To appear in "IEEE Computational Intelligence Magazine"

Autonomous Self-Reconfiguration of Modular Robots by Evolving a Hierarchical Mechanochemical Model

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

[1]Department of Electrical and Computer Engineering
Stevens Institute of Technology, New Jersey, USA
[2]Honda Research Institute Europe, Offenbach, Germany
yan.meng@stevens.edu, yzhang14@stevens.edu, yaochu.jin@honda-ri-de

Abstract— **In this paper, we present a two-layer hierarchical mechanochemical model for self-reconfiguration of modular robots under changing environments. The model, which is inspired by the embryonic development of multi-cellular organisms and chemical morphogenesis, can autonomously generate and form different patterns for modular robots to adapt to environmental changes. Layer 1 of the model utilizes a virtual-cell based mechanochemical model to generate appropriate target patterns (i.e., chemical blueprints) for current environment. Layer 2 is a gene regulatory network (GRN) based controller to coordinate the modules of modular robots for physically realizing the chemical target pattern defined by the first layer. This hierarchical mechanochemical framework is a distributed system in that each module makes decisions based on its local perceptions. To optimize pattern design of modular robots, the covariance matrix adaptation evolution strategy (CMA-ES) is adopted to evolve the pattern parameters of the mechanochemical model. Simulation results demonstrate that the proposed system is effective and robust in autonomously reconfiguring modular robots to adapt to environmental changes.**

I. 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 detach it from other units to form various structures / patterns. Compared with conventional robotic systems, self-reconfigurable robots are potentially more robust and more adaptable under changing environments.

Modular robots can be generally classified into two groups according to their geometric arrangements of the modules: the chain / tree-based architectures [25] [31] [33] [34] and the lattice-based architectures [5] [10] [11] [13] [17] [23] [28] [29] [35]. 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, modules of a robot 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 suited for dynamic environments. However, most existing lattice-based modular robotic systems can only reconfigure them into a few predefined patterns by following a set of user-defined rules. Although self-reconfiguration is believed to be the most important feature of modular robots, the ability to adapt their configuration autonomously under environmental changes largely remains to be demonstrated.

In principle, both centralized and distributed controllers can be used for reconfiguring modular robots. However, centralized high-level controllers for lattice-based modular robots are vulnerable to system failures or malfunctions of robot modules. By contrast, decentralized controllers are more robust and flexible under uncertain environments. However, one major challenge in developing a decentralized controller for self-reconfigurable modular robots lies in the coordination of local behaviors of multiple modules to achieve the desired global pattern to adapt to environmental changes. To address this challenge, researchers have turned their attention to biological systems, where rich examples can be found of which robust and complex emergent behaviors are generated through relatively simple local interactions in the presence of uncertainties [12][21]. Inspired by the biological concept of hormone, Shen et al. [26] proposed a hormone-inspired adaptive communication protocol and adaptive distributed control protocol to allow modules to use hormone-like messages to accomplish locomotion and self-reconfigurations. Recently, Schmickl et al. [24] developed a control system called AHHS (artificial homeostatic hormone systems) for the self-reconfiguration of robotic organisms, where the parameterization of the underlying dynamical system was encoded into a “genome” or the organism that is subject to selection through evolutionary algorithms. Stoy [27] proposed a cellular automata and gradients to control self-reconfigurable modular robots. Pfeiffer and Bongard [22] mainly focused on using evolutionary algorithms to generate robot configurations using an embodied system. In this work, we concentrate on the genetic and cellular mechanisms underlying the morphogenesis of multi-cellular organisms. The connection between reconfigurable modular robots and multi-cellular organisms is straightforward. Each unit in modular robots can be considered as a cell, and similarities can be found in control, communication and physical interactions between cells of multi-cellular organisms and modules of 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 modular robots to self-organize modules using biological morphogenetic mechanisms.

Inspired by the embryonic development of multi-cellular organisms [30] and a mechanochemical model for cell morphogenesis [18], we propose in this paper a two-layer hierarchical morphogenetic mechanochemical framework for self-reconfiguration of modular robots. Layer 1 is responsible for autonomous formation of chemical patterns in a changing environment, which is based on a mechanochemical model. Layer 2 aims to physically realize the target pattern for the modular robot, which is a gene regulatory network (GRN) based controller. Furthermore, to optimize the pattern design of modular robots, the covariance matrix adaptation evolution strategy (CMA-ES) [8][9] is employed to evolve the pattern parameters of the mechanochemical model.

The rest of the paper is organized as follows. The basic mechanics and locomotion design of a simulated lattice-based modular robot, CrossCube, are described in Section II. Section III presents the morphogenetic framework for self-organization of modular robots divided in four parts. First, the multi-cellular morphogenesis and its computational modeling are briefly introduced. Second, the basic idea of the hierarchical framework is introduced. Third, a virtual-cell based mechanochemical model is proposed for chemical pattern formation to generate target patterns. Finally a GRN-based controller for physically realizing the target pattern is described. The CMA-ES for evolving the mechanochemical model is discussed in Section IV. Section V provides extensive simulation results to evaluate the proposed morphogenetic approach to self-reconfiguration of modular robots under changing environments. Section VI concludes the paper with future work.

II. CrossCube: A Simulated Modular Robot

CrossCube is a simulated lattice-based modular robot we developed in a robot simulator using a real time physics engine PhysX. Each module of CrossCube has a flexible single cubic shape like Molecube [35], which does not require much free space for modules to move around, similar to the mechanics of SUPERBOT [4][23][20] and MTRAN [13] [14] [32]. Each module of CrossCube is a cubical structure having its own computing and communication resource and actuation capability. Like other 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 include up, down, right, left, forward, and backward, respectively. Each joint can attach to or detach from the joints of its neighbor modules. The axis of each joint can be actively rotated, extended, and return to its default direction.

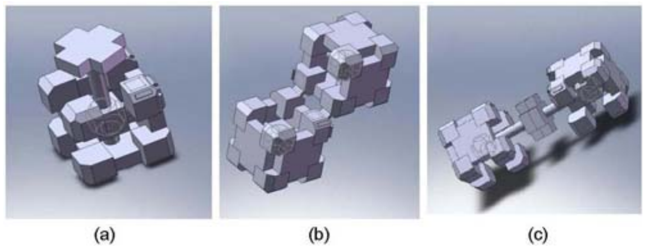


Fig 1. Mechanical demonstration of CrossCube. (a) The joints; (b) The locks on the boundaries of the modules. (c) Rotation and extension of the joints of the modules.

The cross-concaves on each side of the shell restrict the movement trajectory of the joints, as shown 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). The length and angle of the lock mechanism can also be adjusted on the boarders of the modules.

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 an immediate neighboring position. All joints of the modules will stick out slightly to make enough free space for modules to move. 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 readers can refer to [16] for the detailed mechanical design of CrossCube.

III. The Morphogenetic Approach

A. Computational Modeling of Multi-Cellular Morphogenesis

Multi-cellular morphogenesis is under the control of gene regulatory networks. A large number of computational models for GRNs have been suggested [1-3], which can largely be divided into discrete models, such as random Boolean networks and Markov models, and continuous models, such as ordinary differential equations and partial differential equations. GRN models can also distinguish themselves between deterministic models and stochastic models according to their ability to describe stochasticity in gene expression. In artificial life, a few high-level abstraction models have also been used for modeling biological development, such as the L-system [15] and grammar trees [6].

B. *The Proposed Hierarchical Framework*

To develop a morphogenetic approach to self-reconfiguration of modular robots, two steps must be achieved. First, the target pattern that a modular robot needs to form has to be generated automatically based on the current environment, which is to be achieved by the layer 1 in the proposed hierarchical framework. Second, the modular robot needs to self-organize their modules to physically realize the target pattern generated by layer 1, which is the task of layer 2 in our hierarchical framework. Fig. 2 shows a block diagram of this hierarchical morphogenetic framework.

