A Morphogenetic Approach to Self-Reconfigurable Modular Robots using a Hybrid Hierarchical Gene Regulatory Network

Yan Meng[1], Yuyang Zhang[1] and Yaochu Jin[2]

[1]Department of Electrical and Computer Engineering
Stevens Institute of Technology, Hoboken, NJ 07030, USA
[2]Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, UK
yan.meng@stevens.edu, yzhang14@stevens.edu, yaochu.jin@surrey.ac.uk

Abstract

In this paper, we present a morphogenetic approach to self-reconfiguration of a lattice-based simulated modular robot, CrossCube, under dynamic environments. A hybrid hierarchical controller inspired by the embryonic development of multi-cellular organisms is proposed to form different patterns for modular robots to adapt to environmental changes. The first layer is a rule-based controller to generate a number of appropriate target patterns (i.e. configurations) for various environments. The second layer is a gene regulatory network (GRN) based controller to coordinate the modules of CrossCube to transform from its current pattern to the target pattern. This hybrid hierarchical control framework is distributed in the sense that each module makes its own decisions based on its local perception. The global behavior of modular robots emerges from the local interactions with the environment and between the modules. The simulation results demonstrate that the proposed system is efficient and robust in adaptively reconfiguring modular robots to adapt to the changing environment.

Introduction

Self-reconfigurable modular robots are autonomous robots with a variable morphology, where they are able to deliberately change their own shapes by reorganizing the connectivity of their modules to adapt to new environments, perform new tasks, or recover from damages. Each module is an independent unit that is able to connect it to or disconnect it from other units to form various structures/patterns dynamically. Compared with conventional robotic systems, self-reconfigurable robots are potentially more robust and more adaptive under dynamic environments.

Modular robots can be generally classified into two groups according to their geometric arrangements of the modules: the chain/tree-based architectures [16] [19] [21] [22] and the lattice-based architectures [5] [7] [8] [10] [13] [14] [17] [24] [25]. In the chain/tree-based architectures, the modules are connected in a topology of a chain or a tree, where the motion controls of the modules are executed sequentially. It is relatively easier to design and implement this kind of architectures. In the lattice-based architecture, robot modules are usually arranged and connected in 3D patterns, such as a cubical or hexagonal grid, and the motion control of modules are carried out in parallel. Therefore, compared to the chain/tree-based architectures, the lattice-based

architectures are more flexible and efficient to form complex structures although the design and implementation of this kind of architectures are more difficult. From this point of view, lattice-based modular robots are more suitable for dynamic environments. However, most available lattice-based modular robotic systems only have basic locomotion controllers to reconfigure the modular robots to a few predefined patterns by following predefined sequences or rules which have been optimized by human operators as a global controller. These predefined rules or sequences cannot predict all the possible situations that may occur for modular robots under dynamic environments. Although self-reconfiguration is believed to be the most important feature of modular robots, the ability to adapt their configuration autonomously under environmental changes remains to be demonstrated.

Generally, centralized high-level controllers for lattice-based modular robots are vulnerable to system failures or malfunctions of robot modules. On the other side, decentralized controllers are more robust and flexible under dynamic and uncertain environments. However, the main challenge for distributed systems is that it is difficult to predict the emerging behaviors only from local interactions of individual agents; neither is it easy to design rules for local interactions to generate desired global behaviors. Therefore, the major challenge in developing a decentralized controller for self-reconfigurable modular robots is how to coordinate local behaviors of multiple modules to achieve the desired global patterns to adapt to current environmental situations. To this end, we turn our attention to biological systems. Biological systems, from macroscopic swarm systems of social insects to microscopic cellular systems, can generate robust and complex emerging behaviors through relatively simple local interactions subject to various kinds of uncertainties [9]. We are more interested in the morphogenesis procedure in multi-cellular organisms. During the morphogenesis, genes in each cell are expressed, resulting in various cellular functions. The expression of the genes is regulated by their own protein products as well as proteins produced by other genes in the same cell or neighboring cells through intracellular and intercellular diffusion, forming a gene regulatory network that can be described by a set of coupled ordinary differential equations.

The connection between reconfigurable modular robots and multi-cellular organisms appears straightforward. Each unit in modular robots can be seen as a cell, and there are similarities

in control, communication and physical interactions between cells in multi-cellular organisms and modules in modular robots. For example, control in both modular robots and multi-cellular organisms are decentralized. In addition, global behaviors of both modular robots and multi-cellular organisms emerge through local interactions of the units, which include mechanic, magnetic and electronic mechanisms in modular robots, and chemical diffusion and cellular physical interactions such as adhesion in multi-cellular organisms. Therefore, it is a natural idea to develop control algorithms for self-reconfigurable modular robots using biological morphogenetic mechanisms.

Inspired by the embryonic development of multi-cellular organisms [24], in this paper, we propose a morphogenetic approach to self-reconfiguration of a lattice-based simulated modular robot, CrossCube. Basically, each module of CrossCube has a flexible single cubic shape like Molecube [25] [15], which does not require much free space for modules to move around, similar to the mechanics of SUPERBOT [14] [4] and MTRAN [10] [11] [20]. In the high-level controller, a two-layer morphogenetic architecture is proposed. Layer 1 is pattern generation layer, which is a rule-based controller to generate appropriate patterns represented by look-up tables. Layer 2 is a gene regulatory network (GRN) based controller to reconfigure modules automatically to the target patterns generated from layer 1.

