Proceedings of the IEEE 2010 International Conference on Robotics and Automation workshop

“Modular Robots: The State of the Art”

3rd of May 2010, Alaska, AK, USA

Editors

Kasper Stoy
Radhika Nagpal
Wei-Min Shen
Preface

It is a pleasure to present the contributions to the IEEE 2010 International Conference on Robotics and Automation Workshop on Modular Robots: State of the Art. The contributions as a whole gives a wonderful picture of the wealth of ideas currently being explored by the community and is an indication of the high level of activity in the field of modular robots.

The contributions contained within are not conventional research papers since they are accepted based on interest and relevance and not on their scientific merits. The idea behind this, and the workshop in general, is to allow the community to discuss work in progress, which allows participants to get early feedback on their work and avoid well-known problems or be made aware of well-known solutions already known to the community. The overarching goal of this is to accelerate research on modular robots and realize their great potential!

We would like to thank everybody who made a contribution to this workshop and encourage you to continue to participate actively in the modular community in the future.

Sincerely,

Kasper Stoy, University of Southern Denmark

Radhika Nagpal, Harvard University

Wei-Min Shen, University of Southern California's Information Sciences Institute
# Table of Contents

## Overview

Multi-Robot Organisms: State of the Art - 1  
Serge Kernbach, Oliver Scholz, Kanako Harada, Sergej Popesku, Jens Liedke, Humza Raja, Wenguo Liu, Fabio Caparrelli, Jaouhar Jemai, Jiri Havlik, Eugen Meister, and Paul Levi

CKBot Platform for the ICRA 2010 Planetary Challenge - 11  
Shai Revzen, Jimmy Sastra, Nick Eckenstien, and Mark Yim

Recent Progress of SuperBot - 13  
Wei-Min Shen, Feili Hou, Mike Rubenstein, Harris Chiu, and Akiya Kamimura

## Robot Systems

Mechanical Design and Locomotion of Modular Expanding Robots - 17  
Rebecca Belisle, Chih-han Yu, and Radhika Nagpal

Increased Versatility of Modular Robots through Layered Heterogeneity - 24  
Jorgen Christian Larsen, Ricardo Franco Mendoza Garcia, Kasper Stoy

Cellular Slime Mold Robot - 30  
Eunjeong Lee, Jesse Yang, Matthew Jolda, and Robert Wood

Actuated Responsive Truss - 36  
Rehman Merali and David Long

Development of a Reconfigurable & Modular Mobile Robotic Platform - 41  
Taskın Padır, Matthew R. Bienia, Daniel R. Garcia, and Karl B. Wajcs

Kilobot: A Robotic Module for Demonstrating Behaviors in a Large Scale (210 units) Collective - 47  
Michael Rubenstein and Radhika Nagpal

## Mechatronics

Modular Fault Diagnosing for Robotic Arms - 52  
Hamid Abdi and Saeid Nahavandi

A Vacuum-Based Bonding Mechanism for Modular Robotics - 57  
Ricardo F. M. Garcia, Jonathan D. Hiller, and Hod Lipson
Towards the Sense of Touch in Snake Modular Robots for Search and Rescue Operations - 63
Juan Gonzalez-Gomez, Javier Gonzalez-Quijano, Houxiang Zhang, and Mohamed Abderrahim

A Parallel Wireless Radio Communication Architecture for Modular Robots - 69
Victor Kuo and Robert Fitch

Genderless Connection Mechanism for Modular Robots Introducing Torque Transmission Between Modules - 77
Andreas Lyder, Ricardo Franco Mendoza Garcia, and Kasper Stoy

Theory, Software and Control

Characterization of Lattice Modular Systems by Discrete Displacement Groups - 82
Nicolas Brener, Faiz Ben Amar, and Philippe Bidaud

Novel Modular CPG Topologies for Modular Robotics - 89
Fernando Herrero-Carrón, Francisco de Borja Rodríguez, and Pablo Varona

Kyle Gilpin and Daniela Rus

Reconfigurable Software for Reconfigurable Modular Robots - 100
David Ko, Harry. H. Cheng, and Graham G. Ryland

Graph Minor Analysis of Reconfiguration State Spaces - 107
Tom Larkworthy and Subramanian Ramamoorthy

A Hybrid Control Strategy for a Chain type Modular Robot - 111
Rodrigo Moreno and Jonatan Gomez
MULTI-ROBOT ORGANISMS: STATE OF THE ART

Serge Kernbach¹, Oliver Scholz², Kanako Harada³, Sergej Popesku¹, Jens Liedke⁴, Humza Raja², Wenguo Liu³, Fabio Caparrelli⁴, Jaouhar Jemai³, Jiri Havlik⁵, Eugen Meister¹, and Paul Levi¹

¹Institute of Parallel and Distributed Systems, University of Stuttgart, Germany,
²Fraunhofer Institute for Biomedical Engineering, Sankt Ingbert, Germany,
³Center for Applied Research in Micro and Nano Engineering, Scuola Superiore Sant'Anna, Italy,
⁴Institute for Process Control and Robotics, Universität Karlsruhe (TH), Germany,
⁵Bristol Robotics Laboratory (BRL), UWE Bristol, UK,
⁶Materials and Engineering Research Institute, Sheffield Hallam University, UK,
⁷Ubisense AG, Munich, Germany, ⁸IMA, s.r.o., Prague, Czech Republic.

Abstract—This paper represents the state of the art development on the field of artificial multi-robot organisms. It briefly considers mechatronic development, sensor and computational equipment, software framework and introduces one of the Grand Challenges for swarm and reconfigurable robotics.

I. INTRODUCTION

Appearance of multicellular structures is related to one of the greatest moments in the history of life [1]. The rise of multicellular from unicellular is a huge evolutionary step, however we do not exactly know how multicellular organisms appear and which mechanisms take part in this phenomenon. We know multicellular organisms are self-adaptive, self-regulative and self-developing, however we do not know its evolutionary origin and developmental organization. The great vision, which consolidates many interdisciplinary researchers, is a vision of self-adaptive, self-regulative and self-developing robots that reflect multicellularity in nature – a vision of artificial robot organisms [2]. Like multicellular beings, these artificial organisms consist of many small cell-modules, which can act as one structure and can exchange information and energy within this structure. Moreover, these structures can repair themselves and undergo evolutionary development from simple to complex organisms [3].

Technological exploitation of multicellularity provides different practical advantages not only for advanced robotics, but also for autonomous and adaptive systems in general. Three most important advantages are extended reliability, advanced adaptivity and self-evolving properties. Reliability in general context is related to the ability of a system to work durably in different hostile or unexpected circumstances. Artificial organisms can self-disassemble, the destroyed cell-modules should be removed, and then an organism self-assembles again. Capabilities of basic robot modules for autonomous self-assembling and for dynamic change of functionality are key points of the extended reliability. Adaptivity is another key feature of advanced autonomous systems and indicates an ability of a system to cope with a changing environment. Multicellularity introduces a new component into adaptive processes – morphogenesis – the self-development of structure, functionality and behavior during a life cycle of the organism. Both reliability and adaptivity mean a high development plasticity, where an organism can dynamically change itself, modify its own structural and regulatory components. As observed in nature, the developmental plasticity is a necessary condition for evolutionary processes – such processes, which can potentially make a system more complex, increase information capacity and processing power [4].

Exploration of these issues represent a challenge for researchers and engineers. It is firstly related to a good engineering of mechatronic cell-modules, which should demonstrate 2D locomotion on a surface, 3D actuation within a heavy organism, autonomous docking to each other, large on-board energy resources, different sensors and sufficient computation/communication. Of utmost importance is that the modules should be small in size and light in weight. Not only mechatronics, but also software engendering and design of control and regulative structures are of essential importance. This paper is basically devoted to these challenges and represent a snapshot of the research and technological development conducted within the European projects “SYMBRION” [5] and “REPLICATOR” [6].

The paper is organized in the following way. The Sec. II introduces development of heterogeneous reconfigurable platforms. Sec. III treats issues of general architecture, computational power and on-board sensors, whereas Sec. IV briefly considers the software framework. Finally, Sec. V introduces one of the Grand Challenges and Sec. VI concludes this work.

II. MECHATRONIC PLATFORMS

The mechanical characteristics and functionalities of individual robots in a collective symbiotic system are of the utmost importance in order to confer suitable capabilities to the symbiotic robot organisms. However, this does not necessarily mean that the design of individual robots has to
be particularly complex from a mechanical point of view. On the contrary, excessive complexity can lead to several disadvantages in the assembled state of the organism, e.g. higher risk of failures and higher electrical and computational power demand. In addition, considering the manufacturing phase of the individual robots themselves, complexity would lead to high development and assembling costs; this is an issue particularly relevant when a large multi-agent symbiotic system is targeted. Finally, considering miniaturized robots, there are severe volume constraints at the design level that may prevent the possibility to integrate complex mechanisms. Consequently, as a rule of thumb, the individual robots of a large collective symbiotic system can be designed to offer the minimal mechanical functionalities able to allow the symbiotic robotic organism to assemble and develop all those collective configurations and reconfiguration strategies that let specific collective functionalities emerge. That’s inevitably a compromise choice in the design.

As already mentioned, a symbiotic robot organism can be seen as the physical evolution of a swarm system of individual robots into a structural system of connected robots. From this “structural” perspective, the mechanical functionalities of the individual robot could correspond to the behavioral rules of the agents in a swarm system that generates collective emergent behaviors. The mechanical interactions between the robots assembled in the organism expand consequently the collective capabilities of the system to a structural dimension.

On the base of the above considerations, it is clear how the design of suitable mechanical features of the individual robots represents a critical issue. In particular, the robot-to-robot connection mechanisms (docking mechanisms) and the mechanical degrees of freedom implemented in the individual robots deserve a deep investigation.

A. A Heterogeneous Approach in Modular Robotics

The design of each individual robot as a stand-alone unit inevitably ends to favor specific functional characteristics such as locomotion capability, actuation power and robustness, and this can result in multiple design solutions. This is true especially for miniaturized individual robots because focusing on one feature means finally to degrade or loose other features due to obvious space constraints. As a consequence of the above mentioned issues, the design process can follow different paths:

- To try to merge the best features of all the conceived designs into a unique individual robot design by accepting performance compromises of the collective system while making the control of the organism easier. We refer to such a system as collective homogeneous system. This is the path mostly followed by state-of-the-art modular and reconfigurable robotics ([7], [8], [9], [10], [11], [12], etc.).
- To consider having two or more different individual robot types where each robot is optimised for specific functions. Each robot can assemble into a symbiotic organism by means of compatible docking units, thus empowering the global capabilities of the collective system in detriment of more complex control of the symbiotic organism due to its heterogeneity. We refer to such a system as a collective heterogeneous system as introduced in [13].
- To integrate “tool modules” with the above mentioned collective homogeneous system. Tool modules can be generally defined as devices whose functions are dedicated to a specific task. The tool modules can simply dock with the assembled organism, receive commands from the organism and possibly send data to the organism. These tool-modules could be, for instance, wheels, sensors, grippers, etc. By following this path, the system has to accept poor integration of the robot in favor of versatility. This approach is considered to be the evolved version of the collective homogeneous system as demonstrated in [14].
- To integrate “tool modules” with the above mentioned collective heterogeneous system. The main structure of the organism is composed of two or more different individual robots and the organism can be equipped with “tool modules”. The heterogeneity of the system becomes high, making the control more complex. The system is the most versatile and robust to the environment and given tasks. This is a rather new approach in modular robotics as studied in [15], [3], [6].

Taking inspiration from the biological domain, it could be observed that natural swarms are often heterogeneous not only for the different behavioral specialization of each swarm member but also from a strict physical viewpoint (e.g., in a same colony there are insects with different physical capabilities, e.g. in ant colonies). However, differently from natural insect swarms, the conceived collective system should also be able to reach a collective structural level. This goal can be more complicated with heterogeneous individual robots, regarding the assembly process itself and, even more, for what concerns the onboard software (e.g., the self-learning and behavioral control of the symbiotic organism). As a case study, two individual robots, namely a Scout robot and Backbone robot, and one tool module, namely Active wheel, will be described hereafter and shown later in the chapter:

- A “scout” robot equipped with far-range sensors and above all specialized in fast and flexible locomotion that can be used for inspection of the environment and for swift gathering of robots for the assembly. For this purpose, wheeled/caterpillar-like locomotion is advantageous, in particular where challenging terrains have to be engaged. Actuators for the 3D actuation within the organism is mandatory but less powerful actuators are sufficient. It is because the scout robots can be useful when they are docked to the end of a leg or arm of the organism to scan the environment.
- A “backbone” robot, strong in main actuation and stiff in design. The main purpose of this robot is to work as a part of the organism, therefore the casing is strong to provide high stability and the main actuator is able to
lift several docked robots to perform 3D motion. The space for 2D locomotion is limited due to the large main actuator, but the 2D locomotion drive is capable of necessary movements for assembly and docking. In addition, the design of the robot allows to use the single DOF of the main actuator for either bending or rotation of the docked joint. Therefore, the powerful actuation is available for any joint in the assembled organism.

- An “active wheel” module as a tool module. Tool modules are optimised for specific functions and designed in a way to compensate aforementioned deficits of the individual robots. The Active wheel, for example, provides the ability to move omnidirectional, lifting and carrying heavy loads (i.e. other robots or organisms) and at the same time is able to provide an additional energy source. This tool can act in standalone mode as well as in organism mode.

The prototypes of the Backbone robot, the Active wheel and the Scout robot are shown in Fig. 1.

Following the general issues introduced above, several technical key aspects have to be taken in consideration in the mechanical design of the individual robots. The requirements and solutions of the Scout robot, the Backbone robot and the Active Wheel have been defined as shown in Table I.

### B. Locomotion Mechanisms of Backbone and Scout Robots

The locomotion capability allows the individual robots to be active in the environment, carrying on tasks of exploration, for instance. The locomotion capability is evidently fundamental when docking with other robots is necessary in order to reach the symbiotic state. Several approaches can be followed for the design of locomotion mechanisms, depending on the requirements that the individual robots and the symbiotic organism have. In classical modular robotics, the individual robot or module has been considered as a part of the modular system, thus it does not have any mechanisms that let it move as a stand-alone system. Instead, locomotion has generally been considered as a capability of the assembled robot and achieved by means of coordinated actuation among the docked modules in order to realize snake-like locomotion, legged-base walking, etc. This can limit the exploration capability of the whole system to the assembled state. In other words, individual robots or modules need to be manually positioned and docked before initiating the operation. When additional modules are requested by an assembled robot at the operation site, the assembled robot needs to go back to a specific zone where individual modules are deployed, or another assembled robot needs to be formed to reach the operation site. Hence, it is a natural consequence to try to devise individual locomotion solutions on each individual robot. This would guarantee the collective system much higher independence, versatility and flexibility. The system can be autonomous and robust especially in an unknown environment where the number of required robots and appropriate topologies of the organism can be determined after the robots reach the operation site.

Tracked locomotion is adequate for the quick locomotion on rough terrains. The Scout robot with tracked locomotion is capable of going up a slight slope, climbing over small obstacles, passing over a small hole, and also moving in soft ground. The long-range sensors on board can be used to scan the obstacles around then to navigate the organisms (Fig. 2(a)). When the tracked robots are docked together, the assembled robot becomes more robust to the roughness of the terrains as shown in Fig. 2(b). This high locomotive capability also allows the Scout robots to carry the Backbone robot(s) (see Figs. 2(c,d)). The Backbone robots can form an arm or a leg of an organism in advance, then be carried to the operation site so as to save the energy for 3D actuation in the organism. Thus, the Scout robots are adequate to be “feet” of the organism thanks to their robustness and locomotive capability. The disadvantage of the tracked locomotion is the non-holonomic drive characteristic that hinders efficient

---

### Table I

**Scout robot, Backbone robot and Active Wheel: requirements and solutions**

<table>
<thead>
<tr>
<th></th>
<th>Scout robot</th>
<th>Backbone robot</th>
<th>Active Wheel</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alignment</strong></td>
<td>Tracked</td>
<td>Accurate</td>
<td>Omnidirectional</td>
</tr>
<tr>
<td><strong>Locomotion</strong></td>
<td>Tracked</td>
<td>Plain</td>
<td>Nearly</td>
</tr>
<tr>
<td><strong>DOF</strong> of actuation</td>
<td>2 DOF</td>
<td>Bending: 1 DOF</td>
<td>Bending: Rot.</td>
</tr>
<tr>
<td><strong>Torque</strong></td>
<td>Low 3Nm</td>
<td>High 7Nm</td>
<td>High 18Nm</td>
</tr>
<tr>
<td><strong>Speed, act.</strong></td>
<td>30m/s</td>
<td>High 180°/s</td>
<td>Low 50°/s</td>
</tr>
</tbody>
</table>

---

![Fig. 1. First prototypes of robot designs (from left to right): Backbone robot, Active wheel, and Scout robot.](image-url)
docking procedures between the robots.

![Scout robots exploring the surface and guiding the organisms; Connected Scout robot; Scout robots carrying a Backbone robot; Scout robots carrying a chain composed of the Backbone robots. 4-legs shape of an organism; Scorpion-like organism.]

Fig. 2. Scout robots: (a) Scout robots exploring the surface and guiding the organisms; (b) Connected Scout robot; (c) Scout robots carrying a Backbone robot; (d) Scout robots carrying a chain composed of the Backbone robots. (e) 4-legs shape of an organism; (f) Scorpion-like organism.

Regarding the locomotion capability of the Backbone robot, easy assembly of the organism is of utmost importance. Therefore an omnidirectional drive is best since it offers optimal performance to move to a predefined position under a defined angle. This is important because each individual robot provides at least four different docking units and all of them can be used to form the structure of the organism. Every docking unit needs to be reached, regardless of the orientation of the robot which wants to dock. Unfortunately, the integration of an omnidirectional drive requires a lot of space due to the general construction of omnidirectional wheels. Nevertheless, if one takes a closer look at the details of the docking procedure, complete omnidirectional driving characteristics are not required for the Backbone robot, since the orientation of the robot is predefined by the docking units and therefore only certain directions of movement are necessary. In general, the Backbone robot needs to be able to move forward, backward and to turn since these are the minimum requirements for a swarm robot. Furthermore, under the condition of docking orthogonally to the normal drive direction of the robot, it needs to move sideways. A locomotion drive unit which can provide the features of a differential drive plus the possibility to drive to the side is therefore sufficient. Both features are provided by the screw drive, which is used within the Backbone robot. The screw drive locomotion unit itself can be built very small since only two driving motors are required and the driving screws have cylindrical shapes.

Beyond the normal use of the nearly omnidirectional drive of the Backbone robot, the screw drive provides the organism with a possibility to move sideways when the screws of all robots within the organism are synchronised. This can be a very helpful feature if a caterpillar like organism needs to steer to the side. An example of a system composed of reconfigurable heterogeneous mechanical modules, i.e. the Scout robots and the Backbone robots, are shown in the Figs. 2(e)-(g). All individual robots and organisms work as autonomous stand-alone systems.

C. Tool module: Active Wheel

In a heterogeneous system, robots of different design can form an organism together. The two individual robots, namely Scout robot and Backbone robot, have been proposed as basic elements to constitute an organism. The design of this individual robot is a result of compromise to integrate all mechanical and electronic functions into one robot. The features of such individual robots have to be redundant to be adaptable in an unknown environment. The idea of implementing tool modules into the heterogeneous system is to provide a few specially designed tools to compensate for deficits of the individual robots. The design of tool modules needs to be optimized for specific tasks such as sensing with a special sensor, manipulating an object, supplying power to the organism and carrying the individual robots or an organism quickly. The individual robots need to share external dimensions to be a part of the organism and for easy reconfiguration, and they need to be equipped with common electronics, while a tool module may have any shape as long as it can be docked to other individual robots or an organism. As an example of tool modules, we developed a tool module to carry individual robots, named Active Wheel (see Fig. 1). This tool module is intended to carry some individual robots quickly from one place to another without using their energy. The Active Wheel is an autonomous tool robot that is compatible with the other two individual robots platforms (Scout robot and Backbone robot) and used for assistance goals. An Active Wheel consists of two symmetrical arms connected in the middle by a hinge.

This structure gives the opportunity of bending this tool in both directions up to ±90° and hence can drive even upside down. Actually, such a symmetrical design does not require distinguishing between bottom and top or between front and rear side. An additional advantage of this geometry is the uniform weight distribution which is important for stable locomotion. Even if the robot is in a skew position a or b it tilts autonomously back into a stable position a₁ or b₁ (Fig. 3). One of the major tasks of this tool robot is to carry a certain number of individual robots efficiently from one place to another. This condition can be fulfilled only if the Active Wheel can move omnidirectionally. Therefore, two omnidirectional wheels are used on each side on the robot. Such kind of wheels have already been proven to work
Fig. 3. Symmetry and stability of the robot and capability to bend upwards or downwards.

reliably in many robotics projects e.g. in RoboCup [16]. Each wheel consists of many small single rolls which are arranged perpendicularly to the driving axle. This assembly allows an active movement in the driving direction of the wheel and simultaneously allows a passive movement in the normal direction. Each of these wheels is driven by a gear motor. Corresponding sensors which are placed on the driving axle detect the rotation speed of the motor. Those are necessary in order to provide complex manoeuvres such as driving curves or other complex trajectories. The docking between Active Wheel and another robot requires also a very precise control of the wheels.

Additionally to the motor control unit, the Active Wheel is equipped with similar electronic units and components like in the Scout or in the Backbone robot. These comprise for example similar processors, power management, IR sensing units, a ZigBee module, cameras etc. All these electronics are mainly required in order to navigate and to transport other robots autonomously and at the same time allow acting as stand-alone robot and fulfill many different tasks in robot swarms. In stand-alone mode, Active Wheels can be used for separating damaged modules or modules that are not able to move. One possible scenario how an Active Wheel can act as a stand-alone robot, is shown in Fig. 4. Two Active Wheels are placing a module that was flipped over in the right position again.

As an example of a simple organism, topology of three robots can be considered Fig. 5. The idea of this configuration is based on a combination of advanced computational and sensor features, provided by these two individual robots, and fast motion speed, provided by the Active Wheel. Addi-

tionally, the Active Wheel can supply both individual robots with extended energy source. As a common system, these three platforms complement each other and demonstrate commonly very outstanding characteristics. Features of a common system essentially excel the capability of each of these individual robots – this is typically the collective approach.

Fig. 4. Two Active Wheels carry a defective element.

Fig. 5. Simple organism - Active Wheels with two different docked modules.

D. Docking Mechanisms and Strategies

The docking mechanisms are of primary importance in modular robotics as well as in symbiotic multi-robot organisms. They should assure docking and undocking between individual robots, as well as electrical continuity for power sharing and signal transmission. Furthermore, the docking mechanism should tolerate at a certain degree misalignments of individual robots during the docking process [12]. Nilsson et al. have investigated design of a docking unit and summarized desirable connector properties [17]. In this section, the properties required for docking mechanisms are investigated and a guideline for the docking design is proposed. Docking is composed of several phases, and each phase has several requirements to be satisfied.

Approach. The approach of the docking units can be categorized into three modes. The first is the approach of the two locomotive individual robots. Because both robots can move freely, the approach of the docking units is rather easy. The second is the approach of an individual robot to an organism. In this case, the individual robot should be precisely steered. When the individual robot with non-holonomic locomotion capability needs to be docked to the organism, the docking units on the side walls are not available unless the organism itself can approach the individual robot. Thus, the aggregation of an organism must be carefully planned considering the locomotion capability of the individual robots. The last one is the approach of the two assembled robots or two arms/legs of an organism, and this is especially important for a reconfiguration of the organism.
Alignment. Docking design that allows robust self-alignment is crucial for autonomous assembly of a modular robot. Ground roughness needs to be taken into consideration for the docking of locomotive individual robots. In addition, it must be noted that the accuracy of the fabrication and assembly of each robot have a strong influence on the alignment accuracy.

Docking and Locking. A docking unit with hermaphroditic feature is preferable to make the assembly plan easier. The docking must be tight and stable, and the electrical connection between the docked robots must be ensured. In some existing docking designs, the docking is secured by an additional locking mechanism. A simple docking/locking mechanism occupying small space and being actuated with little energy is preferable as well.

Sustainment of the docked status. The docking status must be sustained without or with minimum power supply. The docking status needs to be independent from the actuation of the assembled robots, otherwise, the additional control is necessary to maintain the docking status.

Unlocking and Undocking. Another important feature is the capability to allow undocking between two docked robots in case of an emergency. If one of the individual robots undergoes failure or malfunction, the robot must be removed from the organism by the other robots. Therefore, it is preferable to undock the robot by activating only one of the docked units.

Separation. The individual robots need to be separated and move away from the assembled robot after being undocked so as not to hinder following procedures. When an individual robot with non-holonomic locomotion cannot move away after being undocked, the organism needs to move away from it or another robot needs to come to move it away.

In addition to the above mentioned requirements, easy and low-cost manufacturing for mass production and easy maintenance are important especially when a large multi-agent symbiotic system is targeted. Because multiple docking units are required for an individual robot, the cost of the docking unit is important.

To summarize this section, we have to point out two essential issues: integration with electronics, and a need of software protection from mechanical damages, caused during evolving different controllers. Both issues are essential in a successful design and stepwise improvement of mechatronic platforms.

III. GENERAL HARDWARE ARCHITECTURE

In this section, the electronic hardware and architecture of single robot modules (the first prototype) is described in more detail as another example of self-reconfigurable robots (see Fig. 6). Since in SYMBRION advanced control and evolutionary algorithms, such as on-board genetic evolving, etc. needed to be implemented, here, one major design criterion was the calculation and processing speed. On the other hand, REPLICATOR required a high number of different sensors since the swarm’s objective was to form a highly dynamic sensor network for vast applications, like surveillance, exploration, etc. As shown in Fig. 6, each module hence carries a number of processors/microcontrollers. However, the major control of each robot is performed by the “Core Processor”, an LM3S8970 Cortex microcontroller from LUMINARY MICRO INC. The main purpose of it is to pre-process raw sensor data, to run higher level algorithms such as an artificial immune system (AIS) or artificial homeostatic hormone system (AHHS), to calculate the module’s position, to pass this information to actuators, etc. In order to support this processor, a shadow processor (Blackfin, ADSP-BF537E from ANALOG DEVICES) is included that mainly takes over computationally intensive processing tasks, i.e. of the images taken from the 4 on-board cameras. Due to its high power consumption, the intention is to operate this processor unit only if required. For example, if image processing has to be used to recognize the environment or if the organism size (i.e. number of docked modules) reaches a certain limit so that locomotion tasks require a lot more computational resources.

A dedicated microcontroller (Atmel1280 from ATML INC.) is responsible for A/D-conversion and further processing of analogue sensor signals like microphones, IR-based distance sensors, etc. Since at least 1 brushless motor, whose control occupies many processing resources, is on board a robot module 2 additional Cortex controllers (LM3S8962) have been integrated, dedicated to all major actuation and locomotion tasks. Furthermore, the robots possess a UWB-based localisation unit, a ZigBee radio communication module, a battery management module, Flash and SD memory, a LASER ranging module, and other sensors.

A. General Sensor Capabilities

Following the approach from the previous section, we consider now the general sensor capabilities of the platform. For the application of evolutionary approaches as well as for sensor network applications, the platform should provide a measurement of environmental values, in particular, how
robots do fit to the environment. The local fitness measurement for collective behavior represents a very challenging task, therefore a serious attention during the design of the platform was paid to this issue. From a conceptual viewpoint, the following four ways are available to measure the fitness: approximation of a global state by local sensors, perception of local environment by on-board sensors, different measurements during robot-robot interaction, and finally, measurements of internal states.

TABLE II
OVERVIEW OF ON-BOARD SENSORS.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Name</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environmental Light</td>
<td>ADPS9002</td>
<td>analog</td>
</tr>
<tr>
<td>Air Pressure</td>
<td>SCP1000</td>
<td>I2C</td>
</tr>
<tr>
<td>Directional Sound</td>
<td>SPM0208HD5</td>
<td>analog</td>
</tr>
<tr>
<td>Humidity/Temperature</td>
<td>SHT15</td>
<td>I2C</td>
</tr>
<tr>
<td>IR-reflective</td>
<td>TCRT1000</td>
<td>analog</td>
</tr>
<tr>
<td>Imaging Sensor</td>
<td>OV7660FSL</td>
<td>PPI</td>
</tr>
<tr>
<td>Laser (in the Range Finder)</td>
<td>LS-1-650</td>
<td>digital</td>
</tr>
<tr>
<td>RFID sensor</td>
<td>Lux</td>
<td>no</td>
</tr>
<tr>
<td>Sonar sensor</td>
<td>SRF08(10)</td>
<td>I2C</td>
</tr>
<tr>
<td>Laser RangeFinder</td>
<td>URG-04XL</td>
<td>RS232/USB</td>
</tr>
<tr>
<td>Detecting motion</td>
<td>AMN34111</td>
<td>analog</td>
</tr>
<tr>
<td>Hall effect (magnetic)</td>
<td>US8881EU</td>
<td>analog</td>
</tr>
<tr>
<td>Color Sensor</td>
<td>TCS230</td>
<td>digital</td>
</tr>
<tr>
<td>Capacitive</td>
<td>MT01IN-NR</td>
<td>digital</td>
</tr>
<tr>
<td>Locomotion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D Acceleration</td>
<td>LISL02AL</td>
<td>I2C</td>
</tr>
<tr>
<td>WTL laser mouse</td>
<td>ADNS-7530</td>
<td>SPI</td>
</tr>
<tr>
<td>3D Localization</td>
<td>Ubisense</td>
<td>digital</td>
</tr>
<tr>
<td>Orientation-Sensor</td>
<td>SFH 7710</td>
<td>SPI</td>
</tr>
<tr>
<td>IR-docking sensor</td>
<td>IR-based</td>
<td>analog</td>
</tr>
<tr>
<td>Force measurement sensor</td>
<td>K100N</td>
<td>analog</td>
</tr>
<tr>
<td>Joint angle sensor</td>
<td>2SA-10-LPCC</td>
<td>analog</td>
</tr>
<tr>
<td>Compass</td>
<td>HMC5843</td>
<td>digital</td>
</tr>
<tr>
<td>Internal, Indirect Sensors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voltage, Current</td>
<td>BQ77PL900DL</td>
<td>SMBus</td>
</tr>
<tr>
<td>Bus Load Sensor</td>
<td>no</td>
<td>software</td>
</tr>
<tr>
<td>Center of mass</td>
<td>no</td>
<td>software</td>
</tr>
<tr>
<td>Energy-docking sen.</td>
<td>no</td>
<td>software</td>
</tr>
</tbody>
</table>

1. Approximation of a global state by local sensors. For an application of evolutionary strategies the most appropriate feedback may be provided when knowing a global state of the environment, including internal states of other robots. However, such information is not available for individual robots due to practical reasons. Nevertheless, the global state can be approximated when using the world model and several sensor-fusion approaches. Examples of global states are map-related values, such as explored/unexplored area, coverage of some territory, position of robots in 3D space. The platform includes several sensors, such as localization system or laser rangers, for these purposes.

2. Sensing a local environment. Perception of local environment by on-board sensors is the primary way of receiving information about the environment for both evolving and sensor network applications. The overview of integrated, or considered for integration, sensors is given in Table II.

3. Information provided by a robot-robot interaction and communication. Robot-robot interaction is a very important source of fitness measurement. The corresponding sensors are the force measurement sensors, joint angle, compass or 3D accelerations. Robot-robot communication plays also an important role here, which allows fusing local information from different robots. This is related not only to environmental values, but also to internal states of robots.

4. Internal states of robot organisms. There are different internal sources of information: energy-based, mechanical, load on buses, number of internal failures, CPU/Memory usage and other. The energy-based values are very useful for many purposes, e.g. in estimation of the most efficient structure of organisms. Generally, the number of internal sensors, most of them are virtual sensors, can be very high.

To give a reader an impression about sensing capabilities of the platform, we collect in Table II a brief overview of on-board sensors.

IV. CONTROLLER FRAMEWORK

In robotics, several different control architectures are well-known, as e.g. subsumption/reactive architectures [18], insect-based schemes [19] or structural, synchronous/asynchronous schemes, e.g. [20]. An overview of these and other architectures can be found in [21]. Recently, multiple bio-inspired and swarm-optimized control architectures have appeared, e.g. [22], [23]. In designing the general control architecture, we face several essential challenges:

- **Multiple processes.** Artificial organisms execute many different processes, such as evolutionary development, homeostasis and self - organizing control, learning, middle- and low-level management of software and hardware structures. Several of these processes require simultaneous access to hardware or should be executed under real-time conditions.

- **Distributed execution.** Hardware provides several low-power and high-power microcontrollers and microprocessors in one robot module. Moreover, all modules communicate through a high-speed bus. Thus, the multiprocessor distributed system of an artificial organism provides essential computational resources, however their synchronization and management present a challenge.

- **Multiple fitness.** Fitness evaluation by using local sensors is already mentioned in Sect. III-A. Here we need to mention the problem of credit assignment related to the identification of a responsible controller, see e.g. [24]. Since many different controllers are simultaneously running on-board, the problem of credit assignment as well as interference between controllers is vital.

- **Hardware protection.** Since several controllers use the trial-and-error principle, the hardware of robot platform should be protected from possible damage caused during the controllers' evolution.

Corresponding to the hardware architecture, the general controller framework is shown in Fig. 7. This structure follows the design principles, originating from hybrid deliberative/reactive systems, see e.g. [25]. It includes a strongly rule-
based control component, see e.g. [26] as well as multiple adaptive components [27]. The advantage of the hybrid architecture is that it combines evolvability of reactive controllers, and their high adaptive potential, with deliberative controllers that provide planning and reasoning approaches required for the complex activities of an artificial organism.

Controllers are started as independent computational processes, which can communicate with each other and with different sensor-fusion mechanisms, such as virtual sensors or the world model. Processes are running on different modules, synchronization and interaction between them is performed through message-based middleware system. There are controllers, which use evolutionary engines and their structure is coded in the artificial genome. There are several bio-inspired ideas towards such an artificial genome. It is assumed that there are also a few task-specific controllers, which are placed hierarchically higher than other controllers. These task-specific controllers are in charge of the macroscopic control of an artificial organism. They may use deliberative architectures with different planning approaches, see e.g. [28].

The action-selection mechanism is one of the most complex elements of the general controller framework. This mechanism reflects a common problem of intelligent systems, i.e. “what to do next”, see [29]. Finally, a hardware protection controller closes the fitness evaluation loop for the evolvable part of controllers [30]. This controller has a reactive character and monitors activities between the action-selection mechanism and actuators as well as exceptional events from the middleware. It prevents actions that might immediately lead to destroying the platform, e.g. by mechanical collisions.

V. GRAND CHALLENGES FOR ARTIFICIAL ORGANISM

Issues of challenges in evolutionary, reconfigurable and swarm robotics were mentioned several times since the early 1990s. We can refer to works [31], [32], [33], [34], [35] related to challenges with fitness estimation, “reality gap” and others, whereas more recent work gives overview of challenges in the robotic area [21], such as over-motorization of reconfigurable systems or communication in swarm robotics. However, artificial organisms combine all three areas, resulting not only in a combination of problems and advantages, but also in qualitatively new challenges and breakthroughs. To demonstrate these breakthroughs, two Grand Challenges have been developed. The two following sections discuss underlying ideas of these Grand Challenges and problems in achieving them.

One of the important aspects of artificial organisms is their high degree of adaptivity. Moreover, adaptivity is estimated as one of the major technological challenges, see e.g. [36], [37], [38]. On the other hand, one of the essential general challenges in robotics is a long-term independency of autonomous systems. It seems reasonable that Grand Challenges have to reflect these two issues.

However, adaptivity is addressed by two Grand Challenges in different ways. In Fig. 8 we represented a brief overview of different adaptive mechanisms, related to changes of environment (endogenous factors) and developmental plasticity of regulative mechanisms. This figure can be roughly divided into low-, middle- and highly-rate adaptive parts (for regulative structures and corresponding environmental changes). Due to the nature of the Cognitive and Evolutionary frameworks, they address different adaptive parts: the 1st. Grand Challenge – the medium-rate adaptive part and the 2nd. Grand Challenge – the high-rate adaptive part.

Another split between Grand Challenges can be based on different understanding of artificial evolution. From the first viewpoint, artificial evolution is based on all achievements of natural evolution, including human technological progress, see Fig. 9(a). In other words, artificial evolution can be based on technological artefacts, pre-programmed behavioral patterns or include human-written algorithms. From another viewpoint, shown in Fig. 9(b), artificial evolution is con-
This idea is sketched in Fig. 10, different possible sub-scenarios and evaluation criteria are summarized in Table III.

A. 1st Grand Challenge – 100 Robots, 100 Days

The first Grand Challenge is primarily related to the Cognitive framework and addresses the problems of long-term independency in a medium-rate changeable environment with the assumption that artificial evolution can include technological artifacts. Here we can also find application and utilization of almost all other robotic issues such as e.g. reliability, energetic homeostasis, regulatory autonomy and others. This Grand Challenge may have the following form:

A large-scale system, let assume with 100 heterogeneous modules, is placed in a previously unknown area, which has complex, but structured character. This environment is slowly changing, for example, energetic resources are displaced or their indication is changing. This area contains enough energetic resources, such as power sockets or power cubes, which are sufficient for these 100 modules to survive in such an environment. The main energy source – power sockets – are inaccessible for individual robots, e.g. placed 30-40 cm above ground or in some structural gaps. Moreover, power sockets are switching on and off over the time in different order so that robots should first recognize position and quality of energy. Under these conditions the robots can survive only collectively, when aggregating into organisms with more distributed recognition and and extended affordance and actuation capabilities than individual robots. Aggregated robots perform in this area surveillance and disposal tasks with respect to fellow robots or modules passed away by pulling and carrying them if possible to a ‘graveyard’ - taking the environmental dynamics and the robots energy constraints into account. This experiment takes 100 days and should ideally be performed without any human maintenance work or supervision.
ACKNOWLEDGEMENT

The “SYMBRION” project is funded by the European Commission within the work programme “Future and Emergent Technologies Proactive” under the grant agreement no. 216342. The “REPLICATOR” project is funded within the work programme “Cognitive Systems, Interaction, Robotics” under the grant agreement no. 216240. Additionally, we want to thank all members of the projects for fruitful discussions.

REFERENCES


CKBot Platform for the ICRA 2010 Planetary Challenge

Shai Revzen, Jimmy Sastra, Nick Eckenstien, Mark Yim

I. INTRODUCTION

The ICRA Planetary Contingency Challenge 2010 will include 3 teams that will be using the new CKBot module and software package. This paper will present some of the new aspects of this hardware, underlying software architecture for quickly programming the modules and will include a tutorial for these teams.

II. HARDWARE

The planetary contingency will feature 50 CKBots and 10 CKBot wheel modules. The CKBots are manually reconfigurable modular robots that resemble a 6 centimeter cube on each side.

