Control and interaction strategies for self-reconfigurable modular robots

THÈSE N^{O} 6277 (2014)

PRÉSENTÉE LE 26 SEPTEMBRE 2014

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE BIOROBOTIQUE

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Stéphane BONARDI

acceptée sur proposition du jury:

Prof. B. Faltings, président du jury
Prof. A. Ijspeert, Prof. P. Dillenbourg, directeurs de thèse
Prof. M. Yim, rapporteur
Prof. A. Martinoli, rapporteur
Prof. K. Stoy, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2014

Acknowledgements

FIRST and foremost, I would like to express my sincere gratitude to my advisor, Professor Auke Ijspeert, for his guidance and his complete and unconditional support throughout my PhD as well as for sharing with me his broad knowledge of the field of robotics. I will always be thankful to him for giving me the opportunity to conduct such exciting research and for creating such a wonderful work environment. I wish also to thank Professor Pierre Dillenbourg, who accepted to be my co-supervisor and who gave me precise pointers regarding the topic of interaction, that was not really familiar to me.

It would have been impossible to conduct this research without the help and support of the many people around me, who contributed in their own ways to this work, and to only some of whom it is possible to give particular mention here. I had the chance of having Alexander Spröwitz and Soha Pouya as my first supervisors during my internship project at Biorob, and afterwards as my colleagues, and their scientific rigor and curiosity have been inspirational to me. I latter had the opportunity to closely collaborate with Rico Möckel, who has always been of great help and who always impressed me with his patience and professionalism. I had also the great chance of sharing my work with my colleague Massimo Vespignani who, despite facing his own difficulties, provided me with unconditional help and support. I would not have been able to carry out this work without his support throughout those years. I would also like to thank all my Biorob colleagues (past and present), that have always been supportive and interesting to discuss and interact with. I'm also grateful to all the students I supervised, for their hard work and their help in making progress on this challenging topic of self-reconfigurable modular robots.

I would like to thank the members of the jury who kindly accepted to review this manuscript: Prof. Boi Faltings, Prof. Alcherio Martinoli, Prof. Mark Yim, and Prof. Kasper Stoy.

Last but not least, I would like to thank my family for their support throughout my degree. I especially thank my mom, dad, and sister. My hard-working parents have sacrificed a lot for my sister and myself and provided unconditional love and care. I would not have made it this far without them.

Lausanne, 21st of July 2014 Stéphane Bonardi

Abstract

Research on Self-Reconfigurable Modular Robots (SRMRs) has steadily increased during the past decade. Their ability to change shape dynamically to adapt autonomously to their environment combined with their inherent versatility and their robustness through redundancy make them potentially well suited for a large variety of tasks. For example, the SRMR Roombots from the Biorobotics Laboratory (EPFL, Switzerland) has been developed with the goal of creating assistive and adaptive furniture able to locomote and self-adapt in everyday life environments. This thesis contributes to the field of SRMR by designing algorithms and devising strategies that address three major problems in the domain: self-reconfiguration, locomotion, and user-interaction.

Despite significant efforts conducted in the domain of self-reconfiguration (SR), the current approaches often rely on high level abstractions of the problem and perfect theoretical models of the active units, neglecting the issues of bending and connection misalignment, making the transfer of the considered method to the hardware platforms difficult, if not impossible. Moreover, the constructed structures can only be comprised of active modules (often of the same type) instead of both active and passive units (i.e. units that possess no actuation capability and that are only equipped with passive connectors compatible with the active units), which tends to reduce the range of shapes that can be built using SRMRs. Taking into account these limitations, we first propose incremental modifications of existing techniques to address the SR problem. We extend the state of the art by proposing a novel hierarchical approach that allows the integration of fully passive elements and that computes hardware friendly movements which take into account torque limitation. We explore different ways of characterizing and compensating some of the hardware imperfections such as the bending effects observed in many materials and the alignment error during the connection and disconnection phases.

The ability of SRMRs to rapidly change their morphology make them a suitable tool to study locomotion learning for various topologies. Methods using gait tables have been widely used to manage predefined changes of topology but they cannot deal with unguided self-reconfiguration, where the final structure into which the set of robots reconfigures into is unknown beforehand. We propose a new algorithm that relies on the detection of bio-inspired patterns in the structure combined with the use of symmetries to create a reduced control network that allows a fast convergence towards a reasonably efficient gait in terms of internal collision and forward speed. The Central Pattern Generator (CPG) network used for the locomotion control offers additional robustness and smooth transition between gaits. We demonstrate that our approach significantly outperforms a fully open control network by a

factor of up to 10 in the first 30 iterations, making it particularly well suited for time critical tasks in unknown environments.
With the steady integration of robots into everyday life environments, the question of the interaction strategies and modalities becomes a central one. SRMRs bring the additional challenges of an evolving morphology, both on-grid and off-grid, and a lack of anthropomorphic features. Classical interfaces often confine the user to use a fixed device such as a PC to design a desired shape or to control a group of robots. In order to allow non-expert users to exploit the full potential of SRMRs, we introduce more natural ways of interacting with a group of SRMRs by abstracting away the complexity of SR and locomotion learning through high level interaction strategies. We develop both a tablet-based interface in which the user can arrange virtual structures made of SRMR in an augmented reality representation of a room and a device-free interface based on the principle of embodied interaction in which the user is tracked by external depth sensors and use pointing gestures to control groups of robots. Additional feedbacks are given to the user via visual lighting of the grid setup and of the modules themselves.

Key words: Self-reconfigurable modular robots, self-reconfiguration, locomotion, interaction strategy, user-interfaces

Résumé

Les Robots Modulaires Auto-Reconfigurables (RMARs) n'ont cessé de se développer au cours de la dernière décennie. Leur capacité à changer de forme pour s'adapter à leur environnement de manière autonome combinée à leur polyvalence et leur robustesse les rendent potentiellement bien adaptés pour une grande variété de tâches. Parmi eux, les robots Roombots, développés au Laboratoire de Biorobotique (EPFL, Suisse), ont été créés dans le but de construire des meubles assistifs et adaptatifs, en mesure de se mouvoir librement et de s'adapter à des situations et des environnements de la vie quotidienne. La contribution de cette thèse au domaine des RMARs réside dans le développement d'algorithmes et de stratégies qui répondent à trois problématiques majeures : l'auto-reconfiguration, la locomotion, et l'interaction avec l'utilisateur.

Malgré les efforts importants déployés dans le domaine de l'auto-reconfiguration (AR), les approches actuelles se basent souvent sur des abstractions de haut niveau du problème et considèrent des modèles théoriques parfaits des modules, en négligeant les questions de flexion des matériaux et les erreurs d'alignement du mécanisme de connexion, ce qui rend le transfert de ces méthodes vers les robots réels difficile, voir impossible. En outre, les structures construites ne peuvent être constituées que de modules actifs (souvent du même type), au lieu d'unités actives et passives (c'est à dire d'unités qui ne possèdent pas d'actuateurs, et qui ne sont équipées que de connecteurs passifs compatibles avec les unités actives), ce qui tend à réduire la gamme de formes qui peuvent être construites en utilisant les RMARs. Compte tenu de ces limitations, nous proposons tout d'abord des modifications à des techniques existantes pour résoudre le problème de l'AR. Nous étendons l'état de l'art en proposant une approche hiérarchique qui permet l'intégration d'éléments entièrement passifs et qui calcule des mouvements respectueux du système mécanique en prenant en compte la limite de couple des moteurs. Différentes manières de caractériser et de compenser les imperfections des robots réels, tels que les effets de flexion observés dans de nombreux matériaux, et l'erreur d'alignement au cours des phases de connexion et de déconnexion, sont abordées.

La capacité des RMARs à changer rapidement de morphologie en fait un outil approprié pour étudier l'apprentissage de la locomotion pour différentes topologies. Les méthodes utilisant les tables de paramètres de locomotion pré-calculés ont été largement utilisées pour gérer les changements de topologie prédéfinis, mais elles ne peuvent pas s'appliquer lors d'une auto-reconfiguration arbitraire, où la structure finale en laquelle l'ensemble des robots se reconfigurent, n'est pas connue à l'avance. Nous proposons un nouvel algorithme s'appuyant sur la détection de sous-structures bio-inspirées combinée à l'utilisation des symétries afin

de créer un réseau de contrôle plus optimal qui permet une convergence rapide vers une démarche assez efficace en termes de collision interne et de vitesse d'avancement. Les Central Pattern Generators (CPGs) sont des réseaux d'oscillateurs couplés que nous utilisons pour le contrôle de la locomotion, car ils offrent une robustesse supplémentaire et une transition continue entre les différentes démarches. Nous démontrons que notre approche surpasse de manière significative celles utilisant un réseau de contrôle entièrement ouvert, ce qui la rend particulièrement bien adaptée pour les tâches à forte contrainte de temps dans des environnements inconnus.
Avec l'intégration progressive de robots dans des environnements de la vie quotidienne, la question des stratégies et des modalités d'interaction devient centrale. Les RMARs apportent un défis supplémentaire à cause de leur morphologie évolutive, et par leur manque de caractéristiques anthropomorphiques. Les interfaces classiques obligent souvent l'utilisateur à utiliser un dispositif fixe, comme un ordinateur de bureau, afin de concevoir une forme désirée ou pour contrôler un groupe de robots. Afin de permettre aux utilisateurs non-experts d'exploiter pleinement le potentiel de RMARs, nous introduisons dans cette dissertation des moyens plus naturels d'interaction avec un groupe de RMARs, en nous affranchissant de la complexité de l'AR et de l'apprentissage de la locomotion par l'intermédiaire de stratégies d'interaction de haut niveau. Nous développons à la fois une interface pour appareils mobiles dans laquelle l'utilisateur peut organiser des structures virtuelles faites de RMARs dans une représentation en réalité augmentée d'une pièce, et en proposant une interface basée sur le principe de l'interaction directe dans laquelle l'utilisateur est suivi par des capteurs de distance externes et utilise des gestes pour contrôler des groupes de robots. Des informations supplémentaires concernant l'état du système sont fournies à l'utilisateur grâce à un éclairage des connecteurs placés dans l'environnement et des degrés de liberté des modules eux-mêmes.

Mots clefs : Robots modulaires auto-reconfigurables, reconfiguration, locomotion, stratégies d'interaction, interfaces utilisateur

Contents

General Introduction

Objectives of this dissertation

THE world of robotics has evolved dramatically over the last decade. Robots have seen their capabilities increasing, both in terms of mechanics and electronics but also in terms of control. A growing number of robots are no more limited to lab spaces and are being designed to be integrated in every day life environments. They should provide services, help, and support to a wide variety of end-users, ranging from young children to elderly, all of them having specific needs. These robots appear in many shapes and orders of complexity, from the very advanced humanoid robots, such as Asimo [110], able to walk, run, and manipulate objects, to the simpler vacuum cleaner robot Roomba [123], limited to a specific task. But the complexity of these robots is often linked to their cost, which confines the most advanced ones to lab's environments. They are also often specialized into carrying out a specific set of tasks, such as manipulating objects or exploring unknown environments. More and more robots are being developed to support humans, such as the Keepon [145] robot, used for example as an helper therapy for autistic children, or the RI-MAN robots [192], designed to carry patients from their bed to their wheel chair. But these robots suffer from their high level of specialization into a specific domain and are lacking the ability to adapt to the task to be performed.

As opposed to this rise in complexity trend, the domain of reconfigurable modular robots has emerged as a potential solution. Reconfigurable modular robots are simple interchangeable units able to assemble to form a more complex structure to solve various more complicated tasks. Among them, Self-Reconfigurable Modular Robots (SRMRs) are equipped with active connection mechanisms allowing them to dynamically change shape to adapt to the user needs or to the task to be performed.

The SRMR Roombots developed at the Biorobotics laboratory (EPFL, Switzerland) has been designed to study three major challenges: (i) When being configured in chain or lattice structures we use RB modules as a rapid prototyping set to study distributed locomotion control in unknown terrains. (ii) The self-reconfiguration (SR) capabilities of RB support the exploration of algorithms for self-organization, self-optimization and collaboration between modules. (iii) The name "Roombots" refers to our goal of creating self-reconfigurable adaptive furniture, i.e. furniture that can move and change shape thanks to reconfiguration using

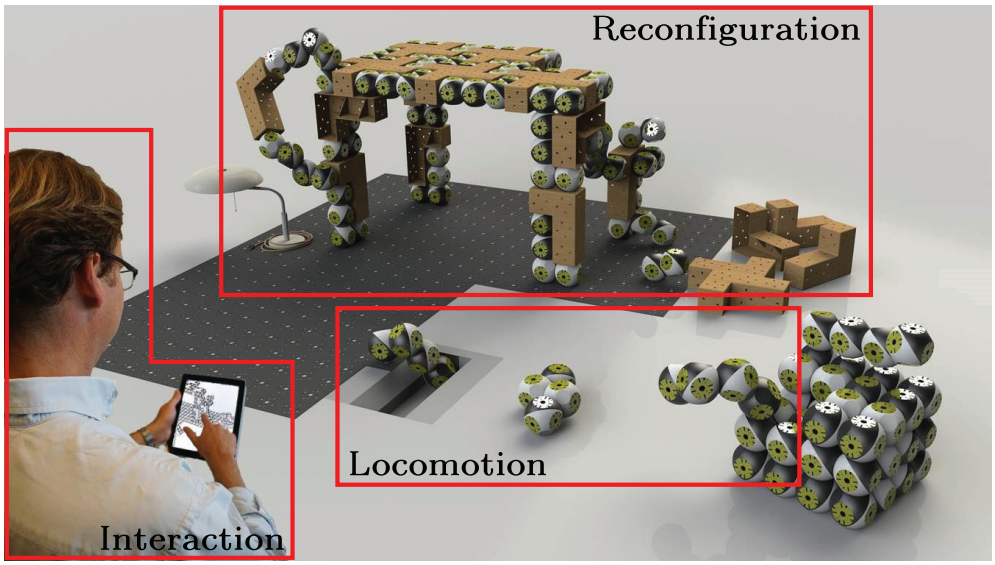


Figure 1 – Rendered picture representing the different aspects of the Roombots project. On this illustration, a table is being constructed out of active modules and passives elements (wooden color) evolving on a 2D grid (in dark grey). A set of modules is located out of the grid and metamodules separate from the main group to perform off-grid locomotion. They reattached to the grid using a sink mechanism included in the ground. A user is controlling the process using a tablet device (illustration adapted from [238]).

dynamic connection mechanisms. RB are made for building reconfigurable living and working environments that adapt to the current needs of human beings. Different research aspects linked to the Roombots project are illustrated in Fig. 1.

In their current development state, SRMRs are still limited in terms of mechanical capabilities (such as torque and weight support), connection mechanism efficiency (regarding the tolerance to misalignment and the connection strength), and their scalability to a large amount of modules, with issues such as communication delays or module variability (inherent to the building process). These limitations are seldom taken into account in the theoretical contributions on locomotion and self-reconfiguration, making the transfer to the hardware platforms tedious, if not impossible.

This thesis contributes to the field of SRMR by designing algorithms and devising strategies that address three major problems in the domain: reconfiguration, locomotion, and user-interactions. The main contributions of this dissertation can be summarized as follows:

1. In the domain of **self-reconfiguration**, we propose novel approaches **tightly linked to the hardware constraints** to ease the transfer to the hardware platforms. We extend the SR to heterogeneous structures composed of modules and fully passive elements to accommodate for the heterogeneous aspects we envision with the assistive and adaptive furniture of the RB project.

2. We describe efficient techniques to **quickly relearn locomotion** parameters after encountering an unknown event during a time critical task.

3. We introduce three novel **interaction strategies** that **abstract away the complexity** of SR and locomotion learning and allow non-expert users to intuitively and naturally interact with groups of modular robots.

In order to automatically change shape, SRMRs need a planner for finding the sequence of moves and connection/disconnection actions to go from a starting configuration to a goal configuration. One of our objectives is to develop simple generic algorithms (able to be used with a large number of hardware platforms) to perform **self-reconfiguration** with homogeneous sets of robots. We would like to additionally incorporate passive elements (i.e. units without actuation) into this SR process, by providing a manipulation framework for modular robots. Through this mean, we aim at improving the properties and the range of structures that can be built by SRMR. One critical requirement that we imposed on our techniques is the **close relationship to the hardware constraints**. In contrast with many existing approaches, we want to **reduce the reality gap** between simulated solutions and their porting to the hardware by incorporating built-in torque checking directly into our algorithm and by carefully characterizing in hardware the connection phase and the bending effects of our SRMR Roombots.

When a group of SRMRs have assumed a given shape, they also have the ability to go off-grid to locomote freely in the environment. We would like to fully exploit the morphology-changing capability of the SRMRs to perform **locomotion** or exploration tasks in unknown environments but also in environments that can be dynamically modified by the user (home environments for example). We want to provide **locomotion schemes** for fast gait generation to handle rapidly changing configurations, these new morphologies being unpredictable because either they have been generated on the fly by the user or independently by the robot to cope with changes in the environment or in its own structure (after hardware failures for example).

The goal of having SRMR into every day life environments requires the development of **natural and intuitive interfaces** to be able to use the reconfiguration and the locomotion techniques in a transparent way. We aim at proposing new ways of interacting with these non-anthropomorphic platforms. In our approaches, we would like to place the end-user as the center of the interaction by abstracting away the complexity of the control techniques inherent to SRMRs.

Thesis outline

This dissertation is composed of four main parts.

The first part of this thesis (part I) introduces general concepts about modular robots. We start by describing usual classifications used for the existing hardware platforms (section

1.1) along with challenges in the domain of SRMR (section 1.2). In chapter 2, we describe the main software and libraries that we used to validate our work and present in details the experimental hardware platform Roombots (RB), that we used to apply our algorithms and to characterize imperfections in the mechanics. Additionally we present the different hardware studies we conducted in order to reduce the reality gap between the simulated results and the hardware platforms, regarding the connection mechanism and the bending effects. A study of exact inverse kinematic solutions for the Roombots platform is presented in Appendix A.

In a second part (part II), we describe in details the different approaches that have been introduced to solve the SR problem and present the advances we made to the state of the art. We start in chapter 3 by precisely defining the problem we are tackling (section 3.1) as well as the related representations (section 3.1.1) and metrics (section 3.1.2). We introduce complexity notions (section 3.2.1 and 3.2.2) and analyse how the fact of using group of modules or metamodules impacts the performances of the reconfiguration algorithm (section 3.2.3). We then present in chapter 4 existing methods to solve the SR problem, both heuristic based (section 4.1) and exact (section 4.2). We describe the advances we made for both types of approach (subsection 4.1.3 and subsection 4.2.3 respectively). After concluding that no method exists to solve the problem of heterogeneous self-reconfiguration without relying on specialized passive elements or modules, we propose our own approach (chapter 5) and emphasize its strong coupling with the constraints imposed by many hardware platforms, such as the torque limitation. It should be noted that the triggering of the reconfiguration process using sensors information or behavioral analysis has not been considered in this work.

We investigate in a third part (part III) different techniques to provide an efficient and robust locomotion capabilities for SRMRs, both on-grid (chapter 6) and off-grid (chapter 7).

The aspect of interaction with SRMRs is treated in part IV. We describe the advances we made in the domain of mobile interfaces for a group of robots (chapter 8) and when considering a fully embodied interface without external device (chapter 9). More preliminary work on computer interfaces used to build arbitrary shapes and convert them automatically into structure made of modular robot is presented in the appendix C along with a review of the existing approaches to create interfaces for mixed team of humans and robots (appendix E).

Part I Preliminaries

1 Background on Modular Robots

Self-reconfigurable 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, such as planetary exploration or disaster relief scenario (where human access is impossible), 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 theoretically increased. In practice, SRMRs suffer from the multiplication of the possible points of failures, e.g. the connection mechanism or the interconnections between the degrees of freedom (that induces bending effects on the entire module). Some of these limitations are linked to the constraints of weight and compactness imposed on the system to allow one module to carry one or more units autonomously, preventing the use of more robust material and larger connection mechanisms.

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 [279].

1.1 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 [245].

First of all, modular self-reconfigurable robotic systems can be composed of replicas of 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 might decrease its robustness and adaptability. Nevertheless, the gain in terms of performance might compensate for this aspect, keeping in mind that the set of basic units should also be able to perform the same 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 talk 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, *hybrid* architectures are able to use both lattice and chain structures, 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 size bigger than the centimeter scale). 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.

More than 80 different reconfigurable modular robotics platforms have been created during the past 25 years [238]. From a hardware perspective, 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,...). Among others, well known platforms are the M-TRAN [183], the Superbot [222] and the Molecubes [291].

1.2 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 [292]. Planning algorithms able to control millions of abstract modules have been introduced [88]. 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 robotic units are often task 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 copies 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 module 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-purpose set of modular robots, efficient algorithms should be developed to determine the optimal shape for a given task.

2 Software and hardware platforms

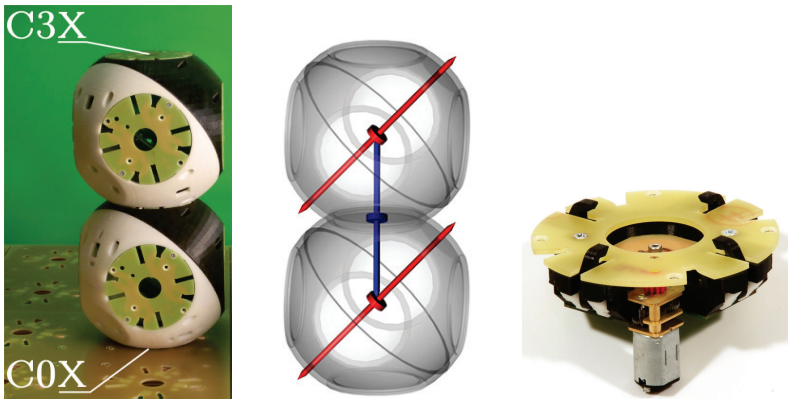
2.1 Simulation environments

In order to simulate our locomotion experiments, we used the physics based simulator Webots, developed by Cyberbotics [275]. Webots is used to model, program and simulate mobile robots in shared and complex environments. Webots is based on the Open Dynamics Engine (ODE) physics engine [233] that allows for realistic and accurate physic simulations. This simulator possesses several features that makes it well suited to simulate reconfigurable modular robots. Among others, Webots offers a large number of possible scripting languages and the possibility to write a specialized physics plugin to extend the capability of the software. Additionally, different models can be easily built using the graphical interface that supports also the direct importation of CAD shapes (for example, to provide precise collision checking).

The main drawback of this simulation platform for modular robots is the lack of scalability when considering the number of connectors that can be handled during one simulation run. This limitation is mainly impacting reconfiguration experiments when we increase the number of modules. In order to alleviate this limitation, we also implemented our own simulation environment based on Bullet Physics and Open Scene Graph (presented in chapter 5).

2.2 Experimental platform: Roombots

Roombots (RB) are self-reconfigurable modular robots developed at the Biorobotics laboratory (EPFL, Switzerland) with the ambitious goal of creating assistive and adaptive pieces of furniture using the reconfiguration and locomotion capabilities of SRMRs. Among the existing modular robots, only a small subset incorporates a mechanism for self-reconfiguration that allows modules to autonomously connect and disconnect like RB since the design of a mechanism for self-reconfiguration in a compact way is already a challenge on its own [283]. RB are designed with the property that a single module can autonomously travel through self-reconfiguration to any position on a 2-dimensional grid by a sequence of attachments



(a) (b) (c)

Figure 2.1 – A single RB module (a). The three degrees of freedom of a RB module (b). The current ACM design (c).

and detachments between the modules' connection mechanism and the grid structure (i.e. panels with regularly spaced connectors) and to overcome concave edges in 3 dimensions with a minimum number of three degrees of freedom (DOFs). Contrary to RB, M-TRAN [180], Molecubes [293], and ATRON [195] robots require more than a single module to change their direction of motion on a 2D grid. More than one RB module is only needed to overcome convex edges in 3-D configurations.

A Roombots module is composed of four half-spheres (see Fig. 2.1a for precise shape) linked together using revolute joints with continuous rotation capabilities (depicted on Fig. 2.1b). Using four-way symmetric compact Active Connection Mechanisms (ACMs, up to 10 per module, illustrated in Fig. 2.1c) each RB module can autonomously connect and disconnect from another module or from a passive connector embedded in the environment. The ACM is genderless and non-back-drivable. This latter property is an advantage since it means that no power is needed to maintain a given position, which prevents disconnection in case of power loss. In the remaining parts of this thesis, we consider that only the most external connectors of a module (C0X and C3X in Fig. 2.1a) are equipped with an ACM, the remaining eight being completely passive. A RB module is controlled through wireless communication and contains two 1200mAh Li-Po battery packs ensuring more than one hour of autonomy in full charge. Each module is driven by a set of distributed embedded electronics. A single module weights around 1.4kg and any of its joints can provide sufficient torque to lift at least one additional RB module. Two RB modules assembled together using the connectors on the outer hemispheres ($C0X$ and $C3X$) form a *metamodule* (MM). Four connection types can be defined (see Fig. 2.5), inducing different kinematic properties and motion capability. A MM has a payload of around 500g on the most external connector (C3X, described in Fig. 2.1a). The upper limit for the nominal torque of the two external DOFs of the RB module is around 4.9Nm whereas for the middle DOF, it is around 3.6Nm. The main characteristics of the RB hardware are summarized in Table 2.1. A detailed description of the hardware can be found in [238].

2.2.1 Active connection mechanism

Hardware design

The design of an active connection mechanism (ACM) for self-reconfigurable modular robots is a great challenge. The mechanism should be strong to lift several modules, compact to allow an overall compact robot design, fast to allow fast reconfiguration sequences, flexible to allow different orientations of connections, and genderless to avoid additional constraints on the reconfiguration options. Furthermore an ideal ACM should only consume power during the connection and disconnection process. It should not stick out during locomotion, should support self-alignment of modules during the connection process and allow the modular robot to be autonomous.

For the Roombots ACM design we evaluated several mechanisms and designs. One of the most sophisticated designs is the ACM of M-TRAN I and II, based on a mechanical mechanism of latches. Since only mechanical solutions match our requirements we as well based our RB ACM design (Fig. 2.1c) on mechanical latches. As the M-TRAN III ACM, the RB ACM is inspired by the work of Terada and Murata [258]. However, while the M-TRAN is only operating on a regular 3-D cubic grid and thus can afford to feature separate male and female connectors, our ACM is genderless. It is based on 4 latches that allow connections and disconnections within 1.7 seconds. A great advantage of the hermaphrodite latching mechanism is that only one side of a connection has to be active to connect. Two ACMs are sufficient to support locomotion through reconfiguration on a grid of passive connectors covering the floor, walls and ceiling of a room (see chapter 6).

The RB ACM is designed to be non-reversible allowing the motor to be switched off while holding the connection. One of the reasons why we are able to reliably perform our reconfiguration experiments is the self-alignment property of the latches.

ACM Characterization

References and contributions
This subsection is based on the following internship project:

E. Stavridis, "Design and Optimization of Active Connection Mechanisms (ACMs) for Roombots modular robots", Internship Project, École Polytechnique Fédérale de Lausanne (EPFL), 2013. Available at: http://biorob.epfl.ch/page-111200.html

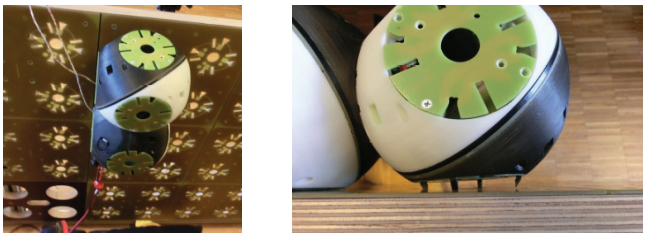
My contributions were:

- *general co-guidance during the project.*
- *proposed methods for the analysis of the data.*

The external contributions were:

- *design of possible connection mechanism solutions.*
- *hardware experiments.*
- *analysis of the collected data.*

We conducted a series of experiments (both qualitative and quantitative) to characterize the actual version of our connection mechanism in terms of tolerance against misalignment and distance from the goal connector. The qualitative experiments consisted in attaching with different orientations a single Roombots module equipped with two ACMs (one for holding and one for gripping) to a wall of passive connectors and gradually changing the motor angles to evaluate when and why the connection mechanism would fail to connect to the grid (see picture 2.2).



(a) The experimental setup.

(b) A failed connection attempt.

Figure 2.2 – ACM characterization experiments. A single RB module is equipped with two ACMs and connected to the grid using one them. The experiment consists in changing the orientation of the module and then varying the motor angles before trying to connect with the second ACM (adapted from [240]).

In order to further characterize the performance of the ACM design we conducted two different quantitative experiments:

1. Gripping distance experiment: we placed one RB module vertically rigidly connected to the wall using screws. The ACM is placed in the upper hemisphere of the module and a target connector plate is kept parallel to it (see figure 2.3). We vary the distance between the ACM and the target from 2mm to 7mm in steps of 1 mm.

2. Gripping performance experiment: we rigidly connected a single module to a wall of connector in the horizontal orientation, so that the effect of gravity are maximized (worst case scenario). We used forward kinematics to select the positions we considered relevant to test during the experiment. We obtained a cloud of reachable solution that we pruned using the previous qualitative experiment results and geometric constraints, and we ended up with 45 cases to check. We repeated them 5 times for the two possible

horizontal orientations. The ACM was able to grip on average 14 and 11 positions for the two horizontal orientation, respectively.

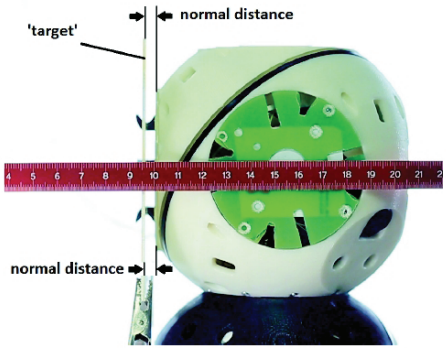


Figure 2.3 – ACM gripping range characterization experiment. A single RB module is rigidly fixed to a wall of passive connectors and we vary the distance between the connecting ACM and the target connector, always keeping the ACM parallel to it. The distance varies from 2mm to 7mm with a 1mm step (adapted from [240]).

We have shown that the current ACM generation can compensate for a misalignment of around 13° degrees in around the normal of the connector and 4mm in translation. Two new designs were proposed to improve those characteristics.

2.2.2 Hardware transfer

In the domain of SRMR, hardware imperfections are inherent to all the developed platforms. Among them, the effects of bending due to gravity are a major cause of failure during self-reconfiguration. This problem arose already in the early stage of modular robots (see for example the misalignment in the 3D self-reconfigurable structure proposed by Murata et al. in 1998 [182]) and have been considered challenging since then. Various approaches have been taken to cope with this issue, both in the hardware development and in the control strategies. Some authors have proposed to improve the connection mechanism in order to widen the connection range [238]. Others have equipped their modules with sensors (for example infrared sensors [276]) and have applied iterative control methods to locate the connection point and correct the misalignment. Different connection means have often been used to ensure a more reliable connection process. For example, the connection mechanism used in the Roombots module relied on the addition of permanent magnets (for the alignment) coupled with mechanical latches. We have demonstrated that combining those two techniques significantly improves the connection rate [238].

In this section, we present two approaches that we followed in order to reduce the reality gap between our simulation and our robotic platform. The first approach is an attempt to model the elasticity effect using a learning algorithm to derive a polynomial expression of the deformation for the end connector of the unit. The second approach relies on the use of

sensors (IMU and camera) to actively compensate for the misalignment during the connection phase.

Modeling the elasticity effects

References and contributions
This section is based on the following master thesis project:

E Badri, "Elasticity compensation using explicit learning", Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2011. Available at: http://biorob.epfl.ch/page-68157.html

My contributions were:

- *general guidance during the project.*
- *proposed modeling methods.*

The external contributions were:

- *help with the hardware experiments.*
- *partial analysis of the results.*

When studying the bending effects of a simple beam in 2D, we can notice that the corresponding equation are expressed by a simple polynomial expression. The exact expression of the deformation depending on the considered point in the body becomes more complicated in 3D and does not exist in closed-form for complex shapes. In order to approximate this deformation, we postulated that the elasticity in the end connector of a kinematic chain made of several modular robots can be modeled on every main axis by a polynomial expression. The coefficient of this polynomial expression can be derived using a regression algorithm call Eureqa, developed by Schmidt et al. [223]. In this study, the chosen alphabet for the regression was inspired by the study of the 3D theoretical expression of the deformation of an homogeneous beam. Eureqa required both experimental data and simulated one to perform the regression. We obtained the set of experimental data by using a motion capture system and positioning the chain of robots in various configurations. The simulated data have been obtained using a simplified model of the RB module to which we applied a Finite Element Analysis method to simulate the deformation.

The results of this study were a bit inconclusive because of the dependencies between the different causes of the deformation. First of all, the home-made gear boxes of the Roombots modules suffer from a highly non linear backlash that introduced a high level of noise in the experiments. Secondly, in real case scenario, the deformation is also induced by the bending

of the connection at the basis of the chain, which is also non-linear and hard to predict.

Improving the connection procedure

References and contributions
This subsection is based on the following internship project:

E. Senft, "Misalignment compensation during the connection phase of the SRMR Roombots", Internship Project, École Polytechnique Fédérale de Lausanne (EPFL), 2013. Available at: http://biorob.epfl.ch/page-111201.html

My contribution was:

- *general guidance during the project.*

The external contributions were:

- *theoretical definition of the problem and derivation of a possible solution.*
- *implementation of the image analysis method.*
- *proof of concept experiments.*

We proposed to equip the end effector of a RB metamodule with a regular camera augmented with an IMU (see 2.4 for an illustration of the camera setup). We used a regular webcam (Cisco VT) that we disassembled to integrate it into the ACM plate. The IMU was external to the module for the testing. The metamodule iteratively approaches the target connection point, using an IK loop to find the minimal changes in the Degrees Of Freedom (DOF) between the two displacements of the chain. The target connector is equipped with two dots (blue and red) that are tracked using the OpenCV library [37]. The red dot is used to find the X and Y real coordinates of the chain while the blue dot serves as an indicator for the connection type. The Z coordinate is fixed using the IMU. A movie of a proof of concept experiment is available at [227]. We are currently working on integrating this procedure into our self-reconfiguration frameworks.

2.2.3 Kinematic structure: case study of the Roombots module

In order to find the right set of angles to direct the ACM of a SRMR towards a goal connector we need to solve the *inverse kinematic* problem corresponding to the kinematic chain of the considered module or metamodule. Our goal is to be able to derive these angles given any desired position and orientation of a chosen ACM, the unreachable cases being detected by the algorithm.The ability of SRMRs to alter their morphology to adapt to the task to be performed brings an additional layer of complexity when we have to derive inverse kinematics

(a) Top view (b) Side view

Figure 2.4 – The camera setup adapted on an ACM plate. The camera is connected via a USB cable to the main computer which performs the image processing. We could improve the compactness and the integration of this unit by designing a custom made camera board.

solutions. Various approaches have been proposed to solve this issue, such as numerical solutions based on Jacobian matrix computation [135], Newton-Raphson algorithm [51], or neural network computation [257]. Numerical solutions often suffer from their lack of robustness, in terms of convergence and completeness, and require a significant amount of time to be computed. Closed form solutions present the advantage of being stable and extremely fast to compute. Unfortunately, closed form solutions can not always be derived for a given robot configuration. Automated methods have been proposed to find those closed form solutions, such as the IKFAST algorithm [77]. The most commonly used tools rely on the Denavit-Hartenberg parametrization method [269], but this method imposes the burden of re-deriving the solution for any specific case considered, in addition of having to manually setup a well suited referential frame system. The Product of Exponential (POE) formula is a geometric formulation that offers an alternative to this parametrization. It has been widely used to study kinematic solutions for classic robots [198, 202] and modular robots [185, 49, 142, 290, 284, 48]. We demonstrate in Appendix A how we can obtain the closed form solution for the IK of a RB metamodule composed of two units using screw theory and a special decomposition of the global IK problem into classical subproblems known as Paden and Kahan subproblems [198, 131]. Our derivation is based on the study described by Murray et al [185].

2.2.4 Roombots working space study

Even if the algorithms that we are presenting in this thesis are platform-independent, a careful study of the working space of the chosen hardware platform can significantly optimize their performance. We briefly present in this section a kinematic study of the working space of two possible Roombots active structures, a single module and a metamodule, that will be used in the following reconfiguration and manipulation algorithms (subsection 4.1.3, subsection 4.2.3, and chapter 5).

Single module

One single RB module can be represented by a kinematic chain composed of three revolute joints (illustrated in figure 2.1b). The diagonal DOF allow the module to approach a vertical surface in a parallel fashion to avoid collision. The main drawback of this DOF is the impossibility for the module to follow a straight line on a grid. The number of possible positions to which the C3X connector can connect on a regular planar grid is two. One of the main disadvantages of the single RB module is the quite large amount of space needed to reach a position.

Metamodule

The metamodule we consider is composed of two modules connected together using the C3X connector from the first module and the C0X connector from the second module. Four different kinds of connection are possible between the two modules depending on the relative rotation of the second module around the z axis. The four metamodule types are illustrated on Fig. 2.5 with their respective names. We consider a regular 2D grid composed of connector plates and count the number of connector states reachable by a fixed metamodule. A connector is said to be reachable with a given orientation (connection type) if there exist an inverse kinematic solution that allows the C3X connector of the second module to connect to it without collision (internal and external) with the considered orientation. We obtained the results presented in Table 2.2. We observed that none of the MM regular reachable space is a subset of one other. Nevertheless, one MM type (PAR) obtains significantly worse results than the three other. This can be explained by the redundancies in the resulting degrees of freedom of the PAR type kinematic chain. In the following discussion we will use primarily PER type when using metamodules, considering their reachable space is the widest on a regular grid.

As we can see, the reachable space of a metamodule is significantly larger than the single module one (in a regular 2D grid, a metamodule is able to reach at least 19 positions against only 2 for a single module). In the remaining part of our study, we favor the use of metamodules as the basic manipulating active unit.

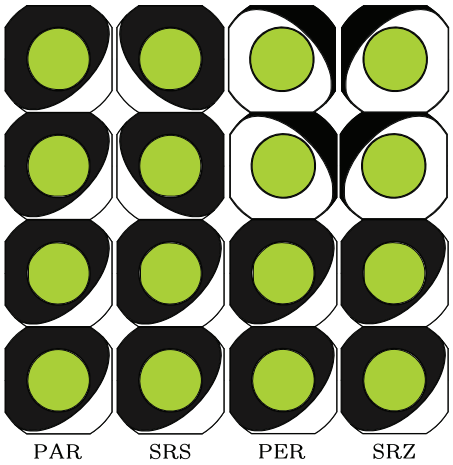


Figure 2.5 – The four different types of metamodules. The Roombots active connection mechanism is a four way symmetric mechanism that allows connecting two units with four different relative orientations. We considered that a module is equipped with two ACMs placed at the bottom connector C0X and at the top connector C3X. The four different types of RB metamodules are obtained by connecting the C3X ACM of the first module to the C0X ACM of the second module with a relative rotation of $\pi/2$ for each type. They are called respectively PARALLEL (PAR), SHEAR-S (SRS), PERPENDICULAR (PER), and SHEAR-Z (SRZ) (corresponding to a relative rotation of 0, $\pi/2$, π, and $3\pi/2$, respectively).

Table 2.1 – Hardware specifications of a Roombots module (table from [272]).

Specification	**Value**
Degrees of freedom	3 (continuous rotational)
Outer motors	Faulhaber 2342 012 CR
Inner motor	Faulhaber 2232 012 SR
Outer gearboxes reduction	305:1
Inner gearbox reduction	366:1
Outer dofs speed (No load)	26.6 RPM
Inner dof speed (No load)	19.4 RPM
Outer dofs nominal torque	4.9 Nm
Inner dof nominal torque	3.6 Nm
Number of connection ports	10 (active or passive)
Active connection type	4-way symmetric genderless mechanical latches
Overall dimensions	110x 110x 220 mm
Weight	1.4 kg
Communication	Bluetooth
Energy source	4-cell LiPo battery, 1200 mAh autonomy ~1 hour

Metamodule types				
S.M.	*PAR*	*SRS*	*PER*	*SRZ*
2	19	38	43	35

Table 2.2 – The number of positions reachable by the different types of metamodules in a regular 2D grid. The number corresponding to the single module (S.M.) is indicated for reference.

Reconfiguration Part II

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 these robots, self-reconfigurable modular robots are able to autonomously change their morphology to better adapt to the task they have to perform. This is done by re-arranging, adding or removing modules inside the main structure (videos illustrating these processes can be found at [21]). 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 configurations increases exponentially with the number of degrees of freedom and the number of connectors in the structure. Hou et al [115, 116] have proved the NP-completeness of the self-reconfiguration process, justifying the use of heuristic methods. Several approaches have been proposed to tackle this issue. Most of them use an abstract representation of the module, called the *sliding cube model* [280]. In this abstraction, the modules are represented by cubes able to slide perfectly on the surface of the structure. Multi-agent frameworks have addressed the question of self-adaptation of a modular structure to real-world perturbations using a consensus-based approach [286] as well as reaching and grasping objects using modular robots with evolving morphology [24]. 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*. The use of scaffolding structures [243] and strict building sequence [129] 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 [89], 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 heuristic based approaches. In order to reduce the complexity of the reconfiguration problem, the notion of metamodules has been developed: instead of considering isolated modules, groups of modules are used and controlled as the basic elements of the structure (see [236] for an example of simulated results and [6] for a review of the related complexity analysis). While 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 [98]).

Those approaches considered a system with only active units. When a structure needs to be built, it might be worth both in terms of structural properties (rigidity, weight distribution, elastic behavior, shape,...) and manufacturing constraints (cost, time,...) to include *passive* elements (i.e. elements without actuation). Such elements need to be manipulated by the active units and placed in the final structure at the right time. Similarly, a defective module can also be treated as a passive unit and be carried towards a maintenance area or can be moved from the working environment. In addition to this new concept of passive pieces inclusion, we are also interested in creating algorithms that are readily transferable to the robotic platforms. Indeed, the previously described theoretical methods suffer from their lack of realism when representing the robots in use. Among others, the effects of bending (due to gravity), deformation, faulty connection or faulty units are often neglected. In this part, we present novel approaches to tackle both the classical self-reconfiguration problem but also the reconfiguration of heterogeneous structures including passive elements. We emphasize our attempt to include more realistic characteristics to our framework such as bending effect active compensation, connection mechanism characterization, or torque limitation.

This part is organized as follows. We first (in chapter 3) introduce the terminology we are going to use throughout the part. We define precisely the problem we are going to tackle (section 3.1) and review the state of the art in terms of complexity analysis (subsection 3.2.1 and subsection 3.2.2), representations (subsection 3.1.1), and similarity measurement (subsection 3.1.2). We also present advances regarding the use of metamodules to reduce the complexity of the SR problem (subsection 3.2.3). In section 4.1, we present existing heuristics based approaches to solve the self-reconfiguration problem. In subsection 4.1.2 we focus on a very successful framework inspired by the gradient mechanism. We present afterwards (section 4.1.3) our extension of this technique in which we introduce close range strategies for the different units and study their impact on the number of deadlocks situations. In section 4.2, we describe exact approaches to solve the SR problem. We cover more in details the method introduced by Fitch et al. [90] (subsection 4.2.2) based on Markov decision process, from which we took inspiration to create our reward based reconfiguration framework (subsection 4.2.3). Finally, in chapter 5, we present a novel hierarchical planner to solve the SR problem with passive elements. Preliminary results on how the augmented self-reconfiguration problem can be reduced to a multi-robots planning problem are presented in Appendix B.

3 Background

3.1 Problem definition

A structure made of an homogeneous set of Reconfigurable Modular Robots (RMR), called a *metamorphic* system [56, 55], has the ability to change the arrangement of its composing modules to adapt to changes in the environment or to a task redefinition.

A *configuration* of RMR is uniquely characterized by the following parameters:

- The spatial arrangement of the modules (their coordinates in space with respect to an absolute referential).
- The value of the different motor positions for every module.
- The state (connected or disconnected) of every connector/linking element (active and passive).

A *shape-configuration* is defined by the geometrical arrangement of its composing elements (i.e. by the geometrical volume it occupies). A shape-configuration can lead to several configurations.

The *configuration space* is the set of all possible configurations that can be assumed by a RMR system.

The problem of *reconfiguration* or *metamorphosis* consists in finding the sequence of movements (motor positions) of the different degrees of freedom of the constituting modules as well as the related connection and disconnection of the linking mechanism to go from a configuration *A* to a configuration *B*. When this process is done *autonomously* (i.e. without intervention of an external operator) we referred to it as *self-reconfiguration* or *self-metamorphosis*.

Hybrid SRMR can use a *substrate*, meaning a structured environment with connection ports (most of the time passive) to perform their self-reconfiguration. By extension, modules can

also be considered as substrate during the process. If a structured environment is not used we talk about *on-site self-reconfiguration* (for example in the case of M-TRAN [183] or Superbot [222]). Otherwise, when an external substrate can be used (in the case of hybrid platforms like Roombots [238] or Smores [74]), we qualify the process of *off-site hybrid self-reconfiguration*.

The self-reconfiguration process can be either *homogeneous*, when all the active robotic units are the same during the reconfiguration process, or *heterogeneous*, when different active units are used. We introduce a new type of self-reconfiguration called *Augmented Self-Reconfiguration* (ASR). In this case, the configurations can be composed not only of active units but also of *passive unactuated elements* only equipped with connecting ports compatible with the active units. Those elements could be partially equipped with sensors.

In order to evaluate our approaches, we consider the following requirements that reflect constraints found in real life scenarios:

1. **The environment can change dynamically** through the reconfiguration process. In particular, obstacles can be added, removed or moved during the process.
2. **The final structure can change dynamically** through the reconfiguration process, meaning that one or more of the characteristics of a configuration mentioned previously can be modified.
3. **The environment is not perfect**: the effect of bending of the robotic units as well as connection failures are taken into account in the proposed methods.

We further assume that all the active units considered are capable of local sensing: they can detect a neighboring module as well as obstacles and passive elements (in terms of geometric shape, position, and orientation).

