Sensor-Coupled Fractal Gene Regulatory Networks for Locomotion Control of a Modular Snake Robot

Payam Zahadat, David Johan Christensen, Serajeddin Katebi, and Kasper Stoy

Abstract. In this paper we study fractal gene regulatory network (FGRN) controllers based on sensory information. The FGRN controllers are evolved to control a snake robot consisting of seven simulated ATRON modules. Each module contains three tilt sensors which represent the direction of gravity in the coordination system of the module. The modules are controlled locally and there is no explicit communication between them. So, they can synchronize implicitly using their sensors, and coordination of their behavior takes place through the environment. In one of our experiments, all the three tilt sensors are available for the FGRNs and a simple controller is evolved. The controller is a linear mapping of one input sensor to the output. It is only based on one sensor input and ignores the other sensors as well as the regulatory part of the network. In another experiment, the controller's input uses one of the other sensors that carries less information. In this case, the evolved controller blends sensory information with the regulatory network capabilities to come up with a proper distributed controller.

1 Introduction and Related Work

Modular robots are distributed robots made up from a number of mechanically coupled modules where each module is typically controlled by its own local controller. These robots are distributed and dynamic by nature and they have limited inter-modular communication and processing capabilities. In this paper we evolve FGRNs as distributed controller for modular robots. The purpose of the paper is to study the FGRN controllers based on sensor information. The FGRNs are evolved as local controllers of modules. Each FGRN controller receives inputs provided by

Payam Zahadat · David Johan Christensen · Kasper Stoy
Modular Robotics Lab, The Maersk Mc-Kinney Moller Institute,
University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark

Payam Zahadat · Serajeddin Katebi
Department of Computer Science and Engineering, School of Engineering,
Shiraz University, Shiraz, Iran

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 517–530.
springerlink.com

the local sensors of the module containing it. The usefulness of the sensor information and the FGRN capability to make proper output patterns are investigated in this paper.

Gene regulatory network (GRN) is a network of genes which interact with each other and regulate each other's activation behavior. Instead of direct mapping of genotypes to phenotypes, nature implements an indirect development of a phenotype using GRNs. In biological cells, genome consists of a number of genes which encode proteins. Proteins play different roles in a cell. They can represent input signals; operate as intermediate substrates to drive the interaction between genes, and shape structure or behavior of the cell which can change during time. Proteins interact with each other and with the genes and this is an ongoing process in the whole life time of a cell. Complex behavior of a cell is the result of this interaction. Production of a protein can be initiated by signals coming from the environment of the cell. The environment might be either the outside world or even the neighboring cells. In this way, the local environment can influence the cell's inner dynamics and changes the behavior of the cell. Differentiation of cells in a multicellular creature takes place through similar processes. In a multi-cellular creature, all cells contain the same genome, but based on the local environment of the call they differentiate during development and may behave differently.

In the field of computation systems different GRN models [2, 12, 14, 18] have been defined to indirectly map genotype to phenotype in order to make more complex phenotypes and behaviors. In some works, models of GRNs are evolved for making mathematical output functions [19], developing neural networks for controlling robots [9, 11, 16] or specifying the morphology of 3D organisms [10]. Also, GRN models have been used to develop the morphology of robots as well as their neural network controllers [6]. A special type of GRNs, which utilizes fractal proteins as the intermediate substrate of gene interaction, is called FGRN [3]. The recursive and self-similar nature of fractal proteins make the fractal genetic space evolvable, complex, and redundant [3, 4, 5]. In a number of previous works, FGRNs are evolved to do different tasks such as producing desired patterns, controlling conventional robots and motion planning [4, 26]. They have also been used [26] as local controllers of modular robots in a simpler version than the current paper such that each FGRN controller selects between different possible commands that can be executed by every module and without any dynamic influence from the outside environment.