CKBots are aimed to be robust, using reliable connectors and high flex life ribbon cable. Cost is kept low by using hobby servos and making them easy to manufacture using lasercut ABS. They support a highly versatile range of locomotion as they can be configured into many shapes and have high torque capabilities. They are designed to be small enough to crawl through a 3 inch pipe, while having enough torque to cantilever 7 modules by itself. Capabilities can be improved by a spring loaded weight compensation mechanism that is easily attached such that it can create an even longer arm.

<table>
<thead>
<tr>
<th>TABLE I CKBot SPECIFICATIONS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size: 60x60x60mm.</td>
</tr>
<tr>
<td>Torque: 417 oz.-in.</td>
</tr>
<tr>
<td>Speed: 0.17s / 60 degrees</td>
</tr>
<tr>
<td>Communication: CANBus, IrDa (local/remote range)</td>
</tr>
<tr>
<td>Weight: 146g.</td>
</tr>
<tr>
<td>Configuration: manual with screws</td>
</tr>
<tr>
<td>Input Power: 24V</td>
</tr>
</tbody>
</table>

Each module has two micro controllers. One is tasked with CANbus and servo communication, the other is equipped with 7 independent serial ports and manages IR communication. IR communication features a local mode where communication only occurs between modules that are attached as well as a remote range mode where communication can occur at a distance of 2 meters if in line of sight. The local mode is useful for configuration recognition. The remote range mode can be used between unconnected clusters or modules connected through passive pieces.

Users can program CKBots using Robotics Bus, a protocol developed on top of CANbus. This software is written in Python.

A. Gravity Compensation

There is a new attachment to the ckBot module that can be used to compensate for gravity. It makes use of a spring attached to the active arm of the module to compensate for the weight of one or more modules. It should be noted that due to technical limitations, it can only be applied to one module in the vertical plane. However, it still makes certain tasks possible that were previously not possible. Most notably, use of an outwardly extended ckBot chain is aided by this attachment, allowing a longer arm before the motor at the base is not able to lift the chain. The attachment simply screws onto the side of the existing ckBot frame, and comes ready to attach. Figure ?? shows one module with a two module load and perfect passive gravity compensation over the full range of motion (180°).

III. SOFTWARE

The CKBot modules are controlled through software comprising several layers. At the lowest layer, modules support a Robotics Bus (RB) protocol layered on top of a CAN bus. RB defines means for nodes to be queried regarding their settable and gettable properties.

On top of RB we developed a python library that dynamically discovers and presents a logical view of a cluster of modules connected to a single CAN bus. The Cluster class can
automatically discover properties of the modules and exposes them to the user via set and get methods in dynamically generated classes. When the module software version is familiar, the Cluster instantiates the appropriate Module subclass to represent it in addition to the automatically discovered properties. For interactive use and prototyping we designed the class structure to facilitate easy use from the Interactive Python shell which supports features like tab-completion for set and get methods and module names. For programming use, the discovered properties, low-level RB addresses and module sub-class methods are all presented with a combined naming scheme allowing them to be listed, read and written. The interface also provide fast communication with the modules using asynchronous IO. An example of the Cluster API is shown in Figure 2.

```python
# Create a Cluster
c = Cluster()
# Expect 4 modules and name 3 of them as specified
c.populate((0,0x91: 'left', 0x9b : 'head', 0x5d : 'right'))
# cluster.at provides logical alias for a numeric module ID
assert c.at.head == c[0x9b]
# Set position of a head servo
cluster.head.set_pos(4500)
# cluster.od exposes auto-discovered properties
cluster[0x9b].od.set_pos(4500)
```

Fig. 2. Example of Cluster API use.

On top of the Cluster interface, we developed a GUI based on the wxPython cross-platform UI library. It presents a tree view of all properties in current modules, and allows the property lists to be quickly customized by editing them as YAML files. The user may select properties to examine and set via the UI, and receives visual indication if modules disconnect.

For high-level programming and interactive operations, we developed an application framework (JoyApp) based on pygame’s event driven architecture and sporting an interface to the Scratch visual programming environment as shown in Figure 3. We enhanced pygame with new event types representing robot module position changes and Scratch events and sensor updates. JoyApp provides a powerful new abstraction: the Plan. A Plan is a sequentially executing behavior with its own incoming event queue and event handler. Plans may be executed sequentially or in parallel using a cooperative multitasking architecture that obviates the need to worry about thread safety.

A library of Plan classes is provided, including plans with the ability to load spreadsheets saved in the commonly supported CSV format as Gait Tables, Figure 4. A Gait Table is a spreadsheet whose columns are mapped by each SheetPlan instance to settable properties in the Cluster and whose rows represent consecutive times. A value in a cell represents the operation of setting the property to the specified value at the specified time. The use of a full-featured spreadsheet program to generate the gait tables makes it particularly easy to edit gaits and to explore gaits with a mathematical relationship between the values written in different times and into different properties.

```python
# ... in JoyApp.onStart()
sheet.ltPinch=SheetPlan(self, loadCSV("gait.csv"),
elbow="Nx35-pos", wrist="Ax55-pos")
sheet.rtPinch=SheetPlan(self, loadCSV("gait.csv"),
elbow="Nx5A-pos", wrist="Ax3A-pos")

# ... in JoyApp.onEvent()
if evt.type==JOYBUTTONDOWN:
  if evt.button==0: self.ltPinch.start()
  elif evt.button==1: self.rtPinch.start()
```

Fig. 3. Example of Cluster API use.

Fig. 4. A “Gait table”.

Fig. 5. Using the same gait table on two different arms.
Recent Progress of SuperBot

Wei-Min Shen, Feili Hou, Mike Rubenstein, Harris Chiu, Akiya Kamimura

Abstract—This paper describes some of the recent progress developed in the context of the self-reconfigurable SuperBot system, including the complexity analysis of self-reconfiguration planning, theories and simulations for scalable self-healing, dynamic control of deformable and recoverable rolling track, the SINGO SuperBot connectors, and switchable magnetic-SINGO connectors for metal bridge inspections.

I. INTRODUCTION

SuperBot [5,6] is a self-reconfigurable robotic system under development at the University of Southern California. It was originated for NASA in 2005 by a research team led by USC/ISI with members from nine organizations, including Ames Research Center, JPL, University of Pennsylvania, University of Hawaii, Lockheed Martin, Raytheon, Alliance Spacesystems Inc., Metrica Inc. and Life Science-Technology Research Inc. The original objective was to develop self-reconfigurable modules and demonstrate them in a desert by starting from a contracted configuration, reconfigure into rolling configuration, travel a distance to a sand dune, reconfigure into a climbing configuration, climb the sand dune to the top, reconfigure into a “greenhouse” platform, and sustain a set of carried seeds for a period of time until they grow into sprouts.

II. COMPLEXITY ANALYSIS OF RECONFIGURATION PLANNING

One of the open problems for self-reconfigurable robots is how to plan a reconfiguration from an initial configuration to a target configuration. This capability is critical for modular self-reconfigurable robots because it determines the usability of reconfigurable robots in practice. By definition, a reconfiguration plan consists of a sequence of actions of “connecting” and “disconnecting”, and a plan is optimal if it has the shortest length among all possible plans.

Most existing algorithms for reconfiguration planning, such as divide-and-conquer, graph matching, etc, do not guarantee optimality. In addition, the complexity of finding an optimal solution is also unknown. One common belief is that it may be computationally intractable because the configuration space is exponential. However, that alone does imply that any polynomial algorithm may not exist. In fact, many optimization problems that have exponential search space do have efficient solutions. For example, the number of paths between two nodes in a graph can grow exponentially with the number of nodes in the graph, but finding the shortest-path without negative cycles can still be solved efficiently.

In a recent paper [1], we have investigated the complexity of optimal reconfiguration planning problem for chain-type modular robots. Our results show that the problem is indeed NP-complete, even if the configuration graphs are acyclic. This offers evidence that a polynomial-time algorithm for finding an optimal reconfiguration plan is unlikely to exist. To facilitate evaluation of reconfiguration algorithms, our result does offer the lower and the upper bounds for the minimum number of reconfiguration steps for any given reconfiguration problem.

We define Acyclic Optimal Reconfiguration as follows. Given acyclic configuration graph $I$ and $G$, and a given integer $n$, does it exist a reconfiguration plan that has $\leq 2n$ reconfiguration steps? One can prove that the AOR problem is NP-Complete by reducing a 3-PARTITION problem to it in polynomial time. The idea is that from an arbitrarily given instance of 3-PARTITION, one can construct an initial configuration $I$ and a goal configuration $G$ as shown in Figure 2. Thus, if the 3-PARTITION problem has a solution, then $I$ can be transformed to $G$ in at most $(6m-2)$ steps. Furthermore, if $I$ can be transformed to $G$ in at most $(6m-2)$ steps, then the 3-PARTITION problem has a solution.
III. SCALABLE SELF-HEALING

Another interesting problem for self-reconfigurable robots is *Scalable Self-Healing*. This is defined as how a collection of mobile robotic modules can self-assemble into a global spatial/temporal pattern (e.g., shape and color) without any centralized controller. If the pattern is disturbed or damaged, the remaining robots can self-heal the pattern with different scales. The process is “scalable” because the size of the pattern is not known in advance and must be adapted to the current number of robots/modules in the collection. For example, if a starfish is cut into two halves, then the two halves must separately self-heal into two new starfishes with smaller scales.

The challenges for this problem include: how the local robots self-organize into a global coordinate system? How do they self-assemble into the desired pattern? How do they repair damages? And how do they dynamically adjust the size of the pattern?

In some recent papers [2,3,4], we have developed distributed solutions for these problems. Figure 3 illustrates the simulation results for this line of work. A collective of randomly scattered robots are tasked to form a five-pointed colored star (3.a). If the star is cut, then the remaining robots will self-heal into a smaller star with the same spatial and color pattern (3.b). If the star is scrambled, then the robots will recover the pattern (3.c). Furthermore, if the current scale is too small (or big) for the number of robots in the collective, the robots will collectively decide and gradually increase (or decrease) the size of the shape until all the robots are in the pattern and pattern is complete (no missing parts). This demonstrates the ability to automatically scale the shape to the number of robots, even if that number changes. The details of these distributed mechanisms are available in the papers and movies can be found at http://www.isi.edu/robots/media-morphos.html. In a companion paper [7] in this workshop, it has been reported that these algorithms are being implemented on real robots.

![Figure 3: Scalable self-assembly and self-healing.](image)

IV. DEFORMABLE DYNAMIC ROLLING TRACK

Unlike a wheel, a modular rolling track propels forward by actively changing its shape. A common problem experienced by the wheel and rolling track is the inability to perform self-recovery, meaning that it cannot stand up without any external help once it has fallen sideways. An example would be a rolling track recovering from a flattened orientation as in Figure 4(a) to a stand-up posture as in Figure 4(b). A rolling track cannot be deemed as “complete” unless it can self-recover, roll and turn. Such robot is able to traverse the environment. Table 1 lists a quick comparison of the existing modular rolling track systems.

![Figure 4: Self-recovery of a modular rolling track.](image)

---

ICRA 2010 Workshop "Modular Robots: State of the Art"
In a recent paper [8], we have reported a new behavior of the SuperBot rolling track system that can dynamically "stand-up" from a flattened position. This rolling track uses the on-board accelerometers to dynamically determine the control values for the motors in order to move forward. Such sensor-based control will enable the rolling track to adapt to the terrain and slopes where it is traveling on. If the robot falls flat due to unexpected events, it can recover its rolling status by standing up by itself. The sequence of self-recovery is illustrated in Figure 5.

![Fig 5. A 6-module SuperBot rolling track self-recovering from falling.](image)

V. PROGRESS IN SYSTEM AND MECHANISMS

III.1. The SINGO Connector Mechanism

Connection mechanisms are critical for self-reconfigurable robots. One highly desired features is single-end-operative, that is, able to establish or disengage a connection even if one end of the connection is not operational. This is essential for self-healing because modules may be out of service unexpectedly, and no connections should be seized permanently or disconnect unintentionally.

The SINGO connector mechanism [9] is designed to offer such features. Shown in Figure 6, it has a base on which four movable connector jaws are formed on one side to provide the single-end operative connection operation. The connector base is structured to have four open slots that are under the sliding rails to expose a motorized circular gear. This circular gear has top spiral or concentric tracks that are engaged to the bottoms of the jaws. As the circular gear rotates, it drives the jaws along their respective sliding rails.

![Fig 6. The SINGO Connector Mechanism.](image)

This SINGO connector is gender-free, compliant against misalignment during docking (5.0mm in x-y cross section, 6.0mm in distance, and 5 degree in rolling), and capable of docking and undocking even when the other side is not cooperative.

III.2. SINGO-based Switchable Magnetic Connector

One additional advantage of the SINGO connector is that its circular driving mechanism can offer multifunctional uses. One such use is to turn on and off a switchable permanent magnet so that the connector can use permanent magnets to connect and disconnect with other modules, or attach and detach from any metal surface.

![Fig 7. SINGO-based switchable permanent magnetic connector](image)

Figure 7 shows two prototypes of this switchable permanent magnetic connector that are installed at the ends of a 2-module SuperBot chain. At the center of this connector is a switchable permanent magnet that we
purchased directly from Home Depot. It can be turned on or off by the SINGO base under the control of on-board electronics. We call this new configuration of SuperBot as “Bridge Climber” as it is designed to climb and inspect metal surface of a bridge for inspection. We have developed control software for this new robot and it can now climb vertically on any metal surface with open-loop gaits. Software is also developed to allow the robot to turn autonomously or under remote control. Figure 8 shows two snapshots as the robot is climbing on a vertical bar in the lab environment. Movie of this behavior is available at http://www.isi.edu/robots/superbot/movies/FirstClimb.avi.

We plan to further develop this behavior so that SuperBot can be used to climb on metal bridges for inspection.

Fig 8. SuperBot climbing a vertical bar in the lab.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

SuperBot is a self-reconfigurable robot designed for multifunctional uses in real-world applications. This paper reports some of the recent progress towards that goal. These progress span from the complexity analysis of reconfiguration planning to theories for scalable self-healing, from novel connection mechanisms to new potential applications for infrastructure applications. Future directions include further harden the systems for large-scale real-world problems (such as exploring and mapping in extreme environments [10]) and increase the intelligence of the robots for surface identification [11] and failure detection and recovery based on learning from surprises [12].

ACKNOWLEDGMENT

We would like to thank the members of USC Polymorphic Robotics Laboratory for their assistance with this work.

REFERENCES


Introduction

Modular robots are a class of robotic systems composed of many identical, physically connected, programmable modules that can coordinate to change the shape of the overall robot. By transforming its shape, a modular robot can achieve many tasks, such as locomotion in complex terrains. An interesting question is how to design these relatively simple modules such that they can be combined to achieve a variety of locomotion mechanisms such as rolling, crawling, and climbing. Modular locomoting robots have the potential of being more adaptive to their environments, and may thus prove useful in applications such as space exploration or search and rescue.

One recent area of interest has been the design of “expandable” modular robots. In 2004, NASA developed a proposal, along with a conceptual video, for a modular robot composed of many expandable links, capable of moving in complex ways: sliding, rolling, climbing, etc [2]. They also constructed a physical version of a 4-link tetrahedral expandable robot, called the 4Tet Walker, capable of conducting a rolling motion; this design was later expanded to a 12-link robot, however the design was quite large and difficult to scale due to the significant weight. Nevertheless it demonstrated an interesting concept for the design of a modular robot. More recently, at the University of Southern Denmark, Lyder et al [1] developed Odin, a fully modular deformable robot based on a similar concept of expandable links with compliant joints; the group demonstrated modular electronics and communication, and also showed how passive expandable links could be introduced in addition to the active links. Our group also explored the design of deformable modular robots, and developed several sensing-based control algorithms, including one for rolling locomotion using the Tetrapod robot with pressure and light sensors [3,4].

One of the main difficulties in the implementation of these expandable modular robots is the mechanical design. The proposed locomotion techniques require large changes in actuation length and significant compliance at the joints during locomotion, but also require rigidity in order for the robot to hold its shape. These conflicting requirements make it difficult to design even the simple rolling tetrahedral walker. For example, the 4Tet Walker telescoping links were hand-crafted and quite heavy, the Odin robot so far has demonstrated only sliding motion rather than rolling motion due to the limited range of both the expandable links and the joints, and the pressure-sensitive Tetrapod design by Yu et al frequently failed due to joint breaks.

In this paper we explore some of the design challenges involved in constructing expandable modular robots. We present the mechanical design of a modular-expandable robot that is capable of multiple configurations and locomotion styles: tetrahedral rolling, 2D sliding, and simple climbing. In addition the modules are easy to manufacture, using only 3d printing and off-the-shelf actuators, and have proven to be robust to repeated use and reassembly. We believe that these mechanical design principles can be incorporated into other similar robots, and move us closer to the types of complex locomotion envisioned by the original NASA project.
Mechanical Design of the Modules

Our expanding modules were first designed to be used in a tetrahedral walker robot. The general configuration and motion of the tetrahedral walker is described in Figure 1, based on the design and algorithms developed in [1,4].

The tetrahedral walker is composed of 6 active links and 4 joints. The walker moves by shifting its center of mass through expansion (linear actuation) of the active links. The body of the walker begins in a pyramidal position and is then contorted until it reaches a critical position causing it to fall over. The structure then returns to its original pyramidal position. During this movement cycle the tetrahedral robot takes on significantly different shapes, changing from a 60° angle between limbs to a 26° angle. The links also require a significant expansion ratio. One of the main challenges to this design is the balance between rigidity and deformability. The modules need to be capable of deformation, both elongating and changing their connection angle, while retaining a rigid form.

The primary requirement in our design was to develop modules capable of the expansion, compliance, and rigidity necessary for this locomotion. In addition we had several secondary requirements. The design needed to be easy to manufacture, using only 3d printed parts and off-the-shelf components, and not require the machining of special metal parts. It also needed to be robust and reusable for long-periods of time, inspite of the manufacturing restrictions and high stresses caused by the rolling locomotion. Finally it needed to be easy to reassemble in new configurations, since the long-term goal was to explore multiple locomotion styles.

These requirements were considered when designing each of the components. There are three main components of the design: the joint, the connector, and the linear actuators with housings. The assembly of the connectors, actuators and housings constitute one module; the joints allow us to combine multiple modules into a larger structure. The assembled parts can be seen in Figure 2. All of the components were designed in Solidworks and printed out of ABS using Fused Deposition Modeling (FDM). Design decisions for each of the components are described in more detail next.
• **CONNECTOR DESIGN**
  The connectors provide the interface between the joint and the active link (linear actuators). The connectors have the unique responsibilities of supplying the compliance in the structure as well as taking the majority of the load when the structure flops over. Thus, when designing the connectors a part was created which would provide secure connections, create compliance, and be able to withstand significant loading.

  The connector is attached securely to the linear actuator using screws, since this connection is meant to be permanent over many structures. To interface the connector with the joint several different options were considered and were evaluated for the security of the connection as well as their ease of assembly and disassembly. Two different sizes of threads (20mm X 1.25 and 20mm X 1.67) were tested as well as a design utilizing tabs (Figure 3L), all of which were directly printed in ABS plastic. The smaller threads were seen to be too thin to be reliable thus the tabbed design and the larger thread design were selected for use. To achieve sufficient compliance in the connector, a ball joint was used having a swivel angle of 80°. This is highlighted in Figure 3Ra.

  Finally, the connector was designed to repeatedly handle the force of the walker tumbling. Stress concentrations were thus avoided by reducing the number of sharp corners. Also, a stiff black tube was placed around the connector, which serves to damp loading without limiting the motion of the connector. This can be seen in Figure 3Rb.

• **JOINT DESIGN**
  The joints are responsible for connecting the different modules together. In the case of the tetrahedral walker this involves holding three modules each 60° from each other. To hold the modules the joints were designed to interface with the connectors. A joint to interface with the tabbed connector (figure 4a) as well as a joint designed to interface with the connector with larger threads (figure 4b) were developed. These two joints can be seen in Figure 4. The smooth rolling surface allows the tetrahedral walker to easily roll over the joints. While the original design was focused on the tetrahedral configuration, several aspects were designed with the long-term goals in mind. For example, the ball unscrews into two halves to allow the easy design of joints with more connections. In addition this provides access to the hollow interior, which can be used to store components such as additional weight or sensors.
FIGURE 3: (L) THREE INITIAL CONNECTOR DESIGNS. DESIGN A UTILIZES SMALL THREADS, DESIGN B UTILIZED TABS, AND DESIGN C WAS THE MOST SUCCESSFUL UTILIZING LARGER THREADS. (R) CONNECTOR FEATURES INCLUDING A BALL JOINT WITH 80° OF SWIVEL (A) AND STIFF TUBING TO DAMPEN LOADING (B).

FIGURE 4: TWO JOINT DESIGNS FOR THE TETRAHEDRAL ROBOT. DESIGN A INTEGRATES WITH TABBED CONNECTORS AND DESIGN B WITH THREADED CONNECTORS.

FIGURE 5: LINEAR ACTUATOR ASSEMBLY. (L) VIEW A SHOWS THE OLD CONFIGURATION WITH A SINGLE LINEAR ACTUATOR. VIEW B SHOWS THE NEW DESIGN PROVIDING ADDITIONAL EXPANSION BY 1:3 EXPANSION BY ATTACHING TWO LINEAR ACTUATORS. (R) VIEW A IS THE INTERIOR OF ONE SIDE OF THE HOUSING. VIEW B SHOWS FULL LINEAR ACTUATOR AND HOUSING ASSEMBLY.
• **Linear Actuators with Housings**
  
The active links, composed of linear actuators, drive the motion in the tetrahedral walker. In order for the tetrahedral robot to locomote by rolling it needs to shift its center of mass significantly; this requires significant expansion of the linear actuators. In previous designs of the walker the configuration of the single linear actuator did not supply sufficient expansion for locomotion. Thus twice the number of actuators were used and arranged to provide a 3:1 expansion of the structure. This change is described in Figure 5L. With this new design utilizing two linear actuators (produced by Firgelli), housings were needed to securely hold the actuators together. The housings were designed to prevent sliding of the actuators as well as provide smooth rolling surface for the walker (Figure 5R).

**Different Configurations and Locomotion**

After being fabricated the modules were assembled into the tetrahedral configuration. The robot was put through many gait cycles. The result can be seen in Figure 8 and movies are available at [5]. These results illustrate that the tetrahedral robot was capable of rolling locomotion. It was able to maintain a rigid pyramidal shape while at the same time deforming its body to shift its center of mass, thus satisfying the requirement of compliance and rigidity. In the process of testing the robot was disassembled, reassembled and tested for multiple hours suggesting the parts are robust.

The modules were next rearranged into a square formation. This new formation allows for a variety of different motions including crawling (Figure 7) and climbing (Figure 8). The crawling locomotion is based on a central-pattern-generator (CPG) style movement described in more detail in [6], where we show that the same control algorithm allows locomotion for more complex, and asymmetric, configurations of the square linkages. The climbing robot uses a similar periodic motion, and in open-loop form can robustly climb the padded tube at 45 to 90 degree angles with only occasional slippage. A more redundant design (more active links) and the use of pressure sensors could potentially make this locomotion even more robust.

In addition to demonstrating each locomotion technique multiple times, this modular robot design has been assembled and reassembled many times, and has been transported and demonstrated at robot exhibitions. The modules have proven to be very robust, and to date no joint breaks have occurred (a sharp contrast from the implementation in [4]); the robot has also proven to be easy to disassemble, reassemble and transport. One of the areas for future improvement is in the design of passive telescoping or spring-based modules, to allow a wider range of flexible structures and potentially easier locomotion algorithms. Another area for improvement is in the design of pressure sensors that interface well with the joints; this requires flexible sensors that can wrap around the joints and current low-cost pressure sensors are not reliable enough.
FIGURE 6: TETRAHEDRAL ROBOT IN MOTION, SINGLE GAIT CYCLE: 1. STRUCTURE AT REST 2. STRUCTURE EXTENDING 3. STRUCTURE FLOPS OVER 4. STRUCTURE LANDS AND RIGHTS ITSELF.

FIGURE 7: 2-SQUARE ROBOT IN MOTION: THROUGH LINEAR ACTUATION THE ROBOT IS ABLE TO SHIFT ITS WEIGHT AND CRAWL FORWARD.

FIGURE 8: 1-SQUARE ROBOT CLIMBING: BY USING LINEAR ACTUATION TO APPLY PRESSURE TO THE WALLS THE SQUARE ASSEMBLY IS CAPABLE OF CLIMBING UP A CHIMNEY-LIKE STRUCTURE.

FIGURE 9: OTHER POTENTIAL EXPANDABLE STRUCTURES: (L) HOBERMAN™ SPHERE, WHERE THE SCISSOR-LIKE LINKS BEHAVE SIMILAR TO AN ACTIVE LINK THAT CHANGES LENGTH. (2) AMORPHOUS 2D ROBOT COMPOSED OF SQUARE UNITS [6].
Through these different locomotion examples, we have demonstrated some of the potential for modular-expandable robots. We have successfully implemented three different forms of locomotion using the same modular hardware, which brings us significantly closer to realizing the behavior suggested in the original NASA concept video for an expandable robot. The possibility of linear actuation in expandable modules also has potential outside of the arrangements presented here. For example, other expanding modules such as the Hoberman Sphere could be deployed using linear actuation.

REFERENCES


ACKNOWLEDGEMENTS
This work was funded in part by the Wyss Institute for Bio-inspired Engineering at Harvard University, the Harvard REU program, and an NSF Emerging Models and Technologies Grant #CCF-0829745.
Abstract—This paper introduces a new class of modular robots, called: “layered heterogeneous modular robots”, which is a type of modular robot, where the functionality of a robot is modularized into three layers of heterogeneous modules: mechanics, actuation and electronics. This novel approach may make it possible to create dynamic, power-efficient and robust locomotive modular robots, extending the usability of modular robots. Early tests show that the system is able to perform dynamic locomotion with speeds up to 11.8 cm/sec with a specific resistance of 9.65. Also static structures have been constructed, forming a tower that is able to withstand a load of 29 times its own weight placed on top of the tower, without any power consumption. These tests show that the system is comparable in performance to those of non-modular robots.

I. INTRO

In robotics there is a special field called “Reconfigurable Modular Robotic Systems”. This field has been inspired by the way nature is built at a cellular level. In nature, complex organisms like mammals contain billions of cells, but only a few hundred different types of cells are found among them. These few hundreds of different cells are the basis of all living creatures. This idea of having a building set, only containing a few hundred different elements, is very interesting to the robotic science field, because it opens a new way to create robots. A way where robots are made from a number of smaller robots which together form the desired robotic structure. In robotics, this approach is called “Reconfigurable Modular Robotic Systems” and has been a field of research for the past 25 years [1]. As opposed to ordinary robots, where a robot is built for one purpose only and is a rigid structure, a modular robot is built from many smaller homogeneous or heterogeneous robots, where each module carries its own power supply, processing, actuation and connection mechanisms in order to connect to other modules. The modular robot can then change its shape into different morphologies dependent on the task the robot has to accomplish, e.g. a snake in order to crawl through a hole, or a walker to travel great distances with high speed. Basically, all these tasks are accomplished with the same set of modules, just configured in different ways. The versatility of such systems gives some advantages over conventional robots known as the three promises of modular robots [4]:

- **Versatility**: The possibility of using each module in many different configurations gives the system a high degree of versatility.
- **Robustness**: Ordinary robots are very vulnerable to damages on the robot. Only one failing sensor could cause the entire robot to fail working. In modular robots this problem is much smaller because of the redundancy. If a module breaks it can easily be replaced by another, even during run-time.

- **Low cost**: As modular robotic systems are composed of homogeneous modules, or a few heterogeneous modules, the modules can be mass-produced.

In the ideal world this would be a perfect solution to the problem of how to create cheap and versatile robots, but the past has shown us that in reality it is not as easy as it may seem. The problems that this vision also holds are the following:

**Problems 1 (Self-sustainability)**: Due to the fact that all modules should be self-contained in order to function as individual small robots forming a bigger structure, all modules will have to contain electronics, batteries, actuation etc. This gives a great overhead in terms of size, weight and power consumption. Some modules in a structure may never even move; they are just part of the structure, carrying the weight of their actuators, and consuming the power necessary to keep the module running in standby mode.

**Problems 2 (Granularity)**: When constructing modular robots, each module should be as small as possible to make it feasible to construct gripping mechanisms, tools, etc. from them. This however has proved to be one of the biggest challenges of all, because the technology for doing that is not yet here. The smallest modules that are self-contained, are still in the 5-10 cm range or bigger, making it impossible to create such structures. The technology will have to facilitate production of a complete module in the millimeter scale or nanometer scale for that to be possible. This is a limiting factor to the versatility of the modules.

**Problems 3 (Performance)**: Modular robots have shown that they are capable of doing various tasks, with different configuration, which is one of the promises of modular robots. However, in order to make modular robots competitive to ordinary robots, the performance of the different tasks that modular robots can do will have to be improved in terms of speed, energy efficiency and reliability.

The problems stated in problem 1-3 are the most limiting ones that modular robots are facing today. In order to solve those, we have formed the following hypotheses.
Based on the hypotheses stated above, this paper describes a new way of designing modular robots by introducing a new class of modular robots, called “layered heterogeneous modular robots”.

II. RELATED WORK

Modular robots have been around for the past 25 years, and a variety of different types of modular robots have been developed in this period. This paper will mainly relate to chain-based modular systems such as CONRO by Shen et al. [2], CKBot by Yim et al., SuperBot by Shen et al. [3], Topobo by Raffel et al. [5] and PolyBot by Yim et al. [6]. Common for these systems is the fact that they mainly have been developed with some sort of locomotion in mind, either snake-like locomotion or by walking on either two, four or six legs. All of them have shown locomotion capabilities in different configurations, and thereby demonstrated that nature’s principle of having a few different building blocks to create a variety of different structures also makes sense in the world of robotics.

Another category that is worth studying is construction kits, aimed at children as toys. The reason why it makes sense to look at this category is that such kits consist of different components (rods, joints, motors etc.) like in modular robots, and that each kit contains a large number of different elements, which increases the number of construction possibilities. The construction-kit category embraces many different kits, but the following have had the main sense to look at this category is that such kits consist of different building blocks to create a variety of different structures also makes sense in the world of robotics.

These three kits make it possible to build different kinds of legged locomotive creatures, perhaps even in a bigger variety than in the ones mentioned above, possibly because of the large number of building blocks available.

Modular robots have shown locomotive capabilities with different configurations, where gaits from animals have been implemented, thus showing the potential of modular robots as elements of a walking structure. Construction kits, on the other hand, have a unique ability to form static structures with great strength from only a few simple components.

Based on the hypotheses stated above, this paper describes a new way of designing modular robots by introducing a new class of modular robots, called “layered heterogeneous modular robots”.

Hypothesis 1 (Increase heterogeneity): The problem of having an overhead is biggest when dealing with homogeneous modular robots, but does still exist, to some degree, in heterogeneous modular robots as well. To overcome this problem, the degree of heterogeneously must be reevaluated in order to create even more specialized modules, where the complexity in each module is reduced to a minimum. By doing that the overhead in terms of electronics and mechanics can be minimized drastically.

Hypothesis 2 (Reduce complexity of modules): As the complexity of the modules rises, so does the size. Therefore it will help to decrease the complexity of the modules, and create more specialized modules in order to make each individual module smaller.

Hypothesis 3 (Narrow area of application): The diversity of tasks that a modular robot can do is huge, so in order not to get lost in trying to increase the performance of modular robots in all of their areas, this paper will focus on one specific task, namely legged locomotion, and try to make modular robots better in terms of embodiment, including robustness, energy efficiency and adaptation.

Based on the hypotheses stated above, this paper describes a new way of designing modular robots by introducing a new class of modular robots, called “layered heterogeneous modular robots”.

III. DESIGN

Modular robots face a number of problems, which are (1) the overhead in terms of electronic, actuation and structure, (2) the granularity and (3) the performance in terms of speed, energy efficiency and reliability, as stated in section I. A number of different attempts have been made over the years to create a system that can solve these problems. This section will present the ideas and visions behind a novel layered heterogeneous modular robotic system called LocoKit. Our motivation for creating a new modular robotic system comes from an interest in studying the interaction between morphology and locomotion. More specifically, we plan to address the following three areas:

- Different morphologies: The system should facilitate the possibility to study different morphologies, and to see how changes influence the locomotive abilities.
- Adaptivity through morphosis: On top of locomotion, the system should also be capable of doing voluntary and involuntary morphosis in run-time. This could be like changing gait parameters while moving on different surfaces, or to adapt if a leg falls off or becomes useless. These adaptive features should not be implemented in control only, but also as part of the structure.
- Dynamic locomotion: To perform locomotion, which is energy efficient, dynamic and adaptive, the system should be interacting with its surroundings, like in nature where the animals adapt their gait pattern to the environment.

Common to all previous modular systems is the fact that they have all proved that the concept of modularity works, and produces results in terms of different kinds of locomotion either as walking or crawling gaits. However, they are still limited by the problems stated in section I. The difference we want to make in the creation of this new system is stated in table I. What is seen from the table is that LocoKit is combining features from both of the other two categories in order to gain what is best from them to fulfill our needs. The area of self-reconfigurable modular robots, has not been discussed until now. A modular robot of this category can change its own shape from e.g. a walker to a snake if it finds this kind of locomotion more appropriate for its current conditions. However, this ability requires the modules to be highly homogeneous in order for the robot to self-reconfigure, which is why this category cannot currently be
combined with our design, because the overhead in terms of electronic, actuation and structure is too huge for our purpose.

A. Modules in the system

Locomotion in robots can in general be split into three groups, namely structure, actuation and electronics. The structural parts are needed to build the skeleton of the desired robot. This could be a robot for legged, jumping, crawling or sidewinding locomotion. When the skeleton is built with its links, joints, and rods some actuation will be needed in order to make it move. With the skeleton and actuation in place, the only thing missing is the electronics, which together with sensors will control all the actuators on the robot and make it walk, run or jump. The following will describe the first iteration in the creation of the new layered heterogeneous modular robot, LocoKit, guided by the hypotheses in section I.

1) Structural parts: To form the skeleton of the robot, a number of small passive building blocks have been created. As opposed to many other modular robots, this system does not form rigid structures, but instead structures that allow for some flexibility. This is controversial, opposed to other systems, where a stiff structure is desirable in order for the system to align its modules for self-reconfiguration, but since this system is only reconfigurable, it is seen as an advantage, because it allows us to create dynamic structures for dynamic locomotion, as is one of our requirements for this system. Figure 1 shows the three components used for structure creation in the system. As connection between joints in the structure, glass-fiber enhanced plastic rods have been used. These are seen in e.g. figure 1(c). This type of rod was selected because of its flexibility and strength. Figure 1(a) shows a freely rotary joint that links two rods together. This joint also contains mounting points, where wires for actuation can be mounted. This joint is used when it is desirable to actuate the joint. In figure 1(b) a fixed joint is shown. This joint can be fixed in 12 different positions, and its primary use is to build the rigid part of the structure. All structural components are mounted firmly onto the rods with screws.

2) Actuation: Now with all the structural components in place, forming the structural basis of the robot, it will need some actuation in order to move. Actuation will have to be applied at the joints where it is needed, but opposed to placing actuators directly onto the joint, this project have been inspired by nature in its way of transferring power from the muscles and onto the joints. In LocoKit, actuation power are transferred from the actuators to the joints via wires. A principle that has been used in e.g. model airplanes, robot hands [17], we now introduce into modular robots. The purpose of not placing the actuator directly at the joint is to minimize the weight and size of each construction module and to keep the weight of the actuators in the centre of the robot. It also allows for a greater control of the dynamics of the robot in terms of weight distribution, because we can place the actuators as desired because of the wires. Transferring the power from the motors onto the joints via wire systems, however, introduces friction to the system every time the wire is bent in some way and also makes each movement less precise. These problems will be addressed at a later point of time. Figure 2 shows configuration examples of the actuation.

3) Electronics: Normally, the structure, actuation and electronics would have been one module, but now it has been shown how structural parts and actuation can been split into separate layered modular systems. The same is true for the electronics, but the electronics has been further divided into separate modules, each representing different functions. This way of dividing the electronics makes it possible to use

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Modular Robots etc.</th>
<th>Erector etc.</th>
<th>LocoKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Granularity</td>
<td>high</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>Degree of heterogeneity</td>
<td>low</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>Deformability</td>
<td>low</td>
<td>variable</td>
<td>variable</td>
</tr>
<tr>
<td>Module autonomy</td>
<td>high</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>Size</td>
<td>high</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>Degrees of freedom</td>
<td>high</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Ease of assembly</td>
<td>high</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Versatility</td>
<td>low</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>Cost</td>
<td>high</td>
<td>low</td>
<td>medium-low</td>
</tr>
<tr>
<td>Self-reconfigurable</td>
<td>medium</td>
<td>some</td>
<td>none</td>
</tr>
</tbody>
</table>

Table I: Comparison between non-layered modular robots, toy kits and LocoKit.
the exact amount of electronics needed at specific places in the robot, and thereby limiting the amount of electronics in the robot. The electric boards have been made so that they can be stacked to form a sandwich structure - see figure 3. One board (the CPU board) is always required to be in the sandwich, but depending on whether the sandwich should be controlling actuation, sensors or just be a computation unit, it can be configured accordingly. Having this sandwich structure makes it easier to create new electronics for the system. It simply just have to fit with a set of pin connections and electronics specifications. In the future there will be boards for communication, sensors and motor control. The electronics for LocoKit is based on the electronics designed for the ODIN modular robots [7].

IV. OBSERVATION / EXPERIMENTS

In this section we will present some experiments that demonstrate the versatility of the LocoKit. Placed between ordinary modular robots and construction toys for kids, LocoKit has the capability of forming structures from both of these worlds, as experiments will show. We will document the following two configurations, (1) Quadruped walker, (2) Static tower.

A. Quadruped walker

To demonstrate locomotion, a quadruped walker has been constructed. The walker has been constructed in a way that makes it possible to use only two motors to actuate four legs. This approach will make the robot less heavy and more power efficient, because the speed of the robot will not be increased by the addition of another two motors, since the step length is limited by the way motors are attached to the legs. The control of the motors is implemented in a simple manner where the motors are oscillating between two fixed angles. Then by connecting the output from the motors, via wire cables to the legs, the actuation power is transferred from the motor onto the legs and thereby allowing the motors to be kept in centre of the robot. The legs are attached in pairs of two to the motors. One pair is the back legs, and one is the front legs. The legs are coupled in anti-phase. To create a gait, the two motors are oscillating with a phase shift, and thereby creating a simple walking gait. The gait was optimized based on the speed and specific resistance, and measurements shows that the selected gait gave the highest speed with the lowest specific resistance, which tells us that the dynamics of the robot helps to improve the performance - see figure 4(b). The speed of the walker was measured to 11.8cm/sec with a specific resistance of 9.65.
The specific resistance [9] was calculated based on the total power consumption of the robot when walking, calculated by equation (1).

$$\epsilon = \frac{E}{Mgd}$$  \hspace{1cm} (1)

The specific resistance in (1) is a dimensionless number. This equation is useful in the evaluation process of mobile robots because it makes it easier to compare performance between different robots. Here, “E” represents the total energy consumed when traveling a distance of “d”. The mass is “M” and the gravitation is “g”, [9]. It is, however, difficult to make a good comparison to other modular robots because very few have actually made such measurements. Measurements carried out by Sastra et al. [8] with a loop configuration are the only one available, but as the configuration is completely different a comparison would be meaningless, because the configuration makes the robot more efficient. To do a fair comparison to other modular robots, the configurations would have to be somewhat identical, e.g. a legged robot.