3.1.1 Configuration representations

In order to encode a configuration of reconfigurable modular robots, different representations techniques have been proposed. Among them, graph representations have a prominent use. Hou et al. [116] introduced the *C-Graph* representation in which every node of the graph corresponds to a module and every edge to a connection, labeled by a tuple indicating the connector of the first module, the orientation of the connection and the connector of the last module. The C-Graph is not directed, making it impossible to determine from the graph which active connection was used for the connection. Freudenstein et al. [93] introduced the concept of kinematics graphs to study mechanical mechanisms composed of links and joints. Baca et al. [12] used a variation of kinematics graphs by introducing connection ports to encode the ability of RMRs to connect and disconnect dynamically between each other. A module is decomposed into a set of joints and links and the graph represents the connection between those sub-elements through connection ports. The corresponding graph

is directed, with every node representing a link and every edge a joint. A number at each sides of the connection link indicates the number of the connection port used (a 0 indicated the lack of connection). The type of joint (spherical or fixed connector) is represented using a different type of connection line (single for spherical and double for fixed connector). This representation fully (and uniquely) described any modular robot assembly. Those information can be incorporated into an Assembly Incidence Matrix (AIM) [51, 50] to ease its use inside numerical algorithms. For a system with N_{link} links and N_{joint} joints the corresponding AIM is a $(N_{link}+1) \times (N_{joint}+1)$ matrix in which every value a_{ij} indicates the port number joining link L_i to joint J_j. The extra line in the AIM contains the type of joint and the extra column the type of link (prismatic, spherical or no link). The advantage of using a graph representation for a RMR structure is the ability to afterwards use the powerful tools developed in Graph Theory [101]. A *reconfiguration graph* [76] can also be used to represent the SR process. In this graph the nodes represent configurations of the structure and the edges linking two nodes, an action (often atomic, i.e. involving only one degree of freedom) allowing to go from one configuration to the other. A cost can be attached to the transition.

3.1.2 Similarity measurement between configurations

The ability to quantify the level of similarity between two given configurations is a key aspect when considering structures that will evolve over time. In order to guide this evolution and to measure the degree of matching between an intermediate configuration and a goal configuration, several metrics have been introduced. Baca et al. [12] proposed to use the AIM representation to quantify the difficulty of moving from one configuration of RMR to another. To do so, they identified repeatable assembly patterns inside the AIM between growing structures in order to reduce the complexity of the displacement planning. Hou et al. [116] proposed to use their *C-Graph* representation to encode the self-reconfiguration problem as a graph matching and graph similarity problem. The authors demonstrate that the complexity of the SR is linked to the number of matching nodes and connections between the initial and the final structure. Despite the similarities between SR and graph similarity, even if the initial and final graphs are acyclic the SR is still NP-Complete contrary to the polynomial time solvable matching problem. Nelson [187] also compared the initial and final configurations using graph matching theory. Castano et al. [45] considered static structures only (in comparison with Chen et al. [50]) and represented them with only directed graphs. This graph only representation allows them to equal the matching of two configurations with the isomorphism search problem (in comparison with [263, 182] for example). This representation can additionally manage loops and modules with multiple connection ports. The directed graph representing a multi-port module can be unambiguously interpreted since it has no automorphism [139]. Park et al. [203] compared approaches using spectral decomposition of incidence matrices, classic isomorphism finding, and the 3DLL method using linked lists to identify identical configurations. Pamecha et al. [199] and Chiang et al. [53] proposed to introduce cost functions to quantify the difficulty to go from a given configuration to another one considering a geometrical representation of the structures. They introduced different

metrics (mainly the Hausdorff distance and the optimal assignment metrics [54]) and showed how those metrics can be applied in the domain of SR. The authors differentiate between single module motions and branch motions where the metrics have to be minimized to create the driving force for the reconfiguration or to minimize the overall effort, respectively. In order to find similarities between graph representation of SR structure, Asadpour et al. [10, 9] introduced the notion of *graph signature* as a unique identifier allowing for a fast isomorphism test between structures (the method of computation of this graph signature was further improved by Golestan et al. [99], see subsection 4.2.1). The structure is represented by a labeled graph (undirected for hermaphrodite connection mechanisms and directed otherwise) in which the nodes are the modules and the edges are the connections between them. The label is uniquely computed based on the two connectors attached in the connection as well as the relative rotation between the modules.

3.2 Evaluation metric and complexity analysis

3.2.1 General concepts of complexity

We present in this section an intuitive explanation of several complexity concepts. For a formal definition of these notions, please refer to [175].

A decision process is a problem that can be binary answered. The main classes of decision process are the following:

- The class P of decision problems contains those that can be decided in polynomial time.
- The class NP contains the problems for which any answer can be verified (not found) in polynomial time.
- An $NP-complete$ problem is an NP problem to which any other NP problem can be reduced to in polynomial time.
- A problem A is NP-hard if there is an $NP-complete$ problem B reducible to A in polynomial time.

3.2.2 Optimality and complexity

A reconfiguration sequence is *optimal* if it minimizes the number of connections and disconnections needed to transform the initial structure I into the final structure F. The problem of SR is intractable, considering that both the state space, i.e. the possible configurations that can be created using a number n of modules, and the actions space, i.e. the set of actions that can be performed by the modules at each time step, grow exponentially with the number of active units considered. It has been proven to be *NP-Complete* [116, 115] for chain type

modular robots using a reduction of the problem to the *3-PARTITION* problem (known to be NP-Complete). As a consequence, the optimal solution for SR cannot be found in polynomial time.

To be closer to the hardware constraints, the optimality of the SR can also be measured using different criteria:

- The time needed to perform the task.
- The energy consumption, which is tightly linked to the angular displacement of the modules and the torque applied to the module.
- The number of modules needed to perform the task .

In the case of Augmented SR, the notion of complexity becomes more a multivariate criterion based on the following parameters:

- The ratio between active units and passive units (in case of shape configuration for example, when those numbers are left open).
- The spatial placement of the passive and active units in the configuration.
- The relative placement of the initial configuration with respect to the final one.

3.2.3 Granularity analysis: use of metamodules

In order to tackle the NP-Completeness of the SR problem, a variety of heuristics has been developed. To improve the time complexity of those heuristics and to limit the motion constraint of their platform, a large number of authors considered using metamodules (i.e. structures composed of several active units) as the basic active blocks of their reconfiguration algorithm. First introduced by Kotay and Nguyen [144, 189], the metamodules have been shown to lead to more efficient SR planning. Nguyen et al. [189] achieve a $O(n)$ in time SR and Prevas et al. [212] a $O(n^2)$, with n corresponding to the number of modules in the structures. The main robots considered were the Crystalline [273] and the Telecube [249] platforms. Depending on the hardware constraints (mainly the strength of the module) for the Crystalline platform, the following time complexity was achieved (n corresponds to the number of modules in the structure):

- $O(n^2)$ for constant strength [221, 271].
- $O(n)$ for linear strength [6].
- $O(\sqrt{n})$ for linear strength and increasing velocities [218] (2D case).

- $O(\log n)$ time for linear strength and acceleration [7] (using $O(n \log n)$ atomic moves) (2D and 3D case).

Metamodules specific to a given hardware platform have been devised (i-Cube [270], Crystalline [221], Atron [60], uniform modular robots [212], and cube style MR [271]). The main drawback of these designs is that they are tightly linked to a given platform or type of platforms, making it difficult to extend the follow up theoretical contribution on planning. To alleviate these constraints, Dewey et al. [76] proposed a generic and ideal metamodule design called *pixel*. By extending the work of Abrams et al. [3], the authors defined a single movement primitive (in this case, the ability for a module to be created or destroyed at any point of the structure) to eliminate local constraints (i.e. the constraints linked to the module hardware). The authors proved that this metamodule configuration was holonomic and demonstrate that any practical system could be reduced to the pixel system. They derived from this result a class of practical metamodules to remove the local constraints (generalization of [189, 75, 243]). In the system, the motion is ensured by transferring modules between the different metamodules while enforcing global constraints such as connectivity or stability. The goal of the approach is to construct a reconfiguration graph based on those metamodules in which two nodes are adjacent if and only if there exist a single motion primitive between those configurations. The authors proved that their planner was complete and effective. Metamodules can be used to create equivalent structures for different types of hardware platforms, making possible to use universal algorithms developed for a given class of platforms for a metamodule build of different hardware elements. Kurokawa et al. [149] demonstrated that 8 M-TRAN modules were equivalent to a 2D Crystalline module. Aloupis et al. [5] showed the equivalence in terms of class of modules between M-TRAN, SuperBot, Molecube, and Roombots (used in a given metamodule configuration allowing contracting and protracting motion) and the Crystalline/Telecube platforms. This result allows to apply the same time complexity results as the ones mentioned previously, without having to consider the intrinsic complexity of the specific hardware platforms. One major drawback of this approach is that it considers structures impossible to build with the current hardware (58 modules in [5] for example, which exceed the torque limitation of any current modular robot). Moreover, as pointed out by Hou et al. [116], the complexity analysis neglects the hardware constraints such as bending, connection failure, and dynamical effects during the moves. A promising work exploring fault tolerance in self-reconfiguration using metamodules has been conducted by Christensen [57].

3.3 Conclusion

We presented in this chapter the foundation of the SR problem in terms of terminology, complexity analysis, and structure representation. In the next chapter we describe the state of the art related to SR algorithms, both those based on heuristics approaches (section 4.1) and those based on exact approaches (section 4.2), and introduce the advances we made in both domains.

4 Self-reconfiguration of homogeneous structures

Using autonomous self-reconfigurable robots to create arbitrary structures is an idea that has been widely studied in the past decades. Many approaches have been proposed and hardware developments have followed. Unfortunately, those theoretical approaches often fail when transferred to existing robotic platforms because they consider almost perfect physical systems. In this chapter, we present first a state of the art of the heuristics based methods used to solve the SR problem (section 4.1) and give more details regarding the gradient approach (subsection 4.1.2) introduced by Stoy et al [243], that has been used to develop our own reconfiguration framework (subsection 4.1.3). We introduce in section 4.2 different exact methods relying on abstract mathematical models allowing a more systematic complexity analysis and ensuring the termination of the process in a given number of moves. We focus (subsection 4.2.2) on a framework describe by R. Fitch [90] that inspired our implementation of a reward based reconfiguration algorithm (section 4.2.3).

4.1 Heuristic approaches

4.1.1 Stochastic methods

Chirikjian [55] defined the concept of *metamorphic systems*. The author introduced four main constraints regarding the design of the basic units for a SRMR system: (i) the modules need to be homogeneous to ease the planning process, (ii) the shape of the modules should allow an efficient filling of the space, (iii) a single module should be self-sufficient in terms of movement (it should be able to locomote autonomously over adjacent modules), and finally (iv) every module should be equipped with an active connection mechanism to allow multiple units to act as a single kinematic entity. A hierarchical set of rules is used to complete the 2D self-reconfiguration process of hexagonal shaped modules actuated using alternating opposite polarities on their faces. The SR rules includes the preservation of the structure connectivity, the conservation of the total number of modules, the synchronous motion of one module per time step, and (intuitively) the impossibility for a unit to move in an occupied spot. The SR process is guided using a cost function quantifying the amount of changes required to go from

one configuration to another, using single action steps.

Murata et al. [181] proposed a 2D stochastic self-reconfiguration algorithm based on the diffusion technique augmented with a leaking factor. In order to spread the value of the fitness function among the different modules, the authors used an analogy with a water reservoir system: every module is a reservoir connected to its neighbors and the level of water between the units will equilibrate as time passes (meaning that the value of the fitness will be propagated throughout the structure). To avoid overflow when the value of the fitness changed (the total volume of water is then not conserved), the authors introduced a leaking factor into the diffusion equation. The authors listed several advantages of using homogeneous units in a SRMR systems: (i) the robustness of the system against failure (fault tolerant and low maintenance cost thanks to self repair capability), (ii) high adaptivity to the environment, and (iii) cost efficiency using the mass production aspect. Penrose [205] in 1959 already pointed out the necessity of having parts with shapes allowing for an easy and complementary pairing when designing mechanical units for self-organization. Murata et al. introduced a 2D modules named *fracta* using magnetic connection to achieve SR. Their design was guided by simplicity to ensure the reliability of the actions of the system. Every unit can have up to 12 different connection states, depending on the type and the number of connections established by the module. A transition diagram to represent the possible change of the connection states of a unit is introduced, where a node corresponds to a given connection state and an edge to a possible transition between two states (a more complex cost function could have been used, like the required energy for example). The distance between two states is measured using a cost function simply defined as the minimal number of edges between the two states. The authors used the unit type and the types of its neighbor to describe a whole shape by strings of connection types. This representation is not unique. The authors also introduced a similarity measure between modules' states based on the current type of unit in comparison with the final type and with the neighboring units' types. This measure is used to define the moving strategy. To communicate those values between the units, a diffusion field with an additional leak constant is introduced. This technique proved to be efficient but suffers from deadlocks and is not complete (due to its stochastic aspect).

Murata et al. [182] introduced the first 3D SRMR and associated planning technique. The authors classified the different types of studies conducted in the field of modular robots as (i) purely theoretical work (cellular automata [188, 152] and swarm intelligence [18]), (ii) RMR (Yim [282], Hamlin et al. [104]) and (iii) 1D/2D SRMR. They introduced an homogeneous 3D SRMR unit composed of a cube with connecting arms attached on its six sides. At the end of each arms an ACM is implemented. The authors pointed out the two main difficulties to go from 2D to 3D systems, namely the effect of gravity and the geometrical constraints imposed by the use of the third dimension as opposed ot the 2D case. They used symmetries to simplify their design and took inspiration from their previous work on fracta [181]. They are the first ones to use MM to achieve *pair-wise* movement of their unit (one unit rotates another unit to its destination). By doing so, they conformed to the constraint of self-sufficiency mentioned

by Chirikjian [55]. They listed several requirements for hardware design. In order to describe a shape they use connection types lists. The SR algorithm is based on relaxation process [97] and Markov random field [97]. The algorithm was implemented in a distributed and parallel way but only tested on a single structure, on which it showed good performance.

Tomita et al. [263] developed a new self-assembly and self-repair method for the 2D fractum system based on the bio-inspired *nucleation method*. They stressed the advantages of homogeneous systems (both in terms of hardware and software) over heterogeneous ones in terms of *replaceability*, *reduction of production and maintenance cost, design freedom*, and *scale extensibility*. They pointed out the analogy with the cells of living organisms that share and store the same genetic information. They proposed a distributed approach considering system homogeneity and local communication between the units. They introduced a method to locally describe the goal shape of the SR process. The concept of self-repair is also introduced as a mean to ensure the self-sustainability of the robotic system. The notion of self-reproducing systems was introduced by Von Neumann in 1966 [188], but those concepts were difficult to implement using real mechanical systems. A significant amount of work has been conducted in this field, mainly on the theoretical level ([278, 153], and references within). Concerning hardware development of self-replicating systems, Penrose [205] proposed a brick model to simulate the metabolism of living organisms. Ichikawa [119] extended this work to create the first 1D self-reproducing robot. Kokaji [141] introduced his "fractal machine" composed of a triangular units and Chirikjian et al. [54] proposed a hexagonal unit equipped with three servo motors to change its shape and a connection mechanism to attach to adjacent units. Pamecha et al. [200] also proposed a square unit using sliding mechanism for its displacements. Ueyama et al. [268] introduce the CEBOT robot, an hexagonal 2D unit able to self-reconfigure. Few 3D systems have also been proposed [281, 143, 182]. Regarding the assembly process, Lindenmayer [162] developed a mathematical model of the development of living organisms, called L-system based on cell division, but the model was difficult to transpose to real hardware because of the impossibility to provide self-replicating capabilities to the units. A self-assembly model has been proposed by Thompson et al [260], but it cannot be applied due to unrealistic connection mechanisms. Applicable and more realistic methods have been proposed by Chirikjian et al. [54] and Beni [18]. The authors use their previously developed 2D *fractum* unit [181]. A fractum unit is composed of six connecting arms, each equipped with either permanent magnets or electromagnets (female and male arms, respectively). They used changes in polarity of the electromagnet to perform three basic actions: change the connection type between two units, cut the connection, and move a unit on a substrate made of other units. They modeled every units as a circle with six branches (corresponding to the arms). The global configuration was described using list of connections types (twelve in total, corresponding to the arrangements of the arms), to encode the state of the connection between the units. The process of self-assembly was driven using random motion. A unit would evaluate the difference between its connection state and the goal state and move randomly if the difference is not null. The frequency of the movement is proportional to the magnitude of the difference between the connection states. This process was well suited

for small structures with symmetries (97% of success for a 10 unit symmetrical structure) but suffered from low success rates when considering larger unsymmetrical structures.

4.1.2 Gradient based approach

In this subsection, we present in details the SR method proposed by Stoy [243] in 2006. 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 [23] 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 [129]. Stoy 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.

Cellular automata

In [243], 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 [243] 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 neighboring 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.

Reconfiguration

The reconfiguration procedure is composed of the following three elements:

1. **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 fills 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.

2. **Gradient generation**

   A *concentration* gradient is used to attract the wandering modules into unfilled positions. The seeds act as sources which emit a simulated chemical in all the neighboring directions. The range of this emission can be controlled. The value of the gradient will be propagated using message passing between neighboring 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 neighboring 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 4.1.

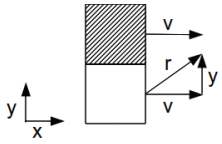


Figure 4.1 – The computation of the vector gradient. The considered module is the white one. The hatched module is the neighbor with the maximum concentration. v is the vector gradient of this module and x and y forms a regular base. r is the resulting vector for the considered module: $\vec{r} = \vec{v} + \vec{y}$ (adapted from [243])

.

3. **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 modules and thus damaged hardware. Moreover, disconnected groups of modules might form. 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 neighbors 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 neighboring modules. This means that the module has no influence on the connectivity of the substructure.
- Module is not a source.

Stoy [243] 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.

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 neighbors. It can also freely slide over the surface of the structure and around neighboring 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 neighbors.
- 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.

Conclusion

K. Stoy [243] 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.

4.1.3 Our approach

References and contributions
This section is based on the following publication:

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. The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, IEEE International Conference on Intelligent Robots and Systems, 2010.

And on the following master thesis project:

P. Laprade, "Distributed Roombot Locomotion and Self-Reconfiguration", Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2010.
Available at: http://biorob.epfl.ch/page-36411.html

My contributions were:

- *proposed ideas for the different strategies.*
- *testing the different strategies between the units.*
- *evaluating the impact of the seeding procedure on the reconfiguration process.*

The external contributions were:

- *the analysis of the results.*
- *the implementation of the framework.*

This section describes the current implementation of our reconfiguration framework for the RB platform. We took inspiration from the work done by K. Stoy [243] and developed a gradient based approach to solve the reconfiguration problem. 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 [236] and [239].

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 the floor, ceiling, and walls, to which modules can attach. MM are the basic units considered in our case. Each metamodule is equipped with only two active connection mechanisms, one at in the bottom hemisphere of the first module (C0X, called afterwards the foot connector), and one in the top hemisphere of the second module (C3X, called the head connector afterwards). 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 (neighbors, obstacles, ...). To perform the basic moves leading to the final position, a precomputed look-up table composed of shape-transitions (motor 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. An overview of the framework is presented on Fig. 4.2.

Metamodule shape and initialization At the beginning of the algorithm 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 $3D-S$ (see Fig. 4.3). The motors angles representing the transition between these shapes are stored in a database (*motion planner database* depicted in Fig. 4.2) and used when needed by the MMs.

Reconfiguration through locomotion During the reconfiguration process, the MMs go from one shape to another using the precomputed 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 neighborhood and requests the precomputed collision cloud corresponding to the desired shape to shape transition. It also broadcasts its position to the neighboring MMs, via the world controller.

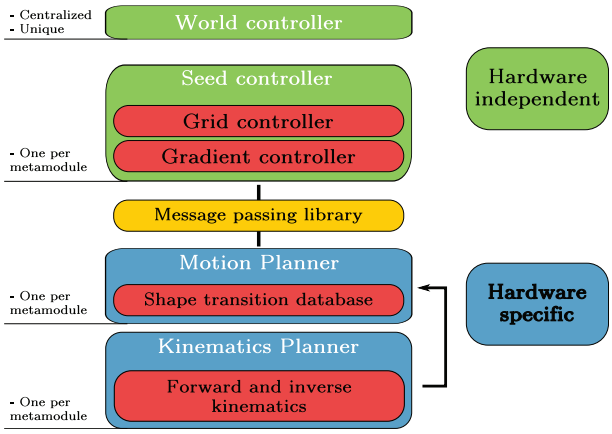


Figure 4.2 – Overview of our gradient based reconfiguration framework. The framework is composed of two main blocks, the low level hardware specific block (depicted in blue) and the high level hardware independent block (depicted in green). They are linked together using a message passing library (in yellow). The hardware specific part of the planner is used to precompute the required moves to achieve shape to shape transition (kinematic planner) and to store them in a database (motion planner). In the high level part, the seed controller is computing the gradient for the different positions on the grid (gradient controller) and it is deciding on the next move to perform based on this value (grid controller). The transition angles to go from the current position to the goal position in a collision free fashion are determined by querying the motion planner database. The state of the world (i.e. the position of the modules, the state of the final structure, and the state of the seeds) is managed by a centralized unit (world controller).

Seeding mechanism In order to guide the MMs during the reconfiguration process, goal positions have to be defined. These final positions 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 *levels* 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 leveling are provided by the user.

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 neighbors. The MM can thus compute the vector force corresponding to the different seeds. The neighboring modules are included in the computation as repulsive sources. Three approaches have been tested regarding the influence of these modules:

1. The *greedy* approach: the neighboring MMs 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.

2. The *slope* approach: we consider a gradual decrease of the influence of the neighbor

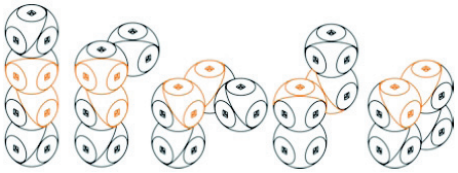
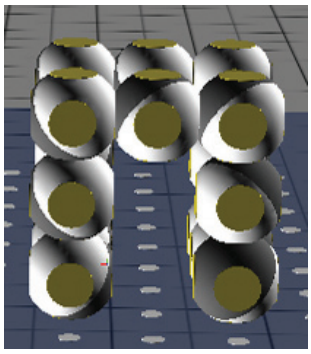
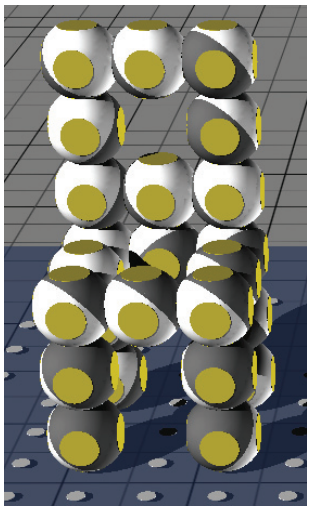


Figure 4.3 – The five metamodule shapes used in our SR framework. From left to right: I-, L-, 3DS-, S- and U-shape (adapted from [235]).



(a) A box like structure (4 MMs).



(b) A chair like structure (6 MMs).

Figure 4.4 – The two goal shapes used to test our gradient based reconfiguration framework. The two shapes are placed in a structured environment equipped with connectors (white circles).

modules with the distance. Only the metamodules in the range of the considered MM will have an influence. This approach is similar to the *temperature test* introduced by De Rosa et al. [75] in their shape sculpting framework via hole motion: the probability of a hole appearing depends on the distance between the site and the closest point on the perimeter of the target geometry, modulated by a decay factor.

3. The *step* approach: the MMs are given the same influence in the whole range of the considered MM. This was intended to minimize collisions between MMs by enforcing a kind of minimal distance policy.

Results We performed several experiments in simulation using up to six metamodules. The goal shape was either a box-like structure (composed of four MMs) or a chair (with six MMs).

We repeated the experiments with the four different types of MMs to analyze their kinematic abilities. We counted the number of deadlock situations (i.e. when the MMs were not able

to build the final shape), the amount of collisions and the overall number of moves needed to complete the reconfiguration task. Each experiment was repeated either three times (for the box shaped final structure) or four times (for the chair-like goal structure) with randomly shifted MMs initial positions. The seeding procedure has been generated by hand. We varied the seeding order for the chair setup using three different strategies: (i) the seeds corresponding to the legs of the chair are given in circular order, followed by the two seeds corresponding to the back of the chair (one after each other); (ii) compared to (i), the seeds for the legs are given in a cross-wise order; (iii) all seeds for the leg are given at the same time followed by the two seeds corresponding to the back of the chair (also at the same time). All the experiments were conducted with the four types of MMs (Fig. 2.5) and the three different gradient strategies. We recorded the number of deadlocks, the number of collisions, and the number of moves needed during the experiment. Those results are summarized in Table 4.1, Table 4.2, and Figure 4.5.

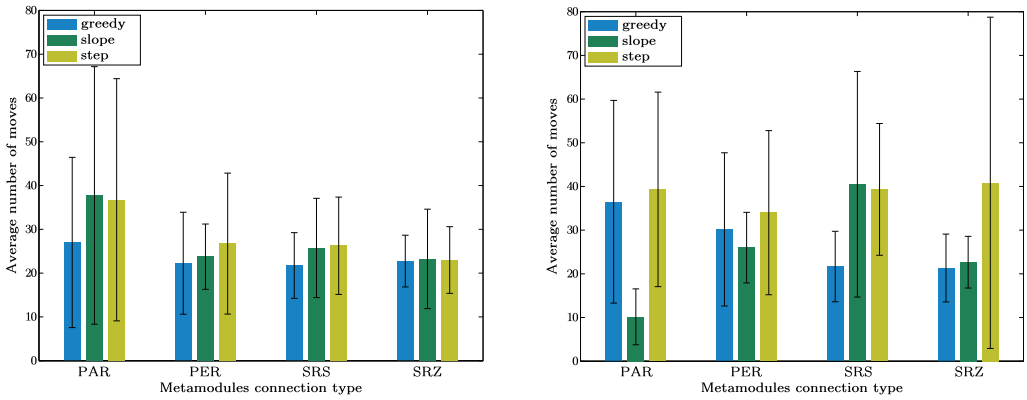
Table 4.1 – Four MMs box assembly: numbers refer to the number of collisions (CL). Deadlocks (DL) are indicated by $*$. Table columns indicate three different strategies: greedy, slope, and step function for the force vector estimation. Rows show the four different metamodule configurations (PER, PAR, SRS, SRZ). Three sets of experiments per configuration are shown, with the initial position of the MMs randomly shifted. The number of collisions happening in deadlocks are excluded from the collision counting (adapted from [236]).

	greedy			slope			step			DL	CL
PAR	4$*$	0	0$*$	0$*$	2	12$*$	0$*$	2	2$*$	6	4
PER	5$*$	0	2$*$	1$*$	1	0	0	1	0	3	2
SRS	0	0	0	0$*$	3	5	0	0	1	1	9
SRZ	0	1	0	0	0	0	1	0	0	0	2
DL	4			4			2			10	
CL	1			4			5				

Table 4.2 – Deadlocks (DL and $*$) and collisions (CL, numerical values) for the reconfiguration into a 6 metamodule chair-like structure. Four experiments for each combination of MMs types and gradient strategies are performed with different *seeding orderings* (adapted from [236]).

	greedy				slope				step				DL	CL
PAR	0$*$	0$*$	1$*$	3	7$*$	1$*$	1$*$	0	9$*$	2$*$	1$*$	2$*$	10	3
PER	1	6	0$*$	8	2	1	2	1	5	1	1	1$*$	2	28
SRS	0$*$	2	1	0	1$*$	1$*$	1	0$*$	8	7	0	6	4	25
SRZ	0	0	2	0$*$	0$*$	2	0	3	2	0$*$	1$*$	4$*$	5	9
DL	6				7				8				21	
CL	25				12				30					

The most relevant observation we made was that metamodules of type *SRS* and *PER* were



(a) 4 metamodule cube

(b) 6 metamodule chair

Figure 4.5 – Bar plot showing the average number of moves four or six metamodules (PAR, PER, SRS, SRZ type) need to build a cube or chair-like structure, respectively. Each bar represents one experiment (repeated 3 and 4 times for the box shaped final structure and the chair-like structure, respectively). Colors indicate the force-field strategies (greedy, slope, step function). (a) Building a box from four MMs is the easiest task of both, with 20 moves required on average. PAR metamodules perform the worst on average, while the SRZ metamodules perform better on average. The greedy strategy tends to result in the least amount of necessary moves. (b) The chair structure is more complex to build and the results show that the step strategy needs longer to assemble. Only the PAR and SRS metamodules type succeed to complete the chair with this strategy. The slope strategy performs very well in both successful cases (for PER and SRZ metamodules) (adapted from [236]).

more successful on average at building the considered structure than the three other types. In terms of gradient strategy, we observed in the more complex case of the chair that the step strategy leads to a larger number of moves whereas the slope strategy performed better than the two others on average. The greedy approach tends to dominate only for the simpler case of the 4 MM box. In terms of seeding strategies, we did not observe significant differences between the three approaches we proposed. Nevertheless, our results tend to show that optimizing the seeding recipe in comparison with a random choice of the seeds placement reduces the number of deadlocks.

4.1.4 Conclusion

The technique that we presented partially matches the requirement we specified in section 3.1:

1. Dynamic environment: since the computation of the different moves takes into account a local sensing of the environment (for other modules and for obstacles), it is possible to add or remove objects from the setup on the fly.
2. Final structure changes: the final structure can also be changed as long as seeds are defined in the added parts.
3. Realistic environment: the active units are considered as perfect (no bending included or failed connections). However failure of some modules can be handled as a special case of adding obstacles into the environment.

This method presents the advantage of being scalable in terms of number of modules. The centralized aspect can be weakened using local communication between the units instead of a central entity managing the state of the world.

One of the main weaknesses of our approach is the possibility of ending into a deadlock situation. We have however observed that the strategies we proposed tended to reduce the number of deadlock situations, especially for the more complex cases.

4.2 Exact approaches

4.2.1 Graph isomorphism

As we have described before (subsection 3.1.1), structures composed of modular robots can be represented by graphs. The SR process can be viewed as the convergence of the initial graph representing the initial structure towards the graph corresponding to the final desired state. Golestan et al [99] proposed an improved version of the method introduced by Asadpour et al. [10, 9] to perform SR for chain type modular robots using graph invariant. The configuration of the structure is captured using a graph in which a node corresponds to a module and an

edge to a connection (undirected for genderless connection mechanisms, directed otherwise). On top of this representation, the authors proposed to use a *transition graph* where nodes correspond to a given configuration of the structure and edges to actions. They used the notion of *graph signature* (GS) introduced in [10] as an isomorphism invariant to encode the 3D structure of a configuration. GS is computed using a modified Depth First Search method on the labelled graph of the structure. They sped up the computation of this invariant by using the notion of *power centrality* [25] in case of symmetric modules. GS is used by the *edit distance* metric to guide the search towards the final configuration together with the RRT [156] planning method to determine the possible actions at each time step. The authors presented encouraging results for the M-TRAN (with four, eight, and twelve modules structures) and SuperBot (with four and eight modules structures) platforms, with a significant decrease of the required computation time to find the first valid solution to the SR problem.

4.2.2 Markov decision process

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

In [90], Fitch et al. 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 [88] where they developed their Markov Decision Process (MDP) formalism, not in the native kinematic space but for an abstract model of sliding cubes [280]. Many authors use the concept of metamodule (a group of two or more modules assembled together) to reduce the number of kinematic constraints in the problem (as presented in subsection 3.2.3). The authors [90] 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 to solve it using dynamic programming [220]. 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 takes 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.

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 whose 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 modules inside the structure is much more challenging. The authors propose a local method based on the definition of *connecting cycles*. For each potentially moving module, a *connectivity graph* composed of the adjacent modules is built. Then a local search is done to find existing paths 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 synchronization 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.

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 [165] 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 is composed of two actions (see Fig. 4.6), and the reward is -1 for each move (the best policy will tend to favor fewer moves). The value function is stored in a distributed fashion. Each module only stores the value of its connectors and updates it using the message passing process when a neighboring module state changes.

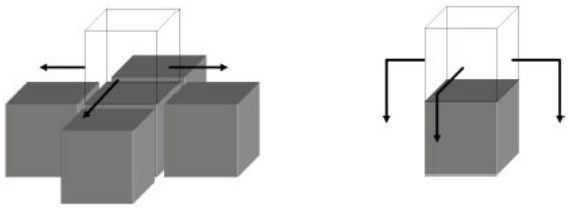


Figure 4.6 – The action space A for the abstract modules (adapted from [88]).

Module kinematics

The main novelty introduced by Fitch et al [90] is the use of module kinematics in a built-in fashion inside the previous MDP formulation. As a consequence the complexity and convergence analysis can directly be 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 degrees 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 a *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.

Results

The authors presented two examples of self-reconfiguration for structures made of Superbot modules: a nine modules line shape evolving into a box shape and a eight cube like structure reconfiguring into a goal shape specified by a given bounding volume.

Conclusion

Fitch et al. [90] 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 managed 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 [150]). 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 prohibitive when dealing with chain or hybrid type modular robots.

4.2.3 Our approach

References and contributions
This section is based on the following master thesis project:

M. Stöckli, "Reconfiguration algorithm for adaptive furniture", Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2012. Available at: http://biorob.epfl.ch/page-81014-en.html

My contributions were:

- *general guidance to develop the implementation of the framework.*
- *proposed evaluation metrics.*

The external contributions were:

- *analysis of the results.*
- *implementation of the framework and critical thinking on the underlying concepts.*

This section present our implementation of a reward based reconfiguration algorithm using a simplified Markov Decision Process (MDP) inspired by the previously described method by Fitch et al. [90]. We provide here a summary of the main steps and results.

Implementation

The basic kinematic units considered in our approach are the metamodules composed of two Roombots modules. They evolve into a finite 3D normalized grid. We define a simplified MDP where the set of state corresponds to the set of available connectors in the structured environment (excluding the metamodules connectors), the set of actions is determined on the fly using an inverse kinematic solver, and the transition are defined depending on whether or not a collision occurs between two states. An overview of the framework is presented on Fig. 4.7.

The goal position of every active units has to be specified beforehand as it impacts on the computation of the reward map. At each time step, the active units try to maximize their reward. In case of collision, the next best action is tested. The reward is computed taking into account three main factors: (i) we want to minimize the number of steps so every action has a reward of -1; (ii) we also want to favor longer moves so we add a "bonus" to the reward depending on the distance between the goal connector and the current position; (iii) we do not want to visit the same connectors several times, so we penalize the already visited states.

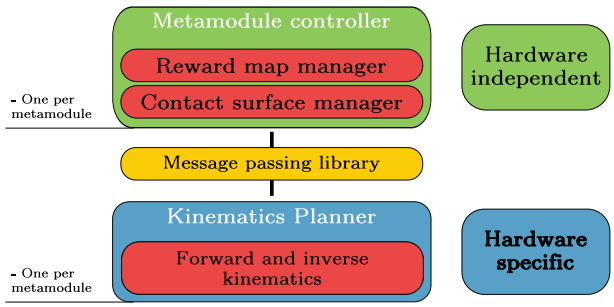


Figure 4.7 – Overview of the reward based reconfiguration framework. As for the gradient based framework, the reward based framework is composed of two main blocks, the low level hardware specific block (depicted in blue) and the high level hardware independent block (depicted in green). They are linked together using a message passing library (in yellow). The hardware specific part of the planner is used to compute on the fly the required move to achieve posture to posture transition (kinematic planner). In the high level part, the reward map manager computes the reward map taking into account the reachable connectors and the desired final position of the modules (off-line computation step).

Testing and results

We tested this approach in three different setups:

1. 2D Grid: a 13 by 13 connector grid in which we vary the connection type to the goal connector and the initial position and orientation of a single metamodule (see figure 4.8), reaching a total of 24 different runs.

2. 3D Box: we place a 6 by 6 by 6 box with connectors on the surface. Two metamodules have to reach predefined goal positions. We only vary the connection type to the goal connector and the initial position and orientation of one of the two metamodules (see figure 4.9), reaching a total of 24 different runs.

3. 2D grid with a narrow passage: connectors are removed from the previous 2D grid to form a channel between the goal positions and the initial positions of four metamodules (see figure 4.10).

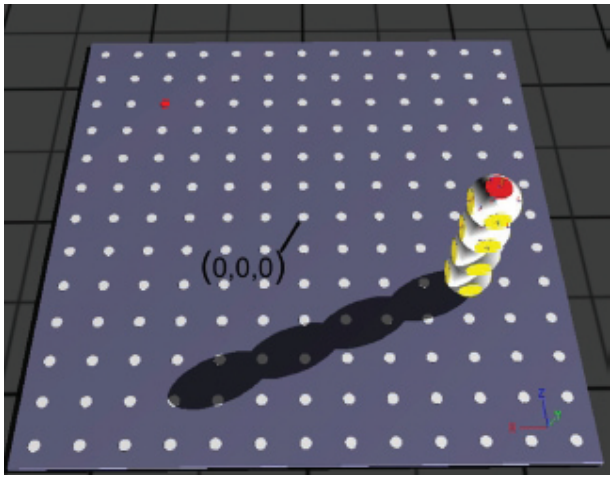


Figure 4.8 – 2D terrain of 13 by 13 connectors (white circles) and one RB metamodule (adapted from [242]).

The success rate of the algorithm reaches 100% but we observed a non negligible number of failed moves (i.e. moves leading to a collision): on average, 1.22 and 0.92 failed moves appeared per minimal required move (read from the reward map) for the 2D and 3D experiments, respectively. The computation time required for the reward map grows exponentially with the number of connectors considered.

Conclusion

The technique that we presented partially matches the requirements we specified in section 3.1:

1. Dynamic environment: since the computation of the different moves takes into account a local sensing of the environment (for other modules and for obstacles), it is possible to add or remove objects from the setup on the fly.

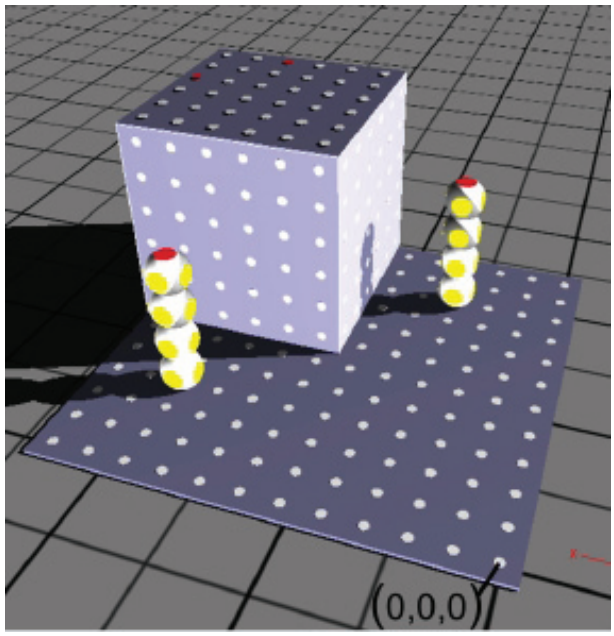


Figure 4.9 – 3D terrain with a box equipped with connectors (white circles) and two RB metamodules (adapted from [242]).

2. Final structure changes: the final and initial position of the modules have to be fixed at the beginning of the run otherwise the reward map needs to be recomputed.

3. Realistic environment: the active units are considered as perfect (no bending included or failed connections). Failure of some modules will lead to an unfilled goal position. If an extra number of modules has been provided, the goal position needs to be re-attributed and the reward map recomputed.

The main weakness of this approach is the need for a precomputed step corresponding to the computation of the reward map. This step is computationally demanding and not scalable in the number of connectors composing the grid. Since the grid topology has to be adapted, this approach also suffers from being unable to integrate modules connectors as potential anchor points during the process.

4.3 Conclusion

In this chapter we have presented existing approaches to solve the self-reconfiguration problem of homogeneous modular robots.

Regarding heuristics approaches, we have described our own framework based on a spatial gradient acting as a force field to guide MMs into predefined seeds positions in the final structure. Our method takes inspiration from the technique introduced by Stoy et al [243] in 2006. The gradient approaches are inherently scalable in the number of modules that

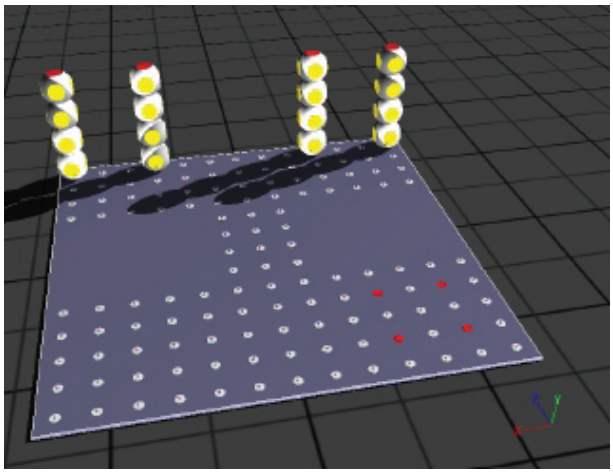


Figure 4.10 – 2D terrain with a narrow passage and four RB metamodules (adapted from [242]).

can be controlled asynchronously and it opens the way for large structures building. In our framework, the world controller is the only centralized element, but it could also be replaced by a message passing mechanism between the units present in a certain neighborhood. A locking mechanism could be locally introduced to prevent discrepancy in the world description for the different modules. The major issue with the gradient based approaches is that the termination of the process is not guaranteed for arbitrary initial and final positions of the units. The definition of the seeds in the structure is a complex task since it requires an a posteriori knowledge of the motion of the units before reaching the target position. We extended previous works by incorporating hardware kinematics in a decoupled fashion in to the SR loop and we have shown that the fact of introducing different interaction strategies between the units and selecting the type of connection between the units to maximize the available workspace tended to reduce the number of deadlocks. Nevertheless in all the presented heuristics based approaches, including ours, only perfect units were considered, making it difficult to transfer such an algorithm to the hardware. Similarly, no strategies have been developed to cope with the failure of a unit or of one motor, since all the moves are precomputed.

In comparison, the exact approach introduced by Fitch et al [90] in 2010, offers a proof of termination as well as strong theoretical tools for the analysis of the reconfiguration process, in terms of complexity and optimization. The fact of integrating directly the kinematics of the modules inside the process to select the available moves for the units as well as the hierarchical organization of the framework (with a clear decoupling between hardware specific parts and high level parts) brings flexibility and robustness to the method. Our adaptation of this technique allows us to provide a complete framework for reconfiguration with built-in convergence thanks to the ordered attribution of the final positions to the different units. Nevertheless, the preprocessing step in the algorithm to compute the reward map, that we introduced to simplify the overall method, impaired the flexibility of changing the goal position or the units initial states.

All the methods presented in this chapter suffer from their lack of consideration of the hardware imperfections. In our methods, we have taken care of creating hierarchical implementation with a clear decoupling of the hardware specific parts, like the inverse kinematic computation or the collision handling, but we still considered fully working units, perfect connection and disconnection processes, and no bending effects in the modules, conditions that are seldom observed in current hardware platforms. In addition, few methods have been proposed so far to handle passive pieces in addition to the active units as part of the final structures. In the next chapter, we present a novel approach to self-reconfiguration including fully passive elements equipped only with connectors matching the ACM of the active units. Our proposed method enforces a strict decoupling between hardware and high level control, and introduces several additional check points regarding the matching between simulation and hardware, based on an on-the-fly torque computation. We focus on a simple planning method based on the $A^\star$ algorithm, and on the high level motion planning technique *RRT-Connect* [146].

5 Augmented self-reconfiguration

As we have seen in the previous chapter, several approaches have been developed to solve the self-reconfiguration problem with homogeneous active units considered as perfect. These methods are promising but they are also prone to lead to results not transferable to the hardware. The goal of this chapter is twofold. We first describe a novel hierarchical manipulation framework allowing to transport passive unit in arbitrary terrains with embedded connectors (section 6.2). We show how we integrate hardware constraints at the core of the algorithm to ensure a better match with the robotic platforms. We afterwards demonstrate the efficiency of our method in various simulated experiments (subsection 7.6).

5.1 Passive object manipulation and transport

References and contributions
This section is based on the following publication:

S. Bonardi, M. Vespignani, R. Moeckel and A. J. Ijspeert. Collaborative Manipulation and Transport of Passive Pieces using the Self-Reconfigurable Modular Robots Roombots, IEEE International Conference on Intelligent Robots and Systems, 2013.

My contributions were:

- *theoretical development and conceptual ideas.*
- *algorithm implementation and testing.*
- *control script for the hardware experiments.*

The external contributions were:

- *building of the test setup.*
- *some of the illustrations of the manipulation process.*

5.1.1 Introduction

Modular robots, as opposed to monolithic ones, are composed of several homogeneous or heterogeneous units (often referred as modules) to improve the overall flexibility, adaptability and robustness of the structure to specific tasks in unknown environments. This modularity comes with the challenge of collaboration between the different modules to form the optimal configuration for a specific task.

Self-reconfigurable modular robots can create a large variety of kinematic structures depending on the applications. One possible use of this versatility is the creation of manipulators able to autonomously locomote in the environment using embedded connectors and to adapt to the object to be carried. Using their self-reconfiguration capabilities, these robots can efficiently move inside a structured environment and dynamically change shape to handle changes in the tasks (e.g. additional objects to be handled) or in the surroundings (e.g. new obstacles). Possible applications for such a system could be the automated construction of arbitrary structures or fully automated warehouses where modular robots are used to carry and store objects in shelves (for example, as a complement of the successful solution proposed by *KIVA systems* [253]).

Our self-reconfigurable modular robot Roombots (RB) has been designed to be used as building block for adaptive pieces of furniture able to move, self-assemble and self-reconfigure. Using the reconfiguration capabilities of RB, we can study distributed locomotion control as

well as self-organization and collaboration between modules [239].

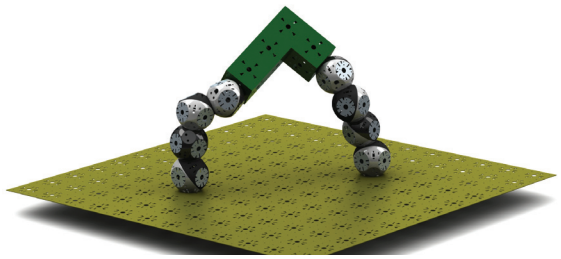


Figure 5.1 – Two metamodules (two RB modules connected together) on a 2D grid collaboratively manipulate a L-shaped object (in green) equipped with passive connectors. The object is transported thanks to a sequence of manipulations and of metamodule on-grid locomotion.

A single RB module can autonomously travel to any position on a 2-dimensional grid by a sequence of connections and disconnections between the modules' active connection mechanisms (ACMs) and the grid structure (i.e. panels with regularly spaced connectors) and overcome concave edges in 3 dimensions.

In order to achieve our goal of furniture that can change shape to adapt to the user's needs, we have to be able to design efficient structures in terms of physical properties and cost. That is the main reason why we envision robotic furniture composed not only of active RB modules but also of passive elements, with the RB modules acting both as manipulators and as components of the structure (an example of the manipulation and transport phase of a passive element is presented in Fig. 5.1). In this application, a set of RB modules needs to perform on-grid locomotion to pass along passive objects. In comparison to the methods presented in Chapter 4, we have to add the constraint of manipulation of a fully passive unit into the SR process. The locomotion through reconfiguration of the different units on a substrate of connectors is similar to the classic SR problem, but the handling of the elements constrains the modules to collaborate to achieve their task.

