Control and Interaction Strategies for Self-Reconfigurable Modular Robots

Stéphane Bonardi
I&C, EPFL

***Abstract*—In this article we present two major challenges in the field of self-reconfigurable modular robotics, namely the reconfiguration problem and the interaction aspect. The former has already been widely addressed in the robotics community. Two different approaches have shown to be particularly well-suited to tackle this issue. The first one is based on a gradient procedure to guide the different modules to the final position. The second approach is built on top of existing methods in control theory. It uses planning techniques to ensure convergence to the desired configuration. We illustrate these approaches by reviewing two very promising works in the field. Contrary to humanoid robotics, the aspect of interaction with end-users has been seldom studied in the community of modular robotics, mainly because the developed platforms were intended to be used in research environments by experts. We give an overview of the work we have already done in these two domains, applying it to the robotic platform Roombots, created at BioRob, EPFL.**

***Index Terms*—Control, Modular robotics, Interactions, Self-reconfiguration**

I. Introduction

THE field of modular robotics addresses the question of the design and control of robots made of multiple units, called modules, able to connect together using a connection mechanism to form more complex entities. Among those robots, self-reconfigurable modular robots are able to autonomously change their shape to fit the task they have to perform. It is done by re-arranging, adding or removing modules inside the main structure (videos illustrating these processes can be found at [1]). The problem of finding the sequence of actions required to go from one configuration to another is known as the ”reconfiguration problem”. This problem is computationally challenging since the number of possible configuration increases exponentially with the number of degrees of freedom and the number of connectors in the structure. Hou et al [2] have proved the NP-completeness of the self-reconfiguration process, justifying the use of heuristic methods. Many different approaches have been proposed to tackle this issue. Most of them use an abstract representation of the module, called the *sliding cube model* [3]. In this abstraction, the modules are represented by cubes able to slide perfectly on the surface of the structure. Multi-agents frameworks have addressed the question of self-adaptation of a modular structure to real-world perturbations using a consensus-based approach [4] as well as reaching and grasping objects using modular robots with evolving morphology [5]. Gradient based methods use a bio-inspired technique similar to hormone driving mechanism: the modules are guided towards their goal position according to a gradient function based on the distance to the desired final position. Unfortunately gradient methods are prone to be trapped in local minima, which might lead to *deadlock situations* in this problem. The use of scaffolding structure [6] and strict building sequence [7] have been introduced to avoid these situations. The self-reconfiguration process can also be viewed as a planning problem. Methods from this domain, such as Markov Decision Process, can be used to create a complete and efficient framework [8], taking into account the kinematics models of the real hardware. Theoretical justifications and complexity analysis are, in this case, available in a more systematic way as opposed to the previous approach. In order to reduce the complexity of the reconfiguration problem, the notion of meta-modules has been developed: instead of considering isolated modules, groups of modules (most of the time two units) are used and controlled as the basic elements of the structure (see for example [9]). Whereas all the previously cited techniques were based on distributed frameworks, a very promising centralized method using graph theory analysis has been developed and recently improved ([10] and [11]).

The human-robot interaction in modular robotics has been

Proposal submitted to committee: January 26th, 2011; Candidacy exam date: February 8th, 2011; Candidacy exam committee: Prof. Boi Faltings, Prof. Auke J. Ijspeert, Prof. Pierre Dillenbourg, Prof. Alcherio Martinoli.

seldom studied until now. While this question is a key component in the domain of humanoid robotics, it has often been eluded in modular robotics, considering that the developed platforms were intended to be used in research environments by experts. Nevertheless, some modular robotics projects specifically target every day life and home application. Among those, the Roombots project developed at the Biorobotics laboratory (EPFL), aims at providing adaptive furniture able to self-assemble, self-reconfigure and locomote in an unknown or changing environment. In this context, interaction with the end-user is crucial: efficient interfaces, able to support the user in the tasks related to the control of the robots, are to be created. Work has already been done in the domain of swarm robotics (see [12] and reference within), but the results might not be directly applicable to the interaction problem we are trying to tackle.

This report is divided into two main parts. In the first part, we review three papers which represent major contributions in the field of modular robotics. We start by presenting the main challenges in the field of self-reconfigurable modular robotics (section II-A) based on the paper by Yim et al [13]. We then present two successful approaches to tackle the reconfiguration problem. The first one by Stoy [6] is based on a gradient method (section II-B) whereas the second one by Fitch et al [8] uses planning techniques and reinforcement learning methods (section II-C). In the second part of this report, we introduce our own approach to the reconfiguration problem and to the other aspect of interaction with the modular robotic system (section III).

