DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD

Self-Adapting Modular Robotics: A Generalized Distributed Consensus Framework

Citation	Yu, Chih-Han and Radhika Nagpal. 2009. Self-adapting modular robotics: A generalized distributed consensus framework. In Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, May 12-17, 2009, Kobe, Japan, 1881-1888. Piscataway, N.J.: Institute of Electrical and Electronics Engineers.
Published Version	doi:10.1109/ROBOT.2009.5152663
Accessed	February 19, 2015 9:15:16 AM EST
Citable Link	http://nrs.harvard.edu/urn-3:HUL.InstRepos:5360186
Terms of Use	This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Open Access Policy Articles, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#OAP

(Article begins on next page)

Self-Adapting Modular Robotics: A Generalized Distributed Consensus Framework

Chih-Han Yu

chyu@fas.harvard.edu

Radhika Nagpal

rad@eecs.harvard.edu

School of Engineering & Applied Sciences, Harvard University, Cambridge, MA, USA
Wyss Institute for Biologically Inspired Engineering, Harvard University, Cambridge, MA, USA

Abstract— **Biological systems achieve amazing adaptive behavior with local agents performing simple sensing and actions. Modular robots with similar properties can potentially achieve self-adaptation tasks robustly. Inspired by this principle, we present a generalized distributed consensus framework for self-adaptation tasks in modular robotics. We demonstrate that a variety of modular robotic systems and tasks can be formulated within such a framework, including (1) an adaptive column that can adapt to external force, (2) a modular gripper that can manipulate fragile objects, and (3) a modular tetrahedral robot that can locomote towards a light source. We also show that control algorithms derived from this framework are provably correct. In real robot experiments, we demonstrate that such a control scheme is robust towards real world sensing and actuation noise. This framework can potentially be applied to a wide range of distributed robotics applications.**

I. Introduction

In nature, biological systems gain a tremendous advantage by using vast numbers of simple and independent agents to collectively achieve group behaviors. This has inspired the area of modular robotics: a class of robotic systems composed of many independent, connected, programmable modules that coordinate among themselves to achieve desired tasks. The incorporation of modularity into robot design grants flexibility to a single type of robotic hardware and allows it to achieve the tasks of many different robotic systems. Most of the algorithmic research in modular robotics has been focused on systematic techniques for configuring the robot into a pre-defined structure or shape and techniques for programming locomotion [1], [2]. Less attention has been focused on how modular robots can autonomously reconfigure themselves to achieve environmental adaptation by sensing the external environment.

On the other hand, biological systems are able to achieve sophisticated tasks in uncertain environments via iterative sensing and self-adaptation. For example, birds and fish rely on simple local observations and actions to travel in groups and adapt to uncertainties. Similarly, a modular robot can potentially achieve many applications if it is capable of performing such sensing and self-adaptation to external environments, e.g., a modular structure that reconfigures to maintain uniform force distribution and a modular gripper that can sense an object and reconfigure to properly grasp it.

In our previous work [3], we proposed an algorithmic approach to self-adaptation in modular robots. We presented a simple and decentralized algorithm by which module agents can cooperate to solve complex tasks specified in terms of distributed constraints. We demonstrated several environmentally adaptive applications using a modular robot with distributed tilt sensors and actuators, e.g. a self-balancing table and a terrain-adaptive bridge. An interesting aspect of this control approach is that it differs significantly from other self-reconfiguration and self-assembly algorithms. Instead, it is closely related to a class of multi-agent algorithms called *distributed consensus* [4]. In [5], we studied this relationship and leveraged this result to prove scalability and robustness of the algorithm. We also outlined a more general set of conditions for reaching consensus. Nevertheless, this framework is strictly limited to shape formation tasks that can be described by distributed orientation sensor constraints. An important open question is whether this self-adaptation approach can be generalized to a larger class of modular robots with different sensors and actuators and different types of tasks.

In this paper, we propose a *generalized distributed consensus* control framework. We extend such a control scheme in several directions and demonstrate how this generalization allows many new application areas in modular robotics. First, we generalize it to new types of sensors, e.g., pressure and light sensors. In these cases, the module agent has an indirect relationship between its sensor and actuator. We demonstrate an example application in hardware: a modular pressure-adaptive column that is capable of reconfiguring itself to absorb uniform pressure (Fig. 1 (a)). Second, we extend it to the case in which the individual agent's action might have long range effects. We show that a modular gripper with this local condition is capable of grasping a fragile object using distributed sensing and actuation (Fig. 1 (b)). Finally, we extend this approach to a more complicated task: We equip modules in a tetrahedral robot with light and pressure sensors and formulate the robot's locomotion as a sequence of self-adaptations. We show that the robot is able to locomote towards the light source with a series of "pressure consensus" reaching processes (Fig. 1 (c-d)). This also shows that our framework is potentially applicable to other dynamic tasks.