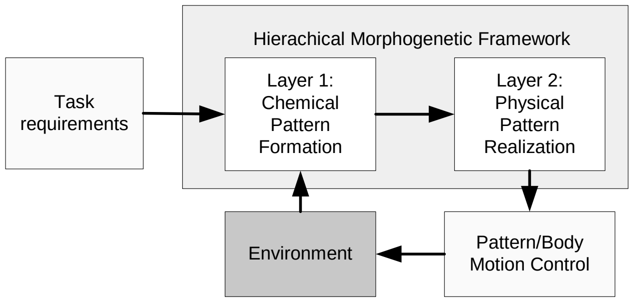

Hierachical Morphogenetic Framework
Task requirements
Layer 1: Chemical Pattern Formation
Layer 2: Physical Pattern Realization
Environment
Pattern/Body Motion Control

Fig. 2. A block diagram of the hierarchical morphogenetic framework.

C. *Layer 1: Chemical Pattern Formation*

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

Interestingly, morphogenesis of multi-cellular systems provides a nice example in which self-organizing pattern formation is realized through cell-cell and cell-environment interactions. One cellular environment is a complex mixture of nonliving material that makes up the extracellular matrix (ECM) [18]. In biological systems, the ECM is the extracellular part of animal tissue secreted by the cells that usually provides structural support to the cells. Meanwhile, the shape of ECM is also affected by the behaviors of the cells through the interactions between ECM and cells. The interaction dynamics between the cells and the ECM can be described by a morphogenetic mechanochemical model proposed by Murray et al. [18][19].

For this reason, the mechnochemical model proposed by Murray et al has been adopted and adapted as layer 1 of the proposed hierarchical model for chemical pattern formation. In the model, pattern formation (chemical blueprint generation) and morphogenesis (physical realization) are considered to occur simultaneously. One major advantage of this simultaneous development is that such mechanism is a closed-loop system where the extracellular matrix (ECM) can be treated as environmental constraints and external forces can be represented by task requirements, which is therefore well suited for constructing layer 1 in a hierarchical GRN framework.

Main challenges in applying this mechanochemical model to self-reconfiguration of modular robots include: (1) the modules can only locate at discrete positions, while cells in organisms can move continuously. (2) In the mechanochemical model, the macro-level behavior of a large number of cells is considered in terms of density change, while in modular robots, the micro-level behavior of a much smaller number of modules is considered.

In this paper, we will tackle these issues using a virtual-cell (v-cell) based approach. The basic hypothesis and assumptions of this approach are listed as follows:

- It is assumed that continuous 3D patterns of modular robots can be divided into discrete grids, which have the same size as the modules of modular robots.
- There are three basic components in this model: v-cells, ECM, and environment (including task requirements and local environmental constraints). In the beginning, a predefined number of v-cells start to proliferate from a fixed grid. V-cells can proliferate, interact with other v-cells, and move in a 3D space. The difference between the density of the v-cells and the ECM value defines the morphogen level of each grid of a modular robot, which will be read in by the GRN of layer 2. The ECM and environment constrain the behaviors of the v-cells.
- Each discrete grid in a 3D space can contain multiple v-cells and one associated ECM value. The ECM is static and each ECM is only associated with one discrete grid. The v-cells can only move to the grids that are either occupied by other modules, or empty grids that are the immediate neighbors. By following these rules, the v-cells will not move to the grids that are occupied by obstacles.
- The ECM value of a grid is determined by the density of v-cells of the immediate neighboring grids. If more than one ECM value is generated on the same grid by v-cells from different grids, the final ECM value is the sum of all the generated ECM values.
- In each discrete grid, the ECM value is inversely proportional to the density of the v-cells. On the other hand, ECM can slow down the generation of v-cells in the grid, i.e., the higher the ECM value, the more slowly the v-cells proliferate. As a result, v-cells tend to flow to the grids with a lower ECM value.

Based on the above assumptions, a new mechanochemical model for self-organization of modular robots is developed.

As shown in Fig. 3, the v-cells (red dots) can move freely in the modules of a modular robot and their neighboring discrete grids, while the ECM and task requirements (for example, a locomotion task to traverse through different terrains) can produce environmental and task constraints to the behavior of the v-cells.

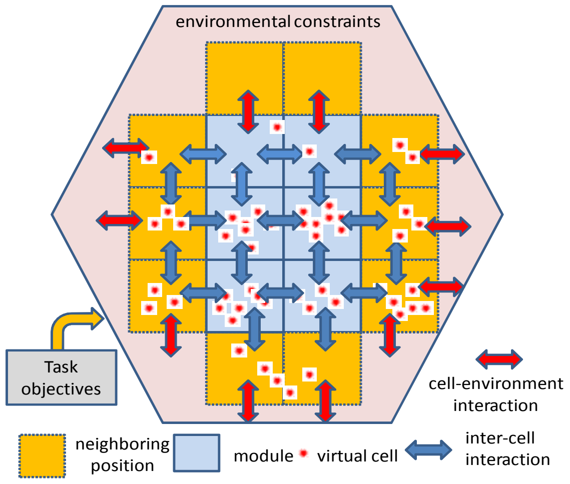


Fig. 3. The virtual-cell based mechanochemical model at a macro-level view.

The density of the v-cells and the ECM value in a grid can be defined as follows:

$$\frac{dn_i}{dt} = r\cdot n_i(N-n_i) + d\cdot K_i\cdot M_i - a\cdot\frac{\rho_i}{n_i+\rho_i} + n_{received}\ , \qquad (1)$$

$$\frac{d\rho_i}{dt} = -b\cdot\frac{n_i}{n_i+\rho_i} + e\cdot f(c_i)\ , \qquad (2)$$

where n_i represents the density of the v-cells and ρ_i represents the ECM value in grid *i*, respectively. The first term on the right-hand side of Eqn. (1) denotes the proliferation rate of the v-cells, where *N* is the predefined maximum number of v-cells allowed in the grid, and *r* is the maximum value of linear proliferation rate of the v-cells in the grid. It can be seen that the proliferation rate of the v-cells will be expedited when the local density of the v-cells is low, and will be reduced when the density of the v-cells approaches to *N*.

The second item in Eqn. (1) denotes the random dispersal of the v-cells, which is dependent on a dispersal control vector K_i and a vector M_i for grid *i*. Since the v-cells are allowed to move only to their immediate neighbors, v-cells have 6 possible directions to move to: up, down, left, right, forward and backward. The dispersal control vector K_i is defined as $K_i = \left[k_i^{up}, k_i^{down}, k_i^{left}, k_i^{right}, k_i^{forward}, k_i^{backward}\right]$, where each element of K_i represents the dispersal rate for each direction. A higher dispersal rate means a faster exchange of the v-cells in that direction. A zero dispersal rate means that the v-cells cannot move to that direction. To consider environmental constraints, for example, when an obstacle is detected, or when a narrower passage is detected, each dispersal control variable in vector K_i can further be defined as:

$$k_i^s = \begin{cases} 0 & \text{if neighbor position } s \text{ is occpied by obstacles} \\ h & \text{otherwise} \end{cases} \qquad (3)$$

where k_i^s is the dispersal control variable in direction *s* (i.e., up, down, left, right, forward, or backward) in grid *i*. *h* is a predefined constant. $M_i = \left[m_i^{up}, m_i^{down}, m_i^{left}, m_i^{right}, m_i^{forward}, m_i^{backward}\right]^T$ is a vector that is affected by the density difference of the v-cells between grid *i* and its 6 neighboring grids. For example, m_i^{up} is the density difference of the v-cells in the upper neighboring grid and that in grid *i*, which is defined as $m_i^{up} = \max(0, n_{up} - n_i)$,where n_{up} and n_i are the v-cell density of upper neighboring grid and current grid *i*, respectively. *d* is a predefined fluid rate.