II. Litterature review

A. Challenges in modular robotics

Modular robots are able to change their morphology to adapt to a given task. Even if this ability might significantly decrease the performance of the system in comparison with a monolithic (specialized) robot, it is particularly suited in the case of multiple tasks assignment and not perfectly defined situations, bringing both *adaptability* and *versatility* to the system. Moreover, the fact of having a robot made of identical entities allows fast repair procedure by interchanging modules, either within the same robot or between robots. The overall system *robustness* is thus increased.

One example of application for self-reconfigurable robots is space exploration. There are several reasons for this. Firstly long term missions require self-sustainable systems, capable of self-repair and self maintenance. Secondly, spatial exploration is always heavily constrained in terms of volume and weight for devices that can be brought into space. Finally, given the unknown environment and tasks requirements, the robotic system should be able to self-adapt and perform multiple tasks autonomously.

We present in the first section the different types of existing architectures in modular robotics. After briefly describing some examples of robots, we state the grand challenges that still need to be overcome in this field.

1) Classification: There are several ways of classifying modular reconfigurable robots, depending on their architectures, the nature of the units and the type of control of the reconfiguration and motion processes.

First of all, modular self-reconfigurable robotic systems can be composed of exactly the same modules (*homogeneous* case) or can include different types of units (*heterogeneous*). One might argue that the fact of having specialized units inside the system will decrease its robustness and adaptability. Nevertheless, the gain in terms of performance will compensate for this aspect, keeping in mind that the set of basic units should also be able to perform the tasks (at the cost of a loss of performance).

Classification between modular robots can also be done based on the architecture of the resulting structures. If the units inside the final robot are arranged into a regular 3D grid, we talked about a *lattice* architecture. In this case, the reconfiguration process is easier since the set of possible moves is reduced to adjacent grid positions. The units can also be connected together in a *tree* configuration. This *chain* architecture is computationally more challenging, but allows for a richer set of reachable points in space. Finally, *mobile* architectures can be considered as an hybrid architecture able to use both lattice and chain structure, but also the environment to move and coordinate actions between multiple sub-robots.

From the control point of view in the reconfiguration process, we can clearly distinguish between *deterministic* and *stochastic* approaches. In the first case, the entire process can be pre-computed and the position of the different units is known at any time. The convergence time (i.e. the time to obtain the desired structure) can also be determined exactly. Most of the time, this type of control is used for *macro-scale* systems (typically with units of more than 10 cm). On the opposite, for *micro-scale* systems, *stochastic* methods are often well-suited. In this case, the connection and disconnection procedures are based on statistical processes. The convergence time can be guaranteed only statistically, and bounds are often used. Often, the environment is active, in the sense that it provides the energy (or part of it) needed for the motion of the modules.

2) Examples: More than 60 different modular robotics platforms have been created during the past 25 years [14]. Two main characteristics are often used to classify these platforms: their degrees of freedom (number, direction,...) and their connection mechanisms (type of connection, number of passive/active connectors,...). Well known platforms are the M-TRAN [15], the Superbot [16] and the Molecubes [17].

3) Challenges: Various achievements have been made in the domain of self-reconfigurable modular robots, both in terms of hardware and software. Robotic systems have been built and demonstrated to be able to self-assemble, self-locomote, self-reconfigure and self-replicate [18]. Planning algorithms able to control millions of abstract modules have been introduced [19]. Several challenges remain to be solved to allow such systems to keep their promise.

Concerning the hardware, some key points still need to be addressed. The current modular robotics units are often tasks specific in their design. Due to space constraints, some choices have to be made on the kind of characteristics the module should exhibit. Another key evolution in the hardware domain is the *self-replication* ability. The module would be able to build copy of itself from basic pieces or even from raw material to ensure real *self-repair* ability.

In terms of software, several ways could be explored to improve the current state of the art in the field. First of all, even if we can control millions of abstract modules, integrating kinematics data from the real hardware (through sensor information fusing) and taking into account failure (mechanical, electronic, communication,...) are still missing aspects. Moreover, being able to recover from modules failure and to handle the defective units inside the overall framework are considerations that have to be included in the current implementations to create real *self-sustainable* systems. Finally, if we envision a real multi-purposes set of modular robots, efficient algorithms should be developed to determine the optimal shape for a given task.

As we have seen, self-reconfiguration planning is one of the main challenges in modular robotics. Several approaches have been proposed to tackle this issue. In the next section we present one very promising method based on gradient techniques and cellular automata representation, developed by K. Stoy [6].