Recently, Stoy proposed cellular automata to control reconfiguration [17]. Both our method and Stoy's method used cellular mechanism to reconfigure the modular robots. However, there are some major differences between our work and his work. Our method is two-layer hierarchical method. In [17], one-layer approach was proposed which corresponds to layer 2 in our model. Layer 2 of our model uses priorities to assign the importance of the positions of the target pattern, which help to improve the balance of target formation. In addition, our proposed method can solve the dead-lock situations of the modules while [17] cannot.

The major contributions of this paper are listed as follows. (1) The mechanics of CrossCube enables highly flexible locomotion compared to that in existing lattice-based modular robots. (2) A hybrid hierarchical morphogenetic controller is proposed, which is a decentralized approach where each module makes its own decisions based on its local perceptions on the environment and interactions with its immediate neighboring modules. (3) The modular robots can autonomously choose an appropriate pattern based on the current environment and then automatically self-reconfigure itself to the target pattern. (4) The proposed system is very robust to system failures.

The rest of the paper is organized as follows. The basic mechanics and locomotion design of CrossCube are described at first, followed by a brief introduction to biological morphogenesis. Then the proposed morphogenetic approach to self-organization of modular robots is presented. Various simulation results on evaluating the proposed morphogenetic approach to modular robots under dynamic environments are described. The paper concludes with a short summary of the current results and future work.

CrossCube – A Simulated Modular Robot

CrossCube is a simulated modular robot we developed in a robot simulator using a real time physics engine PhysX. The detailed information on the simulator will be discussed in the simulation section. CrossCube adopts a lattice-based cube design. Each module is a cubical structure having its own computing and communication resource and actuation capability. Like all modular robots, the connection part of the modules can easily be attached to or detached from other modules. Each module can perceive its local environment and communicate with its neighboring modules using on-board sensors.

Each CrossCube module consists of a core and a shell as shown in Fig. 1(a). The core is a cube with six universal joints. Their default heading directions are bottom, up, right, left, front, and back, respectively. Each joint can attach to or detach from the joints of its neighbor modules. The axis of each joint can actively rotate, extend, and return to its default direction and length.

The cross-concaves on each side of the shell restrict the movement trajectory of the joints, as show in Fig. 1(a). The borders of each module can actively be locked or unlocked with the borders of other modules, as shown in Fig. 1(b).

Basic motions of modules in CrossCube include rotation, climbing and parallel motion. Fig. 1(c) illustrates a rotation movement of two modules. Parallel motion means that a module moves to a next position which is parallel to its current position. During a parallel motion, a module moves from its current position to a parallel position Climbing motion means that a module moves to a diagonal neighboring position. Parallel motion and climbing motion allow a module of CrossCube to move to any position within the modular robot as long as the modules are connected. Since the major focus of this paper is the self-reconfiguration control algorithm, the detailed mechanical design of CrossCube is skipped here.
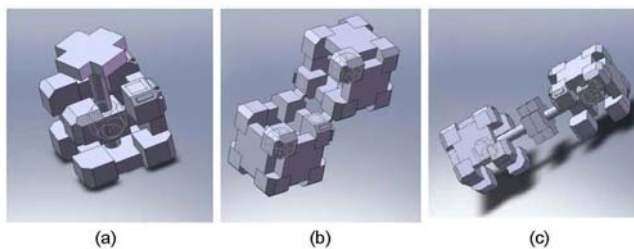

Figure 1: Mechanical demonstration of CrossCube. (a) The joints; (b) The locks on the boundaries of the modules. (c) Rotation and extension of the joints of the modules.

Morphogenetic Approach

Multi-Cellular Morphogenesis

Multi-cellular morphogenesis is under the control of gene regulatory networks. When a gene is expressed, information stored in the genome is transcribed into mRNA and then translated into proteins. Some of these proteins are transcription factors that can regulate the expression of their own or other genes, thus resulting in a complex network of

interacting genes termed as a gene regulatory network (GRN). To understand the emergent morphology resulting from the interactions of genes in a regulatory network, reconstruction of gene regulatory pathways using a computational model has become popular in systems biology [1]. A large number of computational models for GRNs have been suggested [2], [3], which can largely be divided into discrete models, such as random Boolean networks and Markovian models, and continuous models, such as ordinary differential equations and partial differential equations. Sometimes, GRN models also distinguish themselves as deterministic models and stochastic models according to their ability to describe stochasticity in gene expression. Note that in artificial life, a few high-level abstraction models have also been used for modeling development, such as the L-systems [12] and grammar trees [6].

