PERCEPTION, CONTROL AND PLANNING FOR (SELF-)RECONFIGURABLE MODULAR ROBOTS

THIS IS A TEMPORARY TITLE PAGE
It will be replaced for the final print by a version
provided by the service academique.

Thèse n. 1234 2019
présenté le 20 Juin 2019
à la Faculté des Sciences de Base
laboratoire de Biorobotique (BIOROB)
programme doctoral EDRS
École Polytechnique Fédérale de Lausanne

Institute for Systems and Robotics
The Computer and Robot Vision Laboratory (VisLab)
programme doctoral RB-Cog
Instituto Superior Técnico in Lisbon

pour l'obtention du grade de Docteur ès Sciences
par

Mehmet Mutlu

acceptée sur proposition du jury:

Prof Dario Floreano, président du jury
Prof Auke Ijspeert, directeur de thèse
Prof Alexandre Bernardino, (co-)directeur de thèse
Prof Daniela Rus, rapporteur
Prof Michael Rubenstein, rapporteur
Prof Alcherio Martinoli, rapporteur

Lausanne, EPFL, 2019

E! Lunch?
— Certain Biorobian everyday at 11:30

To my parents and Ayşe Nur…

Acknowledgements

I would like express my utmost gratitude to Fundação para a Ciência e Tecnologia (FCT) agency of Ministry for Education and Science of Portugal for offering the PhD program *"Robotics, brain and cognition"* and the *"IST-EPFL"* joint degree which lead to this thesis. FCT (via the PhD grant PD/BD/105781/2014 and project grant UID/EEA/50009/2013) and EPFL financed most of this thesis. I would like to thank Swiss National Science Foundation and National Centre of Competence in Research for their financial support in various projects (SNSF-NCCR, project 153299) that lead to this thesis.

First of all I would like to thank both of my advisors Prof. Auke Ijspeert and Prof. Alexandre Bernardino for their limitless support, understanding and guidance. I particularly appreciate the right amount of freedom they both provided that let me explore many different ideas. I would like particularly highlight Prof. José Santos-Victor for managing to fit high quality supervision in super short time he has.

I also would like to thank to the members of this thesis committee Prof. Daniela Rus, Prof. Michael Rubenstein, Prof. Alcherio Martinoli and the president Prof. Dario Floreano for their time and willingness to participate in my defense.

A big hurray goes to Simon Hauser for shouldering Roombots with me since the very beginning until the very end. High quality time we had together not only in the lab, but also at every single opportunity such as conferences, travels, picnics and parties created a special bond that will last way longer than this PhD.

A special gratitude goes to Massimo Vespignani who introduced me to Roombots, lifted my spirits at hardest times and became my gateway to fun through skiing, drinking and fooling around.

Further, I would like to thank both Biorob family: Tomislav Horvat, Robin Thandiackal, Stéphane Bonardi, Peter Eckert, Behzad Bayat, Alessandro Crespi, Shravan Ramalindagetty, Johnathan Arreguit, Kamilo Melo, Laura Paez, Mostafa Ajallooeia, Hamed Razavi, Jessica Lanini, Luca Colasanto, Alexandre Tuleu, Florin Dzeladini, Amy Wu, Tadej Petric, Nicolas Van der Noot, Sylvie Fiaux, François Longchampl...;

and Vislab family: Ricardo Ribeiro, Atabak Dehban, Mihai Andries, Nino Cauli, Giovanni Saponaro, Moreno Coco,Plinio Moreno, João Avelino, João Martins, Pedro Vicente, Sofija Spasojević, Mirko Raković, Hugo Simão, Paul Schydlo, Rui Figueiredo, Prof. José Gaspar, Ana Santos, Ricardo Nunes...

I gratefully acknowledge the help of Ahmet Safa Öztürk, Stéphane Bussier, Quentin Golay and Théo-Tim Denisart for their contribution to the Roombots project. Special thanks to Jérémy

Blatter for the development of the GUI. Further, we are enormously thankful to François Longchamp for his technical support in the design of the modules. I would like to thank to Ayşe Nur Sunal and Olguta Robu for creating Play-Doh structures and providing alternative construction materials for Ch. 4. I would like to thank anonymous user study participants. I would like to thank all of my students who did a project with me: I. Youssef, H. Khodr, F. Efremov, M. Dujany, E. Klauser, R. Fong, H. Kohli, R. Vasconcelos, Pol Banzet, V. Nigolian, A. Häfliger, F. Freundler, A. Vardi, R. Dryzner, M. Moret.

Lausanne, 20 May 2019 M. M.

Abstract

Modular robots (MRs) consist of similar modules that can be configured into different shapes. MRs introduce a number of benefits over conventional robots specifically designed for a task. Self-reconfigurable modular robots (SRMRs) are a sub-category of MRs that can perform complex morphological changes by making use of autonomous attachments and detachments. This thesis is pointing out various aspects of MRs with particular interest in SRMRs (but not exclusively). First of all, designs of SRMRs are studied and improvements have been demonstrated in Roombots (RB), a SRMR designed in the Biorobotics Laboratory. The design improvements involve both electromechanical hardware, control and planning aspects. After all improvements, complex scenarios can be studied involving more than 10 modules adding up more than 30 DOF system. Such SRMRs are not intuitive for humans to operate. To overcome this issue, three different user interfaces (UIs) are proposed covering different aspects to simplify use of SRMRs. The first UI is used for conveying shape information to SRMRs using a tangible interaction interface as a proxy. The second one does not have any interaction medium and user directly control SRMRs via gestures that are tracked by external vision systems. The last UI shows possibility of assembling SRMRs in a virtual reality environment using a head mounted display and a hand gesture tracking system. All three of proposed UIs are novel and first of their kind for SRMRs. The final focus is locomotion with modular morphologies and presented in three major studies. Initially, one of the most simplest modular structures, a parallel chain structure, is studied to develop a goal driven neuro-visual locomotion model of a swimming lamprey robot. It is followed by a slightly more complex chain structure. A snake robot is simulated during different serpentine gaits to evaluate implications of different on-board camera placement strategies for such robots. In the last phase, the importance of compliance in legged locomotion is experimentally assessed for quadrupedal modular structures. Compliance of legs as well as a control scheme that acts as a compliant element is taken into consideration for a comparative evaluation. Locomotion is a fundamental part of all mobile robots. Three biologically inspired methods are offered to tackle locomotion of MRs that present performance improvements on multiple aspects as well as better understanding of certain concepts. Although the idea of modularity is widely used in industrial production and common daily items, MRs still don't find much use in daily lives. This thesis brings MRs one step closer to wide daily use by addressing such systems from multiple perspectives.

Zusammenfassung

Modulare Roboter (MR) bestehen aus ähnlichen Modulen, die in unterschiedliche Strukturen zusammengebaut werden können. MR besitzen einige Vorteile gegenüber konventionellen Robotern, die speziell für eine Aufgabe konzipiert wurden. Selbst-rekonfigurierbare modulare Roboter (SRMR) formen eine Unterkategorie der MRn, welche komplexe Modifikationen an ihrer Morphologie durch dynamische Verbindungen autonom vornehmen können. Diese These behandelt einige Aspekte von MRn mit Fokus auf SRMRn (aber nicht exklusiv). Als erstes werden verschiedene Designs von SRMRn untersucht, gefolgt von Verbesserungen an dem System Roombots (RB), einem SRMR des Biorobotics Laboratory. Diese Verbesserungen beinhalten elektromechanische Komponenten, Regelung und Bewegungsplanung, worauf komplexe Szenarien mit mehr als 10 Modulen und mehr als 30 Freiheitsgraden studiert werden konnten. Solche komplexen Strukturen erschweren eine intuitive Bedienung. Deshalb werden im zweiten Teil drei verschiedene Bedienoberflächen (UI) vorgeschlagen, welche die Bedienung von SRMR vereinfachen. Die erste UI verwendet ein greifbares Medium, um Forminformationen zu vermitteln. In der zweiten UI können Bediener SRMR direkt über Gesten kontrollieren, welche von einem externen System nachverfolgt werden. Im dritten UI wird die Möglichkeit präsentiert, SRMR mit Hilfe eines Head-Mounted-Display und Gestensystem virtuell zu Strukturen zusammenzubauen. Alle drei vorgeschlagene UI sind neu und erstmals in der Anwendung mit SRMRn. Der dritte Teil dreht sich um Fortbewegung mit modularen Morphologien und beinhaltet drei Studien. Zuerst wird eine einfach Kettenformation präsentiert, mit der ein neuro-visuelles Fortbewegungsmodell eines Neunauge studiert wird. Dann folgt eine etwas komplexere Struktur, wobei ein Schlangenroboter mit unterschiedlichen schlangenartigen Gangarten simuliert wird und die Auswirkungen auf die Platzierung von Kameras in solchen Robotern evaluiert wird. Als letztes wird die Bedeutung von Federelementen in vierbeiniger Fortbewegung experimentell untersucht. Federelemente in den Beinen sowie eine feder-simulierende Regelung werden miteinander verglichen. Fortbewegung ist eine fundamentale Funktion von allen mobilen Robotern. Drei biologisch-inspirierte Methoden der Fortbewegung von MRn werden diskutiert, welche mehrere Leistunsmerkmale verbessern und das allgemeine Verständnis der Fortbewegung fördern. Obwohl die Idee der Modularität in der Industrie sowie für alltägliche Objekte oft verwendet wird, haben MRn noch keine genau definierte Aufgaben im täglichen Leben. Diese These bringt MR einen Schritt näher zu diesem Ziel und betrachtet solche Systeme aus unterschiedlichen Winkeln.

Abstrato

(Portuguese translation of the abstract will be here)

Contents

Acknowledgements **v**

Abstract (English/Deutsch/Português) **vii**

List of figures **xvii**

List of tables **xxi**

1 Introduction **1**
- 1.1 Motivation 2
- 1.2 Challenges 3
- 1.3 Approach and thesis outline 4
- 1.4 Contributions 6

I Self-reconfigurable modular robots: hardware and control **7**

2 Roombots **9**
- 2.1 Introduction 10
- 2.2 Related work 13
- 2.3 Material and methods 15
  - 2.3.1 Mechanics, motors and connections 15
  - 2.3.2 Sensors 17
  - 2.3.3 Spotlight LED 18
  - 2.3.4 Universal gripper 18
  - 2.3.5 Electronics 19
  - 2.3.6 Motor control 20
  - 2.3.7 Connection control 21
  - 2.3.8 Graphical User Interface (GUI) 22
  - 2.3.9 RGB-D vision system 23
  - 2.3.10 Robotic-enhanced furniture 24
- 2.4 Experimental results 25
  - 2.4.1 Easy-to-use user interface: The GUI 26
  - 2.4.2 Scalable self-reconfiguration 27
  - 2.4.3 Mobile furniture 29

2.4.4 Manipulating furniture . . . 31
2.4.5 Interactive furniture . . . 34
2.5 Discussions and future work . . . 34
2.6 Conclusion . . . 37

3 An optimal planning framework to deploy SRMR 39
3.1 Introduction . . . 40
3.2 Terminology . . . 41
3.3 Problem formulation as search . . . 42
3.4 Searching for solutions . . . 44
3.4.1 Search algorithms: Expansion of the search tree . . . 45
3.4.2 Pruning methods . . . 46
3.4.3 Framework performance analysis . . . 48
3.5 Deploying the framework on hardware . . . 51
3.5.1 Addressing ACM alignment problem . . . 52
3.5.2 Approaching/leaving concave corners . . . 53
3.5.3 User feedback . . . 54
3.5.4 Physical limitations allowed in the framework . . . 54
3.6 Hardware validation . . . 54
3.7 Conclusions and future work . . . 55

II User interfaces for modular robots 57

4 From playdough to Roombots 59
4.1 Introduction . . . 60
4.2 Analysis of the problem . . . 61
4.2.1 Expression: Modeling clay . . . 61
4.2.2 Digitization: 3D scanning . . . 62
4.2.3 Abstraction: Voxels and Roombots . . . 62
4.2.4 Constraints and simplifications . . . 63
4.3 Solution method . . . 63
4.3.1 3D Scanning of playdough . . . 63
4.3.2 Pre-processing of Mesh . . . 64
4.3.3 Voxelization of the 3D model . . . 65
4.3.4 Search space definitions and initial module placement . . . 68
4.3.5 Construction with breath first search . . . 69
4.3.6 Construction with depth first search . . . 69
4.3.7 User Feedback . . . 72
4.4 Experimental results and discussions . . . 72
4.4.1 Effect of Initial Module Placement . . . 73
4.4.2 Different construction materials . . . 73
4.4.3 Hardware demonstrations . . . 74

4.4.4 Limitations of the proposed solution 74
4.5 Conclusions and future work 75

5 Natural user interface 77
5.1 Introduction and motivation 78
5.2 System design 80
5.2.1 Overview of the Proposed Human Robot Interface 81
5.2.2 Detecting User's Commands 82
5.2.3 Controlling Roombots 84
5.2.4 Closing the Control Loop 86
5.3 Results and Discussion 88
5.4 Conclusions and Future Work 88

6 Virtual reality for reconfigurable rooms 89
6.1 Introduction and motivation 90
6.2 Background on virtual reality 91
6.3 System design 92
6.3.1 Requirements of the System 93
6.3.2 Overall system 93
6.3.3 Giving more immersive feeling 95
6.3.4 Assembly 97
6.4 Results and discussions 98
6.4.1 User study 98
6.4.2 Discussions 99
6.5 Conclusions and future work 101

III Locomotion with modular morphologies 105

7 Neuro-visual locomotion model for Envirobot 107
7.1 Introduction 108
7.2 Methods 109
7.2.1 Envirobot Design 109
7.2.2 Frame-Based Object Detection 110
7.2.3 Event-Based Object Detection 110
7.2.4 Experimental Setup 111
7.2.5 Neuron Model 111
7.2.6 Neuron Network 111
7.3 Results 115
7.3.1 Behavioral Tests 115
7.3.2 DVS Performance 117
7.4 Discussion 119
7.4.1 Visual Memory 119
7.4.2 Behavior Adaptation & Learning 120

7.4.3 Underwater use of Event Based Cameras 121
7.5 Conclusion 121

8 Camera Placement on a Snake Robots 123
8.1 Introduction 124
8.2 Camera trajectory during locomotion 125
8.2.1 Rolling motion 128
8.2.2 Side-winding motion 129
8.2.3 Linear progression motion 130
8.3 Motion blur of camera during locomotion 131
8.3.1 Rolling motion 133
8.3.2 Side-winding motion 134
8.3.3 Linear progression motion 137
8.4 Discussions and demonstrations 138
8.5 Conclusions 139

9 Locomotion: effects of local passive and active compliance 141
9.1 Introduction 142
9.2 Methods 144
9.2.1 Hardware platform 144
9.2.2 Gait generation 146
9.2.3 Foot trajectory 146
9.2.4 Open loop CPG-based control 146
9.2.5 Tegotae-based control 148
9.3 Experiments 149
9.3.1 Passive compliance 149
9.3.2 Tegotae-based active compliance 150
9.3.3 Experiment terrain 150
9.4 Results and Discussions 151
9.4.1 Performance metrics 151
9.4.2 Overall results 156
9.4.3 Feet trajectories 158
9.4.4 Phase vs compliance in Tegotae-based control 159
9.4.5 Emerging gaits from Tegotae-based control 160
9.4.6 Limitations of the platform: 160
9.5 Conclusions and future work 161

10 Conclusions 163
10.1 Future work 167

Appendix 169

A Cooperative bridge building by SRMR based on ants' stigmergic behaviour 169
A.1 Introduction . . . 170
A.2 Methodology . . . 171
A.2.1 An algorithm from ants . . . 171
A.2.2 Roombot's mechanics . . . 171
A.2.3 Logic-level simulation . . . 172
A.2.4 Experimental set-up . . . 172
A.3 Results . . . 172
A.4 Conclusions . . . 172

B Leg touchdown patterns of the modular quadruped 175

Bibliography 195

Curriculum Vitae 197

List of Figures

1.1 Microbots from the Disney animation Big Hero 6 3
1.2 Thesis outline 6

2.1 Roombots framework with the five tasks indicated 12
2.2 Design of a single Roombots module 16
2.3 Modifications of the connection plates 17
2.4 Integration of a Universal Gripper into a Roombots hemisphere 20
2.5 Roombots electronics 21
2.6 Pitch angle compensation 22
2.7 GUI of Roombots 24
2.8 RGB-D camera and furniture used in the sub-tasks 25
2.9 Circle walk demonstration 27
2.10 Chair formation sequence 28
2.11 Chair following a user 29
2.12 Assistive furniture opens up space 30
2.13 Active slope compensation 31
2.14 Overcoming a ledge 32
2.15 Table picking up an object 33
2.16 Two RB gripper modules passing a pen in mid-air 34
2.17 A gripper metamodule assisting in opening a PET water bottle 35
2.18 Interaction with tracking and lighting 36

3.1 Caption short description for table of contents 42
3.2 Flowchart for a general search algorithm 44
3.3 Illustration of Roombots neighborhoods 47
3.4 Goal state of the two test cases for algorithm comparison. 48
3.5 Comparison of number of problems solved within 30 minutes for (a) gap and (b) ceiling test cases. 49
3.6 Description of the ACM and passive plate components 52
3.7 Example showing the parallel behavior when approaching a concave corner . . 54
3.8 Different stages from the experiment of crossing a gap using 3 RB modules. . . 55

4.1 The cycle from an inspiration to the final Roombots shape 61
4.2 The inspirational LEGO® advertisement 63

4.3 Initial and pre-processed view 65
4.4 High and low resolution voxel space 68
4.5 Initial module placement 69
4.6 Six structures 70
4.7 Construction of modular structures with DFS. 71
4.8 Artificial feedback 72
4.9 Different objects used for benchmarking and the corresponding found Roombots structures 74
4.10 Best two BFS results 75
4.11 The stool example 75

5.1 Rendered picture representing the different aspects of the Roombots project . . 79
5.2 Roombots are being designed to work in daily life environments 80
5.3 Overview of proposed user interface 82
5.4 Experimental robotic spotlight system 83
5.5 Control system 84
5.6 Adjusting brightness of the light 84
5.7 Real time information of the GUI 85
5.8 Kinematic model of Roombots metamodule manipulator 86

6.1 A single Roombots module 90
6.2 Main components of the GUI in the Workshop 95
6.3 Users can place their custom structures anywhere in the Room 96
6.4 The complete set-up required by the system during a user test 97
6.5 A Roombots module in current selection 98
6.6 Loop formation 99
6.7 Operators were asked to perform four successive tasks 100
6.8 The subjective opinions of users 102
6.9 Throne 103

7.1 The Envirobot target bulb and overhead tracking LEDs 110
7.2 8-neuron wide network of the module consisting of an input layer, a response layer, and an auxiliary layer 113
7.3 The architecture of the behavior arbitration sub-network 114
7.4 The sub-network used to produce the system's output 116
7.5 The experimental Envirobot swimming trajectories in four distinct scenarios . 117
7.6 A comparison between the neuron network computation rates achieved by the two types of image capture 118
7.7 An example of trajectories for both vision systems recorded during the experiments119
7.8 A performance comparison between event-based and frame-based cameras . . 120

8.1 Modular Snake Robot Lola-OP 126
8.2 Camera trajectory (on head, straight) during rolling 128

8.3 Camera trajectory (in the middle, sideways) during rolling 129
8.4 Camera trajectory (on head, straight) during sidewinding 130
8.5 Camera trajectory (in the middle, sideways) during sidewinding 130
8.6 Camera trajectory (on head, straight) during linear progression 131
8.7 Camera trajectory (in the middle, sideways) during linear progression 132
8.8 w and μ of camera during rolling (on head, straight) 134
8.9 w and μ of camera during rolling (in the middle, sideways) 135
8.10 w and μ of camera during sidewinding (on head, straight) 135
8.11 w and μ of camera during sidewinding (in the middle, sideways) 136
8.12 w and μ of camera during sidewinding (in the middle, upwards) 136
8.13 w and μ of camera during linear progression (on head, straight) 137
8.14 w and μ of camera during linear progression (in the middle, sideways) 138
8.15 w and μ of camera during linear progression (in the middle, upwards) 138
8.16 Captured images during rolling (on head, straight) 139
8.17 Captured images during sidewinding (on head, straight) 139
8.18 Captured images during linear progression (on head, straight) 140

9.1 Quadruped robot 144
9.2 Three different elements 145
9.3 The designed foot trajectory 147
9.4 Motion based motion blur metric MMBM 155
9.5 Evaluation of the proposed metrics 157
9.6 Ground reaction force and the orientation of the force 158
9.7 Foot trajectories during locomotion on the flat surface 159
9.8 Phase evaluation of fore and hind right limbs 160

A.1 Roombots agent as an ant 170
A.2 Pipeline of the distributed control of RB based on ants 173

B.1 Leg touchdown patterns when all element are rigid 176
B.2 Leg touchdown patterns when all element are hard 176
B.3 Leg touchdown patterns when all element are soft 177
B.4 Leg touchdown patterns when fore limb elements are hard and hind limb elements are rigid 177
B.5 Leg touchdown patterns when fore limb elements are soft and hind limb elements are rigid 178

List of Tables

2.1 The video is available as supplementary material for this article (see link) 25

3.1 Average convergence time of tested algorithms in [s] on both gap and ceiling test cases 50
3.2 Average number of expanded nodes of the tested algorithms on both gap and ceiling test cases 50
3.3 Optimality percentage of tested algorithms on both gap and ceiling experiments 51
3.4 Completeness percentage of both gap and ceiling test cases when history list does not have orientation. 51

4.1 Comparative performance of the suggested methods 73

6.1 Subjective evaluation of the proposed interface 101

8.1 Parameters and Coordinate Transformations of Camera Placement for Examined Gaits 127
8.2 MMBM statistics for different snake gaits 139

1 Introduction

Traditional objects are often made for certain use and do not exhibit modularity. They have a fixed morphology and perform a certain function. When they do not function as expected, they need to be either replaced or repaired which usually takes significant amount of time. A *can opener*, a *shirt* or *bottle* are some examples for such objects.

Modularity is the capability of a system that allows its parts to be combined in different ways to enhance its flexibility in use. It is a very generic term that appears in many fields such as programming, industry, art, science and even daily life. For different uses, the exact meaning slightly changes. For example, a modular software refers to having blocks of code that can be maintained separately and easily merged with the rest. A modular construction refers to fabrication of smaller blocks prior to the assembly and capability of assembling them in different ways. According to [Salvador et al., 2002], there are two main criteria to classify the modularity of a product: (1) existence of a main body (a central unit) connecting modular pieces (or alternatively no main body where modular pieces attach to each other to form the system) and (2) diversity of interface (either single interface or multiple interface types to connect to each other). Combinations of those criteria results in four types of modularity:

1. A system with diverse interface and main body is said to have *component swapping modularity*: Each modular part can be only replaced with a new one. E.g., custom-size battery of specific device.
2. A system with identical interface and main body is said to have *bus modularity*: Different modules can be connected to the same bus and modules are swappable with each other. E.g., universal serial bus (USB) devices being modules and a PC being the main body.
3. A system with identical interface and no main body is said to have *sectional modularity*: Pieces can be combined in any possible way (such as Lego blocks). E.g. Lego bricks.
4. A system with diverse interface and no main body is said to have *combinatorial modularity*: Main components can be different, but they are connected in only one way. E.g. a car (AC unit, steering unit, multimedia unit etc. are assembled in a single way).

Although, an example has been given for each category, they are not limited to single application domain and represent much more generic concepts to be used in different application areas such as mechanics, electronics etc..

A modular robots (MRs) can be built into different morphologies that enables different capabilities [Gilpin and Rus, 2010]. In the modular robotics field, the commonly meant modularities are the *sectional modularity* (independent modules) and the *bus modularity* (modules relying on central computation, power etc.). In other words, the system is mostly expected to be consist of identical modules. However, there are modular robots with heterogenous modules too. Heterogeneity can be in only the functional level (all modules having the same shape, but, different functions) or also in shape. Modular robots can be classified in numerous different ways. [Ahmadzadeh et al., 2015] gives a systematic classification of an extensive list of MRs by considering *module*, *information*, *task* and *environment* (MITE). A specific group of MRs can self reconfigure into different morphologies, thus called **self-reconfigurable modular robots (SRMRs)**. Self reconfiguration adds further autonomy to MRs and usually comes with the increase in the complexity of modules.

1.1 Motivation

SRMRs have various advantages over traditional fixed morphology robots. [Yim et al., 2007] summarizes them in three main points: Versatility, robustness and low cost. The most prominent capability is the shape changing. A SRMR ideally can create any arbitrary shape. Thus, they can adapt to changing demands in time. Having standardized building modules is also preferred for manufacturing reasons. Because mass manufacturing a single type of building block is expected to be cheaper and faster than custom designing dozens of task specific robots. Moreover, replacing a broken module is much easier than detecting and repairing a problem in a big system. SRMRs can in principle self-repair through detecting broken modules and replacing them autonomously.

Ideally, many tasks should be done with a SRMR system. Space applications stand out among all of the possibilities, because the cost of bringing a tool to orbit or deeper space is tremendously high. Another promising application area is intelligent living environments. SRMRs can be used to augment environments we live in by creating furniture, manipulating existing objects and aiding humans. Additionally, SRMRs have an artistic perspective. They can serve as construction blocks for artists to create new interactive experiences.

