DOCTORAT BRETAGNE LOIRE / MATHSTIC

ENSTA BRETAGNE

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
DE TECHNIQUES AVANCÉES BRETAGNE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Automatique, Productique et Robotique*

Par
Aliah MAJED

Sensing-Based Self-Reconfigurable Strategies for Autonomous Modular Robotic Systems

Thèse présentée et soutenue à l'ENSTA Bretagne, le 15 décembre 2022
Unité de recherche : Lab-STICC UMR CNRS 6285

Rapporteurs avant soutenance :

Nadine PIAT	Professeur des Universités, ENSMM / FEMTO-ST, France
Abdelhafid ABOUAISSA	Professeur des Universités, Université de Haute Alsace, France

Composition du Jury :

Président :	Cédric BUCHE	Professeur des Universités, ENIB / IRL CROSSING, Australia
Examinateurs :	Ali JABER	Professor, Lebanese University, Lebanon
	Abbas NASSER	Associate Professor, American University of Culture & Education, Lebanon (co-advisor)
Dir. de thèse :	Benoît CLEMENT	Professeur, ENSTA Bretagne / IRL CROSSING, Australia

Invité(s) :

Hassan HARB Associate Professor, American University of Culture & Education, Lebanon (supervisor)

DOCTORAT BRETAGNE LOIRE / MATHSTIC

ENSTA BRETAGNE

Title: Sensing-Based Self-Reconfigurable Strategies for Autonomous Modular Robotic Systems

Keywords: Modular Robotic System, Self-Reconfiguration, Data Storage Reduction, Decision Making, Module Clustering, Efficient Communication, Embedding Techniques, Multi-Detectors, Roombot.

Abstract: Modular robotic systems (MRSs) have become a highly active research today. It has the ability to change the perspective of robotic systems from machines designed to do certain tasks to multi-purpose tools capable of accomplishing almost any task. They are used in a wide range of applications, including reconnaissance, rescue missions, space exploration, military task, etc. Constantly, MRS is built of “modules” from a few to several hundreds or even thousands . Each module involves actuators, sensors, computational, and communicational capabilities. Usually, these systems are homogeneous where all the modules are identical; however, there could be heterogeneous systems that contain different modules to maximize versatility. One of the advantages of these systems is their ability to operate in harsh environments in which contemporary human-in-the-loop working schemes are risky, inefficient and sometimes infeasible.

In this thesis, we are interested in self-reconfigurable modular robotics. In such systems, it uses a set of detectors in order to continuously sense its surroundings, locate its own position, and then transform to a specific shape to perform the required tasks. Consequently, MRS faces three major challenges. First, it offers a great amount of collected data that overloads the memory storage of the robot. Second it generates redundant data which complicates the decision making about the next morphology in the controller. Third, the self reconfiguration process necessitates massive communication between the modules to reach the target morphology and takes a significant processing time to self-reconfigure the robotic. Therefore, researchers’ strategies are often targeted to minimize the amount of data collected by the modules without considerable loss in fidelity. The goal of this reduction is first to save the storage space in the MRS, and then to facilitate analyzing data and making decision about what morphology to use next in order to adapt to new circumstances and perform new tasks. In this thesis, we propose an efficient mechanism for data processing and self-reconfigurable decision-making dedicated to modular robotic systems. More specifically, we focus on data storage reduction, self-reconfiguration decision-making, and efficient communication management between modules in MRSs with the main goal of ensuring fast self-reconfiguration process.

First, we propose data storage reduction mechanism in order to eliminate redundant data thus, reducing the amount of data needed to be stored in the MRS. Our objective is to aggregate similar data while preserving the dynamicity of the monitored conditions. We study the similarity between detectors readings according to the aggregation approach, we use Sim function based on a predefined threshold; then we observe if the difference between readings is smaller than the threshold then the readings are considered similar. Otherwise, it was considered different.

The second objective of this thesis is to allow MRS to take real-time decisions to decide which morphology, is the most suitable to the current surrounding status. for that reason we propose an efficient model based on the fuzzy logic. Typically, a fuzzy set consists of several elements where each of one has a degree of membership. Then we define a score table (ST) which is a customizable guide used by the robot controller in order to determine the criticality of each monitored condition. The main target of ST is to allow early recognition of events that alert the MRS to change its current configuration to a new one that is suitable to the surrounding. Assume we have a set of predefined events, where each of them imposes MRS to adapt itself to a unique morphology. Then, in order to check the accuracy of the selected morphology, we propose to calculate the strength of the occurred event. This will allow the MRS to periodically check if the current morphology is the most suitable for the monitored condition.

The third objective of this thesis is to provide fast reconfiguration process and reduce the communication between modules in MRS. for that reason we propose a robust mechanism for self-reconfigurable robotics called RUN which consist of two stage. At the first stage, we classify the modules into clusters before performing their transitions. To do that, we first select a set of modules to act as cliques for the robotic. A clique module is responsible to allow an efficient communication between the whole modules during the self-reconfiguration process. On one hand, this will allow to reduce the number of messages transmitted between the modules and, on the other hand, to minimize the number of transitions made by each module (thus saving its energy). At the second stage, we

propose two communication algorithms, the first algorithm is inter-module that allows efficient communication between the cliques of the clusters and aims to minimize the number of transmitted messages in the robotic system to avoid packet congestion and save the module energy, and the second algorithm is intra-module based on the tree structure that reduces the number of communications between the modules in the same clusters.

Forth, we proposed a fast self-reconfiguration technique called FSET, dedicated to MRS. Our proposed technique consists mainly of two stages: root selection and morphology formation. The final goal of these stages is to enhance the time cost to get new morphology of the traditional SET algorithm thus, ensuring fast self-reconfiguration. The root selection stage selects a small number of modules in order to find the best tree roots that affect the topological conditions and lead to the success of the embedding process or not. The morphology formation stage uses the traditional SET algorithm to calculate the embedding truth table where the initial roots used are taken from the first stage.

To evaluate the performance of the proposed techniques, simulations on real robot have been conducted. We have analyzed their performances according to complexity, overhead communication, optimality (minimum number of steps), and time efficiency and we show how our techniques can significantly improve the performance of robotic systems through providing a fast reconfiguration process in MRS and reducing the energy consumption of modules.

Contents

List of Figures

List of Tables

List of Abbreviations

MRS.....**M**odular **R**obotic **S**ystem

ST.....**S**core **T**able

RUN.....**R**obust **U**ster-based pla**N**ning

SET.....**S**tructure **E**mbedding **T**echnique

FSET.....**F**ast **S**elf **R**reconfiguration **T**echnique

ED.....**E**uclidean **D**istance

$\mathcal{M}_i$.....Module number i

$\mathcal{O}$.....Set of Morphology

F_i.....Degree of freedom

D_β.....Set of β Detectors

R_i.....Vector of Reading of Detector i

ε.....Set of thresholds for Similar Function

V_t.....Readings Vector Collected During the Slot Time t

$Sim(V_i, V_j)$.....Similar Function

$wgt(V_i)$.....Weight Function

τ.....Number of Reading during a Period

δ.....Thresholds of Score Table

$\mathcal{E}$.....Set of Predefined Events

s_{k_i}.....Score of Reading r_{k_i} Collected by the Detector D_i

MoE.....Matrix of Events

S_j.....Vector of Scores

$ED(S_j, E_i)$.....Euclidean Distance Between S_j and Event E_i

MoE.....Matrix of Events

Z_i.....Fuzzy Set

S_i.....Set of Element Scores

m_{s_i}..... Degree of Membership of s_i

G_{E_i}.....Strength of E_i

$\mathcal{P}_i$.....Module Positions

p_k.....Coordinates of Module M_k

$\mathcal{B}$.....Set of Independent Modules

$|M_i|$.....Number of Neighbors of Module M_i

μ.....Number of Modules

n.....Spatial Correlation Threshold

E.....Initial Energy of MRS

E_{M_i}.....Initial Energy of Module

N.....Neighbors Threshold

T.....Action Threshold

I.....Energy Threshold

$\mathcal{Q}$.....Set of Cliques

$\mathcal{C}$.....Set of module clusters

$W_{E_{i,j}}$.....Weight of an Edge $E_{i,j}$ Connecting the Modules M_i and M_j

Er_i.....Residual Energy of the Module M_i

Ep_i.....Energy Consumed in the Module M_i During the Transmission and Receiving of Packets

Em_i.....Energy Needed to Move the Module M_i to its Final Position

$Ec_{i,j}$.....Energy Cost

Et_i.....Total Energy Cost During the Data Transmission

$\mathcal{T}$.....Tree

G_i.....Sets of Modules

T_s.....Percentage of Training Set

$T*$.....Embedding Truth Table

$G(M, L)$.....Connected Graph;M denotes the set of nodes; L denotes the set of link

$T[m_k, n_k]$.....Truth Table Entry

Dedication

For the sake of **ALLAH** (Subhanahu Wa Ta'ala) and due to his unique biography, I am dedicating this thesis to 'Abul-Qasim-Muhammad-ibn-Hassan- Askari, "Imam Al-Mahdi", who will fill the earth with justice and equity, after it has been filled with tyranny and oppression. I hope to be graced with his acceptance.

Acknowledgements

I would like to thank all the people who contributed in some way to the work described in this thesis.

I would like to express my sincere appreciation and gratitude to my supervisors: Prof. Benoit Clement at ENSTA Bretagne, and Dr. Abbass Nasser and Dr. Hassan Harb at the American University of Culture and Education (AUCE), for their guidance during my research. Their support, efforts and inspiring suggestions were essential to the birth of this document and to my formation as a future researcher. Particulary, I am very grateful to **Dr. Hassan Harb** who has been a constant source of encouragement and enthusiasm, not only during this thesis but also during my Master intership. I appreciate all his contributions of times and ideas to make my Ph.D productive and stimulating, only my forever prayer will be sufficient to thank him.

For this thesis, I am grateful to my reading committee members Prof. Abdelhafid Abouaissa and Prof. Nadine Piat for their time, interest, and helpful comments. I would also like to thank the other two members of my oral defense committee Prof. Cédric Buche and Prof. Ali Jaber for their time and insightful questions.

I am very gratefully to all people I have met along the way and have contributed to the developement of my research. My appreciation and thanks go to all members of the STICC laboratory at ENSTA Bretagne and all members of the comupter science department at AUCE.

Lastly, my deepest gratitude goes to my family for their unflagging love and unconditional support throughout my life and my studies. For my parents, Ali and Khadija, who raised me with a love of science and supported me in all my pursuits. You made me live the most unique, magic and carefree childhood that has made me who I am now. For my sisters Zeinab and Mariam, and my brothers Hussein and Hassan whose faithful support during this Ph.D. The support and the encouraging of all members of my big family are so appreciated. THANK YOU.

Chapter 1

Introduction

1.1 General Introduction

Robotics system have been defined, according to MIT Technology Review, one of 10 emerging technologies that will change the world [8]. Therefore, they have attracted a great attention of researchers and they have become one of the most interesting areas of research in the few years. Thanks to revolution in the technology sector, a huge number of smart devices have emerged and used everywhere moving most of the daily tasks to machines. Particularly, robotics is one of the most affected domains by the technology revolution due to the great advancements in mechanical and electrical materials. Thus, new features have been added to the robots that totally changes their traditional working way and makes them smarter. One of such features is the self-reconfiguration process that allows the robot to take several shapes, called morphologies, according to the variation of surrounding in which it exists and the task to be accomplished. This leads to a new generation of robots known as modular robotic system (MRS). MRS has the potential to transform robotics from bespoke systems designed to accomplish specific tasks to tools that can be reconfigured to perform an infinite number of tasks. Hence, MRS has found its way quickly into a great number of applications including rescue, healthcare, manufacturing, reconnaissance and military missions

Typically, a modular robotic system (MRS) consists of a set of units, called modules. A module is defined as small, low-cost and limited resources device that communicates wirelessly and has the capabilities of sensing and processing collected data. It has several faces that can connect to or detach from its neighbors allowing MRS to reconfigure it shape to adapt to new circumstances. Each module has a few degrees of freedom (DOFs). Usually, the DOFs are used to define the motion capabilities of robots; more the DOFs increase more the flexibility in changing the MRS morphology is. Subsequently, the MRS monitors the target environments thanks to a set of detector devices attached on its body. These detectors collect data of the MRS surroundings and send them toward the system controller (SC). In its turn, the SC analyses the collected data and decides about the next morphology must be taken by the MRS then, it sends signals to the actuators to reshape the MRS morphology

Currently, researchers attention is focused to the three most critical constraint in MRSs [9,10]: the big data collected by the detectors, the self-reconfiguration process and the communication management between modules. From one hand, and for reliability purposes, the detectors should continuously collect data about MRS surroundings in order to detect any possible variation then, to quickly decide about the new morphology. This leads to a big data collection problem that:1) overloads the

memory storage of the robot; 2) generates redundant data which complicates the decision making about the next morphology. Hence, designing new data reduction storage techniques is essential for MRS in order to avoid the overloading of its storage space. On the other hand, and after data acquisition, the MRS should be able to analyze the collected data in a real-time manner and select the suitable morphology among a set of morphologies defined during its creation. This self-reconfiguration process also requires a massive number of communication between modules when transferring to a new morphology, thus, makes the communication management very difficult. This self-reconfiguration process is a crucial process in MRS because it leads sometimes to reshape the robot into a wrong morphology thus, destruction of the MRS itself. Therefore, proposing efficient data decision techniques for self reconfigurable MRS takes a great attention from researchers and industries

In this thesis, we propose an efficient mechanism for data processing and self-reconfigurable decision-making dedicated to modular robotic systems. More specifically, we focus on data storage reduction, self-reconfiguration decision-making, and efficient communication management between modules in MRSs with the main goal of ensuring fast self-reconfiguration process. Our proposed techniques are validated via simulations on real robot, known as Roombots and comparison with other existing techniques. The results show that the effectiveness of our techniques in terms of improving the complexity, overhead communication, optimality (minimum number of steps), and time efficiency.

1.2 Main Contribution of This Thesis

The main contributions in this thesis concentrate on designing efficient data storage reduction, self-reconfiguration decision-making, and efficient module communication in MRSs with the main goal of ensuring fast self-reconfiguration process.

1. **Data Storage Reduction:** In MRS, the main objective of the robot is to adapt itself to variation of its environments. Hence, a MRS uses a set of detectors in order to sense its surroundings, locate its own position, and then transform to a specific shape to perform the required tasks. Indeed, the continuous monitoring of the environment to detect any possible variation, and to quickly decide about the next morphology, leads to a big data collection problem that: first it overloads the memory storage of the robot; second it generates redundant data which complicates the decision making about the next morphology in the controller. Hence, designing new data reduction storage techniques is essential for MRS in order to avoid the overloading of its storage space and to facilitate data analysis.

   In this thesis we propose a sensing-based processing mechanism for data storage and decision making in MRSs. The goal of this mechanism is to remove the redundancy exiting among the collected data thus, save the storage space and facilitate data analysis. The proposed technique aggregate similar data while preserves the dynamicity of the monitored conditions. According to the aggregation approach, the similarity between readings collected by a detector can be determined according to the *Sim* function based on a predefined threshold; if the difference between readings is less than the threshold then the readings are considered similar. Then we show which data will be saved in the storage space of a MRS using our reduction mechanism. In the first slot

time, the readings collected by all detectors will be saved automatically in the disk storage with a weight sets to 1. Then, for the later collected readings, the controller will search the similarity between these readings and those collected in the previous slot time; if the readings are similar according to the Sim function then the controller removes the new readings while increasing the weight of the previous readings by 1 otherwise, it adds the new collected readings to the disk storage and initializes its weight to 1. We showed via simulations on real detector data, that our approach can be effectively used to reduce the data storage and improving the self-reconfiguration decision.

2. **Self-Reconfiguration Decision-Making:** The self-reconfiguration ability is totally changed the definition of the traditional robotic and allows the emergence of MRS with prominent features including adapt ability, expansibility, versatility and robustness. In MRS, the decision-making is the main mission of the robot in order to self-reconfigure its morphology to the surroundings. After data acquisition, the MRS should be able to analyze the collected data in a real-time manner and select the suitable morphology among a set of morphologies defined during its creation. But the huge number of data collected makes the self-reconfiguration decision very complicated. This self-reconfiguration process is a crucial process in MRS. Because sometimes, in real-time the decision taken by the robot will not be correct due to two reasons: first, a false event happens, or, second, erroneous data is captured by the detectors. This leads to changing the MRS morphology to an unsuitable shape and still until the next occurred event. Thus, destruction of the MRS itself. Therefore, proposing efficient data decision techniques takes great attention from researchers and industries.

   This thesis proposes an efficient model based on the fuzzy logic that allows MRS to take real-time decisions to decide which morphology, among its set of morphologies, is the most suitable to the current surrounding status. Typically, a fuzzy set consists of several elements where each of one has a degree of membership. Then we define a score table (ST) which is a customizable guide used by the robot controller in order to determine the criticality of each monitored condition. The main target of ST is to allow early recognition of events that alert the MRS to change its current configuration to a new one that is suitable to the surrounding. Assume we have a set of predefined events, where each of them imposes MRS to adapt itself to a unique morphology. Then, in order to check the accuracy of the selected morphology, we propose to calculate the strength of the occurred event. This will allow the MRS to periodically check if the current morphology is the most suitable for the monitored condition.

   In addition, we have proposed a fast self-reconfiguration technique called FSET, dedicated to MRS. Our proposed technique consists mainly of two stages: root selection and morphology formation. The final goal of these stages is to enhance the time cost to get new morphology of the traditional SET algorithm thus, ensuring fast self-reconfiguration. The root selection stage selects a small number of modules in order to find the best tree roots that affect the topological conditions and lead to the success of the embedding process or not. The morphology formation stage uses the traditional SET algorithm to calculate the embedding truth table where the initial roots used are taken from the first stage. Finally, our proposed technique, under the two proposed strategies, has been evaluated through simulations on a real scenario using M-TRAN, where

the obtained results were very encouraged in terms of providing a fast reconfiguration process in MRS and reducing the energy consumption of modules.

3. **Module Communication:** Communication is central to module coordination. In MRS, the transition between morphologies is a challenging task due to several reasons: first, it requires a massive communication between the modules to reach the target morphology. Second, it takes a significant processing time to self-reconfigure the robotic, which becomes crucial in critical applications. Third, it consumes the limited available energy of modules. Therefore, in order to ensure a long MRS functionality, we have to conserve the module energies by reducing the communication overhead and the processing time that consumes most of their energies. Hence, designing efficien module communication techniques is essential for MRS in order to provide efficient communication between the modules and save their battery power.

   In this thesis, we propose an efficient and fast self-reconfiguration mechanism for modular robotics called RUN, Robust clUster-based plaNning. The proposed techniques aim to minimize the communication overhead between the modules and enhance the configuration process during the transition between MRS morphologies. Mainly, RUN is based on the clustering of modules and it consists of two stages which can be described as follows: The first stage called as module clustering and aims to group the modules into a set of clusters while assigning a special module for each cluster known as clique; The second stage called as module communication and allows an efficient communication with and within clusters through two proposed algorithms named as inter-module and intra-module respectively.

1.3 Thesis Structure

The thesis is structured as follows:

Chapter 2: Modular Robotic System : An Overview: This chapter provides an introduction to the wide field of modular robotic systems. We present the various concepts related to its objectives, its features, and the different fields of application. We also gives a survey about various MRS prototypes and the architectures adapted to their modules. Then, we show some potential applications of MRS through real projects. Furthermore, we describe the main challenges that face such systems, such as self-reconfiguration, sensing data analysis, hardware design, etc. We highlight self-reconfiguration as a primary challenge in MRS.

Chapter 3: Sensing-Based Self-Reconfigurable Decision-Making Mechanism for MRS: This chapter focuses on the problem of big data collected by the MRS that overloads the memory storage of the robot and complicates the decision-making about the next morphology. We propose an efficient mechanism consisting of two phases: data storage reduction and self-reconfiguration. The first phase searches the similarity between the collected data according to the *Sim* function in order to eliminate redundant data thus, reducing the amount of data needed to be stored in MRS. The second phase uses the fuzzy logic model that allows MRS to be self-reconfigurable by taking the right decision about the desired shape to perform the

required tasks.

Chapter 4: RUN: A Robust Cluster-Based Planning for Fast Self-Reconfigurable Modular Robotic: This chapter is dedicated to accelerate the self-reconfiguration process in MRS by reducing the communication overhead between the modules. We propose a self-reconfiguration mechanism called RUN that works on two stages. The first stage, called as module selection, aims to group the modules into a set of clusters while assigning a special module for each cluster known as clique. The second stage called as module communication and allows an efficient communication with and within clusters through two proposed algorithms named as in-module and on-module respectively.

Chapter 5: FSET: Fast Structure Embedding Technique for Self-Reconfigurable Modular Robotic Systems: In this chapter, we have proposed a fast self-reconfiguration technique called FSET, dedicated to MRS. Our proposed technique consists mainly of two stages: root selection and morphology formation. The final goal of these stages is to enhance the time cost to get new morphology of the traditional SET algorithm thus, ensuring fast self-reconfiguration. The root selection stage selects a small number of modules in order to find the best tree roots which affect the topological conditions that lead to the success of the embedding process or not. The morphology formation stage uses the traditional SET algorithm to calculate the embedding truth table where the initial roots used are taken from the first stage.

Chapter 6: Conclusion and Perspectives: This chapter concludes our work and highlights some aspects of suggested future research work.

Chapter 2

Modular Robotic System (MRS): An Overview

Robotic system have been considered as one of the most important technologies in 21st Century [11]. Recently, engineering inspirations have led to a new generation of robotics called modular robotic systems (MRSs). Compared to conventional robots, the new generation introduces more flexibility and adaptability of its body, and has the ability to change its shape in conformance to the task and the environmental conditions. A modular robotic consists of several units with few degrees of freedom called modules which are usually equipped with connection mechanisms to cooperatively connect to or detach from each other in order to create complex structures. The second chapter in this thesis gives an overview about MRSs. First, we review a number of MRS applications via some existing examples. Then, we describe challenges faced to MRSs while highlighting the self reconfiguration challenge as a real problem for such systems.

2.1 Introduction

Currently, there are some places and regions that humans are not allowed to explore because they are considered too risky. Two of the best examples are deep underwater and space exploration research [12]. As a result, robots are the chosen tools since they can survive a variety of environmental risks and, to a degree, perform tasks that humans can't. Robotic are also used to quickly respond to natural disasters including earthquakes in order to save lives. For example, in order to seek victims, an modular robotic system MRS can change its shape to slip inside ruins and navigate through tiny corridors. When a victim is discovered, the robot can send out a signal indicating its location and transform into a shelter to protect the victim until help arrives [13]. Consequently, MRS have attracted, a significant amount of interest from many researchers as a way of sensing its environment (due to its set of detectors), and then its ability to self-reconfiguring and transforming into a specific shape to adapt to a task or a given situation. Nowadays, Modular robotics has the ability to change the perspective of robotic systems from machines designed to do certain tasks to multi-purpose tools capable of accomplishing almost any task. They are used in a wide range of applications, including reconnaissance, rescue missions, space exploration, military task, etc. Constantly, MRS is built of "modules" from a few to several hundreds or even thousands . Each module involves actuators, sensors, computational, and communicational capabilities. Usually, these systems are homogeneous where all the modules are identical; however, there could be heterogeneous systems that contain different modules to maximize versatility [1].

Modules coordination is critical in self reconfiguration process in MRS. However coordinating these modules is very complicated. This complexity depends on the hardware characteristics of the multiple modules (computation power, communication model, structure organization, motion capabilities, etc.). When it comes to module coordination, communication is essential. Overall network qualities are determined by the communication paradigm and structural organization. Complexities of self reconfiguration algorithms are generally described as a function of network parameter (e.g., number of nodes, number of links, node degree, radius/diameter of the system). Many algorithms are designed to function with a specific network type. In sparse networks, for example, some algorithms are more efficient than in dense networks (e.g, the virtual coordinate-based routing protocol in [14]. therefore, it is critical to consider network features while designing and selecting effective algorithms, particularly in large-scale systems like MRS.

The remainder of this chapter is organized as follows. Section 2.2 introduces a self-reconfigurable modular robotic system and its main objectives. Section 2.3 discuss the main advantages of MRS over traditional robotics. Section 2.4 and 2.5 review a number of MRSs applications via some existing examples. In Section 2.6, we describe the architecture used for MRSs. The main challenges in MRSs are presented in Sections 2.7. We mention in section 2.8 some of well-known simularors for modular robotic systems. Finally, we conclude the chapter in Section 2.9.

2.2 Self-Reconfigurable Modular Robotics: Definition

Modular robotics has evolved as a novel technique to construct robotic systems in recent decades. A modular robot is made up of self-contained, intelligent, and communicative modules that work together as a unit. It creates a distributed system in which modules work together to self-organize, complete tasks, and achieve common goals. Self-Reconfigurable Modular Robots can change their overall shape to adapt to a task or scenario.

2.3 Important Differences From Traditional Robotics

Modular robotics came as an evolution to the traditional robotics where it added many developments in case of mechanisms as well as programming. MSRs have four key advantages over typical robotic systems: versatility, robust- ness, extensibility, and low cost as shown in the Table 2.1. The MSR's versatility attribute stems directly from its ability to self-adapt to a particular, maybe unforeseen environment by reorganizing its global morphology. This allows modules to execute a wide range of tasks, even those that were not even considered during the design process. Modules can be swapped inside a robot, as well as with some external systems. As a result, modular robotic systems are more robust since they can self-repair if a module fails by re- placing the failed module on the spot. Furthermore, modular robotic systems can be scaled up or down as needed. In addition, they also have cost advantages since mass-produced modules may be used to create a wide range of various and sophisticated systems.

Traditional Robotics	Modular Robotics
• Unique solution to each real world application and prototypes that are inflexible for rest of applications	• Unique advantage over traditional robotics in reconfiguration, reusability, and ease in manufacturing
• Lack of adaptive nature if the environment is changed where traditional solutions become inflexible	• Adaptive nature where there is flexibility if the environment is changed
• Repair and maintenance require separate trained personnel for each model	• Development in the perspective of assembly in order to increase the ease of repairing, replacing, and control
• Use integration-oriented way in building	• Using modular way in building

TABLE 2.1: Differences between traditional and modular robotics.

Beside the differences mentioned in the above table, there are also differences based on the programming and manufacturing sides that should be taken into consideration. Figure 2.1 shows the life cycle of the production of modular and traditional robotics in terms of the building side [15].

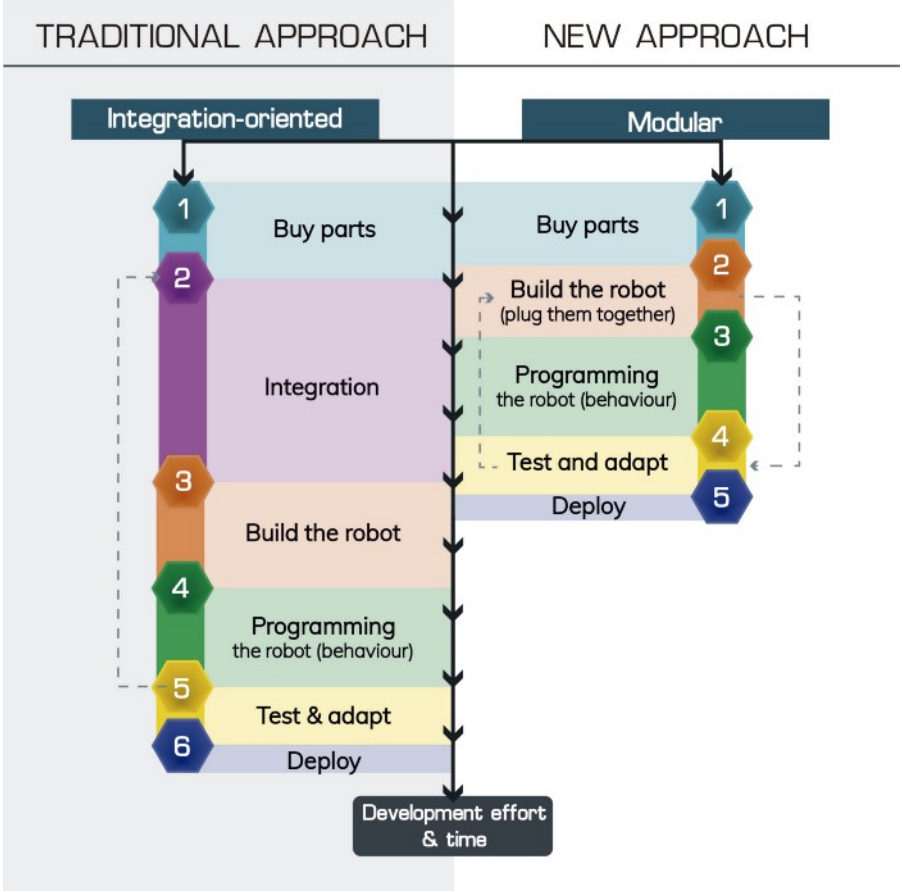


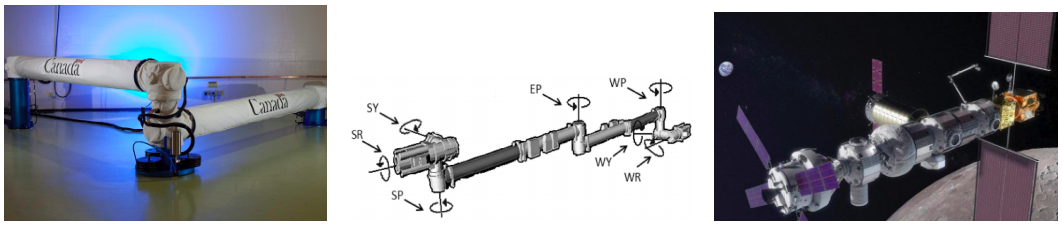
FIGURE 2.1: Comparison between integration-oriented and modular approaches for building robots.

2.4 Applications

Thanks to its capabilities of self reconfiguration, modular robotic systems (MRSs) have attracted significant attention in many applications, such as space exploration, automation, search and rescue, industrial and so forth. In such applications, the main objective of MRS is to change its own shape by rearranging the connectivity of their parts, in order to adapt to new circumstances, perform new tasks, or recover from damage. Next, we give an overview about different MRS applications.

2.4.1 Space Exploration

Robotic technologies have been utilized in space exploration for decades to aid astronauts in orbit and allow our species to discover other planets. Despite this, getting payloads to orbit and beyond is still prohibitively expensive, therefore the more applications a robotic payload has, the easier it is to justify the launch cost. Modular robotics has the potential to help in this aspect, since it provides a platform that can be reconfigured into multiple structures to perform many more jobs than a classical robotic system. Furthermore, the potential of modular robots to deal with failures could be beneficial to space exploration, as robots could self-reconfigure to discard destroyed modules in order to continue on their mission. Despite the fact that such an application may seem far off, a modular robot is already in use on the International Space Station. The Canadarm2, as shown in Figure 2.2, the station's main manipulator, functions as a typical arm for most tasks but can be moved to any location on the station as necessary. Thanks to the symmetric structure of the arm, each end can be docked into one of the many grapple fixtures on the station's outer surface [16], allowing it to move end-over-end from one fixture to the next. Furthermore, as new portions of the space station are delivered and old ones are retired, the arm may be used to reconfigure the various sections of the space station, thereby turning the station into a modular robot.