The flow control vector K_i plays an important role in dispersal of virtual cells. Expanding the second item of Eqn. (1), we have

$$d*(k_i^{left}\cdot m_i^{left} + k_i^{right}\cdot m_i^{right} + k_i^{up}\cdot m_i^{up} + k_i^{down}\cdot m_i^{down} \\ + k_i^{forward}\cdot m_i^{forward} + k_i^{backward}\cdot m_i^{backward}) \qquad (4)$$

which is the total number of v-cells that will flow out of the local position *i*. The diffusion to each direction is scaled by the corresponding element in K_i. For example, if $k_i^{left} = 0$, local v-cells will never diffuse to its left neighboring grid. By setting up the flow control vector K_i, the preferred diffusion direction of the v-cells can be defined. This is very useful for achieving the target patterns. In the experiments, we will demonstrate how to design the flow control vector K_i for the robot to climb on stairs.

The third item in Eqn. (1) describes how the ECM value is reduced by the density of the v-cells. When the ECM value ρ_i is much larger than the density of the v-cells n_i in grid *i*, the density of v-cells will be reduced significantly, where *a* is a predefined scaling coefficient. The last item in Eqn. (1) $n_{received}$ denotes the v-cells coming from neighboring grids.

Eqn. (2) describes the dynamics of the ECM value. The first item on the right-hand side of Eqn. (2) is similar to the third item in Eqn. (1), where *b* is a predefined scaling coefficient. The second item in Eqn. (2) denotes the process how the ECM value is generated by the v-cells in grid *i*. Since the ECM value plays an important role in guiding the movement of the v-cells and in generating the target pattern, the generation of the ECM value should contain information about the preferred grids (position) of the target pattern. For example, if a tower-like pattern is desired, the ECM value in the upper grids should be less than that of its neighboring positions. That is, the smaller ECM value in the upper grids would trigger v-cells move to the upper grids to build up the tower pattern. Therefore, the generation of the ECM should be a function of the grid position, which is represented by $f(c_i)$, where c_i represents the position of grid *i*. *e* is a predefined coefficient.

For different target patterns, the rules for defining $f(c_i)$ are different. To autonomously reconfigure the robot to adapt to environmental changes, when a new environment constraint is detected by the robot, a new $f(c_i)$ will be defined together with the associated flow control vector K_i if needed. The corresponding parameters of $f(c_i)$ and K_i can be selected heuristically in the beginning, and the optimal parameters can be achieved with the help of an evolutionary algorithm, which will be discussed in details in Section V. We will give a detailed description of how to define $f(c_i)$ to generate a vehicle-like pattern and a pattern for climbing in the simulation section.

The target pattern is determined by the morphogen level in each grid. From the assumptions for the v-cell based mechanochemical model, we know that the higher the density of v-cells in the grid, the more likely the grid belongs to the target pattern. By contrast, the higher the ECM value in the grid, the grid has stronger environmental constraints. Therefore, the morphogen level of each grid can be defined as the difference between the density of the v-cells and the ECM value in the grid:

$$mp_i = n_i - \rho_i \tag{5}$$

where mp_i represents the morphogen level of grid *i*, which can be positive or negative depending on the density of the v-cells (n_i) and the ECM value (ρ_i). A positive morphogen level means that the grid should be occupied by a module, i.e., this grid belongs to the target pattern, whereas a negative value denotes that this grid does not belong to the target pattern, and if there is a module in the grid, it should be moved to another grid. A higher value of morphogen level indicates a higher priority for the grid to be filled by a module.

Eventually, a grid whose morphogen level is greater than a predefined pattern threshold *Z* is assigned to the target pattern. This mechenochemical process stops when the number of the assigned target grids reaches the number of the available robot modules, and the formation of chemical target pattern completes.

Chemical pattern formation using the mechanochemical model of layer 1 can be summarized as follows. First, a predefined number of v-cells start to proliferate from a fixed starting grid in a 3D space, and the ECM value is generated in the grids according to Eqns. (1) and (2). Second, the v-cells proliferate and migrate to different grids over time based on the environmental constraints and task requirements. Third, the target pattern is defined based on the density of the virtual cells and the ECM value, which serves as the input to the GRN model in layer 2.

D. *Layer 2: Physical Pattern Realization*

While the target pattern of the modular robot is defined in terms of chemical density by the mechanochemical model in layer 1, the physical realization of the target pattern, i.e., the configuration of the modular robot to its target shape, is controlled by a gene regulatory network (GRN) model. To this end, we need to build a metaphor between a multi-cellular organism and a modular robot. In this metaphor, each unit of the modular robot is considered as a cell containing a DNA. The DNA consists of a number of genes that can produce different proteins. The proteins can diffuse into neighboring modules (cells), through which local communications between the modules can be established. The concentration of the diffused proteins decays over time so that morphogen gradients can be generated.

As described above, the target pattern of the modular robot is defined by morphogen levels associated to the grids. To place the modules in the grids of the target pattern, a local coordinate system must be built up. This can be achieved by setting an arbitrary module to be the origin of the coordinate system [7], and thus all other modules can figure out their relative positions to the module on the origin through local communications. With the help of the relative positions and the morphogen levels of the target pattern, each module can decide: (1) if attracting proteins should be produced to generate positive morphogen gradients for attracting other modules to fill in its neighboring positions, or (2) if repelling proteins should be produced to generate negative morphogen gradients for removing neighboring modules, or (3) if the module should respond to the attraction and repellence requests sent out by other modules. Thus, the reconfiguration of the modular robot is controlled through changing the state of the individual modules.

In the following subsections, we first describe the finite state machine of a module. Then, we introduce a GRN model that has two gene-protein pairs to control the desired state transitions of modules needed for realizing the target pattern.

(1) Finite States of Modules

Each module can have five different states, namely, ‘stable’, ‘unstable’, ‘attracting’, ‘repelling’, and ‘repelled’. The finite state machine of these five states of modules is given in Fig. 4. When an ‘unstable’ module arrives at the destination position (grid), it changes its state to “stable” (arrow *a* in Fig. 4). When a ‘stable’ module has neighboring positions having a positive morphogen level, it can change its state to ‘attracting’ (arrow *b* in Fig. 4) to attract ‘unstable’ modules to fill in those neighboring positions. Once those neighboring positions are occupied by a module, the ‘attracting’ module returns to the ‘stable’ state (arrow *c* in Fig. 4). A ‘stable’ module may also move away from its current position so that it can fill in a more important position in the pattern (with a higher positive gradient) by turning its state to ‘unstable’ (arrow *d* in Fig. 4).

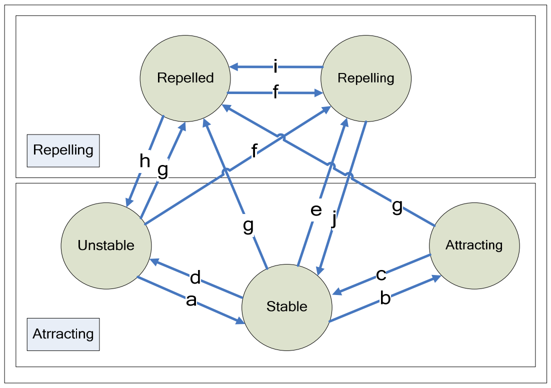


Fig. 4. State transition of each module in CrossCube.

A 'repelling' module repels some or all of its neighbors from their current position. Then the 'repelled' module moves away from those unavailable positions, and switches its state to 'unstable' (arrow *h* in Fig. 4). A module can be triggered to the 'repelling' state in two situations. The first situation is when a 'stable' module finds out that some of its neighboring modules are located in the positions with a negative morphogen level. In this case, it changes its state to 'repelling' (arrow *e* in Fig. 4) and switches the state of those neighbors to 'repelled' (arrow *g* in Fig. 3). When all the 'repelled' modules have left, the 'repelling' module returns to the 'stable' state (arrow *j* in Fig. 4). The second situation is when a deadlock occurs. A deadlock happens when a 'repelled' module is blocked by its neighboring modules. To resolve this deadlock, the 'repelled' module switches its state to 'repelling' (arrow *f* in Fig. 4), and changes the state of all its neighbors to be 'repelled' (arrow *g* in Fig. 4). This removes some of its neighboring modules to make room for the blocked module to move away. Then the 'repelling' module returns to the 'repelled' state (arrow *i* in Fig. 4).

The state transitions are controlled by a GRN 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 grids with a negative morphogen level should be kept empty as long as module migration is still in need for reconfiguration.