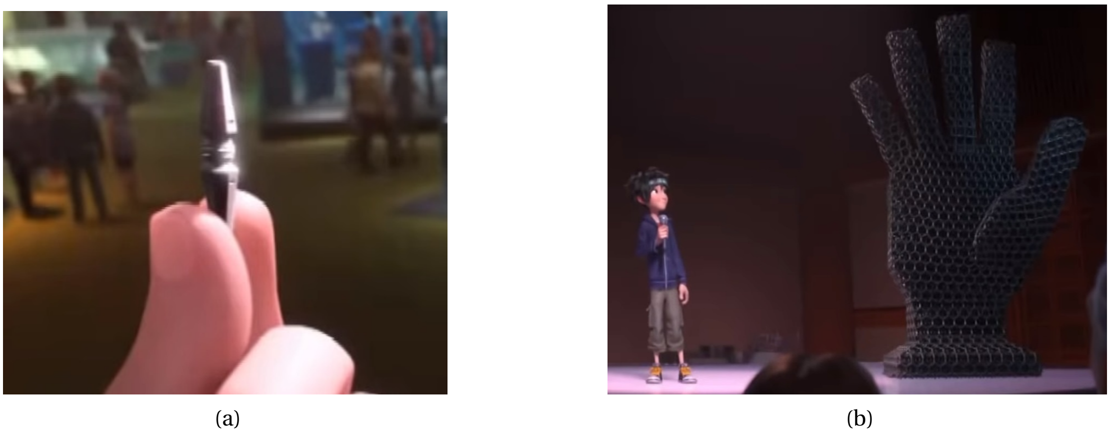
High motion capabilities thanks to the large number of joints is inherently present in SRMRs. It is not intuitive to control such highly redundant systems. Furthermore there is an emerging need for human-robot interaction whenever those robots are used in human presence, which leads to the inevitable question *"How humans can control SRMRs?"*. User interfaces for SRMRs have merely been addressed in the literature thoroughly so far.

Animals have been evolved in millions of years through breeding among specimens, mutations and *survival of the fittest.* Indeed, animals display spectacular characteristics that let them survive to this date. There are various studies focusing on behaviour adaptation in case of loosing a limb such as [Cully et al., 2015]. Further morphological changes in biology can be observed during growth or more severely during metamorphosis. More complex challenges such as morphological additions or complete transformations (like more complex metamorphosis) become available through SRMRs. Hence, they open new directions to study imaginary animals in laboratory environment.

The Biorobotics Laboratory (Biorob) and Computer and Robot Vision Laboratory (VisLab) (two research laboratories I have been affiliated throughout my PhD program *"Robotics, brain and cognition"*) are particularly interested in biologically inspired systems. While Biorob has the major focus in animal locomotion, VisLab stands out in vision and cognition (learning) studies. In real life, animals can display all three of these features in parallel. Being part of an international and multidisciplinary PhD program, this thesis will act as a bridge between conceptually quite different (bio-inspired) robotics studies.

1.2 Challenges

An ultimate robot would be a tiny SRMR which can create any shape and perform any task similar to what is depicted in *"Big Hero 6"*, an animation movie by Disney Studios. Fig. 1.1 shows two scenes where a single Microbot is introduced and its capabilities are shown as a swarm. Naturally, we are bounded by physical limitations and currently available technology. Actuator power to module weight ratio, battery life and material strength are among the most pronounced limitations. Thus, very large scale systems (i.e. 1000+ modules) have still not been realized. The closest are Kilobots [Rubenstein et al., 2012], but those are self assembling robots, not SRMRs. In order to achieve such scales, power management, power sharing and energy harvesting capabilities should be iteratively revisited.



(a)

(b)

Figure 1.1 – Microbots from the Disney animation Big Hero 6: (a) Tiny microbot modules (b) can create any large structure using as many modules as needed.

Current state of the art presents that SRMRs are still inferior to individually designed custom robots for achieving a certain single task. It may as well remain that way due to the definition of the system. In other words, the power of SRMRs comes from the adaptability to different tasks whereas a task-specific design is optimized for the performance on that given task. Nevertheless, the goal is to increase overall performance of SRMRs through both having more diverse task capabilities and improving individual task performances. Modular designs bring extra complexity over specific designs. They may as well be more fragile and expensive.

Sensorization still remains to be one of the biggest challenges in SRMRs. With rapid technological advancements, sensor diversity is increasing while size and cost of existing sensors decrease monotonously. Hence, the number of sensors in our daily lives constantly increase. Nowadays, even a mobile phone comes with dozens of sensors. A good portion of sensors which have found their way into SRMRs are proprioceptive to measure internal states such as motor encoders, battery monitors, accelerometers and gyroscopes. There are also very useful and common exteroceptive sensors such as thermometers, barometers and cameras that are used in SRMRs as well. However, sensing the surrounding objects still (similar to skin) remains to be a challenge especially for self-reconfiguration in 3D. Most of the presented SRMRs in the literature rely on knowing initial states and executing set of actions in open loop fashion. There exits proximity sensors like infrared or ultrasound sensors. But, a large number of them are needed to cover all directions in 3D self-reconfiguration. Planar and 2D scenarios are easier as surrounding the robot with an array of sensors is sufficient. Another popular approach is the use of cameras. However, the use of cameras also require integration of a powerful computation unit to process images. The size, power and cost requirements of SRMR increase proportionally. Existing SRMR solutions do not have truly integrated cameras and powerful computation in all modules, but, mostly keep a camera as a separate passive (non-actuated) module [Barrios et al., 2016], [Zhu et al., 2014], [Murata et al., 2007].

In robotics, modularity has multiple perspectives as it involves the hardware morphology as well as control behind it. SRMRs are high DOF systems. Such high dimensionality can be seen as a blessing since it provides functional flexibility, or at the same time, a curse as the control can get quite complex. High dimensionality of SRMRs still remains as a big challenge to develop control and planning algorithms. Developments on scalable control solutions as well as massively scalable wireless communication systems (in a compact volume) are highly in demand to bring proof of concept systems into real world applications.

1.3 Approach and thesis outline

This thesis focuses on the development of electromechanical hardware and control strategies for (SR)MRs and addresses biologically inspired locomotion models on MRs. The objective of this thesis is trying to answer following research questions:

1. How far can we go on the path of creating the perfect SRMR that can tackle every possible

task?

2. What kind of user interfaces are suitable for using SRMRs in daily lives?
3. How closely can we recreate animal locomotion in laboratories using (SR)MRs?
4. How does the visual system correlate with bio-inspired locomotion of modular robots?
5. What is the role of compliance in bio-inspired locomotion of modular robots?

Answers to these questions are presented in three main parts. The first two questions are addressed in the first two parts while the rest of the questions are collected in the last part. Fig. 1.2 illustrates the visual structure of the thesis. The key concepts binding chapters together are shown on the figure. Each chapter and part are illustrated with blue and green backgrounds respectively in Fig. 1.2. The main similarity of Part I and Part II is the use of Roombots (RB) (a SRMR made in Biorob) as the application platform whereas locomotion studies (Part III) use only modular robots (not self-reconfigurable).

Part I describes the improvements on electromechanical structure of Roombots and complementary control & planning framework.

Part II introduces three novel user interfaces for SRMRs using three different approaches: a tangible, a gesture-based and a virtual reality interface.

Part III studies three different topics on three different locomotion modes to answer respectively the last three of research questions: anguilliform swimming, terrestrial serpentine and legged quadrupedal locomotion. The swimming study had to be conducted on a waterproof robotics platform Envirobot while the others exploited a simpler and less complex modular platform (Robotis Bioloid kit).

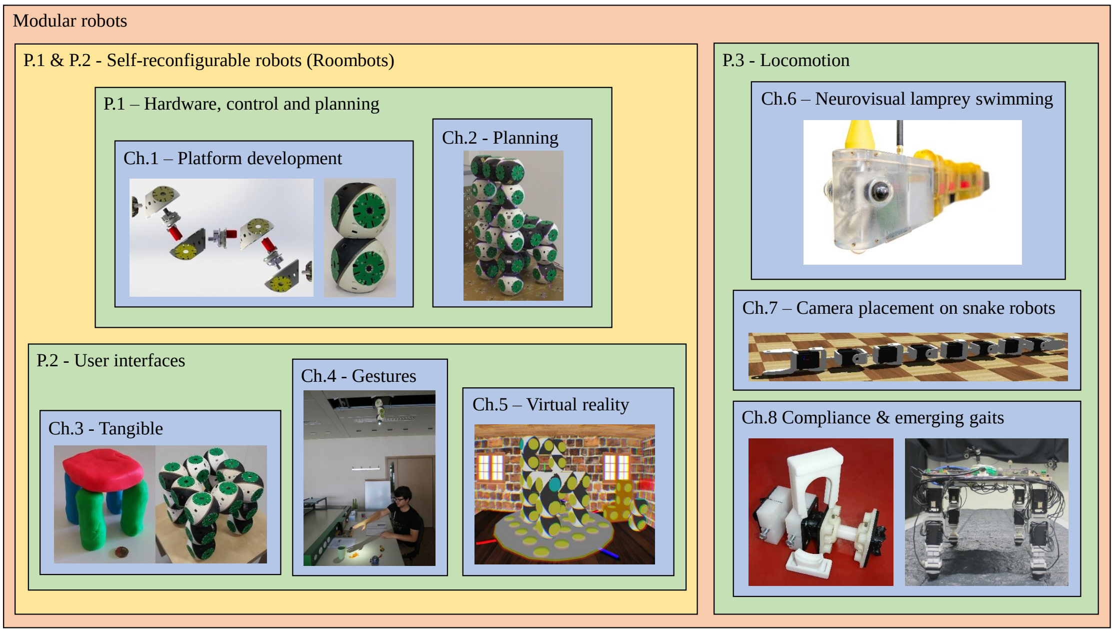

Modular robots
P.1 & P.2 - Self-reconfigurable robots (Roombots)
P.1 – Hardware, control and planning
Ch.1 – Platform development
Ch.2 - Planning
P.2 - User interfaces
Ch.3 - Tangible
Ch.4 - Gestures
Ch.5 – Virtual reality
P.3 - Locomotion
Ch.6 – Neurovisual lamprey swimming
Ch.7 – Camera placement on snake robots
Ch.8 Compliance & emerging gaits

Figure 1.2 – Thesis outline.

1.4 Contributions

The main contributions of the thesis are:

- Hardware development of Roombots by upgrading existing features and adding new ones.
- Developing control strategies and an autonomous planning framework for self-reconfiguration.
- Developing three novel user interfaces for SRMRs.
- Implementing and testing a biologically inspired neurovisual locomotion model for robotic lamprey using event-based cameras.
- Analyzing the relation between serpentine locomotion and visual information.
- Analyzing the effects of compliance in quadrupedal locomotion.

Part I

Self-reconfigurable modular robots: hardware and control

2 Roombots

The first chapter introduces Roombots (RB) that is a SRMR created in the Biorobotics Laboratory. A particular focus of this chapter is to explain improvements and upgrades achieved since October 2014 (the beginning of this thesis). Improvements involve both hardware changes and software/control developments. For the first time, we had a significant number of modules (13) to showcase larger SRMR systems since the beginning of the Roombots project. This chapter serves as the logbook of all the tiny improvements that paved the way towards realization of a scalable SRMR. The platform (Roombots) developed in this chapter forms a basis for the Part I and Part II as all of the SRMR work developed in this thesis uses Roombots for hardware demonstrations.

Reference publications

- This chapter is based on *Simon Hauser, **Mehmet Mutlu**, Pierre Alexandre Léziart, Hala Khodr, Alexandre Bernardino and Auke Ijspeert. "Roombots extended: challenges in the next generationof self-reconfigurable modular robots and their application in adaptive and assistive furniture."* Submitted to Robotics and Autonomous Systems.

  My original contributions

  - Building the required electronics (ACM sensor plate, ACM sensor readout board, universal gripper controller board, LED ring board, spotlight board)
  - Co-supervising Jérémy Blatter for the GUI development
  - Primarily supervising Hala Khodr, Ahmet Safa Öztürk, Stéphane Bussier and co-supervising Quentin Golay and Théo-Tim Denisart who did projects on different aspects of the Roombots project
  - Guiding and conducting planning experiments
  - Partially writing and fully reviewing the manuscript

2.1 Introduction

Self-reconfigurable modular robots (SRMRs) pose interesting interdisciplinary challenges in hardware, software and their application for an end user.

The hardware is an engineering challenge. The general idea of modular robotics is to combine "simple" modules into larger, more complex and thus usually more capable structures that possess the required functionality of a task at hand. A true reversible modular robotic system (MR) has three key functionalities: reversible connection between two modules, reversible disconnection between two modules, and actuation (either within one module or in a structure as a whole). Note that reversible connection does not automatically include disconnection as there can be different mechanical systems for both connection and disconnection. For the design of each of these systems, the overall dimensions of a module should be defined, which often are linked to the expected tasks. At last, it has to be considered if all the sub-units are exact copies of each other (homogeneous MR) or if specialized units collaborate together (heterogeneous MR) [Ahmadzadeh et al., 2015]. In manually reconfigurable modular robot systems [Ahmadzadeh et al., 2015], the connection and disconnection functionalities are outsourced to a human such that an MR mainly has to care only about the actuation system. This means that each module can take on almost any form with any mechanical properties, particularly also soft modules. In contrast, SRMR [Ahmadzadeh et al., 2015] systems have to perform the connection and disconnection functionalities autonomously, which introduces additional constraints on the rigidity: to enable or at the very least simplify the self-reconfiguration, each module should be structurally rigid such that some form of alignment between modules is achievable - a usually necessary condition for self-connection - yet provide some form of movement to initiate self-reconfiguration. The balance between these two properties is already challenging to achieve in 2D SRMR systems and add a significant level of difficulty in 3D SRMR systems.

The software requires algorithms to deal with decentralized systems. It needs to be defined if one module acts as a master commanding all other modules in the system, if each module has the same amount of control or if there is a main external controller unit. Additionally, a way for a human to intuitively interact with one or more groups of modules has to be developed such that the current state of the modular system can be understood and a future state can be communicated.

At last, one has to consider for which application the robotic platform is to be developed. Even though SRMR can have the ability to shape-shift into various morphologies, this does not necessarily mean that one module should be as small as possible build any arbitrary morphology; one module should be as small as the smallest unit required by the application. It also needs to be defined what should be the added value of the ability to self-reconfigure to justify the major development of the challenges mentioned above: what is the usage of the self-reconfiguration?

As one example that aims at tackling some of these challenges, over the last decade we developed “Roombots”(RB), a 3D self-reconfigurable modular robot system with the application - among others - of serving as adaptive and assistive furniture. In this framework, multiple groups of modules adapt to the needs of a user by creating different pieces of furniture or augmenting the capabilities of existing furniture. Additionally, such pieces of furniture can also act somewhat autonomously and provide assistance specifically to elderly by e.g. preventing falls thanks to a following chair, picking up and holding objects and bringing items while retaining their functionality as pieces of furniture.

The first generation of Roombots proposed the general hardware design of our SRMR together with algorithmic work on reconfiguration and locomotion and works in user interfaces [Sproewitz et al., 2008], [Sproewitz et al., 2009], [Sproewitz et al., 2010b], [Sproewitz, 2010], [Sproewitz et al., 2 [Sproewitz et al., 2013], [Sproewitz et al., 2014], [Bonardi et al., 2012b], [Bonardi et al., 2012a], [Bonardi et al., 2013], [Bonardi et al., 2014], [Vespignani et al., 2013], [Ozgur et al., 2014]. As a proof of concept, only a limited number of modules was needed to demonstrate the three key functionalities: connection, disconnection and movement. However, to go further towards the vision of using Roombots for adaptive furniture, more modules were needed and a few adaptations in hardware and software had to be made based on the experience with the developed prototypes.

To showcase some of the new capabilities of the upgraded Roombots system, we propose five new tasks that we regard as core functionalities of adaptive and assistive furniture: self-reconfiguration, mobility, manipulation, human-module-interaction, and user interface. Each of these tasks will be demonstrated by Roombots, often through various sub-tasks (see Fig. 2.1).

The first task is scalable *self-reconfiguration* which stands at the core of an SRMR system and fits well into the vision of creating adaptive furniture. For this demonstration, a disconnected group of RB modules self-reconfigures into a shape resembling that of a common piece of furniture. It should be noted that while this task essentially reduces to a series of connection, disconnection and movement actions, which already have been demonstrated in e.g. [Sproewitz et al., 2014], performing it with a larger number of modules is not trivial due to the scalability of the robotic system: continuous reliable reconfiguration with a group of modules rather than an isolated reconfiguration with two modules adds significant challenges in hardware and software to deal with real-world physical phenomenons that only occur for larger groups of modules. In the case of Roombots, the first hardware iteration experienced major elastic deformations and misalignment issues such that the autonomous formation of larger structures was not possible. Due to the various hardware modifications presented in section 2.3.1 and closed-loop control described in section 2.3.2, such tasks lie now within or at least much closer to the capabilities of the present hardware iteration.

Second, adaptive furniture should possess a certain degree of *mobility* such that pieces of furniture can dynamically move within a living space. Depending on the functionality of the structure, mobility can require certain adaptation of the furniture to the environment as well

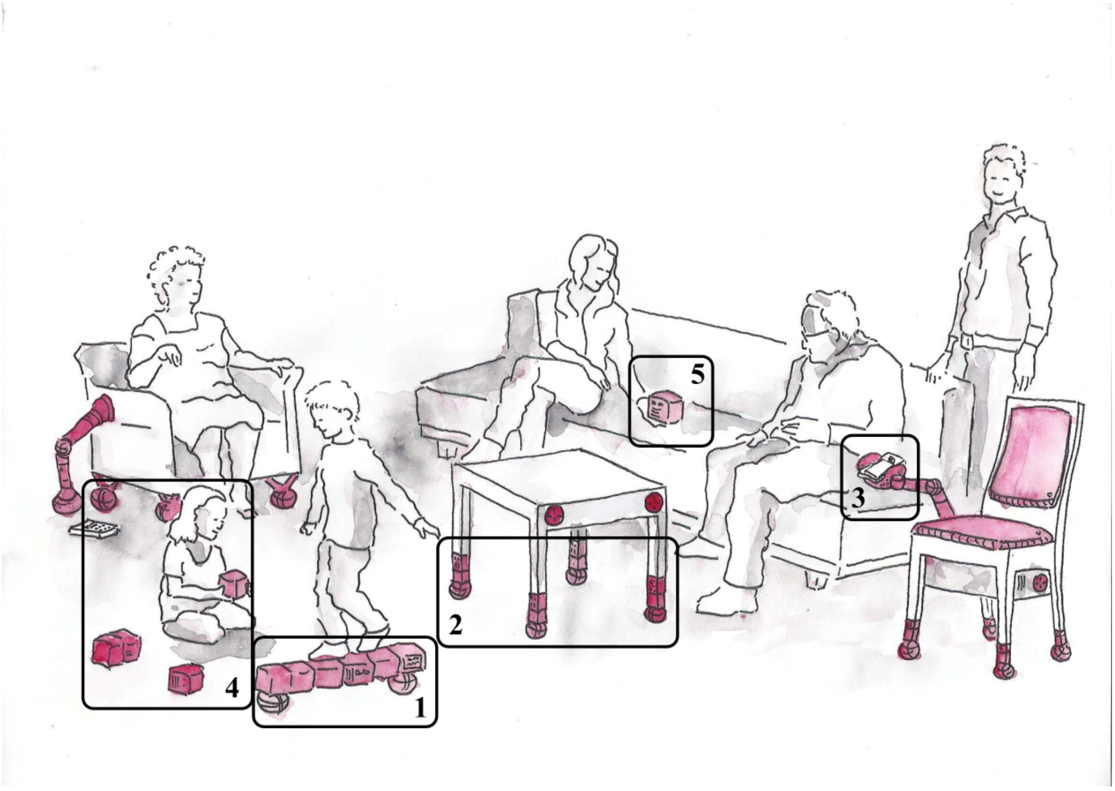


Figure 2.1 – Roombots framework with the five tasks indicated. 1) Self-reconfiguration of a group of Roombots modules into a shape to function as a toy. 2) Mobile furniture where an existing piece of furniture has been enhanced with RB modules to enable it to move around in the living space. 3) Object manipulation where furniture is able to assist in simple tasks such as picking up and holding a remote control. 4) Interactive furniture, allowing users to work together with robotic-enhanced furniture, and 5) an easy-to-use user interface (UI) to monitor the current state of the robotic system.

as autonomy to react to changes in the environment. We performed a set of demonstrations to showcase the mobility aspect, ranging from following and evading pieces of furniture to slope compensation and even stair climbing.

To fulfill more of the assistive aspects of the Roombots vision, robotic-enhanced furniture must possess a way to *manipulate objects*. Simple actions such as holding a book can already provide assistance, and it is clear that many of such actions require a method for manipulation, i.e. a gripper. Such a gripper can, for instance, be used for fetching remote controller fallen on the ground. For the integration of a gripper into the RB framework - which should be able to manipulate a wide range of everyday objects - we revisited the concept of granular jamming that was used to develop a "Universal Gripper". By miniaturizing the mechatronic components, we were able to equip selected modules with this gripper and demonstrate common tasks such as a piece of furniture picking up an object from the ground and an RB structure helping to open a water bottle. Two gripper modules further are able to pass objects from one module to another, which will also be briefly discussed.

Users need an intuitive way of *interacting with modules*. Interaction could be through e.g. speech recognition where modules react to spoken commands, or through gesture control as e.g. in [Ozgur et al., 2014, Mutlu et al., 2016, Nigolian et al., 2017] where commands are created through the tracking of gestures. Further, modules should provide a feedback to the environment such that the current state of a single module or a group of modules can easily be understood. A simple way to achieve this is through light where different colors can be assigned to different states. We revisited our approaches in the past and demonstrate again the interaction capabilities of RB.

At last, when the complete information about the state of the modules is required, an appropriate *visualization tool* is needed. We describe the development of a Graphical User Interface (GUI) that allows for an easy control of RB modules. The GUI greatly facilitated the development of many of the presented demonstrations and will simplify the development of future challenges. Such challenges (e.g. RB metamodule locomotion on-grid) are identified along with this work and analyzed and discussed in section 2.5.

The manuscript is structured as follows. It first details the hardware and software modifications of the newest generation of Roombots and then reports significant demonstrations in hardware of selected scenarios. It then concludes with the limitations of the current platform and with a general discussion of challenges of self-reconfigurable modular robot systems.

2.2 Related work

Over the years, a large variety of 2D and 3D proof-of-concept SRMR systems has been developed. A comprehensive list of both mechanical designs and algorithms of SRMR can be found in [Ahmadzadeh et al., 2015] and [Ahmadzadeh and Masehian, 2015]. If a research includes experiments with prototype hardware modules, the basic functionalities *connection* and *disconnection* are in many cases shown with a minimal number of modules, often in an isolated experiment involving two to five modules. Larger reconfigurations with more modules are usually shown in simulation, whereas structures in hardware experiments with more modules (e.g. some form of collaborative behavior) are mostly manually assembled, e.g. PolyBot [Yim et al., 2000], SMORES [Davey et al., 2012], UBot [Cui et al., 2012], SUPERBOT [Salemi et al., 2006], CoSMO [Liedke et al., 2013], 3D M-Blocks [Romanishin et al., 2015], Soldercubes [Neubert and Lipson, 2015] and AMAS [Terada and Murata, 2008].

With Roombots, the focus is not only task execution with prepared structures, but also the actual formation process in 3D. While some of the systems in [Ahmadzadeh et al., 2015] and [Ahmadzadeh and Masehian, 2015] certainly will mature further and with additional development will be able to demonstrate larger reconfigurations in hardware, most (if not all) SRMR systems have difficulties when the number of modules is increased, especially for 3D systems. Because of significant hardware challenges, only a limited number of such SRMR systems are able to perform 3D self-reconfiguration with more than five modules where all modules are involved in the self-reconfiguration, allowing for a more comprehensive understanding of the potential of reconfigurable systems.

As Roombots are thought to work in groups that can involve up to tens of modules (task 1), we put a closer look at the largest structures formed through self-reconfiguration with 3D SRMRs, and could only find two examples where the formation (or reconfiguration) is explicitly stated in literature: (i) M-TRAN III [Kurokawa et al., 2008] formed a mesh structure with 24 modules (48 DoF) that locomotes by disconnecting modules from one side of the mesh and reattaching them on the other side, and (ii) ATRON presents a self-reconfiguration sequence with 3 three-units meta-modules in [Brandt et al., 2007], resulting in 9 modules (9 DoF). We aim to show that Roombots is one of the few systems where 3D reconfiguration with 10 modules (30 DoF) and more is possible.

