Evolution In Materio : A Real-Time Robot Controller in Liquid Crystal

Simon Harding and Julian F. Miller
Department of Electronics
University of York
York, UK
slh,jfm@evolutioninmaterio.com

Abstract

Although intrinsic evolution has been shown to be capable of exploiting the physical properties of materials to solve problems, most researchers have chosen to limit themselves to using standard electronic components. However, it has been previously argued that because such components are human designed and intentionally have predictable responses, they may not be the most suitable medium to use when trying to get a naturally inspired search technique to solve a problem. Indeed allowing computer controlled evolution (CCE) to manipulate novel physical media can allow much greater scope for the discovery of unconventional solutions. Last year the authors demonstrated, for the first time, that CCE could manipulate liquid crystal to perform signal processing tasks (i.e frequency discrimination). In this paper we show that CCE can use liquid crystal to solve the much harder problem of controlling a robot in real time to navigate in an environment to reach an obstructed destination point.

1 Introduction

It has been argued that evolution in hardware would benefit from access to a richer physical environment [13], however much of the current research still focuses on conventional component based evolution. Evolving in materio may allow us to develop new systems that are based on exploiting the physical properties of a complex system. In [20] we saw that an evolutionary algorithm used some subtle physical properties of an FPGA to solve a problem. It is not fully understood what properties of the FPGA were used. This lack of knowledge of how the system works prevents humans from designing systems that are intended to exploit these subtle and complex physical characteristics. However it does not prevent exploitation through artificial evolution.

In [6] Harding and Miller showed that liquid crystal could be used as a medium for evolution. They were able to rapidly evolve simple transistor like behaviour and in [5] they demonstrated that it was relatively easy to evolve a liquid crystal to discriminate between pairs of dissimilar frequencies. The task was first considered by Adrian Thompson (using an FPGA) [21]. In this paper we present work demonstrating that it is possible to evolve a liquid crystal display to perform as a sophisticated real time controller for a wall avoiding robot.

2 The Field Programmable Matter Array

In [13] a device that the authors referred to as a Field Programmable Matter Array(FPMA) was described. The idea behind the FPMA is that applied voltages may induce physical changes within a substance, and that these changes may interact in unexpected ways that may be exploitable under evolution.

Different candidate materials were cited for possible use as the evolvable substrate in the FPMA. They all share several characteristics : the material should be configurable by an applied voltage/current, the material should affect an incident signal (e.g. optical and electronic) and should be able to be reset back to its original state. Examples of these include electroactive polymers, voltage controlled colloids, bacterial consortia, liquid crystal, nanoparticle suspensions. In previous work we have demonstrated that liquid crystal is indeed a suitable material to form the basis of the FPMA.

2.1 Liquid Crystal

Liquid crystal (LC) is commonly defined as a substance that can exist in a mesomorphic state [3][9]. Mesomorphic states have a degree of molecular order that lies between that of a solid crystal (long-range positional and orientational) and a liquid, gas or amorphous solid (no long-range order). In LC there is long-range orientational order but no long-range positional order.

It is possible to control the orientation of liquid crystal molecules using electric fields. Normally the molecules in a liquid crystal align themselves along a common director, however this orientation is essentially random. By applying an electric field it is possible to change the angle of this director and force the molecules to rotate into a desired orientation. Changing the orientation of the liquid crystal changes its behaviour. The most well known of these effects is the change in its optical properties. Rotating the molecules changes the refractive index of the liquid, it is this effect that is used in liquid crystal displays(LCDs). By applying fields to specific parts of the LCD allows the field affected region to change its optical properties from transparency to opacity - this facilitates the formation of an image.

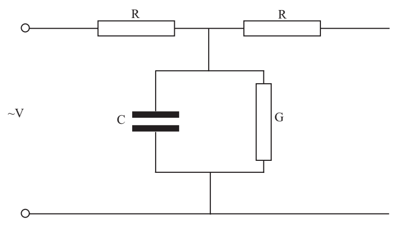

Figure 1. Equivalent circuit for LC

Changing the orientation of the molecules also alters the electrical properties of the liquid crystal. Figure 1 shows the equivalent electrical circuit for liquid crystal between two electrodes when an AC voltage is applied. The distributed resistors, R, are produced by the electrodes. The capacitance, C, and the conductance, G, are produced by the liquid crystal layer[15]. Changing the orientation alters these constants, and in this work we aim to exploit these electrical properties (and possibly other properties) by using applied fields to alter the molecular configuration.

3 An Evolvable Motherboard with a FPMA

3.1 Previous Evolvable Motherboards

An evolvable motherboard(EM) was a term first coined by Layzell, [10] it is a circuit that can be used to investigate intrinsic evolution. The EM is a reconfigurable circuit that rewires a circuit under computer control. Previous EMs have been used to evolve circuits containing electronic components[10][1] - however they can also be used to evolve in materio by replacing the standard components with a candidate material.