When comparing to other quadruped robots that are non-modular, LocoKit is not performing as well as them, which was also expected, because LocoKit is still at a very early state of its development. Most other quadruped walking robots have a specific resistance in the area of 1-10 [10]. However, tests show that LocoKit is able to produce locomotion like other modular robots, and has a performance that is comparable to that of other quadruped robots.

B. Static tower

In this configuration the components are forming a simple tower that are hold in place by four wires attached on two sides of the tower. In figure 5(a) the tower is not stable when the wires are not attached on either of the two sides. In this situation, the only components holding the tower together are the connection joints mounted onto the rods with screws. When the wires are attached, the tower becomes stable, as seen in figure 5(b). In this configuration the tower is able to sustain a load on top of the tower of maximum 7.7kg. The tower by itself weights 262g, which is equivalent of the tower carrying a weight, 20cm over the ground, of roughly 29 times its own weight. Drop tests have shown that the tower could sustain a drop from up to 60cm and still withstand a top load of 7.7kg. With loads higher than 7.7kg or a drop from more than 60cm a random rotary joint would simply break. The broken joint could then be replaced and the tower would regain its strength. This test shows that LocoKit is able to be configured into a stable structure, with the ability to withstand great loads without using any power. It also shows that LocoKit has some of the same capabilities as construction kits like Erector and Meccano.

V. CONCLUSION

This paper introduces a new category of modular robotic systems, which we call “layered heterogeneous modular robots”. The system is divided into layers of heterogeneous modules, namely “Actuation”, “Electronics” and “Structure”, in order to lower the amount of overhead of electronics, motors etc. that normally is seen in modular robots. The purpose of this novel layered system is to create dynamic modular robots with increased locomotive capabilities. Tests have shown that the system at this early state is able to produce walking locomotion with speeds up to 11.8cm/sec and a specific resistance of 9.65. These tests also show that the dynamics of the system contributes to the locomotive performance and makes the performance of the robot comparable to that of other non-modular robots. Stable static structures build with LocoKit have also been tested. These tests show that the system is able to form lightweight structures that are able to carry up to 29 times their own weight 20 cm over the ground. These experiments support our hypothesis that layered heterogeneity increases the versatility of modular robots.

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

REFERENCES


[8] Jimmy Sastra, Sachin Chittay, Mark Yim: Dynamic Rolling for a Modular Loop Robot - GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA


[17] Shadow Robot Company: Design of a Dextrous Hand for advanced CLAWAR applications - 251 Liverpool Road London ENGLAND

ICRA 2010 Workshop "Modular Robots: State of the Art"
Cellular Slime Mold Robot

Eunjeong Lee, Jesse Yang, Matthew Jolda, and Robert Wood

Abstract— This paper presents the fabrication and actuation of biologically-inspired cellular Slime Mold robots. Their body is made of flexible polymer with shape memory alloy wire springs imbedded inside as muscles. Slime Mold robot is a modular robot with remarkable flexibility, shape change, and integration capability. By actuating different sections of the springs, several different shapes can be created from the same module, resulting in shape and size change capability. Hence, it can enhance versatile assembly with other Slime Mold modules. It has dual modes of locomotion: crawling and swimming. Experiments have shown their crawling locomotion and grasping capability successfully.

I. INTRODUCTION

Cellular Slime Mold, Dictyostelium discoideum, has a unique series of developmental events that occurs when they are deprived of bacteria. The cells begin to associate, forming streams of migrating cells which merge in an aggregate consisting of up to 100,000 cells [1]. We have begun the development of a biologically-inspired cellular Slime Mold robot to imitate this behavior. Previous biological studies, as well as previous work on compliant robotic systems, have demonstrated that the mechanical properties of the outer surfaces of mobile systems can contribute to effective locomotion and configuration changes [2]. To develop a soft bodied robot capable of locomotion and shape changing, one of the most valuable capabilities is the spatial control of variable compliance and topologies in a soft material. Here we describe the design and fabrication of one such soft robot, a cellular Slime Mold robot. We also describe a technique to actuate motion of the body to create configuration changes and present experimental results of its locomotion and connection capability.

This paper is organized as follows: Section II describes the design of the cellular Slime Mold robot. Section III explains the fabrication process. Section IV presents experiments, while Section V discusses conclusions and suggestions for future work.

II. DESIGN

The Slime Mold robot is a rectangular shape silicone rubber with three links or sections as shown in figure 1. The muscles of the Slime Mold robot consist of inverted shape memory alloy (SMA) wire springs. There have been some SMA-driven elastomeric robots, such as [3,4]. However, they do not use elastic body with inverted SMA wire springs to imitate behaviors of animals with flexible bodies. The springs are stitched inside the longitudinal grooves on both sides for greater maneuverability. Each groove has a SMA spring with three sections, which can be actuated separately. Through current excitation in the sections of the springs, the Slime Mold robot can be controlled to bend in the desired direction.

 Manuscript received April 11, 2010.
 Eunjeong Lee is with the School of Engineering and Applied Sciences at Harvard University, Cambridge, MA 02138 USA (phone: 617-384-7892; fax: 617-496-9083; e-mail: ellee@seas.harvard.edu).
 Jesse Yang is with the School of Engineering and Applied Sciences at Harvard University, Cambridge, MA 02138 USA (e-mail: jzyang@fas.harvard.edu).
 Matthew Jolda is with the Department of Electronics & Mechanical Engineering at Wentworth Institute of Technology, Boston, MA 02115 USA (e-mail: mjolds@gmail.com).
 Robert Wood is with the School of Engineering and Applied Sciences at Harvard University, Cambridge, MA 02138 USA (e-mail: rjwood@eecs.harvard.edu).

Fig. 1. Pictures of the silicone-molded Slime Mold robot with stitched SMA wires and wire connections (60 mm x 20 mm x 4 mm).

Fig. 2. Different shapes of the Slime Mold robot module. A) Actuation of only the bottom middle section of the SMA wire, B) Actuation of only the top middle section, C) Actuation of all three sections of bottom wires.
Currently, the perpendicular grooves are used to facilitate easy folding. In the future, it will be used to house SMA wires for steering.

A. Modular Shape Change

Robots which mimic the composition and locomotion capability of *Dictyostelium* Mold may travel and perform tasks in unknown and unstructured environments. To meet this challenge, it is important to allow the Slime Mold modules to change into as many shapes as possible. By actuating different sections of the springs, we can create several different shapes, and control the size of the individual Slime Mold modules to make it smaller or larger. Figure 2 depicts examples of different shapes the Slime Mold module can form. They are similar to the most basic unit of organism DNA origami in shape [5]. It suggests that it can enhance versatile assembly with other Slime Mold modules.

B. Locomotion

The Slime Mold robot is designed to have two different modes of locomotion: crawling and swimming. Unlike Shear flow-induced motility of *Dictyostelium* discoideum cells on solid substrate, the Slime Mold robots crawl by changing their morphology [6]. It is also inspired by the behaviors of fire ants. When there is flood to endanger them on the ground, they aggregate to form a ball-like shape collectively to float over the water. While floating, the ants continually rearrange their position to distribute water exposure among its elements. When the water level subsides, they disintegrate to crawl away individually [7]. It may be possible for the Slime Mold robots to swim individually or collectively if they learn from fire ants when they encounter water.

1) Crawling: The Slime Mold module is designed to move on solid ground by sequentially bending its links. The Slime Mold moves by performing six shape changes. The last two sections help control the friction forces between the Slime Mold surface and the ground it travels on. The center section is the primary motor that moves the Slime Mold robot forward in a given direction. Steering is controlled by the perpendicular springs that bend the Slime Mold modules in the direction perpendicular to the longitudinal travel direction.

Theoretically, the Slime Mold module should be capable of moving a fraction of its body length after it performs two full cycles of body movement as in figure 3. The steering is controlled by which perpendicular spring is activated.

2) Swimming: A fish swims by oscillating control surfaces, such as its main body and fins. The flexibility of a fish makes it possible to interact with surrounding fluid in such a way to make effective use of hydrodynamics around them [8]. The Slime Mold robot with flexible body can swim like fish by undulating its body. This action requires the simultaneous use of springs on both sides at significantly higher frequency. The typical fin/tail beat frequency is within the range of 2-7 Hz, during steady swimming and maneuvering, for fish of comparable size to the Slime Mold with Reynolds number greater than 1,000 [9,10,11]. The first prototype can meet this frequency easily since ambient water provides excellent cooling for the SMA wires. Steering for swimming can be controlled by both perpendicular and longitudinal springs.

C. Aggregation and Disintegration

Slime Mold aggregate and migrate for mutual benefit and survival [7]. By changing the overall size and shape of the collective body, they can perform a broader range of tasks and have different modes of locomotion, which were not possible as individual. The viscoelastic silicone body produces large friction upon contact, which helps physical connection with other Slime Mold modules. We can create a larger object if we can link these small Slime Mold modules together. For example, the Slime Mold can link together by curling two sections at the end and locking with each other at those areas, whereas two proteins mediate cell-cell adhesion between the Slime Molds as they form a loosely packed multicellular mass [12]. Aggregating the Slime Mold is beneficial, since it can make them move farther with less power or provide a better method of traveling together. One such possible method to travel together as a group may be obtained by employing Octopus walking gates [13]. Aggregated Slime Mold can be disintegrated by simply switching off the SMA wire actuators, releasing the locked or latched body parts.

III. Fabrication

A. Shape Memory Alloy Actuator

Actuator is chosen based on maximum elongation, robustness, ease of fabrication, power requirements, and cost. The actuator for the Slime Mold robot is a shape memory alloy (SMA) spring made of Nitinol wire. NiTi coil springs are created from straight wire by winding it into a coil shape and annealing approximately at 410 °C for 40 minutes to reset the memorized shape. These NiTi coils result in a 60 percent shape change capability, which is a great improvement over the 5 percent shape change of a straight piece of NiTi wire. Methods have been developed to further improve the response of the coils by inverting or turning the springs inside out. These new Inverted SMA springs are capable of more than 90 percent shape change. There are two methods of inversion as proposed in [14].
The method used for the Slime Mold robot is the central tube method as shown in Figure 4. After heat-treating the SMA spring, this method requires pulling the spring right through its own center in order to invert it. To prevent the coils from getting tangled in each other, a thin-walled plastic tube can be inserted through the center of the spring which also slightly expands the spring. A puller wire with a hook on the end is then threaded through the inside of the spring and connected to the spring at the opposite end. This wire is then pulled back through the center of the spring, in order to turn the spring inside-out, or invert it.

The SMA springs have usually been wound tightly with no spacing between the coils. The inverted SMA springs have been found to respond much faster if the original spring is wound with spacing between the coils. When the spring is wound with spacing and then inverted, the original long gap has the effect of an even tighter inverted wind, thus causing faster response time. The difference becomes very noticeable in lengths over 4 or 5 inches, with a matter of a second or two of difference. For the Slime Mold robot, 0.01 inch SMA was wound around a 0.063 inch core, with a coil gap of 0.045 inch. Because the inverted spring tends to return to the originally loose, uninverted configuration, it has a faster response/actuation time. The coil-gap-wound inverted springs are able to retract almost to the minimum spring length as shown in Figure 5.

The Inverted-SMA actuator was chosen because of its small size, flexibility, and shape change capabilities. In order to make the Slime Mold robot locomote, it needs several actuators spanning sections of its body length. This could be done by using three individual SMA springs, linked together at each end. However, this runs into the issue of shorting signals with different voltages and grounds placed next to each other as in Figure 6. A linear Inverted SMA spring is divided into three sections electrically, with voltages applied in each section as shown in Figure 6. This allows for one long SMA spring to span the length of the Slime Mold robot, while providing three independent sections for actuation. Using one long piece of SMA is also advantageous for fabrication, as it is easier to place into the mold and connect to the mold.

From this, it can be observed that each section is in actuality two resistors in parallel as in Figure 7. It is known that current flows through the path of least resistance, and thus, according to Kirchhoff’s Current Law, it is critical for each resistor in the same section to have the same resistance. Testing was done to see if this was possible for the Inverted SMA springs, in both actuation and nonactuation. The results are indicated in Table I.

From the results in Table I, it is observed that the change in resistance during actuation of the SMA is almost negligible. This means that since both SMA coils have a very close resistance, they heat and contract almost evenly, as desired. Otherwise, more current flows in one direction, resulting in the unbalanced contraction of the springs. In the future, control electronics will be made with H-bridge driver.
A method of creating such an SMA chain actuator is devised as shown in figure 8. All segments are composed of coils of exactly three turns. Each coil has a resistance of $1.3\,\Omega$. Testing of these SMA chain actuators verified that each section can be effectively actuated without affecting each other. During sequences of actuation, the activated section has no significant effect on the unactuated ones, although each section is connected to the next.

### TABLE I

INVERTED-SMA RESISTANCE TEST RESULTS FOR VARIOUS SPRING LENGTHS

<table>
<thead>
<tr>
<th>Spring</th>
<th>Stretched Resistance (ohm)</th>
<th>Actuated Resistance (ohm)</th>
<th>Mid / Cooling Resistance (ohm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.8</td>
<td>13.8</td>
<td>14.0</td>
</tr>
<tr>
<td>2</td>
<td>13.9</td>
<td>13.8</td>
<td>14.2</td>
</tr>
<tr>
<td>3</td>
<td>5.2</td>
<td>5.6</td>
<td>5.8</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>5.8</td>
<td>6.0</td>
</tr>
<tr>
<td>5</td>
<td>6.0</td>
<td>6.8</td>
<td>6.6</td>
</tr>
<tr>
<td>6</td>
<td>6.1</td>
<td>6.8</td>
<td>6.7</td>
</tr>
<tr>
<td>7</td>
<td>13.8</td>
<td>13.0</td>
<td>13.5</td>
</tr>
<tr>
<td>8</td>
<td>13.5</td>
<td>14.5</td>
<td>14.8</td>
</tr>
<tr>
<td>9</td>
<td>13.5</td>
<td>13.4</td>
<td>13.0</td>
</tr>
<tr>
<td>10</td>
<td>12.0</td>
<td>13.0</td>
<td>12.5</td>
</tr>
</tbody>
</table>

B. Silicone Molding and Integration

Our robot body material is chosen to be silicone rubber (Ecoflex Supersoft 0030). This elastomer has a small Young's modulus and hardness and is resistant to tearing and heat. The body is fabricated by molding silicone rubber. Two Mold are produced by rapid prototyping ABS plastic using a fused deposition modeling (FDM) process. The top and bottom Mold sandwich the polymer, where it cures to form the body of Slime Mold robot. The two Mold are prepared such that a polymer layer of 4 mm thickness is manufactured.

First, the design of the Slime Mold robot module, with dimension of 60 mm x 20 mm x 4 mm, is drawn in Solidworks and printed on the DIMENSION ABS 3D printer with P400 material to make Mold for the body. Then, the elastomer is poured on the bottom mold and the top mold is pressed on top of it. The molding process produces a very flexible robot body after 4 hours of curing of the elastomer. Figure 9 shows the pictures of the mold and silicone-molded body. It shows the grooves on the top and bottom surfaces to imbed SMA wire actuators.

IV. EXPERIMENTS

Experiments have been performed to test the locomotive capability of the Slime Mold robot. Figure 10 shows images of the locomotion sequences of a single Slime Mold module. The speed of the first prototype is 0.0024 cm/sec. The main cause of the slow speed is due to the fact that it takes a long time for SMA wires to be released from the contracted shape. This problem will be addressed first by different actuation sequences. Second, design and fabrication need to be improved to allow better ventilation for cooling of the SMA wires.

Also, the head or tail does not bend in a balanced manner. As a result, one side of it touches the ground and acts like a pinned joint. The viscoelastic nature of the Slime Mold surface then produces a large friction force, preventing fast motion. More refined fabrication can eliminate this unbalanced bending. The speed depends mainly on input current and control strategy, such as control electronics timing and actuation sequences. It also depends on passive...
release time of the SMA wire, as well as its actuation response time.
The grasping capability of the Slime Mold robot modules with each other has also been tested and observed. They were placed together manually so that they can be grasped when one Slime Mold module actuates. Further experiments are required to show that each module in proximity can latch onto another autonomously. As shown in figure 11, the Slime Mold modules are capable of latching onto other modules both in longitudinal direction and perpendicular to each other.

Figure 11(a) demonstrates the perpendicular grasping of two Slime Mold robot modules, whereas (b) shows the serial linking of three modules, lifted off the ground. Note that only one module is actuated to make a connection. The flexibility of the body enables the versatile grasping, which allows various modes of connection and group shape.

V. CONCLUSION AND FUTURE WORK
We have developed biologically inspired cellular Slime
Mold modules. Their body is made of flexible polymer with SMA wire imbedded inside as muscles. By actuating different sections of the springs, we can create several different shapes from the same module, and control the size of the individual Slime Mold modules. These basic unit shapes are similar to DNA origami [5]. Hence, it can enhance versatile assembly with other Slime Mold modules. Experiments have successfully shown their locomotion and grasping capability.

They require further refinement of design, fabrication, and control to realize their full potential. Future work includes integration of onboard control electronics and wireless communication devices and cooperative control of various aggregation/migration modes.

Appendix
Movie of the crawling of Slime Mold robot.

REFERENCES
Actuated Responsive Truss

Rehman Merali and David Long

Abstract—This paper presents an actuated truss that uses space more efficiently than a traditional truss, without compromising load capacity. The depth of the truss is variable and dynamically changes in response to dynamic loads. The premise of an actuated responsive truss is novel. This paper discusses the merits of a structure capable of changing its load capacity, as well as present a 30-actuator, physical model of the truss. The truss consists of simple, identical cells that work together to achieve modularity and reconfigurability, independent of a central controller.

I. INTRODUCTION

A. Intelligent Architecture

Intelligent or Responsive architecture combines built forms with integrated systems that are capable of responding to changing conditions. Responsive architecture is not merely concerned with physicality but more so the behaviour of the structure. This emerging genre of architecture is on the leading edge of an imminent paradigm shift in architectural design where buildings will no longer be static but rather dynamic objects. If buildings were more able to adapt to shifting conditions, they would be dramatically more efficient. This emergent architectural practice is generating significant interest among the architectural community because responsive architecture is not interested in aesthetics or style but rather performance.

B. Literature Survey

Intelligent architecture is much more complex than just an automated system because it exhibits complex cybernetic processes and learned behaviour. Communication of its spatial components makes the architecture intelligent [1], [2].

Topotransegrity is a project developed by the 5Subzero group in 2002 [1], [2]. The premise of Topotransegrity is to create a responsive topological surface actuated with a pneumatic spaceframe structure. The structure uses pressure sensitive mats to monitor peoples’ movements, and responds by dynamically changing floors, walls, ceilings, and the building envelope. The project does not, however, aim to optimize the structural strength of the design.

In 2004, the Non-Standard Architecture Exhibition featured The Muscle [3]. The Muscle is a pressurized volume encased in flexible pneumatic actuators. The kinetic structure flexes, expands, contracts, contorts, and mutates. The Muscle’s behaviour is not completely predictable or formulated.

II. RESPONSIVE TRUSS

The building reacts to motion, proximity, and touch sensors. The building is not designed for structural strength, nor are its transformations intended to affect its strength.

A working prototype of the winning submission to the MIT Mini-Skyscraper Design Competition was erected in 2006. MUSCLES is a 35-foot high skyscraper prototype which is designed to change posture and position [4], [5], [6]. In the application of a full scale building, the actuated skyscraper would cancel out movement and to stabilize a tall structure against external forces such as wind and earthquakes. MUSCLES is therefore designed to change its shape to improve the structural stability of the building. The structure is not equipped with sensors, hence, it does not react to its environment.

Similarly, Muscle Tower II is a tower that consists of six distinct trapezoidal segments that are stacked on top of each other [3]. The tower is also actuated by flexible pneumatic actuators. The actuators are capable of changing the strength of the structure, however it does not have sensors and is therefore not reactive.

The Actuated Tensegrity project by ORAMBRA uses linear actuators to transfer loads between tension and compression members [7]. The simplest actuated control module consists of two opposing ‘tripods’, an actuator that connects the structural units together, a processor, a sensor, and a power source [8]. The modules work together with no central processor to create an emergent behaviour. This project uses cables and rigid compression members, which means the structure is capable of supporting more load than the above mentioned projects. In contrast to the research presented here, the tension members are cables, not springs, and are consequently of a static length.

The above mentioned projects all share a common undesirable trait: the structure does not contain a static surface. When a structure is able to change shape, it introduces issues of ‘usable space’ or ‘usable surfaces’. When the floor, walls and ceilings are all changing, the surface must be covered in a stretchable membrane, which may not be able to support the required load. It would be undesirable for a floor to change shape, as it would create additional problems with the objects on that surface.

D. Long holds a Masters from the John H. Daniels Faculty of Architecture, Landscape, and Design, University of Toronto, Toronto, Ontario, Canada david.long@utoronto.ca

R. Merali is a PhD student at the University of Toronto Institute for Aerospace Studies, Toronto, Ontario, Canada rehman.merali@utoronto.ca
If a building could shift its center of gravity dynamically, like a human body, then it would not rely on an over-engineered and redundant structure to resist environmental loadings. Consider how humans react when facing broad side to prevailing wind: humans will shift their body to be more aerodynamic, reducing the amount of wind load incurred, but also widen their stance so they are more stable.

To illustrate the point, this paper will re-examine one of the most fundamental structures in architecture, the truss. The actuated responsive truss is based on the Warren truss. The Warren truss is depicted in Fig. 1. Truss forms ensure that no member experiences torsional or bending forces, but only tension or compression. The amount of load (downward force) that the truss can support, is primarily determined by the depth of the truss. The depth is determined by the length of the web (diagonal) members. Moreover, the downward loading results in axial compression forces in the top chord and axial tension forces in the bottom chord. The forces in the top and bottom chord are equal in magnitude and opposite in direction – the force couple resulting in a bending moment resisting the moment caused by the applied loading. The moment resistance is directly proportional to the perpendicular distance between these axial compression and tension forces in the top and bottom chords, respectively. The forces acting on a section of the truss are illustrated in Fig. 2.

The truss can be made more space efficient by exploiting its geometry. Namely, if a truss is designed for worst-case loading that will only occur once in one-hundred years, as often required by typical building codes [10], then the depth of the truss could be significantly less the rest of the time. Therefore, the actuated responsive truss replaces the web members with linear actuators to control the depth of the truss at any given time. Specifically, the design is a deck truss where the load is applied on the top chord – refer to Fig. 1. The bottom chord cannot be a fixed length since the web members change in length causing the distance between adjacent nodes to change as well. Thus, the bottom chord must span a varying distance. Fig. 3 illustrates the full actuated responsive truss.

A. Swarm Intelligence

Given a point load on the top chord, both webs (actuators) below that load should extend to the same length to distribute the load. To achieve this, the actuated responsive truss is implemented as a swarm; a single cell in the swarm consists of two actuators and a load sensor (positioned between the two actuators, on the top chord). The basic premise of swarm intelligence is that each cell performs a simple task, and has no global knowledge of the entire structure. Each cell responds to a simple set of behavioural rules which permits the group to collectively coordinate its activities and build a global emergent structure. Fig. 4 details the design of a single cell.

III. PROTOTYPE

To demonstrate the actuated responsive truss, a model using 30 actuators and 15 sensors (15 cells) was built. Each actuator is an electric linear actuator – hydraulic cylinders may also be used to support more load and react more quickly. Strain gauges could measure compressive strain in the top chord, but they would come at significant cost. The model uses inexpensive force sensors to measure compressive force in the top chord. Cutting the top chord at each

ICRA 2010 Workshop "Modular Robots: State of the Art"
node (web pivot joint on top chord) yielded poor contact on the force sensor and was difficult to construct to reflect the idealized truss configuration. Consequently, a continuous top chord was used, and a slit was cut in the upper half of the top chord between adjacent nodes. The slit allowed the top chord to deform and apply force to the sensor. The bottom chord is always in tension and required to vary in length. Therefore, surgical tubing was used, which acts as a spring. Fig. 5 shows a time sequence of how the actuated responsive truss reacts to load on the top chord. Essentially, the application of load results in internal axial forces in the chords and web members. In the case of a uniformly distributed and equal downward force, the top chord is put into compression, the web members resist alternating compression and tension forces, and the bottom chord is put into tension. The compression force in the top chord is sensed by the force sensor which controls the length of the actuators, and thus the depth of the truss. As a result, the measured force in the top chord dictates the distance between the chord forces. For a given loading, and therefore bending moment, the magnitude of the chord forces are inversely proportional to the distance between them – see Fig. 2.

The control of each cell is intentionally simple, so it may be implemented in analog circuitry in the future. Both actuators extend to the same length using PID control. As the compression in the top chord increases, the actuators extend to reduce this compression. If the compression is low, the actuators retract to minimize the volume of the truss. The 30-actuator prototype consists of three actuated responsive trusses of five cells each. To experiment with cooperative behaviour, each five-cell truss is controlled by a single microcontroller. The code still functions as described, but this allows ease of implementation and cooperative behaviour in future research. For example, sharing information between neighbouring cells can be used to preempt a moving load. Fig. 6 is a photograph of the final working prototype as described. The blue LEDs on the top surface simply display the reading on each force sensor.
IV. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

The actuated responsive truss is a novel contribution to the field of intelligent architecture. Its actuation is not aimed at aesthetics, but rather performance. The truss allows varying load resistance, and a modular design that allows independent cells to act individually to create a cohesive behaviour.

The truss incorporates the latest research trends of modular and reconfigurable architecture, while still maintaining immediate practicality. The static upper surface allows seamless integration with traditional architecture. Also, given the wide and varied use of trusses, the possible applications for this technology are far-reaching.

For example, if the actuated responsive truss were used as a bridge or roof, the top surface could be covered by a static material – not unlike traditional architecture. To illustrate the point, Fig. 7 depicts a traditional convention center where the trusses supporting the lower ceiling have been replaced with actuated responsive trusses. The result is a higher than normal ceiling in the lower level, while the upper level is not at its maximum load (which would be the majority of the time). As the upper level approaches its worst-case loading, the lower level simply has its ceiling automatically lowered to a ‘normal’ height.

B. Future Works

Future research on this project includes sharing information between cells. Sharing compression information, actuator length, or the time derivative of either may be useful in creating a more cohesive truss. The swarm intelligence can be improved by altering the rules that each cell follows. More data may encourage the cells to extend the actuators to different lengths, thus varying the angle of the web members.

A new prototype would also better identify the strengths and weaknesses of the actuated responsive truss. Specifically, a model of a more idealized truss with pivot joints at each node. Another prototype may be constructed of steel, with hydraulic cylinders as actuators and more accurate force sensors. The bottom chord may also be replaced by linear actuators with their own force sensors. Therefore, its length could be variable as required, and the additional force information could also be fed to the respective cell’s controller.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge the advice of Kevin Schorn, P.Eng., a structural engineer and architecture student, for his invaluable insight. The authors also thank Keith Lee and Seyed Mahdi Olfat (University of Waterloo engineering students) for their help in constructing the prototype. This research was funded in part by The Royal Architectural Institute of Canada through the Canada Green Building Council Scholarship awarded in 2009. The Actuated Responsive Truss project was research conducted in fulfillment of David Long’s Master of Architecture thesis under the guidance of Professor Rodolphe el-Khoury. The prototype would not have been possible without the time and resources.
of the University of Toronto Robotics Association. Vahid Mashatan and Meng Tang were an integral part of the initial design team.

REFERENCES


Development of a Reconfigurable & Modular Mobile Robotic Platform

Taşkin Padır, Matthew R. Bienia, Daniel R. Garcia and Karl B. Wajcs
Robotics Engineering Program
Worcester Polytechnic Institute
Worcester, MA 01609, USA
{tpadir, mbienia, dan.r.garcia, kwajcs}@wpi.edu

Abstract—This paper discusses the development of a reconfigurable modular mobile robotic platform (ReMMRP). The reconfigurability is accomplished by employing a universal hardware, software and communications scheme between the peripheral modules and the robot. In contrast to some existing reconfigurable robots which utilize modular joints, in the ReMMRP, the modules can be articulated legs or arms, wheels, sensors, or another robot. The paper will specifically present the hardware and software design for the robot body and a 3 degree-of-freedom (DOF) leg module. The communications between the leg module and the robot’s body is also discussed in detail. Finally, a complete kinematic model for the system is presented that takes into account the kinematic constraints present in ReMMRP.

Index Terms—reconfigurable robot, modular robot, inverse kinematics, pseudovelocility

I. INTRODUCTION

As the robotics community celebrates "50 years of robotics" [1], there is no doubt that research and development in the field has evolved drastically since the introduction of the first industrial automation robot, the Unimate. With the advances in enabling technologies (electronics, hardware and computation) and components (sensors and actuators), intelligent vehicles are capable of assisting human drivers in urban environments, vacuum cleaning and lawn mowing robots are becoming a common household appliance, medical and rehabilitation robots are assisting with elder care, and search and rescue robots are helping first responders.

The development and control of (self-)reconfigurable and modular robotic platforms have emerged as a new research area in robotics within the past two decades. The field addresses new challenges that come with the design, modeling, implementation and control of autonomous robots whose kinematic structures can vary over time depending on the physical environment that they are in [2], [3], [4], [5]. The reconfigurable modular robots have two important features which make them desirable in applications: flexibility and robustness. They can adapt their shape and form with respect to changes in their environments and they can accommodate failures within modules provided that they possess redundancy [2].

Modularity is due to reusable building blocks with well-defined mechanical and communication interfaces forming the robot. Modularity allows for low-cost development, reusable hardware and software components, ease of maintenance as well as improvements in design time and effort. Reconfigurability is the ability to bring together modules in various configurations to build a robot that can perform a specified task. In this paper, we present the development of a reconfigurable, modular, and mobile robot designed as a research platform to explore kinematic redundancy, adaptive control and motion coordination in reconfigurable robots.

Similar platforms have been discussed in literature. For example, [3] presents a hierarchical design methodology for a reconfigurable walking robot. The approach involves a layered architecture from functional components to modular units to the completed robot system. The design of a reconfigurable planar parallel manipulator is discussed in [4]. The platform can be reconfigured into 3 different revolute-jointed mechanisms for studying redundant actuation. It is noted that the reconfigurability is accomplished by changing the manipulator kinematics only. [5] presents the control system for a multilegged mobile platform in which joint modules make up the leg modules and the leg modules form the scalable body module. The robot can be configured into a 4-, 6-, 8-, etc. legged robot using the body modules. The paper also demonstrates a distributed hierarchical control system for the gait control of a quadruped and a hexapod configuration.

The reconfigurable modular mobile robotic platform (ReMMRP) being developed at Worcester Polytechnic Institute is introduced here. The reconfigurability is accomplished by employing a universal hardware, software and communications scheme between the peripheral modules and robot’s body. In contrast to some existing reconfigurable robots which only utilize modular joints, in the ReMMRP, the modules can be articulated legs or arms, wheels, sensors, or even another ReMMRP robot. The paper will provide the details of the mechanical hardware, electronics and communications between the peripheral devices and the robot itself. We will also introduce the 3 degree of freedom (DOF) leg module and propose a method for kinematic redundancy resolution for establishing a balanced posture of the body when more than two legs are present.

The paper is organized as follows. Section II will discuss the mechanical design of the robot including the body and a leg module. Section III will discuss the on-board electronics and individual control units. The adopted communications
scheme between the peripheral modules and the body is introduced in Section IV. We will present the 3-DOF leg module, introduce its kinematics and discuss the kinematic redundancy resolution in Section V. Finally, Section VI will summarize the results.

II. ReMMRP MECHANICAL DESIGN

Figure 1 illustrates the mechanical design of the ReMMRP when 4 leg modules and 2 infrared range sensor modules are attached to the robot’s body. The robot is composed of a chassis, which serves as the its body, and up to 12 peripheral modules. The chassis contains 12 identical connection ports for the attachment of peripheral devices in a given configuration. Each connection port allows a simple mechanical assembly via 4 bolt holes and a peripheral port for transferring power and communication signals.

The leg module is composed of 3 revolute joints (Figure 2 each driven by a permanent-magnet DC motor. The hip joint houses two of the drive motors separated by 2.86 cm through a connection bracket. The thigh and calf links each has a link length of 26.5 cm. Each leg module weighs approximately 1.56 kg. In designing the leg module, weight, cost, manufacturability, and ease of assembly were among the design considerations.

III. ELECTRONICS AND CONTROL UNITS

The electronics and control units employ a distributed processing architecture which allows for the parallelization of computational tasks as well as modularity in programming. The architecture uses a 2-tiered approach; a processor on the robot body (tier-1) monitors and controls the peripheral units while each peripheral unit (tier-2) has its own processor for communications and low-level control. The design utilizes SPI (Serial Peripheral Interface) communications between tiers 1 and 2.

Leg Control Units (LCUs) handle the processing for each leg module which involves the communication between the Main Control Unit (MCU), calculations of the forward and inverse kinematics, execution of the PID loops for joint position control, and the maintenance of its health status (Figure 3). An LCU uses an ATMEGA164/324/544P family processor which has SPI capability, 32 I/O ports 8 of which are 10-bit ADC channels, and 3 timers which can be used to generate up to 6 PWM signals.

Main Processing Unit (MPU) runs the high-level control algorithms and communicates the desired joint angles to the individual leg modules. The ATMEGA644P is used as the main processor for the ReMMRP. It has the same specifications as the 164/324/544P used in LCUs, except it has 64kB of flash memory, 2kB of EEPROM, and 4kB of internal SRAM.

Between the MPU and LCUs, there exists a Main Communications Board (MCB) which serves as the communications hub. MCB contains a STATUS processor tasked for maintaining the health of the overall system and reporting any status changes to the MPU during the operation of the robot (Figure 4). MCB also houses a decoder for coordinating the data transfer between the MCU and LCU, a step-down DC-DC converter, a battery connection port and a bank of status LEDs.
IV. COMMUNICATIONS

In designing the ReMMRP, the following software design requirements are considered:

1) Communications protocol must exist for data transfer between the MCU and the peripheral processors.
2) MCU software must determine actions for all peripheral devices, mainly LCUs, and delegate commands in a timely fashion.
3) LCU software must respond to commands from MCU with higher priority than any other task inherent to LCU software.
4) STATUS processor software must operate in "real time" allowing MCU to have immediate knowledge of attached peripherals at any given time.

The physical transfer of information between processors in this design is done using SPI communications. This method is a master/slave mode where data transfer is initiated by the master device. The master and slave can simultaneously transfer information blocks of equal size. This process is referred to as Symmetric Data Buffer Exchange (SDBE). The MCU acts as the master in communications with all other processors in the robot architecture. Thus, it is responsible for initiating any and all communications.

The communications scheme used in ReMMRP can be illustrated by an example. During the execution of a balancing algorithm, the MCU simultaneously sends to an LCU the desired joint angles corresponding to its leg module and receives from the LCU the last recorded coordinates using the SDBE. The assumption is that both processors have a data buffer called "Command Coordinates" and the LCU has an additional buffer called "Current Coordinates". The data exchange will occur as follows:

1) The MCU designates its "Command Coordinates" as a send buffer, and initializes an index variable that will correspond to successive bytes of its send and receive buffers.
2) The MCU initiates an SPI communication session with the LCU by enabling the decoder and sending the bit pattern corresponding to the peripheral interface port where the LCU is connected.
3) The MCU sends the LCU a byte indicating that the LCU must designate its "Command Coordinates" as a receive buffer.
4) The MCU sends the LCU a byte indicating that the LCU must designate its "Recorded Coordinates" as a send buffer; at this point the LCU will load the first byte of its Command Coordinates into its SPI register, and initialize an index variable that will correspond to successive bytes in its send and receive buffers.
5) The MCU enters a loop where:
   a) The byte of the send buffer corresponding to the index variable is transmitted. This causes the MCU to receive the first byte from the LCU.
   b) Once the bytes have been exchanged, each processor copies the new data from its SPI register into its respective receive buffer at the index indicated by their respective index variables.
   c) Index variables on both processors are incremented after the copy.
   d) Steps (a) through (c) are repeated until the index variables are equal to the size of the data buffers.
   e) The MCU ends the SPI communication session with the LCU by disabling the decoder.

At the end of this procedure, the MCUs "Command Coordinates" buffer will contain the LCU’s last recorded coordinates, and the LCUs "Command Coordinates" buffer will contain the new set of coordinates from the MCU.

V. KINEMATIC MODEL

For developing a kinematic model for the ReMMRP, the following orthogonal right-handed coordinate frames are introduced: \((OXYZ)\) is the Cartesian world-fixed coordinate frame where origin will be fixed at a point within ReMMRP’s environment; \((OXYZ)\) is the robot-fixed coordinate frame with its origin at the center of gravity of the robot’s body; \((OXYZ)\) is the coordinate frame whose origin is located at the \(i\)-th connection port on the robot’s body \((OXYZ)\) is the base coordinate frame for the \(i\)-th leg module; \((OXYZ)\) is the coordinate system that is assigned to the \(k\)-th link of the \(i\)-th leg, \(k = 1, \dots, 3\). The coordinate frame assignments for the ReMMRP are depicted in Figures 5 and 6.

The mobility or DOF of the ReMMRP with \(N\) identical leg modules each having 3 revolute joints can be calculated using the well-known Gruebler’s equation.

\[
M = 6 + 3N - 3P
\]  \hspace{1cm} (1)

where \(M\) is the mobility or DOF of the system, \(N\) is the total number of leg modules attached to the body, and \(P\) is the number of leg modules in contact with the ground, hence \(P \leq N\) [10]. It is assumed that each leg that is on the ground loses 3 degrees of freedom due to the contact.

