BARC0141: Built Environment Dissertation

"Structurally-driven Self-Reconfiguration: Towards Structurally-aware Assemblies of Modular Robots"

by

Sofia Feist

19162387

September 2020

Word Count: 10 685

Dissertation submitted in part fulfilment of the Degree of Master MSc Architectural Computation

Bartlett School of Architecture
University College London

MSc/MRes Architectural Computation Dissertation Submission Form

A signed and dated copy of the following MUST be inserted after the title page of your dissertation. If you fail to submit this statement duly signed and dated, your submission will not be accepted for marking.

1. DECLARATION OF AUTHORSHIP

I confirm that I have read and understood the guidelines on plagiarism, that I understand the meaning of plagiarism and that I may be penalised for submitting work that has been plagiarised.

I certify that the work submitted is my own and that it has been also submitted electronically and that this can be checked detection service, Turnitin®.

I declare that all material presented in the accompanying work is entirely my own work except where explicitly and individually indicated and that all sources used in its preparation and all quotations are clearly cited.

Should this statement prove to be untrue, I recognise the right of the Board of Examiners to recommend what action should be taken in line with UCL's regulations.

2. COPYRIGHT

Please write your initials in the box if you DO NOT want this report to be made available publicly either electronically or in hard copy.

☐

Name:	Sofia Feist
Signed:	Sofia Feist
Date:	14 September 2020

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my supervisor, Valentina Soana, for all the patience and guidance she provided me throughout the writing of this thesis.

To my family for their unwavering support and helping me focus on my thesis throughout these difficult times.

Finally, to Carmo Cardoso and Teresa Neto for their advice and encouragements to do my best.

ABSTRACT

Self-Reconfigurable Robots have shown great versatility and promise for building dynamic and self-adapting structures of modular robots. Unfortunately, the structural requirements for building structurally-sound robotic structures of modular robots at the architectural scale, where structural performance and stability of the assembly are crucial for success, have yet to be properly addressed.

This thesis addresses these requirements and proposes a structurally-driven control strategy for a self-reconfigurable robotic system based on the structural analysis and performance of not only the final target configuration of a robotic assembly but also of the intermediate transitional configurations achieved during self-reconfiguration. To formulate a structurally feasible target shape, a topology optimization is used to evolve the target shape based specific boundary and loading conditions and to maximize structural stiffness. Thereafter, the control strategy drives the modules' decision-making process using three fitness criteria for action selection: modules' convergence towards the given target configuration, stability of the overall assembly, and the structural performance of the assembly. While the proposed control strategy succeeds in filtering out unstable and structurally unsafe configurations, corrective measures fail in completely dealing with a declining structural performance. Nevertheless, this thesis exposes some of the difficulties in using local decision-making to solve global structural issues and extends the state-of-art on structurally-aware self-reconfigurable robots.

Keywords: Modular Robotics, Self-Reconfigurable Robots, Structurally-aware robots, Robotic-Assembled Structures

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DoF	- Degree of Freedom
FEA	- Finite Element Analysis
SRR	- Self-Reconfigurable Robot
UDP	- User Datagram Protocol

Chapter 1

Introduction

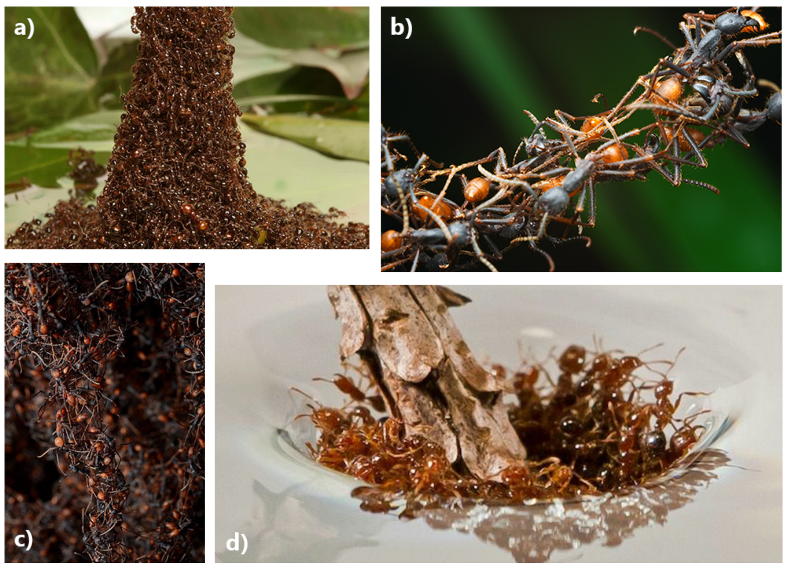

Figure 1.1: ***Ant-assembled Structures:*** *a) a fire ant tower, b) an army ant bridge, c) an army ant bivouac, and d) a fire ant raft*

Social insects are some of nature most skilled builders. For nesting, ants can build large intricate structures of tunnels with multiple entrances and chambers while termites can build towering mounds that can reach up to six or seven meters high, i.e. 600 times the size of the individual (Theraulaz et al., 1998).

Besides nesting architecture, some species are also capable of building temporary structures to deal with obstacles and dynamically respond to new threats using the most readily available building material: their own bodies (see Figure 1.1). Army ants can dynamically assemble bridges and towers with their bodies allowing them to cross gaps, reach higher places and optimize traffic flows for foraging and food transportation. In a flood, fire ants can assemble into rafts of up to 100,000 individuals (Mlot et al., 2012), forming a water-repellent and highly buoyant lattice that allows them to float and maximize their chances of survival. Both army ants and fire ants can also build bivouacs, i.e. temporary nests made with their bodies to protect the queen and their young. These structures are dynamically assembled, responsive to changing conditions and naturally disassembled when no longer needed.

The complexity of these structures has long since fascinated researchers and scientists, who marvel at social insects' ability to collectively coordinate, self-organize and self-assemble without global communication. In modular robotics, this model of a decentralized system comprised of several autonomous units capable of dynamic self-adaptation and collective behaviour has inspired engineers to design robotic systems with similar characteristics. Self-Reconfigurable Robots is one of such systems.

Self-Reconfigurable Robots (SRRs) are robots built from autonomous modules capable of dynamically changing their own shape by self-organization of its parts. Like ants in a colony, the individual modules are able to coordinate with their peers and adapt to collaboratively perform a range of different tasks. SRRs are built for versatility and have a wide range of possible applications. The one this research focuses on is their ability to dynamically self-reconfigure into various structures in times of need such as a temporary bridge or a support structure to hold a collapsing structure, using their constituent modules as building blocks. In the absence of prior preparation or adequate building materials, using the readily available modules of a scout SRR to build self-adapting structures might be a major advantage.

1.1. OBJECTIVES AND CONTRIBUTION

In order to be useful at the architectural scale, robotic structures of modular robots need to be strong and adaptive enough to be able to withstand the internal and external loads resulting from structural formation and functional use. They also need to be stable throughout all stages of self-reconfiguration so that they can stand on their own without falling. Unfortunately, research on SRRs has been mostly focused on the development of cheap and versatile hardware solutions as well as software strategies for scalable and robust self-reconfiguration; the structural requirements to build structurally-sound robotic assemblies has yet to be properly addressed. Moreover, reconfiguration planning strategies rarely address structural issues and stability of robotic assemblies during self-reconfiguration thus making it difficult to physically build assemblies of more than a few modules.

To address the above mentioned problems, this thesis proposes a structurally-driven control strategy for a self-reconfigurable robotic system that can autonomously build structurally-aware structures. In doing so, it seeks explore how the structural requirements of building can impact the motion planning of modular robots and promote the construction of stable and structurally-sound robotic structures at the architectural scale. This strategy is based on the structural analysis and performance of not only the final target configuration of a robotic assembly but also of the intermediate transitional configurations achieved during self-reconfiguration.

This process starts with the formulation of a target configuration using a topology optimization to obtain a structurally optimized shape based on specific local conditions and loads. The goal is to find the most effective load-carrying shape for the given conditions that minimizes the strain energy in the system and maximizes global stiffness.

Once a structurally feasible target shape is found, the control strategy then drives module actions based on distributed control logic and local decision-making to reach the target configuration. The decision-making process is based on the analysis of all reachable configurations of a module according to three fitness criteria: (1) convergence towards the given target configuration, (2) stability of the overall assembly, and (3) the structural performance of the assembly. Actions are chosen according the calculated values of the three fitnesses. By driving self-reconfiguration using real-time structural feedback, unstable and structurally unsafe configurations can be rejected and corrective measures applied to compensate declining structural performances.

This research relies on existing analysis and simulation software for structural analysis and stability simulations. Hardware experiments are not considered in the scope of this thesis.

1.2. STRUCTURE

This thesis is organized into 6 chapters, as follows:

1. Chapter 1 introduces the problem, motivation, objectives and contributions of the thesis;
2. Chapter 2 explains the fundamentals of SRRs, giving some examples of existing robotic systems, as well as how self-reconfiguration works, introducing some of the strategies used to control it;
3. Chapter 3 discusses the structural analysis and assembly planning discrete robotic assemblies and addresses some of the related work for structurally-driven self-reconfigurable robotic assemblies;
4. Chapter 4 explains the research methodology and implementation strategies used to develop the control system and drive the self-reconfiguration process based on structural feedback;
5. Chapter 5 presents the results of the control strategy and discusses some of the problems and potential fixes;
6. Chapter 6 presents the conclusions and future work of the research.

Chapter 2

Self-Reconfigurable Robotic Systems

SRRs are robots made of autonomous modules capable of changing their own shape, allowing them to dynamically adapt to different tasks and environments. This ability makes them ideal for situations where the robot might encounter unexpected obstacles, have to work in unstructured environments or perform a variety of tasks that are not necessarily known a priori. For example, in search and rescue situations after a disaster (Yim et al., 2000a), they can reconfigure to squeeze through tight holes and climb over tall obstacles and reach places that humans cannot. They can also adapt their locomotion gaits to different types of terrains, e.g. forming a rolling track gait for flat terrains and high speed and a legged configuration to move through uneven and debris-filled terrains.

Other possible applications include management of large facilities (Baca et al., 2015), underwater inspection and monitoring (Christensen et al., 2015), planetary exploration (Yim et al., 2003), construction of space structures (Shen et al., 2003; O'Callaghan, 2019), self-assembled structures (Saldana et al., 2017), programmable matter, physical rendering and even synthetic realities (Goldstein et al., 2005; Bourgeois et al., 2016).

Yim et al. (2007a) define the main motivators of SRRs as:

VERSATILITY Versatility in SRRs pertains to their ability to perform many different tasks in many different environment. Modules can be connected in various ways and the robot can reconfigure between a variety of different shapes, allowing them to adapt to their tasks and the environment.

ROBUSTNESS Robustness derives from the robot's ability to handle and adapt to hardware and software failures. In conventional robots, component malfunctions can cause the entire robot to fail in its task but in SRRs, modules are interchangeable and capable of self-repair. As a result, the performance of SRRs does not fail catastrophically but declines gracefully with the number of malfunctioning modules.

LOW-COST Thanks to their modular nature, SRRs can be mass-produced which can reduce the costs of production when economies of scale come into play. Low-cost can also be achieved through simplicity in the design of the individual modules as well as through the reduced computational power needed due to the distributed nature of the modular robots. Finally, versatility and generality allows SRRs to deal with a variety of different tasks, saving costs through reuse.

In the following sections, different Self-Reconfigurable Robotic Systems will be presented, along with some of the methods and algorithmic strategies used to control them.

2.1. CLASSIFICATION AND EXISTING SELF-RECONFIGURABLE ROBOTIC SYSTEMS

Self-Reconfigurable Robotic systems can generally be categorized into 3 types: chain-based, lattice-based or hybrids of the two. Chain-based systems are systems where modules form linearly connected (chain-like) or branching (tree-like) bodies and use module coordination to move with animal-like fluidity. Lattice-based systems are organized into a 2D or 3D lattice of cells like atoms in a crystal, which modules can navigate through using a *cluster-flow* [1] style of locomotion.

Both chain-based and lattice-based systems offer different advantages regarding locomotion and self-reconfiguration. Chain-based systems are better for locomotion and for navigating through low obstacles and tunnels, while lattice-based systems are best suited for self-reconfiguration, building a wider variety structures and climbing over tall obstacles. Hybrid systems can form both chain and lattice structures and combine the benefits of the two into a single system: the fluidity of locomotion of chain-based systems and the versatility of self-reconfiguration of lattice-based systems.

Using this classification, the following sections present several examples of existing self-reconfigurable robotic systems. These examples do not form an exhaustive survey of all robotic systems developed until today but provide a comprehensive and sufficiently diverse overview of the systems developed. For a more extensive survey of SRRs, see Seo et al. (2019), Chennareddy et al. (2017), or Jinguo et al. (2016). For a review on current robotic trends in SRRs, see Brunete et al. (2017).

2.1.1. Chain-based Systems

The roots of Self-Reconfigurable Robots can be traced back to 1988 with the introduction of the Cellular Robot (CEBOT), the first dynamically reconfigurable robot (Fukuda and Nakagawa, 1988; Fukuda and Kawauchi, 1990). The CEBOT proposed the idea of modular robots as cells in a living organism, capable of self-organizing and adapting to their tasks and environment. While it could not yet autonomously self-reconfigure on its own, it paved the conceptual groundwork for Self-Reconfigurable Robots as we know today.

The PolyPod was the first chain-based reconfigurable robot, designed and built by Yim (1993, 1994a) at Stanford University. It was composed of two types of modules: a rigid cubic Node module for branching and a Segment module for actuation (see a) in Figure 2.1). The PolyPod showed the versatility of modular design for different types of locomotion gaits, such as the Caterpillar, the Snake, the Spider and the Rolling-Track locomotion gaits (Yim, 1994b), but could also not yet change shape by itself. Years later, its successor, the PolyBot Yim et al. (2000b), improved on that by introducing autonomous self-reconfiguration via electromechanical connectors as well as increasing the load carrying capacity of the robot for object manipulation.

Unlike the CEBOT and PolyPod, CONRO (CONfigurable RObot) (Castano et al., 2000) was designed on the idea of homogeneity, autonomy, and self-sufficiency of the modules. CONRO is composed of homogeneous modules consisting of three segments connected in a chain: a passive connector, a body, and an active connector. Each module is autonomous and self-sufficient, containing all components and functionalities to continue working when detached from the main robot. This allows modules to work independently, simplifies the self-reconfiguration process since all modules work in the exact same way, adds redundancy to the system, allowing faulty modules to be easily replaced, and increasing the robot's robustness. CONRO showed versatility in performing a wide range of locomotion gaits and self-reconfiguration capabilities. It also introduced a Hormone-inspired algorithm for synchronization and coordination between independent module (Shen et al., 2000, 2002),

[1] Cluster flow is a type of locomotion where modules move one by one from the back of the robot to the front (Stoy et al., 2010).

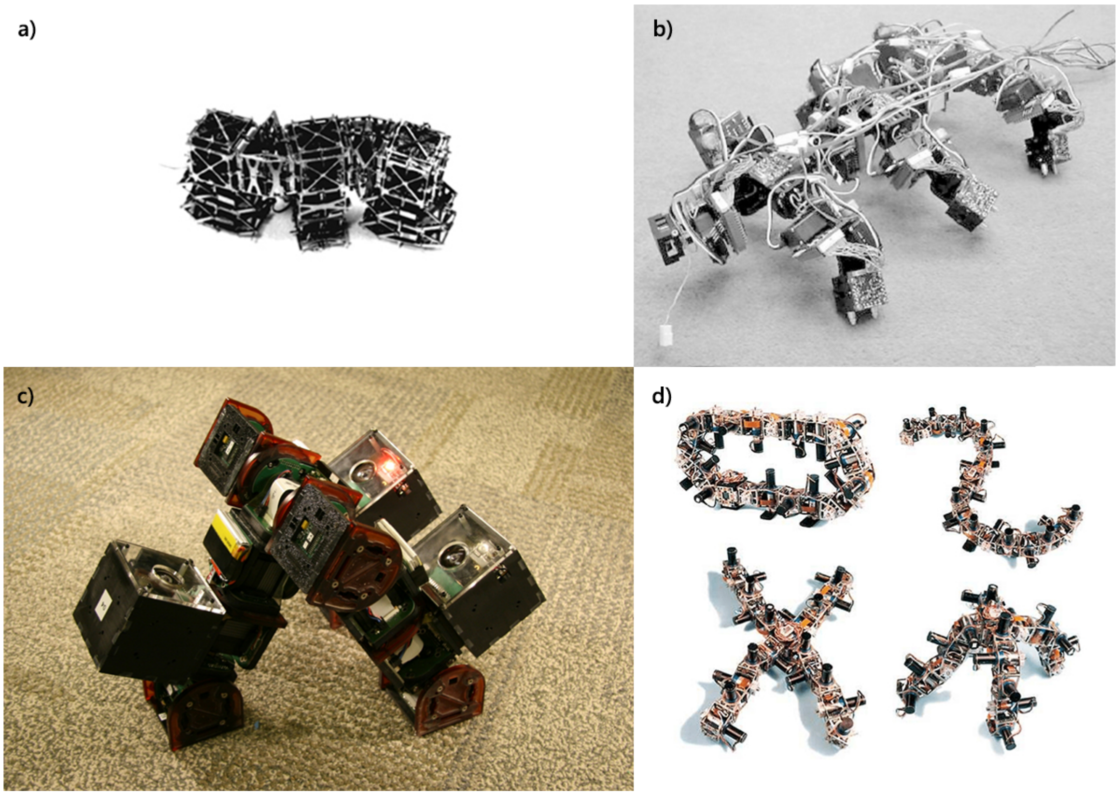

Figure 2.1: ***Chain-based Robots:*** *a) The PolyPod using a Caterpillar locomotion gait (Source: Yim (1994a)); b) CONRO in a hexapod configuration (Source: Castano et al. (2002)); c) The CKBot with 3 clusters of 5 modules each reassembling after a disassembly (Source: Yim et al. (2007b)); d) The second generation of the PolyBot in different locomotion configurations (Source: Yim et al. (2002)).*

which became an important contribution for the distributed control and scalability of chain-based self-reconfigurable robots.