The main contribution of this paper is further investigation of the usefulness of FRGN for control of modular robots; in particular, we extend on previous work [26] by looking at how sensor-inputs can be integrated with FGRN. The controllers we develop in this paper are tied to the physics of the ATRON self-reconfigurable robot and are thus not directly applicable to control of other modular robots such as M-TRAN [20], SuperBot [22], CKBot [25] due to their differences in weight, actuator strength, placement of sensors, etc. However, it is a general problem of all embodied controllers that they rely on the specific physical properties of the robot on which they run. For the same reason, the controllers cannot be directly applied to control of non-reconfigurable snake robots either (see [23] for an overview). However, the idea of a model-free approach relying on

tilt-sensors for both local control and implicit synchronization between segments of the snake may be transferable to other robots as well. More importantly, we expect our development method based on evolution of FGRN can be applied to these systems. In modular and multi-segment robotics, controllers for snake robots have been extensively studied based on gait control tables [24], Central Pattern Generators (CPGs) [15, 17], artificial hormones [13], and role-based control [22]. However, opposed to our controllers these controllers except [17] are open-loop and in the case of the latter two rely on explicit communication between modules for synchronization.

The paper is organized as follows. The next section reviews the biological inspiration and computational implementation of FGRN. Then, the application of FGRN as a sensor-based controller is described. Consequently, the controllers which respectively are evolved with unrestricted and restricted access to sensor information are investigated and the achieved behaviors are compared.

2 Gene Regulatory Networks

2.1 Biological Inspiration

Development of phenotypes can be thought of as a product of interaction between genes and proteins in their environment. Proteins drive development and functioning of a cell and are used for communication between a cell and its environment that might include other cells.

A cell contains a genome and a cytoplasm which are surrounded by a membrane (Fig. 1) [1]. The membrane separates the interior of a cell from the outside environment. Receptor proteins are embedded in the membrane and control the movement of environmental proteins into the cell. The cytoplasm contains a compound of proteins inside the cell. The genome consists of a set of genes. Every gene contains a sequence that encodes a protein (coding region) and a sequence that determines the conditions for activation or suppression of that gene (promoter region) (Fig. 1).

An active gene expresses and produces its appropriate protein as encoded in its coding region. For a gene to be activated, the similarity between the cytoplasm content and the promoter region of the gene has to reach a threshold.

Fig. 1 An example cell (left) and a gene (right)

The cytoplasm content is altered by proteins produced by genes inside the cell or the environmental proteins which have entered the cell passing through receptors.

During the development of a cell, the protein content of the cytoplasm might match against the promoter of some genes and get them to suppress or express proteins. Every produced protein will enter to the cytoplasm and alter its content. The new content, in turn, affects the expression of genes in the next step. In this way, every protein inside a cell either produced by the genes or from environment might influence the expression of the genes directly or indirectly. On the other hand, the functional behavior of a cell is determined by special proteins in the cell and is controlled by the cytoplasm content.

The ongoing interaction between proteins and genes continues for the whole lifetime of a cell and is considered a network of genes which regulate the expression of each other and is called a Gene Regulatory Network (GRN).

2.2 Fractals and Gene Regulatory Networks

In a series of works reported by Bentley [3, 4, 5] a protein model called fractal protein is developed as an abstraction of the protein substance of gene regulatory networks in an evolutionary system.

Fractal proteins are square windows on the Mandelbrot fractal set with a fixed resolution (Fig. 2). Each fractal protein is represented by a square matrix of integer values, but it is encoded by only three values (x, y, z). (x, y) is the coordination of the center of the window on the fractal set and z is the length of the sides. Therefore, by changing these three values we can reach different locations and different scales of the fractal set which benefit the evolvability due to the self-similarity found in fractals. This property makes a desirable redundancy which means the same potential solution can be found in indefinite number of points in genotype space and it facilitates the evolutionary process. Fig. 2 represents an example fractal protein.

In addition to a square matrix of integer values, a single integer value relates to each fractal protein as its concentration level. The concentration level represents the current amount of the protein. The value increases when more of the protein is produced and decreases slowly over time to resemble normal degradation that happens in biological cells. The value is constant for the receptor proteins.

