values at each time step. These are combined to give a score F according to Equation 1:

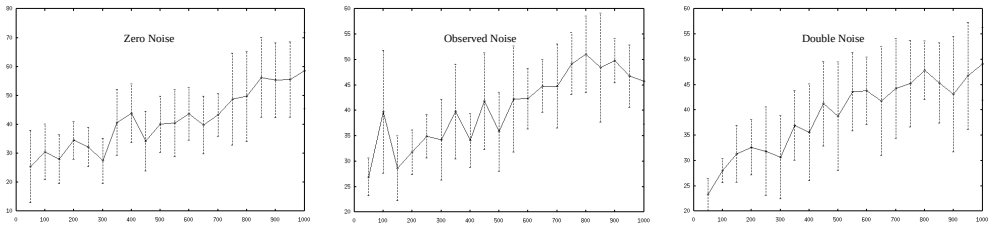

Figure 4: Obstacle avoidance. The average fitnesses, over each set of five trials, of the fittest individuals on each evaluation in simulation.

$$F = V(1 - \sqrt{D})(1 - I) \tag{1}$$

A slight variation of this fitness function was used in the experiments reported here. The $(1 - I)$ term in Equation 1 was found to be implicit if the environment is cluttered enough. This is because, in the environment used (Figure 1), the robot will have to learn to avoid objects if it is to go as fast and as straight as possible. Also the D term in Equation 1 was changed to the unsigned sum of the signed differences between the wheel speeds. This was thought to be a more effective way of forcing the robot to turn both ways to avoid objects while traveling in as straight a line as possible. Slight variations to left and right will tend to cancel out whereas variations biased either to the left or to the right will add. Thus the fitness equation actually used was:

$$F = V(1 - \sqrt{D}) \tag{2}$$

where D is now the absolute value of the sum of the *signed* differences between the wheel speeds.

Each evolutionary run consisted of one thousand fitness evaluations with the GA operating upon an initially random population of sixty four individuals. Each evaluation consisted of two trials started from a random position and random orientation near the centre of the environment shown in Figure 1. The fitness value was derived from the average of the scores resulting from the two trials. The trial time was twenty simulated seconds. Each run on a single user SPARC-10 took about 40 minutes.

Figure 4 shows the average fitnesses, over each set of five trials, of the fittest individuals on each evaluation. Table 1 shows the results of the subjective scoring process as described above. Figure 3 shows the correspondence between simulation and reality for some of the controllers listed in the table. See Section 7 for a discussion of these results.

6.2 Light Seeking

This behaviour is perhaps easier to evolve than obstacle avoidance as everything happens on a much slower scale. An obstacle avoider using short range IR

Zero Noise				
#	*type of behaviour*	*behaviour score*	*differences*	*correspondence score*
1	one way turner	5	a little noisier	8
2	wall follower	3	qualitatively similar	7
3	one way turner	4	noisier	5
4	one way turner	3	a little noisier	7
5	one way looper	5	noisier	4
		average 4		average 6.2
Normal Noise				
#	*type of behaviour*	*behaviour score*	*differences*	*correspondence score*
1	two way turner	8	a little noisier	8
2	one way looper	5	very similar	9
3	wall follower	4	a little noisier	7
4	wall follower	6	very similar	9
5	wall follower	7	a little noisier	7
		average 6		average 8
Double Noise				
#	*type of behaviour*	*behaviour score*	*differences*	*correspondence score*
1	one way turner	4	a little less responsive	8
2	two way turner	8	slightly less noisy	8
3	one way looper	4	less responsive	5
4	one way turner	5	less responsive	6
5	wall follower	3	less responsive	7
		average 4.8		average 6.8

Table 1: Obstacle avoidance. This table shows the scores subjectively given by our panel of judges to the evolved neural networks on the basis of the quality of their behaviours in simulation and the correspondence between their behaviour in simulation and their behaviour in reality. All scores are out of a maximum of 10.

sensors must turn the moment (or very soon afterwards) it senses an obstacle. A light seeking robot may employ a number of different strategies that will take it to the light eventually. It may react instantly to the light *or* take slow curving paths. The fitness landscape of such a task is therefore much smoother and light seeking behaviour emerges fairly early on in the evolutionary process.