This paper makes the following contributions: (1) We demonstrate that a variety of modular robot tasks can be formulated and solved as self-adaptation processes based on environmental feedback, including structure adaptation,
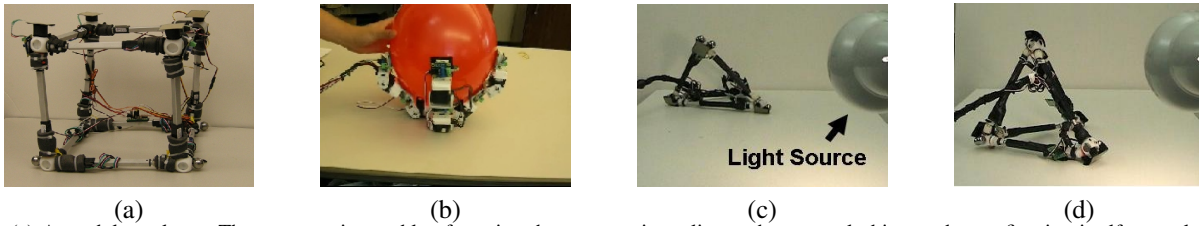
Fig. 1. (a) A modular column. The structure is capable of sensing the pressure it applies to the external object and reconfiguring itself to apply uniform pressure on it. (b) A module-formed robotic hand. The hand structure can form a configuration that grasps a fragile object, e.g., a balloon, with each module coordinating only with its local neighbors. (c-d) A modular tetrahedral robot performs locomotion by a sequence of self-adaptations to the external environment. The structure is capable of moving toward the light source.

gripper manipulation, and locomotion. (2) We use the same underlying distributed control principle to derive control laws for these tasks. The control laws are robust and simple to implement. (3) We show that the control laws are provably correct: we can guarantee convergence to the tasks we consider. (4) We implement the proposed control framework on three different hardware robot prototypes and show that it is robust toward sensing/actuation noise and exogenous perturbations. (5) We show that this framework can potentially be applied to many distributed robotics applications beyond modular robotics.

The rest of the paper is organized as follows: we start with a literature review in Section II. We define our robot model in Section III. We describe the generalized distributed consensus framework in Section IV. We then present three different self-adaptation tasks and their respective control laws in V. We evaluate the performance of this framework with real robots in Section VI. Finally, we provide a discussion and then draw conclusions in Section VII.

II. RELATED WORK

A modular robot is a new class of robots that are composed of many independent modules. Each module can communicate locally with other modules that are physically connected to it. While applying appropriate control, modular robots are capable of changing their configurations to become different structures or shapes, so they are sometimes referred as (self-) reconfigurable robots. There are mainly two types of hardware design for modular robots. The first type is the "chain-based" modular robot where modules are normally connected in a chain and perform tasks such as locomotion by controlling their actuators [2], [6], [7]. Another common style is the "lattice-based" modular robot, where overall shape change is achieved by modules changing their local connectivity [1]. More recently, several groups have proposed strut-based modular robot in which shape formation is achieved by modules self-deformation [8], [9].

Several groups have demonstrated centralized and decentralized control in modular robots [1], [2], [6], [7]. However, there are only a few that focus on self-adaptation tasks based on sensory feedbacks. In chain-based robots, Yim et al. demonstrate robot locomotion that conforms to the environment via a hand-designed gait table and distributed force feedback [6]. However, there is no theoretical guarantee for the control laws they propose. Another type of adaptive

locomotion strategy for chain-based robots is based on CPG. Kamimura et al. and Sproewitz et al. have demonstrated such an approach in the M-Tran [10] and YaMoR [11] modular robots, respectively. In a lattice-based system, Rus et al. have demonstrated distributed algorithms for locomotion over obstacles. Bojinov et al. presented control algorithms for several interesting examples that were tested in simulations: a hand that grasps an object and a table that adaptively supports a weight [12]. One major limitation of lattice-based systems in self-adaptive tasks is that shape change can only be achieved through module movement, which is slow in the hardware implementation. In this work, we mainly consider the chain-based system which can achieve fast adaptation.

Distributed consensus [4] has been widely applied in distributed (robotics) systems, including autonomous vehicle formation control [13], sensor network time synchronization [14], and the sensor coverage problem [15]. In most cases, agents observation space and control space are assumed to be the same[1]. In our previous work, we took a first step to generalize it to the heterogenous sensing/control space, but it was restricted to tasks that can be described by orientation sensor constraints [3]. In this work, we propose a generalized distributed consensus framework that can be applied to a variety of sensor-actuator networks.

Our work is also inspired by Hiroses work on a distributed feedback controller for snake robots [16]. He demonstrated that the snake robot can coil around an object using distributed controller and force feedback. When applied to a coiling task, our decentralized framework further allows each element of the robot to achieve equal pressure or any desired pressure distribution around the object. In addition, the control law provably converges to the desired state.

III. ROBOT MODEL

In this section, we describe the robot model and the capabilities assumed in our framework. Our primary focus is on modular robotic systems in which the whole robot is composed of many independent and autonomous modules. Nevertheless, this decentralized control framework is applicable to many other distributed robotic systems as long as the assumptions described in this section are satisfied.

In our model, each module is an *independent agent* that has computation, communication, and actuation capabilities.

---

[1]For example, in sensor networks time synchronization, an agent can observe its neighbors firing time and thus control its own firing time
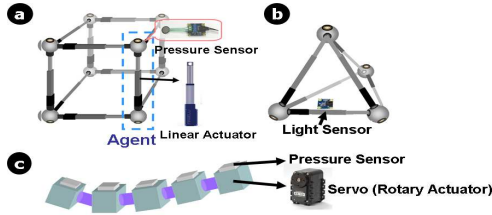
Fig. 2. Different modular robot configurations based on our robot model: (a) A pressure-adaptive column where each module is equipped with a linear actuator and a pressure sensor, (b) A tetrahedral robot. Each agent's configuration is similar to that of the pressure-adaptive column except that a light sensor is mounted on each surface to receive environmental trigger, (c) A modular gripper in which each module is composed of a rotary servo and a pressure sensor.