Finally, the Connector Kinetic roBot (CKBot), also created by Yim et al., was developed with the focus on self-repair and self-reassembly after an explosive disassembly of the robot due to an unexpected external force, e.g. an impact (Yim et al., 2007b). The CKBot is also an heterogeneous robot with different types of modules: two different-shaped actuation modules with one rotational Degree of Freedom (DoF) each, and several specialized modules which give the CKBot additional capabilities such as modules with infra-red proximity sensors, a gripper module, or a camera module. By grouping together clusters of different modules (see c) in Figure 2.1), the CKBot showed how modules can work collaboratively to self-reassemble after an explosive disassembly (see video at UPenn (2009)).

2.1.2. Lattice-based Systems

Developed a year after the PolyPod, Fracta (Murata et al., 1994) and the Metamorphic Robot Chirikjian (1994) were the first SRRs to successfully implement autonomous self-reconfiguration, without manual human intervention, albeit still only in two dimensions. Fracta achieved this using only electromagnetic forces for actuation while the Metamorphic achieved this using mechanical actuation.

The first SRR to achieve the same in three dimensions was the Molecule (Rus and Kotay, 1998). One Molecule module is composed of two 'atoms' connected by a right-angle rigid bond with 2 DoFs. With this design, the Molecule showed different types of motion capabilities, including linear walking, concave and convex transitions of individual modules (Kotay et al.,

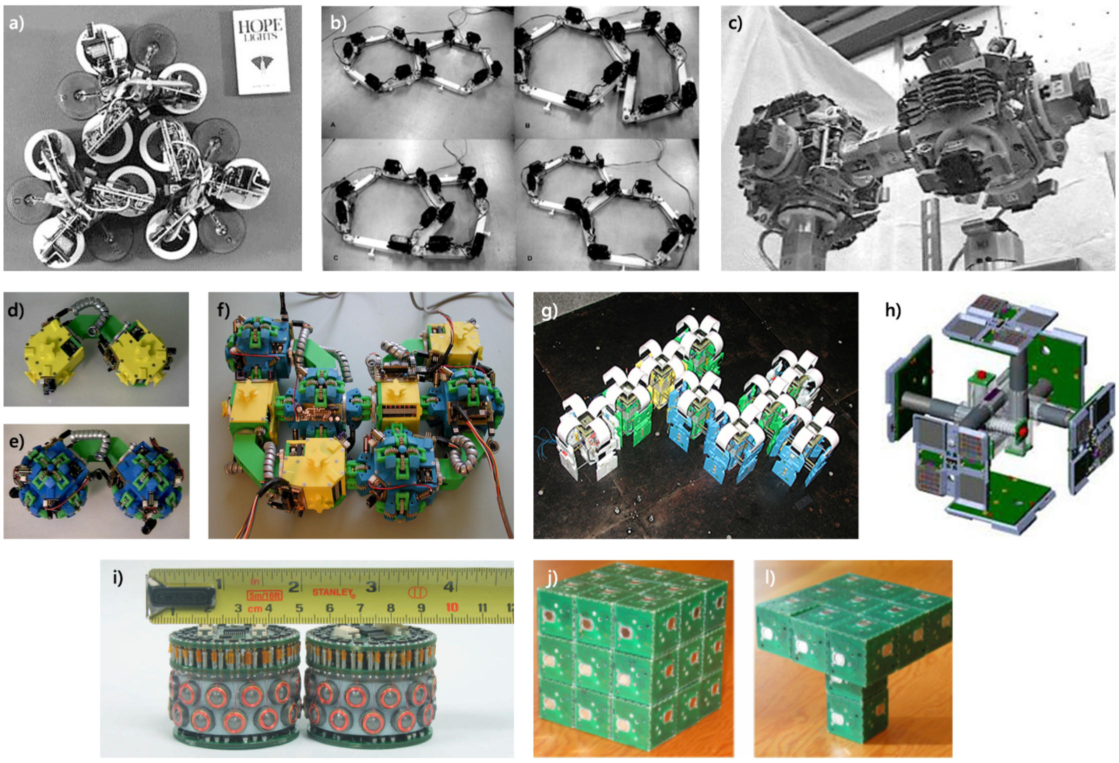

Figure 2.2: ***Lattice-based Robots:*** *a) Three connected Fracta modules (Source: Murata et al. (1994)); b) Locomomotion of one metamorphic module around another (Source: Pamecha et al. (1996)); c) Photo of the two connected 3D-Unit modules (Source: Murata et al. (1998)); d)-f) Third generation of the Molecule robot with female/passive (yellow) and male/active (blue) modules (Source: Rus Robotics Laboratory (2000b)); g) Nine Crystalline modules (Source: Rus Robotics Laboratory (2000a); h) A Telecube module (Source: Suh et al. (2002)); i) Two mini Catom Modules (Source: Goldstein et al. (2005)); j)-l) A T structure obtained by self-disassembly of the Miche Robot (Source: Gilpin et al. (2008)).*

1998), stair climbing (Rus and Kotay, 1998) and two types of lattice-based locomotion: a tumbling algorithm for cluster-flow locomotion and a dynamic rolling gait achieved by dynamically moving the center of mass of the robot Kotay and Rus (2005). It also introduced the concept of *Scaffolding* (Kotay and Rus, 2000), a strategy to simplify self-reconfiguration planning by introducing a regular structure with tunnels for modules to navigate through.

Like the Molecule, the 3D-Unit (Murata et al., 1998) was another important contribution for SRRs in three dimensions. However, unlike the Molecule, the 3D-Unit is based on compactness and spatial symmetry and modules have no locomotion autonomy. Instead, modules use a pair-wise strategy for self-reconfiguration: since modules have no locomotion capabilities on their own, they rely on neighbouring modules to act as pivot and carry them around the structure of modules (Kurokawa et al., 1998). The 3D-Unit also introduced a distributed self-reconfiguration algorithm for lattice-based systems based on simulated annealing to achieve convergence (Yoshida et al., 1998), which became one of the first distributed self-reconfiguration algorithms for SRRs in 3D.

The Crystalline Robot and the Telecubes are two other lattice-based SRR developed by Rus and Vona (1999) and Suh et al. (2002). Together, they introduced a new self-reconfiguration approach based on module contraction and expansion, the Crystalline Robot in 2 dimensions and the Telecubes in 3 dimensions: by contracting, two modules can be squeezed into one lattice cell, which allows a supporting module to be dragged one cell and support the expansion of the contracted modules to their new cells, moving the structure of modules by one cell.

Both the Catoms (Goldstein et al., 2005) and Miche (Gilpin et al., 2008) explored the use of SRRs for constructing programmable matter. The Catoms were 44mm-diameter modular robots capable of 2 dimensional self-assembly using electrostatic forces for locomotion. Miche explored 3 dimensional self-reconfiguration through self-disassembly, i.e, by disconnecting unwanted modules from the structure, and letting them fall with gravity.

2.1.3. Hybrid Systems

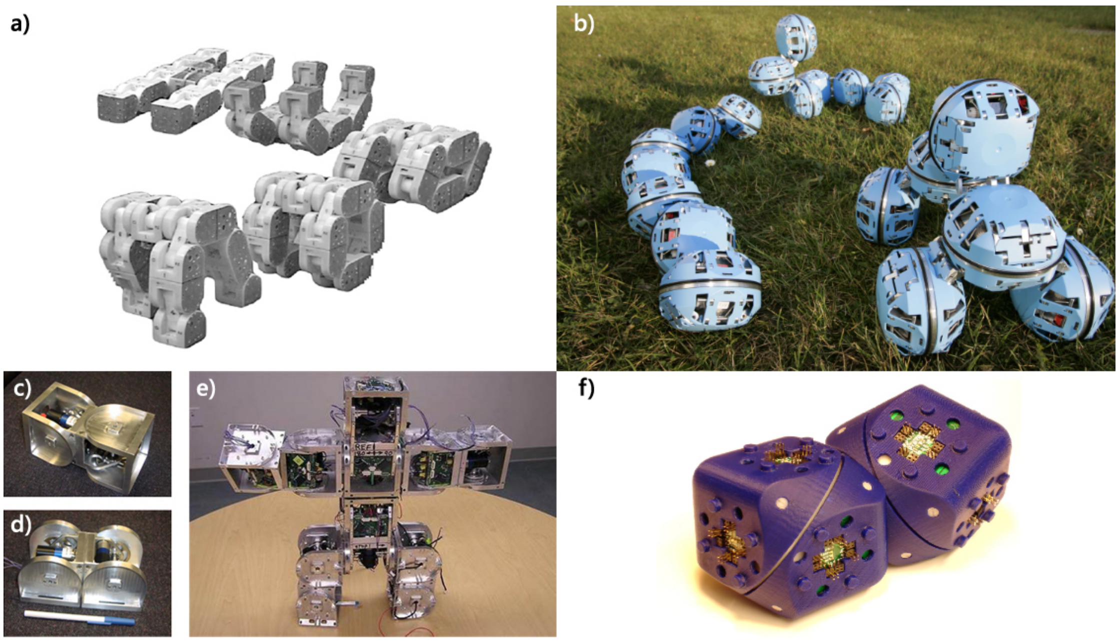

Figure 2.3: ***Hybrid Robots:*** *a) The M-TRAN reconfiguring between different locomotion gaits (Source: AIST); b) ATRON in three different configurations: from left to right, snake, cluster-walk, car (Source: Østergaard et al. (2006)); c) A SuperBot C-module; d) A SuperBot M-module; e) Six Superbot modules in a humanoid configuration (Source: Salemi et al. (2006)); f) Two Molecubes.*

In 1999, the M-TRAN (Modular TRANsformer) introduced the idea of a hybrid SRR, capable of both the 3D self-reconfiguration capabilities of lattice-based systems and the locomotion capabilities of chain-based systems Murata et al. (2000). One M-TRAN module consists of two linked semi-cylindrical cubes with one DoF each, allowing them to rotate from -90 to +90 degrees around its axis. M-TRAN showed both lattice-based cluster-flow locomotion and chain-based locomotion gaits. Regarding the former, M-TRAN used meta-modules (Yoshida et al., 2002) and regular structures of modules (Kurokawa et al., 2005) to simplify self-reconfiguration planning and facilitate cluster-flow locomotion. Regarding the latter, it introduces a method for the automatic generation of locomotion patterns based on a generalized Central Pattern Generator network and genetic algorithms to optimize network parameters for a more dynamic and adaptive locomotion control of modular robots (Kamimura et al., 2003).

ATRON (Jorgensen et al., 2004) was the second hybrid SRR. One ATRON module has a spherical design with only one rotational DoF around the module's central axis. The simplicity of the ATRON's modular design showed that self-reconfiguration was possible with minimal DoFs. It did so by arranging modules' rotational axes perpendicular to each other (see b) in Figure 2.3) and using meta-modules to overcome the limitations of the individual modules (Christensen et al., 2004; Christensen and Stoy, 2006). Similarly, the Molecubes (Zykov et al., 2007) also explored self-reconfiguration with only one DoF, although they did so to physically demonstrate kinematic self-reproduction.

Inspired by other SRRs such as the M-TRAN and CONRO, the SuperBot (Shen et al., 2006b) is a hybrid SRR designed for NASA's space exploration programs and support life on other planets. Similar to the M-TRAN module, a SuperBot module consists of two semi-cylindrical cubes bonded together, with an additional DoF that allows the bond itself to rotate in both directions. By rotating the middle bond by 90º, the M-TRAN-like module (called "M-module", see d) in Figure 2.3) becomes similar to a CONRO module (called "C-module", see c) in Figure 2.3), capable of performing pitching and yawing motion. All electronics and mechanical components are internally protected from possible dust, moisture, and physical impact. Overall, SuperBot showed capable of self-reconfiguration and manipulation tasks as well as multiple modes of locomotion, including on uneven and challenging terrains such as climbing a sand dune (Shen et al., 2006a, 2008).

2.2. SELF-RECONFIGURATION

The main advantage of SRRs is that they can self-reconfigure and, thus, adapt to different tasks, environments and situations. However, self-reconfiguration is a complex problem with no simple or general way to solve it (Christensen et al., 2004).

Self-reconfiguration allows a robot to change shape which involves a sequence of module movements until a target shape, designed to deal with a specific task, is reached. There are several reasons that make self-reconfiguration a complex and challenging process. The first and most obvious is that, in SRRs, instead of controlling a single robot, we need to control and coordinate several autonomous modules, sometimes including heterogeneous modules with different roles and control strategies.

Usually, modular robots have a minimalistic hardware design and functionality to allow cheap production, which consequently puts heavy constraints on their ability to move and interact with the environment. As a result, a very simple move might require a series of complicated intermediate steps to work around the motion limitations of individual modules. Usually, strong coordination and synchronization between modules is required to accomplish any given task. This coordination is also needed to avoid collision between multiple modules potentially moving at the same time.

Secondly, it is important that modules remain connected during self-reconfiguration to maintain electronic communications between modules but also so that no module becomes stranded or left behind and unsupported modules do not fall off the robot. Fallen modules might break upon impact and stranded modules might be unable to autonomously return to the main body. This problem might not be trivial to detect and solve (Stoy et al., 2010).

Thirdly, if a self-reconfiguration sequence is not carefully planned, modules might get stuck in hollows, solid subconfigurations, local minima or leave unfillable holes in the robot. Hollows refers to holes in the goal configuration inside which modules can get stuck, leaving them unable to exit and complete the self-reconfiguration process. Similarly, solid subconfigurations refers to modules getting stuck on the outside of the goal configuration, unable to fill holes on the inside. Local minima refers to when modules following the most direct path to the goal get stuck in configurations from which they cannot escape, due to either problems in the self-reconfiguration sequence or motion constraints. Finally, motion constraints might also make modules unable to reach certain positions, leaving holes in the final goal shape, impossible to be filled (Tucci et al., 2018). These problems are illustrated in Figure 2.4.

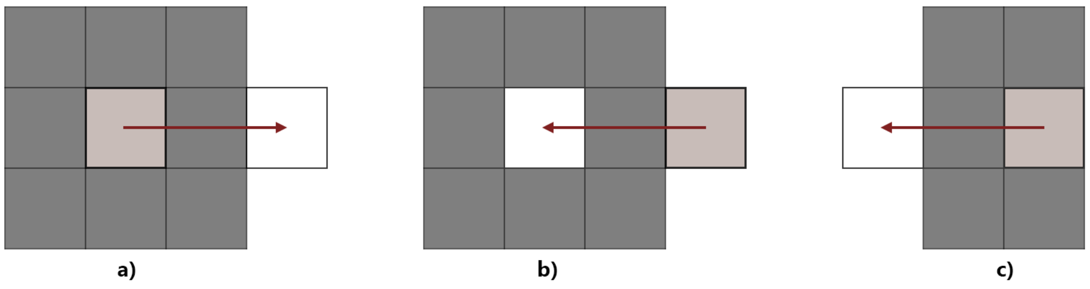

Figure 2.4: *Problematic subconfigurations where modules might get stuck: a) Hollow, b) Solid, c) Local Minima*

These issues stress the need for strong control and careful planning of SRRs. Thus, a variety of methods have been developed to control and plan the self-reconfiguration process. The following section will briefly discuss some of these methods. For a more extensive review of the methods and algorithms used to control SRR, see Ahmadzadeh and Masehian (2015).

2.2.1. Control of Self-Reconfigurable Robots

Control in SRRs generally falls into one of two categories: centralized control and distributed control. In centralized control, a single controller is responsible for managing all of the modules. The state of the whole system is known to the centralized controller, which coordinates all module movements and allocates individual tasks for the global execution of the given task.

Centralized approaches typically involve graph-based and path planning approaches (e.g. Kotay and Rus, 1998; Yoshida et al., 2001). This involves representing the system as a graph of nodes (representing configurations) and connections (representing module actions) and using search algorithms to find the most desirable sequence of steps to reach a final goal configuration. This is not a trivial problem due to the shear amount of possible configurations in the system, a problem which is only exponentially aggravated by the increase in the number of modules to manage. In the Metamorphic Robot, Pamecha et al. (1997) address this problem by introducing heuristics based on simulated annealing to explore the search space and converge towards the goal configuration. For the I-Cubes, Ünsal and Khosla (2001) use multilayered planners to divide the motion planning problem into smaller, tractable sub-problems that can be more easily and quickly solved using heuristic methods.

Unfortunately, centralized approaches are neither scalable nor robust. As the number of modules increases, so does the complexity and number of actions that the centralized controller needs to manage and coordinate which can quickly overload the system. Moreover, if the centralized host fails, so does the entire robot. For these reasons, distributed control is usually preferred in SRRs.

In distributed control, all modules are equipped with their own individual controller. They can only perceive part of the system around them and react locally to their surroundings, resulting in an emergent behaviour. Distributed control is scalable and more robust than centralized control since all modules function autonomously and do not rely on a centralized controller to operate but introduces new challenges regarding global coordination and convergence: if modules can only react locally and autonomously, how can we make the desired global behaviour emerge? Generally, in a distributed approach, all modules need a behaviour strategy that addresses these issues.