(2) 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 Fig. 4. In the beginning of the physical pattern realization, all modules are set as 'unstable'. After they are initialized with the target pattern and the relative position to the origin of the pattern, modules that are located in the grids with a positive morphogen level become 'stable'. A new 'stable' module initializes the gene expression level of its attracting gene G_A to zero, and generates an attracting protein P_A for all of the empty neighboring grids having a positive morphogen level so that 'unstable' modules can be attracted to fill in these empty grids. Here, P_A is defined as

$$P_A^{ij} = \{AP^{ij}, S^i, C_A^{ij}\} \qquad (6)$$

where P_A^{ij} is the attracting protein generated by *i*-th module for its *j*-th neighbor. AP^{ij} is the *j*-th neighboring attracting position of *i*-th module. S^i is the identification label of *i*-th module, and C_A^{ij} is the concentration of the protein P_A^{ij}, which equals to the morphogen level of AP^{ij}. P_A can regulate local G_A and diffuse into neighboring modules to regulate G_A of neighboring modules as well.

The dynamics of G_A and P_A can be described by the following GRN model:

$$\frac{dg_A(t)}{dt} = -k_1 \cdot g_A(t) + k_2 \cdot \sum p_{A\_local} - k_3 \cdot \sum p_{A\_rec} \qquad (7)$$

where $g_A(t)$ is the gene expression level of G_A at time *t*. p_{A_local} and p_{A_rec} are protein concentrations of locally generated protein and received protein from neighboring modules, respectively. k_1, k_2, and k_3 are constant coefficients.

Based on the expression level of g_A, the state of the module can be regulated according to the following rule:

$$\text{state} = \begin{cases} \text{'unstable'} & \text{if } g_A < G_{A\_L} \\ \text{'stable'} & \text{if } G_{A\_L} < g_A < G_{A\_H} \\ \text{'attracting'} & \text{if } g_A > G_{A\_H} \end{cases} \qquad (8)$$

where G_{A_L} is a negative threshold and G_{A_H} is a positive threshold. According to Eqn. (8), g_A falls below a negative threshold G_{A_L} with the increase of $k_3 \cdot \sum p_{A\_rec}$.

A higher value of $k_3 \cdot \sum p_{A\_rec}$ means that there are some more important positions to be filled in. So the module needs to change its state from stable to unstable and move to a more important position following the attracting morphogen gradient. An unstable module chooses a P_A with the highest concentration value from all the received attracting proteins. Then the module migrates to the attracting position requested by that P_A. In order to guide the unstable modules to migrate to their destination, each module can detect the proteins within its local environment and choose the position with the highest protein concentration as its destination. Once they reach their destination, the unstable modules become stable.

The expression level of g_A will be enhanced when $k_2 \cdot \sum p_{A\_local}$ is increased, which means that the module detects that there are some important neighboring positions it needs to fill in. So the module changes its current state to the attracting state. The attracting module emits attracting proteins in the grid in which it sits, and the emitted proteins will then diffuse into other modules. The attracting module will become stable again when its neighboring attracting positions are all occupied.

To summarize, the gene-protein pair ($G_A - P_A$) can regulate each other according to the GRN model described in Eqns. (7) and (8). More specifically, P_A can regulate G_A through Eqn. (7). On the other hand, G_A can determine when P_A is allowed to diffuse based on Eqn. (8). That is to say, only if the expression level of G_A is above G_{A_H} and G_{A_L} , P_A can be generated; and only if the expression level of G_A is above G_{A_H} , P_A is allowed to diffuse.

(3) 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

$$P_P^{ij} = \left\{ RP^{ij}, S^{ij}, C_P^{ij} \right\} \tag{9}$$

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 position around *i*-th module. S^i is the identification label of repelling module *i*, and C_P^{ij} is the concentration of the protein P_P^{ij}, which equals to a predefined positive constant. As we mentioned before, when a stable module finds out that some of its neighbors are located in the positions with a negative morphogen level, it changes its state to the repelling and turns the state of these neighbors to repelled. If the repelling module is triggered in this situation, RP^{ij} is reset in such a way that P_P can only repel the specific neighbor modules that is located in RP^{ij}. If the repelling module is triggered by a deadlock, RP^{ij} is not set because P_P should be detected by all the neighboring modules of the repellent module.

The gene expression level of g_P is initialized as the morphogen level of the current position of the module. It can be regulated by P_P through the following equation:

$$\begin{aligned} \frac{dg_P(t)}{dt} &= -k_4 \cdot g_P(t) - k_5 \cdot \sum p_{P\_rec} \\ \text{state} &= \text{repelled when } g_P < G_{P\_L} \end{aligned} \tag{10}$$

where $g_p(t)$ is the gene expression level of the repellent gene G_P at time *t*. p_{P_rec} is the concentration of the received repellent protein. G_{P_L} is a negative constant threshold, and k_4 and k_5 are constant coefficients.

When a module receives P_P, the concentration of g_P will be reduced. If $g_p < G_{P\_L}$, the module changes its state to ‘repelled’. Obviously modules with a lower morphogen level are more easily to be repelled.

To summarize, P_P can regulate G_P through Eqn. (10). G_P can produce P_P under the condition that G_P is below G_{P_L} and the module is blocked.

IV. Evolving the Pattern Model using the CMA-ES

To optimize the chemical pattern formation using the mechanochemical model, the covariance matrix adaption evolution strategy (CMA-ES) is employed for tuning the pattern formation parameters in the model. CMA-ES is a stochastic, iterative optimization method belonging to the class of evolutionary algorithms (EAs) proposed by Hansen and Ostermeier [8][9]. CMA-ES has shown to work efficiently with a small population size.

Based on the CMA-ES [8], λ offspring are generated from μ parents, where μ≤λ. Offspring individuals are generated by adding a random number generated from a normal distribution to a parent, or a recombination of a number of parents:

$$x_i^{(g+1)} \sim m^{(g)} + \sigma^{(g)} N_i(0, C^{(g)}) \qquad \text{for } i = 1, \ldots, \lambda \tag{11}$$

where “~” denotes the same distribution on the left and right side. $x_i^{(g+1)}$ is the *i*-th offspring individual in generation $g+1$, and x is a vector $x \in R^n$, which consists of the parameters need to be evolved. Vector $m^{(g)} \in R^n$ represents a recombination of a few parent individuals of generation g. $\sigma^{(g)}$ is the step-size that controls the step length of mutation in generation g. $C^{(g)}$ denotes the covariance matrix in generation g and $N_i(0, C^{(g)})$ is the *i*-th multivariate normal distribution with zero mean and covariance matrix $C^{(g)}$.

To determine the complete iteration, $m^{(g+1)}$, $\sigma^{(g+1)}$ and $C^{(g+1)}$ are defined as:

$$m^{(g+1)} = m^{(g)} + \sigma^{(g)} y_w^{(g)} \tag{12}$$

$$\sigma^{(g+1)} = \sigma^{(g)} \times \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\left\| p_\sigma^{(g+1)} \right\|}{E\left\| N(0, I) \right\|} - 1 \right) \right) \tag{13}$$

$$\begin{aligned} C^{(g+1)} &= (1 - c_1 - c_\mu) C^{(g)} + c_1 p_c^{(g+1)} p_c^{(g+1)^T} \\ &+ c_\mu \sum_{i=1}^{\mu} w_i y_{i:k}^{(g+1)} y_{i:k}^{(g+1)^T} \end{aligned} \tag{14}$$

where $c_\sigma < 1$ is the learning rate for the step-size adaptation, $d_\sigma \approx 1$ is the damping parameter for step-size update. c_1 and c_μ are the learning rates for the rank-one update and the rank-μ update of the covariance matrix, respectively. w_i is the i-th positive weight coefficient for recombination. In addition, functions $y_w^{(g+1)}$, $p_\sigma^{(g+1)}$, and $p_c^{(g+1)}$ can be obtained from generation $g+1$ as follows,

$$y_w^{(g+1)} = \prod_{i+1}^{\mu} w_i y_{i:k}^{(g+1)} \tag{15}$$

$$\begin{aligned} p_\sigma^{(g+1)} &= (1 - c_\sigma) p_\sigma^{(g)} \\ &+ \sqrt{c_\sigma (2 - c_\sigma) \mu_{eff}}\, C^{(g)^{-\frac{1}{2}}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \end{aligned} \tag{16}$$

$$p_c^{(g+1)} = (1 - c_c) p_c^{(g)} + \sqrt{c_c (2 - c_c) \mu_{eff}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \tag{17}$$

where $\mu_{eff} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1}$ is the variance effective selection mass and $y_{i:k}^{(g+1)} = \left(x_{i:k}^{(g+1)} - m^{(g)}\right)/\sigma$ .The reader is referred to [9] for a detailed description of CMA-ES.