The Hierarchical Framework

The metaphor between reconfigurable modular robots and multi-cellular organisms is straightforward. We can treat each module in modular robots as a single cell. And the similarities in control, communication and physical interactions between cells in multi-cellular organisms and modules in modular robots are obvious. For example, the control in both modular robots and multi-cellular organisms in decentralized. Furthermore, the global behaviors of both modular robots and multi-cellular organisms emerge through local interactions of the units, which include mechanic, magnetic and electronic mechanisms in modular robots, and chemical diffusion and cellular physical interactions such as adhesion in multi-cellular organisms.
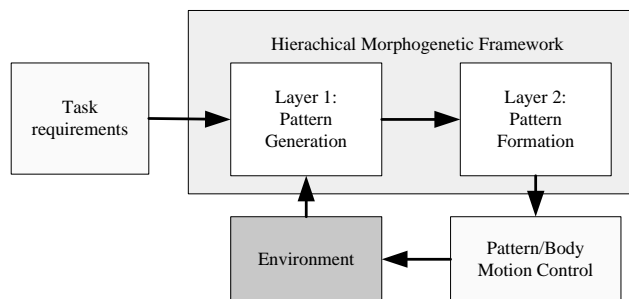

Figure 2: The block diagram of the hierarchical framework for the morphogenetic approach.

Based on this metaphor, a hybrid hierarchical morphogenetic approach is developed in this paper for self-reconfiguration of modular robots. First, the target pattern (i.e. final configuration) that a modular robot needs to form has to be generated automatically based on the current environments and mission at hand using some heuristic rules, which is the layer 1 controller of the hierarchical framework. Then, the modules in a modular robot need to self-organize themselves to form the target pattern generated by layer 1 using a GRN-based controller, which is the layer 2 controller. Fig. 2 shows the block diagram of this hierarchical GRN framework. Each unit of the modular robots contains a chromosome consisting of several genes that can produce different proteins. The local communications between the modules can be setup by diffusing the proteins into neighboring modules. The

concentration of the diffused proteins decays over time and distance.

Layer 1: Pattern Generation

Adaptation to environmental changes is of paramount importance in reconfigurable modular robots. A mechanism is needed to define and modify the target configuration of the modular robot adaptively. Adaptation of the global configuration of the modular robot, i.e., change in morphogen values, can be triggered by local sensory feedback. For such tasks, it is assumed that each module is equipped with a sensor to detect the distance(s) between the module and obstacle(s) in the environment. Once a module receives such sensory feedback, this information will be passed on to its neighbors through local communication. In this way, a global change in configuration can be achieved.

The target pattern of the modular robot is defined by morphogen values of each grid. Grids are discretized from the space in which the modular robot is located. Each grid has the same size of with a robot module. The morphogen value can be either positive or negative. A positive morphogen value means that the grid should be occupied by a module, while a negative gradient suggests that the module in the grid, if any, should be removed. A higher value of morphogen value indicates a higher priority for the grid to be filled by a module.

For the sake of simplicity, a number of basic configurations for different environments can be represented in terms of a look-up-table for a given mission, for instance locomotion. An example of defining the configuration of a vehicle is provided in Table 1. In the table, x, y, and z are 3D coordinates of grid positions, MG denotes morphogen level and PID stands for position identification. Additionally, we define some joints' behaviors to enable the vehicle to move once the configuration is completed. Joints can be identified by its PID and RD means joint rotate direction.

Then the question is how to generate the look-up-table and decide the morphogen value for each position of a pattern under current environmental situations. A rule-based controller is developed for this purpose. In this paper, we only focus on the generation of some specific vehicle patterns to explain the basic ideas. We will investigate a more generic controller for different patterns in the future.

It is assumed that initially all robot modules know the heading direction of the vehicle pattern. When a robot needs to traverse a path whose width is narrower than that of the robot, the width of the front row will be first adapted to fit in the path. The remaining rows of the vehicle will be adapted row by row in a decentralized manner through local communication. The basic rules for this procedure can be summarized as follows:

- Rule 1: Once a module in the front row detects obstacle(s), it passes this information through local communication to its neighboring modules until all the modules are reset to the unstable state for initialization. Refer to the next section for a definition of different states of the robot modules.
- Rule 2: If some of the modules in the first row detect an obstacle, they will estimate whether the robot need to reconfigure itself to avoid the obstacle. If yes, these modules will estimate how many modules need to be removed and this information is passed to other modules in

the same row through local communication. Therefore, the mophogen gradients of these need-to-remove positions are set up as negative values while others as positive values. As a result, those positions with positive mophogen values are head of the new vehicle pattern.

- Rule3: After the GRN-based pattern formation controller finishes the reorganization of the modules in one row, the states of these modules are set to be 'stable'. If a row of a vehicle pattern is filled in by stable modules, these modules can set the positive morphogen values for the position in the next row. One exception is that if the module is used as a wheel for the vehicle pattern, the morphogen value of its next position should be set as negative because two neighboring wheel modules causes fault pattern.
- Rule4: The pattern generation procedure stops when all the modules change to the stable state.

Layer 2: Pattern Formation

By setting any single module as the origin, all other modules can figure out their relative positions to this origin easily through local communications. Based on the relative positions and the information on the target pattern, each module can produce different types of proteins to attract other modules to fill in its neighboring positions with positive morphogen values, or repel its neighbor modules from positions with negative morphgen values.