The fitness function for this behaviour was simply calculated as the reciprocal of the sum of the squares of the distance from the light source at any particular time step. In other words if D_i is the distance to the light source at time step i, then the fitness F after n time steps is calculated as:

$$F = \frac{1}{\sum_{i=1}^{n} D_i^2} \tag{3}$$

Again, each evolutionary run consisted of one thousand fitness evaluations with an initially random population of sixty four individuals. A single evaluation consisted of two separate trials started from a fixed position and random

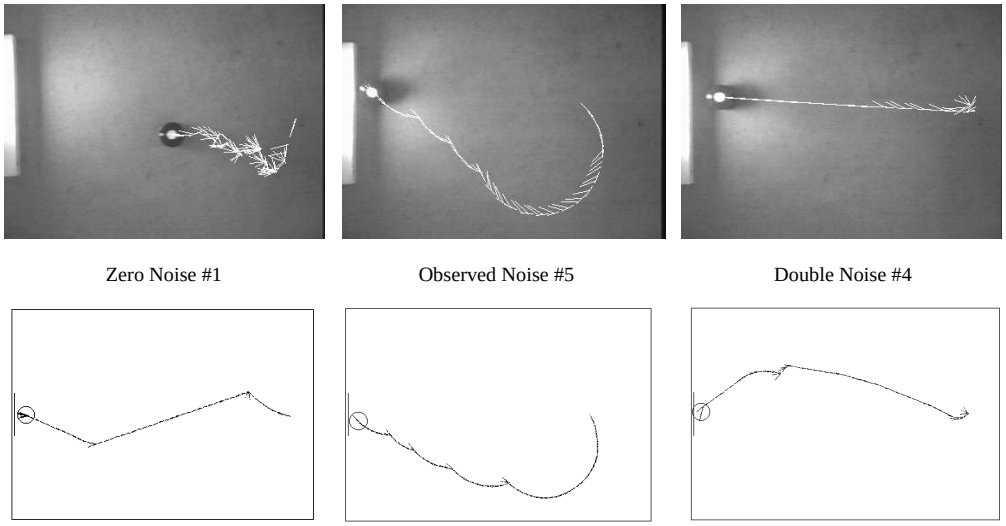

Figure 5: Light seeking: from simulation to reality. These six pictures display the situated and simulated behaviours of three different neural network controllers, one taken from each noise class. The #s refer to Table 2.

orientation near the opposite end of the environment (shown in Figure 2) to the light source. The fitness value was the average of the two scores. The trial time was twenty simulated seconds. Each run on a single user SPARC-10 took about an hour.

Figure 6 shows the average fitnesses, over each set of five trials, of the fittest individuals on each evaluation. Table 2 shows the results of the subjective scoring process as described above. Figure 5 shows the correspondence between simulation and reality for some of the controllers listed in the table. See Section 7 for a discussion of these results.

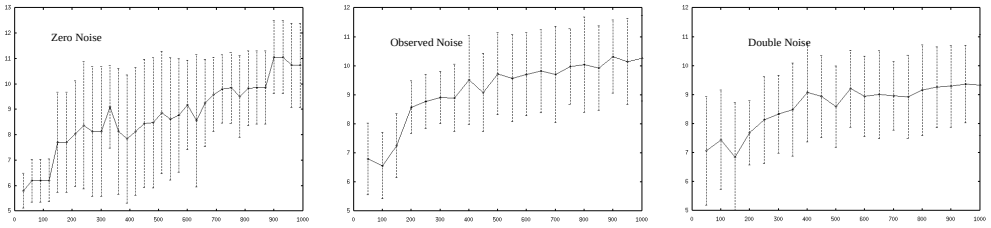

Figure 6: Light seeking. The average fitnesses, over each set of five trials, of the fittest individuals on each evaluation in simulation.