B. First approach: gradient based method

The idea of this approach is to use an automatically generated cellular automata to control the growth of a modular structure made of ideal cube units. The growth is guided from seed modules using three different types of gradients. This method does not rely on planning procedure explicitly, by introducing non-deterministic building sequences. The main issue encountered when using such techniques is the difficulty of ensuring the convergence of the algorithm. Indeed, gradient based methods are prone to local minima issue. To solve this problem, Bojinov et al [20] have introduced the idea of functional properties of the final structure: there is no need to build exactly the final configuration and it is sufficient to create a similar structure in terms of functionality. Another way is to impose a strict order in the construction of the structure [7]. [6] uses a scaffolding technique to avoid local minima. The CAD model is approximated by a structure made of cubes, itself approximated by basic substructures constituting the "skeleton" on the configuration.

1) Cellular automata: In [6], cellular automata (CA) are used to represent the desired structure to be build by the modules. The most difficult part in designing CA is the creation of the right set of local rules that will lead to the final configuration. K. Stoy [6] proposes an automatic method to create these rules from the CAD representation. The four main required steps of this part can be summarized as follows:

1) **Approximation of the CAD model**: the 3D model is filled with cubes.
2) **Scaffolding**: building blocks are used to further approximate the previous structure, avoiding deadlocks and local minima in the process. This step will ensure *built-in convergence* of the algorithm.
3) **Numbering**: each cube is given a unique ID in the final structure.
4) **Rules generation**: a rule is generated for each neighbouring pair of modules (i, j). The rule will look like: the CA in the direction $\vec{ij}$ should change its state to $s(i)$ if it is in the state $s(j)$

The final cellular automaton is composed of all of these rules. The initial state, called the *wandering* state, is chosen different from any already existing state.

2) Reconfiguration: The reconfiguration procedure is composed of the following three elements:

a) **State propagation:** At the beginning of the reconfiguration procedure, each module is initialized using the same copy of the CA. One module will be randomly chosen to be the seed and will be given a state among the available states. The structure will then evolve according to the CA rules. If needed a module is able to attract wandering modules. When a module filled a position, it becomes a seed. If this position is part of the final structure, the module is considered as *finalized*. The process ends when all the rules have been fulfilled.

b) **Gradient generation:** A *concentration* gradient is used to attract the wandering modules into unfilled position. The seeds act as sources which emit in all the neighbouring directions, a simulated chemical. The range of this emission can be controlled. The value of the gradient will be propagated using message passing between neighbouring modules. The non-source modules will compute the concentration of the gradient at their position using the following formula:

$$c_{module} = \max_{i \in R}(c_i)$$

where R is the set of received values from the neighbouring modules.

In order to avoid unnecessary moves to locate the sources, a *vector* gradient (VG) is used. The value of the gradient will be made locally available by computing VG as illustrated on figure 1.

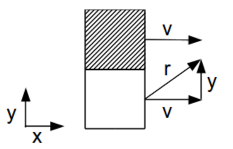

Fig. 1: The computation of the vector gradient. The considered module is the white one. The hatched module is the neighbour with the maximum concentration. v is the vector gradient of this module and x and y forms a regular base. r is the resulting vector for the considered module: $\vec{r} = \vec{v} + \vec{y}$

c) **Connectivity check:** One strong constraint in the self-reconfiguration process is to maintain the connectivity of the structure. Disconnection during the process might lead to falling module and thus damaged hardware. Moreover, disconnected groups of modules might get formed. These groups will not be able to reconnect, leading to deadlock situations. Since the different modules can move asynchronously and simultaneously, some rules are required to ensure the connectivity of the structure. The only modules which are static are the finalized one, which form a connected structure. They can thus become the sources of a new gradient, called the *connection* gradient (CG). This gradient is propagated the same way as the concentration one. The following set of rules is introduced to define when a module can move without any risk of connectivity break:

- The concentration of the CG in the module and its neighbours is strictly greater than zero. Indeed if the concentration of the CG in the module is equal to zero it is considered as a wandering module, i.e. a module that is currently moving.
- The fact of moving this module doesn't change the CG in the neighbouring modules. This means that the module has no influence on the connectivity of the substructure.
- Module is not a source.

[6] proves by induction that these rules are sufficient. Using this checking procedure, several modules are allowed to move at the same time. The only strong limitation introduced by this connectivity constraint is that sources cannot be removed from the structure during the process. As a consequence locomotion through reconfiguration is impossible.

3) Experiments: In order to perform the experiment in a simulated environment, the modules were considered as perfect cubes able to move in a regular 3D grid (lattice system). Each module has 6 hermaphrodite connectors and can sense its neighbours. It can also freely slide over the surface of the structure and around neighbouring units. The simulated system is thus more powerful than current hardware. Furthermore the connection/disconnection sequences are not considered. During each time step, the module does the following:

- Process received messages.
- Send messages to neighbours.
- Move if possible.

