Evolution and Analysis of a Robot Controller Based on a Gene Regulatory Network

Martin A. Trefzer, Tüze Kuyucu, Julian F. Miller and Andy M. Tyrrell
{mt540,tk519,jfm7,amt}@ohm.york.ac.uk

Department of Electronics, University of York, UK

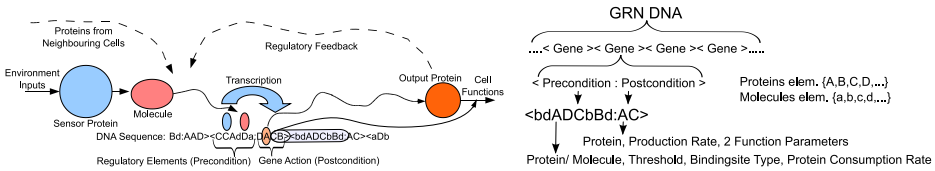
Abstract. This paper explores the application of an artificial developmental system (ADS) to the field of evolutionary robotics by investigating the capability of a gene regulatory network (GRN) to specify a general purpose obstacle avoidance controller both in simulation and on a real robot. Experiments are carried out using the e-puck robot platform. It is further proposed to use cross-correlation between inputs and outputs in order to assess the quality of robot controllers more accurately than with observing its behaviour alone.

1 Introduction

Biological development encompasses a variety of complex dynamic systems and processes at different levels, ranging from chemical reactions at the molecular level, to single cells or groups of cells dedicated to specific tasks to complex multicellular organisms that are capable of adapting to changing environments and exhibit remarkable capabilities of scalability, robustness and damage recovery [1]. It is remarkable how this large number of complex mechanisms work together in nature over long periods of time in an effective and productive manner. This makes biological development a source of inspiration for research into modelling its principles and applying them to engineered real-time systems.

Current research in the area of artificial developmental systems (ADS), gene regulatory networks (GRNs) and artificial life (ALife) concentrates on both studying developmental processes from a complex dynamic systems point of view and regarding their versatility in providing an indirect mapping mechanisms between genotype and phenotype. In the first case, the properties of ADSs, particularly GRNs, are investigated by identifying transient states and attractors of such systems [2, 3]. Hence, these approaches offer a more theoretical approach to modelling biological development. In the second case, there are examples where GRNs are utilised to grow neural networks or nervous systems for artificial agents [4]. Research that is undertaken into growing large, complex organisms that can represent a variety of things, such as patterns [5] morphogenesis in general [6, 7] or designs [8] also fits into the second category.

A third research thread seeks to exploit inherent properties of ADSs, such as the ongoing interaction between cell/organism and environment, multicellularity, chemical based gene regulation and homoeostasis, in order to achieve



(a) Mechanisms of the GRN (b) Structure of genes and GRN

Fig. 1. Protein interaction and regulatory feedback mechanisms of the ADS are shown on the left. On the right, it is illustrated how genes are divided into precondition and postcondition. Proteins can occur in both pre- and postcondition whereas molecules can only occur in the precondition.

adaptive, robust and scalable control mechanisms for robotic systems. Research is undertaken into emergent, autonomous, collaborative behaviours [9–11] and modular robotics [12]. In [13] it is shown that GRNs are a viable architecture for the on-line, real-time control of a robot.

This paper introduces a GRN based robot controller, similar to the one presented in [13]. It is investigated whether the chemical regulation based GRN mechanisms of the ADS introduced in [14] are suitable to specify a general purpose obstacle avoidance controller for the e-puck robot platform[1]. Evolutionary experiments have been conducted in simulation using the Player/Stage simulator and are validated on a real robot. Furthermore, the paper proposes using cross-correlation between inputs and outputs of the GRN controller to assess its quality and ability to adapt beyond observing behaviour alone.