We henceforth refer to an autonomous module as an *agent* in the remainder of the paper. These agents can be reconfigured into different robotic systems. In this work, we assume agents have been connected into a certain configuration. They need to coordinate with each other to complete a desired task. We now describe the assumptions that each agent is assumed to satisfy, and we use three different modular robots (an adaptive column, a modular gripper, and a modular tetrahedral robot) that we built as examples.

Sensor: Each agent is equipped with one or more sensors suited to different robotics applications. Sensors are used to measure the current state of the agent. In the pressure-adaptive structure (Fig. 2 (a)), a pressure sensor is mounted on each agent. In the tetrahedral robot (Fig. 2 (b)), we also supply agents with light sensors to provide additional environmental triggers.

Actuator: Each agent is equipped with an actuator. We consider several types of actuators in our framework. In the pressure-adaptive structure and tetrahedral robots, each agent is equipped with a linear actuator. In the modular gripper, a rotary servo (AX-12) is mounted on each agent (Fig. 2 (c)).

Computation/Communication: Each agent is capable of performing simple computations such as addition and multiplication. Each agent is able to communicate with its immediate neighbors that are physically connected to it. Most of the current modular robots have these stated capabilities, e.g. Superbot [2], M-Tran [7], and Odin [8].

Task: The task is described as inter-agent sensor constraints. An agents task is complete when it has satisfied sensory state constraints between it and its neighbors. A consensus is formed when all agents have satisfied their constraints with their neighbors. In our framework, a task can be composed of one or more processes for reaching consensus

IV. GENERALIZED DISTRIBUTED CONSENSUS

In this section, we begin by briefly reviewing the standard distributed consensus algorithm. We then present a more general form of the algorithm and sufficient conditions for agents to reach consensus, as described in our previous theoretical study [5]. This generalized framework allows us

to extend the control law to a wide range of applications which we will cover in Section V.

A. Distributed Consensus

Distributed consensus is a process by which a group of networked agents come to a state of agreement by communicating only with *neighbors*. At each time step, each agent updates its new state according to the difference between its own state and its neighbors' states. This process can be formally written as:

$$x_i(t+1) \quad = \quad x_i(t) + \alpha \sum_{a_j \in N_i} x_j(t) - x_i(t) \qquad (1)$$

where a_i indicates agent i, and $x_i(t)$ and $x_i(t+1)$ are actuation states[2] of agent i at time step t and $t + 1$, respectively. N_i indicates the set of all one-hop neighbors of a_i. α is a small constant, and is sometimes called damping factor. There are two main assumptions buried in Eq. 1: First, each agent is capable of directly observing its state and its neighbors' states. Second, each agent is capable of freely driving itself to a new state $x_i(t+1)$.

B. Generalized Distributed Consensus Algorithm

In many cases, the mapping between sensor space and agent's actuation state is not precisely known. For example, in the modular gripper (Fig. 2 (c)), the mapping between the actuator's rotational angle and agent sensor value cannot be directly computed. In [5], we propose a more general form of the agent update equation:

ALGORITHM 1: **Generalized Distributed Consensus**

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} g(\theta_i, \theta_j). \qquad (2)$$

where θ_i is agent a_is sensor reading and θ_j indicates sensor reading of a_is neighbor, a_j. $g(\theta_i, \theta_j)$ is a sensory feedback function that agent a_i receives from its neighbor a_j. We note that Eq. 2 assumes agents perform update with round synchronization, i.e. each agent updates according sensory feedback at every time step. Let $T(\cdot)$ be a function that *maps the agent's actuation changes to sensor changes*. We show that $g(\cdot)$ can be *any function* satisfying the following conditions:

$$g(\theta_i, \theta_j) = 0 \Leftrightarrow \theta_i = \theta_j \qquad (3)$$
$$sign(T(g(\theta_i, \theta_j))) = sign(\theta_j - \theta_i) \qquad (4)$$
$$g(-\theta_i, -\theta_j) = -g(\theta_i, \theta_j) \qquad (5)$$

Intuitively, condition 1 (Eq. 3) means that g only "thinks" the system is solved when it actually is; condition 2 (Eq. 4) means that when not solved, each sensory feedback g at least points the agent *in the correct direction* to satisfy the local constraint with a neighboring agent; and condition 3 (Eq. 5) means that g is *anti-symmetric*.

In addition, we need to ensure that $\sum_{a_j \in N_i} \alpha \cdot \frac{g}{x_j(t) - x_i(t) - \Delta_{ij}^*} < 1$ holds for all a_i and for all t where Δ_{ij}^*

---

[2]If the agent's actuator is a linear actuator, $x_i(t)$ would represent the length of the actuator. If the actuator is a rotary one, it would represent the angle of the actuator

is the desired state difference between agents that achieves $\theta_j = \theta_i$. This will ensure agents' states from fluctuation while reaching the consensus state, and it is usually done by selecting an appropriate α constant and choosing g as a function that is proportional to distance from the desired state. This formulation is capable of being applied to a large class of distributed control tasks provided that one can create local agent rules that satisfy the conditions. In the next section, we describe three different generalizations and their applications.

V. SELF-ADAPTATION TASKS

We have described our robot model in Section III and sufficient conditions for reaching consensus in Section IV. When solving modular robot tasks, there are still two main challenges we need to address to apply this framework. First, we need to represent a new task in terms of inter-agent sensor difference or consensus. Second, we need to design an appropriate sensor feedback function g so that the conditions outlined in Section IV are provably satisfied.

In this section, we illustrate solutions to these challenges using three different example applications: (A) A pressure-adaptive column in which case each agent's sensor and actuator has an indirect relationship. (B) A modular gripper in which case each agent's actuator has a long range effect. (C) A modular tetrahedral robot which extends the agents' task space from forming a single consensus to a sequence of consensuses. In the last subsection, we further illustrate some other example tasks that can be formulated within our framework.