In order to build a heterogeneous structure using RB, we design a manipulation and transport framework that can be generalized to different self-reconfigurable modular robots able to use passive connectors to locomote. The requirement for environments equipped with connectors can be partially relaxed considering the off-grid locomotion capabilities of the RB platform [210]. Our goal is to find the sequence of motor movements and connections/disconnections for a group of active units to collaboratively carry a set of passive objects from an initial position to a final one in an arbitrary 3D non regular grid with obstacles (illustrated in Fig. 5.2).

In section 5.1.2 we review some successful approaches in the field of objects manipulation and structures building using mobile and modular robots. We then describe our manipulation architecture in section 5.1.3. We test our approach in simulation and describe afterwards a proof of concept experiment using the RB hardware (section 5.1.4).

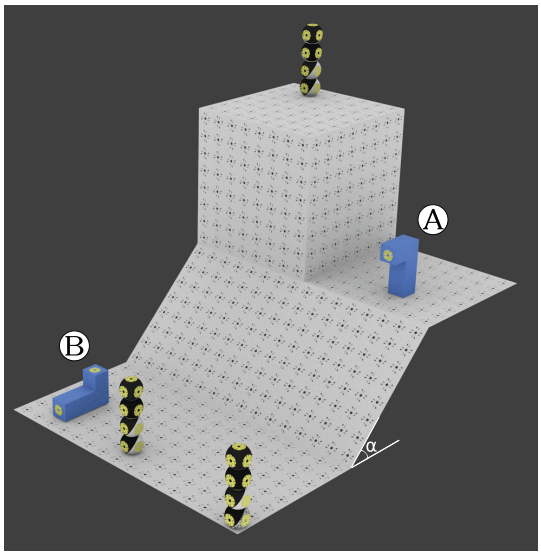


Figure 5.2 – Example of a manipulation and transport scenario: three metamodules have to carry an L-shaped object (in blue) with connectors, from a point A to a point B. This requires (i) that the metamodules move by sequentially attaching and detaching to and from connectors in the environment (represented as black circle, randomly made available on the grid plates), (ii) that they attach to and manipulate the object, and (iii) that they collaborate to bring the object to the target position B.

5.1.2 Related work

Manipulation and transport of objects using mobile platforms equipped with robotic arms is a well studied research area. However using reconfigurable modular robots for manipulation of passive objects has been scarcely explored so far. Terada et al. [258] proposed a complete framework to build arbitrarily layered structures using a specialized manipulator with four Degrees Of Freedom (DOF) and specific building blocks. The robot uses inch-worm locomotion on the structure and occasionally rotation to change direction. The sequence of moves of the robot is controlled using a gradient approach and a local negotiation via blackboard to avoid collisions between several manipulators. One of the main limitation of this approach is the need for active connection mechanisms on the external faces of the elements being carried around, as opposed to the arbitrarily shaped fully passive elements we are considering. Additionally, the limited degrees of freedom of the manipulator constrain the structure to be built in a layered fashion as opposed to the fully 3D manipulation problem we are tackling. Another very successful approach has been proposed by Petersen et al. [207]. The authors use mobile units to grab specially designed elements to build an arbitrary structure from a high level representation. The path chosen by the robots to go from the supply spot for passive elements to the goal position is determined using a depth-first search algorithm coupled with a set of rules to prevent inaccessible positions. The task of manipulation is simple since it mainly consists in deposing the piece in the given spot with a rotation of a one DOF actuator. In this case the complexity of the manipulation is shared between the manipulator and the

design of the element. The main difference with our method is that we can easily transform everyday life objects into movable objects simply by adding passive connector plates to them and connect active units in a plug and play fashion. This aspect brings more flexibility in the type of structures that can be built using the manipulation and transport method we present. Groß et al. [102] presented a framework in which several Swarm Bots modular robots [178] collaborate to move an object from one position to another on a flat terrain. The modular aspect comes from the fact that the wheeled robots used can dynamically connect between each other using a gripper based mechanism, to form larger chains able to move bigger objects using traction. The main strength of the approach used in this paper is the careful experimental validation of the transportation task. The main limitation of the proposed approach is the difficulty for the platform used to locomote in irregular 3D environment (limitation to almost 2D terrain) as well as the use of pure traction to move the object. Several studies have been conducted in the domain of automated truss assembly using modular robots [285, 111], but they are limited by the fact that the robots cannot physically attached the passive elements to the structure. A framework using aerial swarm robots has been introduced by Lindsey et al. [163] but the method suffers from the need for specialized elements and the limited payload of the aerial vehicles. In the factory floor model proposed by Galloway et al [94] specialized tiles composed of a manipulator arm made of CKbots modules, an elevator unit, and a guiding mechanism for the truss elements, are being designed. Trusses are fully passive elements that are manipulated by the robotic arms and attached together using nodes. One of the main limitation of this approach is that any change in design in the structure requires the deconstruction and reassembly of the total structure. A stochastic control method for multi-robot collaborative task has been proposed by Napp et al. [186] and demonstrated on the problem of the assembly of truss structures using this factory floor tile system. Another promising control approach using truss climbing robots and specialized elements has been proposed by Yun et al. [288]. In our approach we neither impose a specific design or structure in the arrangement of the manipulator or of the connector substrate nor require specialized passive elements or active units. In addition, the active units are part of the built structure and can be used to increase the potential for adaptation of the final structure that is not restricted to a 2D or 3D grid layout. As we have seen in the previous chapter 4, a large number of successful approaches have been developed to achieve displacement of modular robots to form arbitrary structures [244, 98, 90] but they only consider active units as building blocks. In this chapter we proposed a manipulation framework using self-reconfigurable robots to manipulate fully passive elements in an arbitrary 3D environment equipped with connectors. The only constraints on the elements are the need for at least two anchor points compatible with the active units and a weight that does not exceed the payload of the active units.

5.1.3 Hierarchical planner

The task that we are solving is to find the complete sequence of motor angles and connections/disconnections for a set of active elements to collaboratively bring a set of passive elements from an initial to a final position. We assume that the passive elements are not actuated and

that they have to be always connected to at least one active unit during the transportation task. The former requirement arises from the need of maintaining the element at a given position before the next active unit connects to it. An equivalent solution would be to design holders at predefined points to store the pieces between two handling actions. Nevertheless we prefer to consider solutions that would require the least amount of extra facilities to solve the task we defined. The world (i.e. the available connectors and the obstacles, their position and orientation) is supposed to be known. The information about the shape and available connectors of the passive elements are also assumed to be known beforehand. No parallel motion with multiple active units is considered. As a consequence, the weight of the passive element should not exceed the possible payload of one active unit.

We decomposed the handling task into four main elements, (i) a low level kinematic planner, (ii) a motion planner, (iii) a path planning algorithm, and (iv) a handling method. Each of these components is incrementally added into the next one. This decomposition brings flexibility in terms of hardware platforms by decoupling the kinematic constraints from the high level planning.

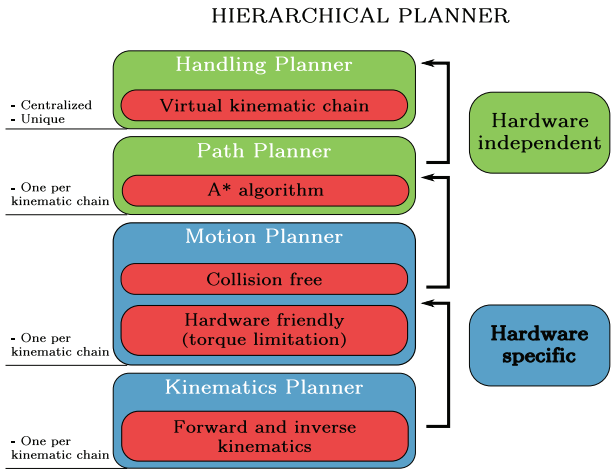


Figure 5.3 – Overview of our hierarchical manipulation framework. The kinematic planner generates forward and inverse kinematics solutions for a specific type of chain. The motion planner provides collision free motion and hardware friendly movements using a torque estimation routine. The path planner, finds the complete sequence of moves and connection-disconnection to go from one initial structure state to a final one. Finally the handling planner defines the connection points between the passive element and the active structures handling it. It also defines the postures of the active structures and their connection type to the grid. Those elements are connected in a bottom up scheme.

Level 1: kinematic planner

Any assembly of Roombots modules and passive elements can be viewed as a set of kinematic chains. Despite the torque restriction on the actual version of the RB hardware (the fact that one metamodule composed of two RB modules can only lift one passive element), we

used a very general representation of the kinematic chain of the structure to allow future generalizations. One module is represented by a 3 rotational DOF chain with 10 connection points. We derive the inverse kinematic solution using the iterative damped Levenberg-Marquardt algorithm [170] provided in the Rigid Body Dynamic library [86]. This algorithm, also called damped least-square (DLS) method, is an iterative minimization method close to the Gauss-Newton algorithm and the gradient method, but generally more stable. Using this technique, we can impose a complete final posture for any chain or tree configurations. Passive elements can be easily integrated into the structure as pure sets of connection points.

Level 2: motion planner

In order to find a collision-free path between two postures of a considered structure given by the previously mentioned kinematic planner, we use a variation of the classical Rapidly-exploring Random Trees (RRT-Connect [146]) motion planning algorithm available in the Open Motion Planning Library [73]. The search for a possible path is done using a discretization of the movement of the chain: instead of considering a continuous movement from a posture A to a posture B, we consider several intermediate static postures that lead from A to B. The validity of every intermediate posture is evaluated using the following two conditions:

1. The posture is collision free: we use the exact model of the hardware module and passive elements to compute the collision manifold of any structure.

2. The posture does not lead to impractical stress constraints on the motors. We compute for every posture candidate an approximation of the resulting torque on each motor and check whether this value is inferior to the nominal torque of the motor. We consider two different torque estimates, corresponding respectively to the worst case scenario (denoted by T^w), and to a more reasonable estimate (denoted by T^r) of the needed torque to achieve a move. To compute T^w, we project each pivot point (corresponding to each motor) on the plane perpendicular to the gravity force and multiply this value by the distance L between this projected point and the projection of the center of mass of the remaining segments on the same plane: $T^w_{motor_i} = m_i * g * L_i$ (m_i corresponding to the mass of the remaining segments in the direction of the lever). This computation gives a crude upper-bound estimate of the real torque applied to the motor and neglects both the friction and the dynamics during the movement, since we consider a completely rigid structure and a fine grain discretization of the movement of the robot leading to an almost static analysis. This overestimation considers that the degrees of freedom are perpendicular to the gravity vector and it will favor moves that prevent over-stressing the hardware. The computation of T^r is similar, but instead of projecting the pivot point on the gravity plane, we compute the lever arm, d, as the perpendicular distance from the motor rotation axis to the line along the gravity force. Let L be the line defining the motor axis, Q the pivot point, and $\overrightarrow{u}$ the vector corresponding to the rotation axis of the motor. We have $L : \overrightarrow{r}(t) = Q + t\overrightarrow{u}$. Similarly, let M be the line defining the force axis, P

the application point of the force, and $\overrightarrow{v}$ the force vector. We have $M : \overrightarrow{s}(t) = P + t\overrightarrow{v}$. We obtain the following relationship:

$$d = \frac{|(\overrightarrow{PQ}).(\overrightarrow{u} \times \overrightarrow{v})|}{\|\overrightarrow{u} \times \overrightarrow{v}\|} \tag{5.1}$$

We consider that the gravity force is applied to the last pivot point of the chain. We obtain:

$$T^r_{motor_i} = m_i \times g \times d_i \tag{5.2}$$

T^r is preferred to T^w because it is less restrictive and allows for a more realistic control of the torque limit of the different motors.

Additional constraints on the posture, such as orientation constraints for a carried object, can easily be added.

Level 3: path planner

The goal of this planner is to find the complete sequence of moves and connection/disconnection to go from one initial structure state (i.e. position, orientation, type of connection and posture) to a final one. The problem of finding a path on a 2-D grid can be viewed as a path-finding problem in a graph. The sequence of grid positions to go from the initial position to the final one is generated using the $A^\star$ algorithm, a popular algorithm for solving path planning in 2-dimensional grids [105]. This algorithm is based on the evaluation of a cost function f which takes into account the distance from the start position and a heuristic estimate of the distance to the goal position (often chosen to be the distance to the goal along a straight line).

$$\forall s = (x, y) \in Grid \quad f(s) = g(s) + h(s) \tag{5.3}$$

where $g(s)$ corresponds to the distance from the start position to the current position and $h(s)$ corresponds to an *estimate* of the distance to the final goal. h is defined in our case as the Euclidean distance from the current position to the goal position, in order to favor paths with fewer and longer moves.

The search space S is composed of connector position p and orientation o as well as type of connection c (there are four main types of connections since the ACM is four ways symmetric):

$$\forall s \in S \quad s = (p, o, c) \text{ with } p \text{ and } o \in \Re^3 \text{ and } c \in [0..3] \tag{5.4}$$

For each state space in S, a neighborhood of reachable states is computed based on the previ-

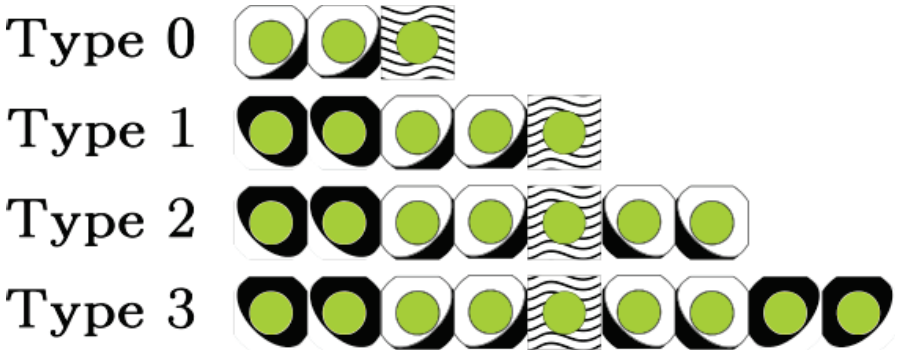


Figure 5.4 – The four different virtual chains. The passive element is represented by a square with wavy lines and the green circles correspond to the passive connectors.

ous motion planner. This computation is done inside a sub-routine which can be modified to integrate further constraints, such as a minimal length required to the next connector or a blocked degree of freedom.

Level 4: handling planner

In order to handle a passive element, we need to define two main parameters: (1) the connection points between the element and the handling active structures, (2) the postures of the active structures and their connection type to the grid. We use the notion of *virtual chain* (VC) to tackle this problem. A virtual chain is defined as a movable structure composed of at least one active unit and one passive element. A structure is said to be movable if it is not blocked (i.e. with elements around that would prevent movement) and if it possesses at least one active unit. We define four basic types of virtual chains (illustrated in Fig. 5.4) depending on the number of active units they are composed of. We assume that the passive element is at first not connected to the active units.

The displacement of one passive element e from a state $A \in S$ to a state $B \in S$ is planned as follows:

1. Depending on the number of active units available, we form the widest (in the sense of the wider kinematic space) virtual chain among the four types by connecting virtual active units to e. For example, virtual chain of type 3 would be favored over virtual chain of type 2. The choice of the connectors on the passive element is made so that the length of the total virtual chain is maximized.

2. Once the algorithm decides on a given VC, it considers the passive element as a fixed point and it uses the motion planner previously defined to find the possible grid connection states (called $S_{available}$) for the active unit closer to the final position of the passive element.

3. The algorithm sorts the available active units based on the distance from their current

connection point to the center of the passive element. The available units configuration is fixed, meaning a metamodule cannot split to form two single modules.

4. The algorithm computes the path from the current position of the active units to the closest grid connection state in $S_{available}$ to determine if the structure is reachable using any of the active units. It iterates over the states in $S_{available}$ until it finds a path or we switch to another active unit. If no solution is found, it changes the type of virtual chain and restart the process from step 1.

5. If the passive element is reachable the algorithm can now compute the set of connector states towards the final state B. The path planner described in subsection 5.1.3 is used with an adapted version of the torque limit constraint to provide motors angles and connection states from A to B. The torque limit is only applied to the final posture of the active units in the final position of the chain. The validation function contains an extra constraint to ensure that any selected state is reachable by at least one active unit, tested in sorted order according to their Euclidean distance to this grid state. This validation is based on the path planner from subsection 5.1.3 including the complete set of constraints on the collision and the torque limit. The final state of the connected active unit is integrated as an obstacle to the collision world to ensure a collision free path for the second moving unit.

6. If the final state is not reachable using the current VC we switch to a smaller type and repeat from point 1.

The main steps of this manipulation routine are illustrated in Fig. 5.5.

5.1.4 Experimental results

We consider, for our experiments, a centralized implementation of the above method, but a fully distributed version could be achieved, if we still consider that the map of the environment is known by every active unit beforehand. We conducted two main experiments in simulation to test our framework. In a first experiment, we quantify the impact of the complexity of the terrain (number of available connectors in the world and inclination of a connecting plane) on the handling process. In the second experiment, we propose to test the influence of the torque limitation on the result of our framework.

Using the RB hardware, we illustrate one step of the handling algorithm we presented earlier using one passive element and two metamodules.

Terrain description and metrics

We test our approach using our own simulation environment based on Open Scene Graph [41] and Bullet Physics [70]. We consider the same terrain template for all the experiments

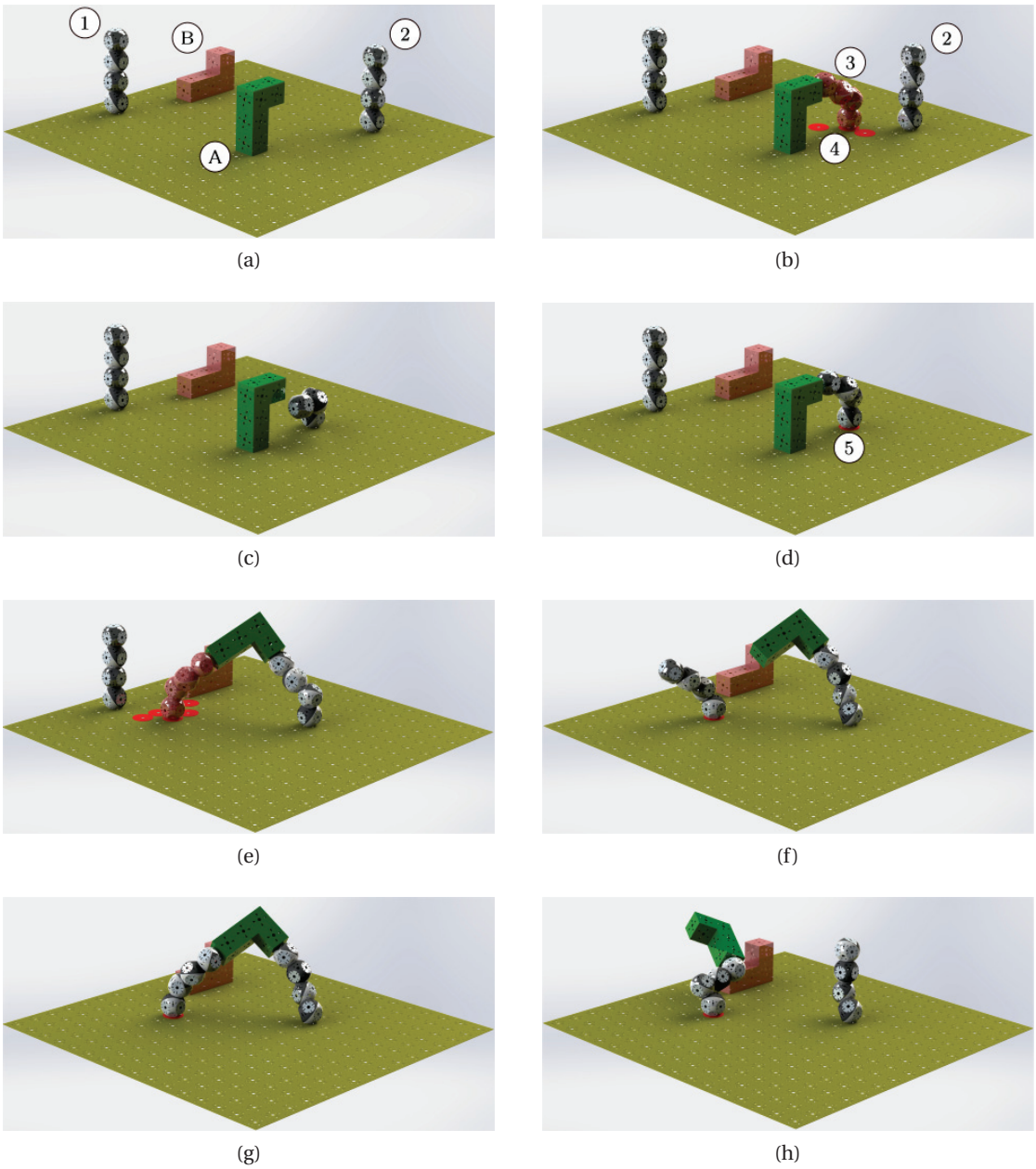


Figure 5.5 – The main steps of the manipulation routine, with one passive element and two metamodules (labelled 1 and 2 in (a)). The connector are indicated by small white circles. The passive object is modelled as a green L-shaped element (labelled A in (a) in its initial position). The red transparent element in the different images represents the desired final state of the passive object and the virtual state of the active units (for example labelled 3 in (b)). In (a) we present the initial configuration of the terrain. The final position of the passive element is displayed in transparent red (labelled B). In (b) the red connectors correspond to the $S_{available}$ set (labelled 4 and mentioned at step 2 in the previous description) determined using the closest metamodule as active unit (labelled 2). (c) and (d) show respectively an intermediate state to get to the chosen connector (in red, labelled 5) by the first metamodule and the connection of the first metamodule to the passive element. (e) depicts the position of the virtual chain when checking the available connection point (red connectors) for the second metamodule (step 5). (f) and (g) show respectively an intermediate state to get to the chosen connector (in red) by the second metamodule and the connection of the second metamodule to the passive element. Finally, (h) represents the final move of the second metamodule to place the passive element into its final position and orientation.

(depicted in Fig. 5.2) composed of several initially perpendicular planes and a maximum of 458 connectors. The connectors are numbered using their coordinate in the regular grid of unit equal to the smaller dimension of the RB module ($0.11m$) . In order to speed up the simulation, we approximate the shape of a RB hemisphere using a sphere of diameter $0.055m$ tangent to the connector plates and a set of spheres of diameter $0.009m$ placed on the rig of the shell (see Figure 5.6 for an illustration). Those spheres would be tangent to the sphere of diameter $0.128m$ corresponding to the outer shape of the RB hemisphere. They prevent unrealistic moves (for example, a continuous rotation in the ground), while being resource friendly in terms of collision detection.

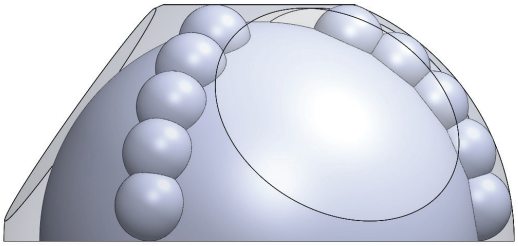
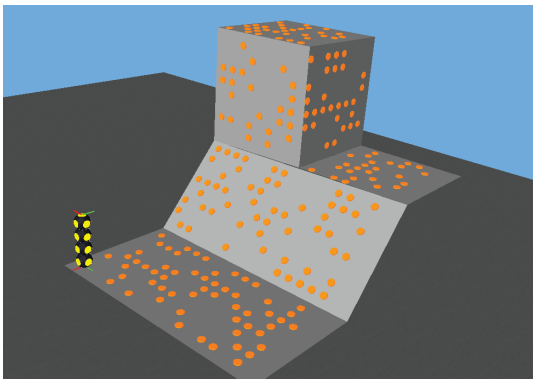


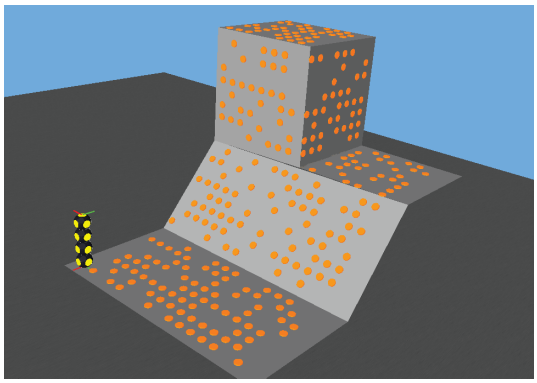
Figure 5.6 – The collision shape considered for a RB hemisphere. This shape corresponds to the union of a sphere tangent to the connector of the hemisphere and centered at the joint origin and of five spheres centered on each rib of the half-sphere and tangent to the outer sphere in which the hemisphere is included (represented on this picture by a transparent layer).

We consider the following quantities as an evaluation of the efficiency of the algorithm:

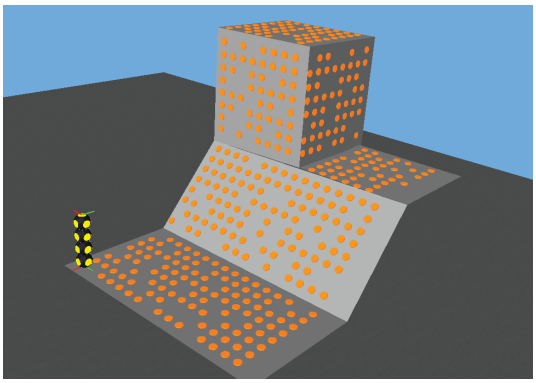
- Successful reaching or not of the final position.
- Number of moves needed to reach the final position: a move is considered as the sequence of motor positions between two connections. The number of moves corresponds to the number of connections.
- Average angular displacement of the active units for the completion of the task, computed using an estimate of the real time needed to perform a move assuming a constant angular velocity.
- Average torque used during the process and per move.
- The number of modules used to carry out the task (when applicable).



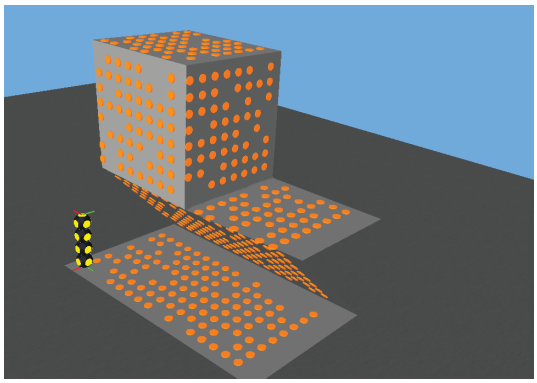
(a) p=0.4 and $\alpha = \frac{\pi}{4}$



(b) p=0.6 and $\alpha = \frac{\pi}{4}$



(c) p=0.8 and $\alpha = \frac{\pi}{4}$



(d) p=0.8 and $\alpha = \frac{3\pi}{4}$

Figure 5.7 – Examples of terrain with various probability p and slope α. Connectors are represented by orange circles and only one metamodule is represented at position (0,0).

Experiment one: terrain complexity

Setup We use in this experiment two metamodules and a single passive cube-shaped object. We vary the number of connectors per terrain by introducing a probability p which determines whether a connector in the regular grid is available or not. We choose four values for p (0.2, 0.4, 0.6 or 0.8) and we randomly generate a set of 50 terrains per value of p by varying the angle of two of the main planes of the terrain (angle α depicted in Fig. 5.2) in the set $\{pi/4; pi/2; 3pi/4\}$ radians as well as the final object position and orientation. Examples of terrains are illustrated in Figure 5.7.

Results The results are summarized in Fig. 5.8 and Fig. 5.9.

Throughout the simulated experiments described in subsection 5.1.4, we observed that the chosen VC was always of type 3. This can be explained by the significantly bigger working space offered by the metamodule in comparison with the single module. We postulate that the use of the chain of lower type would arise only when considering a transport task in

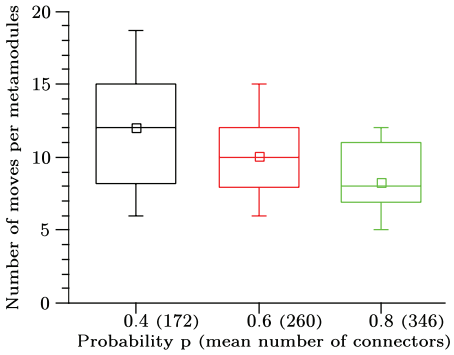


Figure 5.8 – Box-plot representing the number of moves for one metamodule during the successful runs of the algorithm for the different values of p.

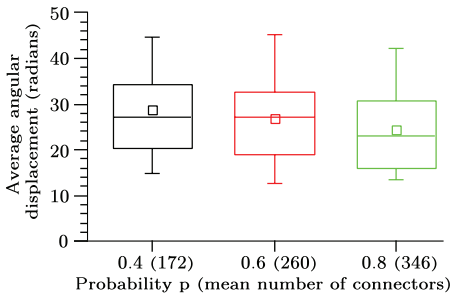


Figure 5.9 – Box-plot corresponding to the average of the modulus of the angular displacement for one metamodule during the successful runs of the algorithm for the different values of p.

which the active units would be allowed to let the passive object on the ground, disconnect and reconnect to reach a previously inaccessible position (due for example to low hanging obstacles) and take back again the object.

Since the number of impossible worlds generated when the probability p was equal to 0.2 was too high to compare it to the other cases, we chose to discard the results related to this value. Some unsolvable worlds include those with no existing path to the final position (too spaced connectors) or with a passive object placed below the slope (when α is equal to $\frac{3\pi}{4}$), in such a way that the passive element is inaccessible without creating collisions. Similarly, when the passive element is situated close to one of the inclined surfaces, the complexity of the manipulation task increases. Given the degrees of freedom of the RB platform considered for the tests, we can discard some of the generated terrains considering they cannot be solved using the kinematic chains involved. Finally, the overall success rate of the algorithm was around 97% for all the solvable worlds generated. The reason for failure in some of the solvable worlds is due to the robot kinematic that prevents some moves in the given configuration of the terrain.

We can see on Fig. 5.8 that the number of moves required to reach the final position increases with the decrease in the number of connectors. This can be explained by the need for the active units to go back and forth on some positions before being able to reach a position with

the correct orientation for the next step. When considering a sufficiently large number of connectors the effect of the heuristic function selected for the path planning (described in subsection 5.1.3) can be observed, with a significant decrease in the number of moves needed to reach the goal (the average number of moves to perform the manipulation task was 11 per metamodule). Nevertheless, the average angular displacement per metamodule (depicted in Fig. 5.9) remains almost constant for varying p. This can be related to the previous observation about the number of moves, since more small moves will be equivalent to less large moves in terms of angular displacement.

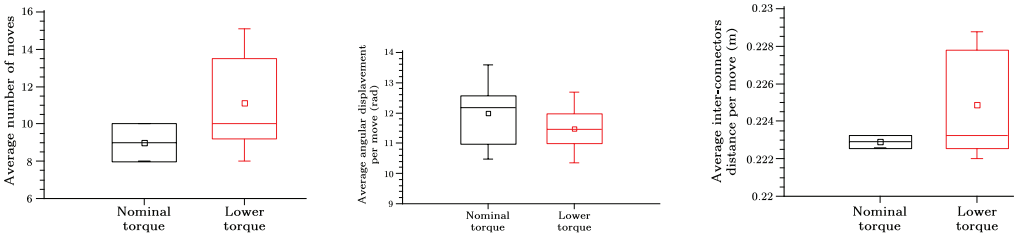
We also observed that the value of the angle α has no significant effect on the success rate or on the number of moves required. A possible explanation would be that the variation of the angle α does not induce a fundamental change in the topology of the terrain when moved. This topological stability of the terrain coupled with the randomness of the final position of the object, does not favor a given strategy in terms of number of moves or movement amplitude.

Experiment two: torque limitation

The goal of this experiment is to analyze the impact of the torque limitation imposed on the motors of the active units on the number of moves needed to reach a given position as well as on the corresponding average angular displacement per move.

Setup We use the same experimental terrain as the one used in our first experiment except we only consider the first plateau (that lies on the ground) as our grid setup. We do not vary the number of connectors since it would have a correlated effect with the torque value on the size of the moves (fewer connectors imposes larger moves). We choose two values for the torque for each type of motors: a low value corresponding to a fifth less than the nominal torque, and a value equal to the nominal torque of the motor (which will serve as a control case). We must stress out that the torque estimate T^r, even if more realistic than T^w, remains an upper bound of the actual torque, so that any moves generated with any of those three selected value will require a strictly lower torque than the nominal torque of the motor (neglecting the friction effects and the dynamics of the movement). We only consider one metamodule of type PAR placed at position $(0,0)$ on the grid and having to reach the connector $(12,5)$ with the same orientation and type of connection. We choose as nominal torque values $5Nm$ and $2.5Nm$ for the diagonal joints and the central one, respectively and $4Nm$ and $2Nm$ for the lower torque values. We repeated the experiment ten times for both torque values to take into account the variability of the solutions resulting from the IK solver. The results are summarized in Fig. 5.10.

Results We observe that the number of moves needed to reach a position is higher when considering a lower value of the torque for the joint (Fig. 5.10a), which can be explained by the necessity for the active units to restrain the amplitude of its movements (this trend can be



(a) Average number of moves.

(b) Average angular displacement per move (in radian).

(c) Average inter-connector distance per move (in meter).

Figure 5.10 – The different results from the torque experiment. The number of moves tends to be higher for lower torque values (a) which correlates with an overall smaller angular displacement (b). However, the motion of the metamodules on the grid in terms of distance between two consecutive connections is almost stable between the two torque values (c) which tends to indicate that the higher torque value allows for more direct moves than the lower one.

seen in Fig. 5.10b). Overall, the distance between two consecutive connectors part of the path to reach the goal connector is almost the same for both torque value (Fig. 5.10c). It means that the torque constraint impacts the type of move itself, since the decrease in the average angular displacement induces a lower amplitude of the overall movement.

Hardware results

We tested our framework using RB hardware modules (this experiment can be seen at [1])[1]: we use a metamodule placed on a 2D grid to grab a passive cube of $0.11m$ edge-length with connection plates on every side and hand it over to another metamodule attached to a grid of connectors placed above the first one (the setup is depicted on Fig. 5.11). The passive element is placed at its initial position in a holder that allows easy picking and avoid sliding of the passive element. The element is also maintained in position using small magnets. The positions of the metamodules have been computed using the planner described in section 6.2. The experiment is performed in open-loop and the environment is fully known. In order to facilitate the alignment between the active connection mechanism and the connectors on the passive object and on the grid, we equipped every ACM and passive connector with small magnets. The magnets are used for guidance only and the connection/disconnection sequence is performed using the grippers of the ACM.

Results During the hardware experiment, we observed that the elasticity in the metamodule structures (at the level of the joints and the level of the ACM connected to the grid) induced a significant error in the final position of the connecting surface. That is the reason why we added magnets to provide the compensation needed to achieve a successful connection

[1] It should be noted that the two metamodules were remotely controlled at low speed to avoid dynamic effects.

Figure 5.11 – The experimental setup: one metamodule is connected to the grid (number 1) above the second metamodule (number 2). A passive element (black cube, labeled as 3) is maintained in position using a magnet. Metamodule 2 will grab the passive element and hand it to the first metamodule.

between the passive element and the metamodule ACM.

5.1.5 Conclusion

We presented in this chapter a complete collaborative manipulation and transport framework using self-reconfigurable modular robots to handle passive elements in a structured environment equipped with connectors. Our method is based on a hierarchical planner that uses the notion of virtual kinematic chain to compute way-points and collision free paths. We also included an on-line computation of the applied torque to the different motors of the active units to favor hardware friendly moves. Our approach proved to be robust and efficient in arbitrary simulated environments, with a success rate of around 97%. An example of a manipulation step using two RB metamodules and one cube-shaped passive element has been successfully demonstrated in hardware.

5.2 Conclusion

In this chapter, we presented a novel approach to handle passive elements using SRMR. We proposed a hierarchical manipulation framework enforcing a strict decoupling between hardware specific elements and high level platform independent parts. Our low level kinematics planner is based on the Levenberg-Marquart IK algorithm that provides a generally more stable and fast convergence than a classic Newton Raphson approach (although it could also be improved to make it complete, using for example the method mentioned in [247, 248]). Collision detection is provided by the RTT-Connect algorithm and we further prune the valid states

by introducing a torque estimate for the different move, ensuring more hardware friendly movements.

We tested our method in simulated environments for various conditions. We investigated the influence of the number of available connectors in the terrain on the number of moves and average angular displacement (which we proposed as an estimate of the energy needed to perform a move). We noticed that the fewer connectors the fewer and larger the moves. We also checked the impact of two different torque limitation higher bounds and noticed that the higher the torque the longer the moves (in terms of average angular displacement of the different joints). We presented a proof of concept of manipulation with the RB hardware platform using a cube with six connectors and a two metamodules. In Appendix B, we describe our preliminary results demonstrating how the augmented self-reconfiguration problem can be reduced to a multi-robots planning problem using our previously define hierarchical manipulation framework and a deconstruction procedure based on centrality measurements in graphs.

Conclusion

SELF-RECONFIGURATION is a very challenging problem for modular robots, considering the extremely large search spaces resulting from the number of connection ports and degrees of freedom available in the modular structures considered. Throughout this part, we have seen how those constraints forced researchers to come up with novel solutions to be able to fully exploit the great flexibility of SRMR.

We have first precisely defined the problem of self-reconfiguration, as well as the related tools and metrics to evaluate the complexity of an approach and its efficiency. The use of metamodules has been described in details since they can be used to build a theoretical optimal planner for SR, but at the cost of the transferability to actual hardware platforms due to its lack of realism.

Among the different methods proposed to provide SR capability to groups of SRMR, two main categories can be distinguished.

The first one comprises the techniques relying on heuristics while the second one includes the methods supported by exact approaches. The heuristics based techniques are very diverse, ranging from genetic algorithms to hormone based control. We focused on a very promising and scalable approach based on the gradient attraction mechanism. In this method, the robots are guided towards seed positions (or attractors) in a fully distributed fashion. We described our own contribution based on the work by K. Stoy [243]. We introduced different strategies to manage the close range interaction between the units and show how they impact on the number of deadlock situations and collisions. Our proposed method tends to decrease the number of deadlocks for complex structures but we would need more large scale experiments to fully validate this point. The scalability of the method comes at the cost of completeness since no built-in convergence can be ensured with arbitrary dense goal structures.

The exact approaches are supported by theoretical results that guarantee the completion of the task and provide complexity measurements. One of the most promising techniques has been proposed by R. Fitch et al. [90] and rely on a Markov Decision Process formulation of the SR problem with a hierarchical planner decoupling the hardware kinematics from the high level planning process. We simplified this approach and introduced a reward based reconfiguration framework using metamodules as basic active units. We have demonstrated

that our technique was efficient in complex environments. Nevertheless the simplification we made comes with the requirement of an exponentially complex (in the number of available connectors) pre-computing step and the necessity of fixing at the beginning of the algorithm the final desired position of the different units.

Both categories were comprised of methods relying on simplified versions of the hardware, abstracting away the imperfections of the different platforms, such as bending effects or connection misalignment. None of them were also able to consider heterogeneous systems composed of fully passive elements and active units. We filled this gap by proposing a novel manipulation framework based on a hierarchical planner decoupling hardware specific routine from the high level planning function. We integrated a built in torque estimation into our motion planner to further close the gap with the hardware. We tested our approach on terrains with various complexities and investigated the impact of the torque on the number of moves needed to complete the task and on the average angular movements. We have shown that the smaller the number of connectors or the larger the torque limit value the longer the moves (in terms of average angular displacement). We introduced preliminary results showing how the augmented self-reconfiguration problem can be reduced to a multi-robot planning problem using the manipulation planner that we proposed.

We briefly presented different studies we conducted in hardware to try to close the gap when porting our algorithm to the experimental platform. We described an exploratory work on the characterization of the elasticity effects in a metamodule. We show how we could compensate for connection misalignment using a low end camera integrated in a module as one of the connector. We finally mentioned a study we conducted to better evaluate the performance of our connection mechanism. Those approaches still need to be integrated into our reconfiguration planners to be tested and evaluated.

Locomotion Part III

Introduction

SELF-RECONFIGURABLE modular robots can dynamically change their topology which makes them suitable platforms to be used as rapid prototyping tools to study locomotion in and adaptation to unknown environments. In addition to this off-grid locomotion capability, hybrid self-reconfigurable modular robots can use embedded connectors in the environment to perform locomotion through reconfiguration, also referred as on-grid locomotion.

This adaptation capability brings additional challenges in terms of locomotion control of the resulting structure, since the morphology of such a structure might not be known beforehand. One successful approach to control the locomotion of modular structures are bio-inspired Central Pattern Generators (CPGs) [121], a network of coupled oscillators that allows to generate complex locomotion behaviors with a reduced set of control parameters. One of the main difficulties when using CPGs is the design of the best suited network for a given morphology. This step is most of the time based on trial and error and can quickly become time-consuming for large irregular structures. In order to find the most suited set of control parameters for the CPG network, optimization methods, such as Powell's Method [211] or Particle Swarm Optimization [136, 209], can be used [210]. In these methods, the time required to optimize the gait of a structure is highly correlated to the number of parameters to optimize.

In this part, we first present (chapter 6) an efficient planner to perform locomotion through reconfiguration using movement primitive and the well-known $D^\star$ algorithm. We present hardware results supporting our work using a single RB module. We then describe (chapter 7) an automated method to generate reduced control networks for the locomotion of arbitrary structures made of modular robots for time critical applications.

6 On-grid locomotion

References and contributions
This chapter is based on the following publication:

S. Bonardi, R. Möckel, A. Spröwitz, M. Vespignani, and A. J. Ijspeert. Locomotion through reconfiguration based on motor primitives for Roombots self-reconfigurable modular robots. ROBOTIK Conference, Munich, Germany, 2012.

My contributions were:

- *theoretical description of the planner and implementation.*
- *writing of the control scripts for the hardware experiments.*

The external contributions were:

- *help for the hardware experiments.*

6.1 Introduction

The problem of on-grid locomotion can be viewed as a sub-case of the SR problem, in which the robotic structure has to reach a given position in space, either on embedded connectors in the environment or using the connectors of its composing modules as substrate. The main difference between the approach we proposed in this section and the SR frameworks described in part II is that there is no high level coordination manager to take into account multiple active units and the order in which to build a final structure. On the contrary, the previously mentioned SR frameworks can also be used for locomotion through reconfiguration but add an additional degree of complexity in comparison with the method we propose in this chapter. In the following experiments, we propose to control a single RB module through a hierarchical, online running locomotion-through-reconfiguration planner based on the $D^\star$

algorithm [241] and *composed motor primitives*. A state-of-the-art reconfiguration planner that takes the kinematics of modular robots into account has been presented by Fitch et al. [90] in simulation (see subsection 4.2.2 for a detail review of this approach). However, this planner has not yet been demonstrated on real hardware. In fact, most hardware experiments so far have been using pre-computed reconfiguration sequences [180, 132]. The major advantage of our online planner over these techniques is that we can take into account online changes of the environment and incorporate readings from sensors.

We start by presenting in section 6.2 the RB movement planner allowing a RB module to reach any position on a 2-D grid. We then describe in section 6.3 experimental results, both simulated and using the actual RB hardware.

6.2 Planner

The goal of our planner is to compute a path (not necessarily optimal) on a 2-D grid from a start (S) to a goal (G) position. We built a hierarchical planner based on $D^\star$, a well established low-level path planning algorithm [241]. On top of it we added a high level planner which transforms the path computed by $D^\star$ into a sequence of basic *movement primitives*. In section 6.2.1 we describe the algorithm used to compute the shortest path inside our 2-D grid. In section 6.2.2 we define the motor primitives alphabet on which we base our high level planner presented in section 6.2.3.

6.2.1 Low level planner

The problem of finding a path on a 2-D grid can be viewed as a path-finding problem in a graph. We consider a grid in which regular obstacles can be inserted but we assume that the dimensions of the grid are constant and the positions of the obstacles are fixed. One popular algorithm for solving path planning in 2-dimensional grids is the $A^\star$ algorithm [105]. Unfortunately, this algorithm is not really efficient at managing movable obstacles: a re-planning of the path is executed each time an obstacle is added or moved. Since we would like to allow moving obstacles (e.g. to represent other moving modules), we decided to use the $D^\star$ algorithm [241] which can be viewed as an evolved version of the $A^\star$ [140]. A detailed comparison between $A^\star$ and $D^\star$ can be found in [166].

6.2.2 Motor primitives

In order to ease the control of a RB module, we introduce a set of basic moves called *motor primitives*. Our goal is to perform locomotion through reconfiguration by using a sequence of attachments and detachments of the module on a 2-D grid. Although a RB module can contain up to 10 ACMs, in our experiments we consider only one ACM per outer hemisphere ($H0$ and $H3$, represented in Fig. 2.1a). When a RB module is connected to a grid using the

	Atomic motor primitives			
	P_1^0	P_2^0	P_3^0	P_4^0
$\boldsymbol{\tau}$	$(0,1)$	$(0,1)$	$(1,0)$	$(1,0)$
$\boldsymbol{\rho}$	π	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\pi$

Table 6.1 – The four different atomic motor primitives for H_0 connected to the grid.

ACM in $H0$ or $H3$, it can only move in one of two directions (Fig. 6.1b). These two orthogonal directions form a coordinate system, the *relative coordinate system* (R^i_{rel}), where i is either 0 or 3 and corresponds to the connected hemisphere.

Atomic motor primitives

We define an *atomic* motor primitive (AMP) as a set of motor angles allowing the module to translate by a distance of one unit of the grid. The translation is represented by a vector τ. The direction of this translation is parallel to one of the axis of R^i_{rel}. During an AMP the two relative coordinate systems are inverted (R^0_{rel} becomes R^3_{rel} and vice versa), and an absolute rotation ρ between them takes place. An AMP is fully characterized by the couple (τ, ρ). We define four different AMPs valid when H_0 is connected to the grid and their equivalent when H_3 is connected. The main difference between P^0_i and P^3_i is the order in which the motor angles are sent to the module. The AMPs will be denoted by P^j_i with $i \in [1..4]$ and $j \in \{0,3\}$. The four AMPs for H_0 connected to the grid are summarized in Table 6.1.

Composed motor primitives

We introduce a set of *composed* motor primitives (CMP), defined as the concatenation of one or more AMP, to simplify the planning of the sequence of motor primitives to move from the start to the goal position. We define eight CMPs, that represent the motor primitive *alphabet*, to cover the eight direct neighbour cells (A-H shown in Fig. 6.3a) on a flat grid. The sequence of AMP composing the CMP alternate between P^0_i and P^3_i, following the connection and disconnection of the two RB hemispheres.