An EM is connected to an Evolvatron (a term coined by Adrian Thompson). This is essentially a PC that is used to control the evolutionary processes. The Evolvatron also has digital and analog I/O, and can be used to provide test signals and record the response of the material under evolution. In 2 we give a depiction of an evolvatron manipulating a piece of liquid crystal.

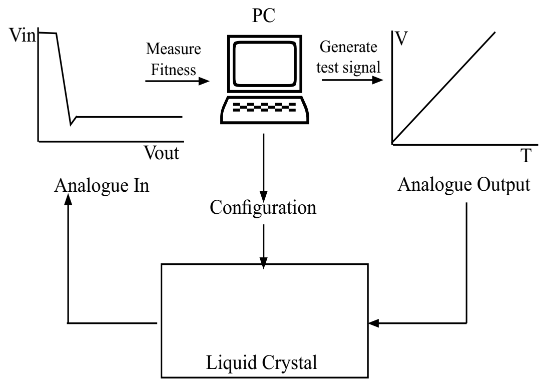

Figure 2. Equipment configuration

3.2 The Liquid Crystal EM

In the experiments presented here, a standard liquid crystal display with twisted nematic liquid crystals was used as the medium for evolution. The display is a monochromatic matrix LCD with a resolution for 180 by 120 pixels. The displays are made up of several layers, as shown in figure 3. The liquid crystal layer(c) is sandwiched between the two sheets which are coated in electric connections(b,d). These layers are then positioned between two polarising filters, one in a horizontal orientation(a) the other vertically(e).

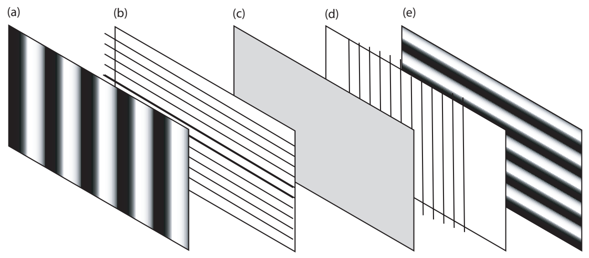

Figure 3. Layers in a LCD

It is assumed that the electrodes are indium tin oxide. Typically such a display would be connected to a driver circuit. The driver circuit has a configuration bus on which commands can be given for writing text or individually addressing pixels so that images can be displayed. The driver circuit has a large number of outputs that connect to the wires on the matrix display. When displaying an image appropriate connections are held high, at a fixed voltage - the outputs are typically either fully on or fully off.

Such a driver circuit is unsuitable for our task of intrinsic evolution. We need to be able to apply both control signals and incident signals to the display, and also record the response from a particular connector. Evolution should be

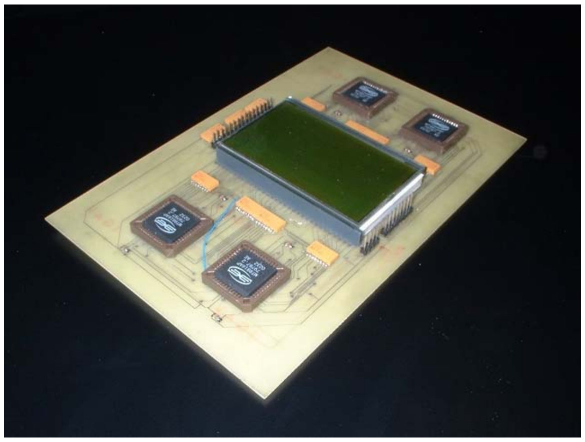

Figure 4. The LCEM

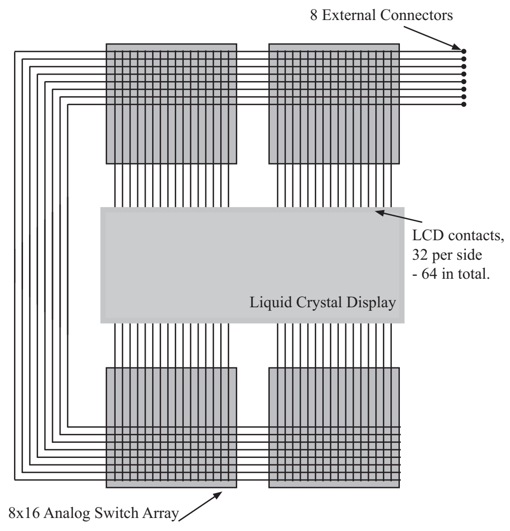

Figure 5. Schematic of LCEM

allowed to determine the correct voltages to apply, and may choose to apply several different values. The evolutionary algorithm should also be able to select suitable positions to apply and record values. A standard driver circuit would be unable to do this satisfactorily.

Hence a variation of the evolvable motherboard was developed in order to meet these requirements.