V. Experimental Results

To evaluate the efficiency and robustness of the morphogentic framework for self-reconfiguration of the modular robot, CrossCube, several case studies have been conducted in a robot simulator, as shown in Fig. 5. 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 of the hierarchical model are set up as follows. The parameters of the GRN model in layer 2 are: $k_1 = 0.7$, $k_2 = 1$, $k_3 = 1$, $k_4 = 0.5$, $k_5 = 2$, G_{A_L} = -1, G_{A_H} = 1, $G_{P\_L} = -2$, C_P^{ij} = 0.7. The parameters of the model in layer 1 are: *r*=0.005, *N*=200, *d*=1, *a*=0.05, *b*=20 and *e*=1. Protein concentration decays to 80% of its previous level when it diffuses into a neighboring module. For case studies 3 and 4, the CMA-ES method is employed to evolve the pattern parameters in the mechanochemical model to achieve optimal pattern formation, and the evolved parameters are provided in Sections V(C) and V(D).

A. Case Study 1: Pattern Formation

In case study 1, we focus on the vehicle pattern formation. As we mentioned before, different $f(c_i)$ should be defined in Eqn. (2) for different target pattern. We first provide the following design guidelines for the vehicle patterns before discussing the experimental results.

- The chassis of a vehicle pattern should be a rectangle or similar to a rectangle. The total width of the vehicle is denoted by $W = x_{\max} - x_{\min}$, where $x_{\max}$ and $x_{\min}$ are the rightmost and leftmost positions of the vehicle pattern.
- There have to be modules served as wheels on the left and right boundary of the chassis of a vehicle pattern. Wheel modules cannot be immediate neighbors, otherwise they cannot rotate freely.
- The target pattern should be built up from the bottom floor to the top. Only when the bottom floor is filled up and there are still modules left, the top floor will be filled. For example, a pattern that only has a single floor is preferred to the pattern with multiple floors in an open space.

Based on these guidelines, the function $f(c_i)$ can be defined in the following manner. To follow the first two guidelines, when the input position c_i is at the side-boundary positions (i.e., the rightmost or leftmost positions), the v-cells on this position will generate ECMs with the level of ECM_{wheel} on all immediate neighboring positions except for the one on "inner" direction toward the pattern. This rule can be used to generate rectangle chassis and wheel shapes on both sides of the chassis. To follow the third guideline, when the input position (the current grid position) does not contain any v-cells previously and is occupied by v-cells for the first time, the v-cells will generate an ECM with the level of ECM_{up} on the upper neighboring position to prevent the system from building top-floor modules.

Three experiments are carried out to verify the effectiveness of the proposed hierarchical morphogenetic model. In the first experiment, 16 robot modules are initially deployed in a 4x4 square. The space available for robot modules for self- reconfiguration is a 4x5 space as shown in Fig. 5(a).

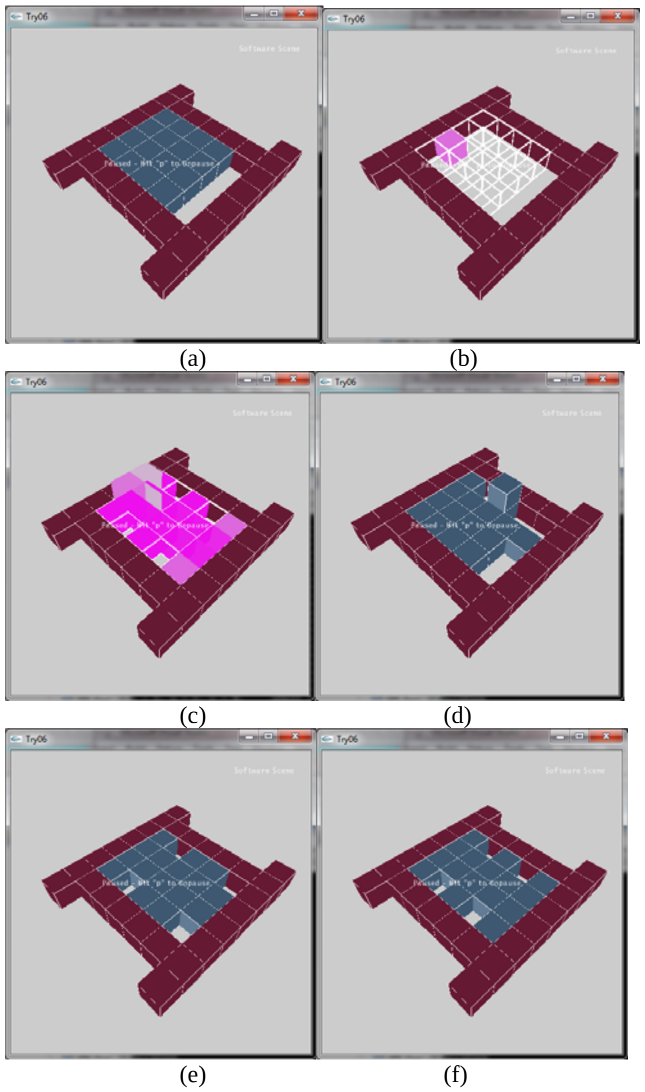


Fig. 5. A set of snapshots of the self-reconfiguration process using the proposed hierarchical morphogenetic model in a 4x5 space. (a) The initial configuration of a modular robot in a 4x5 space. (b) The starting stage of the running v-cell model. (c) The final stage of the density distribution of v-cells. (d)(e) The GRN-based algorithm is applied to mechanically build up the target shape produced by v-cell based model. (f) The final target pattern has been constructed by the modular robot.

The parameters in $f(c_i)$ are defined as $x_{\min}$ = 0, $x_{\max}$ = 4, ECM_{wheel} =n*100, ECM_{up} =130, where n is the density of v-cells in the input position c_i . Pattern threshold Z is set to 100, which means that a grid will be considered to belong to the target pattern if the density of the v-cells in the grid is higher than 100. Since there is no special request for the grow trend of the v-cell flow, the flow control vector of v-cells is defined as follows:

$$\left[k_i^{up}=1, k_i^{down}=1, k_i^{left}=1, k_i^{right}=1, k_i^{forward}=1, k_i^{backward}=1\right].$$

Figs. 5(b)-(f) demonstrate the procedure of the self-configuration of a modular robot using the hierarchical morphogenetic model. First, the v-cell based mechanochemical model is applied to create the target pattern based on the environmental constraints and target pattern requirements. Then, the GRN model is employed to self-reconfigure the modules to physically realize the target pattern defined by chemical concentrations. Fig. 5(b) shows that 100 v-cells start to proliferate from a randomly picked starting position and create ECMs on different grids according to Eqns. (1) and (2). Fig. 5(c) shows the final state of the distribution of the v-cells, where the candidate positions of the target pattern have been selected, and the morphogen level of each position is defined based on the difference between the density of the v-cells and the ECM value in the grid, which can be used as the input to the GRN model of layer 2. Figs. 5(d)-(e) show the procedure of the physical pattern realization using the GRN model of layer 2. Fig. 5(f) shows the final constructed target pattern of the modular robots.