2 The Artificial Developmental System

The GRN based model for artificial development used in this paper is based on the one that has been introduced in [14]. The design considerations of the original ADS are retained, namely the use of data structures and operations in the GRN core that are suitable for embedded systems, i.e. boolean, integers, no division and to keep the mechanisms of the ADS as close as possible to their biological counterparts within the boundaries of the chosen data types. Whilst not crucial for the experiments in this paper, the choice of the data structures imposes no loss of generality and is therefore unchanged. However, some improvements are made to the ADS for this paper: first, a dedicated diffusion layer is added and only chemicals that are released to this layer by the cells are subject to diffusion. Chemicals need to be absorbed by the cells from the diffusion layer first, before they affect gene regulation. This is motivated by natural development. Second, a genetic representation that allows for variable length GRNs is used in this paper, which allows for a more flexible and compact encoding of the genes, as shown in Figure 1(b).

[1] http://www.e-puck.org/

An overview of the mechanisms of the ADS is provided in the following sections. The term chemicals refers to both proteins and molecules. As the experiments in this paper are performed using one single cell, the description of cell signalling mechanisms and growth are omitted. A more detailed description of the ADS can be found in [14].

2.1 Representation and Gene Regulation

The core of the developmental model is represented by a GRN, as shown in Figure 1(a). The genotype is implemented as a string of symbols that encode start and end of genes, separation of pre- and postcondition within genes, binding sites and chemicals as shown in Figure 1(b). Genes interact through chemicals and form a regulatory network. There is at least one major difference between the artificial model and biology: in the ADS used the binding sites match exactly one chemical, whereas in natural genes binding sites are defined by certain upstream and downstream gene sequences that accept a number of proteins to bind and transcribe their genetic code. The binding sites in natural DNA therefore allow for smooth binding, i.e. the probability that certain chemicals (transcription factors) bind to the DNA is given by how well the binding sites of the chemical matches the one of the DNA.

The current GRN works with four proteins ($A \dots D$) and eight molecules ($a \dots h$). Proteins are directly produced by the GRN, whereas molecules are only a product of a gene function as a result of a measurement or interaction that is performed by a protein. In addition to gene regulation, proteins implement dedicated functions and mechanisms of the ADS. Protein A (*structuring/functional*) defines the cell type, B (*sensory*) translates sensory inputs into molecules, C (*diffusion*) manages chemical diffusion and D (*Plasmodesmata*) manages chemical sharing between cells (tunnelling) and growth. Note that the additional roles of chemicals for robot control are described in Section 2.3 and 3.

2.2 Evolution of the Genotype

The genotype is derived from a genome that is evolved using a $1+4$ evolutionary strategy (ES). The genome is represented by a string of integers and mutation takes place by replacing them with a new random value at a rate of 2% of the genome length. The GRN is obtained by mapping the string of integers to GRN symbols using the modulus operation on the genome in a straight forward fashion. Variable length genes are achieved via active/inactive flags encoded in the genes.

2.3 Developing Organisms that Control Robots

The application in this paper is to control a robot via a GRN. Therefore, the GRN has to be able to process input signals from the robot's infra-red (IR) range sensors and the outputs have to be translated into motor commands. Since

the GRN operates on chemical concentrations, this is achieved by mapping the distance measures to input chemical concentrations and by computing speed and turning rate of the robot from output chemical concentrations. Molecules are suitable to present the sensory inputs to the GRN since they affect gene regulation and can be consumed, but not directly produced. Hence, it is not possible for the GRN to directly generate input signals which are not actually present in the environment. However, molecules can be indirectly produced via the sensory protein B (Section 2.1). Contrary to the molecules, the GRN is able to quickly change the levels of the proteins (A-D) as they can be both consumed and directly produced, hence, proteins naturally represent the outputs of the system. Thus, values for speed and turning rate of the robots are calculated from protein levels. Furthermore, as proteins occur in the precondition, they provide feedback of the states of the outputs to the GRN, which can be exploited by the organism for adaptation and self-regulation. Due to the fact that one GRN with a sufficient number of proteins is able to process the inputs of one robot, a single-cell organism is used to control the robot in the experiments described.