The Liquid Crystal Evolvable Motherboard (LCEM) is circuit that uses four cross-switch matrix devices to dynamically configure circuits connecting to the liquid crystal. The switches are used to wire the 64 connections on the LCD to one of 8 external connections. The external connections are: input voltages, grounding, signals and connections to measurement devices. Each of the external connectors can be wired to any of the connections to the LCD.

The external connections of the LCEM are connected to the Evolvatron's analogue inputs and outputs. Connections can be assigned for the input signals, measurement, and for fixed voltages (plus a ground connection). The value of the fixed voltages is determined by a genetic algorithm[8], but is constant throughout each evaluation.

In these experiments the liquid crystal glass sandwich was removed from the display controller it was originally mounted on, and placed on the LCEM. The display has a large number of connections (in excess of 200), however because of PCB manufacturing constraints we are limited in the size of connection we can make, and hence the number of connections. The LCD is therefore roughly positioned over the pads on the PCB, with many of the PCB pads touching more than 1 of the connectors on the LCD. This means that we are applying configuration voltages to several areas of LC at the same time.

Unfortunately neither the internal structure nor the electrical characteristics of the LCD are known. This raises the possibility that a configuration may be applied that would damage the device. The wires inside the LCD are made of an extremely thin material that could easily be burnt out if too much current flows through them. To guard against this, each connection to the LCD is made through a 4.7Kohm resistor in order to provide protection against short circuits and to help limit the current in the LCD. The current supplied to the LCD is limited to 100mA. The software controlling the evolution is also responsible for avoiding configurations that may endanger the device (such as short circuits).

It is important to note that other than the control circuitry for the switch arrays there are no other active components on the motherboard - only analog switches, smoothing capacitors, resistors and the LCD are present.

3.3 Evolved Robot Controllers

There are many examples of evolved robot controllers, and in general they fall into one of three categories : Genetic Programming, Evolved neural network (either with discrete or continuous models) and Evolvable hardware.

A typical task for robot controller evolution is to produce wall avoiding (or wall following) behaviour and there are many examples of this in the literature [12, 11, 2, 16, 4, 17, 18, 19].

Generally evolution is performed in simulation. Solutions based on genetic programming or neural networks can run faster in simulation than those based on real robots, as they can ignore (to a degree) the physical properties of the robot and its hardware. However, in this instance we cannot simulate the controller and have to perform the control in hardware. In [19] Thompson demonstrates that it is not only possible to evolve a robot controller in hardware, but to evolve one that utilises the properties of

a physical system. This work was based on his previous research that demonstrated that evolution was capable of ”using the properties of the hardware that the designer could never know about”. In the experiment, a control system was evolved to move a robot, ”Mr Chips”, around an empty rectangular room without colliding with the walls. The control hardware for Mr Chips is simply two sonar distance sensors, two motor units and a RAM chip. This RAM chips implements a version of a finite state machine, described as a Dynamic State Machine (DSM). The contents of the RAM chip are defined by evolution, and provide a mapping between input and output states. The outputs of the RAM chip are connected directly to the motors, and the output addressing lines of the chip are connected directly to the sonar outputs. All of the input and output occurs asynchronously. Evolution was allowed to define the operating frequency of the two clocks that are required to transfer output from the DSM to the input of the DSM and to control the motor speeds.
Using the DSM in this way removes the sensory/control/motor functional decomposition. Having the inputs(sonar readings) and the outputs(motor pulses) connected means that the ”control system is intimately linked to the dynamics of the sensor/motor signals and the environment, with time now able to play an important role throughout the system”.
The robot was operated in a kind of virtual reality. In order to be able to exploit the physical properties of the robot the experiments had to be performed intrinsically, however the robot was kept stationery with sensor readings simulated using a computer. Realistic levels of noise were added to the sonar readings to increase the realism of the simulated results. The robots wheels were allowed to run in free air, with their speed measured and used to calculate the movement of a virtual robot.
Figure 6 shows the behaviour of an evolved solution. The bottom right hand image shows the result when the robot is allowed to move around its environment and use the the real sonar readings. Thompson comments ”given that the DSM receives the raw echo signals from the sonars and directly drives the motors (one of which happens to be more powerful than the other), with only two internal state variables, then this performance in surprisingly good.” There are many subtle interactions between components in the system, and these cannot easily be discovered or described. It does however appear that the operation of this control system is dependent on the physical properties of the hardware, and it for this reason that the following experiments with robot control systems in liquid crystal were performed. In liquid crystal there is no concept of a program, and it is unclear how a neural network like structure could be developed. Hence a control system that utilises the properties of the system is required.

The following work takes a similar approach to Thompson, however instead of using a RAM chip as the DSM, liquid crystal is used. This is the first time that such an approach has been taken.