The experiments consisted of making an initial squared structure to reconfigure into a disk and then into a sphere. The experiments illustrate the almost linear dependency between the reconfiguration time and the number of modules in the structure. The evolution of the total number of moves was also shown to be faster than linear. In all the cases the system converged to the desired shape. Finally, the majority of local messages was used to propagate gradient in the structure.

4) Conclusion: K. Stoy [6] presented a new approach in the domain of self-reconfigurable modular robots based on a cellular automaton to represent and generate the final structure and several gradients to guide the modules into the final position.

One of the strong contributions of the article is the development of a complete framework for performing reconfiguration: the desired structure is created using a CAD software and can be directly converted into a configuration to be reconfigured into. The use of a scaffolding structure ensures the convergence of the process.

One weak point of this work is the lack of hardware consideration. The algorithm uses a perfect model of a module without taking into account the connection/disconnection procedure. The case of two modules trying to fill the same position has also been eluded. Moreover the message passing between the units is considered perfect, without any loss. Finally, theoretical analysis is missing for the convergence induced by the scaffolding method.

In the next section, we describe a new framework developed by Fitch et al [8] to tackle the reconfiguration problem.

C. Second approach: planning strategy

Reconfiguration planning can be defined has the problem of finding the sequence of module moves to go from a configuration A to a configuration B.

In [8] the authors developed a flexible reconfiguration framework allowing the use of different kinematic models. In this article, the main idea is to represent the reconfiguration problem as a path planning problem directly inside the kinematic action space of the considered modules. This work is a follow up of a previous paper [19] where they developed their Markov Decision Process (MDP) formalism, not in the native kinematic space but for an abstract model of sliding cubes. Many authors use the concept of meta-module (a group of two or more modules assembled together) to reduce the number of kinematics constraints in the problem. The authors [8] have chosen not to use MM to exploit the possibility of dynamic grouping during the reconfiguration process. A method taken from the field of reinforcement learning (MDP) is used to represent their path planning problem and solve it using dynamic programming [21]. A *navigation* function is defined and updated as modules move. The module kinematics will be implemented through the *transition* function of the MDP. The algorithm allows locomotion through reconfiguration: the goal shape is made of convex or non convex elements and the modules move to fill this shape, which can then be shifted. This framework also take into account obstacles in the way.

The MDP planning is composed of two main elements: a connectivity checking procedure and the actual planning using a global navigation function. We first describe the connectivity checking method and the formulation of the planning problem as a MDP. Finally we present how the MDP has to be modified to integrate the module kinematics.

1) Connectivity graph: Before a module is allowed to move, we have to ensure that it remains connected with the main structure. In graph theory, the notion of articulation nodes fits perfectly with this situation. An articulation node in a graph is defined as a node which removal would lead to a disconnected graph. It would then seem natural to check for the "articulation modules" before moving and to consider them as locked. The main problem with this technique is that checking for simultaneous removal of module inside the structure is much more challenging. The author propose a local method based on the definition of *connecting cycles*. For each potential moving module, a *connectivity graph* composed of the adjacent modules is built. Then a local search is done to find existing path between all the nodes of the connectivity graph without including the considered module. If a path exists between them these nodes form a connectivity cycle and the module can be moved. The depth of the search is fixed at the beginning but can be increased if required. The overall process relies on a message passing procedure between adjacent modules. The modules along the path of a moving module are locked. This locking corresponds to a synchronisation of the modules to prevent collisions. If two modules want to fill the same position, the moving one is chosen at random. Since this process is local, many modules can move asynchronously at the same time.

2) Planning using Markov Decision Process: The planner is based on a *value function* updated continuously to take into account topological changes within the structure. This function is used to globally guide the modules towards the final configuration. A general MDP is a sequential decision making method composed of four main elements:

- A state set S: all the possible states of an agent.
- An action set A: all the possible actions that can be taken by an agent.
- A transition function T: a function mapping the state-action space into the state space.
- A reward function R: a function mapping the action space into $\mathbb{R}$ or $\mathbb{N}$.

Most of the time, the goal of an agent is to find the set of actions (known as the *policy*) that lead to maximum reward. The transition function can either be deterministic or stochastic, known or unknown. If T is known, then the MDP can be solved in polynomial time [22] in the number of states using dynamics programming.

In the case of the abstract module representation (sliding cubes model), S corresponds to the set of faces, A was composed of two actions (see Fig. 2), and the reward was -1 for each move (the best policy will tend to favour fewer moves). The value function is stored in a distributed fashion. Each module only stores the value of its connectors and update it using the message passing process when a neighbouring module state changes.