Please be noted that the blue grids in Fig. 5 represent the real robot modules in the stable state. The black grids represent the modules in the unstable state. The pink grids represent the density distribution of v-cells in the robot modules, where the darker the color, the higher the density of the v-cells in the modules is. The grids in magenta color represent the obstacles in the environment. These representations apply to all the following simulation results, unless otherwise stated.

In the second experiment, the same number of robot modules is initialized in the same way as in the first experiment but in a smaller 4x4 space. The parameters in $f(c_i)$ are the same as in the first experiment. Based on the new environmental constraints (i.e., a 4x4 square space) with the same pattern constraints (i.e., a vehicle pattern with wheels on both sides), a two-floor vehicle pattern is constructed following the same procedure of the previous experiment, as shown in Fig. 6. Although the final pattern as shown in Fig. 6(f) is not fully symmetric, the requirements of vehicle pattern are still well met.

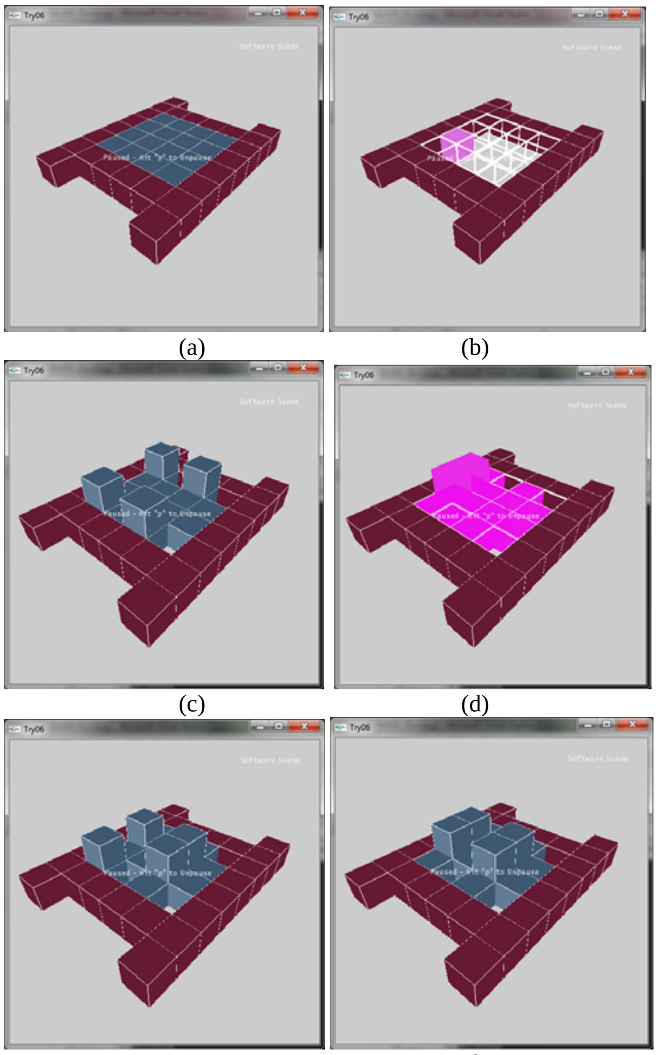


Fig. 6. A set of snapshots of the self-reconfiguration process using the proposed hierarchical morphogenetic model in a 4x4 space. The explanations of Fig. 6(a) to Fig. 6(f) are similar to Fig. 5.

In the third experiment, a vehicle pattern in an open space is generated using the proposed hierarchical model. The parameters in $f(c_i)$ are defined as follows: The parameters in $f(c_i)$ are: $x_{\min}$ = 0, $x_{\max}$ = 4, ECM_{up} =100, ECM_{wheel} =n*100, where n is the density of v-cells in input position. The pattern threshold Z is set to 170. The robots are distributed in a 4x6 rectangle.

Since there is no environmental constraints in an open space, a one-floor vehicle pattern will be generated, which matches the guidelines of the vehicle-like pattern design. The ECM valued generated by the v-cells on the left and the right boarders can be used to generate the wheels. The generated ECM value by a wheel slows down the diffusion of the v-cells into the wheel's neighboring grids. As a result, no two wheel modules can be immediate neighbors so that wheels can freely rotate. A set of snapshots of this experiment are shown in Fig. 7.

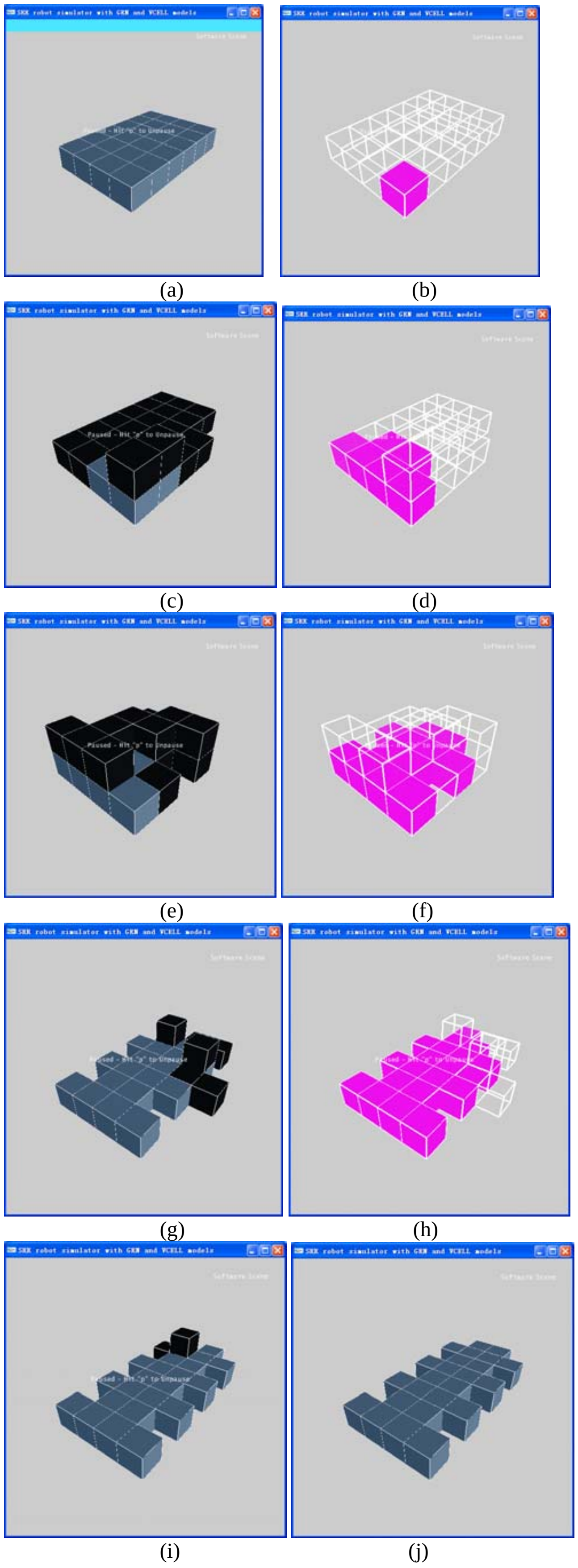


Fig. 7. A set of snapshots of a vehicle pattern generation of a modular robot in an open space.

B. *Case Study 2: Self-Repairing*

One important feature of reconfigurable modular robots is their ability to self-repair malfunctioned 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 level 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 level.

To evaluate the self-repairing performance of the GRN-based control in layer 2, another experiment is conducted here. The bottom-floor modules are functional modules in the vehicle pattern. The top-floor modules are backup modules that are used for self-repairing the malfunctioned parts of the vehicle pattern. Therefore, the backup modules have a lower morphogen level 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 move to fill in the damaged modules. Fig. 8 shows a snapshot of this self-repairing procedure using the proposed hierarchical framework. This experiment demonstrates that the proposed approach is able to self-repair a modular robot in case of failed modules.