For each CMP we defined the notion of a *spanning area* (SA), which corresponds to the grid positions crossed by the module during the execution of this CMP. This notion will be used later for checking obstacle avoidance.

6.2.3 High-level planner

We introduce a high level planner to find the sequence of CMPs required to follow the path found by the low-level planner. From the set of grid positions found by the low level planner,

the high level planner computes the sequence of CMPs allowing the module to follow the path toward the final position. Using the low level planner, we are able to find the set of grid positions that needs to be crossed to reach the final position.

The main steps of this high-level planner can be summarized as follows:

1. Find the shortest path $C = (c_0, c_1, ..., c_n)$ to the goal position using the low-level planner (c_0 is the initial position, c_n the final position and $c_{i\in[1..n-1]}$ are the intermediate positions).
2. For every point $c_i \in C$ starting from the initial position, use the CMP that follows direction $c_i c_{i+1}$.
3. For every selected CMP, check whether its spanning area intersects with an obstacle.
   - If *yes*, compute the set of neighbouring positions of c_{i+1} and select the reachable one (denoted as N_r). For every point in N_r:
     - Modify the initial path to incorporate this new point.
     - Find the new CMPs matching this new path.

     If no points lead to a valid sequence, go back to the previous location c_{i-1} and repeat the process.
   - Otherwise, continue.

The process ends when the goal position has been reached.

6.3 Experimental results

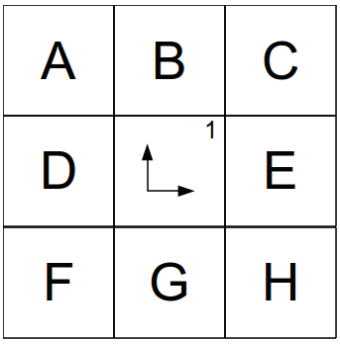
6.3.1 Hardware experiments

We extensively tested the RB hardware performing both locomotion through reconfiguration experiments on horizontal and vertical grids as well as RB modules performing a transition between horizontal and vertical grids (see Fig. 6.2 for an illustration). Movies can be found on the Roombots website [2]. We concentrated first on open-loop experiments: RB modules are PID position-controlled through relative position sensors at each joint but no additional sensors for example for sensing the alignment of a module with the grid have been used.

We tested all sequences to reach all neighbouring positions on a horizontal 2-dimensional grid shown in Fig. 6.3a. Fig. 6.1 shows snapshots from the CMP sequence A. For locomotion on a horizontal 2-D grid the RB hardware is sufficiently reliable. The design of the ACM supports the modules during reconfiguration to overcome elasticity in the joints and connectors as well as backlash in the gear boxes and supports self-alignment of a module with the grid without the need of additional control or sensing. We had only 1 out of 20 connection trials failed

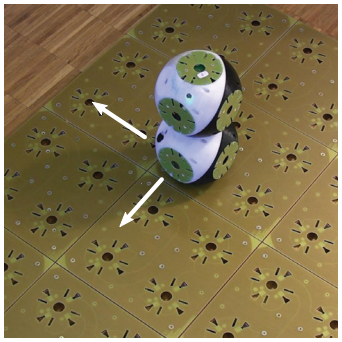
(a) Single RB module.

(b) Initial position.

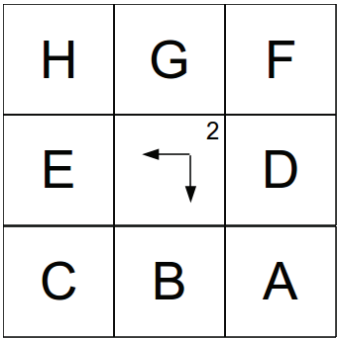


(c)

(d) Connection and disconnection.

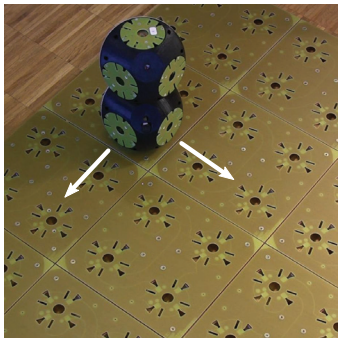


(e) Position B.

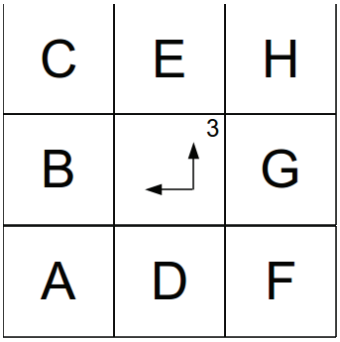


(f)

(g) Connection and disconnection.

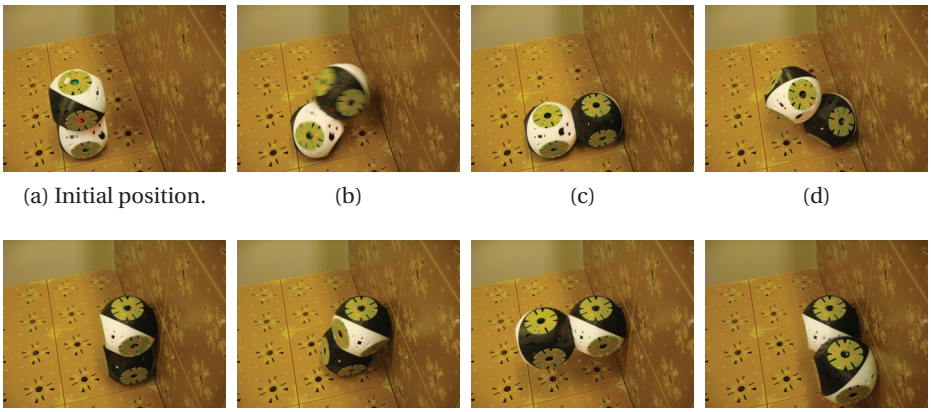


(h) Position A.



(i)

Figure 6.1 – Composed motor primitive *A* (white arrows indicate the 2 possible directions of movements): (a) RB module on a 2-D grid. (b), (d), (e), (g), and (h) illustrate the RB module following CMP *A*. (c),(f), and (i) show the relative referential and CMP alphabet for RB module configuration in (b), (e), and (h), respectively.



(a) Initial position. (b) (c) (d)

(e) (f) (g) (h) Final position.

Figure 6.2 – By a well coordinated sequence of DOF movements that we calculated using inverse kinematics a single RB module can approach a concave corner and climb a wall. This experiment was done in open loop. We could not climb further than position (h) because of elasticity effects in the RB hardware.

(success rate of 95%) where the gripping range of the ACM was not sufficient to overcome elasticity in the module's joints. This was typically because the ACM that was supposed to form a connection with the grid was slightly rotated with respect to the grid so that only one or two of the four ACM grippers could grip into a hole on the grid while the other grippers collided with the grid.

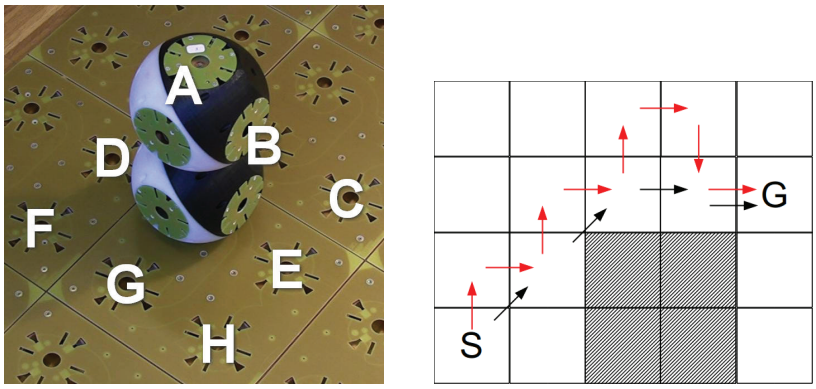
When climbing on vertical surfaces the connection process fails more often since gravity is bending the modules due to the elasticity in the joints, RB shells and connectors. To improve the performance and further increase the success rate, we are currently working on an improved RB design featuring less elasticity in the joints and a bigger ACM gripping range. We are also working on bending detection using infrared distance sensors and active compensation.

6.3.2 Planner results

In order to test our planner, we performed three different types of experiments in a simulated environment representing a 20×20 regular 2-D grid. The initial condition of the module (orientation, values of the degrees of freedom,...) is the same in all the experiments.

In the *first experiment*, we exhaustively tested the planner by trying to reach all the positions around the initial position of the module, within a range of 2 grid units. We did not include any obstacles on the grid. The success rate for this experiment was 100%.

In the *second experiment*, we generated a single squared obstacle of a dimension randomly chosen between 1 and 15 grid units that we randomly placed on the grid. The start and goal



(a) (b)

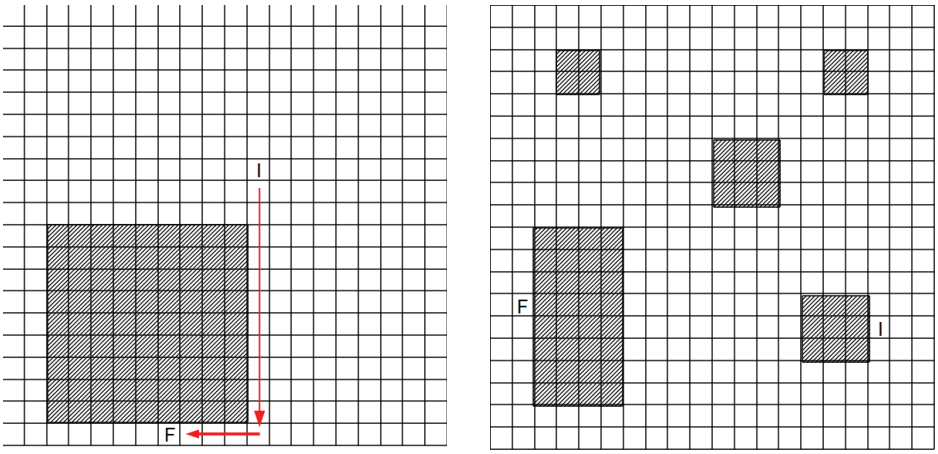
Figure 6.3 – (a) shows the grid with Roombots module. The eight nearest neighbour cells are labelled with letters A-H. In (b) an example of planning result is depicted: the goal position, G, is reached from the start position S using the CMPs C, H, G and B (in red) based on the path found by the $D^\star$ algorithm (in black). The hatched area represents an obstacle in the grid.

position of the module were randomly chosen as well. We generated and tested 300 worlds. The success rate of the planner is 100% for worlds that contain at least a solution. The worlds for which the planner was not able to find a path are in fact worlds with no existing path between the initial and the final position (see Fig. 6.4a for an example of such a world).

In the *third experiment*, we fixed the number of obstacles, their dimensions and their position (as illustrated in Fig. 6.4b) and we randomly chose the start and goal position of the module. We performed 300 trials. The success rate of the planner was 70% on average. The worlds in which no paths were found correspond to those where either the goal position or the start position of the module were at the boundaries of an obstacle and/or of the world so that the module could not leave or reach this position due to kinematic constraints of the RB module. Although we successfully presented climbing experiments with the real hardware, the planner is currently not capable of using motor primitives for approaching or getting away from walls. Thus all initial and final positions with a distance of less than one cell from an obstacle or world boundary cannot be reached or left with the current state of the planner. We will be able to increase the success rate once we include the same motor primitives in the planning process that we use to approach and leave obstacles for overcoming concave corners. Other world configurations where the planner failed to find a path included those where the initial position of the RB module was placed so that its two possible directions of motion where blocked by an obstacle and/or by the border of the grid, as illustrated in Fig. 6.4a.

6.4 Conclusion

Roombots are designed with the property that a single module can fully autonomously travel through self-reconfiguration to any position on a 2-dimensional grid with a minimum number of three degrees of freedom. We presented a simple but effective online locomotion-through-



(a) A world with no path (b) The fixed world with obstacles

Figure 6.4 – Worlds used during the experiments. I and F are the initial and final position of the module, respectively. The hatched areas represent the obstacles. (a) illustrates a world in which no path can be found. The red arrows indicates the path found by the $D^\star$ algorithm. (b) depicts the fixed world used in the third set of experiment with an example of initial and final positions for which no path was found by the planner.

reconfiguration planner based on the $D^\star$ algorithm and composed motor primitives that is closely linked to the real hardware and allows steering RB modules on a grid by simply giving a goal position. This method can be generalized to various hybrid platforms as well as to metamodules, after characterizing their kinematic ability. We presented experimental results illustrating the reliability of RB moving on 2-dimensional grids.

7 Off-grid locomotion

References and contributions
This chapter is based on the following publication:

S. Bonardi, M. Vespignani, R. Möckel, S. Pouya, J.v.d. Kieboom, A. Spröwitz, and A. J. Ijspeert. Automatic Generation of Reduced CPG Control Networks for Locomotion of Arbitrary Modular Robot Structures. Robotics Science (RSS), Berkeley, USA, 2014.

My contributions were:

- *theoretical development of the bio-inspired structures recognition and implementation.*
- *analysis of the data.*

The external contributions were:

- *conducting of the simulated experiments.*
- *help for the analysis of the data.*

7.1 Introduction

In this chapter, we propose an automated method to generate reduced control networks for the off-grid locomotion of arbitrary structures made of modular robots, instead of considering a fully connected network with many parameters. In this work we consider structures that are neither fully linear (i.e. being composed of modules connected only in a open chain) nor fully cyclic (i.e. being composed of modules connected only in closed chain). Our approach is based on the decomposition of the robotic structure into morphologically relevant sub-structures, like body and limbs, and on the automated identification of bio-inspired articulation joints inside the structure. The number of optimization parameters is further reduced using ex-

isting symmetries in the structure. This method can be applied for self-recovery and fast re-optimization after structural changes due to hardware failure or voluntary morphological modifications, for example. A test situation could be the deployment of self-reconfigurable modular robots in an unknown environment where they would quickly need to re-learn some efficient gaits after reconfiguring during a time-critical task.

Our work is driven by the two following hypotheses:

- **Hypothesis 1**: the use of bio-inspired functional patterns and symmetries to generate the architecture of a CPG network controller for locomotion significantly increases the speed of convergence towards an acceptable[1] solution in terms of forward velocity and collision, compared to using a fully open CPG network.
- **Hypothesis 2**: the quality of the solution (in our case the velocity after convergence and the potential internal collisions between modules) is not significantly modified in comparison to a fully open optimization.

This chapter is organized as follows. In section 7.2, we review the existing approaches for controlling modular structures with dynamically changing morphologies. We then introduce in section 7.3 the basic control architecture used in this work. Afterwards we describe our method to find relevant sub-structures inside any modular configuration (section 7.4) and explain how a reduced control network can be generated based on this differentiation and on the concept of distance-based symmetry (section 7.5). We validate our approach using three structures in simulation (section 7.6) and discuss our results (section 7.7) before concluding (section 7.8).

7.2 Related Work

Modular robots offer the advantage of morphology that can change depending on external factors (e.g. changes in the environment) or internal ones (e.g. sudden hardware failure). This flexibility brings an additional challenge in comparison with monolithic robots in terms of design of efficient controllers. Moreover the increase in the number of degrees of freedom with each module added to the structure makes it difficult to hand-design specific gaits. Monolithic robots can also have to cope with a change in their morphology due to hardware issues, requiring as a consequence a re-design of their locomotion controller. The optimization of the set of parameters to generate efficient locomotion is often time consuming.

Since the early work by Yim [282] on the caterpillar locomotion of Polypod robots, several approaches have been proposed for the control of the locomotion of structures made of modular robots. For example, Shen et al. [228] proposed a hormone based method to control

[1] In our case, *acceptable* means capable of moving at a reasonable velocity above some minimum threshold.

the locomotion of CONRO robots, Stoy et al. [246] used role-based control and cellular automata, and Yu et al. [287] described a consensus based approach for the locomotion control of 2D modular robots. CPGs, implemented as systems of coupled oscillators, have been applied for locomotion control by several researchers for distributed locomotion control and various techniques have been investigated [121, 167, 168, 72, 237]. The main drawbacks of the presented approaches is that they consider a fixed morphology and require the manual design of the CPG network, which might prove to be a tedious task for large structures. Some authors [231, 28, 164] used evolutionary methods and co-evolution to make the robot discover its own morphology, or used genetic algorithms to evolve possible gaits for given structures [133]. Those methods are often computationally demanding and time consuming, making them difficult to transfer on-board and on-line. More recently, accelerated learning methods have been investigated [62, 59, 61] based on a distributed and morphology independent learning process. The main difference with our approach is that we propose to optimize beforehand the control network itself instead of approximating the learning reward for the different possible actions. Christensen et al. [58] described a control framework to generate full body behavior based on the decomposition of the structure into bio-inspired parts (like muscle or bones) with pre-defined function (e.g. muscles can contract). The control is then done at the level of those sub-parts, abstracting away their individual components. Although this approach is similar in essence to our method, the main difference is that we propose an automatic detection of bio-inspired joints and symmetries in any arbitrary structure instead of considering predefined structures built from known sub-parts.

7.3 Control Framework

We test our techniques on a simulated model of our self-reconfigurable modular robot Roombots (RB) [238]. Compared to other SRMR, we chose to use RB because of the large variety of gaits that can be obtained with few modules, thanks to their three degrees of freedom capable of both oscillation and continuous rotation.

We considered as locomotion controller a network of coupled non-linear oscillators mimicking the Central Pattern Generators (CPGs) found in many vertebrates [121]. The control inputs for this CPG are the amplitude A_i, the offset X_i, and the phase lags ψ_{ij} of each oscillator i connected to oscillator j. We use one common frequency for all oscillators (ν = 0.2 Hz, according to [177]), bi-directional couplings follow the rule such that ψ_{ij} = -ψ_{ji} and all coupling weights are set to 2. We set the CPG output to produce oscillatory joint angle signals. The coupled phase oscillators are implemented by the following coupled differential equations:

$$\dot{\phi}_i = 2\pi \cdot \nu \cdot \sum_j w_{ij} \cdot r_j \cdot \sin\left(\phi_j - \phi_i - \psi_{ij}\right) \tag{7.1}$$

$$\dot{r}_i = a_i \left(A_i - r_i\right) \tag{7.2}$$

$$\theta_i = r_i \cdot \sin\left(\phi_i\right) + X_i \tag{7.3}$$

where i and j are the indexes of the oscillator, θ_i is the oscillator output controlling the position set point of the Degree Of Freedom (DOF) number i, r_i is the signal amplitude, and ϕ_i the phase. Each oscillator i has a maximum of three parameters that are subject to optimization: the desired amplitude A_i, offset X_i and the phase lag ψ_{ij} to the following neighbor j. More information about CPGs can be found in [121].

In order to find the most efficient gait for each structure, we use a population-based algorithm based on Particle Swarm Optimization (PSO) [136, 63] to generate the set of CPG control input parameters. In this work, we used simulated gait optimization in the simulation software Webots [275].

7.4 Body/Limb Finder

In many vertebrates, the body and limbs are clearly differentiated and play different roles in the chosen locomotion strategy. In order to benefit from this definition of specific sub-structures, we developed an automatic centralized algorithm, called *Body/Limb Finder* (BLF), to automatically identify body and limbs in an arbitrary modular structure. This structure is represented as an undirected graph in which each node represents a module and each edge represents a connection between two modules (as illustrated in Fig. 7.2 top right). The main idea of our approach is that the removal of the body from a given structure will lead to several disconnected elements that represent the limbs. Additionally, the body can be further decomposed into a linear part (or chain part) and/or a cyclic part. A cyclic part is defined as a closed loop of connected modules. The actuation strategy will vary depending on the type of body part considered. In the scope of our bio-inspired control approach, we have introduced a set of rules to identify relevant articulations within the structure: we are able to differentiate between spines, hips, knees and ankles. A special control pattern for each of those units will be introduced in section 7.5. We first present the theoretical aspects related to the detection of those different elements inside a given structure and then describe the validation of our method using a statistical approach.

7.4.1 Theory

Body/Limbs differentiation

The BLF algorithm is primarily based on the notion of *bi-connected components* (bcc). A bi-connected component of a graph is a graph with no articulation vertices, meaning no vertices that, if removed, would lead to a disconnected graph. The BFL algorithm is composed of three main steps (illustrated in Fig. 7.1):

Step 1: decomposition into bi-connected components We first obtain the different components of the graph. This gives us the linear parts (i.e. bcc composed of less than 2 nodes) and the cycles (i.e. bcc composed of strictly more than 2 nodes), if any.

Step 2: finding the cyclic parts of the body We use the following rule to find the cyclic parts of the body. The cyclic parts of the body correspond to groups of modules that are fully linked together, meaning that at least two paths exist between any pair of the group. For each cycle found at step 1, we check the connectivity of the graph resulting from the removal of this cycle: if the remaining graph is still connected then the cycle is not part of the body.

Step 3: finding the linear parts of the body For this step we consider the different nodes which compose the 2-nodes bcc found at step 1. We select the nodes using the following rules (the 2 conditions have to be validated):

i Clustering power: if the removal of the node leads to a number of components for the remaining graph strictly greater than 2 then the node is a linear part of the body[2].

ii Articulation: the node must be an articulation of the graph.

After that, we calculate the shortest path between the selected nodes and we include it in the linear part of the body (minus the intersection with the nodes found at step 2). The limbs are the disconnected components remaining after the removal of the previously found body.

Articulation rules

Spine Every joint inside the linear part of the body is part of the spine, except for hip joints. We chose to consider the cyclic parts of the body as unactuated.

[2]We need to impose this condition because of the case of "long" linear limbs: in a linear limb composed of strictly more than 2 nodes, the central node has a clustering power of 2 and is an articulation, but it is not part of the body.

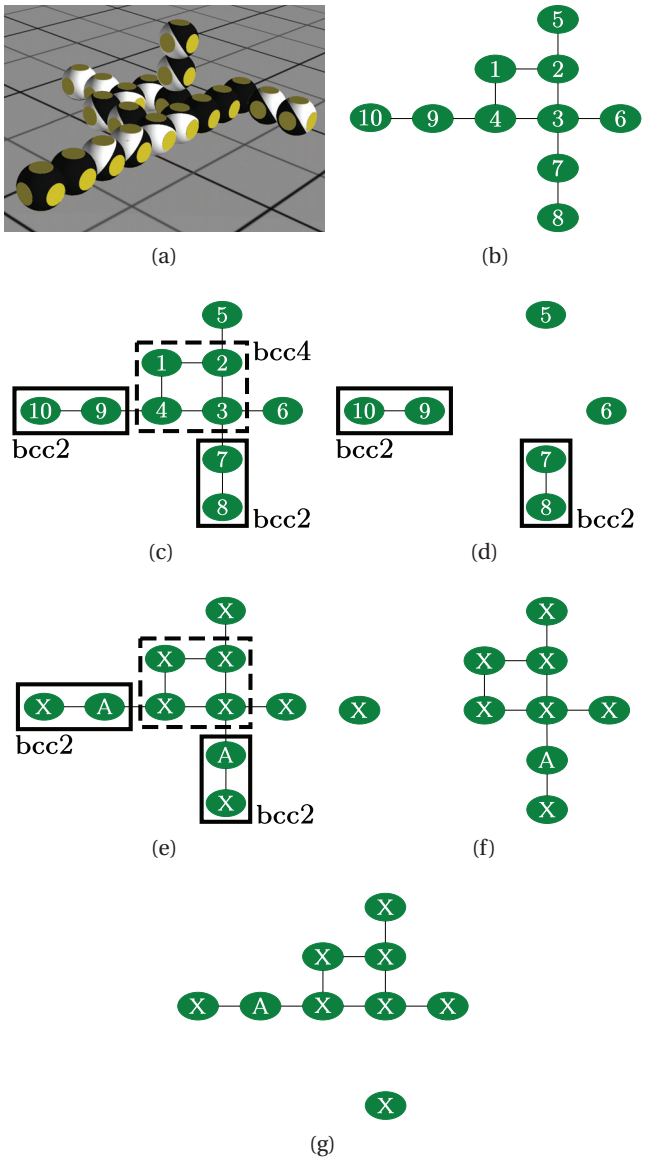


Figure 7.1 – The different steps of the body limbs finder illustrated on *struct10* (a), a structure with 10 modules used later in our experimental validation. (b) We start by converting the robot structure into an undirected graph in which each node represents a module and each edge represents a physical connection. (c) At step 1, we detect the bi-connected components (bcc) in the graph and sort them depending on the number of nodes they contain. (d) At step 2, we detect whether the previously found cycles (i.e. the bcc containing more than 3 nodes) are part of the body by testing if their removal leads to a disconnected graph: since we obtain 4 sub-graphs after the removal of bcc4, it is part of the body. (e) At step 3, we detect in the remaining bcc the articulation nodes, indicated with an A, and check whether they lead to more than two sub-graphs or not (f and g): since the removal of those articulations leads to only two sub-graphs, they are not part of the body.

Hip

- A hip joint can belong either to a limb or to the body.
- A hip is a joint at the frontier between a limb and the body. This joint must therefore have at least one neighboring joint being part of the body.
- We want a hip joint to be as proximal as possible, which means that a hip joint defined as part of the body will be preferred over a hip joint belonging to a limb.

Knee

- A knee joint must be part of a limb. There is only one knee per limb.
- A knee joint is at the center of the limb, between the pod and the hip (we define the pod as the most distal joint of the limb).
- If a hip joint and a foot joint are connected to each other with only one additional joint, no knee joint can be defined in the limb.
- If possible, a knee joint should be at equal distance from a foot joint and a hip joint. If such a joint cannot be found, we would choose the more proximal joint situated at the center of the limb as the knee joint.

Ankle The rules to define an ankle joint are the same as the one describing a knee joint but considering the limb is starting at the knee joint. There is only one ankle per limb.

This set of rules, as well as the result from the BLF algorithm, is illustrated on Fig. 7.2 on a test structure with 9 modules shaped as a quadruped. The unclassified degrees of freedom are considered as *locked*.

7.4.2 Results

Since the notion of body is difficult to define and to characterize, we manually evaluated the "recognition rate" of our algorithm applied to *twenty* randomly generated structures. The number of modules per structure varies from 12 to 32. We considered the RB modules presented in section 7.3 as a test platform, but our method could be applied to other modular robots. Given that the goal of our method is to improve the locomotion control of a structure, we discarded unusable structures, for example, the ones with no limbs at all. The results matched our manual tagging for all the tested structures (see Fig. 7.2 for an example).

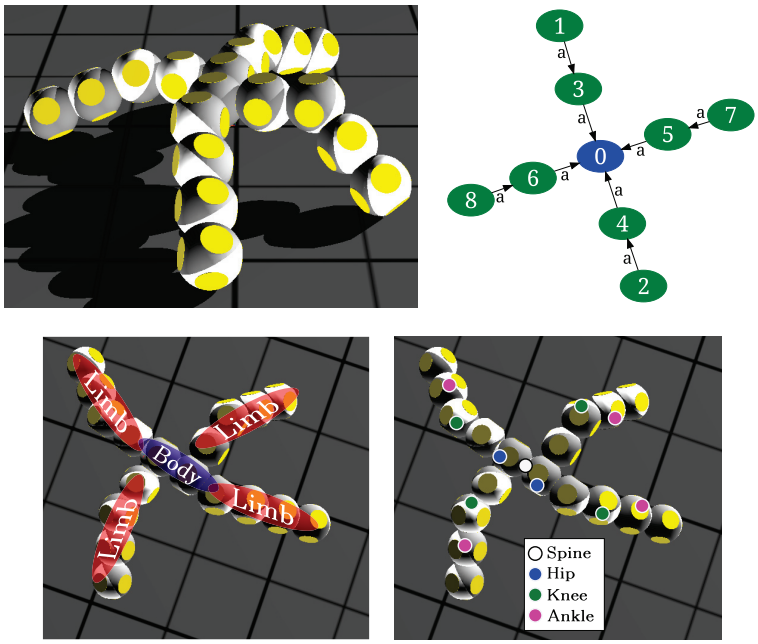


Figure 7.2 – Top left: A quadruped structure composed of 9 RB modules. Top right: the corresponding graph representation, used in the BLF. Bottom left: the detected body and limbs. Bottom right: the articulations detected using the set of rules described in subsection 7.4.1.

7.5 Automatic Generation of Reduced CPG Networks

In this section, we describe the rules applied to design the control network of a modular structure using the coupled oscillators introduced in section 7.3 depending on the results from the BLF. Additionally, we introduce the notion of *distance-based symmetry* using a labeling function for the connection between modules and show how those symmetries can be used to further reduce the number of parameters in the control network.

7.5.1 Articulation network

The CPG network we derive from the articulations found using the BLF is inspired by the typical bone connection network present in many vertebrates (for example, the knees are usually connected to the hips and the hips to the spine). Each spine, hip, knee and ankle joint is driven by a single oscillator. The other degrees of freedom are considered as locked. We also assume that only the linear parts of the body are actuated (the cyclic parts being blocked) and that each spine is composed of a single joint (the most central one of the linear part, tie being solved at random) driven by a single oscillator. If more than one linear part is present in the body, each is controlled using a single oscillator. The parameter boundaries for the amplitude of an oscillator depend on the type of articulation. The coupling rules between the oscillators

are the following (illustrated in Table 7.4):

- The spine oscillators are fully coupled together.
- The hip oscillators are fully coupled together. They are further coupled to the closest spine oscillator in the structure.
- The knee oscillators are only coupled to the corresponding hip oscillator (the one located in the same limb). If no hip is present in the limbs, the knee oscillator will act as a substitute and the same hip coupling rules will apply to it.
- The ankle oscillators are only coupled to the corresponding knee oscillator (the one located in the same limb).

Using this technique, the maximum number of parameters depends only on the number of limbs in the structure and is independent from the number of modules per limb. We consider only three parameters for each oscillator: the amplitude, the offset and the phase shift between the different oscillators. In this formula, we also consider bi-connected connection between the oscillators. If n represent the number of limbs, n_s the number of spine oscillators, and n_h the number of hip oscillators, the number of parameters $P_{reduced}$ can be computed as follows:

$$P_{reduced} = \sum_{i}^{n} 2(\delta_i^a + \delta_i^k + \delta_i^h) + 2 \times \sum_{i}^{n_h} (\delta_{h \in Body}) + 2n_s + \sum_{i}^{n} 2(\delta_i^a + \delta_i^k) + n_h(n_h - 1) + n_s(n_s - 1) + \sum_{i}^{n_h} 2(\delta_i^s) \quad (7.4)$$

where δ_i^a, δ_i^k, and δ_i^h equal 0 or 1 depending if the limb i contains an ankle, a knee or a hip, respectively, $\delta_{h \in Body}$ equals 0 or 1 depending if the hip is inside the body or not, and δ_i^s is equal to 0 or 1 if the hip is connected to a spine or not.

For a fully open network controlling a structure with m joints, each of them represented by one oscillator coupled to its closest neighbor, the total number of parameters P_{open} is equal to

$$P_{open} = 2 \times m + \sum_{i}^{m} (\delta_i^c) \quad (7.5)$$

where δ_i^c is the number of connections between oscillator i and its neighbors.

7.5.2 Distance-based symmetry

In order to further reduce the number of parameters required in our control network, we use geometrical symmetries between the limbs in the structure. If two limbs are considered symmetric, the corresponding oscillators share the same amplitude and the same offset (the

phase shift remaining open to avoid restricting the possible gait patterns). To capture the geometrical organization of a given structure, a label for the connection between modules has to be defined. The label expresses the physical relationship between modules by encoding the source connector, the destination connector and the rotation between the 2 modules. We propose to use *Cantor polynomials* as a labeling function. Cantor polynomials are the only polynomials of degree 2 bijective from $\mathbb{N}^2$ to $\mathbb{N}$. They are defined by the following formulas:

$$f: \begin{array}{ccc} \mathbb{N}^2 & \longmapsto & \mathbb{N} \\ (a,b) & \rightarrow & \frac{[(a+b)^2+3a+b]}{2} \end{array} \tag{7.6}$$

and

$$g: \begin{array}{ccc} \mathbb{N}^2 & \longmapsto & \mathbb{N} \\ (a,b) & \rightarrow & \frac{[(a+b)^2+3b+a]}{2} \end{array}, \tag{7.7}$$

where $\forall (a,b) \in \mathbb{N}^2 \quad f(a,b) = g(b,a)$.

In order to obtain a bijection from $\mathbb{N}^n$ to $\mathbb{N}$ we only need to compose f or g by itself. In the case of the RB platform, we need to associate a unique integer to any given set of 3 values representing respectively the source connector (we called it a), the destination connector (we called it b) and the type of connection (we called it c). Thus, the labeling function that we are going to use is defined as follows:

$$l: \begin{array}{ccc} \mathbb{N}^3 & \longmapsto & \mathbb{N} \\ (a,b,c) & \rightarrow & \frac{[(f(a,b)+c)^2+3f(a,b)+c]}{2} \end{array}$$

We can now associate any tuple $(source, destination, type)$ with a unique positive integer. The only issue with this labeling system is that we have to take into account the orientation of the connection (a bijective function cannot be symmetrical and, as a consequence, if we switch the source connector and the destination connector, the computed label will be different).

The RB platform is equipped with 10 connection ports. Nevertheless the connectors placed on one outer hemisphere are equivalent considering a rotation of $\pi/3$. Similarly those two hemispheres can also be flipped (which corresponds to flipping the module) without modifying the functional characteristics of the connection. As a consequence the range for the connection ports is reduced to [0..4] instead of [0..9].

The distance-based symmetries in a structure are determined using the information provided by the body/limb finder applied to the graph representing the structure and labeled using the previously mentioned labeling function. Only limbs of the same length are compared. The use of symmetries inside the structure is coupled with the information about the localization of the joint with respect to the body. The connections between modules are sorted into different groups depending on their distance from the body. The labels of the connection inside each limb are iteratively compared among groups: only fully identical limbs are considered symmetric.

7.6 Experimental Results

We considered three RB structures as test cases to evaluate our method. The first structure is a quadruped made of 5 modules with all limbs symmetrical (called *quad5-sym*, shown in Fig. 7.3 on the left). The second structure is the same quadruped but with a limb connected to the spine with a different orientation, so that only 3 limbs are now symmetric (called *quad5-unsym*, depicted in Fig. 7.3 on the right). The last structure is a pseudo random asymmetric structure made of 10 modules (called *struct10*, shown in Fig. subfig:struct10webots). The first two structures were chosen to represent bio-inspired structures, with the distinction between fully symmetric and partially symmetric one. We decided to use *struct10* to test our method on a much larger structure in which no intuitive gait could be engineered and also for which a fully open optimization requires a significant amount of time to converge.

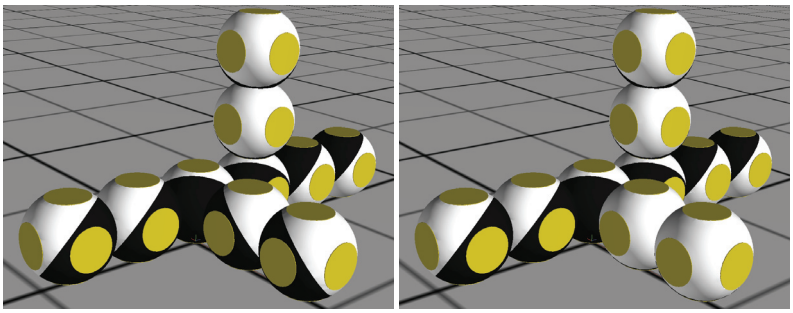


Figure 7.3 – Two of the three test structures: *quad5-sym* (left) and *quad5-unsym* (right). The difference between the two structures is that one of the modules (the one at the bottom right of the picture) is connected using a different orientation.

We compare an optimization of the parameters of the corresponding CPG network for each structure in the following four conditions:

1. Fully open optimization (FO): all the parameters of the network are considered open. One oscillator per dof is used. For each oscillator the amplitude is only constrained to $[0;\pi]$.
2. Body Limbs Finder reduced network (BLF): we use the technique described in subsec-

tion 7.5.1 to generate a reduced network for the structure. The amplitude parameter is constrained depending on the type of articulation considered (see Table 7.1).

3. BLF network and symmetry finder (BLF-SYM): additionally to using the reduced network generated by the BLF, we consider symmetries as described in subsection 7.5.2 to further decrease the number of parameters to optimize. The amplitude parameter is constrained depending on the type of articulation considered (see Table 7.1).

4. Symmetry finder (SYM): we applied distance-based symmetries between the limbs to reduce the number of parameters in the fully open CPG network controlling the structure. This step requires the use of the BLF to determine body and limbs in the structure, but, contrary to the previous case, no specific network structure is derived from this detection.

In terms of search space, the BLF, BLF-SYM, and SYM cases are sub-sets of the FO case. The number of parameters for each structure in the different cases are summarized in Table 7.2. The parameters used for the PSO optimization for each case can be found in Table 7.3. The corresponding CPG networks are depicted in Table 7.4.

Table 7.1 – The boundaries for the amplitude parameter depending on the type of articulation considered.

	Spine	Hip	Knee	Ankle
Min	0	0	0	0
Max	$\frac{2}{3}\pi$	$\pi/2$	$\pi/6$	$\pi/6$

Table 7.2 – The number of optimized network parameters for the three case structures in the four different conditions. The number in parenthesis indicates that the network is the same as one previously defined, and as a consequence, that it was not used.

	quad5-sym	quad5-unsym	struct10
FO	44	44	90
BLF	21	21	26
BLF-SYM	15	17	(26)
SYM	26	30	(90)

We ran the PSO optimization twenty times with different initial random populations for the three structures *quad5-sym*, *quad5-unsym*, and *struct10*. For the latter, only the FO and BLF networks were tested, since no apparent symmetries are present in the structure. The fitness function f chosen for the optimization process takes into account the displacement of the structure and penalizes collisions between modules:

$$f = \frac{d}{t_{total}} \times c \tag{7.8}$$

where d corresponds to the displacement of the robot during the total experiment time t_{total}

Table 7.3 – The fixed parameters for the PSO optimizations for the different structures.

Parameters	**quad5-(un)sym**	**struct10**
No. particles	80	160
No. iterations	800	
maximum velocity	0.6	
social factor	2.05	
cognitive factor	2.05	
constriction factor	0.729	
exp. duration t_{total}	30s	

and c is a penalization factor used in case of self-collision equal to 0.001 if there is a collision and 1 otherwise. c was determined experimentally.

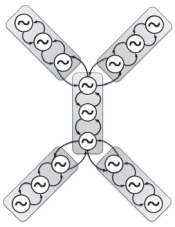
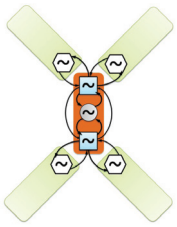
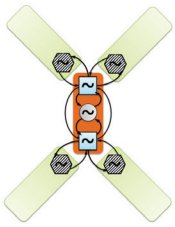
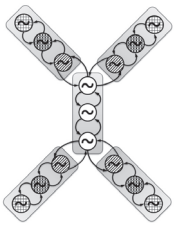
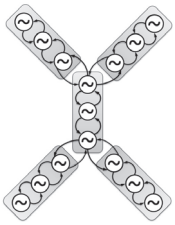
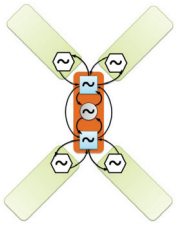
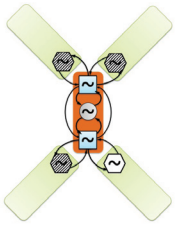
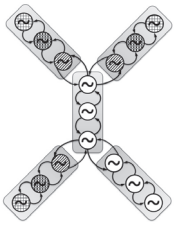
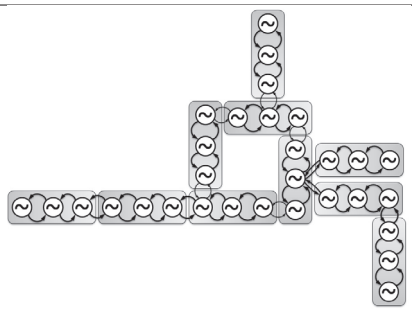
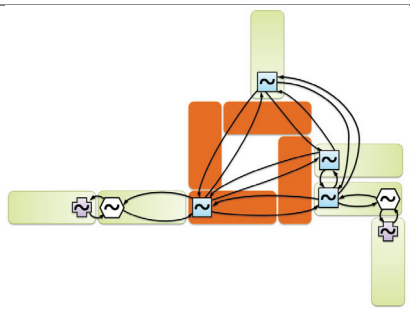
At each iteration and for each of the twenty optimization runs, we only consider the solution with the highest fitness. We then computed the mean value of these sets of twenty best solutions and repeated the process for the three structures considered. The results are depicted on Fig. 7.4.

In order to compare the results of the best particles obtained at each iteration using the different network topologies, we performed single factor ANOVA tests (we tested the homoscedasticity of the residuals using the Levene's test and we assumed they were normally distributed). The results are summarized in Table 7.5.

7.7 Discussion

Our first hypothesis was that the fact of using a reduced CPG network generated using bio-inspired rules would significantly reduce the number of iterations needed to obtain an acceptable gait for a given structure. As we can see in Table 7.5, the solutions generated using the reduced CPG network dominated the fully open population at least to the $30th$ iteration for the *quad5-sym* and *quad5-unsym* structures. For the bigger structure, we can clearly notice that restricting the search space by introducing automatically generated prior knowledge and boundaries to the parameters positively impacts the results: the fitness values are significantly better up to the 200th iterations and the convergence is significantly faster. We also observed that out of twenty runs of the *struct10* FO cases, we only obtain four valid solutions that converge to a set of parameters that did not induce self-collisions, emphasizing the need for a more robust method. If we select those solutions and compare them to four randomly chosen solutions from the BLF set, we observed that the BLF solutions are significantly better at the beginning of the optimization process (until iteration 71) before being dominated by the FO solutions (as illustrated in Fig. 7.5), which remains consistent with our first hypothesis. Similarly, we observed in all cases that no significant differences could be found between our three proposed methods and the standard FO case at convergence, which remains consistent with our second hypothesis.

Table 7.4 – The different CPG networks for the three tested structures. In the fully open case, the circles represent the generic oscillators. For the BLF and BLF-SYM cases, the limbs are represented in green, the body in orange, and the shape coding is as follows: the spine oscillator are circles, the hip oscillators are squares, the knee oscillators are hexagons, and the ankle oscillators are crosses. For the BLF-SYM and SYM cases, the symmetric oscillators are indicated with the same stripe type.

Structures	FO	BLF	BLF-SYM	SYM
quad5-sym				
quad5-unsym				
	FO		BLF	
struct10				

The reduced networks seems to be less sensitive to local minima resulting from the complex optimization landscape, as illustrated on Fig. 7.4c. The results we obtained are as expected, since reducing the search space is known to have a positive effect on the speed of convergence, but through our study, we managed to validate our hypothesis and to quantify for how many iterations it is still valid.

One typical test situation for our method would be some hardware failure of a self-modular robot during a time critical mission: the robot is then forced to reconfigure into a new shape and to re-learn how to move. It can, for example, connect to a remote cluster service to ask for new possible gaits but it cannot wait until the full convergence (meaning hours of computation for large structures). A similar scenario could involve a monolithic robot having to deal with a change in its morphology after some hardware issue. Our approach could be used in those two cases to characterize the new configuration of the robot and to propose corresponding reduced

Table 7.5 – Iteration number after which no significant difference (with a $p-value < 0.05$) can be found between the samples of best individuals corresponding to the four network topologies. The values marked with a star indicate the iteration number before which no significant difference (with a $p-value < 0.05$) can be found between the samples of best individuals. The three numbers correspond to the three structures, respectively, from left to right, *quad5-sym*, *quad5-unsym*, and *struct10*. Before this iteration number, the ordering between the different network topologies can be seen on Fig. 7.4. *None* means that the two samples tested were not significantly different. *X* means that the test was not performed because the networks were not tested.

	BLF	**BLF-SYM**	**SYM**
FO	32/96/202	30/88/X	80/224/X
BLF	-	92/None/X	57/84*/X
BLF-SYM	-	-	35/82*/X

CPG networks to speed-up the optimization of the gait. With our proposed technique, after only five iterations (around one minute of optimization on average on our computer cluster[3]) we manage to provide a gait with a fitness value of 0.017, 0.024, and 0.016 (BLF, BLFSYM, and SYM) against only 0.005 in the FO case for *quad5-sym* (at least 3 times less on average, as depicted on Fig. 7.6). Similarly, for the *quad5-unsym*, the fitness after five iterations is almost an order of magnitude bigger with the reduced network (minimum 0.017) in comparison with the fully open case (0.002), as illustrated on Fig. 7.7. Similar trends are can be observed at iterations 25, 50 and 100 (5, 10 and 20 minutes of computation, respectively), as shown on Fig. 7.6 and Fig. 7.7. The solutions found in the FO case in the early exploration phase were often heavily penalized because of the self-collision induced by the large boundaries set for the CPG parameters.

We can also observe on Fig. 7.4a and 7.4b that the reduced networks generated using the distance-based symmetry technique (SYM) obtain better results relatively to the two other reduced networks (BLF and BLF-SYM). This can be explained by the fact that the amplitude for the oscillators has larger boundaries than in the two other cases. A qualitative analysis of the resulting gait showed that in the SYM case, as well as on the FO case, the structure tends to rely much on almost rolling movement of some joints to increase its momentum. On the contrary, in the BLF and BLF-SYM cases, the structure tends to have a smaller amplitude of oscillation and favor animal-like displacement of the limb, making the obtained gaits more hardware friendly. The different solutions are illustrated in the video attachment.

Regarding the portability of the solutions to the hardware platform, we set the parameters of the simulation environment according to the results of a work that used meta-optimization on the RB robot in order to reduce the reality gap between simulation and hardware [177]. This should ensure that the gaits we obtained in this study remained consistent when transferred to a hardware platform.

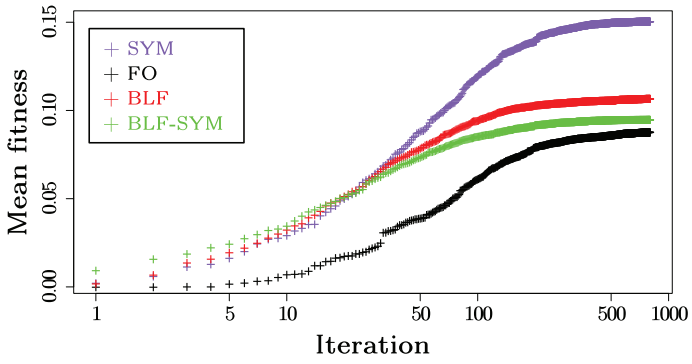
[3]Our cluster is composed of forty 2.00GHz quad-core Intel Xeon E5504 processors.