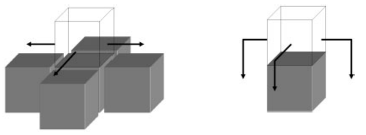
Fig. 2: The action space A for the abstract modules (adapted from [19]).

3) Module kinematics: The main novelty introduced by Fitch et al [8] is the use of module kinematics in a built-in fashion inside the previous MDP formulation. As a consequence the complexity and convergence analysis will directly applied to the modified formulation. The MDP will be adapted as follows. The abstract state set and action set are replaced respectively by the real possible joint angles and the connector state. The new state space is determined by the *transition function*: if the state is reachable, then it will be added to S.

The new action space is based on the kinematic model of the robots. The actions are generated iteratively by incrementing the different joint angles of the module. More precisely the following algorithm is used, for a single module move:

1) Given the actual state of the module, its lattice position, the value of its joints, use forward kinematics to compute connectors position.
2) Iteratively generate the set of actions:
    - Permute the degree of freedom (i.e. increment or decrement their value of a multiple of $\frac{\pi}{2}$)
    - If no connection is possible, the configuration is discarded
    - Otherwise a collision checking is performed

Some moves require the use of two modules (for example, when a convex edge needs to be overcome using Roombots modules). The additional required module is called an *helper* module. In this case the previous algorithm is modified by considering the joint angles of both modules. Since this algorithm is exponential in the number of considered joint angles it is more suited for lattice systems.

4) Conclusion: Fitch et al [8] present a reconfiguration framework based on a path planning method directly into the kinematic space of the considered modular robotic units.

The contribution of this paper is twofold. Firstly, the authors have refined the usual sliding cube abstraction to directly include the kinematic constraints of the robots. Their framework, beyond the fact of being more realistic in terms of hardware representation, allows the use of virtually any modular platforms which kinematics model is known. Secondly they manage to provide a strong theoretical justification and analysis of the problem, showing the expected complexity of the reconfiguration process using their algorithm. By using a Semi Markov Decision Process, they ensure that the Markov property holds, i.e. that the future states of the system will only depend on its present state.

One of the main weaknesses of this article is the lack of real hardware experiments. The authors use ”hardware in the loop” composed of communication and computation boards to simulate the distribution of tasks, the message passing and the actual computational power of a real modules (see [23]). Nevertheless, the complexity of the reconfiguration process often comes from the mechanical parts (backlash, elasticity effects, ...) and the connection/disconnection procedure (misalignment, incomplete connection, ...). The algorithm does not take into account possible failures of modules that might then be obstacles during the process. Loss of messages and corrupted data are also ignored. Finally, the complexity of the algorithm for generating the action space (which is in fact a brute force approach) might become restrictive when dealing with chain or hybrid type modular robots.

III. Research Proposal

At BioRob (EPFL) a homogeneous, self-reconfiguring modular robotic platform named Roombots (RB) is being developed (see Fig. 3 for an illustration). Roombots are designed as basic units for adaptive furniture able to move, self-reconfigure and self-repair. In this context, a reconfiguration framework would be required to allow assembly of RB units into furniture-like structures. Moreover, since the project is targeting everyday life environment, interaction with end users need to be apprehended.

In this section we first present the robotic platform Roombots. We then describe the two main aspects of our work, namely the control algorithms and the interaction strategies we have already implemented. We explain the possible improvements we are planning to integrate and the ways we would like to explore.

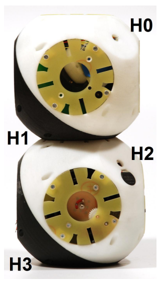

Fig. 3: A roombots module

A. Platform

The RB modules are composed of two cube-like parts with 110 mm edge length (Fig. 3). Each module has three continuous rotational degrees of freedom (depicted in Fig. 5a) and up to ten active connection mechanism (Fig. 5b). This four-way symmetric connection mechanism is based on a fully retractable latching mechanism. Using these hermaphrodite connectors, the modules can connect to each other to form more complex structures. Two RB modules assembled together using the connectors on the outer hemispheres ($H0$ and $H3$ in Fig. 3) form a *metamodule*. Four connection types can be defined (see Fig. 4), inducing different kinematic properties and motion capability. A RB module weights about 1.4 kg, with batteries included. The estimated autonomy of one module in continuous actuation is 30 min. Any joints in the RB module can deliver sufficient torque to lift an horizontally stretched metamodule.