3 E-Puck, Player/Stage and GRN

The experiments presented in this paper are carried out using the e-puck robot platform. Evolution of the ADS that controls the robot and testing on different maps is performed using the open-source robot simulation platform Player/Stage[2]. Verification of the controller is achieved on a real e-puck robot.

As described in Section 2.3, sensory inputs and motor signals are mapped to chemical concentrations which can be processed by the GRN. Due to the 16bit processor available on the e-puck, the maximum protein level is 65535 and the mapping functions for input and output signals are designed in such a way that the full protein value range is utilised. An important and still open question is how the time scales of development and the robot should be related to each other. In biology, for instance, neural networks operate at a greater speed than gene regulation, which inherently constrains those systems to certain tasks. In the case of engineering and computer science, those boundaries are not existent and therefore subject to research. In this paper, one developmental step of the controller (GRN) corresponds to one sensor/motor update cycle of 10 Hz. The latter value is given by the e-puck robot and is set accordingly in the simulation model.

3.1 Mapping Sensory Inputs

The e-puck provides 8 IR distance sensors, which are positioned around the outside of the robot at $10°, 45°, 90°, 150°, -150°, -90°, -45°$ and $-10°$. The range of the IR sensors is theoretically about 10 cm for the real robot. However, measuring and calibrating the actual IR sensor ranges of the e-puck used for these

[2] http://playerstage.sourceforge.net/

experiments shows linear behaviour only up to a maximum range of 5 cm, which is assumed as an approximation for the Player/Stage simulation, whereas the different sensors are assigned different maximum ranges in the case of the real e-puck according to the measurements taken (see max_range below). For simplicity, linear behaviour is also assumed in the case of the e-puck, despite the fact that an accurate calibration would have to take the exponential characteristics of the IR diodes into account.

This leads to the following equation for mapping IR sensor readings to input chemical levels:

$$\text{chem\_level}_i = \begin{cases} 65535 \times (1 - \frac{\text{sensor}_i}{\text{max\_range}_i}) & \text{if sensor}_i < \text{max\_range}_i \\ 0 & \text{if sensor}_i \geq \text{max\_range}_i \end{cases} \quad (1)$$

with $\text{max\_range} = 0.05$ for all i in the uncalibrated case and $\text{max\_range}_{0..7} = 0.03, 0.05, 0.05, 0.05, 0.005, 0.03, 0.015, 0.005$ in the calibrated case.

3.2 Deriving Motor Command Signals

Both the e-puck and its simulation model provide an interface that enables the speed and turning rate to be set. While speed is a value without unit between -1 and 1 (maximum reverse speed/ forward speed), the turning rate is expected in radian. Hence, computing a value for speed and turning rate from the output chemical levels can be achieved in a straight forward manner:

$$\text{newspeed} = 0.15 \times ((\text{protein}_\text{A} - \text{protein}_\text{B})/65535) + 0.05 \quad (2a)$$

$$\text{newturnrate} = 3.0 \times ((\text{protein}_\text{C} - \text{protein}_\text{D})/65535). \quad (2b)$$

where maximum speeds between $-0.1 \cdots + 0.2$, the forward speed bias of $+0.05$ and possible turning rates between $-171^\circ \cdots +171^\circ$ are arbitrarily chosen.

4 Evolution and Analysis of a GRN Based Robot Controller

The task is to optimise a GRN based controller for an e-puck robot via an EA. This experiment is carried out using a simulation model of the e-puck in Player/Stage. The aim is to achieve obstacle avoidance and area coverage in the map shown in Figure 2(a). A relatively basic map is chosen, since the aim is to achieve a low-level controller, which interacts directly with the robot's sensors and actuators rather than operating on a higher abstraction level using predefined actions or behaviours. The size of the map is 1.6 m × 1.6 m with x/y coordinates between -0.8 m.. $+ 0.8$ m.