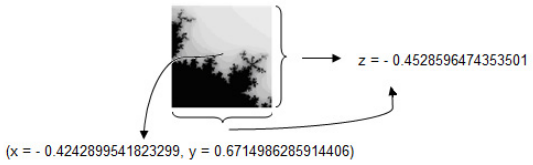

Fig. 2 An example fractal protein and the three values which specify it

Fractal proteins can merge together and make protein compounds. A fractal protein compound is represented by a square matrix of integer values in the same way as fractal proteins. Merging is a pixel-wise max operation between the corresponding matrices. See Fig. 3(a) for an example.

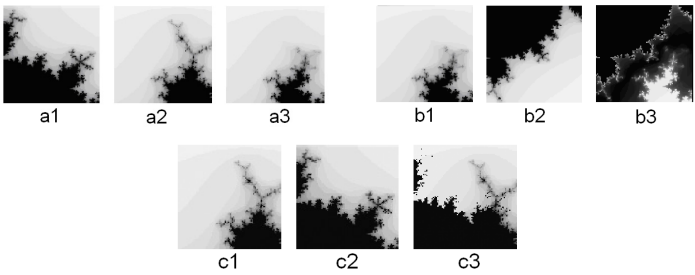

Fig. 3 Fractal Protein Operators: a) Merge: Two proteins a1 and a2 are merged as a3. b) Match: The cytoplasm protein compound b1 matches against the promoter of a gene (b2) and b3 is resulted as the calculated absolute difference. c) Mask: Environmental protein c1 passes through the receptor protein c2 and some portions of it (c3) which are corresponding to non-black pixels of c2 are allowed to enter the cytoplasm.

The cytoplasm of an FGRN cell is a compound of all the proteins inside the cell. Every protein that is produced in the cell or enters the cell from outside will be merged into the content of the cytoplasm.

A genome in an FGRN cell consists of a set of genes. Genes consist of a sequence of values representing promoter region, coding region, threshold parameters, and type of the gene.

The coding region contains the three real values which encode a fractal protein. In the same way as the coding region, the promoter region consists of three real values that encode a square matrix of fractal values as well. This matrix works as a window that will be put on the cytoplasm protein compound matrix and is used to calculate the matching degree between the promoter of the gene and cytoplasm content (See Fig. 3(b) for an example). The matching degree along with the total concentration of matched proteins on promoter region, determine the degree of activation (or suppression) of the gene and might specify its protein production rate. Threshold parameters are used to calculate the matching degree and protein production rate of each gene. To assimilate different types of genes in a cell, every gene belongs to one of the types represented in Table 1. Each gene contains an integer value that represents its type. The lifetime of an FGRN cell consists of a number of developmental cycles which can be summarized as the steps represented in Fig. 4. For more detailed descriptions of FGRN systems and the corresponding formulas see [3, 4, 26].

Table 1 Different gene types

Gene Type	Description
Regulatory	Includes both promoter and coding region. Its encoded protein will be produced and merged into cytoplasm and participate in regulation of gene expression.
Environmental	Determines the proteins which might be present in the environment of the cell.
Cell receptor	Contains a coding region and produces a receptor protein. Receptor proteins merge together and act as a mask to permit variable portions of environmental proteins to the cytoplasm (See Fig. 3(c)).
Behavioral	Comprises a promoter region and a coding region. The values in the coding region can directly participate to determine the outputs of the cell.

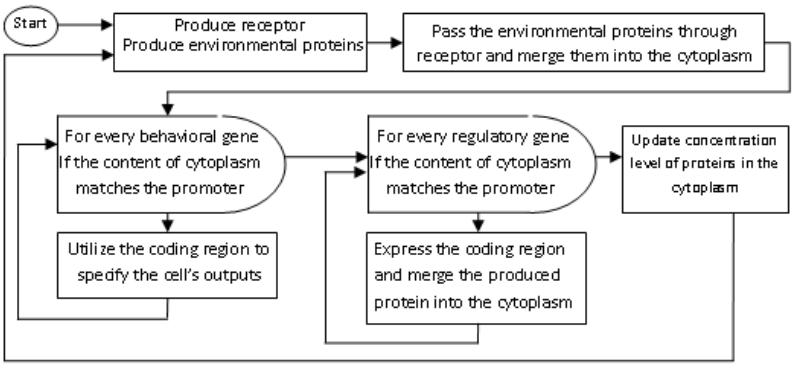

Fig. 4 A developmental cycle of an FGRN cell

3 Evolving FGRN Local Controllers with Tilt-Sensor Input

In this work, FGRN controllers are evolved for the ATRON robot [21] which is a homogenous, lattice-based self-reconfigurable modular robot. An ATRON module weighs 0.850kg and has a diameter of 110mm. A module consists of two hemispheres which can rotate infinitely relative to each other with a speed of 60 degrees per second. Each hemisphere contains two passive (bars) and two active connectors (hooks), see Fig. 5.