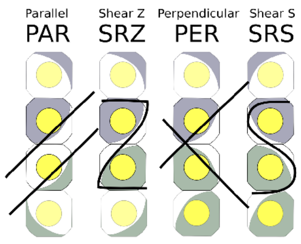

Fig. 4: The four different types of metamodule connection (adapted from [24])

B. Control

This section describes first the current implementation of our reconfiguration framework for the RB platform. We took inspiration from the work done by K. Stoy [6] and developed a gradient based approach to solve the reconfiguration problem.

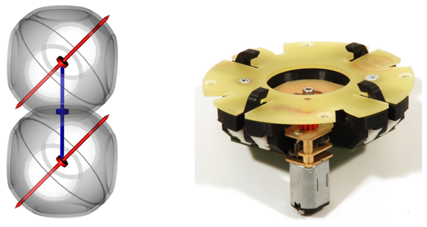

Fig. 5: In (a) the three degree of freedom of a RB module are depicted. (b) shows the current design of the Active Connection Mechanism (ACM).

We provide here a summary of the main steps of the methods followed by suggested improvements and their expected influence. A more detailed description of our framework can be found in [9] and [25].

1) Current implementation: The problem that we are trying to tackle is the reconfiguration of several metamodules (MM) into a final shape. This reconfiguration through locomotion takes place in a structured environment, i.e. with embedded connectors in floor and ceiling walls to which modules can attach. MM are the basic units considered in our case. They are guided towards their final position using a force field approach. MM are able to broadcast messages between each other to acquire the necessary knowledge about their surroundings (neighbours, obstacles, ...). To perform the basic moves leading to the final position, a look-up table composed of shape-transitions (servo angles) is used by the MM along with a precomputed collision cloud to avoid self-collision and collision with other MM. The moves are done in a fully asynchronous fashion, allowing several MM to reconfigure at the same time.

a) Metamodule shape and initialization: At the beginning of the algorithms the metamodules are randomly placed on a $2D$ structured environment made of passive connectors. The MM are attached by their foot connector. They are restricted to be in five different shapes during the reconfiguration process: I, L, S, U and $3D-S$. The servos angles representing the transition between these shapes are stored in a database and used when needed by the MM.

b) Reconfiguration through locomotion: During the reconfiguration process, the MM go from one shape to another using the transition database. They use only two active connection mechanisms (one at the top of the MM and one in the foot) which are alternatively connected to the ground. After each move the MM checks the current state of its neighbourhood and requests the precomputed collision cloud corresponding to the desired shape to shape transition. It also broadcasts its position to the neighbouring MM.

c) Seeding mechanism: In order to guide the MM during the reconfiguration process, goal position have to be defined. These final position will be the *seeds* of the shape and play the role of attractors. To ensure the feasibility of the building procedure, a bottom-up approach is imposed: different *level* are defined in the final shape and the corresponding seeds are only available when the seeds positions in the previous level have been filled. The seeding and levelling are provided by the user.

d) Gradient: In the framework, the MM knows its absolute position in the $3D$ grid as well as the position of the active seeds and the one of its neighbours. The MM can thus compute the vector force corresponding to the different seeds. The neighbouring modules are including in the computation as repulsive sources. Different approaches have been tested regarding the influence of these modules. In the first one, they do not have any influence and the modules tend to go straight to the seeds. The collisions are prevented by locking modules which are too close from each other before deciding which one should move first. The second approach consider a gradual decrease of the influence of the neighbour with the distance. Only the metamodules in the range of the considered MM will have an influence. Finally we have tested also an approach in which the MM are given the same influence in the all range of the considered MM. This was intended to minimize collision between MM by enforcing a kind of minimal distance policy.

e) Results: We performed several experiments in simulation using up to six metamodules. The goal shape was either a box-like structure (composed of four MM) or a chair (with six metamodules). We repeated the experiments with the four different types of MM to analyse their kinematic abilities. We counted the number of deadlocks situation (i.e. when the MM were not able to build the final shape), the amount of collisions and the overall number of moves. The most relevant observation we made was that metamodules of type SRS were more successful in average at building the considered structure than the three other types. This result needs to be further investigated taking into account the kinematic properties of the four type of MM.

2) Improvements: There are several ways we are currently exploring to improve our current reconfiguration framework.

a) Granularity adjustment: Previous studies [9] shows the type of meta-module used in the reconfiguration process plays a central role in the "buildable" aspect of the structure. The meta-modules should be able to switch between types, according the reachable space[1] that this new configuration allows. The reachable space is defined by the kinematic model of the meta-module. The switching sequence can be summarized in four main steps:

- The meta-module goes to a U-shape
- The inner *ACM* is disconnected
- The two outer spheres of both modules rotate and move to achieve the new type of connection
- The inner *ACM* is connected again

This ability of the MM to switch from one type to another might allow an optimization of the reachable space before each move and thus a reduction in the number of deadlock situations.