A. Pressure-Adaptive Column

One potential application for modular robotics is a reconfigurable structure: a structure that can reconfigure itself to achieve functional requirements irrespective of external environment changes. Examples include forming the supporting structure for a building that absorbs uniform force, and a modular seat back that adapts to apply uniform pressure on the user. Motivated by this application area, we construct a pressure-adaptive column with a modular robot.

As shown in Fig. 3 (1), each agent is equipped with a linear actuator whose length can be precisely controlled and a pressure sensor that can sense the force applied on each agent. We program agents to achieve a state where each agent absorbs equal force when an unknown object or structure is placed on it.

The algorithmic overview of the self-adapting process is shown in Fig. 3. **Step 1:** An unknown object is placed on the robot. **Step 2:** Each agent starts exchanging current pressure sensor feedback with its neighbors. **Step 3:** Each agent computes its actuators new parameters based on the sensor feedback that it receives from all its neighbors. Each agent iterates between Step 2 and Step 3 until the desired state has been reached: $\theta_i - \theta_j < \varepsilon$ where ε is a constant. When the environment starts changing again, the robot automatically goes back to Step 2.
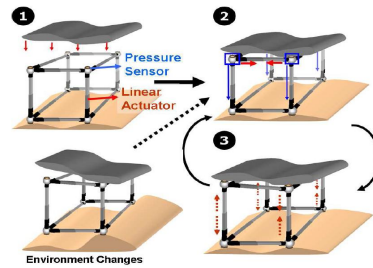

Fig. 3. The algorithmic overview of the pressure-adaptive column. Step 1: an unknown object is placed on the robot. Step 2: Each agent sends its pressure reading to neighbors. Step 3: Step 3: Each agent continuously adapts based on its neighbor's states.

In Step 3, each agent runs a control law to change the length of its linear actuator x_i based on sensory feedback from its neighbors. This control law can be written as:

$$x_i(t + 1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} (\theta_j - \theta_i) \tag{6}$$

Here, the feedback function g is simply $g(\theta_j, \theta_i) = \theta_j - \theta_i$. g satisfies conditions Eq. 3 – Eq. 5, since: (1) when $\theta_i = \theta_j$, $g(\theta_j, \theta_i) = \theta_j - \theta_i = 0$, (2) when sensory θ_i is smaller than θ_j, $g(\theta_j, \theta_i) > 0$ such that agent a_i increases its length to increase its pressure state θ_i. Therefore, $T(g(\theta_i, \theta_j))$ is moving in the same direction as $\theta_j - \theta_i$, (3) g function is anti-symmetric. Therefore, the control law (Eq. 6) will allow the robot to converge to the desired state. This leads to the following theorem:

Theorem 1: Let Θ^0 be the initial condition of the robot and Θ^* be the desired state, so that $\theta_j = \theta_i$ for every agent a_i and its neighbor a_j. If configurations between Θ^0 and Θ^* are reachable, the control law Eq. 6 will lead all agents to converge to Θ^* at an exponential rate.

Proof: see [17]. ∎

B. Modular Gripper

In this section, we illustrate another application a modular gripper. The gripper is capable of reconfiguring itself to grasp an object using distributed sensing and actuation. The control law design follows a similar procedure as in the previous example. However, the analysis of the convergence property is somewhat different due to the fact that each agents actuation affects more than its own sensor state.

As shown in Fig. 4 (1), a modular gripper is composed of a chain of modular agents, where each agent is equipped with a rotary servo and a pressure sensor. The goal of the agents is to grasp a convex object, e.g. a balloon, such that all of the agents apply equal pressure $\theta_p(\theta_{\min} \leq \theta_p \leq \theta_{\max})$.

The illustration of the algorithmic procedure is shown as Fig. 4. It can be divided into the following steps:

Step 1: One of the agents starts sensing the object. When the sensor reading is in between $\theta_{\min}$ and $\theta_{\max}$, it starts sending messages to neighboring agents. Upon receiving a message, each agent propagates the message and its ID to neighboring agents (shown in Fig. 4 (1)). We denote R_i as the agent ID from which agent a_i receives the message and S_i as the ID of the agent to which it sends the message
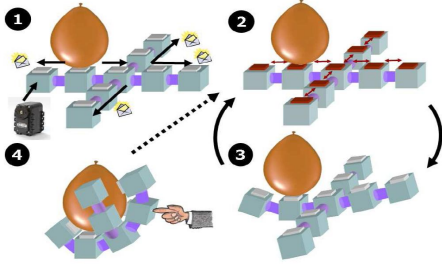
Fig. 4. The algorithmic overview of the grasping task. Step 1: The first module starts sensing the presence of the object. It starts sending messages to it neighbors. Steps 2 and 3: Agents perform iterative sensing and actuation until they converge to the desired state. Step 4: When the robot is perturbed by exogenous force, it goes back to Step 2.
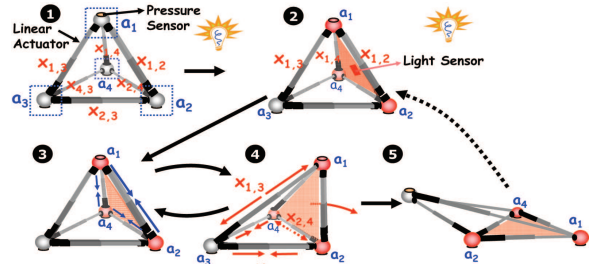

Fig. 5. An overview of the tetrahedral robot locomotion task. Step 2: one of the surfaces is triggered by light. Step 3: the activated agents exchange sensor readings with their neighbors. Step 4: Agents control actuators to let the triggered surface lean forward until agents reach the pressure consensus state. Step 5: The pressure consensus state has been reached, the robot goes back to the default configuration, and a new surface is triggered.