The challenge doesn't stop once a structure is formed. The dynamic shape-shifting capabilities of SRMRs raises the question of how humans can interact with such complex robot systems [Weller et al., 2011]. Vision is one of the most powerful sense that enables accurate remote sensing and allows for interaction. Particularly the sense of depth plays a crucial role in almost all of our fundamental sub-tasks such as grasping and locomotion. Similar to humans, robotics systems can benefit from vision too. One of the easiest way of getting the depth map information is using RGB-D cameras. There exists numerous ways to utilize RGB-D cameras for human tracking [Camplani et al., 2017, Han et al., 2012, Liu et al., 2015], detecting gestures [Ramey et al., 2011] as well as other human motion [Ye et al., 2013] or learning object affordances [Koppula et al., 2013]. Modular robots can benefit from the vision capability as well, although the integration of a vision system into a modular robot system is not straightforward. There are several possibilities (e.g. integrated in a module, a specialized module, an external system), each with their own advantages and disadvantages and engineering difficulties. A recent use of RGB-D sensor in the scope of SRMR has been reported in [Tosun et al., 2018] where a system made out of SMORES carry the RGB-D sensor around and use it for closed loop self reconfiguration as well as navigation. The simplest way however for tracking is to have the RGB-D camera externally. Such an external camera could be placed on top of a structure built out of modular robots as e.g. in [Magnússon et al., 2013]. However, because the camera in this work is used to track multiple objects in a simulated living space, we use a stationary mounted external camera with an overhead view.

Concerning the vision of providing assistance in everyday tasks, robots started entering houses to do chores much later than their industrial counterparts that are used in process automation. One of the pioneering service robots which became widely available is Roomba [Forlizzi and DiSalvo, 2006]. Although consumer robots are not very widespread, there is a deep literature on the research of service robots reporting various tasks such as cloth folding [Maitin-Shepard et al., 2010, van den Berg et al., 2011], social interaction with humans [Broekens et al., 2009, Martin et al., 2009, Demiris, 2009], doing a combination of different chores [Beetz et al., 2012, Srinivasa et al., 2012, Asfour et al., 2013] and helping disabled people with an autonomous wheelchair [Carlson and Demiris, 2012] or bidepal robot [Sugahara et al., 2004]. All of those robots are either partially of generic human form or specifically designed for a given task. Here, we explore the possibility of such tasks being attempted by distributed systems such as SRMRs, specifically in the scope of creating an assistive environment.

Whereas conventional furniture consists of passive elements that stays where they are placed, advancements in technology and robotics call for smarter and interactive houses. Examples are a robotic ottoman system guessing your intention and responding to your needs [Sirkin et al., 2015] or a robotic wardrobe to modulate living space to needs of the user [OriLiving, 2019]. Thanks to the expansion of internet of things, there is an increasing number of architectural efforts to sensorize furniture using cameras and other complementary sensors [Georgoulas et al., 2014] and in designing actuation systems for furniture [Gross and Green, 2012], [Yu et al., 2007].

The vision of Roombots is to introduce dynamically reconfigurable furniture into our future living space, redefining room arrangements and available space. In this work, we show proof-of-concepts of functionalities enabled by such reconfigurable furniture, namely dynamic adaptivity and assistance. We are aware that our current work does not contain much quantitative data; we rather report qualitative data by showcasing the many abilities of a reconfigurable system. We believe that such progress is also important to report as it can serve as a benchmark for future iterations and generations of reconfigurable systems.

2.3 Material and methods

The following subsections describe the different modifications of the RB hardware and software in detail: the adaptations on the module itself, the added sensors and the new control possibilities, the new gripper, added electronics, the Graphical User Interface (GUI), the external RGB-D camera and the passive parts (furniture).

2.3.1 Mechanics, motors and connections

The overall design and dimensions of one Roombots module are kept as the original design described in [Sproewitz et al., 2014], thus only the main features are described here. One module consists of 4 interconnected half-spheres (hemispheres) that can continuously rotate around each other by 3 motor units. Each module has 10 connection plates of which two possess actively retractable hooks that can latch onto any other connection plate. This Active Connection Mechanism (ACM) is the mechanism that allows one module to attach to another module or to any surface equipped with our connection plate, forming dynamic multi-module structures such as adaptive furniture. One module contains all necessary electronics for operation, a bluetooth module for communication and batteries, making it fully autonomous. Fig. 2.2 gives an overview of the design of a single module.

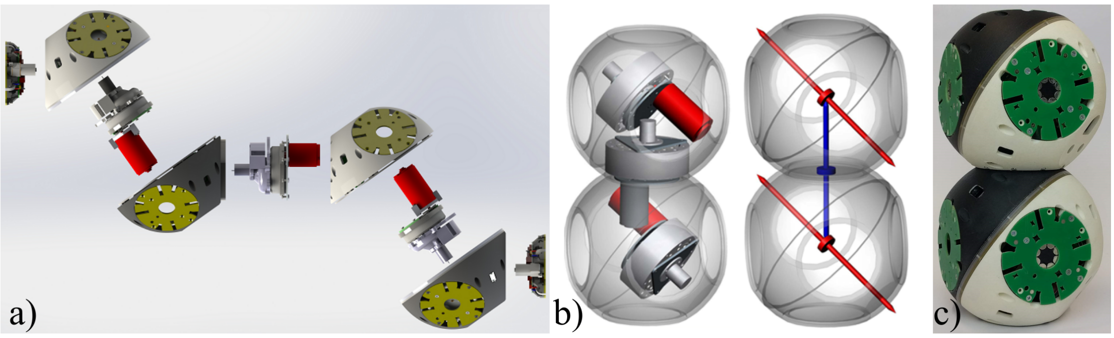


Figure 2.2 – Design of a single module with a) an exploded view of the main components Active Connection Mechanism ACM (2), hemispheres (4) and motor units (3), b) arrangement of the motor units inside the assembled module and orientation of the rotation axes (red and blue axes), and c) fully assembled Roombots module of the newest iteration with dimensions 11x11x22 cm. Parts of this image have been adapted from [Sproewitz et al., 2009].

As already pointed out in [Sproewitz et al., 2014], a stiffer construction and stronger actuators were needed to overcome some of the limitations of the initial design. From previous iterations of the hardware, it became clear that the design needed major improvements in three areas: motor strength, control precision, and connection reliability. As a first step, the material of the three main gearboxes that are driving each module was changed from plastic to brass together with replacing the respective motors to more powerful ones. On the one hand, this significantly increased the torque of the Degrees of Freedom (DoFs) of one module (19% increase for the middle DoF with a Maxon RE-max 24 DC motor to 4.3 Nm, and 70% increase for the outer DoF with a Maxon RE 25 DC motor to 8.4 Nm, [Vespignani, 2015]). On the other hand, due to the much more precise metal gearboxes, the backlash from each gearbox was reduced by a factor of 10 (from 2 ° to 0.2 °) which was necessary to improve the control precision and thus the connection reliability of a module. Many difficulties in the ACM caused by the use of 3D-printed parts and plastic gearboxes also required a change of material. This mechanical latch needs to withstand the full weight of a few connected modules. Thus, we decided to also implement metallic gearboxes and manufacture the structural parts of the ACM out of aluminum which also helped to increase the connection reliability of a module.

In an effort to compensate for small misalignments during reconfiguration, we reconsidered the hybrid ACM mentioned in [Sproewitz et al., 2014] in which permanent magnets are used to assist the connection action. It consisted of one permanent magnet in the middle of the ACM and connection plates that would need a specific polarity to create attraction, thus unfavorably changing the hermaphrodite ACM into a gendered connection mechanism. We achieved a hermaphrodite connection with permanent magnets by creating a *ring* of magnets whose polarities are oriented in a specific manner (see Fig. 2.3) such that an attraction is created every 90 ° which corresponds to how two modules can connect to each other. A standard ring is formed by 8 disk-shaped magnets with alternating poles facing the surface; due to space constraints, ACM only contains 6 magnets, so the ring is missing a quadrant. Each ACM and every possible connection plate both of a module and of a passive grid structure

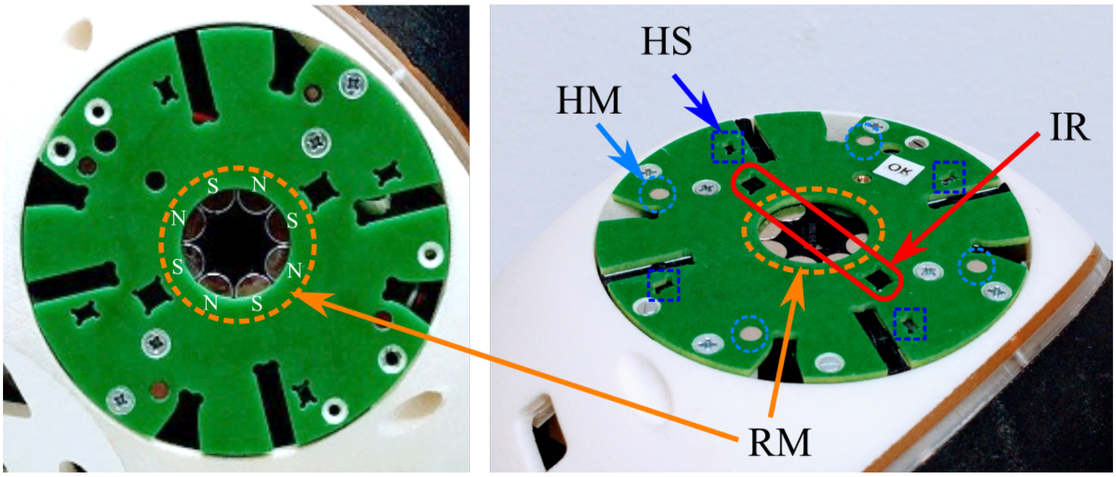


Figure 2.3 – Modifications of the connection plates. In the middle of each plate, a ring of permanent magnets with alternating poles (N and S) facing the surface (RM; 8 magnets for a standard plate, 6 magnets for an ACM) assists in forming contact between two plates. Two infrared sensors (IR) give information about the proximity of neighboring plates or the environment whereas four pairs of a hall-effect sensor (HS) and corresponding magnet (HM) indicate if two touching plates are oriented appropriately such that a connection with the ACM can be formed.

has been equipped with this ring of permanent magnets. They are helping to form an aligned contact between two connection plates that are supposed to be touching. This magnetic connection is strong enough to form a small region of attraction when two plates are close, however it is not capable of transferring any load and the connection is easily broken by any movement of a module. The researches in [Kurokawa et al., 2008] and [Davey et al., 2012] are using a similar magnetic ring idea as a main connection method. Using permanent magnets as main connection method results in harder disconnection process. However, the magnet ring is used as only a complementary method to the mechanical ACM in Roombots. Thus, advantages of both methods are integrated while minimizing disadvantages of each approach.

In the end, we built 11 new fully functional modules with the presented hardware modifications (with a total of 13 modules with 2 modules of the previous iteration).

2.3.2 Sensors

The most important sensorization was done on the rotary joints by adding absolute encoders (12-bit capacitive absolute encoder AMT203). Initially, Roombots had relative encoders which meant that modules were supposed to be powered up only after all joints were manually aligned to a certain angle such as 0°. New absolute encoders ensure that the control of a module does no longer depend on the initial position when switched on, thus the initialization of experiments became easier. Although both relative and absolute encoders could have been used in conjunction, relative encoders have been removed to open up some space for stronger

DC motors.

There has been a high demand on proximity sensors on Roombots for various applications, particularly for reliable self reconfiguration. Each ACM plate is equipped with two infrared (QRE1113GR) and four single-axis linear hall-effect sensors (DRV5053RA) (shown as IR and HS respectively in Fig. 2.3). Infrared sensors detect existence and distance of a nearby object within the range of 1 mm to 15 mm. The reading of the infrared sensor relies on the reflectivity and orientation of a facing surface. On the other hand, hall effect sensors report the magnetic flux, Φ_B, crossing the sensor area which correlates to the proximity of a magnet. The actual Φ_B reading depends on distance, strength, orientation and misalignment of a facing magnet. Every possible docking surface (matching surface of ACMs) are equipped with four cylindrical magnets with a diameter of 3 mm (illustrated as HM in Fig. 2.3). All four magnets are situated right in front of hall effect sensors when an ACM is connected. Thus, the viability of docking can be assessed. Additionally, each ACM is equipped with a three-axis accelerometer (ADXL337) to detect the gravity vector. The gravity vector helps to detect external disturbances or internally accumulated errors such as gear backlashes, manufacturing/assembly misalignments or local control imperfections. The selected sensors are quite commonly used in many embedded systems. The use of similar sensors has been reported in different modular robots too. For instance Sambot in [Wei et al., 2011] uses infrared sensors, accelerometer and gyroscope integrated in modules whereas [Zhu et al., 2014] presents a non-actuated sensor module just for carrying linear hall effect sensors and a camera.

2.3.3 Spotlight LED

A powerful RGBW-spotlight LED board has been developed to illuminate a workspace whenever the ambient lighting is insufficient. This spotlight can be integrated within any Roombots module at multiple possible locations. Particularly when Roombots need to incorporate a camera, the environment needs to be well-lit. Whenever there is a lack of light, a module can carry the spotlight to the desired location and enable the computer vision approaches. Furthermore, such a colourful and strong light can be used to induce emotions by playing with color and intensity functions. For example creating a sunset effect in a bedroom without any windows or fireplace feeling in a living room without a fireplace.

2.3.4 Universal gripper

In the vision of Roombots as depicted in Fig. 2.1, the functionality of object manipulation had to be implemented in the Roombots platform. There were essentially two options available: a manipulator either could be designed as a standalone system that could be attached to a module with the ACM, or it could be implemented directly into a module. For the first option, the method of manipulation would be more open since the space needed for it is not restricted as a standalone gripper unit could have arbitrary dimensions. As a disadvantage, a new power management and communication line would have to be designed for such a system.

For the latter option, only methods that do not require much space can be considered as everything has to fit into one hemisphere, although power and communication can be shared from the module itself. This advantage led to the decision to integrate a manipulator into a module, and thus an appropriate gripper technology had to be found with two constraints: (i) to work in limited space and (ii) to be able to manipulate a variety of everyday objects (glasses, remote control, USB stick, cutlery, etc.). After much consideration, the interesting concept of granular jamming was used to develop a “Universal Gripper” (UG) similar to the one in [Brown et al., 2010]. This gripper consists of a closed, flexible membrane (often a party latex balloon) filled with small granules (e.g. sand, ground coffee, etc.). The membrane is normally soft and adapts to the shape of an object when pushed onto them. The actual gripping is achieved by transitioning the fluid state of the granules into the solid (jammed) state by creating a vacuum inside the membrane. This effectively locks the shape of the membrane which now is able to exert a gripping force on the object. Due to this interplay of a soft, shape-adapting state and a solid, gripping state, this gripper is capable of manipulating a wide range of objects as demonstrated in various previous works (e.g. [Brown et al., 2010, Amend et al., 2012, Amend et al., 2016]).

Integrating a UG into a Roombots hemisphere still posed a number of challenges, mostly around downsizing the mechatronic components, especially the required and usually large vacuum pump. A careful selection of a miniature vacuum pump (Schwarzer SP 100 EC) and small-scale solenoid valves (SMC S070C-SAG-32) resulted, to the best of our knowledge, in the first-of-its-kind mobile jamming gripper integration within an MR. Two modules have been equipped with this type of gripper which is filled with ground coffee as in the original version in [Brown et al., 2010]. In addition of being able to manipulate various objects, the control of the gripper is extremely simple and robust where the vacuum pump - with the solenoids - either inflates or deflates the membrane to set pressures, measured by a single pressure sensor (Honeywell 015PAAA5). Details of the implementation are depicted in Fig. 2.4 left, and the final Roombots gripper module is shown in Fig. 2.4 on the right. Although the Universal Gripper is very capable, it has few limitations such as the object size and weight which depends on the size of the gripper, pressure in the membrane chamber and type of granules and membrane. Furthermore, it needs to press against to object. Hence, it is not always suitable to handle standing objects that has sensitive balance or soft objects.

2.3.5 Electronics

A considerable part of the electronics was redesigned to accommodate for the hardware changes (see Fig. 2.5 for details). The power board, motor control boards and ACM control boards went through major updates. The technology of communication with the modules (Bluetooth) as well as the communication bus inside a module (RS-485) were kept the same. Three new board types were designed to control the additional hardware. The hemisphere with an integrated Universal Gripper contains a gripper board that is responsible for controlling the pressure inside the membrane by using a pressure sensor, vacuum pump and valves. Further,

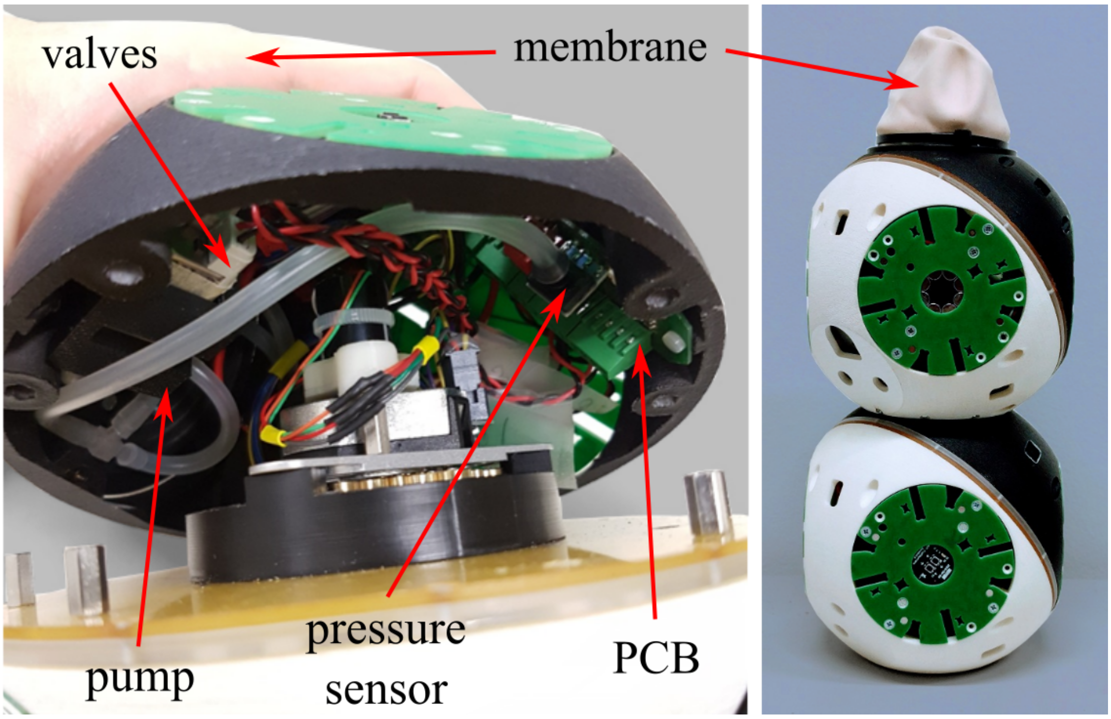


Figure 2.4 – Integration of a Universal Gripper into a Roombots hemisphere. Left: a miniature vacuum pump and small-scale solenoid valves allowed the components to be fully contained and a specialized gripper PCB alike the existing electronics measures and controls the pressure inside the flexible membrane. Right: a gripper module with the Universal Gripper sticking out on top, replacing one of the ACMs.

a specialized spotlight board controls the powerful RGBW-LED integrated in one hemisphere which is used for illumination purposes in different colors and brightness. Finally, a LED-ring board gives feedback to a nearby user (this board has been developed in [Ozgur et al., 2014]).

Further, readout boards for other new sensors have been integrated in Roombots. The values of the proximity sensors and accelerometer are not directly used by modules, however they can be reported to an external PC which can execute higher level controllers (as e.g. in [Khodr et al., 2019] for a local connection search algorithm), summarized in section 2.3.7.

2.3.6 Motor control

Changing the hardware led to changes in the firmware of various boards in Roombots. The new absolute encoders have a much lower resolution (number of ticks per rotation) than relative encoders, which particularly affected the derivative control. Hence, the frequency of the control loop was reduced and PID parameters were adapted. Absolute encoders read over SPI and the communication channel is going through slip rings. Although the communication with absolute encoders is mostly smooth, rare encoder reading errors can occur. Various reading error detection and safety procedures have been implemented. For instance, all the

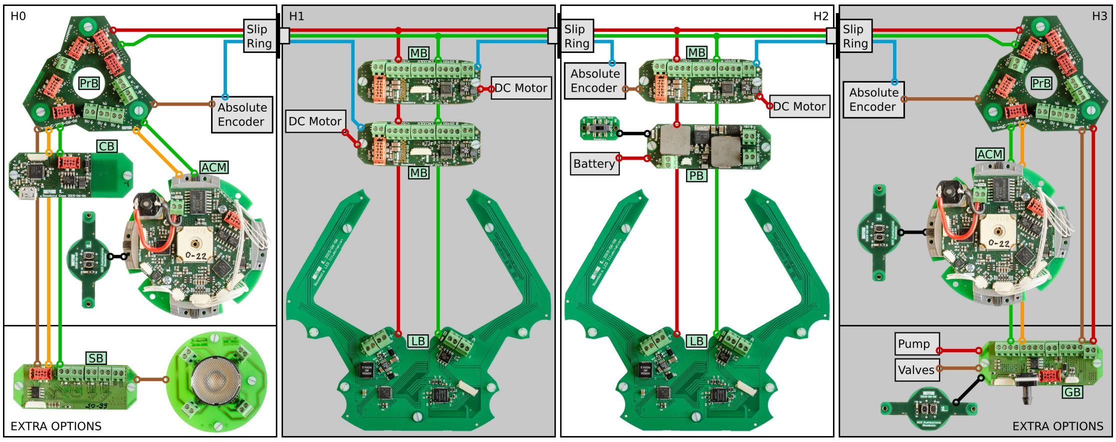


Figure 2.5 – Roombots electronics is also modular and distributed throughout a module in hemispheres H0-H3. The background of each hemisphere region is shaded to match colors of the real hemispheres. In each Roombots module, there exists 3 motor driver boards (MB), 2 active connection mechanisms boards (ACM), 2 proxy boards (PrB, used to convert voltages and ease assembly), 2 LED boards (LB, used to give feedback to the operator), 1 communication board (CB, provides Bluetooth connection and drives the inner RS-485 bus) and 1 power regulation board (PB). Furthermore, each module can support extra options such as spotlight board (SB) or Universal Gripper control board (GB) in the outer hemispheres. Extra option slots are identical and each electro-mechanical subsystem can be placed in either of the outer hemispheres. The extra options are not considered as essential for the basic operation of Roombots modules; only selected modules are equipped with them. Electrical connections within a module are illustrated with different colors: Red (15V), orange (6V), brown (5V), green (RS-485 bus), blue (SPI) and black (support boards for easy user access). Each physical connection is shown with a circle whereas unconnected lines pass through a board without a circle.

DC motors are disabled if the position error gets unexpectedly large which may occur due to external disturbances or persistent erroneous encoder readings.

2.3.7 Connection control

The proximity sensors and accelerometer data can be used to check the validity of a connection. A connection can only be formed if two plates are close enough and correctly oriented which can be checked by the IR-sensors and hall effect sensors. Additionally, in a lattice-type configuration, the gravity vector must be collinear to either the global x-, y- or z-coordinate, measured by the accelerometer; other directions indicate non-idealities such as body elasticity, local control error, or gear and ACM backlashes.

These properties are used in [Khodr et al., 2019] to develop a local search algorithm for reliable connections that has two stages. In our case, the condition for a successful connection is the proper alignment of the four hall effect sensors. Since their values is maximal when directly opposite a magnet, it is enough to threshold the sum of the four sensors to validate a possible

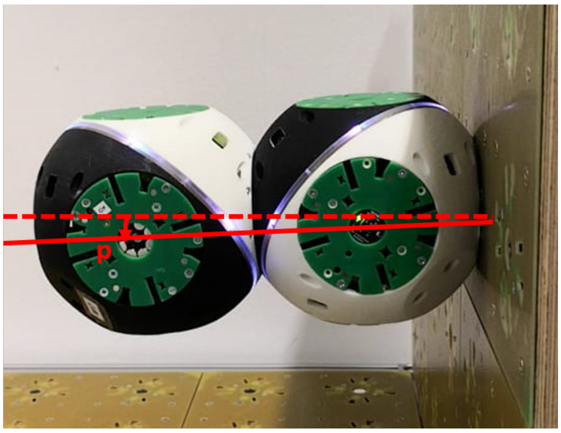


Figure 2.6 – Pitch angle compensation. The accelerometer detects deviations (red solid line) from the global orthogonal frame (red dashed line) and uses an inverse kinematics model of the module to correct imperfect attachments and module deflections.

connection. If the sum is above the threshold when a connection sequence is initiated, the command list continues normally. If the sum is below the threshold, the first stage of the local search engages, where the accelerometer data is converted into a *pitch* angle. The pitch then is used to generate motor commands for a new desired angle by an inverse kinematics model of the module (Fig. 2.6) to reorient the plates orthogonally. If this is still not satisfying the threshold condition, the second stage with random movements engages. This is because the occurence of ACM misalignments strongly depends on the movement history of each module such that modeling this effect is infeasible. Instead, one of the three motors is randomly chosen and moves by either +1° or -1°, followed by a new evaluation of the hall effect sensor values. This process is repeated until the threshold condition is satisfied.

The result of this local search algorithm was a drastic improvement of single-module on-grid locomotion: while one module before could only locomote on a tabletop and had a 0% success rate for certain movements on a wall or ceiling, the new connection control was tested in three of the most challenging scenarios (i) going up when attached to ceiling, (ii) going up when attached to a side wall, and (iii) going left when attached to a side wall. Each scenario was repeated 10 times. The new docking position search method works with a 100% reliability for all movements (average convergence time for a connection by random movement is 43.53 s), a crucial property for any self-reconfiguration with the hardware.