Zero Noise				
#	*type of behaviour*	*behaviour score*	*differences*	*correspondence score*
1	three point turner	6	noisier	4
2	straight looper	4	loops more	6
3	two way turner	6	similar	8
4	two way turner	8	noisier	4
5	two way turner	8	a little noisier	6
		average 6.4		average 5.6
Normal Noise				
#	*type of behaviour*	*behaviour score*	*differences*	*correspondence score*
1	straight looper	5	a little noisier	8
2	two way turner	8	a little noisier	6
3	two way turner	9	very similar	9
4	two way turner	7	a little noisier	6
5	bobber	5	observably identical	10
		average 6.8		average 7.8
Double Noise				
#	*type of behaviour*	*behaviour score*	*differences*	*correspondence score*
1	two way turner	7	much noisier	2
2	bobber	5	less noisy	6
3	straight looper	3	less noisy	7
4	two way turner	8	less noisy	6
5	bobber	6	less noisy	6
		average 5.8		average 5.4

Table 2: Light seeking behaviours. This table shows the scores subjectively given by our panel of judges to the evolved neural networks on the basis of the quality of their behaviours in simulation and the correspondence between their behaviour in simulation and their behaviour in reality. All scores are out of a maximum of 10.

7 Discussion

7.1 The General Picture

The overall picture to be gleaned from a study of Tables 1 and 2 is perhaps not very surprising. 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 seems to be maximised when the noise levels of the simulation have similar amplitudes to those observed in reality. The behaviours shown in Figures 3 and 5 graphically illustrate this.

Noise also plays a part in determining the quality (see Tables 1 and 2) of behaviours that evolve. This was a score subjectively attributed to behaviours on the grounds of how robust and optimal they proved to be *in simulation*

after repeated testing. Since a fitness evaluation only involves two trials, these scores do not necessarily correspond well to the fitness values ascribed by the GA. This correspondence could be improved by increasing the number of trials that make up a fitness evaluation, but at the expense of lengthening the time taken to perform an evolutionary run. The behaviour scores, then, show which noise levels give the best 'value for money' in terms of robustness against the number of trials that make up a fitness evaluation.

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 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. However, the fact that the particular level of noise that most favours the evolution of robust behaviours in simulation is also the level of noise that achieves the highest simulation to reality correspondence should probably not be regarded as anything other than a coincidence.

7.2 The Noise Level Has to Be Right

If the noise levels used in the simulation differ significantly from those present in reality, whole different classes of behaviours become available which, while acquiring high fitness scores in simulation, necessarily fail to work in reality. The zero noise behaviour shown in Figure 3 is one example. Evolution has taken advantage of the fact that, in a zero noise environment, the simulated robot will react identically in similar situations. By always turning through exactly ninety degrees, the simulated robot circumnavigates the room, ad infinitum, scoring well on the fitness test. In reality, the Khepera will never respond in exactly the same way twice. As can be seen from the picture, its behaviour is qualitatively similar, but the fact that its turns are never exactly ninety degrees means that it cannot settle down into a steady state of circumnavigation. Instead, because the neural network controlling it is 'blind' on one side, it ends up hitting objects and displaying far from optimal behaviour.

It is perhaps less obvious that evolution can also take advantage of *too much* noise in the simulation to produce networks that rely on the extra noise and are thus incapable of reproducing their behaviours in reality. The first double-noise light-seeker in Table 2 is one such example. Here a network that displayed near optimal (if noisy) light-seeking behaviour in simulation proved totally useless when downloaded onto the Khepera. It would jitter rapidly from side to side,

approaching the light only very slowly, and occasionally backing right away from it into the rear wall. The neural network in question uses two sensors, one on either side, to find the light. Each sensor controls the wheel on the opposite side of the robot in a Braitenberg fashion, but they are so arranged that, when the Khepera exactly faces the light-source, they are both *only just* illuminated. Since each input of the neural network is divided into bands, they are never both illuminated sufficiently (in a low noise world) to provide positive input to the neural network at the same time, and thus the robot jitters on the spot. However, in the double noise environment, there is enough noise present to push each sensor input's value up from the lowest band every now and then, thus providing positive inputs from both sensors at the same time, driving the robot forwards.