4.1 Fitness Function

The fitness is the averaged score of three rounds of 1000 time steps (= developmental steps) each. The chemical levels are initialised to 0 before the first round.

Algorithm 1 Pseudo-code of the fitness function used.

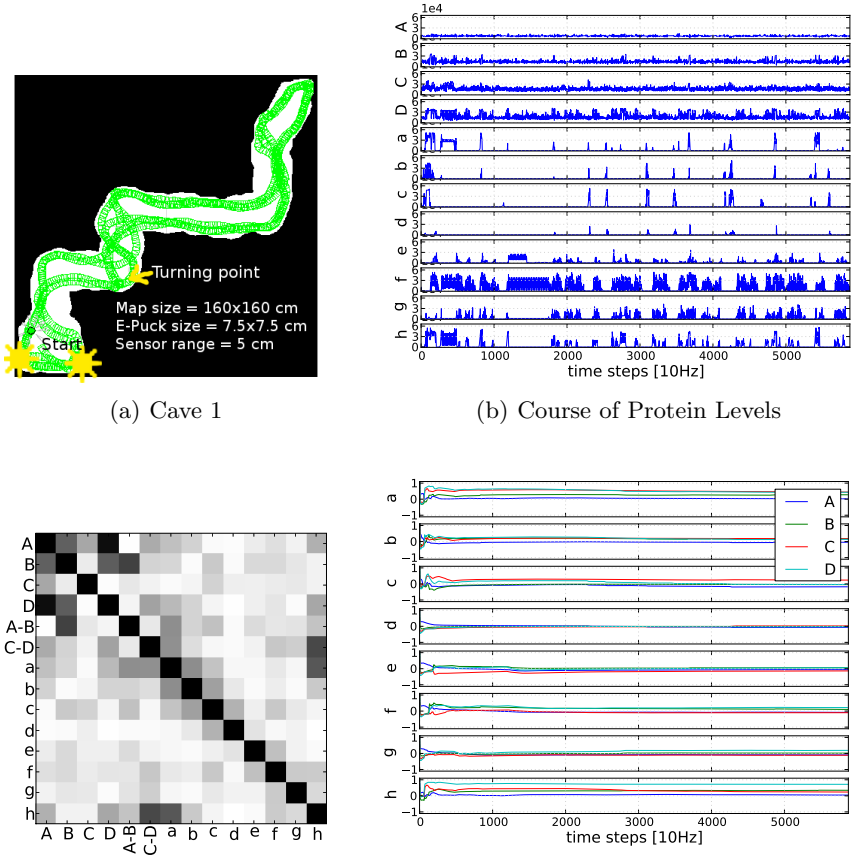
```
for three rounds do
  reset score
  reset previous distance
  randomise starting position and angle of the e-puck with
  x,y in the range of −0.65.. − 0.75 m (lower left corner)
  angle in the range of 0..360°

  for 1000 time steps do
    perform sensor reading
    map distance values to molecule levels (a-h)

    perform one developmental step
    calculate new speed and turning rate from protein levels (A-D)
    send motor commands

    // stimulating covering distance:
    if current distance to starting point > previous distance to starting point then
      score = score + distance
    end if

    // stimulating obstacle avoidance:
    if robot bumps into obstacle or wall then
      end this round (and the chance to increase score)
    end if
  end for
  add score to fitness
end for

divide fitness by number of rounds
```

For subsequent rounds, the state of the developmental system is retained in order to allow for the ADS to adapt to the environment. The fitness is calculated as shown in Algorithm 1.

4.2 Assessing Task Based Performance

In the case of a robot controller, it is possible to qualitatively assess its performance by observing the behaviour of the robot for a period of time and count the number of times it fails to avoid walls or obstacles. The ability of the robot to explore the map and reach the opposite end of the map can be observed by tracking its path. This can be easily achieved in simulation by enabling path highlighting, which is a feature of Player/Stage. In the case of the real robot this becomes more difficult as generally either a video recording or a tracking system are required.