One way to tackle this is by giving each individual module a set of local rules and a representation the goal shape. A rule consists of a possible action that a module can take based on a specific local configuration. For example, Butler et al. (2004) show how Cellular-Automata-inspired rule sets, based on purely local configuration information, can be used to generate cluster-flow locomotion.

Following these rules gives modules a clear movement strategy while the representation of the goal shape gives them a goal. If modules know their own global position in the configuration and their goal position, all they need to do is to execute the rules that will bring them closer to their goal (Christensen and Stoy, 2006). Alternatively, if modules do not know their own global and goal position, artificial attraction gradients can be used to guide them towards an empty goal position (Bojinov et al., 2002). In this strategy, one module next to an empty goal cell acts as the source (the *Seed*), emitting a virtual "scent" that is propagated throughout the system, informing and attracting surrounding modules towards that unfilled position.

A problem with these approaches is that, without global path planning, modules can get stuck in local minima or other problematic subconfigurations as seen in the previous section. The introduction of meta-modules (Butler et al., 2004) and scaffolding (Kotay and Rus, 2000) can help alleviate some of these issues and simplify the self-reconfiguration problem. Finally, distributed search can also be used to solve small localized problems. Using this strategy, modules calculate the graph of local reachable positions in the neighbourhood and locate the most desirable path that will lead them closer to the goal (Christensen, 2006).

2.3. SUMMARY

Name	Type	2D/3D	DoF	Homo/Hetero	References
CEBOT	Chain	3D	Various	Heterogeneous	Fukuda and Nakagawa (1988)
PolyPod	Chain	3D	2	Heterogeneous	Yim (1993)
Fracta	Lattice	2D	0	Homogeneous	Murata et al. (1994)
Metamorphic	Lattice	2D	3	Homogeneous	Chirikjian (1994)
Molecule	Lattice	3D	4	Homogeneous	Kotay et al. (1998)
3D-Unit	Lattice	3D	6	Homogeneous	Murata et al. (1998)
Micro-Unit	Lattice	2D	2	Homogeneous	Yoshida et al. (1999)
Crystalline	Lattice	2D	1	Homogeneous	Rus and Vona (1999)
I-Cubes	Lattice	3D	3	Homogeneous	Unsal et al. (1999)
M-TRAN	Hybrid	3D	2	Homogeneous	Murata et al. (2000)
CONRO	Chain	3D	2	Homogeneous	Castano et al. (2000)
PolyBot	Chain	3D	1	Heterogeneous	Yim et al. (2000b)
Telecubes	Lattice	3D	1	Homogeneous	Suh et al. (2002)
Chobie II	Lattice	2.5D	1	Homogeneous	Koseki et al. (2004)
ATRON	Hybrid	3D	1	Homogeneous	Jorgensen et al. (2004)
Catoms	Lattice	2D	0	Homogeneous	Goldstein et al. (2005)
SuperBot	Hybrid	3D	3	Homogeneous	Shen et al. (2006b)
Molecubes	Hybrid	3D	1	Homogeneous	Zykov et al. (2007)
CKBot	Chain	3D	1	Heterogeneous	Yim et al. (2007b)
Miche	Lattice	3D	0	Homogeneous	Gilpin et al. (2008)
ModRED	Chain	3D	4	Homogeneous	Nelson et al. (2010)
Roombots	Hybrid	3D	3	Homogeneous	Spröwitz et al. (2010)
SMORES	Hybrid	3D	4	Homogeneous	Davey et al. (2012)
CoSMO	Hybrid	3D	1	Homogeneous	Liedke et al. (2013)
M-Blocks	Lattice	3D	0	Homogeneous	Romanishin et al. (2013)
Soldercubes	Lattice	3D	1	Heterogeneous	Neubert and Lipson (2016)

Table 2.1: Overview of existing Self-Reconfigurable Robots

In this chapter, we saw different self-reconfigurable robotic systems, as well as some of the methods and strategies used to control them. SRRs can be divided into 3 types. Chain-based system are better for locomotion, lattice-based systems for self-reconfiguration, while hybrids combine the advantages of the two to achieve a system with both good locomotion and self-reconfiguration capabilities.

SRRs can also be either homogeneous or heterogeneous robots, depending on the type(s) of modules that they use. Homogeneous modules simplify self-reconfiguration since all modules are physically and functionally identical and also increases the robustness of the system since faulty modules can be easily replaceable with any other module in the robot. Homogeneity is also advantageous for cost and mass production. On the other hand, heterogeneous modules allows the robot to split components and functionalities among different types of modules as well as handle specialized modules or tools which can increase the capabilities of the robot. This however can come at the cost of system robustness and reconfiguration simplicity since we now have different types of modules to control and keep track of (positions, orientations, and tasks).

Regarding the control of SRRs, the system is centralized if they rely on a single centralized controller or distributed if each module is equipped with their own controller. Centralized control achieves better coordination and convergence but lacks in robustness and scalability, requiring computationally expensive search algorithms to plan module paths. Distributed control is scalable and more robust but does not guarantee emergence of the desired global behaviour.

With SRRs, there is no simple and general way to design a self-reconfigurable robotic system (Yim et al., 2007a) and no SRR yet fully explores the promise of Versatility, Robustness and Low Cost that have becoming the driving motivator to designing SRR (Stoy et al., 2010). The potential for self-reconfigurable robots is vast but the field is still young. While the initial challenges of creating robots capable of self-reconfiguration have been achieved, these systems have only just matured to a degree where application is possible. Now, more research is needed towards putting the acquired knowledge into practice, and making SRRs capable of facing the physical environment.

Chapter 3

Discrete Assemblies of Modular Robots

Discrete structures are structures made of discrete blocks stacked and bonded together to form a whole. In construction, interest in studying discrete structures has been mainly lead by interest to study the performance of masonry structures. More recently, advances in digital fabrication techniques and robotics have enabled the creation of novel and complex discrete assemblies and have re-wakened the interest in structurally analysing these structures in order to materialize them.

In modular robotics, assemblies obtained by self-assembly and self-reconfiguration of modular robots present a unique type of structure since we are not stacking passive building blocks but active modular robots, which move themselves into their own stacking position. Nonetheless, structures built by assemblies of modular robots can still be considered discrete-element structures, since they are made of many discrete modular robots stacked together and connected to their neighbouring modules by the connectors in their joints.

This chapter will discuss how to study the structural performance and assembly planning of discrete robotic assemblies, grounded in research of discrete structures. It will also introduce some relevant related work on structurally-driven self-reconfigured robotic assemblies.

3.1. STRUCTURAL PERFORMANCE OF DISCRETE ROBOTIC ASSEMBLIES

In structural engineering, approaches to studying the structural performance of discrete structures can generally be categorized into one of two categories: material performance and geometry performance. Material performance considers material strength and limits under the influence of stresses in the structure. For example in masonry, these include analysing the compressive and tensile strength of the bricks and mortar and identifying points of material rupture in the structure. In contrast, geometry performance considers the stability conditions of the overall geometry of stacked elements and material failures are not considered (Whiting, 2012).

In modular robotics, material performance considers module and connector strength while geometry performance considers the overall stability of the robotic assembly. The strength of a module depends on its design and materials of the module's shell and internal structure. Østergaard et al. (2006) show how a Finite Element Analysis (FEA) can be used to calculate the modulus of elasticity and the yield stress of an ATRON module. On larger assemblies, this information could be used to identify modules stressed beyond their yield strength.

However, discrete structures are usually weakest at the joints and the same is true for robotic assemblies of modular robots. The stiffness and cohesion of an assembly of modular robots is strongly dependent on connector strength which are typically much weaker than modules' compressive strength. This in turn depends on the type, material and stiffness of the connector. For example, magnetic connectors are only as strong as the strength of the magnetic forces of the magnet while mechanical connectors depend on the design and materials of the connection mechanism. Weak connectors can break easily while loose connectors can cause large module displacements in the structure, as illustrated on the left in Figure 3.1. To address this, ATRON (right in Figure 3.1) uses aluminium connectors with three points of contact for stiffer and stronger connectors.

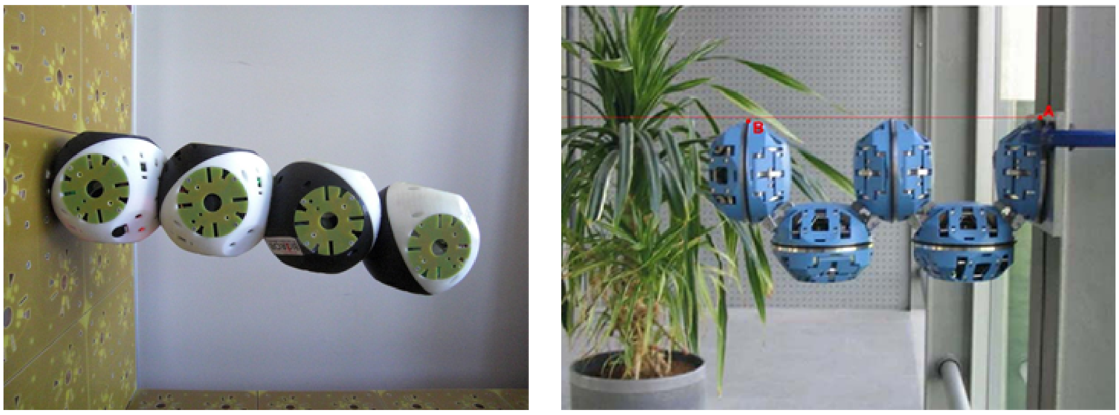

Figure 3.1: ***Module displacement of (Left) two Roombot modules and (Right) five ATRON modules caused by gravity**: (Courtesy of BioRobotics Laboratory, EPFL and Østergaard et al. (2006))*

Finally, the stability of the robotic assembly depends on the geometric configuration of the modules and the conditions of equilibrium. An unstable configuration could make the robotic assembly fall before the self-reconfiguration process is complete or cause tension forces in the structure that could put too much strain the connectors. Thus, it's important to consider both the stability of the final configuration as well as the stability of intermediate configurations during self-reconfiguration.

3.2. ASSEMBLY PLANNING OF DISCRETE ASSEMBLIES

In SRRs, it is not enough to simply optimize the structural performance of the static target structure. During self-reconfiguration, the robotic assembly will transition between a sequence of geometric configurations until the target structure is reached but if this reconfiguration sequence is not structurally-feasible, the assembly will never be able to converge into the desired target configuration. Unfortunately, assembly planning in SRR usually only involves calculating the sequence of steps modules need to take to self-reconfigure from an initial configuration to a goal configuration. The global structural performance of the robotic assembly is not usually considered during the intermediate stages of self-reconfiguration.

In construction, support structures and falsework are usually needed to support the unfinished structure during assembly. Alternatively, assembly-aware techniques have been proposed to reduce the need for these support structures. The assembly-by-disassembly approach is one of them. The idea is that by reversing the process of disassembly (i.e. starting from the complete structure and removing one block at a time), we can obtain a complete structurally feasible assembly sequence (Kao et al., 2017).

Another option is to use counterbalancing. This can done by adding material in the opposite direction of the potential instability, thus using the weight distribution of the structure to provide stability (Melenbrink et al., 2017).

These methods can be repurposed for the assembly planning of structurally-aware robotic assemblies.

3.3. RELATED WORK ON STRUCTURALLY-DRIVEN ROBOTIC-ASSEMBLIES

As previously mentioned, structural performance and stability conditions of robotic assemblies are not usually considered during design and self-reconfiguration of modular robots. This section discusses some of the exceptions to that.

Many works simulate the robot's physical behaviour under the action of gravity or in different types of environment, but not a lot of them actively address stability as a driving force of motion selection and self-reconfiguration. In Ünsal et al. (2000), the I-Cubes' centralized motion planner actively addresses the issue of stability by executing a stability analysis of each action to ensure that a motion step will not cause the robot to overturn. This is achieved by comparing the location of the robot's center of gravity on the plane normal to the direction of gravity with the footprint of the robot, which will be stationary during a projected move. If the center of gravity falls outside of the robot's footprint, the configuration is considered unstable and the action is rejected. Thus, unstable configurations are filtered out of the motion planning.

White et al. (2010) addresses the structural functional requirements for self-reconfigurable robotic programmable matter to be usable in the physical world. Connector strength is recognized as the critical factor in holding the system together and a strength analysis is proposed based on a 6×6 stiffness matrix of the heterogeneous system, composed of both rigid and soft connections. They show how heterogeneous stiffness properties displayed by folded chain structures can be exploited to form structures that best suit a given task.

Similarly, in O'hara et al. (2014), the strength requirements of floating structures assembled by self-reconfigurable robotic boats in the open ocean are addressed. A simplified wave and structural strength analysis is presented. The results show that, to conform to the large forces and moments caused by waves, connection stiffness should vary between modules. To address that, active stiffness connections are introduced to reduce structural stress when needed.

Finally, in Tolley et al. (2011), both the structural optimization of the target shape and assembly planning of the optimized shape are addressed in a fluid environment. The target shape is evolved using evolutionary algorithms to minimize the strain energy in the structures with FEA. Once a satisfactory shape is found, the structure is assembled using a self-assembling algorithm based on disassembly: by starting with the final configuration, one cube is removed at a time until no cubes remain. This sequence is then reversed to achieve a top-down centralized assembly strategy.

Unfortunately, of the above examples, only the I-Cubes addresses structurally-driven self-reconfiguration and the structural requirements of human-scale robotic structures on a non-fluid environment.

3.4. SUMMARY

In their review of the current state-of-the-art of SRRs, Seo et al. (2019) recognize structural weakness as the main issue of robotic-assembled structures. Unfortunately, the structural performance of these robotic structures has yet to be properly addressed on a human and architectural scale, where forces and stability plays a crucial role for a structure's success.

To address that, this chapter discusses some of the key factors to consider in the structural study and assembly planning of architectural-scale robotic assemblies, such as module strength, connector strength and stability, and addresses some of the related work on structurally-driven robotic assemblies. In doing that, it hopes to equip the reader with the knowledge needed to address the problem of building structurally-aware robotic-assembled structures.

Chapter 4

Structurally-driven Self-Reconfiguration

In the previous chapter, we discussed how the structural performance and assembly planning of robotic assemblies are crucial for building structurally-aware structures. In this chapter, we propose structurally-driven control strategy for a self-reconfigurable robotic system based on the structural analysis and performance of the resulting robotic configurations, both target and intermediate. The following sections explain the overall methodology of this strategy and the implementation strategies used to develop the control system and drive the self-reconfiguration process based on structural feedback.

4.1. RESEARCH PLAN AND METHODOLOGY

In order to build structurally-aware robotic assemblies, a robotic system of modular robots capable of forming robotic assemblies was developed, which will serve as the test bed for our implementation. The developed system is based on a lattice architecture with homogeneous, functionally identical modules. A cubic lattice architecture was chosen for its spatial symmetry and form-filling properties, which is better for the self-reconfiguration of a wider variety of structures. The system was designed based on strategies of distributed control and local interactions to improve scalability and potentially enable the construction of larger assemblies of modules.

In order to simulate the developed robotic system's physical behaviour and analyse the structural performance of the resulting robotic assemblies, two different simulation environments were used. The first is Unity 3D, which uses the built-in Nvidia PhysX 3.3 physics engine (NVIDIA, 2020) to simulate the robot's physical behaviour and validate stability calculations with rigid-body simulations.

The other simulation environment used was Grasshopper (Davidson, 2020) for structural analysis and optimization. Grasshopper was chosen due to the variety of complementary state-of-the-art plug-ins available that allow designers and engineers to analyse and optimize different types of structures. Specifically, we used tOpos (Białkowski, 2020) for the structural and topology optimization of the target configuration and then Karamba3D (Karamba3D, 2020) to analyse the resulting structures and provide real-time structural feedback with FEA during robotic self-reconfiguration.

In order to have a real-time feedback loop for the self-reconfiguration process, these two environments needed to be tightly connected. To realize that, a User Datagram Protocol (UDP) connection was developed between Unity and Grasshopper for data exchange. From an initial configuration, the robot determines the possible actions based on the local configuration and sends the corresponding geometric data of the resulting configurations to Grasshopper for structural analysis. The feedback of each action is subsequently used in an action selection process that determines the next action the robot should take. This selection process is based on both the structural performance of the robotic configurations as well as convergence

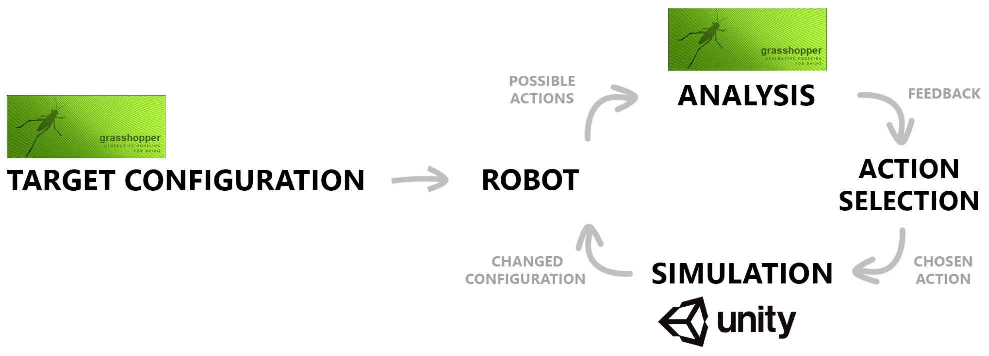

Figure 4.1: *Self-Reconfiguration Workflow between Unity and Grasshopper*

criteria towards the target configuration. Finally, the chosen action is executed and simulated in Unity, resulting in a changed configuration. This process is repeated until the target configuration is reached. This workflow, illustrated in Figure 4.1, allowed us to drive the self-reconfiguration decision-making process based on structural feedback.