2.3.8 Graphical User Interface (GUI)

For designing different complex tasks as outlined in the introduction, an easy-to-use User Interface (UI) was needed. Different user interfaces with Roombots were already explored in the past, e.g. using Playdough to form structures [Mutlu et al., 2018] and an approach based on Virtual Reality (VR) [Nigolian et al., 2017], however these previous works could not be adapted to larger scale dynamic formations. The Playdough interface does not work in real-time, and the VR only allows a limited kinematic range for simplicity reasons. Since Roombots are able to form complex geometrical structures, it seemed natural to develop a Graphical User Interface (GUI) to be able to visualize such formations. This GUI has been developed with Unity[1], a 3D animation software that often is used for creating video games. Even though Unity has the capability to function as a physics simulator, we only use the visualization aspect of the software. Naturally, the visualization does not always represent the real configuration of modules perfectly due to the physics present in the real world, causing e.g. elastic deformation of modules and attachment plates under load. This effect as well as others, arguably could be modelled and hence included in the visualization. However, throughout our experience with the real modules, we found these effects and other undefined sources of noise to be too stochastic for modelling and decided to implement control routines in the modules to account for discrepancies. These routines will be discussed below in section 2.4.1; the GUI itself was kept simple and assumes perfect conditions.

Fig. 2.7 shows screenshots of an example usage of the GUI. There are three main parts: control panel, visualization panel and message panel. The control panel has three sub-categories: manual commands, operation mode and list of modules. In “manual commands” (left), single commands (motor, ACM, LED, gripper, spotlight) can be created and sent to specific modules. These commands can also be added to a list which is created on the fly. In “operation mode” (second left), this list of commands is stored, and it can be executed in a single sequence to perform a specific task. Command lists also can be saved and loaded here. This category also allows the execution of predefined movements of a Central Pattern Generator (CPG) as well as executing external plug-ins, e.g. control of a motor with a computer keyboard. The yellow border around the panel indicates that the control has been switched from the virtual modules to the actual real-world modules. Connected modules provide real-time feedback of all the motor and sensor states which then are displayed in the GUI. In “list of modules” (second right), all virtual modules are listed and modules can be created or deleted. The current state and positions of all modules can be exported as a “scene”, and scenes can also be imported again, allowing a quick setup for new tasks. Here, each virtual module can also be connected to a real module by bluetooth, which then allows an easy switch between performing a list of commands with the virtual modules alone or together with the real connected modules. On the bottom, the visualization panel has an arrangement of virtual Roombots modules is visible. This panel also contains and a (empty) message panel to display warnings and other messages (not shown here).

2.3.9 RGB-D vision system

Some of the proposed tasks require an external system that tracks a user and furniture present in a setup. We have already used the Microsoft Kinect sensor in previous works for tracking purposes and reimplemented the same system also for the current chapter. Here, a Kinect v2 sensor is mounted on the ceiling and observes a designated ground area. It functions as

---

[1] https://unity3d.com

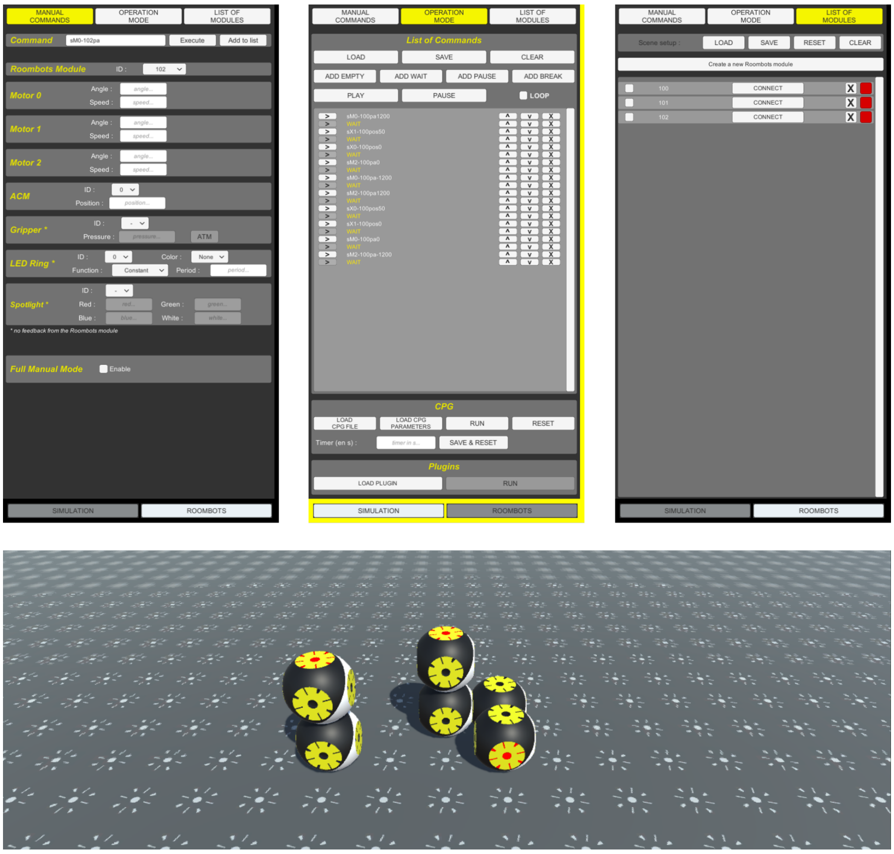


Figure 2.7 – Screenshots of the GUI. The control panel can be switched between “manual commands” where specific commands can be sent to single modules, “operation mode” where a sequence of commands can be played and additional modes are available (CPG and plug-ins) and “list of modules” where all modules are listed and can be connected by bluetooth to real Roombots modules. On the bottom, a visualization panel shows a Roombots arrangement, and the message panel below can display various informations during runtime (not shown here).

a depth sensor and can distinguish objects with different heights in the scene. It is assumed that the highest object in the scene is the head of a user, and multiple pieces of furniture are marked in specific ways to make them unique and define their orientation. Fig. 2.8 a) and b) give an overview of the tracking system. A recent use of RGB-D sensor in the scope of SRMR has been reported in [Tosun et al., 2018] where a system made out of Smores carry the RGB-D sensor around and use it for closed loop self reconfiguration as well as navigation.

2.3.10 Robotic-enhanced furniture

As two example pieces of furniture to demonstrate the proposed tasks, we equipped a small table and a chair with Roombots modules to extend their functionalities. Fig. 2.8 c) and d) show both pieces of furniture. The chair is put on passive caster wheels such that it can roll

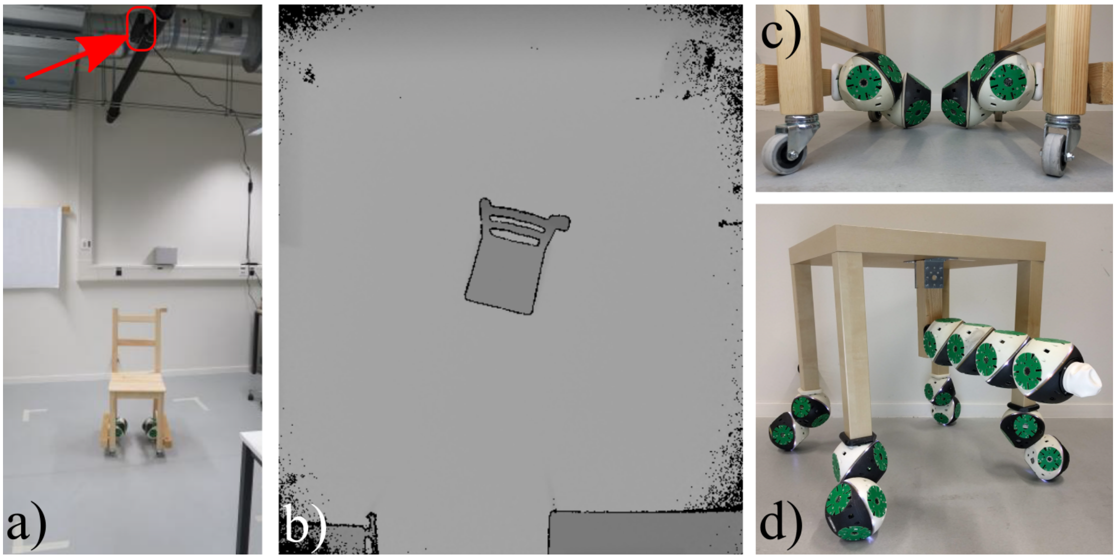


Figure 2.8 – RGB-D camera and furniture used in the sub-tasks. a) An overhead Microsoft Kinect depth sensor (red arrow) can track objects within the area marked with masking tape. A chair is shown in this area. b) View of the camera; depth is mapped onto pixel gray-scale making the chair easily identifiable. c) Close-up view of the caster wheels and the differential drive-like wheel modules of the used chair. d) The used table with four wheel modules and a meta-module with a gripper attached underneath.

in any direction, and two modules use their outer hemispheres as wheels to drive the chair similar to a differential drive. Since the main load is borne by the passive wheels, a user can sit on the chair, however the chair is only moved by the modules under no additional load. For the table, each leg is extended by one module, and again using the outer hemispheres for rolling, the table can move almost omni-directionally. Additionally, a gripper structure consisting of one gripper module in series with a standard module can be mounted underneath the table. This allows the table to pick up objects from the ground and place them onto itself as will be shown later.

2.4 Experimental results

To demonstrate the augmented capabilities of the modified RB system, we designed five main tasks which are demonstrated in various sub-tasks. The results of these experiments are presented in this section and the link to the supplementary video including all described experiments can be found in Tab. 2.1.

Table 2.1 – The video is available as supplementary material for this article (see link)

	Description	Location
1	RB sub-tasks	link

2.4.1 Easy-to-use user interface: The GUI

GUI is almost the backbone of Roombots. Although, it is possible to control Roombots without a GUI, it significantly eases any use. There is no quantitative data to report in this section. However as an example to present the work flow of the GUI from task definition to hardware execution, we here describe the development of a simple demonstration, “Circle walk”. The demo consists of a sequence of movement commands that make a module locomote on-grid in a circular manner.

At startup, the GUI visualization area is empty, and a new virtual RB module is added. The module is initialized with both ACMs opened and all motors in their zero state. By default, it is placed in the GUI vertically onto the grid with one ACM facing the grid. First, an initialization script is loaded into the GUI that attaches the module to the grid and brings the motors in the correct position to start the circular loop. Then, the main list is loaded and the loop-box ticked, which makes the full sequence loop as long as needed. This main list of commands consists of a predefined sequence of ACM opening and closing actions and performing 90 ° rotations of motors in between where looping these actions makes the module move in a circular fashion on a single 2x2 grid plate. The top row in Fig. 2.9 shows the initialization and the first half of the circular loop demonstration in the GUI.

Once the demonstration has been developed in the GUI, it is only a matter of connecting a real module to the virtual module in the GUI. Once connected, the mode is switched from “Simulation” to “Roombots” to send commands to the bluetooth connection, and the demonstration is performed with the hardware (bottom row of Fig. 2.9). Additional control routines are implemented to increase the rate of success for executing commands. The GUI continuously crosschecks the setpoints of motors and ACMs with the actual values fed back from the module. If read values do not lie within a small margin, GUI sends the corresponding commands again. The most crucial part of a reconfiguration is forming the connection to a new attachment point. Handling misalignments during this process required more elaborate strategies. In our case, the sensors implemented on the ACM plates give the necessary information concerning the correct alignment. If a misalignment is detected, a search algorithm performs small actions around the setpoints that usually result in a successful connection. Details of this algorithm can be found in [Khodr et al., 2019].

The GUI possesses more features, e.g. forcing the starting point for the forward kinematics of connected RB modules in simulation, adding pauses to a sequence of commands, running CPG networks and external plugins and more. The described example only present the basic functions of the GUI which are enough to make the development of simple demonstrations relatively quick and easy.

Research on most of the (SR)MR systems begins with simulation to study the feasibility of the system because simulation is easier and faster to set up compared to robot hardware and can give valuable information. Moreover, very complex scenarios can be explored in simulation environment. Such a simulation is not necessarily user friendly. Nevertheless,

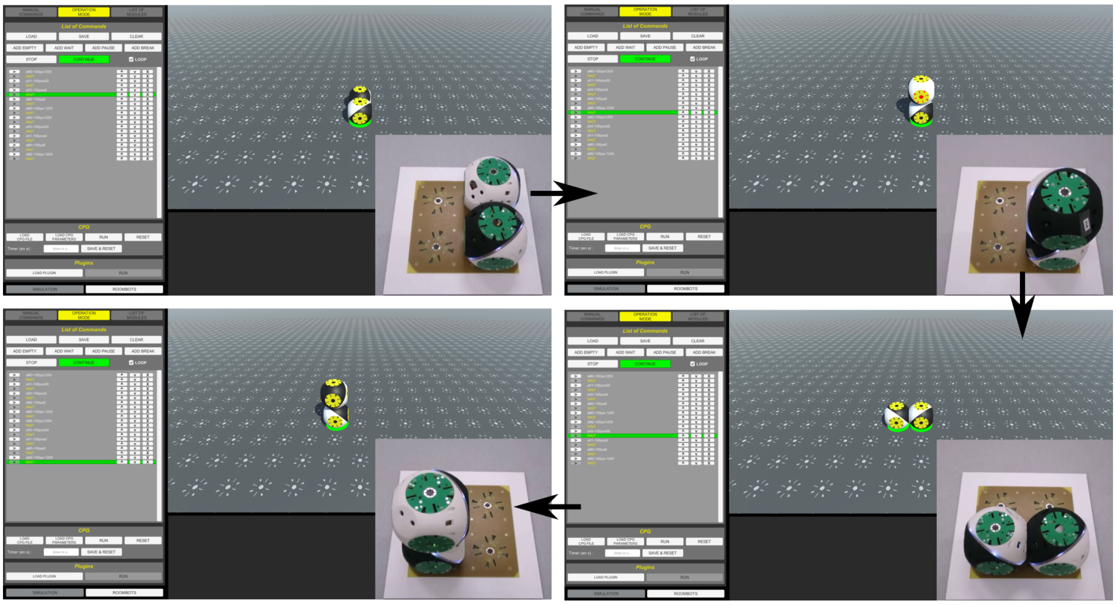


Figure 2.9 – Circle walk demonstration. The GUI is shown in “operation mode” where a preprogrammed list of commands is executed in sequence. The first half of the circle walk is shown; looping the script makes one module go around on a 2x2 grid. The corresponding hardware configuration is shown as insets.

most of the modular robotic systems are accompanied with simulation/GUI tools. For instance, [Krupke et al., 2012] explains a simulation environment for a modular system whereas [Kurokawa et al., 2000] includes self-reconfiguration steps too. One of the most comprehensive and recent SRMR GUI (VSPARC) is designed for Smores in [Jing et al., 2016]. It is using Unity engine similar to our approach. Most of the features presented for VSPARC also exists in the Roombots GUI, adapted for the Roombots hardware.

2.4.2 Scalable self-reconfiguration

As an example of a larger self-reconfiguration task (i.e. with more modules and more steps) that goes together with the application of adaptive furniture, we show the formation of a small chair with 12 modules. The details of the planning of this demonstration can be found in [Khodr et al., 2019]. The work includes the development of a local search algorithm and closed-loop control during the connection process that enables more reliable reconfiguration, as well as the derivation of an A* search algorithm that finds the motor command sequence to form a given structure from arbitrary initial conditions.

Here, only the final formation sequence is shown, starting from 12 modules standing on a flat grid. In order to decrease the number of nodes on the search tree for such a large number of possible motor actions at each state, pruning methods are included in the search algorithm to avoid exploring not promising branches. The formation of the full chair is divided in to 6 subconfigurations (manually preprocessed by an operator) that must be reached in a specified

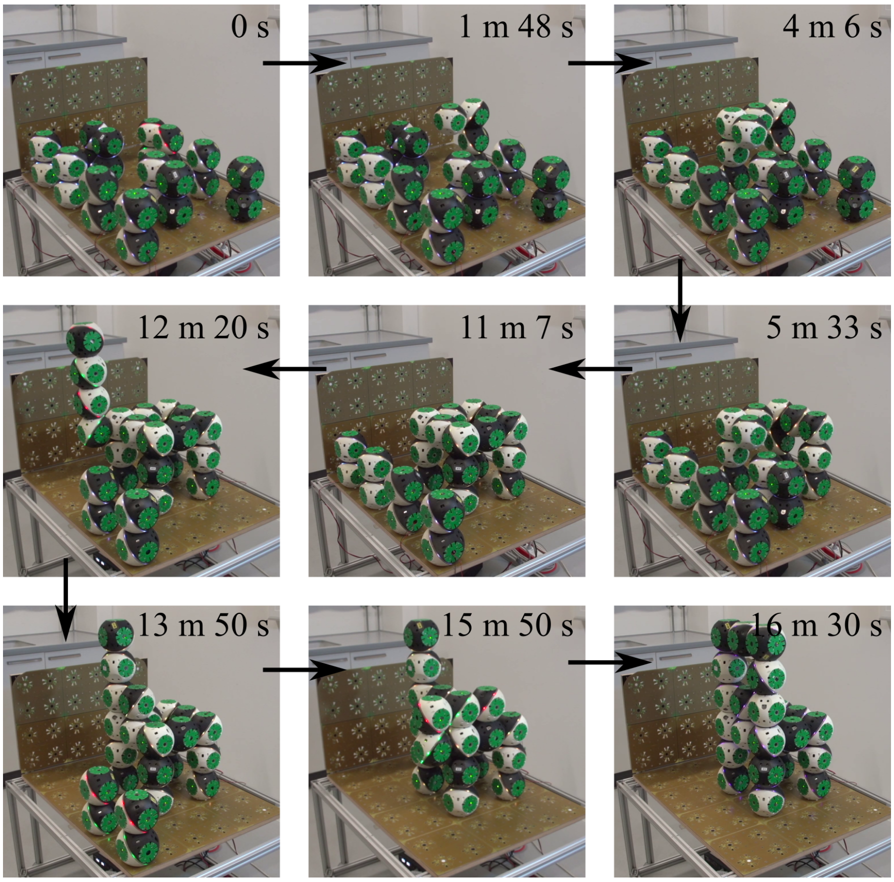


Figure 2.10 – Snapshots of the chair formation sequence. The chair is built in 6 stages (4 for the legs and 2 for the backrest). The large time loss in the middle is caused by a battery change and a manual intervention to connect the leg segments, however as shown in the supplementary video, the self-reconfiguration is largely autonomous.

order and build up the chair in stages. Fig. 2.10 shows snapshots of the formation process.

In 3D SRMRs, we could only find a few systems that demonstrated self-reconfiguration with more than 8 modules. M-TRAN demonstrated cluster flow with 24 modules (48 DOF total) in [Kurokawa et al., 2008]. However, in most of the demonstrations of M-TRAN, the structure stays connected. ATRON [Brandt et al., 2007] presents a sequence with 9 DoF. Similarly, SMORES demonstrate a system with 6 up to 9 modules (4 DOF each) with considerable autonomous (closed-loop) shape changing with disconnections and re-connections in [Tosun et al., 2018] and [Daudelin et al., 2018]. Although M-TRAN, ATRON, SMORES and various other SRMR are potentially capable of showing a similar application shown in 2.10, to the best of our knowledge this is the first time a 3D self-reconfiguration is shown using more than 30 DOF (36) with all modules initialized as separate (detached). Further details of this sub-section will be explained in Ch. 3.

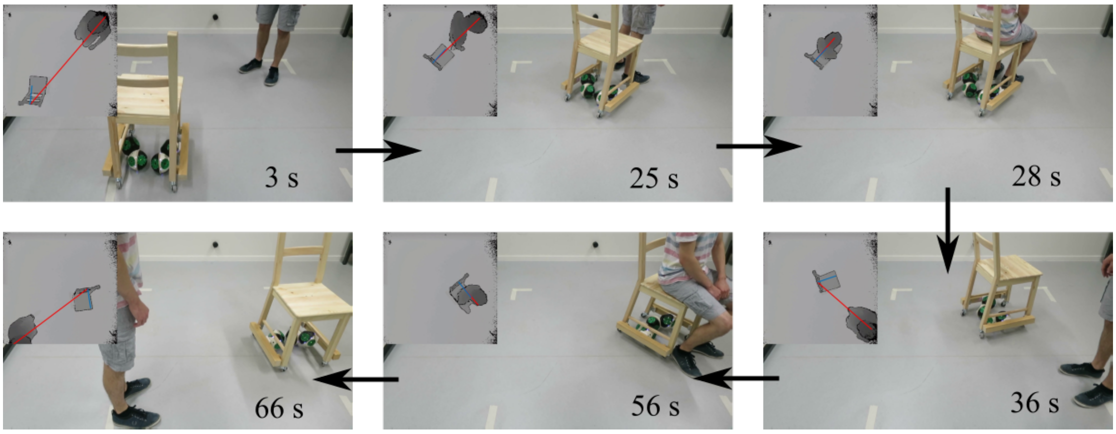


Figure 2.11 – Following chair. An overhead Kinect (top left panels in each snapshot) tracks the position and orientation of an assistive chair (short blue line) and the path to the patient (long red line). The user can sit whenever needed as the chair follows the user inside the predefined area.

2.4.3 Mobile furniture

The task of mobile furniture is demonstrated in four sub-tasks: following chair, evading chair, adaptation to environment and overcoming obstacles.

Following chair

We explore the possibility of an assistive chair that follows a user to always be available in case of the user needing to rest. The system is aimed at preventing falls - a common problem among elderly - and providing assistance for manual recovery after a fall. It consists of an RGB-D camera tracking the position and orientation of the user and of the assistive chair in a predefined area. Two Roombots modules are attached to the chair which allows the chair to move anywhere and in any orientation in the area, similar to a two-wheeled differential drive robot. A controller (PC) computes the difference between the position and orientation of the user and the chair and sends movement commands to the modules such that the chair is always next to the user ready for the user to sit down (Fig. 2.11).

Evading chair

We also demonstrate the inverse scenario where Roombots furniture has to make space to let a user pass. This scenario could take place in a packed apartment where the available space is limited and shared between user and furniture, or a user is physically unable (e.g. in a wheelchair) to move furniture out of the way. We again used the depth map to track a user and the same chair as in the previous demonstration. The chair initially is located in the middle of a defined area and the user would like to cross this area. A controller (PC) extrapolates the desired path of the user and moves the chair orthogonally to this path out of the way to make

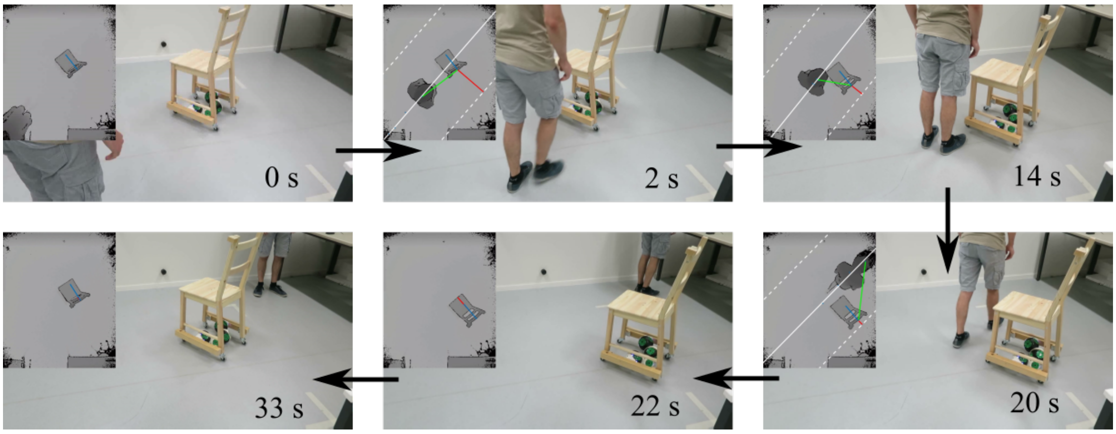


Figure 2.12 – Assistive furniture opens up space. An overhead RGB-D camera (top left panels in each snapshot) tracks an assistive chair (short blue line) and the user's location with respect to the chair (green line). It extrapolates the path of the user (white lines) and moves the chair orthogonally to it (short red line) before returning to its original position after letting the user pass.

space for the user to pass. After creating enough space and the user passing, the chair moves back to its original position (Fig. 2.12).

It is important to note that both "following and evading chair" demonstrations are not complete and comprehensive solutions to solve the problem. The full solution would require significantly more research which falls out of the scope of this chapter and the SRMR field. Such a solution would involve detailed intention detection considering a much wider spectrum of actions and would incorporate safety and emergency protocols. Nevertheless, they serve as a proof-of-concept that SRMR can be used in such applications.

Adaptation to environment

Roombots-enhanced furniture can have additional functionalities besides providing mobility to existing pieces of furniture. In this demonstration, we equip a small table with four modules, fixed to the end of each leg of the table. We use the outer hemispheres (one DoF) as wheels to enable the table to act similarly to an omnidirectional vehicle. In this specific example, the two other DoFs of a module can move in a null-space such that the total height of a leg can be varied. The table transports a set of objects and this height adjusting ability can be used to keep the tabletop horizontal on uneven terrain to prevent the objects from falling off the tabletop. We let the table drive from a straight area into a sloped area. In the first case without adaptation, the set of objects falls off the table after a short distance on the slope area. If the slope compensation is activated, the tabletop can be kept closer to the horizontal plane which prevents the objects from falling on the sloped area. Fig. 2.13 depicts snapshots of both cases.