(a) (b) (c)

FIGURE 2.2: (a) The next-generation Canadarm based on the telescopic link reconfiguration concept (CSA) [2], (b) A 3D representation of Canadarm2 illustrating its seven joints [3], (c) The gateway extravehicular robotic system (GERS) [2]

Figure 2.2

2.4.2 Programmable Matter

A matter that may modify its physical properties in response to external and programmed events is known as programmable matter (PM). In the literature, various strategies and technologies for realizing PM have been proposed, including PM

employing 4D printing [17], quantum wellstone [18], DNA architectures [19], and robotic-based techniques. The latter include the utilization of robotic materials [20], tendon-driven robotic chains [21], swarm robotic systems [22], Self-folding robots [23] and modular self-reconfigurable robots [24,25] .

To construct PM, the Clay-Electronics (Claytronics) project [24] proposes using large-scale micro MRSs made up of millions of Claytronics Atoms (Catoms). Every Catom is a mass-producible tiny robot with very limited (and strictly required) functions. PM promises synthetic reality and has a wide range of uses (e.g., sending/downloading copies of physical items, morpheable objects that can be reshaped at will, injectable surgical devices, 3D interactive life-size TV, and so on). People will be able to shape their surroundings as well as control it.

PM, for example, allows a significant advancement of the computer-aided design process, as shown in Fig. 2.3. In this vision, a computer stores a virtual representation of an object that may be transferred to programmable matter to create a physical version of the object. The virtual and physical representations remain always consistent; if one changes, the other changes as well. The user can alter the virtual representation, which will immediately affect the physical representation of the item in question. He can even adjust the physical representation manually as he desires, which will update the virtual representation immediately [4,11,26]. As a result, designers will be able to design a model and a prototype of their item at the same time, considerably lowering prototype time. Furthermore, because the matter may be reused and molded indefinitely, this approach reduces resource waste.

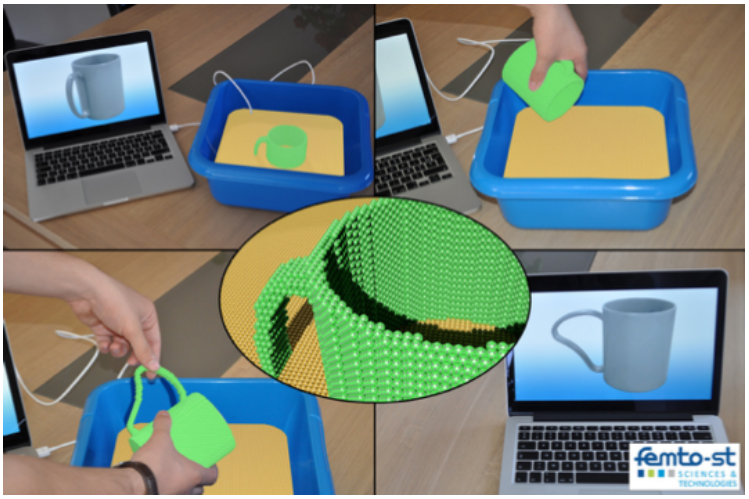


FIGURE 2.3: Computer aided-design tool using programmable matter [4].

2.4.3 Search and Rescue

In search and rescue, the ability of a response team to respond quickly to a crisis is critical for saving lives [27]. In some natural disasters, there is a difficulty in finding humans under destruction. Indeed, during chemical explosions, it is risky for people to look for human evidence or even attempt to repair the damage. As

a solution, the need for modular robotics is a must [13] in search and rescue operations. Hence, AMOEBA-I was developed to help in assisting disaster relief work (Figure 2.4). It has a diversity of sensors aimed to perceive the environment nearby. Specifically, the laser sensors applied to know the obstacles in front of it as well as the installation of GPS system to recognize the exact location of victims. When a victim is found, the robot can broadcast its location and morph into a shelt. It also had a microphone and an IR camera to help in searching for information. Moreover, it has a track mobile modules called caterpillar in order to assure the strong movement ability on the ground and the adaptability of different terrains.

FIGURE 2.4: AMOEBA-I in Ya'an earthquake.

2.4.4 Industrial Production

Today most robots are used in manufacturing operations. Modular robots could have applications in a variety of fields, including reconfigurable manufacturing[28]. The goal of reconfigurable manufacturing is to develop production processes that can adjust to changing product demands more quickly than present manufacturing methods can [29]. By incorporating self-reconfigurability, modular robotics has the potential to take reconfigurable manufacturing to the next level. Having manufacturing systems that can self-reconfigure based on the product to be produced allows for more product customization because the system can adjust its structure to fit for the product's various manufacturing requirements.

2.5 Modular Robotic Prototypes: A Historical View

There are massive enterprises, universities, government sectors, and specialized companies working on modular robotic systems either to ease their works or to upgrade and develop them. Moreover, the number of MRS prototypes are increasing on a daily basis to fit our routine work as well as make it much faster, and easier. In this section, we are going to overview the timeline of prototypes found in the market that range from 1988 until 2016.

Figure 2.5 shows the illustration of a chronogram with each prototype year, name, and image related to this prototype [30–129].

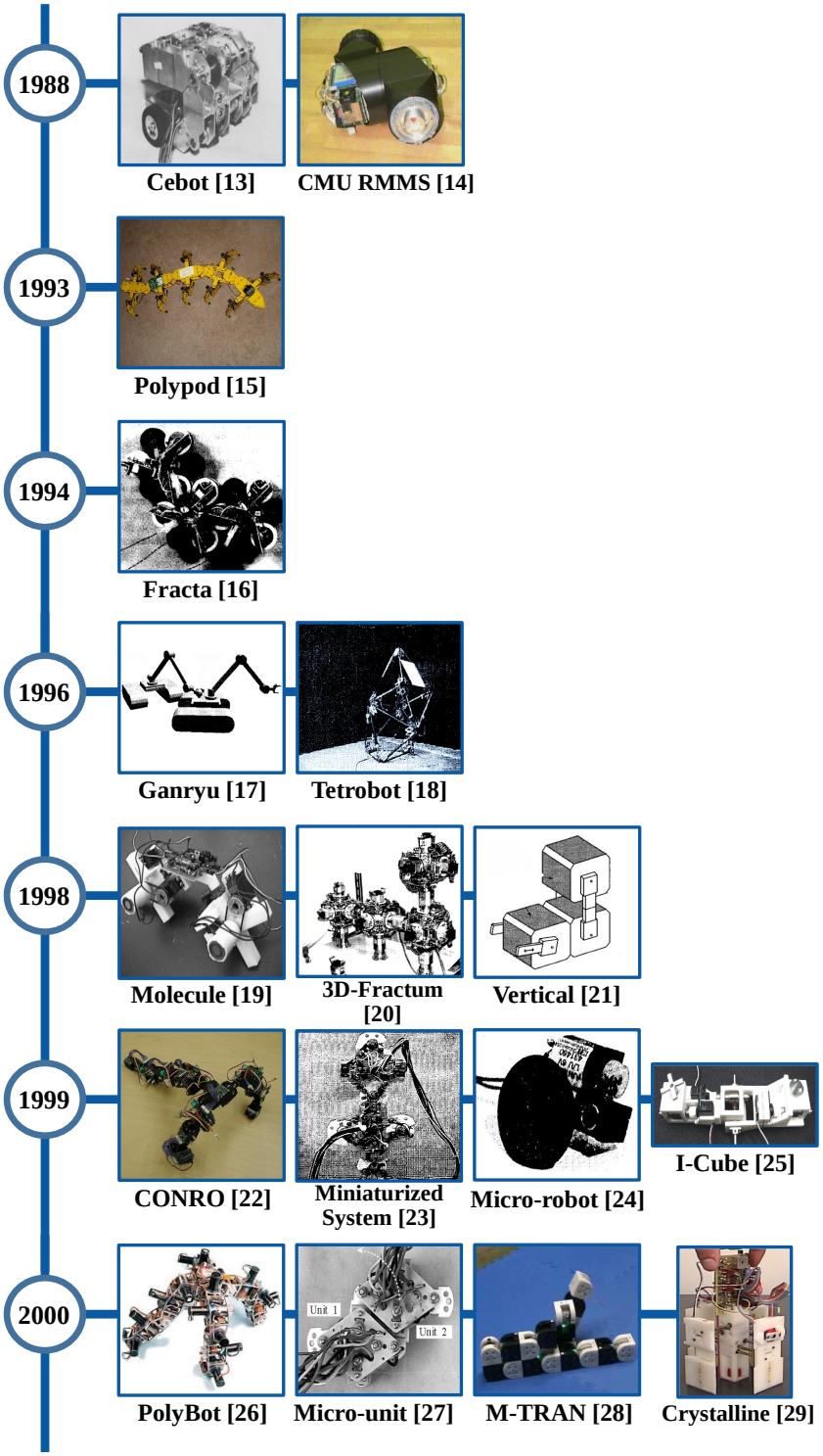
1. **M-TRAN (2000)**
   M-TRAN (Modular Transformer) is a distributed lattice-based self-reconfigurable robotic system that can change into numerous configurations, such as a legged machine that can walk. The actual system was constructed from ten modules and successfully exhibited basic self-reconfiguration and motion creation activities. A set of software programs, including a kinematics simulator, a user interface for creating configurations and motion sequences, and an autonomous motion planner, have been developed to operate M-TRAN hardware [130].

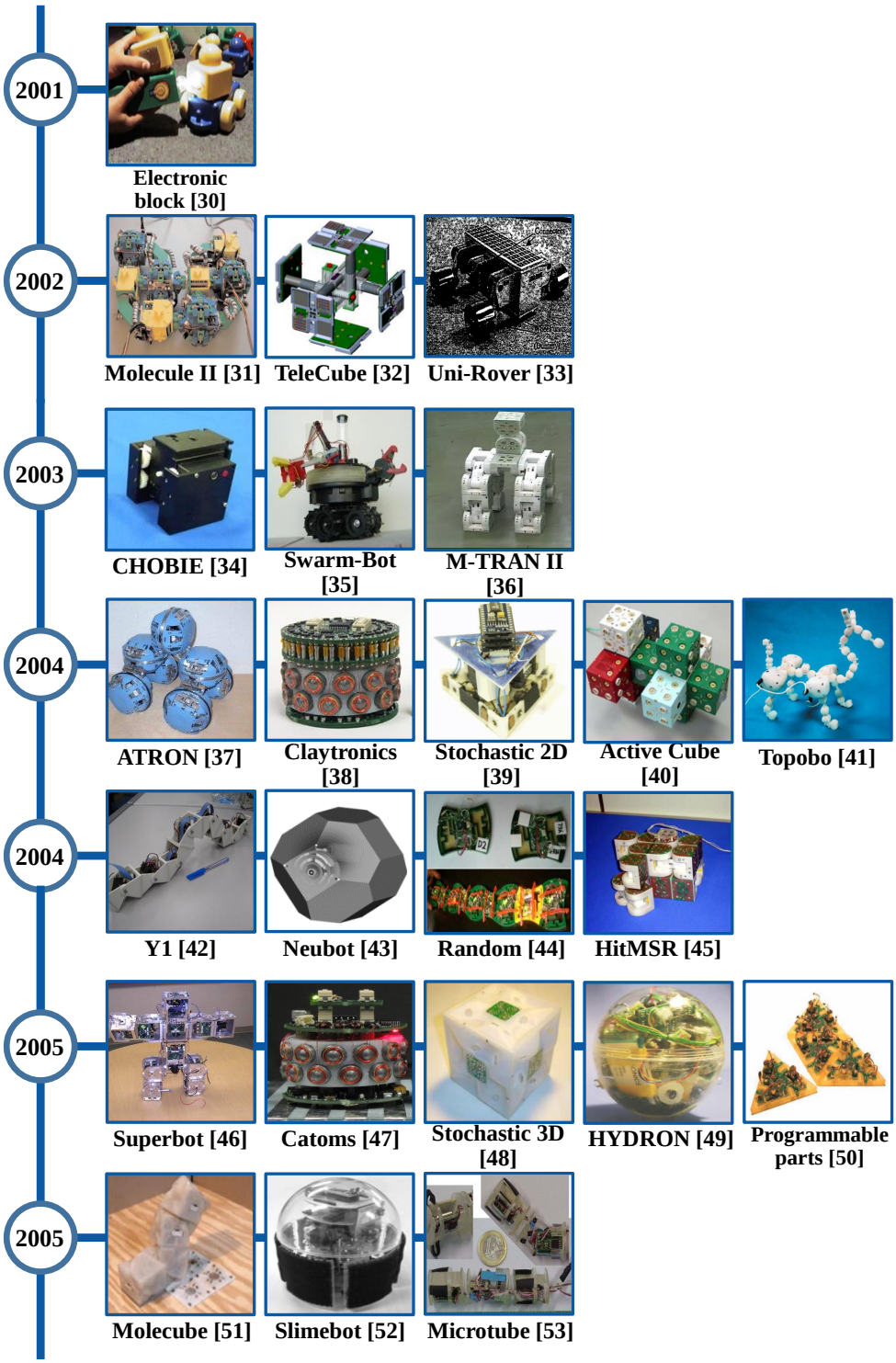
   M-TRAN II is the second prototype, where numerous enhancements done that allow for a wide range of whole-body motions and complex reconfigurations. A reliable connection mechanism, high-speed inter-module communication, on-board multi-computers, precise motor control, and low energy consumption are among the enhancements. Reconfiguration operations have also been improved by the software [131].

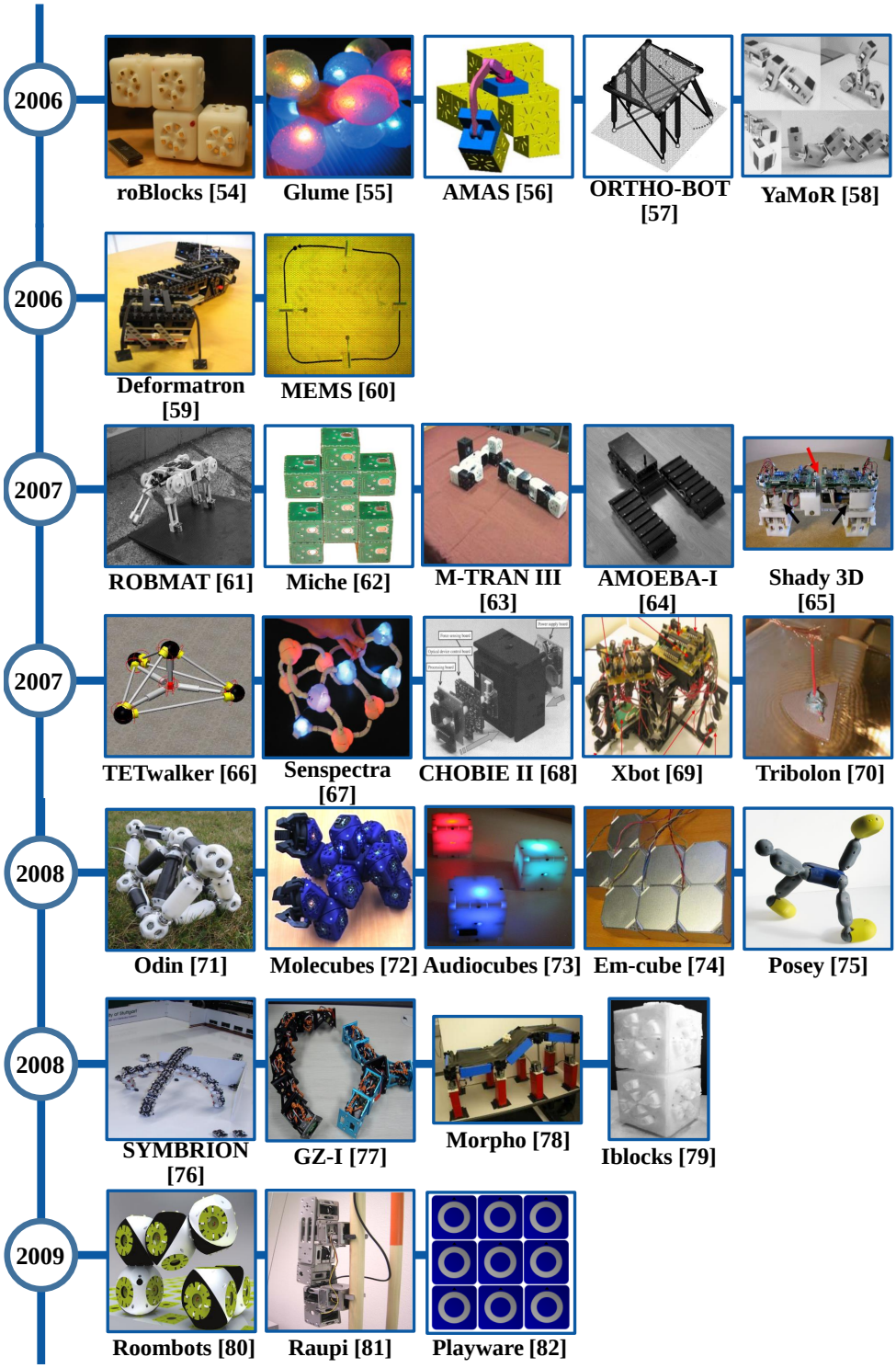
   The third prototype, M-TRAN III, was designed with a better connection method. Single-master, worldwide synchronous control, and parallel asynchronous control are some of the control styles that have been employed with distributed control. Self-reconfiguration experimients with up to 24 units were conducted out using centralized and decentralized control. The system scalability and homogeneity were maintained in all experiments [132].

2. **PolyBot (2000)**
   7 PolyBot is a self-configurable modular robot that was created to see how feasible it is to build robots out of numerous homogeneous hardware modules. PolyBot modules were prototyped in three generations, each of which addressed a number of vulnerabilities uncovered in the preceding one. The first generation (G1) is made up of two sorts of modules: node and segment. The segment modules are rectangular prisms with one rotating degree of freedom (DOF) separating the two connection ports. The node modules are fixed passive cubes with 6 connection ports. The second generation (G2) connection ports, unlike their ancestors, include electromechanical latches that are controlled by software. These hook into the projecting pins on the other side. To improve performance, the third generation (G3) modules are smaller, while the connectors are larger and have a higher contact force for higher current loads.

   PolyBot's first two generations demonstrate its versatility by traversing a variety of terrains. PolyBot can self-reconfigure by changing its morphology and locomotion mode based on the terrain type — rolling over flat terrain, moving around obstacles like an earthworm, and stepping over mountainous terrain like a spider. Because the dimension of this space is exponential in the number of modules but proportionate to the number of DOF, planning self-collision-free motions might be difficult. A defined set of configurations is sufficient for many applications. In this instance, reconfigurations can be planned off-line and stored in a table for ease of reconfiguration [133].


1988
Cebot [13]
CMU RMMS [14]
1993
Polypod [15]
1994
Fracta [16]
1996
Ganryu [17]
Tetrobot [18]
1998
Molecule [19]
3D-Fractum [20]
Vertical [21]
1999
CONRO [22]
Miniaturized System [23]
Micro-robot [24]
I-Cube [25]
2000
PolyBot [26]
Micro-unit [27]
M-TRAN [28]
Crystalline [29]


2001
Electronic block [30]
2002
Molecule II [31]
TeleCube [32]
Uni-Rover [33]
2003
CHOBIE [34]
Swarm-Bot [35]
M-TRAN II [36]
2004
ATRON [37]
Claytronics [38]
Stochastic 2D [39]
Active Cube [40]
Topobo [41]
2004
Y1 [42]
Neubot [43]
Random [44]
HitMSR [45]
2005
Superbot [46]
Catoms [47]
Stochastic 3D [48]
HYDRON [49]
Programmable parts [50]
2005
Molecube [51]
Slimebot [52]
Microtube [53]


2006
roBlocks [54]
Glume [55]
AMAS [56]
ORTHO-BOT [57]
YaMoR [58]
2006
Deformatron [59]
MEMS [60]
2007
ROBMAT [61]
Miche [62]
M-TRAN III [63]
AMOEBA-I [64]
Shady 3D [65]
2007
TETwalker [66]
Senspectra [67]
CHOBIE II [68]
Xbot [69]
Tribolon [70]
2008
Odin [71]
Molecubes [72]
Audiocubes [73]
Em-cube [74]
Posey [75]
2008
SYMBRION [76]
GZ-I [77]
Morpho [78]
Iblocks [79]
2009
Roombots [80]
Raupi [81]
Playware [82]

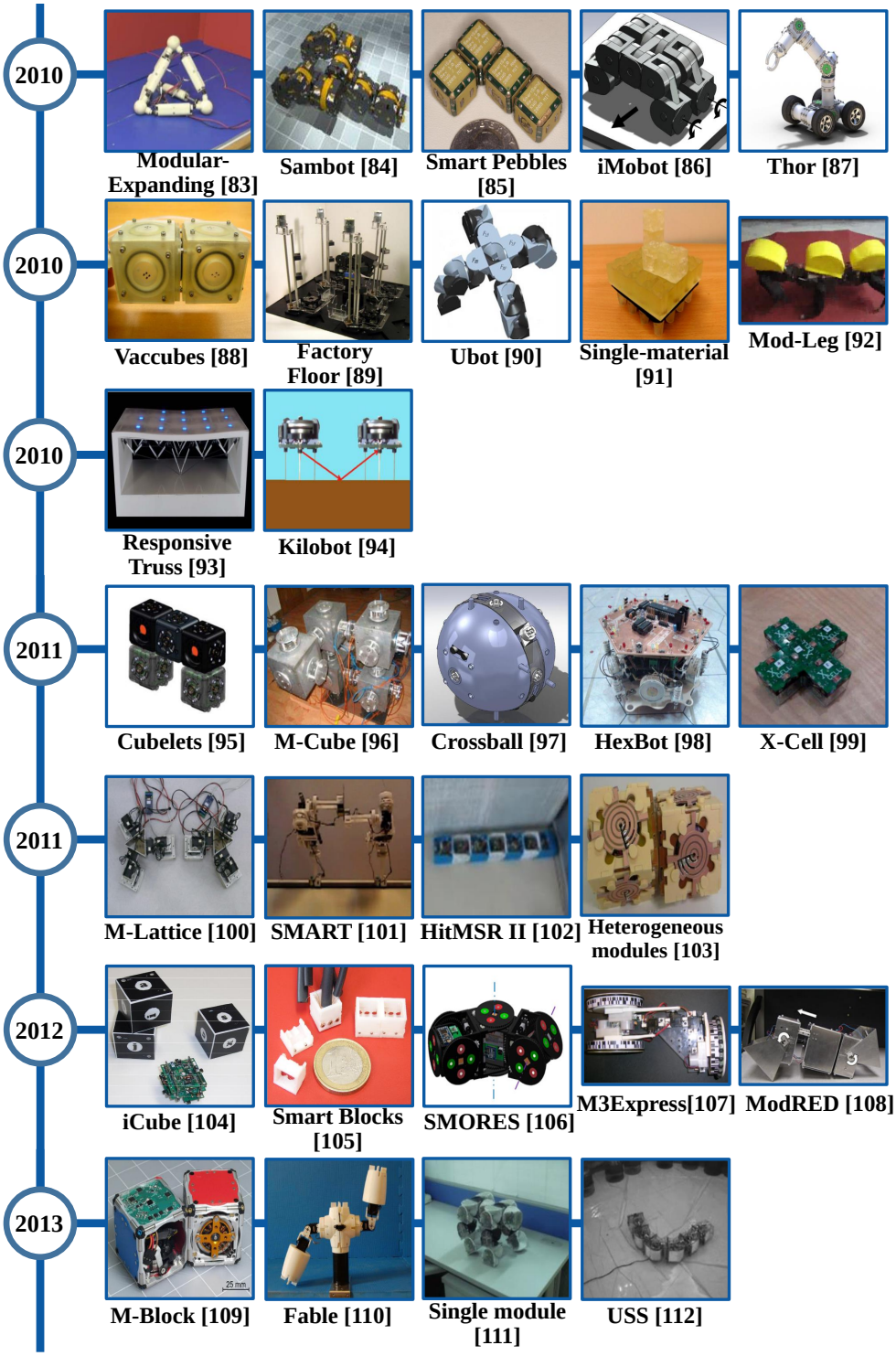


FIGURE 2.5: Modular robotic prototype timeline.

3. **Molecubes (2005)**
   The Molecubes system is a modular robotics open hardware and software platform that was created to lower entry barriers and speed progress. The system is made up of modules with one rotational degree of freedom (DOF). There were a variety of active modules on display, including grippers, actuated joints, controllers, cameras, and wheels, as well as a number of passive modules. Each module is a cube with round corners that is made up of about two triangular pyramidal pieces that are joined at their bases so that their main axes are aligned. Each of the module's six sides contains an electromechanical connector that may be used to link two modules. Symmetric connector design enables for four alternative relative orientations of two connected module interfaces, each of which results in different robot kinematics [134].

4. **ATRON (2004)**
   Another modular self-reconfigurable robot is ATRON [135], a lattice-based system made up of roughly spherical modules connected by an endless revolute join, Fig.2.5. Actuation is accomplished by rotating each module 360 degrees around the equator around an axis running diagonally across the sphere. Because a relatively big area is accessible for mechanics, this design provides for a fairly stable construction surrounding the actuated joint. The spherical basic module design, on the other hand, makes connecting huge flat surfaces difficult. Connectors for spherical modules must provide critical point-to-point interactions between modules, which are undesirable due to the high likelihood of collision. Because of ATRON's restricted mobility and other motion limits, the ATRON meta-module is used to reduce motion constraints. The meta-module is made up of three modules: a body in the center and two legs on either side.

   Modular ATRON control comprises three Artificial Neural Networks; the first one to decide when to emerge, the second to decide when to stop, and the third to calculate the fitness value of every state in the self-reconfiguration and self- repair processes. Genetic Algorithm is used to improve the weights of the ANNs. Despite the fact that ATRON modules have only one actuated DOF, a group of them proved capable of self-reconfiguring in 3D simulation. Similarly, ATRON modules successfully exhibited self-repair in simulation [135,136].

2.6 MRS Architecture

Because of their ability to self-reconfigure, modular robotics has recently attracted the interest of robotics experts [137]. Modular self-reconfigurable robots are made up of several modules that can combine themselves autonomously into meta-modules capable of executing a variety of jobs in a variety of situations [138]. These metamorphosis robots have the ability to self-reconfigure, allowing them to perform various kinematics. Based on how reconfigurable robots reconfigure, Yim et al. divided them into three architecture types in 2002: lattice, chain, and hybrid [139]. After that, in 2007, they introduced both deterministic and stochastic reconfigurations [140].

2.6.1 Lattice Architectures :

Modules in lattice structures are placed in a 2D or 3D grid that can be used as a guide for selecting module places and constructing the new design [140]. All modules are connected to the main body to facilitate planning and control. Lattice structures can also be easier to reconfigure than other classes because control and motion can be carried out in simultaneously [137]. Lattice-type systems use the regularity of the lattice to align connectors during self-reconfiguration, making self-reconfiguration faster and easier. When aligning connectors during self-reconfiguration, lattice-type systems take advantage of lattice regularity to make the process faster and easier. For systems with a large number of modules, however, presuming that all modules follow the lattice can be difficult [141]. M-TRAN is an example of a lattice-based self-reconfigurable robot [142,143].

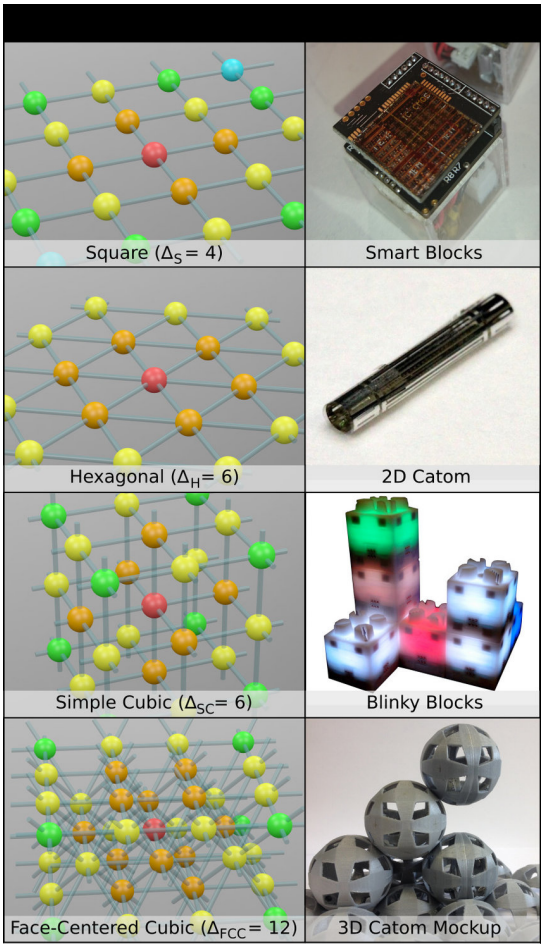


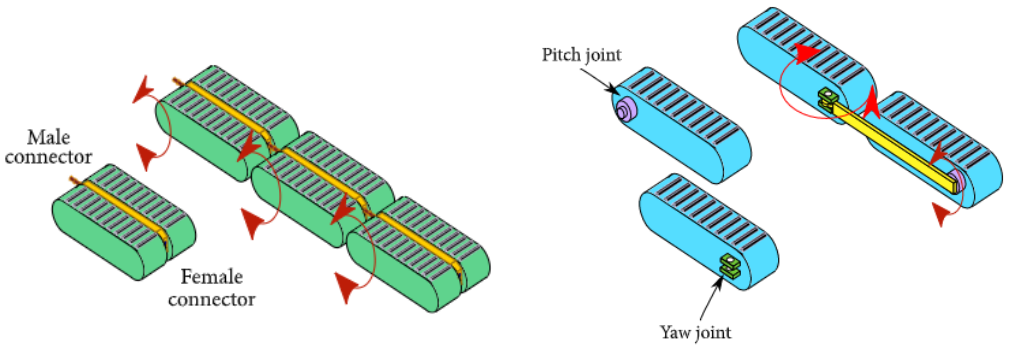
FIGURE 2.6: Different lattice arrangements associated with modular robotic systems developed in the Smart Blocks and the Claytronics projects [5]

2.6.2 Chain Architectures :

In a chain architecture, modules are connected in a string or tree structure (Figure 2.7). There are many chain-structured systems that are related to MRS, such as CEBOT, which is a mobility category with heterogeneous modules and 2 hardware

prototypes known as series 1 and series 2, ACM, which resembles a snake structure and has 3 different versions, Millibot, which is a 2D structured for climbing on stairs and uneven terrains, Amoeba-I (Figure 2.7(b)) which is a self-mobility tracked MRS, JL 1 and JL 2 are a developed versions of Millibot (Figure 2.7(a)). Each chain is always connected to the rest of the modules at one or more locations due to the serial underlying architecture, and modules reconfigure by connecting and detaching from each other. Robotic limbs, legs, and tentacles can all be made out of the chains [140]. Chain architectures are more adaptable than other systems because they can articulate to any point in space, but they are more difficult to control and show and analyze computationally [137]. PolyBot is an example of a self-reconfigurable robot built on chains [144–146].

It's worth mentioning that lattice and chain topologies aren't usually mutually exclusive, and many systems, like SuperBot [147] and UBot [148], can have both [139].



(a) Millibot (b) Amoeba

FIGURE 2.7: Chain hardware model.