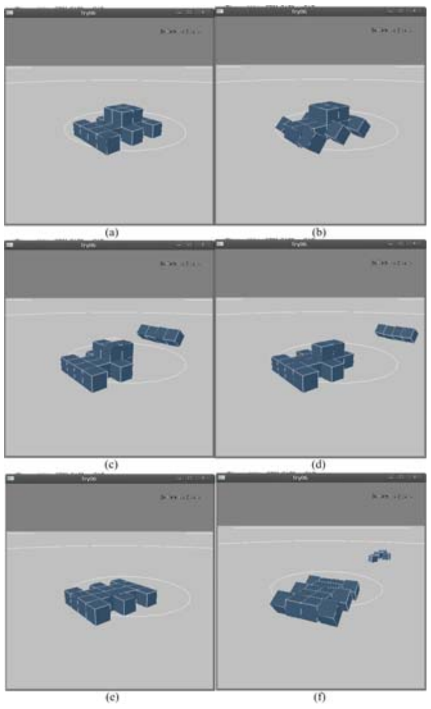


Fig. 8. A set of snapshots of the self-repairing of CrossCube using the GRN-based controller. (a) A vehicle pattern is formed. (b) The vehicle pattern moves forward. (c) Some modules are blown off when the explosion happens. (d) The failed part is filled up by the backup modules. (e) The vehicle is repaired. (f) The repaired vehicle continues moving.

C. *Case Study 3: Pattern Adaptation in a Changing Environment – Climbing Stairs*

This case study is to verify the efficiency and robustness of the hierarchical model for a modular robot to adapt to environmental changes. More specifically, we want to show that a modular robot using the proposed hierarchical model is able to climb stairs by changing its pattern autonomously. The starting position of the robot is initialized with 4000 v-cells and the robot is distributed in a 3x3 square.

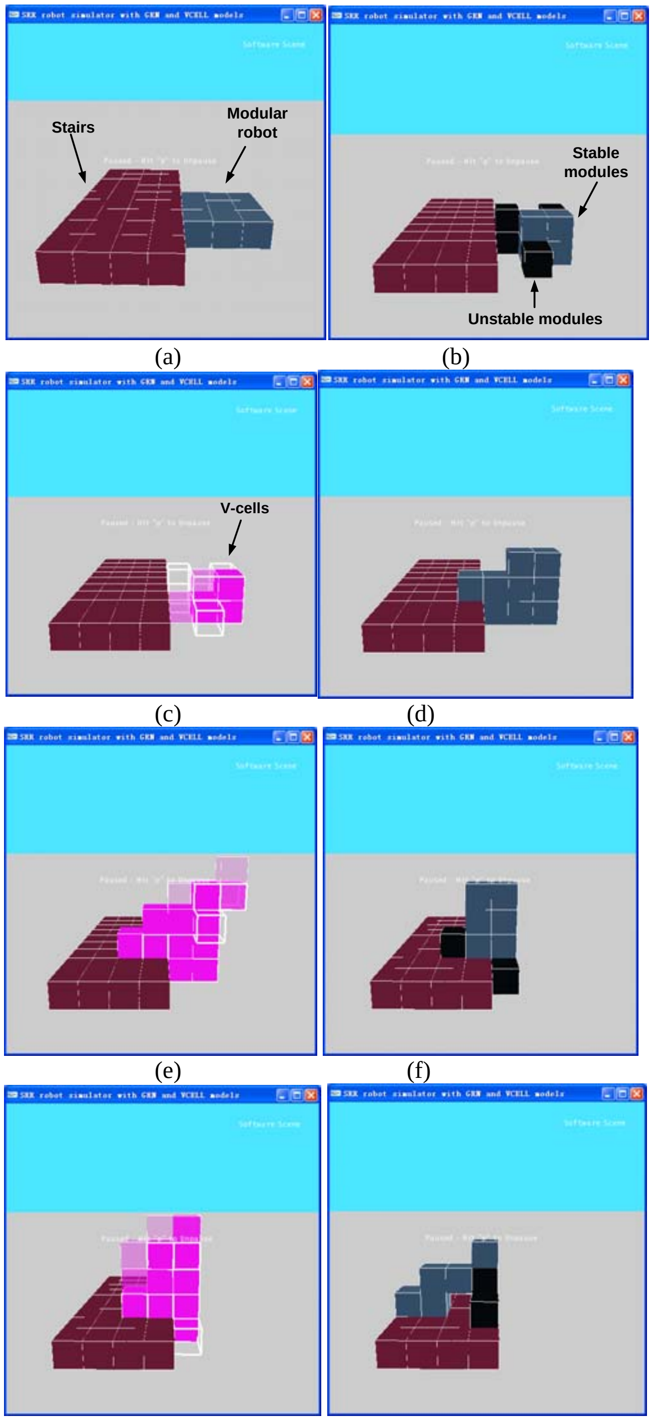
First, in order to trigger the robot to move forward, a forward "force" needs to be applied on all v-cells to push them to move forward. To update the density of v-cells in each grid, this forward "force" can be obtained from the last term $n_{received}$ in Eqn. (1). In other words, each grid will provide Q v-cells to its front neighboring position assuming that this front neighboring position is a valid position to hold v-cells (it is either occupied by a robot module or by an immediate neighbor of a robot module, however, it should not be occupied by obstacles).

Second, we need to define $f(c_i)$ generic pattern for climbing. Since we want the pattern to be able to move forward and climb up the stairs, the major environmental constraint is that the v-cells should not diffuse in other directions, such as right, left, or backward directions. Diffusion of v-cells in the right or left direction will increase the width of the pattern during the traveling, which may eventually make the movement impossible. Diffusion of the v-cells to the backward rear direction will make the robot moving backward. Therefore, the parameters of $f(c_i)$ include the width of the pattern, the v-cell fluid Q to the front direction, and the pattern threshold Z. If the input position c_i is at the leftmost or rightmost position of the pattern, it will release ECM to its immediate neighboring left or right grid with the level of n*100, where n is the v-cell density of the input position, so that the pattern will keep the same width all the time.

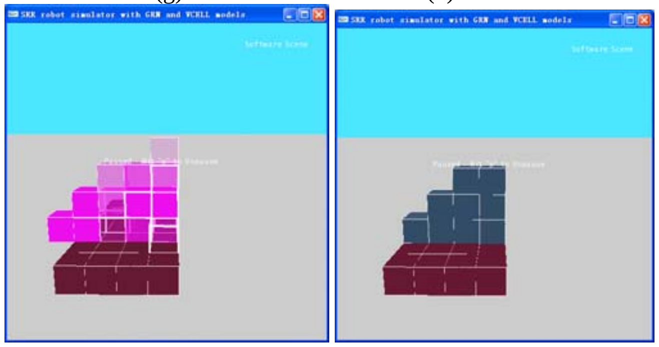
Since the robot has to climb up on the stairs, the v-cells flowing to an upper direction should be encouraged. Combined with the constraints of $f(c_i)$, we can also set up the flow control vector of v-cells to help for the climbing. .

Then, the CMA-ES is applied to evolve the above parameters of $f(c_i)$ and the flow control vector to achieve the optimal pattern through evolutionary learning. Since CMA-ES does not require a large population size, we set $\lambda = 20, \mu = 5$ and $\sigma = 50$. The fitness function is defined as the time needed for all the modules of the pattern can accomplish climbing on the stairs. The shorter the climbing time, the higher the fitness value is. The evolved parameters are Q=3, Z=30, $k_i^{up} = 1, k_i^{down} = 0$, $k_i^{left} = 0.5$, $k_i^{right} = 0.5, k_i^{forward} = 0.7$, and $k_i^{backward} = 0.7$ assuming that the width of the pattern keeps the same value 4 all the time.

A set of snapshots showing the adaptation of the vehicle pattern to environmental changes is provided in Fig. 9. When the stairs in the environment are detected in front of the robot, new target patterns are automatically generated to allow the robots to climb the stairs. During this procedure, the GRN-based controller for physical pattern formation automatically reconfigures the modules to form the new target patterns. Since this is a distributed system, where each module makes its own movement decisions based only on its local information, the dynamics of layer 1 and layer 2 is updated asynchronously. More specifically, the v-cells first flow to an intermediate position (not yet the target position), then the modules will move to these positions controlled by the GRN model. The two processes interleave until the target pattern is realized or the robot climbs up the stairs.