The experimental results provided some (inconclusive) support for the envelope-of-noise conjecture mentioned in the Introduction – that inevitable deficiencies in the simulation could be blurred by noise. In this case, sensor noise *profiles* were not modelled, rather a simple distribution of noise at the right level was used. Also differences between individual sensors were not modelled.

8 Conclusions

It has been shown that it is possible to artificially evolve successful network-based control systems in simulation that generate almost identical behaviours in reality. However, great care must be taken in building the simulation and appropriate levels of noise *must* be included.

The robot-environment interactions modelled here are relatively simple. Difficulties in simulating interference between the IR and ambient light modes of the Khepera's sensors, suggest that the approach taken here rather quickly become less feasible as the interaction dynamics become more complex. We still feel that real-world evolution techniques such as those described in [7, 5] are necessary, at least for the time being, to deal with these more complex couplings. However, it does appear that simulations are not quite the dead-end some had suggested. For simpler cases at least, it has been shown that they can be made accurate enough. Their attractive qualities of speed and ease of data collection can then be made use of.

Acknowledgements

Nick Jakobi is supported by a COGS postgraduate bursary. Thanks to colleagues in the Evolutionary and Adaptive Systems Group for useful discussions. Special thanks to David Young and Bob Ives for help in conducting the experiments described in this paper.

References

[1] R.A. Brooks. Intelligence without reason. In *Proceedings IJCAI-91*, pages 569–595. Morgan Kaufmann, 1991.

[2] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

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

[4] R. Collins and D. Jefferson. Selection in massively parallel genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 249–256. Morgan Kaufmann, 1991.

[5] 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.

[6] A.S. Glasner, editor. *An Introduction To Ray Tracing*. Academic Press, London, 1989.

[7] I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision. 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*, pages 392–401. MIT Press/Bradford Books, 1994.

[8] P. Husbands and I. Harvey. Evolution versus design: Controlling autonomous robots. In *Integrating Perception, Planning and Action, Proceedings of 3rd Annual Conference on Artificial Intelligence, Simulation and Planning*, pages 139–146. IEEE Press, 1992.

[9] P. Husbands, I. Harvey, and D. Cliff. An evolutionary approach to situated AI. In A. Sloman et al., editor, *Proc. 9th bi-annual conference of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISB 93)*, pages 61–70. IOS Press, 1993.

[10] N. Jakobi. Evolving sensorimotor control architectures in simulation for a real robot. Master's thesis, School of Cognitive and Computing Sciences, University of Sussex, 1994.

[11] K-Team. Khepera users manual. EPFL,Lausanne, June 1993.

[12] J. Kunt. *An Introduction to Control Theory*. Huddersfield and Wallstone, 1964.

[13] O. Miglino, C. Nafasi, and C. Taylor. Selection for wandering behavior in a small robot. Technical Report UCLA-CRSP-94-01, Dept. Cognitive Science, UCLA, 1994.

[14] F. Mondada and P. Verschure. Modeling system-environment interaction: The complementary roles of simulations and real world artifacts. In *Proceedings of Second European Conference on Artificial Life, ECAL93*, pages 808–817. Brussels, May 1993, 1993.

[15] 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.

[16] Tim Smithers. On why better robots make it harder. 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*, pages 54–72. MIT Press/Bradford Books, 1994.

[17] A. Thompson. Evolving electronic robot controllers that exploit hardware resources. In F. Moran, A. Moreno, J.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.

[18] 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, Proc. of 3rd Intl. Conf. on Simulation of Adaptive Behavior, SAB'94*, pages 382–391. MIT Press/Bradford Books, 1994.