4.2. IMPLEMENTATION

4.2.1. Target Shape Optimization

Before starting the reconfiguration process, we need to make sure that the target configuration itself is structurally feasible and stable, otherwise the assembly might never be able to converge into it. To obtain a structurally feasible and optimized target shape, we perform a topology optimization. Topology optimization aims at finding the optimal solid-void pattern of material distribution within a given design domain for a given set of load and boundary conditions (Zhu and Gao, 2016). It uses a FEA to find the most effective load-carrying paths in a structure and thus the shape that minimizes the strain energy in the system and maximizes the global stiffness. This can be useful for the robot to dynamically formulate a target shape that responds to specific on-site load and boundary conditions.

To implement this, we turned to Grasshopper for the tOpos plugin, which uses the Solid Isotropic Material with Penalty (SIMP) method described in Bendsoe and Sigmund (2013) for Topology Optimization. To perform the optimization, we need a boundary domain, support conditions, load conditions and a resolution. The boundary domain defines the boundary space within which the structure should be contained. The support conditions define the area and conditions of support of the structure while the load conditions define the loads the structure will be subjected to. Finally, the resolution is used to discretize the domain space into a voxelized space of nodes and elements to perform the FEA. By matching the resolution size to the module size (5cm for the developed module), we can use this voxelized space to directly obtain the modules' distribution in the domain. Karamba is then used to verify the performance of the resulting shape.

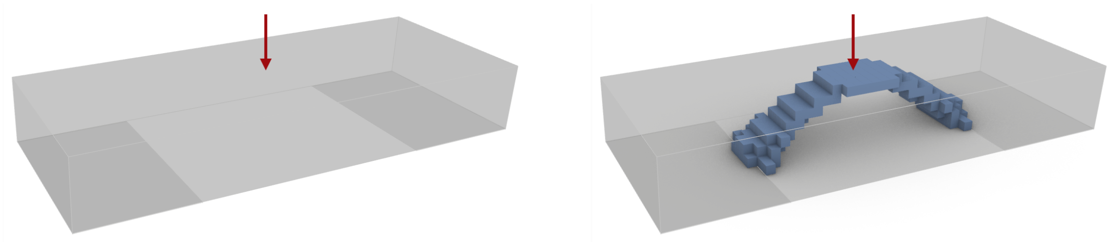

Figure 4.2: *To perform the Topology Optimization, we need a boundary domain (light grey volume), the support conditions (dark grey areas) and the load conditions (red arrow). The resolution defines the scale and discretization of the space for the analysis.*

Once a target shape is found, each module is given a representation of this shape in the form of a list of goal cell coordinates. This list can be used for modules to check the location of target cells in the vicinity as well as in relation to their own position. Once this is done, the reconfiguration process can now start.

4.2.2. Module Design and Motion Control

The design of the developed modular robots was inspired by the 3D-Unit SRR (Murata et al., 1998). Each module has 3 rotational DoFs in orthogonal directions that allow modules to perform ± 90º degree rotations around the center, as shown in Figure 4.3.

This design imposes a few motion constraints on self-reconfiguration: since modules have no autonomy of movement, they have to be carried by a neighbouring module, which we call the Pivot, in a rotating motion. Moreover, to support the pivot's rotating motion, at least one Support module is needed to fix the pivot's axis of rotation. These motion constraints are illustrated in Figure 4.4.

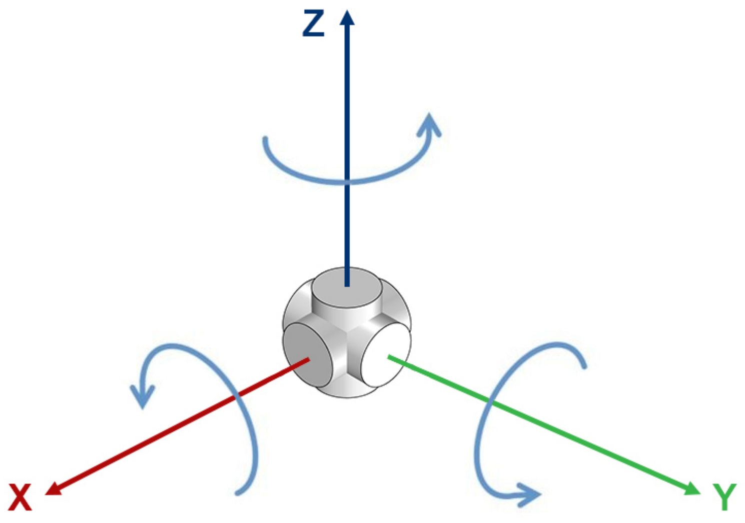

Figure 4.3: *Module design and degrees of freedom*

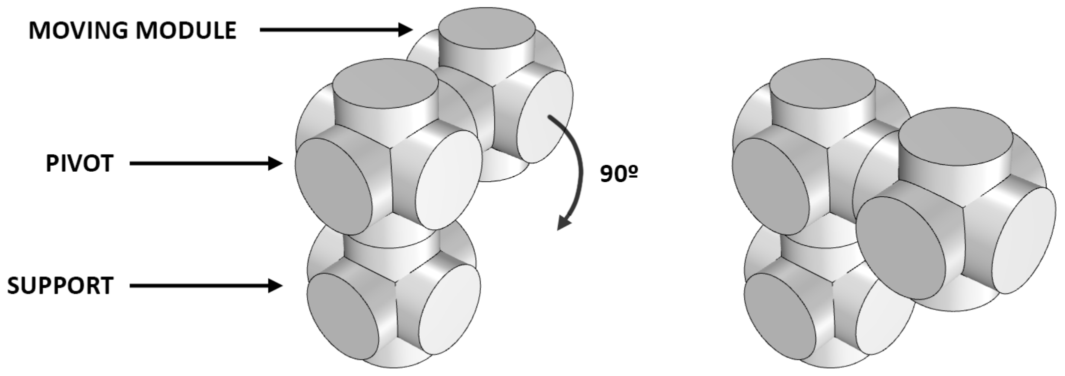

Figure 4.4: ***Module motion criteria**: To perform any motion, a module needs a Pivot module to carry it and a Support module in the direction of the axis of rotation to support the rotating pivot*

To reduce some of the difficulties imposed by these motion constrains and facilitate self-reconfiguration planning, modules always move in meta-modules of two modules, where one module coordinates with a partner module to mutually move each other around the robotic assembly (see Figure 4.5). This is done to avoid lone modules getting stuck in configurations from which they cannot move due to their own motion constraints (e.g., see d) in Figure 4.5).

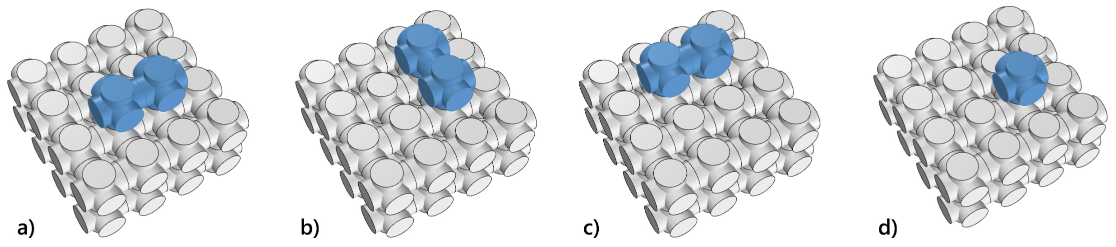

Figure 4.5: ***Meta-module pair-wise motion:** a)-c): by using each other as pivots, two modules can coordinate together to move towards their goal positions; d) If by itself, a lone module can get stuck, unable to move.*

Each module is assumed to be equipped with its own controller and to be able to communicate with its immediate connected neighbors. To sustain larger assembled structures, a sturdy internal framing and a rigid shell is assumed to increase the modules' compressive strength and allow them to support the weight of several other modules. A strong and light material such as aluminium for the shell and internal structure is considered.

Regarding the connectors, mechanical connectors are usually the strongest and the best option for increasing connector strength between modules. All six faces of the modules should have mechanical metallic connectors with three or more points of contact to more evenly distribute stress between modules and provide stiffer and stronger connections to torques from all directions. For this robotic design and motion constraints, unisex connectors could make for a more uniform system but aren't strictly needed: since modules always move in a checkerboard grid of cells (see Figure 4.6), having male and female modules with active and passive connectors could help reduce costs and hardware complexity.

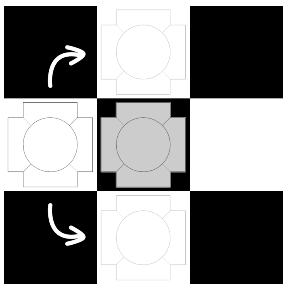

Figure 4.6: ***Modules always move in a Checkerboard grid of cells:*** Modules in white cells can never move into black cells and vice versa.

To make the robotic system more scalable, we use a distributed strategy of motion control based on local rules for the modules. As mentioned previously, a rule consists of a possible action that a module can take based on a specific local neighbourhood (see Figure 4.7). Depending on the existence and distribution of other modules in this neighborhood, several actions are possible. In order to be executable, these actions need to comply with the modules' motion constraints, i.e. a module needs a pivot and at least one support module to move.

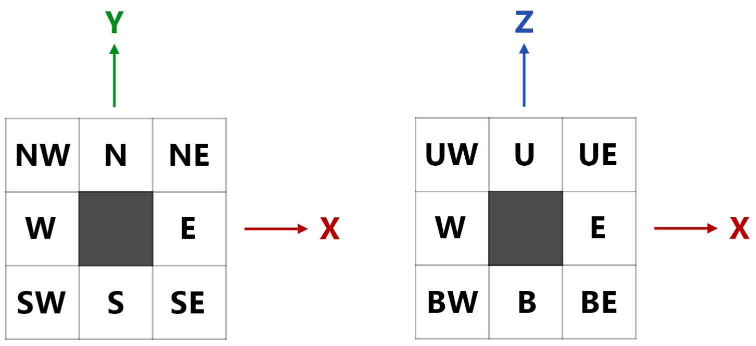

Figure 4.7: ***The local neighbourhood of a module consists of its immediate surrounding neighbours. For convenience, these are named based on cardinal directions (North, East, South, West), and their elevation (Up, Bottom) in relation to the given module.***

The local rules for our robotic system are as follows:

PIVOT LOCATION	SUPPORT LOCATION	ACTION	TARGET CELL
North (N)	Pivot X-axis	X90	Upper North (UN)
		X-90	Bottom North (BN)
	Pivot Z-axis	Z90	North West (NW)
		Z-90	North East (NE)
South (S)	Pivot X-axis	X90	Bottom South (BS)
		X-90	Upper South (US)
	Pivot Z-axis	Z90	South East (SE)
		Z-90	South West (SW)
Up (U)	Pivot X-axis	X90	Upper South (US)
		X-90	Upper North (UN)
	Pivot Y-axis	Y90	Upper East (UE)
		Y-90	Upper West (UW))
Bottom (B)	Pivot X-axis	X90	Bottom North (BN)
		X-90	Bottom South (BS)
	Pivot Y-axis	Y90	Bottom West (BW)
		Y-90	Bottom East (BE)
East (E)	Pivot Y-axis	Y90	Bottom East (BE)
		Y-90	Upper East (UE)
	Pivot Z-axis	Z90	North East (NE)
		Z-90	South East (SE)
West (W)	Pivot Y-axis	Y90	Upper West (UW)
		Y-90	Bottom West (BW)
	Pivot Z-axis	Z90	South West (SW)
		Z-90	North West (NW)
-	-	No Action	Current Cell

Table 4.1: Module Local Rules; the rule highlighted in blue is illustrated in Figure 4.8.

As can be seen in Table 4.1, each module can have six possible pivots according to the existence of a neighbouring module connected in the corresponding direction: North, South, Up, Bottom, East or West. If any of those pivots exists, an action is only possible if a corresponding support module exists in the direction of the pivot's axis of rotation. If both of these conditions are verified, an action is applicable to the current neighbourhood.

Actions are named according to the axis and angle of rotation of the pivot rotation. For example, X90 represents a 90 degrees rotation of the pivot around the X axis. Similarly, Z-90 represents a -90 degrees rotation around the Z axis. Once an action is executed, the module will move towards the corresponding target cell in the neighbourhood, carried by the pivot. To simplify, only one module within a meta-module can move at a time and pivots can also only carry one module. Finally, modules can also choose not to move and remain in their current cell to allow their partner module to perform a second 90 degrees rotation.

To illustrate this rule execution process, the rule highlighted in blue in Table 4.1 can be seen in Figure 4.8.

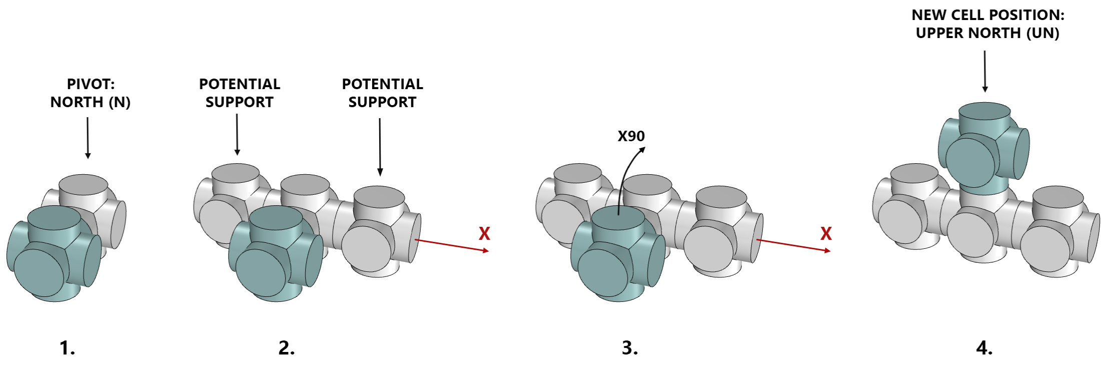

Figure 4.8: ***Rule X90 with a North Pivot:*** *(1.) If there is a connected pivot in the North cell and (2.) at least one support module on the pivot's X axis then (3.) action X90 is applicable. (4.) The Pivot will rotate the module to the Upper North cell.*

Before an action can be executed, one final applicability check is performed for each matched rule from two aspects: collision avoidance and connectivity. Collision avoidance verifies if the target cell and all transition cells where the module will pass through on its way to the target cell are empty. If any of those are occupied, a collision will occur so the rule is removed from the possible rules. Connectivity checks if a module action will cause the robot to split in two. If so, the rule is also removed from the possible rules.

In the end, all applicable rules for a given local neighbourhood are stored. If no rule applies, the module will not move. If only one rule applies, the module will execute the corresponding action. If more than one rule applies, the corresponding actions will undergo a selection process based on the structural analysis and fitness of the resulting configurations to chose the action to be executed. This process is illustrated in Figure 4.9.

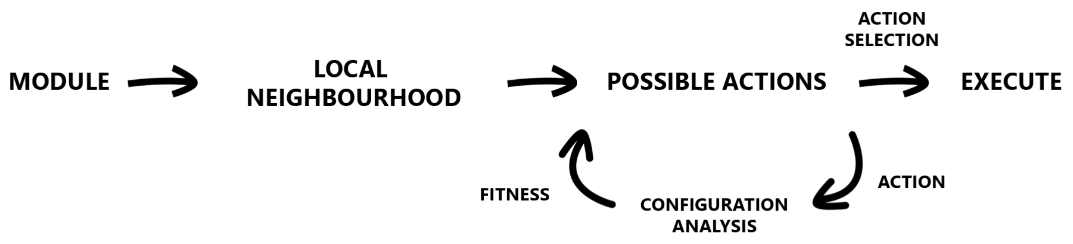

Figure 4.9: ***Action Selection process based on analysis and fitness of the resulting configurations***

4.2.3. Action Analysis and Fitness

Each module action results in a different configuration. If more than one rule applies, all of those possible configurations will have different corresponding fitnesses. We evaluate the fitness of each action based on three fitness criteria: (1) convergence towards the given target configuration, (2) stability of the overall assembly, or (3) the structural performance of the assembly.

Convergence

The Convergence fitness calculates if a module action will cause the module to move closer or farther away from its own goal position in the assembly. In an ideal self-reconfiguration planning sequence, if more than one path is possible, the best one is the one that minimizes the amount of steps modules need to take to reconfigure from an initial configuration to a final goal configuration. Similarly, if more than one rule applies, the most convergent action is the one that brings the module closer to its goal.

In the developed robotic system, assembly sequence and convergence are achieved through attraction gradients. When the system is initialized, one of the modules located next to an empty goal position will be selected to act as the Seed, emitting a virtual scent gradient which will be propagated throughout the modules. This is represented by a numeric value that gradually decreases the further away a module is from the source (i.e. the Seed), as illustrated in Figure 4.10. Using the scent values propagated throughout the modules as reference, a moving module can calculate if a given action will result in an increase or decrease of its own scent value. Convergence is achieve by following the scent gradient back to the source, which will bring the module towards the empty goal position. Once this happens, a new Seed will be selected and a new scent gradient will be propagated to attract other modules. For one meta-module, two consecutive Seeds will always be selected next to each other.