7.8 Conclusions and Future Work

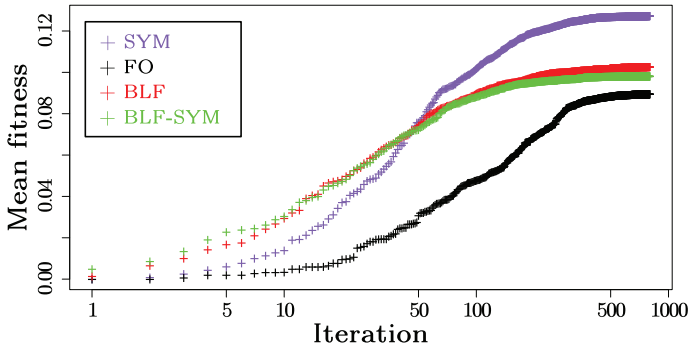
In this chapter, we proposed an automated method to generate a reduced CPG control network for the locomotion control of modular robots. We based our approach on the recognition of bio-inspired patterns in the structure playing the role of spine, hip, knee or ankle. Each of them are driven by a single oscillator with particular boundaries for the optimization parameters and specific coupling rules with the neighboring oscillators, with the goal of reducing the optimization time needed to find acceptable gait. We further reduced the number of parameters required in the optimization by automatically considering the existing symmetries in the structure.

By comparing the results obtained with three different structures, two quadrupeds and one pseudo-random structure composed of 10 modules, we noticed that our method leads to significantly better results during the first iterations, making the goal of re-optimizing a locomotion strategy (for example to cope with an unexpected change in the morphology of the robot due to a hardware failure) online and on-board a reachable goal.

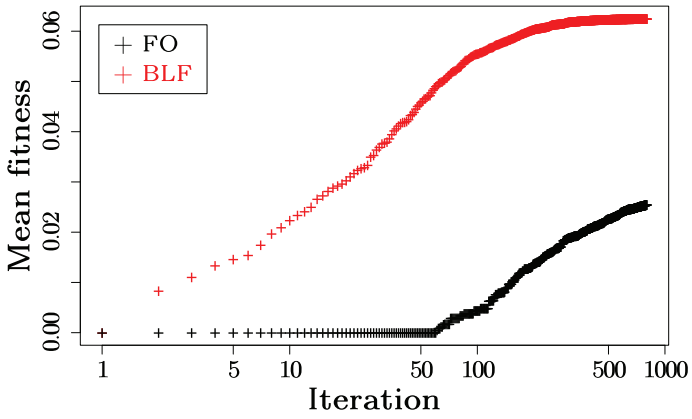
Despite our method being generic, our preliminary study involved a restricted number of structures and was focused on a particular robotic platform. We are planning in the future to further extend our work to different types of modular robots and to increase the number of modules per structure to emphasize the gain induced when using our method.



(a) Quad5-sym



(b) Quad5-unsym



(c) Struct10

Figure 7.4 – The mean value of the fitness function over twenty runs for (from top to bottom) the *quad5-sym*, the *quad5-unsym* structure, and the *struct10* structure. The results are displayed in semi-log scale.

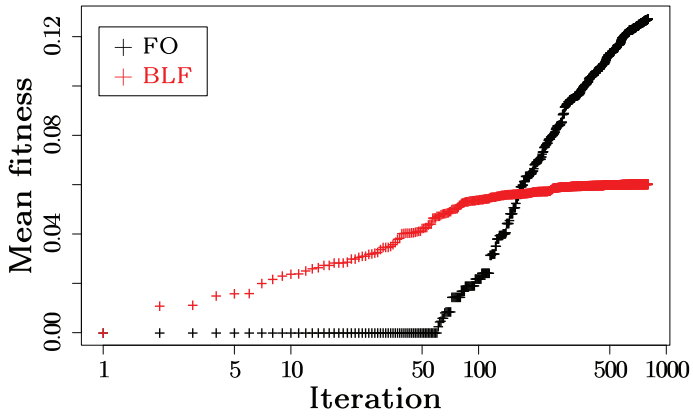


Figure 7.5 – In black, the mean value of the fitness function for the four optimization runs of the *struct10* structure in the FO cases in which no collisions were observed in the optimization best solution. In red, the mean fitness value of four randomly chosen optimization solutions in the BLF case.

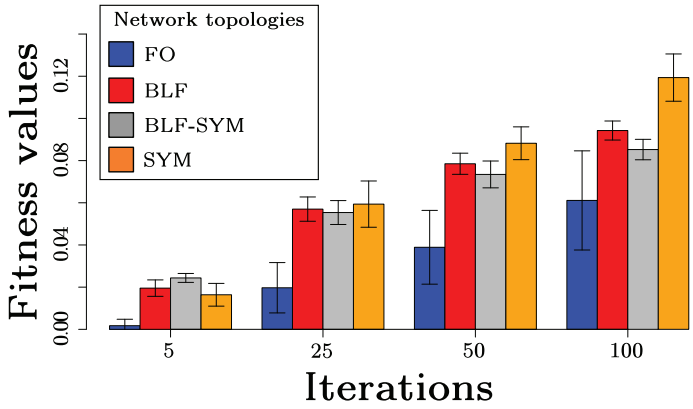


Figure 7.6 – The mean fitness values and standard deviation at iteration 5, 25, 50, and 100 for the different network topologies applied to the *quad5-sym* structure.

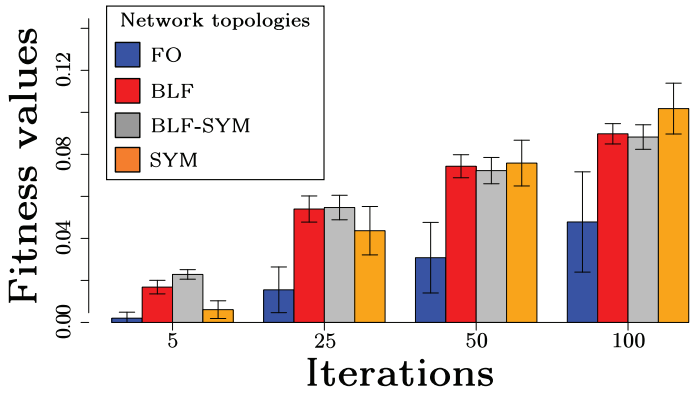


Figure 7.7 – The mean fitness values and standard deviation at iteration 5, 25, 50, and 100 for the different network topologies applied to the *quad5-unsym* structure.

Conclusion

HYBRID SRMRs have the ability to use embedded connectors in the environment to perform a kind a locomotion through reconfiguration, via connection and disconnection of their docking mechanisms. They can additionally freely locomote off-grid using various strategies. In this part we have illustrated both aspects and proposed efficient approaches to tackle the locomotion problem.

We have first described (Chapter 6) a novel planning method to perform on-grid locomotion through reconfiguration based on composed motor primitives and the well known $D^\star$ planning algorithm. Our technique was demonstrated using simulation of the RB platform but also through proof of concept hardware experiments using hardware modules. Our planner proved to be robust and we demonstrated the ability of a single RB module to reach any position on a 2D grid, fully autonomously.

In Chapter 7, we have tackled the issue of re-learning off-grid locomotion parameters under time-critical situations. A test situation that we are considering is a structure made of several robots and deployed in an unknown environment to perform a time-critical task. At a given point these robots, while locomoting using a known set of parameters, have to reconfigure into a not previously known shape to cope with an unexpected change in the environment or with some hardware failure. The resulting shape after reconfiguration was unknown beforehand, removing the opportunity of using gait look-up tables. We use Central Pattern Generators (CPGs) for the control of our modular structures and the population based optimization technique Particule Swarm Optimization (PSO) for finding our set of control parameters. Our strategy for this fast relearning of the locomotion parameters is to identify bio-inspired patterns such as body and limbs and articulation degree of freedom (such as spine, hip, knee, and ankles) to reduce the number of degree of freedom to consider in the locomotion parameter search. Additionally we proposed to use symmetries in the structure to further reduce the number of parameters to optimize, since two symmetric oscillators share their amplitude and phase. Our goal was to increase the speed of convergence towards a reasonable solution in terms of forward speed and internal collision. Using those reduction techniques we were able to generate reduced CPG networks for three different structures made of simulated RB modules, two quadruped like structures with five modules and one arbitrary random structure made of 10 modules. We proved that our method significantly outperformed the gait resulting from the fully open network in the first few iterations. Our method can be apply with

any arbitrary structures (except the fully linear or cyclic ones) and can be adapted to various optimization techniques and control methods.

In the long term goals of the RB project, the locomotion techniques we developed in this part will allow a quick, robust, and efficient deployment of the different robotic structure in the environment, as well as a strong capability to adapt to unknown and unexpected changes.

Interaction Part IV

Introduction

SELF-RECONFIGURABLE modular robots are composed of several independent units working together to achieve a particular task. They are different from more classical bio-inspired and anthropomorphic robots since they do not necessarily exhibit traits that would allow for an intuitive way of interaction (such has a head with cameras or hands with embedded tactile sensors). We can wonder whether it makes the design of control interfaces and interaction strategies easier or on the contrary more complex. The needs for a natural way of interacting with such robots is growing, especially if we envision to deploy them in everyday life environments, as it is the case in the Roombots project. When considering interaction inside homes or public spaces, we have to keep in mind that the proposed interaction solution should be non-intrusive but also easy to handle for non-experts or people with disabilities.

Our search for natural interaction strategies was guided by three main scenarios, tightly linked to the ultimate goal of the Roombots project of building assistive and adaptive pieces of furniture, but still applicable for various existing robotic platforms.

In a first scenario, called the *building scenario*, the user would like to design a particular shape to be constructed by the robot. For example, the user has in mind the building of a table with a unusually shaped table-top and he/she has to shape it and this shape has then to be transform into an interpretable file to serve as an input to one of the previously define reconfiguration frameworks (chapters 4 and 5).

For our second scenario, we imagine a user who needs to arrange a complete room with furniture made of modular robots. This scenario is called the *arrangement scenario*. The user can use the shapes that he/she design in the building scenario and should be able to place them in a room to have a better idea of its global arrangement. The user should also be able to freely move into the room to create the desired arrangement.

A last scenario, referred as the *direct control scenario*, would be when the user needs to indicate to a given robot or group of robots where to move or what action to perform. This control can be done for robots connected to a structured environment or for robotic structures performing off-grid locomotion.

In this part, we are going to present the advances we made to the state of art in the domain of interaction with a group of modular robots. We will examine in each chapter a particular

solution we propose for the above mentioned scenarios. For all of them the two main questions we are trying to answer are the following:

1. What would be an intuitive and natural way to interact with a group of modular robots in a given situation?
2. How can we capture and measure the efficiency and the effectiveness of an interaction strategy?

This part is organized as follows. Since our results and solutions for the first scenario are still at a early stage, we decided to move our study to the appendix C. The first chapter (chapter 8) of this part present the solution we derived to propose an interaction strategy allowing the user to freely arrange a complete room with furniture made of modular robots. We describe an evaluation of our solution based on a medium scale user study conducted on a representative sample of potential users. In a second chapter (chapter 9) we explore a device free control interface to solve the direct control scenario. We will finally conclude on the different strategies we have developed to tackle this challenging issue. To go beyond the interaction paradigms that we described in this part we present in appendix E a critical review of the existing approaches to create interfaces for mixed team of humans and robots, that can be used to pave the way for a collaborative framework involving heterogeneous teams of robots and human inside an everyday life environment.

8 Mobile interfaces

References and contributions
This section is based on the following publication:

S. Bonardi, J. Blatter, J. Fink, R. Möckel, P. Jermann, P. Dillenbourg, and A. J. Ijspeert. "Design and evaluation fo a graphical iPad interface application for arranging adaptive furniture". The 21st IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Paris, France, 2012.

And on the following master thesis project:

J. Blatter, "Mobile control interface for modular robots", Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2011. Available at: http://biorob.epfl.ch/page-75451.html

My contributions were:

- *proposed ideas for the mobile interface.*
- *provide supporting code for the user tracking.*
- *help to design, conduct, and analyse the results from the user study.*

The external contributions were:

- *help for the design and the conduction of the user study.*
- *help for the analysis of the results.*
- *the implementation of the interface.*

8.1 Introduction

Mobile devices such as smart-phones or tablets have recently become more and more popular. The field of robotics in everyday life could benefit from these technologies by using them as a platform for human-robot interaction (HRI). Modular robots are aiming at achieving robustness and versatility by using basic elements as building units for more complex structures able to autonomously adapt to changes in the environment [279]. The Roombots (RB) project aims at designing and controlling modular robots to be used as building blocks for adaptive furniture able to self-reconfigure, self-assemble, and self-locomote [239]. Through this, Roombots are meant to become robots for everyday life which defines a broad user group including people with no background in the field of robotics.

To make Roombots attractive and usable for a broader range of people, we designed a new interface for potential end-users. For the arrangement of furniture, a mobile device such as the iPad allows the user to intuitively interact with the environment while walking around the room to change perspective. This matches one of the main tasks that we envision users to perform: arranging furniture in a room according to the user needs. Thus, an interface should not only allow this arrangement but further let the user pre-visualize the result, with the different Roombots units performing the required moves in a simulated representation of the room. To improve this pre-visualization aspect, Augmented Reality (AR), in which virtual objects are superimposed to a real view of the environment, can be used. As a first step towards this goal, a preliminary version of such an interface was developed [22] and evaluated in a user study.

This chapter is organised as follows: we start by describing our hypotheses and the related works in the field. We then briefly describe our application before presenting the outline of the user study we conducted. Results are afterwards presented in both a quantitative and qualitative way. Finally we discuss our findings, conclude our main results and give an outlook to future work.

8.2 Theory and hypothesis

We conducted a user study to evaluate our approach of using the iPad as a platform for HRI. Additionally, we aimed at exploring how participants experienced the interaction and used the application. We further expected insights into the usability of this first version of the interface to refine it in the future.

We were particularly interested in finding out how far participants took advantage of the mobility of the device and walked around the room to change their perspective while arranging the robotic furniture through the interface. This topic addresses a main challenge in HRI research: how a single user can operate distributed mobile robots. We focus here on the human factor rather than on the technical aspects.

A second topic originates from the particularity that robots are physically embodied and share the same space with humans. This creates one of the main differences between interacting with a robotic technology and a traditional human-computer interface. We addressed this aspect by using an augmented-reality environment in which abstract virtual representations of adaptive furniture were used. Augmented reality enables a direct interaction within the real world and enhances it with computer-generated sensory inputs.

8.2.1 Augmented reality

Augmented Reality (AR) allows virtual objects to be combined with a real world representation. The three key characteristics of AR have been identified to be: (1) combining real and virtual images, (2) the virtual imagery is registered with the real world, and (3) real time interaction of both virtual and real objects [11]. During the last decades, research in AR focused on the development of techniques to provide such a user experience. Applications of AR can now be found in various domains such as medicine [30] or games [208]. With the sophistication of mobile devices the new field of mobile AR has emerged with its own challenges. The main limitations of current mobile devices for AR are the limited input/output options, the screen size and the graphical/computational power [20]. AR has often been opposed to pure virtual reality (VR) mainly in the domain of medical training and has shown to lead to a better perception of the task [30, 14], especially when coupled with haptic feedbacks. An AR based approach has recently been used as a new interaction technique to increase safety for industrial robots [191]: the user can pre-visualize the moves of a robotic arm and display information regarding the current state of the motors using a head-mounted display.

For our case study, we thus formalize the following hypothesis:

Hypothesis
The use of augmented reality eases the placement of virtual pieces of furniture using the iPad and improves the user experience. The precision of this arrangement and the completion time will also be positively affected.

8.2.2 Mobility

Contrary to classical wearable AR devices, such as head-mounted displays, mobile AR units are held in hand instead of being head mounted. This tends to increase peripheral view but greatly challenge the input/output design strategy [107, 108]. When using a mobile device, the fact that the user does not have both hands free induces more constraints on the interface design [134]. Nevertheless mobile AR has recently been used in the domain of room arrangement in the CMAR project [8]. In this project, multiple people can collaborate to arrange a room by dragging virtual representations of the furniture displayed on a mobile phone on a printed floor plan of the room. The collaborative aspect was emphasized in this study while participants' ability of moving was reduced since they had to remain seated in front of the table holding the AR marker. Some applications are also using the motion of the device itself as an input method

[20, 52]. It has been shown that even if the use of device motion leads to faster translational displacement of the virtual object, participants tend to be slower during the rotation phase of the object [20]. Additionally, the use of a small screen tends to negatively impact presence but the ability to move with these devices can compensate this effect [118]. To the best of our knowledge no user study has been conducted to evaluate the effect of the mobility of the participant in a room arrangement task.

So we postulate the following hypothesis:
Hypothesis
The ability to move in the room, as opposed to having a fixed standing point, improves the precision of the furniture arrangement and decreases the completion time.

8.3 Application and setup

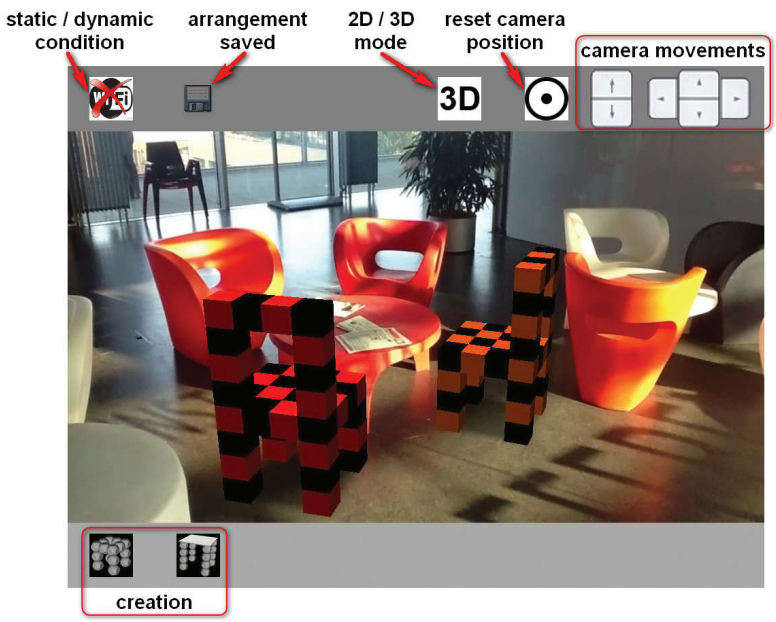
Our iPad application (see Fig. 8.1a for an illustration) allows placing two different types of furniture (tables and chairs) inside a room, to move and rotate them as well as to change their color (between five possible choices). The room configuration of our lab experiment is illustrated in Fig. 8.1b. An external device has been used to track the user inside the area delimited by a tape frame on the floor. The application is completely marker free. The required data regarding the experiment (number of actions, final placement of the furniture, ...) were recorded. More details about the technical aspects of the application can be found in [22].

8.3.1 Tracking system

In order to track participants during the experiment we used the Kinect sensor. We used the open source library Nestk [43] to estimate the position of the participants' body joints. This tracking library requires a calibration procedure to reduce ambiguities regarding the user orientation. The position of the iPad was estimated to be equal to the position of the user's neck, plus $30cm$ in the direction of the normal vector of the plane containing the torso and the two shoulders of the user. The computed position was sent in real time by WiFi to the iPad via a BSD socket with the UDP connectionless protocol. The overall precision obtained using this approximation is around $10cm$ on average (computed by comparing 10 measured values with computed ones). One of the main limitations of this method is the impossibility to detect a change in the iPad's position if the user does not move, for example by extending her arms. A second restriction was that the user had to stay in a predefined area determined by the field of view of the Kinect (about $3 \times 2.50m$). The orientation of the iPad was obtained directly using the iPad Inertial Measurement Unit (IMU).

8.3.2 Implementation

In order to render the different virtual elements of our application we used the open source graphics toolkit OpenSceneGraph (OSG) [42]. In the virtual condition, the virtual environment



(a) The application

(b) The experiment room

Figure 8.1 – A screenshot of the application is depicted in (a). The user can choose between five different colors (top row) for two types of furniture (chairs and tables, depicted in the bottom row). The room used during the experiment is shown in (b). The tape on the floor delimited the area where the participant could move in the dynamic modality. The stand where the iPad was fixed during the static condition can be seen in the back of the room, near the wall.

was fully constructed and rendered using OSG. In the augmented reality condition, virtual objects needed to be superimposed to the real time camera view. The camera view was

managed using the *Objective-C* language. This view was added as a sub-view of the window along with the OSG scene that was rendered in another sub-view of the window, placed on top.

8.3.3 Interactions

The application is composed of two main elements (depicted in Fig. 8.1a): (1) the view of the room in the background (either purely virtual or augmented) and (2) two Head Up Displays (HUD). The lower HUD allows the user to create and delete pieces of furniture using a single tap. The upper HUD can be used to change the furniture color and to check the state of the WiFi connection between the iPad and the Kinect software. The user can select and deselect pieces of furniture using a single tap and translate them using one finger once they are selected. The furniture can also be rotated using two fingers.

8.4 Method

In order to test our hypotheses we conducted a 2×2 laboratory experiment with 24 subjects. We defined the following manipulations:

1. The nature of the room representation:
    - **Virtual representation (V)**: the room is modeled by a pure 3D environment.
    - **Augmented Reality representation (A)**: the iPad camera is used to display the room and the virtual pieces of furniture are superimposed to this view (see Fig. 8.1a).
2. The ability to move:
    - **Static (S)**: the iPad is fixed on a stand facing the room. It can be rotated to have a different angle of view of the room but it cannot be translated.
    - **Dynamic (D)**: the participant can freely move inside the delimited area of the room (Fig. 8.1b) holding the iPad in her hands.

The experiment was a within-subject design, where each participant performed two of the four conditions (SV, SA, DV, and DA). Subjects performed the two respective conditions they were assigned to in a counterbalanced way to handle order effects and keeping one of the two modalities constant (row or columns in Table 8.1). The control group was composed of users in the "static" condition using a virtual representation of the room (modalities S and V).

8.4.1 Participants

We recruited 24 subjects among students of different levels (bachelor, master, phd) and staff members (post docs, assistants, administrative staff). People were invited to participate in a

Table 8.1 – The four different groups of participants.

	Room representation	
	Virtual (V)	Augmented (A)
Ability to move		
Static (S)	SV	SA
Dynamic (D)	DV	DA

one-hour session to evaluate a new software application. The mean age of participants was 27.3 and 8 of the 24 participants were women. We collected no data on participants' cultural background or other demographic factors.

8.4.2 Task and procedure

The main goal of the study was to gain knowledge about the effective use of the user's ability to move (opposition between static and dynamic conditions) and about the suitable accuracy of the environment representation. We designed the following task to test these aspects: the user was asked to create a circle of 10 chairs, two of each color, and place a table in the middle (see Fig. 8.3). The room was completely empty and no instructions about the size of the circle or the color order were given. In this task, we tested the space perception and the accuracy of the relative positioning of virtual object.

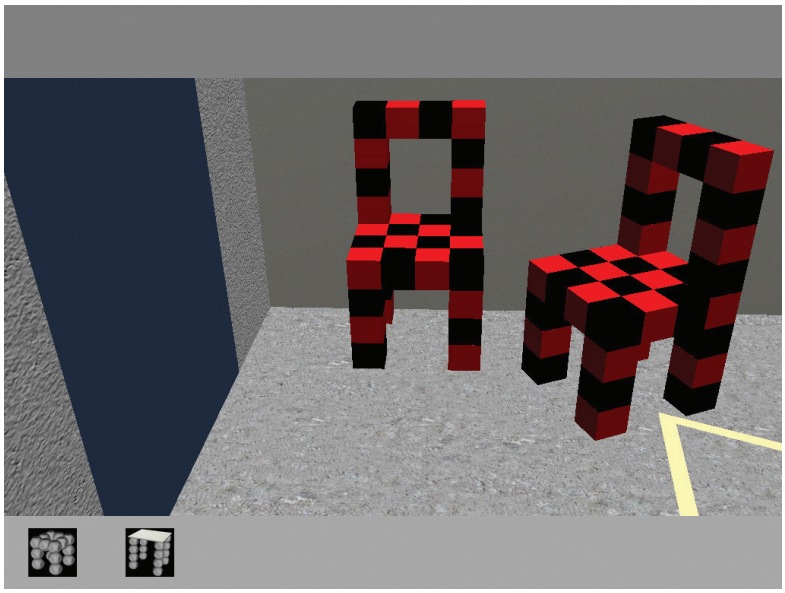
8.4.3 Protocol

Participants were brought to the experiment room, given a short introduction to the project and asked to sign a consent form, to agree being videotaped during the interaction and recorded while the post-interview took place. Participants were shortly introduced to how the application was working and given about one minute to become familiar with it. They were explained the basic operations of the software, e.g. how to create, move, rotate, select or delete a piece of furniture. This try-out always took place using a virtual representation of the room (Virtual) and while the iPad was fixed on a stand (Static). Each subject performed the task in each of their two respective conditions. No special instructions were given to participants regarding precision or completion time. After this, participants were asked to complete a questionnaire which was integrated in a qualitative semi-structured interview.

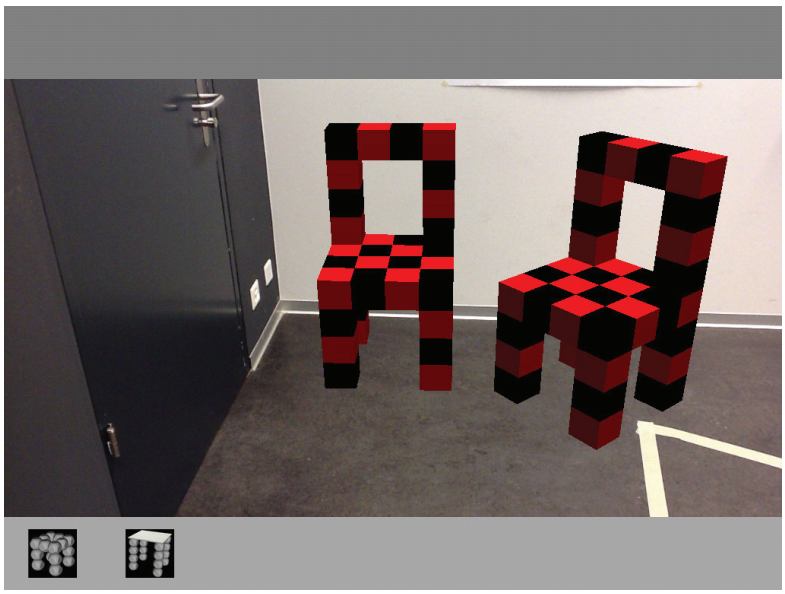
8.4.4 Measures

Dependent variables We identified five main dependent variables:

- The **completion time**, measured from the first interaction with the interface to the last



(a) Virtual view



(b) Augmented view

Figure 8.2 – Virtual (a) and augmented (b) representations of the room with two chairs.

one.

- The **number of errors participants made in the final arrangement** (e.g. additional piece of furniture or wrongly colored chair).
- The **number of actions** and their **types** (e.g. rotations, translations, selections,...)
- The **position of the user** during the experiment (in the dynamic condition).

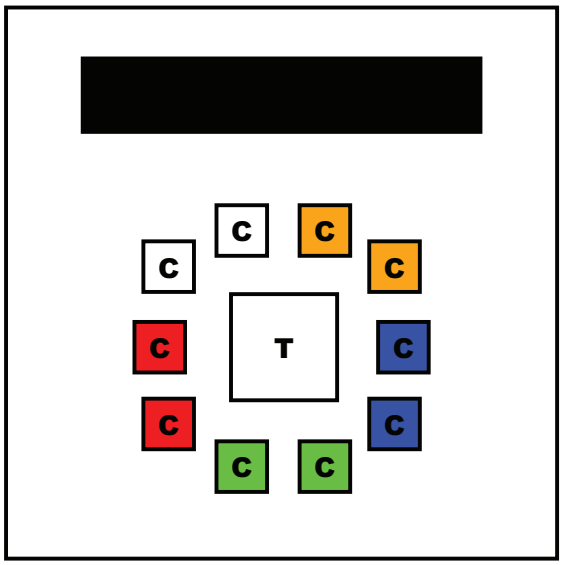


Figure 8.3 – This picture depicts the circle of chairs (**C**) with the table (**T**) in the middle used in the task. Static virtual objects are represented by black rectangles.

- The **precision** of the final arrangement. We estimated the precision of the arrangement considering both the overall placement of the chairs in a circle (as a global error measurement, denoted by p_g) and the regularity of the positioning of the furniture with respect to each other (as a local error measurement, denoted by p_l). To evaluate the deviation from a perfect global arrangement (circle) we computed the ellipse fit based on a mean squared error algorithm taking the position of the chairs center as data points. The ratio between the long and the short axis of this ellipse captures the deformation of the global placement. The local precision error p_l takes into account the standard deviation of the distance between the chairs and their centroids ($\sigma(d_{cc})$) as well as the distance between the centroid of the group of chairs and the table center (d_{ct}):

$$p_l = \sigma(d_{cc}) + d_{ct} \tag{8.1}$$

Questionnaire After performing the task, participants were asked to fill a questionnaire during the interview part. The questionnaire assessed using Likert scales people's previous expertise with tablets and smart-phones as well as with 3D object manipulation. The goal was to check whether participant's expertise impacted how the task was solved and how the device was interacted with. The questionnaire further addressed aspects such as ease of use, usefulness and learnability of the application. One questionnaire per group was created since some questions were specific to one condition (for example the ease of manipulation of the stand). Common questions included a relative evaluation of the task (difficulty, clarity and entertainment aspect) and a global appreciation of the experience.

8.5 Results

The mean and standard deviation related to the local precision error, the global precision error, and the completion time are summarized in Table 8.2. We performed a two factors ANOVA with the room representation (virtual or augmented) as first factor and the mobility (static or dynamic) as a second factor. The data of the control group (SV) are compared against those obtained in the other conditions. Results are described in more details below.

Table 8.2 – The mean and standard deviation of the local precision error, the global precision error, and the completion time for the four modalities.

	Modalities			
	V		**A**	
	S	**D**	**S**	**D**
Local precision p_l (m)				
M	.73	.55	.81	.60
SD	.26	.29	.39	.13
Global precision p_g				
M	1.23	1.13	1.19	1.12
SD	.19	.069	.10	.081
Completion time (s)				
M	194.13	225.64	185.76	229.38
SD	102.31	151.96	67.61	69.97

8.5.1 Effects of the room representation

In our first hypothesis, we postulated that using augmented reality would improve the user experience, increase the precision of the arrangement, and decrease the completion time. Our data does not support this hypothesis. There is no significant difference in terms of precision for the room arrangement ($F(1,46) = .65$, $p = .42$ and $F(1,46) = .57$, $p = .45$ for local and global precision respectively) . With the same factors, no significant difference was found regarding the completion time ($F(1,46) = .006$, $p = .94$).

8.5.2 Effects of the ability to move

In hypothesis 2, we argued that the ability to move would positively impact both the completion time and the precision of the room arrangement. We found that there was a main effect of the mobility on both the local precision ($F(1,46) = 5.86$, $p = .0196$) and the global precision ($F(1,46) = 5.94$, $p = .0187$). No significant difference has been found between these two modalities in terms of completion time ($F(1,46) = 1.57$, $p = .22$).

8.5.3 Interaction effects

We did not observe interaction effects between the two factors for the two types of precision as well as for the completion time: as illustrated in Fig. 8.4 the augmented and virtual conditions do not impact the trend observed for both the local and global precision errors between the dynamic and the static condition.

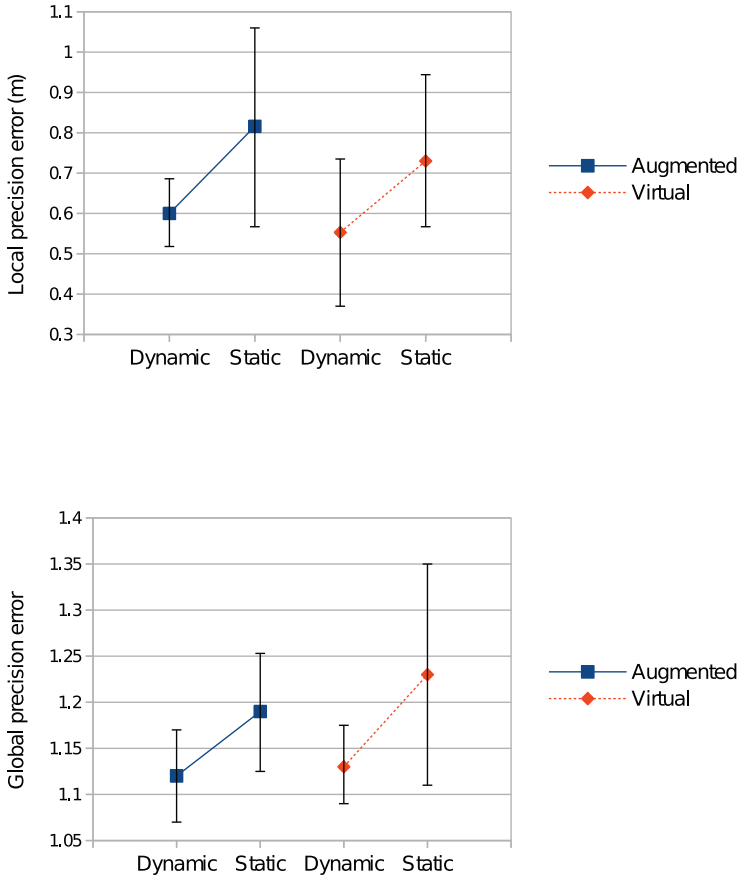


Figure 8.4 – The upper figure depicts the means and confidence intervals for the mobility and the representation factors for the local precision error p_l. The same indicators for the global precision p_g error are shown in the lower graph.

8.5.4 Qualitative study

For all the illustrations of this section (Fig. 8.5) the legend is as follows[1]: for the static condition, the pink star represents the position of the stand on which the iPad was placed; the circles and the square represent respectively the chairs and the table. The color inside these shapes

[1] In black and white printed versions the color gradients mentioned go from dark to light grey.

indicates the order in which the furniture were placed, starting from dark red to white. The position of the participant at every 0.1s is represented by stars colored continuously from red to yellow (as represented by the color scale in Fig. 8.5) following the time evolution.

Subjects' trajectory types for the dynamic modality We represented the trajectory of the participants during the task based on their tracked position. A classification of the displacement types of the participants can be made, based on four main criteria:

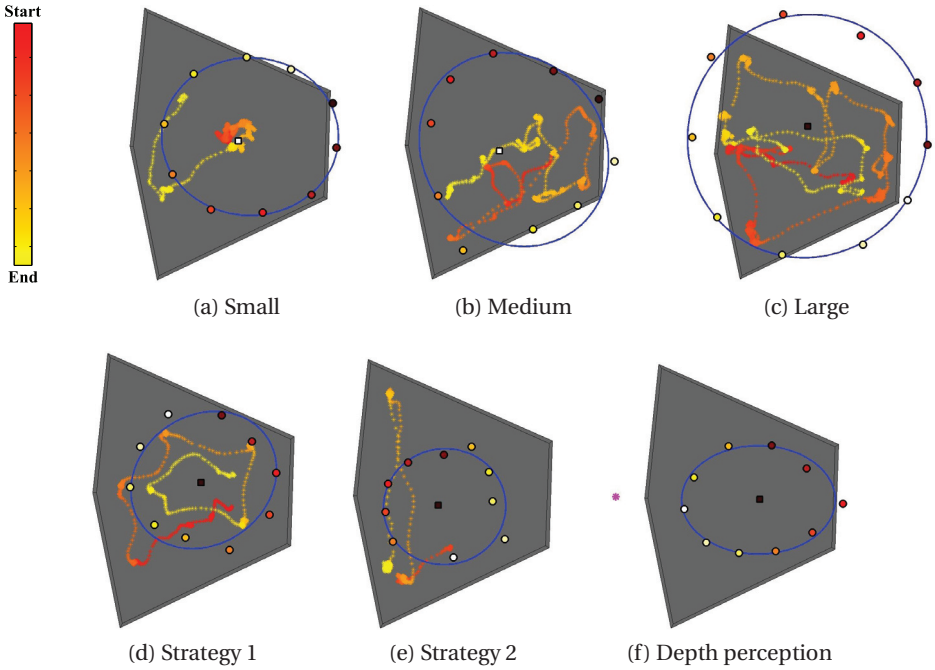
1. the overall surface covered.
2. the number of control points and interaction points. A *control point* is a position where the user stands not for interacting with the furniture but to check the current status of the arrangement. On the opposite, the user stops on an *interacting point* to place, translate or rotate a piece of furniture. These points have been determined by cross-checking the data from the log files and the videos of the different experiments. We can thus distinguish between displacement phase and action phase.
3. the ratio of inner and outer points. *Inner point* are located inside the ellipse used to define p_g whereas the *outer points* are outside the ellipse.
4. the speed of the displacements.

Based on these criteria we identified trends in the trajectory of the participants that we classified into three main categories: the *Small* category (see Fig. 8.5a), the *Medium* category (see Fig. 8.5b), and the *Large* category (see Fig. 8.5c). In the *Small* category, the user mainly remains at the same position and only uses one external point at the end of the experiment for checking the arrangement. In the *Medium* category, several interaction and observation points can be observed, mainly inside the ellipse but also outside. The available moving area is not fully used. Finally, in the *Large* category, several interaction and observation points can be seen as well as an exhaustive use of the space. We observed in addition that the participants who performed the two dynamic conditions (DV and DA) remain consistent in their strategy.

For the two modalities tested in the dynamic condition the repartition between these three categories are summarized in Table 8.3.

We observed that most participants (10/12 and 11/12 in the DV and DA group, respectively) effectively used the available space. Nevertheless no significant differences were found in terms of completion time, precision, or number of actions between those groups.

Perception of depth, distance and alignment As previously mentioned, we observed a significantly higher deformation of the ellipse (p_g) in the static experiment than in the dynamic one (see Fig. 8.5f for an illustration): the user tends to place the table too far away and has a



(a) Small (b) Medium (c) Large

(d) Strategy 1 (e) Strategy 2 (f) Depth perception

Figure 8.5 – In the *Small* category (a), the user mainly remains in the middle of the circle of chairs at the same position and only uses one external point at the end of the experiment for controlling the arrangement. In the *Medium* category (b), several interaction and observation points can be observed, mainly inside the ellipse but also outside. The grey surface is not fully used. Finally, in the *Large* category (c), several interaction and observation points can be seen as well as an exhaustive use of the space. In (d), a common strategy for furniture placing is depicted: the participant face the area where s/he would like to add a piece of furniture and do a few steps backward to have a better view. (e) shows the "outside" strategy: the participant stays outside the circle of furniture and only move at the end to control the placement. (f) illustrates the depth perception issue in the static condition (the pink star represents the position of the stand on which the iPad was attached).

Table 8.3 – The repartition of the participants based on their displacements during the task.

	Categories		
	Small	Medium	Large
Groups			
DV	2	1	9
DA	1	5	6

distorted perception of the depth inside the room. The fact that both AR and VR conditions were equally affected is consistent with previous studies mentioning a similar bias in the perception of distance between these two representations [251]. Some of the classical issues of AR [79], such as the distance estimation and the alignment difficulty, have also been reported by the participants during the interview session.

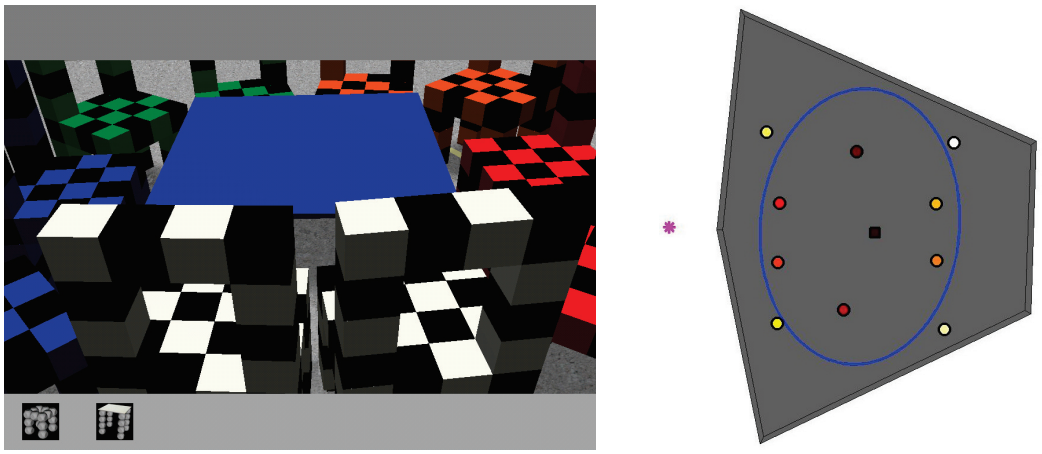
Furniture placement We observed that participants tend to go away from the position they want to place the furniture in while facing it (see Fig. 8.5d). Another commonly observed strategy was to adopt an external point of view during the task: the participant would stay outside the circle of chairs and only change position to have a different perspective or to control their placement (see Fig. 8.5e).

Regarding the order of placement for the furniture, there were no significant differences between subjects starting with the table and subjects starting with the chairs, but most of the participants (20/24 and 19/24 in the static and dynamic condition, respectively) placed the table first. We can infer that they needed a fixed point as a guiding cue to place the other pieces of furniture.

Additionally, many participants mentioned during the interview that they have often been annoyed by already placed furniture in the static case because those were blocking their view of the room (see Fig. 8.6 for an illustration).

8.6 Discussion

The analysis of our questionnaire revealed that the participants preferred the dynamic condition over the static one (10 among 12). This preference is supported by the significantly better results they obtained in terms of precision but no correlation has been found with the completion time or the number of actions. Moreover, 9 participants among 12 declared having preferred the augmented representation of the room but no significant differences have been observed between virtual and augmented representation. It can be explained by the fact that the task involved a very simple environment without any dynamic elements. Several participants mentioned during the interviews that they would tend to favor the augmented



(a) The iPad screenshot (b) The representation

Figure 8.6 – An example of blocked view: the participant placed first chairs in front the stand (a) and then experienced difficulties to complete the circle of chair mainly because of the obstructed view (b).

reality environment because it was more "realistic", especially regarding the lighting condition. Unfortunately, the current state of the application did not allow the use of more complex environments.

We have noticed that the strategy to place furniture greatly varies between the participants. Nevertheless a commonly observed behavior was that participants tend to stay at a given position while interacting with a given piece of furniture. This tendency, coupled with the fact of taking one or two steps back before placing the furniture can be explained by the restricted angle of view provided by the iPad camera. In the questionnaire, 16/24 participants declared that the field of view of the camera was not big enough to comfortably complete the task.

We observed that the previous quantitative results remained the same when considering 2 groups of subjects classified using a median split based on their expertise. These results are further supported by the replies in the questionnaire: the intuitiveness and the ease of learning of the application have a mean rating of 3.68/4 while its ease of use has been evaluated to 3.59/4 on average. We conclude that the application was sufficiently intuitive to balance the difference in expertise between the participants. In addition, the participants were asked whether they found the software frustrating or confusing and more than 3/4 strongly rejected this statement (meaning it was rated 0/5 in the questionnaire). We observed the same rating regarding the responsiveness of the application. The overall number of errors is also low (3 errors in the final arrangement among all the different experiments performed).

Many participants mentioned during the interview session that a 2D top view of the room would have been helpful to check the arrangement. This feature is available in many architecture software (see for example *Sweet Home 3D* [252]) and is often coupled with a 3D view

for rendering only. The main reason why we discarded this option is that our application is meant to be used in any existing room (with already placed furniture inside), meaning that the environment is unknown before the start of the furniture placement activity. Indeed the relevant features of the room will be tracked in order to reconstruct the corresponding model of the room used in the application. On the opposite, the 2D view is well suited for not yet existing buildings or for completely known environments.

8.7 Conclusion and future work

Revisiting our initial hypotheses, our study has shown that participants took advantage of being able to move inside the environment and performed significantly better in term of precision during the task. We can conclude that mobile devices are more suited for arrangement tasks and preferred to fixed devices such as desktop PCs. Nevertheless the data that we collected did not lend any support to our first hypothesis regarding the level of details in the room representation: no differences were observed between a pure virtual representation of the room and the use of an augmented reality environment. This last statement might be due to the simplicity of the task we considered for this study, mainly regarding the integration of dynamical objects inside the scene. We nonetheless observed that allowing the user to move inside the environment while using augmented reality to integrate virtual elements enhanced the user experience and eased the interaction between users and virtual artefacts.

Although our results were somewhat inconclusive regarding the effect of augmented reality, we believe that they were greatly influenced by the technical limitations of the current version of the interface. We are thus planning to improve our software to offer a more natural inclusion of the pieces of furniture in the environment (for example by adding shadows). Further study will then be needed in a more dynamical and unknown environment to ascertain the preference of the users regarding the type of representation.

In order to remove the need for an external device to track the user, we investigated a new way of detecting the user position using only points of interest in the image and a SLAM like algorithm called PTAM [137, 138]. Our preliminary results shows that we can efficiently draw a stable augmented scene inside the environment without relying on external sensors. More details are given in appendix D.

9 Device-free interface

References and contributions
This chapter is based on the following publication:

A. Özgür, S. Bonardi, M. Vespignani, R. Möckel, and A. J. Ijspeert. Natural User Interface for Roombots. The 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Edinburgh, Scotland, UK, 2014.

My contributions was:

- *general guidance to develop the different parts of the framework.*

The external contributions were:

- *the implementation of the framework.*
- *the development of the electronic parts.*
- *critical thinking on the more suitable interaction strategies to implement.*

The Roombots modules have been designed with the objective of creating adaptive furniture for home environments. Modules would perform both on-grid reconfiguration but also off-grid locomotion to match the user's needs. Up to now we have mainly considered three different ways of interacting with RB: (i) for lab's experiments, we are sending commands using a custom-made ASCII protocol; (ii) to build or to control a structure composed of several modules, we developed different GUI running on a PC (see appendix C for a presentation of the building GUI); (iii) we introduced a new tablet-based interface allowing non-expert users to quickly and efficiently arrange virtual pieces of furniture made of modular robots in an augmented reality rendering of a room (see chapter 8). All these methods come with limitations when considering the envisioned use cases of the RB: the first one is restricted to expert user, while the second one requires the user to stay in front of the computer, preventing

her/him to freely move in the environment while arranging it; as for the last one, the user needs to carry an external device with him to interact with the robots. We would like to introduce a more intuitive way of interacting with the group of modular robots using physical gestures for selection and control of the robots, but also by relying on visual sensory feedbacks to provide information to the user on the state of the system. The proposed interface uses Kinect depth sensor to track the user and detect where she/he is pointing at, removing the need to carry an extra device. The visual feedbacks are provided via LED rings installed on the two diagonal degrees of freedom of the RB modules (see Fig. 9.1b for an illustration) as well as by LED plates mounted directly on the grid setup. The test setup that we consider is a grid with a vertical plane where two Roombots modules are connected (see Fig. 9.1a for an illustration).