2.6.3 Hybrid Architectures :

it provides more advantages compared to lattice and chain robotic structures due to their capabilities in easy adaptation to surroundings by forming both lattice, chained and mixture of both. Some of Hybrid structured systems related to MRS are as follow: S-BOT [149–151] which is a hybrid category because of the combination between lattice 2D structure and chain 3D structure, M3 is a combination of 3D chain and lattice which also has 2 version of M3 and M3 Express, Imobot is cuboid structured made of two semi cylindrical modules, SMORES is basically similar to iMobot but i has a singular semi cylindrical cubic structure, Trimobot is an integrated mobile category hexagonal which is made of 2D lattice structures and 3D chain structures, M-Tran (Figure 2.8) is has 3D structures of both lattice and chain which also has 3 versions, M-Tran I, M-Tran II, and M-Tran III, Superbot module is made of bonding semi cylindrical cells like iMobot, The CKbot is like SMORES but has reduction in self-mobility as well as rolling capabilities in individual units, Molecubes are UBot both are hybrid cubic structured, Roombots is a hybrid made of 3D structures of lattice and chain, and Soldercubesis individualized Soldercube just

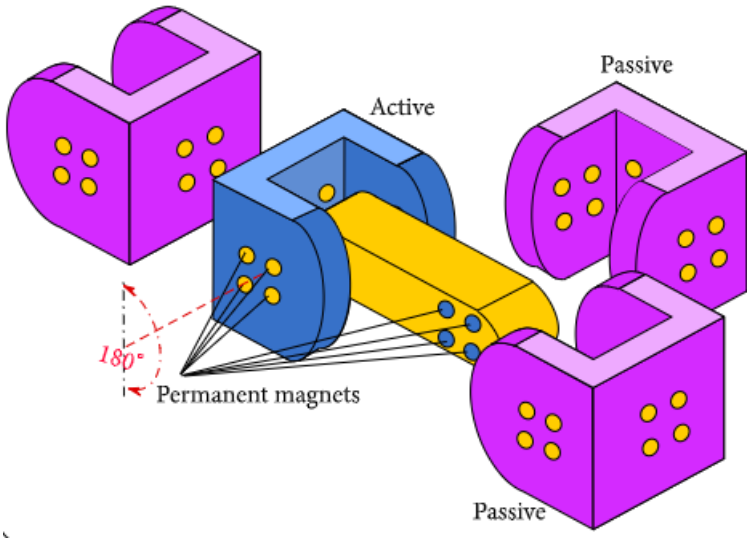


FIGURE 2.8: A hybrid structured system, M-Tran.

like a cell in dual-cell structure of roombots.

2.6.4 Truss:

it supports formation of random structures due to the employment of telescopic links and heterogeneous units for forming structures but require complex algorithms for handling assembly and formation of structures. Some of Truss based structured systems include tetrobot which uses heterogeneous units like links and Joints in order to form random structures, ORTHOBOT that has telescopic links with split toroids and with one toroid that is connected to link using revolute joint, Odin has heterogeneous units which are Cubic Closed Packe joints and telescopic links as well as the capability to form 3D structures (Figure 2.9), Morpho is made of passive and active links and well as joints, Hing is used for reconfiguration of Truss structures, and factory floor is used for auto assembly of truss structured systems.

FIGURE 2.9: A truss structured system, Odin.

2.7 Modular Robotic System Challenges

The development of effective Modular Robotic Systems (MRS) is loaded with hardware and software problems. In this section, we describe the main challenges involved by MRS (some of them can be common to the traditional RS). They must be able to (1) reconfigure rapidly, (2) be sufficiently robust and fault-tolerant, (3) be scalable to many-module configurations, (4) be reliable in constructing solid and robust structures, (5) exhibit self-reconfiguration for an extended period of time, (6) deal with uncertainty in the environment and sensed data, (7) handle communication unreliability among modules, and (8) deal with the limitations of mechatronic devices in terms of resources

2.7.1 Sensing Analysis Challenge:

One of the key processes of MRS is sensing, or perception. It enables the direct acquisition of internal or exterior inputs through a variety of sensors. Unfortunately, the continuous monitoring of the environment, as well as the large number of sensors in use, makes decision-making in the controller a complicated task. Modular robots, on the other hand, have limited processing power and so are unable to accept all data streams from sensors. As a result, developing novel process and sensing strategies has become a critical task that must be addressed in MRS, based on the robot's mission.

2.7.2 Self-Reconfiguration Challenge:

The process by which an MRS transforms itself from an initial configuration to a desired configuration is known as self-reconfiguration. Creating a self-reconfiguring robot systems poses many engineering challenges. These challenges are centered around designing the basic self-reconfiguring module and the inter-module connections, and aggregating distributed systems from these modules. This method can be used in a variety of application. It allows an MRS to take on different shapes in the context of programmable matter. Self-reconfiguration can also be used to adapt an MRS to environmental changes or specialized tasks. For example, the authors of [152] employ self-reconfiguration to reorganize module connectivity in order to achieve an ideal network topology. Self-reconfiguration brings with it a slew of software difficulties. Self-reconfiguration raises a number of software challenges. To begin with, planning is difficult due to the large number of possible unique configurations: $(x \times t)m$, where m is the number of modules, x is the number of possible connections per module, and t is the number of ways to connect the modules together [153]. Modules can often move simultanously depending on physical restrictions, causing the configuration space to grow at the rate of $O(vm)$, where v is the number of possible movements and m is the number of modules free to move [154]. The exploration space for reconfiguration between two random configurations is therefore exponential in the number of modules, which prevents us from finding a complete optimal planning for all but the simplest configurations. For chain-type MSRs, the optimal self-reconfiguration planning is then NP-complete issue [155], and nothing has been proven for lattice-based MSRs to our knowledge. Second, aside from the path planning difficulty, the self-reconfiguration process is

difficult because it is a distributed process that necessitates the distributed coordination of mobile autonomous modules connected in time-varying ways. Modules, in particular, must coordinate their movements to avoid colliding with one another. Self-reconfiguration algorithms are frequently adapted for a certain class of modular robots, each with its own set of motion constraints.

2.7.3 Time Synchronization:

Time synchronization is a significant and costly challenge in modular robotic system. Coordination among a group of modules often relies on availability of a shared concept of time. Every module has its own concept of time provided by its own hardware clock. Local clocks tend to run at somewhat distinct and dynamic frequencies, sliding apart from one other over time due to the imperfections of common hardware clocks. As a result, time synchronization is required to keep each module's local clock in sync. There are several techniques to time synchronization (continuous vs on-demand, network-wide vs clustering, timescale transformation vs clock synchronization,etc.) [156]. The approach to take is determined by the target application. Nodes in the continuous model seek to maintain synchronization at all times. This method is in contrast to the on-demand synchronization model, in which nodes can either a posteriori agree on the time of an event or predict synchronizations in order to initiate coordinated actions at a specific moment.

2.7.4 Battery Power:

The major challenge that faces MRS is improving the lifetime of the robot; in other words ensuring long-time functionality for a given task [157]. Indeed, the robotic lifetime is highly related to the power consumption during the module transition or the transmission of data during self reconfiguration process. In addition, it is difficult and cost-ineffective to recharge the robot's batteries in most cases, especially in harsh or hostile environments. Therefore, one of the major objectives today is to efficiently manage the robot's energy in order to increase the system lifetime. Hence, the module must minimize its data transmission and reduce the number of transitions during the self-configuration, in order to save its available energy.

2.7.5 Number of Modules:

A modular robotic consists of several units with few degrees of freedom called modules which are usually equipped with connection mechanisms to cooperatively connect to or detach from each other in order to create complex structures. Obviously, transforming a bunch of uniform modules into a versatile robot is not a simple task. To put together a useful system, solutions to the challenges of programming a large number of coupled but autonomous robotic units must be discovered. Worse, as more modules are added, many of the programming issues get exponentially harder. These include controlling, communication and coordinating modules to work together effectively and not collide or otherwise interfere with each other.

2.8 Common Modular Robotic Simulators

Nowadays, there is a lot of simulators known for their efficiency in certain ways and methods as well as their inefficiency in some points. We mentioned in the Table 2.2 some of well-known simulators in the market mentioning their names, simulated

modules, operating system, and programming languages. In that table, we can understand the difference between the mentioned simulators.

Simulator	Simulated Modules	Operating System	Programming Language
M-TRAN [158]	M-TRAN	Windows, Linux	C++
Adam [159]	Hinge-like	Windows, Linux, Mac OS	C++
Webots [160]	YaMor, M-TRAN, RoomBots	Windows, Linux, Mac OS	C++, Java, Python, Matlab
USSR [161]	ATRON, M-TRAN, Odin	Windows, Linux	Java, C
CubeInterface [162]	Molecube	Windows	C++
Symbricator3D [163]	SYMBRON, Replicator	Linux	C++
OpenMR [164]	Y1	Windows, Linux	C++, Python
ReMod3D [165]	M-TRAN, ATRON	Windows, Linux	VC++
Micromult [166]	Chain-like	Windows	VC++

TABLE 2.2: Common MRS simulators.

2.9 Conclusion

In this chapter, we introduced modular robotic system in which self reconfiguring robots are able to deliberately change their own shape by rearranging their connectivity of their parts, in order to adapt to new circumstances perform new tasks, or recover from damage. Then, we presented some examples of MRS applications including space exploration, programmable matter, industrial and search and rescue fields. The MRS architectures is also described in this chapter. Finally, we have presented challenges that face MRSs while highlighting the self reconfiguration as the primary challenge to be optimized in order to ensure a fast self-reconfiguration process. Next, we present in more details our proposed techniques for data processing and self- reconfigurable decision making dedicated to modular robotic systems (MRS).

Chapter 3

Sensing-Based Self-Reconfigurable Decision-Making Mechanism for MRS

Nowadays, robotic technology finds its way quickly across industries affecting business and people's lives. In addition, the rapid growth in communication technologies has lead to a new generation of robotics called as Modular Robotic System (MRS). Basically, MRS is an autonomous kinematic machine with variable morphology, structure, and functionality. The detectors represent the eyes of the MRS that collect data about different environments and states, while the controller forms the brain of the MRS that must analyze the collected data and take decision about the suitable reconfiguration morphology. However, the huge number of data sensed by the detectors provides two main challenges for MRS; first, it overloads the limited storage of the MRS and, second, it complicates the self-reconfiguration decision required to change its shape according to the monitored environment. In this chapter, we propose a sensing-based processing mechanism for data storage and decision making in modular robotic systems. Our mechanism consists of two phases: data storage reduction and self-reconfiguration. The first phase uses aggregation process in order to eliminate redundant data collected thus, reduces the amount of data needed to be stored in MRS. The second phase uses the fuzzy logic model and allows MRS to be self-reconfigurable by taking the right decision about the desired shape.

3.1 Introduction

From the beginning of 1980's, the robotic industry has witnessed a boom in the development and creating millions of robotics with various types and missions. These robots change our daily life and help to make our work more effective and faster. In addition, the fast growth in technologies area emerged in the new century was moving towards a new robotic world: Modular robotic system (MRS). One of the main advantages of MRS is to self-reconfigure its shape - known as *morphology* - according to the monitored environments through its own intelligent system to achieve the established functional goals. Recently, MRS has taken a great attention in a huge number of applications include reconnaissance, rescue missions, space exploration, military task, etc. In almost such applications, modular robotic has constantly changed the traditional production mode of people, which greatly improving human productivity and reducing human production risks [167].

Typically, a modular robotic system (MRS) consists of a set of units, called modules, where each module has a few degrees of freedom (DOFs). Usually, the DOFs are used to define the motion capabilities of robots; more the DOFs increase more the flexibility in changing the MRS morphology is. Subsequently, the MRS monitors the target environments thanks to a set of detector devices attached on its body. These detectors capture the readings of the MRS surroundings and send them toward the system controller (SC). In its turn, the SC analyses the collected data and decides about the next morphology must be taken by the MRS then, it sends signals to the actuators to reshape the MRS morphology. Indeed, this self-reconfiguration process offered by MRS imposes two main challenges: the big data collected by the detectors and the self-reconfiguration itself. From one hand, and for reliability purposes, the detectors should continuously collect data about its surroundings in order to detect any possible variation then, to quickly decide about the new morphology. This leads to a huge number of collected data that must be saved in the MRS storage for a later analysis. Indeed, such data is almost redundant and useless, and it consumes the storage space of the robot. Hence, designing new data reduction storage techniques is essential for MRS in order to avoid the overloading of its storage space. On the other hand, and after data acquisition, the MRS should be able to analyze the collected data in a real-time manner and select the suitable morphology among a set of morphologies defined during its creation. This self-reconfiguration process is a crucial process in MRS because it leads sometimes to reshape the robot into a wrong morphology thus, destruction of the MRS itself. Therefore, proposing efficient data decision techniques takes a great attention from researchers and industries.

In this chapter, we propose an efficient mechanism for data processing and decision making in robotic systems. The proposed mechanism is mainly based on two phases: the first one, called data storage reduction, uses aggregation method in order to remove redundancy among data collected by the detectors, thus reduce the data storage needed. The second phase, called self-reconfiguration decision-making, is based on the fuzzy logic model and aims to analyze the real-time data collected then, it allows MRS to be reconfigured to the suitable morphology. We simulate our mechanism according to a proposed MRS scenario with real sensor data.

The remainder of this chapter is organized as follows. section 3.2 gives an overview about the existing techniques in sensing mechanisms and self-reconfiguration techniques used in MRS. Section 3.3 introduces the MRS design while overviewing real projects existing in MRS. Sections 3.4 and 3.5 respectively detail the data storage reduction and self-reconfiguration decision-making phases proposed in our mechanism. The system demonstration and the results are presented in Section 3.6. Finally, Section 3.7 concludes our paper and gives some perspectives.

3.2 Sensing Data Analysis and Self-Reconfiguration: A Background

In the literature, we can find various techniques related to sensing data analysis and self-reconfiguration proposed for MRS [168–170]. The objective behind the first approach, e.g. sensing data analysis, is to eliminate the redundancy among the data sensed by the detectors thus, remove the useless data and save the storage space.

Whilst, the self-reconfiguration approach allows MRS to reshape to the proper morphology according to the dynamicity of the monitored environments. Recently, the authors of [9,171] give an overview about different algorithms proposed for modeling, control, sensing analysis and self-reconfiguration in MRS.

From one hand, the authors of [172–180] have proposed techniques for data management (collection, reduction and analysis) in MRS. In [172], the authors propose a real-time data sensing framework for people tracking using mobile robots. The framework uses laser detectors to collect and merge data of several detectors using data association and Kalman filtering. Then, the framework uses Hidden Markov model in order to convert data into qualitative spatial relations thus, the robot will detect humans and decide about the next step. The authors in [178] propose a distributed and compressed sensing algorithm for the communication in robotic networks. The objective of the proposed algorithm is to reduce the data traffic among robots using a compressed sensing, e.g. sum of scalar values, and to avoid collision with obstacles and between each other. In [179], a data aggregation technique based on the compressed sensing has been proposed in order to reduce the energy consumption in wireless sensor networks (WSN) as well as recovery of the data fidelity. The proposed aggregated scheme uses the diffusion wavelets to find a sparse basis characterizing the spatio-temporal correlations between sensor nodes. Finally, the authors in [181] propose an efficient and robust compression method named Sequential Lossless Entropy Compression (S-LEC). S-LEC uses a differential predictor that arranges the alphabet of integer residues into a number of groups. For each group, two codes are assigned: entropy and binary codes. The first code specifies the group where the second one represents the index inside the group.

On the other hand, a lot of techniques have been proposed for self-reconfiguration problem in MRS [182–187]. In [182], the authors have proposed a distributed scaffold-based self-reconfiguration technique for MRS which is based on three models: scaffold, local rules and coordination. First, the authors define the goal and the shape of scaffold followed by a deterministic method to build the shape, based on a sandbox of modules, and finally they generalize their technique into a set of self-reconfiguration schemes that can be adapted to several structures. The authors of [183] propose a reconfiguration framework applied at SuperBot-style modules. The idea behind the proposed reconfiguration process is to the cube-shaped modules into a set of module's kinematic model in the form of a transition function. Then, the reconfiguration process is divided into two levels: The top level that allows to move a single module in the robot based on the Markov decision and the distributed dynamic programming; and, the lower level that accepts a transition function for the kinematic model of the chosen module type as input. In [184], the authors propose a self-reconfiguration technique for underwater MRS that consists of a cluster of connected modular vehicles. The proposed technique uses Theta* algorithm adapted to energy heuristics, with low restoring forces, in order to compute the order of vehicle movements to change from a start to an end morphology. Finally, a multiple attribute decision making (MADM) methodology for self-reconfiguration MRS has been proposed in [185]. MADM uses 86 attributes of MRS and provides a preference technique for coding, evaluation, comparison raking for a best self-reconfiguration of a robot.

Unfortunately, although the proposed techniques and strategies carry many advantages and solve data storage and self-reconfiguration problem in MRS, but almost all have several drawbacks: 1) complexity which is not suitable to limited resources of MRS; 2) they are interested either in data storage reduction or self-reconfiguration but not both; 3) the proposed techniques are mostly dedicated to

one type of robot but cannot be generalized to all ones. In this chapter, we propose an efficient and less complex data storage and self-reconfiguration mechanism suitable for low resources robotic systems.

3.3 MRS Design

Indeed, the efficiency of any MRS is highly dependent on its ability to adapt its morphology to the variation of the monitored environment. Hence, the DoF of the modules plays a vital role in the designing of any MRS. In this section, our objective is to first formulate the design of a MRS then we present real projects in MRS that show the efficiency of design process in such system

3.3.1 MRS Notation

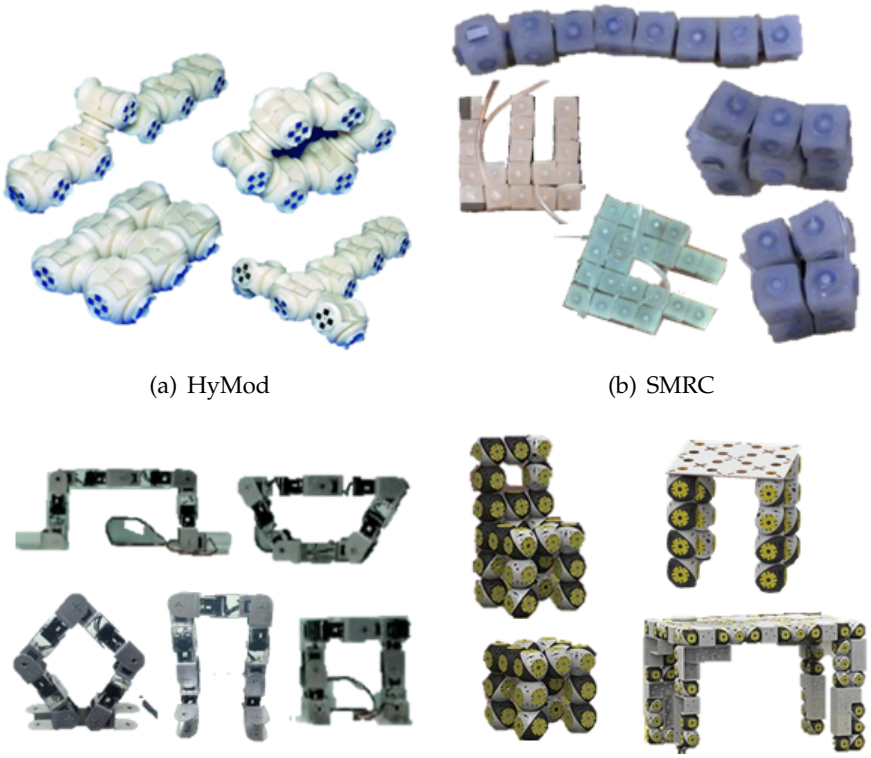
Typically, a modular robotic system consists of a set of μ modules, e.g. $\mathcal{M} = \{M_1, M_2, \ldots, M_\mu\}$, that are connected together, using connection mechanisms, in order to cooperate and create the shape of the MRS, e.g. morphology. We define the set of morphologies that a MRS can be configured to as follows: $\mathcal{O} = \{O_1, O_2, \ldots, O_\theta\}$, where θ is the number of possible morphologies of a MRS. For instance, $O_i \in \mathcal{O}$ could be a snack to crawl into restricted tunnels, a boat to cross a river, a hexapod to explore harsh terrains, or a circle to traverse flat territories. Subsequently, each module $M_i \in \mathcal{M}$ has some degrees of freedom $\mathcal{F}_i$ in order to allows MRS configuration from one morphology to another. $\mathcal{F}_i$ can be represented as a set of actions, $\{F_1, F_2, \ldots, F_\alpha\}$, where α is the degree of freedom of the module and $F_k \in \mathcal{F}_i$ could be mostly a rotate or a move (to left, to right, to top, to down) action. Finally, a MRS can have a set of β detectors, e.g. $\mathcal{D} = \{D_1, D_2, \ldots, D_\beta\}$, where each of them allows to monitor the changing variation of one condition of the monitored environment then to reshape accordingly.

3.3.2 Real MRS Designs

In MRS, one can find a huge number of robots that have been designed to various types of applications. In order to show the efficiency of MRS design, our objective, in this section, is to introduce some of the most real projects in MRS proposed recently in the literature, while focusing on their designs.

In [188], a HyMod robotic system is designed which consists of 6 modules. Each module has 3 rotational degrees of freedom ranging from -90 to $+90$ degrees. HyMod can take 4 morphologies (snake, wheeled vehicle, loop and crawler) and it can be used in manipulator arm and omni-directional rover (Figure 3.1(a)). In [189], the authors propose a soft modular robotic cubes (SMRC) which can be deployed in terrestrial and underwater zones. In the terrestrial zones, SMRC uses 8 modules to form 3 possible morphologies: worm, $2 \times 2 \times 2$ cube and two legged machine. Whilst, in the submerged water, SMRC uses more number of modules (from 8 to 24) and generates several morphologies such as C-shape, T-shape, E-shape, etc. (Figure 3.1(b)). Indeed, the modules in both zones have 4 degrees of freedom (move to left, right, top, down). In [190], the authors design ReBis robot, e.g. Reconfigurable Bipedal Snake, which has the ability to configure to several walking morphologies like square, rectangle, diamond, polygon, etc. (Figure 3.1(c)). ReBis consists of 8 modules with a single degree of freedom that revolute to ± 120 degrees. In [191], the authors propose a self-reconfigurable modular robotic system called Roombots dedicated to adaptive furniture. Roombots can be transformed to many furniture

morphologies like chairs, tables, stools (Figure 3.1(d)). To show its relevance, the authors create a Roombot consisting in 4 modules with 6 degrees of freedom to perform both oscillatory movements and continuous rotation.



(a) HyMod (b) SMRC

(c) ReBiS (d) Roombots

FIGURE 3.1: Real projects in MRS with multiple morphologies.

3.4 Data Storage Reduction Phase

In MRS, the main objective of the robot is to adapt itself to variation of its environments. Hence, a MRS uses a set of detectors in order to sense its surroundings, locate its own position, and then transform to a specific shape to perform the required tasks [192,193]. Indeed, the continuous monitoring of the environment leads to a big data collection problem that: 1) overloads the memory storage of the robot; 2) generates redundant data which complicates the decision making in the controller. Therefore, in order to overcome these challenges, the first phase in our mechanism, e.g. data storage reduction phase, aims to remove the redundancy exiting among the collected data thus, save the storage space and facilitate data analysis.

3.4.1 Periodic Sensing Detection Model

In order to detect any change of its surrounding, detectors in MRS have to periodically sense the environment and send the collected data to the controller for later analysis. In such periodic model, we assume that each detector $D_i \in \mathcal{D}$ allows MRS to monitor one environment condition during a period p then it forms a vector R_i of τ readings as follows: $R_i = [r_{i_1}, r_{i_2}, \ldots, r_{i_\tau}]$. Therefore, during each period p,

the detectors $\mathcal{D}$ of a MRS will form a matrix of readings with dimensions $\beta \times \tau$ as follows:

$$\begin{matrix} D_1 = \\ D_2 = \\ \vdots \\ D_\beta = \end{matrix} \begin{pmatrix} r_{1_0} & r_{1_1} & \dots & r_{1_\tau} \\ r_{2_0} & r_{2_1} & \dots & r_{2_\tau} \\ \vdots & \vdots & \dots & \vdots \\ r_{\beta_0} & r_{\beta_1} & \dots & r_{\beta_\tau} \end{pmatrix}$$

Indeed, readings sensed by the detectors in successive slot times are mostly redundant according to the variation of the monitored condition; more the condition slows down more the redundancy is. Hence, in order to remove data redundancy and reduce reading stored in MRS, we propose to aggregate similar readings while preserving the dynamicity of the monitored conditions. According to the aggregation approach, the similarity between readings collected by a detector can be determined based on a predefined threshold; if the difference between readings is less than the threshold then the readings are considered similar. However, in order to enhance this approach which is already introduced in [194,195], we propose to extend the aggregation process to several readings detectors at the same time. Therefore, assume we have two vectors of readings $V_i = [r_{1_i}, r_{2_i}, \dots, r_{\beta_i}]$ and $V_j = [r_{1_j}, r_{2_j}, \dots, r_{\beta_j}]$ collected by $\mathcal{D}$ during two successive slot times and a set of predefined thresholds $\varepsilon = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_\beta\}$ then, we define the following function:

$$Sim(V_i, V_j) = \begin{cases} 1 & if \ \ |r_{1_j} - r_{1_i}| \leq \varepsilon_1 \ \ \& \ \ |r_{2_j} - r_{2_i}| \leq \varepsilon_2 \\ & \quad \& \ \ \dots \ \ \& \ \ |r_{\beta_j} - r_{\beta_i}| \leq \varepsilon_\beta \\ 0 & otherwise \end{cases} \tag{3.1}$$

Consequently, V_i and V_j are considered similar if the similarity function, e.g. Sim, between them is equal to 1. Furthermore, the set of thresholds ε is heavily dependent on the monitored conditions and their values should be determined by the experts.

Finally, we define the weight function, e.g. wgt, of a vector of readings V_i in order to preserve the accuracy of the stored reading sets as follows:

Definition 1 *Weight function, $wgt(V_i)$. Given a vector of readings V_i collected by the set of detectors $\mathcal{D}$ during a slot time. Then, the weight of V_i, denoted by $wgt(V_i)$, represents the number of vectors V_j that are similar to V_i according to the similar function ($Sim(V_i, V_j)$).*

3.4.2 Data Storage Reduction Algorithm

In this section, our objective is to show which data collected by the detectors will be saved in the storage space of a MRS using our reduction mechanism (Algorithm 1). In the first slot time, the readings collected by all detectors will be saved automatically in the disk storage with a weight sets to 1 (lines 1-4). Then, for the later collected readings, the controller will search the similarity between these readings and those collected in the previous slot time; if the readings are similar according to the Sim function then the controller removes the new readings while increasing the weight of the previous readings by 1 (lines 6-8) otherwise, it adds the new collected readings and initializes its weight to 1 (lines 9-11).

Algorithm 1 Data Storage Reduction Algorithm.

Require: Set of detectors: $\mathcal{D} = \{D_1, D_2, \ldots, D_\beta\}$; New readings vector collected during the slot time t: $V_t = [r_{1_t}, r_{2_t}, \ldots, r_{\beta_t}]$; Set of thresholds: $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_\beta\}$.
Ensure: Stored reading sets: $\mathcal{V}$.

$\mathcal{V} \leftarrow \emptyset$
if $t = 1$ // readings collected at the first slot time **then**
  $wgt(V_t) \leftarrow 1$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{(V_t, wgt(V_t))\}$
else
  if $Sim(V_t, V_{t-1}) == 1$ **then**
    $wgt(V_{t-1}) \leftarrow wgt(V_{t-1}) + 1$
    discard V_t;
  else
    $wgt(V_t) \leftarrow 1$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{(V_t, wgt(V_t))\}$
  end if
end if
return $\mathcal{V}$

In order to illustrate the process of Algorithm 1, we assume three detectors (D_1, D_2 and D_3) collecting data during a period of 5 slots as shown in the below matrix. Also, we assume the set of threshold values for all detectors as follows: $\varepsilon = \{0.5, 1, 2\}$. First, the algorithm stores the first set of readings V_1 with a weight of 1 in $\mathcal{V}$, e.g. $\mathcal{V} = \{(V_1, 1)\}$. Then, for the set V_2, it searches its similarity with the set V_1 using the function Sim; $Sim(V_1, V_2) = 1$ because $|20.3 - 20| \leq 0.5$, $|34.5 - 35| \leq 1$ and $|121 - 120| \leq 2$. Then, we increase by one the weight of V_1 while deleting V_2, e.g. $\mathcal{V} = \{(V_1, 2)\}$. The same process is repeated for V_3 while we calculate its similarity with the set stored in $\mathcal{V}$, e.g. V_1; again $Sim(V_1, V_3) = 1$ then the weight of V_1 is increased by one and V_3 is deleted, e.g. $\mathcal{V} = \{(V_1, 3)\}$. Now, V_4 and V_1 are not similar according to Sim thus we add V_4 to $\mathcal{V}$ with a weight of 1, e.g. $\mathcal{V} = \{(V_1, 3)(V_4, 1)\}$. Finally, we calculate the similarity between V_5 and the last set stored in $\mathcal{V}$, e.g. V_4; since they are similar we add by one the weight of V_4 and we delete V_5, e.g. $\mathcal{V} = \{(V_1, 3)(V_4, 2)\}$. Subsequently, Algorithm 1 calculates the similarity between sets of readings collected in sequential time slots in order to maintain the temporally information of the stored data.

$$\begin{array}{c} \\ t_1 = \\ t_2 = \\ t_3 = \\ t_4 = \\ t_5 = \end{array} \begin{array}{c} \begin{array}{ccc} D_1 & D_2 & D_3 \end{array} \\ \begin{pmatrix} 20 & 35 & 120 \\ 20.3 & 34.5 & 121 \\ 20.5 & 34 & 122 \\ 21 & 33 & 130 \\ 21 & 32 & 132 \end{pmatrix} \end{array} \begin{array}{l} \\ = V_1 \\ = V_2 \\ = V_3 \\ = V_4 \\ = V_5 \end{array}$$

3.5 Self-Reconfiguration Decision-Making Phase

In MRS, the decision making is the main mission of the robot in order to self-reconfigure its morphology to the surroundings. In the second phase of mechanism, we propose an efficient model that allows MRS to take real-time decision to decide

Score	3	2	1	0	1	2	3
D_1	$\leq r_{1_l} - 2\delta_1$	$]r_{1_l} - 2\delta_1, r_{1_u} - 2\delta_1[$	$]r_{1_l} - \delta_1, r_{1_u} - \delta_1[$	$]r_{1_l}, r_{1_u}[$	$]r_{1_l} + \delta_1, r_{1_u} + \delta_1[$	$]r_{1_l} + 2\delta_1, r_{1_u} + 2\delta_1[$	$\geq r_{1_u} + 2\delta_1$
D_2	$\leq r_{2_l} - 2\delta_2$	$]r_{2_l} - 2\delta_2, r_{2_u} - 2\delta_2[$	$]r_{2_l} - \delta_2, r_{2_u} - \delta_2[$	$]r_{2_l}, r_{2_u}[$	$]r_{2_l} + \delta_2, r_{2_u} + \delta_2[$	$]r_{2_l} + 2\delta_2, r_{2_u} + 2\delta_2[$	$\geq r_{2_u} + 2\delta_2$
...	...	...	...	...	...	...	...
D_β	$\leq r_{\beta_l} - 2\delta_\beta$	$]r_{\beta_l} - 2\delta_\beta, r_{\beta_u} - 2\delta_\beta[$	$]r_{\beta_l} - \delta_\beta, r_{\beta_u} - \delta_\beta[$	$]r_{\beta_l}, r_{\beta_u}[$	$]r_{\beta_l} + \delta_\beta, r_{\beta_u} + \delta_\beta[$	$]r_{\beta_l} + 2\delta_\beta, r_{\beta_u} + 2\delta_\beta[$	$\geq r_{\beta_u} + 2\delta_\beta$

FIGURE 3.2: Score table.

which morphology, among its set of morphologies, is the most suitable to the current surrounding status. In the next sections, we detail the two main pillars of the decision-making self-reconfiguration phase: Score table and Fuzzy set.