Fig. 5 From left to right: An ATRON module, a seven-segment snake robot

Simulation experiments are performed in an open-source simulator named Unified Simulator for Self-Reconfigurable Robots (USSR) [8]. The simulator is based on Open Dynamics Engine which provides simulation of collisions and rigid body dynamics. Physical forces like gravity and friction are implemented and the parameters, e.g. strength, speed, weight, etc., has been calibrated with the existing hardware. The physical validity of the mechanical simulation has been demonstrated in the previous works [7] where the controllers were successfully transferred from simulation to the real modules. The implemented sensors are ideal tilt sensors and not still verified.

FGRN local controllers with access to tilt-sensor inputs are evolved. The controller is evolved for a snake-shaped robot consists of seven ATRON modules (See Fig. 5) and there is no explicit communication or synchronization between the modules. Every module contains three tilt sensors as (TiltX, TiltY, TiltZ). The sensors specify the direction of gravity related to the coordination system of the module (Fig. 5). The initial tilt sensor values of a module are different for the neighbor modules because of the positioning of the connectors in ATRON. The initial values are (0, -90, 0) for the modules in the odd positions of the snake and (-90, 0, 0) for the ones in the even positions.

Evolution searches for FGRN genomes which are used in the local FGRN controllers to solve a locomotion task. To evaluate a genome, an identical version of genome is copied to all the FGRN cells which are situated in the modules. Each cell receives tilt sensor values from the module's local sensors. Initially, one input gene is related to each sensor. The level of protein expression of each input gene is determined by the value received from the related sensor. The development cycle in Fig. 4 is performed and the new concentration level of each protein in the cytoplasm is specified. In order to make an actuator command for each module in every step, each module independently run its own FGRN cell for one developmental cycle and receives an output from the cell. The FGRN output is calculated on the basis of activation level of behavioral genes and the real values of the coding region [4]. The output value received from the cell is scaled and used as the

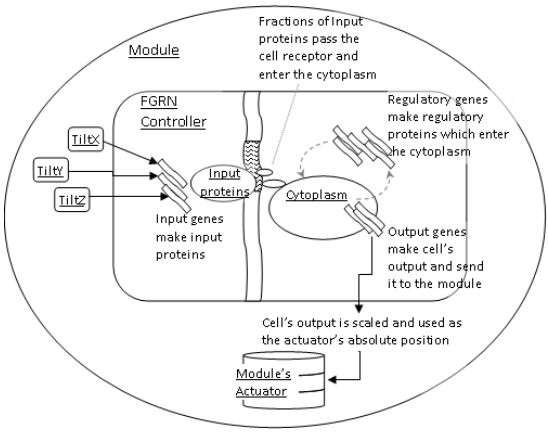

Fig. 6 Each module contains an FGRN controller that specifies the actuator's absolute position

absolute position of the module's actuator which is between -180 to 180 degrees (See Fig. 6). Modules use the nearest rotation angle to reach the desired absolute position. Robot runs for a specific time period (50 sec.) and fitness is simply evaluated as the average speed of locomotion of the robot. For each evolutionary run, a population of 50 FGRN genomes is evolved for 250 generations using a version of steady-state genetic algorithm with lifespan limits [2]. Each genome is initialized with randomly generated regulatory, receptor, environmental, and behavioral genes. Evolution is allowed to regulate the number of each type of genes (See [3, 26] for more details).

4 Experimental Results and Discussion

In order to investigate the usefulness and properties of integrating sensors with FGRN controllers for the snake robot, we performed two experiments. First we studied the FGRNs that are evolved when the input is available from all of the local tilt sensors and observed the usefulness of the sensor-inputs. Then we examined the ability of FGRN to produce proper output patterns when the input is limited to a sensor with less information.