A controller with a good performance is re-run for 6000 time steps and the resulting path of the robot is shown in Figure 2(a). The starting point of the robot is in the lower left corner of the map. At the beginning of the run, the robot bumps into walls twice, indicated by the star symbols. After that, it manages



(a) Cave 1

(b) Course of Protein Levels

(c) Correlation Matrix

(d) Course of Correlation

Fig. 2. The maze that is used to evolve the GRN robot controller, protein levels and correlation matrix are shown in the figure. $a - b$ are inputs, $A - D$ are outputs.

to navigate through the cave with no further collisions. Also, it can be seen from Figure 2(a) that the robot roams the entire cave by following the wall on its left hand side. However, the controller achieves slightly more than just wall-following, as it automatically starts turning and returns to the left wall in case it loses track of it. As can be seen from the tracks at the turning point in Figure 2(a), where the robot turns left in one round and right in another, it is not default behaviour to stop and always turn right when approaching a wall.

From this it can be concluded that the GRN achieves control of the robot in a manner that satisfies the requirements of the fitness function: the robot avoids walls and navigates as far away as possible from the starting point. The fact that the robot hits a wall only twice in the beginning of the run suggests some kind of adaptivity of the GRN based controller. Hence, the controller's ability to adapt is further investigated in Section 5.

4.3 Measuring Performance Using Cross-Correlation

Although tracking the path of the robot and counting the number of collisions are suitable to verify whether the evolved controller satisfies the behavioural requirements of the fitness function, this provides no information of the complexity of the states and the dynamics of the supposedly adaptive, GRN based controller. It would be particularly useful to have information about how the controller makes use of the input sensor data and in what way the inputs are related to the outputs, i.e. the actions of the robot, since a common problem [15] (although not analysed and published very often) with evolved controllers is the fact that they are likely to ignore inputs to the system but still manage to find partially optimal solutions.

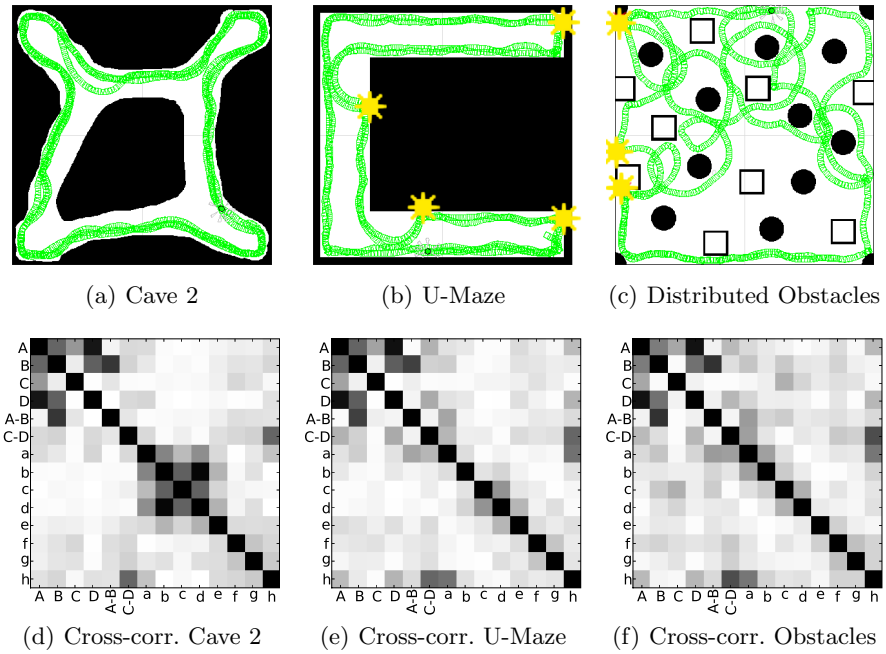
In this paper, it is proposed to use cross-correlation as a measure of dependency between sensory inputs and motor outputs. Cross-correlation is a measure of similarity of two continuous functions, which in general also considers a time-lag applied to one of them. In this case, it is assumed that the time-shift between input and output is 0. In order to obtain values in a bounded range, normalised cross-correlation is used for the experiments presented:

$$(f * g) = \frac{1}{n-1} \cdot \sum_{t} \frac{(f(t) - \bar{f}) \cdot (g(t) - \bar{g})}{\sigma_f \cdot \sigma_t}, \tag{3}$$

where $\bar{f}$, $\bar{g}$ are the mean values, σ_f, σ_g are standard deviations and n is the number of samples of the time series. Note that the usage of mean and standard deviation might be problematic, as the statistical distribution of the samples is unknown. However, using Equation 3 is convenient as the output value range is $-1 \ldots 1$, where $-1/1$ denote maximum negative/positive correlation and 0 means the signals are uncorrelated.

The measured input chemical levels (a-h) and output chemical levels ($A - D$) for 6000 time steps are shown in Figure 2(b) and the development of the cross-correlation of the chemical levels is shown in Figure 2(d). As can be seen from Figure 2(b) and 4(c), input sensors f,g,h (front, left) show almost constant activity, a,b,c (front, right) show only occasional peaks and d,e (rear) are almost always zero. This corresponds to the observed behaviour where the robot follows the left wall and only occasionally encounters a wall on its right hand side.

In order to answer the question whether the inputs are actually considered by the controller when generating the output chemical levels A, B, C, D—which define speed and turning rate of the robot according to Equation 1—, the course of the cross-correlation values over time (at each point in time from $0 \ldots t$) for each input/output chemical pair is shown in Figure 2(d). At the beginning of the run, the cross-correlation values keep changing before settling to particular values (although there are still slight adjustments taking place at later iterations, e.g. in the case of a,c,h). Again, this suggests that, to a certain extent, an adaptation process is taking place at the beginning of the run. However, this needs to be further consolidated by the following sections, when the controller is tested on different maps and on a real robot. For a better overview, the cross-correlation matrix of time step 5000 is shown in Figure 2(c), including the differences $A - B$



(a) Cave 2 (b) U-Maze (c) Distributed Obstacles

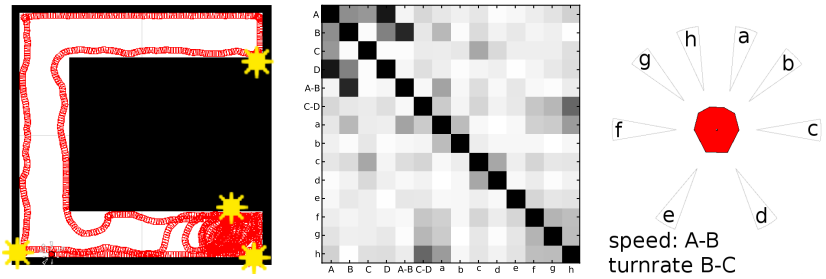
(d) Cross-corr. Cave 2 (e) Cross-corr. U-Maze (f) Cross-corr. Obstacles

Fig. 3. The figure shows a comparison of the behaviour of the GRN controller in different maps.

($\propto$ speed) and $C - D$ ($\propto$ turning rate). Looking at the correlation matrix (black = max. absolute correlation, white = no correlation), it can be confirmed that there is a correlation between speed and turning rate and the sensors at the front and the sides. It is interesting to see that $A-B$ appears to be correlated to B, but not A, and $C-D$ appears to be more correlated to C than D. This suggests that the evolved controller keeps A and D relatively constant and achieves changes in speed and turning rate by adjusting the chemical levels of B and C.