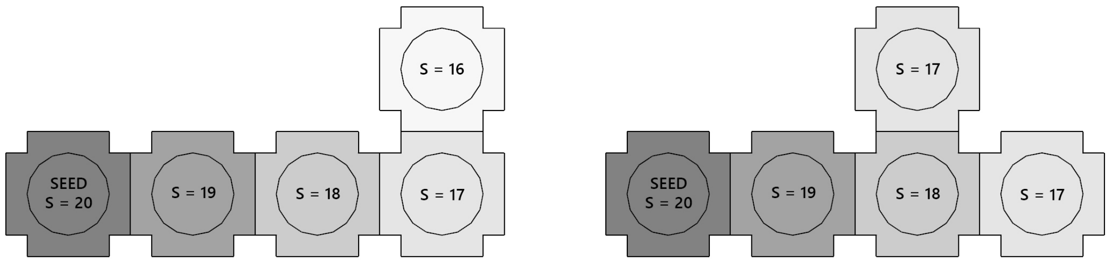

Figure 4.10: ***Attraction Gradients:*** *The Seed module propagates a "scent" value that points modules towards the location of an unfilled goal position. Each time a module moves, it's own scent value is updated according to the gradient.*

Stability

The Stability fitness calculates the stability of the overall robotic structure. Stability is calculated using a similar analysis method as Ünsal et al. (2000). First, the center of gravity of the entire robot is calculated as the average between the center of gravity of all modules in the assembly and projected onto ground plane. If the projected point falls within the area corresponding to the stationary footprint of the robot, the assembly is considered stable; if not, it is considered unstable. For configurations with only one support area, the stationary footprint area of the robot corresponds to the area where stationary modules are in contact with the ground. For configurations with more than one support area, it also considers the projected 'shadow' area between the supports (see Figure 4.12).

For the selection process, we also implemented a *safety threshold*. Before reaching the point of instability, if the center of gravity of the robot falls too close to the boundary of the footprint area of the robot, the resulting axial forces might cause the structure to form a hinge and fall. Thus, we use a safety threshold factor to uniformly reduce the footprint area of the robot on all sides. This divides the area into three zones: the safe and stable zone, the threshold zone and the unstable zone. If during the course of a module move, the center of gravity of the robot falls within the safe and stable zone (light grey area in Figure 4.11), the assembly proceeds according to convergence. If it falls within the threshold zone (dark grey area in Figure 4.11), the action is accepted but a subsequent corrective action that maximizes stability becomes necessary. This is done by choosing the action that minimizes the distance between the center of gravity of the robot and the center of the footprint area (distance d in Figure 4.11). As a result, the chosen action is the one that best counterbalances the weight distribution of the robot. Lastly, all actions that fall into the unstable zone are eliminated.

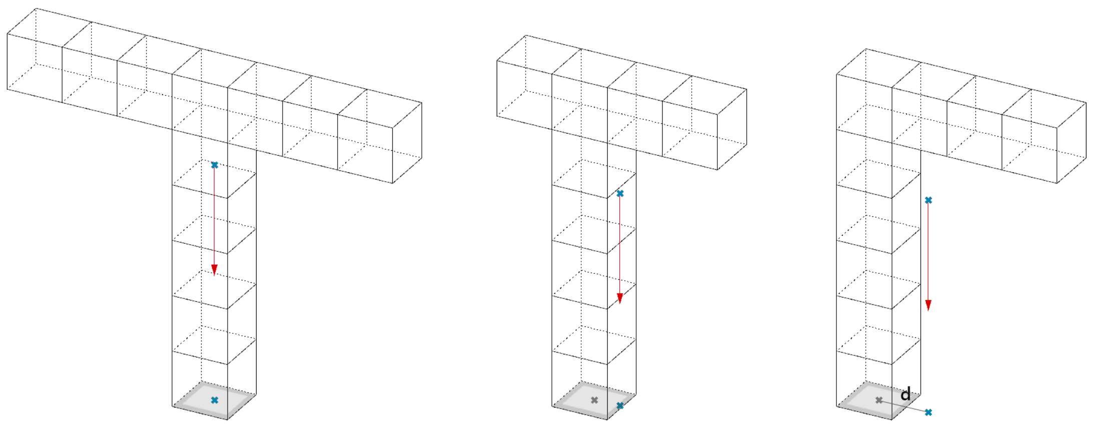

Figure 4.11: ***Stability Analysis:*** *If the projected center of gravity of the robot falls within the safe and stable zone (light grey area), the assembly is stable. If it falls within the threshold zone (dark grey area), the assembly is stable but approaching instability; a corrective action is needed. If it falls in the unstable zone (i.e. outside of the footprint area of the robot), the assembly is unstable.*

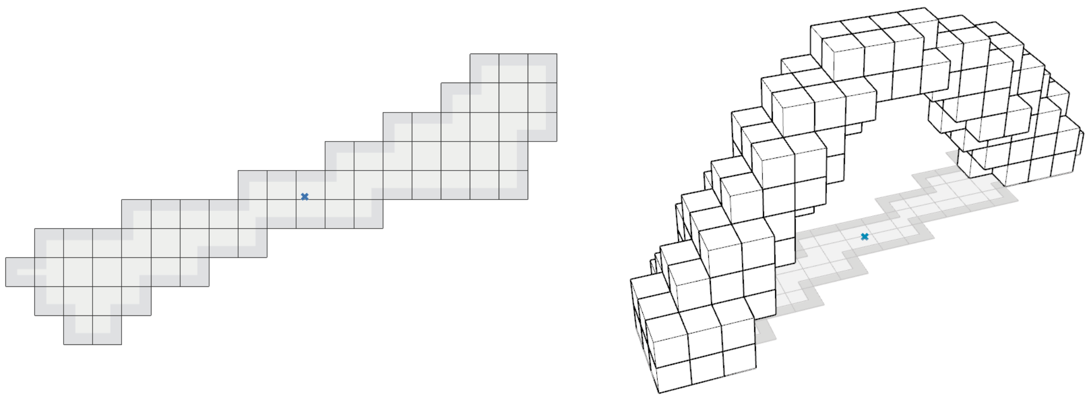

Figure 4.12: ***Footprint Area of the robot for configurations with two or more support areas.***

Structural Performance

The assembly's structural performance was analyzed with Karamba3D. To do that, Unity encodes and sends the robotic assembly's geometric data, i.e. module coordinates, size and active connections, over to Grasshopper via the established UDP connection, which receives that information and uses it to build a simplistic analytical model. This model is represented with nodes as module centers and aluminium beams as connections between modules. Each node has a point load representing module self-weight, assumed to be 500g for each module, and modules touching the ground plane are considered support nodes. The support nodes have all of their translation and rotational DoFs fixed for analysis purposes. This analytical model is illustrated in Figure 4.13.

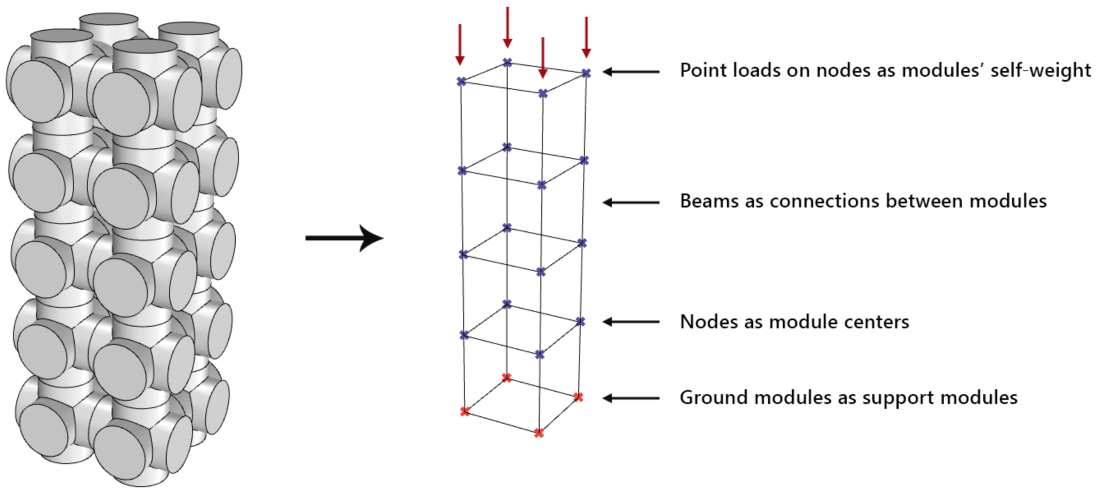
Point loads on nodes as modules' self-weight
Beams as connections between modules
Nodes as module centers
Ground modules as support modules

Figure 4.13: *Analytical model for Structural Analysis*

Using the described analytical model, Karamba calculates the deflection model for the assembled structure and the nodal displacement for each module, which was the chosen fitness for structural feedback. Nodal displacement is representative of the bending of the structure, which, when big enough, can cause excessive stress on modules' connectors. Once calculated, the maximum nodal displacement between all modules is sent back to Unity. Since each module action represents a different resulting geometric configuration, the nodal displacements of the different actions can be compared to find the action with the best performance, i.e. the smaller displacement.

However, not all nodal displacements are problematic. Small nodal displacements are unlikely to cause too much stress in the structure and for that reason, like with the stability analysis, a safety threshold and a danger threshold were implemented. If the maximum nodal displacement falls under the safety threshold, the assembly proceeds according to convergence; if it falls over the safety threshold but under the danger threshold, a corrective action that minimizes nodal displacement is needed. All actions that fall over the danger threshold are eliminated.

4.2.4. Action Selection

As we saw in the previous sections, we have three fitness criteria that modules need to consider when choosing an action. Choosing which fitness to prioritize depends on the calculated values of the different fitnesses and whether they fall within their respective thresholds. Actions that maximize convergence are preferable for a quicker and more efficient self-reconfiguration process. However, always seeking the most convergent actions can cause the structure to break or become unstable if no structural factors are considered. In those cases, convergence must be sacrificed for a safer and more stable assembly. This is done through the thresholds.

An action that maximizes convergence and falls under the stability and nodal displacement thresholds will always be preferred. If that is not possible, actions that fall over the safety threshold are still allowed but require an immediate corrective action to counterbalance the loss of performance. Actions considered unstable or dangerous for structural performance are disregarded altogether. Table 4.2 shows how priorities for corrective actions are chosen according to the calculated fitnesses.

STABILITY / DISPLACEMENT	UNDER THRESHOLD	OVER THRESHOLD
UNDER THRESHOLD	Maximize Convergence	Maximize Stability
OVER THRESHOLD	Minimize Nodal Displacement	(1) Maximize Stability; (2) Minimize Nodal Displacement

Table 4.2: ***Priority Selection***

4.2.5. Rigid-body Simulations

To simulate the behaviour of the robotic assembly under the action of gravity and validate the stability calculations of the structure, Unity's physics engine was used. Unity uses rigid-body dynamics to study the movement of interconnected rigid-bodies under the action of external forces. Bodies are assumed to be rigid and not deform under applied loads, which simplifies computationally expensive calculations and produces faster results without significant loss of accuracy (Kao et al., 2017).

Using this method, modules are assumed to be rigid-bodies with negligible deformation, bonded together by joints. For simulation purposes, modules have a static and dynamic surface friction coefficient of 1.05 and 1.4 respectively, i.e. the coefficients for aluminium, and joints between modules have a break Force and Torque of 500N. The ground surface is assumed to be solid and flat; uneven and obstacle-ridden environments were not considered in the scope of this thesis.

4.3. SUMMARY

In this chapter, we went over the methodology and implementation details of the developed robotic system and structurally-driven self-reconfiguration strategy. In the following chapter, we will go over the results of this strategy.

Chapter 5

Results

In order to evaluate the developed self-reconfiguration strategy, we studied the target shape and assembly sequence. The results will be showed in the following sections.

5.1. GOAL SHAPE OPTIMIZATION

If the robot encounters an unexpected obstacle that requires load-bearing capacity, it might have to reconfigure into a configuration capable of handling these loads. For that, instead of having a set of pre-programmed configurations that the robot can morph into, it's better if the robot can dynamically adapt its shape to specific on-site conditions. With topology optimization, the goal shape itself is irrelevant; what matters is its ability to carry the loads and the global stiffness of the system. This allows the robot to formulate a configuration that would best fit a given task.

Revisiting the bridge example previously shown in Figure 4.2, consider a situation where the robot needs to build a bridge structure capable of carrying a single person across a gap. For that, we need a boundary domain, the support conditions and a loading condition. The boundary domain is given by a general geometry that defines the available unobstructed space where the structure can grow. The support conditions are defined by two areas within the boundary domain with a gap of 2m between them. In an on-site situation, these areas could be defined according to local information and terrain conditions. The applied load for this example corresponds to the weight of a single person (80kg or 0.78kN) in order to build a structure capable of supporting one person. Figure 5.1 shows the evolution of the goal shape during the topology optimization, where unnecessary modules are gradually discarded from the structure until only the most essential modules for structural stiffness remain. Within ten iterations, an acceptable solution can be found.

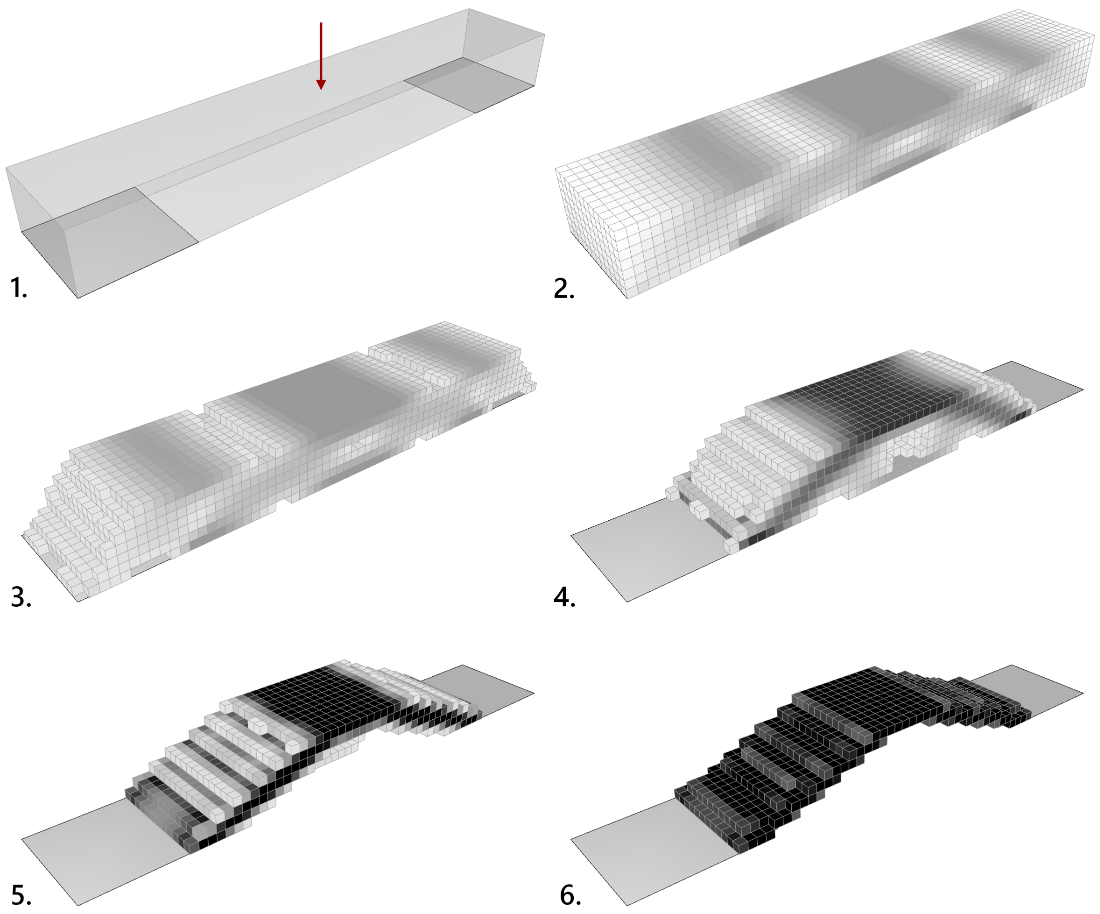

Figure 5.1: ***Evolution of the Goal shape during the Topology Optimization:*** *1. Given boundary and load conditions; 2. First iteration; 3. Second iteration; 4. Third iteration, 5. Fifth iteration; 6. Tenth iteration.*

Ultimately, the topology optimization only gives us a general optimized shape designed to deal with specific load and boundary conditions. It does not necessarily guarantee a good performing structure. For that, we need to make sure that the performance of the resulting shape is adequate for the expected use. This final verification was done with Karamba.

One important factor in this performance is the solid/void ratio of the topology optimization, i.e. how much material (modules) should be distributed in the boundary domain. If too few modules are used, the structure will be fragile and likely break under the expected load. Conversely, using too many modules is both impractical and can overload modules at the bottom of the structure with self-weight alone. Ideally, we should find the balance between the number of modules and the resulting performance. Figure 5.2 shows how the solid/void ratio and, consequently the number of modules, can impact the assembly's structural performance. Overall, to build a bridge capable of supporting one person spanning 2m and with a width of 0.6m, structures with a thickness of less than two layers of modules proved to be too brittle to support the expected loads.