3.5.1 Score Table (ST)

The score table (ST) is a customizable guide used by the robot controller in order to determine the criticality of each monitored condition. The main target of ST is to allow early recognition of events that alert the MRS to change its current configuration to a new one that is suitable to the surrounding. Basically, ST defines a normal range of readings, e.g. $]r_{i_l}, r_{i_u}[$, sensed by each detector D_i. Readings outside of this range are assigned a weighted score indicating the criticality degree of the collected readings; more the reading is deviated from the range, more the criticality degree is. Fig. 3.2 shows the score table that determines the severity of the collected readings. After determining the lower (r_{i_l}) and upper (r_{i_u}) bounds of the normal range, a threshold δ_i is defined in order to determine the deviation of the readings from the normal range. δ_i is a user-defined threshold determined according to the application requirements. Finally, we define the set of all detector thresholds as $\delta = \{\delta_1, \delta_2, \ldots, \delta_\beta\}$.

3.5.2 Event-Sensing Decision Making

Sometimes, the robot should be reconfigured according to a sudden event in the monitored environment. For instance, if the robot moves in a forest and a fire is detected then it should change its morphology in order to protect itself and avoid possible failure. Hence, a real-time analysis must be performed to quickly adapt to the happened events. Assume we have a set of predefined events, noted as $\mathcal{E} = \{E_1, E_2, \ldots, E_\gamma\}$, where each of them imposes MRS to adapt itself to a unique morphology $O_i \in \mathcal{O}$. Indeed, each event E_i is highly dependent on the reading severity collected by the detectors $\mathcal{D}$. Thus, E_i can be represented as $[s_{1_i}, s_{2_i}, \ldots, s_{\beta_i}]$ where s_{k_i} is the score, calculated according to the score table, of reading r_{k_i} collected by the detector D_i during the current slot time. Therefore, the set of events $\mathcal{E}$ can be converted to a matrix of events MoE as follows:

$$MoE = \begin{array}{c} \\ E_1 = \\ E_2 = \\ \vdots \\ E_\gamma = \end{array} \begin{array}{c} \begin{array}{cccc} D_1 & D_2 & \ldots & D_\beta \end{array} \\ \begin{pmatrix} s_{1_1} & s_{2_1} & \ldots & s_{\beta_1} \\ s_{1_2} & s_{2_2} & \ldots & s_{\beta_2} \\ \vdots & \vdots & \ldots & \vdots \\ s_{1_\gamma} & s_{2_\gamma} & \ldots & s_{\beta_\gamma} \end{pmatrix} \end{array}$$

Subsequently, assume the set of detectors $\mathcal{D}$ collect the vector of readings $V_i = [r_{1_i}, r_{2_i}, \ldots, r_{\beta_i}]$ during a slot time, where r_{k_i} corresponds to detector $D_i \in \mathcal{D}$. First, the controller finds the score of each reading in V_i according to ST and forms a vector of

scores $S_j = [s_{1_j}, s_{2_j}, \ldots, s_{\beta_j}]$. Then, it calculates the Euclidean distance (ED) between S_j and every event in MoE (Equation 3.2); more the distance between S_j and E_i is smaller more the readings indicate the event. Thus, the controller will change the morphology of the MRS to O_i that corresponds to the minimum distance to the event E_i.

$$ED(S_j, E_i) = \sqrt{\sum_{k=1}^{\beta} (s_{k_i} - s_{k_j})^2} \tag{3.2}$$

where $s_{k_i} \in E_i$ and $s_{k_j} \in S_j$.

Formally, Algorithm 2 shows the event-sensing decision making process applied after collection of readings at each slot time. Briefly, the controller calculates the score of each collected reading based on the score table (lines 1-5). Then, it searches the minimum Euclidean distance between the score vector and all the predefined events (lines 6-13). Finally, the controller finds the suitable morphology corresponds to the detected event then it sends signal to the actuators of modules in order to self-reconfigured to the desired shape.

Algorithm 2 Event-Based Decision Making Algorithm.

Require: Set of detectors: $\mathcal{D} = \{D_1, D_2, \ldots, D_\beta\}$; New readings vector collected during the slot time t: $V_t = [r_{1_t}, r_{2_t}, \ldots, r_{\beta_t}]$; Matrix of events: $MoE = \{E_1, E_2, \ldots, E_\gamma\}$.
Ensure: New morphology: O_i.

1: $S_j \leftarrow \emptyset$
2: **for** each reading $r_{i_t} \in V_t$ **do**
3: calculate score s_{i_t} of r_{i_t}
4: $S_j \leftarrow S_j \cup \{s_{i_t}\}$
5: **end for**
6: $index = 1$
7: $ED_{min} = ED(S_j, E_1)$
8: **for** each event $E_i \in MoE$ **do**
9: **if** $ED(S_j, E_i) < ED_{min}$ **then**
10: $index = i$
11: $ED_{min} = ED(S_j, E_i)$
12: **end if**
13: **end for**
14: **return** morphology O correponds to E_{index}

3.5.3 Periodic-Sensing Decision Making

Sometimes, the real-time decision taken by the robot will not be correct due to two reasons: first, a false event is happened or, second, erroneous data is captured by the detectors. This leads to change the MRS morphology to unsuitable shape and still until the next occurred event. Therefore, in order to overcome this problem, we propose to periodically check the MRS morphology according to the collected data; thus, the MRS will be reconfigured to the suitable morphology at the end of the period. Our decision-making mechanism is based on the Fuzzy sets and aims to select the best morphology decision among a set of them.

1. Fuzzy Sets: Typically, a fuzzy set consists of several elements where each of one has a degree of membership. Let assume a set of readings $R_i = [r_{i_1}, r_{i_2}, \ldots, r_{i_\tau}]$

collected by a detector D_i during a period p, where $S_i = [s_{i_1}, s_{i_2}, \ldots, s_{i_\tau}]$ is the set of calculated element scores. Then, we define the fuzzy set of R_i as follows: $Z_i = \{(s_i, m_{s_i}),\ \text{such that}\ i \in [0,3]\}$, where m_{s_i} is the degree of membership of s_i that can be calculated according to the following equation:

$$m_{s_i} = \frac{wgt(s_i)}{\sum_{k=0}^{3} wgt(s_k)} \tag{3.3}$$

where $wgt(s_k)$ is the weight of s_k in S_i. Thus, in this case, the weight function represents the degree of membership of each score in S_i. For instance, if $S_i = [1,0,0,1,1,2,2,3,1,0]$ then $Z_i = [(0,0.3)(1,0.4)(2,0.2)(3,0.1)]$.

Accodingly, we calculate the fuzzy sets for all reading vectors collected by the detectors $\mathcal{D}$ during a period p, e.g. $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_\beta\}$ where Z_i corresponds to R_i collected by D_i. Then, in order to check the accuracy of the selected morphology, we propose to calculate the strength of the occured event. This will allow the MRS to periodically check if the current morphology is the most suitable for the monitored condition. According to MoE, an event E_i is defined as $[s_{1_i}, s_{2_i}, \ldots, s_{\beta_i}]$ where s_{k_i} corresponds to the detector D_k. Then, we define the strength of E_i as follows:

$$G_{E_i} = \min(\max_{j=0,k=1}^{3,\beta} (m_{s_j} \times e^{-(s_j - s_{i_k})^2})) \tag{3.4}$$

Based on the above equation, the max function is used to find the exponential distances to each event in MoE and the min function finds the weakest distance among the calculated ones. Furthermore, the max function allows to remove readings that either have low confidence/membership or large distance to the ideal value for each detector. Also, the min function tries to find out the strength of a decision/event with the strength of its weakest condition reading. It is important to notice that other aggregate functions could be used like the mean instead of the min and the max functions.

2. Periodic-Sensing Decision-Making technique: Algorithm 3 describes the decision making technique which is applied by the controller at the end of each period. The algorithm takes the sets of readings collected by all detectors during a period (and saved at the controller where the process takes place) as well as the list of possible events and it returns the morhology suitable to the current monitored status. First, the controller calculates the score set for each collected reading set then it finds its corresponding fuzzy set (lines 1-9). After that, the conroller searhes for the strongest decision for each event according to the equation (4) (lines 10-18). Finally, it decides about the most suitable morphology based on the strongest decision.

3.6 System Demonstration and Results Discussion

In order to evaluate our proposed mechanism, we consider a scenario of a MRS which consists of three detectors, e.g. $\mathcal{D} = \{temperature, salinity, humidity\}$. Accordingly, we assume that the robot can be reconfigured to three morphologies $\mathcal{O} = \{boat, wheel, snake\}$; the boat shape is suitable for missions in ocean (detecting mines,

Score	3	2	1	0	1	2	3
Temperature	≤ 16.4	[16.4,18.0]	]18.0,19.6[	]19.6,21.2[	]21.2,22.8[	[22.8,24.4]	≥ 24.4
Humidity	≤ 25.3	[25.3,28.6]	]28.6,31.9[	]31.9,35.2[	]35.2,38.5[	[38.5,41.8]	≥ 41.8
Salinity	≤ 34.1	[34.1,34.3]	]34.3,34.5[	]34.5,34.6[	]34.6,34.8[	[34.8,34.9]	≥ 34.9

FIGURE 3.3: Customizable score table.

water currents, etc.), the wheel is used in flat surfaces, and the snake shape is used in narrow places like tunnels. In our simulation, we used real sensing data collected from Intel Berkeley lab [196] and Argo project [197]. We fixed the period size τ to 100 readings and we varied the thresholds of temperature, salinity and humidity according to following cases: $\varepsilon_1 = 0.2$, $\varepsilon_2 = 0.5$, $\varepsilon_3 = 0.75$, $\varepsilon_4 = 1$. Furthermore, we fixed the thresholds of score table to $\delta = \{1.6, 3.3, 0.16\}$ thus the score table is assumed as shown in Fig. 3.3. The effectiveness of our mechanism is tested and compared to a data compression technique (S-LEC) proposed recently in [181].

Algorithm 3 Periodic-Sensing Decision-Making Algorithm.

Require: Set of detectors: $\mathcal{D} = \{D_1, D_2, \ldots, D_\beta\}$; Set of readings: $\mathcal{R} = \{R_1, R_2, \ldots, R_\beta\}$; Matrix of events: $MoE = \{E_1, E_2, \ldots, E_\gamma\}$; Period: p.
Ensure: New morphology: O_i.

$\mathcal{S} \leftarrow \emptyset$
for each reading set $R_i \in \mathcal{R}$ **do**
  $S_i \leftarrow \emptyset$
  for each reading $r_{i_j} \in R_i$ **do**
    calculate score s_{i_j} of r_{i_j}
    $S_i \leftarrow S_i \cup \{s_{i_j}\}$
  end for
  calculate Z_i of S_i based on equation (3)
end for
$index = 1$
$G_{min} = G_{E_0}$ //G_{E_0} is calculated based on equation (4)
for each event $E_i \in MoE$ **do**
  calculate G_{E_i}
  if $G_{E_i} < G_{min}$ **then**
    $index = i$
    $G_{min} = G_{E_i}$
  end if
end for
return morphology O correponds to E_{index}

In addition, in our scenario, we assumed two events in order to test the efficiency of event-sensing decision making algorithm as follows:

$$\begin{array}{c} \\ E_1 = \\ E_2 = \end{array} \begin{array}{c} \begin{array}{ccc} D_1 & D_2 & D_3 \end{array} \\ \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

3.6.1 Selection of Thresholds

Obviously, the efficiency of our mechanism is highly related to the selection of threshold sets ε and δ. From one hand, increasing the values of ε in *Sim* function leads to decrease the amount of data stored in the MRS, and vice versa. On the other hand, the lowest the values of δ-thresholds are, the better accuracy and relevant decisions and analysis could be made in the MRS, and vice versa. Therefore, selecting the appropriate values of thresholds is very essential in our mechanism. Indeed, we believe that these values should be determined by the decision makers or experts depending on the application requirements. For instance, in critical applications like healthcare and disasters, thresholds must be lower than weather monitoring applications. Therefore, these parameters are based on the application criticality and the studied phenomenon. After selecting their values, the decision makers assign the thresholds accordingly into all detectors in MRS prior to deployment.

3.6.2 Data Storage Reduction Study

In this section, we show the efficiency of the first phase of our mechanism in terms of reducing the amount of data collected and stored in the MRS, compared to S-LEC technique. Fig. 3.4 shows the percentage of remaining data after applying the first phase when varying the threshold values as mentioned in the simulation description. The obtained results show that our data storage algorithm can reduce the data storage more than S-LEC in all cases, except for a small value of ε (e.g. $\varepsilon_1 = 0.2$). Subsequently, our algorithm reduces the stored data from 85% to 90% compared to S-LEC technique. We can also notice that our phase is more efficient when increasing the value of the thresholds from $\varepsilon_1 = 0.2$ to $\varepsilon_4 = 1$. This is because the redundancy among data collected increases when ε increases thus our algorithm remove more data.

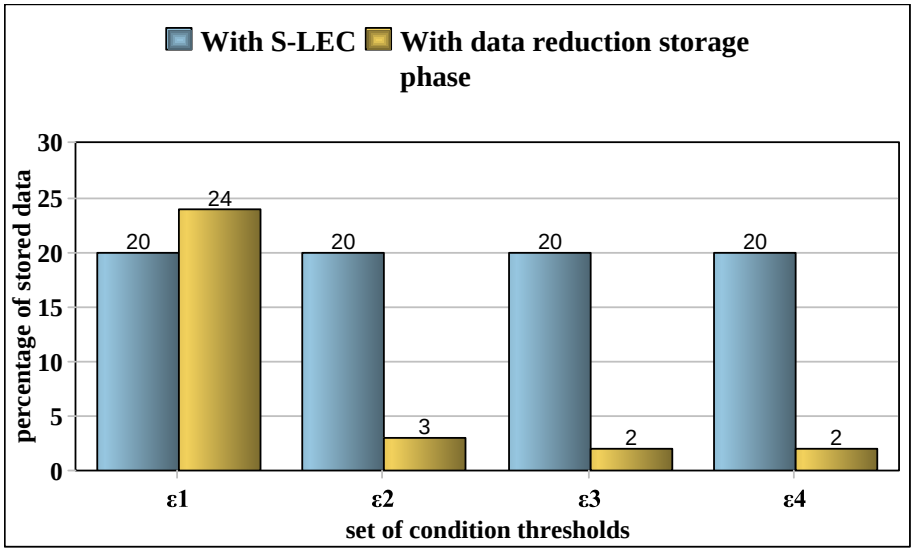


FIGURE 3.4: Amount of data stored in MRS.

Fig. 3.5 shows the variation of data stored in MRS during 10 periods of simulation compared to those obtained with S-LEC, according to the various threshold values ε_1 to ε_4. First, we show that, in both techniques, the amount of data storage varies from period to another dependently of the redundancy among the collected

data; more the redundancy among the collected data is, more our algorithm will reduce the amount of stored data as well as more the data is compressed using S-LEC. This means that the monitored condition is not fixed and it dynamically changes over time. Second, we can clearly show the efficiency of the proposed reduction mechanism compared to S-LEC in almost all periods, except for a small value of ε.

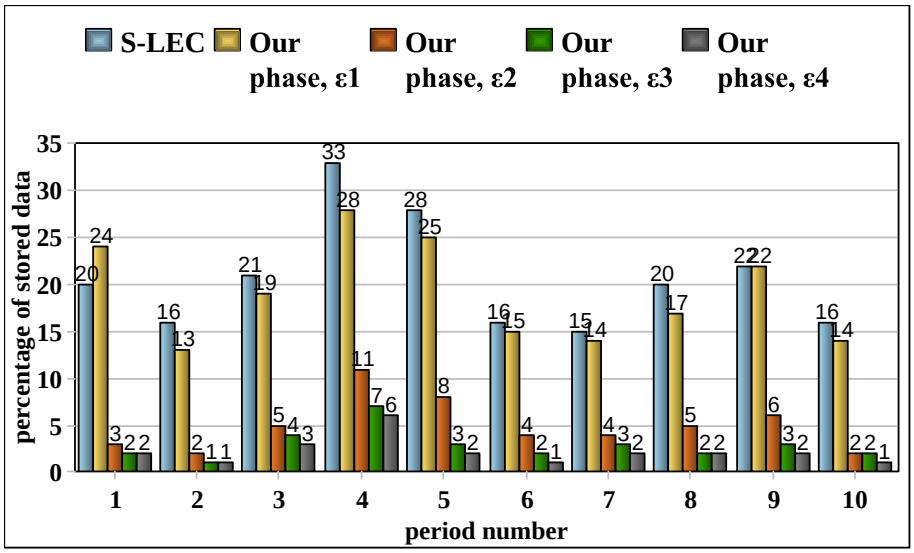


FIGURE 3.5: Amount of data stored in MRS during period progress.

3.6.3 MRS Morphologies Variation during Periods

Fig. 3.6 shows the efficiency of the decision making phase proposed in our mechanism in terms of adapting MRS morphologies according to the dynamic of the monitored surroundings. As mentioned before, the scenario simulates three robot morphologies (wheel, boat and snake). Indeed, the obtained results show that the robot only changes its shape to two of them, e.g. wheel and boat, during 10 periods of surrounding monitoring. As shown, the robot takes the wheel shape during the first three periods which means that it is moving in terrestrial/flat environment. Then, the morphology is changed to boat indicating that the robot meets an ocean or water-based environment. Lastly, the robot rechanged its morphology to the wheel shape according to the changing of the water environment.

3.6.4 Variation of Decision Strength during Period Progress

In MRS, taking the right morphology that suits the surrounding is on if the critical mission for the robot. Otherwise, unsuitable morphology may lead to degrade the robot itself. In our mechanism, the morphology adaptation is highly dependent on the decision strength calculated on the second phase. Fig. 3.7 shows the variation of the strength of decision during each period to select the suitable robot morphology among wheel, snake and boat. The results show a high accuracy of our proposed mechanism in terms of deciding about the suitable morphology in each period. For instance, in the first three periods, the decision strength of wheel shape is very large

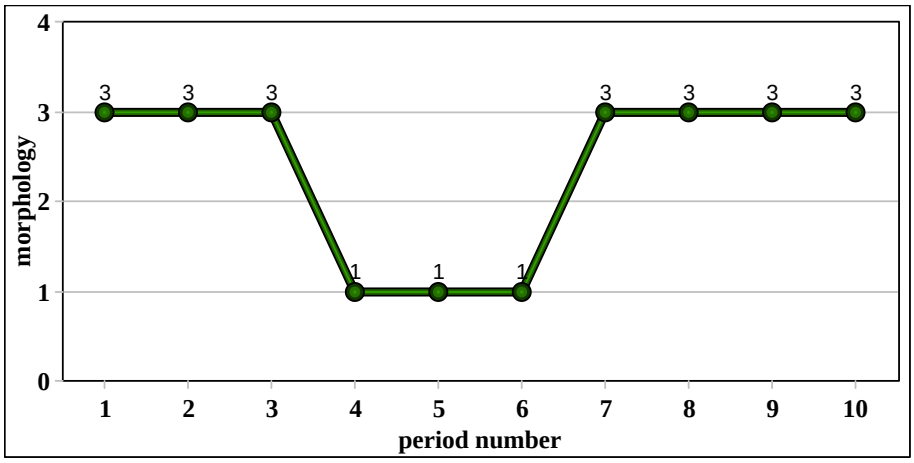


FIGURE 3.6: Variation of robot morphologies during periods.

compared to those for other morphologies. This confirms the behavior of our mechanism and shows its efficiency for self-reconfiguration MRS.

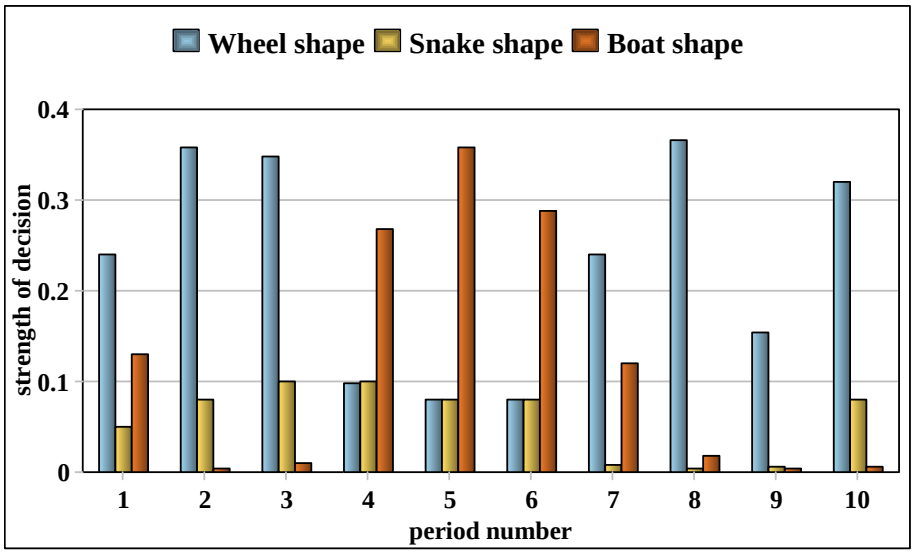


FIGURE 3.7: Decision strength variation during periods.

3.6.5 Surrounding Criticality Variation Study

Finally, we study the variation of the surrounding criticality in terms of the score of readings collected at every period (Fig. 3.8); more the reading scores are more the surrounding is critical. The obtained results are highly dependent on the decision strength shown in Fig. 3.7. First, we can observe the criticality of the surrounding is dynamically changed from period to another as well as within the same period. For instance, in period 4, we clearly show that the surrounding criticality is changed to low, medium and high simultaneously. Second, we also observe that the reading

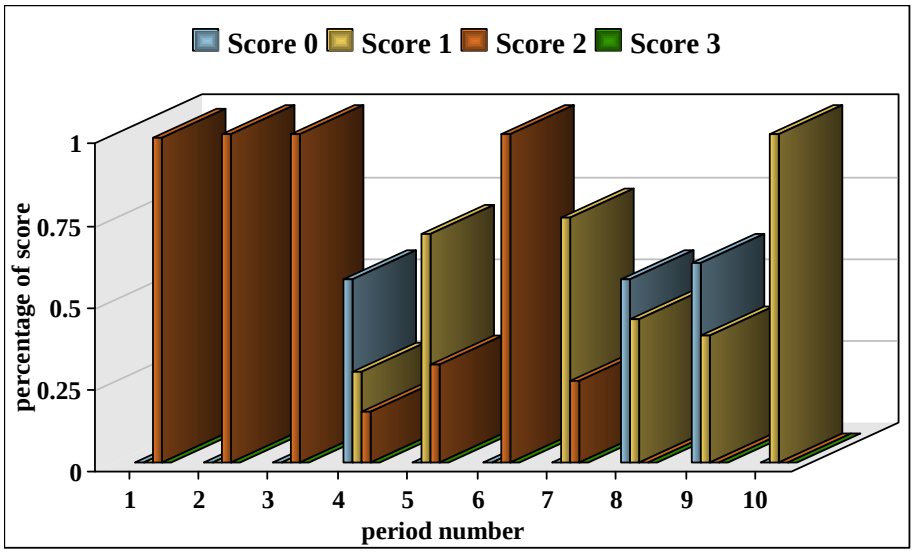


FIGURE 3.8: Surrounding criticality variation during periods.

scores are mostly varied to 1 and 2 which means that the criticality of the surrounding is almost at medium level.

3.7 Conclusion and Future Work

MRS is a promising domain that allows to build swarms of robots which can choose the right configuration for whatever jobs they are called on to do. Hence, sensing data analysis and self-reconfiguration challenges will take more attention from researchers. In this chapter, we have proposed an efficient mechanism for data storage and decision making in modular robotic systems. Our mechanism consists of two phases: data storage reduction and self-reconfiguration decision-making. Whilst the objective of the first phase is to reduce the amount of data stored in the MRS by performing aggregation, the second phase uses the fuzzy logic model in order to allow MRS to self-reconfigured according to the variation of the monitored environments. Through simulation on real detector data, we demonstrated the efficiency of our mechanism in terms of data storage and self-reconfiguration decision.

Chapter 4

RUN: A Robust Cluster-Based Planning for Fast Self-Reconfigurable Modular Robotic Systems

There are several challenging technological and theoretical research issues that need to be addressed before putting modular reconfigurable robotic systems into work. In order to achieve configuration independence, a MRS must extend the notion of modularity to include mechanical design, electronics design, control algorithms, software, and communications. However, the most fundamental challenge for the MRS functionality id the self-reconfiguration process. By definition, self-reconfigurability refers to the transitions of MRS from one shape to another according to the monitored surroundings and without the intervention of humans. Hence, creating self-reconfiguring robot systems poses many engineering issues. These issues are centered around designing the basic self-reconfiguring module and the inter-module connections, and aggregating distributed systems from these modules. In this work, we focus on the self-reconfiguration challenge in MRS and we propose a novel mechanism called Robust clUster-based plaNning, referred to RUN, for fast self-reconfigurable modular robots. The proposed mechanism aims to reduce the communication overhead between the modules by group them into a set of clusters before applying the self-reconfiguration process. Basically, RUN works on two stages: module selection and module communication. In the first stage, we select a set of modules, called as cliques, according to several criteria then the whole modules are divided into clusters based on the number of connections between modules and cliques. Once the clusters are formulated, we introduce two algorithms in the second stage; the inter-module algorithm that allows an efficient communication between the cliques of the clusters and the intra-module algorithm that reduces the communications between modules in the same clusters. We show the efficiency of RUN in terms of communication reduction and fast reconfiguration process, through simulations on Roombots compared to other exiting techniques.

4.1 Introduction

The twentieth century has witness a revolution in the technology sector that highly affects our lives and those of the future generations. Thanks to such revolution, a huge number of smart devices have emerged and used everywhere moving most of the daily tasks to machines. Particularly, robotics is one the most affected

domains by the technology revolution due to the great advancements in mechanical and electrical materials. Thus, new features have been added to the robots that totally changes their traditional working way and makes them smarter. One of such features is the self-reconfiguration process that allows the robot to take several shapes, called morphologies, according to the variation of surrounding in which it exists and the task to be accomplished. This leads to a new generation of robots known as modular robotic system (MRS). One of the most advantages of the MRS is that it can be programmed to carry out several missions and tasks, which are too complex, dangerous, dirty or boring for humans. Hence, MRS has found its way quickly into a great number of applications including rescue, healthcare, manufacturing, reconnaissance and military missions

Generally, a MRS consists of a set of independent modules, each of them is equipped with a battery. In addition, each module has the ability to sense the environment, compute the collected data and move on the space according to some degrees of freedom. Thus, the transitions of the module positions allow the MRS to be reconfigured from an initial morphology to the desired one. Such self-reconfiguration process acts as the most central activity of MRS and it is considered as the main challenge for such technology [198–200]. From one hand, modelling and controlling the self-reconfiguration is an extremely hard task and it is a costly operation in terms of time and computation. On the other hand, the self-reconfiguration requires a massive number of communication between modules when transferring to a new morphology. Furthermore, this challenge becomes more crucial when the MRS contains a high number of modules or the applications require a fast transition between morphologies. Therefore, designing new planning and mechanisms for self-reconfiguration process has becoming essential for MRS.

In the literature, one can find various centralized and decentralized control models that have been proposed for self-reconfiguration MRS [201–204]. Although such models show a great potential by offering a smoothing transitions between morphologies but they mostly suffer from the processing needed to perform such transitions. In this chapter, we take advantages from both centralized and decentralized models and we propose a fast self-reconfiguration mechanism for modular robotics called RUN, Robust clUster-based plaNning. RUN aims to minimize the communication overhead between the modules and enhance the configuration process during the transition between MRS morphologies. Mainly, RUN is based on the clustering of modules and it consists of two stages which can be described as follows:

- The first stage called as module clustering and aims to group the modules into a set of clusters while assigning a special module for each cluster known as clique.
- The second stage called as module communication and allows an efficient communication with and within clusters through two proposed algorithms named as inter-module and intra-module respectively.
- We tested the performance of RUN through simulations on real robot, known as Roombots, and we demonstrated its efficiency in terms of complexity, overhead communication, optimality (minimum number of steps), and time efficiency.

The rest of the chapter is organized as follows. Section 4.2 gives an overview about the existing work for self- reconfiguration in MRS. Section 4.3 formulates the problem of self-reconfiguration and defines some terminologies used in MRS design. In section 4.4, we detail the RUN mechanism with the two proposed stages.

The system demonstration and the results are presented in Section 4.5. Finally, we conclude the chapter and provide directions for future work in section 4.6.

4.2 Self-Reconfiguration: A Background

Nowadays, MRS is a widely studied research field and it has been extensible reviewed in many publications [205,206]. The self-reconfiguration takes a great attention from researchers as the major challenge facing the deployment of modular robotics. Thus, a significant number of works have been proposed to study and analyze the transition of relative positions of modules to allow the robotic to take the suitable morphology and meet the surrounding requirement. Recently, the authors of [207] and [208] give an extensive overview of the current state-of-the-art of self-reconfiguration models and algorithms proposed for MRS while comparing their sufficiency according to several metrics.

The authors of [209–215] propose centralized models for self-reconfiguration in modular robotics. In [209], the authors propose an optimal planning framework (OPF) based on search algorithms for self-reconfiguration of Roombots. OPF uses four approaches (e.g. conventional search heuristic, transition model, offline computation and pruning strategies) in order to reconfigure the modules whether they are connected or disconnected in the space. The main goal of such mechanism is to reduce the reconfiguration complexity, the number of transitions, and the time needed in changing the morphology. The authors of [210] propose a self-reconfiguration technique for 2D square grid MRS where each module occupies a grid in the space. Then, in order to convert the 2D-MRS from one morphology to another, the authors introduce two parallelism-based models for specific and universal centralised transformations with and without preserving the connectivity of the original shape. In [211], the authors uses a tool called Petri nets for task planning and execution in modular robotics. Mainly, Petri nets consists of higher-level and lower-level MRS controllers; the first controller executes a global net model of task plan representing cooperative behaviors performed by the modules while the second controller performs a synchronization of the module activities through the transmission of status requests between them. Lastly, the authors of [212] propose a virtual pheromone (VP) base network data flow control for efficient communication in MRS. The proposed technique uses a gradient routing algorithm with an edutainment game to find the optimal path selection between the communicated modules in order to reconfigure the robotic systems.