Step 2: Each agent starts sending its pressure sensor reading to its neighbors (as shown in Fig. 4 (2)). We note that this sensory reading message is passed only between an agent and its immediate neighbors.

Step 3: Each agent computes its new actuation state based on the sensor readings that it receives from its neighbors. The control law run by each agent is:

$$x_i(t+1) = x_i(t) + \alpha \cdot (\theta_{R_i} - \theta_i) \qquad (7)$$

Agents iterate between Step 2 and Step 3 until all agents have reached the desired state. When the robot is perturbed by exogenous force, it goes back to Step 2.

The control law we showed in Eq. 7 satisfies condition 1, since sensory feedback $g(\cdot) = \theta_{R_i} - \theta_i = 0$ only when agent a_i's sensor reading equals to its neighbor a_{R_i}. In addition, g is also anti-symmetric. However, it is nontrivial to evaluate whether the control law satisfies condition 2. This is primarily due to the fact that all agents are connected together in a chain and changing an agent's actuation parameter can potentially change more than its own sensor state. The details of the proof are in the Appendix. We can state the following theorem for the Eq. 7 control law:

Theorem 2: Let Θ^0 be the initial sensory condition of the gripper and Θ^* be the desired sensory state such that $\theta_j = \theta_i$, for each agent a_i and its neighbor a_j. If configurations between Θ^0 and Θ^* are reachable, the control law Eq. 7 will lead the gripper to apply uniform pressure on the object, and the system's state will converge to Θ^*.

Proof: see [17]. ∎

Most of the controllers designed for grasping tasks have used a centralized architecture. The decentralized and modular robot approach that we propose here allows the whole system to adapt to local perturbations more efficiently. In addition, given any initial contacting module, the gripper is able to form a grasping configuration that conforms to the shape of the object. This control scheme is also applicable to different kinds of gripper configurations. We will provide demonstrations of these capabilities in Section VI.

C. Modular Tetrahedral Robot Locomotion

So far, we have presented generalizations in forming a single consensus state between agents. In this section, we show how agents can achieve more complicated tasks by forming a sequence of consensus states. We demonstrate a modular tetrahedral robot is capable of performing locomotion towards a light source with a sequence of such tasks. This approach can be potentially applied to many other modular robot locomotion tasks.

As shown in Fig. 5 (1), an agent is equipped with a pressure sensor and is capable of controlling actuators that are connected to it. We denote x_{ij} as the linear actuator mounted between agent a_i and a_j. In the example in Fig. 5, agent a_1 can control actuators $x_{1,2}$, $x_{1,3}$, and $x_{1,4}$. In addition, a light sensor is mounted on each surface of the tetrahedron (Fig. 5 (2)), and agents on the surface can access the sensor reading. At each locomotion step, a subset of agents is selected to form consensus with the light trigger. The selected agents perform actuation to achieve nearly equal pressure readings. The detailed steps are as follows:

Step 1: Each agent starts passing messages to its neighbors, allowing it to identify its neighboring agents and the linear actuators between them.

Step 2: The surface that is closest to the light source is triggered[3]. We denote the subset of agents on the triggered surface as Ω. In our example of Fig. 5, $\Omega = \{a_1, a_2, a_4\}$.

Step 3: The activated agents start sending pressure readings to their other activated neighbors.

Step 4: We denote linear actuators that are on the triggered surface as surface actuators, and those attached to the triggered surface as supporting actuators; e.g., agent 1 in Fig. 5 (2) has surface actuators x_{12} and x_{13} and supporting actuator x_{14}. In this step, each agent actuates the supporting linear actuator (the linear actuator that it is connected to but not on the triggered surface) by running the following control law:

$$x_{ik}(t+1) = x_{ik}(t) + \alpha \cdot \sum_{a_j \in N_i \cap a_j \in \Omega} (\theta_j - \theta_i) \qquad (8)$$

---

[3]In our implementation, we set a threshold $\bar{\theta}$ to determine whether a surface is triggered. In a tetrahedral structure, only one surface is nearly perpendicular to the light direction, so only one surface will be triggered. In some other structures where ambiguities might arise, a maximum value consensus algorithm can be run on them to identify which surface is to be triggered.

where x_{ik} is a_i's supporting actuator. This control law will allow the activated surface to lean forward until the tetrahedron rolls over to put all three activated agents in contact with the ground. In our Fig. 5 example, this is achieved with $x_{2,3}$ and $x_{4,3}$'s contraction and $x_{1,3}$'s expansion. In our hardware implementation, all actuators are fully contracted in the default state, so $x_{2,3}$ and $x_{4,3}$ are not able to further contract. Alternatively, we program agents that have contacted with ground (a_2 and a_4) to actuate the surface actuator between them $(x_{2,4})^4$. Agents iterate between Steps 3 and 4 until they converge to the consensus state.

Step 5: The consensus state is formed when all activated agents have contacted the ground and $\|\theta_j - \theta_i\| \leq \epsilon$ for all agents a_i and their neighbors $a_j{}^5$. After agents have achieved consensus, they reset to the default configuration (Step 2), and a new surface is triggered.

The verification of sufficient conditions for reaching consensus with the control law Eq. 8 is similar to that of Eq. 6. To avoid repetition, we omit the details here.