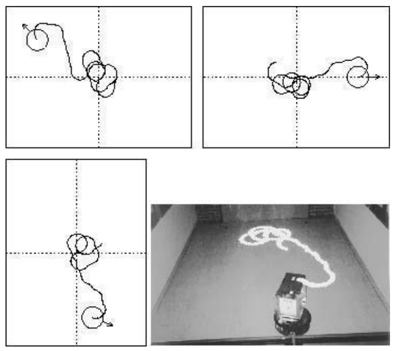

Figure 6. Wall avoidance in virtual reality and (bottom right) in the real world, after 35 generations.[19]

4 A Liquid Crystal Robot Controller

For convenience, a simulated robot was used for these experiments. As the simulation could be considered to occur in real time, a physical robot could be substituted as the embodiment of the liquid crystal. However, real robots would bring unnecessary complexity to this experiment. The simulated robot has two such sensors (mounted with 30 degrees of separation) and two wheels for mobility. The simulated sensor readings are converted into signals fed to the evolvable motherboard. Signals read from the evolvable motherboard are then used to control the behaviour of the simulated robot. The intention being that the signal processing, and majority of the robot control should be performed in the liquid crystal. Figure 7 shows how the liquid crystal is connected to act as a robot controller. Two sonar distances sensors and two motors can be considered to be ”directly” connected to the evolvable motherboard, and then routed to the liquid crystal. If implemented in real hardware, extra circuitry would be required as the output from the liquid crystal may not be suitable to drive the motors directly.

We defined each distance sensor to output a square wave with a frequency proportional to the distance in a straight line from the sensor to an obstacle. For near objects the output was 1Hz, for far objects the output frequency is 5000Hz. No artificial noise is added to the distance measured, however the mechanism by which the waves are generated by the computer will add noise and timing problems. There is also an expected 50ms delay between a

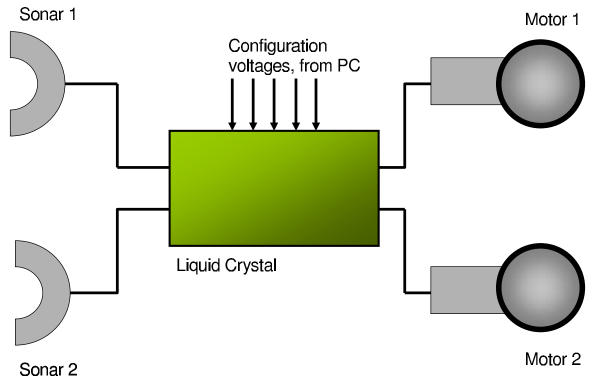

Figure 7. Liquid Crystal Robot Controller

distance reading and a change in frequency.
Two connections from the LCEM are used as inputs to the two motor controllers. The two motors are mounted either side of the simulated robot, and allow for the robot to be steered. If the voltage is high (i.e. above 0.3V) a motor is switched fully on, low the motor is set to a slow speed. If both inputs are high the robot drives forward, with both off the robot is stationery. If only one motor is enabled, the robot turns. The threshold voltage for enabling a motor was chosen arbitrarily. The robot has a small turning circle, and does not pivot on the switched off wheel.

5 The Genetic Algorithm

5.1 The Genotype

The genetic representation for each individual is made of two parts. The first part specifies the connectivity; the second part determines the configuration voltages applied to the the LCD.

Each of the 64 connectors on the LCD can be connected to one of the eight external connectors or left to float (see figure 5). Each of the connectors is represented by a number from 0 to 7 and no connection is represented by 8. Hence the genotype for connectivity is a string of 64 integers in the range 0 to 8.

The remainder of the genotype specifies the voltages applied to the pins on the external connector that are not used for signal injection / monitoring. One of the external connectors is always connected to ground. Two are reserved for the incident signals (distance readings) and two connections for motor control. The remaining three connectors have static voltages applied to them that are determined by evolution. All these connectors can be routed to various places in the liquid crystal display according to the connection scheme decided by evolution (as described in the previous paragraph). Each voltage is represented as a 16-bit integer, the 65536 possible values map to the voltage levels output from -10V to +10V. The second section of the genotype is therefore represented as a string of three 16bit integers.

To clarify this further, the evolutionary algorithm determines three possible voltages and where they may be applied to any of the 64 connectors on the LCD. The algorithm also determines to which of the connectors on the LCD the incident signals will be applied, the connector used to read the output signals from and which connectors should be grounded.

In contrast to previous work, where the liquid crystal had one input and one output these experiments require two inputs and two outputs. This reduces the number of configuration voltages to 3 (since we used the same LCEM as that used in our previous frequency discriminator experiments).

5.2 Constraints

To help prevent damage and misreading output signals, the genotype has to be limited to configurations that will not be harmful to its phenotypic expression, for example shorting connections together. To achieve this certain connections (for example where the output is measured) are limited to a certain number of appearances within the genotype.

By preventing the genotype from going outside these constraints it is hoped that no damaging configurations can be downloaded into the LCEM. We also only allow the sensor signals and motor outputs to be connected to a *single* place in the LCD (subject to the ambiguity caused by inaccurate physical mounting previously discussed.