4.1 Evolving Controllers with Unrestricted Access Sensors

In order to study if FGRN controller can gain any benefit from the tilt sensor inputs, we evolved the controllers with access to all the three local sensors. Evolution was free to use all or some of the sensor inputs for the controllers. The evolved controllers were evaluated in the locomotion task and the speed of locomotion was measured as the distance between the initial position and the end position of the center of mass of the robot and used as the fitness value. We repeated the experiment for 10 independent runs. The average speed of the best controllers from the ten runs was 0.0334 m/s (with standard deviation of 0.0032) and all the runs evolved controllers that generated rolling locomotion.

In order to investigate the effects of different sensor values in producing robot behavior, we limited access of the evolved controller to different combinations of the sensors and set the others to zero. The achieved results demonstrated that for 9 runs out of 10, there is no detectable effect for the TiltY and TiltZ sensors. In the only other run, output was produced based on both TiltY and TiltZ sensor values and no use of TiltX detected. This controller had the speed of 0.027 m/s.

In the same way as the sensor values, we removed regulatory genes of the evolved FGRN controllers in order to investigate their influence on the controllers' behavior. The investigation demonstrated that only in one of the evolved solutions, regulatory genes were participating in producing the controllers' output. No significant difference was observed between the speed of this controller and the rest.

Based on the above investigations, for the eight runs out of the 10 runs, the evolved controllers produced output merely from TiltX sensor value. This means the controller directly maps one input to the output which is a simple controller for this robot.

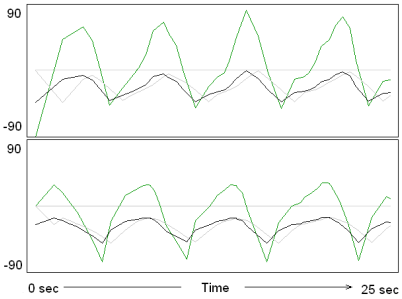

Fig. 7 Internal dynamics of the controllers of the first experiment for the two first modules of the snake. Green lines represent sensor value, black lines represent the output for the actuator absolute position, and the gray lines represent the actuator's real position.

For a typical evolved FGRN controller, the internal dynamics of the modules are represented in Fig. 7. As it is demonstrated in the figure, the output values can be simply calculated using a linear equation. We derived the equation from the related input and output data as:

$$\text{Output} = \text{TiltX} * 0.33 - 60.8$$

4.2 FGRN with Restricted Sensor Information

In the second experiment, we investigated whether the regulating dynamics of FGRN can make proper output patterns when the instant values of input sensors doesn't carry enough information. As the results of the last experiment demonstrated, TiltZ sensor has no detectable effect in producing the control outputs. It made us suspect that this sensor doesn't have enough information for this control task. Therefore, we first tried to evolve a linear equation solely based on TiltZ sensor. We implemented a real-valued genetic algorithm to evolve a population of 50 individuals for 250 generations. The experiment was repeated ten times and we observed that evolution failed to find a proper controller. Then, we evolved FGRN controllers which have only access to TiltZ sensor value to investigate if FGRN can exploit this restricted sensor information.

We repeated the evolutionary process for 10 independent runs and observed different locomotion-types for the best controllers of the different runs. The locomotion-types are discussed in three groups. The first group consists of the controllers which generate rolling-type locomotion for the robot. In order to study which parts of the network are involved in the control process, we disabled the sensor and each of the regulatory genes one by one. In all cases the controller failed to make proper locomotion. It demonstrates that both regulatory genes and sensor input are used by the controller. The Internal dynamics of one of the best controllers we achieved in this group is represented in Fig. 8. In order to get an informal impression of the robustness of the controllers in case a module breaks which lead to restarting controller, we randomly chose a module and restarted its controller to the initial state during the robot's run. We repeated the experiment several times and observed that the robot continues its normal locomotion after a short while.