Similarly we are investigating the possibility of allowing the MM to disconnect and act as two separated module to overcome obstacles or avoid deadlocks. In [8] the authors mention the possibility of a dynamic connection into a substructure during the reconfiguration process to optimize it. We are studying a way to characterize this problem of *granularity adjustment* such that each MM can decide on the fly what its best configuration should be given the current neighbouring situation.

b) Building process:

1) Seeding procedure
During preliminary experiments, we have observed that both the *levelling* and the *seeds* choices have a significant impact on the building process. An exhaustive study of these two factors should be performed to clearly understand their influence. It would result in a better understanding of the pre-processing steps of seeding and levelling and possibly lead to a global improvement of the reconfiguration process.
2) Cooperation/Consensus based approach
In the current version of the framework the metamodules have limited communication range. Message broadcasting is used to manage seeds priority, possible collision issues, ... In [4] the different modules are able to adapt to a change in their environment by reaching a *consensus*. We would like to incorporate a similar mechanism in the case of variable granularity. The modules might need to cooperate to overcome convex edges or to ask for the help of a MM to optimize the building process (for example a metamodule could carry a module over an obstacle).

c) Hardware consideration: The main weakness of the current implementations for reconfiguration is too little consideration of hardware constraints of the underlying platforms. Through our first experiments using the RB platform, we have faced many problems intrinsically linked to hardware imperfection: misalignment of the connection mechanism due to gravity, elastic effects, backlash effects in the gear boxes, imprecision in the control loop, ...

In order to facilitate the transfer of our framework to the real hardware we are currently integrating several solutions to better fit the reality. Firstly we are implementing a new approach for using inverse kinematics to both optimize the collision cloud of the modules/metamodules but also to compensate for the imperfection in the hardware. These corrections will be provided by different sensors (accelerometer, force sensor and infra-red sensor) which data will be fused. We plan on considering a more straight-toward movement of the connector during the attachment procedure to avoid collision between modules.

As it was done in [8] we aim at developing a framework versatile enough to be used with different modular robotic platforms.

d) Use of meta-structure: We are using mainly metamodules as building block and basic units in the current version of the framework. In [26], the use of an optimal metamodule abstraction has been investigated and seems to simplify the planning process for the reconfiguration problem. We plan on allowing the modules to assemble in various struc-

[1]The reachable space can be defined as the number of connectors or cubes that can be reached by the head/tail *ACM* of the module. This indicator can be further extended to take into account the number of possible future collisions at the next step, for example.

tures to form the final *meta-structure* [2]. This should lead to a gain in time to completion for the process and to flexibility in terms of controllable structure. We could also relax some of the constraints regarding the structured environment: by allowing some complex assembled structures we can think of using locomotion controller on quadruped shape (for example) and then go back to more simple structures for the reconfiguration process.

C. Interaction

Throughout this report we have presented reconfiguration methods for modular robots mainly design for research purposes. Few works have been done on the use of modular robots in every day life environment.

One of the goals of the Roombots project is to allow non-expert user to exploit the possibilities of the modules, without being held back by the complexity of the tasks or of the robots itself.

1) Interface and recommender system:

a) Current interface: We have started developing an interface to ease the building of a structure made of RB modules. The first version of the interface allows the user to build a shape made of cubes and convert it in real time into a configuration made of Roombots modules. The conversion process is based on a perfect matching algorithm from graph theory. More details of this all project can be found in [27].

b) Improvements: Apart from the possible technical improvements presented in [27], some more high level considerations can be derived from this preliminary work. We are looking for a complete framework allowing the user to efficiently build structures, place them into her environment and have a preview of the reconfiguration process, with a quantification in terms of number of moves or time. Recommendations could be made to the user as a guidance during the building process. Modifications could be proposed to reduce the complexity of the desired structure. A complexity and a similarity measure (between structures) should thus be developed.

2) Interaction: We need to study the possible media that could be used to interact with the non-human like modular robotics platform we are using. We will based our work on the research on interaction and coordination between human and non-anthropomorphic robots in mixed robotics swarms (see [12] and references within). The use of *tangible* interfaces will also be explored.