5 Test and Analysis on Different Maps

In order to investigate whether the controller exhibits—at least to a certain extent—adaptive behaviour, it is tested in simulation on three different maps, shown in Figure 3(a,b,c). As can be seen from the recorded tracks, the controller successfully navigates the robot through the three maps, which feature different characteristics: the first map (Figure 3(a)) is similar to the one in which the controller is evolved. Hence it is expected that the robot does not collide with walls in this case. Since the primary behaviour of the evolved controller is wall-following, the robot explores the additional branches present in this map, rather than going straight to the opposite side of the map. The second map (Figure 3(b)) represents a simple U-maze. The important feature of this map are the straight edges and the sharp 90° turns, which impose challenges on the controller. It



(a) Trace of the real e-puck. (b) Correlation Matrix (c) E-Puck sensors

Fig. 4. Results from experiments with the real e-puck are shown in the figure.

is observed that, although the robot manages to navigate through the entire map, it hits the wall in all cases where it approaches the wall under a right angle. In those cases, the controller is unable to decide in which direction to turn. The third map (Figure 3(c)) is significantly different one to the one used for the evolution of the controller. Despite that, the robot successfully navigates around the obstacles and explores the map. It is interesting to see that in this case the robot bumps into obstacles only in the beginning of the run (starting point is in the lower right corner of the map) and manages to successfully avoid all obstacles as time goes on. This is again a hint that an adaptation process is actually happening.

When comparing the cross-correlation matrices in Figures 3(d,e,f), it can be observed that the cross-correlation values at which the controller settles at time step 5000 are different for different environments. Particularly in the case of the third map (Figure 3(c)), the correlation between the outputs and sensors c,d,e,f has increased. This indicates that those sensors are playing a more important role in the case of the third map.

The results show that the cross-correlation matrix looks different for different maps, which indicates that the controller indeed features different states of operation, depending on the environment. It is an open question and subject of future work how to investigate whether this emergent adaptivity is a general, inherent property of 'soft' controllers, rather than ones based on thresholds and decisions, or whether it is a particular feature of GRN based, developmental controllers.

6 Test and Analysis on an E-Puck Robot

In order to show the relevance of the presented experiments for real-world applications, the evolved GRN robot controller is tested on an e-puck robot. The only modifications that are made for the real robot is using the calibrated maximum sensor range values rather than the same for each sensor, as described in Section 3 and Equation 1. The results obtained with the real robot are shown in Figure 4. For visualisation, the path of the robot has been manually traced in Figure 4(a) using Player/Stage.

It is observed that the e-puck is trapped for more than 2500 out of 6000 time steps in the lower-right corner of the map shown in Figure 4(a), before it successfully manages to resume its primary wall-following behaviour and navigate through the map. The fact that the behaviour is similar to the one observed in simulation (see Figure 3(b)) after it manages to escape the corner raises again the question whether this is achieved as a result of adaptation.

It can be seen from the cross-correlation matrix that the controller settles in a state that looks similar to the one from simulation, but the development of the cross-correlation values over time is significantly noisier. In order to quantitatively compare the cross-correlation matrices, it will be necessary to define a distance measure or visualise the state space given by the cross-correlation matrices as part of future work.

7 Discussion

This paper[3] has explored the application of an ADS to the field of evolutionary robotics by investigating the capability of a GRN to control an e-puck robot. A GRN controller has been successfully evolved that exhibits a general ability to avoid obstacles in different maps as well as when transferred to a real robot. It has been shown that GRN based controllers have the potential to adapt to different environments, due to the fact that the robot successfully managed to navigate through previously unknown maps and could be successfully transferred to a real robot without further modification of the controller. Hence, it is concluded that GRNs are a suitable approach for real-time robot control and can cope with variations inferred by changing environments and sensor noise of a real robot. The results further suggest that it is possible to specify a general purpose obstacle avoidance behaviour via a GRN.