The second group includes the controllers that make crawling-type locomotion. Investigation of the internal dynamics of the controller demonstrates that these solutions are mainly based on the regulatory genes and doesn't really exploit the input information. We observed that these robots are not robust against randomly restarting of the controllers.

The third group consists of the controllers which make efficient locomotion once in a while. Benefiting from the robot's body accidental flips over, these controllers sometimes make fast locomotion, and otherwise they do not produce locomotion. Since evolution only searches for fast controllers and there was no selection pressure towards the robustness and reproducability of the locomotion, the large fitness that these controllers gain by the accidental success is enough to pick them up among the other controllers in the evolutionary process. These controllers are not robust even during normal locomotion. Average and standard deviation of speed reached by the different controller groups are shown in Table. 2.

Table 2 Speeds reached by different types of locomotion

	All	Rolling	Crawling	Others
Average speed	0.0209	0.0248	0.0168	0.0212
Standard deviation	0.0076	0.0047	0.0015	0.0112

The inner dynamics of a typical controller is represented in Fig. 8. The controller is selected from the rolling-type group which demonstrates an efficient and robust behavior. As it is represented in the figure, the TiltZ value is zero for all the modules on the start of the execution. Therefore, there is no difference between the cells of a robot at the beginning and all of them make the same output for their module actuators. Rotating actuators as a result of command execution, changes module's orientation. This might lead to different TiltZ sensor values for different

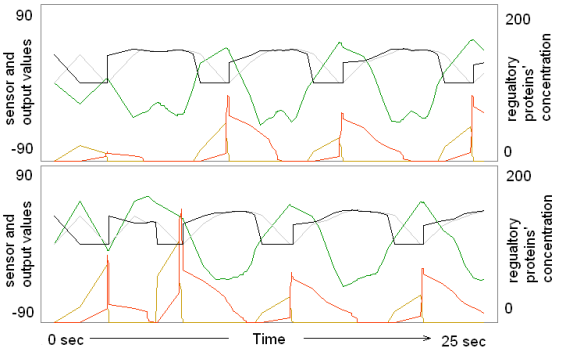

Fig. 8 Internal dynamics of the selected controller of the second experiment for the two first modules. Green lines represent the sensor values, black lines represent the output for the actuator absolute position; and the red and brown lines represent the concentration level of the two regulatory proteins.

modules. After a short while of chaotic behaviors, modules start to synchronize and coordinate their behaviors through environmental feedback which is received in the form of the sensor values.

4.3 Comparison of Behaviors from the Best Evolved Controllers of the Two Experiments

In order to have an impression of the rolling behavior produced by the best controllers of each of the two experiments, we studied the actuator's absolute positions for one typical controller evolved in the first experiment and one typical controller from the second one.

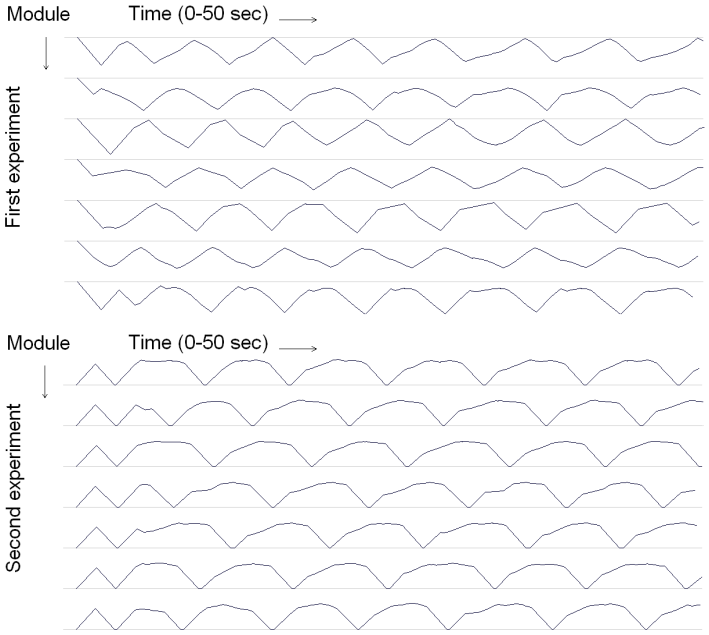

Fig. 9 Actuators' absolute positions of all the modules for a typical controller of the first and second experiments respectively

As it is demonstrated in Fig. 9, the module actuators have oscillatory behaviors. For the first experiment, the average period of estimated oscillation of the actuator absolute positions is 6.46 sec (with standard deviation of 0.63). The estimated phase shifts between the actuator signals of the consecutive modules is represented in Table 3.