Unconstrained, the number of possible configurations is $9^{64}X2^{48}$.

5.3 Genetic Operators

A mutation is defined as randomly taking an element in one part of the genotype and setting it to a randomly selected new value. Constraints are enforced to prevent illegal configurations.

We chose not to use genetic recombination as the constraints imposed on this representation would make it difficult to implement and would require many arbitrary decisions to be made concerning suitable repair techniques. For example, it is unclear what strategy should be used to fix a genotype where multiple connections on the LCD are linked to the analogue recorder and where only one connection is allowed. For this reason, the evolutionary algorithm used here has no crossover operator.

5.4 Parameters

In all the following experiments, a population of 40 individuals was used. The mutation rate was set to 5 mutations per individual. Elitism was used, with 5 individuals selected from the population going through to the next generation. Selection was performed using tournament selection based on a sample of 5 individuals.

Evolutionary runs were limited to 200 generations. With each individual taking approximately 60 seconds to evaluate.

6 The Fitness Function

Typically for obstacle avoiding robots, fitness values are computed based on factors such as time spent moving forward, total path length and Euclidean distance travelled. For example, Thompson uses the following calculation:

$$fitness = \frac{1}{T}\int_0^T \left(e^{-k_x c_x(t)^2} + e^{-k_y c_y(t)^2} - s(t)\right)$$

$$where\ s(t) =^{1whenstationary}_{0otherwise}$$

Where the distance of the robot from the centre of the room in the x and y directions at time t was c_x and c_y, then after an evaluation for T seconds. However, during initial experiments it was appeared that this method for calculating fitness had many drawbacks including local minimas which resulted in poor evolutionary characteristics (e.g. mean fitness did not increase smoothly). In environments with obstacles this style of fitness function fails to capture the difficulty of getting to hard-to-reach locations in the map. To address this a fitness map of the environment was calculated, where each area in the map had an absolute measure of the difficulty in reaching that point. The fitness for the robot was calculated as the highest fitness measure seen during the robot's movement around the map.
The fitness values for the map were calculated by modelling chemical diffusion within the environment. Although it may be difficult to see the variation in colour, figures 9 and 11 show the fitness maps for the two environments used. The darker colours show the areas of greatest difficulty to reach from the robots starting position in the top left corner. Robots that can successfully navigate around the obstacles, and explore large amounts of the map will pass through areas marked as having high fitness. Solutions where the robot does not move far or travels in a circle will obtain low fitness. In these experiments the robots were allowed to travel until they collided with a wall, or a timeout situation occurs. Robots were initially given 30 seconds before timing out, with extra time awarded if they travelled into new parts of the map.
In [7], a robot controller is evolved using Cartesian Genetic Programming, with the fitness function used here. It was found that the smoother fitness landscape allowed for rapid evolution.

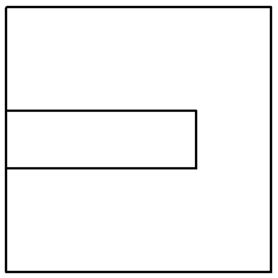

Figure 8. Map 1

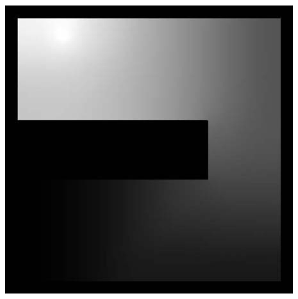

Figure 9. Fitness values for map 1

7 Results

The results for solving the first map are shown in table **??**. Solutions that have a fitness of over 6700 represent robots that have navigated to leave the top section of the map. Solutions below this score fail to fully explore the map - however they may cover large areas of the top half of the map but never escape through the gap. The five solutions that do escape, continue and fully explore the map. This gives a success rate of 36%. The average number of generations to find a good solution is 62, with the fastest solution found within 22 generations.
In figure 12 graphs are shown that illustrate the evolution of each run. It can be seen that the fitness often increases rapidly - with the overall fitness increasing in several (on average 5) of these steps. These jumps in fitness suggest that only small changes are needed in the genotype to cause large changes in the phenotype. As we are able to evolve solutions, there must be a strong relationship between the genotype/phenotype mapping - however it must be a highly nonlinear mapping. Such a genotype-phenotype mapping would make training the liquid crystal by hand, or with another machine learning technique, very difficult as it would be hard to predict the outcome of a parameter change - especially when so many parameters are available.