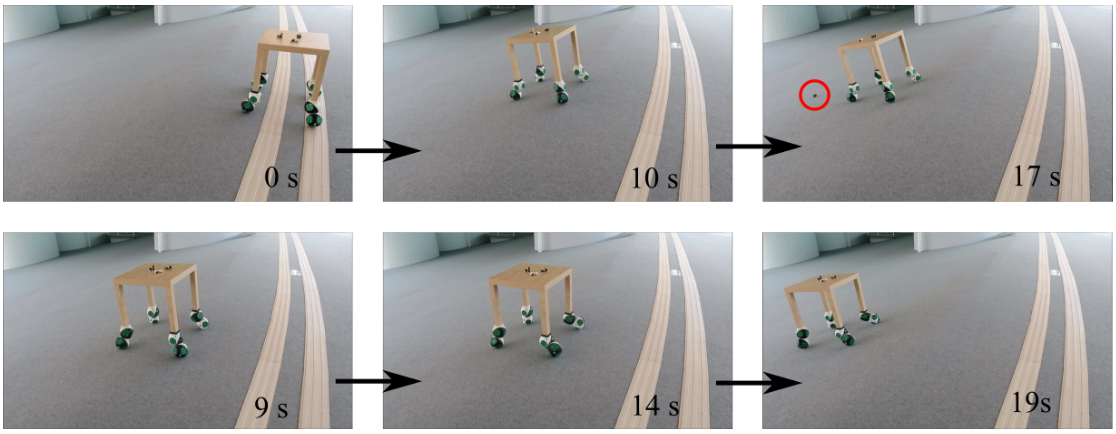


Figure 2.13 – Active slope compensation. Top row: a table driving over a sloped area (left side of double stripes) tilts its tabletop, causing objects on top of it to fall (red circle). Bottom row: the attached RB modules can partly compensate for the slope by shortening the hind legs of the table, keeping the tabletop more horizontally such that the objects stay on top.

Overcoming obstacles

For the table in the previous demonstration, an additional behavior is available by "rotating" the RB modules around their attachment point at the end of each leg. This causes a module to move in a circular manner around the tip of the leg, performing a movement similar to a step. We use this movement to demonstrate climbing a small ledge that the table cannot drive over. The table drives towards the ledge until the front legs are aligned and touch the ledge. The front RB modules then rotate, moving on top of the ledge by doing so. The same process then repeats for the hind legs, after which the table has successfully overcome the ledge. Fig. 2.14 depicts snapshots of this process. In this demonstration, the table is piloted by a human and the sequences are manually initiated.

2.4.4 Manipulating furniture

One of the most needed and repeated task in daily life (as well as industrial settings) is object manipulation. Hence, it is an implicit requirement for a SRMR to be able to manipulate the surrounding objects, particularly when humans and SRMR need to cohabit.

Object pick-up

A small table is equipped with one module on each leg that enables the table to move similarly to an omnidirectional vehicle. Additionally, a meta-module structure with an implemented Universal Gripper is attached to the underside of the table. The task is to start at location X, pick up a pen at location Y and bring it to location Z. The table is again manually controlled and rolls into position to pick up the pen. The gripping sequence is preprogrammed and executed once the table is in the correct position. It picks up the pen from the ground and

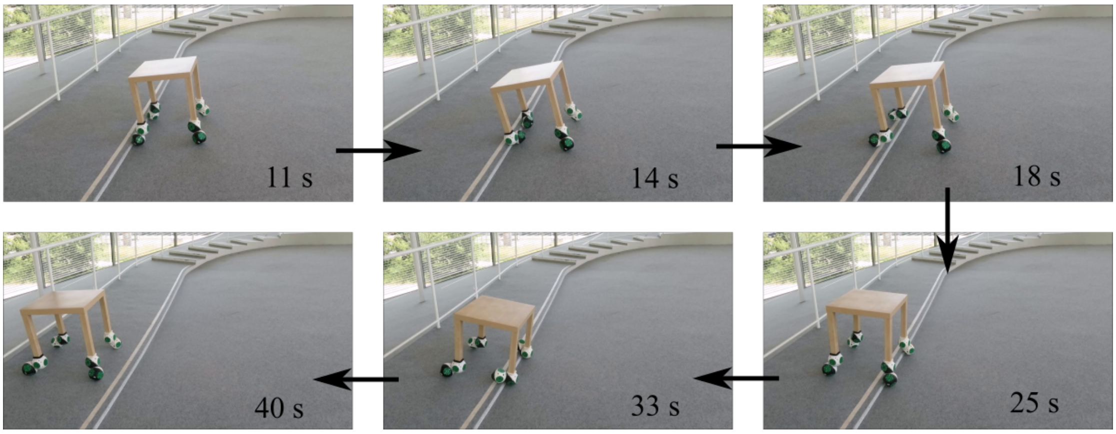


Figure 2.14 – Overcoming a ledge. A table drives until its front legs touch the ledge. The attached RB modules then perform a rotation around their attachment point, causing them to move on top of the ledge in the process. The same movement then is repeated for the hind legs after which the table has overcome the ledge.

puts it on top of the table which then rolls to the final location. Fig. 2.15 shows snapshots of this manipulation task.

Passing objects

One of the advantages of the Universal Gripper is that it can grip objects regardless of their shape and orientation (within a size limit given by the size of the gripper). This is promising for passing objects from one gripper to another since the passing sequence does not need to take the object's shape or orientation into account. In the framework of multiple RB furnitures collaborating with each other, such a scenario of passing an object between pieces of furniture can easily be imagined. We adopt this task in a simpler setup where an object is passed from one RB gripper module to another. The first module picks up an object (pen) from a tabletop and rotates its gripper to face the gripper of the second module. In this position, it is important to notice that simply making the two grippers touch orthogonally causes issues for the second gripper to actually grip the object as it would require the surface on the object that is currently occupied by the first gripper. There are several ways to deal with this issue; here we present the preliminary results of inducing a "shift" such that the second gripper touches a free part of the object (this also requires that the object is somewhat elongated to possess such a free part). The second gripper then grips the object upon which the first gripper releases, transferring the object to the second gripper. Fig. 2.16 is showing snapshots of transferring a pen from one to another module.

Even though we only show one example of such mid-air sensor-less passing of objects between two modules, we see much potential in using this unique way of object transfer in a collaborative environment as depicted in Fig. 2.1.

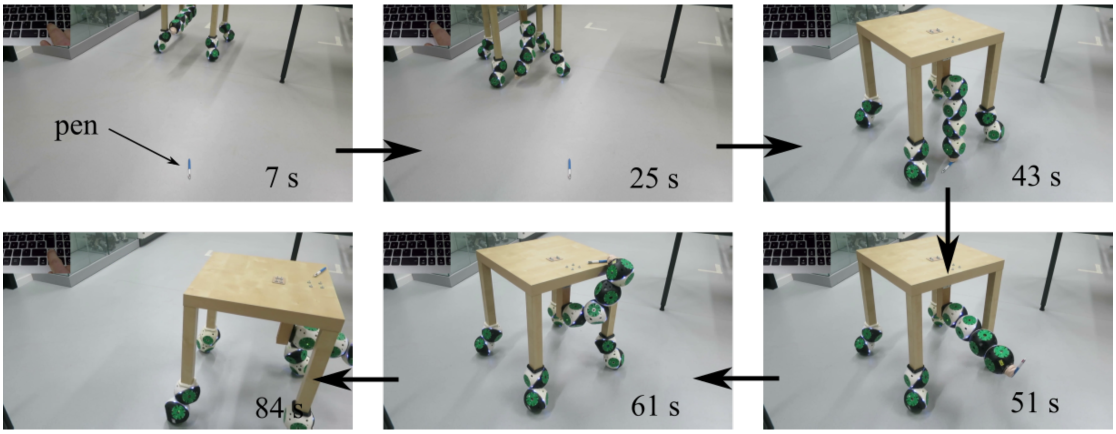


Figure 2.15 – Table picking up an object. An assistive table with manipulating capabilities moves towards a pen dropped on the floor. It picks up the pen with a Universal Gripper, places it on top of the table and brings it to the user.

Opening water bottle

The third demonstration in this task concerns a more assistive subtask of manipulation. Here, we briefly validate if a gripper structure can help with opening a PET water bottle. A user has to hold the bottle and push it into the gripper of a waiting RB structure. The gripper then actively grips the cap of the bottle and performs a rotation to open the cap. Once opened, the user can remove the bottle and the modules place the cap on the table (Fig. 2.17).

Object manipulation capabilities of SRMRs

Almost all SRMRs have means of connecting and detaching to each other. Some use magnetic attraction whereas a big portion of them use mechanisms. For example, the Active Connection Mechanism (ACM) is the mechanical latch Roombots use for the purpose of connection. In theory, this connection mechanism - and similar ones in other systems - could also be used for object manipulation. However, most of the times these self-reconfiguration methods are not suitable for manipulating arbitrary objects. It is possible to design a gripper module as an extension for almost all of the SRMRs similar to [Brandt et al., 2007]. Also, some of the SRMRs have already integrated gripper modules such as [Mondada et al., 2005] and [Gross et al., 2006]. These existing grippers are similar to conventional manipulator end effectors. An integrated universal manipulator is demonstrated for the first time with this work.

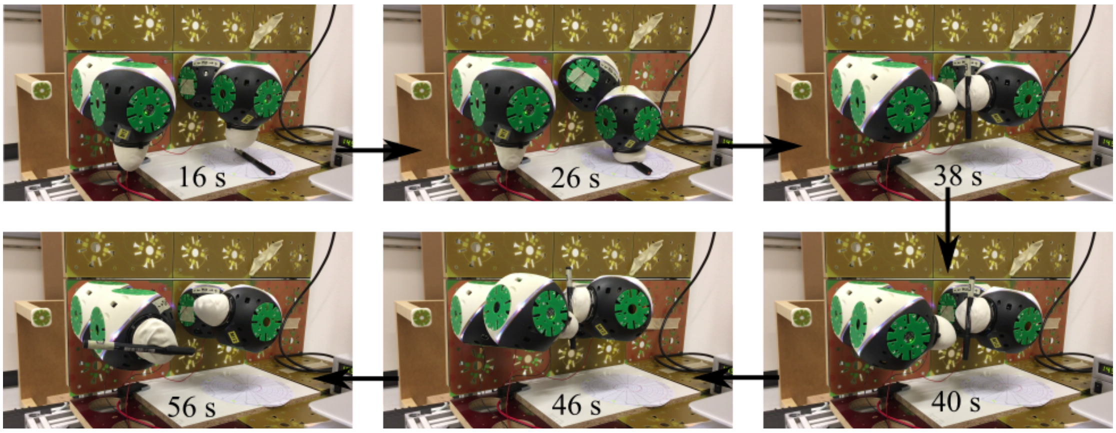


Figure 2.16 – Two RB gripper modules passing a pen in mid-air. The first module picks up the pen from the table and orients it to face the gripper of the second module. A shifting motion then takes place such that the second gripper has a free part of the pen to grab onto. The first module then presses the pen into the gripper of the second module and releases it after the pass is completed.

2.4.5 Interactive furniture

Kinect tracking control

We can couple movement control of RBs with the tracking abilities of the Kinect. Using depth information and the body segregation method of the Kinect, we can define intuitive hand gestures for certain commands. Such a user interface was explored in [Ozgur et al., 2014], however not in real-time. There, a user would first point to an RB module and then to a desired goal position, and a planning algorithm then computes and executes a movement sequence. Here, we use real-time inverse kinematics to convert the position of a users hand, tracked by the Kinect, into movements commands to make a meta-module follow the hand. The top row of Fig. 2.18 gives an example of this interaction.

LED capabilities

At last, we showcase LED lighting capabilities as an interaction method. Each module contains two rings of 6 RGBW-LEDs that can be used to e.g. display the state of a module to a user or indicate if an ACM is open or closed. Additionally, one module possesses a powerful RGBW-LED that can be used as a spotlight to illuminate a specific region (as already discussed in [Mutlu et al., 2016]). In the bottom row of Fig. 2.18, some lighting examples are shown.

2.5 Discussions and future work

The presented hardware demonstrations showcase the capabilities of the upgraded Roombots modules. We are aware that the outcomes of the demonstrations are rather of qualitative

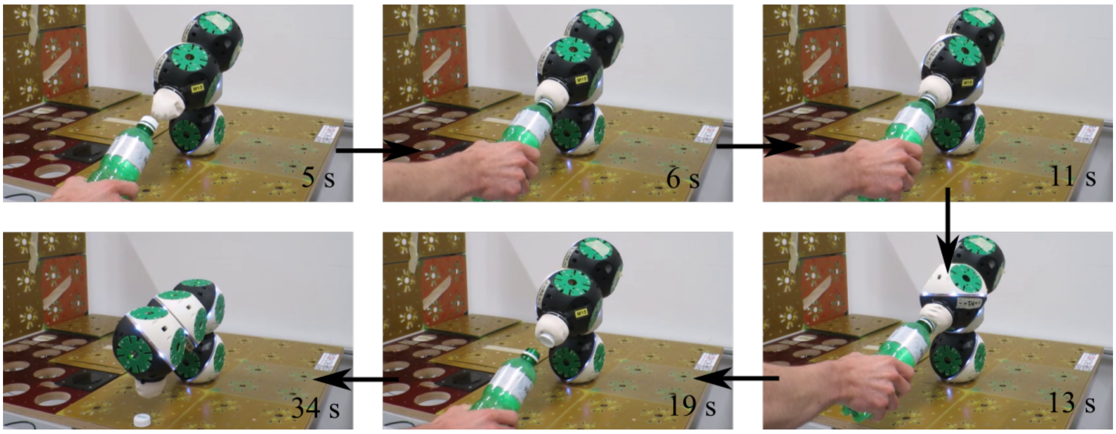


Figure 2.17 – A gripper metamodule assisting in opening a PET water bottle. The user pushes the bottle into the gripper which grips the cap and then performs a rotating movement to open the bottle.

than of quantitative nature as this chapter aims at exploring a potential use of (SR)MR for adaptive and assistive furniture. In this context, we successfully presented proof-of-concept demonstrations of RB modules completing a large and complex self-reconfiguration task involving a significant number of autonomous connections with minimal human assistance. Further, RB modules are used to create mobile furniture that can follow and evade users, adapt to the environment and can overcome obstacles. Manipulation tasks involved RB modules picking up an pen, opening a PET bottle and a proof-of-concept demonstration of passing objects between two modules. At last, RB modules possess various LED lighting capabilities. We have showed examples of how to use simple gestures to control an RB metamodule and developed a dedicated GUI for an easy monitoring of modules and creation of demonstrations.

The challenges for the presented experiments were to increase the connection reliability and to develop an interface to quickly create demonstrations with a large number of modules. For the reliability, on the one hand we could significantly improve the movement precision of all DoFs by changing the materials of the main gearbox and ACM from plastic to aluminum and by integrating a new absolute encoder. The increase in weight was compensated for with stronger motors. On the other hand, reconfiguration sequences now use a closed-loop controller to better align an ACM with a neighboring plate thanks to the integration of infrared and hall-effect sensors. Additionally, a ring of permanent magnets locally helps to form a connection while retaining the hermaphroditism of the ACM. Concerning the interface, the developed GUI contains all the key functionalities to rapidly develop Roombots scenarios. It allows for a safe creation of motor sequences in its virtual mode, and due to the improved connection reliability, the switch from virtual to real modules has a high chance of correctly performing the demonstration despite the disturbances present in the real world.

While a single module now works effectively, reliable position control of multiple modules in series remains a challenge. With Roombots, an example are the necessary more complex on-

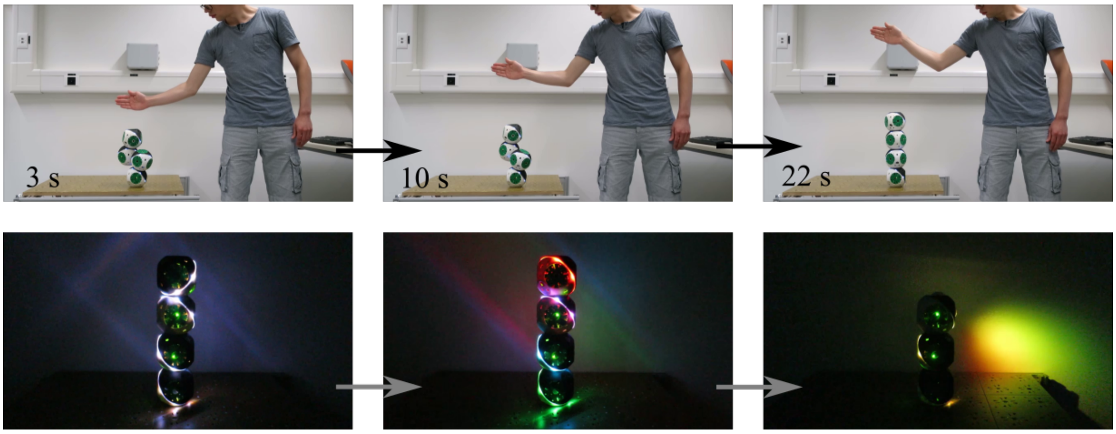


Figure 2.18 – Interaction with tracking and lighting. Top row: a users hand is tracked with the Kinect sensor and its position converted into inverse kinematic commands that make the meta-module follow the hand. Bottom row: lighting capabilities of Roombots with the integrated LED-rings and the powerful RGBW-spotlight.

grid locomotion capabilities of a meta-module to perform a self-reconfiguration as in task 1, for which the capabilities of a single module are too limiting. Even though the mechanics have been upgraded to cause less gearbox backlash, there are two main sources of misalignment that hinders the autonomy of this functionality: (i) the connection between two modules possesses unidentified mechanical play, and (ii) two modules in series still cause significant elastic deformation of the module shells.

For the experiments in this chapter, only the chair reconfiguration task involved meta-module reconfiguration where human assistance thus was needed in a few cases. Nevertheless, we regard the successful demonstration of the autonomous chair formation using 12 modules as a significant milestone in Roombots project. It validates both the capabilities of the hardware and the self-reconfiguration framework. It should be noted that relatively minor human intervention during the hardware demonstration was needed despite the efforts for autonomy: 3 manual interventions were needed in 107 executions of motor commands and connection or deconnection events (of which 16 events are independent connections only). In particular, the random local search algorithm to align the ACMs does not handle all possible cases equally well and may not converge because the positions of the two connecting plates are initially too misaligned due to elastic deformations and mechanical play. As a result, the local motor actions caused by the algorithm do not allow a meaningful quantitative comparison of the sensory feedback to steer the connection to a more aligned state, in which case the external assistance was needed. A more elaborate algorithm together with a proper modelling of all physical effects could be able to bring the reconfiguration to true autonomy. However, the Roombots robotic platform in its current form has rather reached an upper limit in terms of complexity and weight, making it difficult to investigate such research. Further hardware validation would thus likely require a new platform which could be part of future research in the topic of SRMRs.

Another major future work is the integration of additional structural passive parts. In the full Roombots vision, furniture consists not only of RB modules but also of lightweight structural parts that will help to reduce the weight of the formations and allow the creation of larger pieces of furniture such as a table. The addition of such passive parts requires a close collaboration of RB modules to position them appropriately in a structure for which new reconfiguration algorithms will have to be developed and tested, such as e.g. in [Bonardi et al., 2013].

By operating the upgraded Roombots design at the limitations of the platform through successfully running the many different sub-tasks, we could identify the following additional hardware and control challenges: (i) exploration of machine vision capabilities (such as e.g. in visual servoing for autonomous docking) to allow modules to perceive their environment, (ii) autonomous adaptation of morphology according to the task, either with visual feedback and/or other means, (iii) exploring safe human-module interaction for a safe integration of modular robots into our living space, (iv) improved autonomy and self-reconfiguration performance through deeper research on AI methods, potentially with a new design of an SRMR.

The continuous development of SRMRs gives us insights into current challenges and limitations of the general concept of reconfigurable systems. In the long term, we could imagine the technology being used in many more applications, e.g. reconfigurable factory lines where it could be used both for reducing factory space and manufacture costs through machine reconfiguration, or futuristic visions of reconfigurable satellites, exploration space robots or even space stations where autonomous reconfiguration systems could prove invaluable for their versatility and robustness. This work is another steppingstone towards the grand vision of SRMRs.

2.6 Conclusion

In this chapter we presented the capabilities of the latest generation of our self-reconfigurable modular robotic system “Roombots”. We outlined five key tasks that we consider relevant to the vision of modular robotics for adaptive and assistive furniture and were able to successfully demonstrate various sub-tasks in hardware. This required significant design improvements - especially in the mechanics of the modules - which are described in detail.

Concerning the demonstrations, modules performed a large-scale self-reconfiguration into a chair and provided mobility capabilities - such as following and evading a patient and overcoming obstacles - to off-the-shelf pieces of furniture by being used similarly to omnidirectional wheels. Specialized gripper modules were used to demonstrate basic manipulation capabilities such as picking up a pen, and could show the passing of objects between two modules. Interaction functionalities with colored LED and with the use of a Kinect depth camera were presented and we discussed the development of an easy-to-use Graphical User Interface (GUI) to control and monitor groups of modules.

Much of the presented results are of qualitative nature with a set of Roombots modules achieving a defined sub-task. Our goal in this chapter was not to optimize a single task leading to quantitative data to analyze, but to show the versatility of our SRMR system in many different scenarios. We were interested in the capabilities as well as limitations of the current iteration of our system to build new experience in hardware experiments that can be used for future development of this or another SRMR, creating a benchmark of demonstrations that other systems can compare to. Even though some of the demonstrations were partly controlled by a human operator, autonomy is only one aspect of SRMRs and all of the presented demonstrations have been crucial milestones for this project.

While the majority of the presented demonstrations is focused on adaptive and assistive furniture, the tasks in general share the main challenges present in generic SRMR functionalities in mechanics (connection, disconnection, movement, alignment), electronics (autonomy, communication, sensors), software (control, computation, robustness, safety) and interaction (user interface, user feedback). Roombots as a system presents one approach in tackling these various and interdisciplinary challenges, which could inspire other systems with the solutions described here, and we hope that this work is stimulating the general field of moduluar robots.

As for the future of the Roombots project, we envision the design and integration of lightweight passive, structural parts that can be used in conjunction with active RB modules to form larger furniture for our everyday environment. New collaboration algorithms (i.e. adapting existing control framework or creating a new distributed control framework) will need to be developed, especially regarding safety when interacting with modules, coming one step closer to the integration of such robotic systems into our living space.

3 An optimal planning framework to deploy SRMR

Ch. 2 presented a brief introduction of hardware and control improvements on RB. However, autonomous planning for self-reconfiguration is a highly challenging task mostly due to the sheer size of the inherent high dimensionality of SRMRs and unusual orientation of the exes of rotation. The problem is highly nonintuitive for humans as it poses sort of a puzzle. Luckily, self reconfiguration of SRMRs can be formally represented as discrete states and well-defined actions causing states to evolve under certain assumptions (i.e. restricting the motion capabilities to certain discrete angles and using SRMRs on an engineered discrete environment). Thus, self reconfiguration can be formulated as a conventional search problem where computers are much better than humans for crunching numbers. In this chapter, we present further details of the planning framework introduced in Ch. 2. The framework has been shown to work for Roombots and in principle it can be adopted to other SRMRs with minimal changes. Chronologically, this work has been completed in a later phase of this thesis. Otherwise, it could have been integrated in Ch. 4 and Ch. 6 to boost the autonomy of self reconfiguration.

Reference publications

- This chapter is based on *Hala Khodr, **Mehmet Mutlu**, Simon Hauser, Alexandre Bernardino and Auke Ijspeert. "An Optimal Planning Framework to Deploy Self-Reconfigurable Modular Robots ."* Submitted to IEEE Robotics and Automation Letters.

  My original contributions
  - Definition and supervision of MS thesis project of Hala Khodr
  - Building the required electronics (Sensor plate and readout boards)
  - Providing the base code for the framework
  - Guiding and planning of experiments
  - Performing experiments
  - Reviewing the manuscript

3.1 Introduction

Self-reconfigurable modular robots (SRMRs) are “Swiss Army knives” of robotics. A set of robotic modules are expected to perform a great variability of tasks by adapting to changing demands. The task space flexibility of SRMRs arises from the high number of degrees of freedom (DOF) since each active module introduces at least a DOF to the system. On the other hand, the same high dimensionality can be a curse too. Self-reconfiguration is proved to be a NP complete problem for chain type of SRMRs by [Hou and Shen, 2010] and [Hou and Shen, 2014] which renders conventional search methods impractical for large DOF SRMR systems.

In order to overcome the limitation of search space explosion, [Arancha Casal, 1999], [Unsal et al., 2001], [Yoshida et al., 2002] and [Tosun et al., 2018] introduce divide-and-conquer methods to reduce the problem into smaller substructures. Another way to overcome the high dimensionality is introducing randomness is the search algorithm. For instance, [Brandt, 2006] uses RRT-connect on ATRON to show possible benefits of randomness in hard problems. Yet another approach is to efficiently restrict the motion capabilities of robotic modules for computational benefits as shown in [Larkworthy and Ramamoorthy, 2010]. Further extensive surveys on control strategies proposed for SRMRs are presented [Ahmadzadeh and Masehian, 2015], [Stoy et al., 2010] and [Kotay and Rus, 2000].