The authors of [216–222] propose decentralized models for self-reconfiguration in modular robotics. In [216], the authors propose a tunneling-based algorithm for reconfiguration of cubic modular robots in severe space requirements. Unlike existing methods, the proposed algorithm removes the limitation on the arrangement of the initial and desired morphologies and is available for cases with multi-overlapped parts between both morphologies. Furthermore, the proposed tunneling-based reconfiguration uses the meta-modules in order to maintain the connectivity and mobility of the MRS structure. The authors of [217] propose a deterministic and distributed method for fast reconstructing the scaffold of a MRS based on a set of rotating-only modules. The proposed method relies on the face-centered-cubic lattice structure and operates at two levels of planning. The first level aims to schedule the configuration of the scaffold modules to void deadlocks while the second level allows handling the transition of modules to avoid collisions. In [218], the authors

propose a self-reconfiguration framework that combines between a cluster-flow locomotion based on cellular automata and a decentralized local representation of the spatial geometry based on membrane computing. The proposed framework introduces both theoretical and practical sides for self-reconfigurable robots; on the first side, it provides an abstract presentation of the robots while, on the second side, it develops of new abilities of physical robots inspired by the local relative position of the modules and the local encapsulation of the information. Lastly, a biological method inspired by the plant growth for the distributed self-reconfiguration of UBot systems has been proposed in [219]. First, the reconfiguration process can converge to the target morphology based the implementation of L-systems and turtle interpretations. Then, a set of reproduction rules are parameterized to introduce the influence to the reconfiguration process by distributed modules' local sensing.

Despite both centralized and decentralized models provide good solutions for self-reconfiguration in modular robotics, they mainly suffer from several drawbacks. First, most of them are highly time consumed during the reconfiguration process which is not suitable for many MRS applications, especially the critical ones. Second, the self-reconfiguration models proposed in most of the works are dedicated to one type of modular robot and cannot be generalized to other ones. Third, the computation complexity in the most models are at high level and are suitable to the limited MRS resources. In this chapter, we take advantages from both centralized and decentralized aspects and we propose a fast and robust self-reconfiguration model for robotic systems. Our mechanism may be implemented in almost all modular robotic types as well as it is very scalable to the increase number of modules. Furthermore, the proposed model consists of a set of low complexity algorithms, allowing an efficient communication between the modules, that work on two stages: module selection and module communication

4.3 Problem Formulation and Terminologies

The self-reconfiguration ability is totally changed the definition of the traditional robotic and allows the emergence of MRS with prominent features including adaptability, expansibility, versatility and robustness. By definition, the self-reconfiguration problem can be settled as follows: given an initial and goal morphologies, the modular robotic has to find the sequence of module moves that allow its reconfiguration from the first morphology to the second one. This requires a massive communication between the modules in order to rearrange their connectivity to autonomously adapt to new circumstances or perform new missions. For instance, the modular robotic can change into a snake-like morphology to go through a narrow pipe, transform into a spider-like to cross the harsh landscape, or reorganize as a loop to move rapidly over a flat ground. Hence, the performance of the MRS is highly dependent on the dexterity of modules, thus designing new modules with excellent abilities of locomotion and manipulation has become essential to increase the dynamicity of robotic systems.

Mathematically, we define a modular robotic system, $\mathcal{B}$, as a set of independent modules as follows: $\mathcal{B} = \{M_1, M_2, \ldots, M_\mu\}$, where μ is the number of total modules. By connecting together, the modules can take several shapes that form the set of robotic morphologies, e.g. $\mathcal{O} = \{O_1, O_2, \ldots, O_\theta\}$; θ is the number of possible morphologies (like flag, rolling, snake, etc.) defined prior to the application requirements and the dynamic of the module configuration. Subsequently, each module $M_i \in \mathcal{B}$ is defined according to its components or its configuration ability. From one

hand, M_i can be seen as an independent robot consisting of several components that mainly include motors, sensors, actuators, control and transmission units. On the other hand, M_i has some flexibilities regarding its reconfiguration which is mostly known as degree of freedom. We assume that each module has a α degrees of freedom represented as follows: $\mathcal{F}_i = \{F_1, F_2, \ldots, F_\alpha\}$. In most existing robotic systems, $F_j \in \mathcal{F}_i$ indicates an action of rotation (for some angle degrees to any direction) or move (to left, to right, to top, to down). Thus, more the value of α is, more the robotic system can take morphologies. Furthermore, we consider that each morphology O_i is represented according to the module positions $\mathcal{P}_i = \{p_1, p_2, \ldots, p_\mu\}$ in a 3D plane; p_k is the coordinates of the module M_k where $p_k = \{x_k, y_k, z_k\}$. Therefore, the problem of self-reconfiguration in MRS is formulated as the transition from a morphology O_i to another one O_j by moving the module positions from $\mathcal{P}_i$ to $\mathcal{P}_j$.

4.4 RUN Mechanism

In MRS, the transition between morphologies is challenging task due to several reasons: first, it requires a massive communication between the modules to reach the target morphology. Second, it takes a significant processing time to self-reconfigure the robotic, which becomes crucial in critical applications. Third, it consumes the limited available energy of modules. Therefore, in order to ensure a long MRS functionality, we have to conserve the module energies by reducing the communication overhead and the processing time that consume most of their energies. In the next sections, we describe the RUN mechanism that consists of two stages aiming to provide an efficient communication between the modules and saving their battery power.

4.4.1 Module Clustering Stage

In the first stage, we aim to classify the modules into clusters before performing their transitions. To do that, we first select a set of modules to act as cliques for the robot. A clique module is responsible to ensure an efficient communication between the whole modules during the self-reconfiguration process. On one hand, this will allow to reduce the number of messages transmitted between the modules and, on the other hand, to minimize the number of transitions made by each module (thus saving its energy). The module clustering stage takes into account several parameters to select the clique modules described as follows:

- *The neighbors correlation*: the spatial correlation plays an important role in the configuration of the robotic. Thus, the more the modules are geographically closer, the more the modules are spatially correlated. Hence, the spatial correlation does not mean the modules that are directly connected but those having a short path between them and they do not overload the system during the message exchange. Therefore, we propose to set a threshold n that allows to determine the set of neighbors for each module. Subsequently, two modules M_i and M_j are considered spatially correlated if the shortest path length, e.g. the number of modules, between them is less than n. Accordingly, we define $|M_i|$ as the number of neighbors of the module M_i.

- *The transition steps*: it is defined as the number of actions made by each module M_i to reshape the robotic from its current morphology O_j to the desired one O_k. Indeed, the transition steps are highly dependent on the degree of freedom of the module where the step may be any move or rotate action performed by the

module. Hence, we define the variable A_i as the transition steps or the number of actions required by a module M_i to switch from O_i to O_k.

- *The battery power*: in MRS, each module is equipped with a small battery that cannot be mostly replaced or recharged especially in harsh or hostile environments. Commonly, the battery is consumed during the module transition or the transmission of data. Hence, in order to save its available energy, the module must minimize its data transmission and reduce the number of transitions during the self-configuration. Therefore, the clique modules should be selected among those having more power than the others to efficiently manage the power constraint of a modular robotic. In this stage, we define a threshold I (in joules) allowing to check if the residual energy, Er_i, of the module M_i is at high or low levels; subsequently, M_i is considered at a high energy level if its residual energy is greater than Er_i; otherwise, it is considered at a low battery power.

Based on the above parameters, a module is selected to be a clique node if it meets the following three rules:

- It has a higher number of neighbors compared to other modules as follows:

$$|M_i| \geq N, \text{ where } N \in [1, \mu] \tag{4.1}$$

- It requires a less number of actions to switch from morphology to another as follows:

$$A_i \leq T \tag{4.2}$$

- It has a high level of energy power as follows:

$$Er_i \geq I \tag{4.3}$$

where N, T and I are three thresholds defined by the experts for the number of neighbors, the number of actions and the residual energy respectively. The values of all thresholds are highly dependent on the number of robotic modules and the application requirements.

Algorithm 4 describes the process of the module clustering adopted in RUN mechanism. The algorithm takes, as input, the set of cliques $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_K\}$ and returns the set of module clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$, where K is the number of obtained clusters. For each module M_i in the robotic system, the algorithm finds the shortest path from M_i to each selected clique according to the number of modules between them (lines 1-3). Then, the module is assigned to the cluster having the minimum path length to its clique (line 4). The process is repeated until all the modules are grouped into clusters.

Algorithm 4 Module Clustering Algorithm.

Require: Set of modules: $\mathcal{B} = \{M_1, M_2, \ldots, M_\mu\}$; Set of cliques: $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_K\}$.
Ensure: Set of module clusters: $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$.
1: **for** each module $M_i \in \mathcal{B}$ **do**
2: **for** each clique $Q_j \in \mathcal{Q}$ **do**

```
    calculate the length n for the shortest path between M_i and Q_j
    assign M_i to the cluster C_k having the minimum value of n
  end for
end for
return C
```

4.4.2 Module Communication Stage

Once the module grouping is done, the communication process is starting in order to provide an efficient transition between morphologies. The second stage, e.g. module communication, in our mechanism aims to reduce the number of transitions to conserve the modules' energies and ensure a fast self-reconfiguration of the robotic. Given a transition from O_i to O_j, this stage allows two types of communications to switch the modules from their current positions to the new ones: inter-module and intra-module communications. In the next sections, we detail each type of communication.

Inter-Module Communication

In this step, we propose an efficient scheme for data communication between the cliques of the clusters. Our objective is to minimize the number of transmitted messages in the robotic system to avoid packet congestion and save the module energy. The proposed scheme works according to the following steps:

- *Graph construction*: we consider the robotic system as a connected graph $\mathcal{G}$ ($\mathcal{V}$, $\mathcal{E}$): $\mathcal{V}$ is the set of vertices representing the modules and $\mathcal{E}$ is the set of edges that connecting such modules. Indeed, two vertices are linked with an edge if their corresponding modules are connected directly on the 3D plane. Thus, the cost of any edge is set to one because of the one-step spatially correlated between the modules, e.g. $n = 1$.

- *Tree formulation*: this step aims to convert the graph into a tree while considering the shortest paths between the modules. Hence, we propose to use the Dijkstra's algorithm [223] that allows to find the shortest paths between nodes in a graph and transform it into a minimum spanning tree. Basically, Dijkstra's algorithm works on two steps: first, it selects a vertex as the source node then it finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree. In our mechanism, we select the clique module having the most residual energy as the source node and it is considered as the root of the tree. After that, the source node sends broadcast messages to all modules in order to find shortest paths to other cliques based on the Dijkstra's algorithm.

- *Cliques communication*: once the tree is constructed, the communication between the cliques is started to determine the transitions needed to switch to the new morphology. Indeed, each clique is responsible to manage the transitions of modules inside its cluster. Moreover, it is important to notice that the selection of cliques and the construction of tree will differ from one transition to another depending on the residual energies of the modules. This will lead to balance the energy consumption of the whole robotic system and increase its lifetime.

Intra-Module Communication

Obviously, the number of cliques is controlled by the experts by adjusting the values of parameters defined in the module clustering stage. However, the number of modules may vary from application to another and could be of a high number especially in critical and multi-task applications. This will lead to complicate the mission of the clique when managing a significant number of modules in a cluster. In this section, we propose an intra-module communication mechanism based on the tree structure that allows an efficient transition of modules inside each cluster. In the proposed scheme, the clique is considered as the root of the tree while the modules are reorganized into parent-child relationships.

Indeed, the construction of any tree suffers from the number of relayed packets that can deplete the available energy of modules. Particularly, the modules having more children or those close to the clique will be firstly died because of the significant number of relayed packets. In order to overcome such problem, we propose an energy-balanced and communication-efficient tree that maximizes the lifetime of the MRS and provide a fast reconfiguration process of their modules. The proposed tree is constructed with respecting to the following properties: first, the modules with low residual energy should have less number of children to forward less number of packets; second, the modules with high residual energy are penalized to have more children and perform more transition steps to reshape the robotic system. By achieving that, the residual energy of all modules will be maximized and the energy consumption during each reconfiguration will be minimized.

The first step to construct our tree is to convert the modules inside each cluster into a connected graph where two modules are connected if they have a direct connection between them. Then, the weight $W_{E_{i,j}}$ of an edge $E_{i,j}$ connecting the modules M_i and M_j is calculated according to the following equation:

$$W_{E_{i,j}} = \frac{Er_i}{Ep_i + Em_i} \tag{4.4}$$

where:

- Er_i is the residual energy of the module M_i.
- Ep_i is the energy consumed in the module M_i during the transmission and receiving of packets.
- Em_i is the energy consumption needed to move the module M_i to its final position.

Based on the equation 4.4, the Algorithm 5 shows the process of the tree construction in respecting to the above properties. First, the clique module is added to the tree $\mathcal{T}$ as the root of the cluster. Then, a module M_j is added to $\mathcal{T}$ if its corresponding edge $E_{i,j}$ has a minimum energy cost as follows:

$$Ec_{i,j} = Et_i + \frac{1}{W_{E_{i,j}}} \tag{4.5}$$

where Et_i is the total energy cost during the data transmission from the module M_i to the clique through the shortest path. Subsequently, the selection of the minimum cost edge will maximize the tree lifetime and balance the module energies according to the following facts:

- If the number of packets communicated between the modules M_i and M_j is small then the weight of edge $W_{E_{i,j}}$ tends to be high. This will decrease the

value of $\frac{1}{W_{E_{i,j}}}$ and the module M_j will have high possibility to be added to the tree and connect to M_i. Hence, our objective of minimizing the energy consumption of the modules will be verified at each reconfiguration. In a similar way, the same consequences will be obtained if the number of transitions performed by the module M_i is minimized.

- If the residual energy of the module M_i is high then the weight of edge will tend to be high which decreases the value of $\frac{1}{W_{E_{i,j}}}$. In this case, the value of Et_i needed to reach the clique will be decreased which leads to increase the possibility of the module M_j bein added to the tree.

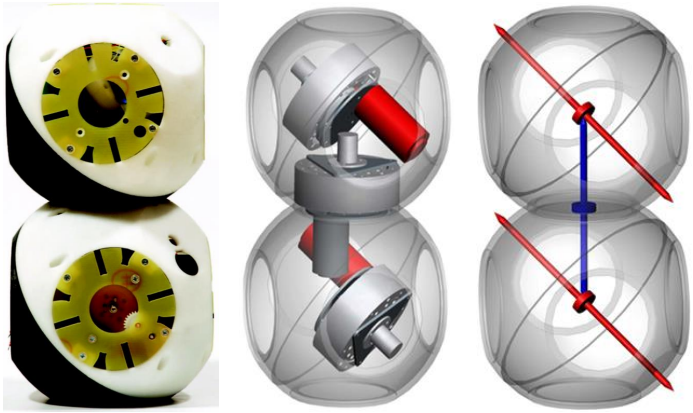
Algorithm 5 Intra-Module Communication Algorithm.

Require: A clique: Q_i; Cluster of modules: $\mathcal{C}_i = \{M_1, M_2, \ldots, M_k\}$; A connected graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
Ensure: A tree: $\mathcal{T}$.
1: $\mathcal{T} \leftarrow \mathcal{T} \cup \{Q_i\}$
2: // set the total energy cost of each module to 0
3: **for** each module $M_q \in \mathcal{C}_i$ **do**
4: $Et_q = 0$
5: **end for**
6: **repeat**
7: find the edge $E_{i,j}$ having the minimum $Ec_{i,j}$
8: $\mathcal{T} \leftarrow \mathcal{T} \cup \{M_j\}$
9: $Et_j = Et_i + \frac{1}{W_{E_{i,j}}}$
10: **until** (no more modules in $\mathcal{V}$)
11: **return** $\mathcal{T}$

4.5 Simulation and Results

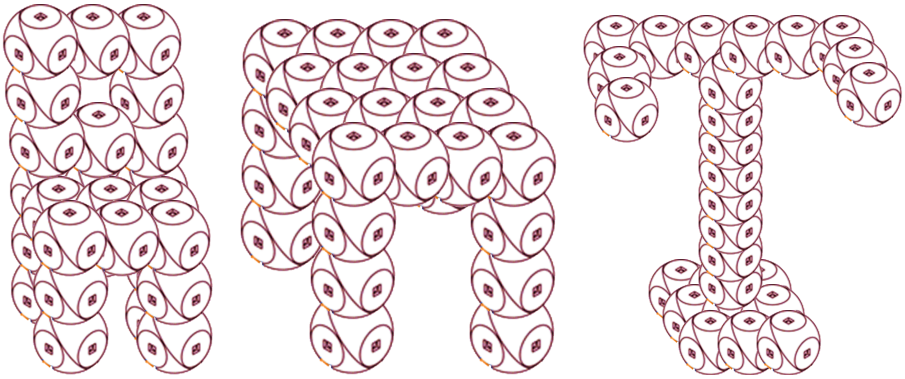
In order to evaluate its efficiency, we tested our mechanism on one of the most used MRS proposed in recent years, e.g. Roombots [224]. Indeed, Roombots is characterized by its high flexibility and it is designed for adaptive furniture. Typically, each Roombots module has three rotational motors and is equipped with two connectors to attach to other modules. Figure 4.1 presents the components of a Roombots module. Thanks to the motors, the module has 7 degrees of freedom as follows: the first and third motors can rotate to angle position $-120°$, $0°$ or $120°$ (2 actions for each) and the second one rotates to $-90°$, $0°$, $90°$ or $180°$ (3 actions). In our simulations, we used the Roombots modules to build a MRS with three morphologies, e.g. $\mathcal{O} = \{table, chair, cane\}$. Our goal is to build a MRS that helps the elderly persons, staying at their homes, during their movements. Subsequently, the table can help the elderly person to eat when he desires, the chair allows him to sit down while the cane assists him during the walking. Figure 4.2 shows the morphologies of our robotic design in a 3D plane where each module occupies a grid. In Table 4.1, we show the parameters adapted in our simulation. We assumed the following sequence of morphology transitions which includes all the possibilities of transitions pair: $\{chair \xrightarrow{1} table \xrightarrow{2} chair \xrightarrow{3} cane \xrightarrow{4} chair \xrightarrow{5} table \xrightarrow{6} cane \xrightarrow{7} table\}$; the

numbers above the arrows represent the transitions' number. Finally, we compared the results of RUN to those obtained with VP [212] and OPF [209].



(a) one module (b) motors (c) axes of rotation

FIGURE 4.1: Roombots components.



(a) chair (b) table (c) cane

FIGURE 4.2: Roombots morphologies adapted in our simulation.

4.5.1 Study of Cliques Number Variation

In Figure 4.3, we shows the average number of obtained cliques during ach morphology transition after applying the module clustering stage. The obtained results are highly dependent on the morphology type (Figure 4.3(a)), the number of modules (Figure 4.3(b)), the module neighbors (Figure 4.3(c)) and the number of actions (Figure 4.3(d)). The figure shows that the cliques' number forms 13% to 36% on the total number of modules. Furthermore, the following observations are eminent:

- The number of cliques obtained in chair morphology is greater than those obtained with other morphologies (Figure 4.3(a)). This is due to the complexity of the chair morphology and the number of actions needed to convert to other ones.

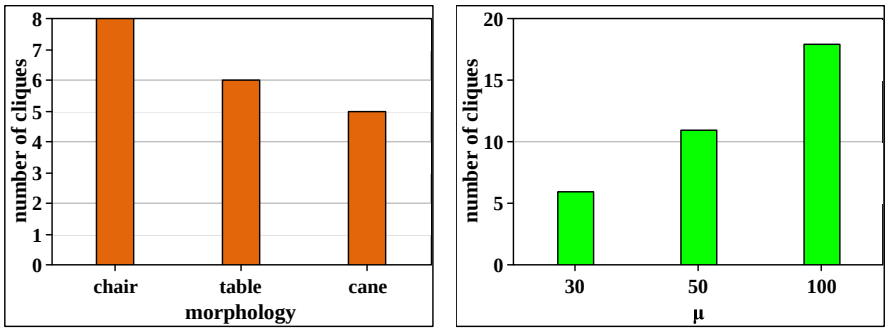
TABLE 4.1: Simulation environment.

Parameter	Symbol	Values
Number of modules	μ	30, 50, 100
Spatial correlation threshold	n	fixed to 3
Initial energy of MRS	E	600 units for $\mu = 30$ 1200 units for $\mu = 50$ 2800 units for $\mu = 100$
Initial energy of module	E_{M_i}	E/μ every action requires 0.8 unit every packet requires 0.2 unit
Neighbors threshold	N	3, 5, 7
Action threshold	T	6, 9, 12
Energy threshold	I	fixed to $E_{M_i}/2$

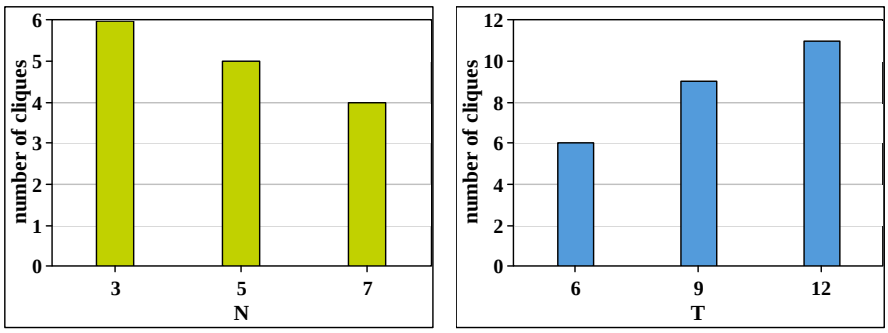
- The average number of cliques during each transition increases with the increasing of the modules' number (Figure 4.3(b)). For instance, the cliques' number increased by 200% when the number of modules varies from 30 to 100. This is because the number of neighbors of each module will increase when μ increases thus, the probability of a module to become a clique will increase.
- The cliques' number decreases with the increasing of the neighbors threshold value (Figure 4.3(c)). For instance, the cliques' number decreased by 33% when the value of N increased from 3 to 7. This is because the number of modules meeting the clique conditions will diminish when complicating the neighbors' threshold.
- The number of cliques increases when the action threshold increases (Figure 4.3(d)). For instance, the number of cliques increases by 83% when T increases from 6 to 12. This is because when T increases each module will have more number of transitions to reach its final position thus its possibility to become a clique module will increase.

4.5.2 Study of Exchanged Packets Number Variation

In this section, we study the efficiency of the module communication stage proposed in RUN in terms of optimizing the number of packets circulated in the robotic system, compared to VP and OPF. Figure 4.4 shows the average number of packets relayed during each morphology transition by all modules when varying the value of μ. The results show that the communication between the modules is highly dependent on the type of morphology. For instance, the second transition, e.g. transition from table to chair, requires less communications compared to other transitions while the third transition, e.g. from chair to cane, requires more communications than the other ones. In RUN, this is due to the complexity of the morphology that requires more/less number of cliques thus increasing/decreasing the inter and intra communications to reshape the robotic system. We can also observe that RUN mechanism outperforms VP and OPF in terms of reducing the number of communicated packets in almost all cases. Therefore, this confirms the behavior of our mechanism



(a) $\mu = 30, N = 3, T = 6$. (b) $N = 3, T = 6$.



(c) $\mu = 30, T = 6$. (d) $\mu = 30, N = 3$.

FIGURE 4.3: Average number of obtained cliques during each morpholoy transition.

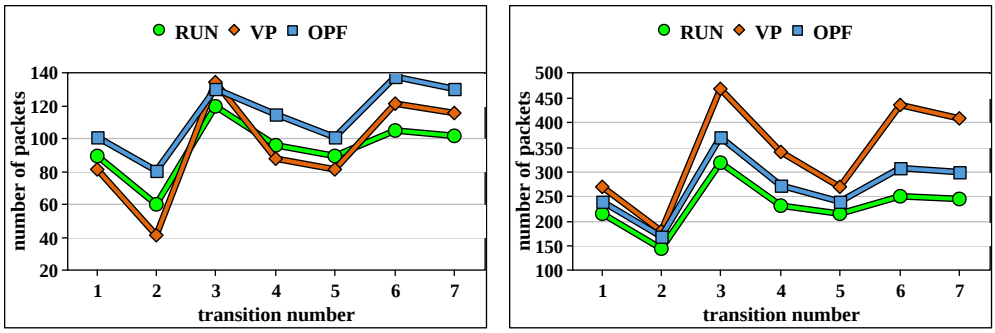
by providing a less number of relayed packets for tree construction and robotic transition. Thus, RUN mechanism allows avoiding the packet congestion and system overflow that results in decreasing the processing complexity of the MRS. In addition, the following observations are eminent:

- The number of relayed packets increases with the increase number of robotic modules. For instance, by applying RUN mechanism, each module relays an average number of 3, 4.5 and 5.2 packets during each transition when μ changes to 30, 50 and 100 respectively. This is due to the number of cliques which increases when increasing the number of modules (see Figure 4.3).

- RUN and OPF give better results than VP in case of large number of modules or complex morphology transitions (Fig. 4.4(b) and 4.4(c)). Whilst, VP gives better results than the other mechanisms only in the cases of small number of modules, e.g. $\mu = 30$, or simple robotic morphology (Fig. 4.4(a)).

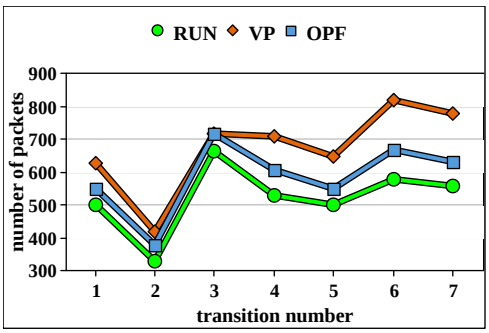
4.5.3 Study of Actions Number Variation

In this section, we study the efficiency of RUN mechanism in terms of optimizing the number of actions performed by the robotic to switch between its morphologies, compared to VP and OPF. Figure 4.5 shows the number of actions made by the Roombots during each transition to reach its desired morphology, when varying the number of modules. First, the results show that the number of actions is varying from transition to another and it is dynamically adapted according to the robotic



(a) $\mu = 30$.

(b) $\mu = 100$.



(c) $\mu = 100$.

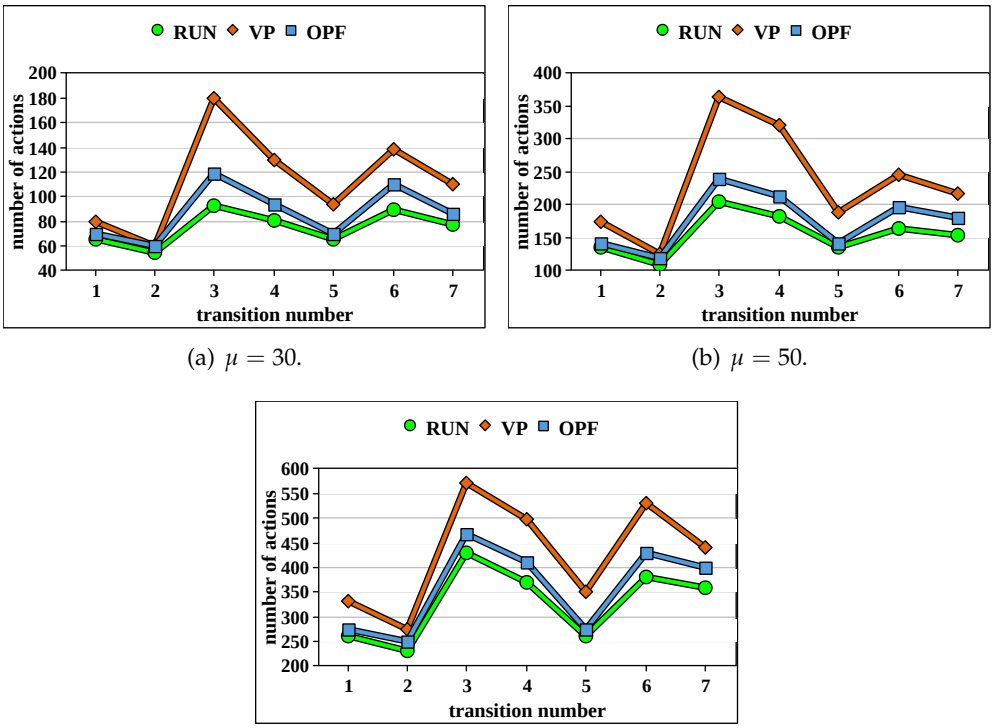
FIGURE 4.4: Average number of packets exchanged during each morpholoy transition, $N = 3$, $T = 6$.

morphology. For instance, we can observe that the most number of actions is noticed during the third transition, e.g. from chair to cane, while the lowest actions' number is detected during the transition from table to chair. We can also observe that RUN mechanism outperforms VP and OPF in terms of reducing the number of actions performed by the whole robotic system in all cases. This is due to the module communication stage proposed in RUN that allows to find the optimal path transition between morphologies compared to the routing protocol proposed in VP and search-based algorithm proposed in OPF. Furthermore, we can notice the following observations:

- The number of actions performed by the MRS increases with the increasing of its number of modules in all mechanisms. For instance, by applying RUN mechanism, the MRS requires an average number of 75, 155 and 327 actions during each transition when μ varies from 30 to 100 modules respectively. This is logical since to the number of actions made by each module to move from its current position to the final one will increase when the dimension of the 3D plane increases.

- RUN reduces the average number of actions performed during each morphology up to 33% and 15% compared to VP and OPF respectively.

4.5.4 Study of Energy Consumption



(a) $\mu = 30$.

(b) $\mu = 50$.

(c) $\mu = 100$.

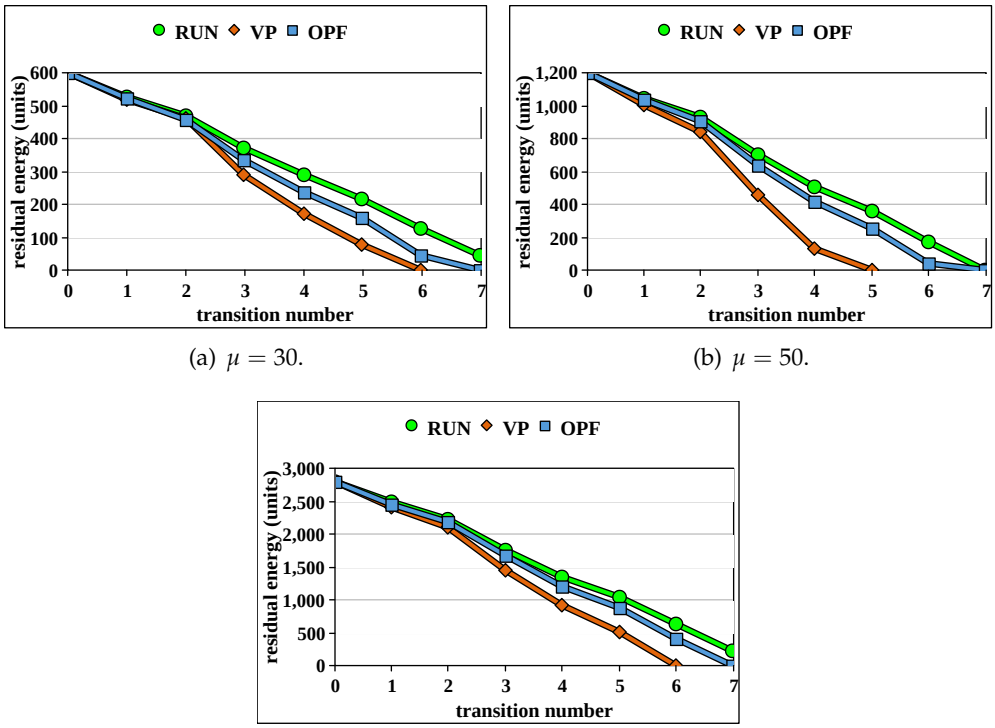
FIGURE 4.5: Average number of actions performed during each morpholoy transition, $N = 3$, $T = 6$.