In order to describe the motions of the overall system, a set of generalized coordinates will be selected for each
leg module. The joint angles define uniquely a certain configuration for each leg, thus forming a set of generalized coordinates. In the sequel, \( q_i, i = 1, \ldots, N \), will denote the vector of generalized positions for the \( i \)-th leg module.

\[
q_i = [\theta_{1,i} \quad \theta_{2,i} \quad \theta_{3,i}]^T
\]  

(2)

where the subscript \( i = 1, \ldots, N \), \( 3 < N < 12 \), assuming that ReMMRP has at least 3 legs necessary for static balancing) signifies the variables and the parameters of the \( i \)-th leg module. The vector of generalized positions for the ReMMRP is denoted with

\[
q = [q_0^T \quad q_1^T \quad q_2^T \quad \ldots \quad q_N^T]^T
\]

(3)

where \( q_0^T \in \mathbb{R}^6 \) is position and orientation vector for the robot’s body and \( q_i \)'s are as described in equation 2. Similarly, a point in the task-space will be described by a set of task-space coordinates. The vector of task-space coordinates will be signified by \( p \).

\[
p = [p_1 \quad p_2 \quad \ldots \quad p_K]^T
\]

(4)

where \( p_1, p_2, \ldots, p_K \) describe the position coordinates for the task to be accomplished. For example, if the task is to keep the center of gravity of the robot at a certain distance from the \( xy \)-plane of the world-fixed coordinate system as well as maintaining the robot’s roll angle, \( p = [z_r \quad \phi_r]^T \) where \( z_r \) is the position of the robot’s center of gravity in the direction of the \( z \)-axis of the world-fixed coordinate frame, and \( \phi_r \) is the roll angle of the robot’s body about the \( x \)-axis of the robot-fixed coordinate frame.

The vector of generalized velocities and the task-space velocity vector will be denoted by \( \dot{q} \) and \( \dot{p} \), respectively. It should be noted that \( \dot{q} \in \mathbb{R}^{N+6} \) and \( \dot{p} \in \mathbb{R}^k \), in general.

For the ReMMRP, the inverse kinematics problem can be described as follows: Given the vector of task-space coordinates \( p \), at a fixed time \( t \), the inverse kinematics problem is to find the generalized positions for each leg module, \( q_i \) and the body pose, \( q_b \). Due to the kinematics redundancy of the system, the inverse kinematics problem is (in general) an under-specified problem, i.e., an infinite number of generalized positions, \( q_i \) and \( q_b \), can be calculated for the same task-space vector, \( p \).

The kinematic position relations for each leg module can be obtained by defining the homogeneous transformation matrices between the coordinate frames assigned. The foot position and orientation (pose) of a leg module can be expressed as a nonlinear function in terms of its generalized positions, \( q_i \).

\[
p_{f,i} = \kappa_i(q_i), \quad i = 1, \ldots, N
\]

(5)

where \( p_{f,i} \) is the foot pose vector and \( q_i \) is the vector of the generalized positions for the \( i \)-th leg module. \( \kappa_i(q_i) \) is a nonlinear differentiable function of \( q_i \). The velocity relations can then be obtained by differentiating equations (5) with respect to time.

\[
\dot{p}_{f,i} = J_i(q_i)\dot{q}_i, \quad i = 1, \ldots, N
\]

(6)

where \( \dot{p}_{f,i} \) denote the end-effector (end point of the distal link of leg \( i \)) velocity vector and \( \dot{q}_i \) is the generalized velocity vector of the \( i \)-th leg and \( J_i , i = 1, \ldots, N \), represent the corresponding Jacobian matrices.

In view of equation (6), one can relate the task-space velocity vector, \( \dot{p} \) to the generalized velocities \( \dot{q} \) of the ReMMRP.

\[
\dot{p} = J(q)\dot{q}
\]

(7)

When the ReMMRP moves to execute a balancing or locomotion task, the motions are constrained by the legs that are in contact with the ground plane. In other words, the velocities of the \( P \) legs whose feet are on the ground are related due to the fact that the legs and the ground link form closed kinematic chains. This leads to the kinematic constraint equations that will be expressed in terms of the ReMMRP’s generalized velocities. The velocity of the COG of the robot’s body must be expressed in terms of the generalized velocities of each leg in order to obtain the kinematic constraint equations. The derivation of the kinematic constraints is similar to the case of two manipulators with fixed bases holding a rigid object. The kinematic constraint
equations can then be expressed in the following compact form.

\[ A(q) \dot{q} = 0 \quad (8) \]

where \( A(q) \) is a \((3P + 6) \times (3N + 6)\) dimensional matrix. In view of equation (8), it seems to be possible to obtain a reduction in the number of unknowns in equation (7). In its most general form, equation (8) describes \(3P + 6\) independent kinematic constraint equations. Theoretically, one could solve equation (8) for \(3P + 6\) of the generalized velocity components and eliminate them in (7). However, this is usually not possible due to the nonlinear nature of the equations involved in (8). Instead, a linear transformation of velocities can be introduced in order to reduce the number of unknowns in (7) \([11], [12]\).

Pseudovelocities for the system can be introduced by choosing the transformation matrix \(B(q)\) such that
\[
\dot{\mu} = B(q) \dot{q} \quad (9)
\]
where \(\dot{\mu}\) is \((3(N - P))\)-dimensional pseudovelocity vector, and matrix \(B\) is chosen so that \([A^T B^T]^T\) is invertible. That is, one can write
\[
\begin{bmatrix} A \\ B \end{bmatrix}^{-1} = [\Sigma \ \Gamma] \quad (10)
\]
where \(\Sigma\) and \(\Gamma\) must satisfy
\[
A \Sigma = I_{(3P+6)}, \quad B \Sigma = 0, \quad A \Gamma = 0, \quad B \Gamma = I_{(3(N-P))}.
\]
The pseudovelocities given by equation (9), may or may not correspond to actual system velocities depending on the choice of matrix \(B\).

The generalized velocities \(\dot{\mu}\) can now be calculated in terms of the pseudovelocities \(\dot{\mu}\) using equations (8), (9) and (10).
\[
\dot{\mu} = \Gamma \dot{\mu} \quad (11)
\]
Then, equations (7) and (11) will give
\[
\dot{p} = J \Gamma \dot{\mu} \quad (12)
\]
Equation (12) relates the task-space velocities to the pseudovelocities through the Jacobian matrix \((J \Gamma)\) and therefore represents a kinematic model for the system. By introducing the linear transformation of equation (9), the kinematic constraint equations are incorporated into the system kinematics given by (7).

VI. RESULTS

Figure 7 depicts the ReMMRP when a single leg module is connected to the robot’s body. As the development of the ReMMRP continues, the robot body and a leg module has been successfully built and integrated. The Denavit-Hartenberg parameters for the leg module is presented in Table I.

In addition to the realization of the ReMMRP mechanically, the operation of all electronics (LCU, MPU and MCB) has been tested and validated. The forward and inverse kinematics for the leg module and the PID position control for joint angles have been successfully implemented. The real-time operational characteristics of the LCU software are tested by running various functions of the LCU software in a loop and toggling an output signal from the ATmega324P processor at the beginning of this loop. As a result, the LCU is capable of:

- running a single PID loop to control a DC motor position at the sampling rate of 2.5kHz,
- running three sequential PID loops for controlling the entire leg module at the sampling rate of 850Hz,
- calculating the leg inverse kinematics in 2.8 ms.
- implementing the leg module PID control at the sampling rate of 200Hz.

The full implementation of the ReMMRP is to be completed in April 2010. Authors will be able to present more comprehensive results in the final submission of the paper.

VII. CONCLUSION

The paper discussed the development of a reconfigurable and modular mobile platform called ReMMRP being developed at Worcester Polytechnic Institute. The reconfigurability is accomplished by developing modular mechanical hardware, electronics, control and software architecture. Our approach is unique in the sense that the definition of a module is rather general including legs, arms, ranging sensors, non-ranging sensors, additional processing units or even similar ReMMRP robots. The paper also outlined the formulation of a kinematic model for the robot incorporating the kinematic constraints present in the system. As the development phase continues, initial results on the successful integration of a leg module and the robot body have been presented. The modular design makes ReMMRP a capable robot to further
research in legged robot locomotion and adaptive control of reconfigurable robots.

REFERENCES


APPENDIX I

LEG MODULE KINEMATICS

A. Homogeneous Transformation Matrices

The subscript $i$ is dropped for convenience.

$T_1^0 = \begin{bmatrix} \cos \theta_1 & 0 & \sin \theta_1 & a_1 \cos \theta_1 \\ \sin \theta_1 & 0 & -\cos \theta_1 & a_1 \sin \theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$T_2 = \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 & a_2 \cos \theta_2 \\ \sin \theta_2 & \cos \theta_2 & 0 & a_2 \sin \theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$T_3 = \begin{bmatrix} \cos \theta_3 & -\sin \theta_3 & 0 & a_3 \cos \theta_3 \\ \sin \theta_3 & \cos \theta_3 & 0 & a_3 \sin \theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

B. Inverse Kinematics Solution

Given $(p_x, p_y, p_z)$ as the foot location, the joint angles are given by,

$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right)$

$\theta_3 = \pm \tan^{-1}\left(\frac{\sqrt{4a_2^2a_3^2 - (p_x / \cos \theta_1 - a_1)^2 + p_z^2 - a_3^2} - a_3^2/2}{(p_x / \cos \theta_1 - a_1)^2 + p_z^2 - a_3^2 - a_3^2/2}\right)$

$\theta_2 = -\tan^{-1}\left(\frac{a_3 \sin \theta_3}{a_2 + a_3 \cos \theta_3}\right) + \sin^{-1}\left(\frac{p_z}{\sqrt{(a_2 + a_3 \cos \theta_3)^2 + a_3 \sin^2 \theta_3}}\right)$

C. Jacobian Matrix

$J = \begin{bmatrix} -a_1 s_1 - a_2 s_1 c_2 - a_3 s_1 c_{23} & -a_2 c_1 s_2 - a_3 c_1 s_{23} & -a_3 c_1 s_{23} \\ a_1 c_1 + a_2 c_1 c_2 + a_3 c_1 c_{23} & -a_2 s_1 s_2 - a_3 s_1 s_{23} & -a_3 s_1 s_{23} \\ 0 & a_2 c_2 + a_3 c_{23} & a_3 c_{23} \\ 0 & s_1 & s_1 \\ 1 & -c_1 & -c_1 \end{bmatrix}$
Kilobot: A Robotic Module for Demonstrating Behaviors in a Large Scale \( (2^{10}) \) units Collective

Michael Rubenstein, Radhika Nagpal

Abstract—A collective of robots can together complete a task that is beyond the capabilities of any of its individual robots. One property of a robotic collective that allows it to complete such a task is the shape of the collective. This paper presents Kilobot, a simple modular robot designed to work in a collective to self-assemble and self-heal that collective’s shape.

In previous work, an algorithm is given that allows a simulated collective of robots to self-assemble and self-heal a desired shape, keeping the shape sized proportional to the number of robots in the collective. In this abstract, the current work of producing a robotic collective that can demonstrate that algorithm is presented.

I. INTRODUCTION

It is possible for a group of robots, called a collective, to complete a goal that cannot be completed by any of the individual robots. One way to accomplish this is if the collective forms a specific shape. For example, imagine a single SWARM-BOT [1] reaches a canyon-like obstacle with a goal on the other side. By itself, a single SWARM-BOT is not capable of crossing the canyon to reach the goal on the other side. However, if the SWARM-BOT joins a collective of other SWARM-BOTs, and forms a collective shaped like a bridge, the collective’s shape enables it to cross the canyon and reach the goal. In another example, a single Superbot robot [2] needs to locomode as far as it can until its battery pack empties. As a solitary Superbot, it can only travel 200 meters until the battery is empty. If this single Superbot can form a collective with five other Superbot robots in the shape of a wheel, then it can move over 1000 meters until its battery pack depletes. In this case, the shape of the Superbot collective enables the collective to travel five times as far as any single Superbot can travel.

In previous work [3,4] a distributed control method called S-DASH was presented which enables a collective of robots to form a given shape at a scale proportional to the total number of robots. If the shape of the collective is damaged, for example by removing some robots, then S-DASH will reform the shape at a new, smaller scale, proportional to the new number of robots. A demonstration of this behavior in a simulated collective is shown in Fig. 1.

Figure 1. A demonstration of S-DASH running on a collective of robots, forming the desired shape of a star at a proportional scale. (A) An initial formation at too small a scale. (B-C) S-DASH increasing the scale. (D) The shape reaches the correct scale. (E) Half the robots are removed. (F-G) S-DASH reduces the scale. (H) The shape reaches a new correct scale.

In the previous work of S-DASH, the collective behaviors are demonstrated only in simulation. These simulated robots are very simple in their capabilities. They are capable of...
moving forward, rotating, communicating with local neighbors, and measuring the distance between themselves and their local neighbors. Many robot platforms exist that have these capabilities, for example [2,5,6], and therefore in theory are capable of demonstrating S-DASH. However, these robots are not practical to operate as a collective on the order of $2^{10}$ robots.

For an example of why they are not practical in such large numbers, consider the simple task of powering on the robots in the collective. With a standard robot, for example the E-PUCK [6] turning on the robot requires a user to toggle a power switch located on the side of the individual robot. If one robot could be powered up this way on average every 2 seconds, it would still take a single user over 30 minutes to power on all 1024 robots in the collective! Some other reasons these robots are not practical for such a large scale collective include: robot cost, operability (powering, charging, programming, etc...), and physical size.

This abstract will discuss our work-in-progress at creating Kilobot, a robot module designed to easily operate in collective sizes of $2^{10}$ robots or greater. In section 2, the Kilobot hardware is reviewed, including the locomotion, communication, control and power systems. In section 3 Kilobot operations are discussed which include: charging methods, programming, and power control. In section 4, some demonstrations of Kilobot’s current capabilities will be given. The abstract will then conclude with a discussion on what work is left to operate S-DASH on $2^{10}$ Kilobots so that they demonstrate self-assembly and self healing.

II. KILOBOT HARDWARE

The Kilobot module is designed to meet the requirements of S-DASH, while at the same time easily operate in large collectives (more than $2^{10}$ robots). The requirements of S-DASH are that the robot be able to move forward, turn, communicate with neighbors, and measure distances to neighbors. Kilobot meets these requirements, while keeping the design balanced against other needs for operating a large collective, such as keeping the cost per robot under $15 (US), and ease in programming. Fig. 2 shows a prototype version of Kilobot.

A. Locomotion

Kilobots uses two vibration motors along with passive directional legs, a method inspired by [7], for simple, low cost locomotion. Fig. 3 demonstrates the principal of this vibration based locomotion. In Fig. 3a, a vibration motor (black) is mounted on a green chassis. The platform is supported above the ground by two slanted legs attached to the chassis. When the motor vibrates, it causes forces that move the platform up and down, as shown in Fig. 3b. That up and down movement is translated to forward movement from the slanted shape of the legs, as shown in Fig. 3c.
There are three reasons for choosing this form of locomotion over traditional wheeled locomotion. The first reason is cost. Due to the complexity of the wheeled drive train, they are at least 10 times the cost of the vibration system used for Kilobot. The second reason is that the structure for a wheeled system is generally a few times larger than the vibration system used. Thirdly, the vibration system allows for minimal structure between the bottom of the chassis and the floor below. The only structures in that space are the two slanted legs on the sides, and a third straight leg in the front used to complete the tripod of legs. The reason minimal structure in this space is important will be explained in the next section.

B. Communication

The communication system on each robot serves two purposes: to send digital information to neighboring robots, and to measure the distance between itself and a neighboring robot. This is accomplished using a wide field-of-view infra-red emitter and transmitter that are located at the center of the chassis underside. The emitter and transmitter are pointed down at the floor below. As shown in Fig. 5, the infra-red light reflects off the floor below, and then is received by a neighboring robot. This communication channel can be used to send digital information and estimate the distance of that communication using the received signal strength. This communication path is the reason that it is important to minimize the structure between the chassis and the floor, because a large structure would block some or all of the communication path.

\[ \text{Figure 5. A path for infra-red communication between two robots.} \]

C. Controller

Each Kilobot has a microcontroller onboard to handle the entire control of the robot, including communication, motor control and decision making. The microcontroller used is a 32 pin Atmega 168, which can be purchased in bulk for under $3 (US). This microcontroller has 16 KB of memory.

D. Power System

Each Kilobot is powered by a 160 milliamp hour, 3.4 volt lithium ion battery. The Atmega microcontroller is directly connected to the battery, while the rest of the circuitry is connected to a 3V switchable regulator. This regulator can be turned on and off by the microcontroller. Under normal usage the battery should provide approximately 4 hours of operational power for the robot. There is also a battery charging circuit on each robot, which will charge the battery if 6 volts is applied to the two slanted legs of the robot.

III. Operations

There are some operations for a standard robot that are simple, and hardly considered when designing that robot, for example: how to turn on the robot, how to charge its battery, and how to program it. These simple operations must be carefully considered when dealing with a collective of thousands of robots, to prevent them from becoming expensive, tedious, or time consuming.

A. Power Control

As demonstrated before, even a simple power switch for each robot is too time consuming for a large collective. Kilobots are designed so that the entire collective can be turned on or off in under one minute. Instead of completely disconnecting the battery from the robot when it is not in use, the robot just enters a low power state when commanded to by the user. When the user wants to activate the robots, they are sent a wake up message, and the robots will exit the low power state, entering the operational mode. This is done by having the microcontroller turn off the switchable power regulator, and then entering a deep sleep mode for 1 minute when the robot is not in use. While Kilobot is in this deep sleep mode, the onboard circuitry is only drawing 50μA of current from the battery. After about a minute of this deep sleep, the microcontroller will wake up, turn on the switchable power regulator, and for 10ms, check to see if it detects a wake up message over the infra red communication channel. If this message is detected, then the robot will enter the operational mode. If the message is not detected, the Kilobot will re-enter the deep sleep mode for another minute, repeating the cycle. The experimenter uses an infra-red remote control to send messages to all the robots in the collective to either enter the deep sleep cycle, or to wake up from the deep sleep cycle.

B. Charging

As stated earlier, each robot has a built in lithium ion battery charger, that will charge the on board battery when 6 volts is applied to the two slanted legs. The charger will automatically cease charging when the battery becomes full. This allows bulk charging of the Kilobots by placing each robot on a set of conducting strips attached to a 6v power supply, as visualized in Fig 6. This charging method is inspired from [8].

\[ \text{Figure 6. A visualization of the Kilobot charging scheme.} \]
C. Programming

The standard way to program a robot is by connecting it to a computer and then programming it, which is too time consuming for a large collective. Two possible methods for programming a Kilobot speed up the programming process by taking advantage of the Atmega’s self-programmable memory. The first way would be to program one Kilobot the traditional way, using a cable connected to a computer. Next, place this newly programmed Kilobot into a group of robots you wish to program. This Kilobot will then broadcast over the infra red communication channel its new program to all of its surrounding neighbors, who will overwrite their old program with the newer program. These neighbors will then propagate the new program to all of their neighbors with the older program. This process will continue until all the robots in the group have the new program. A second possible method to program a group of Kilobots is to use a powerful infra-red light that can broadcast the new program to all the robots at once. When the robots receive the program from the infra-red light, they again use the self-programmable memory feature of the Atmega and replace their old program with the new one.

IV. Current Capabilities

The development of the Kilobot system is currently still in progress, so not all of the goals of the Kilobot system have been achieved yet. This section will describe what capabilities are needed for the Kilobot to achieve the collective self-assembling and self-healing behaviors of S-DASH. Furthermore, it will discuss which ones have already been achieved, and which ones are still incomplete.

A. Movement

The robots must have the capability to move to any location in a 2D plane. This capability has already been achieved using vibration based locomotion. Using this technique, a Kilobot is capable of turning in a circle of less than 3 robot radii, and moving at speeds greater than 2 robot radii per second.

B. Power Control

The ability to put the robots into a deep sleep, and then wake them up to the operational mode is still incomplete. The circuitry for the deep sleep has been built and tested, however the full functionality has not yet been programmed into the Kilobots.

C. Programming

The ability to program the robots as a group is incomplete. Again, the circuitry is built and tested, but this capability has not been fully programmed yet.

D. Charging

The large scale charging system for the Kilobot has been fully tested, and is operating as described.

E. Communication

The communication system on the Kilobot has been built and fully tested. Each robot is capable of sending a four byte data packet to its neighbors up to 750 times a second. A Kilobot can communicate to any neighbor that is closer than 8 robot radii away.

F. Distance Sensing

Every time a communication packet is received by a Kilobot it also has the ability to measure the distance between itself and the transmitting robot. This distance measurement has the accuracy of ±2mm.

G. Triangulation

Kilobot has demonstrated the capability to triangulate its location in the environment. The triangulation works as follows: given three neighboring robots in a triangle formation, where each robot knows its location in the environment, a forth robot can then determine its position in the environment. This fourth robot does so after receiving a data packet from each neighbor that gives that neighbor’s location in the environment. That communication also allows the fourth robot to measure the distance from itself to each of its neighbors. From this information, the fourth robot can determine its location.

H. Trilateration

Using trilateration, a method described in [9,10], each robot should be able to assign itself a unique location in a coordinate system. This coordinate system is agreed upon by all the other Kilobots in the collective. The unique location for each robot can be developed solely from communication between neighboring robots, and the measuring of that distance. While the hardware is ready for trilateration, the actual software is still in production, and has not been fully tested.

I. Demonstrations

There are two demonstrations that can be presented at the workshop either by video, or bringing the robots. In the first demonstration, one Kilobot is stationary, and another orbits around the first at a fixed distance. This demonstrates two capabilities: first, that distance between Kilobots can be reliably sensed, and second, that the method used for locomotion is reliable and accurate when feedback from the environment is available.

A second demonstration shows triangulation in action. In this demonstration, three Kilobots are placed in a triangle. A fourth robot is moved about in the environment, and using a RGB LED, it indicates its location in the coordinate system. This demonstration shows both communication and distance sensing.

CONCLUSION

This abstract presents the Kilobot robot, designed specifically for operation in a large collective. While the Kilobot is a simple low cost robot, it is expected to be able to self-assemble and self-heal its collective shape. Currently there are just a few Kilobots produced for early prototyping and testing, however once the design is finalized, we plan to produce 1024 robots.
REFERENCES


Modular Fault Diagnosing for Robotic Arms

Hamid Abdi, IEEE Member
Centre for intelligent systems research
Deakin University, Australia
Hamid.Abdi@deakin.edu.au

Saeid Nahavandi, SMIEEE
Centre for intelligent systems research
Deakin University, Australia
Saeid.Nahavandi@deakin.edu.au


INDEX TERMS: Fault Identification for robotic arm, Fault fusion and fault inferring methods, distributed fault diagnosing

I. INTRODUCTION

Fault diagnosing and isolation is a useful feature for electromechanical system. It is very suitable if the system is complex and the user is not a professional. Therefore if a robotic arm is applied for non professional users then it is better to have fault diagnosing. Even in professional applications when a robotic arm is used in a hazardous environments and the minimum down time is desired then fault tolerant is essential. These arms tolerate partial faults into the joint or control systems such as those introduced in [1][2][3]. Fault detection of the arms is critical for their fault tolerance and fail-safe functionality [3]. The diagnosing adds into the availability of the system when the system is modular and it has some kinds of redundancies. Then the fault diagnosing contributes to maintain the availability of the systems [4]. Fault diagnosing has been applied for several applications such as electronics chips, control systems of hazardous process and robotic applications. Generally self diagnosing of systems, is equipping the system with a supervisory facilities to look into the measured or predicted states of an electromechanical system then infer the fault. This supervisory system is assessing the system states to ensure safety for each component or the healthiness of the operation of the whole system.

The literature of the fault tolerant of robotic arms has focused on design level (Design of fault tolerant arm or fault tolerant structures) or control level (Fault analysis and fault tolerant motion planning or control). In design different structure have been designed to be fault tolerant like [5][6][7]. Work within the control level has focused on Fault Detection, Fault Isolation and Identification and Fault Recovery techniques. Different strategies such as model base analysis or AI solution have been proposed such as those in [8][9][10]. Fault diagnosing for robotic arms brings all the aforementioned advantages within the arm [10]. This promotes the arm reliability, which is required for the tasks in surgery, handling of explosive and nuclear materials and tasks in hazardous environment [10].

Four types of faults are considered for robotic arms including joint actuator failures, sensor failures, control system failures and communication or interfacing failures [10]. Essentially these failures cause partial to catastrophic fault into the motion or the control of the arm. Self diagnosing of these faults provides the ability to use an improved or adapted strategy for control of the arm. It also provides the ability for faster troubleshooting of the arm. Using the information from the self diagnosing in the control system such as adaptive or robust control maintains the availability of the arm under some fault circumstances. Therefore the arm continues its tasks under these faults. In [9] and [12], kinematic faults of the robot joint are investigated. In [13], a Neuro Fuzzy mechanism has been used for fault detection and in [14] a self diagnosing power recharge system is designed for mobile robots. In [15], the same system is introduced for whole operation of mobile robot. But a distributed fault diagnosing structure in [16] has been used for a flight system. Recently in industrial robotics community MOTOMAN and other manufactures have released their robots products with central self diagnosing features. This paper is to design of a decentralized self diagnosing for robotic arms.

In reminder of the paper presents a distributed architecture and conceptual hardware of the self diagnosing modules for robotic arms. Then it proposes the fault inferring methodology by utilizing fault fusion framework. Then it introduces the fault diagnosing network. Then fault diagnosing function block library is incorporated to make the solution expandable. Finally a designed self diagnosing system for an arm is validated through simulation study and the work is concluded.

II. MODULAR SELF DIAGNOSING

A. Modular Hardware on Open LAN

Centralized and decentralized architecture can be used to design a self diagnosing of the robotic arms [14][15][16]. In this study, motivated from similar functionality of the joints of the arm and the advantages of a distributed systems a self diagnosing solution is proposed. The main objective is to increase the availability of the
For a 6-DOF robotic arm, the design of self fault diagnosing is proposed to be composed by 8 distributed self diagnosing modules which include 6 modules for the joints, one for the main controller and the last one for electric/power circuits. All these modules and the main controller can communicate with each other through a communication channel which will introduce later.

It is possible to have other dedicated modules which can be used as self diagnosing for peripherals such as the digital cameras or the grippers. And the open model of the networks provides the required expandability for these peripheral’s diagnosing modules. The main focus of this study is design and functionality of the modules for fault diagnosing for the joints of the arm. Fig. 1 indicates the proposed distributed self fault diagnosing system for a 6-DOF robot arm.

As the joint of the arm are working very similar then types of the faults within each joint are similar [10]. Then the distributed architecture proposes a self diagnosing module for each joint. Each module is mounted on the corresponding joint of the arm. The function of the module is to identify the fault the joints. These modules receive the general or their dedicated sensors and switched information on the joint which is incorporated with the joint. The module is designed to have an inbuilt redundant microcontroller to provide the required reliability for the module. A range of sensors are connected to provide real-time information from the joint and related electric or mechanical components. The sensors include encoders which provide the position, speed and acceleration information, strain gauge, several micro switches, temperature, current sensors.

**B. Expandable Software**

From software perspective the functionality of the operation of very similar the distributed self diagnosing is a right decision but a centralized fault diagnosing is required as well. Because more over than the health of the joint is essential the health and safety of whole arm need to be assessed. In joint level an expandable library of self diagnosing function block with configurable parameters are considered for the modules. Two general types of signals are considered for self diagnosing information including digital and analogue input (DI and AI) signals. Currently for a standard block following parameters are considered. These are called the parameters of the self diagnosing DI function block (FB) and the self diagnosing AI function block as following.

For a DI channel following configuration parameter exists

1.1. Channel address
1.2. Channel ID
1.3. Channel unit
1.4. Active low/high
1.5. Min pulse width time
1.6. Fault comment

For AI channels

2.1. Channel address
2.2. Channel ID
2.3. Channel unit
2.4. High and High-High threshold
2.5. Low and Low-Low threshold
2.6. Increasing rate limit
2.7. Decreasing rate limit
2.8. Frequency spectrum analysis
2.9. Fault comment

For each analogue signal four level of Low-Low, Low, High and High-High are determining the levels which critical for the safe operation of the signal. For example for the current of a motor of a joint this values can be Low-Low=0A, Low=4A, High=5A and High-High=7A. Which means less than 5 amperes the operation of the motor from current is safe but while it more than 5 then it is higher than what it is expected and more than 7A is like the command to stop to protect the motor. For each of these signals a standard library of self diagnosing functions is provided such as those simple described for Low-Low, Low, High and High-High. Specifically, the frequency domain sub library consist a number of spectral analysis functions such as harmonic analysis, resonance frequency measure, vibration analysis and bandwidth analysis which are used for more advanced fault diagnosing. The open structure of the library of the function block provides a medium in which new functionality can be added in easily.

More over than the specific fault diagnosing function blocks a sub library of common math or logical function blocks is provided. Therefore a more advanced function blocks can be programmed by using other function blocks.

The function blocks are uploaded into the modules as they are required. The module operates with a standalone operating system which runs the configured function blocks in a sequential format similar to programmable logic controllers.

**III. Fault Inferring**

Different faults at three levels of part failures, joint failures and overall arm failures are required to be identified by the fault diagnosing system of the arm. Therefore following conceptual structure is designed.

**C. Fault Inferring Hierarchy**

Having a high performance self diagnosing requires appropriate use of all information is provided by the sensors, rules or even modules. A hierarchical approach is used for fault analysis within
the fault diagnosing system. The conceptual fault inferring hierarchy for an arm is indicated in Fig.2. The inferring is classified into the part level, joints level and the overall arm level. In the part level only the functionality of the mechanical or electrical parts of the joints is diagnosed using the basic diagnosing function blocks on the received sensor information. At the joint level the part level results are used by an expert or rule base inferring to evaluate the fault within the joints. Then at the arm level the inferred faults at two prior levels are used in an expert and rule base inferring process to determine the overall fault of the arm. The first two levels are implemented in modules while the third level is processed by the main controller of the arm.

**D. Fault Detection Method**

Failures detection from the sensory information can be classified into 4 types including

1. Faults can be certainly identified by single sensor or switch
2. Faults can be certainly identified by multi sensor or switch
3. Faults can be uncertainly identified by single sensor or switch
4. Faults can be uncertainly identified by multi sensor or switch

For the first case only a simple logic/mathematic operation with function block introduced previously identifies the fault. However in some circumstances the transient behavior of a sensor or spectrum of is required to be analyzed or user function block can be deployed. The function block structure of the diagnosing programs and programmability of the modules make it suitable for any expansion and adding of specific diagnosing function blocks.

For the third case as the information is uncertain the fuzzy or estimation base approach is required which is the ongoing further research as those introduced in [13, 15]. And finally for multisensory fault diagnosing a data fusion block is required as following.

**E. Data fusion for overlapped Inferred Faults**

Because of the overlap between the information provided with the sensors or switches, data fusion can be used to increase the reliability of the inferred faults or even make the inferred faults with more details [7],[8]. Inside the data fusion, least square data fusing, data blending and expert base inferring can be used for fault diagnosing.

The data fusion block is very applicable especially at feature level. As the information at this level is multimodal or multi dimensional. Fig.3 shows the block diagram of the inferred fault through data fusion mechanism. Defining a standard data fusion block requires more research but weighted data blending can be achieved through regression and least square which is well known in data fusion systems.

**IV. HARDWARE AND INTERFACING**

**F. Redundant Hardware Design for Self Diagnosing Modules**

The modules should have a redundant power supply, redundant Controller Unit (CU), redundant sensor interfacing, redundant memory and finally redundant network connection. Fig 4 in the last page of the paper indicates the block diagram and internal redundancy of the module. It indicated the data a control flows within the proposed hardware. This system is designed by using dual Atmel AVR 128 micro controller for a master and redundant controller, dual power supply, and dual bus driver. Also this has to be noted that the internal features of the Atmel microcontrollers results into dual RAM and ROM, dual oscillator dual ports and others. The selected microcontroller provides 8 analogue input (AI) channels for analogue sensors, 12 digital inputs (DI) channels for the switches and 4 pulse counters. Table 1 indicates the internal facilities which are used. In [11] the design provides a self healing hardware is studied.

**G. Swapping between redundant controller units**

When a failure occurs within the hardware of the modules a suitable swapping mechanism between the master and redundant microcontroller is required with following routine. The swapping can be through Event base watchdog or Time base watchdog.

In both cases an internal fault is detected through watchdog routine a copy of internal data memory of the master controller is saved in the internal data memory of the redundant CPU, the communication is done through asynchronous user defined protocol AUDP by two master redundant Controller Units. Four status pins of the master controller are connected to the redundant one to inform the internal hardware fault of the master microcontroller (2 hardware status and 2 for software status). Therefore in the case of hardware and software failure in the master controller, the redundant sends a command to stop the ICRA 2010 Workshop "Modular Robots: State of the Art"
master and it continues with the last saved data while their program and configurations are the same.

H. Communication system

Two communication mechanisms are considered for the self diagnosing module. The first one is the master CU and redundant CU communication network which is the peer to peer and is called internal communication. And the next one is a standard communication between modules which is called external network. For the internal communication, we used eight bit parallel transferring with four hand shaking bits (Port C and Half of Port D of microcontroller) as RTS (request to send), CTS (clear to send) and Parity and Error acknowledging information. The other necessary hand shaking is provided with software protocol.

For the external communication an open standard network is required. In each module both the master and the redundant CPU have their own bus controller and the bus is handled by the master bus driver normally when it is a functional. By using two CAN bus [19] drivers along with the two AVR controllers; it is assured that the module accesses to the bus when a fault occurs to the master controller. The open structure of the CAN network which is used as fault diagnosing network provides the expandability of the network. Also as the bandwidth of the CAN bus is enough to handle the fault data traffic between the modules and the main controller of the arm.

I. Design Validation

Based on the design proposed in the paper a simulation study has been implemented to develop the fault diagnosing and fault inferring for a 6-DOF arm. The PUMA560 model in MATLAB robotics toolbox [18] has been selected. The required controls have been added into the model of the robot to impose different fault scenarios for the joints. This included locked joints, free swing, and non proper torques, current of the motor of the joint, temperature of the motor, vibration.

The fault diagnosing modules include the function blocks of digital and analogue diagnosing was developed using MATLAB Simulink. The block diagram of the self diagnosing composed by the modules and fault diagnosing bus is indicated in Fig 5 in next page. The middle block is the model of a PUMA560. The fault generators are in the left side and the fault interfering blocks are in the right side.

This developed model provided a framework to incorporate the specific fault diagnosing function block for example those functions based on spectrum analysis. Through this simulation different fault scenarios have been studied. The result indicated that the proposed self diagnosing is functional.

V. Conclusion

The paper presented a modular fault diagnosing system for robotic arm joints. The design incorporated a dedicated module for self fault diagnosing for each joint of the arm. The specifications and the design details of the modules have been addressed. The self fault diagnosing methodology and fault inferring hierarchy have been discussed. The internal hardware redundancy and network redundancy to provide the necessary reliability of the self diagnosing modules were considered. Then an expandable library of fault diagnosing functions was introduced for the modules and a preliminary list of functions in the library was defined using Simulink library. The design of the fault diagnosing modules was validated through simulation study for an arm, indicated the self diagnosing modules and the network functionality. The simulation framework provided and environment to develop different fault diagnosing function blocks prior to implementation on the modules. As future work the authors are interested to have a standard function library for fault diagnosing of joints in any robotics arm.

REFERENCES


ICRA 2010 Workshop "Modular Robots: State of the Art"
<table>
<thead>
<tr>
<th>Facility</th>
<th>Function</th>
<th>Redundancy</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI Channel</td>
<td>Analogue Sensors</td>
<td>√</td>
<td>Temp, Strain, Force, Acceleration</td>
</tr>
<tr>
<td>DI Channel</td>
<td>Digital Sensors</td>
<td>√</td>
<td>Proximity and limit, Vibration switches</td>
</tr>
<tr>
<td>AUDP* communication</td>
<td>Redundant System</td>
<td>√</td>
<td>Master- Redundant CPU communication</td>
</tr>
<tr>
<td>Program Memory</td>
<td>System Program</td>
<td>√</td>
<td>Saving System Program</td>
</tr>
<tr>
<td>RAM</td>
<td>Data</td>
<td>√</td>
<td>Saving and Data Backup</td>
</tr>
<tr>
<td>Interrupt</td>
<td>Hardware control</td>
<td>√</td>
<td>For AUDP communication and Master fault detection</td>
</tr>
<tr>
<td>Timer/Counter</td>
<td>Encoder Pulse Inputs</td>
<td>√</td>
<td>For system calculation, /encoder</td>
</tr>
<tr>
<td>Frequency Oscillator</td>
<td>System Clock</td>
<td>√</td>
<td>Hardware Clock</td>
</tr>
</tbody>
</table>

Fig. 4 Block diagram of the software and hardware redundancies for the fault diagnosing modules

Fig. 5 The Simulink model for fault analysis of a PUMA560

ICRA 2010 Workshop "Modular Robots: State of the Art"
A vacuum-based bonding mechanism for modular robotics

Ricardo F. M. Garcia, Jonathan D. Hiller, Hod Lipson

Abstract—We present our progress on the design and implementation of Vacuubes, a set of robotic modules that exploit vacuum as adhesive force to form and hold structures. We use a first prototype to perform basic experiments that demonstrate the vacuum sealing capabilities of the modules, as well as the proper actuation of a valve designed to propagate vacuum between two of these modules. Based on these results, we expect that vacuum is a viable connection principle for any modular technology where easy attachment and detachment is required.

I. INTRODUCTION

A modular robot is a robot built from many similar modules that attach to each other by using connection mechanisms. Although single modules have limited uses, they are able to combine into more functional structures according to the task being performed [1].

The advantages of modular robots over robots made from a few special-purpose parts are three. First, their ability of assembling task-suitable structures makes modular robots more flexible. Second, their redundancy of modules makes them more robust. Third, their similarity between modules makes production of modular robots potentially cheaper [2].

As reconfiguration plays an important role in these robots, modules must be easy to attach and detach. Additionally, connection between modules must be strong enough to hold large configurations and to support additional loads of, for example, manipulating tasks. Hence, a very important part of design of modules involves the connection mechanism [3], [4].

One connection mechanism of modular robots is active hooks. In this approach, modules are loaded with motor-driven protrusions that hook either holes or complementary protrusions in neighbouring modules to achieve attachment [5], [6], [7], [8]. Although strong, these mechanisms require many motors to drive many connectors (one per active face) with the consequent increase in complexity and price of modules.

Another connection mechanism is passive hooks. Here, modules are loaded with passive connectors that are manually plugged into complementary sockets of neighbouring modules to achieve attachment [9], [10], [11]. Although this approach is also strong, autonomous attachment, if desired, would demand dexterous robots and precise alignment of parts.

The next connection mechanism is magnetic attachment. In this approach, faces of modules loaded with magnets, such as permanent magnets [12], electro-magnets [13] or Magswitches1 [14], are brought into close proximity to achieve attachment. Although this is perhaps the easiest way to put modules together, connection strength may not be suitable for large constructions. Additionally, permanent magnets can be (one-by-one) difficult to detach, electro-magnets consume too much power, and Magswitches also demand a motor per connector to turn them on or off.

A final connection mechanism is Velcro. In this case, faces of modules covered with Velcro stripes are brought into close proximity to modules with complementary stripes to achieve attachment [15]. Unfortunately, the strength of this approach is also not suitable for large constructions.