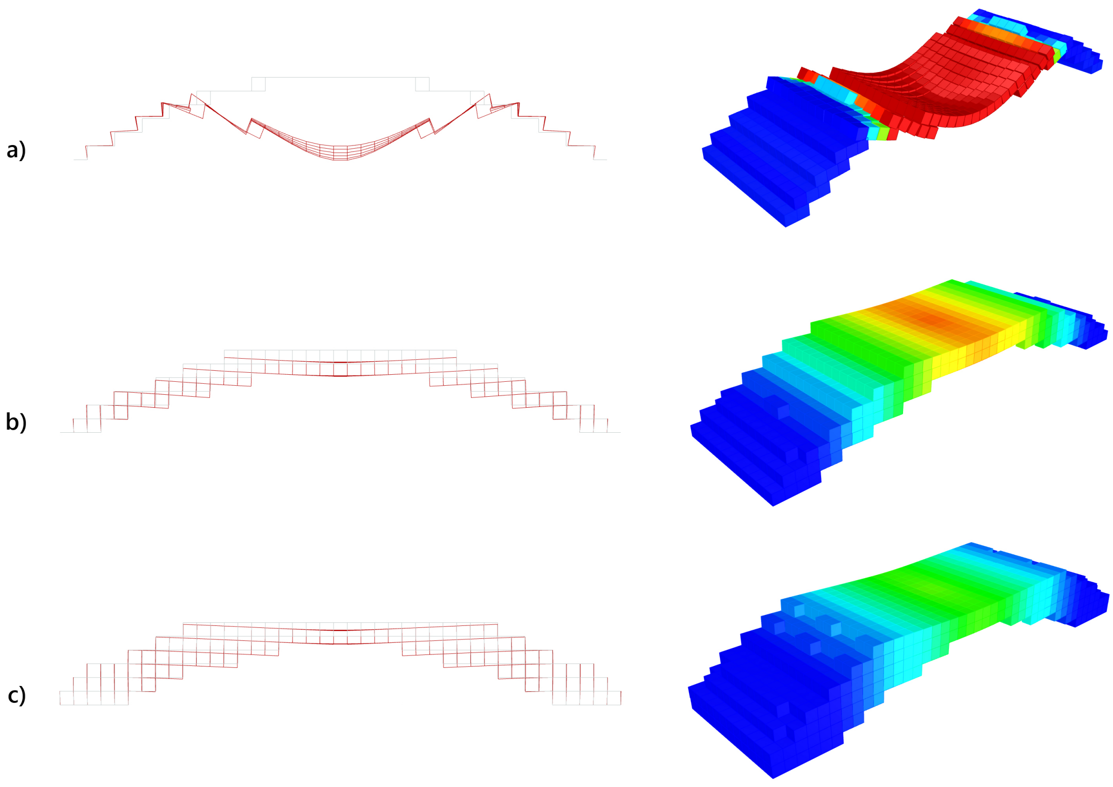

Figure 5.2: ***Solid/void ratio variations:*** *a) 20%, corresponding to 608 modules, b) 30%, corresponding to 1184 modules, c) 50%, corresponding to 1632 modules. The image on the left shows the structure's deflection; the image on the right shows the displacement of each module.*

Once an acceptable target shape is found, the target coordinates of the goal solution are sent to the modules to start the reconfiguration process.

5.2. RECONFIGURATION SEQUENCE

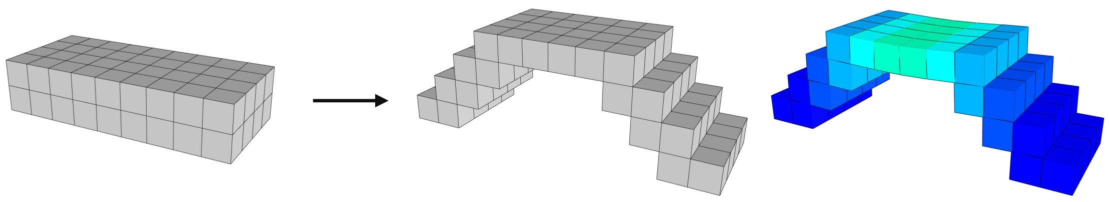

Figure 5.3: ***Initial and Target Configuration***

To evaluate the proposed self-reconfiguration strategy, let us consider the transition from an initial compact configuration to a smaller-scaled bridge configuration (see Figure 5.3), obtained using the method explained in the previous section. This shape was obtained using a smaller domain space, support areas spaced by 0.4m between them and a 15kg load. The resulting shape is made of 56 modules.

Once the reconfiguration process is initialized, all modules will check their own position according to the list of target cell coordinates and update their own state according to the initial conditions. All modules already located in a goal position will update their state to Final and will not move anymore. All other modules, will update their own state to Inactive. From among the modules in their Final state, one of the ones next to an empty goal position will then be selected as the Seed module and propagate a scent value that will inform nearby modules of the presence and proximity of an unfilled goal position. Finally, the farthest two Inactive modules from the Seed, i.e., the modules with the lowest scent value, will be selected as a meta-module of two modules and will update their state to Active to start moving.

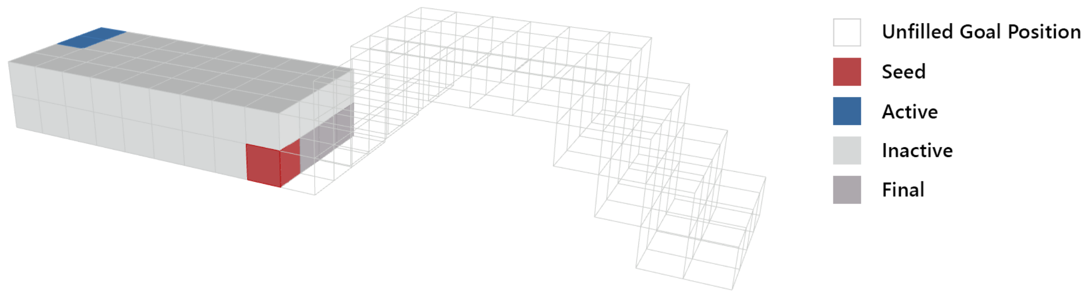

Figure 5.4: ***System Initialization and Module States***

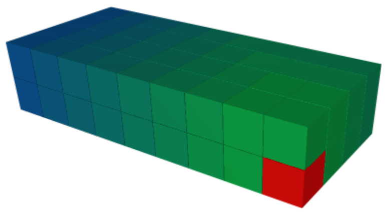

Figure 5.5: ***Scent Propagation: The red Seed module represents the origin of the virtual scent. The further from the Seed, the lower modules' scent values are.***

Once the reconfiguration process starts, modules in the meta-module will alternate in choosing an action to execute. To do that, they will check their current local neighbourhood to determine the applicable rules and possible actions, as explained in section 4.2.2. Each of these actions will result in a different configuration and corresponding fitness. For example, for the left-most module in the meta-module, Figure 5.6 shows the possible actions and corresponding calculated fitnesses of these actions.

In this example, all actions result in stable configurations and negligibly small nodal displacements that fall under their respective safety thresholds so actions are chosen according to convergence. Both action X90 and Z90 result in positive convergence, i.e. the module getting closer to the Seed. However, action Z90, causes the module to move outside of the range of its meta-module partner so action Z90 is disregarded to prevent meta-module separation. Thus, action X90 is the clear winner for this configuration. In case there is more than one best action with the same convergence fitness (see Figure 5.7), one of them will be randomly selected. Action 'No Action' is only selected when no other action has a better or same performance. Figure 5.8 shows a sequence of selected actions taken for a meta-module to reach the Seed and, consequently, their respective goal positions.

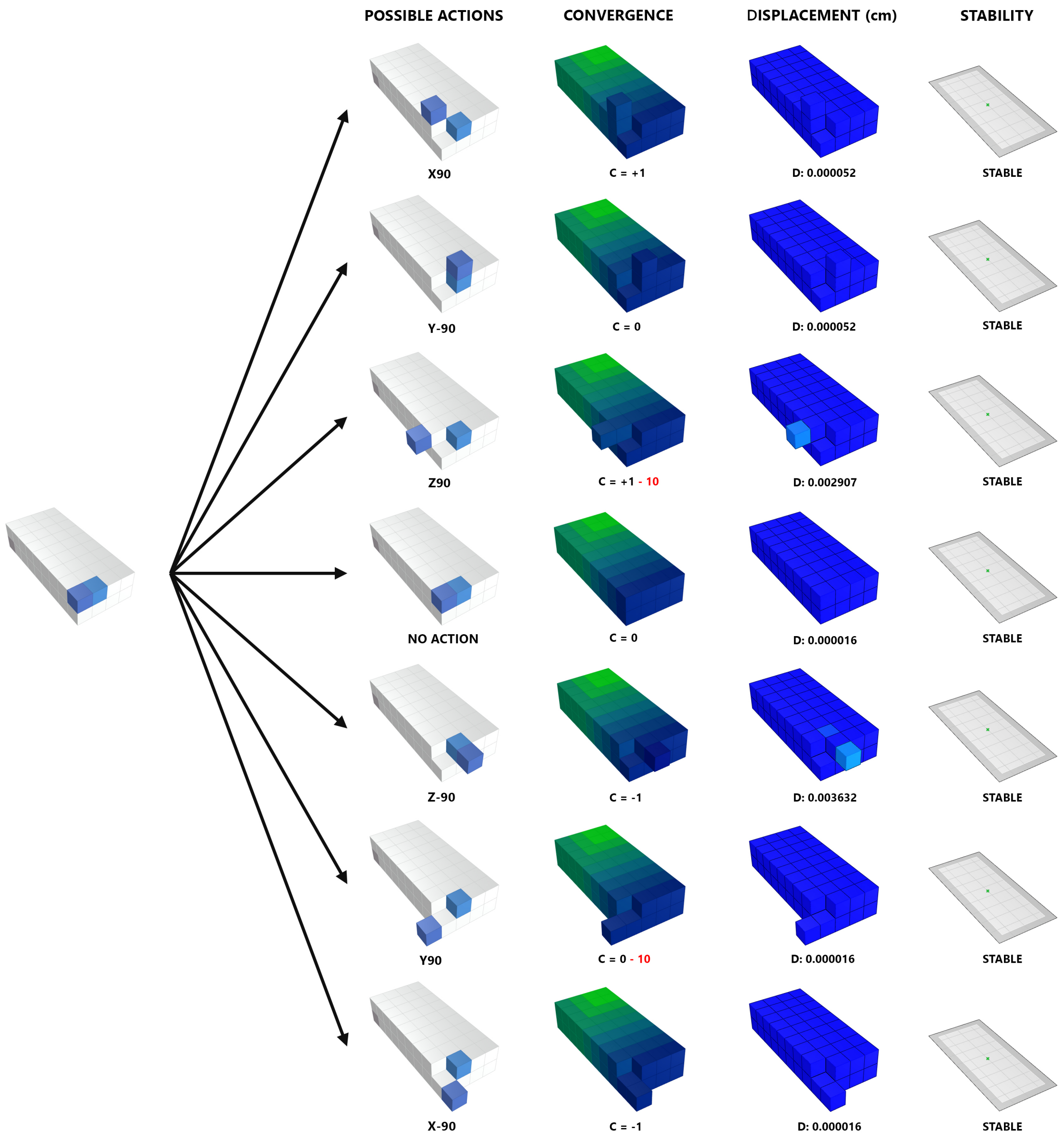

Figure 5.6: *Module actions and corresponding fitnesses.*

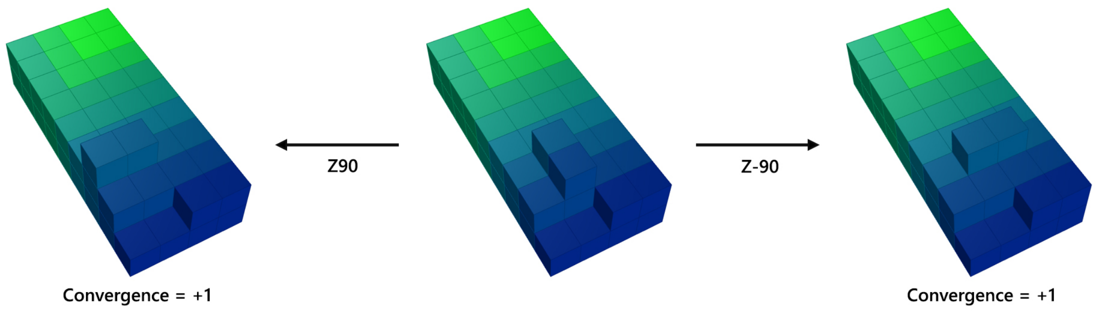

Figure 5.7: *Module actions resulting in the same best convergence fitness. Both actions are equally valid but only one of them will be randomly selected.*

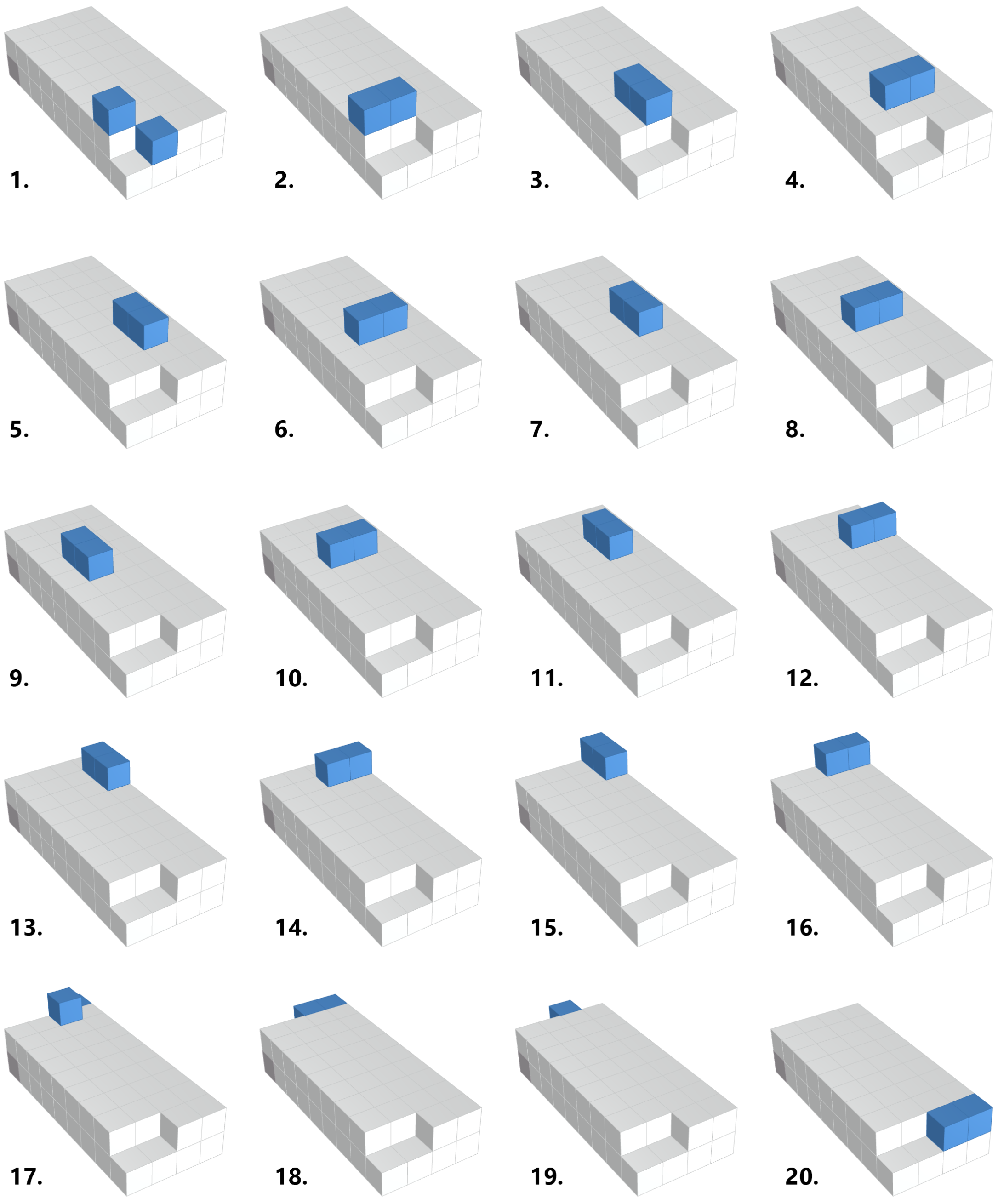

Figure 5.8: ***A Meta-module reconfiguration Sequence:*** *As previously explained, modules will alternate actions to move forward and reach the Seed. For the two modules in the meta-module, two consecutive Seeds will point towards two unfilled goal positions right next to each. Once both modules reach their final goal positions, a new meta-module will be chosen.*

As the assembly grows, it eventually reaches a point where the structural performance of the assembly starts to approach structural failure, i.e. the calculated fitnesses fall over the established safety thresholds for nodal displacement and stability. At this point if the robot were to continue building without any structural considerations, the structure would eventually fail before reaching its target configuration (see Figure 5.9).