For the second experiment, the average period of oscillation of the actuator absolute positions is 7.6 sec (with standard deviation of 0.2). The estimated phase shifts of the consecutive modules are represented in Table 3.

Table 3 Phase shift (per period) between the neighbor modules for the typical controller of each experiment

Module number	#1	#2	#3	#4	#5	#6	#7
first experiment	-	0.21	0.52	0.34	0.34	0.41	0.48
second experiment	-	0.56	0.37	0.41	0.40	0.36	0.38

It is interesting to note that the phase difference between neighbor modules is not constant. We suspect this is because modules are subject to different forces and dynamics depending on their position in the snake.

5 Conclusions

In this paper we explored fractal gene regulatory network controllers for a snake-shaped modular robot where only the tilt sensor inputs are available for the controllers. First, we provided the controllers with all the three tilt sensor inputs. The evolved controllers were simple linear equation which exploits only one of the three sensor inputs and ignores the regulatory abilities. In the next step, we restricted the controller's access to one of the other sensors and tried to evolve new linear equation controllers based on this information. Since evolution couldn't find the proper controllers, we suspect that the information provided by that sensor is not enough to be used by such a simple controller.

Then we evolved FGRN controllers with access to this sensor information. The resulting controllers made appropriate oscillatory output patterns to control the modules. Investigating the different parts of the FGRN genome demonstrated that the system exploits both sensor values and regulatory network capabilities to make the proper controller commands. As it might be expected, the generated outputs of the controllers were oscillatory patterns shifted for each module. Furthermore, we performed some preliminary tests towards robustness of the controllers in both cases and observed that the controllers can drive the robot properly in the case of random restarting of the controllers during locomotion.

All in all, as an early step to use FGRN as a modular robot controller, it is demonstrated that FGRN can be evolved to both simple and relatively complex controllers depending on the problem. Furthermore, when the capability of FGRN to make oscillatory patterns is coupled with the sensor information, the controllers show some degree of adaptability. In this way, the identical controllers generate different oscillatory outputs when situated in different modules and may provide some levels of robustness for the whole system. While it has not been verified we think that the idea of using sensor-coupled FGRN controllers for local control and synchronization between segments can be transferable to other modular robots as well.

Acknowledgments. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 - Future Emerging Technologies, Embodied Intelligence, under grant agreement no. 231688.