In MRS, energy consumption is an important metric to evaluate since it affects the functionality of the whole system. Indeed, the limited energy of a module is mostly consumed during the data transmission or the action performed during the transition. In our simulations, we assumed that each module has fixed energy units, depending on the MRS size, then we considered that each packet transmission consumes 0.2 unit and each action consumes 0.8 unit. Figure 4.6 shows the residual energy of the whole MRS after each transition in function of the modules' number for the three mechanisms. The obtained results shown in this figure are highly dependent on those obtained in Figures 4.3 and 4.4. Thus, RUN can save the robotic energy and extend its lifetime more than VP and OPF mechanisms. Furthermore, the following observations are eminent:

- The energy consumption of the MRS proportionally increases with the increasing of the number of modules. This is due to the more number of relayed packets (see Figure 4.4) and actions (see Figure 4.5) required when μ increases.
- The residual energy of the MRS is differently changed depending on the final morphology. Thus, when the morphology is more complex, such as the chair shape, the residual energy will be quickly changed otherwise, e.g. in table shape, it will be more conserved.

4.6 Conclusion and Future Work

Due to competition and innovation, the modular robotics is expecting to grow



(a) $\mu = 30$.

(b) $\mu = 50$.

(c) $\mu = 100$.

FIGURE 4.6: Total energy consumption after each morpholoy transition, $N = 3$, $T = 6$.

exponentially in the next years leading to a radical change in automation systems. Obviously, self-reconfiguration will remain the central problem faced by scientists and engineers where designing efficient solutions can lead to enhance the flexibility of such technology and its provided services. In this paper, we proposed a robust mechanism for self-reconfigurable robotics called as RUN that consists of two stages: module clustering and module communication. The first stage aimed to group the modules into a set of clusters based on the number of connections between them where each cluster is assigned a clique module. The second stage proposed two algorithms where the first one, e.g. inter-module, allows an efficient communication between the cliques of the clusters and the second one, e.g. intra-module, aims to reduce the communications and the transitions within the clusters. Through simulations on real scenario using Roombots, we demonstrated the efficiency of our mechanism in terms of reducing the communications in MRS and providing a fast reconfiguration process.

Chapter 5

FSET: Fast Structure Embedding Technique for Self-Reconfigurable Modular Robotic Systems

The most crucial process in MRS is self-reconfiguration, which is regarded as the major challenge for such technology. Self-reconfiguration is the mechanism by which a modular robot's initial arrangement of modules is changed into a target configuration. Indeed, MRS uses self-reconfiguration to get new morphology and new behavior to performs specified tasks. But, creating new morphology and behaviors manually is a time-consuming and costly process, especially when dealing with complex structures.In this chapter, we have proposed a fast self reconfiguration technique called FSET, i.e. Fast SET, dedicated to MRSs. Our proposed technique consists mainly in two stages: root selection and morphology formation. The final goal of these stages is to enhance the time cost to get new morphology of the traditional SET algorithm thus, ensure fast self reconfiguration. The root selection stage selects a small number of modules in order to find the best tree roots that effects the topological conditions that leads to successful of the embedding process or not. The morphology formation stage uses the traditional SET algorithm to calculate the embedding truth table where the initial roots used are taken from the first stage. Finally, we show the efficiency of our mechanism through simulations on real scenario using M-TRAN, in terms providing a fast reconfiguration process in MRS and reducing the energy consumption of modules thus, increasing its lifetime.

5.1 Introduction

Since the early 1980's, the robotics industry has seen an increase in the production and manufacture of millions of robots of different types and missions. These robots are altering our everyday lives and assisting us in making our jobs more efficient and successful. Furthermore, the rapid advancement of technology in the new century has ushered in a new robotic era: Modular Robotic System (MRS).

The most important attribute of modular robots, regardless of their design, is their ability to reconfigure their morphology, a mechanism known as self-reconfiguration (See Fig. 5.1), which is regarded as the major challenge for such technology [225,226]. Self-reconfiguration is the mechanism by which a modular robot's initial arrangement of modules (and thus initial form, also known as configuration) is changed into a target configuration. Indeed, MRS use self-reconfiguration to get new morphology and new behavior to performs specified tasks. However, creating new morphology

and behaviors manually is a time-consuming and costly process, especially when dealing with complex structures. As a result, designing an algorithm that creates new morphology and new behavior from already existing modular robotic structures, takes a great attention from researchers and communities and became an active research field nowadays.

Therefore, to avoid the above-mentioned challenge, Structure Embedding Technique (SET) for the self-reconfigurable modular robotic system has been introduced. SET decides if a given modular robot structure can be embedded into another structure in order to form new morphology. In this chapter, we present a fast SET, abbreviated FSET, a technique for modular robotic systems to minimize the delay to get new morphology. The proposed technique consists of a two-stage algorithm and can highly outperform the traditional SET in terms of the time cost to get new morphology and energy consumption of modules. The first stage of our technique, called root selection, has an objective to find the best roots of the initial trees, by selecting a small number of modules instead of the whole sets. The second stage, which is called morphology formation, uses the first stage's root of trees, to calculate the embedding truth table between modules in order to check the embeddability of the two modular robotic designs, resulting in the formation of new morphologies. Consequently, the calculation time cost of our FSET will highly minimize that of traditional SET due to the small number of the training modules used in the first stage and the low number of iteration loops needed in the second stage.

The rest of the chapter is organized as follows. In section 5.2, we present related works in self-reconfiguration techniques used in MRS. Sections 5.3 detail the SET mechanism. The system demonstration and the results are presented in Section 5.4. Finally, Section 5.5 concludes our paper and gives some perspectives.

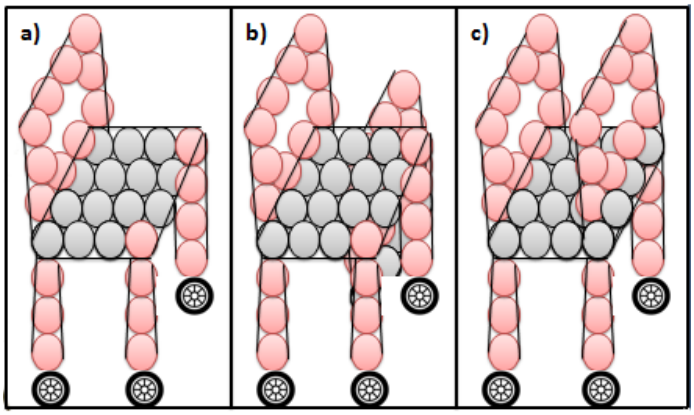


FIGURE 5.1: Sample self-reconfiguration of about 52 3D modules from a chair into a stroller. (a) Chair initial configuration; (b) An intermediate configuration from the self-reconfiguration process; (c) Stroller goal configuration..

5.2 Self-Reconfiguration: A Background

Various methods for self-reconfiguration introduced for MRS can be found in the literature. Initially, researchers in the field of modular self-reconfigurable robotics

based their focus on the hardware issue of creating metamorphic robots; then, research interest eventually arose in the generalized management of categories of these systems and various software frameworks were proposed; More recently, in the field of Self-Organizing Particle Systems, researchers have begun to show interest in what may be called a theoretical kind of metamorphic system.

In fact, three categories can be extracted from these three methods. We would then refer to self-reconfiguration algorithm, which Bottom-Up, Top-Down, and Theoretical, as well as. It can be seen in Figure 2, as presented in [227]. Such methods vary by how they relate to their target execution platform, and therefore, by the nature of the constraints that make up the algorithm model used. We'll go through each of the three self-reconfiguration methods mentioned above, as well as the possible solutions.

From one hand, the authors of [228–233] have proposed Bottom-Up methods to self-reconfiguration in MRS. The Bottom-Up method involves a focus on modular robotic hardware at first. They've come up with a range of module models, ranging from UCMs like Telecube [228] and Crystalline [229] to hybrids like M-TRAN [230] and Roombot [231]. bi-partite models like the Robotic Molecule [233] and I-Cubes [234,235], as well as self-reconfigurable structures like Fracta [232]. There are several other models in the literature, but these are the ones that are used in the algorithms under consideration.

Due to the complexities of the geometry of hardware modules or their motion capacities, this approach credibly provides a very complicated self-reconfiguration preparation. Non-holonomic motion constraints are typical in these systems, complicating the reconfiguration method. Motion constraints can either be local: caused by module geometry and blocking constraints; or they can be global: like the connectivity constraint which specifies that the whole system's graph must stay connected at all times. Several strategies for achieving holonomy at the expense of granularity have been devised, including using higher holonomy module aggregates (meta-modules) [234] or arranging the device into a porous structure [233] from which modules can flow unconstrainedly (a scaffold). Although the kinematics in the Bottom-Up method are typically more complicated, modules are more likely to expect a greater understanding of their surroundings. Sensor data about their orientation, location in the system, neighborhood, and other factors are used to produce these environmental information.

The Roombot hybrid modular robot was designed by the authors of [231], which relieves some of the strong constraints imposed on MSR that rendered planning extremely difficult. Instead of traditional neighbor-to-neighbor communication, Roombots can communicate with other modules through broadcast, which does not require the robot to be connected at all times (which is a major constraint in virtually all other systems), but it does necessitate the existence of a structured ground surface with passive connectors. They introduced a decentralized self-reconfiguration algorithm based on meta-modules comprised of two stacked Roombots, ensuring that individual modules can still shift. Their method is based on the locomotion of disconnected Roombot meta-module structures that converge into the desired configuration due to the attraction of a force-field and a predetermined assembly order.

The authors in [233] have suggested a centralized solution for their bipartite

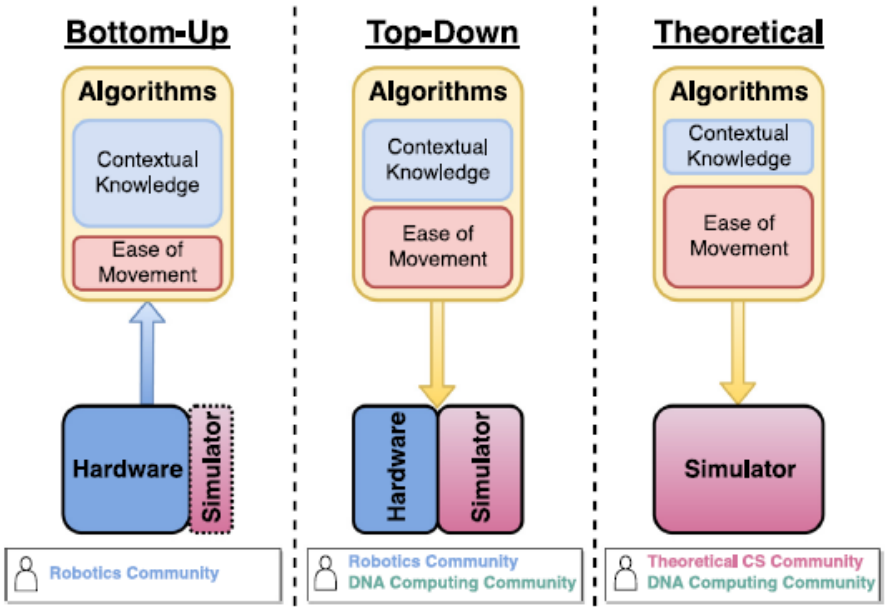


FIGURE 5.2: The three methods for developing self-reconfiguration techniques, as well as their features [6].

Molecule robot, depending on a three-level hierarchical planner. The highest level of preparation is task-level planning, which chooses a configuration that is suitable for the task at hand. It then admits on a motion plan for Molecules to turn the initial configuration into the target configuration using configuration planning. The configuration planner uses trajectory planning to shift individual modules to their target positions at a lower stage. They also implemented the aforementioned scaffolding principle to ensure that Molecules reached into the target configuration, but this increased the granularity of their device dramatically since a single scaffold tile consisted of 54 modules. While these early works using centralized planners laid the foundations for most of the field and implemented useful problem simplification techniques, they lack the robustness, scalability, and autonomy that self-reconfiguration needs. As a result, researchers transformed to the decentralized self-reconfiguration process, as we'll see below.

In [234] the authors propose a similar method for the I-Cubes bipartite system, which uses three-degree-offreedom connections for communication and actuation, as well as passive cubes for the modules. They used a centralized two-level planner where the high level planner specifies the location of modules in the target configuration by using the low level planner to find a viable plan of individual connection movements that will shift the module to the desired location. This approach was later developed iteratively by implementing meta-modules to simplify planning and, as a result, adding a third layer of planning on top of the existing two, at the meta-module level. Another project with I-Cubes used a centralized divide and conquer technique, in which the issue of planning the motion of a module from one position to another was broken down into subproblems. They also used a two-level hierarchical planner in this research, with (1) the low level planner looking for solutions to subproblems and (2) the high level planner looking for the actual motion of the module, incorporating solutions from the lower level [235].

In [232], the authors proposes a distributed algorithm for 3D reconfigurable structures with star-shaped modules based on local knowledge. They suggested a definition of the target shape based on link types, which they had previously used in some of their 2D hardware works. They used stochastic relaxation based on simulated annealing to add randomness to local laws.

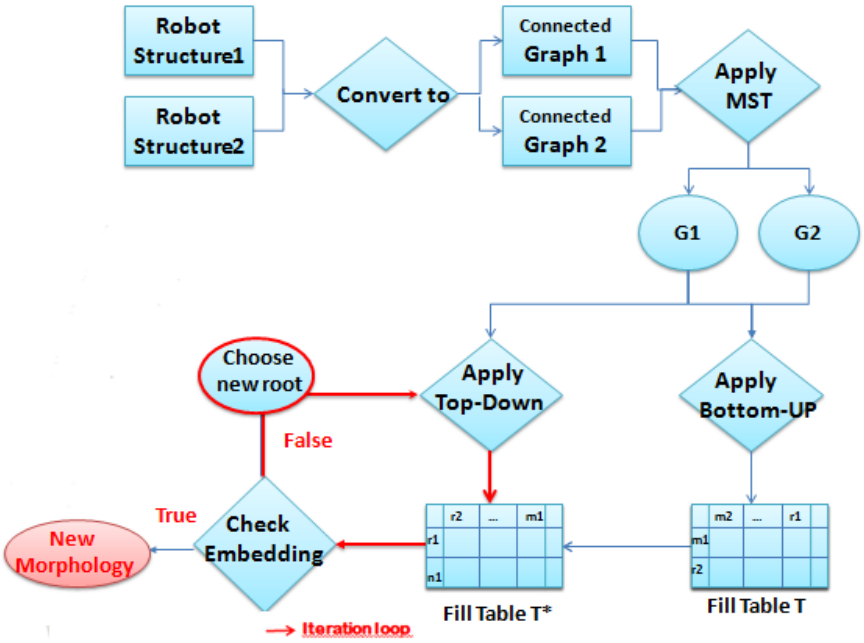


FIGURE 5.3: SET Algorithm Flowchart.

On the other hand, a lot of Top-Down methods have proposed to self-reconfiguration in MRS [236–241]. The Top-Down method plays a critical role in constructing shape formulation techniques that aren't related to a particular hardware application and can be applicable to a variety of MSR in a generalized way.

The authors in [236] developed the Pixel meta-module framework for lattice-based modular robots, which could greatly simplify reconfiguration planning in large modular robots. The key idea is to split the reconfiguration problem into two tasks: planning and resource allocation. The former's job is to figure out the meta-module positions in the target configuration need to be filled next, while the latter's job is to figure out where the meta-modules that will fill that position should come from.

In [241] the authors presents a two-level hierarchical approach to completely generic algorithms (for any architecture), in which the planning problem is formulated as a distributed Markov Decision Process (MDP). An MDP is defined by the four-tuple S, A, T, R, where S is the set of states, represented by open positions to be filled by modules, and is equal to the number of faces of the modules; A is the set of actions, represented by the disconnection of a connector from a neighbor module and the reconnection to another, potentially using a different connector; T is a deterministic or stochastic transition function that determines the next action to take; R is the estimated reward, which is set to 1 to reduce the number of moves.

The authors overcome this MDP by introducing dynamic programming in a distributed environment using message passing. The MDP works at the planner's higher levels, deciding for each moving module which other modules and connectors should link to during the next time phase. The low-level planner then calculates the sequence of individual module movements that the mobile module can take to detach from its current neighbor and reconnect at its new anchor point. Modules look at the structure to make sure they aren't an articulation point in the system's graph, then determine if they are mobile or not and lock a portion of it during their motion if they are. Since several modules can lock the same part of the structure, they can shift in parallel, speeding up the reconfiguration phase. Designing an effective kinematic planner to serve as the transition function T is the most difficult part of this scenario.

The authors in [242] developed their PacMan self-reconfiguration algorithm for two-dimensional unit-compressible modules to three-dimensional structures. As compared to surface moving modules, one advantage of UCM is that they can migrate through the volume of the system, theoretically benefiting from a higher number of parallel motions and a shorter distance to their destination. The authors use a technique known as virtual relocation to transfer modules from one end of the configuration to the other, switching their identities as they compress and decompress along the path to their target point.

PacMan depends on a two-stage distributed planning algorithm in which: (1) modules locally calculate the difference between the current and target shapes to determine which modules should move; and (2) A suitable search (depth-first search) for a mobile module is carried out from the desired locations, with pellets dropped along the way to mark the route that the selected module should follow. Our proposed method consists of two phases, where it applies one of the bottom-up algorithms in its first phase, while in the second phase it applies one of the top-down algorithms. Recognizing if two complete configurations are the same [243], detecting graph automorphisms[244], and recognizing similar substructures for efficient reconfiguration are all examples of existing work in graph representations of modular robots. Our work stands out by incorporating task implications on configurations and specifying criteria for replicating a design's capabilities by replicating its design.

5.3 FSET Technique

In the literature, one can find a huge number of self reconfiguration algorithms like SET, PacMan, scaffold-based etc. However, SET is one of the most popular algorithms used in self reconfiguration. Unfortunately, traditional SET suffer from its huge calculation time cost needed to obtain the new morphology. In order to overcome this problem, we propose a new version of SET called FSET, Fast SET, which highly enhances the time cost of traditional SET. Our FSET consists of two stages, root selection and morphology formation stages, and calculate the embedding truth table according to the topological embedding condition. In the next sections, we first recall the traditional SET and its topological embedding conditions then we detail the two stages of our technique.

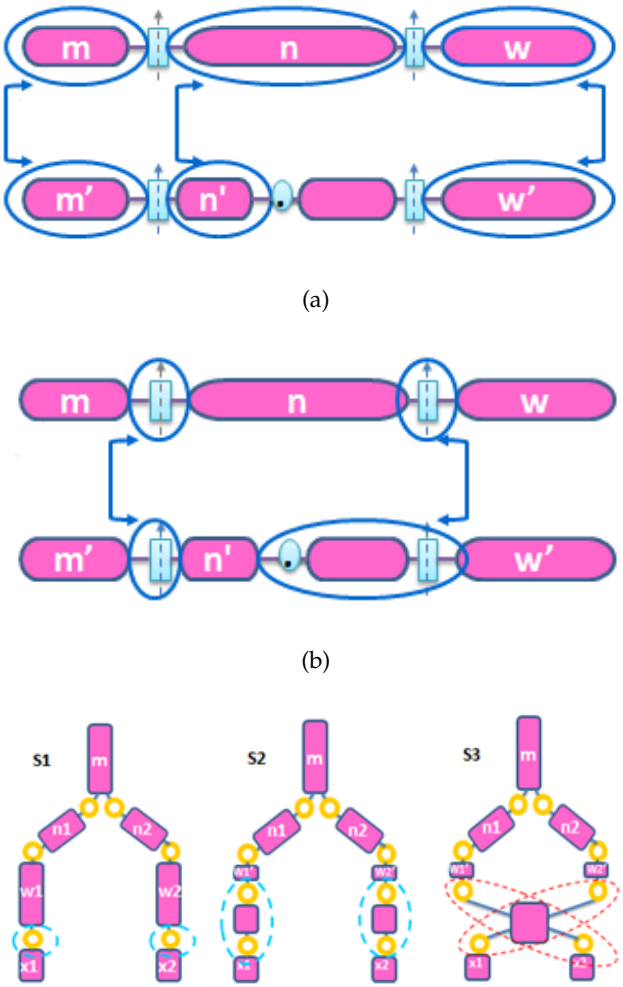


FIGURE 5.4: Topological conditions for embedding [7]

5.3.1 Recall of SET Algorithm

SET is one of self reconfiguration algorithm that decides if a given modular robot structure can be embedded into another structure to form new morphology (Fig. 3.). The process of SET starts by taking the two robotic structures and convert them into two connected graph. Then, it applies MST to them and randomly selects the initial roots for the trees.

We consider that the robotic system is modeled as a connected graph G (M, L): M denotes the set of nodes representing the modules, and L $\subseteq$ M $\times$ M denotes the set of links that connecting such modules.

SET maintain a $|M_1| \times |M_2|$ truth table T, where T[m_1, m_2] is true, under a specified rooting $r_1 \in M_1$, $r_2 \in M_2$. At the end of the algorithm, T[r_1, r_2] answers whether structure (S_1) embeds in stucture (S_2) under r_1 and r_2; if the answer is negative, it repeat the process for a new rooting until it either get a positive answer or we exhaust all possible rootings, in which case we conclude that S_1 does not embeds in S_2.

SET is a two pass algorithm. At first, all entries of the truth table are false. After that, it start proceed bottom-up, starting from the leaves of S_1, and keep going gradually towards r_1. As a starting point, it consider a leaf $m_1 \in M_1$ and check whether it embeds in the leaves of S_2, to calculate the truth table that give us the new morphology according to the topological condition (see [7] for details). Basically, it fill the truth table by traversing G_1 in reverse pre-order, where at each step of the traversal process the nodes of G_2 in reverse preorder.

After first pass, the second pass is lunched, where it involves a top-down message passing. It iterate top-down, starting from the roots of S_1, and progressively down to v_1. Then it compute a new table called T^*, based on the topological condition in top-down and the preceding truth table in bottom-up pass. However, $T^*[r_1, n]$ = true iff $T_n[r_1, n]$ = true, $\forall n \in M_2$. It is then not hard to see that S_1 embeds in S_2 iff at least one entry of the $r_1 - th$ row of T^* is true. if the answer is negative, This process is repeated until we either get a positive answer or we exhaust all possible rootings.

Indeed, it is proved that the loop process generated in the algorithm will always end. Subsequently, SETis highly dependent on the randomly initial tree roots. SET algorithm is one of the simple method in the self reconfiguration approach that has been used in wide range of domains.

5.3.2 Topological Embedding Condition

This section introduces the fundamental graph-theoretical principles and the concept of topological structure embedding and formally introduce the graph representation of modular robotic structure that we will use throughout our discussion of topology. The robotic system is modeled as a connected graph G (M, L): M denotes the set of nodes representing the modules, and L $\subseteq$ M $\times$ M denotes the set of links that connecting such modules. Indeed, two nodes are linked with an edge if their corresponding modules are connected.

Definition 1. (Unit Block). A unit block $B = < \phi >$ is an elementary rigid body capable of implementing a prespecified set of built-in functionalities $\phi \in F$.
Built-in functionality is independent of topology; e.g., consider a block equipped with sensors, a processor unit or a battery. We define a partial order on unit blocks on a functional basis: $B_1 1 \preceq B!2$ if and only if $\phi_1 \in \phi_2$.

Definition2. (Modular Robot Structure). Given a set of unit blocks **B**, a robot design $D = \langle$ (V, L),$\beta \rangle$, defined on **B** is a labelled, undirected graph G, where nodes of G correspond to unit blocks through β : M $\rightarrow$ **B**, and edges between two nodes m and n represent a revolute joint connecting β(u) to β(v).

Definition3. (Structure Embedding). Given two Structure $S1 = \langle G_1(M_1 , L_1), \beta_1 \rangle$ and $S2 = \langle G_2(M_2 , L_2), \beta_2 \rangle$ defined on a set of unit blocks **B**, and an injective mapping $h : M_1 \rightarrow M_2$, we say that S_1 embeds in S_2 with respect to h, and write $S_1 \subseteq_h S_2$, if and only if:
1. Functionality subsumption: $\forall m \in M_1$, we have $\beta_1(m)$ to $\beta_2(h(m))$.

2. Connectivity preservation: $\forall(m,n) \in L_1$, there exists a simple path $\pi_{m_n} = h(m) \rightsquigarrow h(n) \in G_2$.

3. Path disjointness: for any pair of edges with distinct endpoints (m_1, n_1), (m_2, n_2) $\in L_1$, the corresponding paths $\pi_{m_1 n_1}$ and $\pi_{m_2 n_2}$ in G_2 are vertex-disjoint. In addition, for any (m, n_1), (m, n_2) $\in L_1$, π_{mn_1} and π_{mn_2} share only $h(m)$.

In general, we refer to S_1 as the substructure and S_2 as the superstructure. Where there is no chance of confusion, we omit h and write $S_1 \subseteq S_2$.

Fig. 4 offers the intuition behind the definition, as presented in [7]:

1. Condition1: requires every vertex in the substructure to map to a vertex of equal or superior functionality in the superstructure.

2. Condition2: preserves the connectivity of the substructure once it is embedded: nodes which were able to interact through the joints can still do it, albeit maybe through longer paths.

3. condition3: ensures that degrees of freedom which are independent in the substructure remain independent in the superstructure.

From a topological perspective, embeddability is equivalent to whether the substructure is a topological minor of the superstructure; see [245] and references therein. We are now ready to state our main result.

Theorem 1. Given two structures $S1 = \langle G_1(M_1 , L_1), \beta_1 \rangle$ and $S2 = \langle G_2(M_2 , L_2), \beta_2 \rangle$, defined over a set of unit blocks **B**, where G_1 and G_2 are trees of maximum degree d, there exists a deterministic algorithm that decides whether $S_1 \subseteq S_2$ in time $O(|M_1| \cdot |M_2| \cdot d^3)$. Note that d is the maximum number of edges incident on any node. For most real robot applications, $d \leq 5$.

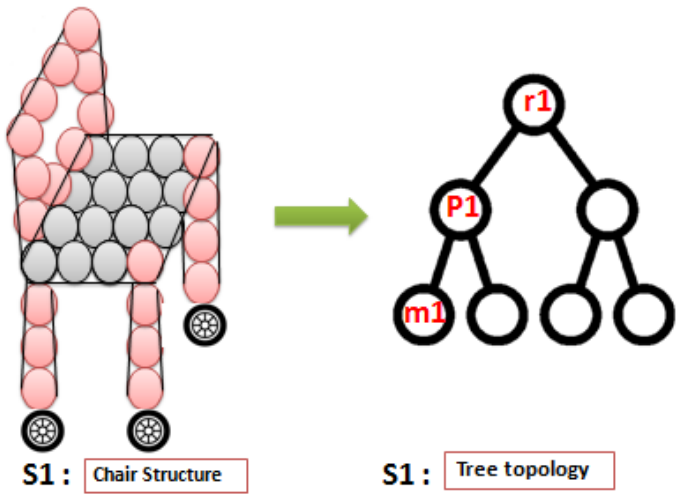


FIGURE 5.5: Convert MRS to Connected Graph.

5.3.3 FSET: Fast SET Algorithm

1. Root Selection Stage
   Mostly, the efficiency and performance of the SET algorithm are greatly affected by initial tree roots as different initial tree roots often lead to different embedding or failure in the embedding and thus, failure to obtain the new morphology. Therefore, the calculation time cost for the embedding truth table between the modules of the two robotic structures to form new morphology will be high. Hence, the selection of the initial root trees is becoming a challenge for the SET algorithm.

   The first stage of our adapted SET is called root selection and aims to solve the above problem. We propose to select a subset/training from the two sets of modules G_1 and G_2 of the connected graph that represents the two modular robotic structures, in order to find the approximate final tree roots $R = r_i, r_j$ that generate the final morphology. Our intuition is to reduce the number of iterations needed in the traditional SET to obtain the final morphology as fast as possible, thus enhancing the processing time of the SET.

   Obviously, the efficiency of the selection root stage is highly related to the percentage, represented by Ts (i.e. training size), of a training set of modules. Subsequently, increasing the value of Ts leads to an increase in the calculation time of FSET so no profit will be noticed compared to traditional SET. On the other hand, the lowest the value of Ts is the better the processing time, but the error in the final obtained morphology will increase. Therefore, selecting the appropriate value of Ts is very essential in the first stage of our technique. Indeed, we believe that Ts should be determined by the decision-makers or experts depending on the application requirements.

2. Morphology Formation Stage
   After having the approximate initial root of trees R, the second stage is lunched which aims to reduce the number of iteration loops in SET. So, it take the obtained roots in the first stage and the whole sets of modules G , and then it apply SET over G in order to get the final morphology.

   Algorithm 6 describes the procedure of the second stage of our technique. First, we determine the number of modules needed to find the tree roots in the first stage of our technique (line 2). Based on this number, we randomly select the training sets among the whole sets of modules G1 and G2 (lines 3-6). The modules in the training set represent now the approximate roots of the trees. Then, we calculate the T* embedding truth table. This process is repeated until we either get a positive answer or we exhaust all possible rootings (lines 14-20). At this moment, the first stage is accomplished and the initial roots are determined (line 21). After that, the second stage is running where the process starts by considering the roots obtained in the first stage as the initial roots of the trees. Then, we calculate the T* embedding truth table based on the obtained roots from the first stage. Again, the loop is repeated until we either get a positive answer or we exhaust all possible rootings that give us the embedding sequence that form the new morphology. (lines 21-30).

Algorithm 6 FSET Algorithm.

Require: Two sets of modules: $G_1 = \{m_1, m_2, \ldots, m_k\}$; $G_2 = \{n_1, n_2, \ldots, n_q\}$; Percentage of training set: T_s.

Ensure: Embedding truth table T* that generate the new morphology.
$G_s \leftarrow \emptyset$
$N_s \leftarrow [(T_s \times k)/100]$
for $i \leftarrow 1$ `to` N_s **do**
  // randomly selects the training set of modules among G_1 and G_2
  $G_s \leftarrow G_s U G_i$
end for
for $i \leftarrow 1$ `to` 2 **do**
  $T_i \leftarrow \emptyset$
  // randomly choose roots r_i among G_s belongs T_i
end for
// Initially, all entries are false
(i.e $T[m_i, n_j]$=false)
trace the two tree in a bottom-up to calculate the embedding truth table T between modules
repeat
  for $i \leftarrow 1$ `to` s **do**
    for $j \leftarrow 1$ `to` s **do**
      // involves a top-down message passing to calculate embedding truth table $T^*[m_i, n_j]$ between modules
    end for
  end for
until $r_t h$ row of T^* contain at least one true value or exhaust all possible rootings
extract the roots r_i and r_j from the previous T*
//use the whole set of modules G_1 and G_2
repeat
  for $i \leftarrow 1$ `to` k **do**
    for $j \leftarrow 1$ `to` q **do**
      // involves a top-down message passing to calculate embedding truth table $T^*[r_i, r_j]$ between modules
    end for
  end for
until $r_t h$ row of T^* contain at least one true value or exhaust all possible rootings
return T^* that generate the new morphology

5.4 Simulation and Results

In order to evaluate its efficiency, we tested our mechanism on one of the most used MRS proposed in recent years, e.g. M-TRAN [246]. Indeed, M-TRAN is a three-dimensional modular robotic system, with characteristics of both lattice and chain (linear) types of modular robot. Each M-TRAN module is made up of two semi-cylindrical pieces that can rotate 180 degrees around their axis and have an independent battery, two degrees of freedom motion, six surface connections, and intelligence with inter-module communication. The M-TRAN system can perform flexible and adaptive locomotion in various configurations using coordination control based on a CPG [247]. Figure 6 presents the components of a M-TRAN module.