Finite States of Modules

The attraction and repellent behaviors of the modules are regulated by a GRN-based controller, which can adaptively set the state of the modules to one of the following five states, namely, 'stable', 'unstable', 'attracting', 'repelling', and 'repelled'. The transition relationships between the five states of modules are given in Figure 3.
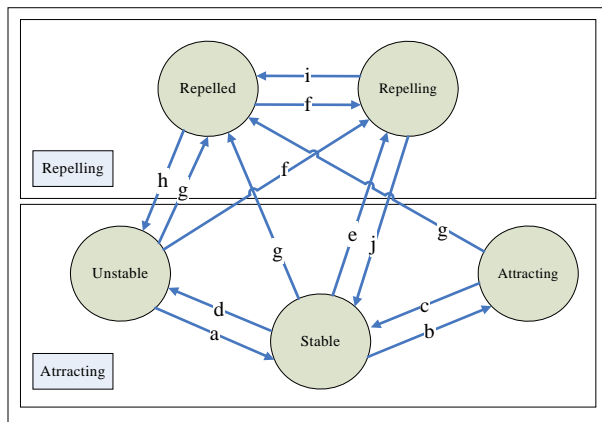

Figure 3: State transition of each module in CrossCube.

The "stable" state means the final state of the module. The "attracting" state means the module can attract other modules to fill in some of its neighboring positions. The "unstable" state means the module can respond to attractions. The "repelling" state means the module can repel specific neighboring modules away. The "repelled" state means that

the module responds to repelling requests and move away from the current position.

When an 'unstable' module arrives at the destination position (grid), it changes its state to "stable" (arrow *a* in Figure 3). A 'stable' module can change its state to 'attracting' (arrow *b* in Figure 3) if it has neighboring positions with a positive morphogen value. When those neighboring positions are occupied by modules, the 'attracting' module returns to the 'stable' state (arrow *c* in Figure 3). A 'stable' module may also give up its current position so that it can fill in some more important positions in the pattern (with a high positive gradient) by turning its state to 'unstable' (arrow *d* in Fig. 2).

When the 'repelled' module moves away from its current position it switches its state to 'unstable' (arrow *h* in Figure 3). A module can be triggered to be 'repelling' state under two situations. First one is when a 'stable' module finds out that some of its neighboring modules are located in the positions with negative morphogen value, it changes its state to 'repelling' (arrow *e* in Figure 3) and switches the state of those neighbors to be 'repelled' (arrow *g* in Figure 3). When all the 'repelled' modules have left, the 'repelling' module returns to the 'stable' state (arrow *j* in Figure 3). The second situation is a deadlock situation. A deadlock happens when a module is blocked by its neighboring modules. To resolve this deadlock, the blocked module switches its state to be 'repelling' (arrow *f* in Figure 3), and trying to change the state of all its neighbors to be 'repelled' (arrow *g* in Figure 3). This removes some of its neighboring modules to make room for the blocked module to move away. Then the 'repelling' module turns back to the 'repelled' state (arrow *i* in Figure 3).

The state transitions are controlled by a GRN-based model having two gene-protein pairs: an attracting gene-protein pair (g_A, p_A) and a repelling gene-protein pair (g_p, p_p). We assume that the repellent states always have a higher priority than the attracting states. As a result, all the states triggered by the attracting behaviors can be overwritten by the states triggered by the repelling behaviors. The reason for this assumption is that the positions with a repelling (negative) morphogen value should be kept empty as long as migration modules are still in need during reconfiguration.

Gene-Protein Pair for Attraction

The attracting gene-protein pair (g_A, p_A) is used to control the transitions between 'attracting', 'stable' and 'unstable' states in Figure 3. Basically the expression level of g_A affects the state as shown in (1). And protein p_A will regulate g_A's expression level.

$$\text{state} = \begin{cases} \text{'unstable'} & \text{if } g_A < G_{A\_L} \\ \text{'stable'} & \text{if } G_{A\_L} < g_A < G_{A\_H} \\ \text{'attracting'} & \text{if } g_A > G_{A\_H} \end{cases} \quad (1)$$

where G_{A_L} is a negative threshold and G_{A_H} is a positive threshold.

At the initial stage of pattern formation, all modules are set as 'unstable'. After they are initialized with the target pattern and the relative position information to the origin, modules that are located in the grids with a positive morphogen value become 'stable'. A new 'stable' module initializes the gene expression level of its attracting gene g_A to zero.

Each 'stable' module generates attracting protein p_A for all of the empty neighboring grids having a positive morphogen value. The local generated p_A and received p_A from other modules will regulate the expression level of g_A. When g_A is high enough to trigger the module to be 'attracting', the local generated p_A will be diffused to other modules. During diffusion, the concentration of p_A are weaken by a fix rate each time when it enters a cell. Here, p_A is defined as

$$p_A^{ij} = \{AP^{ij}, M_A^{ij}\} \tag{2}$$

where p_A^{ij} is the attracting protein generated by i-th module for its j-th neighbor position. AP^{ij} is the position, and M_A^{ij} is the concentration of the protein p_A^{ij}, which is discounted from the morphogen value of AP^{ij} defined by layer 1 of the control framework.