In [Fitch and Butler, 2008], the million module march algorithm was introduced as a fully decentralized path planning using a Markov Decision Process (MDP) for sliding cube module abstraction. It is extended in [Fitch and McAllister, 2013] to include kinematic model for a physical module and it was applied on Superbot [Salemi et al., 2006].

Although, for certain type of SRMR (near) optimal distributed control methods exist, in most of the SRMR systems search-based centralized methods are needed for complete and optimal solutions. By optimality, we denote that the algorithm finds the minimum number of steps among all possibilities from start to goal. By completeness, we imply that it guarantees to find a solution if one exists.

So far, Roombots is more commonly used for free space locomotion experiments, even though, on-grid (lattice) self-reconfiguration is studied to a certain extent [Sproewitz et al., 2010a],[Bonardi et al., 2012b]. In the locomotion through reconfiguration approach introduced in [Bonardi et al., 2012b], only one RB module is moving; furthermore, it is neither allowed to manipulate another one nor interact with other modules. This limits the possible reachable positions when encountering cases that one module alone cannot overcome, for instance a gap. When more modules interact with each other, this becomes a subset of the self-reconfiguration problem. [Sproewitz et al., 2010a] proposes a gradient based navigation for Roombots. The basic motion units considered in [Sproewitz et al., 2010a] are meta-modules (structure formed of two modules connected together). This choice considerably augments the reachable space of one basic unit; however it also increases the difficulty of manipulation with the increase of the degrees of freedom, since more combinations of motor actions are possible to reach a desired position. One major issue in [Sproewitz et al., 2010a] is the inability to guarantee the completeness of the process because of local minima.

The motivation of this chapter is to create a complete framework that can guarantee a solution, provided that it exists, to move RB modules from initial poses to desired goal poses. The smallest unit of the system is required to be a single RB module and collaboration between modules must emerge whenever needed. Such planning capability is essential to unlock the full potential of Roombots in self-reconfiguration, self-assembly, self-disassembly, locomotion and flow applications. The framework should take into account all physical capabilities and limitations of real RB modules in lattice environment (for instance alignment, maximum weight, motor rotation, work space..). Thus, the generated plan should be executable in real hardware.

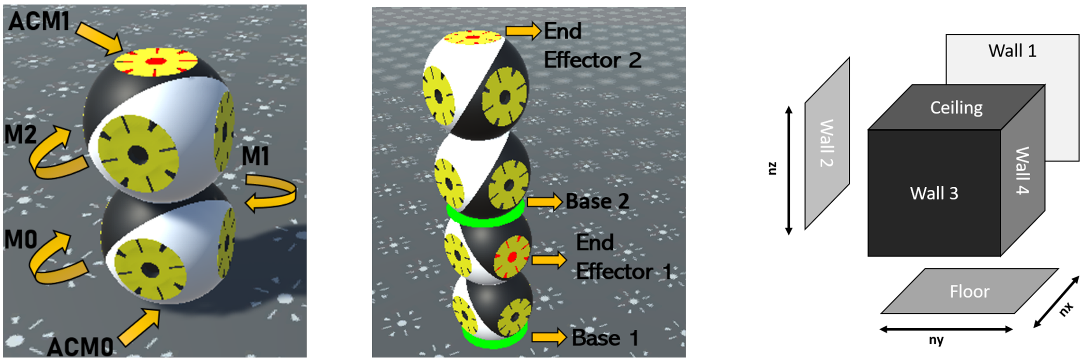
In this chapter we propose an optimal planning framework for RB based on conventional search algorithms. We compare blind and informed search methods while incorporating crucial pruning methods and offline steps to accelerate the search to a point where we can always obtain a solution for configurations involving up to a few RB modules (double digit DOF) in medium-sized or restricted workspaces where gradient based methods fail. Thus, we have a kernel functionality which can later be used in higher level planners to overcome larger problems.

Contributions of this chapter are four-fold: (i) deriving an abstraction layer which is a novel transition model from a configuration to another one, (ii) introducing a novel RB module specific search heuristic which can be adopted to other SRMRs, (iii) complementary generic methods based on look up tables and pruning rules to accelerate the search and (iv) comparative study on trade-offs of different search methods in terms of complexity, completeness, optimality and execution speed.

The rest of the chapter is organized as follows. Sec. 3.2 gives the notation, Sec. 3.3 explains our formulation of self reconfiguration as a search problem, Sec. 3.4 compares different conventional search algorithms on the self-reconfiguration problem, Sec. 3.5 describes hardware considerations for the plan to be deployable in the real world, Sec. 3.6 briefly demonstrates the framework on real hardware and Sec. 3.7 concludes the chapter.

3.2 Terminology

A **Roombots (RB) module** is the basic robotic structure in this work. Each RB module has three rotational motors (M0, M1, and M2) and two active connection mechanisms (ACM0 and ACM1) that allow it to attach to the grid, to other modules or to any passive surface equipped with the corresponding connection plate. Fig. 3.1a illustrates these components. Each RB_i configuration, denoted C_i, can be uniquely abstracted to C_i=$\{H_{ACM_0}, H_{ACM_1}, S_{ACM_0}, S_{ACM_1}\}$ where H_{ACM_i} is the homogeneous matrix that represents the position and orientation of ACM_i in 3D space and S_{ACM_i} is the status (open or closed) of ACM_i. We define a **RB meta-module** the structure composed of two RB modules connected together to form a 6 DOF system as seen in Fig. 3.1b. We consider our **environment** as a confined 3D space modeled as a box with width n_x, length n_y and height n_z as illustrated in Fig. 3.1c. The 6 sides of the box can have **fixed supports** with passive connectors to which RB modules can attach. We create a **3D occupancy grid** where each occupied voxel represents one sphere of an RB module. We denote the number of RB modules present in the environment with N.



(a) (b) (c)

Figure 3.1 – (a) Single RB module components (b) RB meta-module example(c) Environment Abstraction

3.3 Problem formulation as search

The goal of our framework is to compute the sequence of RB actions from an initial (I) to a goal (G) state. By considering our task as a problem-solving agent, it is formally defined by five main components [Russell and Norvig, 2009]:

The initial state that the agent starts in

Any arrangement of the N RB modules in a given environment is a state S={$C_0, C_1, .., C_N, O$} where C_i is the RB configuration of each module RB_i, O the resulting 3D occupancy grid.

The possible actions (a) available to the agent given a particular state

We map the actions to the motors' motion but restrict them to discrete movements. In other words, at each state, there are mainly 9 actions per module:

- Rotate M0 to angle position −120°, 0°, or 120° (2 possible actions).
- Rotate M1 to angle position −90°, 0°, 90° or 180°(3 possible actions).
- Rotate M2 to angle position −120°, 0°, or 120°(2 possible actions).

- Toggle (close if open and vice versa) each ACM (2 possible actions).

The transition model that determines the resulting state from the current one after doing a specific action a

We abstract one RB module to a set of joints and links and use the popular Denavit-Hartenberg (DH) convention to attach the reference frames. We consider the fixed ACM as base (b) while the other ACM as end-effector (e), and the 3 motors as revolute joints. Using the DH parameters, one can derive the homogeneous transform matrix T_i^{i-1} specifying the position and orientation of the i^{th} frame with respect to the previous $(i-1)^{th}$ coordinate frame. Then, the position and orientation of e with respect to b can be computed as

$$T_e^b = T_0^b \cdot T_1^0 \cdot T_2^1 \cdot T_e^2 \tag{3.1}$$

where 0, 1, 2 refer to the frames attached to the revolute joints. One final multiplication is needed to know the absolute position and orientation of e in the world frame :

$$H_{ACM_e} = H_{ACM_b} \cdot T_e^b \tag{3.2}$$

In the case of a meta-module, we will further need to update the state of the second (manipulated) RB module. The base and end-effector of the first and the second modules (b_1, e_1, b_2 and e_2) are illustrated in Fig. 3.1b. Since the homogeneous matrix ($H_{ACM_{e_2}}$ and $H_{ACM_{b_2}}$) for each ACM of the second RB module before action a is known, the transformations between e_1 and b_2 (T_{b2}^{e1}), as well as between e_1 and e_2 (T_{e2}^{e1}) are fixed and they can be calculated as:

$$T_{e2}^{e1} = (H_{ACM_{e_1}})_{current}^T \cdot H_{ACM_{e_2}} \tag{3.3}$$

and

$$T_{b1}^{e1} = (H_{ACM_{e_1}})_{current}^T \cdot H_{ACM_{b_2}} \tag{3.4}$$

Finally, the homogeneous matrix for each the ACMs of the second RB module after applying an action a can be calculated as:

$$(H_{ACM_{e_2}})_{new} = (H_{ACM_{e_1}})_{new} \cdot T_{e_2}^{e_1} \tag{3.5}$$

and

$$(H_{ACM_{b_2}})_{new} = (H_{ACM_{e_1}})_{new} \cdot T_{b_2}^{e_1} \tag{3.6}$$

where *current* denotes the instance before applying a, and *new* the one after applying it. Lastly, knowing the positions of the bases and end-effectors, the 3D occupancy grid is updated by computing the new voxels occupied given this action.

The goal test that determines whether a given state is the goal

we compare the 3D occupancy grid of the current state and that of the desired goal state. We note that the goal state can be any distribution of RB modules, e.g. a complex structure or disconnected modules distributed in space.

The step cost of taking action a in state s to reach state s'

we assume each step cost equals to +1.

3.4 Searching for solutions

Having formulated the problem, the next step is to find a solution, i.e. a series of actions to go from state I to state G. Starting at the initial state I, the possible action sequences form a search tree with I at the root, the branches are the actions and the nodes correspond to states in the search space of the problem. The strategy adopted for searching follows the flowchart shown in Fig. 3.2. We highlight two of its main blocks throughout the rest of this section: (i) how to select the next node from the search list, and (ii) how to prune some branches.

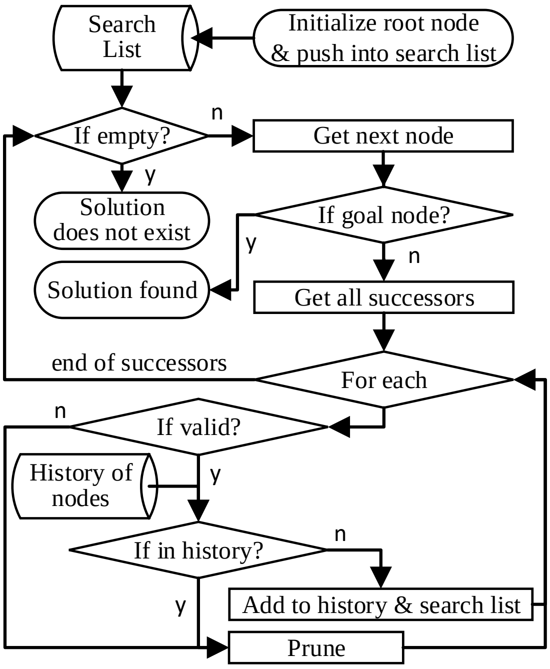


Figure 3.2 – Flowchart for a general search algorithm

3.4.1 Search algorithms: Expansion of the search tree

We keep track of nodes that have been generated but not yet expanded in a *search list.* The selection of the next node from the *search list* depends on the search algorithm implementation. We consider and compare four approaches: A* search, bidirectional A* search, breadth-first search (BFS) and depth-first search (DFS). Only the bidirectional search has one additional layer: two searches are initiated in parallel, one forward tree with the root being the initial state and another backward tree with the root being the goal state. When two branches from each search tree meet, the algorithm converges. In this case, it is important to highlight two points: (i) in addition to specifying the goal 3D occupancy grid, we also define the configuration of the RB modules at the goal state to be able to consider it as an initial state for the backwards tree, and (ii) the check goal stage will not only include the goal test but an additional test to check if the current node is in the history list of the other tree.

Heuristic choice for A*

Estimating the number of actions between two states is a very challenging task. One approach is using machine learning algorithms to create effective heuristics in large search spaces as in [Jabbari Arfaee et al., 2011]. Although it is interesting to investigate this approach, it does not guarantee admissibility. In [Asadpour et al., 2009], they introduce a heuristic based on the graph edit distance between two graphs. However, this chapter is limited to closed chain reconfiguration. The adopted approach here is inspired from the optimal assignment metric introduced in [Pamecha et al., 1997]. We explain in what follows our strategy. Since the goal test is comparing current and goal occupancy grids, the heuristic function should have a measure of the path cost between two 3D occupancy grids. Given a 3D occupancy grid in the current state O_c and the goal 3D occupancy grid O_G, we find a bijection between the $2N$ occupied voxels in O_c and the $2N$ occupied voxels in O_G which minimizes the sum of all voxel-to-voxel costs using a minimum-cost-maximum-flow approach [Edmonds and Karp, 1972]:

$$\alpha_G = \arg\min_{\alpha} \sum_{i=1}^{2N} f(g_\alpha(i), p(i)) \tag{3.7}$$

where α is the assignment function, f is the voxel-to-voxel cost function $p(i)$ is the position of voxel$_i$, $g_\alpha(i)$ is the position of the assigned goal voxel$_i$. And the final heuristic will be equal to

$$h = \left\lceil \sum_{i=1}^{2N} f(g_{\alpha_G}(i), p(i)) \right\rceil \tag{3.8}$$

The voxel-to-voxel cost f is calculated as the sum of distances defined by neighborhood sequences. In other words, starting from the given voxel, we specify the set of reachable voxels by only one action: we denote this set as the voxel neighborhood. The smallest number of neighbors traversed to reach a goal determines the voxel-to-voxel cost. We considered two types of voxel neighborhood:

1. “26-neighborhood” (8-neighborhood in 2D) illustrated in Fig. 3.3a. Mathematically, this can be calculated as

   $$f(g_{\alpha_G}, p) = \max(|x_G - x_p|, |y_G - y_p|, |z_G - z_p|) \tag{3.9}$$

   where (x_p,y_p,z_p) and (x_G,y_G,z_G) are the coordinates of the current and goal voxel positions respectively.

2. Taking into account the capabilities of RB modules, we devise a new “RB neighborhood" representing the reachable space of one voxel in one action with a RB meta-module (Fig. 3.3b). For faster computation, we pre-calculate the cost between any two voxels using this neighborhood given the environment defined by n_x, n_y and n_z and save it in a 3D array denoted RBN. Moreover, since a single RB action can move up to three voxels, we divide the cost by 3 in order not to overestimate the actual path cost. Mathematically, the cost corresponds to

   $$f(g_{\alpha_G}, p) = \frac{RBN(|x_G - x_p|, |y_G - y_p|, |z_G - z_p|)}{3} \tag{3.10}$$

   where (x_i,y_i,z_i) and (x_G,y_G,z_G) are the coordinates of the current and goal positions respectively.

The 26-neighborhood breaks the admissibility of the heuristic function since it doesn’t take into account the possibility of forming a meta-module leading sometimes to an overestimated cost. The RB-neighborhood, on the other hand, guarantees the admissibility of the heuristic, and therefore the optimality of the solution. Although this heuristic is admissible, it sets the path cost low, leading to slow convergence. For this reason, we relax the restriction of optimality constraint in favor to a faster convergence and also consider the 26-neighborhood heuristic, not-admissible in this case. In conclusion, we implement and compare two versions of A*: the first with admissible heuristic using the RB neighborhood (denoted as A*) and the second with not admissible heuristic using 26-neighborhood (denoted as A* nah).

3.4.2 Pruning methods

Since the branching factor of the search space is very high, it is important to prune some actions to avoid exploring not promising branches. We apply two methods for pruning branches during the search: (i) validity check and (ii) history keeping.

Validity check

We define not valid actions as:

- ACM closing action when no valid connector plate is present. Although this action is

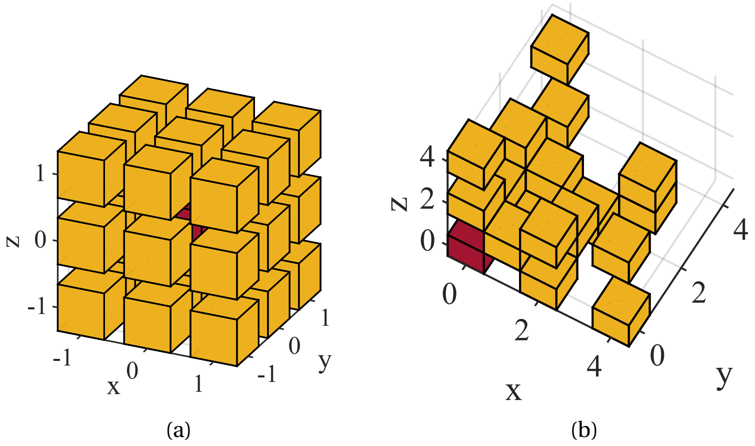


Figure 3.3 – Different voxel neighborhoods: (a) 26-neighborhood and (b) one octant of the Roombots neighborhood. The darker color indicates the voxel in consideration. The voxels with lighter color represent the reachable space in one action.

physically possible, it is actually not effective, so we prune it.

- ACM open action resulting in floating (not connected) RB modules.
- ACM open action resulting in heavy weight (more than 2 RB modules) on a single module.
- Blocked motor actions when the kinematic chain is closed by being attached to a fixed support from both ends.
- Motor actions resulting in a collision.

In order to fast compute if an action results in a collision, we devise a collision-check database (offline) containing the occupied voxels during each motor movement for every RB configuration. The action is considered valid only if these voxels are empty in the 3D occupancy grid. For each motor action, the collision database is implemented as a sorted list with unique keys representing the RB configuration. The list is loaded only once at the start of the program, and used only for look-up during the program execution which is only an $O(log(n))$ operation where n is the length of the list.

History keeping

Keeping track of a history and performing similarity checks with newly considered nodes is crucial to avoid going in unnecessary loops. Similar to collision lists, the history is also implemented with a sorted list with unique keys generated for each state. We consider two ways to check the similarities between a current state and one in the history.

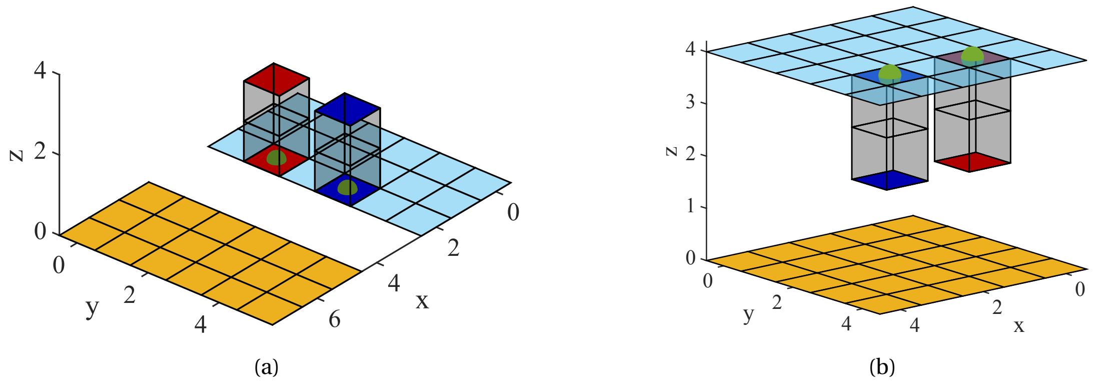
- The complete way is to compare both the ACMs positions and orientations of each RB module between the two states. We denote it as *with orientation (w. ori.)*.

- The incomplete way to check state similarity is to only compare the ACMs positions of each RB module between the two states. We denote it as *without orientation (w/ ori.)*.

The second option (*w/ ori.*) is justified by the fact that many paths can exist between two states due to redundancy. It is also motivated by the desire to decrease the convergence time by reducing the effective branching factor with the price of giving up completeness and optimality. It is important to note that with a bidirectional search the second option (*w/ ori.*) approach cannot be applied since when the two branches meet, RB modules need to have the same RB orientation for the sequence to be feasibly connected.

3.4.3 Framework performance analysis

We consider two test cases as shown in Fig. 3.4 to compare the performance of the search algorithms implemented.



(a)

(b)

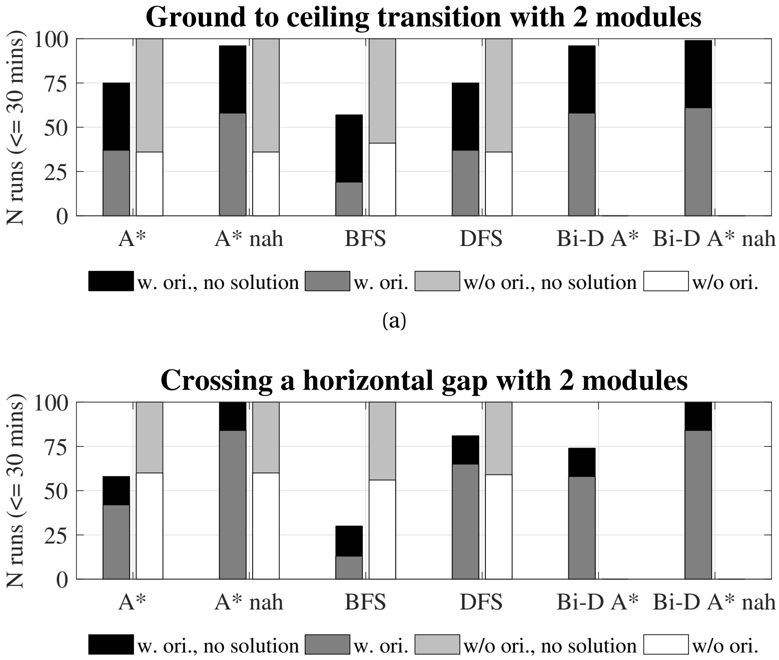
Figure 3.4 – Goal state of the two test cases for algorithm comparison. (a) Crossing a horizontal gap with 2 RB modules in a 6x8x4 environment where all the sides except the floor are disabled. The middle cells of the floor are disabled emulating the presence of a gap. The goal state are 2 RB modules in the positions shown irrespective of their orientation. (b) Ground to ceiling transition with 2 RB modules, in a 5x5x4 box where the side walls are disabled. The goal state is 2 RB modules on the ceiling in the positions as shown irrespective of their orientation. In both cases (a) and (b), the initial state is a random placement (pose) of the 2 RB modules on the yellow area.

We compare 6 algorithms explained (A*, A* nah, Bidirectional A*, Bidirectional A* nah, BFS, and DFS) with and without orientation history keeping when feasible. For each test case and for each algorithm, we repeat the simulation 100 times with random initial states using the same seed for the random number generator to guarantee a fair comparison. We also introduce a timeout of 30 minutes to avoid very long runs. Computations are done using a

PC with an Intel Core i5-750@2.67GHz and 12 GB RAM. Finally, we extract 5 main features to compare the considered algorithms:

The total number of solved problems within the limit of 30 minutes

we further categorize how many of them converged to a found solution, and how many converged to no existing solution as shown in Fig. 3.5. By comparing the total number of solved problems, we notice that in both test cases, all the runs with the approach “w/o ori." resulted in 100% convergence to a solution in less than 30 minutes. Whereas, with to the approach “w. ori.", only A * and bidirectional A* (non-admissible) converged to 100% in the gap test case, and above 95% in the ceiling test case. The worst result is the BFS with orientation algorithm with a score of only 30% in gap test case and 55% in ceiling test case.



(a)

(b)

Figure 3.5 – Comparison of number of problems solved within 30 minutes for (a) gap and (b) ceiling test cases.

The mean convergence time

execution times are reported in Table 3.1. BFS results in the longest convergence time. On the other hand, A* with non-admissible heuristic and without orientation approach has shown the fastest convergence time in both test cases.

Table 3.1 – Average convergence time of tested algorithms in [s] on both gap and ceiling test cases

	Gap test case		**ceiling test case**	
Algorithm	**w/ orient.**	**w/o orient.**	**w/ orient.**	**w/o orient.**
A*	671.5	48.9	441.5	261.0
A* nah	123.8	32.3	378.5	125.2
BFS	1049.2	139.1	924.0	196.7
DFS	327.6	107.7	224.8	238.1
BiD A*	584.8	NA	245.0	NA
BiD A* nah	85.3	NA	128.8	NA

The mean number of expanded nodes

we keep track of the number of nodes popped off the *search list* as an indication of the memory used. Results are reported in Table 3.2. It is important to note that the numbers given in the table are only valid and processed nodes. However, in total, we visit in the order of millions of nodes (about 5 millions), most of which are pruned.

Table 3.2 – Average number of expanded nodes of the tested algorithms on both gap and ceiling test cases

	Gap test case		**Ceiling test case**	
Algorithm	**w/ orient.**	**w/o orient.**	**w/ orient.**	**w/o orient.**
A*	168.4k	17.1k	43.3k	12.1k
A* nah	24.6k	4.6k	30.3k	3.7k
BFS	167.4k	24.9k	67.7k	24.1k
DFS	23.9k	11.8k	9.9k	7.1k
BiD A*	220.2k	NA	66.2k	NA
BiD A* nah	23.9k	NA	14.5k	NA

The mean optimality percentage

since the fastest solutions were found from algorithms that don't guarantee optimality(minimum number of actions), we compare how far the found solutions are from optimal ones. Results are reported in Table 3.3. The solutions found by A* “with orientation" are optimal by definition and are used as a baseline for comparison. We notice that all algorithms except DFS are near optimal with around 70 % of confidence for gap test case and 80 % of confidence for ceiling test case. Moreover, it is interesting to note that DFS appears to be way too far from optimal although its convergence time is competitive with the other algorithms: if we look at the number of actions in the solution path, we find it to be around 2000 steps. This comes from the fact that DFS always expands nodes along one branch until a solution is found.