We explore vacuum as adhesive force to form and hold structures made of modules. Our idea is to keep vacuum inside a cubic module loaded with normally-closed pneumatic valves in all faces, and then propagate vacuum to another module when their faces are brought into contact, as shown in Fig. 1. Besides simplicity and easy attachment, we believe that this approach allows for easy detachment of modules by releasing vacuum to atmospheric pressure, as shown in Fig. 2 for a structure made of many modules. We call our modules Vacuubes.

In the coming sections we show initial design and implementation of our modules as well as basic experiments that demonstrate fulfillment of important requirements, such as

Fig. 1. Vacuum holding two modules together. Vacuum is at first generated inside the body of the cube on the left and then propagated to the cube on the right by using a pneumatic valve that opens only when their faces are brought into contact.

1 A Magswitch is a magnet that can be mechanically turned on or off. http://www.magswitch.com.au/
vacuum sealing and adhesion capabilities of the modules, and the proper actuation of a valve designed to propagate vacuum between two of these modules. Based on these results, we expect that vacuum is a viable connection principle for any modular technology where easy attachment and detachment is required.

II. REQUIREMENTS

Considering easy attachment, easy detachment, strong connection, and that attachment forces are described by the equation \( F = P \times A \), the modules should:

- reach good vacuum levels, and
- maximize contact-area to weight ratio.

Considering also our lab resources, the modules should:

- be 3D-printed during prototyping, and
- have a maximum dimension of 30x30x30mm.

The dimensional requirement relates to the printing speed of our 3D-printer. 30x30x30mm is a good balance between size and printing time.

III. DESIGN

A. Modularity

As we wanted to isolate sources of problems, we decided to design our modules in incremental steps. We wanted to test, at first, the permeability of the 3D-printing material, then, different approaches to pneumatic valves, and, finally, different alignment mechanisms for the faces of the modules. To this end, we followed a modular approach in which the cubes have exchangeable faces that can be loaded with different valves, alignment mechanisms, or whatever is needed to perform tests (e.g., a nozzle to connect a vacuum pump). Fig. 3 shows the CAD models of the cubes’ skeleton and of a “nozzle” face.

B. Pneumatic valves

Besides sealing vacuum to a high degree, the normally-closed valves had to open in response to contact with an approaching face of a neighbour module. By doing so, the valves could establish a vacuum channel between neighbour modules and, therefore, propagate vacuum. That said, we needed a push-actuated mechanism able to drive the valves from close to open state.

Our first idea to solve this problem was to use commercial
push-button valves, but, unfortunately, the smallest ones we found\(^2\) were too big to fit in our modules. Afterwards, we figured out two 3D-printable mechanisms, which are shown in Fig. 4. The first mechanism (Figs. 4a and 4b) consists of a spring that pushes a rubber ball up in the vertical axis to seal a vacuum channel in the horizontal axis, and the second mechanism (Figs. 4c and 4d) consists of a pivoted cover that seals a vacuum channel along the vertical axis with the assistance of an o-ring and a spring. Notice that, in both cases, forces exerted by vacuum cooperate with the sealing strategy, and also that springs enable operation of these valves in all orientations.

**C. Simulation**

At this point, we wanted to know the influence of the internal volume of the modules and the resistance to flow of the vacuum channels on the strength of the structure when adding a new module. In other words, we wanted to know which combination volume-resistance affects the vacuum levels in the modules the least. Big volume and small resistance? Big volume and big resistance? As we did not have the answer, we decided to simulate the pneumatic system determined by our modules.

We first considered internal volume as a pneumatic accumulator and channel's resistance as a constriction in a pipe, and then determined the equivalent electronic circuit, as shown in Fig. 5. We then simulated the behaviour of the sequential connection of four modules into a chain by using the LTspice\(^3\) electronic simulator and the equivalent circuit shown in Fig. 6. In our setup, we gave arbitrary initial values to capacitors (volumes) and resistances (constrictions), and, then, we proceeded to vary these values to perform comparisons. Thus, our analysis is only qualitative.

Our simulation results, shown in Fig. 7, provided interesting insights. For example, we observed that, when adding a new module to the structure, the entry module receiving this new module suffers the highest decrement in the vacuum level and recovery time. We also observed that the larger the chain the worse is the effect on the entry module (which is always the last module in the chain). Most interestingly, we observed that increasing volume and keeping resistance small or keeping volume small and increasing resistance produces exactly the same output: the magnitudes of the vacuum drops in the cubes are the same as in the initial simulation, but the times required to recover from these drops are, in these cases, larger and proportional to the product of both parameters (i.e., \(\tau = RC\) constant). Keeping in mind that vacuum level is dependant on the amount of air molecules enclosed in a volume (the less the better), we interpret last observation as follows:

\(^2\)http://www.valve-push-button.com/
\(^3\)http://www.linear.com/
• big volumes resist more air molecules in the system before collapsing but also bring more of these molecules to the system when adding a new module, and
• big resistances slow down the rate at which air molecules enter the system but also the rate at which they leave.

Thus, according to our interpretations of the simulation results, small volume and small resistance (i.e., big vacuum channels) would improve recovery times but would do little in terms of vacuum drops. This is problematic because structures would still collapse with big vacuum drops for short periods of time, and these times would take longer as the structure grow. Despite that, we now believe that a possible solution to the problem of vacuum drops would be: big resistance to the entry of air molecules and small resistance to the exit of them.

IV. IMPLEMENTATION

A. 3D-printing material permeability

We began implementation by 3D-printing the skeleton of our modules, a nozzle face to connect a vacuum pump, and simple faces without any valve or hole just to test the sealing properties of our material (sealing faces). Fig. 8a shows the implementation of the skeleton and the sealing faces. We 3D-printed the parts in an Object Eden 260V 3D-printer loaded with a translucent acrylic-based photopolymer material called Fullcure 720\(^4\), and, with a thickness of 1.5mm, the material allowed us to reach and keep vacuum levels of 27.5 in.-Hg. We interpret this results as positive in terms of permeability.

B. Pneumatic valves

Unfortunately, the 3D-printed implementation of the ball valve design (Figs. 4a and 4b) was not able to keep vacuum. Even with a protrusion at the sealing height to pull the ball towards the hole in the horizontal axis, the sealing achieved was not acceptable. We believe that the reason of this problem is the high precision demanded on the spring extension to position the valve at the correct sealing height.

Despite that, the previous mechanism allowed us to test an interesting approach. We simply pre-stressed the spring pushing the ball up, so that the sealing point would now be at the top hole (not at the horizontal axis), and we succeeded at sealing. This approach is similar to “Schrader” valves of cars and bikes, with the difference that, now, the spring has to overcome the vacuum force pushing the ball down. Fig. 8b shows a diagram of this modified ball mechanism, and Fig. 8c shows the implementation of the same valve. Notice that we replaced the ball with a button with o-rings in order to adjust the point of actuation by varying the length of the button’s knob, and also to play with the resistance of the vacuum channel by varying the cross-section of the same knob. Our idea to apply the insights from simulation (i.e., big/small resistance to entry/exit of air, respectively) is to vary the knob’s cross-section, going from wide to narrow and then from narrow to wide again.

As the modified ball valve mechanism worked, we did not test the pivoted valve mechanism (Figs. 4c and 4d).

C. Alignment mechanisms

Up to this point, we have not designed nor tested any alignment mechanism.

V. EXPERIMENTS

A. Vacuum holding-releasing

Fig. 9 shows an experiment to demonstrate the vacuum holding capabilities of Vacuubes as well as the proper actuation of the valve. In this experiment, we assemble a module with two nozzle faces (one to connect a manual vacuum pump and another to connect a pressure gauge), a valve, and three sealing faces. Then, we build up vacuum inside the body of the module, and, after a couple of seconds, we release vacuum by pushing the button of the valve.

B. Attachment-detachment

Fig. 10 shows an experiment to demonstrate the idea of modules sticking together by using vacuum. In this experiment, we basically use the same setup as the previous experiment, but we also assemble a second module with five sealing faces and a valve. We perform the experiment as follow: first, we place the two valves of the modules face-to-face with an o-ring in between, then, we build up vacuum

\(^4\)http://www.objet.com/Materials/FullCure720_Transparent/
in the body of the modules, after that, we manipulate the structure for a while, and, finally, we release the vacuum to atmospheric pressure with the consequent detachment of the structure.

VI. CONCLUSIONS AND FUTURE WORK

We presented our progress on the design and implementation of modules able to attach to each other by using vacuum as adhesive force. In addition to that, we performed simple experiments that demonstrated fulfillment of important requirements, such as vacuum sealing and adhesion capabilities of the modules, and the proper actuation of a valve designed to propagate vacuum between two of these modules. Based on these results, we expect that vacuum is a viable connection principle for any modular technology where easy attachment and detachment is required.

In the near future, we will address not only alignment mechanisms and assembly of larger structures but also the construction of a robot able to manipulate our modules. The idea is then to explore reconfiguration of passive structures; a challenge known as machine metabolism.

ACKNOWLEDGMENT

This work was supported in part by the U.S. National Science Foundation’s (NSF) Office of Emerging Frontiers in Research and Innovation (EFRI), grant #0735953, and in part by the Chilean Ministry of Education through the project Mecesup UTA0304.

REFERENCES


Towards the sense of touch in snake modular robots for search and rescue operations

Juan Gonzalez-Gomez¹, Javier Gonzalez-Quijano¹, Houxiang Zhang², Mohamed Abderrahim¹
¹Robotics Lab. Universidad Carlos III de Madrid, Spain
²TAMS, Department of Informatics, University of Hamburg, Germany
{juan.gonzalez.gomez, javier.gonzalez-quijano, Mohamed.Abderrahim}@uc3m.es, hzhang@informatik.uni-hamburg.de

Abstract—Snake modular robots are good candidates for being used in Urban Search And Rescue (USAR) operations because of their flexibility, good adaptation to the terrain and small section. We propose to design and build snake robots that combine three capabilities: locomotion, climbing and grasping. The last one allows the robot to remove objects for clearing the path to the trapped people. The sense of touch is key to achieving these three capabilities. To implement it, our novel approach is based on the idea of touch rings and touch strips. In this paper some preliminary ideas are presented.

I. INTRODUCTION

Unfortunately disasters caused by earthquakes, floods, tsunamis or hurricanes are very often in the news. In these scenarios the priority is to find survivals quickly, otherwise they will die. In urban disasters the environment is extremely unstructured, chaotic and full of pieces of rubble everywhere (fig. 1). This makes very difficult and dangerous for the USAR (Urban Search And Rescue) teams to move and locate people. Besides, it is clear that if a building has collapsed, there will be likely people inside that should be rescued as soon as possible.

During the last decade, researchers around the world have been facing the problem of designing tele-operated robots capable of moving in such environments and to assist the USAR teams. Properties such as versatility, adaptability and flexibility are taken into account when designing these robots.

Yim et al.[1] were the first to propose modular robots for this purpose, due to their three promises of versatility, robustness and low cost. Miller[2] suggested snakes robots (consisting of yaw modules with passive wheels) as good candidates for moving in unstructured terrains, in a similar manner than real snakes do. Wolf et al.[3] from the Department of Mechanics at CMU developed a new concept consisting of an elephant trunk-like robot mounted on a mobile base. A camera located at the end of the trunk was used for inspecting unreachable areas such as small cracks and pipes.

The Japanese government is specially interested in USAR applications. They are funding and promoting research for the development of practical search-and-rescue robots. Prof. Hirose and their colleagues at the Tokyo Institute of Technology have been working on these robots for many years [4]. They have developed robots with tracks and wheels, such as Souryu, Genbu, Kohga, Gunryu and the latest Helios VIII[5]. They have also proposed the snake robots as candidates for USAR applications because of their advanced motion capabilities: their bodies can act as “legs”, “arms” or “fingers” depending on the situation. Since mid 70’s they have been working on the ACM snake-like robot family[6]. The ACM-R3 and R4 have passive wheels larger than the profile of the link that fully cover the whole body. This enables the snake to slide smoothly among the rubble.

In the Institute of Robotics at Beihang University, Zhang et al. have developed the JL-I robot[7]. It is composed of three identical autonomous modules with 3 DOF joints. It can climb obstacles, stairs and recover itself in case of turning over.

One of the latest USAR robots is Amoeba-I[8] developed at the Shenyang Institute of Automation in China. It is composed of three tracked modules and has nine locomotion configurations. It can change its configuration automatically to adapt to the environment.

Modular robots with a 1D topology can be divided into two groups: serpentine and snake robots. The former use active tracks or wheels for self-propulsion while the latter use body motions in a similar manner than their biological counterparts. A very successful serpentine robot is Omnithread developed by Granosik, Hansen and Borenstein at the Mobile Robotics Laboratory in the University of Michigan. It is a novel design
consisting of 5 segments with moving tracks on all their four sides to assure propulsion even when the vehicle rolls over. The latest prototype is OT-4[9]. Rimassa et al. [10] have developed a serpentine robot with climbing abilities.

Even though serpentine robots are very promising for USAR applications, they lack the flexibility of real snakes and it is difficult to attach to them artificial skins because of their wheels or tracks. Snake robots, on the contrary, are similar to real snakes and inherit some of their properties. Perambulator[11], developed by Ye et al. at the Shenyang Institute of Automation in China, is able to perform a powerful propulsion and a high mobility. Every segment contains a kind of omni-directional wheel with free rollers that enables this robot to perform the serpentine locomotion (like real snakes).

Another modular robot with high locomotion capabilities is Hypercube[12], developed in our group. It can perform at least 5 different gaits: moving in a straight line (forward and backward), moving in a circular path, side-winding, rotating and rolling.

Our goal is to develop a new snake modular robot for search and rescue applications. In this paper we propose a preliminary novel design of its tactile system that will be used for improving the robot locomotion as well as its climbing and grasping capabilities.

II. SNAKE MODULAR ROBOTS FOR SEARCH AND RESCUE OPERATIONS

Modular snake robots have a lot of potential for search and rescue operations. Consider the scenario shown in figure 1 where there is a collapsed building (image of Puerto Principe, in Haiti). A USAR robot would be very helpful to explore inside the house and searching for people. The are two possible alternatives. On one hand, the robot can go through the bars of the window if its size is small enough. On the other hand, it can climb the wall and enter through the upper hole, but the robot should be flexible and have climbing capabilities. Snake robots could meet these requirements.

Now let's consider a general scenario as the one shown in figure 2 where there is a person trapped inside a house. First (fig. 2-1) the snake robot approaches the zone. Obviously, locomotion capabilities are needed. Second (fig. 2-2), the snake arrives to a wall and has to climb it. Therefore, climbing capabilities are necessary. Third (fig. 2-3), there are some pieces of rubble blocking the access to the person. The snake clears the path by grasping the objects and getting rid of them. Grasping capabilities are required. Finally (fig. 2-4), the robot reaches the person.

This scenario suggests that the robot should have at least locomotion, climbing and grasping capabilities. In previous work, our group has researched on these three areas. The locomotion of snake modular robots of any length has been widely studied and implemented on real robots[13] using sinusoidal oscillators that can be programmed with low cost 8-bit micro-controllers. A novel inspired modular climbing caterpillar that combines climbing techniques with locomotion capabilities was designed and tested in [14]. For climbing, a low-frequency vibrating passive attachment principle was successfully developed. Moreover, grasping capabilities are being studied and simulated[15].

The sense of touch is key to achieve these three capabilities. For locomotion, touch sensors provide the necessary feedback for enabling the robot to adapt its body to the terrain and therefore moving more efficiently. In addition, situations in which the robot get caught on the ground can be detected. For climbing, the attaching forces can be measured to guarantee that the robot will not fall down. Finally, the sense of touch in grasping allows the robot to apply the necessary force to remove objects from the path as well as optionally detecting the shape of these objects.

III. SENSE OF TOUCH

A. Introduction

Perceiving the environment in animals needs multimodal sensing capabilities. Humans combine sensory information from different sources as touch, vision, and hearing. If not using any of them, there is a gap in knowledge that makes differences between what is sensed and what is perceived[16]. Simple experiments of exploring objects after putting hands on an ice block for a while, therefore anesthetizing the skin, shows the difficulty of achieving stable grasp[17]. This occurs even when participants can see what they were doing.

There are several technologies and transduction principles that have been traditionally employed for the development of tactile sensing for robots, such as resistive, force, magnetic, optical, piezoelectric and capacitive based sensors[18]. Among them, capacitive sensors are commonly used in robotics applications. Despite their two major drawbacks, stray capacity and severe hysteresis, they are very small and sensitive. In addition they can be easily arranged in dense arrays. As an example of their high degree of integration, Gray et al. developed in 1996 an 8x8 capacitive tactile sensing array with 1 mm² area and spatial resolution at least 10 times better than the human limit of 1 mm. Schmidt et al. [19] designed a novel array of capacitive sensors for grasping purposes, which couple to the object by means of little brushes of fiber. Maggiali et at.[20] proposed the novel idea of using a mesh of triangles for covering three-dimensional surfaces. They have designed a triangle module witch contains 16 capacitive sensors and all the electronics.

Currently, it can be found commercially capacitive based touch array sensors such as the ones provided by ‘RoboTouch’
and ‘DigiTacts’\(^1\). Also, there exists various commercial ‘capacitance to digital converter’ chips such as the AD7147 ‘Cap-touch’ chip from Analog Devices. It have been successfully employed for the triangle module implementation[20].

B. Touch rings

Real snakes have a continuous skin, like the mammals, in which the sense of touch is distributed uniformly throughout the whole body. Designing an artificial skin similar to that of the snakes, with the same flexibility and equipped with touch sensors, is a big challenge. It still has not been realized, to the best of our knowledge. In our approach, we propose the idea of touch rings (figure 3) located at fixed positions along the body axis. Therefore, the sense of touch is not continuous and uniformly distributed but discrete and concentrated in touch zones. Although it is not continuous, the distance between two consecutive rings (d) can be set according to the application. If higher touch resolution is required, this distance can be reduced. Thus, this approach is very versatile giving the designer the possibility of changing it.

In our opinion, USAR operations do not require a very high resolution. It should be high enough to allow the robot to perform the locomotion, climbing and grasping efficiently. As a starting point we are planning to use one touch ring per module.

C. Touch strips

Each touch ring consist of a flexible capacitive strip that is bended (figure 4). One advantage of this design is that it can be fitted into different snakes with different sections of the body trunk, due to the fact that the strip is flexible. In the example of figure 3 the section is circular, but it can also be applied to snakes with other section shape, like a square. The strip will fit regardless of the shape of the body section.

Therefore, our design is very versatile and it is valid for a lot of different snakes. Moreover, the width and length of the strip can also be changed according the application. The wider it is, the more capacitive sensors can be embedded.

D. Capacitive sensor principle

The touch strips are based on capacitive sensors. The strip is composed of three layers (figure 4). The total thickness is expected to be less than 1mm. The first layer is a flexible printed circuit board (PCB) containing the electrodes for the capacitors. The second layer is the dielectric. This material should be compressible and expandable. We are testing different candidates, such as silicone (S60) and polyester (PEPT) based ones. The third layer is the ground plane.

When normal forces are applied to the strip surface, the dielectric layer is compressed and the distance between the electrodes and the ground plates is reduced. Therefore there is a change in the capacitance of the capacitors that can be measured. The material in the second layer must recover its initial thickness when no forces are applied.

E. Electronics

Various commercial capacitive-to-digital converters chips are available. In this work, we are using the MPR121 from Freescale\(^2\) because of its low cost and \(I^2C\) bus compatibility. Its main features are summed up in table I. It can measure capacitance ranging from 10 pF to 2000 pF. Minimum size of the electrodes should be less than 0.25 cm, if high dielectric materials are used. Once capacitance is calculated, it runs through a couple of levels of digital filtering allowing for good noise immunity in different environments without sacrificing response time or power consumption.

Some of the major advantages comparing with the AD7147 chip include an increased electrode count, a hardware configurable \(I^2C\) address, an expanded filtering system, and completely independent electrodes with auto-configuration built in. Regardless every chip can read only 12 capacitive sensors, bigger arrays can be made by connecting different MPR121 chips through \(I^2C\) bus. When bandwidth is not enough for

---

\(^1\)http://www.pressureprofile.com

\(^2\)http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPR121
transmitting all chip readings, digital multiplexers for communications or other microcontrollers should be used.

### IV. Mechanics

#### A. Modules

Since 2004 our groups have been working on modular robots. We have designed different modules, grouped into two families: Y1 and Cube-M (figure 5). The Y1 family is very cheap and very easy to build. It is intended for fast-prototyping, testings and educational purposes. The modules have one DOF actuated by a low-end RC servo (futaba 3003). The electronics (not shown in the figure) includes an 8-bit PIC16F876A microcontroller. This family is composed of three modules: Y1, RepY1 and MY1. The Y1 consist of 6 laser-cut pieces of plastic glued together. It is mainly intended for educational purposes as it is very cheap and easy to build by students. RepY1 modules are the “printable” version that can be manufactured using a commercial 3D printer or a low-cost open Reprap machine\(^3\). The MY1 version is made of aluminum. It has three pieces that are joined together using screws and nuts.

The Cube-M family[21] is an improvement over Y1. These modules are made of aluminum and are designed to carry more weight and to build bigger modular robots with 1D and 2D topologies. The electronics has been improved and embedded into the modules.

#### B. Robot prototypes

Some of the modular robots we have built are shown in figure 6. We have mainly concentrated on studying the locomotion of modular robots with 1D topology. This group can be divided into two subgroups: pitch-pitch and pitch-yaw connecting modular robots. The former is composed of those robots in which all the modules pitch up and down (fig. 6c and 6f). This configuration is kinematically similar to that of the caterpillars in nature. Therefore, these robots are very useful for studying the climbing capabilities, even though they only can move forward and backward. The later comprises the robots with pitch and yaw modules. These snakes can move on a plane (fig. 6b and 6e). When sinusoidal oscillators are used for their control, they can perform at least five different locomotion gaits[13]: moving in a straight line, circular path, side-winding, rotating and rolling. The relationships between the oscillator’s parameters (amplitudes and phase differences) determine the type of movement that will be executed.

Modular robots with a 2D topology have also been developed (fig6d) but their use for search and rescue operations is left for future work.

#### C. Touch strip prototype

The first prototype of touch strip is being made and designed, but a preliminary proof of concept has been built. It is shown in figure 7a. It consist of one layer with 12 electrodes in a flexible PCB, which has been taken from a commercial flexible keyboard. The strip containing the 12 electrodes has been cut and three wires have been soldered for performing preliminary tests. The total strip length is 208mm.

In figure 7b the strip is placed around one MY1 module, which has a square section. As the strip is flexible, it can be fitted to any shape. There are three electrodes per side, but only the three at the bottom are currently being tested. The proof of concept is being done with the pitch-pitch minimum configuration robot consisting of two MY1 modules. This is the minimum modular robot with locomotion capabilities[13].

\(^3\)http://reprap.org/wiki/Main_Page

---

**Table 1**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of electrodes</td>
<td>12 electrodes</td>
</tr>
<tr>
<td>Measurement range</td>
<td>10 pF to 2000 pF</td>
</tr>
<tr>
<td>Sensitivity range</td>
<td>-0.01953 to -0.19531 pF / ADC count</td>
</tr>
<tr>
<td>I2c bandwidth</td>
<td>400 Khz</td>
</tr>
<tr>
<td>Size</td>
<td>3x3x0.6 mm(^2)</td>
</tr>
</tbody>
</table>

---

**Figure 5.** The Y1 and Cube-M families of modules.

**Figure 6.** Some of the modular robot prototypes. a) Dr. Zhang and Dr. Gonzalez-Gomez sitting by their modular robots b) Hypercube snake, composed of 8 pitch-yaw Y1 modules. c) Cube Revolutions: a caterpillar with 8 pitch Y1 modules. d) A 2D topology star robot. e) A 5 pitch-yaw Cube-M module snake. f) A four MY-1 module pitch caterpillar going through a tube.
V. EXPERIMENTS

A. Touch strip experiments

First, we have tested and measured standard capacitive sensors provided by the Freescale Sensor Toolbox MPR121 Evaluation Kit shown in figure 8a. Then we have replaced them by our touch strip prototype to test the viability of our idea. Even if these preliminary electrodes are not meant for being used as capacitive sensors, because the come from a flexible commercial keyboard, they are working good enough and the MPR121 chip is able to perform measurements. The pressure exerted with the fingers is detected, as shown in fig. 8b. As can be seen, the idea is working. These results are very promising and let us continue to the next step: building another touch strip prototype that includes the three layers and the 12 electrodes.

B. Modular Grasping simulation

Some preliminary grasping experiments have been performed in simulation. The OpenRave simulator[22] has been used along with our OpenMR\(^4\) plugin for modular robots. The testing program obtain the contact points between the snake and the cylinder, as well as the applied forces. In figure 9a a

---

\(^4\)OpenRave Modular Robots plugin: http://bit.ly/9a3fXk

---

Figure 7. Preliminary touch strip prototype. Only one layer is being tested. a) The touch strip with 12 electrodes along with the pitch-pitch minimum configuration robot. Only three electrodes are connected. b) The touch strip is placed around one module to form a square “touch ring”

Figure 8. Testing the touch strip with the Freescale Sensor Toolbox MPR121 Evaluation Kit. a) The kit is connected to the standard electrodes supplied by the manufacturer. b) Screenshot of the measures taken from the three electrodes of the touch strip, using the software provided by Freescale

Figure 9. Simulation of a 30 module snake robot grasping a cylinder. a) Simulation in OpenRave. b) The contact point cloud obtained from the simulation. c) The contact point cloud view from another angle, so that the cylinder contour can be seen
30 module snake robot grasping the cylinder can be seen. The contact point cloud is shown in fig. 9b. When the orientation of these points is aligned with the observer, the cylinder contour can be glimpsed.

VI. CONCLUSIONS

We have presented a novel idea for implementing the sense of touch in snake modular robots. It is based on touch strips placed around the snake section, forming touch rings with embedded capacitive sensors. Even if this work is in a very preliminary stage of development, the experiments carried out confirm the viability of this design. In addition, experiments on modular grasping have been simulated for obtaining the contact point and forces applied when a snake robot grasp an object.

Currently we are working in combining the locomotion, climbing and grasping capabilities with the sense of touch for the realization of a modular snake robot for urban search and rescue operations.

VII. ACKNOWLEDGMENTS

This research is partially supported by the HANDLE project, which has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement ICT 231640.

REFERENCES

Abstract—Inter-module communication is integral to modular robot systems. Communication links must tolerate module misalignment and implement the neighbor-to-neighbor communication model. We propose a system based on wireless radios that scales to robots of arbitrary size. We present a multi-radio single-channel architecture and validate its performance through hardware experiments. Results show that wireless radios can provide reliable neighbor-to-neighbor communication suitable for modular robots.

I. INTRODUCTION

Self-reconfiguring (SR) modular robots rely on communication between modules in coordinating movement for reconfiguration and locomotion. Communication is also essential for sensing and perception in fusing data from many distributed sensors. We are interested in communication systems that are based on wireless radio frequency (RF) links. We envision wireless RF communication as a robust solution for implementing communication systems in SR robots.

Despite the central role communication plays in SR systems, it remains a major research issue [1]. Communication in existing SR robots is based on infrared (IR) or wired links. The problem is that these physical layers are not well-matched to module connection and disconnection.

The main challenge is that communication links must tolerate uncertainty in module alignment. Precise alignment presents a difficulty to connector design but is a requirement for IR or wired systems. If the precision requirement is relaxed, problems with crosstalk and neighbor detection emerge [2]. Coupling the physical implementation of the communication system with the mechanical design of the connection system increases the difficulty of both problems. We believe this coupling is unnecessary.

Our approach is to use wireless RF links. Radios do not require precise module alignment and can provide global as well as local communication. The most obvious implementation is an ad-hoc mesh network, but mesh networks introduce challenges of their own. Standard mesh networks act like a communication bus; nodes cannot transmit in parallel. The seminal paper by Gupta and Kumar [3] shows that the capacity of mesh networks scales by \( \theta(\sqrt{n}) \) as the number of nodes \( n \) grows to infinity. This implies that the bandwidth available to each node is \( \theta(1/\sqrt{n}) \) and hence mesh networks are not suitable for applications with large numbers of nodes. However, this bound does not hold when communication is local [4]. The key insight in the modular robots case is that in implementing decentralized algorithms, communication is predominantly neighbor-to-neighbor.

In previous work, we proposed a new architecture where each communication node uses multiple radios and multiple channels [5]. A single communication channel can be safely “recycled” assuming a minimum physical separation distance. This threshold is determined analytically for the
given radio hardware. The idea is that it is possible to compute a channel assignment that allows all neighbor pairs to communicate using their assigned channel without interference. A special case is when all pairs of radios are above the separation threshold. In this special case, only a single channel is needed.

This paper proposes a system based on the multi-radio single-channel case and empirically evaluates the performance of a system with ten unactuated modules. A typical setup is shown in Fig. 1a. Results support our main hypothesis that, unlike a standard mesh network, the performance of our proposed system stays constant as network size increases (Fig. 1b). We also validate tolerance to misalignment and hardware support for neighbor detection.

II. RELATED WORK

Communication systems based on IR or wired links are discussed in [6]. The majority of SR robots fall into this category. Bluetooth, a low-latency mesh network system with a master-slave architecture, is successfully used in the YaMor robot[7]. YaMor is the most notable example of RF communication used for control. Bluetooth is effective in small robots, but still faces the same scalability issues as other mesh networks. The other common use of wireless radios in SR robots is for human-robot interaction [8], [9], [10], [11].

The scalability of an off-the-shelf ZigBee mesh network in a typical SR robot workload was empirically evaluated in our previous work [12]. ZigBee is sufficient for small robots, but the system quickly saturates with more than fifteen modules. The architecture presented in this paper is a special case of the general multi-radio multi-channel approach presented in [5].

The communications community has devoted many years of effort to developing efficient ad-hoc mesh networks [13]. However, commercial applications do not satisfy the locality assumption that SR systems can exploit. The problem of designing a true neighbor-to-neighbor architecture has not been addressed by mesh network researchers.

III. NETWORK ARCHITECTURE

Our parallel wireless communication system is a multi-radio network architecture. A separate radio is assigned to each connecting face, as exemplified by radios $A_1$ and $B_1$ in Fig. 2.

If there exists an infinite number of orthogonal channels then we can simply assign a globally unique channel to each pair of neighbor radios $(A_i, B_j)$. However since there exists only a finite number of channels, in general a channel allocation scheme is required to ensure radios on the same channel do not interfere with each other. In our previous work [5] we investigated how to compute the minimum number of channels required and found that in some special cases only a single channel is required.

The architecture we propose in this paper is a single-channel scheme, where the same channel must be reused at every neighbor-neighbor pair $(A_i, B_j)$. This is possible because the radios are physically close together. The signal strength between each pair is greater than the interference signal from all other transmitting radios by the co-channel rejection ratio (CCRR) of the radio hardware. The CCRR quantifies the necessary signal to interference ratio (SIR) such that the desired signal can be correctly received without corruption.

To achieve the necessary SIR, neighbor-neighbor pairs must be located close together whilst non-neighbor radios must be separated far apart in Euclidean space. Using the free-space path loss (FSPL) equation [14], we derive an equation to determine the Euclidean separation ratio required between a neighbor-neighbor pair and other interfering radios to achieve an SIR greater than the CCRR of the radios:

$$\frac{d_{\text{interferer_distance}}}{D_{\text{neighbor_distance}}} \geq 10 \frac{\text{CCRR}}{20}$$

where CCRR is the co-channel rejection ratio in dB, $D_{\text{neighbor_distance}}$ is the separation between neighbor-neighbor radios $(A_i, B_j)$ and $d_{\text{interferer_distance}}$ is minimum separation to any other radio $\{ (A_j, B_j); j \neq i \}$.

From Eq. 1 we can see that we want to minimize the separation between the neighbor-neighbor pairs and maximize the separation between radios as this increases the SIR. Given that the CCRR is typically 2dB then $d_{\text{interferer_distance}}$ only needs to be 1.26 times $D_{\text{neighbor_distance}}$ which is easily achievable. From a practical perspective however a larger separation ratio of approximately 3 would be needed to negate the effects of multi-path fading, hardware tolerances, anisotropic antennas and for neighbor detection which we shall discuss in Sec. III-A.

Our architecture assumes that Carrier Sense Multiple Access (CSMA) is not used. CSMA is a channel access scheme in which a transmitting radio will check that the channel is idle before broadcasting so that the signals are not corrupted [15]. As we already guarantee neighbor-neighbor pairs have a SIR greater than the CCRR, the CSMA algorithm becomes redundant.
A. Neighbor Detection

Each module in the robot is assigned a globally unique network identifier and all transceivers within a single module will take on the same network ID. In addition, each radio is allocated a unique address corresponding to the connector to which it is assigned.

To detect a new neighbor module, all radios with open connectors periodically activate and listen for connection request messages. If a connection request is received a connection protocol is initialized. At the same time, as a module is maneuvering into a new position, it periodically sends out a connection requests and listens for a response from an open connector.

As a connection request from a reconfiguring module will most likely be heard by multiple open connectors as crosstalk, we must prevent radios from commencing the connection protocol with the incorrect connector interface and/or module. This is done by checking the signal strength of the connection request message. If the suggested separation ratio of 3 is provided, then a connection request message from an intended neighbor will be at least 9.5dB greater than a connection request message received as crosstalk. The possibility of communicating with the wrong connector can be further reduced through determining the global position a module through its network identifier and ensuring that the module’s position corresponds with the connector on which it is connected. In Sec. V-E we demonstrate in hardware the robustness of using signal strength thresholds to eliminate crosstalk and assist in neighbor detection.

IV. EXPERIMENT SETUP

In this section we discuss the hardware and software that is used in the experiments. We begin by discussing our choice of radio hardware.

A. Radio Selection

In selecting the radio hardware, the following criteria were considered:

1) A high CCRR is preferable (-3dB or more).
2) Radios can operate without using Carrier Sense Multiple Access (CSMA).

Based on the above criteria, IEEE 802.15.4 and ZigBee standard radios were used for the experiments. IEEE 802.15.4 operates in the unlicensed RF spectrum of 868, 915 and 2400MHz with respective data rates of 20, 40 and 250kbps. In our experiments, 2400MHz radios were selected for the higher data rate. The IEEE 802.15.4 standard also includes the capacity for a time division multiple access (TDMA) scheme which does not use CSMA. The ability to operate the radios without CSMA is important as a single radio can easily transmit a signal sufficiently strong to force all modules in the SR robot to back-off in the CSMA algorithm.

IEEE 802.11a/b/g and Bluetooth radios were also considered however they consume more power and use a more complicated protocol stack. However their high bandwidth could make them better suited to other applications.

B. Hardware and Testbed Implementation

Microchip MRF24J40 2.4GHz IEEE802.15.4 compliant modules [16] were chosen to conduct the experiments. These modules contain an onboard PCB antenna which saves the need to consider complex antenna and RF circuitry design issues. The modules are interfaced to a custom-built control board via an SPI bus. The custom control board contains a 32-bit STM32F101C6 microprocessor operating at 36MHz with 32KB flash and 6KB RAM as seen in Fig. 3. User control of the setup is achieved via RS232.

![Image](image-url)

**Fig. 3:** Six Microchip 802.15.4 modules connected to a custom STM32F microprocessor board.

In an effort to create results that are realistic, the radio architecture is tested in a representative modular robot to model the operational environment. The modular robot is made from pieces of 3mm thick acrylic that are assembled into a 100x100x100mm cube as seen in Fig. 4a. This provides a \( D_{\text{neighbor\_distance}} \) of 6mm and a \( D_{\text{interferer\_distance}} \) of 70mm and results in a \( \frac{D_{\text{neighbor\_distance}}}{D_{\text{interferer\_distance}}} \) separation ratio of 11.66. This meets the minimum 1.26 separation ratio calculated in Sec. III. Radio modules are attached to the inside surface of the cubes using adhesive putty and connected to the microprocessor board which slots into the acrylic cube as seen in Fig. 4a.

C. Software

The testing software is based on the IEEE 802.15.4 protocol stack. Each radio can be dynamically configured via RS232 by the user as a transmitter or receiver and assigned addresses and network identifiers allowing radios to distinguish between a desired transmitter and interference transmitters. The purpose of the software is to measure throughput between pairs of radios on opposite sides of an inter-module connector by configuring the transmitter to continuously transmit data at the fastest possible rate allowable by the radio hardware (250kbps). The corresponding receiver radio then counts the number of messages received and also performs cyclic redundancy checks (CRCs) on all incoming data to ensure that interference has not corrupted the messages. The receiver radio automatically performs hardware checks on the address and network identifier of the sender hence ensuring that the radio is not receiving data from other interfering transmitters.
In the first experiment we establish the baseline throughput rates of two connected radios with increasing numbers of interfering transmitters as shown in Fig. 4b. The setup consists of two modules with radios on all six faces placed side by side as seen in Fig. 4a. When there are no interfering transmitters, the baseline throughput rate is approximately 224 messages per second averaged over a fifteen-minute period. This throughput rate is close to the carrier data rate of IEEE 802.15.4 with the difference accounted for in the inter-frame spacing and preamble bits required by the hardware. Each message at the physical layer is a 128-byte packet of data.

Following this additional radios are turned on and set to the same channel to act as interference as shown in Fig. 4b. In total ten interfering transmitters are turned on. Theoretically the additional radios should not have an impact on the throughput rate provided that the communicating pair have an SIR greater than the radios’ CCRR which we confirm after each interfering radio is enabled. However in reality some interference still exists. Fortunately the results show the throughput loss from the interference is not directly proportional to the number of interferers. Hence as the number of radios scale up, message losses will not increase.

To further verify this principle, the same experiment was also conducted with Jennic IEEE 802.15.4 ZigBee radios which are compatible with the Microchip radios. Using the Jennic radios message losses did not occur even with eleven interfering Microchip transmitters enabled. This substantiates the notion that the message losses are a result of the Microchip hardware and that switching to another hardware such as Jennic ZigBee radios could potentially solve the anomaly.

A comparison between our hardware results and the theoretical maximum throughput of a standard mesh network (ZigBee or otherwise) is given in Fig. 4b. In a standard mesh network, the throughput capacity of each radio decreases at the rate of $\frac{1}{n}$ where $n$ represents the number of modules in the robot. On the other hand, throughput is constant in our parallel communication architecture which allows our system to outperform a standard mesh network even in robots with few modules.

1) Misalignment and Orientation Tolerance: An existing problem with current communication hardware such as wired connections and IR is the need for accurate alignment, orientation and proximity between two connecting modules to within a few millimeters. The use of RF transceivers provides a much greater tolerance for misalignment and orientation. The setup shown in Fig. 4a is used to measure the misalignment and orientation and inter-module gap tolerances of the hardware as depicted in Fig. 5a. During the test, all ten interfering transmitters are also enabled.