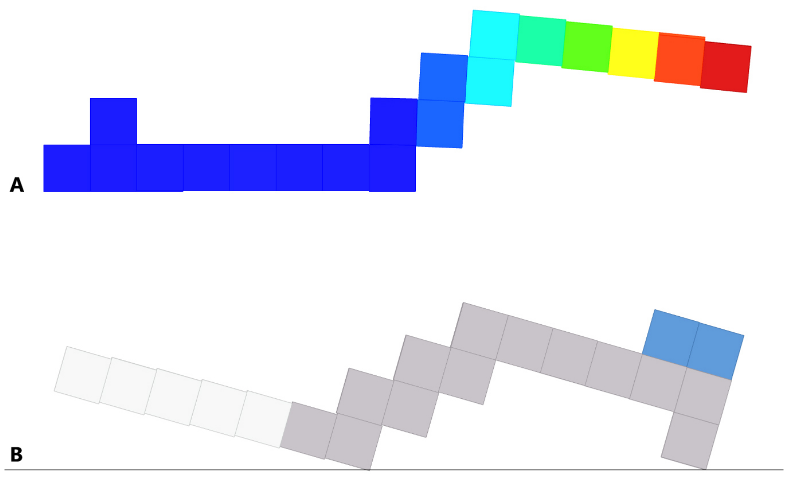

Figure 5.9: ***Structural Failures: A - Displacement failure:*** *the maximum displacement of the structure is over the given Danger Threshold, i.e. the equivalent of half a module (2.5 cm), which can put excessive stress on the modules' connectors and can cause them to break;* ***B - Stability Failure:*** *the structure became unstable, which caused it to topple*

Unfortunately, while the proposed strategy is successful in preventing the structure from ever reaching structural failure, it also prevents the structure from reaching the target configuration since convergence and structural performance are opposite goals in this bridge example. In order to build the bridge, modules need to build a large cantilever whose structural performance only worsens as the cantilever gets longer. As a result, when the structure reaches the safety thresholds, modules will try to compensate for the declining performance by choosing actions opposite to the direction of the cantilever, preventing the structure from finishing.

If we only consider the stability of the structure, the order of assembly of the modules into their goal positions is actually a major factor in achieving a complete and stable reconfiguration sequence. By prioritizing reaching the other side of the bridge with a small arm of modules, the robot can find support for stability on the other side while most of its weight still rests on the initial side of the bridge. Once the gap is bridged, other modules can then start widening the bridge and distributing the weight of the robot between the two supported sides. In the developed robotic system, this is achieved by driving Seed selection to prioritize unfilled goal cells in the direction of the gap first. The rule selection strategy then makes sure that no intermediate configurations, i.e. the paths modules take, compromise the stability of the assembly. This sequence is illustrated in Figure 5.12.

This approach, while stable, does not solve the bending problem which is still unsolved in developed system. If the arm is too thin, the weight of the cantilever will cause the structure to bend and the resulting stress in the module connectors might cause them to break before the assembly reaches the other side. To solve this, a reinforcement strategy would be needed in addition to the stable approach, where modules are gradually recruited to reinforce the suspended cantilever and relieve the stress in the overstrained modules until the structure reaches the other side. The idea behind this strategy is explained in Figure 5.10. To implement this, increasing the redundancy of modules in the robot might be needed to allow spare modules to take care of structural support, i.e., additional weight for stability and reinforcement of overstrained modules

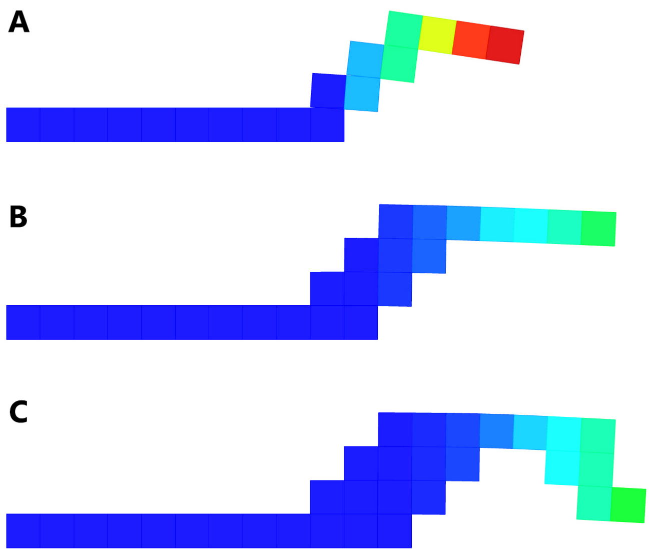

Figure 5.10: *Reinforcement strategy to reduce structural bending.*

Figures 5.11 and 5.12 show different reconfiguration sequences according to different fitness criteria: 1) convergence, and 2) convergence and stability, respectively. A reconfiguration sequence that could achieve all of the above as well as adapt and compensate for the declining performance of the modular bending would still be need to be implemented to successfully build the bridge example.

Finally, sometimes a module moving into its final goal position will cause it's state to shift to Final while the partner module is in a position from which it cannot move alone or reach its own final goal position (see Figure 5.13). When that happens, because the settled module will not move anymore to help the module left behind move to its goal position, the module becomes stranded. This means that it will stop in place and act as a temporary Seed while a new meta-module is chosen and attracted towards the stranded module.

When the new meta-module reaches the stranded module, the stranded module will join it to form a meta-module of three modules and all three modules will coordinate to reach the original Seed. Most of the times, this strategy is successful in rescuing the stranded module. Other times, new stranded modules are born from the difficult coordination between the three modules and separation of the meta-module. This is another obstacle to convergence which prevents the reconfiguration from being completed. More work needs to be done to solve this problem and eliminate the occurrence of stranded modules entirely.

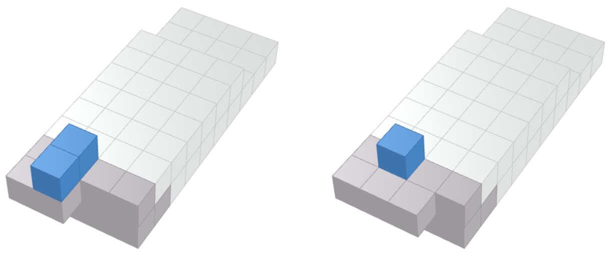

Figure 5.13: *A Stranded Module is caused by a module reaching its goal position while it's partner module is in position from which it cannot move.*

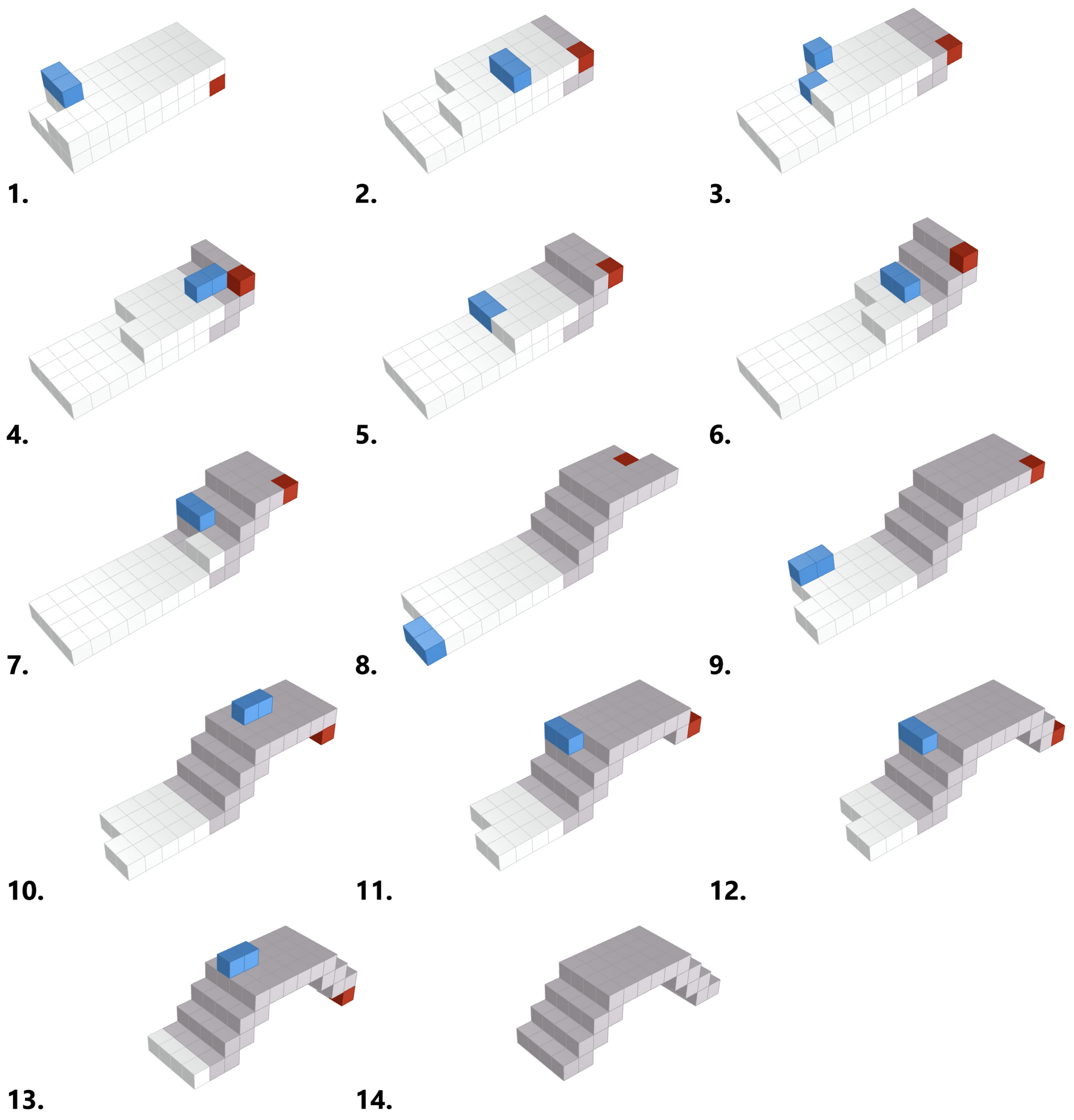

Figure 5.11: ***Reconfiguration sequence according to Convergence.*** *In reality, the structure would not be able to grow past step 8 without breaking and step 9 without toppling (see structural failures in Figure 5.9).*

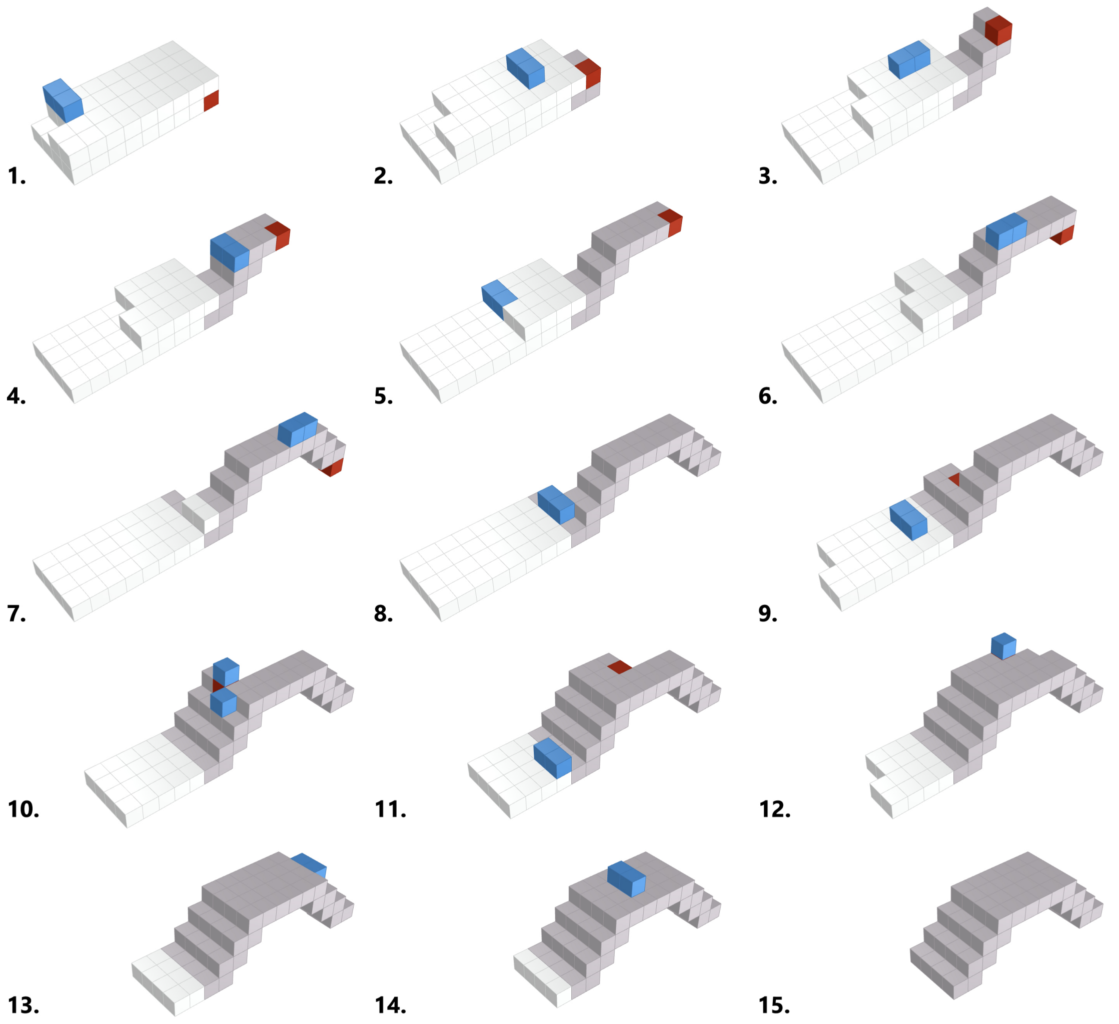

Figure 5.12: ***Reconfiguration sequence according to Convergence and Stability.*** *This sequence remained stable throughout the assembly but would still face issues regarding structural bending, which would cause it to break after step 5.*

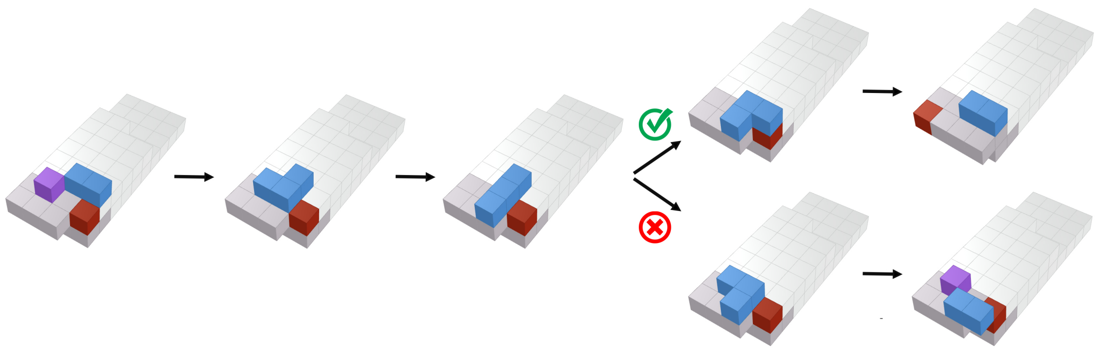

Figure 5.14: *A new meta-module rescuing a stranded module and failing to do so. The stranded meta-modules is represented in purple*

5.3. DISCUSSION

The approach used in this research relies on a distributed self-reconfiguration strategy based on local motion rules and structurally-driven action selection with a centralized analysis workflow to calculate global structural performance of both the target and intermediate configurations. Regarding the target configuration, a topology optimization allows the robot to formulate a target shape for specific local conditions and loads. With the help of sensors and camera modules, this information could be collected from the surrounding environment to map areas for the robot to use. The obtained shape does not have to be the *best* shape for a specific task but a minimum requirement of structural robustness is needed for the structure to be built and stand on its own. This verification was done with a FEA, where the number of modules needed to achieve a robust structure could be determined for the obtained shape.

Regarding the reconfiguration sequence, while the proposed strategy succeeds in preventing the structure from ever reaching structural failure by analyzing every reachable intermediate configuration of a module, filtering out dangerous or unstable configurations, and warning the robot when it approaches dangerous configurations, it fails to adapt when structural performance and convergence are opposite goals. The corrective measures in this case are not effective in dealing with the declining performance.

It turned out to be more difficult than anticipated to achieve both convergence and the desired global structural behaviour using only local decision-making. Considering just the next local action and resulting configuration as the driving force of decision-making turned out to be too limited of a scope to achieve a complete and efficient reconfiguration process for all three performance criteria: convergence, stability and nodal displacement. When considering the stability of a structure, the positioning of modules in the configuration and the weight distribution of the assembly as a whole play a major role in achieving stable configurations. It is therefore difficult to plan the reconfiguration sequence of a complete and stable target configuration on a purely local scale.

Regarding the bending of the structure and modules' nodal displacement, a localized reinforcement strategy could be used to temporarily relieve the stress in overstrained modules and support the cantilever until it reaches the other side. Future work should address all three of these criteria to successfully build structurally-driven assemblies.

The analysis workflow of structural performance based on the exchange between different software, Unity and Grasshopper, was adequate as a proof of concept to analyze and simulate the behaviour of the robotic assembly but ended up significantly slowing down the reconfiguration process. Each configuration calculation (i.e. sending the geometric information to Grasshopper, calculating the structural performance with Karamba and returning the feedback to Unity) takes averagely 160ms per configuration. When each module has between 2 to 7 possible actions and resulting configurations to analyse, this process can take up to

1.12s per module motion. This becomes even more significant when we consider that a complete reconfiguration sequence for 56 modules can take on average 27 module motions per module. Implementing an integrated and simplified analysis method to measure structural performance could significantly improve the performance of the system.

Chapter 6

Conclusion

This research addresses the structural requirements for building robotic assemblies of modular robots with SRRs. It proposes a structurally-driven control strategy for a SRR system based on the structural analysis and performance of the robotic configurations, both target and intermediate during self-reconfiguration.

Using topology optimization, the target structure is evolved to obtain a structurally feasible target shape that responds to specific boundary and loading conditions. Once a suitable target configuration is found, the distributed control strategy drives the modules' action selection process based on the fitnesses of the resulting robotic configurations according to convergence, stability and modules' nodal displacement. A safety threshold is implemented for both the assembly's stability as well and modules' nodal displacement to inform the robot of approaching unstable or dangerous configurations and allow it take the necessary corrective actions to compensate for the declining performance as needed. An instability and danger thresholds are used to filter out actions that result in unstable and overly deformed configurations from the reconfiguration sequence entirely.