The dynamics of regulation can be described by the following GRN model:

$$\frac{dg_A^i(t)}{dt} = -k_1 \cdot g_A^i(t) + k_2 \cdot \sum_j p_A^{ij} - k_3 \cdot \sum p_{A\_received} \tag{3}$$

where $g_A^i(t)$ is g_A's concentration of the i-th module. The first term indicates that $g_A^i(t)$ will decay over time. The second term represents the sum of all locally generated p_A by grid i. The more proteins a module (which is associated with grid i) generates for its empty neighboring grid, the higher the g_A expression level this grid will be, which means it will have better chance to change its state from "stable" to "attracting". Meanwhile, $g_A(t)$ will decrease if it receives p_A from other modules. The module may turn to 'unstable' if outer attraction is strong enough. $k_1, k_2,$ and k_3 are constant coefficients. Unstable modules choose the attracting position with the highest P_A from all the received attracting proteins to fill in, and move to the destination by following morphogen gradient. Once a module reaches its destination, it will become stable.

To summarize, the gene-protein pair (g_A, p_A) can regulate each other according to the GRN model described in Eqns. (1) and (3). More specifically, p_A can regulate g_A through Eqn. (3). Meanwhile, p_A can diffuse only if g_A is greater than G_{A_H} based on Eqn. (1).

Gene-Protein Pair for Repelling
The 'repelling' states are controlled by the repelling gene-protein pair (g_p, p_p). The repelling modules produce p_p, which is defined as

$$p_P^{ij} = \{RP^{ij}, M_P^{ij}\} \tag{4}$$

where p_P^{ij} is the repellent protein generated by i-th module for its j-th neighbor. RP^{ij} is the j-th repellent grid around i-th module, and M_P^{ij} is the concentration of the protein p_P^{ij}, which equals to a predefined positive constant. Each module has repelling gene whose concentration affects whether the module should change to 'repelled' state, that is, to respond to a 'repelling' module. The gene expression level of g_P is initialized as 0 and can be regulated by p_P through Eqn. (5)

$$\frac{dg_P^i(t)}{dt} = -k_4 \cdot g_P^i(t) - k_5 \cdot \sum p_{P\_rec} \tag{5}$$

$$\text{state} = \text{repelled when } g_P < -MG^i$$

where $g_p(t)$ is the gene expression level of the repellent gene at time t. p_{P_rec} is the concentration of the received repellent protein. MG^i is the morphogen value of the current position. k_4 and k_5 are constant coefficients. The first item denotes $g_p(t)$ will decays to zero along time. The second term indicates that when a module receives p_P, the concentration of g_P is reduced. Obviously modules with a lower morphogen value are more likely to be repelled.

To summarize, p_P can regulate g_P through Equation (5). g_P can produce p_P under the condition that g_P is below MG^i and the module is blocked.

Simulation Results

To evaluate the efficiency and robustness of the morphogenetic approach to the self-reconfiguration of CrossCube, several case studies have been conducted in a robot simulator, as shown in Figure 4. This simulator is used to simulate the behaviors and interaction of CrossCube with a physical world using C++ and the PhysX engine from nVidia (http://en.wikipedia.org/wiki/PhysX). In the following experiments, the system parameters are setup as follows: $k_1 = 0.7$, $k_2 = 1$, $k_3 = 1$, $k_4 = 0.5$, $k_5 = 2$, $G_{A\_L} = -1$, $G_{A\_H} = 1$, $G_{P\_L} = -2$, $C_P^{ij} = 0.7$. Protein concentration decays to 80% of its previous level when it diffuses into a neighbor module.

Case Study 1: Pattern Formation
To evaluate the performance of the GRN-based controller for pattern formation layer, first, we can predefine a fixed target pattern using a look-up table. For example a vehicle pattern, can be defined as Table 1.

Positions (X, Y, Z, MG, PID)		Joints (PID1, PID2, RD)
(0, 0, 0, 10, 0)	(1, 0, 3, 10, 10)	(0, 1, 0)
(1, 0, 0, 10, 1)	(2, 0, 3, 10, 11)	(2, 3, 1)
(2, 0, 0, 10,2)	(0, 0, 4, 10,12)	(6, 7, 0)
(3, 0, 0, 10, 3)	(1, 0, 4, 10, 13)	(8, 9, 1)
(1, 0, 1, 10, 4)	(2, 0, 4, 10, 14)	(12, 13, 0)
(2, 0, 1, 10, 5)	(3, 0, 4, 10, 15)	(14, 15, 1)
(0, 0, 2, 10, 6)	(0, 0, 1, -1, 16)	
(1, 0, 2, 10, 7)	(3, 0, 1, -1, 17)	
(2, 0, 2, 10, 8)	(0, 0, 3, -1, 18)	
(3, 0, 2, 10, 9)	(3, 0, 3, -1, 19)	

Table 1: Definition of a vehicle pattern for case study 1. In the table, x, y, and z are 3D coordinates of grid positions, MG denotes morphogen level and PID stands for position identification.