Table 3.3 – Optimality percentage of tested algorithms on both gap and ceiling experiments

	Gap test case		Ceiling test case	
Algorithm	**w/ orient.**	**w/o orient.**	**w/ orient.**	**w/o orient.**
A*	100%	69.1%	100%	78.1%
A* nah	97.2%	72.6%	98.9%	77.7%
BFS	100%	65.7%	100%	74.9%
DFS	ϵ%	ϵ%	ϵ%	ϵ%
Bi-D A*	88.2%	NA	97.8%	NA
Bi-D A* nah	91.5%	NA	96.4%	NA

The completeness percentage

we calculate completeness as the percentage of convergence to not-found solutions where an actual solution exists if we use the “with orientation" approach. By this measure, we can see the effect of giving up the completeness of the similarity check on the completeness of the search algorithm. Results are reported in Table 3.4.

Table 3.4 – Completeness percentage of both gap and ceiling test cases when history list does not have orientation.

Algorithm	Gap test case	Ceiling test case
A*	93 %	83 %
A* nah	76 %	77 %
BFS	95 %	89 %
DFS	82 %	82 %

In the light of presented results, the strategy we propose to find solutions is to first run the fastest algorithm which is A* (nah) without orientation. Since completeness is not guaranteed with this algorithm, if solution was not found in a given limited time, we try the second fastest but complete search algorithm: bidirectional A* nah, or A* nah with orientation. For an optimal and complete search, we run A* with orientation.

3.5 Deploying the framework on hardware

The output of our framework is the sequence of actions leading from an initial to a desired goal state. However, to apply this sequence on real hardware, some considerations listed below should be taken into account.

3.5.1 Addressing ACM alignment problem

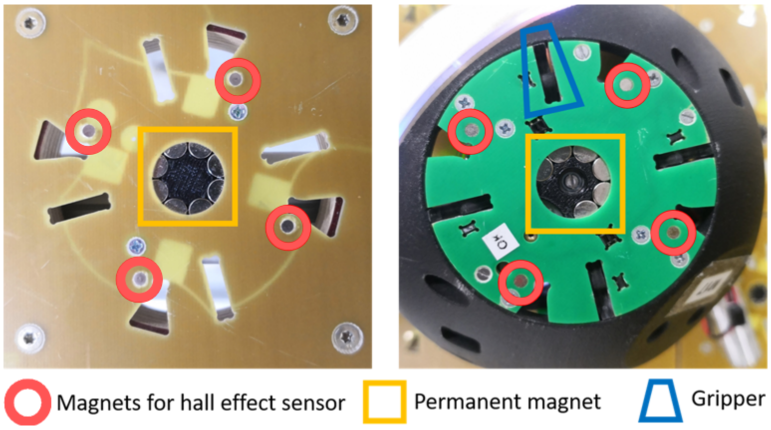
One of the most limiting challenges with on-grid locomotion using RB modules is the misalignment during the connection phase between the gripper of the ACM and the passive connectors on other modules or on the grid. Since the emergence of ACM misalignment is highly dependent on the real time experiment, it cannot be modeled and was not included in the transition model described in section 3.3. Therefore, the need for an active alignment mechanism arises.

Strategy

Each ACM plate has 4 pairs of hall-effect sensor and magnet. The four magnets are considered as the source of the magnetic flux density whose change can be detected by the 4 hall-effect sensors. On the other hand, each passive plate also has small magnets in the same positions as the ones on the ACM plate as shown in Fig. 3.6a. Therefore, if these magnets are aligned with the hall-effect sensors on the ACM side, the sensed magnetic flux density will be maximum. This design allows us to detect the presence of any misalignment before closing the ACM. Moreover, each ACM plate has also an accelerometer that can be used to detect its orientation. More specifically in our case, the accelerometer is used to calculate the initial pitch of the RB module when attached horizontally as illustrated in Fig. 2.6. This initial pitch can be obtained from the accelerometer data using the following formula:

$$\beta = arctan(\frac{-a_z}{\sqrt{(a_x)^2+(a_y)^2}}) \tag{3.11}$$

where a_x, a_y, and a_z are the accelerations measured around x-, y-, and z-axis respectively. Knowing the initial pitch β, we calculate the new desired angles using the inverse kinematics model to compensate for this initial rotation at the base of the RB kinematic chain.



(a)

Figure 3.6 – Description of the ACM and passive plate components.

Algorithm 1 Random Search

```
function RandomSearch($s_i$)
    $s \leftarrow s_i$
    while Hall-effect sensors above threshold t do
        Pick N random neighbors of s:
        $neigh=\{s_0, s_1, ..., s_N\}$
        for each: $s_j$ in $neigh$
            Evaluate its fitness function F($s_j$)
        $s_{new} = \arg\min_{s_j} (F(s_j))$
        $s \leftarrow s_{new}$
    Successful alignment : Quit
```

Pitch compensation alone is only valid when a RB module is attached horizontally and even then it is not always enough, requiring an additional local search. The strategy for the local search is to change the three motor angles in the neighborhood (+/- 1° for any motor) of the desired angles and use the value of the hall effect sensors as a feedback for the quality of alignment. Since this relation is highly non-linear, we chose a random search approach (Alg. 1). The fitness function is considered to be the sum of the sensor values. Three setups were tested: 1) going up when attached to ceiling, 2) going up when attached to a side wall, and 3) going left when attached to a side wall. For each setup, the experiment is repeated 10 times. We note that in these experiments, the success rate of a successful connection is 0% before introducing the plugin and 100% after the plugin with an average convergence time of 43.53 seconds. We also tried the simulated annealing search, but the mean convergence time increases to 106.48 seconds, so we proceeded with the random search choice. Alg. 2 depicts the overall strategy developed to actively compensate for any connection misalignment.

Algorithm 2 ACM Alignment Plugin

```
procedure ACM Alignment
    Go to desired motor angles
    if Hall-effect sensors below threshold t then
        Successful alignment : Quit
    else
        Perform pitch compensation (update desired motor angles with IK)
        if Hall-effect sensors below threshold t then
            Successful alignment : Quit
        else
            RandomSearch(motor angles)
```

3.5.2 Approaching/leaving concave corners

Approaching/leaving a concave corner with a single motor action would result in a collision. However, since the outer motors (M0 and M2) have the same velocity profile, this movement is possible by rotating both motors at the same time, which leads to a parallel behavior as shown in Fig. 3.7. On the other hand, allowing this behavior as a possible action in our framework would increase the branching factor. Instead, we consider the single action in case of approaching/leaving a concave corner valid and rather do a post-processing step in the final list of actions by replacing the single motor command (M0) by the equivalent collision-free parallel approach (simultaneous command for M0 and M2).

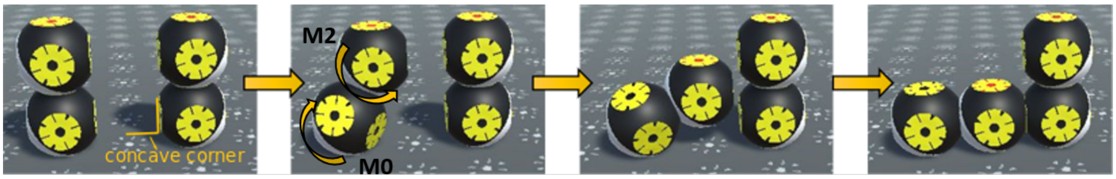


Figure 3.7 – Example showing the parallel behavior when approaching a concave corner

3.5.3 User feedback

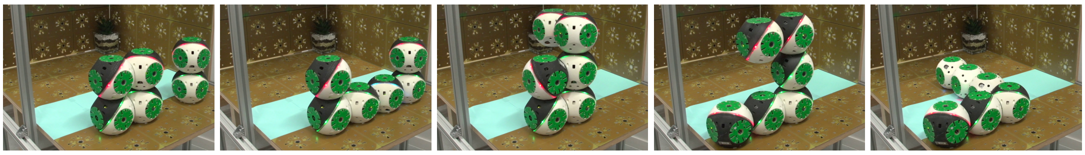
We add a visual feedback for the ACM states using the LED rings on the RB module: a red color indicates an open ACM, a green color a closed ACM, and a blue color an ACM in the alignment plugin. Lastly, all LEDs starts blinking violet when the goal is reached.

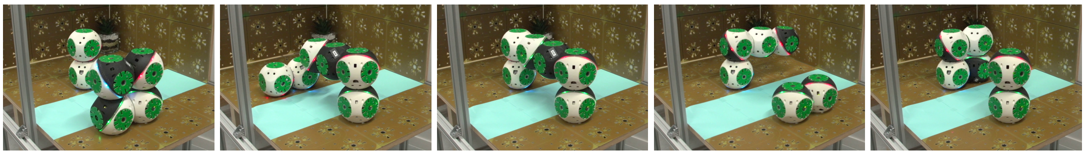
3.5.4 Physical limitations allowed in the framework

RB modules are approximately 1.6kg. Although a single module can dock successfully on every scenario on a rigid grid, a metamodule consisting of at least 2 modules has higher uncertainties at the end effector pose. Thus, it cannot always dock reliably. Nevertheless, we ignored this limitation in the proposed framework to have a more capable framework since future hardware iterations may overcome this challenge. During the real experiments, we allowed human intervention to help docking in such cases. The number of interventions depends on how many times such motions are needed. On average, there were two interferences per experiment in the scenarios shown in the video attachment.

3.6 Hardware validation

Six experiments with the real hardware were done in total. These were inspired by real life examples such as ants collaborating to cross a gap, or humans cooperating to overcome a high wall. Whereas the experiments are shown in the attached video, we discuss one of the experiments in this section: Crossing a horizontal gap with 3 modules. The algorithm used was *A* nah w/o ori.* from section 3.4. It was solved in 139.34 seconds and resulted in a sequence of 38 actions. We present the main stages of the reconfiguration and highlight the emergence of collaboration between RB modules to achieve the desired task in Fig. 3.8. In the first 5 stages (Fig. 3.8a-e), the modules demonstrate a collaboration so that one module will end up on the second side of the gap (Fig. 3.8f): one RB module manipulates another one to bring it to the other side, while the third RB module is moving away to guarantee a collision free movement for the formed meta-module. In the following 4 stages (Fig. 3.8f-i), we notice the formation of a bridge-like structure when one RB module "hands over" its neighbor to the RB module on the other side of the gap. In Fig. 3.8j, we see 2 RB modules already on the other side of the gap. In the last stages, the two RB modules on each side on the gap connect to form a meta-module and move to the other side. In Fig. 3.8n, all RB modules form a wall structure on the other side of the gap satisfying the goal target. The experiment took around 4 minutes and required a single human intervention.


(a) t=0s (b) t=24s (c) t=32s (d) t=40s (e) t=56s


(f) t=80s (g) t=104s (h) t=120s (i) t=144s (j) t=152s

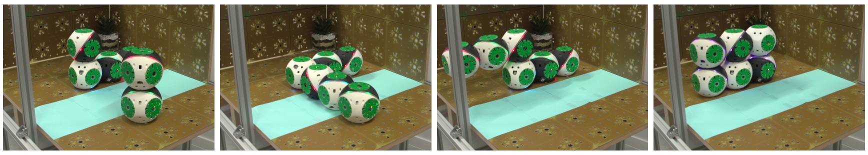

(k) t=168s (l) t=192s (m) t=224s (n) t=240s

Figure 3.8 – Different stages from the experiment of crossing a gap using 3 RB modules.

3.7 Conclusions and future work

In this chapter, a formulation of self-reconfiguration as a search problem for Roombots modules is presented. The presented framework can easily be adopted to various SRMRs by adapting the transition model to the kinematics of the used modules.

The proposed framework is complete and optimal. Therefore, it finds a solution as long as a solution exists. Furthermore, a novel transition model and a heuristic are proposed based on motion capability of RB modules. The proposed heuristic can successfully guide the search. However, there is still a room for improvement on the heuristic function. Thanks to history keeping and active pruning strategies, the search could be accelerated to enable small to medium size problems in reasonable time. The proposed framework can already solve various scenarios. Moreover, the plan is ready to be deployed in real modules owing to hardware and control extensions added to Roombots to increase docking reliability.

The next step will be developing a higher level planner using the framework explained in this chapter as the core element. Similar to all other search-based self-reconfiguration algorithms in the literature, the proposed framework is also limited with the size of the search problem. If the problem is too big, the higher level planner could divide the task into sub-tasks and each sub-task can be solved with this framework.

User interfaces for modular robots Part II

4 From playdough to Roombots

How would you tell a SRMR to form a chair? It could be possible to give voice commands to initiate it. Especially when a library of chair types is available, the operator could just say *"Build me chair # 3!"*. What if you want something new and not predefined? One can take the modules one by one and assemble, but, it beats the purpose of an autonomous robotic platform. One can open a PC and start clicking buttons, dragging modules around on a PC screen until he/she is happy with it. It would work as most of the SRMR research is using exactly this method. But, is it really the most user friendly way of interacting with SRMRs? We believe not. To prove it, we allocate the whole Part II for the study of user interfaces (UIs) for SRMRs.

This chapter explains a novel user interface making use of tangible media (malleable sculpture dough). We believe the easiest way of conveying 3D shape information is through manipulating playdough. Even very young kids know how to create shapes of everyday objects with playdough. Once user creates a sculpture, it is 3D scanned into a mesh to digitize the shape information for further processing. The goal of this chapter is to investigate the feasibility of commanding Roombots to form desired shapes by using playdough.

Reference publications

- This chapter is based on ***Mehmet Mutlu***, *Simon Hauser, Alexandre Bernardino and Auke Ijspeert. "Playdough to Roombots: Towards a Novel Tangible User Interface for Self-reconfigurable Modular Robots."* IEEE International Conference on Robotics and Automation (ICRA) 2018.

  My original contributions

  - Designing the user interface
  - All of the coding and testing
  - Partially doing the hardware demonstrations
  - Writing the manuscript

4.1 Introduction

One of the grand challenges [Yim et al., 2007] for the intended usage of structures built with SRMR is the method of how the inspiration for a structure by a user is translated into an real-world representation with SRMR. This process is illustrated in Fig. 4.1. We formally split the full process into five subprocesses: (i) expression, where an idea of a user takes shape in the real world, (ii) digitization, where this idea gets digitized and put into a PC, (iii) abstraction, where the raw digital representation is post-processed into a representation that can be used by an SRMR system, (iv) planning where algorithms produce an instruction plan that creates the desired shape in the form of a building sequence and finally (v) formation, where the SRMR form into the initial inspiration in the real world by following the building plan provided by the planning step. It is important to notice that only the first and the last processes take place in the real world and only they are usually able to provide any other than visual feedback (e.g. haptic feedback) which is the primary type of feedback in the digital representation (Fig. 4.1).

Earlier works on Roombots focus mostly on only single processes of Fig. 4.1. In [Bonardi et al., 2012a], a user utilizes a tablet PC and augmented reality to virtually place pre-defined structures into a living room; the inspiration is never expressed in real world and digitization does not take place since the abstraction step is directly performed manually. [Ozgur et al., 2014] proposes a direct human robot interaction (HRI) to control the location of modules in discrete grid environment in the real world, whereas [Mutlu et al., 2016] extends this idea to the continuous space; the expression and digitization steps are skipped in both cases. In contrast, much stand-alone work has been done in the planning step (e.g. [Bonardi et al., 2012b]), where a selected set of abstracted structures have been provided to reconfiguration algorithms. The formation step has not yet been the primary focus and is subject of ongoing hardware development but ultimately necessary to demonstrate the complete cycle.

This chapter focuses on the three aspects expression, digitization and abstraction as a whole. In particular, we seek a method that is able to rapidly produce the abstracted representation of an idea; as one of the main strengths of SRMR is their ability to shape-shift and create (almost) arbitrary structures, a fast abstraction of the inspiration is necessary to allow a purposeful interaction with an SRMR-system. We present a method that combines the relatively novel technology of 3D-scanning with an existing algorithmic approach (DFS) with adjustments to autonomously generate the building instructions for Roombots, bringing us one step closer to the vision of Roombots to be used as “fast-protyping” user-created furniture. At last, the formation (self-reconfiguration) step is subject of ongoing hardware development and will not be discussed in this chapter.

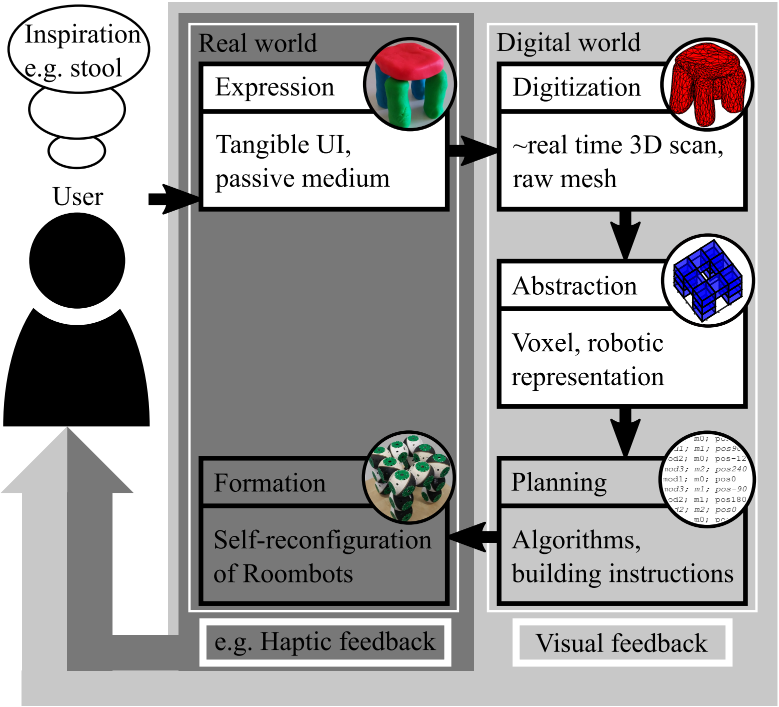


Figure 4.1 – The cycle from an inspiration to the final Roombots shape. The inspiration first is expressed in the real world. A digitization step transforms it into the digital world where the processing steps abstraction and planning can take place. At last, the inspiration takes shape with the SRMR by the formation.

4.2 Analysis of the problem

The main purpose of this chapter is to create a user friendly interface to generate a final *shape of structures* to be self assembled by Roombots. The interface should come up with a construction plan of the a desired arbitrary structure. For this, each of the three processes (see Fig. 4.1) leading to the processing step are explained in detail below, motivating our choice for each of the used methods. The explanation of each method can be found in further sections.

4.2.1 Expression: Modeling clay

An inspiration can be expressed in the real world in many different ways. It can be in the form of a drawing, a spoken or written description or even as a pantomimic gesture. As the purpose of Roombots is to form 3-dimensional shapes, expressing the inspiration also in a 3-dimensional way seemed to be the most intuitive approach. However, the size of the structures formed by Roombots is roughly in the order of magnitude of a human. Hence, a method to form a *model* of the inspiration was desired: a Tangible User Interface (TUI) [Ishii et al., 2012, Nakagaki et al., 2016]. There exist diverse interaction media proposed in the literature that could benefit a TUI. For instance, computational building blocks in [Anderson et al., 2000] and active cubes proposed in [Watanabe et al., 2004] can immediately transform the physical structure that a user is building into the digital domain as a 3D voxel array. Furthermore, [Raffle et al., 2004] offers a construction method with kinetic memory which includes the ability of a user to input the shape of the structure as well as the expected motion. The Roombots is a complex robotic system due to the nature of SRMRs. In order to simplify the interaction, we inclined for a passive medium. In this work, the passive medium can be anything (LEGO® blocks, wood pieces etc.) that allows creation of structures that can be replicated with Roombots. Modeling clays are widely used for 3D shape modeling in games, e.g. Cranium® and Cluzzle®, or professional activities like architecture or art. Play-Doh® or similar modeling clays are very easy to shape and made for kids to create 3D shapes. Modeling clay is also one of the most studied tangible user interface (TUI) material used in the literature. For instance, [Piper et al., 2002, Ishii et al., 2004] use the modeling clay for fascinating landscape analysis and digital 3D design. [Johnson and Thomas, 2010] creates electronic circuits using conductive and insulating playdough. Another usage is explained in [Gao et al., 2015] where modeling clay is used to design an ergonomic PC mouse.

4.2.2 Digitization: 3D scanning

Having a passive interaction medium requires an additional step to digitize the shape information that is formed by the playdough (note that if the inspiration is directly digitized e.g. by modeling a structure in a PC, the expression step can be skipped). In this regard, we rely on commercial scanning solutions, which are becoming widely available and cheaper by the day, to convert the real object to the digital data. The result of the 3D-scanning process is a mesh grid that approximates the shape of an object with surface triangles.

4.2.3 Abstraction: Voxels and Roombots

For representing the 3D information, we use a voxel representation. Voxels are commonly used to represent SRMRs, particularly lattice types like Roombots. Example usage of cubic SRMR representation can be seen in [Stoy, 2006, Stoy and Nagpal, 2004] and [Romanishin et al., 2013]. The quality of voxelization greatly depends on the voxel resolution and generally improves with a higher resolution, for which small and many voxels are required. However, Roombots modules are relatively big such that a representation in which a voxel corresponds to a half Roombots module would result in big voxels and low resolution. Low resolution voxelization is likely to result in undesired information loss and a voxel array that possibly is not resembling the original shape anymore. Nevertheless, it is possible to preserve the rough shape information even with big/coarse voxels when voxelization is done in a smart way. An advertisement of LEGO® is cleverly illustrating this challenge in Fig. 4.2.

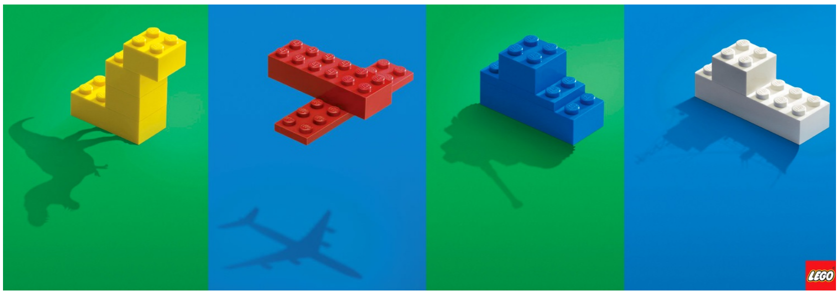


Figure 4.2 – The inspirational LEGO® advertisement emphasizing the shape representation capabilities of primitive construction bricks.

4.2.4 Constraints and simplifications

Some shapes cannot be built with Roombots due to their structure which consists of two consecutive cubic/spheric shape. For instance, the **T** shaped block (in its elementary, symmetric form) in the famous Tetris® game is physically impossible to build with Roombots modules.

Each Roombots module has three degrees of freedom (DOF) that can continuously rotate and two active connection mechanisms (ACM) that allows modules to connect to each other or to an environment equipped with connection plates [Sproewitz et al., 2014]. Such motion capabilities allow a single module to locomote by itself and gives it functional flexibility. However, the same blessing of motion capabilities turn into a curse of dimensionality in a search problem. Nevertheless, most of the furniture-like structures made out of Roombots that have been demonstrated so far do not make use of DOFs, i.e. each joint of a Roombots module is set to zero degrees and the module resembles two consecutive voxels. Even though the continuous rotation of each DOF increases the number of possible structures that can be built with Roombots, in this chapter we use modules as if they are construction bricks. Hence, construction is done only in orthogonal axes.

4.3 Solution method

The construction of structures with a shape given by the playdough is a multi stage process that is almost completely autonomous. Once the user shapes the playdough, the following stages are (i) 3D scanning, (ii) mesh pre-processing, (iii) voxelization, (iv) initial module placement (v) construction search and (vi) the user feedback.

4.3.1 3D Scanning of playdough

3D scanning is needed to capture the information sculpted into playdough. There is a deep research literature on 3D scanning focusing on many different methodologies. An early study in [Rusinkiewicz et al., 2002] captures 3D mesh in real time using a camera. In a more recent study, [Izadi et al., 2011] shows how to fuse color and depth information to have high quality

3D scans. [Follmer and Ishii, 2012] suggests a method to scan modeling clay in 2.5D in real time. There are also commercial 4D scanners (3D scanners that can capture temporal changes). However, they are usually expensive systems. [Straub et al., 2015] offers an open source and one of the most economical instantaneous 3D scanners.

It is possible to replicate the system in [Straub et al., 2015] in a smaller scale for smaller objects. However, developing a 3D scanning system is out of scope of this chapter. We picked the cheapest commercially available solution: a 3D scanner from XYZprinting® for 200$. It is using an Intel® Real-Sense™F200 camera to obtain depth and color images. The Intel RealSense SDK and 3D Scan tool (11.0.27.8892) is used for 3D scanning. As a result, we compromised on real-time scanning for the sake of cost efficiency. Nevertheless, a playdough structure similar to the one shown in Fig. 4.1 can be scanned in approximately 45 to 90 seconds.