We first present the structure of the tracking framework (section 9.1) and then we give an overview of the interaction strategies we developed (section 9.2).

9.1 Tracking framework

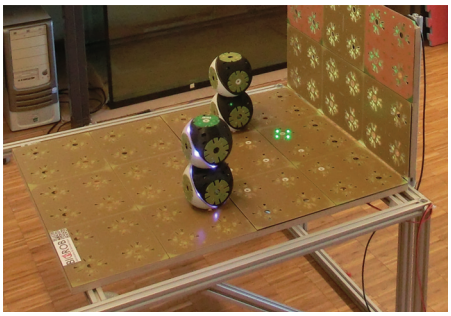
The overall architecture of the tracking framework can be seen in Fig. 9.2. There are two main tracking routines that have to be developed to ensure a proper user experience. First of all we need to detect the position of the grid in the environment using the depth sensors of the Kinects. Afterwards we need to robustly track the user to be able to detect where she/he is pointing at.

In order to track the environment, we have to decide on the number of Kinects that we are going to use as well as on their placement (position and orientation) in the room. By evaluating the interference between the different sensing units and considering the resulting space covered by the tracking, we found a trade-off with two Kinects, one near vertical pointing at the grid setup and one almost horizontal and pointing at the user.

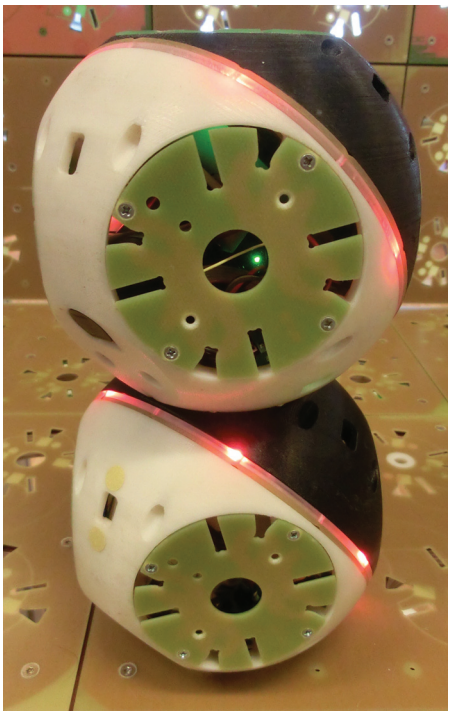
The two kinects sensors need to be "synchronized" to construct a unified 3D coordinate system. This step is done using an extrinsic calibration routine relying on two main steps: (i) A coarse calibration in which the user has to manually[1] align the cloud of point coming from the two kinects using a 3D visualization; (ii) A precise calibration performed using a variant of the Iterative Closest Point algorithm [19].

After having a coherent point cloud, we need to detect the user. To do so we use the skeleton tracker provided by the NiTE middleware [213] of the OpenNI framework [194]. To improve the quality of the tracking we apply median filtering on the input of the skeleton tracker and a weighted average running average filter on the determined joint positions. The pointing gestures are detected by considering the head to hand vector as the direction of interest.

[1] It should be kept in mind that this step only has to be performed once by the user, and, as such, does not imply an overload for her/him.



(a) The grid setup



(b)

Figure 9.1 – (a) The setup being tracked is composed of a grid with a vertical panel and two Roombots modules connected to the horizontal plane. (b) A single RB module equipped with two LED rings, lighted in red (adapted from [197]).

9.2 Interaction strategies

In order to provide visual feedbacks to the user and improve the usability of the interface, we designed two LED-based systems (with four colors LEDs) equipping both the RB modules, as rings on the two diagonal degrees of freedom, and the grid, as additional tiles that superimposed to the existing connectors. The LED rings can exhibit three main behaviors: (i) Constant lighting with a single color; (ii) Breathing effect where the intensity of the LEDs increases and decreases periodically; (iii) Turning effects, in which the LEDs are lit one after the other to simulate the movement of the degree of freedom of the RB module. The user can select or

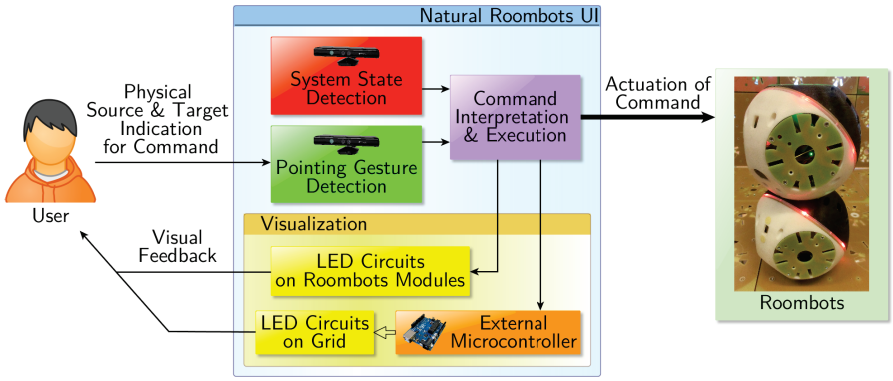


Figure 9.2 – Overview of the natural interface. Two main units comprise the interface: a tracking unit (green, red, and purple squares) and a visual feedback unit (visualisation unit in light orange). The tracking unit relies on two Kinect depth sensors to detect the state of the grid (in red) and to track the user (in green). The output of those two blocks are combined to give a coherent interpretation of the state of the system and to dispatch the appropriate command (in purple). Those instructions are then sent to the visualisation unit which will directly command the robotic platform and the grid board to provide visual feedback to the user (adapted from [197]).

deselect a tile by pointing at it for more than two seconds. By this mean the user provides to the system the starting grid position and the end grid position that can be used as an input to one of our locomotion through reconfiguration framework (see for example chapter 6). An complete example of interaction with the framework can be seen at [196].

9.3 Conclusion

In this chapter, we presented a novel and natural way of interacting with a group of modular robots. We tracked both the environment and the user using two depth sensors provided by two Kinects camera and performed skeleton detection to determine the pointing gestures of the user. To enhance the user experience and to enrich the interaction modalities of our framework, we designed and implemented visual feedback LED rings and tiles to equip our RB modules as well as the grid. A first test has been conducted inside our lab and proved to be convincing in terms of robustness of the tracking and ease of use of the application.

Conclusion

THROUGHOUT this part we have examined various interaction strategies to solve the scenarios that we considered as being archetypal to the problem encountered when using modular robots.

For our building scenario, we proposed to let the user build a 3D shape in a rendering interface using cubes aligned with a 3D grid. The created structure was then converted, using a perfect matching algorithm from graph theory, into a structure made of RB modules. This converted shape can be used as an input for one of the reconfiguration frameworks we described in chapter 4 and 5. Although our proposed solution is still at a early stage and we have not yet fully evaluated it, we believe it will prove to be a valid approach to solve the problem at hand.

To tackle the arrangement scenario we developed and evaluated an interface for tablet, in which the user can place virtual pieces of furniture into an augmented reality view of the actual room. We have shown trough a user study involving 20 participants that the ability to move inside the room was a significant improvement in comparison with being fixed in a given position, as it is the case for PC based interface. The ease of use and the intuitiveness of our solution positively impacted the user experience. To alleviate the burden of having to rely on an external device to track the user while performing the arrangement task, we briefly presented a recent advance we made using a SLAM like algorithm called PTAM [137, 138], that allows us to draw a stable scene in an augmented reality setting with only one camera video stream as input.

Finally, we proposed to solve the direct control scenario using a Kinect based interface tracking the user pointing gesture. This physical embodiment of the interaction with both the robots and the humans interacting directly in the same space without external devices, was coupled with a LED-based visual feedbacks both on the robots and on the grid setup. Our preliminary tests show the robustness of the tracking and the ease of use of the interface.

All our proposed solutions are complementary in the scope of modular robots integrated in everyday life environments. Furthermore, we have seen how they can be used to bridge the approaches we proposed for locomotion and reconfiguration of SRMR by providing to the user an intuitive and natural mean of exploiting the true potential of SRMR without having to consider their related inherent complexity.

General Conclusion

SELF-RECONFIGURABLE modular robots have been created to bring flexibility and adaptability to the world of robotics by dramatically changing the paradigms in place so far for solving a task or react to an unknown environment. But they are still mostly confined to lab environments, where perfectly controlled conditions bias the control techniques developed. Furthermore they still suffer from hardware limitations, such as bending effects and connection misalignment. Simulations and abstract models are often developed to study self-reconfiguration and locomotion problems, but few have really taken into account the inherent imperfections of the hardware platforms. Similarly, self-reconfiguration and locomotion are most of the time considered separately when it comes to create interfaces to control a set of modular robots. With the democratization of robots into our societies and their ever growing use in everyday life environments for services and assistance, a new opportunity to exploit the advantages of SRMR arose. In this dissertation we shed light on the necessity of offering to non-expert users a complete, robust, and natural control over any sets of modular robots, abstracting away the complexity linked to shape changing and gait learning.

Our contribution is threefold:

1. We proposed novel and generic self-reconfiguration techniques with built-in hardware constraints consideration, such as torque limitation and an exploratory connection misalignment compensation technique in hardware.

   We described a self-reconfiguration technique based on a gradient based approach inspired by the work of K. Stoy [243]. Instead of scaffolding techniques to ensure a built-in convergence, we introduced different low level interaction strategies between the actives units to avoid deadlock situations. We have shown that our strategies tend to reduce the number of deadlocks, especially for more complex structures.

   We modified an approach by R. Fitch et al [90] originally based on a full Markov Decision Process formulation of the SR problem. We derived a reward based reconfiguration framework that simplifies the overall approach and keeps the built-in convergence aspect of it but at the cost of a computationally demanding precomputing step.

   We extended the classical self-reconfiguration towards what we refer as *augmented self-reconfiguration* to include fully passive elements (which can potentially be damaged modules) into the SR process. We proposed a novel manipulation method using SRMR

(a very recent work by Cohen et al. [64] introduced a planner for manipulation tasks using a set of robotic arms) based on an efficient, yet simple, hierarchical centralized approach to perform manipulation of fully passive pieces in arbitrary 3D environments. We described preliminary results on how the augmented SR problem can be reduced to a multi-robot path planning problem. By integrating these external passive elements in the final structure of the SR process, we have opened the way for creating a significantly larger set of shapes, increasing at the same time the range of tasks that SRMR can carry out.

We have demonstrated the influence of the torque limitation on the number of moves (defined as the displacement between two connection and disconnection phases) needed to reach a position on a regular grid, showing that a lower torque value induces a larger number of moves of smaller angular displacement.

2. We developed innovative control methods to provide efficient locomotion strategies, both using connectors embedded in the structured environment (on-grid locomotion through self-reconfiguration) but also completely off-grid.

   We described a simple, yet robust, planner based on composed motor primitives to perform locomotion through reconfiguration. In comparison with existing approaches, such as [88], we fully tested our approach on our self-reconfigurable modular robot Roombots through various hardware experiments including 2D grid locomotion through reconfiguration and concave edge overcoming (followed by wall-climbing).

   We have additionally shown how the detection of bio-inspired patterns and the use of symmetries in a given structure could allow us to generate reduced Central Pattern Generator control networks, that would lead to a faster convergence towards an acceptable gait. Once again we were concerned by how cope with potential hardware failure or unexpected changes in the environment during a time critical task, potentially falsely detected by imperfect sensors. Our approach differs from those using gait-tables or classical CPG networks control with predefined structures (see for example [148]) because we are able to deal with topologies unknown before the reconfiguration process and to quickly provide new control parameters, thus widening the range of possible structures that can be considered and allowing for a fine grain adaptation to the task uniquely constrained by the objective to be achieved and the reconfiguration capabilities of the active units, and no more by the available control scheme.

3. We explored various interaction strategies with modular robots to capture their specificities and allow non-expert users to exploit their full potential. We first designed building interfaces providing a way to the users to create new shapes using cubes in a 3D computer based environment. We provided a conversion algorithm to directly convert any built shape into structures made of modular robots that can be used as an input for our SR or locomotion methods. Considering the next step of home arrangement, we designed and evaluated a tablet interface that lets the user place virtual pieces of furniture into an augmented reality rendering of the room. We recently alleviated the

need for an external tracking device using a SLAM like algorithm, called PTAM [137, 138], which should further enhance the user experience. We pushed forwards the concept of device free interfaces by developing a gesture based interface, with a tracking relying on the Kinect depth sensors, as well as an LED-based visual system allowing the user to direct the modular robots on the grid and to receive visual cues of the system state. This physical embodiment of the interface should improve the ease of use and the acceptability of our techniques.

All our approaches have been guided by our vision of modular robots being used as assistive and adaptive pieces of furniture. We have derived a set of strategies allowing any non expert user to use a set of modular robots to its full potential via intuitive and natural interfaces, and relying on robust and efficient locomotion and self-reconfiguration techniques.

Future challenges

Although our global approach advances the state of the art in reconfigurable modular robotics, many challenges still remains to be tackled.

One of the main advantages of SRMR lies in their capabilities to morph onto different shapes to adapt their topology to the task to be performed or to the user needs. But up to now, we have mainly considered a triggering of the SR process by the user himself. The next step to further improve our system and its integration into home environments would be to rely on sensors inputs, both to trigger the SR process but also to guide it towards the most optimal shape to solve the considered task. These sensors inputs could be produced using behavioral analysis of the users in their environment.

Modular robots are naturally well-suited for decentralized approaches since they provide a set of computing units that can be linked together using message passing techniques. Our methods are all centralized to keep them as simple as possible, but this comes at the cost of robustness against electronic failure. Nevertheless, these techniques can be extended and modified to accustom with partially or fully distributed implementations. For the scenarios we had in mind, namely the deployment of groups of robots into home environments, we considered that the robustness aspect related to computation did not represent a major concern.

We have proposed a reduction technique for the augmented SR process that we are planning to test extensively with the integration of our preliminary results on a more realistic connection procedure. This integration should mainly impact the time needed to build a given structure, since the visual analysis of the images coming from the camera device requires a low motion speed to avoid motion blur effects. Additionally the approach to the goal connector needs to be constrained to favor a straight line motion.

Finally, we have yet to test our global approach with real end users and various modular

robotic platforms to evaluate and, most probably iterate over, our interaction strategies and the underlying methods for locomotion and self-reconfiguration.

A Kinematic structure: case study of the Roombots module

A.1 Screw theory

The POE formulation of a kinematic chain with n joints is the following [185]:

$$\prod_{k=1}^{k=n} e^{\xi_k \theta_k} g_o = g_f \tag{A.1}$$

where:
g_o and g_f are the initial and final pose (position and orientation) of the end effector.
ξ_i is the twist corresponding to joint i. If joint i is revolute then $\xi_i = (-w_i \times q_i, w_i)$ with w_i being a unit vector in the direction of the joint axis and q_i an arbitrary point on the joint axis. If joint i is prismatic then $\xi_i = (v_i, 0)$ with v_i being a unit vector in the direction of the translation.

In order to simplify equation A.1, we use two main properties of the twist [290]:

1. Position preservation: if a point p is on the axis of a revolute twist, then $e^{\xi_k \theta_k} p = p$.
2. Distance preservation: for any point p and q, we have $\| e^{\xi_k \theta_k} (p - q) \| = \| p - q \|$

The POE formulation can be reduced into three main subproblems with known solutions [198, 131, 185] (note that additional subproblems can also be derived, such as the one presented in [142]):

1. Subproblem 1: Rotation about a single axis
   Let ξ_k be a zero pitch twist and q and p two arbitrary points in $\mathbb{R}^3$. The subproblem 1 can be written

$$e^{\xi_k \theta_k} p = q \tag{A.2}$$

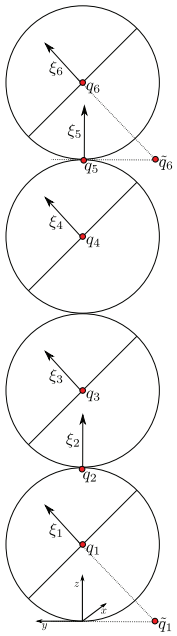


Figure A.1 – The naming conventions used to apply the POE formula to a RB metamodule. $q_{i\in[1..6]}$ corresponds to the rotation point of the joint i. $\xi_{i\in[1..6]}$ is the twist for joint i. $\tilde{q}_1$ and $\tilde{q}_6$ are the projections of q_1 and q_6 on the plane xOy and the plane define by ξ_5 and q_5, respectively.

2. Subproblem 2: Rotation about two subsequent axis
   Let ξ_1 and ξ_2 be two zero pitch, unit magnitude twists with intersecting axis and q and p two arbitrary points in $\mathbb{R}^3$. The subproblem 2 can be written

$$e^{\xi_1\theta_1}e^{\xi_2\theta_2}p = q \tag{A.3}$$

3. Subproblem 3: Rotation to a given distance
   Let ξ_k be a zero pitch twist, q and p two arbitrary points in $\mathbb{R}^3$, and $\delta \in \mathbb{R}^3$ with $\delta > 0$. The subproblem 3 can be written

$$\|q - e^{\xi_k\theta_k}p\| = \delta \tag{A.4}$$

The solutions to these subproblems are given in [185], at page 100, 102, and 103 respectively, with the corresponding restriction on their validity.

A.2 Roombots metamodule IK

We define $G_i = g_f g_o^{-1} q_i$ and $G = g_f g_0^{-1}$. Solving the IK problem for the end connector $C3X$ of the second module of a RB metamodule consists in solving the following equality:

$$\prod_{k=1}^{k=6} e^{\xi_k \theta_k} g_o = g_f \Leftrightarrow \prod_{k=1}^{k=6} e^{\xi_k \theta_k} = G \tag{A.5}$$

We first consider ξ_6 fixed. Equation A.5 is reduced to:

$$\prod_{k=1}^{k=5} e^{\xi_k \theta_k} = G \tag{A.6}$$

We start by finding angle θ_3:

We apply q_4 on both sides of equation A.6 and we obtain:

$$\prod_{k=1}^{k=5} e^{\xi_k \theta_k} q_4 = G q_4 \Leftrightarrow \prod_{k=1}^{k=3} e^{\xi_k \theta_k} q_4 = G_4 \tag{A.7}$$

We then subtract q_1 and apply the norm to both sides:

$$\prod_{k=1}^{k=3} e^{\xi_k \theta_k} q_4 = G_4 \Leftrightarrow \prod_{k=1}^{k=3} e^{\xi_k \theta_k} q_4 - q_1 = G_4 - q_1 \tag{A.8}$$

$$\prod_{k=1}^{k=3} e^{\xi_k \theta_k} q_4 = G_4 \Leftrightarrow e^{\xi_1 \theta_1} e^{\xi_2 \theta_2} (e^{\xi_3 \theta_3} q_4 - q_1) = G_4 - q_1 \tag{A.9}$$

$$\| \prod_{k=1}^{k=3} e^{\xi_k \theta_k} q_4 \| = \| G_4 \| \Rightarrow \| e^{\xi_3 \theta_3} q_4 - q_1 \| = \| G_4 - q_1 \| \tag{A.10}$$

Which reduces the problem to subproblem 3.

For angle θ_2 we have:

Let's define $p_1 = e^{\xi_3\theta_3} q_4 - q_1$. Applying $\tilde{q}_1$ and norm to both sides of A.8 leads to:

$$e^{\xi_1\theta_1}(e^{\xi_2\theta_2} p_1 - \tilde{q}_1) = G_4 - q_1 - \tilde{q}_1 \tag{A.11}$$

and

$$\| e^{\xi_2\theta_2} p_1 - \tilde{q}_1 \| = \| G_4 - q_1 - \tilde{q}_1 \| \tag{A.12}$$

Which reduces the problem to subproblem 3.

For angle θ_1, θ_4, and θ_5 we reduce the problem to subproblem 1. We have:

For θ_1:

$$e^{\xi_1\theta_1} p_2 = G_4 \tag{A.13}$$

where $p_2 = e^{\xi_2\theta_2} e^{\xi_3\theta_3} q_4$

For θ_4:

$$\prod_{k=1}^{k=5} e^{\xi_k\theta_k} = G \Leftrightarrow e^{\xi_4\theta_4} e^{\xi_5\theta_5} = e^{\xi_{-3}\theta_{-3}} e^{\xi_{-2}\theta_{-2}} e^{\xi_{-1}\theta_{-1}} G \tag{A.14}$$

Applying q_6 we obtain:

$$\prod_{k=1}^{k=5} e^{\xi_k\theta_k} = G \Leftrightarrow e^{\xi_4\theta_4} q_6 = e^{-\xi_3\theta_3} e^{-\xi_2\theta_2} e^{-\xi_1\theta_1} G q_6 \tag{A.15}$$

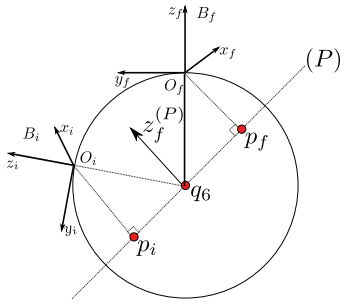


Figure A.2 – The definition of the new problem to determine $\theta 6$. P is the plane define by q_6 and ξ_6. $z_f^{(P)}$ is the unit vector corresponding to the rotation axis of the twist ξ_6. B_f is the base frame corresponding to the desired connector position and orientation, B_i is the initial frame of connector $C3X$ of the second module. p_i and p_f are the projection of the bases B_i and B_f origins on the plane P, respectively.

Finally, for θ_5:

$$\prod_{k=1}^{k=5} e^{\xi_k \theta_k} = G \Leftrightarrow e^{\xi_5 \theta_5} = e^{-\xi_4 \theta_4} e^{-\xi_3 \theta_3} e^{-\xi_2 \theta_2} e^{-\xi_1 \theta_1} G \tag{A.16}$$

Applying $\tilde{q_6}$ we obtain:

$$\prod_{k=1}^{k=5} e^{\xi_k \theta_k} = G \Leftrightarrow e^{\xi_5 \theta_5} \tilde{q_6} = e^{-\xi_4 \theta_4} e^{-\xi_3 \theta_3} e^{-\xi_2 \theta_2} e^{-\xi_1 \theta_1} G \tilde{q_6} \tag{A.17}$$

In order to deduce the value of θ_6 we derive the new following problem. Let's denote by θ_6 the rotation angle around z_f^P (previously named q_6) and β the rotation angle around $O_f p_f$. The problem we need to solve is the following:

Find θ_6 and β such that:

$$e^{\xi_6 \theta_6} e^{\xi_\beta \beta} B_i = B_f \tag{A.18}$$

We apply first p_f to equation A.18:

$$e^{\xi_6 \theta_6} p_f = B p_f \tag{A.19}$$

where $B = B_f B_i^{-1}$

We obtain subproblem 1.

We apply p_i to equation A.18:

$$e^{\xi_\beta \beta} p_i = e^{-\xi_6 \theta_6} B p_i \tag{A.20}$$

Which is equivalent to subproblem 1.

The main limitation of this approach is that a closed form solution might not exist for more complex structures, such as the kinematic chains we are going to consider in the following chapters. That is the main reason why we decided to use a numerical method based on the Levenberg-Marquart algorithm [170].

B Reconfiguration of heterogeneous structures

In Chapter E, we introduced a novel method for the manipulation of passive objects using SRMR. This method is used as one of the main building blocks of a complete framework to allow the construction and deconstruction of arbitrary heterogeneous structures made of passive elements and active units. The problem we aim at solving is the self-reconfiguration of a group of modular robots robots into a structure that includes both active units and fully passive elements. In this chapter we show how the augmented self-reconfiguration problem can be reduced to a multi-robots planning problem and we present a theoretical procedure to perform this reduction.

B.1 Starting configuration

The planning of the reconfiguration process depends on the type of configuration considered. In the case of shape configurations, we reduced the constraints on the placement of the different units but we need a method to convert this defined volume into a set of units (passive and active). We define an alphabet of passive elements and virtual kinematic chains (KC) to fill the structure. The final configuration is filled using the widest kinematic chains that can be created using the available module in the initial configuration. The configuration is filled using a greedy approach starting from the module closer to the connectors inserted in the structured environment and using the virtual kinematic chains sorted accorded to the size of their kinematic space. If only the initial or the final configuration is fully defined, we use the existing units as alphabet for the fitting.

B.2 Kinematic chain identification

We consider a structure composed of active and passive units evolving in a structure environment equipped with connectors. To identify kinematic chains (KC) in the structure, we use the concept of sub-isomorphism in graph. Two graphs G and H are isomorphic if there exists a bijection from the set of vertex of G to the set of vertex of H such that two adjacent

vertex in G are also adjacent in H. For two graphs G_1 and G_2, the sub-graph isomorphism problem consists in determining whether a graph G_1 contains a subgraph that is isomorphic to G_2. In our case, we want to be able to identify sub-graphs that corresponds to virtual kinematic chains inside the structure. The sub-graph isomorphism problem has been proven to be NP-complete [67]. We consider two different algorithms with exponential complexity to solve this problem: the VF2 algorithm [68] and the McGregor's algorithm [172]. They are both implemented in the Boost Graph library [229].

B.3 Representation

The final configuration of the active and passive units is represented by two undirected graphs. The first graph, called the *connection graph* (similar to the *C-Graph* mentioned in subsection 3.1.1), represents the physical connection between the units and the structured environment: one node corresponds to one unit and one edge to a physical connection. We differentiate between active and passive units using two different colors (attribute) for the nodes. The connectors in the structured environment are also represented as node in the graph with a different color. The second graph, called the *density graph*, is a fully connected weighted graph in which each node corresponds to a unit or to a connector. For this graph, the weights are the Euclidean distances between the two center of mass of the connected units. For each graph we introduced a metric, respectively the *level* and the *isolation* for the connection graph and the density graph respectively, that will be later used to define an ordering in the assembly process. They are defined as follows:

1. The *isolation*, I, of a node captures the geometric occupancy of the space near a unit. It can be seen as a kind of voxel density measurement. It reflects the reachability of the units and its ability to move in its surrounding environment (the more isolated a node is, the easier it is for it to move). To account for those aspects, we explore the notion of *centrality* in graph [29], widely used for real world networks study (see [29] and references within). We investigate different expressions for the isolation of a node using the notion of centrality measure in graphs. We can compare different notions of centrality: the *degree centrality* [92], named DC, and the *sub-graphs centrality*, named SC, introduced by Estrada et al [84]. The DC allows for a local definition of the density influenced only by the nearest neighbors connectivity. It corresponds to the marginals of the adjacency matrix of the considered graph. The SC additionally takes into account the sub-graphs including the selected node, with a decreasing weight with respect to the size of the sub-graph. It has been used in various real world applications [83], and prominently in molecular chemistry to study the protein folding process [84]. Their formal definitions are as follows: let $G = (V, E)$ be a graph of order N and A the associated adjacency matrix. Let $v_1, v_2, ..., v_N$ and $\lambda_1, \lambda_2, ..., \lambda_N$ be the eigenvectors and eigenvalues

associated to A and w_{ij} be the weight on the edge between node i and j. We have:

$$\forall i \in E \quad DC(i) = \sum_j a_{ij} = \sum_j w_{ij} \tag{B.1}$$

and

$$\forall i \in E \quad SC(i) = \sum_{j=1}^{N} (v_j^i)^2 e^{\lambda_j} \tag{B.2}$$

2. The *level*, H, of a node: to ensure that the final configuration can be physically constructed, we have to ensure that the units are placed following a bottom up approach starting from the structured environment connectors. The level of a node corresponds to the shortest distance between the connectors inside this set and the node. The level of a node is close to the definition of *closeness centrality* [92], except we only consider the set of embedded connectors as starting point for the measurement of the distance (as opposed to the complete set of nodes).

B.4 Disassembly planning

The usual approaches for planning the building of a given structure using SRMR take as an input the final shape of the structure that we would like to construct, the initial position of the different active units, as well as the potential obstacles in the working space. A plan for the moves of the different units is then computed using various heuristics or exact approaches (as presented in chapter 4). These techniques are well suited for homogeneous structures, but lack the aspect of collaborative manipulation of passive objects. To cope with this issue we proposed to a use technique similar to the one developed to solve automated assembly and disassembly of industrial products [154]. In our case, instead of trying to reach the final configuration starting from the initial one, we "de-construct" sequentially the final structure depending on the available kinematic chains we have previously identified in both configurations. We assume that the assembly and disassembly processes are reversible. This assumption is reasonable considering that no blocking actions are performed by the units during the process. Several approaches have been introduced to find the sequence of steps that would lead to the disassembly of the structure but none have been applied to the self-reconfiguration problem. *Precedence graph* (see Lambert 2003 [151] for a general review) have been proposed based on *liaison matrices* that encode the physical relationship between the different parts of an object (for example, two plates linked by a screw). An early approach to obtain those precedence graph has been developed by Laperrière et al. in 1991 [154]. More recent techniques include semantic planning [157], Petri nets [179], Octrees [173], and manipulation primitives nets [259].

Our problem differs from the one tackle in the previously cited contributions in the sense that we not only have to take into account the initial structure but also the final arrangement we

would like to achieve. We propose to capture the precedence constraints in both structures by introducing a measure of the voxel density around the units coupled with a measure of distance between the units and the closest anchor point in the structured environment. The method to define the different disassembly and assembly action or tasks, to be performed at each step is described in the following section.

B.5 Task definition

In order to schedule the different tasks that the active units will have to perform, we use the previously introduced metrics, isolation and level, both on the final and initial configuration. At each step of the procedure, we maintain two lists containing the available units in the initial configurations, L_i, and the required units in the final configuration, L_f. The lists are filled as follows: at each step k, we compute the indexes H and L for all the nodes, and

- In the final configuration, we sort the nodes by increasing order of level and then, by increasing number of isolation. In other words, the first elements in L_f^k will be the bottom layer of the final structure and the less isolated ones (meaning the most difficult to reach) will be included first. We then identify the potential KC using the sub-graph isomorphism introduced in subsection B.2. We only allow KC to contain units with consecutive levels. We compute an average value of the isolation of the KC equals to the average of the isolation of the single units and sort the KC based on this indicator.
- In the initial configuration, we sort the nodes by decreasing order of level and then, by decreasing order of isolation. Intuitively, it means that the first elements of L_i^k will be in the "top" layer of the initial structure, sorted according to the amount of free space in their surroundings. We add another constraint to the nodes in the L_i list to ensure the structural consistency of the initial configuration after the removal of the node: we simulate the disconnection of the node and check whether the structure is still stable using an approximate model of the structure (a physical simulation). At the end L_i^k contains the units that can move at step k.

In order to allow for the reuse of kinematic chains during the reconfiguration, we create in the structured environment a zone in which the units that are currently not used but that can still move (and that might prevent the access to more internal units in the structure) can go and play the role of reserve units or "helper" units. We called this space the *Helper zone*, abbreviated Z_h. This zone is defined using the previously introduced metrics by choosing the connectors with the highest isolation index. After the filling of the two lists, we compare the elements of L_i^k and L_f^k and try to match them. Among the units in L_i^k we match the KC that could be used to fill the corresponding position in L_f^k using the planner that we previously developed. If no match is found (meaning that either there are no units that could be matched to the KC or that the planner was not able to find the path to the final position), the units are added to the helper zone. We then update the list L_i^k and start again the matching process. If

there exists a match between a unit in L_i^k and in L_f^k we compute the required path towards the final configuration. We then switch to the next matching KC. The matching will then also be checked also taking into account the helper units. The process continue as long as L_i is not empty: at the end all the units should either be in L_f or in Z_h. A unit or a KC with the highest level in the structure at step k and with a high isolation (allowing it to move) in the final structure can also be considered as an helper unit.

One of the main drawbacks of this method is that the structured environment should be large enough to be able to create an helper zone.

The previous method output at each step a list of units that should reach a given position and orientation. At each step, the initial and final configuration are updated, but the timing between the actions (moves, connection, and disconnection) leading from one step to the next still need to be defined. Now that we have derived at each step the required task to be performed, we need to find the optimal scheduling between them in terms of time constraint.

Using our hierarchical manipulation planner and the aforementioned decomposition technique we manage to abstract away the constraints and complexity linked to the joint manipulation of passive elements and their inclusion into an heterogeneous structure. This way we have shown that the ASR problem can be reduced, at each step, to a "classic" multi-robot planning problem to which we can apply existing algorithms. We present some of them in the next section.

B.6 Task scheduling

The planning algorithm that we have presented in the previous section relies on the $A^\star$ algorithm. Many planning techniques have been proposed to find a near optimal scheduling of one or multiple units [155, 157]. A planning algorithm can be evaluated using three main criteria [204]: completeness (if one solution exists, then the algorithm will find it), complexity, and optimality (the algorithm output the optimal solution). The methods for planning of multiple units fall in two main categories: *coupled* methods that often rely on a complete search algorithm like $A^\star$ to achieve completeness and optimality, and *decoupled* methods that combines single states from the different units to create a complete plan. Unfortunately, it has been shown [114] that the motion planning of multiple units is a PSPACE-Hard problem. The configuration space grows exponentially with the number of robots and the search performed by the centralized algorithms underlying the coupled methods quickly becomes intractable. Methods have been introduced to reduce the search space (e.g. probabilistic roadmaps [250]), but they are lacking scalability. To tackle this issue, the decoupled methods sacrify optimality and completeness for the sake of complexity. In these methods, the planning of the motion is done at the level of each individual robotic unit and then combined to ensure collision free paths. Those methods can be either centralized or decentralized. A recent distributed decoupled approach introduced by Peasgood et al [204] has been shown to be scalable (with a

complexity linear in the number of moving units) and complete. It is based on a multiphase planning using a topological graph and spanning tree representation of the problem. Another approach [95] propose a new sensor-based Path Planner based on Voronoi Graph. The method has been shown to be fast for both local or global motion planning and able to take into account new obstacles included in the terrain. Other successful tasks scheduling methods include [289, 16, 44].

B.7 Conclusion

In this section we presented preliminary results on a method to reduce the problem of augmented self-reconfiguration to a multi-robot path planning problem. To do so, we introduced a deconstruction planning algorithm based on the notion of centrality in graph theory. This reduction relies on the use of the hierarchical planner that we introduced in section 5.1. This planner is applied on a discretization of the problem based on the creation of two lists of modules at each step, one corresponding to the positions (i.e. the position and orientation active units and potentially the passive elements) to be filled in the final structure, the other containing the available active units in the initial configuration at a given step.

C Computer based interfaces

In this part, we tackle the problem of designing an easy to use $3D$ interface for building pieces of furniture made of Roombots modules. We explore the different possibilities for simplifying the problem without restraining to much the capabilities of the modules. We present an intermediate solution where a $3D$ regular grid is used with cubes as basic units. Algorithms for converting the cube structure into a real Roombots shape are presented. A complete designing application incorporating all these elements is fully described.

C.1 Introduction

The Roombots project aims at designing adaptive furniture able to self-reconfigure and locomote. A Roombots (RB) module has three degrees of freedom (DOF) and ten connectors. Several modules can connect together using an active connection mechanism to form a meta-structure (also called shape).

Due to the complexity of the RB module, the building of shape can become tedious for a non expert user. To ease this process and allow fast prototyping, small replica of the real module were built (figure C.1). These *mockups* use a passive magnetic connection mechanism to connect to each other. Despite this simplification in the connection mechanism, the complexity induced by the three degrees of freedom remains. The goal of this part is to proposed a new interface to allow a fast assembling of a RB structure for lay users.

This chapter is organized as follows. In a first section (section C.2) we define more precisely the problem we are trying to tackle. We then describe some related works in the field of space filling and designing tools, with a particular focus on techniques and softwares applied and used in modular robotics (section C.5). In section C.6 we describe our proposed solutions. We then present the actual implementation of our software (section C.7) . Finally (section C.8) we mentioned some possible improvements to the current design after concluding.

Figure C.1 – A structure made of mockups.

C.2 Problem definition

In this section, we aim at giving an overview of the different possibilities that could be considered when designing an end-user interface for building purposes. We start by identifying the basic elements of the assembled structure. For each of them we briefly describe the proposed solution and present some of the related pros and cons. We then examine the possible type of interfaces we could consider for our problem.

C.3 The basic elements

Several options can be considered for the basic elements of the interface: spheres, cubes, roombot modules or lines. We present each of them in this section.

C.3.1 Spheres and cubes

It might be the most intuitive structure to consider. The user will build the desired configuration by using cubes or spheres, which can be viewed as “half” a roombot module.

- **Pros**
  - Easy to manipulate
  - Good representation of the structure for the user
- **Cons**
  - Requires an algorithm to convert it into a “true” roombot structure
  - The user might end up building impossible structure (in this case, we need a “recommender” system to modify or help the user modifying the structure)

C.3.2 Roombots module

We could also use directly the entire roombot module (or a slightly modified/simplified version).

- **Pros**
  - The full "potential" of the roombots module (degrees of freedom, size, ...) can be used
  - The structure can always be created
  - The user can create richer structures
- **Cons**
  - More complicated for the user
  - More difficult to implement

C.3.3 Lines (sketch mode)

We could imagine that the user only have a very rough idea of the structure she wants to create. In this case, she might not be willing to use precise shapes like spheres, cubes or roombots modules, but only draw some sketch of the shape she would like to design.

- **Pros**
  - Easy to use
  - No complex interface (basic drawing tool in 3D)
- **Cons**
  - The final structure might be difficult to imagine for the user
  - An algorithm need to be used to convert this drawing into a valid roombots structure

C.3.4 Conclusion

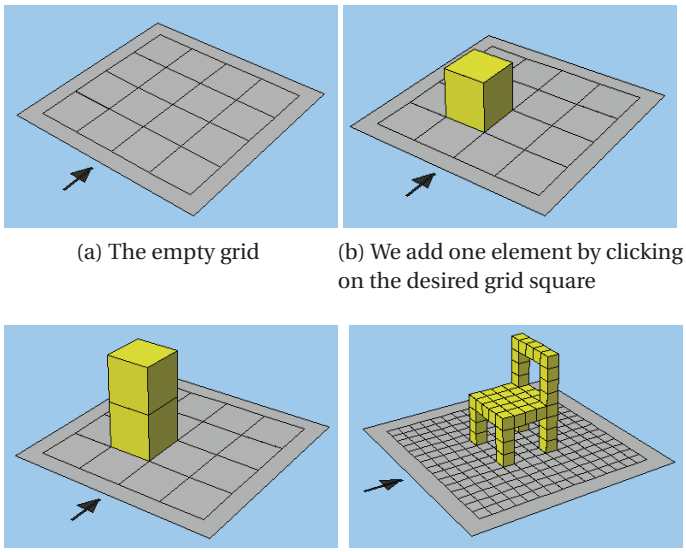
The structure that seems to best fit our need, both in terms of complexity to manipulate and time to implement, would be the cube or sphere unit. Indeed it is not too far from the real RB module, as opposed to the line sketch, but induces a valuable simplification in comparison with the use of RB shaped elements.

C.4 The interfaces

After selecting the type of basic elements we want to use, we need to choose the type of interface.

C.4.1 Single layer

In this type of interface, the user build the structure starting from a $2D$ grid (representing the ground), as shown in figure C.2. At each action of the mouse corresponds a unique result: insertion or deletion of a object. An example of such an interface can be tested at [122].



(a) The empty grid

(b) We add one element by clicking on the desired grid square

(c) Another element can be added on top of the previous one

(d) An example of final structure

Figure C.2 – The process of building a simple structure (screenshots from [122]).

- **Pros**
  - Clear mouse interaction
  - Not too hard to implement (existing code can be re-used)
- **Cons**
  - The user can not build the structure from any point (see figure C.3 for an example)
  - The user can not copy/past multiple elements

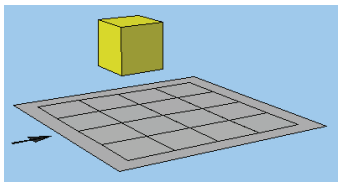


Figure C.3 – This structure can only be built by removing a module (after steps C.2b and C.2c in figure C.2). Screenshot taken from [122].

C.4.2 Multiple layer/3*D* grid

This solution can be viewed as an extension of the previous one: instead of having only one layer (equivalent to the ground), we consider multiple layers which create a kind a 3*D* grid inside the space. The user can switch between the different layers to put a basic element at any position in the 3*D* grid.

- **Pros**
  - More flexible than the previous one
- **Cons**
  - Might be less intuitive to use
  - The user is still limited by the 3*D* grid

C.4.3 "Free" interface

In this kind of interface, the user can place the elements wherever she wants inside the space. The possibilities of the interface are the following:

- Drag/Drop elements
- Align elements (with respect to symmetry axis, external objects,..)
- Display reachable connectors (hover events)
- Allowing rotation of multiple parts
- Multiple elements selection (as well as copy/past actions)

As for the previous interfaces some advantages and drawbacks can be enlightened:

- **Pros**

  - Highly flexible

- **Cons**

  - Hard and long to code
  - The representation in a $3D$ space might be difficult to apprehend for the user

C.4.4 Conclusion

The natural interface for the design of a structure would be the "free" interface, since the user is not limited in her creation. Nevertheless the complexity of the representation might become an hindrance in the process. As a consequence, we have chosen to start by implementing a single layer interface which can then be extended in a multilayer architecture in a pretty straightforward way.

C.5 Related works

We have previously described the basic elements and the type of interface we were planning to use to tackle our problem. The final structure will be represented as a $3D$ volume made of basic regular units. The problem of fitting RB modules inside this volume can be seen as a packing problem. In this section we first review existing methods to efficiently solve the bin-packing problem as well as the space filling problematic. We then review existing software in the field of design tools for modular structure.

C.5.1 Packing and space filling problems

Our problem is similar to two famous computational optimization problems: the bin-packing problem and the space filling problem. Indeed, on the one hand, we could think of letting the user design a volume in $3D$ and use the algorithm to fill this volume with the RB modules. On the other hand, we could imagine having to fold an already existing RB structure inside a given volume optimally, i.e. by maximizing the filling ratio.

The packing problem can be defined as finding the best arrangement of a set of objects inside a given volume in the sense of the minimization of the unoccupied space. The items can be homogeneous (*uniform* packing problems) or heterogeneous. If we impose a perfect packing (i.e. with no gaps), the problem is called a *tessellation* or *tilling* problem. An extension of the basic packing problem consists in optimizing the number of containers (in size and/or in number) to carry a given set of objects. This problem is particularly relevant in stocks management.

The three dimensional packing algorithms can mainly have two goals. Starting with a given set of items (referred as I) and a given volume to be filled (referred as V), the algorithm has to maximizing the occupied volume in V. It can also have to minimize the number of containers required to carry the set I. Unfortunately, these two classical combinatorial optimization problems have been shown to be strongly NP-hard ([81]). Several heuristics have been proposed to tackle this issue. A pattern based method using a tree search approach developed in [40] has proved to be fast on large scale problems but also to exhibit a good filling ratio. Another approach uses multi-faced building process to improve the filling factor of the algorithm ([161]). Exact methods have been presented to solve this problem (see for example [171]), but the computational explosion leads to very poor time performance.

The problem of space filling is widely studied in the domain of biology: proteins folding mechanisms allow a fast change in property as well as a huge gain in space. Similar technique have been applied to modular robotics. In [13] for example, the authors use Hamiltonian path method to fill space with a lattice type modular robots composed of tetrahedral units.

C.5.2 Existing software

We briefly present the main existing solutions related to the creation of $3D$ structures using basic units as building blocks.

The LEGO Digital Designer

This software has been developed by the *Lego* firm. The user can use a set of existing Lego pieces to build her structure. She can also copy/paste, drag and drop elements and manage multiple shapes in the same environment. The interface is really intuitive and user-friendly. Unfortunately, this software is not open source and is developed for *Mac* and *Windows* platforms only. Some demos can be found at [158] (in the "get started" section).

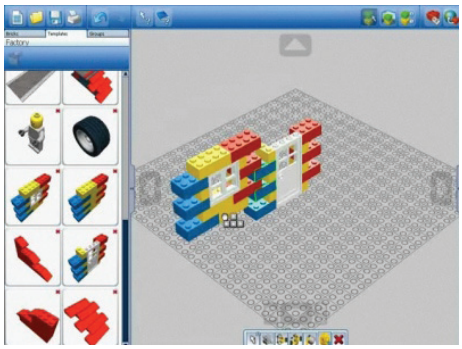


Figure C.4 – The Lego designer software (adapted from [158])

An open source software: LDraw

LDraw [124] offers almost the same functionalities as the “official” Lego designer. Nevertheless, its interface seems to be more similar to professional CAD design tools: multiple views, tree structure for the description of an object, ... This leads to a less intuitive software, not very well suited to lay users. Moreover the development of the linux libraries seems to have been stopped in 2004 whereas the windows equivalent is still being maintained.

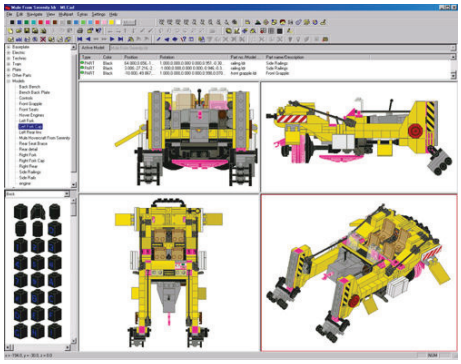


Figure C.5 – The LDRAW application (adapted from [124]).

IMOROD

A previous project called *IMOROD* aimed at designing a 3*D* interface to allow users to create a structure composed of modular robots. The elements chosen for the interface were the *YAMOR* module [176]. This software was more intended for expert user and lab environment. More detailed information regarding this project can be found in [96].

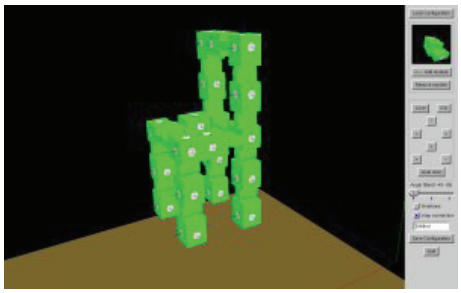


Figure C.6 – IMOROD: the simulation environment for the YAMOR robots (adapted from [96])

VUG

V.U.G stands for Virtual Universe Generator [85]. This open source project aims at proposing an easy to use user interface for virtual world creation. The user is able to create and modify the different elements of the world (objects, robots, ...) as well as its intrinsic properties (physic laws,...). Objects can be added or removed. The user can also move in the world and interact

with it.

Figure C.7 – The VUG software: a manipulator is represented inside a house environment (adapted from [85]).