Based on this predefined target pattern, the modules of CrossCube modules need to autonomously configure themselves to form the target pattern using the GRN-based controller in layer 2. A set of snapshots of this pattern

formation procedure in the experiment is depicted in Figure 4. From Figure 4, we can see that the CrossCube can automatically form a given target pattern through self-reconfiguration using the proposed GRN-based controller.
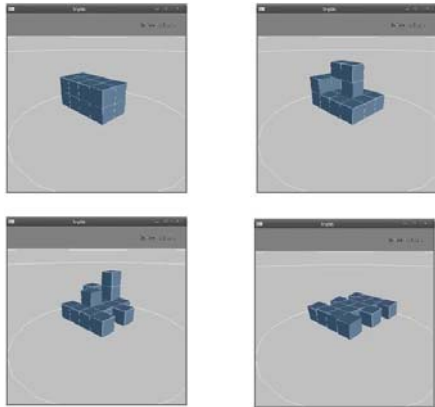

Figure 4: Autonomous reconfiguration of a CrossCube from a rectangle to a vehicle using the GRN-based model.
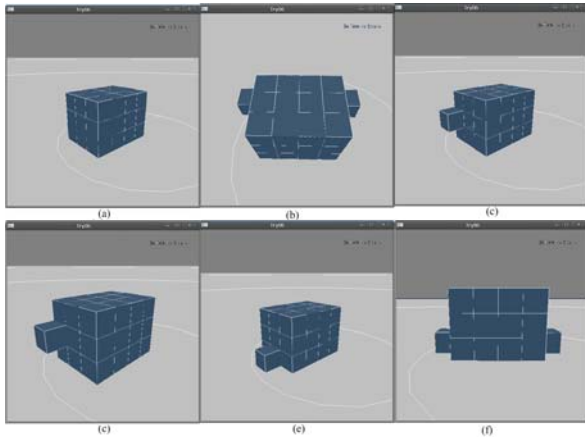

Figure 5: A set of snapshots for the simulation using the repelling feature of the GRN-based controller to resolve a deadlock problem. (a) The original pattern of the robot. (b)(c) Two modules are repelled by the central modules. (d)(e) The central modules move away from blocked positions. (f) The target pattern is finished.

Case Study 2: Resolving Deadlock

In this case study, a deadlock problem is resolved using the repelling function of the GRN-based controller in layer 2. Robot modules are initialized in a 4x3x3 solid cube, starting at (0, 0, 0) and ending at (3, 2, 2). The target pattern is predefined in Table 2 which is a center-empty box plus two additional modules at sides. To build the pattern, the modules in the center of the solid cube should move out the module that is blocked by the modules on surface. Then the GRN-based controller of layer 2 is conducted to solve the deadlock problem to form the target pattern. Figure 5 shows the successful procedure of solving this deadlock problem using this morphogenetic approach on CrossCube simulator. It is

shown that the modules with lower morphogen value are repelled which is consistent with our design.

Positions (x, y, z)	Morphogen value
(-1, 0, 1), (4, 0, 1), (0, 1, 1), (3, 1, 1)	2
(1, 1, 1), (2, 1, 1)	-10
Other positions	10

Table 2 Definition of a vehicle pattern for case study 2

Case Study 3: Self-Repairing

One important feature of a reconfigurable modular robot is being able to dynamically self-repair itself from the malfunctions of modules or damaged modules. For example, if some of the modules are damaged, the remaining modules will release new attracting proteins to repel those damaged modules and attract existing modules in the positions with a low morphogen value to fill in the positions of the damaged modules. In other words, modules that are located in less important positions of the target pattern will automatically migrate to the positions originally occupied by the damaged modules with a higher morphogen value. To evaluate the self-repairing performance of the GRN-based control in layer 2, another experiment is conducted here. First, the look-up table for the target pattern (i.e., a vehicle patter here) is given in Table 3 as a fixed predefined layer 2. The bottom modules (y equals to 0) are functional modules in the vehicle pattern. The top modules (y equals to 1) are backup modules, which are used to repair the malfunctioned parts of the vehicle pattern. Therefore, the backup modules have a lower morphogen value than that of the functional modules.

When the vehicle is moving, an "explosion" occurs and some functional modules are blown away. The backup modules then automatically move to fill in the damaged modules. Figure 6 shows a snapshot of this self-repairing procedure using the proposed hierarchical framework on CrossCube modules. This experiment demonstrates that the proposed approach is efficient for self-repair of a modular robot in the presence of some failed modules.

Positions (X, Y, Z, MG, PID)		Joints (PID1, PID2, RD)
(0, 0, 0, 10, 0)	(0, 0, 4, 10,12)	(0, 1, 0)
(1, 0, 0, 10, 1)	(1, 0, 4, 10, 13)	(2, 3, 1)
(2, 0, 0, 10,2)	(2, 0, 4, 10, 14)	(6, 7, 0)
(3, 0, 0, 10, 3)	(3, 0, 4, 10, 15)	(8, 9, 1)
(1, 0, 1, 10, 4)	(0, 0, 1, -1, 16)	(12, 13, 0)
(2, 0, 1, 10, 5)	(3, 0, 1, -1, 17)	(14, 15, 1)
(0, 0, 2, 10, 6)	(0, 0, 3, -1, 18)	
(1, 0, 2, 10, 7)	(3, 0, 3, -1, 19)	
(2, 0, 2, 10, 8)	(1, 1, 1, 1, 20)	
(3, 0, 2, 10, 9)	(2, 1, 1, 1, 21)	
(1, 0, 3, 10, 10)	(1, 1, 2, 1, 22)	
(2, 0, 3, 10, 11)	(2, 1, 2, 1, 23)	

Table 3 Definition of a vehicle pattern for case study 3

Pattern Adaptation in a Changing Environment

To verify the efficiency and robustness of the rule-based controller for pattern generation, a transformable vehicle is

developed. During the pattern generation process, the positive morphogen value is set as 10 and the negative morphogen value is -10.