(g) (h)

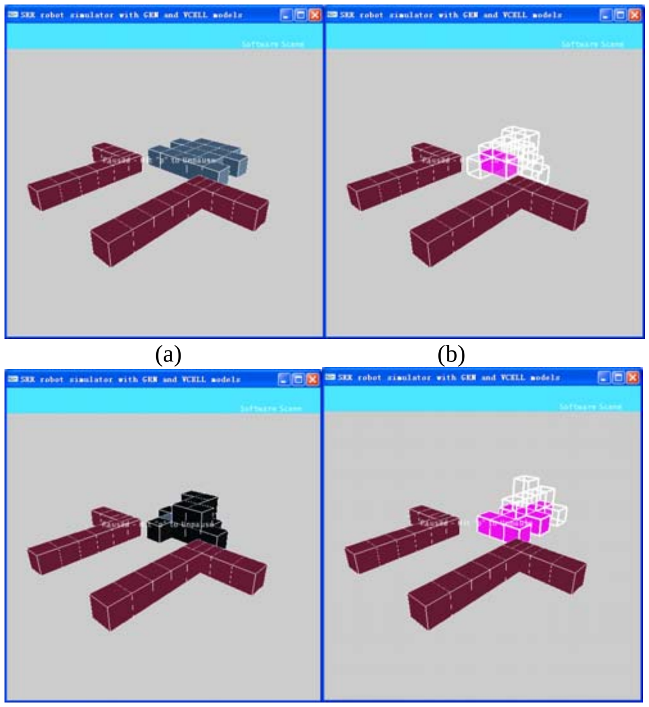


(i) (j)

Fig. 9. A set of snapshots demonstrating the reconfigurable processes in the stair-climbing case using the proposed hierarchical model.

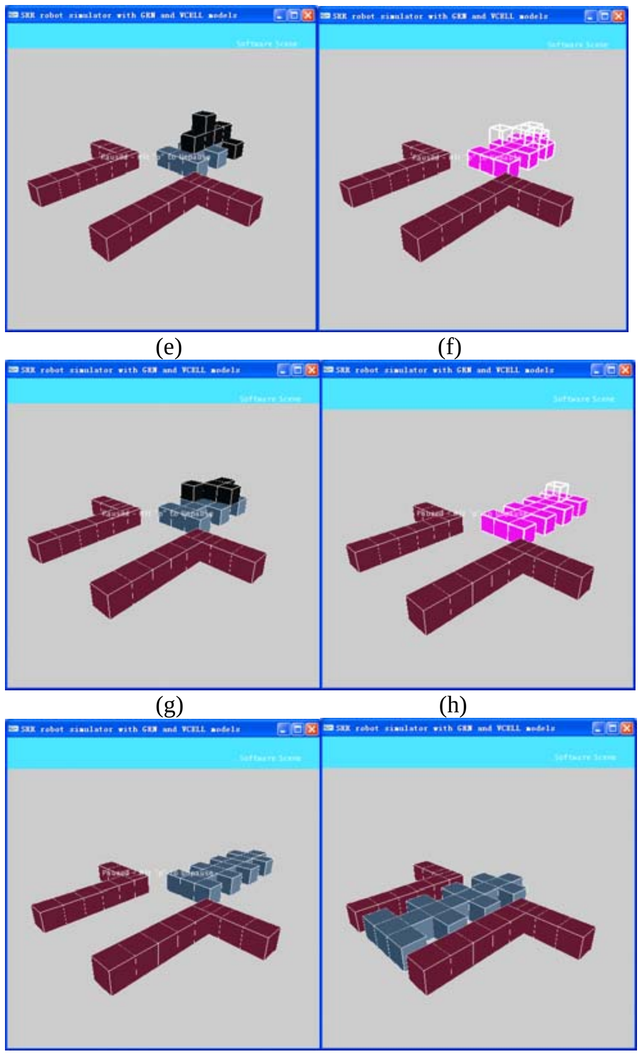
D. *Case Study 4: Pattern Adaptation in a Changing Environment – Traversing a Narrow Passage*

To demonstrate that the modular robot using the proposed hierarchical model can autonomously adapt to changing environments, another experiment was conducted. To optimize the pattern design of the modular robot for a locomotion task, the CMA-ES is applied to evolve the pattern parameters of $f(c_i)$, which include the width of the robot (*W*), the ECM_{up} value generated by the v-cells in the current grid to upper grids, and the threshold *Z* above which a grid will be considered as one in the target pattern.



(a) (b)

(c) (d)



(e) (f)

(g) (h)

(i) (j)

Fig. 10. A set of snapshots demonstrating a series of reconfigurable processes during locomotion in a changing environment where the robot adapted its width to a narrow path.

Here, for a locomotion pattern, the fitness function is defined as the travel distance within a certain time period. The longer the robot can travel within a certain time period, the better the current pattern is for the locomotion task. Similar to case study 3, we set $\lambda = 20, \mu = 5$ and $\sigma = 50$. The parameters of the best evolved vehicle-like pattern are: *W* = 3, ECM_{up} = 100, and *Z* = 136. Fig. 10 shows a set of snapshots of the self-adaption procedure of the robot in a changing environment, where the robot dynamically change its pattern when a narrower passage is detected.

VI. Conclusion and Future Work

In this paper, we presented a two-layer hierarchical morphogenetic approach to self-reconfiguration of modular robots, which is inspired by multi-cellular morphogenesis. Layer 1 of this morphogenetic framework is a virtual-cell based mechanochemical model that can generate appropriate target patterns for modular robots to adapt to environmental changes. Layer 2 is a gene regulatory network (GRN) based controller that can automatically self-reconfigure the modular robots to achieve the desired configuration generated by layer 1. In addition, the CMA-ES is employed to evolve the parameters of the mechanochemical model so that a modular robot can generate new patterns to adapt to environmental changes. 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 morphogenesis. In response to the environmental changes, the layer for defining the target pattern of the modular robot is able to adapt the target configuration, based on which the second layer can re-organize the modules autonomously to realize the target configuration.

In the future, we will investigate the following two issues. First, we will develop a physical modular robot of CrossCube and implement the proposed hierarchical morphogenetic framework on real modular robots. Second, more complex environmental changes will be implemented to evaluate the robustness of the proposed approach.

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] Guo, H., Meng, Y., and Jin, Y., 2010, A Multi-Cellular Self-Organization Algorithm for Swarm Robotic Systems using Local Relative Information, Submitted to 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems.

[8] Hansen, N, and Ostermeier, A., Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 9(2) pp.159-195. 2001.

[9] Hansen, N., Müller, S. D., and Koumoutsakos, P., Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation, 11(1) pp.1-18. 2003.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] Meng, Y., Zhang, Y., Wierzchowski, J., and Jin, Y., 2010, CrossCube: A Self-Reconfigurable Modular Robot using a Morphogenetic Approach. Under preparation.

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

[18] Murray, J. D., Modelling biological pattern formation in embryology, ISI Atlas of Science: Animal and Plant Sciences 1: 270-274, 1988.

[19] Murray, J. D., and Maini, P. K., Mechanochemical models for generating biological pattern and form in development, Physics Reports, 171, no. 2, pp.59-84, 1988.

[20] 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.

[21] Nolfi, S., and Floreano, D., Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, MIT Press, 2004.

[22] Pfeiffer, R., and Bongard, J. C., How the Body Shapes the Way We Think. MIT Press, 2006.

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

[24] Schmickl, T., Hamann, H., Stradner, J., and Crailsheim, K., Hormone-based control for multi-modular robotics. In Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution, P. Levi and S. Kernbach, Eds. Springer, Feb. 2010.

[25] 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.

[26] Shen, W.-M, Salemi, B., and Will, P., Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots, IEEE Trans. on Robotics and Automation, vol. 8, no. 5, pp.700-712, Oct. 2002.

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

[28] 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.

[29] 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.

[30] Wolpert. L, 2002. Principles of Development. Oxford University Press.

[31] 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.

[32] 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.

[33] 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.

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

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