Conclusion

We have seen that several solutions exist both in terms of algorithm for the theoretical aspects of our problem but also from a practical point of view, with multiple design softwares. The Lego designer seems to be a really mature solution regarding the aspect of user-friendliness. LDraw and VUG are promising but, whereas the first one is complicated to apprehend because of its CAD like interface, the second one is still at a development stage. IMOROD was a great tool for expert users but lack the ease of use we are looking for with our interface. Consequently, we plan on using an interface similar to the Lego designer, but considering a very limited amount of possible pieces (mainly cubes and/or a few passive elements).

C.6 Proposed solution

In this section we present a possible solution to the problem of constructing in a $3D$ environment, a structure made of RB modules. As we have seen in the previous sections, we will consider that the constructed structure is made of regular identical cubes aligned in a $3D$ grid.

We will first describe how the created structure is represented using a planar graph (section C.6.1). Then we explain the two main solutions we have explored to perform the conversion between the cube structure and the RB shape (sections C.6.2 and C.6.3). Finally we describe the recommender system that assist the user in the building process of the RB structure (section C.6.4).

C.6.1 Roombots structure representation

The built structure is composed of cubes connected together. A natural representation of this shape is to use a graph with a node corresponding to a cube and a vertex to a connection between two cubes. An example of such a representation is presented in figure C.8.

C.6.2 First solution: biconnected components

Algorithm

We consider in this section that the following assumptions hold:

1. The cubes can be moved only in a Cartesian $3D$ grid
2. Only face to face connections are allowed

We need first a characterization of a *valid* cube structure. For now, the only requirement that will be imposed on it will be to have an even number of cubes. Nevertheless we will also use the following rule to identify problematic nodes:

If a node contains more than 2 *isolated nodes* [1] *in its neighborhood (level* 1*) then the structure is invalid*

Now, we need an algorithm to convert the cubes into modules. We can propose a first method, based on graph theory notions. It can be decomposed into 7 main steps.

1. **Check the structure validity**
2. **Find body and limbs in the structure**
   This step is based on a previous work done on body/limbs recognition in RB structures. More details can be found in [26].
3. **Compute the distance to the body for each spheres**
   If no body has been found in the structure, we consider that each sphere is at a distance 0.
4. **Find the bi-connected components (bcc) of size 2** [2]
5. **Create distance based groups of bcc**
   The distance of a component to the body is defined as the mean distance of its modules. A group contains all the bcc at the same distance from the body.

[1] A node is said to be isolated if and only if it is connected to one or less other nodes
[2] see [27] for definitions

6. **Make pair with the components in the group furthest from the body**
   This step ends when all the components in the group have been used. In case of tie, a random selection mechanism is used.

7. **Remove the previous bcc from the initial structure and go back to step** 2
   If there are no more cubes the algorithm stops.

The major issue is that the structure build by the user might be impossible to convert into a roombots structure. For example, consider the structure presented in figure C.9 (represented in $2D$ for convenience and without connections for the sack of readability). As one can see, the number of module is valid, but the structure can not be created due to the circled part.

Several options can be considered in order to tackle this issue:

1. **Ask the user to modify the structure**
   The system can point out which part of the structure is invalid using the rule previously defined.

2. **Modify the structure automatically based on:**

   (a) **Symmetries preservation**: one module can be added or removed to maintain or increase the global number of symmetries

   (b) **Shape preservation**: try to apply a transformation (dilation, refinement,...) which preserves the global shape of the structure

Tests and results

Generating random graphs In order to test the algorithm, we needed to generate random graphs. We used one of the randomized graph generators implemented in the *igraph* library ([120]): the Erdös-Rényi graph. More details about this type of graphs and their properties can be found in [27].

Results We tested the algorithm on randomly generated graphs. The number of nodes cannot be controlled but we nevertheless imposed a minimum size of 20 nodes. We eliminated unconnected graphs as well as graphs with an odd number of nodes. We also took into account the number of invalid structures (as defined by *rule 1* in the previous report). The results are the following: for 1000 valid structures the algorithm was able to make pairs with the nodes in $\sim 90\%$ of cases.

The remaining cases are "impossible", meaning that a pairing for the structure cannot be found. One example of such a structure is represented in figure C.10.

Conclusion Despite the good results exhibited by this solution, it lacks simplicity and fast execution time for large structures. As a consequence, we looked for a more direct method in the field of graph analysis and computational graph theory.

C.6.3 Second solution: perfect matching

The RB modules are made of two cube-like parts connected together by a central degree of freedom. Considering the previous representation, it would seem consistent to convert two cubes linked together into a RB modules. More formally, we are looking for a matching between the graph G_i, representing the structure made of cubes, and the final graph G_f, composed of node corresponding to a real RB module. Each node in G_f should correspond to two connected nodes in G_i. The problem of finding a set of edges without common vertices in a graph is known as the *matching* problem. The matching is said to be *perfect* if every vertex of the graph is incident to one and only one edge of the matching [27]. This concept is illustrated in figure C.11.

The existing matching algorithms can also deal with weighted connections in the graph: the goal is then to find a perfect matching while minimizing or maximizing its overall weight. This problem can be solved for bipartite graphs using the Bellman-Ford algorithm or the Hungarian algorithm with a complexity of $O(C^2 log(C) + CN)$ where C is the number of connections and N the number of nodes in the graph. For non bipartite graphs, algorithms with a complexity of $O(\sqrt{C}N)$ have been proposed (see [174] for example).

Since this method is already efficiently implemented in the classical graph analysis libraries, we decided to chose it to solve our conversion problem.

C.6.4 Recommender system

In order to guide the user in the building of the desired structure, we propose a basic recommender system. It will be based on the following rules, which will be checked throughout the construction process:

1. In order to be built, the structure has to be composed of an even number of cubes.
2. In case of *disconnective cycles*[3] (DC) the user will be proposed two solutions:
    - Add node(s) to the DC in the direct neighborhood of the isolated nodes. In order not to break the first rule, two nodes has to be added. We try to ensure that we neither introduce nor break the symmetry of the structure by placing these two nodes the furthest away from the isolated nodes and from each other.

[3] A set of nodes in a graph will be said to be "disconnective" if its removal would lead to isolated node(s), i.e. to node(s) with no connections.

- Remove isolated node(s) until a perfect matching is found. This solution might end up breaking the symmetries in the structure. We thus prefer the previous option.

C.6.5 Conclusion

We have explored in this section the ways of converting the constructed structure made of cubes into a shape composed of real RB modules. After proposing our own algorithm based on previous work, we presented a solution from the computational graph theory field which allows for faster results. Finally we proposed some possible rules for our recommender system.

We describe in the next section the actual implementation of our design software.

C.7 Implementation

In this section we describe the implementation of the designing software we proposed to incorporate all the previously defined features. After briefly reminding the interface requirements, we present our solution. We then summarize the code structure after giving a short introduction to $3D$ designing applications architecture.

C.7.1 Requirements and proposed solution

The goal of the interface is to provide an easy to use $3D$ environment to build a structure made of cubes and visualize its equivalent in RB modules. The user should be able to see in real time how the changes on the cube structure impact on the RB representation, to rotate the constructed structure, to zoom in and out and to save and load structures.

We propose to create an application composed of two views and a side panel. The first view will be the construction view, in which the user can build in an invisible $3D$ grid her structure starting from a single cube. The second view represents the structure made of a rendered version of the RB modules. It will be dynamically generated each time the user modify the building view. The side panel will store the already built structures in a list like way. The general organization of the interface can be seen in figure C.12.

C.7.2 Code structure

General structure of a $3D$ designing application

Most of the time, a $3D$ modeling application is composed of 3 main elements:

- **The physics engine:**
  This module is used to simulate physics models (gravity, velocity,...). Several physical engines exist: *PhysX* (proprietary, by Nvidia), *Newton* (closed source), *Bullet* (open source),...
- **The *3D* engine:**
  This module is mainly used to render the different graphical elements. As for the previous module, there exists plenty of different 3*D* engines: *Ogre3D* (open source LGPL), *Open Scene Graph* (open source),...
- **The *GUI* library:**
  The *GUI* library is used to design the interface of the application (widgets, buttons, layouts, ...).

Most of the time, a **wrapper** is used to facilitate the use of the physics engine functions with those provide by the 3*D* engine. It should be kept in mind that the wrapper is, in a lot of cases, a work done by "volunteers". Thus, only a part of the functionalities of the 3*D* engine is available through it. The 3*D* engine can use two main types of graphical libraries: **OpenGL** (open source and cross-platform) or **DirectX** (windows specific).

Example: the *molecubes* simulator

The *molecubes* project ([256]) is a modular robotics project aiming at designing cheap and easy to use robotics modules. The molecubes team has developed an advanced simulator to manipulate and control created structures. The following elements were used: *Ogre3D* as a 3*D* engine, *PhysX* as a physic engine *NxOgre* as a wrapper and *CEGUI* as a *GUI* library.

Choices for our application

We have chosen a free opensource 3*D* engine optimized for OpenGL called *Open Scene Graph* ([66]). The main reasons why we decided to use this application are the stability and the cleanliness of the code, the optimization for OpenGL and its great portability. For the windows manager we used the *Qt* framework ([69]) for the richness of its features and its stability. For now we do not use a physics engine since our application is mainly intended as a designing tool. For the sake of consistency, the application and all the library used are written in C++. For the graph analysis, we have used the *Lemon* library ([160]) because of its efficiency.

Structure

The overall structure of the code is depicted in figure C.13.

The application is composed of three main blocks: the Input/Output module, the Graph module and the View module. They are linked together inside a main application called the

MainWindow of the software.

Input/Output module This class manages the loading and saving of the built structure. The configuration is saved into an *xml* file listing the position of the different faces of the cubes in the $3D$ space.

Graph class The conversion of the cube structure into an undirected graph and its analysis are done in this module. The main function called the perfect matching routine of the *lemon* library. The detection of isolated node and odd configuration is also handled by this module. Unfortunately, due to time constraints, the recommender system was not fully implemented and only a basic checking can be performed, without real interaction with the user.

View The view is composed of two main elements:

1. The OpenGL representation: the structure is represented by a set of basic elements (cube, lines, ...) on the screen.
2. The Interaction Handler: this module manages all the interaction between the input devices (mouse and keyboard) and the $3D$ representation. Their is one interaction handler for each view.

C.8 Conclusion and future works

C.8.1 Conclusion

In this chapter, we have described a new interface to easily build structure made of Roombots modules. We use an intermediate representation made of cubes to simplify the process. This structure is then converted into a configuration made of real RB units by using a perfect matching algorithm applied to the graph equivalent of the cube structure. We have designed a complete software which allows the user to create the desired shape, interact with it and visualise the equivalent RB configuration in real time.

C.8.2 Discussion and future work

On a theoretical and analytical level, several aspects still need to be investigated.

Firstly, the fact of restricting the building of the structure to a $3D$ grid restrains the capability of the modules and, as a consequence, the possible shapes that can be created out of them.

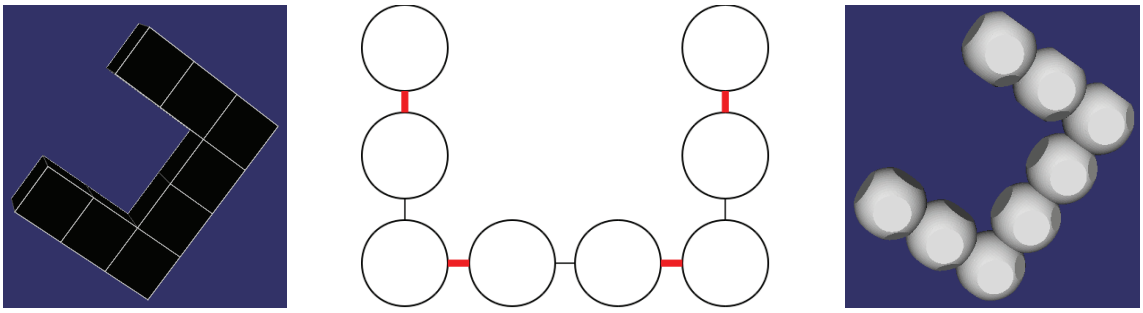
This constraint could be partially lifted if the size ratio of our structure would not be taken into account. Indeed, any structure can be approximated up to a certain precision by increasing the number of cubes. Nevertheless this granularity increase does not preserve the overall size factor of the object, since the RB modules have fixed dimensions.

Secondly, the perfect matching technique use in this project does not take into account the future use intended for the structure. For example, we might think of designing the resulting structure so that it would be easy to build using the current reconfiguration algorithm already developed at BioRob ([236]). Similarly the locomotion of the RB shape depends on the orientation of the different modules and the type of connections between them. These information are currently not included when performing the conversion from the cubes configuration to the RB shape.

In terms of software functionality, we can identify some natural extensions and improvements of the current version.

If we intend to create a tool for designing real piece of furniture, we need to provide to the user the ability of managing several structures at the same time. We should also allow her to place the different configuration inside a virtual environment, as it is done in the VUG framework. To create the shapes, it might definitely be useful to be able to drag, drop and copy existing pieces of structure to ease the creation process. Including passive elements would also allow for more variety in the resulting configuration.

The recommender system should also be improved to handle more difficult cases. A measure of the complexity of the building process for the given shape has to be developed as well as a similarity measurement between structures, so that a structure difficult to build could be approximated by an easier similar one.



(a) (b) (c)

Figure C.8 – The three representations of the structure: the cube structure (a), the corresponding graph with the perfect matching shown in red (b) and the created Roombots structure (c).

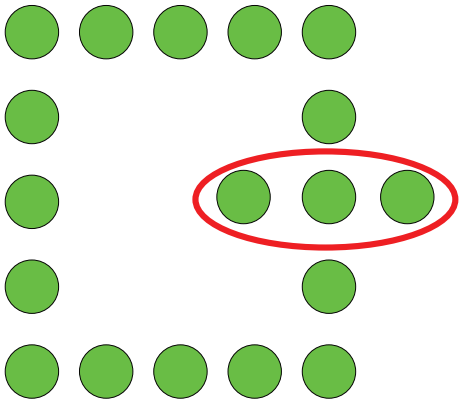


Figure C.9 – The spheres structure

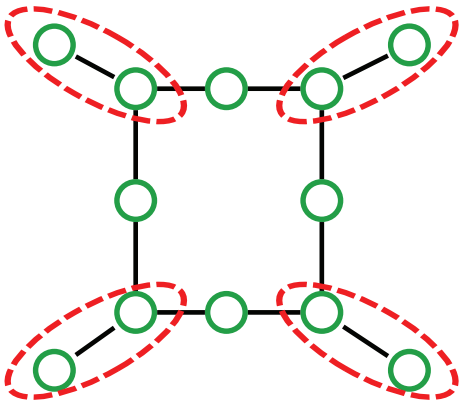


Figure C.10 – An “impossible” structure

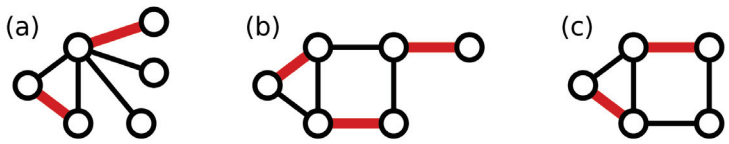


Figure C.11 – Illustration of graph matching. For a) and c) no perfect matching exists (adapted from [277])

.

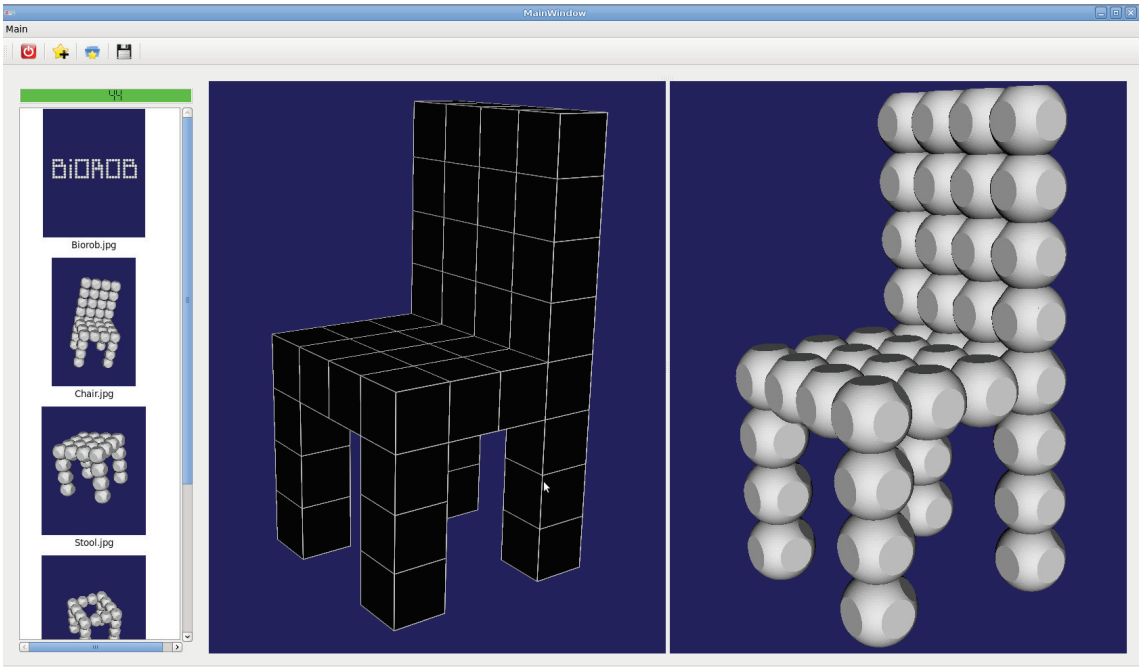


Figure C.12 – A screenshot of the proposed interface. On the left, the panel displays the saved configuration. The left view shows the cube structure and the right view the rendered structure.

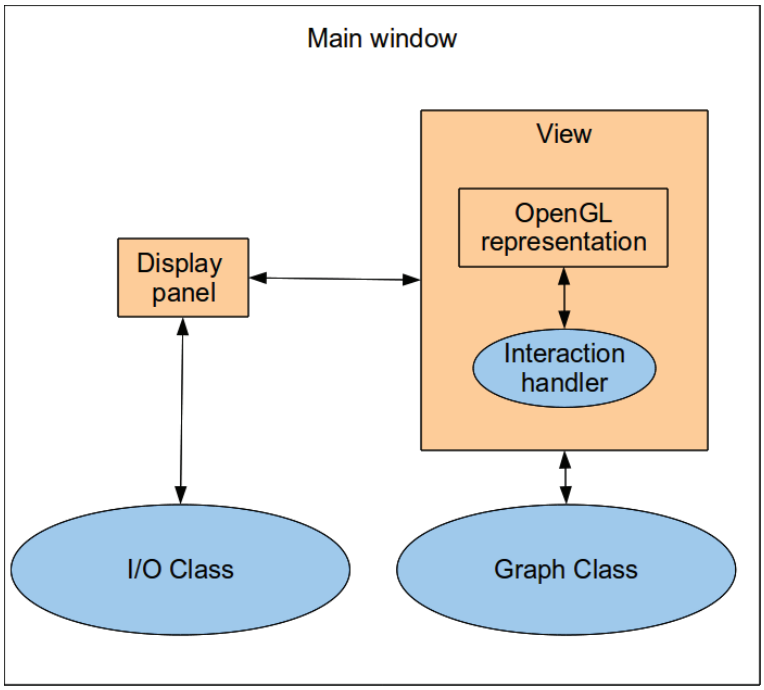


Figure C.13 – The structure of the 3*D* assembler code

D Mobile control interface for modular robots

References and contributions
This section is based on the following semester project:

L. Girod, "Mobile control interface for modular robots", Semester Project, École Polytechnique Fédérale de Lausanne (EPFL), 2014. Available at: http://biorob.epfl.ch/page-110120-en.html

My contributions were:

- *general guidance during the project.*
- *proposed SLAM like method (PTAM).*

The external contributions were:

- *implementation of the method.*
- *test of the method and possible improvement.*

The initial goal of this project was to improve our previously developed tablet interface (described in chapter 8) by removing the need for an external tracking device. We found an already developed augmented reality software by Klein at al. [138, 137] to demonstrate the efficiency of their SLAM like method called PTAM (Parallel Tracking and Mapping). PTAM is similar to SLAM algorithms: it initially recognizes points of interest in the pictures and construct an estimate of the camera position and orientation called keyframe; at each frame the algorithm tracks the displacement of these points of interest and create a new keyframe. The position of the camera is corrected using probabilistic methods. One of the main limitation of PTAM is the restriction to a unique map of points of interest that confines the algorithm to work on a limited area. An extension of this algorithm called PTAMM (Parallel Tracking and Multiple Mapping) [46] proposed a solution to this problem by managing a multiple maps. We

used an easier approach to the problem by triggering the rendering of the scene only when the probabilistic score corresponding to the confidence interval of recognizing the point of interest was high enough. In other words, we only draw the AR scene when there is a high probability that we are pointing in the direction of the scene.

E Collaborative interface for mixed team of humans and robots

References and contributions
This section is based on a personal project conducted during the "Autonomous Robots" course in 2011. We received guidance from our supervisors, José Nuno Ferreira Maia Pereira and Professor Alcherio Martinoli

In this chapter, we review the different aspects of Human Robot Interaction (HRI) in the domain of coordination of mixed team of humans and robots. We explore the different existing metrics in this domain and emphasize the need for common, task independent metrics. We insist on the aspects of coordination and cooperation and present a possible taxonomy for the evaluation in this domain. We describe also some high level implementation in the domain of multi-agent systems and analyse examples of real world applications. We investigate the concept of autonomy and awareness and mention a new approach for designing collaborative architectures. Finally, we present some social implications induced by these mixed teams of humans and robots, such as trust, cognitive models or roles playing.

E.1 Introduction

The field of *Human-Computer Interaction* (HCI) has been extremely active in the past ten years, allowing great progresses in the domain of interaction of humans with computer-based technologies but also in the one of social studies, with the implication of these interaction in our everyday life. Most of the time these studies have been performed in a situation of one to one interaction between a single robot and a human. The *control* of groups of robots has also greatly evolved and matured since its early days and it is now possible to drive efficiently swarms of robots to achieve particular tasks. Nevertheless much fewer work has been done in the study of mixed team of humans and robots, both at the level of the control and at the level of interaction, cooperation and coordination. The domain of *Human Robot Interaction* has often been reduced to a particular case of HCI, neglecting the differences induced by the implication of the devices in the real environment (in terms of perception of the robots for

example) and the underlying social issues. As the robots became more and more autonomous and efficient, this branch has tended to specialise by affirming its differences. Nonetheless, most of the studies have been made in this field considering the humans as external to the task (like supervisors) rather than as a member of the team.

In this work we investigate a more general architecture in which not only the number of robots and humans is not limited to one but also the team is made of humans and robots interacting, collaborating and cooperating together to achieve a particular task.

In the first section (section E.2) we describe the possible metrics that can be applied to this problem. Then (section E.3) we discuss the implementation of such system both in terms of abstract architecture and concrete applications. The notion of autonomy and the related concept of awareness are explained in section E.4. Finally, we explore in section E.5 the social implication of such an heterogeneous system.

E.2 Metrics

The notion of metrics is crucial in many domains as it allows evaluation of methods as well as comparison between them, but also a prediction of the system performance, effectiveness and robustness. In HRI, the main difficulty is often the task dependency of the measurement: a metric can be well suited to a particular task but meaningless in another context.

The newly used mixed human/robots teams in HRI raise a major issue in term of evaluation. At a high level, three main criteria are often used to evaluate these formations. The *autonomy* can be defined as the capability of a system to analyse, plan, make decisions, communicate, or achieve goals (a task, for example). The *robustness* of a system corresponds to its ability to achieve its goals when facing uncertainty and disturbances (noise, perturbations,...). The *stability* can be seen as a subcategory of the robustness which represents the ability of a system to maintain its behavior in face of disturbances. For example, a team of robots able to maintain their formation over time might be considered as stable. Finally, the *efficiency* of a system corresponds to its performances given some criteria like number of successful missions or the time to completion. Unfortunately these metrics are intrinsically task dependent, leading to domain dependent measurements and thus restraining the comparison opportunity. That is the main reason why common metrics are needed.

E.2.1 Common metrics

Even if the definition of a global, task independent metric remains a difficult problem, it has been shown ([91]) that common metrics can be constructed. Three main categories can be created depending on the point of view we want to consider: the system (humans and robots) as an entity, the robots alone or the humans alone.

System

The robots and the humans are evaluated as a team ([4]). The quantitative measurement of the team can be done using two main criteria: the *effectiveness*, which corresponds to the percentage of successful missions of the team, and the the *efficiency*, which measures the *time to completion*[1] of the task. These measures can be completed using *subjective* rating: the overall impression of "easiness" in the performing of the mission for example (no brusque interruption, no long immobilisation time,...). The use of *mixed initiative* can also be quantified: the percentage of request for assistance from the robots or from the humans, the interaction effort (mainly from the human to work with the robot, [193]), i.e. the right mix of competencies of the team members, and the correct leveling of the autonomy of the team mates.

Humans

To evaluate the performance of the operators in HRI, different factors can be taken into account. The *situation awareness* (SA) has been shown to be critical for decision making in dynamic systems management and highly related to the notion of *workload* (see [82] for example of tools to measure SA and [130, 226] for the implication of SA in the decision making process and workload evolution). The workload of the operators is almost always related to the need for tele-operation of the robots: most of the time the workload decreases with the need for tele-operation (see [78, 130, 225] for examples of workload measure). Finally, the *accuracy of mental models* often plays a major role in HRI. The main domain of application of this measurement is the *Search and Rescue* problem, in which a mixed team of humans and robots has to perform a rescue operation in a rugged terrain. It has been shown ([184]) that better representation of the environment by the humans can be achieved by cooperating with other fellow humans to explicit and improve the situation model.

Robots

The robots performance depends on multiple factors. The *self-awareness* of the robot, i.e. its ability to know its own capabilities can become crucial in many situations. A self-aware robot will be able to recognize the case in which a human is needed and, on the contrary, avoid the need for monitoring otherwise. This capability can be qualitatively measured using three main evaluators: the *intrinsic* limitation of the robots (at the level of the hardware), for example the type of sensors, the *self-monitoring* capacity (the ability to evaluate its current status and state), and finally the capacity to detect, recover and isolate fault.

[1] This measure can be further refined using other measurements

E.2.2 Coordination and cooperation

When we have to consider mixed team of robots and humans, the notions of *cooperation* and *coordination* have to be considered carefully.

Most of the time the cooperation in a team is measured using the *neglect tolerance* ([100, 193]) or the *fan-out* ([71]) criteria. The neglect tolerance (NT) corresponds to the maximum duration between two human interventions before the performance goes under a certain threshold (see figure E.1).

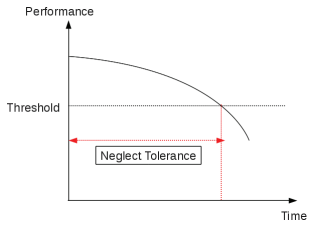
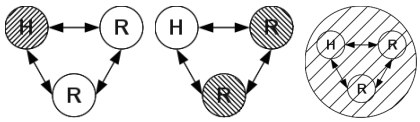


Figure E.1 – The neglect tolerance criterion

The fan-out (FO) estimates the number of robots that can be controlled by a human without a decrease of the global performance under a certain threshold.

The main issue with both FO and NT is the fact of considering team of homogeneous robots, which considerably limits the versatility of the resulting group. Nevertheless, the framework of NT can be extended ([274]) to take into account the heterogeneity among the robots and to measure the coordination demand in the team. The *cooperation effort* in the team depends not only on the capacities of both humans and robots but also on the global coordination capacity of the team (see figure E.2).



(a) For the humans (b) For the robots (c) For the team

Figure E.2 – Capacity in a mixed team: to evaluate the performance of a team, one has to take into account not only the capacities of the human (E.2a) or of the robots (E.2b) alone, but also the capacity of the team as a whole (E.2c) considering the additional abilities than can emerge from the cooperative and collaborative behaviors (double sided arrows).

The robot autonomy has a direct impact on the decision making load ([15]) of the team: in the case of multiple robots, the humans often have to *shift attention* from one robot to another one leading to a degradation of the situation awareness. The average demand on human attention for each particular robot can be measured ([71]): most of the time an increase in the autonomy of the robot leads to an increase in the duration of the neglect tolerance time.

A new criteria based on the NT can be introduced to better capture the coordination between the different team mates when strong coordination is needed: the *coordination demand* ([274]). Both the NT and the FO are not very well suited for tightly cooperative tasks because they are based on a sequential division of the control task. The coordination demand can be formally defined as

$$CD = \frac{\sum OT}{NT}$$

where OT corresponds to the time devoted by the human to synchronize (with) the robots. It measures the time devoted to cooperation during a given task. This criteria can be further extended to the case of sub-team inside a global group, where robots and humans can decide to form a subgroup to perform a given sub-task.

E.2.3 Taxonomy

Different taxonomies exist in HRI. We present three main categories which are particularly relevant to our problem (see [159] and references inside).

Autonomy level

we can differentiate between autonomy as the amount of time the robot carries a task independently from the *intervention time*, which is the percentage of time the operator needs to operate the robot.

Interaction ratio

it corresponds to the level of interaction between robots and humans (for example, one robot - one human, multiple robots - one human, human team - one robot,...).

Team composition

in this domain, different measures exist. The human-robot ratio gives an idea of the relative proportion of humans and robots in the team. The notions of *homogeneity* and *heterogeneity* are the basic considerations for this criterion.

To characterise the task, three main factors can be taken into account. The *criticality* measures the potential harm that can be caused to the humans, the robots or to the environment in case of failure. The *time* corresponds to the synchronous or asynchronous aspect of the interaction between humans and robots. The *space* is the indication of the physical proximity of humans and robots: do they perform their action in a co-located area or remotely.

E.2.4 Implied design

The previous indicators can greatly influence and shape the design of the control architecture and of the interface in the team.

For example, in the problem of search and rescue, it has been shown ([184]) that several guidelines must be followed when designing a HRI protocol. Firstly, the awareness must be enhanced by improving the quality of the spatial information given to the operators to allow a better grasp of the robots immediate surroundings. Secondly, the cognitive load of the humans need to be lowered, by fusing the different information before transmitting them to the users and only displaying important information. Thirdly, the use of multiple windows should be avoided and finally the system should assist humans in the choice of the right modalities to consider from the robots as well as to chose the appropriate autonomy level.

E.3 Implementation

In order to effectively design a control architecture for a mixed team of humans and robots, two main approaches can be considered: the centralised and decentralised one. Nevertheless, the former tends to suffer from major drawbacks: lack of scalability, multiple points of failure possible,... On the contrary, the use of a distributed system to abstractly represent a team of human and robots leads to robust task execution in dynamic environment, with no centralized bottlenecks or points of failure. We explain in the following subsections the use of the abstraction concept of *Multi-Agent Systems* (MAS) to achieve this decentralised control architecture and present examples of applications of teamwork frameworks.

E.3.1 Multi-agents systems

The integration of humans in robots teams make the classical approach, like the *Adaptive Agent Architecture* ([147]), difficult to use, not only because of the strongly heterogeneous aspect of the resulting groups but also considering the lack of teamwork implementation in this case. The notion of *team programming* ([261, 262]) has been developed to integrate team behavior into multiagent framework. The *SharedPlans* ([103]) and *joint intentions* ([65]) theory have been used along with the frameworks of *coordinate agent* ([125, 254, 255]) to extend and adapt the existing architecture to deal with heterogeneous teams of humans and robots ([214]). *Policy* based methods ([31]) are also used, as they allow the regulation of dynamic system of heterogeneous entities without requiring cooperation ability between the group members. One of the most promising approach in the *MAS* domain is the work done by Tambe and Pynadath in the developement of the *Teamcore* architecture ([214]). In this framework, the major improvement over classical approaches ([219, 125, 127, 232, 17]) is the use of proxies to access the different agents. It allows a reuse of the same rules in the team even if the members change, as it considers the different agents as black boxes. The main challenges are to manage the *role* between the heterogeneous agents and to coordinate the level of planning needed

(at the level of the agent and at the level of the team). Most of the time, an agent *coordinator* is used to handle this task. The role of coordinator can be shifted from on agent to an other one. In this context, an agent can be viewed as a proxy with team work capability (using the *STEAM* teamwork model, [254]) and adjustable autonomy (due to the team work capacity, the agent can defer some decision making to other agents depending in the circumstances). Thus the architecture itself, through the proxies, can adapt to the needs and performances of the agents (for example, a difference in response time). More globally, *dynamic plan alteration* are made possible by combining the adaptive properties of the agents. Finally, the possibility of reusing specialized proxies as building blocks of the application ([106, 117, 126]) make the efficiency of a newly created team much higher.

E.3.2 Examples

The *Teamcore* ([214]) architecture has been applied in various domains. In simulation, an evacuation rehearsal has been represented where the different agents were in charge of the control of the helicopters for the evacuation of the civilians, planning the route (avoiding the obstacles but also locate possible enemy threats) and reacting to the commander inputs. This experiment was useful to illustrate the power of the proxy abstraction layer considering the heterogeneity of the agents in terms of architecture and code (all of them ran on different architectures and were coded in different languages): without any modification of the agents, *Teamcore* was able to successfully achieve the mission. This architecture has also been tested in a real environment with a team of humans. It was in charge of organizing the meetings of a team of researchers (planning the meetings, informing of possible delay,...). In this example the challenge was to take into account the different role played by the lab members, their heterogeneity and their numbers (the experiment has been performed in a large team). Moreover the group was also composed of multiple subgroups and the coordination of the common tasks was also needed. This example has inspired other possible applications in different domains (e.g. *Electric Elves*, [47]).

The notion of policy has been successfully implemented in the *KAoS* architecture ([215]). This framework is compatible with many mobile agent environment platforms and allows, in one of its extension called *Kaa* ([35]), the use of adjustable autonomy and policy learning. Concrete examples of application of policies to the robotic field can be found in the domain of space exploration (the NASA personal satellite assistant,[36], or the simulation of space exploration, [230]) and search tasks with joint activity constraints ([33]). Some metrics have been proposed in these cases to evaluate the policies: the *survivability* (ability to maintain effectiveness when facing unforeseen events), the *predictability* (correlation between human judgment of predicted behaviors in comparison with actual behaviors) or *safety* (capacity of preventing certain classes of dangerous actions or situations) for example.

E.4 Autonomy and awareness

The notion of autonomy (subsection E.4.1) is often used to classify robots and, most of the time, different types of scales can be used. The *awareness* (subsection E.4.2) concept also plays a crucial role in HRI. Finally, being able to adapt the design of the implementation by using the intrinsic requirements of the joint activity is the purpose of a new approach called *co-active design*, that we present in subsection E.4.3.

E.4.1 Autonomy

The *autonomy* of robots can be measured in various ways. The example of guidance in an unknown environment illustrates quite well the different levels of autonomy. The first level is the *tele-operation* level ([201]): the robot has no autonomy and the operator has to fully control it (speed of the wheel, orientation of the camera,...) to complete the task. The second level, called *waypoint control* ([71, 190, 201, 267]), consists in giving to the robot only positions in space where it has to go (like checkpoints) and let it manage its behavior in between. The autonomy in this case can also be compared to the one corresponding to the *prescribed behavior* strategy ([184]): the human select a type of behavior for the robots according the environment and to the situation. If we increase the level of autonomy of the robots and allow for different level of individual autonomy, it has been shown that the efficiency of the team increases only if the robots are able to cooperate ([274]), even if the workload on the humans was inferior. This classification can be qualified of "human centered" because we mainly consider the autonomy of the robots in terms of differences for the humans operator or team mates.

Another way of comparing the autonomy in a more "team centered" manner is as follows ([169]). The tele-operation remains the first level of autonomy. The second one is called *safe mode*: the robot has the authority to protect itself from the environment if the operator commands are evaluated as dangerous. In the next level, or *shared control* level, the robot can choose its own path in response to the global direction of the human. Finally, the last level is the *full autonomy* one: the robot react to high level input ("Go search this area" for example) without the need to be operated by a human at any moment.

Systems in which the robots autonomy can be dynamically changed to adapt to the situation have also been studied ([169]). The notion of *mixed-initiative* is used in this case: the autonomy can be shifted between humans and robots depending on the environment. The robots are always responsible for the low level task and the human of defining the high level goals, but the robots behavior can be overridden if the humans *infer* some possible outcomes in the environment. A similar idea supports the concept of *adjustable* autonomy ([34]). It is based on an optimal allocation of the task based of the capabilities of the agents and of the humans. This allocation can also be made adjustable depending on the context.

E.4.2 Awareness

The concept of awareness plays a crucial role in HRI. It has been first defined in [80] for two entities collaborating synchronously on a given task: the awareness is the understanding that one of the entity has of the presence, identity and activities of the other. However in HRI, the situations might differ from this one because interactions can take place between several robots and humans. Moreover the relationship between humans and robots is not symmetrical because of the intrinsic limitations of robots in terms of free will, cognitive skills and autonomy. The awareness can be decomposed into five different categories depending of the perspective chosen. The *human-robot* awareness is the understanding the humans have of the environment, status, identity and activities of the robots, as well as the certainty of this understanding. The *human-human* awareness is similar to the previous one except we consider human-human interaction. The *robot-human* awareness corresponds to the robots' understanding or knowledge about the human that is needed to understand the commands given and to shape the activity depending on human need or status. The *robot-robot* awareness take into account the possible command one robot can be given by other robot and the collaboration/coordination plans needed to dynamically reaffect task among the robots. The human or robot *overall* awareness of the mission is the last type of awareness we can consider. It corresponds to the understanding of the goals of the joints activities but also the ability of measuring the *progress* towards these goals.

E.4.3 Co-active design

The previous approaches can be seen as *autonomy centered* in the sense that the main concern for the design of the system is either to compensate the low capabilities of the robots using tele-operation or to reduce the humans work load by increasing the autonomy of the robots. Some authors ([128]) have adopted a more *team-work centered* approach ([32]). Key concepts in this method are the notions of *group participatory action* and *interdependency*: instead of seeing the global goal as a sequence of individual tasks performed by the different actors, a global activity is considered. In this activity, *collective obligations* ([87]) emerge of the needed joint actions. This leads to design requirements which are going to shape the implementation of the robots proportionally to the interdependence needed in the joint activity. For example, if the users inputs are needed to improve the navigation task, then the corresponding algorithm should be able to incorporate them.

E.5 Social interaction

The relationships between human and robots have always been difficult to define and to analyse. Robots can be considered as true members of a team ([169]) or only as active information sources ([184]). The humans might need to be willing to accept robots initiatives and "trust" the system to ensure true integration of the robots inside the team.

E.5.1 Roles

Different roles can be played by the humans in an interaction with a robot. They can act as a *supervisor* and be in charge of one or more robots. She need then to have a global understanding of the mission. In the *operator* role, the interactions will depend on the level of autonomy defined. The human need to be aware of the status of the robot as well as its surroundings. Finally, as a *team-mate*, the human need to understand the restriction in the robots capabilities to be able to interact optimally with it.

In an interaction, a human migh assume multiple roles for a given robot. Several people can also interact with a robot and play different roles. The type of interaction and the role played by the humans in it can determine or model the design of the interface between humans and robots.

In human teams, the roles of the different team mates evolve depending on the information obtained or elicited about the different members (capacities, behaviors,..). The direct application of *role theory* to HRI has shown ([39]) that robots often have the role of tools in a team and scarcely the one of peers, and even less often the one of leaders. Nevertheless, robots capabilities can exceed those of a human in some cases (mainly for low level functions). The main limitation of the robot comes from its difficulty to recover from failure and to adapt to changes in the environment. *Role shifting* is useful in this case to compensate these limitations or failures and ensure a more optimal modulation of the level of autonomy in terms of robots initiative.

A robot should also be *human-aware* to improve its reactivity and its performance in the team. This aspect will be dependent on both the autonomy of the robots and the role played by the humans ([224]). The robots will be able to construct a “user model” to tune its behavior according to the humans it interacts with (mainly by using monitoring).

E.5.2 Trust and social behavior

Trust can be defined as the disposition to firmly rely on a person or thing ([35]). It is based on a judgement of competence, benevolence and compliance. Different studies have shown ([169]) that a key concept to ensure trust between team members is a *basic understanding* of the action of the others members (the incomprehension leading to frustration). In HRI the humans need to be able to understand and predict the robot responses to accept it as a team member. If this aspect is important for the human, it can also be relevant and helpful for robots, as they will be able to anticipate and better respond to human behavior or needs. For example the robots could detect the level of stress or workload of the user and adjust accordingly their level of autonomy (for a more complete study of the change in robots behavior in response to human state, see [216, 217]). It has also been shown that it is possible to ensure comparative results in a task between expert and novice humans by allowing the robot to adjust its level of autonomy.

E.5.3 Fluency in the task

Some authors ([113]) have imagined to give to the robot a *cognitive* architecture based on *anticipation* and *perceptual simulation* to improve the interaction between humans and robots. It allows the robots to adapt dynamically to its human "colleagues". It has been shown that the more the robot is able to anticipate the human needs the more the human expects a full coordination with the robot ([112]). In order to decrease the reaction time, anticipatory simulation (a predicted response of the human counterpart is used as a simulated reply) and Hebbian inter-modal reinforcement learning (a neural network using the Hebbian's rule) are used. This new architecture leads to better results in all tests performed. The learning curve of both the humans and the robots is similar. The notion of *fluency* captures the high level of coordination and adaptation of agents who perform a joint action. In all the tests performed, the level of fluency was higher for the team in which the robots were driven by the previous cognitive architecture.

E.5.4 Anthropomorphic behavior

The way humans consider the robots is highly dependant on the robots capabilities (in terms of adaptation, evolution but also considering the available means of communication with it) and less on their appearance. For example, it has been observed in a search and rescue task that humans were prone to adopt anthropomorphic behavior ([184]) even with non human like robot: they make eye contacts with the robot, try to incite him to follow them using gesture and also maintain personal space etiquette. In the case of the previous cognitive architecture, the human interacting with the robot tends to give it human attributes (sex, eye,..) and to have a more self-deprecated attitude ([113]).

E.5.5 Computational Cognitive models:

Some works ([264, 265]) have been done in the field of computational cognitive models applied to HRI. The main hypothesis is that the fact of having a basic and identical representation and reasoning mechanism will lead to a better collaboration. To a certain extent, we could say that a system should be able to act "naturally" in order to improve the "compatibility" with the humans. As a consequence the robots should accommodate to their human counterparts in such a way that the team can exploit optimally the capabilty of the two "worlds". The main "cognitive skills" that have to be mastered by the system are of different orders. Firstly, the robots have to appropriate the knowledge representations characteristic of the problem. For example, if we consider a guiding problem on a map, the spatial representation will induce a spatial reasoning. The representation of the problem will have to be adapted to the method used for solving it. Secondly, the system should be able to learn to recognize and anticipate its team-mates behaviors, as well as to elicit and determine their different capabilities to better react and adapt to the situation. Finally some features have to be mastered in specific domains: the permanence and tracking of objects [234] and the gestures recognition [206]

for example. More high level considerations can be taken into account. Humans are able to switch between different perspective depending on the situation (from spatial to social for example,[264, 265, 38]). Our capacities of anticipation and temporal reasoning also play a crucial role in our ability to perform a given task. An interesting experiment that requires the use of several of these skills is the problem of Hide and Seek [109, 266]. This kind of experiment allows to develop computational cognitive models of high-level human cognitive skills that will be used as reasoning mechanisms for robots.

E.6 Conclusion

The overview given in this chapter illustrates both the complexity of the HRI field and the promising perspectives it can offer in the domain of coordination of mixed teams. We have seen that different metrics exist to measure the efficiency of a coordination task and that the implementation of this collaborative aspect into a complete architecture can be made possible by using an abstract representation based on a multi-agent system. We have also investigated the notions of autonomy and awareness, showing that both are fundamental in this domain, but also that the emergent aspect of co-active design can influence the implementation of the control architecture and of the robots in the early stages of the system conception. Finally, we have described the social aspects related to this problem, by pointing out that the more robots capacities are similar to human ones or accordingly adapted, the more robots will be considered as team-mates and not as simple tools in the group.

Bibliography

[1] Manipulation of a passive elements using two roombots metamodules. http://biorob2.epfl.ch/utils/movieplayer.php?id=278.

[2] Roombots website, www.roombots.org, 2011.

[3] Aaron Abrams and Robert Ghrist. State complexes for metamorphic robots. *The International Journal of Robotics Research*, 23(7-8):811–826, 2004.

[4] AIAA. Guide to human performance measurements, 1993.

[5] Greg Aloupis, Nadia Benbernou, Mirela Damian, Erik D Demaine, Robin Y Flatland, John Iacono, and Stefanie Wuhrer. Efficient reconfiguration of lattice-based modular robots. In *ECMR*, pages 81–86. Citeseer, 2009.

[6] Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhrer. Linear reconfiguration of cube-style modular robots. *Computational geometry*, 42(6):652–663, 2009.

[7] Greg Aloupis, Sébastien Collette, Erik D Demaine, Stefan Langerman, Vera Sacristán, and Stefanie Wuhrer. Reconfiguration of cube-style modular robots using o (logn) parallel moves. In *Algorithms and Computation*, pages 342–353. Springer, 2008.

[8] M. Andel, A. Petrovski, A. Henrysson, and M. Ollila. Interactive collaborative scene assembly using ar on mobile phones. *Advances in Artificial Reality and Tele-Existence*, pages 1008–1017, 2006.

[9] Masoud Asadpour, M Ashtiani, Alexander Sproewitz, and Auke Ijspeert. Graph signature for self-reconfiguration planning of modules with symmetry. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5295–5300. IEEE, 2009.

[10] Masoud Asadpour, Alexander Sproewitz, Aude Billard, Pierre Dillenbourg, and Auke Jan Ijspeert. Graph signature for self-reconfiguration planning. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 863–869. IEEE, 2008.

[11] R.T. Azuma et al. A survey of augmented reality. *Presence-Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[12] José Baca, Ariadna Yerpes, Manuel Ferre, Juan A Escalera, and Rafael Aracil. Modelling of modular robot configurations using graph theory. In *Hybrid Artificial Intelligence Systems*, pages 649–656. Springer, 2008.

[13] J. Bachrach, V. Zykov, and S. Griffith. Folding Arbitrary 3D Shapes with Space-Filling Chain Robots: Folded Configuration Design as 3D Hamiltonian Path through Target Solid.

[14] M. Bajura, H. Fuchs, and R. Ohbuchi. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 203–210. ACM, 1992.

[15] S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V.A. Ziparo. Towards heterogeneous robot teams for disaster mitigation: Results and Performance Metrics from RoboCup Rescue. *Journal of Field Robotics*, 24(11):943–968, 2007.