Figures 13 to 23 shows sections of the "fossil record" of the evolution of one controller (result N). We can see that after learning not to drive in circles, the robot learns to move

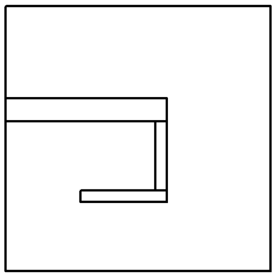

Figure 10. Map 2

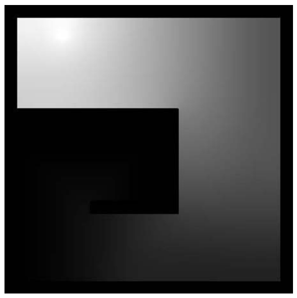

Figure 11. Fitness values for map 2

forward, and then learns to turn when it approaches a wall. After it learns to start following the wall it quickly searches the entire map, and gets the highest fitness.

7.1 Generalisation

The robot controllers were also tested to see if the behaviour generalised. To do this, the evolved robot controller were also tested in a second, unseen environment. This map is shown in figure 10. It was found that 35% of evolved solutions correctly navigated the second map.

7.2 Symmetry in wall following behaviour

An interesting observation we made after examining the robot results was the symmetry of wall following behaviour. The behaviour of the robot on both sides of the obstacle is similar. Figures 24,25 and 26 show some examples of this. This behaviour appears to demonstrates that a consistent and generalised control system has been developed. If the robot had "learnt" the map, it would be unlikely for it to follow the same behaviour when presented with the similar but "'inverted"' environmental features.

Experiment	Maximum Fitness	Generation
A	6696	68
B	6695	197
C	5431	74
D	6700	47
E	5797	14
F	6690	80
G	6667	42
H	5050	20
I	10000	46
J	10000	73
K	9056	74
L	10000	97
M	4542	68
N	9958	22

Table 1. Summary of experiment results, showing for each experiment the maximum fitness and the generation at which that fitness was achieved.

8 Comparison to Cartesian Genetic Programming

In [7] maps using the same robot simulator are solved using a robot controlled by an evolved program (i.e. no liquid crystal). In this work Cartesian Genetic Programming(CGP) was used to evolve a graph based control scheme for the robot. A complete description of CGP can be found in [14]. Using CGP the success rate was 100% (based on 100 runs) for this map - with the average time to find a solution of 10 generations.
There are several key differences in the implementation of the CGP robot controller compared with the work presented here. The first difference is the lack of real time control. The robot controlled by liquid crystal could be considered to be controlled in real time, where the robot controlled by CGP moved in individual time steps, with the CGP program was run between time steps. We would expect a real time controller to be a harder task than a discrete time controller. When evolving using CGP, the robot has access to many different mathematical operators - liquid crystal, we assume, does not have this well defined functionality built in. Evolution, if it wishes to use such operators, would first have to evolve them in the liquid crystal. The third difference is the manner in which sonar readings are passed to the controller. The liquid crystal receives the information as a square wave signal whose frequency is dependent on distance. The CGP program is given the actual distance as an integer. This removes even more time dependence from the problem. We would assume the liquid crystal would