The generalization of single consensus formation to a sequence of consensuses allows this framework to extend from solving static shape/structure adaptations to dynamic tasks such as locomotion. Utilizing agents' sensor consensus provides a way for modular robots to adapt to different environmental conditions. In the tetrahedral robot example, the cycle time of locomotion is determined by the pressure states of the agents. When the environmental condition allows agents to reach consensus state sooner, e.g., when the robot is on a steeper slope, the locomotion cycle time will adapt to become shorter.

D. Other Applications & Potential Extensions/Limitations

There are many potential applications that can be generalized from this framework. Here we illustrate some of them: (1) Light-adaptive modular panel: We can change the pressure sensors we mount on the robot to light sensors. Each agent is programmed to achieve the same light absorption as its neighbors. A similar concept can be applied in many environmental sensory adaptation tasks. (2) Adaptive prosthetic structure: Existing prosthetic devices for children require manual reconfiguration to adapt to limb growth. If force (pressure) sensors are mounted on the device, it is possible to construct a self-reconfigurable prosthetic device. (3) A similar concept can be applied to a support structure for plants. The structure is capable of self-adaptation based on the growth of the plant and lighting conditions. (4) In the dynamic task domain, robotic systems that locomote by shape/structure deformation can potentially apply our

---

[4]The modified control law is:

$$x_{il}(t+1) = x_{il}(t) - \alpha \cdot \sum_{a_j \in N_i \cap a_j \in \Omega} (\theta_j - \theta_i)$$

where a_i and a_l are agents have contacted with ground. We note that this control law will allow surface actuator x_{il} to expand, achieving the same effect as contracting two supporting linear actuators.

[5]In this application, we allow larger ϵ to identify whether consensus has been formed.
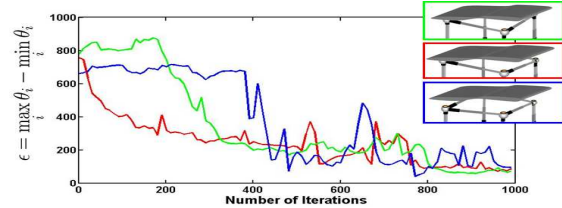

Fig. 6. Pressure-adaptive column with different initial conditions. We let the number of initial contacting agents to be one (green curve), two (red curve), and three (blue curve) respectively and examine how the column respond with different initializations. After 1000 iterations of running the control law, the distance measure, ϵ, decreases to less than 100.

framework for locomotion tasks, e.g., an amoebic modular robot and a cubic modular robot.

The type of tasks we illustrate with this framework share with one similarity: they can be expressed as a single consensus state, e.g. modular gripper grasping tasks, or a sequence of identical consensus states, tetrahedral robot's locomotion. To further extend such a framework to solve more sophisticated tasks, it is necessary to have the mechanism that can decide *different* consensus states (it is also called *biased* consensus states [5]) based on different external states. For example, we might need different pressure distributions for modular gripper to optimally grasp different types of objects, instead of using uniform pressure distribution to grasp all objects. Understanding the scope of self-adaptation tasks this limitation is an important future direction of this research.

VI. EXPERIMENTAL RESULTS

In this section, we present experimental results of applying this framework in three different real robots. Our results show that our decentralized control approach is able to cope with real world sensing and actuation noise to achieve self-adaptation tasks. In the pressure-adaptive column experiments, we show that agents are capable of converging to an equal pressure state irrespective of different initializations when an unknown object is placed on it. In the modular gripper experiments, we show that our control law is capable of leading agents to grasp around a balloon while applying equal pressure on it. Furthermore, agents are capable of achieving the desired state regardless of initial contact locations. They can also maintain the desired state when facing exogenous perturbations. In the modular tetrahedral robot experiments, we show that our robot is capable of moving toward a light source through a sequence of consensus formation processes.

A. Pressure Adaptive Column

In this experiment, we examine the control laws convergence property with different initial conditions. Each agent is equipped with a pressure sensor (force sensing resistor) with sensory readings ranging from 0 to 900. Agents are programmed to achieve equal pressure with their neighbors. The weight of the unknown object is roughly 1.5 pound. The robot starts in three different configurations, such that the number of initial contacting agents is different, ranging from
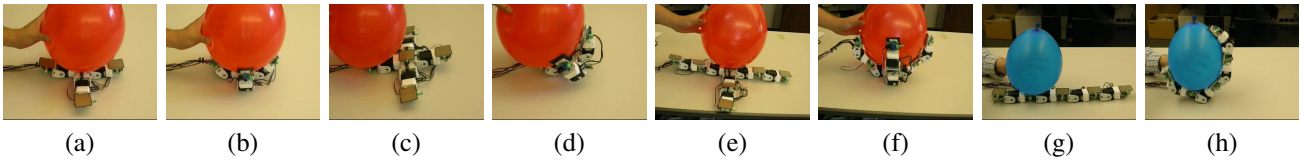
Fig. 7. (a-d) Different initial conditions for the grasping task. The robot is capable of completing the task irrespective of initial conditions. (e-f) Scalability experiment. More modules are added to the robot. Empirically, the robot scales successfully to the number of module agents. (g-h) The robot performs the grasping task with a different gripper configuration.
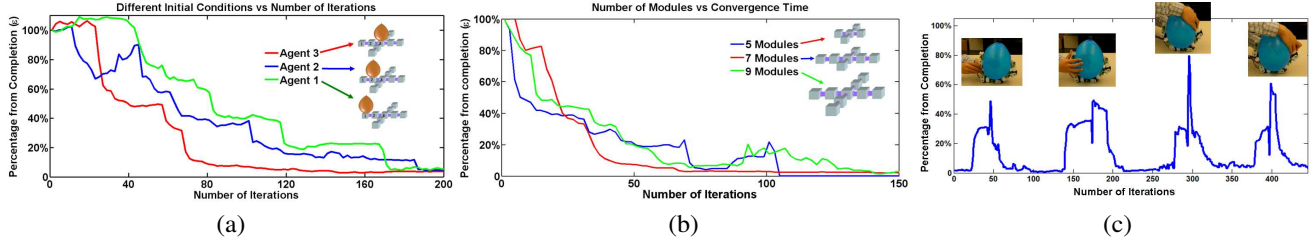