A set of snapshots showing the adaptation of the vehicle pattern to environmental changes is provided in Figure 7. First the pattern generation controller generates a vehicle pattern based on the width of path it needs to traverse using the rule-based method. As the vehicle is moving forward, a narrower path is detected. Consequently, a new vehicle pattern that can fit in this narrower tunnel is generated. Then steps are detected in front of the robot, new target patterns are dynamically generated to allow the robots to climb the steps, and eventually a new vehicle pattern is generated to continue its locomotion task after finishing the climbing. During this procedure, the GRN-based controller for pattern formation layer would automatically reconfigure the modules to form the new target patterns.
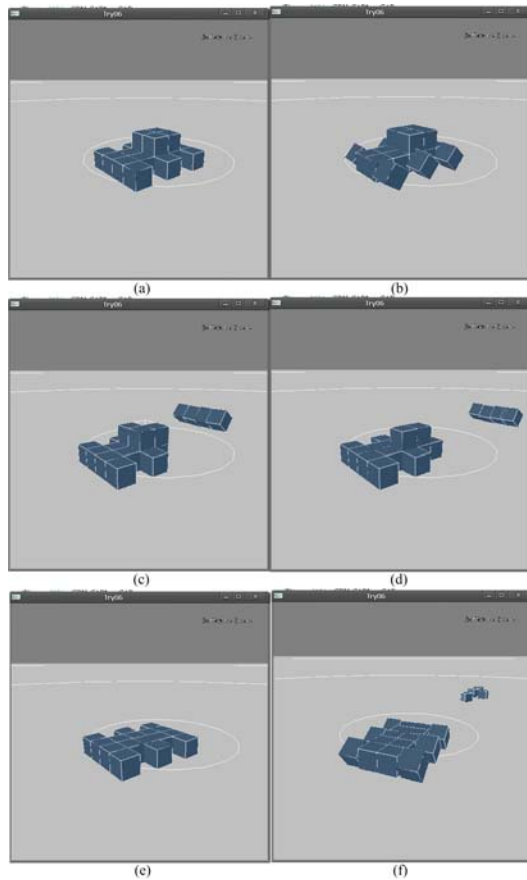

Figure 6: A set of snapshots of the self-repairing of CrossCube using the GRN-based controller. (a) A vehicle pattern is formed. (b) The vehicle pattern moves forward. (c) Some modules are blown off when the explosion happens. (d) The failed part is filled up by the backup modules. (e) The vehicle is repaired. (f) The repaired vehicle continues moving.

Conclusion and Future Work

In this paper, we presented a hybrid hierarchical approach to self-reconfiguration of a simulated modular robot, CrossCube, which is inspired by multi-cellular morphogenesis. First layer defines the desired configuration of the modular robots while the other layer organizes the modules autonomously to achieve the desired configuration. Such a hierarchical structure makes it possible to separate the control mechanisms for defining a target configuration from those for realizing it, similar to biological gene regulatory networks. In response to the environment changes, the layer for defining the robot configuration is able to adapt the target configuration, based on which the second layer can re-organize the modules autonomously to realize the target configuration.

The current system is only based on simulated modular robots with considerations of physical constraints. In the future, we will develop the real modular robots based on the current mechanical design. Furthermore, since the current design of the first layer is a heuristic rule-based method, it has some limitations to generate various patterns for dynamic environments, only some simply patterns are possible. In the future, we will investigate a more general approach for the design of layer 1 so that more general patterns can be automatically generated to adapt to various dynamic environmental changes.
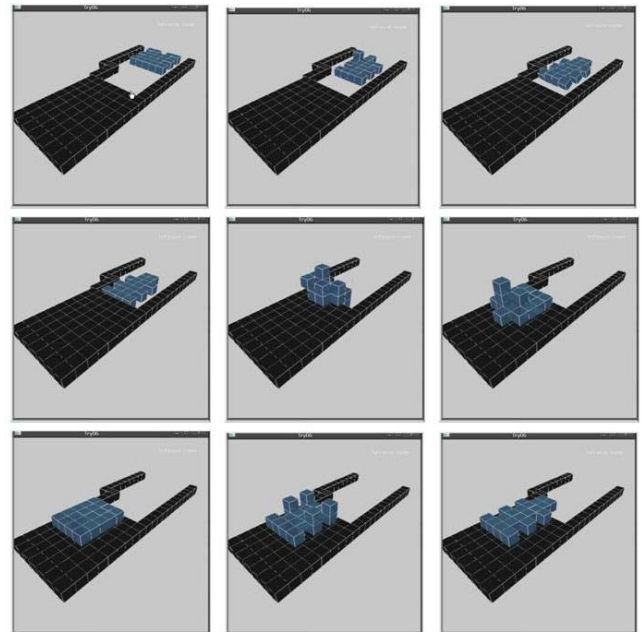

Figure 7. A set of snapshots demonstrating a series of reconfigurable processes during locomotion and climbing. The robot first adapted its width to the narrow path, then changed its configuration for climbing up a step, and finally reconfigured itself into a vehicle again to move forward.