In our simulations, we used two robotic that have chair and wall design and are made out of M-TRAN modules. We obtain a new design with stroller morphology

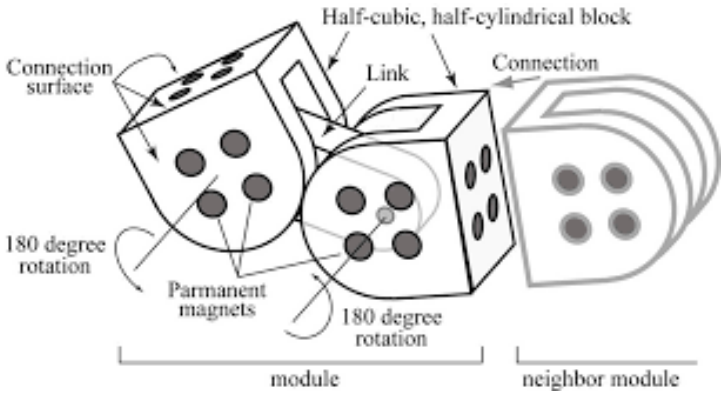


FIGURE 5.6: M-TRAN components.

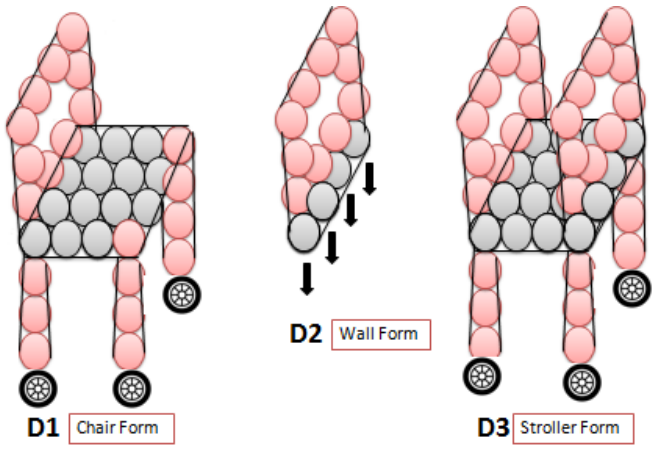


FIGURE 5.7: Morphologies adapted in our simulation.

quickly after using the FSET method (Fig. 7).

The objective of our simulations was to confirm that our technique can successfully achieve intended results for reducing the delay to obtain new morphology and reducing the energy consumption in modules that leads to extend the MRS lifetime. In order to evaluate the performance, we compare our results to the traditional SET. In our simulations, we evaluated the performance using the following parameters:

- The Number of Modules for substructure n_b, takes the following values: 100, 200, 300, 1000 and 2000.
- the Number of Modules for superstructure n_p, takes the following values: 500, 950, 2000 and 4000.
- the percentage of module chosen, T_s, takes the following values: 5, 10, 15 and 20.

5.4.1 Execution Time

Sometimes, getting new morphology fast time as possible to the end-user is a crucial operation especially in e-health and military applications. Fig. 8, shows the execution time for both FSET and SET when varying the number of modules (for both the substructure and the superstructure respectively). The results show that FSET can optimize the execution time, comparing always to the SET, from 10%

(while varying number of module from (100,500) to (300,2k) module) to 37% (while varying number of module from (500, 2k) to (2k,4k) module).

Obviously, the execution time of FSET will be highly affected by the selection of the tree roots as well as the number of iteration loops to obtain the final morphology. Therefore, FSET outperforms the SET where the processing time to get new morphology is twice accelerated when using FSET, compared to SET algorithm

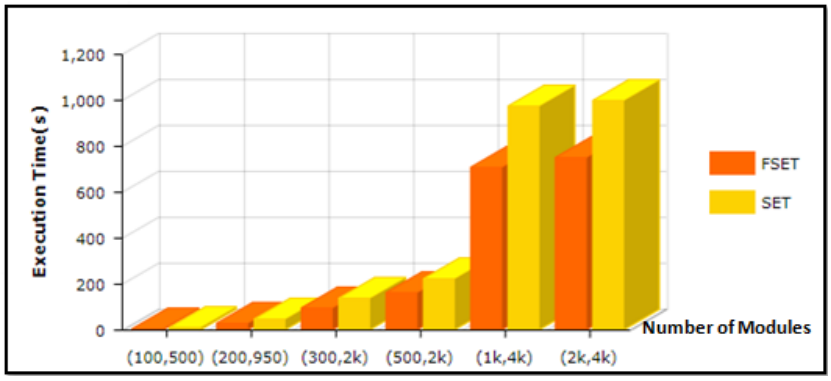


FIGURE 5.8: Processing time for FSET and SET..

5.4.2 Iteration Loop

One of the factor that can delay the obtain of new morphology is the number of iterations. In Fig. 9, we show how many iterations are generated by the the two robotic structures to find the final morphology for both FSET and the SET. It is important to know that a high number of iterations can increase the complexity of the proposed algorithm. The obtained results show that, The number of iterations is reduced by at least 30% as shown in these figure when applying FSET on the SuperBot modules. Therefore, FSET minimize the morphology delay by reducing the number of iterations.

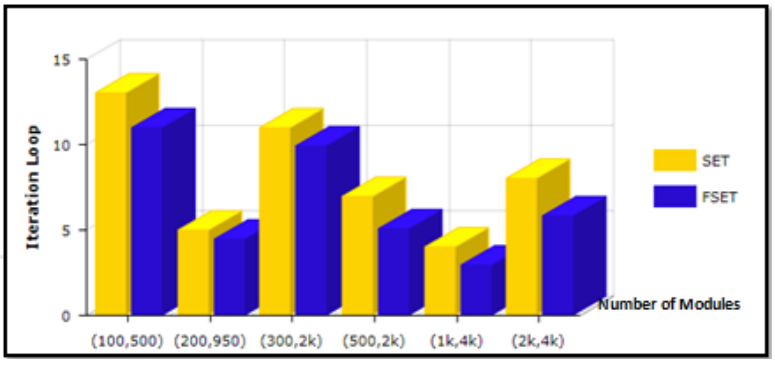


FIGURE 5.9: Iteration loop number for FSET and SET.

5.4.3 Energy Consumption

Energy consumption is a crucial parameter to assess in MRS since it impacts the overall system's functionality. Indeed, the data transmission or activity done during the transition consumes the majority of a module's limited energy. In our simulation, we implemented the same energy model that used in [240] to calculate the energy consumption in SuperBot modules, assuming that each module has fixed energy units, based on the MRS size, then we considered that each message transmission consumes 0:2 unit and each successful embedding modules consumes 0:8 unit.

Fig. 10, shows the energy consumed in the SuperBot robot depending on modules number. The obtained results show that the energy consumption increases with the increasing of the modules number while it is optimized, using FSET, up to 68% compared to the SET approach. Therefore, our proposed technique can be considered very efficiently in terms of reducing the energy consumption of the modular robotic system, thus, increasing its lifetime.

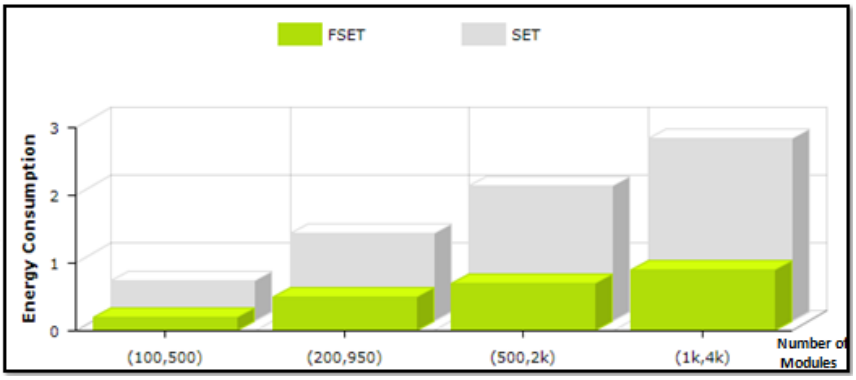


FIGURE 5.10: Energy consumption for FSET and SET.

5.5 Conclusion and Future Work

Modular robotic systems (MRSs) are one of the most advanced technologies nowadays. Therefore, researchers have paid great attention in the past years to this hot field by exploring the different challenges that cover it, while presenting different solutions. Unfortunately, MRS suffers from two major challenges: The self reconfiguration process which is a crucial process in MRS because it leads sometimes to reshape the robot into a wrong morphology thus, destruction of the MRS itself, and the energy consumption during modules transformation and communication that decreases the lifetime of the MRS.

In this chapter, we have proposed a fast self reconfiguration technique called FSET, i.e. Fast SET, dedicated to MRSs. Our proposed technique consists mainly in two stages: root selection and morphology formation. The final goal of these stages is to enhance the time cost of creating new morphology of traditional SET algorithm thus, ensure fast self reconfiguration. The root selection stage selects a small number of modules in order to find the best tree roots that effects the topological conditions that leads to successful of the embedding process or not. The morphology formation stage uses the traditional SET algorithm to calculate the embedding truth table where the initial roots used are taken from the first stage. Finally, we demonstrated

FSET's efficiency in terms of complexity, optimality (lowest number of steps), and time efficiency by simulating its performance on a real robot called SuperBot.

Chapter 6

Conclusions and Perspectives

6.1 Conclusions

Modular robotic systems hold promise for a wide range of applications, including rescue, space exploration, manufacturing, reconnaissance, and military missions, among others. MRS is one of ten new technologies that will revolutionize the world, according to MIT Technology Review.

In this thesis, we have proposed an efficient mechanism for data processing and self-reconfigurable decision making dedicated to modular robotic systems. We showed that such systems face three major challenges; first, they generate a huge amount of collected data that: first it overloads the memory storage of the robot; Second it generates redundant data which complicates the decision making about the next morphology in the controller; Third, the self reconfiguration challenge which necessitates massive communication between the modules to reach the target morphology and takes a significant processing time to self-reconfigure the robotic. In addition, it depletes the module's limited energy supply. Therefore, data storage and self-reconfigurable decision making in modular robotic systems proposed in this thesis were targeted to remove redundancy among data collected by the detectors. The goal of this reduction is to save the storage space in the MRS, and then to facilitate analyzing data and making decision about next morphology should be taken by the robot according to the surrounding. Indeed, we focused on data storage reduction, self-reconfiguration decision-making, and efficient communication management between modules in MRSs with the main goal of ensuring fast self-reconfiguration process.

First, in data storage reduction model, we have proposed to aggregate similar data while preserving the dynamicity of the monitored conditions, in order to eliminate redundant data thus, reducing the amount of data needed to be stored in MRS. According to the aggregation approach, the similarity between readings collected by a detector can be determined according to the *Sim* function based on a predefined threshold; if the difference between readings is less than the threshold then the readings are considered similar. Then we show which data will be saved in the storage space of a MRS using our reduction mechanism. In the first slot time, the readings collected by all detectors will be saved automatically in the disk storage with a weight sets to 1. Then, for the later collected readings, the controller will search the similarity between these readings and those collected in the previous slot time; if the readings are similar according to the Sim function then the controller removes the new readings while increasing the weight of the previous readings by 1 otherwise, it adds the new collected readings to the disk storage and initializes its weight to 1. We

showed via simulations on real detector data, that our approach can be effectively used to reduce the data storage and improving the self-reconfiguration decision.

Second, we have proposed a robust mechanism for self-reconfigurable robotics called RUN with the main objective is to reduce the communications in MRS and provide a fast reconfiguration process through two stages. At the first stage, we classify the modules into clusters before performing their transitions. To do that, we first select a set of modules to act as cliques for the robotic. A clique module is responsible to allow an efficient communication between the whole modules during the self-reconfiguration process. On one hand, this will allow to reduce the number of messages transmitted between the modules and, on the other hand, to minimize the number of transitions made by each module (thus saving its energy). Then, once the clusters are formulated, we introduce two communication algorithms in the second stage; the first algorithm is inter-module that allows efficient communication between the cliques of the clusters and aims to minimize the number of transmitted messages in the robotic system to avoid packet congestion and save the module energy, and the second algorithm is intra-module based on the tree structure that reduces the number of communications between the modules in the same clusters. In the proposed scheme, we consider the clique as the root of the tree while the modules are reorganized into parent-child relationships. Indeed, the construction of any tree suffers from the number of relayed packets that can deplete the available energy of modules. Particularly, the modules having more children or those close to the clique will be firstly died because of the significant number of relayed packets. In order to overcome such problem, we propose an energy-balanced and efficient-communication tree that maximizes the lifetime of the MRS and provide a fast reconfiguration process of their modules. Compared to other existing techniques, we showed through simulations on real robot, known as Roombots, the efficiency of our proposed technique in terms of complexity, overhead communication, optimality (minimum number of steps), and time efficiency.

In a third step, we have proposed a fast self-reconfiguration technique called FSET, dedicated to MRS. Our proposed technique consists mainly of two stages: root selection and morphology formation. The final goal of these stages is to enhance the time cost to get new morphology of the traditional SET algorithm thus, ensuring fast self-reconfiguration. The root selection stage selects a small number of modules in order to find the best tree roots that affect the topological conditions and lead to the success of the embedding process or not. The morphology formation stage uses the traditional SET algorithm to calculate the embedding truth table where the initial roots used are taken from the first stage. Finally, our proposed technique, under the two proposed strategies, has been evaluated through simulations on a real scenario using M-TRAN, where the obtained results were very encouraged in terms of providing a fast reconfiguration process in MRS and reducing the energy consumption of modules.

6.2 Perspectives

As perspectives of this thesis, we propose two categories. The first one is direct perspectives which are related to the techniques proposed in this work. While the second one is general perspectives and open issues in self reconfiguration for MRS.

6.2.1 Direct Perspectives

In this section, we give some perspectives in order to improve, extend or continue the proposed techniques on self-reconfiguration process, and modules communication management presented in this work.

First, we seek to adapt our self-reconfiguration mechanism to use machine learning algorithms in order to allow MRS to predict sensed data then to self-reconfigured accordingly.

Second, we seek to enhance the decision making mechanism in a way to select the best morphology, according to some criteria's (like energy, delay, etc.), in case the MRS can take many morphologies at the same time.

Third, we seek to apply RUN on other real world scenarios in order to validate its efficiency regarding different existing modular robotics such as Polybot, Kilobot, Sambot, etc.

In addition, we seek to adapt our FSET mechanism to investigate how to reduce the kinematic checking runtime. We'll move away from detecting embeddability and toward design synthesis in the long run. We feel our embedding strategy is a good place to start for this line of research, and preliminary results are promising.

Finally, it is interesting to perform real experiments in order to evaluate the performance of our proposed techniques in real world applications.

6.2.2 General Perspectives and Open Issues

Despite the vast amount of studies on self-reconfiguration in modular robotic systems, the field is still substantially unexplored. There are a number of important open research questions in self reconfiguration that have not yet been fully investigated or sometimes need to be more explored. In order to enhance the performance of such systems, particularly in "self-reconfiguration decision-making, time synchronization, data analysis and energy consumption," researchers should focus more on these areas.

First, the difficulties of improving MRS go beyond simply designing dependable, responsive, strong, and scalable hardware components. They also require creating solvers in the form of planning/control methods and algorithms that can increase the adaptability of such systems. We can benefits from integrating AI in modular robotic system to increasing the cognitive abilities of robots for better decision-making capabilities about next morphology should be taken to perform the new task according to its surrounding changes. We will study the Critical success factors for integrating artificial intelligence and modular robotic systems. Thus integrating AI in modular robotic systems will enable various useful applications. As a future work, we may focus on trying to apply AI methods to manage self reconfigurable modular robotic system in different areas such as space exploration, environment, healthcare, and industrial and designing novel solutions to the challenging problems.

Second, time synchronization is a significant challenge in MRSs. In particular, we plan to address the issue of time synchronization during the self-reconfiguration process, in which modules move to reorganize the modular robot's overall structure

in order to complete the essential task. since any module mobility and failures may occur frequently can change the time synchronization at the modules which raises a problem in self-reconfiguration decision-making. Therefore, more techniques need to be proposed in order to guarantee an accurate time module mobility in MRSs.

Finally, the detectors of our MRS are collecting huge amounts of data from different fields because it works in periodic way, and due to their limited memory, computation capabilities, and battery lifetime, it is difficult to sense, store, transfer, and analyze such amount of big data. Nowadays, recent advances in big data are allowing huge amounts of data to be properly captured, structured, processed, and stored. Therefore, big data technology is complementing these smart detectors of MRSs. Thus integrating these two technologies will enable various useful applications. As a future work, we may focus on use Edge AI technologies have emerged as a powerful solution for gathering and processing data in real-time, supporting the development of advanced applications for process monitoring, planning and control.

PUBLICATION

ACCEPTED AND PUBLISHED JOURNAL

[1] Aliah Majed, Hassan Harb, Abbass Nasser, Benoit Clement and Olivier Reynet. Sensing-Based Self-Reconfigurable Decision-Making Mechanism for Autonomous Modular Robotic System. IEEE Sensors Journal, IEEE publisher, Vol. 20, number=13, pages 7097–7106, 2020.

SUBMITTED JOURNAL AND CONFERENCE

[2] Aliah Majed, Hassan Harb, Abbass Nasser, Benoit Clement and Olivier Reynet. RUN: A Robust Cluster-Based Planning for Fast Self-Reconfigurable Modular Robotic Systems. Submitted to Intelligent Service Robotics (Springer Nature Journals).

[3] Aliah Majed, Hassan Harb, Abbass Nasser, Benoit Clement and Olivier Reynet. FSET: Fast Structure Embedding Technique for Self-Reconfigurable Modular Robotic Systems. Submitted to 37th International Conference on Advanced Information Networking and Applications (AINA-2023).

Bibliography

[1] A. Faíña, F. Bellas, F. López-Peña, and R. J. Duro, "Edhmor: Evolutionary designer of heterogeneous modular robots," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2408–2423, 2013.

[2] E. Papadopoulos, F. Aghili, O. Ma, and R. Lampariello, "Robotic manipulation and capture in space: A survey," *Frontiers in Robotics and AI*, p. 228, 2021.

[3] P. Fournier-Viger, R. Nkambou, and A. Mayers, "Evaluating spatial representations and skills in a simulator-based tutoring system," *IEEE Transactions on Learning Technologies*, vol. 1, no. 1, pp. 63–74, 2008.

[4] J. Bourgeois, B. Piranda, A. Naz, N. Boillot, H. Mabed, D. Dhoutaut, T. Tucci, and H. Lakhlef, "Programmable matter as a cyber-physical conjugation," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2016, pp. 002 942–002 947.

[5] A. Naz, "Distributed algorithms for large-scale robotic ensembles: centrality, synchronization and self-reconfiguration," Ph.D. dissertation, Université Bourgogne Franche-Comté, 2017.

[6] P. Thalamy, B. Piranda, and J. Bourgeois, "A survey of autonomous self-reconfiguration methods for robot-based programmable matter," *Robotics and Autonomous Systems*, vol. 120, p. 103242, 2019.

[7] Y. Mantzouratos, T. Tosun, S. Khanna, and M. Yim, "On embeddability of modular robot designs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1911–1918.

[8] G. Huang, "10 emerging technologies that will change your world," *Technology Review*, vol. 107, no. 1, pp. 32–+, 2004.

[9] H. Ahmadzadeh and E. Masehian, "Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization," *Artificial Intelligence*, vol. 223, pp. 27–64, 2015.

[10] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. D. Schutter, "A comparison of decision making criteria and optimization methods for active robotic sensing," in *International Conference on Numerical Methods and Applications*. Springer, 2002, pp. 316–324.

[11] A. Eguchi, "Educational robotics to promote 21 st century skills and technological understanding among underprivileged undergraduate students," in *2015 IEEE Integrated STEM Education Conference*. IEEE, 2015, pp. 76–82.

[12] M. D. Hancher and G. S. Hornby, "A modular robotic system with applications to space exploration," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*. IEEE, 2006, pp. 8–pp.

[13] J. Seo, S. Gray, V. Kumar, and M. Yim, "Reconfiguring chain-type modular robots based on the carpenter's rule theorem," in *Algorithmic Foundations of Robotics IX*. Springer, 2010, pp. 105–120.

[14] Y. Zhao, Y. Chen, B. Li, and Q. Zhang, "Hop id: A virtual coordinate based routing for sparse mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1075–1089, 2007.

[15] R. A. Brooks, "New approaches to robotics," *Science*, vol. 253, no. 5025, pp. 1227–1232, 1991.

[16] M. Stieber, C. Trudel, and D. Hunter, "Robotic systems for the international space station," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. IEEE, 1997, pp. 3068–3073.

[17] S. Tibbits, C. McKnelly, C. Olguin, D. Dikovsky, and S. Hirsch, "4d printing and universal transformation," 2014.

[18] W. McCarthy, "Programmable matter," *Nature*, vol. 407, no. 6804, pp. 569–569, 2000.

[19] Y. Ke, L. L. Ong, W. M. Shih, and P. Yin, "Three-dimensional structures self-assembled from dna bricks," *science*, vol. 338, no. 6111, pp. 1177–1183, 2012.

[20] M. A. McEvoy and N. Correll, "Materials that couple sensing, actuation, computation, and communication," *Science*, vol. 347, no. 6228, p. 1261689, 2015.

[21] M. Lasagni and K. Römer, "Dynamic model of tendon-driven robotic chains forming a shape-shifting surface," in *Smart Materials, Adaptive Structures and Intelligent Systems*, vol. 50497. American Society of Mechanical Engineers, 2016, p. V002T03A024.

[22] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[23] E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood, "Programmable matter by folding," *Proceedings of the National Academy of Sciences*, vol. 107, no. 28, pp. 12 441–12 445, 2010.

[24] S. C. Goldstein and T. C. Mowry, "Claytronics: An instance of programmable matter," *Wild and crazy ideas session of ASPLOS*, vol. 12, p. 1, 2004.

[25] K. Gilpin, A. Knaian, and D. Rus, "Robot pebbles: One centimeter modules for programmable matter through self-disassembly," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2485–2492.

[26] B. Piranda and J. Bourgeois, "Geometrical study of a quasi-spherical module for building programmable matter," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 387–400.

[27] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[28] B. Benhabib and M. Dai, "Mechanical design of a modular robot for industrial applications," *Journal of Manufacturing Systems*, vol. 10, no. 4, pp. 297–306, 1991.

[29] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable manufacturing systems," *CIRP annals*, vol. 48, no. 2, pp. 527–540, 1999.

[30] T. Fukuda and S. Nakagawa, "Approach to the dynamically reconfigurable robotic system," *Journal of Intelligent and Robotic Systems*, vol. 1, no. 1, pp. 55–72, 1988.

[31] D. Schmitz, P. Khosla, and T. Kanade, "The cmu reconfigurable modular manipulator system," 1988.

[32] M. Yim, "A reconfigurable modular robot with many modes of locomotion," in *International Conference on Advanced Mechatronics*. IEEE, 1993, pp. 283–288.

[33] S. Murata, H. Kurokawa, and S. Kokaji, "Self-assembling machine," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 441–448.

[34] S. Hirose, T. Shirasu, and E. F. Fukushima, "Proposal for cooperative robot "gunryu" composed of autonomous segments," *Robotics and Autonomous Systems*, vol. 17, no. 1-2, pp. 107–118, 1996.

[35] G. J. Hamlin and A. C. Sanderson, "Tetrobot modular robotics: Prototype and experiments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, vol. 2. IEEE, 1996, pp. 390–395.

[36] K. Kotay, D. Rus, M. Vona, and C. McGray, "The self-reconfiguring robotic molecule," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 1. IEEE, 1998, pp. 424–431.

[37] H. Kurokawa, S. Murata, E. Yoshida, K. Tomita, and S. Kokaji, "A 3-d self-reconfigurable structure and experiments," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, vol. 2. IEEE, 1998, pp. 860–865.

[38] K. Hosokawa, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, and I. Endo, "Self-organizing collective robots with morphogenesis in a vertical plane," *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing*, vol. 42, no. 1, pp. 195–202, 1999.

[39] P. M. Will, A. Castaño, and W.-M. Shen, "Robot modularity for self-reconfiguration," in *Sensor Fusion and Decentralized Control in Robotic Systems II*, vol. 3839. International Society for Optics and Photonics, 1999, pp. 236–245.

[40] E. Yoshida, S. Kokaji, S. Murata, H. Kurokawa, and K. Tomita, "Miniaturized self-reconfigurable system using shape memory alloy," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, vol. 3. IEEE, 1999, pp. 1579–1585.

[41] R. L. Tummala, R. Mukherjee, D. Aslam, N. Xi, S. Mahadevan, and J. Weng, "Reconfigurable adaptable micro-robot," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, vol. 6. IEEE, 1999, pp. 687–691.

[42] C. Unsal, H. Kiliccote, and P. K. Khosla, "I (ces)-cubes: a modular self-reconfigurable bipartite robotic system," in *Sensor Fusion and Decentralized Control in Robotic Systems II*, vol. 3839. International Society for Optics and Photonics, 1999, pp. 258–269.

[43] M. Yim, D. G. Duff, and K. D. Roufas, "Polybot: a modular reconfigurable robot," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 514–520.

[44] E. Yoshida, S. Murata, S. Kokaji, K. Tomita, and H. Kurokawa, "Micro self-reconfigurable robotic system using shape memory alloy," in *Distributed Autonomous Robotic Systems 4*. Springer, 2000, pp. 145–154.

[45] H. Kurokawa, K. Tomita, E. Yoshida, S. Murata, and S. Kokaji, "Motion simulation of a modular robotic system," in *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies*, vol. 4. IEEE, 2000, pp. 2473–2478.

[46] D. Rus and M. Vona, "A physical implementation of the self-reconfiguring crystalline robot," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 1726–1733.

[47] C. Khairallah, "Modular articulated robot structure," Nov. 27 2001, uS Patent 6,323,615.

[48] D. Rus, Z. Butler, K. Kotay, and M. Vona, "Self-reconfiguring robots," *Communications of the ACM*, vol. 45, no. 3, pp. 39–45, 2002.

[49] J. W. Suh, S. B. Homans, and M. Yim, "Telecubes: Mechanical design of a module for self-reconfigurable robotics," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 4. IEEE, 2002, pp. 4095–4101.

[50] A. Kawakami, A. Torii, K. Motomura, and S. Hirose, "Smc rover: planetary rover with transformable wheels," in *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, vol. 1. IEEE, 2002, pp. 157–162.

[51] N. Inou, K. Minami, and M. Koseki, "Group robots forming a mechanical structure-development of slide motion mechanism and estimation of energy consumption of the structural formation," in *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No. 03EX694)*, vol. 2. IEEE, 2003, pp. 874–879.

[52] F. Mondada, A. Guignard, M. Bonani, D. Bar, M. Lauria, and D. Floreano, "Swarm-bot: From concept to implementation," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1626–1631.

[53] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata, "M-tran ii: Metamorphosis from a four-legged walker to a caterpillar," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2454–2459.

[54] M. W. Jorgensen, E. H. Ostergaard, and H. H. Lund, "Modular atron: Modules for a self-reconfigurable robot," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. Ieee, 2004, pp. 2068–2073.

[55] S. C. Goldstein and T. C. Mowry, "Claytronics: An instance of programmable matter," *Wild and crazy ideas session of ASPLOS*, vol. 12, p. 1, 2004.

[56] P. White, K. Kopanski, and H. Lipson, "Stochastic self-reconfigurable cellular robotics," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3. IEEE, 2004, pp. 2888–2893.

[57] R. Watanabe, Y. Itoh, M. Asai, Y. Kitamura, F. Kishino, and H. Kikuchi, "The soul of activecube: implementing a flexible, multimodal, three-dimensional spatial tangible interface," *Computers in Entertainment (CIE)*, vol. 2, no. 4, pp. 15–15, 2004.

[58] H. S. Raffle, A. J. Parkes, and H. Ishii, "Topobo: a constructive assembly system with kinetic memory," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004, pp. 647–654.

[59] J. Gonzalez-Gomez, E. Aguayo, and E. Boemo, "Locomotion of a modular worm-like robot using a fpga-based embedded microblaze soft-processor," in *Climbing and walking robots*. Springer, 2005, pp. 869–878.

[60] T. Bertolote and V. Hentsch, "Design and prototyping of an underwater modular robot," *Unpublished Master Thesis, Laboratory of Intelligent Systems, EPFL*, 2004.

[61] S. T. Griffith, "Growing machines," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[62] L. Zhang, J. Zhao, and H. G. Cai, "A substructure based motion planning method for a modular self-reconfigurable robot," in *Proceedings of the Fourth International Workshop on Robot Motion and Control (IEEE Cat. No. 04EX891)*. IEEE, 2004, pp. 371–376.

[63] W.-M. Shen, J. Bogdanowicz, W. Chun, M. Yim, P. M. Will, M. Sims, S. Colombano, D. Kortenkamp, S. Vanderzyl, E. Baumgartener *et al.*, "Superbots: Modular, multifunctional, reconfigurable robotic system for space exploration," *LPICo*, vol. 1287, p. 80, 2005.

[64] B. Kirby, J. Campbell, B. Aksak, P. Pillai, J. Hoburg, T. Mowry, and S. C. Goldstein, "Catoms: Moving robots without moving parts," in *Proceedings of the national conference on artificial intelligence*, vol. 20, no. 4. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005, p. 1730.

[65] P. White, V. Zykov, J. C. Bongard, and H. Lipson, "Three dimensional stochastic reconfiguration of modular robots." in *Robotics: Science and Systems*. Cambridge, 2005, pp. 161–168.

[66] E. H. Østergaard, D. J. Christensen, P. Eggenberger, T. Taylor, P. Ottery, and H. H. Lund, "Hydra: From cellular biology to shape-changing artefacts," in *International Conference on Artificial Neural Networks*. Springer, 2005, pp. 275–281.

[67] J. Bishop, S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen, "Programmable parts: A demonstration of the grammatical approach to self-organization," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3684–3691.

[68] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson, "Self-reproducing machines," *Nature*, vol. 435, no. 7039, pp. 163–164, 2005.

[69] M. Shimizu, A. Ishiguro, and T. Kawakatsu, "A modular robot that exploits a spontaneous connectivity control mechanism," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1899–1904.

[70] A. Brunete, M. Hernando, and E. Gambao, "Modular multiconfigurable architecture for low diameter pipe inspection microrobots," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 490–495.

[71] E. Schweikardt and M. D. Gross, "roblocks: a robotic construction kit for mathematics and science education," in *Proceedings of the 8th international conference on Multimodal interfaces*, 2006, pp. 72–75.

[72] A. Parkes, V. LeClerc, and H. Ishii, "Glume: exploring materiality in a soft augmented modular modeling system," in *CHI'06 extended abstracts on Human factors in computing systems*, 2006, pp. 1211–1216.