Fig. 5b shows the throughput rate of two connected radios with varying misalignment and orientation as illustrated in Fig. 5a. The results indicate that even with 2cm misalignment, the lowest throughput is still over 192 messages/second or 87% of the initial throughput. This result is significantly better than the misalignment tolerances of IR and hardwired connections which are conservatively no better than 0.5cm.

Fig. 5c shows the throughput rate as the gap between two modules is increased. This test is done with no misalignment but with 0, 90 and 180 degrees of rotation. The results indicate that little throughput capacity is lost within first 1cm regardless of orientation and reasonable data rates are still achievable even with a separation gap 2.5-3cm with 0 and 180 degrees of rotation.

In both the misalignment and inter-module gap experiments, the relative orientation of the two radio antennas has a considerable impact on misalignment and gap tolerances. This is due to the fact that the gain pattern of the printed PCB antenna is not omni-directional but a typical torus shape which is not ideal for this application. The use of a patch or
Throughput with Inter-Module Misalignment and Varying Module Orientations

(a) Inter-Module gap
Rotation
Inter-Module Misalignment

(b) Throughput with Inter-Module Gap and Varying Orientations

(c) Throughput with Inter-Module Gap and Varying Orientations

Throughput with Inter-Module Gap and Varying Orientations

(c) Throughput with Inter-Module Gap and Varying Orientations

Fig. 5: (a) Illustration of module misalignment and rotation setup. (b) Throughput of modules with varying inter-module misalignment and varying orientations. (c) Throughput of modules with varying inter-module gap and varying orientations.

Throughput with Inter-Module Gap and Varying Orientations

(c) Throughput with Inter-Module Gap and Varying Orientations

B. Immunity from External 2.4GHz Interference

With the rapid proliferation of 2.4GHz RF transmitters from sources such as WiFi and Bluetooth, interference from external sources is inevitable. A benefit of our architecture is its immunity from interference due to the large SIR it achieves from the physical proximity of adjacent communicating pairs. The setup shown in Fig. 4a is also used to demonstrate noise immunity. All ten interfering Microchip transmitters are enabled and set to transmit on channel 22 which has a center frequency of 2.460GHz and a spread of 5MHz. The 802.11b/g network is set to channel 11 which has a center frequency of 2.462GHz and a spread of 22MHz.

Two cases are considered in testing noise immunity: an 802.11b/g transmitter close by, and an 802.11b/g network operating in the same area but with transmitters far away. To test the first scenario, a laptop with a wireless card is placed approximately 10cm from the modules. In this case the laptop is unable to connect the local access point which is expected whilst the modular robot network continued to function normally. Since 802.11b/g transmitters employ a CSMA algorithm, the transmit protocol would prohibit the radio from transmitting if it detects signals on the same channel above a certain energy level. This will always be the case given the proximity of the modular robot transmitters to the laptop. Note that the Microchip radios are still set to transmit at -30dBm which is significantly below its 0dBm transmit capacity.

To test the second case, the laptop and modular robot network are separated by approximately 5m. Now both networks are able to co-exist and the performance of the modular robot network is the same as the baseline experiment. The 802.11b/g network also began to function normally.

C. Three-Dimensional Networks

Most analytical and hardware implementations of mesh networks are performed on two-dimensional planes which inherently results in a lower density of transmitters per volume of space. Our system works no differently in two and three dimensions as can be seen in Fig. 6a and Fig. 1a which shows two 3D setups with seven and ten modules respectively. In the first setup as shown in Fig. 6a, the module in the center has radios attached to all six sides of the module. Each radio also has a corresponding neighbor pair on the connected neighbor modules. Finally all radios in the setup are tuned to the same channel.

This experiment represents a worse case scenario test where all six sides of the module are communicating at the full data rate of the radio hardware over a 40-minute period. Fig. 6b shows the throughput rate for all six sides of the inner module which indicates consistently high throughput over the 40-minute period. In a standard mesh network configuration, the theoretical maximum throughput would
Throughput in a 7-Module 3D Network

(a)

(b)

Throughput in a 10-Module 3D Network

Fig. 7: Throughput results in a network of ten modules in a 3D network layout.

D. Chain Network Topology

The chain network topology is a commonly used benchmark for analyzing mesh networks as it is commonly encountered in real networks. Typical mesh networks perform extremely inefficiently in chain networks. This is because only one link can be active at any one time if the range of the radios encompasses the entire area of the network which is the case in modular robots. A parallel communication architecture on the other hand is able to utilize all links simultaneously and hence greatly increases the speed and bandwidth capacity of the network. Fig. 8a illustrates a setup containing seven modules and six separate inter-module connections. The graph in Fig. 8b shows the throughput rate at each of the six inter-module connections averaged over a one-hour period. Whilst a few links are not achieving the theoretical 224 messages/second of throughput, the overall performance is still significantly better than a mesh network which would only achieve 37 messages/second.

Fig. 8: (a) Setup with seven modules in a chain topology. (b) Throughput of seven modules in a chain topology.
E. Automatic Neighbor Detection and Crosstalk Elimination

Neighbor detection and crosstalk elimination are common problems associated with the use of IR communication in modular robots [2]. RF communication suffers from the same problems except to a much greater extent as radio waves easily permeate through and refract around modules. In this parallel communication architecture, both neighbor detection and crosstalk elimination is addressed by observing the receive signal strength indicator (RSSI) of incoming data as discussed in Sec.III-A.

Using the setup shown in Fig. 4a, we measure the RSSI difference between the pair of connected neighbors and the ten other interfering radios and find a minimum RSSI difference of -15dBm. This large RSSI difference means that using simple threshold techniques for neighbor detection and crosstalk elimination is very robust even without higher-level algorithms that would normally be available in a real modular robot implementation. In fact using this simple RSSI threshold technique, the radios are able to correctly detect neighbors and eliminate crosstalk with misalignments of up to 2cm and inter-module gaps of 1cm regardless of orientation.

Fig. 9 shows the results and setup used to test the neighbor detection and crosstalk elimination properties in a realistic 3D network. The setup consists of one mobile module containing four radios (top, front, left and right) as annotated in Fig 9c. This mobile module is then surrounded by a static network of modules. The mobile module is manually moved between three different positions as shown by Fig. 9a-9c.

The throughput versus time plots for the four radios in the mobile module are shown in Fig. 9d, which also shows when a new neighbor is detected. The time at which the mobile module moves is clearly visible by the throughput dropping to zero for a few seconds before returning to full throughput and the detection of a new neighbor at the same time. In some cases the throughput drops to zero and remains there if no new neighbor is detected as seen in plot “Left” at time 156 seconds and similarly in plots “Top” and “Front.”

Crosstalk elimination is achieved through a combination of software and hardware. In hardware, crosstalk elimination occurs naturally when the SIR is greater than the CCRR and crosstalk is treated as interference that is filtered out. However if the neighbor radio is not transmitting and interfering radios are, software is needed to remove crosstalk. Each time a new neighbor is detected, both radios record the address and network ID of their neighbor and all future transmissions is stamped with the neighbor’s address and network ID. Both radios will then be set to discard all packets that are not addressed correctly.

In this experiment the modules detect new neighbors using only the RSSI of messages sent by their new neighbors. The mobile module is not informed that a move is about to occur and similarly static modules are not informed that the mobile module is about to disconnect or connect. Given these circumstances the results already show accurate and fast neighbor detection. However robustness can be further improved in a real modular robot implementation as modules would know a priori of planned connections or disconnections.
VI. DISCUSSION

We have presented a new wireless communication architecture designed to allow simultaneous neighbor-to-neighbor communication at constant bandwidth for SR robots of any size. The neighbor-to-neighbor model is required by sensing and control algorithms, but not possible with standard wireless radio networks that do not allow multiple nodes to transmit in parallel.

We performed experiments that validate the performance of the system in hardware. These hardware experiments are significant because the true performance of wireless radios is notoriously difficult to simulate. Our experiments give us confidence that the system will satisfy challenges of misalignment, crosstalk, and neighbor detection in real SR robots. Results also indicate that the system will be tolerant to outside interference sources, such as WiFi.

There are several possible extensions to the system. Infrequent global communication is possible by reserving an auxiliary channel for a standard mesh network implementation. An additional radio in each module would be dedicated to the global channel. Also, bandwidth or power consumption can be varied through choice of radio hardware.

Although we exploited the single-channel case in this paper, the full multi-channel architecture is more broadly applicable in distributed robotics. We are currently investigating an application to robot teams that assume the neighbor-to-neighbor communication model.

ACKNOWLEDGMENTS

This work is supported by the Australian Research Council (ARC) Centre of Excellence programme, funded by the ARC and the New South Wales (NSW) State Government. We would like to thank Ritesh Lal for the support he provided with the radio hardware.

REFERENCES

Genderless Connection Mechanism for Modular Robots Introducing Torque Transmission Between Modules

Andreas Lyder, Ricardo Franco Mendoza Garcia, and Kasper Stoy

Abstract—We describe the design and implementation of a genderless connection mechanism for modular robots, capable of transferring torque between two modules. The connection mechanism allows the usually integrated motor to be removed from an actuator module, and then to drive the actuator module by a separate motor module. Additionally, we introduce Thor, a new modular robot taking advantage of the proposed connection mechanism, and, consequently, achieves high levels of heterogeneity and versatility. Based on these results, we believe that the proposed connection mechanism has the potential to simplify module design, and reduce the overall cost of modular robots.

I. INTRODUCTION

Heterogeneous modular robots consist of a set of different modules with varying functionality. Typically, these modules will not work individually, but when put together, each module’s functionality adds up to form a robot capable of performing a task. The modules with some sort of actuation usually integrate a motor, increasing the cost and complexity of the module. In this paper we propose a connection mechanism capable of transferring torque between two modules. This makes it possible to drive actuated modules by general motor modules and remove the integrated motors. By being able to drive various actuator modules using general motor modules, the design and development of new actuator modules is simplified, and the usage of expensive high-grade motors are optimized.

One of the most difficult but also one of the most frequently asked questions in modular robotics research is: “Which applications are modular robots suitable for, and why do they need to be modular?” People are mainly concerned that modular robots are not able to solve a specific task better and more efficiently than a robot developed for that specific task. This is partly due to the fact that the research areas of modular robotics are still being explored, but merely that modular robots are not meant for specific application domains. Instead, modular robots are meant for a large variety of application domains. This means that they may not be able to solve a specific task very well, however, they may be able to solve many different tasks to a satisfactory level. Some of the application domains, which has been suggested, are maintenance [1], space applications [2], search and rescue [3], and morphing production lines [4].

The main focus in the design of the state-of-the-art modular robots has been on self-reconfiguration [5], [6], [7], and locomotion [8], [9], [10], where in some cases manipulation capabilities have been added later. In this paper we present a new modular robot called Thor, which utilizes the proposed connection mechanism to achieve higher versatility with both locomotion and manipulation in mind. The Thor modular robot is partly inspired by the experience from our participation in the ICRA 2008 Planetary Robotic Contingency challenge. In the following sections we will describe the design and implementation of the Thor modular robot and the proposed connection mechanism. The Thor modular robot will be tested for the first time at the ICRA 2010 Planetary Robotic Contingency challenge, but we conclude that the proposed connection mechanism has the potential to simplify module design and reduce the overall cost of modular robots.

II. THE ICRA PLANETARY ROBOTIC CONTINGENCY

The ICRA Planetary Robotic Contingency competition was held for the first time in May 2008. “This event simulates an unexpected problem occurring at a planetary habitat, where a robotic solution must be quickly developed and deployed, using only existing resources. ... The competition drives not only the development of versatile robotic hardware and on-board software, but also the design and development of programming and assembly tools capable of rapidly implementing a wide variety of capabilities.” [11]
The ICRA Planetary Robotic Contingency competition inspire researchers to show the potential of modular robots in an environment that resembles the real world by competing against each other. The research teams are given 4-5 hours to solve various tasks using a limited set of robotic tools, which were either provided by the organizers or brought by the teams themselves. The most successful teams were able to solve a majority of the tasks using a mobile platform with a reconfigurable manipulator on top. Between tasks the teams reconfigured their robots for the specific tasks. With that in mind we decided to develop a new modular robot, with which we should be able to create, at least but not only, various mobile platforms various mobile platforms with various manipulations on top, using the same set of modules. We call this new robot Thor.

### III. The Thor Modular Robot

The name Thor comes from the Norse mythology, where Thor is the God of Thunder. Thor’s powerful hammer, Mjöllnir, is believed to be one of the most powerful weapons in the Norse mythology, and is said to be capable of leveling mountains. The Norse God Thor is also a direct descendant of the Norse Chief God Odin, after whom the Odin robot [12] is named. Since the Thor robot is partly inspired by the experience we have had with the Odin robot, it was a suitable name for this new robot.

The Thor modular robot is a robotic building kit, with which we can build a variety of robots solving different tasks in different environments. For this robot we assume that a human will be present to assemble and reconfigure the robot, therefore the robot is not able to self-reconfigure. We can imagine having a suitcase full of modules, which on demand can be brought to different scenes, where a robot can be build on site to solve the required task. Since the environments and tasks are not known beforehand, and the suitcase can only hold a limited set of modules, it is crucial that this set of modules are very versatile. By developing a new connection mechanism capable of transferring torque between two modules, we can remove the usually integrated module, and drive the actuator module by a general motor module.

The Thor modular robot must be prepared for unexpected problems happening in an unknown environment, and it must be able to be rapidly configured into different robots performing both locomotion and manipulation. Depending on the task and environment, there may sometimes be a need for many wheels, or sometimes there may be a need for many hinge joints. The motor module is able to drive a large variety of actuator modules, such as a wheel, a hinge joint, or a variety of end-effectors, such as a gripper. A motor does not need to be added to each module, which simplifies and decreases the cost of developing and producing new modules with new functionality. This allows us to develop a more versatile set of tools for the Thor modular robot. However, we will not be able to actuate more joints than the number of available motor modules, and the total size of a motor module and an actuator module is in comparison larger than an integrated module, but this is countered by the versatility.

### IV. The Thor Connector

The usability of a modular robot relies heavily on the interconnection between modules, and the connector is the most crucial component of the robot. One of the definitions of a modular robot is that it is dynamically reconfigurable [13], which means that it should allow for easy reconfiguration of modules to change the physical morphology of the robot. In terms of reconfiguration, we can divide the existing modular robots into two different categories, the self-reconfigurable and the manually reconfigurable. Self-reconfigurable modular robots are able to reconfigure its modules and change its morphology without human intervention, whereas the manually reconfigurable modular robots must be reconfigured by hand. The connection mechanisms for self-reconfigurable robots tend to be very advanced both mechanically and in terms of control. The connection mechanism proposed in this paper is developed for a robotic toolkit, where a human will be present. Therefore the Thor connector is designed to be manually reconfigurable, allowing it to be simpler.

The Thor connector is a genderless magnetic 8 redundant connector with optional rotational drive shaft. It consists of a few simple components, a base ring, a pcb with 6 spring contacts, a bearing and 16 magnets. Its dimensions are Ø70x5mm.

The base ring has 8 symmetrically placed pairs of a pin and a hole. Each pin and hole has a small 5x2mm disc magnet with opposite polarity. This allows two connectors to magnetically connect and align in 8 different angles. The connector also has 8 screw holes, in case the magnets are not strong enough to withstand the forces in the robot or exerted on the robot.

Inside the base ring sits a pcb with 6 spring contacts. Each spring contact is assigned to establish dedicated electrical connection between two connected modules, as follows: 2 Power lines, Vcc and Ground, 2 lines for a RS-485 differential bus, 1 line for detecting the angle of connection allowing the robot to detect its topology, and 1 line is still unassigned. The hole in the center of the pcb can either
be blinded or it can fit a bearing. The bearing can hold a 5mm hexagonal socket which in combination with a 5mm hexagonal shaft can transfer rotational power between two modules. This means that a motor module can drive other modules which requires rotation.

Each module may have several connectors, and from the backside of the PCB all the connectors of a module can be interconnected with power and communication. The backside of the PCB also allows us to add on a microcontroller board facilitating an Atmel AT91SAM7S256 microcontroller with an ARM-7 core, and RS-485 communication. The microcontroller board can be combined with additional electronics for specific modules, such as a motor controller and power electronics for a motor module. Though the connector is prepared for topology detection, the electronics is not fully developed and does not enable this yet.

V. THOR MODULES - THE INITIAL SET

Modular robots are designed to fit a chain-based, lattice-based, or a hybrid structure. In chain-based modular robots the modules are configured in chains, which typically can be branched into trees. When a robot is configured in a chain, the modules are able to actuate more freely. If the chains, however, are connected in a loop, the actuation becomes parallel. Chain-based modular robots are usually able to change their shape significantly by actuating their modules, and it is easier to create arms for manipulation and legs for locomotion. In lattice-based modular robots the modules are configured in discrete positions defined by a lattice. The lattice gives a modular robot a bit more substance, and since its modules are connected in parallel, its strength and stability do not only rely on a single module. To change the shape of a lattice-based modular robot it must either be flexible, so that the modules can actuate within the structure, or it must be reconfigured.

The Thor modular robot is a hybrid, and combines the ability to configure the modules in a chain and a cubic-lattice structure. By default, it has a rigid structure, and its modules are, therefore, not able to actuate within a lattice-based sub-structure. To enable this, flexibility would have to be added using flexible modules. However, the rigid lattice-based structure enables us to create sturdy bases, on which we can mount actuated chain-based structures.

To fit the design of modules within a cubic lattice we have defined a unit of measure called Thor Cubic Units (TCU). 1 TCU = 18mm. The Thor connector has a diameter of 70mm and is centered on a face which is 4x4x4 TCU. Fig. 3 shows the dimensions in TCU of different Thor Modules.

We have developed an initial set of 7 different modules, as follows:

1) Motor: The motor module is able to drive other modules that require rotation to actuate, such as the rotation, wheel and gripper module. Having separate motor modules simplifies developing new actuated modules, since they do not need to have their own motor. The motor module has a 5:1 planetary gearing driven by a maxon brushless DC-motor. The brushless DC-motor has a stall torque of 255 mNm, which should give us at least a torque of 1 Nm on the output shaft. It is all contained, including electronics, in a 3D printed (ABS) cylinder with a connector at each end. The cubic dimensions of the motor module are 4x4x4 TCU.

2) Cubic Node: The cubic node module is used for branching modules in a cubic lattice. The node is milled in aluminum and holds six connectors. The cubic dimensions of the node module are 4x4x4 TCU.

3) Rotation 165: The rotation modules can be driven by the motor module to rotate another module ±165 degrees. It is contained in a cylinder with a connector at each end. The cubic dimensions of the node module are 4x4x2 TCU (the cylinder is 2 TCU high).

4) Angle 90: The angle module has an L-shape and branches up to three modules in a 90-degree angle. Two of the connectors are directly opposite on the side face and allow rotation to pass through. The third connector is placed
5) Wheel: The wheel module can be driven by a motor to rotate. The cubic dimensions of the wheel are 4x6x6 TCU, where the connector is centered on the 6x6 TCU face concentric with the wheel.

6) Gripper: The gripper can be driven by a motor to grasp objects or other modules. The cubic dimensions of the gripper base are 4x4x2 TCU. Since the gripper is an end-effector, the dimensions of the gripper arms are not defined.

7) Battery: The battery module provides battery power to the robot. By connecting a 12V power supply the robot can run tethered, or the battery can be charged directly. The cubic dimensions of the battery module are not yet defined.

8) Wireless: The wireless modules allow for PC-to-robot or robot-to-robot communication using wireless ZigBee. The cubic dimensions of the battery module are not yet defined.

VI. FUTURE

The Thor modular robot is not yet able to detect its own configuration, and it must, therefore, be programmed for each configuration by the user. Detecting its own configuration will be necessary in the future for the robot to work autonomously. The connector is prepared for topology detection, however, the electronics does not yet support this. To enable the robot to detect its own configuration, and allow each module to identify itself in the configuration, the electronics should be updated.

The Thor modular robot also induces some new interesting control issues that rely on fast and stable communication. Since the motor can drive different actuated modules, the control of the motor must also be dynamic. If we e.g. connect a wheel to the motor, we need a speed controller, but if we connect a rotational or bending joint, we need an absolute position controller. Since a module driven by the motor module may have additional gearing, the absolute position must be measured as close to the output as possible. The controller then becomes inter-modular, and relies on the communication between the motor module and the module it drives. A solution to this could be to make a speed controller regulating the speed of the output shaft on the motor module. An actuated module is then able to ask for a specific speed on its input shaft, which can be provided by the motor module.

VII. DISCUSSION

The Thor modular robot is a robotic building kit, with which a human can assemble different robots capable of both locomotion and manipulation. To create a versatile robotic building kit, we need a variety of functionalities, such as actuated joints, end-effectors, wheels, legs, sensors, batteries. The functionality needed vary depending on the task and environment, either of which is not necessarily known beforehand. If a homogeneous modular robot on a cm-scale were to incorporate all of this functionality, each module would become extremely complex and big. The Thor modular robot is highly heterogeneous, and the functionality is divided into a variety of modules which can connect to each other.

Since Thor is heterogenous, only a fraction of the modules will typically be used for a single task, leaving a number of unused modules. The actuated modules are usually the most mechanically complex and expensive modules in a modular robot, partially due to high-grade motors. Therefore, if each actuated module has an integrated motor the overhead in cost and complexity from the unused modules could be fairly high. By developing a connector capable of transferring mechanical rotation, we have removed the integrated motor from the actuated modules. Instead we have developed a simple motor module which is able to drive a variety of actuated modules. This has made the actuated modules simpler and easier to develop and produce, and it allows us to have a larger variety of actuated modules. However, the number of actuated modules, which can be used in a single configuration of the Thor modular robot, are limited to the number of motor modules available. The total size of an actuated module plus a motor module will also be larger than an integrated module. Dividing the functionality of the Thor modules into simpler modules with less functionality requires at least two or more modules to create an actuated joint or end-effector. This will increase the size of the robot and also require a higher number of connectors. By making the connector simple with cheap components, and easy to produce, the overhead in cost and assembly is minimized.

The ICRA Planetary Robotic Contingency competition encourages researchers to show the potential of their modular robots in an environment resembling a planetary habitat. The planetary habitat is an example of an environment where unexpected problems may occur and must be solved by a robot. Other examples could be a crash site, where humans could be trapped in the ruins of a collapsed building, a factory, where a broken-down machine needs maintenance, or other environments where unexpected problems may occur. The competition is designed to simulate a situation that could happen in the real world, which is a great opportunity to test the potential of a modular robot. The Thor modular robot was inspired by the Planetary Robotic Contingency competition held at ICRA 2008, and will be tested for the first time at the same event held at ICRA 2010.

VIII. CONCLUSIONS

In this paper we have described the design and implementation of a new modular robot called Thor, emphasizing a genderless connection mechanism capable of transferring torque between modules. By introducing this connection mechanism we have removed the usually integrated motor from the actuated modules, and developed a general motor module. We believe that this has simplified the process of developing additional modules with new functionality, and decreased the overall cost of expanding the functionality of a modular robot. The Thor modular robot will participate in the ICRA 2010 Planetary Robotic Contingency, which will
show whether its high level of heterogeneity and versatility will pay off.

IX. ACKNOWLEDGEMENTS

The authors would like to thank all members of the Modular Robotics Lab at University of Southern Denmark, and Jesper Tordrup, Lars Peder Nielsen, and Stanislav Kucirek for their help and support.

REFERENCES

Characterization of Lattice Modular Systems by Discrete Displacement Groups

Nicolas Brener, Faiz Ben Amar, Philippe Bidaud
Université Pierre et Marie Curie - Paris 6
Institut des Systèmes Intelligents et de Robotique
4 Place Jussieu, 75252 Paris Cedex 05, France
e-mail : {brener,amar,bidaud}@isir.upmc.fr

Abstract—In this workshop we provide a method to determine and compare the reconfigurability of lattice systems. First it shows the difference that exists between the reconfigurability and self-reconfigurability features of a lattice system. Then a method using displacement groups is introduced to characterize these features. Based on this method, these features are then compared for some existing lattice systems.

I. INTRODUCTION

As stated [1] lattice systems are a class of modular self-reconfigurable robots having their “modules arranged nominally in a 2D or 3D grid structure. For this category, there are discrete positions that a given module can occupy”. Most lattice systems have continuous actuation mechanism [2], [3], [4], [5], [6], others use discrete [7] or passive [8] actuation. Some systems [9], [10], [11] with continuous actuation may also use both lattice and non-lattice configurations modes. In the sequel we consider only the lattice utilization, and therefore all these systems are considered as lattice ones. Only lattice systems performed successfully autonomous self-reconfiguration because the discrete configurations helps to bring the connectors to matching positions, and simplify the planning of the reconfiguration sequences. Nevertheless, the kinematical conception of lattice robots is difficult because modules with arbitrary discrete joint configurations will not necessarily lead to lattice systems; conversely, non-discrete joint configurations may lead to lattice systems because of redundant mechanisms or singular configurations. This issue was addressed in [12], [13] by proposing a framework for the kinematical design of lattice robots relying on discrete displacement groups theory. The configuration of a module $M_i$ is the $n$-tuple $X_i = (x_i^1, \ldots, x_i^n)$ where $x_i^1, \ldots, x_i^n \in \mathbb{R}^6$ denote the $n$ poses of its $n$ connectors $c_i^1, \ldots, c_i^n$. For a lattice system it is assumed that a module can have only a finite number of relative poses of its connectors. Any configuration of a system $S$ of $k$ modules is given by the set $\mathcal{X} = \{X_1, \ldots, X_k\}$. As stated in [12], [13], a system is said lattice if, for any configuration $\mathcal{X}$, the pose $x_i^j$ of a connector $c_i^j$ belongs to a discrete set of poses $\mathcal{O}_i^j$, called orbit, and that all the orbits $\mathcal{O}_1^1, \ldots, \mathcal{O}_k^n$ are generated by an unique discrete displacement group $G$. Thus, for any poses $x$ and $y$ of a connector, $x$ and $y$ belongs to the same orbit and there exists $g \in G$ such that $y = gx$.

In this workshop we propose an opposite approach by using discrete displacement groups to characterize the kinematical features of existing lattice robots. This shows that our definition of lattice robots is consistent with the previous one by characterizing successfully existing “lattice” systems. The first section introduces and compares the concepts of reconfiguration and manipulation group. The second section proposes a method to establish the configuration and manipulation groups for any lattice system. Based on this result, the third section describes and compares the feature of several lattice system, before concluding.

II. RECONFIGURATION VS SELF-RECONFIGURATION

In a “Self-reconfiguration” the lattice system changes its topology by disconnecting, manipulating and reconnecting its modules. In a “reconfiguration” an external device manipulates the modules of the lattice system to reconfigure it. The external device may be, for example, a human operator, and is considered as a manipulator that can reach arbitrary poses in an unlimited workspace. The following example illustrates that some feasible reconfigurations may not be done by self-reconfigurations.

Consider the module depicted in figure 1. It has a revolute joint with two configurations and three hermaphrodite connectors represented by the triangles.

Fig. 1. A module with two joint configurations

The figure 2 represents a set of modules assembled together with a lattice represented by grey and white chessboard cells. Obviously, in a reconfiguration it is possible for an external device to exchange two modules belonging to arbitrary cells (with same color or different

Several works about modular robots used group theory for various purposes as reconfiguration planning [14], [15] and non-isomorph assembly enumeration [16].
colors). Nevertheless, in a self-reconfiguration, the system can displace the modules only by rotation of $90^\circ$ around the revolute joints located at the centers of the cells. Such displacements may only displace modules between cells of same color. Therefore a self-reconfiguration cannot exchange modules on cells with different colors. Another example: the modules of Telecube[6] can have their orientation reconfigured by hand, while a self-reconfiguration cannot change their orientation. These examples illustrate the difference between reconfiguration and self-reconfiguration capacity.

In the latter these features will be characterized respectively by the **reconfiguration group** and the **manipulation group** of a system. These groups can be determined by their generating sets. The next section explains how to construct these generating sets for lattice systems.

### III. Generating sets of the reconfiguration group and the manipulation group

As a preliminary the terminology and symmetry operations concerning the connectors of lattice modules, introduced in [12], [13], are reminded.

#### A. Connectors symmetries

1) A connector without symmetry has no symmetry axis. It can be connected to a connector with opposite gender. Fig. 3(a) represents two compatible connectors with genders + and −.

2) A connector may have a “transverse axis”2 corresponding to a 2-fold rotation axis as represented Fig. 3(b). Such a connector is said “hermaphrodite” and can be connected to another one identical to itself (the two connectors in Fig. 3(b) are identical).

3) A connector may have a “normal symmetry” corresponding to a $n$-fold rotation axis as represented Fig. 3(c). The order $n$ of the rotation equals the number of orientations the connector can be connected to another one having an opposite gender. In the example Fig. 3(c) the normal symmetry axis of the connector (considered rectangular) is a 2-fold rotation axis, therefore the connector + can be connected with 2 different orientations to the connector − (if the connector is considered square the order of the rotation is 4 and it has 4 connecting orientations).

4) A connector may have both types of symmetries described previously: it has one normal symmetry axis of order $n$, and $n$ transverse symmetry axes of order 2. For instance, Fig. 3(d) represents a connector with one normal symmetry of order 2 and two transverse axis with order 2. Such a connector is hermaphrodite and can be connected to another one identical to itself.

Terminologically, we call configuration of a connector the set of points that it occupies in the space. Therefore several poses can correspond to the same configuration because of a normal symmetry axis of the connector. Two compatible connectors with opposite genders + and − are called “opposite connectors”. When two such connectors are connected (or have configurations such that they can be connected) we say that they have “opposite configurations”. Connectors which are compatible (hermaphrodite or with opposite gender) or identical (hermaphrodite or with same gender) are said of “same type”.

#### B. Geometrical features

We can now describe the geometrical features of the modules that will be used to construct the generating sets of the reconfiguration group and the manipulation group. There are three types of displacements that must be taken in account:

1) The first type of displacement corresponds to the **symmetry operations** of the connectors of a module as recalled section III-A. For instance, the Fig 4(a) represents a configuration of a module of the system M-Tran[10] which has 6 connectors. The symmetry operations of the connectors are the $90^\circ$
Rotation around their normal axes represented by dotted lines.

2) The second type of displacement will be called **inter-connectors** displacement. It can correspond to:

a) a displacement that brings a connector of a module to the same configuration as an another identical connector of the module. For instance, in Fig.4(b) the connector A and B are identical and the $90^\circ$ rotations around the dotted line brings the connector A to the same configuration as the connector B.

b) a displacement that brings a connector to a configuration such that it could be connected to an opposite connector of the module (the connectors reach opposite configurations). Such a displacement exists only for non-hermaphroditic connectors. For example, in Fig.4(c) the connectors B and C are opposite and the $180^\circ$ rotation around the dotted line move the connector B at a configuration opposite to the configuration of the connector C.

3) The third type of displacement will be called **inter-configurations** displacement. Is is a displacement of a connector of a module from one pose to a new pose when another connector of the same module is assumed immobile. Such displacements correspond to the relative displacements of connectors of a module due to its actuation mechanisms. If the joints of the module are independent (this is mostly the case), any motion produced by one joint corresponds to an inter-configuration displacements. For instance, Fig.4(d) shows the $90^\circ$ rotation axes of displacements produced by the joints M1 and M2. For redundant mechanism, parallel mechanism, singular configurations, or coupled joints[17], it is not possible to assimilate joints motions with connectors displacements.

C. Reconfiguration group

Consider one module $M$ having $n$ connectors $c_1, \ldots, c_n$ with a constant configuration $X = (x_1, \ldots, x_n)$. We denote by $\Omega \subset SE(3)$ the set containing all the symmetry operations of the connectors of the module, by $\Delta \subset SE(3)$ the set of inter-connectors displacement of the module, and by $\Gamma \subset SE(3)$ the set of the inter-configurations displacements from the configuration $X$ of the module to any other configuration $X' = (x'_1, \ldots, x'_n)$ such that at least one connector keeps the same pose: for any $X'$ there exists $i$ such that $x'_i = x_i$. Moreover, it is assumed (1) that the module has finitely many relative poses of its connectors, (2) that it has finitely many connectors and (3) that the number of symmetry operations for each connector is finite, so $\Gamma$, $\Delta$ and $\Omega$ are finite sets.

**Definition 1 (Reconfiguration group):** The **reconfiguration group** of the (homogeneous) modular system defined by the module $M$ is the group generated by the set $S = \Omega \cup \Delta \cup \Gamma$ with configuration $X = (x_1, \ldots, x_n)$. Moreover, if the group generated by $S$ is a discrete displacement group, we say that the modular system is *lattice*, otherwise it is not a lattice system.

**Definition 2 (Group type):** Two displacements groups $G \subset SE(3)$ and $G' \subset SE(3)$ are of same type if they are conjugate by a displacement $h \in SE(3)$: $g \in G$ iff $hgh^{-1} \in G'$.

**Theorem 1 (shall be demonstrated elsewhere [19]):**

The type of the reconfiguration group $G$ of the module $M$ does not depends on the configuration $X = (x_1, \ldots, x_n)$ of $M$.

For instance, for the module M-Tran[10] Fig.4, $S$ contains 4 symmetries for each 6 connectors, plus 5 inter-connectors displacement for each 6 connectors, plus 8 inter-configurations when the connectors are assumed immobile and 8 other inter-configurations when the connectors are assumed immobile, providing $24 + 30 + 16 = 70$ generators.

1) **Simplification:** By definition $S$ is a generating set of the reconfiguration group, but any subset of $S$ which generates the same group as $S$ may be used instead of $S$. Considering a module $M$ with a constant configuration $X$, a reduced generating set, denoted $\tilde{S}$ can consist of:

1) for each type of connector, the symmetry operations of only one connector (chosen arbitrarily) of this type;

2) for each type of connector, the inter-connectors displacements from only one connector (chosen arbitrarily) to the others of same type;

In crystallography a similar definition[18] considers an **isometry** instead of a displacement.
3) the inter-configurations displacements from the configuration \( X \) to the others, when only one connector, chosen arbitrarily, is assumed immobile. The validity of these simplification shall be demonstrated elsewhere [19]. For the module M-Tran[10] Fig.4, \( \bar{S} \) contains the 4 symmetries of 1 connector (there is one type of connector), 5 inter-connectors displacements from one connector (chosen arbitrarily) to the others, 8 inter-connectors displacements from one configuration to the others when a connector is immobile, providing 4+5+8 = 17 generators.

2) Further simplifications: If the joints of a module are not coupled (which is the case for most lattice system, an exception is [17]) a further simplification is to put in \( \bar{\Gamma} \) the sets of inter-configurations displacements produced by each joint separately, because the other inter-configurations displacements will result as combinations of the previous ones. For the module M-Tran this leads to consider only 3 displacements for one joint and 3 for the other.

We have seen that \( S \) is redundant because the smaller set \( \bar{S} \) generates the same group. Nevertheless, in most cases, the generating set \( \bar{S} \) his itself redundant because its displacements occur along common symmetry axes. Therefore it may be advantageous to represent the generating set \( \bar{S} \) geometrically by using symmetry axes. In the following example we will see that the generating set \( \bar{S} \) of M-Tran Fig.4 can be represented by four 4-fold rotation axes and one translation.

Fig. 5. The kinematics of a module M-Tran.

3) Example: The kinematics of the system M-Tran[10] is represented on the Fig.5. It has two revolute joints axes M1 and M2, and 6 connectors: three connectors C1, C2 and C3 with gender + and three opposite connectors C4, C5 and C6 with gender −.

The displacements of \( \bar{S} \) (see section III-C.1) are represented geometrically by symmetry axes:

1) Fig. 6(a) shows a 4-fold rotation axis corresponding to the symmetry operations of one connector.

2) Fig. 6(b) shows the symmetry axes corresponding to the inter-connectors displacements. The rotations around the 4-fold axis Z1 provides the displacements to bring the connector C2 to the same configurations as the identical connectors C1 and C3. Moreover, the translation along the axis T and the rotations around the 4-fold axis Z2 allows to bring the connector C2 on the opposite connectors C4, C5 and C6.

3) Fig. 6(c) shows the 4-fold rotation axes corresponding to the displacements generated by the joints M1 and M2.

The resulting set of axes is recapitulated Fig. 6(d).

4) Recognizing the generated reconfiguration group: The chiral space groups\(^4\) are all the discrete displacement groups containing translations. Therefore a modular system is lattice and has a translational periodicity in the space if and only if its reconfiguration group is a chiral space group. To find the corresponding space group, one must find the minimum group generated by \( \bar{S} \) in the tables [18], [20] or find the minimum group whose symmetry axes represented in [18], [21] match the geometrical representation of \( \bar{S} \) (as in Fig. 6(d)). If \( \bar{S} \) does not generate a chiral space group, the system is not lattice or has no periodicity in three directions (for example,

---

\(^4\)The chiral space groups (also called Sohncke groups) where introduced to describe the (chiral) crystals symmetry because they contains three independent translations allowing translational periodicity in the space. There exists 65 types of such groups described in [18] and listed in [12]. In two dimensions the discrete groups having two independent translations are called the plane groups. The 17 types of plane groups are described in [18]. Exhaustive data about the 230 space groups and their hierarchy can be found on line in [20], [21].
we consider the Caisson Fig. 35 in [17] as a lattice system with a periodicity in only one direction so its reconfiguration group is not a chiral space group). For the system M-Tran, \( G \) is the face centered space group F432. The Fig.7 represents the 4-fold rotation axes (Wyckoff position \( e \), see [18]) of the space group F432 and a module of the system M-Tran. The white spheres represent the position equivalent by translation. The 4-fold rotation axes of F432 coincide with the 4-fold symmetry axes of the revolute joints in any possible lattice configuration of the module. All the “equivalent” poses of a connector can be obtained by applying all the displacements of the group to one pose of this connector. On the figure, all the equivalent poses of the connectors + and − are represented respectively by the black and white squares (poses corresponding to the same connector configuration are indistinguishable because of its normal symmetry). In any lattice configuration of the system, the connectors of the modules will coincide with the equivalent poses of these connectors.

**D. Manipulation group**

The manipulation group is the minimum group containing the displacements of the lattice on which the module can move in any configuration of the module. Let \( D \) be the set (assumed finite) of displacements of a mechanism of a module \( M \) can produce when a connector \( c \) of \( M \) has a constant pose. Moving \( c \) (and consequently \( M \)) by a displacement \( d \in SE(3) \) brings \( c \) to a new constant pose. The new set of displacement produced by the mechanism is the previous one conjugated by \( d \):

\[
D' = dDd^{-1}
\]

The set \( D \) corresponds to the inter-configurations displacements \( \Gamma \) of the module, and the displacement \( d \) may be any element of the reconfiguration group \( \mathcal{R} \) generated by \( S = \Omega \cup \Delta \cup \Gamma \).