Acknowledgements

References

[1] Alon, U. (2007). Network motifs: theory and experimental approaches, *Nature Review Genetics*, vol. 8, pp. 450–461.

[2] DeJong, H. (2002). Modeling and simulation of genetic regulatory systems: A literature review, *Journal of Computational Biology*, vol. 9, pp. 67–103.

[3] Endy D., and Brent, R. (2001). Modeling cellular behavior, *Nature*, vol. 409,pp. 391–395.

[4] Everist, J., Hou, F., and Shen, W. (2006). Transformation of Control in Congruent Self-Reconfigurable Robot Topologies, in Proc. of *International Conference on Intelligent Robots and Systems*.

[5] Gilpin, K., Kotay, K. and Rus, D. (2007). Miche: Modular Shape Formation by Self-Dissasembly, in Proc. of *IEEE International Conference on Robotics and Automation*.

[6] Gruau, F. (1993). Genetic synthesis of modular neural networks, in International Conferences on Neural Networks. *Morgan Kaufmann*, pp. 318–325.

[7] Jorgensen, M. W., Ostergaard, E. H., and Lund, H. H. (2004). Modular ATRON: Moduels for a self-reconfigurable robot, in Proc. of *IEEE International Conference on Intelligent Robots and Systems*.

[8] Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., and Kokaji, S. (2004). Distributed adaptive locomotion by a modular robotic system, M-TRAN II, in Proc. of *IEEE International Conference on Intelligent Robots and Systems*.

[9] Kelly, K. (1994). Out of Control – The New Biology of machines, *Social Systems and Economic World. Basic Books.*

[10] Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., and Murata, S. (2008). Distributed self-reconfiguration of M-TRAN III modular, *The International Journal of Robotics Research*, 27(3-4):373–386.

[11] Kurokawa, H., Tomita, K., Kamimura, A., Yoshida, E., Kokaji, S., and Murata, S. (2005). Distributed Self-reconfiguration Control of Modular Robot M-TRAN, in Proc. of *International Conference on Mechatronics and Automation.*

[12] Lindenmayer, A. (1968). Mathematical models for cellular interaction indevelopmental. Parts I and II, *Journal of Theoretical Biology*, vol. 18, pp. 280–315.

[13] Murata, S., Yoshida, E., Kurokawa, H., Tomita, K., and Kokaji, S. (2001). Self-repairing mechanical systems, *Autonomous Robots*, vol. 10, pp. 7–21.

[14] Salemi, B., Moll, M., and Shen, W. (2006). SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System, in Proc. of *IEEE International Conference on Intelligent Robots and Systems*.

[15] Moll, M., Will, P., Krivokon, M., and Shen, W. (2006). Distributed Control of the Center of Mass of a Modular Robot, in Proc. of *IEEE International Conference on Intelligent Robots and Systems.*

[16] Shen, W., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., and Venkatesh, J. (2006). Multimode locomotion for reconfigurable robots, *Autonomous Robots*, vol. 20, no. 2, pp. 165-177.

[17] Stoy, K. (2006), Using cellular automata and gradients to control self-reconfiguration, *Robotics and Autonomous Systems*, vol. 54, pp.135-141.

[18] Unsal, C., Killiccote, H., and Kholsa, P.K. (2001). A Modular Self-Reconfigurable Bipartite Robotic System: Implementation and Motion Planning, *Autonomous Robots*, vol. 10, pp.23–40.

[19] Yim, M., Eldershaw, C., Zhang, Y., and Duff, D. G. (2004). Limbless conforming gaits with modular robots, in Proc. of *International Symposium on Experimental Robotics.*

[20] Yoshida, E., Kurokawa, H., Kamimura, A., Tomita, K., Kokaji, A., and Murata, S. (2004). Planning Behaviors of a Modular Robot: an Approach Applying a Randomized Planner to Coherent Structure, in Proc. *IEEE International Conference on Intelligent Robots and Systems.*

[21] Yu, C.H., Haller, K., Ingber, D., and Nagpal, R. (2009). Morpho: A selfdeformable modular robot inspired by cellular structure, in Proc. of *International Conference on Robotics and Automation*.

[22] Yu, C., and Nagpal, R. (2009). Self-Adapting Modular Robotics: A Generalized Distributed Consensus Framework, in Proc. of *International Conference on Robotics and Automation.*

[23] White, P., Zykov, V., Bongard, J., and Lipson, H. (2006). Three Dimensional Stochastic Reconfiguration of Modular Robots, in Proc. of *Robotics: Science and Systems Conference*, pp. 161–168.

[24] Wolpert. L. (2002). *Principles of Development.* Oxford University Press.

[25] Zykov, V., Chan, A., and Lipson, H. (2007). Molecubes: An Open-Source Modular Robotics Kit, in Proc. of *International Conference on Robotics and Automation.*