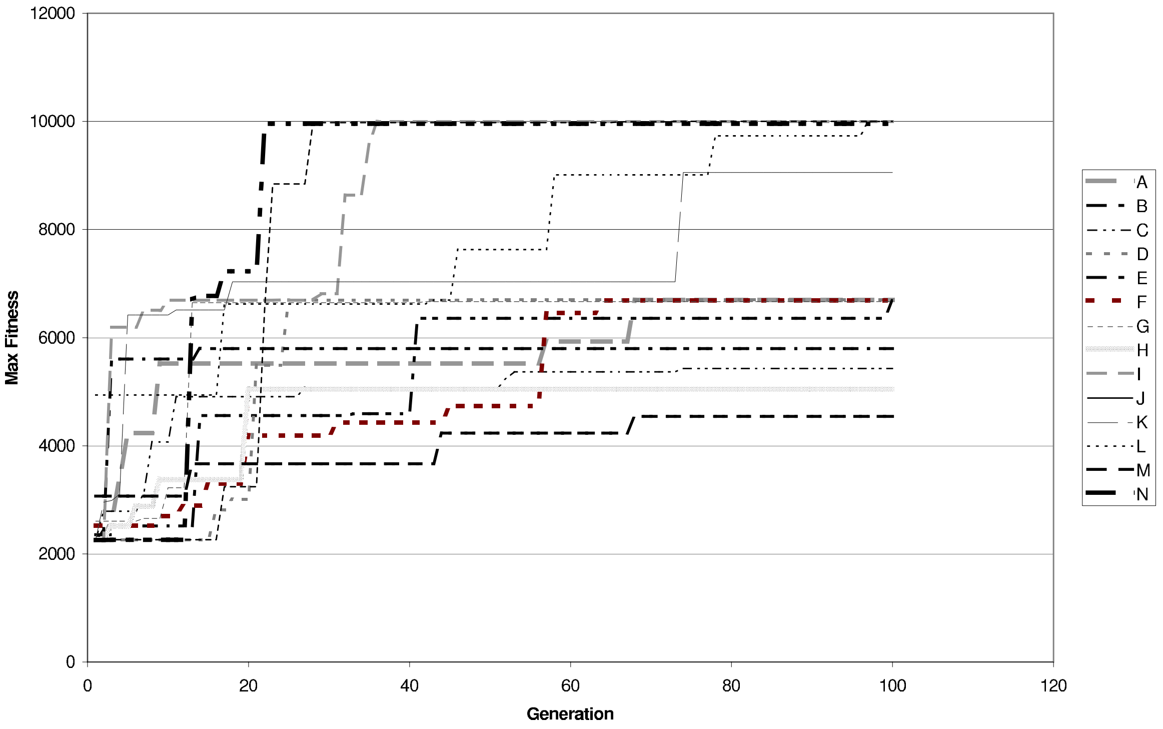

Figure 12. Evolution of controller

need to perform some sort of integration to understand the distance readings, but the CGP program essentially has this work performed for it. Another major difference is the absence of noise - the real world controller is likely to suffer from various types of noise from sensors, cables and digital-analogue conversions. If implemented with a real robot, the CGP controller may perform worse as it was not evolved with noise in the system.
These differences demonstrate that evolving a controller in liquid crystal is not a simple task when all the differences are considered. We would expect a CGP controller that had all the constraints and limitations of the liquid crystal to take far longer to evolve.

9 Conclusions

We have demonstrated, for the first time, that it is possible to evolve a robot controller in liquid crystal. The task is significantly harder than that of our previous work with liquid crystal (if only because the number of inputs and outputs to the display device has been doubled). Yet we found that it was relatively easy (in evolutionary terms) to evolve a sophisticated robot controller.
The quality of results when compared to previous work is also high. The environment is more complex than that of [19] and unlike much work on evolving GP robot controllers or neural network controllers, we solve a real-time control task. The results also indicated an evolutionary computational effort that is comparable to other examples of evolved controller (with simpler tasks). We feel our work, aside from its novelty, demonstrates that evolution in materio offers some advantages over more traditional techniques. In future work we intend to explore a wider variety of tasks and demonstrate more examples using liquid crystal as an evolvable platform. We also intend to construct our own purpose built in materio chamber so that we can investigate other materials and also have more control over the way we can configure the material.