[16] Ashis Gopal Banerjee, Sagar Chowdhury, Wolfgang Losert, and Satyandra K Gupta. Real-time path planning for coordinated transport of multiple particles using optical tweezers. *Automation Science and Engineering, IEEE Transactions on*, 9(4):669–678, 2012.

[17] M. Barbuceanu and M. Fox. The architecture of an agent building shell. *Intelligent Agents II Agent Theories, Architectures, and Languages*, pages 235–250, 1995.

[18] G Beni. Research perspectives in swarm intelligence the reconfiguration problem. In *Proc. Int. Symp. Syst. Life*, pages 51–59, 1997.

[19] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[20] M. Billinghurst, H. Kato, and S. Myojin. Advanced interaction techniques for augmented reality applications. *Virtual and Mixed Reality*, pages 13–22, 2009.

[21] Biorob. Biorobotics laboratory. http://biorob.epfl.ch/.

[22] J. Blatter. Mobile control interface for modular robots. Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2011.

[23] 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*, volume 2, pages 1734–1741. IEEE, 2002.

[24] H. Bojinov, A. Casal, and T. Hogg. Multiagent control of self-reconfigurable robots. *Artificial Intelligence*, 142(2):99–120, 2002.

[25] Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, pages 1170–1182, 1987.

[26] Stéphane Bonardi. Complexity reduction in optimization of modular robots locomotion using body/limbs recognition and spatial symmetries. Project report, June 2010.

[27] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 6. Macmillan London, 1976.

[28] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[29] Stephen P Borgatti and Martin G Everett. A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484, 2006.

[30] S.M.B.I. Botden, S.N. Buzink, M.P. Schijven, and J.J. Jakimowicz. Augmented versus virtual reality laparoscopic simulation: What is the difference? *World journal of surgery*, 31(4):764–772, 2007.

[31] J. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, et al. Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, page 842. ACM, 2003.

[32] J.M. Bradshaw, A. Acquisti, J. Allen, MR Breedy, L. Bunch, N. Chambers, P. Feltovich, L. Galescu, M.A. Goodrich, R. Jeffers, et al. Teamwork-centered autonomy for extended human-agent interaction in space applications. In *AAAI 2004 Spring Symposium*, pages 22–24, 2004.

[33] JM Bradshaw, P. Feltovich, M. Johnson, L. Bunch, MR Breedy, H. Jung, J. Lott, and A. Uszok. Coordination in Human-Agent Teamwork. Invited Paper and Presentation. In *AAAI Fall Symposium*, pages 8–10, 2007.

[34] J.M. Bradshaw, PJ. Feltovich, H. Jung, S. Kulkarni, W. Taysom, and A. Uszok. Dimensions of adjustable autonomy and mixed-initiative interaction. *Agents and Computational Autonomy*, pages 17–39, 2003.

[35] J.M. Bradshaw, H. Jung, S. Kulkarni, M. Johnson, P. Feltovich, J. Allen, L. Bunch, N. Chambers, L. Galescu, R. Jeffers, et al. Toward trustworthy adjustable autonomy in KAoS. *Trusting Agents for Trusting Electronic Societies*, pages 18–42, 2005.

[36] J.M. Bradshaw, M. Sierhuis, Y. Gawdiak, R. Jeffers, N. Suri, and M. Greaves. Adjustable autonomy and teamwork for the Personal Satellite Assistant. In *Proceedings of IJCAI-01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*. Citeseer, 2001.

[37] G. Bradski. *Dr. Dobb's Journal of Software Tools*.

[38] C. Breazeal, M. Berlin, A. Brooks, J. Gray, and A.L. Thomaz. Using perspective taking to learn from ambiguous demonstrations. *Robotics and Autonomous Systems*, 54(5):385–393, 2006.

[39] D.J. Bruemmer, J.L. Marble, and D.D. Dudenhoeffer. Mutual initiative in human-machine teams. In *IEEE Conference on Human Factors and Power Plants*, volume 7, pages 22–7, 2002.

[40] L. Brunetta and P. Gregoire. A general purpose algorithm for three-dimensional packing. *INFORMS Journal on Computing*, 17(3):328, 2005.

[41] Don Burns and Robert Osfield. Open scene graph. In *Proceedings of the IEEE Virtual Reality*, page 265, 2004.

[42] Don Burns and Robert Osfield. Open scene graph a: Introduction, b: Examples and applications. In *Proceedings of the IEEE Virtual Reality 2004*, VR '04, pages 265–, Washington, DC, USA, 2004. IEEE Computer Society.

[43] Nicolas Burrus. http://labs.manctl.com/rgbdemo/index.php. Kinect toolkit.

[44] J Capitan, L Merino, and A Ollero. Multirobot coordinated decision making under mixed observability through decentralized data fusion. In *Proceedings of the 11th International Conference on Mobile Robots and Competitions (Robotica 2011)*, 2011.

[45] Andres Castano and Peter Will. Representing and discovering the configuration of conro robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3503–3509. IEEE, 2001.

[46] R. O. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Proc 12th IEEE Int Symp on Wearable Computers, Pittsburgh PA, Sept 28 - Oct 1, 2008*, pages 15–22, 2008.

[47] H. Chalupsky, Y. Gil, C.A. Knoblock, K. Lerman, J. Oh, D.V. Pynadath, T.A. Russ, and M. Tambe. Electric Elves: Agent technology for supporting human organizations. *AI Magazine*, 23(2):11, 2002.

[48] Zong-yu CHANG, Xian-qi YANG, and Jun-zhe TAN. Application of exponential product method in spatial movement analysis of mechanism [j]. *Machine Design*, 7:008, 2002.

[49] I-M Chen and Yan Gao. Closed-form inverse kinematics solver for reconfigurable robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2395–2400. IEEE, 2001.

[50] I-Ming Chen and Joel W Burdick. Enumerating the non-isomorphic assembly configurations of modular robotic systems. *The International Journal of Robotics Research*, 17(7):702–719, 1998.

[51] I-Ming Chen, Guilin Yang, In-Gyu Kang, et al. Numerical inverse kinematics for modular reconfigurable robots. *Journal of Robotic Systems*, 16(4):213–225, 1999.

[52] K.Y. Cheng, Y.H. Lin, Y.H. Lin, B.Y. Chen, and T. Igarashi. Grab-carry-release: manipulating physical objects in a real scene through a smart phone. In *SIGGRAPH Asia 2011 Emerging Technologies*, page 13. ACM, 2011.

[53] C.J. Chiang and G.S. Chirikjian. Modular robot motion planning using similarity metrics. *Autonomous Robots*, 10(1):91–106, 2001.

[54] Gregory Chirikjian, Amit Pamecha, and Imme Ebert-Uphoff. Evaluating efficiency of self-reconfiguration in a class of modular robots. *Journal of robotic systems*, 13(5):317–338, 1996.

[55] Gregory S Chirikjian. Metamorphic hyper-redundant manipulators. In *Proc. of Intl. Conf. on Advanced Mechatronics*, pages 467–472, 1993.

[56] G.S. Chirikjian. Kinematics of a metamorphic robotic system. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, page 449–455, 1994.

[57] David Johan Christensen. Experiments on fault-tolerant self-reconfiguration and emergent self-repair. In *Artificial Life, 2007. ALIFE'07. IEEE Symposium on*, pages 355–361. IEEE, 2007.

[58] David Johan Christensen, Jason Campbell, and Kasper Stoy. Anatomy-based organization of morphology and control in self-reconfigurable modular robots. *Neural Computing and Applications*, 19(6):787–805, 2010.

[59] David Johan Christensen, Jørgen Christian Larsen, and Kasper Stoy. Adaptive strategy for online gait learning evaluated on the polymorphic robotic locokit. In *Evolving and Adaptive Intelligent Systems (EAIS), 2012 IEEE Conference on*, pages 63–68. IEEE, 2012.

[60] David Johan Christensen, Esben Hallundbok Ostergaard, and Henrik Hautop Lund. Metamodule control for the atron self-reconfigurable robotic system. In *Proceedings of the The 8th Conference on Intelligent Autonomous Systems (IAS-8)*, pages 685–692. Citeseer, 2004.

[61] David Johan Christensen, Ulrik Pagh Schultz, and Kasper Stoy. A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots. *Robotics and Autonomous Systems*, 2013.

[62] David Johan Christensen, Alexander Spröwitz, and Auke Jan Ijspeert. Distributed online learning of central pattern generators in modular robots. In *From Animals to Animats 11*, pages 402–412. Springer, 2010.

[63] Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002.

[64] Benjamin Cohen, Mike Phillips, and Maxim Likhachev. Planning single-arm manipulations with n-arm robots. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[65] P.R. Cohen and H.J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[66] OSG Community. Openscenegraph. http://www.openscenegraph.org/projects/osg. [Online; accessed 17-January-2011].

[67] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[68] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159, 2001.

[69] Nokia Corporation. Qt. http://qt.nokia.com/products/. [Online; accessed 17-January-2011].

[70] Erwin Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 2006.

[71] JW Crandall, MA Goodrich, DR Olsen Jr, and CW Nielsen. Validating human-robot interaction schemes in multitasking environments. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(4):438–449, 2005.

[72] Alessandro Crespi and Auke Jan Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *Robotics, IEEE Transactions on*, 24(1):75–87, 2008.

[73] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. http://ompl.kavrakilab.org.

[74] Jay Davey, Ngai Kwok, and Mark Yim. Emulating self-reconfigurable robots-design of the smores system. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4464–4469. IEEE, 2012.

[75] Michael De Rosa, Seth Goldstein, Peter Lee, Jason Campbell, and Padmanabhan Pillai. Scalable shape sculpting via hole motion: Motion planning in lattice-constrained modular robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1462–1468. IEEE, 2006.

[76] D.J. Dewey, M.P. Ashley-Rollman, M. De Rosa, S.C. Goldstein, T.C. Mowry, S.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*, pages 1338–1345. IEEE, 2008.

[77] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[78] JV Draper and LM Blair. Workload, flow, and telepresence during teleoperation. In *1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings.*, volume 2, 1996.

[79] D. Drascic and P. Milgram. Perceptual issues in augmented reality. In *Proceedings of SPIE*, volume 2653, page 123, 1996.

[80] J. Drury. Developing heuristics for synchronous collaborative systems. In *CHI'01 extended abstracts on Human factors in computing systems*, page 448. ACM, 2001.

[81] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

[82] M.R. Endsley. Measurement of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):65–84, 1995.

[83] Ernesto Estrada and Örjan Bodin. Using network centrality measures to manage landscape connectivity. *Ecological Applications*, 18(7):1810–1825, 2008.

[84] Ernesto Estrada and Juan A Rodriguez-Velazquez. Subgraph centrality in complex networks. *Physical Review E*, 71(5):056103, 2005.

[85] J.berriault et *al.* V.U.G Project - Virtual Universe Generator. http://code.google.com/p/vug/.

[86] Martin Felis. Rigid body dynamics library. http://rbdl.bitbucket.org/index.html.

[87] P. Feltovich, J. Bradshaw, W. Clancey, and M. Johnson. Toward an ontology of regulation: Socially-based support for coordination in human and machine joint activity. *Engineering Societies in the Agents World VII*, pages 175–192.

[88] R. Fitch and Z. Butler. Million module march: Scalable locomotion for large self-reconfiguring robots. *The International Journal of Robotics Research*, 27(3-4):331, 2008.

[89] R. Fitch and R. McAllister. Hierarchical Planning for Self-Reconfiguring Robots Using Module Kinematics. 2010.

[90] Robert Fitch and Rowan McAllister. Hierarchical planning for self-reconfiguring robots using module kinematics. In *Distributed Autonomous Robotic Systems*, pages 477–490. Springer, 2013.

[91] T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and A. Steinfeld. Common metrics for human-robot interaction. In *IEEE 2004 International Conference on Intelligent Robots and Systems, Sendai, Japan*. Citeseer, 2004.

[92] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[93] F Freudenstein and L Dobrjanskyj. On a theory for the type synthesis of mechanisms. In *Applied Mechanics*, pages 420–428. Springer, 1966.

[94] Kevin C Galloway, Rekha Jois, and Mark Yim. Factory floor: A robotically reconfigurable construction platform. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2467–2472. IEEE, 2010.

[95] Santiago Garrido, Luis Moreno, Dolores Blanco, and Piotr Jurewicz. Path planning for mobile robot navigation using voronoi diagram and fast marching. *International Journal of Robotics and Automation (IJRA)*, 2(1):42–64, 2011.

[96] Sébastien Gay. Engineer Diploma Project : IMOROD : Interface for MOdular RObots Design. Conception and Development of a 3D Robotic Interface for the Roombots project. http://birg.epfl.ch/page66475.html.

[97] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[98] K. Golestan, M. Asadpour, and H. Moradi. A New Graph Signature Calculation Method Based on Power Centrality for Modular Robots.

[99] Keyvan Golestan, Masoud Asadpour, and Hadi Moradi. A new graph signature calculation method based on power centrality for modular robots. In *Distributed Autonomous Robotic Systems*, pages 505–516. Springer, 2013.

[100] M. Goodrich and D. Olsen. Seven principles of efficient human robot interaction. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3943–3948, 2003.

[101] Jonathan L Gross and Jay Yellen. *Graph theory and its applications*. CRC press, 2005.

[102] Roderich Groß, Elio Tuci, Marco Dorigo, Michael Bonani, and Francesco Mondada. Object transport by modular robots that self-assemble. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2558–2564. IEEE, 2006.

[103] B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[104] Gregory J Hamlin and Arthur C Sanderson. Tetrobot modular robotics: Prototype and experiments. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 2, pages 390–395. IEEE, 1996.

[105] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybernetics*, 4(2):100–107, 1968.

[106] J. Hendler and R. Metzeger. Putting it all together–the control of agent-based systems program. *IEEE Intelligent Systems and their applications*, 14, 1999.

[107] A. Henrysson, M. Billinghurst, and M. Ollila. Face to face collaborative ar on mobile phones. In *Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on*, pages 80–89. IEEE, 2005.

[108] A. Henrysson, M. Billinghurst, and M. Ollila. Virtual object manipulation using a mobile phone. In *Proceedings of the 2005 international conference on Augmented tele-existence*, pages 164–171. ACM, 2005.

[109] L.M. Hiatt, J.G. Trafton, AM Harrison, and A.C. Schultz. A cognitive model for spatial perspective taking. In *Proc. 6th International Conference on Cognitive Modelling*. Citeseer, 2004.

[110] Masato Hirose. Development of humanoid robot asimo. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems (Oct. 29, 2001)*, 2001.

[111] David Hjelle and Hod Lipson. A robotically reconfigurable truss. In *Reconfigurable Mechanisms and Robots, 2009. ReMAR 2009. ASME/IFToMM International Conference on*, pages 73–78. IEEE, 2009.

[112] G. Hoffman. Ensemble: fluency and embodiment for robots acting with humans. 2007.

[113] G. Hoffman and C. Breazeal. Effects of anticipatory perceptual simulation on practiced human-robot tasks. *Autonomous Robots*, pages 1–21, 2009.

[114] John E Hopcroft, Jacob T Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem". *The International Journal of Robotics Research*, 3(4):76–88, 1984.

[115] F. Hou and W.M. Shen. On the complexity of optimal reconfiguration planning for modular reconfigurable robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2791–2796. IEEE, 2010.

[116] Feili Hou and Wei-Min Shen. Graph-based optimal reconfiguration planning for self-reconfigurable robots. *Robotics and Autonomous Systems*, 2013.

[117] MN Huhns and MP Singh. All agents are not created equal. *IEEE Internet Computing*, 2(3):94–96, 1998.

[118] J. Hwang, J. Jung, and G.J. Kim. Hand-held virtual reality: a feasibility study. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 356–363. ACM, 2006.

[119] Yoshiaki Ichikawa and Fumiyasu Okido. One-dimensional self-reproducing robot. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*, pages 963–966. IEEE, 1991.

[120] The igraph Project. The igraph library. http://igraph.sourceforge.net/.

[121] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

[122] Freudenthal Institute. Building houses 1. http://www.mathsnet.net/geometry/solid/houses1.html, 2009.

[123] iRobot. http://www.irobot.com/us/. Robots designer and builder.

[124] James Jessiman. LDraw, 2014.

[125] NR Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.

[126] N.R. Jennings. Agent-based computing: Promise and perils. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1429–1436. Citeseer, 1999.

[127] NR Jennings, TJ Norman, and P. Faratin. ADEPT: An agent-based approach to business process management. *ACM Sigmod Record*, 27(4):39, 1998.

[128] M. Johnson, C. Jonker, and M. Sierhuis. Coactive Design. In *Proceeding of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 49–56. AAMAS, 2010.

[129] C. Jones and M.J. Mataric. From local to global behavior in intelligent self-assembly. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 721–726. IEEE, 2003.

[130] D.B. Kaber, E. Onal, and M.R. Endsley. Design of automation for telerobots and the effect on performance, operator situation awareness, and subjective workload. *Human Factors and Ergonomics in Manufacturing*, 10(4):409–430, 2000.

[131] W Kahan. Lectures on computational aspects of geometry. department of electrical engineering and computer sciences. *University of California, Berkeley. Unpublished*, 1983.

[132] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji. Distributed adaptive locomotion by a modular robotic system, M-TRAN II. In *IROS*, pages 2370–2377. 2004.

[133] Akiya Kamimura, Haruhisa Kurokawa, E Toshida, Kohji Tomita, Satoshi Murata, and Shigeru Kokaji. Automatic locomotion pattern generation for modular robots. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 714–720. IEEE, 2003.

[134] H. Kato, M. Billinghurst, I. Poupyrev, N. Tetsutani, and K. Tachibana. Tangible augmented reality for human computer interaction. *The Journal of the Society for Art and Science*, 1(2):97–104, 2002.

[135] Laura Kelmar and Pradeep K Khosla. Automatic generation of kinematics for a reconfigurable modular manipulator system. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 663–668. IEEE, 1988.

[136] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[137] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[138] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 83–86. IEEE, 2009.

[139] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity*. Birkhauser Verlag, 1994.

[140] S. Koenig, M. Likhachev, and D. Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1-2):93–146, 2004.

[141] Shigeru Kokaji. A fractal mechanism and a decentralized control method. In *Proc. USA-Japan Symp. Flexible Automation*, pages 1129–1134, 1988.

[142] Minxiu Kong, Zhijiang Du, Lining Sun, and Yong Zhang. Solution and application of two inverse kinematics subproblems. In *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, pages 1164–1168. IEEE, 2006.

[143] Keith Kotay, Daniela Rus, Marsette Vona, and Craig McGray. The self-reconfiguring robotic molecule: Design and control algorithms. In *Workshop on Algorithmic Foundations of Robotics*, pages 376–386. Citeseer, 1998.

[144] Keith D Kotay and Daniela L Rus. Algorithms for self-reconfiguring molecule motion planning. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 2184–2193. IEEE, 2000.

[145] Hideki Kozima, Marek P Michalowski, and Cocoro Nakagawa. Keepon. *International Journal of Social Robotics*, 1(1):3–18, 2009.

[146] James J Kuffner Jr and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

[147] S. Kumar, PR Cohen, and HJ Levesque. The adaptive agent architecture: Achieving fault-tolerance usingpersistent broker teams. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 159–166, 2000.

[148] Haruhisa Kurokawa, Kohji Tomita, Akiya Kamimura, Shigeru Kokaji, Takashi Hasuo, and Satoshi Murata. Self-reconfigurable modular robot m-tran: distributed control and communication. In *Proceedings of the 1st international conference on Robot communication and coordination*, page 21. IEEE Press, 2007.

[149] Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, Akiya Kamimura, Satoshi Murata, and Shigeru Kokaji. Self-reconfigurable m-tran structures and walker generation. *Robotics and Autonomous Systems*, 54(2):142–149, 2006.

[150] 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.

[151] A. J. D. Lambert. Disassembly sequencing: A survey. *International Journal of Production Research*, 41(16):3721–3759, January 2003.

[152] Christopher G Langton. Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):135–144, 1984.

[153] Christopher G Langton and Katsunori Shimohara. *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, volume 5. MIT Press, 1997.

[154] L. Laperrière and H. A. ElMaraghy. Automatic generation of robotic assembly sequences. *The International Journal of Advanced Manufacturing Technology*, 6(4):299–316, November 1991.

[155] Jean-Claude Latombe. Robot motion planning, chapter. 1996.

[156] Steven M LaValle. Rapidly-exploring random trees a new tool for path planning. 1998.

[157] Steven Michael LaValle. *Planning algorithms*. Cambridge university press, 2006.

[158] LEGO. Lego digital designer. http://ldd.lego.com/, 2009.

[159] J. Leitner. Literature Review Multi-Robot Cooperation in Space Applications. 2009.

[160] Lemon Graph Library. Department of Operations Research, Eötvös Loránd University, Budapest, Hungary. http://lemon.cs.elte.hu/trac/lemon.

[161] A. Lim, B. Rodrigues, and Y. Wang. A multi-faced buildup algorithm for three-dimensional packing problems. *Omega*, 31(6):471–481, 2003.

[162] Aristid Lindenmayer and Grzegorz Rozenberg. Automata, languages, development. 1976.

[163] Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction with quadrotor teams. *Autonomous Robots*, 33(3):323–336, 2012.

[164] Hod Lipson and Jordan B Pollack. Towards continuously reconfigurable self-designing robotics. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1761–1766. IEEE, 2000.

[165] M.L. Littman, T.L. Dean, and L.P. Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 394–402. Citeseer, 1995.

[166] D. Mackay. Path planning with d*-lite. *DRDC Suffield TM*, 242, 2005.

[167] Daniel Marbach and Auke Jan Ijspeert. Co-evolution of configuration and control for homogenous modular robots. In *Proceedings of the eighth conference on intelligent autonomous systems (IAS8)*, pages 712–719, 2004.

[168] Daniel Marbach and Auke Jan Ijspeert. Online optimization of modular robot locomotion. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 1, pages 248–253. IEEE, 2005.

[169] J.L. Marble, D.J. Bruemmer, D.A. Few, and D.D. Dudenhoeffer. Evaluation of supervisory vs. peer-peer interaction with human-robot teams. In *Proceedings of the 37th Annual Hawaii International Conference on Systems Sciences*, pages 4–8. Citeseer, 2004.

[170] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[171] S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, pages 256–267, 2000.

[172] James J McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice and Experience*, 12(1):23–34, 1982.

[173] H. Medellin, J. Corney, J. Ritchie, and T. Lim. Automatic generation of robot and manual assembly plans using octrees. *Assembly Automation*, 30(2):173–183, 2010.

[174] S. Micali and V.V. Vazirani. An O (v| v| c| E|) algorithm for finding maximum matching in general graphs. In *| 21st Annual Symposium on Foundations of Computer Science*, pages 17–27. IEEE, 1980.

[175] R Garey Michael and S Johnson David. Computers and intractability: a guide to the theory of np-completeness. *WH Freeman & Co., San Francisco*, 1979.

[176] Rico Möckel, Cyril Jaquier, Kevin Drapel, Elmar Dittrich, Andres Upegui, and A Ijspeert. Yamor and bluemove—an autonomous modular robot with bluetooth interface for exploring adaptive locomotion. In *Climbing and Walking Robots*, pages 685–692. Springer, 2006.

[177] R Moeckel, Y. Perov, A. The Nguyen, M Vespignani, S Bonardi, S. Pouya, A. Sproewitz, J. Van den Kieboom, F. Wilhelm, and AJ Ijspeert. Gait optimization for roombots modular robots - matching simulation and reality. In *Proceedings of IEEE/RSJ IROS 2013, Tokyo, Japan, November 3*. IEEE, 2013.

[178] Francesco Mondada, André Guignard, Alexandre Colot, Dario Floreano, Jean-Louis Deneubourg, Luca Gambardella, Stefano Nolfi, and Marco Dorigo. Swarm-bot: A new concept of robust all-terrain mobile robotic system. 2002.

[179] Kendra E. Moore, Aşkıner Güngör, and Surendra M. Gupta. Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships. *European Journal of Operational Research*, 135(2):428–449, December 2001.

[180] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-TRAN: self-reconfigurable modular robotic system. *IEEE/ASME Trans. Mechatron.*, 7(4):431–441, 2002.

[181] Satoshi Murata, Haruhisa Kurokawa, and Shigeru Kokaji. Self-assembling machine. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 441–448. IEEE, 1994.

[182] Satoshi Murata, Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, and Shigeru Kokaji. A 3-d self-reconfigurable structure. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 432–439. IEEE, 1998.

[183] Satoshi Murata, Eiichi Yoshida, Akiya Kamimura, Haruhisa Kurokawa, Kohji Tomita, and Shigeru Kokaji. M-tran: Self-reconfigurable modular robotic system. *Mechatronics, IEEE/ASME Transactions on*, 7(4):431–441, 2002.

[184] R.R. Murphy and J.L. Burke. Up from the rubble: Lessons learned about HRI from search and rescue. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, volume 49, pages 437–441. Human Factors and Ergonomics Society, 2005.

[185] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[186] Nils Napp and Eric Klavins. A compositional framework for programming stochastically interacting robots. *The International Journal of Robotics Research*, 30(6):713–729, 2011.

[187] Carl A Nelson. A framework for self-reconfiguration planning for unit-modular robots. 2005.

[188] John von Neumann and Arthur W Burks. Theory of self-reproducing automata. 1966.

[189] An Nguyen, Leonidas J Guibas, and Mark Yim. Controlled module density helps reconfiguration planning. In *Proc. of 4th International Workshop on Algorithmic Foundations of Robotics*, pages 23–36, 2000.

[190] C.W. Nielsen, M.A. Goodrich, and J.W. Crandall. Experiments in human-robot teams. In *Multi-robot systems: from swarms to intelligent automata: proceedings from the 2003 International Workshop on Multi-Robot Systems*, page 241. Springer Netherlands, 2003.

[191] Simon Notheis, Wilhelm August, Björn Hein, and Heinz Wörn. Ar-based approach for evaluation of new model-based control algorithms. In *Proceedings of the 7th German Conference on Robotics*, 2012.

[192] Tadashi Odashima, Masaki Onishi, Kenji Tahara, Kentaro Takagi, Fumihiko Asano, Yo Kato, Hiromichi Nakashima, Yuichi Kobayashi, Toshiharu Mukai, Zhiwei Luo, et al. A soft human-interactive robot ri-man. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1–1. IEEE, 2006.

[193] D.R. Olsen and M.A. Goodrich. Metrics for evaluating human-robot interactions. In *Proceedings of PERMIS*, volume 2003. Citeseer, 2003.

[194] OpenNI. OpenNI, 2014.

[195] E. H. Ostergaard, K. Kassow, R. Beck, and H. H. Lund. ATRON lattice-based self-reconfigurable robot. *Autonomous Robots*, 21(2):165–183, 2006.

[196] Ayberk Ozgur. Natural User Interface for Roombots. http://biorob2.epfl.ch/utils/movieplayer.php?id=274.

[197] Ayberk Özgür, Stéphane S. Bonardi, Massimo Vespignani, Rico Möckel, and Auke Jan Ijspeert. Natural user interface for roombots. In *RO-MAN, 2014 IEEE*. IEEE, 2014.

[198] Bradley Evan Paden. Kinematics and control of robot manipulators. 1985.

[199] A. Pamecha, I. Ebert-Uphoff, and G.S. Chirikjian. Useful metrics for modular robot motion planning. *IEEE Transactions on Robotics and Automation*, 13(4):531–545, 1997.

[200] Amit Pamecha, Chih-Jung Chiang, David Stein, and Gregory Chirikjian. Design and implementation of metamorphic robots. In *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, volume 10. Irvine, California, USA: ASME, 1996.

[201] R. Parasuraman, S. Galster, P. Squire, H. Furukawa, and C. Miller. A flexible delegation-type interface enhances system performance in human supervision of multiple robots: Empirical studies with RoboFlag. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(4):481–493, 2005.

[202] F.C. Park. Computational aspects of the product-of-exponentials formula for robot kinematics. *Automatic Control, IEEE Transactions on*, 39(3):643–647, 1994.

[203] Michael Park, Sachin Chitta, Alex Teichman, and Mark Yim. Automatic configuration recognition methods in modular robots. *The International Journal of Robotics Research*, 27(3-4):403–421, 2008.

[204] Mike Peasgood, Christopher M Clark, and John McPhee. A complete and scalable strategy for coordinating multiple robots within roadmaps. *Robotics, IEEE Transactions on*, 24(2):283–292, 2008.

[205] Lionel S Penrose. Self-reproducing machines. *Scientific American*, 200(6):105–114, 1959.

[206] D. Perzanowski, AC Schultz, W. Adams, E. Marsh, and M. Bugajska. Building a multimodal human-robot interface. *IEEE intelligent systems*, 16(1):16–21, 2001.

[207] Kirstin Petersen, Radhika Nagpal, and Justin Werfel. Termes: An autonomous robotic system for three-dimensional collective construction. *Proc. Robotics: Science & Systems VII*, 2011.

[208] W. Piekarski and B. Thomas. Arquake: the outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, 2002.

[209] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[210] Soha Pouya, Jesse Van Den Kieboom, Alexander Spröwitz, and Auke Ijspeert. Automatic gait generation in modular robots: to oscillate or to rotate? that is the question. *Proceedings of IEEE/RSJ IROS 2010, Taipei, Taiwan, October 18*, 22, 2010.

[211] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

[212] Konstantine C Prevas, Cem Unsal, Mehmet Onder Efe, and Pradeep K Khosla. A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 787–792. IEEE, 2002.

[213] PrimeSense Inc. PrimeSense™ NITE Algorithms 1.5, 2013.

[214] D.V. Pynadath and M. Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems*, 7(1):71–100, 2003.

[215] AK Raj, JM Bradshaw, RW Carff, M. Johnson, and S. Kulkarni. An agent based approach for Aug Cog integration and interaction. In *Proceedings of Augmented Cognition-Improving Warfighter Information Intake Under Stress, Scientific Investigators Meeting*, pages 6–8, 2004.

[216] P. Rani, N. Sarkar, and CA Smith. Affect-sensitive human-robot cooperation-theory and experiments. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA'03*, volume 2, 2003.

[217] P. Rani, J. Sims, R. Brackin, and N. Sarkar. Online stress detection using psychophysiological signals for implicit human-robot cooperation. *Robotica*, 20(06):673–685, 2002.

[218] John H Reif and Sam Slee. Optimal kinodynamic motion planning for self-reconfigurable robots between arbitrary 2d configurations. In *Georgia Institute of Technology*. Citeseer, 2007.

[219] C. Rich and C.L. Sidner. COLLAGEN: When agents collaborate with people. In *Proceedings of the first international conference on Autonomous agents*, pages 284–291. ACM, 1997.

[220] A.H. RONALD. Dynamic Programming and Markov Processes, 1960.

[221] Daniela Rus and Marsette Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, 2001.

[222] Behnam Salemi, Mark Moll, and Wei-Min Shen. Superbot: A deployable, multifunctional, and modular self-reconfigurable robotic system. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3636–3641. IEEE, 2006.

[223] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[224] J. Scholtz. Theory and evaluation of human robot interactions. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, page 10, 2003.

[225] J. Scholtz, B. Antonishek, and J. Young. Evaluation of operator interventions in autonomous off-road driving. In *Proc. NIST Performance Metrics for Intelligent Systems Workshop*, 2003.

[226] J. Scholtz, B. Antonishek, and J. Young. Evaluation of a human-robot interface: development of a situational awareness methodology. In *Proceedings of the Hawaii International Conference on Systems Sciences*. Citeseer, 2004.

[227] Emmanuel Senft. Misalignment compensation during the connection phase of the SRMR Roombots. http://biorob2.epfl.ch/utils/movieplayer.php?id=277.

[228] Wei-Min Shen, Behnam Salemi, and Peter Will. Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. *Robotics and Automation, IEEE Transactions on*, 18(5):700–712, 2002.

[229] Jeremy G Siek, Lie-Quan Lee, and Andrew Lumsdaine. *Boost Graph Library: User Guide and Reference Manual, The*. Pearson Education, 2001.

[230] M. Sierhuis, J.M. Bradshaw, A. Acquisti, R. van Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice. In *Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2003.

[231] Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.

[232] M. Singh. A customizable coordination service for autonomous agents. *Intelligent Agents IV Agent Theories, Architectures, and Languages*, pages 93–106.

[233] Russell Smith et al. Open dynamics engine, 2005.

[234] D. Sofge, D. Perzanowski, M. Skubic, M. Bugajska, J.G. Trafton, N. Cassimatis, D. Brock, W. Adams, A. Schultz, and Naval Research Lab Washington DC Center for Applied Research in Artificial Intelligence. Cognitive tools for humanoid robots in space. In *Proceedings of IFAC Symposium on Automatic Control in Aerospace*. Citeseer, 2004.

[235] Alexander Sproewitz, Aude Billard, Pierre Dillenbourg, and Auke Jan Ijspeert. Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4259–4264. IEEE, 2009.

[236] Alexander Sproewitz, Philippe Laprade, Stéphane Bonardi, Mikael Mayer, Rico Moeckel, P-A Mudry, and Auke Jan Ijspeert. Roombots—towards decentralized reconfiguration with self-reconfiguring modular robotic metamodules. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1126–1132. IEEE, 2010.

[237] Alexander Sproewitz, Rico Moeckel, Jérôme Maye, and Auke Jan Ijspeert. Learning to move in modular robots using central pattern generators and online optimization. *The International Journal of Robotics Research*, 27(3-4):423–443, 2008.

[238] Alexander Spröwitz, Rico Möckel, Massimo Vespignani, Stéphane Bonardi, and AJ Ijspeert. Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot. *Robotics and Autonomous Systems*, 62(7):1016–1033, 2014.

[239] Alexander Spröwitz, Soha Pouya, Stéphane Bonardi, Jesse van den Kieboom, Rico Möckel, Aude Billard, Pierre Dillenbourg, and Auke Ijspeert. Roombots: Reconfigurable Robots for Adaptive Furniture. *IEEE Computational Intelligence Magazine, special issue on "Evolutionary and developmental approaches to robotics"*, 5(3):20–32, 2010.

[240] E. Stavridis. Design and optimization of active connection mechanisms (acms) for roombots modular robots. Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2013.

[241] A. Stentz. Optimal and efficient path planning for partially-known environments. In *ICRA*, pages 3310–3317. IEEE, 1994.

[242] M. Stöckli. Reconfiguration algorithm for adaptive furniture. Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2012.

[243] K. Stoy. Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems*, 54(2):135–141, 2006.

[244] K. Stoy. Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems*, 54(2):135–141, February 2006.

[245] Kasper Stoy, David Brandt, and David Johan Christensen. *Self-reconfigurable robots: an introduction*. MIT Press, 2010.

[246] Kasper Stoy, Wei-Min Shen, and Peter M Will. Using role-based control to produce locomotion in chain-type self-reconfigurable robots. *Mechatronics, IEEE/ASME Transactions on*, 7(4):410–417, 2002.

[247] Tomomichi Sugihara. Solvability-unconcerned inverse kinematics based on levenberg-marquardt method with robust damping. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 555–560. IEEE, 2009.

[248] Tomomichi Sugihara. Solvability-unconcerned inverse kinematics by the levenberg–marquardt method. *Robotics, IEEE Transactions on*, 27(5):984–991, 2011.

[249] John W Suh, Samuel B Homans, and Mark Yim. Telecubes: Mechanical design of a module for self-reconfigurable robotics. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 4095–4101. IEEE, 2002.

[250] Petr Švestka and Mark H Overmars. Coordinated path planning for multiple robots. *Robotics and autonomous systems*, 23(3):125–152, 1998.

[251] J.E. Swan, A. Jones, E. Kolstad, M.A. Livingston, and H.S. Smallman. Egocentric depth judgments in optical, see-through augmented reality. *Visualization and Computer Graphics, IEEE Transactions on*, 13(3):429–442, 2007.

[252] SweetHome3D. http://www.sweethome3d.com/index.jsp. Free interior design application.

[253] Kiva Systems. Kiva systems. http://www.kivasystems.com/.

[254] M. Tambe. Towards flexible teamwork. *Arxiv preprint cs/9709101*, 1997.

[255] M. Tambe, J. Adibi, Y. Al-Onaizan, A. Erdem, G.A. Kaminka, S.C. Marsella, and I. Muslea. Building agent teams using an explicit teamwork model and learning. *Artificial intelligence*, 110(2):215–239, 1999.

[256] Molecubes team. Molecubes For Everyone. http://fabathome.mae.cornell.edu/cubes/index.php?title=Molecubes_For_Everyone.

[257] Sreenivas Tejomurtula and Subhash Kak. Inverse kinematics in robotics using neural networks. *Information Sciences*, 116(2):147–164, 1999.

[258] Yuzuru Terada and Satoshi Murata. Automatic modular assembly system and its distributed control. *IJRR*, 27(3-4):445–462, March 2008.

[259] Ulrike Thomas and Friedrich M Wahl. Assembly planning and task planning—two prerequisites for automated robot programming. In *Robotic Systems for Handling and Assembly*, pages 333–354. Springer, 2011.

[260] Richard L Thompson and Narendra S Goel. Movable finite automata (mfa) models for biological systems i: Bacteriophage assembly and operation. *Journal of Theoretical Biology*, 131(3):351–385, 1988.

[261] G. Tidhar. Team-oriented programming: Preliminary report. *Technical Note*, 41, 1993.

[262] G. Tidhar. Team-oriented programming: Social structures. *Technical Notes*, 47, 1993.

[263] Kohji Tomita, Satoshi Murata, Haruhisa Kurokawa, Eiichi Yoshida, and Shigeru Kokaji. Self-assembly and self-repair method for a distributed mechanical system. *Robotics and Automation, IEEE Transactions on*, 15(6):1035–1045, 1999.

[264] G. Trafton, A.C. Schultz, N.L. Cassimatis, L. Hiatt, D. Perzanowski, D.P. Brock, M. Bugajska, and W. Adams. Using similar representations to improve human-robot interaction. *Agents and Architectures (to appear)*, 2004.

[265] JG Trafton, AC Schultz, M. Bugajska, and F. Mintz. Perspective-taking with robots: experiments and models. In *IEEE International Workshop on Robot and Human Interactive Communication, 2005. ROMAN 2005*, pages 580–584, 2005.

[266] J.G. Trafton, A.C. Schultz, N.L. Cassimatis, L.M. Hiatt, D. Perzanowski, D.P. Brock, M.D. Bugajska, and W. Adams. Communicating and collaborating with robotic agents. *Cognition and multi-agent interaction: from cognitive modeling to social simulation*, page 252, 2006.

[267] B. Trouvain and HL Wolf. Evaluation of multi-robot control and monitoring performance. In *11th IEEE International Workshop on Robot and Human Interactive Communication, 2002. Proceedings*, pages 111–116, 2002.

[268] T Ueyama. A study on dynamically reconfigurable robotic systems (10th report, distributed control structure for organization using an evaluation of network energy for group structure of cebot). *J. JSME (C)*, 58(549):132–139, 1992.

[269] JJ Uicker, J Denavit, and RS Hartenberg. An iterative method for the displacement analysis of spatial mechanisms. *Journal of Applied Mechanics*, 31(2):309–314, 1964.

[270] Cem Unsal and Pradeep K Khosla. A multi-layered planner for self-reconfiguration of a uniform group of i-cube modules. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 598–605. IEEE, 2001.

[271] Serguei Vassilvitskii, Jeremy Kubica, Eleanor Rieffel, John Suh, and Mark Yim. On the general reconfiguration problem for expanding cube style modular robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 801–808. IEEE, 2002.

[272] Massimo Vespignani, Emmanuel Senft, Stéphane Bonardi, Rico Moeckel, and Auke J Ijspeert. An experimental study on the role of compliant elements on the locomotion of the self-reconfigurable modular robots roombots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4308–4313. Ieee, 2013.

[273] M Vona and DL Rus. A physical implementation of the self-reconfiguring cristalline robot. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1726–1733, 2000.

[274] J. Wang and M. Lewis. Assessing cooperation in human control of heterogeneous robots. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 9–16. ACM, 2008.

[275] Webots. http://www.cyberbotics.com. Commercial Mobile Robot Simulation Software.

[276] Hongxing Wei, Youdong Chen, Jindong Tan, and Tianmiao Wang. Sambot: A self-assembly modular robot system. *Mechatronics, IEEE/ASME Transactions on*, 16(4):745–757, 2011.

[277] Wikipedia. Matching (graph theory) — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Matching_(graph_theory)&oldid=403424165, 2010. [Online; accessed 17-January-2011].

[278] Stephen Wolfram. *Cellular automata and complexity: collected papers*, volume 1. Addison-Wesley Reading, 1994.

[279] M. Yim, W.M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G.S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):43–52, 2007.

[280] M. Yim, Y. Zhang, J. Lamping, and E. Mao. Distributed control for 3D metamorphosis. *Autonomous Robots*, 10(1):41–56, 2001.

[281] Mark Yim. *Locomotion with a unit-modular reconfigurable robot*. PhD thesis, Citeseer, 1994.

[282] Mark Yim. New locomotion gaits. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 2508–2514. IEEE, 1994.

[283] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S. Chirikjian. Modular Self-Reconfigurable Robot Systems. *IEEE Robot. Automat. Mag.*, 14(1):43–52, March 2007.

[284] Jungwon Yoon and Jeha Ryu. Design, fabrication, and evaluation of a new haptic device using a parallel mechanism. *Mechatronics, IEEE/ASME Transactions on*, 6(3):221–233, 2001.

[285] Yeoreum Yoon and Daniela Rus. Shady3d: A robot that climbs 3d trusses. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4071–4076. IEEE, 2007.

[286] C.H. Yu and R. Nagpal. Self-adapting modular robotics: A generalized distributed consensus framework. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1881–1888. IEEE, 2009.

[287] Chih-Han Yu, Justin Werfel, and Radhika Nagpal. Coordinating collective locomotion in an amorphous modular robot. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2777–2784. IEEE, 2010.

[288] Seung-kook Yun, David Alan Hjelle, Eric Schweikardt, Hod Lipson, and Daniela Rus. Planning the reconfiguration of grounded truss structures with truss climbing robots that carry truss elements. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1327–1333. IEEE, 2009.

[289] Yu Zhang and Lynne E Parker. Multi-robot task scheduling. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2992–2998. IEEE, 2013.

[290] Jie Zhao, Weizhong Wang, Yongsheng Gao, and Hegao Cai. Generation of closed-form inverse kinematics for reconfigurable robots. *Frontiers of Mechanical Engineering in China*, 3(1):91–96, 2008.

[291] V. Zykov, A. Chan, and H. Lipson. Molecubes: An open-source modular robotics kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, 2007.

[292] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson. Evolved and designed self-reproducing modular robotics. *Robotics, IEEE Transactions on*, 23(2):308–319, 2007.

[293] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson. Evolved and Designed Self-Reproducing Modular Robotics. *IEEE Trans. Robotics*, 23(2):308–319, April 2007.

Curriculum Vitae

Stéphane Bonardi

Education

2010 – 2014	Ph.D. in Computer Science École Polytechnique Fédérale de Lausanne (EPFL, Switzerland)
2005 – 2009	M.Sc. in Modeling and Scientific Computing ISTIL (Engineering School of Science and Technology of Lyon), Lyon, France.

Experience

2007-2008	Tutoring in Mathematics for Highschool students (baccalaureate preparation).
January 2007	Internship, Advanced Accelerator Applications (pharmaceutical firm), St Genis Pouilly, France Development of a new Software application (data base).

Publications

1. **S. Bonardi**, M. Vespignani, R. Möckel, S. Pouya, J.v.d. Kieboom, A. Spröwitz, and A. J. Ijspeert. *Automatic Generation of Reduced CPG Control Networks for Locomotion of Arbitrary Modular Robot Structures* . Robotics Science (RSS), Berkeley, USA, 2014.
2. A. Özgür, **S. Bonardi**, M. Vespignani, R. Möckel, and A. J. Ijspeert. *Natural User Interface for Roombots*. The 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Edinburgh, Scotland, UK, 2014.
3. A. Spröwitz, R. Möckel, M. Vespignani, **S. Bonardi** and A. Ijspeert. *Roombots: A Hardware Perspective on 3D Self-Reconfiguration and Locomotion with a Homogeneous Modular Robot*, in Robotics and Autonomous Systems, vol. 62, num. 7, p. 1016-1033, 2014.
4. M. Vespignani, E. Senft, **S. Bonardi**, R. Moeckel and A. J. Ijspeert. *An experimental study on the role of compliant elements on the locomotion of the self-reconfigurable modular robots Roombots*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

5. R. Moeckel, Y. N. Perov, T. N. Anh, M. Vespignani and **S. Bonardi** et al. *Gait Optimization for Roombots Modular Robots - Matching Simulation and Reality*. IEEE International Conference on Intelligent Robots and Systems (IROS), 2013.
6. **S. Bonardi**, M. Vespignani, R. Moeckel and A. J. Ijspeert. *Collaborative Manipulation and Transport of Passive Pieces using the Self-Reconfigurable Modular Robots Roombots*. IEEE International Conference on Intelligent Robots and Systems (IROS), 2013.
7. **S. Bonardi**, J. Blatter, J. Fink, R. Möckel and P. Jermann et al. *Design and Evaluation of a Graphical iPad Application for Arranging Adaptive Furniture*. 21st IEEE International Symposium on Robot and Human Interactive Communication, Paris, France, 2012.
8. A. Giusti, J. Nagi, L. M. Gambardella, **S. Bonardi** and G. A. Di Caro. *Human-Swarm Interaction through Distributed Cooperative Gesture Recognition*. International Conference on Human-Robot Interaction (HRI), 2012 7th ACM/IEEE, Boston, Massachusetts, USA, 2012.
9. **S. Bonardi**, R. Möckel, A. Spröwitz, M. Vespignani and A. Ijspeert. *Locomotion through Reconfiguration based on Motor Primitives for Roombots Self-Reconfigurable Modular Robots*. 7th German Conference on Robotics - Robotik 2012, Munich, Germany, 2012.
10. A. Spröwitz, P. Laprade, **S. Bonardi**, M. Mayer and R. Möckel et al. *Roombots-Towards Decentralized Reconfiguration with Self-Reconfiguring Modular Robotic Metamodules.* The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010.
11. A. Spröwitz, S. Pouya, **S. Bonardi**, J. van den Kieboom and R. Möckel et al. *Roombots: Reconfigurable Robots for Adaptive Furniture*, in IEEE Computational Intelligence Magazine, special issue on "Evolutionary and developmental approaches to robotics", vol. 5, num. 3, p. 20-32, 2010.

Personal Details and Coordinates

Age:	28	Stéphane Bonardi
Place of birth:	Annemasse, France	Rue du Lac 26
Citizenship:	French	CH-1020 Renens (VD)
Marital status:	Single	stephane.bonardi@epfl.ch