References

1. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Molecular Biology of the Cell, 4th edn., Garland (2002)
2. Banzhaf, W.: On evolutionary design, embodiment and artificial regulatory networks,". In: Iida, F., Pfeifer, R., Steels, L., Kuniyoshi, Y. (eds.) Embodied Artificial Intelligence, pp. 284–292. Springer (2004)
3. Bentley, P.J.: Fractal proteins. J. Genetic Programming and Evolvable Machines 5(1), 71–101 (2004)
4. Bentley, P.J.: Adaptive Fractal Gene Regulatory Networks for Robot Control. In: Genetic and Evolutionary Computation Conference, Seattle, USA (2004)
5. Bentley, P.J.: Methods for Improving Simulations of Biological Systems: Systemic Computation and Fractal Proteins. J. R Soc Interface (2009)
6. Bongard, J.C., Pfeifer, R.: Evolving Complete Agents Using Artificial Ontogeny. In: Hara, F., Pfeifer, R. (eds.) Morpho-functional Machines: The New Species (Designing Embodied Intelligence), pp. 237–258. Springer (2003)
7. Christensen, D.J., Bordignon, M., Schultz, U.P., Shaikh, D., Stoy, K.: Morphology Independent Learning in Modular Robots. In: Int. Symposium on Distributed Autonomous Robotic Systems, pp. 379–391 (2008)
8. Christensen, D.J., Schultz, U.P., Brandt, D., Stoy, K.: A Unified Simulator for Self-reconfigurable Robots. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2008)
9. Dellaert, F., Beer, R.: A Developmental Model for the Evolution of Complete Autonomous Agents. In: 4th Int. Conf. on Simulation of Adaptive Behavior, pp. 393–401. MIT Press, Cambridge (1996)
10. Eggenberger, P.: Evolving Morphologies of Simulated 3D Organisms Based on Differential Gene Expression. In: Husbands, P., Harvey, I. (eds.) 4th European Conf. on Artificial Life (ECAL), pp. 205–213. MIT Press, Cambridge (1997)
11. Federici, D.: Evolving a Neurocontroller through a Process of Embryogeny. In: Schaal, S., et al. (eds.) 8th Int. Conf. of Simulation and Adaptive Behavior, pp. 373–384. MIT Press (2004)
12. Federici, D., Downing, K.: Evolution and Development of a Multi-Cellular Organism: Scalability, Resilience and Neutral Complexification. J. Artificial Life 12(3), 381–409 (2006)
13. Hamann, H., Stradner, J., Schmickl, T., Crailsheim, K.: Artificial Hormone Reaction Networks: Towards Higher Evolvability in Evolutionary Multi-Modular Robotics. In: The 12th Int. Conf. on Artificial Life (2010)
14. Hornby, G.S., Pollak, B.: The Advantages of Generative Grammatical Encodings for Physical Design. In: Congress on Evolutionary Computation, pp. 600–607. IEEE Press (2001)
15. Ijspeert, A.J., Crespi, A.: Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. In: IEEE Int. Conf. on Robotics and Automation, pp. 262–268 (2007)
16. Jakobi, N.: Harnessing Morphogenesis. In: Paton, R. (ed.) Int. Conf. on Information Processing in Cells and Tissues, Liverpool, UK, pp. 29–41 (1995)
17. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic Locomotion Design and Experiments for a Modular Robotic System. IEEE/ASME Transactions on Mechatronics 10(3), 314–325 (2005)
18. Kennedy, P.J., Osborn, T.R.: A Model of Gene Expression and Regulation in an Artificial Cellular Organism. J. Complex Systems 13(1), 1–28 (2001)

19. Kuo, P.D., Leier, A., Banzhaf, W.: Evolving Dynamics in an Artificial Regulatory Network Model. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 571–580. Springer, Heidelberg (2004)
20. Murata, S., Tomita, K., Yoshida, E., Kurokawa, H., Kokaji, S.: Self-reconfigurable robot-module design and simulation. In: Proc. 6th Int. Conf. on Intelligent Autonomous Systems, Venice, Italy, pp. 911–917 (2000)
21. Ostergaard, E.H., Kassow, K., Beck, R., Lund, H.H.: Design of the Atron Lattice-Based Self-Reconfigurable Robot. J. Auton. Robots 21(2), 165–183 (2006)
22. Stoy, K., Shen, W.M., Will, P.: How to make a self-reconfigurable robot run. In: Proc. First Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2002), Bologna, Italy, pp. 813–820 (2002)
23. Transeth, A.A., Pettersen, K.Y., Liljeb, P.: A survey on snake robot modeling and locomotion. J. Robotica, 999–1015 (2009)
24. Yim, M.: Locomotion with a unit-modular reconfigurable robot. PhD thesis, Department of Mechanical Engineering, Stanford University, Stanford, CA (1994)
25. Yim, M., Shirmohammadi, B., Sastra, J., Park, M., Dugan, M., Taylor, C.J.: Towards robotic selfreassembly after explosion. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, San Diego, CA, pp. 2767–2772 (2007)
26. Zahadat, P., Katebi, S.D.: Tartarus and Fractal Gene Regulatory Networks with Input. J. Adv. Complex Sys. 11(6), 803–829 (2008)
27. Zahadat, P., Christensen, D.J., Schultz, U.P., Katebi, S.D., Stoy, K.: Fractal gene regulatory networks for robust locomotion control of modular robots, In: The 11th Int. Conf. on Simulation of Adaptive Behavior (2010)