Fig. 8. (a) Experiments with different initial conditions. After ~ 180 iterations, agents are capable of achieving less 3% distance from the consensus state in all three cases. (b) Scalability experiment. The decentralized algorithm is scalable with the number of agents. On the other hand, the network structure might affect the convergence speed. In the 7 and 9 agents cases, the diameter of network is 2 and it leads to longer time for the robot to complete the task. (c) After the robot has reached the desired state, we constantly perturb the gripper by applying force. The robot is able to re-adapt after being perturbed.

one to three[6]. $\epsilon = \max_i \theta_i - \min_i \theta_i$, the difference between maximal and minimal sensory reading among agents, as a measure of distance from reaching consensus. We can see from Fig. 6 that ϵ decreases from 800 to around 100 after 1000 iterations (~ 10 sec. in real time) in all three cases. We note that the sensor we use is very noisy and sensitive to slight perturbations of the linear actuators. Therefore, we set the α in Eq. 6 to be a very small constant to avoid the column from being over-sensitive to perturbations. This naturally leads to a longer convergence time. We also note that the larger fluctuations in the blue curve is primarily due to the object significantly shifted its center of mass when more agents contact it.

B. Modular Gripper

Here, we present an empirical evaluation of this control framework when applied to a modular gripper. The robotic configuration of the gripper was described in Section V-B. Agents are programmed to apply equal pressure on a balloon. We test Eq. 7's convergence properties under different initial conditions and different numbers of agents. We also assess its adaptability towards repetitive perturbations.

Different Initial Conditions: We connect the agents to form a cross configuration as shown in Fig. 7. We let different agents start to touch the balloon to examine the systems behavior under different initial conditions. Fig. 7 (a-h) shows a sequence of robot configurations while grasping the object. We use k to denote the first activated (contacted) agents index. Let $\theta_i(t)$ be the pressure sensor reading of agent i at time t. After the first contact between the object and the robot, the object is held in place. This will lead all other agents to approach agent a_ks sensor reading $\theta_k(t)$ while reaching the consensus state. Therefore, we define

[6]In the case of one or two initial contacting agents, we provide slight external support to the object to prevent the rest of the agents from contacting the object.

the *percentage from achieving the task*, ϵ, as a ratio of the current distance for all agents to reach the first contacted agents sensor reading $\theta_k(t)$ to the initial distance. This can be formally written as: $\epsilon = \frac{\sum_i \|\theta_i(t) - \theta_k(t)\|}{\sum_i \|\theta_i(0) - \theta_k(0)\|}$. Fig. 8 (a) shows ϵ's value changing over time. We can see that the agents are capable of converging to $\sim 3\%$ from completing the task after 180 iterations, regardless of initial conditions. From this figure, we can also see that there is a correlation between the position of the first activated agent and the convergence time. The red curve shows the case when the middle agent is first activated. The maximum communication hop between it and all other agents is two. In this case, agents achieve faster convergence as compared to the case where the maximal hop is three and four respectively (blue and green curve).

Scalability: We further evaluate the algorithms scalability towards the number of agents. In Fig. 8 (b), we increase the number of agents from 5 to 9. We can see from the figure that there is no significant increase in convergence time when we increase the number of agents. ϵ converges to less than 3% after 150 iterations in all three cases. However, we can see that the convergence time is slightly shorter in the 5-agent case in which the diameter of the agent network is only one (in contrast to two in the other cases). This coincides with our previous theoretical result [5] that decreasing the diameter of the agent network can increase convergence speed.

Adaptation Towards Perturbations: After all agents achieve the desired state, we start applying an external force on the gripper. Fig. 8 (c) shows ϵ vs time as the gripper encounters four different perturbations. We can see that decreases to less than 3% after $50 - 70$ iterations in each case. This shows that our decentralize control law can efficiently lead agents recover from exogenous perturbations. We specifically note that the gripper achieves faster adaptation than the pressure-adaptive column is due to: (1) Each agents actuation has a long range effect, an agent is likely to assist
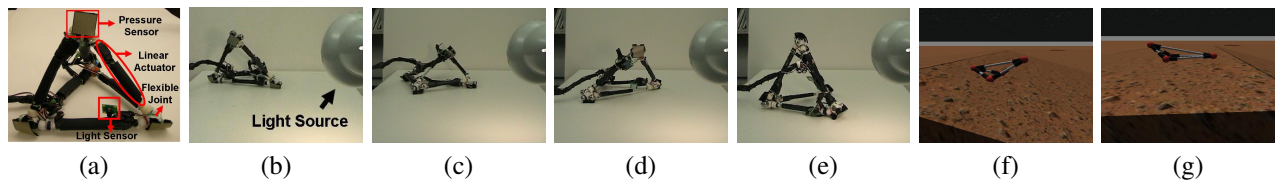
Fig. 9. (a) The tetrahedral robot configuration. Each agent is equipped with a pressure sensor. A linear actuator is mounted between each pair of neighboring agents. Flexible joints allow the robot to deform its shape. A light sensor is mounted on each surface of the tetrahedral. (b-e) Tetrahedral robot moving towards the light source. The average locomotion cycle time is ~ 5 seconds, and the robot is capable of moving at a speed of 10cm/sec. (f-g) Tetrahedral robot's locomotion cycle autonomously adjusts with on different slopes. The locomotion cycle time becomes longer when the robot is on a inclining slope (f) while the cycle becomes shorter when the robot is on a declining slope (g).

more than its neighbors in the process. (2) The rotary servos we use here has better precision than the linear actuators.