[73] Y. Terada and S. Murata, "Modular stucture assembly using blackboard path planning systems," in *International Symposium on Automation and Robotics in Construction*, 2006, pp. 852–857.

[74] M. T. G. V. M. Hossain, "Climbing and walking robots," 2006.

[75] J. Dietsch, R. Moeckel, C. Jaquier, K. Drapel, E. Dittrich, A. Upegui, and A. J. Ijspeert, "Exploring adaptive locomotion with yamor, a novel autonomous modular robot with bluetooth interface," *Industrial Robot: An International Journal*, 2006.

[76] K. Stoy, "The deformatron robot: a biologically inspired homogeneous modular robot," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2527–2531.

[77] B. R. Donald, C. G. Levey, C. D. McGray, I. Paprotny, and D. Rus, "An untethered, electrostatic, globally controllable mems micro-robot," *Journal of microelectromechanical systems*, vol. 15, no. 1, pp. 1–15, 2006.

[78] J. A. Escalera, M. Ferre, R. Aracil, and J. Baca, "Robmat: Teleoperation of a modular robot for collaborative manipulation," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2007, pp. 1204–1213.

[79] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu, "Miche: Modular shape formation by self-disassembly," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 345–372, 2008.

[80] S. Murata, K. Kakomura, and H. Kurokawa, "Toward a scalable modular robotic system," *IEEE Robotics & Automation Magazine*, vol. 14, no. 4, pp. 56–63, 2007.

[81] J. Liu, Y. Wang, B. Li, S. Ma, and D. Tan, "Center-configuration selection technique for the reconfigurable modular robot," *Science in China Series F: Information Sciences*, vol. 50, no. 5, pp. 697–710, 2007.

[82] Y. Yoon and D. Rus, "Shady3d: A robot that climbs 3d trusses," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 4071–4076.

[83] B. L. Carmichael and C. M. Gifford, "Modeling and simulation of the seismic tetwalker concept," *Center for Remote Sensing of Ice Sheets, University of Kansas, KS, Tech. Rep., CReSIS TR*, vol. 134, 2007.

[84] V. LeClerc, A. Parkes, and H. Ishii, "Senspectra: A computationally augmented physical modeling toolkit for sensing and visualization of structural strain," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007, pp. 801–804.

[85] M. Koseki, K. Minami, and N. Inou, "Cellular robots forming a mechanical structure," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 139–148.

[86] P. J. White and M. Yim, "Scalable modular self-reconfigurable robots using external actuation," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 2773–2778.

[87] S. Miyashita, M. Hadorn, and P. E. Hotz, "Water floating self-assembling agents," in *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*. Springer, 2007, pp. 665–674.

[88] A. Lyder, R. F. M. Garcia, and K. Stoy, "Mechanical design of odin, an extendable heterogeneous deformable modular robot," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ieee, 2008, pp. 883–888.

[89] V. Zykov, W. Phelps, N. Lassabe, and H. Lipson, "Molecubes extended: Diversifying capabilities of open-source modular robotics," in *IROS-2008 Self-Reconfigurable Robotics Workshop*, 2008, pp. 22–26.

[90] B. Schiettecatte and J. Vanderdonckt, "Audiocubes: a distributed cube tangible interface based on interaction range for sound design," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*, 2008, pp. 3–10.

[91] B. K. An, "Em-cube: cube-shaped, self-reconfigurable robots sliding on structure surfaces," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 3149–3155.

[92] M. P. Weller, E. Y.-L. Do, and M. D. Gross, "Posey: instrumenting a poseable hub and strut construction toy," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*, 2008, pp. 39–46.

[93] S. Kernbach, E. Meister, F. Schlachter, K. Jebens, M. Szymanski, J. Liedke, D. Laneri, L. Winkler, T. Schmickl, R. Thenius *et al.*, "Symbiotic robot organisms: Replicator and symbrion projects," in *Proceedings of the 8th workshop on performance metrics for intelligent systems*, 2008, pp. 62–69.

[94] H. Zhang, J. Gonzalez-Gomez, Z. Me, S. Cheng, and J. Zhang, "Development of a low-cost flexible modular robot gz-i," in *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2008, pp. 223–228.

[95] C.-H. Yu, K. Haller, D. Ingber, and R. Nagpal, "Morpho: A self-deformable modular robot inspired by cellular structure," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3571–3578.

[96] J. Nielsen and H. H. Lund, "Modular robotics as a tool for education and entertainment," *Computers in human behavior*, vol. 24, no. 2, pp. 234–248, 2008.

[97] A. Sproewitz, A. Billard, P. Dillenbourg, and A. J. Ijspeert, "Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 4259–4264.

[98] J. Mampel, K. Gerlach, C. Schilling, and H. Witte, "A modular robot climbing on pipe-like structures," in *2009 4th International Conference on Autonomous Robots and Agents*. IEEE, 2009, pp. 87–91.

[99] H. H. Lund and P. Marti, "Designing modular robotic playware," in *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2009, pp. 115–121.

[100] R. Belisle, C.-H. Yu, and R. Nagpal, "Mechanical design and locomotion of modular-expanding robots," 2010.

[101] H. Wei, Y. Chen, J. Tan, and T. Wang, "Sambot: A self-assembly modular robot system," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 4, pp. 745–757, 2010.

[102] K. Gilpin, A. Knaian, and D. Rus, "Robot pebbles: One centimeter modules for programmable matter through self-disassembly," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2485–2492.

[103] G. G. Ryland and H. H. Cheng, "Design of imobot, an intelligent reconfigurable mobile robot with novel locomotion," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 60–65.

[104] A. Lyder, R. F. M. Garcia, and K. Stoy, "Genderless connection mechanism for modular robots introducing torque transmission between modules," in *Proceedings of the ICRA Workshop on Modular Robots, State of the Art*, 2010, pp. 77–81.

[105] R. F. M. Garcia, J. D. Hiller, K. Stoy, and H. Lipson, "A vacuum-based bonding mechanism for modular robotics," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 876–890, 2011.

[106] K. C. Galloway, R. Jois, and M. Yim, "Factory floor: A robotically reconfigurable construction platform," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2467–2472.

[107] J. Zhao, X. Cui, Y. Zhu, and S. Tang, "A new self-reconfigurable modular robotic system ubot: Multi-mode locomotion and self-reconfiguration," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1020–1025.

[108] M. T. Tolley and H. Lipson, "Fluidic manipulation for scalable stochastic 3d assembly of modular robots," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 2473–2478.

[109] S. Rasakatla, K. M. Krishna, and B. Indurkhya, "" mod-leg" a modular legged robotic system," in *ACM SIGGRAPH 2010 Posters*, 2010, pp. 1–1.

[110] R. Merali and D. Long, "Actuated responsive truss," *Modular Robots: The State of the Art*, vol. 36, 2010.

[111] M. Rubenstein and R. Nagpal, "Kilobot: a robotic module for demonstrating behaviors in a large scale (\(2^{10}\) units) collective," in *Proceedings of the IEEE 2010 international conference on robotics and automation workshop, modular robotics: state of the art*. Institute of Electrical and Electronics Engineers, 2010.

[112] E. Schweikardt, "Modular robotics studio," in *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, 2010, pp. 353–356.

[113] Q. Wu, Y. Luo, X. Chi, and X. Lou, "Motion error analysis of modular self-reconfigurable robot m-cubes based screw theory," in *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, vol. 2. IEEE, 2011, pp. 859–866.

[114] Y. Meng, Y. Zhang, A. Sampath, Y. Jin, and B. Sendhoff, "Cross-ball: a new morphogenetic self-reconfigurable modular robot," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 267–272.

[115] H. Sadjadi, O. Mohareri, M. A. Al-Jarrah, and K. Assaleh, "Design and implementation of hexbot: A modular self-reconfigurable robotic system," *Journal of the Franklin Institute*, vol. 349, no. 7, pp. 2281–2293, 2012.

[116] W. Hong, S. Wang, and D. Shui, "Reconfigurable robot system based on electromagnetic design," in *Proceedings of 2011 International Conference on Fluid Power and Mechatronics*. IEEE, 2011, pp. 570–575.

[117] E. Guan, Z. Fu, W. Yan, D. Jiang, and Y. Zhao, "Self-reconfiguration path planning design for m-lattice robot based on genetic algorithm," in *International Conference on Intelligent Robotics and Applications*. Springer, 2011, pp. 505–514.

[118] J. Baca, M. Ferre, and R. Aracil, "A heterogeneous modular robotic design for fast response to a diversity of tasks," *Robotics and Autonomous Systems*, vol. 60, no. 4, pp. 522–531, 2012.

[119] Y. Zhu, X. Wang, X. Cui, J. Yin, and J. Zhao, "Research on locomotive evolution based on worm-shaped configuration of self-reconfigurable robot hitmsr ii," in *Electrical Power Systems and Computers*. Springer, 2011, pp. 245–252.

[120] A. Faiña, F. Orjales, F. Bellas, R. Duro *et al.*, "First steps towards a heterogeneous modular robotic architecture for intelligent industrial operation," in *Workshop on Reconfigurable Modular Robotics at the IROS*, 2011.

[121] W. B. Goh, L. C. Kasun, J. Tan, and W. Shou, "The i-cube: design considerations for block-based digital manipulatives and their applications," in *Proceedings of the Designing Interactive Systems Conference*, 2012, pp. 398–407.

[122] S. Mobes, G. J. Laurent, C. Clevy, N. Le Fort-Piat, B. Piranda, and J. Bourgeois, "Toward a 2d modular and self-reconfigurable robot for conveying microparts," in *2012 Second Workshop on Design, Control and Software Implementation for Distributed MEMS*. IEEE, 2012, pp. 7–13.

[123] J. Davey, N. Kwok, and M. Yim, "Emulating self-reconfigurable robots-design of the smores system," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4464–4469.

[124] K. C. Wolfe, M. S. Moses, M. D. Kutzer, and G. S. Chirikjian, "M 3 express: a low-cost independently-mobile reconfigurable modular robot," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2704–2710.

[125] S. Hossain, C. A. Nelson, and P. Dasgupta, "Hardware design and testing of modred: A modular self-reconfigurable robot system," in *Advances in Reconfigurable Mechanisms and Robots I*. Springer, 2012, pp. 515–523.

[126] J. W. Romanishin, K. Gilpin, and D. Rus, "M-blocks: Momentum-driven, magnetic modular robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4288–4295.

[127] M. Pacheco, M. Moghadam, A. Magnússon, B. Silverman, H. H. Lund, and D. J. Christensen, "Fable: Design of a modular robotic playware platform," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 544–550.

[128] G. Qiao, G. Song, W. Wang, Y. Zhang, and Y. Wang, "Design and implementation of a modular self-reconfigurable robot," *International Journal of Advanced Robotic Systems*, vol. 11, no. 3, p. 47, 2014.

[129] C. Wu, X.-y. Wang, G.-j. Zhuang, M. Zhao, and T. Ge, "Motion of an underwater self-reconfigurable robot with tree-like configurations," *Journal of Shanghai Jiaotong University (Science)*, vol. 18, no. 5, pp. 598–605, 2013.

[130] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME transactions on mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[131] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata, "M-tran ii: metamorphosis from a four-legged walker to a caterpillar," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2454–2459.

[132] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Distributed self-reconfiguration of m-tran iii modular robotic system," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 373–386, 2008.

[133] M. Yim, D. G. Duff, and K. D. Roufas, "Polybot: a modular reconfigurable robot," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 514–520.

[134] V. Zykov, A. Chan, and H. Lipson, "Molecubes: An open-source modular robotics kit," in *IROS-2007 Self-Reconfigurable Robotics Workshop*. Citeseer, 2007, pp. 3–6.

[135] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, "Design of the atron lattice-based self-reconfigurable robot," *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, 2006.

[136] M. Yim, "A reconfigurable modular robot with multiple modes of locomotion," in *Proc. of the 1993 JSME Conference on Advanced Mechatronics, Tokyo, Japan*, 1993.

[137] P. White, V. Zykov, J. C. Bongard, H. Lipson *et al.*, "Three dimensional stochastic reconfiguration of modular robots." in *Robotics: Science and Systems*. Citeseer, 2005, pp. 161–168.

[138] A. Faíña, F. Bellas, F. López-Peña, and R. J. Duro, "Edhmor: Evolutionary designer of heterogeneous modular robots," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2408–2423, 2013.

[139] M. Yim, Y. Zhang, and D. Duff, "Modular robots," *IEEE Spectrum*, vol. 39, no. 2, pp. 30–34, 2002.

[140] 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]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[141] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, "Design of the atron lattice-based self-reconfigurable robot," *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, 2006.

[142] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME transactions on mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[143] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata, "M-tran ii: metamorphosis from a four-legged walker to a caterpillar," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2454–2459.

[144] M. Yim, C. Eldershaw, Y. Zhang, and D. Duff, "Self-reconfigurable robot systems: Polybot," *Journal of the Robotics Society of Japan*, vol. 21, no. 8, pp. 851–854, 2003.

[145] A. Golovinsky, M. Yim, Y. Zhang, C. Eldershaw, and D. Duff, "Polybot and polykinetic/spl trade/system: a modular robotic platform for education," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 2. IEEE, 2004, pp. 1381–1386.

[146] D. Duff, M. Yim, and K. Roufas, "Evolution of polybot: A modular reconfigurable robot," in *Proc. of the Harmonic Drive Intl. Symposium, Nagano, Japan*. Citeseer, 2001.

[147] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw, "Connecting and disconnecting for chain self-reconfiguration with polybot," *IEEE/ASME Transactions on mechatronics*, vol. 7, no. 4, pp. 442–451, 2002.

[148] J. Zhao, X. Cui, Y. Zhu, and S. Tang, "A new self-reconfigurable modular robotic system ubot: Multi-mode locomotion and self-reconfiguration," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1020–1025.

[149] T. Fukuda and Y. Kawauchi, "Cellular robotic system (cebot) as one of the realization of self-organizing intelligent universal manipulator," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, 1990, pp. 662–667.

[150] T. FUKUDA and S. NAKAGAWA, "Method of autonomous approach, docking and detaching between cells for dynamically recontigurable robotic system cebot," *JSME international journal. Ser. 3, Vibration, control engineering, engineering for industry*, vol. 33, no. 2, pp. 263–268, 1990.

[151] T. Fukuda, M. Buss, H. Hosokai, and Y. Kawauchi, "Cell structured robotic system cebot: control, planning and communication methods," *Robotics and autonomous systems*, vol. 7, no. 2-3, pp. 239–248, 1991.

[152] H. Lakhlef, H. Mabed, and J. Bourgeois, "Distributed and efficient algorithm for self-reconfiguration of mems microrobots," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 560–566.

[153] M. Park, S. Chitta, A. Teichman, and M. Yim, "Automatic configuration recognition methods in modular robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 403–421, 2008.

[154] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[155] F. Hou and W.-M. Shen, "Graph-based optimal reconfiguration planning for self-reconfigurable robots," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 1047–1059, 2014.

[156] K. Römer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks." *Handbook of sensor networks*, vol. 33, no. 1, pp. 199–237, 2005.

[157] G. Li, I. Svogor, and G. Beltrame, "Long-term pattern formation and maintenance for battery-powered robots," *Swarm Intelligence*, vol. 13, no. 1, pp. 21–57, 2019.

[158] H. Kurokawa, K. Tomita, E. Yoshida, S. Murata, and S. Kokaji, "Motion simulation of a modular robotic system," in *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies*, vol. 4. IEEE, 2000, pp. 2473–2478.

[159] D. Marbach and A. J. Ijspeert, "Co-evolution of configuration and control for homogenous modular robots," in *Proceedings of the eighth conference on intelligent autonomous systems (IAS8)*, no. CONF. IOS Press, 2004, pp. 712–719.

[160] O. Michel, "Cyberbotics ltd. webots: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[161] D. Christensen, D. Brandt, K. Stoy, and U. P. Schultz, "A unified simulator for self-reconfigurable robots," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 870–876.

[162] V. Zykov, W. Phelps, N. Lassabe, and H. Lipson, "Molecubes extended: Diversifying capabilities of open-source modular robotics," in *IROS-2008 Self-Reconfigurable Robotics Workshop*, 2008, pp. 22–26.

[163] L. Winkler and H. Wörn, "Symbricator3d–a distributed simulation environment for modular robots," in *International Conference on Intelligent Robotics and Applications*. Springer, 2009, pp. 1266–1277.

[164] J. Gonzalez-Gomez, J. Gonzalez-Quijano, H. Zhang, and M. Abderrahim, "Toward the sense of touch in snake modular robots for search and rescue operations," in *Proc. ICRA 2010 Workshop "Modular Robots: State of the Art*, 2010, pp. 63–68.

[165] T. Collins, N. O. Ranasinghe, and W.-M. Shen, "Remod3d: A high-performance simulator for autonomous, self-reconfigurable robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4281–4287.

[166] A. Brunete, M. Hernando, and E. Gambao, "A simulation environment for bio-inspired heterogeneous chained modular robots," *International Journal of Advanced Robotic Systems*, vol. 11, no. 2, p. 17, 2014.

[167] R. Xu, "Path planning of mobile robot based on multi-sensor information fusion," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 44, 2019.

[168] S. Chennareddy, A. Agrawal, and A. Karuppiah, "Modular self-reconfigurable robotic systems: a survey on hardware architectures," *Journal of Robotics*, vol. 2017, 2017.

[169] P. Thalamy, B. Piranda, and J. Bourgeois, "A survey of autonomous self-reconfiguration methods for robot-based programmable matter," *Robotics and Autonomous Systems*, vol. 120, p. 103242, 2019.

[170] U. Jahn, C. Wolff, and P. Schulz, "Concepts of a modular system architecture for distributed robotic systems," *Computers*, vol. 8, no. 1, p. 25, 2019.

[171] R. J. Alattas, S. Patel, and T. M. Sobh, "Evolutionary modular robotics: Survey and analysis," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 3-4, pp. 815–828, 2019.

[172] C. Dondrup, N. Bellotto, F. Jovan, M. Hanheide *et al.*, "Real-time multisensor people tracking for human-robot spatial interaction," in *Workshop on Machine Learning for Social Robotics at ICRA*. IEEE, 2015, pp. 1–6.

[173] J. M. Santos, T. Krajník, J. P. Fentanes, and T. Duckett, "Lifelong information-driven exploration to complete and refine 4-d spatio-temporal maps," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 684–691, 2016.

[174] S. Erfani, A. Jafari, and A. Hajiahmad, "Comparison of two data fusion methods for localization of wheeled mobile robot in farm conditions," *Artificial Intelligence in Agriculture*, vol. 1, pp. 48–55, 2019.

[175] T. Fäulhammer, R. Ambruş, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, "Autonomous learning of object models on a mobile robot," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 26–33, 2016.

[176] T. Tosun, J. Daudelin, G. Jing, H. Kress-Gazit, M. Campbell, and M. Yim, "Perception-informed autonomous environment augmentation with modular robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6818–6824.

[177] K. Nagla, M. Uddin, and D. Singh, "Multisensor data fusion and integration for mobile robots: A review," *IAES International Journal of Robotics and Automation*, vol. 3, no. 2, p. 131, 2014.

[178] M. T. Nguyen, H. M. La, and K. A. Teague, "Collaborative and compressed mobile sensing for data collection in distributed robotic networks," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1729–1740, 2017.

[179] L. Xiang, J. Luo, and C. Rosenberg, "Compressed data aggregation: Energy-efficient and high-fidelity data collection," *IEEE/ACM transactions on Networking*, vol. 21, no. 6, pp. 1722–1735, 2012.

[180] C. Tang, G. Y. Tian, K. Li, R. Sutthaweekul, and J. Wu, "Smart compressed sensing for online evaluation of cfrp structure integrity," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 12, pp. 9608–9617, 2017.

[181] Y. Liang and Y. Li, "An efficient and robust data compression algorithm in wireless sensor networks," *IEEE Communications Letters*, vol. 18, no. 3, pp. 439–442, 2014.

[182] P. Thalamy, B. Piranda, and J. Bourgeois, "Distributed self-reconfiguration using a deterministic autonomous scaffolding structure," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 140–148.

[183] R. Fitch and R. McAllister, "Hierarchical planning for self-reconfiguring robots using module kinematics," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 477–490.

[184] L. Furno, M. Blanke, R. Galeazzi, and D. J. Christensen, "Self-reconfiguration of modular underwater robots using an energy heuristic," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6277–6284.

[185] M. Singh, V. Agrawal, and G. Bhatti, "Attribute based comparison and selection of modular self-reconfigurable robot using multiple attribute decision making approach," *International Journal of Mechanical and Mechatronics Engineering*, vol. 12, no. 5, pp. 478–484, 2018.

[186] D. Bie, I. Sajid, J. Han, J. Zhao, and Y. Zhu, "Natural growth-inspired distributed self-reconfiguration of ubot robots," *Complexity*, vol. 2019, 2019.

[187] T. Tucci, B. Piranda, and J. Bourgeois, "A distributed self-assembly planning algorithm for modular robots," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 550–558.

[188] C. Parrott, T. J. Dodd, and R. Groß, "Hymod: A 3-dof hybrid mobile and self-reconfigurable modular robot and its extensions," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 401–414.

[189] A. Vergara, Y.-s. Lau, R.-F. Mendoza-Garcia, and J. C. Zagal, "Soft modular robotic cubes: toward replicating morphogenetic movements of the embryo," *PloS one*, vol. 12, no. 1, p. e0169179, 2017.

[190] R. Thakker, A. Kamat, S. Bharambe, S. Chiddarwar, and K. M. Bhurchandi, "Rebis-reconfigurable bipedal snake robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 309–314.

[191] A. Spröwitz, R. Moeckel, M. Vespignani, S. Bonardi, and A. J. Ijspeert, "Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 1016–1033, 2014.

[192] Z.-X. Liu, C.-X. Xie, M. Xie, and J. Mao, "Mobile robot positioning method based on multi-sensor information fusion laser slam," *Cluster Computing*, pp. 1–7, 2018.

[193] J. Yuan, J. Zhang, S. Ding, and X. Dong, "Cooperative localization for disconnected sensor networks and a mobile robot in friendly environments," *Information Fusion*, vol. 37, pp. 22–36, 2017.

[194] J. M. Bahi, A. Makhoul, and M. Medlej, "Frequency filtering approach for data aggregation in periodic sensor networks," in *2012 IEEE Network Operations and Management Symposium*. IEEE, 2012, pp. 570–573.

[195] I. Atoui, A. Ahmad, M. Medlej, A. Makhoul, S. Tawbe, and A. Hijazi, "Tree-based data aggregation approach in wireless sensor network using fitting functions," in *2016 Sixth international conference on digital information processing and communications (ICDIPC)*. IEEE, 2016, pp. 146–150.

[196] S. Madden, "Intel lab data," *http://db.csail.mit.edu/labdata/labdata.html*, 2004.

[197] M. Belbeoch, "Argo project," *http://www.argo.ucsd.edu/index.html*, 2003.

[198] P. Thalamy, B. Piranda, and J. Bourgeois, "Distributed self-reconfiguration using a deterministic autonomous scaffolding structure," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 140–148.

[199] R. Fitch and R. McAllister, "Hierarchical planning for self-reconfiguring robots using module kinematics," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 477–490.

[200] R. Thakker, A. Kamat, S. Bharambe, S. Chiddarwar, and K. M. Bhurchandi, "Rebis-reconfigurable bipedal snake robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 309–314.

[201] R. Xu, "Path planning of mobile robot based on multi-sensor information fusion," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 44, 2019.

[202] U. Jahn, C. Wolff, and P. Schulz, "Concepts of a modular system architecture for distributed robotic systems," *Computers*, vol. 8, no. 1, p. 25, 2019.

[203] K. Nagla, M. Uddin, and D. Singh, "Multisensor data fusion and integration for mobile robots: A review," *IAES International Journal of Robotics and Automation*, vol. 3, no. 2, p. 131, 2014.

[204] M. Singh, V. Agrawal, and G. Bhatti, "Attribute based comparison and selection of modular self-reconfigurable robot using multiple attribute decision making approach," *International Journal of Mechanical and Mechatronics Engineering*, vol. 12, no. 5, pp. 478–484, 2018.

[205] R. J. Alattas, S. Patel, and T. M. Sobh, "Evolutionary modular robotics: Survey and analysis," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 3-4, pp. 815–828, 2019.

[206] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–41, 2019.

[207] N. Tan, A. A. Hayat, M. R. Elara, and K. L. Wood, "A framework for taxonomy and evaluation of self-reconfigurable robotic systems," *IEEE Access*, vol. 8, pp. 13 969–13 986, 2020.

[208] P. Thalamy, B. Piranda, and J. Bourgeois, "A survey of autonomous self-reconfiguration methods for robot-based programmable matter," *Robotics and Autonomous Systems*, vol. 120, p. 103242, 2019.

[209] H. Khodr, M. Mutlu, S. Hauser, A. Bernardino, and A. Ijspeert, "An optimal planning framework to deploy self-reconfigurable modular robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4278–4285, 2019.

[210] A. Almethen, O. Michail, and I. Potapov, "Pushing lines helps: Efficient universal centralised transformations for programmable matter," *Theoretical Computer Science*, 2020.

[211] G. Yasuda, "Design and implementation of distributed autonomous coordinators for cooperative multi-robot systems," in *Robotic Systems: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2020, pp. 324–339.

[212] V. T. Le, T. D. Ngo *et al.*, "Virtual pheromone based network flow control for modular robotic systems," *Electronics*, vol. 9, no. 3, p. 481, 2020.

[213] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots," *IEEE Access*, vol. 7, pp. 94 642–94 657, 2019.

[214] L. Furno, M. Blanke, R. Galeazzi, and D. J. Christensen, "Self-reconfiguration of modular underwater robots using an energy heuristic," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6277–6284.

[215] C. Dondrup, N. Bellotto, F. Jovan, M. Hanheide *et al.*, "Real-time multisensor people tracking for human-robot spatial interaction," in *Workshop on Machine Learning for Social Robotics at ICRA*. IEEE, 2015, pp. 1–6.

[216] H. Kawano, "Distributed tunneling reconfiguration of cubic modular robots without meta-module's disassembling in severe space requirement," *Robotics and Autonomous Systems*, vol. 124, p. 103369, 2020.

[217] P. Thalamy, B. Piranda, F. Lassabe, and J. Bourgeois, "Deterministic scaffold assembly by self-reconfiguring micro-robotic swarms," *Swarm and Evolutionary Computation*, p. 100722, 2020.

[218] D. Bie, M. A. Gutiérrez-Naranjo, J. Zhao, and Y. Zhu, "A membrane computing framework for self-reconfigurable robots," *Natural Computing*, vol. 18, no. 3, pp. 635–646, 2019.

[219] D. Bie, I. Sajid, J. Han, J. Zhao, and Y. Zhu, "Natural growth-inspired distributed self-reconfiguration of ubot robots," *Complexity*, vol. 2019, 2019.

[220] D. Bie, Y. Zhu, X. Wang, Y. Zhang, and J. Zhao, "L-systems driven self-reconfiguration of modular robots," *International Journal of Advanced Robotic Systems*, vol. 13, no. 5, p. 1729881416669349, 2016.

[221] A. Majed, H. Harb, A. Nasser, B. Clement, and O. Reynet, "Sensing-based self-reconfigurable decision-making mechanism for autonomous modular robotic system," *IEEE Sensors Journal*, vol. 20, no. 13, pp. 7097–7106, 2020.

[222] M. T. Nguyen, H. M. La, and K. A. Teague, "Collaborative and compressed mobile sensing for data collection in distributed robotic networks," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1729–1740, 2017.

[223] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[224] A. Sproewitz, A. Billard, P. Dillenbourg, and A. J. Ijspeert, "Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 4259–4264.

[225] P. Thalamy, B. Piranda, and J. Bourgeois, "Distributed self-reconfiguration using a deterministic autonomous scaffolding structure," Ph.D. dissertation, UBFC, 2019.

[226] R. Thakker, A. Kamat, S. Bharambe, S. Chiddarwar, and K. Bhurchandi, "Rebis-reconfigurable bipedal snake robot. 2014 ieee," in *RSJ International Conference on Intelligent Robots and Systems, Chicago-USA*, pp. 309–314.

[227] P. Thalamy, B. Piranda, and J. Bourgeois, "A survey of autonomous self-reconfiguration methods for robot-based programmable matter," *Robotics and Autonomous Systems*, vol. 120, p. 103242, 2019.

[228] S. Vassilvitskii, M. Yim, and J. Suh, "A complete, local and parallel reconfiguration algorithm for cube style modular robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1. IEEE, 2002, pp. 117–122.

[229] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, and S. Kokaji, "Hardware design of modular robotic system," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 2210–2217.

[230] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME transactions on mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[231] A. Sproewitz, P. Laprade, S. Bonardi, M. Mayer, R. Moeckel, P.-A. Mudry, and A. J. Ijspeert, "Roombots—towards decentralized reconfiguration with self-reconfiguring modular robotic metamodules," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1126–1132.

[232] E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, and S. Kokaji, "A distributed method for reconfiguration of a three-dimensional homogeneous structure," *Advanced Robotics*, vol. 13, no. 4, pp. 363–379, 1998.

[233] K. D. Kotay and D. L. Rus, "Algorithms for self-reconfiguring molecule motion planning," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 2184–2193.

[234] C. Unsal and P. K. Khosla, "A multi-layered planner for self-reconfiguration of a uniform group of i-cube modules," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 598–605.

[235] C. Ünsal, H. Kiliççöte, and P. K. Khosla, "A modular self-reconfigurable bipartite robotic system: Implementation and motion planning," *Autonomous Robots*, vol. 10, no. 1, pp. 23–40, 2001.

[236] 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 *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1338–1345.

[237] M. Yim, Y. Zhang, J. Lamping, and E. Mao, "Distributed control for 3d metamorphosis," *Autonomous Robots*, vol. 10, no. 1, pp. 41–56, 2001.

[238] R. Fitch, Z. Butler, and D. Rus, "Reconfiguration planning for heterogeneous self-reconfiguring robots," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2460–2467.

[239] ——, "In-place distributed heterogeneous reconfiguration planning," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 159–168.

[240] P. Thalamy, B. Piranda, and J. Bourgeois, "A survey of autonomous self-reconfiguration methods for robot-based programmable matter," *Robotics and Autonomous Systems*, vol. 120, p. 103242, 2019.

[241] R. Fitch and R. McAllister, "Hierarchical planning for self-reconfiguring robots using module kinematics," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 477–490.

[242] Z. Butler and D. Rus, "Distributed planning and control for modular robots with unit-compressible modules," *The International Journal of Robotics Research*, vol. 22, no. 9, pp. 699–715, 2003.

[243] M. Park, S. Chitta, A. Teichman, and M. Yim, "Automatic configuration recognition methods in modular robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 403–421, 2008.

[244] B. McKay, "Nauty user's guide (v2. 4)," *Computer Science Dept., Australian Nat. Univ*, 2007.

[245] M. Grohe, K.-i. Kawarabayashi, D. Marx, and P. Wollan, "Finding topological subgraphs is fixed-parameter tractable," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011, pp. 479–488.

[246] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME transactions on mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[247] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Self-reconfigurable modular robot m-tran: distributed control and communication," in *Proceedings of the 1st international conference on Robot communication and coordination*, 2007, pp. 1–7.