Unfortunately, the control strategy used fails to adapt when structural performance and convergence are opposite goals and corrective measures to fix declining performances are ineffective. Making sure that both target and intermediate configurations are safe and stable does not guarantee the emergence of a structurally-feasible complete reconfiguration sequence. A alternative reinforcement strategy where modules are gradually recruited to relieve the stress in overstrained modules was proposed but still needs to be tested for feasibility and effectiveness.

Moreover, the reconfiguration sequence still faces some problems with modules becoming stranded due to the unsuccessful module coordination in meta-modules, which reduces the efficiency of the overall self-reconfiguration algorithm. Finally, the analysis workflow can be improved by implementing an integrated structural analysis method that modules can quickly perform without sacrificing performance. Future work would need to fix all of these issues to more efficiently drive the self-reconfiguration process to build structurally-driven assemblies.

Using local decision-making to solve global structural issues can be challenging and more research needs to address this in order to build structurally-sound robotic assemblies. This thesis only scratches the surface on this issue but addresses some important structural requirements for building these structures and extends the state-of-art on structurally-aware SRRs. In the future, these would need to be developed on a larger scale in order to operate at the architectural scale.

Bibliography

Hossein Ahmadzadeh and Ellips Masehian. Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization. *Artificial Intelligence*, 223:27–64, 2015.

José Baca, Prithvi Pagala, Claudio Rossi, and Manuel Ferre. Modular robot systems towards the execution of cooperative tasks in large facilities. *Robotics and Autonomous Systems*, 66:159–174, 2015.

Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

Sebastian Białkowski. topos, 2020. URL https://www.food4rhino.com/app/topos. Accessed: 2020-08-24.

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

Julien Bourgeois, Benoit Piranda, Andre Naz, Nicolas Boillot, Hakim Mabed, Dominique Dhoutaut, Thadeu Tucci, and Hicham Lakhlef. Programmable matter as a cyber-physical conjugation. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002942–002947. IEEE, 2016.

Alberto Brunete, Avinash Ranganath, Sergio Segovia, Javier Perez de Frutos, Miguel Hernando, and Ernesto Gambao. Current trends in reconfigurable modular robots design. *International Journal of Advanced Robotic Systems*, 14(3), 2017.

Zack Butler, Keith Kotay, Daniela Rus, and Kohji Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *The International Journal of Robotics Research*, 23(9):919–937, 2004.

Andres Castano, Wei-Min Shen, and Peter Will. Conro: Towards deployable robots with inter-robots metamorphic capabilities. *Autonomous Robots*, 8(3):309–324, 2000.

Andres Castano, Alberto Behar, and Peter M Will. The conro modules for reconfigurable robots. *IEEE/ASME transactions on mechatronics*, 7(4):403–409, 2002.

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

Gregory S Chirikjian. Kinematics of a metamorphic robotic system. In *proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 449–455. IEEE, 1994.

David Johan Christensen. Evolution of shape-changing and self-repairing control for the atron self-reconfigurable robot. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, pages 2539–2545, 2006.

David Johan Christensen and Kasper Stoy. Selecting a meta-module to shape-change the atron self-reconfigurable robot. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2532–2538. IEEE, 2006.

David Johan Christensen, Esben Hallundbok Ostergaard, and Henrik Hautop Lund. Metamodule control for the atron self-reconfigurable robotic system. In *Proceedings of the the 8th Conference on Intelligent Autonomous Systems (IAS-8)*, pages 685–692. Citeseer, 2004.

David Johan Christensen, Jens Christian Andersen, Mogens Blanke, Lidia Furno, Roberto Galeazzi, Peter Nicholas Hansen, and Mikkel Cornelius Nielsen. Collective modular underwater robotic system for long-term autonomous operation. In *ICRA Workshop on Persistent Autonomy for Aquatic Robotics: the Role of Control and Learning in Single and Multi-Robot Systems*, 2015.

Jay Davey, Ngai Kwok, and Mark Yim. Emulating self-reconfigurable robots-design of the smores system. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4464–4469. IEEE, 2012.

Scott Davidson. Grasshopper, 2020. URL https://www.grasshopper3d.com/. Accessed: 2020-08-13.

Toshio Fukuda and Yoshio 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*, pages 662–667. IEEE, 1990.

Toshio Fukuda and Seiya Nakagawa. Dynamically reconfigurable robotic system. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1581–1586. IEEE, 1988.

Kyle Gilpin, Keith Kotay, Daniela Rus, and Iuliu Vasilescu. Miche: Modular shape formation by self-disassembly. *The International Journal of Robotics Research*, 27(3-4):345–372, 2008.

Seth Copen Goldstein, Jason D Campbell, and Todd C Mowry. Programmable matter. *Computer*, 38(6):99–101, 2005.

L. Jinguo, Z. Xin, and H. Guangbo. Survey on research and development of reconfigurable modular robots. *Advances in Mechanical Engineering*, 8(8), 2016. doi: 10.1177/1687814016659597.

Morten Winkler Jorgensen, Esben Hallundbk Ostergaard, and Henrik Hautop 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)*, volume 2, pages 2068–2073. IEEE, 2004.

Akiya Kamimura, Haruhisa Kurokawa, E Toshida, Kohji Tomita, Satoshi Murata, and Shigeru Kokaji. Automatic locomotion pattern generation for modular robots. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 714–720. IEEE, 2003.

Gene T.C. Kao, Axel Körner, Daniel Sonntag, Long Nguyen, Achim Menges, and Jan Knippers. Assembly-aware design of masonry shell structures: a computational approach. In *Proceedings of the IASS Annual Symposium 2017*, volume 23, pages 1–10. International Association for Shell and Spatial Structures (IASS), 2017.

Karamba3D. Karamba3d, 2020. URL https://www.karamba3d.com/. Accessed: 2020-08-24.

Michihiko Koseki, Kengo Minami, and Norio Inou. Cellular robots forming a mechanical structure (evaluation of structural formation and hardware design of “chobie ii. In *in Distributed Autonomous Robotic Systems*. Citeseer, 2004.

Keith Kotay and Daniela Rus. Efficient locomotion for a self-reconfiguring robot. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2963–2969. IEEE, 2005.

Keith Kotay, Daniela Rus, Marsette Vona, and Craig McGray. The self-reconfiguring robotic molecule. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 424–431. IEEE, 1998.

Keith D Kotay and Daniela L Rus. Motion synthesis for the self-reconfiguring molecule. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, volume 2, pages 843–851. IEEE, 1998.

Keith D Kotay and Daniela 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)*, volume 3, pages 2184–2193. IEEE, 2000.

Haruhisa Kurokawa, Satoshi Murata, Eiichi Yoshida, Kohji Tomita, and Shigeru 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)*, volume 2, pages 860–865. IEEE, 1998.

Haruhisa Kurokawa, Kohji Tomita, Akiya Kamimura, Eiichi Yoshida, Shigeru Kokaji, and Satoshi Murata. Distributed self-reconfiguration control of modular robot m-tran. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 1, pages 254–259. IEEE, 2005.

Rus Robotics Laboratory. Crystal robot, 2000a. URL `https://groups.csail.mit.edu/drl/modular_robots/crystal/crystal.html`. Accessed: 2020-08-09.

Rus Robotics Laboratory. Molecule robot, 2000b. URL `https://groups.csail.mit.edu/drl/modular_robots/molecule/molecule.html`. Accessed: 2020-08-09.

Jens Liedke, Rene Matthias, Lutz Winkler, and Heinz Wörn. The collective self-reconfigurable modular organism (cosmo). In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1–6. IEEE, 2013.

Nathan Melenbrink, Paul Kassabian, Achim Menges, and Justin Werfel. Towards force-aware robot collectives for on-site construction. In *DISCIPLINES & DISRUPTION: Proceedings of the Annual Conference of the Association for Computer-Aided Design in Architecture (ACADIA)*, volume 37, pages 382–391. CumInCAD, 2017.

Nathan J Mlot, Craig Tovey, and David L Hu. Dynamics and shape of large fire ant rafts. *Communicative & integrative biology*, 5(6):590–597, 2012.

Satoshi Murata, Haruhisa Kurokawa, and Shigeru Kokaji. Self-assembling machine. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 441–448. IEEE, 1994.

Satoshi Murata, Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, and Shigeru Kokaji. A 3-d self-reconfigurable structure. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 432–439. IEEE, 1998.

Satoshi Murata, Eiichi Yoshida, Kohji Tomita, Haruhisa Kurokawa, Akiya Kamimura, and Shigeru Kokaji. Hardware design of modular robotic system. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, volume 3, pages 2210–2217. IEEE, 2000.

CA Nelson, Khoa Chu, and Prithviraj Dasgupta. Modred: a modular self-reconfigurable robot for autonomous extra-terrestrial exploration and discovery. In *IEEE Planetary Rovers Workshop, International Conference for Robotics and Automation (ICRA) 2010*, 2010.

Jonas Neubert and Hod Lipson. Soldercubes: a self-soldering self-reconfiguring modular robot system. *Autonomous Robots*, 40(1):139–158, 2016.

NVIDIA. Nvidia physx sdk, 2020. URL https://developer.nvidia.com/physx-sdk. Accessed: 2020-07-13.

Ian O'hara, James Paulos, Jay Davey, Nick Eckenstein, Neel Doshi, Tarik Tosun, Jonathan Greco, Jungwon Seo, Matt Turpin, Vijay Kumar, et al. Self-assembly of a swarm of autonomous boats into floating structures. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1240. IEEE, 2014.

Esben Hallundbæk Østergaard, Kristian Kassow, Richard Beck, and Henrik Hautop Lund. Design of the atron lattice-based self-reconfigurable robot. *Autonomous Robots*, 21(2):165–183, 2006.

Eckersley O'Callaghan. Mars habitat project, 2019. URL https://www.eocengineers.com/en/news/hassell--eoc-one-step-closer-to-life-on-mars. Accessed: 2020-07-14.

Amit Pamecha, Chih-Jung Chiang, David Stein, and Gregory Chirikjian. Design and implementation of metamorphic robots. In *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, volume 10. Irvine, California, USA: ASME, 1996.

Amit Pamecha, Imme Ebert-Uphoff, and Gregory S Chirikjian. Useful metrics for modular robot motion planning. *IEEE Transactions on Robotics and Automation*, 13(4):531–545, 1997.

John W Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295. IEEE, 2013.

Daniela Rus and Keith Kotay. Versatility for unknown worlds: Mobile sensors and self-reconfiguring robots. In *Field and Service Robotics*, pages 477–484. Springer, 1998.

Daniela Rus and Marsette Vona. Self-reconfiguration planning with compressible unit modules. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, volume 4, pages 2513–2520. IEEE, 1999.

David Saldana, Bruno Gabrich, Michael Whitzer, Amanda Prorok, Mario FM Campos, Mark Yim, and Vijay Kumar. A decentralized algorithm for assembling structures with modular robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2736–2743. IEEE, 2017.

Behnam Salemi, Mark Moll, and Wei-Min Shen. Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3636–3641. IEEE, 2006.

Jungwon Seo, Jamie Paik, and Mark Yim. Modular reconfigurable robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:63–88, 2019.

Wei-Min Shen, Yimin Lu, and Peter Will. Hormone-based control for self-reconfigurable robots. In *Proceedings of the fourth international conference on Autonomous agents*, pages 1–8, 2000.

Wei-Min Shen, Behnam Salemi, and Peter Will. Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. *IEEE transactions on Robotics and Automation*, 18(5):700–712, 2002.

Wei-Min Shen, Peter Will, and Berok Khoshnevis. Self-assembly in space via self-reconfigurable robots. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 2516–2521. IEEE, 2003.

Wei-Min Shen, Maks Krivokon, Harris Chiu, Jacob Everist, Michael Rubenstein, and Jagadesh Venkatesh. Multimode locomotion via superbot reconfigurable robots. *Autonomous Robots*, 20(2):165–177, 2006a.

Wei-Min Shen, Behnam Salemi, and Mark Moll. Modular, multifunctional and reconfigurable superbot for space applications. In *Space 2006*, page 7405. The American Institute of Aeronautics and Astronautics, 2006b.

Wei-Min Shen, Harris CH Chiu, Mike Rubenstein, and Behnam Salemi. Rolling and climbing by the multifunctional superbot reconfigurable robotic system. In *AIP Conference Proceedings*, volume 969, pages 839–848. American Institute of Physics, 2008.

Alexander Spröwitz, Soha Pouya, Stéphane Bonardi, Jesse Van Den Kieboom, Rico Möckel, Aude Billard, Pierre Dillenbourg, and Auke Jan Ijspeert. Roombots: reconfigurable robots for adaptive furniture. *IEEE Computational Intelligence Magazine*, 5(3):20–32, 2010.

K. Stoy, D. Brandt, and D. J. Christensen. *Self-reconfigurable robots: an introduction*. MIT press Cambridge, 2010.

John W Suh, Samuel B Homans, and Mark Yim. Telecubes: Mechanical design of a module for self-reconfigurable robotics. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 4095–4101. IEEE, 2002.

Guy Theraulaz, Eric Bonabeau, and Jean-Louis Deneubourg. The origin of nest complexity in social insects. *Complexity*, 3(6):15–25, 1998.

Michael T Tolley, Jonathan D Hiller, and Hod Lipson. Evolutionary design and assembly planning for stochastic modular robots. In *New Horizons in Evolutionary Robotics*, pages 211–225. Springer, 2011.

Thadeu Knychala Tucci, Benoit Piranda, and Julien Bourgeois. A distributed self-assembly planning algorithm for modular robots. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 550–558. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Cem Ünsal and Pradeep 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)*, volume 1, pages 598–605. IEEE, 2001.

Cem Unsal, Han Kiliccote, and Pradeep K Khosla. I (ces)-cubes: a modular self-reconfigurable bipartite robotic system. In *Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, pages 258–269. International Society for Optics and Photonics, 1999.

Cem Ünsal, Mark E Patton, Pradeep K Khosla, et al. Motion planning for a modular self-reconfiguring robotic system. In *Distributed Autonomous Robotic Systems 4*, pages 165–175. Springer, 2000.

ModLab UPenn. Self re-assembly after explosion, 2009. URL https://www.modlabupenn.org/2009/09/23/self-re-assembly-after-explosion/. Accessed: 2020-07-28.

Paul J White, Michael L Posner, and Mark Yim. Strength analysis of miniature folded right angle tetrahedron chain programmable matter. In *2010 IEEE International Conference on Robotics and Automation*, pages 2785–2790. IEEE, 2010.

Emily Jing Wei Whiting. *Design of structurally-sound masonry buildings using 3d static analysis*. PhD thesis, Massachusetts Institute of Technology, 2012.

Mark Yim. A reconfigurable modular robot with many modes of locomotion. In *Proc. of the 1993 JSME Conference on Advanced Mechatronics, Tokyo, Japan*, 1993.

Mark Yim. *Locomotion with a unit-modular reconfigurable robot*. PhD thesis, Stanford University Palo Alto, CA, 1994a.

Mark Yim. New locomotion gaits. In *Proceedings of the 1994 IEEE International conference on Robotics and Automation*, pages 2508–2514. IEEE, 1994b.

Mark Yim, David G Duff, and Kimon Roufas. Modular reconfigurable robots, an approach to urban search and rescue. In *the Proceedings of the 1st International Workshop onHuman-friendly Welfare Robotics Systems, Taejon, Korea*, 2000a.

Mark Yim, David G Duff, and Kimon 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)*, volume 1, pages 514–520. IEEE, 2000b.

Mark Yim, Ying Zhang, and David Duff. Modular robots. *IEEE Spectrum*, 39(2):30–34, 2002.

Mark Yim, Kimon Roufas, David Duff, Ying Zhang, Craig Eldershaw, and Sam Homans. Modular reconfigurable robots in space applications. *Autonomous Robots*, 14(2-3):225–237, 2003.

Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007a.

Mark Yim, Babak Shirmohammadi, Jimmy Sastra, Michael Park, Michael Dugan, and Camillo J Taylor. Towards robotic self-reassembly after explosion. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2767–2772. IEEE, 2007b.

Eiichi Yoshida, Satoshi Murata, Haruhisa Kurokawa, Kohji Tomita, and Shigeru Kokaji. A distributed reconfiguration method for 3d homogeneous structure. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, volume 2, pages 852–859. IEEE, 1998.

Eiichi Yoshida, Shigeru Kokaji, Satoshi Murata, Haruhisa Kurokawa, and Kohji 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)*, volume 3, pages 1579–1585. IEEE, 1999.

Eiichi Yoshida, Satoshi Murata, Akiya Kamimura, Kohji Tomita, Haruhisa Kurokawa, and Shigeru Kokaji. Reconfiguration planning for a self-assembling modular robot. In *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century.(Cat. No. 01TH8560)*, pages 276–281. IEEE, 2001.

Eiichi Yoshida, Satoshi Murata, Akiya Kamimura, Kohji Tomita, Haruhisa Kurokawa, and Shigeru Kokaji. A self-reconfigurable modular robot: Reconfiguration planning and experiments. *The International Journal of Robotics Research*, 21(10-11):903–915, 2002.

Jihong Zhu and Tong Gao. *Topology optimization in engineering structure design*. Elsevier, 2016.

Victor Zykov, Efstathios Mytilinaios, Mark Desnoyer, and Hod Lipson. Evolved and designed self-reproducing modular robotics. *IEEE Transactions on robotics*, 23(2):308–319, 2007.