References

[1] Biorob, “Biorobotics laboratory,” http://biorob.epfl.ch/roombots.
[2] F. Hou and W. Shen, “On the complexity of optimal reconfiguration planning for modular reconfigurable robots,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2791–2796.
[3] M. Yim, Y. Zhang, J. Lamping, and E. Mao, “Distributed control for 3D metamorphosis,” *Autonomous Robots*, vol. 10, no. 1, pp. 41–56, 2001.
[4] C. Yu and R. Nagpal, “Self-adapting modular robotics: A generalized distributed consensus framework,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE, 2009, pp. 1881–1888.
[5] H. Bojinov, A. Casal, and T. Hogg, “Multiagent control of self-reconfigurable robots,” in *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*. IEEE, 2002, pp. 143–150.
[6] K. Stoy, “Using cellular automata and gradients to control self-reconfiguration,” *Robotics and Autonomous Systems*, vol. 54, no. 2, pp. 135–141, 2006.
[7] C. Jones and M. Mataric, “From local to global behavior in intelligent self-assembly,” in *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 721–726.
[8] R. Fitch and R. McAllister, “Hierarchical Planning for Self-Reconfiguring Robots Using Module Kinematics,” 2010.
[9] A. Spröwitz, P. Laprade, S. Bonardi, M. Mayer, R. Möckel, P.-A. Mudry, and A. Ijspeert, “Roombots-Towards Decentralized Reconfiguration with Self-Reconfiguring Modular Robotic Metamodules,” in *Proceedings of IEEE IROS2010*, 2010, pp. 1126–1132.
[10] M. Asadpour, A. Sproewitz, A. Billard, P. Dillenbourg, and A. Ijspeert, “Graph signature for self-reconfiguration planning,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 863–869.
[11] K. Golestan, M. Asadpour, and H. Moradi, “A New Graph Signature Calculation Method Based on Power Centrality for Modular Robots.”
[12] S. Bonardi, “Human-robot interface for multi-robot systems coordination,” 2010.
[13] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, “Modular self-reconfigurable robot systems [grand challenges of robotics],” *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 43–52, 2007.
[14] A. Spröwitz, “Roombots: Design and Implementation of a Modular Robot for Reconfiguration and Locomotion,” Ph.D. dissertation, Lausanne, 2010. [Online]. Available: http://library.epfl.ch/theses/?nr=4803
[15] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, “M-TRAN: Self-reconfigurable modular robotic system,” *Mechatronics, IEEE/ASME Transactions on*, vol. 7, no. 4, pp. 431–441, 2002.
[16] B. Salemi, M. Moll, and W. Shen, “SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system,” in *Proc. 2006 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. Citeseer, 2006.
[17] V. Zykov, A. Chan, and H. Lipson, “Molecubes: An open-source modular robotics kit,” in *IROS-2007 Self-Reconfigurable Robotics Workshop*, 2007.
[18] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson, “Evolved and designed self-reproducing modular robotics,” *Robotics, IEEE Transactions on*, vol. 23, no. 2, pp. 308–319, 2007.
[19] R. Fitch and Z. Butler, “Million module march: Scalable locomotion for large self-reconfiguring robots,” *The International Journal of Robotics Research*, vol. 27, no. 3-4, p. 331, 2008.
[20] H. Bojinov, A. Casal, and T. Hogg, “Emergent structures in modular self-reconfigurable robots,” in *Robotics and Automation, 2000. Proceedings. ICRA’00. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1734–1741.
[21] A. RONALD, “Dynamic Programming and Markov Processes,” 1960.
[22] M. Littman, T. Dean, and L. Kaelbling, “On the complexity of solving Markov decision problems,” in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Citeseer, 1995, pp. 394–402.
[23] R. Lal and R. Fitch, “A hardware-in-the-loop simulator for distributed robotics,” in *Proc. of ARAA Australasian Conference on Robotics and Automation (ACRA)*, 2009.
[24] M. Mayer, “Roombot modules - kinematics considerations for moving optimizations,” 2009.
[25] A. Spröwitz, S. Pouya, S. Bonardi, J. van den Kieboom, R. Möckel, A. Billard, P. Dillenbourg, and A. Ijspeert, “Roombots: Reconfigurable Robots for Adaptive Furniture,” *IEEE Computational Intelligence Magazine, special issue on ”Evolutionary and developmental approaches to robotics”*, vol. 5, no. 3, pp. 20–32, 2010.
[26] D. Dewey, M. Ashley-Rollman, M. De Rosa, S. Goldstein, T. Mowry, S. Srinivasa, P. Pillai, and J. Campbell, “Generalizing metamodules to simplify planning in modular robotic systems,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1338–1345.
[27] S. Bonardi, “Graphical interface for building roombots structures,” 2010.

[2]We use the word ”meta-structure” to indicate that the final configuration is made of different and non-homogeneous structure.