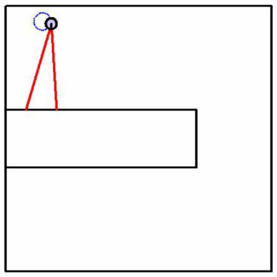
Figure 13. Fitness=515

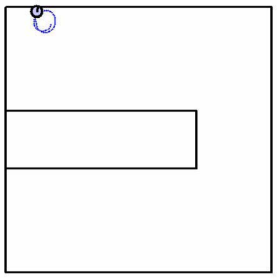
Figure 14. Fitness=517

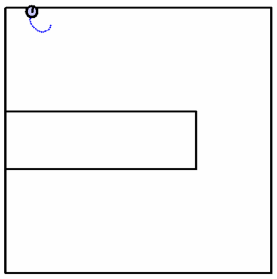
Figure 15. Fitness=518

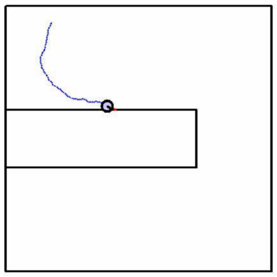
Figure 16. fitness=2809

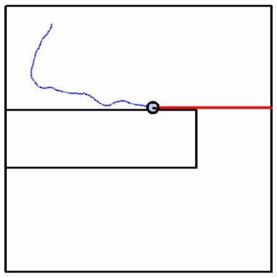
Figure 17. fitness=3819

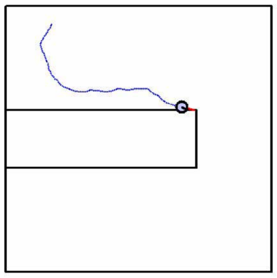
Figure 18. fitness=4607

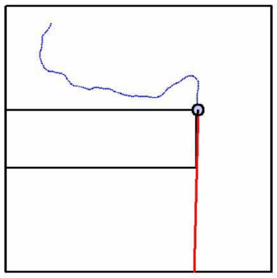
Figure 19. fitness=5569

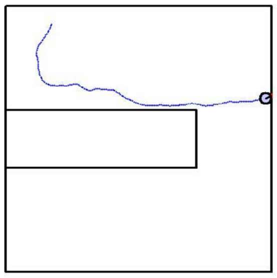
Figure 20. fitness=6686

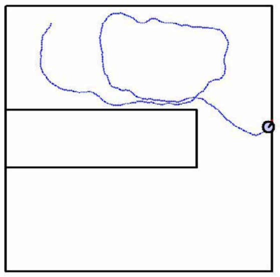
Figure 21. fitness=6772

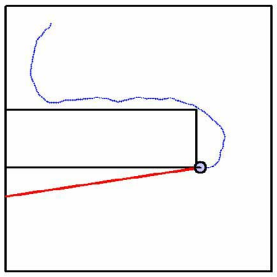
Figure 22. fitness=7229

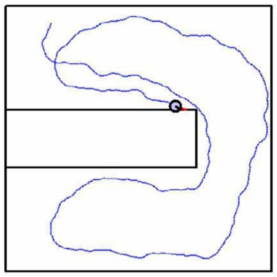
Figure 23. fitness=9796

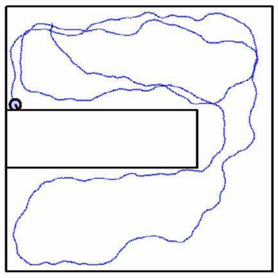
Figure 24. Symmetry in wall following behaviour: fitness=9993

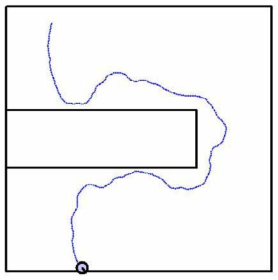
Figure 25. Symmetry in wall following behaviour: fitness=9594

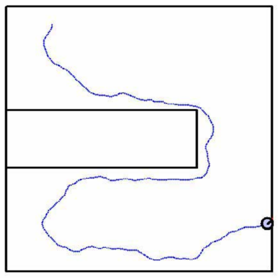
Figure 26. Symmetry in wall following behaviour: fitness=9862

References

[1] J. Crooks. Evolvable analogue hardware. Meng project report, The University Of York, 2002.

[2] R. Dain. Developing mobile robot wall-following algorithms using genetic programming. In *Applied Intelligence*, volume 8, pages 33–41. Kluwer Academic Publishers, 1998.

[3] D. Demus, J. Goodby, G. W. Gray, H. W. Spiess, and V. Vill, editors. *Handbook of Liquid Crystals*, volume 1,2A,2B,3. July 1998.

[4] M. Ebner. Evolution of a control architecture for a mobile robot. In *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES 98), Lausanne, Switzerland*, pages 303–310. Springer-Verlag, 1998.

[5] S. Harding and J. F. Miller. Evolution in materio: A tone discriminator in liquid crystal. In *In Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004)*, volume 2, pages 1800–1807, 2004.

[6] S. Harding and J. F. Miller. Evolution in materio: Initial experiments with liquid crystal. In *Proceedings of 2004 NASA/DoD Conference on Evolvable Hardware (EH'04)*, pages 298–305, 2004.

[7] S. Harding and J. F. Miller. Evolution of robot controller using cartesian genetic programming. In *To be published in proceedings of EURO GP 2005*, 2005.

[8] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[9] I. C. Khoo. *Liquid Crystals: physical properties and nonlinear optical phenomena*. Wiley, 1995.

[10] P. Layzell. A new research tool for intrinsic hardware evolution. *Proceedings of The Second International Conference on Evolvable Systems: From Biology to Hardware, LNCS*, 1478:47–56, 1998.

[11] C. Lazarus and H. Hu. Using genetic programming to evolve robot behaviours, 2001.

[12] M. J. Mataric. A distributed model for mobile robot environment-learning and navigation. Technical report, Cambridge, MA, USA, 1990.

[13] J. F. Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In *Proceedings of NASA/DoD Evolvable Hardware Workshop*, pages 167–176, 2002.

[14] J. F. Miller and P. Thomson. Cartesian genetic programming. In R. Poli and W. B. et al., editors, *Proc. of EuroGP 2000*, volume 1802 of *LNCS*, pages 121–132. Springer-Verlag, 2000.

[15] A. F. Naumov, M. Y. Loktev, I. R. Guralnik, and G. Vdovin. Liquid-crystal adaptive lenses with modal control. In *Optics Letters*, volume 23, pages 992–994, 1998.

[16] P. Nordin and W. Banzhaf. Genetic programming controlling a miniature robot. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 61–67, MIT, Cambridge, MA, USA, 10–12 1995. AAAI.

[17] C. W. Reynolds. An evolved, vision-based behavioral model of obstacle avoidance behaviour. In C. G. Langton, editor, *Artificial Life III*, volume 16 of *SFI Studies in the Sciences of Complexity*. Addison-Wesley, 1993.

[18] C. W. Reynolds. Evolution of corridor following behavior in a noisy world. In *Simulation of Adaptive Behaviour (SAB-94)*, 1994.

[19] A. Thompson. Evolving electronic robot controllers that exploit hardware resources. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacon, editors, *Advances in Artificial Life: Proc. 3rd Eur. Conf. on Artificial Life (ECAL95)*, volume 929 of *LNAI*, pages 640–656. Springer-Verlag, 1995.

[20] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *ICES*, pages 390–405, 1996.

[21] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In T. Higuchi, M. Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, volume 1259 of *LNCS*, pages 390–405. Springer-Verlag, 1997.