It is proposed that cross-correlation between inputs and outputs is a suitable measure to quantitatively assess the quality of robot controllers (particularly evolved ones) beyond observing whether the robot exhibits the desired behaviour only. It has been shown that the cross-correlation settles at different values for different environments. On the one hand, this simply confirms that the level of activity and importance of the sensors changes for different environments. On the other hand, in conjunction with the observation that the robot still exhibits the desired behaviour, different cross-correlation matrices for different environments indicate that the controller features different stable states of operation and shows the ability of the controller to autonomously adapt to a certain extent. As the experiments show, this is the case for different maps in simulation and when transferring the controller to a real robot.

However, it is an open question and subject to future work how to investigate whether this emergent adaptivity is a general, inherent property of 'soft' controllers, rather than ones based on thresholds and decisions, or whether it is a specific feature of the GRN based, developmental controllers like the one

[3] This work is part of a project that is funded by EPSRC - EP/E028381/1.

introduced in this paper. One of the the greatest challenges in evolutionary computation (EC) is the design of the fitness function. This is particularly true in the case of behavioural fitness functions and real-world systems which can be extremely noisy, i.e. good solutions have a significant probability of being discarded during the optimisation process simply because of unlucky initial conditions at one iteration of the EA. Therefore, we will explore the possibility of including the cross-correlation measure in the fitness function, in order to provide an additional quality measure which is independent of the behaviour. Even if the robot does not solve the task, it will be possible to emphasise correlation between inputs and outputs which will prevent evolution from ignoring the inputs and might offer a means to overcome sub-minimally competent controllers, particularly in the beginning of the optimisation process.

References

1. Wolpert, L., Beddington, R., Jessel, T., Lawrence, P., Meyerowitz, E., Smith, J.: Principles of Development. 2 edn. Oxford University Press Inc., New York (2002)
2. Kauffman, S.: Metabolic stability and epigenesis in randomly constructed genetic nets. Journal of Theoretical Biology **22**(3) (March 1969) 437–467
3. De Jong, H.: Hybrid modeling and simulation of genetic regulatory networks: a qualitative approach. In: ERCIM News, Springer (2003) 267–282
4. Astor, J.C.: A developmental model for the evolution of artificial neural networks: Design, implementation and evaluation. Artificial Life **6** (1998) 189–218
5. Miller, J.F.: Evolving developmental programs for adaptation, morphogenesis, and self-repair. In: 7th European Conf. on Artificial Life, Springer LNAI (2003) 256–265
6. Eggenberger, P.: Evolving morphologies of simulated 3d organisms based on differential gene expression. In: Proc. of 4th European Conf. on ALife. (1997) 205–213
7. Bentley, P., Kumar, S.: Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In: Proc. of the Genetic and Evolutionary Computation Conf., Orlando, Florida, USA, Morgan Kaufmann (1999) 35–43
8. Hornby, G.S.: Generative representations for evolving families of designs. In: Genetic and Evolutionary Computation (GECCO), Springer (2003) 1678–1689
9. Quick, T., Nehaniv, C.L., Dautenhahn, K., Roberts, G.: Evolving embodied genetic regulatory networks-driven control systems. In: Proc. of the 7th European Conf. on Artificial Life (ECAL). (2003)
10. Floreano, D., Mondada, F.: Evolution of homing navigation in a real mobile robot. IEEE Trans. on Systems, Man, and Cybernetics–Part B (1996) 39640–7
11. Ziegler, J., Banzhaf, W.: Evolving control metabolisms for a robot. Artificial Life **7** (2001) 171–190
12. Gro, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarmbots. IEEE Trans. Robot (2006) 1115–1130
13. Kumar, S.: A developmental genetics-inspired approach to robot control. In: Proc. of the Workshops on Genetic and Evolutionary Computation (GECCO), New York, NY, USA, ACM Press (2005) 304–309
14. Trefzer, M.A., Kuyucu, T., Miller, J.F., Tyrrell, A.M.: A model for intrinsic artificial development featuring structural feedback and emergent growth. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC), Norway (2009)
15. Tarapore, D., Lungarella, M., Gomez, G.: Quantifying patterns of agent-environment interaction. Robotics and Autonomous Systems **54**(2) (2006) 150–158