Hence, the manipulation group \( \mathcal{M} \) is the smallest group containing the inter-configurations displacements of the module conjugated by the elements of its reconfiguration group:

\[
\mathcal{M} = \langle gdg^{-1} | g \in \mathcal{R}, d \in \Gamma \rangle
\]

Or, more in detail:

\[
\mathcal{M} = \langle gdg^{-1} | g \in (\Omega \cup \Delta \cup \Gamma), d \in \Gamma \rangle
\]

For any element \( m \) in \( \mathcal{M} \), \( m = gd_1g^{-1}gd_2g^{-1}\ldots gd_ng^{-1} = gd_1d_2d_n g^{-1} = gdg^{-1} \) where \( d \) is an element of the group \( \langle \Gamma \rangle \) generated by \( \Gamma \). Therefore, \( \mathcal{M} \) is the normalizer of \( \langle \Gamma \rangle \) in \( \mathcal{R} \):

\[
\mathcal{M} = \langle gdg^{-1} | g \in \mathcal{R}, d \in \langle \Gamma \rangle \rangle = \text{Norm}_\mathcal{R} \langle \Gamma \rangle
\]

**E. Heterogeneous system**

In the presented method, the modular systems had only one type of module. To generalize to heterogeneous systems with \( n \) types of modules one must consider \( n \) modules connected together, this assembly forms a “meta-module”. The generating sets are determined on the meta-module in the same manner as for one module of an homogeneous system, however the connectors linking adjacent modules of the meta-module must be considered as the other connectors.

**IV. COMPARISON OF LATTICE SYSTEMS**

The previous principles where used to compare the reconfiguration group and manipulation group of several existing lattice systems. The results are represented in Table I. For each system the table give the types of the reconfiguration group and manipulation group in Hermann-Mauguin notation. For these systems, the groups are 2 or 3 dimensional chiral space groups, described in [18]. The groups in lowercase (like P432) are three dimensional space groups. The manipulation groups are necessarily subgroups of the reconfiguration groups, but may be equals (“The same” is displayed in the last column). If the manipulation group is not the same, it may nevertheless have the same type. This occurs for the system Micro-unit: in the corresponding row the scale difference is displayed. If the manipulation and reconfiguration groups have different types, then the reconfiguration group is a maximum subgroup of the manipulation group (the contrary does not occur for the existing lattice systems). For the system Stochastic[8] there is no manipulation group, because the system uses only external devices to be reconfigured.

The symmetries of the connector are also displayed. To note the symmetry of a connector (see [12], [13]) we use two digits: the first digit gives the order of the normal symmetry, while the second gives the order of the transverse symmetry (which is 1 or 2).
non hermaphrodite connectors, the transverse symmetry order equals 1, and it is possible to distinguish their gender by replacing the digit 1 by + or −. To locate the symmetry axes of the connectors in relation with the reconfiguration group (not the manipulation group) the table gives the Wyckoff positions (see [18], [12], [13] of these axes. Likewise, the table locates the joints axes in relation with the reconfiguration group by giving their corresponding Wyckoff positions. For the heterogeneous system I-Cube, the Wyckoff positions of the axes are given separately for each modules, while for the system Molecule V2 both module are described in the same row (only the gender of the connectors changes).

When the reconfiguration and manipulation groups are different this means that some identical connectors of modules cannot have their configurations exchanged by self-reconfiguration (while this is possible by reconfiguration). For example, the system Atron[9] has its modules equipped with two pair of connectors + and connectors −, but it cannot exchange the configurations of two identical connectors of a module by self-reconfiguration. Another example is the system Fracta[3], for which it is not possible to exchange the configurations of two adjacent modules by self-reconfiguration.

V. Concluding Remarks

The introduced method allowed to characterize successfully the existing lattice systems by discrete displacement groups. This prove that our definition of lattice systems is consistent with the former one, and consequently that the conception approach introduced in [12], [13] is appropriate for the design of lattice systems. Moreover, this characterization helps to estimate the kinematical features of the lattice systems by giving the sets of possible displacement occurring on the module during reconfigurations or self-reconfigurations. For example if the manipulation group is the identity, the system cannot self-reconfigure. If the manipulation group is two dimensional, while the reconfiguration group has three dimensions, the system can only move its modules along a plane. These results can be used as a complement for the conception of lattice systems because it allows to discriminate between candidate solutions.

Furthermore, our method can be used to analyse the effect of some modifications of lattice systems. For example, if we put hermaphrodite connectors for the modules of M-Tran[10] we will find that the new system has the same reconfiguration and self-reconfiguration groups than the previous one. If we alternate passive and active modules for the system Fracta[3] we will find that the new system keeps the same self-reconfiguration group. The Table I shows that nearly all 3D systems correspond to groups of type P432 and F432 which have respectively a cubic and cubic face centered lattice. These groups are on the top of the chiral space groups hierarchy. The reason to this is that only these groups have orthogonal 4-fold rotation axes, and most lattice systems have orthogonal 90° rotations in the generating sets of their manipulation or reconfiguration groups. Nevertheless, it is surprising that no lattice system corresponds to the group of type I432 with a cubic centered lattice.

### Table I

**Comparison of the kinematical features of some lattice systems.**

<table>
<thead>
<tr>
<th>System</th>
<th>Reconfiguration group</th>
<th>Connectors</th>
<th>Joints</th>
<th>Manipulation group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Symmetry</td>
<td>Wyckoff Position</td>
<td>Type and number</td>
</tr>
<tr>
<td>Atron</td>
<td>P432</td>
<td>11</td>
<td>k (general position)</td>
<td>4-fold revolute joint: 1</td>
</tr>
<tr>
<td>Crystalline</td>
<td>p1 (2D)</td>
<td>11</td>
<td>a</td>
<td>Prismatic joints: 4</td>
</tr>
<tr>
<td>Fracta 3D</td>
<td>P432</td>
<td>22</td>
<td>d</td>
<td>4-fold revolute joints: 6</td>
</tr>
<tr>
<td>I-Cube</td>
<td>P432</td>
<td>Cube: 4-</td>
<td>e</td>
<td>Square: none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Link: 4+</td>
<td></td>
<td>Link: 4-fold revolute joint: 3</td>
</tr>
<tr>
<td>Micro Unit</td>
<td>p4 (2D)</td>
<td>11</td>
<td>d</td>
<td>4-fold revolute joints: 2</td>
</tr>
<tr>
<td>Molecule</td>
<td>P432</td>
<td>42</td>
<td>c</td>
<td>3-fold rotation joints: 1</td>
</tr>
<tr>
<td>Molecule v1</td>
<td>P432</td>
<td>42</td>
<td>d</td>
<td>4-fold rotation joints: 4</td>
</tr>
<tr>
<td>Molecule v2</td>
<td>F432</td>
<td>4+, 4-</td>
<td>e</td>
<td>4-fold rotation joints: 4</td>
</tr>
<tr>
<td>M-trans</td>
<td>F432</td>
<td>4+, 4-</td>
<td>e</td>
<td>4-fold rotation joints: 2</td>
</tr>
<tr>
<td>Stochastic</td>
<td>P432</td>
<td>42</td>
<td>e</td>
<td>None</td>
</tr>
<tr>
<td>SuperBot</td>
<td>P432</td>
<td>42</td>
<td>e</td>
<td>4-fold rotation joints: 3</td>
</tr>
<tr>
<td>Telecube</td>
<td>P23</td>
<td>22</td>
<td>f</td>
<td>Prismatic joints: 6</td>
</tr>
</tbody>
</table>

ICRA 2010 Workshop "Modular Robots: State of the Art"
References


Novel modular CPG topologies for modular robotics

Fernando Herrero-Carrón, Francisco de Borja Rodríguez, Pablo Varona

Abstract—We present our work on novel neural controllers for a modular, worm-like robot. We first develop a model of neural oscillator based on a neuron model with multiple time scales and a kinematic synapse model. We then study different connectivity mechanisms that allow individual modules to negotiate the overall rhythm based only on first-neighbors communications. All of the studied topologies are capable of self-organization and they sustain a stable rhythm.

I. INTRODUCTION

Synchronization is, in our opinion, one of the most fundamental problems in modular robotics. In most cases, modular robots are built of units (modules) with only one degree of freedom. It is perhaps surprising that this simplicity does in no way limit the power of modular robots. But this power and flexibility must be under control. The different modules of the robot must be coordinated to carry out any useful activity. The last years have witnessed an increasing focus of attention towards living neural circuits to solve this problem. In particular, Central Pattern Generators have enjoyed widespread acceptance and use in the community [1].

CPGs are known for their ability to robustly generate a rhythmic sequence of motion commands, while still being very flexible and adaptable to the requirements of their host [2], [3]. In this paper we introduce some CPGs designed to control a modular worm robot by González et al. [4]. This platform is very powerful, in terms of locomotion capabilities, while still being very accessible and easy to control. For simplicity, we have focused on horizontal ground displacement, one of the many locomotion modes this robot is capable of.

The robot, illustrated diagramatically in Fig. 1, consists of several modules attached side by side through special connection points. Each of these modules consists of two triangle shaped rigid pieces, joint by one vertex of the triangle, and a servomotor controlling the angle between these two pieces. In horizontal locomotion mode, modules are connected sequentially, each of them oscillating on the same plane. One solution to the control problem here posed is undulating locomotion. Each module must oscillate periodically, at a given phase lag from the neighboring ones. Thus, the CPG must solve the problem of individual oscillation and global coordination.

The choice of this platform has been motivated by its versatility (the reader is again referred to [4]), low cost and ease of construction. The chassis is built of methacrylate panels, assembled by hand in less than one hour. The servos are futaba s3003, readily available in any RC store and with an approximate cost of $15 apiece. Finally, there being no wheels, limbs or any other movable parts besides the servos, control of the robot is exclusively a problem of synchronization among modules, a problem that CPG control will solve in a robust and flexible manner.

II. MATHEMATICAL MODELS

In this paper we explore a neuron model that displays multiple time-scales, that is, its output is a slow wave with fast spikes sitting on top of each wave. This regime is called bursting dynamics, and may be seen in living neurons [3]. We have used a neuron model that is mathematically very simple, yet very rich in its possible repertoire of behaviors [5].

1) Neuron model: The mathematical description of Rulkov’s model as used in this work is as follows:

\[
\begin{align*}
  f(x, y) &= \begin{cases} 
    \frac{\alpha}{1-x} + y, & \text{if } x \leq 0 \\
    \alpha + y, & \text{if } 0 \leq x < \alpha + y \\
    -1, & \text{otherwise}
  \end{cases} \\
  x_{n+1} &= f(x_n, y_n) \\
  y_{n+1} &= y_n - \mu(x_n + 1) + \mu \sigma + \mu I_n
\end{align*}
\]

with \( \mu = 0.001 \) in all experiments.

This is a bi-dimensional model, where variable \( x_n \) represents a neuron’s membrane voltage and \( y_n \) is a slow dynamics variable with no direct biological meaning, but with similar meaning as gating variables in biological models that represent the fraction of open ion-channels in the cell. While \( x_n \) oscillates on a fast time scale, representing individual
spikes of the neuron, \( y_n \) keeps track of the bursting cycle, a sort of context memory. Units are dimensionless, that is, one can rescale them to match the requirements of the robot. The combination of \( \sigma \) and \( \alpha \) selects the working regime of the model: silent, tonic spiking or tonic bursting.

2) Synapse model: Coupling is now performed using a dynamical model which acts as a filter of the fast spikes and enables synchronization between bursts of the two neurons. The model used is by Destexhe et al. [6] and its equations are as follows:

\[
\dot{r} = \begin{cases} 
\lambda[T](1 - r) - \beta r, & \text{if } t_f < t < t_f + t_r \\ 
-\beta r, & \text{otherwise}
\end{cases}
\]  

(4)

This equation defines the ratio of bound chemical receptors in the post-synaptic neuron, where \( r \) is the fraction of bound receptors, \( \lambda \) and \( \beta \) are the forward and backward rate constants for transmitter binding and \( [T] \) is neurotransmitter concentration. The equation is defined piecewise, depending on the specific times when the presynaptic neuron fires \( (t_f) \): during \( t_r \) units of time, the synapse is considered to be releasing neurotransmitters that bind to the post-synaptic neuron. After the release period, no more neurotransmitter is released and the only active process is that of unbinding, as described by the second part of the equation. Times \( t_f \) are determined as the times when the presynaptic neuron’s membrane potential crosses a given threshold \( \theta \).

Synaptic current is then calculated as follows:

\[
I(t) = g \cdot r(t) \cdot (X_{post}(t) - E_{syn})
\]

(5)

where \( I(t) \) is post-synaptic current at time \( t \), \( g \) is synaptic conductance, \( r(t) \) is the fraction of bound receptors at time \( t \), \( X_{post}(t) \) is the post-synaptic neuron’s membrane potential and \( E_{syn} \) its reversal potential, the potential at which the net ionic flow through the membrane is zero. When coupling two Rulkov map neurons we will need to use a discrete synaptic function. We will build a sequence, let us call it \( I_m \), by simulating \( I(t) \) as a continuous function and then taking samples every 0.01 time units (for our choice of kinetic parameters as outlined in the different figures).

3) Output stage: Movement information is robustly encoded in the neurons’ bursting episodes. A neuron called motoneuron is then responsible of decoding this information and generating the signal that will finally be sent to the servo controller. This signal tells the angle at which the servo should be positioned, in degrees. See Fig. 3 for an example pattern of activity of an oscillator in its steady state.

The synapses that connect R and P neurons to the motoneurons are governed by a very simple threshold equation:

\[
s(x, \nu) = \begin{cases} 
1, & \text{if } x > \nu \\ 
0, & \text{otherwise}
\end{cases}
\]

(6)

This function is used to detect individual spikes of neurons. By setting the threshold to, for example, \( \nu = -1.5 \text{ a.u.} \), this function applied to the potential trace of one neuron will have value 1 during individual spikes and 0 otherwise.

The output of the motoneuron M is calculated from its input (signals generated by R and P filtered through (6)) following this equation:

\[
\tau \dot{m}(t) = -m(t) + \gamma s(P_z(t), \nu) - \gamma s(R_z(t), \nu)
\]

(7)

where, \( m(t) \) is the output of neuron M (in degrees), the \( s(\ldots) \) terms are the threshold function (6) applied to input from R and P, and \( \tau \) is a time constant that controls how quick the output signal \( m(t) \) will change, and is therefore a scaling factor of the amplitude of the generated signal.

In this equation, if \( \gamma \) is positive, P will contribute positively and R negatively. Given the fact that P and R oscillate in anti-phase, the solution \( m(t) \) is an oscillatory function bounded between \( -\gamma \) and \( \gamma \). When the motoneuron receives no input because P and R are silent, it will go back to zero due to the leak term \( (-m) \) in (7) (see decay of \( m(t) \) between bursts in Fig. 3).

III. FIRST NEIGHBORS TOPOLOGIES

We have studied different mechanisms for coordination between modules, based on different assumptions about what the governing principle should be. Our goal is to find what invariant must be implemented in our CPGs for it to generate an effective locomotion on our robot. In this section we discuss the motivation behind each approach and the resulting architecture.

A. Asymmetric Inhibition

This is our most basic assumption, and the least restrictive. We want neighboring modules to inhibit each other. This will prevent them from oscillating in-phase, which would never generate an effective undulating locomotion. However, preventing in-phase synchronization does not guarantee it either.

The idea is shown in Fig. 4. Two inhibitory pathways are established: ascending and descending. If both pathways have similar strength (synaptic conductance and other factors), the system does not generally converge to a stable state. However, if one pathway is dominant over the other, the
B. Phase Enforcement

The previous was a very general restriction. We now want to study if we can find a way to enforce a specific phase difference between modules. Fig. 5 shows two different mechanisms we have devised.

The first one is what we have called the inhibitory loop. It takes advantage of the dynamics of synapses in order to create a temporal delay in the interaction between adjacent modules. Let’s take only one P neuron and its “side” neuron (the one it excites and receives inhibition from). The P neuron excites the “side” neuron when it (P) bursts. Under excitation, the “side” neuron enters a tonic spiking regime, which activates the inhibitory synapse to P. This inhibition is weak, so it does not completely inhibit P, but it reduces its frequency. The “side” neuron will remain in the tonic spiking regime for some time, even past P’s burst, until it eventually returns to the silent regime. At a steady state, P’s frequency under “side”’s influence is lower than its nominal frequency.

We now introduce one neighbor P neuron, let’s call it P’, to which P excites and who, in turn, inhibits “side”. P’ at its nominal frequency oscillates faster than P with “side”’s influence. If P’ receives excitation it will oscillate even faster. Furthermore, at equal frequency of oscillation, excitation will tend to synchronize both P and P’ in phase.

Let’s see how this subsystem with two promotor neurons and one side neuron works. P’ is oscillating faster than P, so it will slowly reduce its phase difference with the latter. By reducing phase difference it inhibits “side” earlier every cycle, so its spiking episodes become shorter. With less inhibition in each cycle, P begins to oscillate faster.

There is a given phase difference between P and P’ that is a stable state. That means that both P and P’ oscillate at the same frequency, which seems surprising given that P only receives inhibition. The fact is that at this point, inhibition from side only affects burst length, making it shorter, but does not affect P’s recovery process, thus effectively making frequency higher than its nominal frequency. Here, “side” acts as a phase detector.

We will now examine the push-pull architecture. In this model one excitatory and one inhibitory synapse project from one module to one of its neighbors. Total synaptic input from one module to its neighbor depends on phase difference. Both excitatory and inhibitory synapses have the same time constants, the only difference being their reversal potential. Recalling equation (5), the combined result of both will depend on whether the post-synaptic neuron’s potential is closer to the excitatory synapse’s $E_{syn}$ or to the inhibitory’s. That is, if the post-synaptic neuron is hyper-polarized (at a very negative value) when synapses are active, the end result will be inhibitory (membrane potential will be further from the inhibitory synapse’s reversal potential and will contribute more than the excitatory’s). On the other hand, if it is depolarized (bursting or spiking), its membrane potential will be closer to the inhibitory’s reversal potential, thus excitation will contribute more.

C. Winnerless Competition

We are interested in implementing a particular kind of network dynamics named Winnerless competition [7]. In this type of dynamics, all neurons compete with each other through inhibition. When one neuron is active, it will inhibit some other neurons, preventing them from activating as well. The key point is that there must be a mechanism by which this inhibition is released. When this happens, the previously inhibited neurons are allowed to become active, inhibiting other neurons in turn. With this release mechanism, it is ensured that no single neuron will inhibit all other neurons permanently, hence the term “winnerless”
competition. Beyond this, we seek a mechanism to implement a winnerless competition in which the sequence of activation is reproducible, in order for the robot to undulate properly.

In summary, three principles are required for generic winnerless competition dynamics: non-open topologies, asymmetric inhibition and a mechanism by which inhibition is released, guaranteeing that no neuron will be permanently inhibited. Together with these, we consider that the topology of the CPG must be modular, as that of the robot, and that the sequence of activation must be reproducible. The CPG is shown in Fig. 6.

We have added two bistable neurons (see (9) through (11)), which are a modified Rulkov map (see (1) through (3)) to every module. Their role is to inhibit the promotor and the remotor neurons respectively. They can be either silent or in a tonic spiking regime: when they are excited they switch to a silent regime. Under absence of input, they will remain in the same state they were. ‘B’ neurons effectively restrict the allowed intervals through a strong synapse. When a ‘B’ neuron is in its active state, it will completely inhibit one ‘P’ (respectively ‘R’) neuron until it goes back to the silent regime.

to the desired winnerless competition.

Border neurons would only receive signals from one side, not from both. If they were left to burst freely, they would do so at a higher frequency than the rest, since they would be receiving no inhibition. Thus the need for a non-open topology emerges naturally. That is, neuron borders need to receive some feedback from the rest of the CPG. Adding a “border synapse” regularizes the CPG and a stable rhythm may be achieved.

IV. CONCLUSION

We present a new oscillator model, based on bursting neurons and kinetic synapses, and propose different connectivity models for inter-module coordination. Our experimental results show that a robot under control of these CPGs is capable of effective locomotion. In future work we will study how this models can be augmented to achieve locomotion in two planes rather than only one.

REFERENCES

Self-Disassembling Robots Pebbles: New Results and Ideas for Self-Assembly of 3D Structures

Kyle Gilpin and Daniela Rus

I. INTRODUCTION

We present our newest algorithms, results, and future plans for the robotic pebble system shown in Figure 1 which is capable of forming shapes through uniform self-assembly followed by selective self-disassembly. In general, programmable matter systems are composed of small, intelligent modules able to form a variety of macroscale objects in response to external commands or stimuli. Our system is composed of 12mm cubic autonomous robotic pebbles, (first presented in [1]), capable of bonding and communicating with their neighbors. Starting from a loose collection of disjoint modules, we hope to show that our system, with the assistance of external stochastic forces, is capable of self-assembling into a uniform crystalline structure. Once this initial block of material is formed, the system is able to self-disassemble to form complex 2D shapes. Like geologic forces compact sediment into blocks of sandstone and a sculptor removes the extra stone to reveal a statue underneath, our system forms an initial uniform grid of modules and then subtracts the unnecessary modules to form a goal structure.

A. System Functionality

We aim to create a system of sand grain sized modules that can form arbitrary structures on demand. Imagine a bag of these intelligent particles. If, for example, one needs a specific type or size of wrench, one communicates this to the bag. The modules contained within first crystallize into a regular structure and then self-disassemble in an organized fashion to form the requested object. One reaches in, grabs the tool, and uses it to accomplish a meaningful task. When one is done with the tool, it goes back into the bag where it disintegrates, and the particles can be reused to form the next tool. Such a system would be immensely useful for an astronaut on an inter-planetary mission or a scientist isolated at the South Pole. Even for the average mechanic or surgeon, the ability to form arbitrary, task-specific, tools would be immensely valuable in inspecting and working in tight spaces.

B. Advantages of Self-Assembly/Disassembly

Designing an electromagnetic module capable of exerting the force necessary to attract or repel other modules from a distance greater than the size of a module has proven challenging. Shape formation with electrostatic or magnetic modules is more feasible when driven by stochastic forces, so that the actuators only need to operate over short distances.

Traditional self-assembling systems aim to form complex shapes in a direct manner. As these structures grow from a single module, new modules are only allowed to attach to the structure in specific locations. By carefully controlling these locations and waiting for a sufficiently lengthy period of time, the desired structure grows in an organic manner. In contrast, our system greatly simplifies the assembly process by initially aiming to form a regular crystalline block of fully connected modules. We make only limited attempts to restrict which modules or faces are allowed to bond with the growing structure. These restrictions are only to ensure that we achieve a regular structure. As illustrated in Figure 2, after we form this initial block of material, we complete the shape formation process through self-disassembly and subtraction of the unwanted modules.

Subtraction has one distinct advantage over existing self-assembly techniques. Subtraction does not rely on complicated attachment mechanisms that require precise alignment or careful planning. Subtraction excels at shape formation because it is relatively easy, quick, and robust. The drawback associated with subtraction is that the initial mass of material must be pre-assembled. While we do this by hand for our experiments, it could be automated. Our modules, due to symmetry in their magnet-endowed faces, are rotation...
of freedom that are able to modify their topology in some way. There are also hybrid systems [10]-[13] in which neighboring modules join to accomplish relative actuation.

Other research has focused more directly on the concept of programmable matter. One particular system [14], uses rigid cylindrical modules covered with electromagnets to achieve 2D shape formation. Theoretical research has previously investigated the use of sub-millimeter intelligent particles as 3D sensing and replication devices [15]. More recent developments are utilizing deformable modules [16] as a way to realize programmable matter. Finally, the system described in [17] has no actuation ability, but demonstrates what may be termed ‘virtual’ programmable matter through the use of 1000 distributed modules to form an intelligent paintable display capable of forming text and images.

A limited amount of past research has focused specifically on self-disassembling systems as a basis for shape formation [18]. This past work was based on large modules (45cm cubes) with internal moving parts. Additionally, the modules lacked symmetry so they had to be assembled in a particular orientation. The work presented in this paper is an outgrowth of the Miche system presented in [18], but we have reduced the module size, eliminated all moving parts, and added symmetry to allow for arbitrary module orientations. Finally, the system presented here shows promise as both a self-disassembling and self-assembling system.

**II. Hardware Summary**

Figure 1 shows a collection of identical programmable matter pebbles and the components that comprise one module. The modules are 12mm cubes capable of autonomously communicating with and latching to four neighboring cubes using electopermanent (EP) magnets. Each module is formed by wrapping a flexible circuit around a brass frame. All electronic components are mounted to the inside of the flex circuit. The four EP magnets are able to draw in other modules from a distance, mechanically hold modules together against outside forces (with zero power dissipation), communicate data between modules, and transport power from module to module. The EP magnets are soldered directly to the flex circuits so that their pole pieces protrude slightly through four sets of holes in the cube faces. A capacitor mounted inside each module provides local energy storage eliminating the need for batteries in each module. Instead, DC voltage and current are injected into the system at one root module and transferred from module-to-module by the electrically isolated poles of the EP magnets. For more information about the system hardware, consult [1].

**III. Self-Assembly**

As mentioned above, we wish to demonstrate that our system of robot pebbles can form arbitrary shapes through a two-step process. First, we want a loose collection of modules to self-assemble into a regular crystalline structure. Once this initial block of material is formed, we wish to sculpt it into an arbitrary shape using self-disassembly.

![Figure 2](image_url)

**Fig. 2.** To form shapes through subtraction, modules initially assemble into a regular block of material (a). Once this initial structure is complete and all modules are fully latched to their neighbors (b), the modules not needed in the final structure detach from the neighbors (c). Once, these extra modules are removed, we are left with the final shape (d).

---

ICRA 2010 Workshop "Modular Robots: State of the Art"
Previous work [18] has successfully demonstrated the self-disassembly process. This work aims to explore the self-assembly step by using external stochastic forces to drive the modules around small platform until they bond into a uniform structure.

A. Hardware

As shown in Figure 3, we built a simple vibration table to test the ability of the pebbles to self-assemble in the presence of stochastic environmental forces. The table was built from an inexpensive back massager. We anchored the massager to a heavy aluminum plate and mounted an acrylic assembly platform to the top of the massager. The pebbles are placed on the top of this platform and allowed to move freely. The acrylic tray has edges which keep the pebbles from falling of the platform. Additionally, there is a universal joint between the massager and the assembly platform which allows us to arbitrarily incline the platform.

Fig. 3. To aid in self-assembling, we built a simple vibration table from a $100 back massager (a) and a Panavise universal joint (b) which allows us to control the angle of the assembly table (c). A variac (d) allows us to control the speed of vibration.

For self-assembly to occur, the pebbles must be provided with power. The acrylic assembly platform contains three locations where a single root pebble may be anchored: the center of the platform, the middle of an edge, and a corner. Below these three anchor point there are gold-plated pogopins which transfer power and communication signals to matching pads on the bottom of the pebble. When self-assembling, the module at this anchor point activates its EP magnets to help attract unpowered modules as they pass by. When an unpowered, unattached module comes close enough to the root to be drawn in, the module will bond with the root and immediately receive power. Once it has power, it communicates with the root module and activates its EP magnet to significantly strengthen their mechanical bond. It also activates its three other magnets to attract additional free pebbles that are circulating on the platform. This process repeats until all free modules have been attached to the growing structure.

B. Algorithms

During the self-assembly process, we want to ensure that no gaps are formed in the growing structure. By preventing gaps, the self-disassembly process is able to form the widest variety of shapes. Additionally, gaps weaken the structure and reduce the available communication paths. If we allow new modules to be accreted at any location on the growing structure, it is easy to create concavities in the structure that are theoretically difficult and practically impossible to fill. For example, a loose module will never fill a spot in the crystalline lattice that is already surrounded on three sides.

To avoid holes in the self-assembled structure, we propose a simple distributed algorithm that only requires local information. Based on this information, each free module coming into contact with a potential bonding site on the solidified structure must decide whether to permanently bond with the structure or move on and look for another bonding site.

The algorithm makes two assumptions. First, all modules know the location of the root module. This is easy to hard-code into each module's process as location (0,0), for example. Second, once each module is added to the structure, it can determine it's (x,y) position. This requirement is also easy to meet. The user informs the one module anchored to the assembly platform that it is the root and therefore at location (0,0) and that it is rotated 0°. Using this information, the root can inform the module added to its right that the new module’s location is (1,0). Likewise, the module added below the root is at location (0,-1), etc. Based on which of its faces the new module receives this message, it can determine its orientation. Now that the newest module knows its location and orientation, it inform its newest neighbors of their location. More details, and a proof that this algorithm is correct are proved in [18].

The algorithm, begins as the free module receives power when it comes into contact with a module already a part of the crystallized structure. Immediately, the module queries its neighbor to determine its location. Based on this location, the module then constructs a root vector pointing back to the root module. The vector may have x- and y-components. The new module permanently bonds with the structure if it detects that it has neighbors in both the x- and y-directions of the root vector, if they exist. For example, consider a new module that determines its location is (10,2). As shown in Figure 4, the root vector is then (-10,-2) which has both x- and y-components. As a result, the module only bonds with the structure if it has neighbors at (9,2) and (10,1). Instead, if the new module were located at (0,-5) and the root vector was (0,5), the module in question would only bond if it detected a neighbor at (0,4).

If the new module does not detect neighbors in the appropriate locations, it informs whatever neighbors it is contacting, and they deactivate their connectors allowing the pebble to continue moving under the influence the table’s vibrations. The already solidified module keeps this connector deactivated for a fixed period of time to allow the rejected module to move out of range of its attractive
During self-assembly, modules only permanently attach to the already assembled structure if they detect immediate neighbors along a vector that points back to the root module. The module at (2,1) does not attach because, while it has a neighbor along the y-component of its root vector at (2,0), it does not detect a neighbor at (1,1) along the x-component of the vector. The module at (0,-2) does attach to the crystallized structure because it detects a neighbor at (0,-1), along the y-component of its root vector. The root vector does not have an x-component, so the module does not attempt to detect neighbors at (-1,-2) or (1,-2).

Eventually, the connector is reactived in hopes that the bonding site will have become valid.

C. Experiments

In preliminary experiments, the vibration table works well to align the pebbles into a grid pattern. When the assembly table is tilted about 10° along its diagonal, a set of 15 pebbles forms a lattice within 15–20 seconds. As expected, the resulting shape is often concave. To test our self-assembly algorithm, we plan to compare it to the naive algorithm which bonds two neighboring modules wherever they come into contact. We will perform these experiments in the following weeks leading up to the workshop.

IV. EXTENSIONS TO 3D

The current generation of robot pebbles is only able to operate in the plane. Furthermore, the pebbles cannot be flipped upside down. If they are, the EP magnets, when activated, change from attracting to repelling. To expand the number of practical applications of the system, it needs to be able to operate in three dimensions. We see three different approaches to achieving a 3D programmable matter systems.

A. Rotation Invariant Pebbles

The first option is the obvious solution: place electropermanant magnets on all six faces of the pebbles making them invariant to any 90° rotation. This solution provides the greatest flexibility and highest degree of redundancy when assembling the modules into a 3D structure.

The six-connector solution is not without drawbacks. The flex circuits in the current version of the pebbles are already severely space limited. By adding two additional EP magnets, we would eliminate the area currently dedicated to the processor and power conditioning circuitry. (The additional EP magnet would also require additional drivers further increasing the component density.) The EP magnets are large components with respect to the size of the flex circuit and must be placed in the center of each face. As a result, they subdivide the remaining flex circuit area into many small parcels that are difficult to utilize for components other than surface mount resistors and capacitors. This awkward division of flex circuit area would make it difficult to utilize an ASIC that combined all of the circuitry into one IC. One way to avoid this problem may be to modify the design of the flex circuit to create an additional “floating tab” that occupies the middle of the cube and is large enough to contain the ASIC.

The second problem with placing EP magnet connectors on all six faces is that the connectors would need to be redesigned. Currently, the connectors are only 2-way symmetric, but they would need to be 8-way or axially symmetric in the 3D system. Figure 5 shows a cross-section of one possible design of an axially symmetric EP magnet.

B. Out of Plane Connectors

An alternative to employing six active faces in each pebble is to create two or three distinct types of pebbles, each capable of bonding with neighbors in separate planes as shown in Figure 6 One can think of this strategy as forming a structure as a stack of unbonded layers and then bonding the neighboring layers together with special “out of plane” pebbles. We could continue using our current set of pebbles for bonding in the X-Y plane, but we would then design two new types of pebble (still with just four connectors) capable of bonding in the X-Z and Y-Z planes. Starting with a sheet of X-Y type pebbles, we could replace some of the modules with X-Z and Y-Z modules. On top of each of these new pebbles we would place another X-Z or Y-Z module, respectively. Then, the remainder of the second layer could be filled with the standard X-Y pebbles.

Fig. 4. During self-assembly, modules only permanently attach to the already assembled structure if they detect immediate neighbors along a vector that points back to the root module. The module at (2,1) does not attach because, while it has a neighbor along the y-component of its root vector at (2,0), it does not detect a neighbor at (1,1) along the x-component of the vector. The module at (0,-2) does attach to the crystallized structure because it detects a neighbor at (0,-1), along the y-component of its root vector. The root vector does not have an x-component, so the module does not attempt to detect neighbors at (-1,-2) or (1,-2).

Fig. 5. An axially symmetric electropermanent magnet could be created by placing two half round magnets (a,b) next to each other to form a core than is then wound with a coil (c) and placed inside of a ferromagnetic cup (d). A small cap (e) is attached to the exposed end of the magnetic core to prevent fringing fields from giving rise to attractive forces when the EP magnet is deactivated.
For a large structure containing an equal proportion of all three types of modules, there will, on average, be one third of neighboring faces which are not connected. In comparison to a system in which there are EP magnets on all six faces, this will weaken the structure and limit the communication pathways through it. As with the rotation invariant system, the connectors will need to include additional degrees of symmetry because stochastic forces will ensure that the modules touch in every possible orientation.

C. Active and Passive Faces

The third option for creating a 3D system is to use a combination of active and passive connectors like the configure in the Miche system [18]. Three connectors could be ferromagnetic plates and the other three could be EP magnets. This setup alleviates the space constraints inside the pebbles, but forces the all pebbles in a structure to be oriented identically. If their orientations are not homogenized, there will be locations were two passive connectors are adjacent. If we ever hope for the system to self-assemble in the presence in stochastic forces, the active/passive connection mechanism is untenable.

V. Future Directions

The self-assembly algorithm presented here is completely distributed and only relies on information from a module’s closest neighbors. As a result, the assembly process can happen in parallel over the whole perimeter of the growing structure—an important feature when assembling structures from hundreds or thousands of modules. The algorithm also extends to 3D systems. Instead of checking just the x- and y-components of the root vector for neighbors, the module attempting to attach to the structure will also need to check for a neighbor in the z-direction.

One aspect of the algorithm that needs optimization is the time delay between when a module rejects a bonding site and when that bonding site is reactivated. This time is likely dependent on the number of free modules in the system and the severity of the stochastic forces acting on it. We suspect the delay could be shortened if the EP magnets had the ability to repel, in addition to attract. Such functionality is with reach, but requires additional mechanical and electrical complexity.

Given that we know how to form an initial structure devoid of holes, future research should also focus on how to explicitly form holes in the structure. Without holes in the initial structure, the final shape formed after the self-disassembly process will also not have holes. By purposefully including holes in the self-assembly process, the completed shapes can also have holes.

We hope that by uniting the ideas for self-assembly and 3D shape formation presented here that we can move the robotic pebbles one step closer to smart sand. While there is still work required to further miniaturize the pebbles, the current system is a useful testbed that allows us to quickly test the concepts and algorithms that drive the self-assembly and self-disassembly processes.

VI. ACKNOWLEDGMENTS

This work is supported by the DARPA Programmable Matter program and the US Army Research Office under grant number W911NF-08-1-0228, NSF EFRI, Intel, and the NDSEG fellowship program.

REFERENCES


Reconfigurable Software for Reconfigurable Modular Robots

David Ko
Graduate Student Researcher
Integration Engineering Laboratory
Department of Mechanical and Aerospace Engineering
University of California, Davis

Harry. H. Cheng∗, Senior Member, IEEE
Professor and Director
Integration Engineering Laboratory
Department of Mechanical and Aerospace Engineering
University of California, Davis
hhcheng@ucdavis.edu

Graham G. Ryland, Member, IEEE
Barobo, Inc.

Abstract

Highly reconfigurable modular robots face unique control and programming challenges due to the large scale of the robotic systems, high level of reconfigurability of the systems, and high number of controllable degrees of freedom in the system. Modular robot systems such as iMobot which are designed with high degrees of freedom for reconfigurability must face these challenges in novel ways. Reconfigurable software systems are effective solutions for control and programming of reconfigurable modular robots. C based interpretable computing environments such as Ch provide high level scripting capabilities as well as hardware interface required for programming mechatronic systems. C based embeddable interpreters and mobile agent systems can provide a base of reconfigurable software for the control and cooperation of modular robots. Reconfigurable software systems may also be used for high level cooperative computational tasks performed on the robot modules. They provide an effective, straightforward, and familiar way to easily program and configure modular robotic systems.

1. Introduction

Modular robotics has received a great deal of research interest in recent years. Much of their potential is due to the fact that they are highly flexible due to their reconfigurability. The same set of robots may be reconfigured to perform a variety of different tasks, and each configuration may have unique locomotive possibilities available to it. They are also more fault tolerant due to the usage of identical modules. A modular robot system may be able to shed damaged modules and replace them with identical ones. Modular robots also promise to be cost effective because each module is identical, thus simplifying the manufacturing process. There are many possible application areas for a robust modular robotic system, including first response applications [1] and space exploration [2].

Various designs of modular robots have been developed in recent years. The wide range of designs include lattice type robots such as ATRON [3, 4] and Telecubes [5], to chain type robots such as MTRAN [6], Superbot [7], and iMobot [8, 9, 10].

Among the chain type modular robots, the majority of the systems possess only one or two controllable degrees of freedom per module. Furthermore, none of the degrees of freedom in these modules are capable of free rotation. This limits the possible locomotive options for these modules. For instance, a single module of these types of robots is only capable of traversing along a line, unable to turn. A relatively large number of modules is needed for the robot to be able to perform a rolling locomotion as shown in [11], where the modules form a loop. However, should the loop topple over, it is difficult or impossible for the modules to reposition themselves into a standing loop.

The iMobot module has four controllable degrees of freedom. The arrangement of the degrees of freedom enables iMobot modules to have unique forms of locomotion.
2. iMobot: an Intelligent Reconfigurable Modular Robot

The novel design of iMobot incorporates four controllable degrees of freedom into each module, and two of the degrees of freedom are provided by rotating faceplates at each end of the module as shown in Figure 1. This enables an iMobot module to perform novel locomotive tasks, using the faceplates as wheels as shown in Figure 2.

The geometric design of the iMobot allows it to have six mounting locations per module, greatly increasing the possible configurations for even a modest number of modules. The mounting locations are located on the end of each faceplate, and one on each of the four side plates.