C. Modular Tetrahedral Robot

We also implement the sequential consensus formation process as described in Section V-C on a tetrahedral robot. As shown in Fig. 9 (a), each agent is equipped with a pressure sensor, and each surface has a light sensor. Figelli L12 linear actuators are mounted between agents, and the actuators maximal speed is 2.3 cm/sec. We also create flexible joints on the connecting points between linear actuators and agents to allow deformation of the tetrahedron. The height of the tetrahedron is ~ 20 cm. Fig. 9 (b-e) shows a sequence of the robots locomotion actions. We note that due to mechanical restrictions[7], we place the robot on a slope of roughly 10 degrees. This allows the robot to roll over more easily. As shown in Fig. 9, agents on the surface that is closest to the light source are activated in each cycle. The average locomotion cycle time is 5 sec, and the robot is capable of moving towards the light source at a speed of 10 cm/sec. We are also interested in further exploring how different terrain conditions might affect the cycle of locomotion.

We also constructed a simulation environment using open dynamics engine to examine how tetrahedral robot's locomotion adapts to terrains of different slopes. When the robot is placed on a declining slope of -54 degree (Fig. 9 (f)), the locomotion cycle time has become much shorter: its average cycle time is 48% of the average cycle time when it locomotes on a $+18$ degree inclining slope (Fig. 9 (g)). The robot travels in a more efficient way when it can exploit gravity to assist locomotion.

VII. CONCLUSIONS

We have presented a generalized distributed consensus framework for self-adaptation tasks in modular robotics. We also demonstrated three example applications in hardware using this framework, including (1) a pressure-adaptive column; (2) an adaptive modular gripper; (3) a modular tetrahedral robot. We also show that the proposed control laws are provably correct and robust toward different initial conditions and constant perturbations. These applications represent a small set of what is achievable within this framework.

We plan to extend this work in several directions. First, we have illustrated several potential applications to which

we can further apply this framework, examples include a self-adaptive support structure. Second, we are interested in applying this framework in other distributed robotics applications beyond modular robots, e.g. on a team of mobile robots. Finally, we are interested in exploring a mixed strategy that is composed of centralized and decentralized controllers. For example, a humanoid robot utilizes a centralized controller to reach an object and decentralized controllers run on the gripper allowing it to grasp around the object.

REFERENCES

[1] D. Rus, Z. Butler, K. Kotay, and M. Vona, "Self-reconfiguring robots," *Communications of the ACM*, vol. 45, no. 3, pp. 39–45, 2002.
[2] W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh, "Multimode locomotion for reconfigurable robots," *Autonomous Robots*, vol. 20, no. 2, pp. 165–177, 2006.
[3] C.-H. Yu, F.-X. Willems, D. Ingber, and R. Nagpal, "Self-organization of environmentally-adaptive shapes on a modular robot," in *Proc. IROS*, 2007.
[4] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proc. of IEEE*, 2007.
[5] C.-H. Yu and R. Nagpal, "Sensing-based shape formation tasks on modular multi-robot systems: A theoretical study," in *Proc. of AAMAS*, 2008.
[6] M. Yim, C. Eldershaw, Y. Zhang, and D. G. Duff, "Limbless conforming gaits with modular robots," in *Proc. of ISER*, 2004.
[7] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME Trans. Mechatron*, vol. 7, no. 4, pp. 431–441, 2002.
[8] A. Lyder, R. Garcia, and K. Stoy, "Mechanical design of odin, an extendable heterogeneous deformable modular robot," in *Proc. of IROS*, 2008.
[9] C.-H. Yu, K. Haller, D. Ingber, and R. Nagpal, "Morpho: A selfdeformable modular robot inspired by cellular structure," in *Proc. of IROS*, 2008.
[10] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Distributed adaptive locomotion by a modular robotic system, m-tran ii," in *Proc. of IROS*, 2004.
[11] R. Moeckel, C. Jaquier, K. Drapel, A. Upegui, and A. Ijspeert, "Yamor and bluemove - an autonomous modular robot with bluetooth interface for exploring adaptive locomotion," in *Proc. of CLAWAR*, 2005.
[12] H. Bojinov, A. Casal, and T. Hogg, "Emergent structures in modular self-reconfigurable robots," in *Proc. of ICRA*, 2000.
[13] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. on Automatic Control*, 2002.
[14] D. Lucarelli and I. Wang, "Decentralized synchronization protocols with nearest neighbor communication,," in *Proc. of Sensys*, 2004.
[15] M. Schwager, J.-J. E. Slotine, and D. Rus, "Consensus learning for distributed coverage control," in *Proc. of ICRA*, 2008.
[16] S. Hirose, *Biologically-Inspired Robots*. Oxford Sci. Pub., 1993.
[17] Online Appendix URL: http://www.eecs.harvard.edu/~chyu/icra09-appendix.pdf.

---

[7]Our linear actuator can only perform up to 80%'s extension. This restriction can be removed if the actuation range is longer.