This unique mechanical design presents us with a platform which has great potential for performing a large variety of tasks. Cluster configurations such as one shown in Figure 4 may utilize a large number of possible gaits to overcome obstacles and carry heavy loads. However, the complexity of the system also presents novel challenges, such as the design of a distributed controller and efficient communication protocols.

3. Reconfigurable Software for Modular Robots

One of the main challenges faced with any modular robotic system due to the high level of reconfigurability and large number of degrees of freedom is the development of control programs quickly and efficiently in Section 3.1. Another major challenge is the development of an efficient communication method for collaboration of modular systems. We will present our usage of mobile agent systems to coordinate multiple robotic systems in Section 3.3.

3.1. Ch: A C/C++ Interpretive Programming Environment

Research and application experience of robotic systems indicate that an ideal programming language for robotic systems must be a sophisticated computer programming language. The language should appeal to both expert programmers and novice users. In most cases, sophisticated users will write high-level functions that can be readily used by less experienced users. The language must be deterministic for real-time programming. The language should be interpretive with a quick system response. The program compilation presents a serious problem for real-time manipulation of mechatronic systems. For a real-time robotic system, the external environment may be different at each execution, so the testing scenario may not be repeatable. During debugging and testing, it is impractical to restart a program from the very beginning every time a change is made or a problem is diagnosed. The programming environment should support command or function execution interactively. Although the language is interpretive, some time critical code such as control algorithms for servo loop might be compiled for fast execution. Therefore, interpretive scripts should be able to interface to binary objects. It is typical that multi-tasks or threads are executed for servo update, IO checking, user interface, etc. The language should support concurrent multi-task processing. In addition, the language should be a superset of an established computer programming language rather than a subset. Programming of robotic systems can then draw upon a large body of existing user and code base. The base language should be an open language with an international standard so that it will keep abreast advance of the new technology. The language should be easy to learn. It will be used not only for programming of robotic systems, but also for daily programming tasks. The object-oriented nature of the language will ensure that the code is relatively easier to develop, maintain, and reuse. Programming of robotic systems should be similar to very high-level shell programming. Application programs are created not by writing large programs starting from scratch. Instead, they are combined by relatively small components. These components are small and concentrate on simple tasks so that they are easy to build, understand, describe and maintain. The language should support modular programming. A set of high-level commands and functions developed by experienced researchers and engineers can be readily used by novice users. The programming environment should support a user-friendly graphical interface. However, an entirely visual based programming environment without a base of procedural programming language is difficult to program for complicated tasks, especially for sensor fusion. The language should be supported in different platforms such as Windows and Linux so that application programs will be portable. The language should be architecture-independent so that high-level application programs can be developed to
Figure 2. Various forms of locomotive possibilities for a single iMobot module, including rolling (a) (b) (c), inchworm (d), turning (e), and standing (f).

relieve programmers of the task of learning different programming languages for different operating systems or architectures. It will enable integration of mechatronic devices from different vendors for consistent interfaces external to robotic platforms. The language should support programming of several robotic systems concurrently. The language should support secure network computing with the standard networking protocol TCP/IP. It is desirable that a program can be dynamically downloaded through the network and executed securely so that the robotic system can be adaptive to the external sensory information. Many complicated algorithms are used in robotic systems, so it is desirable that the language supports advanced numerical features such as matrix computations.

With these considerations in mind, the Ch [12, 13] language environment offers many features desirable for building a control system for modular robots. As shown in Figure 3, Ch is an embeddable C/C++ interpreter that supports all features of the ISO C90 standard. Ch also supports many features of the latest C99 standard applicable to engineering, such as IEEE floating-point arithmetic [14], complex numbers [15] and variable-length arrays [16]. Other features provided by Ch that are desirable for programming modular robotic systems include the following:

1. Ch supports computational arrays [17] as first-class objects as in FORTRAN 90 and MATLAB for linear al-

Figure 3. The Ch language relationship with other popular languages.
gebra and matrix computations which greatly simplifies the forward and reverse kinematics required for multi-linkage motion systems.

2. Ch supports real-time control [18]. Real-time control depends on guarantee against any delay caused by page faulting. This requirement is needed to ensure stable and accurate control of actuators.

3. As a superset of C, Ch retains C’s low-level features for interfacing to hardware.

4. The interpretive system will help speed up the prototyping process. Since the robotic system is novel in many aspects, many iterations of control code must be generated to test the robot. By skipping the compile/link step of code development, lots of time may be saved during the code development step.

5. The Ch interpreter has been ported to a variety of systems which makes Ch code architecture independent. A control algorithm may be quickly tested on a desktop system before it is tried on the actual robotic platform.

6. Ch supports network computing for inter-module communication.

Ch has been used for real-time control and programming of robotic systems [19, 20].

3.2. Embedded Ch: An Embeddable C/C++ Interpreter

The Embedded Ch library is an embeddable C/C++ interpreter [21]. It allows a programmer to embed the Ch interpreter into their own C/C++ application. This allows a user’s C/C++ binary applications to call Ch scripts and functions, and also allows Ch scripts to call back C/C++ binary functions.

The embedded Ch library makes binary applications to be highly reconfigurable. For instance, consider a binary control program which executes various control algorithms residing on Ch scripts. The application may perform optimizations and other tasks in binary space for faster performance and execute Ch scripts for configurable behavior. A well designed system following this template may be quickly and easily reconfigured by simply modifying or replacing the Ch scripts. The Ch scripts may even be modified in real time without interrupting the executing binary program. This allows the robotic system to be configured dynamically in real time with virtually no down-time between tests.

3.3. Mobile-C: A C/C++ Based Mobile Agent System

In order to help facilitate module communication and cooperation, a mobile agent system known as Mobile-C [22, 23, 24] can be incorporated into the modules. Agent-based computing emerged in the past decade as a promising strategy for developing distributed complex systems. A mobile agent is a software component that can travel among different execution environments autonomously [25]. Agents are able to migrate between hosts while keeping their code and data state intact, as shown in Figure 5. While Figure 5 illustrates the agents migrating using the internet, any communication method could feasibly be used to as a medium of a migrating agent.

An embeddable mobile agent system known as Mobile-C which utilizes Embedded Ch [21] as its agent execution engine can be used with the iMobot modular robots. Mobile-C is a C library which implements a Foundation for Intelligent Physical Agents (FIPA) compliant mobile agency framework. Unlike other agency platforms such as JADE...
Mobile-C runs on heterogeneous platforms with various operating systems, such as Windows, Linux, Solaris, HP-UX, FreeBSD, Mac OS X, and QNX. Mobile-C has already been used in several robotic systems, such as robotic workcells [20]. In the robotic workcells, the agents take advantage of agent synchronization methods provided by Mobile-C to perform a coordinated task. Mobile-C has also been used on mobile robots performing distributed vision sensor fusion [27]. By utilizing mobile agents, image processing is done in situ on the robots, thereby saving network bandwidth and energy.

The Mobile-C library allows a mobile agent platform to be embedded in a program to support C/C++ mobile agents. The host program space is defined as the C/C++ binary space where a host program and the Mobile-C agency reside. The mobile agent space is defined as the C/C++ script space where a mobile agent resides. Since an agency is embedded in a host program to support mobile agents, a host program can protect itself from malicious agents by controlling the operation of the embedded agency and mobile agents. The Mobile-C library was designed to provide APIs relevant to an agency, different modules of an agency, agents, and other functionalities of an agency. These APIs can be called in a host program to have control over the embedded agency, different modules of the agency, and mobile agents operating within the agency. Also, agents are able to interact with the host program space through a set of mobile agent space APIs.

There are many possible uses for Mobile-C on a modular robotic system. Since Mobile-C agents are FIPA compatible, they may communicate with any other FIPA compatible agent, perhaps sharing or distributing tasks or resources. Mobile-C has also been used in distributed computing applications, such as computational steering [28]. These techniques may be applied to large clusters, and Mobile-C may be used to integrate the processors on board each module into a powerful computational platform. Such a platform may be used for any number of tasks, such as gait generation using evolutionary algorithms, or inverse kinematics to solve a specific robotic pose problem as described below.

4. Functionalities and Applications of Reconfigurable Software for Modular Robots

There are many potential uses of flexible reconfigurable software on modular robotic systems. Some sample applications are described below.

4.1. Distributed Computing

Reconfigurable software can be used to create a distributed computing system out of the module clusters. Each iMobot module uses a high powered Gumstix [29] tiny computer for high level processing. It is possible to utilize each of the processors in a distributed fashion to perform computational tasks in parallel. However, the communication speeds of our cluster, and of most modular systems, are fairly low compared to the bandwidth attainable with ethernet connections used by typical cluster computing systems. The task is then to divide up computational tasks in such a manner that all processors may be utilized, but at the same time be highly communication efficient.

4.2. Agent-Based Distributed Computing

A unique feature about agent based computing is that the agents may intelligently migrate. For example, the agents might selectively migrate to processors that have more computational resources available at hand for maximum computational efficiency. For instance, if an agent is currently residing on a processor that is bogged down with high priority real-time tasks, the agent may poll its neighboring hosts regarding their computational resources. If a better host is found, the agent may migrate to the other host, thus evening out the computational load. Since it is a fully distributed process, there is no single point of failure.

Another benefit provided by an agent’s mobility is its ability to migrate away from hosts which might be failing. For instance, if a host is about to run out of batteries or detects a malfunction, it may instruct its agents to migrate to another host. In this manner, no computational data is lost if the agents are able to successfully migrate away before the host terminates.

Agents may also track or poll the status of their neighboring hosts for a higher degree of fault tolerance. If a host dies unexpectedly, its control program and code state may be replicated by its neighbors. This gives clusters of robots desirable self-healing capabilities.

4.3. Distributed Agent-Based Genetic Algorithm

One possible technique that may be used is a distributed agent-based genetic algorithm. A number of agents may be created, each carrying its own unique population for use in a genetic algorithm. The agents may use the Genetic Algorithm Utility Library [30, 31] and run a genetic algorithm on its population, searching for an optimum genotype. The genotype might control any number of things, from a robotic gait to a simulated communication profile for maximum energy efficiency.

For example, if a single module in a cluster system dies, it may adversely affect whatever gait the cluster was using beforehand. If the robot is completely isolated, it will need to generate a new gait on its own. The cluster will generate a number of agents and give them the cooperative task of finding another optimum gait taking the inoperative module into account. The agents will efficiently utilize all on-board computational power to generate a new gait in the minimum amount of time.
4.4. Distributed Gait Generation

Gait generation may also be performed by each module independently, with each module working towards a different goal. In this manner, other algorithms which are not themselves suitable for distributed computation may be used. For example, module A might perform a simulated annealing process on a gait that moves the robot forward, while module B performs the same analysis, but on a gait that moves the robot backward. In this manner, multiple modules may be utilized to generate a comprehensive set of gaits for robot motion.

4.5. Distributed Sensor Fusion

Agent based computing may also be utilized to perform distributed sensor fusion. Agents are able to process sensor data in situ aboard whatever module it is currently residing on. For instance, agents can be utilized in distributed vision sensor fusion [27]. By processing image information on the host that the sensors are attached to, raw image information, which is very large, does not need to be sent over the network. The lightweight agent can then carry only the results of image processing with it as it migrates and analyzes other sensors.

5. Conclusions

Reconfigurable software systems, such as C/C++ based interpreters, embeddable interpreters, and mobile agent systems may prove to be extremely effective and useful for the design and implementation of modular robot software. A C/C++ interpreter Ch allows control programs to be quickly written and tested on a variety of different architectures without the need of compiling and linking programs. An embeddable interpreter Ch gives binary programs a high degree of reconfigurability by allowing computational components to be easily reconfigured. An embeddable mobile agent system Mobile-C takes advantage of both interpretative environment and embeddable interpreter to provide a mobile-agent based framework, on which distributed tasks may be easily composed and tested.

References


Graph Minor Analysis of Reconfiguration State Spaces

Tom Larkworthy\(^1\) and Subramanian Ramamoorthy\(^1\)

**Abstract**—Efficiently overcoming difficult motion constraints is the prime problem in development of efficient motion planning algorithms for self-reconfiguring systems (SRSs). Meta-modularization, and other related techniques, deal with the problem by adding further constraints in a way that simplifies planning. If \(R_n\) denotes a raw state space for configurations containing \(n\) sub-units, and \(C_n\) a further constrained version of \(R_n\) then \(R_n \leq C_n\) where \(\leq\) denotes the graph minor relation. Often the choice of \(C_n\) is ad hoc (although made on clever intuitions). We wish to study whether there are principles that may guide this choice. We demonstrate one such principle, that is planning is tractable, e.g. in meta-modularized sub-spaces, when \(C_n \leq C_{n+1}\), which captures a smooth increase in state-space complexity as more modules are added.

I. INTRODUCTION

Developing motion planning algorithms for self-reconfiguring systems (SRSs) is hard. It seems that SRS designs that are easy to manufacture contain difficult motion constraints that are hard to accommodate into a planning algorithm. A popular method for dealing with difficult local constraints has been to encapsulate a group of sub-units into a meta-module. Sequences of underlying motion moves can be developed that permit meta-modules to move in relation to one another on a coarser embedding space. With care, a new motion catalog operating at the meta-module level can be synthesized that has less motion constraints than the underlying motion catalog of the individual sub-units, and motion planning is simplified for the meta-modules reconfiguration state space.

However, meta-modularization has its drawbacks. General configurations in the underlying state space that adhere to the meta-module definitions are the exception, not the rule. So much of the generality of the SRS is lost. A local approach to adding constraints to a state-space is presented in [1] which generates a sub-space of the hexagonal metamorphic robot (HMR) called the Surface space. This sub-space occupies a larger proportion of the underlying state space compared to meta-modularization, but still simplifies planning effectively. In fact, the remainder of the underlying state space that is not included in the Surface space is small enough that planning using greedy search methods becomes tractable, and the general motion planning problem for the entire HMR state space becomes feasible for high numbers of units.

Clearly, adding constraints to an underlying state space can simplify things. Adding the fewest constraints possible is also desirable, as less generality is lost and plans in the efficient sub-space can be fused with other planning methods in the more general sub-space effectively. An important question that has not been studied is this: what are the attributes required of the constraints to make planning more efficient? If we understood the answer, then perhaps we can invent a minimal set of constraints to add to an underlying model in order to simplify planning.

We use a fundamental concept from graph theory, that of graph minor relationships, to discover a striking property that published meta-modularization state spaces and the Surface state space possess, that difficult planning state spaces do not. Specifically, the reconfiguration state spaces generated by adding a unit are well ordered by the minor relationship. The fact that difficult state spaces do not seem to possess this quality suggests that minor ordering in state spaces is one of the mechanisms in play that describes when a state space admits a scalable motion planning algorithm.

II. PRELIMS

An undirected graph, \(H\), is said to be a minor of a graph, \(G\), denoted \(H \leq G\) if there exists a sequence of vertex deletions, edge deletions and edge contractions that modify \(G\) to \(H\). An edge deletion is a removal of an edge from the graph. An edge contraction merges the two connected vertices, \(v_1\) and \(v_2\) of an edge into a single vertex that inherits the connections of \(v_1\) and \(v_2\). An example is shown in Figure 1.

The topic of graph minors is deep, any topological space that permits an embedding of \(G\) will permit an embedding of \(H[2]\). Many important classes of graphs can be described by the minors they do not contain, for example, trees are all graphs that do not contain a clique of three vertices as a minor, all planar graphs forbid two specific minors.

III. STATE SPACES

We perform an analysis of graph minor ordering on different state spaces of the hexagonal metamorphic robot (HMR). The HMR is a collection of sub-units arranged on a hexagonal
lattice. All sub-units must form a single connected region, which may or may not contain enclosed space. Thus, an HMR configuration state can be represented as \( n \) unique hexagonal lattice coordinates. A single sub-unit can move to an adjacent hex location per iteration of time, subject to the constraints of the motion model. The single move (or short move) reconfiguration graph represents the entire motion state space for a given model of reconfiguration. The vertices of the reconfiguration graph represent an HMR configuration, and an edge represents that a permitted move exists between the two connected configurations.

Different motion models generate different reconfiguration graphs. We use the graphical nomenclature of Ghrist[5] to describe the motion generators for each state space. A Ghrist motion generator identifies the local context required for a move to occur (called the support) and the locations with which the move occurs (the trace). If a given sub-unit can move from location \( a \) to an empty location \( b \) this implies that a rigid body transformation exists that moves the trace of a generator over \( a \) and \( b \), and that its support matches the area around \( a \) and \( b \). In addition, the global property that all sub-units remain connected may require addition checking.

The original definition of an HMR was developed by Chirikjian[3], and was the least restrictive. A unit could move if it had a robotic neighbor to pivot around, Figure 2 A. We denote the state space of this model as \( S_a^S \). A more restrictive HMR can be found in the motion model of the Claytronics project HMR[4], which requires empty space opposite the pivot location of the moving unit, Figure 2 B; we will denote this state space as \( S_b^C \). Both \( S_a^S \) and \( S_b^C \) motion catalogs do not enforce the global constraint that all sub-units must remain as a single connected component, so this needs to be checked explicitly before a move can be considered valid.

Ghrist presented his own motion catalog that did not require this check, the generators are shown in Figure 2 C[5]. This model implicitly does not permit gross topological changes in the configurations, such as the introduction or removal of enclosed space, as in the move shown in 2 D. We will denote the Ghrist reconfiguration state space as \( G^S \); note this reconfiguration graph is disconnected for \( n \geq 6 \) when enclosed space is possible.

A new HMR motion catalog is presented called the Surface model[1] which is a further constrained version of Ghrist’s \( G^S \). A Surface adhering configuration, \( c \in S^L \), is defined as a configuration that permits an unviolated Euler tour to be wrapped around the adjacent external locations. Violations of the Euler tour comes in two varieties. A kink violation is defined as a location where the Euler tour leaves through the same edge it enters from, and a dual path violation is where the Euler tour traverses through the same location more than once (Figure 3).

A sub-unit of the Surface model can move according to the Ghrist catalog, but is prevented from stopping if the resulting configuration contains a Euler tour violation. This model of reconfiguration does not lend itself to compact representation in Ghrist’s nomenclature, because the move’s trace might not be in adjacent hex locations. However, the validity of a start or end trace location for a move can still be expressed with a local support only, Figure 4. A valid surface move is then any valid removal followed by any valid addition. We denote the Surface state space as \( S^L \), the \( L \) suffix indicating the state space is comprised of long-moves. By construction, and unlike \( G^S \), if a sub-unit can move, it can move anywhere on the perimeter of the configuration, if it does not lead to an Euler tour violation (Figure 3).

Meta-modularization of a state space normally involves dividing the embedding space into a coarser lattice[6]. Figure
5 demonstrates an example meta-modularization that would simplify planning under the Claytronics motion constraints. Using a tunneling procedure meta-modules are capable of virtually moving between locations, if the meta-module reconfiguration graph remains globally connected. The decoupled start and end local constraints are much less than $S^L_i$ and are shown in Figure 5 C. We will denote the state space of this example meta-modularization as $M^L_i$.

To compare the state spaces of each model we first must convert the single-move state spaces into long-move state space\(^1\). The long-move reconfiguration state space has an edge between two configuration vertices if their exists a consecutive sequence of single-moves acting upon an individual sub-unit that links the two configurations i.e. a long-move is simply the locations that a single unit can reach while all others remain stationary. In this view $\bigcup J^L_i \subseteq \bigcup C^L_i \subseteq \bigcup S^L_i$ and $\bigcup C^L_i \subseteq \bigcup M^L_i$. The minor relationship compactly represents what we mean by adding further constraints to a motion model, even if the constrained motion model uses intermediate configurations transiently that are not exposed to the higher level planning space (these edges are contracted, rather than deleted).

\(^1\)This is not strictly necessary, but simpler than describing single move analogies for the Surface and meta-module models

\[\text{Definition 1. The } \text{local structure} \text{ for a configuration, } v, \text{ is all possible configurations generated by applying an ADD from the Surface catalog (Figure 4).}\]

\[\text{Definition 2. The } \text{inherited local structure} \text{ for a configuration, } v, \text{ is all possible configurations generated by applying an ADD from the Surface catalog (Figure 4).}\]

\[\text{Theorem 5. When there are 6 or more units in a configuration, there are 6 or more possible applicable ADD locations, of which, given any two, a third shall exist with a surface perimeter distance of greater than 2.}\]

\[\text{Proof: Omitted for brevity}\]

\[\text{Lemma 6. For every move } x \rightarrow y \text{ in the local structure between configurations } v \text{ and } u, \text{ there exists at least one pair of configurations } v' \text{ and } u' \text{ in the inherited structure with the same move valid.}\]

\[\text{Proof: Let } v = X \cup \{x\} \text{ and } u = X \cup \{y\}. \text{ We simply need to find an ADD location } z \text{ that can be added to } v \text{ and } u \text{ such that it does not interfere with the local contexts that enabled the move } x \rightarrow y \text{ to take place. Interference with local contexts can only occur when the perimeter distance from } z \text{ to } x \text{ and } z \text{ to } y \text{ is less than 3 (Figure 4). Lemma 5, shows such a } z \text{ exists in all configurations when there are more than 6 sub-units in the configuration. For smaller configuration brute force enumeration has shown such a } z \text{ exists in all possibilities containing 5 or less sub-units (not shown). Thus a } z \text{ always exists such } v' = X \cup \{x, z\}, u = X \cup \{y, z\} \text{ admits } x \rightarrow y.\]

\[\text{Lemma 7. The local structure of a vertex, } v, \text{ is a minor of its inherited local structure.}\]

\[\text{Proof: By Lemma 6, for each neighbor, } u_i, \text{ in the local structure representing a move } v \rightarrow u_i, \text{ there exists a } z_i \text{ of an additional unit that permits the same move to take place, } v \cup \{z_i\} \rightarrow u_i \cup \{z_i\}. \text{ By definition, } v \cup \{z_i\} \text{ is in the inherited local structure. By Lemma 3 there exists a move between all } v \cup \{z_x\} \text{ pairings. If all moves between } v \cup \{z_x\} \text{ are contracted and all edges not } v \cup \{z_x\} \rightarrow u_i \cup \{z_i\} \text{ in the inherited structure are deleted, the remaining edges are the local structure (Figure 6).}\]

\[\text{Theorem 6. } S^L_1 \leq S^L_2 \leq \ldots\]

\[\text{Proof: By Lemma 7 every vertex in the } i \text{ graph is a minor of the inherited graph. For a pair of configurations in the } i \text{ graph, } u = X \cup \{x\}, v = X \cup \{y\} \text{ with a move between them, } x \rightarrow y. \text{ Lemma 7 states an ADD location on each, } z_u \text{ and } z_v \text{ exists such that the same move can take place in the inherited structure, } X \cup \{x, z_u\} \rightarrow X \cup \{y, z_v\} \text{ and } X \cup \{y, z_v\} \rightarrow\]
Figure 6. A local structure (left) is a minor of the inherited local structure (right). The left central vertex is surrounded by all configurations reachable by a move (its local structure). The right central vertex contains the inherited structure for that vertex (a clique), yellow denoting where the additional sub-unit has been added. For every move in the local structure (white to purple), a comparable move can be found in the inherited structure with an addition sub-unit added, denoted by the vertexes joining the central vertex. The red lines within the inherited structure shows which moves are required to move the addition sub-unit around to "get out of the way" so that all comparable moves can execute. Deleting all black edges in the inherited clique followed by contracting the red edges reproduces the local structure.

\[ X \cup \{x, z_v\}. \]

By Lemma 3, a connecting move between the local minors exists between \( X \cup \{x, z_v\} \) and \( X \cup \{x, z_u\} \) and thus we can compose all the local minors of Lemma 7 into a graph; edge contracting the connecting moves to produce the reconfiguration graph containing \( i \) sub-units.

**Remark** 9. For an alternate viewpoint on the same result, we could entirely skip the composition of local graphs. An extra sub-unit can be added, and moved out of the way in order to realize all sequences of realizable moves (Lemma 6). However, this loses sight that there is a notion of locality relating the local structure to the inherited local structure through the embedding space. This becomes important when we consider the counter example for the Ghrist model.

**Conjecture 10.** For some \( i > k \), the Ghrist reconfiguration graph containing \( i \) units is not a minor of \( i + 1 \).

Our above construction of minor for the Surface model does not hold for the Ghrist configuration graph because an additional sub-unit in the inherited local structures does not, in general, form a clique structure. Thus, while a location may exist for every local move that permits the move to take place in the inherited structure, there may not be connections between these locations. Figure 7 shows and example where the inherited structure is disconnected. In these cases the local structures are not minors of the inherited structures, and so a global minor cannot be constructed from a composition of local minors. The "trick" within this proof can be understood visually by careful comparison of Figures 6 and 7.

**Theorem 11.** \( \mathcal{M}_1 \leq \mathcal{M}_2 \leq \ldots \)

**Proof:** Omitted for brevity, but the proof largely follows the logic for the Surface model.

Decoupling the start and end positions of moves is the primary reason why minor ordering is found in the reconfiguration graphs of the easy planning spaces. It must be noted though, that the minor ordering is a global structural property of the reconfiguration graphs, and not a consequence of the language used to describe the motion catalogs. An implication of minor ordering is that reconfiguration solutions can be reused across complexity levels, and suggests planning can be solved recursively, locally and iteratively. This desirable quality can be detected in polynomial time[2].

**V. Conclusion**

The graph minor relationship is a powerful mathematical tool for analysis of SRS state spaces, with deep theoretical implications. It provides a compact notation for describing when one state space is a constrained version of another. We have provided several examples but there are other examples within the literature. Butler et al.'s generic work [7] can be compactly restated as: the state space of the SRS motion model within is a minor of the Molecule, Crystal, M-Tran and HMR state spaces.

Furthermore, state spaces that are easy to form plans with, such as meta-modularization, seem to be well ordered by the minor relationship across graphs containing different numbers of units. So the most important consequence of this work is that we now have a method for expressing why we add constraint to models, and a mechanism for how those constraints simplify planning. Perhaps we are now in a position to automatically find efficient planning sub-spaces for more awkward SRS motion catalogs.

**References**

A Hybrid Control Strategy for a Chain type Modular Robot

Rodrigo Moreno and Jonatan Gomez

Abstract—This paper presents a hybrid strategy for controlling a chain type modular robot to achieve a better interaction with the environment based on the combination of Central Pattern Generators (CPG) and Hormone inspired control. Locomotion movements of robot chains are generated by the interaction of the CPGs, modeled as nonlinear oscillators and run one per module, with its nearest neighbors. Each module contains a set of proximity sensors to detect nearby objects and obstacles. Once an obstacle is detected a hormone message will be propagated gradually along the structure triggering different reactions depending on the topology and orientation of each module. The reaction is achieved by adding or subtracting predefined values from the CPG parameters like its amplitude, offset and phase difference with its neighbors. Finally, the complete behavior of the module is the sum of all external stimuli brought to it by the hormone messages and its current state. Simulation of a possible reaction to a sensor stimulation is presented.

I. INTRODUCTION

All over the world there has been an accelerated development in research related to robotics and its applications, within that is the field of cooperative robotics and modular robots. These robots are built from basic units, called modules, with or without autonomy, that can reconfigure themselves to perform different tasks [1]. The systems posses the characteristics of being scalable, robust and reconfigurable. The main applications for this kind of robots include search and rescue on natural disasters, space exploration, and monitoring and repair of non-accessible areas in factories.

In general, systems of multiple robots have a greater advantage over single robot systems because when faced to multiple tasks the last one is, one way or another, spatially limited. For example, if a robot was designed to have four legs, it will have four legs forever. This problem is solved by the reconfiguration ability (also known as morphogenesis [2]), which can be automatic or manual, modular robots have. However this advantage has its problems, such like finding the best configuration under certain environment [3].

Modular systems provide versatile solutions to many kinds of problems thanks to their reconfiguration ability. But this capacity also brings the problem of more complex control algorithms than that of monolithic robots. Chain type robots are an special subgroup of modular robots. These robotic modules form chains when connected and can be arranged in rather arbitrary configurations. This kind of robots have been widely studied and many prototypes have been built ([4], [5] and [6]). Many control techniques have been developed for controlling modular robots and specifically chain type modular robots, these techniques vary in complexity and can be as simpler as movement tables [7], or as complex as hormone inspired control [8].

This article focus only on the main features of motion generation using CPGs (Central Pattern Generators) and hormone inspired messages, and how they are used to create a hybrid algorithm to control a chain type modular robot. CPG techniques and hormone inspired messages are described on sections II and III, section IV describes a simple chain type modular robot model to help describing the algorithm, section V describes the hybrid approach, section VI shows some simulation examples of the algorithm, and finally section VII provides conclusions and discusses future work.

II. CENTRAL PATTERN GENERATORS

The main idea of CPGs is that in many animals, like quadrupeds, some kind of coordination is needed between body parts to achieve some kind of displacement. In this animals a central pattern of movement is generated at the spine and is propagated to the muscles that control limbs. Using this model in the case of a modular robot in snake configuration, a sinusoidal pattern is generated in one of the modules and is propagated along the chain of modules with different phases to create a coordinated movement.

Central pattern generators provide a basic structure for generating coordinated motion between different parts of a robot with multiple degrees of freedom. Generators are often modeled as coupled nonlinear oscillators that can synchronize in an automatic fashion due to interaction with its neighbors, this avoids the usage of external synchronization methods. Another advantage of CPGs is that they are robust under perturbation coming from the environment [9], and react in a smooth way to changes on its parameters, avoiding abrupt changes on the actuators set points and preventing damage.

Despite the advantages mentioned, interaction between oscillators and it’s parameters must be optimized (using genetic algorithms or other methods like Powell’s) to achieve effective locomotion patterns. Even though the nonlinear oscillators which are used to model the CPGs have few control parameters to optimize [10].

III. HORMONE INSPIRED CONTROL

Hormone messages inspired control is another case of a biological inspired system. This time it is inspired on the
hormonal messages used by human beings. Hormones are chemical substances that are segregated by certain organs which travel all along the body triggering different reactions. Only a group of special cells with specific receptors react to each kind of hormone. Hormone molecules bind to the receptors and trigger different effects on the actions and properties of each cell. Hormone messages are defined as messages that can be interpreted in different ways by different kinds of receptors, and that can even be modified or delayed before transmitted to other cells [8].

The hormone inspired strategy constitutes a multi agent distributed control. With this type of control every module is considered an independent agent that works with local information. Using some features of role based control [11], each agent can decide which behavior to perform based on its topology. Topology of each individual module can be discovered by checking how it is connected to its neighbors and the paths that messages take to reach the module, this way it can check not only its local topology but its neighbors topology. Local topology and neighbor topology must be discovered in order to determine an unique position on every configuration [12].

Once an unique position is determined on the structure (it does not have to be the global structure) the module can use this information and the information brought to him by hormone messages as well to choose the most suited behavior in order to help the total system to perform a certain global task. The global task can be assigned by an external human operator or controller, or it can be a reaction to a sensor stimulus coming from the environment. To successfully perform any of these tasks enough modules are needed to agree on the task, this can be achieved by using the algorithm proposed in [12].

The main advantage of this strategy is that there is no necessity for unique identifiers for each module so that it can be generalized to almost every configuration of the robot. Also, the system is simple enough to implement it on the hardware present modular robots have. Its main disadvantage is that a great part of the algorithm is used to generate movements and synchronize them, using system resources on the storage of movement commands that can be generated by other means.

IV. CHAIN TYPE MODULAR ROBOT MODEL

The algorithm described on section V combines both strategies from sections II and III to achieve a more advanced control regarding environment interaction, creating a hybrid control system. To illustrate the general features of this algorithm it is necessary to define a basic modular robot model. Figure 1 shows the simplest case of a chain type modular robot model. The module has only one degree of freedom, i.e. it has only two parts attached to each other through a rotational joint, the north part (red) and the south part (blue). Each part has only one connector face to connect itself to other modules, having a total of two connectors on the module. The connectors are modeled as rigid mechanical connectors so rotation and translation are restrained once the module is connected. By using this type of basic modules simple chains can be formed, thus providing a basic structure to describe the algorithm.

V. HYBRID ALGORITHM

The hybrid algorithm seeks to use the movement generation features of the CPGs, executing one nonlinear oscillator per module (Figure 2). This frees the hormone based control from this task and allows it to focus its advantages on managing tasks and reactions to external stimuli. For example, a sinusoidal locomotion movement could be established on a snake configuration, controlled by CPGs, and by the time an obstacle appears on the robot’s trajectory a hormone message could be propagated to modify the behavior of certain modules in order to avoid the obstacle.

The behavior of each module is defined when the amplitude, offset and phase difference of its CPG are determined. The coupling of each CPG is only with its nearest neighbors so the phase difference is only established between the module and them. The coupling information is sent to neighbors on every cycle of the module’s program. The equations that determine the behavior of the CPGs are shown at (1), where $\theta_i$ is the output of the oscillator, $\phi_i$ represents the phase, $x_i$ the offset, and $r_i$ the amplitude of oscillator $i$. $w_{ij}$ represents the coupling weights between oscillators and $\varphi_{ij}$ the phase difference. These equations are similar to those used in [13] and ensure that the phase difference, amplitude and offset will converge to $\varphi_{ij}$, $R_i$ and $X_i$ respectively, showing a limit cycle behavior.

While connected in any configuration the modules can send or receive messages to other modules by some communication method. This method depends on the physical system in which the algorithm is implemented. Communication between modules is only between nearest neighbors for simplicity, so broadcast messages are not taken into account. Hormone messages transmitted by modules have special characteristics. The first one is that they do not possess a destination, instead they travel from module to module along the structure, with a header that is identified by each module depending on various conditions like its topology and orientation. If the module is interested in the message it will proceed to change its properties based on the information of the message.
Each module runs one CPG modeled as an oscillator with its corresponding amplitude, offset and phase difference between neighbors. The oscillators are coupled only with their nearest neighbors.

\[
\dot{\phi}_i = w_i + \sum_j (w_{ij} \sin(\phi_j - \phi_i - \varphi_{ij}))
\]

\[
\ddot{r}_i = a_r \left( \frac{a_r}{4} (R_i - r_i) - \dot{r}_i \right)
\]

\[
\ddot{x}_i = a_x \left( \frac{a_x}{4} (X_i - x_i) - \dot{x}_i \right)
\]

\[
\theta_i = x_i + r_i \cos(\phi_i)
\]

Each module has 6 proximity sensors, one for each spatial orientation in 3D (See figure 3). The module produces a different message for each one of these orientations. If there is an important change in orientation this will be detected and the messages will be emitted in agreement to the new orientation, i.e. the new sensor facing up after the change in orientation will produce the corresponding up message. Any message is only emitted if there is a change in the sensor state, i.e. if its value is different from zero, and only one message is generated at a time. If there is more than one sensor which value is different from zero the one determining the message to emit will be chosen by a priority system. For example the sensor in the front will have bigger priority than a side sensor.

The message is emitted only while the stimulus is active and the module ceases its transmission if the sensor goes back to zero. The message generated by the module is then transmitted to its nearest neighbors. The module neighbors propagate the message only to the modules from which they did not received it and so on as described in [12]. This ensures the message to be propagated along the structure.

Propagation of messages must be done gradually, this is meant to avoid deadlocks, and is because there could exist various sensor stimulated on different parts of the structure and therefore parallel reactions. Messages are then propagated only if they are received a predefined number of times. Gradual propagation ensures messages not to be propagated indefinitely along the structure, even if there are cyclic pathways.

Messages sent by other modules are received by the modules in the propagation path. Each individual module checks the header of the message and if it is accepted by its current configuration it will change its behavior. In a specific way, each message accepted by the module sums or subtracts a predefined value to modify a variable (See figure 4). The total behavior of the module is then the sum of all the modifications due to the acceptance of the messages and its current state.

Message acceptance, the modified variable, and how much such variable is modified, are determined by the topology and orientation of the module. If a message corresponding to an upper sensor is received by a module which orientation is sideways, it will respond in a different way to if it were on another orientation, or it would not even respond at all.

VI. SIMULATION

The system described on section IV and the algorithm on section V have been implemented on simulation by using the Unified Simulator for Self Reconfigurable Robots (USSR) [14]. The USSR is a simulation platform based on Java that allows to build physics simulations based on the Open Dynamics Engine (ODE), and that contains useful examples and classes to model modular robots. The possible orientations of the modules have been limited to two, straight and sideways (See figure 3). The variables to modify and how
Fig. 4. Here an upper view of a chain with one sideways oriented module and four straight oriented modules is shown. Since the only module who can turn left or right is the sideways one it will modify its properties due to messages propagated by modules detecting obstacles from left and right. In this case, for example, it modifies its amplitude by adding a predefined value if it receives a right message, or subtracts the same value if it receives a left message.

Fig. 5. Side view of a possible reaction to an obstacle detected by the front sensor. Once detected the module emits a front type message that is gradually propagated along the chain formed by straight oriented modules. Each module reacts depending on its topology and the overall effect is to raise the structure over the obstacle.

much they are changed are determined manually, so reaction behaviors to different messages are designed beforehand. In figures 5 and 6 an example of an external stimulus due to an obstacle on the robot’s movement direction is shown. After detecting the obstacle on its front sensor, the first module of the chain emits a message to the next module connected to it. The message is propagated gradually along the structure and the general movement of the modules reached by the message is to raise the first module above the obstacle to avoid it. Each module reacts based on its own orientation and topology.

VII. CONCLUSIONS AND FUTURE WORK

The hybrid approach to control a modular robot combines the features of motion generation without explicit synchronization of the CPGs and the simple way of propagating information and modifying individual module behavior of hormonal messages strategy.

While the CPGs are engaged on generating movements, hormonal messages are used to propagate information on external stimuli to achieve a better interaction with the environment. The interaction translates into the modification of movement parameters to, for example, navigate through rough terrain or avoid obstacles. Indeed, this kind of information could be used to determine whether to reconfigure the whole structure.

Establishing the changes of behavior manually forces to design the specific behavior of a module in every configuration to different kinds of stimuli beforehand. The design of different reaction to different stimuli can be a very tedious and complicated task if more degrees of freedom and connection possibilities are added to the module. This can be solved by letting the module choose its own response to the stimulus in a random way or studying the evolution of the system under a machine learning algorithm.

The system uses asynchronous messages, i.e. the coupling information of the CPGs, the emission and propagation of hormone messages are done separately and at any moment, so the only limit to manage multiple stimuli (i.e. various messages) in the system at the same time is the capacity of the real system to receive and process those messages inside each module. This capacity is determined by the processing power of the module and the buffer size of the communication system.

The concept of capacity to manage multiple stimuli is similar to the concept of capacity of a medium to contain a number of hormone molecules or neurotransmitters, and the number of receptors that each cell have in order to receive this molecules at a given time. This capacity is finite even in biological systems and it is worth studying it when implementing the algorithm on a real system.

VIII. ACKNOWLEDGMENTS

This research is supported by Bogota Investigation Direction (DIB) of the Universidad Nacional de Colombia under grant 10986 of the Postgraduate Thesis support program.

REFERENCES