4.3.2 Pre-processing of Mesh

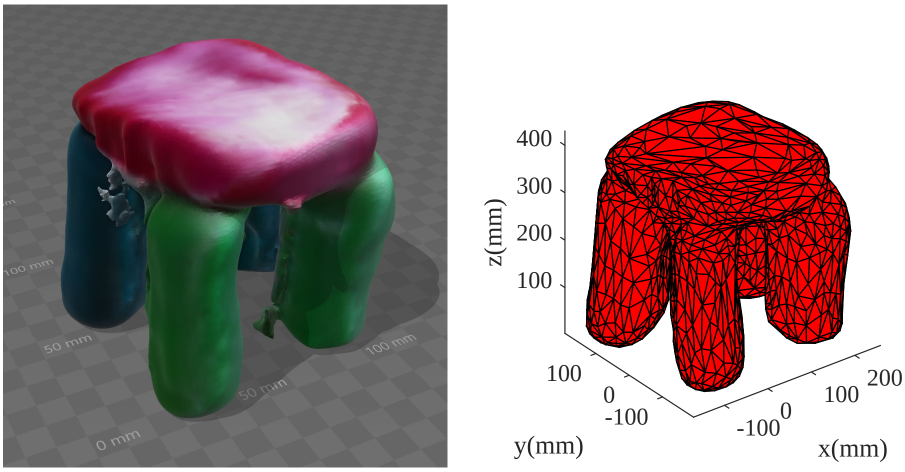
3D scanned meshes need a quick and almost automatized preprocessing. The initial meshes are solidified (to obtain closed surfaces), simplified by reducing the triangle patch count to accelerate voxelization and saved as .stl file format. Finally, they are imported into MATLAB R2015b for the rest of the abstraction. Scanned parts can occasionally have small artifacts as seen in Fig. 4.3a However, those artifacts are insignificant, and rarely need to be explicitly removed.

Scaling

The scanned playdough model is a smaller mock-up of the desired structure where the user has to specify the size of the desired final structure. In other words, the user is giving only the shape information with the playdough and the final size of the desired object is another parameter. There could be a hard-coded constant gain to scale up the 3D model to get the real-life object size. However, making an exactly scaled model means more effort for the user since removing or adding material to change the scale of the sculpture may not be simple. Additionally, the user may want to change the size of the object after some use. In such scenarios, having an adjustable parameter κ eases the user's role. In this chapter, κ is assumed to be the maximum length of the object along any x-y-z axes and it can be set on the fly. We envision that the user can use the GUI or a physical object such as knob or slider bar to define the scale. In summary, κ is the only user parameter other than the playdough shape in the proposed interface. Fig. 4.3b shows the scaled model with $\kappa = 420$ mm.

Orientation

Orientation of the mesh is critical for the processing step of the proposed interface since the Roombots modules will be placed along the orthogonal axes as explained in Sec. 4.2.4. In a stationary 3D scanning system, the orientation of the coordinate frame of playdough



(a) (b)

Figure 4.3 – (a)Initial view of the 3D scan and (b) after pre-processing when the mesh is ready for the voxelization.

structure would be fixed with respect to the world coordinate frame and the user could simply rotate the playdough sculpture to input the desired orientation. Although this could be a nice feature, our orientation information is not reliable due to the handheld 3D scanner. A workaround is to rotate the mesh manually. Alternatively, we propose an auto rotate method in the pre-processing step.

We assume that the user is working on a flat surface, e.g. table, with the surface normal pointing in reverse gravity direction. As a result, the only unknown orientation axes that needs to be optimized is rotation with respect to the surface normal. The problem can be formulated as

$$\underset{\theta \in [0^\circ, 90^\circ)}{\arg\max} f(\theta, \mathscr{M}) \tag{4.1}$$

where f is the occupancy function, θ is the rotation with respect to the surface normal where the sculpture is placed and $\mathscr{M}$ denotes the scanned and scaled mesh.

4.3.3 Voxelization of the 3D model

The power of the shape representation of a voxel space highly depends on the voxel resolution. When the resolution is low, the voxel space suffers from aliasing. Voxelizing the $\mathscr{M}$ with Roombots-size voxels is likely to result in drastic shape deformations. Hence, we firstly voxelize the $\mathscr{M}$ with relatively high resolution to minimize the severe effects of aliasing and obtain the high resolution voxel set $\mathscr{V}_h$. The voxelization for $\mathscr{V}_h$ is based on the generic ray intersection method similar to the one that is described by [Patil and Ravi, 2005]. The whole space is discretized uniformly (center of each cell space representing a voxel area) and voxels confined in the $\mathcal{M}$ are assigned into $\mathcal{V}_h$. The voxels of $\mathcal{V}_h$ obtained by voxelizing the mesh in Fig. 4.3b are plotted in Fig. 4.4a. Secondly, we create a uniform low resolution voxel set $\mathcal{V}_l$ on the same space. Voxels in $\mathcal{V}_l$ are large and have the size of half of a Roombots module. Unlike $\mathcal{V}_h$, where all voxels are binary, $\mathcal{V}_l$ associates an occupancy value to each voxel. That is one of the main breaking points where the proposed solution goes from a generic to a specific application for Roombots. In our implementation, we set the size of large voxels of $\mathcal{V}_l$ (they will be called v_l) to be five times larger than the small voxels of $\mathcal{V}_h$ (they will be called v_s). That ratio between the voxel sizes will be called ρ. So, the space covered by a half Roombots module is represented by 5^3 v_s.

Grid offset selection

The accuracy of shape representation capability with large voxel set $\mathcal{V}_l$ relies on the discrete offset (o) value between $\mathcal{V}_l$ and $\mathcal{V}_h$. The offset is the discrete position translation (in 3D space) of $\mathcal{V}_h$ on top of $\mathcal{V}_l$ with the step size of v_s edge length. The occupancy value of a single voxel in ($\mathcal{V}_l$) is defined as

$$\mathcal{V}_l(x_i, y_i, z_i) := \sum_{x_i-\lfloor\rho/2\rfloor}^{x_i+\lfloor\rho/2\rfloor} \sum_{y_i-\lfloor\rho/2\rfloor}^{y_i+\lfloor\rho/2\rfloor} \sum_{z_i-\lfloor\rho/2\rfloor}^{z_i+\lfloor\rho/2\rfloor} \frac{\mathcal{V}_h(x, y, z)}{\rho^3} \tag{4.2}$$

which corresponds to total number of v_s in the space defined by the given v_l divided by total number of v_s that could fit in the same space. When $\mathcal{V}_l$ is shifted by the size of one v_s in any x-y-z axis, the occupancy of voxels in $\mathcal{V}_l$ changes. Before proceeding further, we define another set $\mathcal{V}_{lo}$ which a subset of $\mathcal{V}_l$ that consists of elements of $\mathcal{V}_l$ that have minimum 0.5 occupancy.

For better shape representation of $\mathcal{V}_l$, its voxels should be occupied as densely as possible. In more rigorous terms, the optimization problem can be formulated as

$$\underset{p\in[0,\rho-1)}{\arg\max}\, g(p, \mathcal{V}_{lo}, \mathcal{V}_h, \rho) \tag{4.3}$$

where $p \in \mathbb{N}$ and the aim is finding the optimal position offset $p = [p_x, p_y, p_z]$ for $\mathcal{V}_h$ that maximizes the fitness function g which is defined as

$$g(p, \mathcal{V}_{lo}(p), \mathcal{V}_h, \rho) := \frac{[2\sum \mathcal{V}_{lo}(p) - \mathbf{card}(\mathcal{V}_{lo}(p))] \cdot \rho^3}{\mathbf{card}(\mathcal{V}_h)} \tag{4.4}$$

In Eq. 4.4, the first term in the numerator (when multiplied by ρ^3) denotes the number of occupied v_s in the set $\mathcal{V}_{lo}$ (i.e. occupied space) and the second term comes from the number of extra voxels that could fit in $\mathcal{V}_{lo}$ (i.e. empty space). Thus, the fitness function is rewarding the higher occupancy, whereas it is penalizing low occupancy. Finally, the term is normalized with the total number of voxels in $\mathcal{V}_h$. Finding the optimal offset between $\mathcal{V}_h$ and $\mathcal{V}_l$ is automated and the implementation is given in Alg. 3.

Algorithm 3 Find the offset yielding the highest occupancy

```
Require: Voxel set $\mathcal{V}_h$ obtained
procedure FINDBESTOFFSET($\mathcal{V}_h, \rho$)
    $o_{max} \leftarrow 0$                                        ▷ $o$ is occupancy
    $p_{opt} \leftarrow [0,0,0]$
    for (each $p$ offset) do                                  ▷ p is 3D in xyz
        $\mathcal{V}_{lo} \leftarrow$ CREATEVLO(p,$\mathcal{V}_h$,$\rho$)                    ▷ Eq. 4.2
        $o \leftarrow$ CALCULATEFITNESS(p,$\mathcal{V}_h$,$\mathcal{V}_{lo}$,$\rho$)          ▷ Eq. 4.4
        if $o > o_{max}$ then
            $o_{max} \leftarrow o$
            $p_{opt} \leftarrow p$
    return $o_{max}, p_{opt}$
```

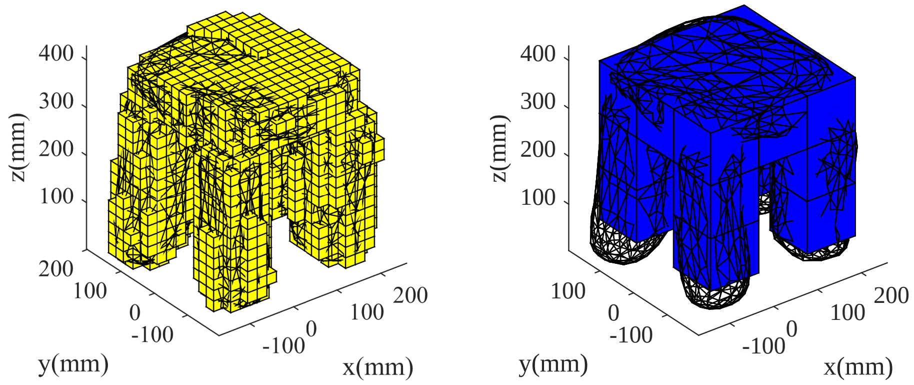
For better clarification we will revisit the auto-rotate method. The computational implementation of the rotation procedure, is given in Alg. 4. The implementation of Eq. 4.1 is a brute force method in which the resolution of θ is taken to be 5°.

Algorithm 4 Rotate mesh to maximize occupancy

```
Require: Mesh $\mathcal{M}$ is scanned
procedure ROTATEMESH($\mathcal{M}$)
    $\theta_{max} \leftarrow 0$
    $o_{max} \leftarrow 0$
    for (each angle $\theta$) do
        $\mathcal{M}_t \leftarrow$ ROTATEZ($\mathcal{M}$,$\theta$)
        $\mathcal{V}_h \leftarrow$ VOXELIZE($\mathcal{M}_t$)                       ▷ $\mathcal{V}_h$ is voxel set
        $o, p \leftarrow$ FINDBESTOFFSET($\mathcal{V}_h, \rho$)
        if $o > o_{max}$ then
            $o_{max} \leftarrow o$
            $\theta_{max} \leftarrow \theta$
    $\mathcal{M} \leftarrow$ ROTATEZ($\mathcal{M}$,$\theta_{max}$)
    return $\mathcal{M}$
```

Search space selection (reduction)

The following subsections will be explaining the construction/search part of the interface. Having a small search space usually reduces the computation time of the search algorithm. Hence we select only $\mathcal{V}_{lo}$ to be filled with Roombots modules. The resulting $\mathcal{V}_{lo}$ of the mesh shown in Fig. 4.3b can be seen in Fig. 4.4b.



(a)

(b)

Figure 4.4 – (a) High resolution voxel space to approximate the 3D model accurately and (b) low resolution voxel space in which the voxel size is equal to a half Roombots module.

4.3.4 Search space definitions and initial module placement

In the rest of the section, the search approach used to fill $\mathcal{V}_{lo}$ with Roombots modules is explained. For a comparative study, two of the well known exhaustive search algorithms, breadth first search (BFS) and depth first search (DFS) are implemented. The discrete search space is $\mathcal{V}_{lo}$ which is a 3D voxel array. Each node is defined as a connected structure that is possible to be constructed with Roombots. Each layer corresponds to total number of modules in the structure. For instance layer zero is the empty $\mathcal{V}_{lo}$ and layer-1 is a only single Roombots module placed in $\mathcal{V}_{lo}$. The goal node (n_G) is the state where either all voxels of $\mathcal{V}_{lo}$ are occupied (when $\textbf{card}(\mathcal{V}_{lo})$ is an even number) or just a single voxel is left empty (when $\textbf{card}(\mathcal{V}_{lo})$ is an odd number) assuming the search problem is optimally solvable. The globally optimal solution is the structure which yields the highest possible occupation. By the term optimality, we also refer to set of structures that are completely constructible with Roombots, unlike the case discussed in Sec. 4.2.4. When the $\mathcal{V}_{lo}$ does not lead to an optimal solution, we need to search all possible structures to come up with the best matching structure.

In order to ensure globally optimal solution, the start node (n_S) should be the empty $\mathcal{V}_{lo}$ and in the first iteration all possible initial placements should be inserted into the search queue/stack. Eventually, all initial placements lead to the optimal solutions when $\textbf{card}(\mathcal{V}_{lo})\%2 = 0$, provided that the initial placement does not result in non-optimal free space (i.e. dividing the free space and each new space blob having odd number of voxels). Further discussion on optimality will be given in Sec. 4.3.6. Thus, we pick a random optimal initial seed module and start the construction search with that one. Fig. 4.5 illustrates two different initial configurations.

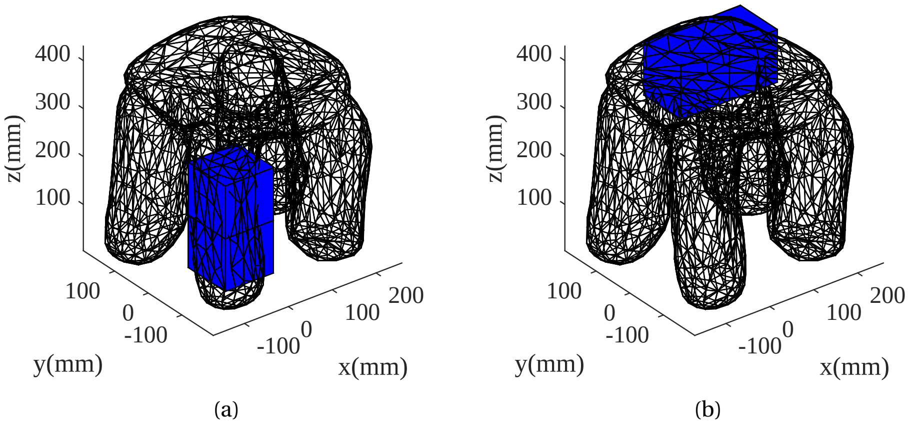


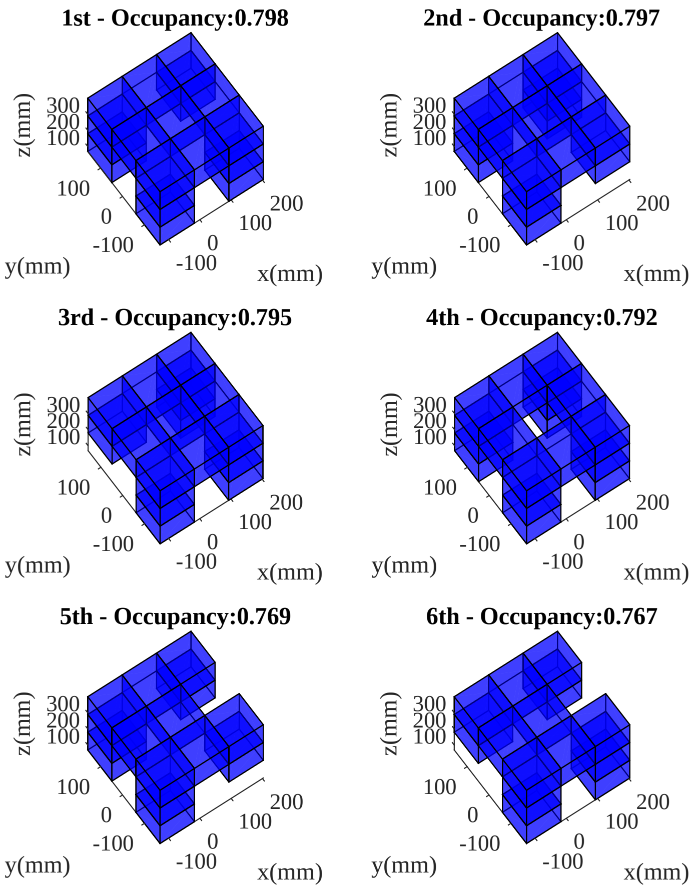
Figure 4.5 – The initial module can be placed in any location in $\mathcal{V}_{lo}$. Two examples can be seen in (a) and (b)

4.3.5 Construction with breath first search

BFS gives a structured baseline to analyze the construction process. It exhaustively searches for the all possible structures and stops only when all alternatives are tested. However, the expansion factor of the search space is very high exponential. A single Roombots module has 10 surfaces. In free space, another module can attach to any of the surfaces in five different ways, resulting in 46 (50 – 4 repetitions removed) different possibilities (children nodes) in the second layer. The number of surfaces increases, resulting in even higher number of children per parent node in the 3rd layer. In such an exponential search space, any branch that can be pruned helps to improve the total computation time. Hence, we do the pruning with (i) history keeping and (ii) optimal placement check. The history all of the opened nodes is recorded and we do not push the child in the search queue if it already exists in the history. The history is implemented with *map container* with unique keys generated for each structure to optimize the time spent checking if the node has already been opened. The flow of the procedure can be seen in Alg. 5. Once BFS ends, the user has a selection of possibilities listed in highest occupancy order as shown in Fig. 4.6.

4.3.6 Construction with depth first search

The structure of the DFS is more suitable for the addressed construction problem particularly when the structure can be built with Roombots and the optimal solution exists. Therefore, we implemented DFS with the same structure of Alg. 5 by only changing the queue to stack. DFS marches towards goal and stops as soon as the goal state is reached. If we let the DFS to search all possibilities, the results would be the same as BFS since they are both exhaustive methods. However, our DFS implementation is intended to find a quick solution for large search space.



1st - Occupancy:0.798

2nd - Occupancy:0.797

3rd - Occupancy:0.795

4th - Occupancy:0.792

5th - Occupancy:0.769

6th - Occupancy:0.767

Figure 4.6 – Six structures resulting from the construction based on BFS with n_s shown in Fig 4.5a. Structures are ordered in the occupancy order. Note that the $\mathcal{V}_{lo}$ has odd number of Voxels for the desired stool structure which means one voxel remains unoccupied. The user can pick any of the resulting structure. Even though the 1st structure has a higher occupancy, the 4th structure would be a better stool in terms of functionality.

Fig. 4.7 gives the resulting structures of the DFS approach.

Structures that cannot be built completely

When the structure cannot be built optimally with Roombots, DFS cannot terminate since n_G does never appear and it searches for all possibilities like BFS. While the exhaustive search still continues, the best results can be displayed to the user and user can stop the search anytime when satisfied with one of the offered solutions.

Algorithm 5 Construction with BFS

```
Require: $\mathcal{V}_{lo}$ is obtained
procedure ConstructBFS($\mathcal{V}_{lo}$)
    Initialize empty $h$                                   ▷ h is the history
    Initialize $q$ with seed module $n_S$                  ▷ q is the queue
    while q is not empty do
        $n_p \leftarrow$ Pop($q$)                          ▷ parent node
        $\mathcal{N}_c \leftarrow$ GetChildren($n_p$)
        for (each $n_c$ in $\mathcal{N}_c$) do
            if $n_c$ is not in h and optimal then
                Push($n_c$,$q$)                            ▷ child into q
                Push($n_c$,$h$)                            ▷ child into h
    $n_G$ = FindHighestOccuancy($h$)
    return $n_G$
```

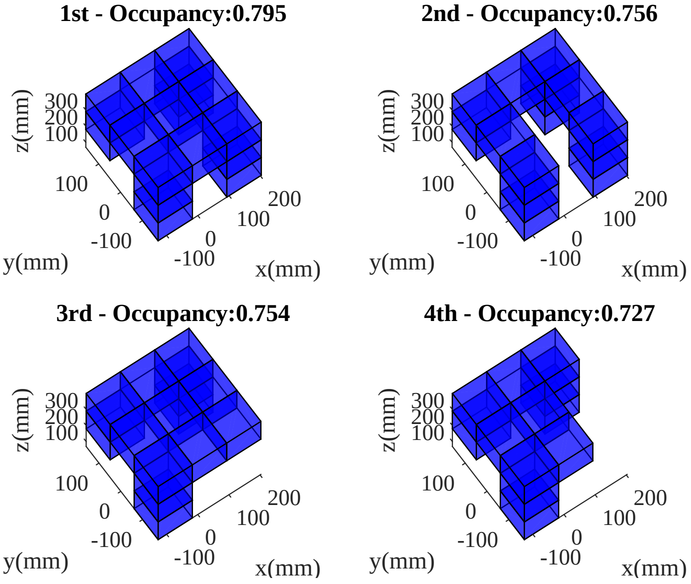


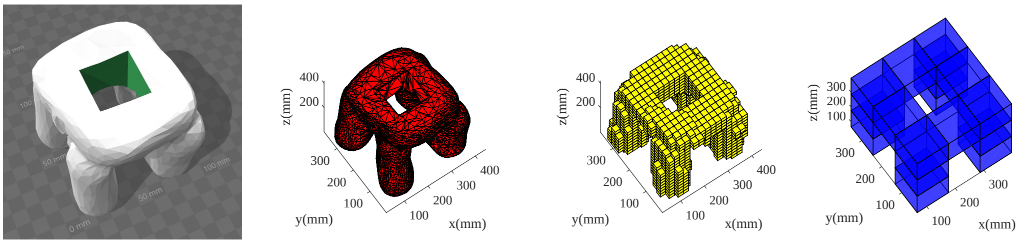
Figure 4.7 – Construction of modular structures with DFS.

Pruning method based on optimal placement

We propose a pruning method for the implemented search algorithm that is mentioned as *optimality check* on the eight line of Alg. 5. If the $\mathcal{V}_{lo}$ can be fully filled with Roombots, we can immediately decide if an opened child will result in non-optimal solution. If the child is dividing the remaining free space into more then one 3D blobs, all blobs should have an even number of voxels. Otherwise, we can prune that branch by discarding the child.

4.3.7 User Feedback

One crucial observation in Fig. 4.7 is that DFS does not suggest a nice functional stool. The main reason is that $\mathcal{V}_{lo}$ has an odd number of voxels and one voxel need to get discarded. However, the displayed results are only for one scan (snapshot) of the playdough. In an interactive continuous case, assuming that the 3D-scanning can be done much faster, the user can still guide the interface towards a desired structure (see Sec.4.3.7). In order to demonstrate the role of the user feedback, then 3D model is modified to emulate what the user could do. Applying the modification, this time the resulting $\mathcal{V}_{lo}$ has an even number of voxels as seen in Fig. 4.8 and a functional stool is quickly found by the search method.



(a) (b) (c) (d)

Figure 4.8 – (a) Creating an artificial feedback by modifying the 3D model directly and running the same procedures on the modified model. Resulting (b) mesh, (c) high resolution voxel space and (d) resulting structure.

4.4 Experimental results and discussions

In order to test the performance of the computational part of the proposed interface, we created the test set seen in the Fig. 4.9 (a)-(h). The common property of all of those models is that they have an even number of v_l and they can be filled optimally after scaling them to Roombots size. For the scaling of the models, we have considered smaller mock-up furnitures rather than the human size because we only have 13 Roombots modules in Biorobotics Laboratory and we focused more on the physical experiments we can do in the future. Fig. 4.9 (i)-(p) shows the resulting Roombots structures of each model in the test set. The number of modules needed for each model can be seen in Tab. 4.1. The same table also reports the computation time for each model when processed with the BFS, DFS and DFS with no pruning check as well as number of different structures tested in history (h_d) until finding the solution with DFS. Computations are done on a PC with an Intel i7-6700HQ CPU and 16GB RAM.

The Tab. 4.1 suggests that the proposed pruning method enables the search to be used in real-time applications in the Roombots project. The BFS time clearly shows the explosive effect in computation time since the search space is highly exponential with respect to the size of the structure. The *NA* marks in the table represents that not all possible structures could be tested in 30 minutes. To put this into perspective: when the *Sofa 1* object is searched for approximately 7 hours, roughly 12 million structures are tested and BFS was still in the 11th layer (11 modules placed out of 25 module model).

Table 4.1 – Comparative performance of the suggested methods

Object	# **mod.**	# $\mathbf{h_d}$	$\mathbf{t_d}$ **(s)**	$\mathbf{t_{d,np}}$ **(s)**	$\mathbf{t_b}$ **(s)**
Stairs wood (h)	7	15	0.10	0.06	0.19
Stool (Fig. 4.8)	8	16	0.10	0.14	0.27
Fancy chair (d)	10	38	0.14	0.10	17.9
Armchair 1 (a)	11	58	0.16	19.6	85.4
Stairs LEGO (e)	14	66	0.19	105	1171
Sofa 2 (g)	16	108	0.24	NA	NA
Armchair 2 (b)	17	187	0.32	0.24	NA
Bookshelf (c)	20	387	2.70	545	NA
Sofa 1 (f)	25	329	1.91	NA	NA

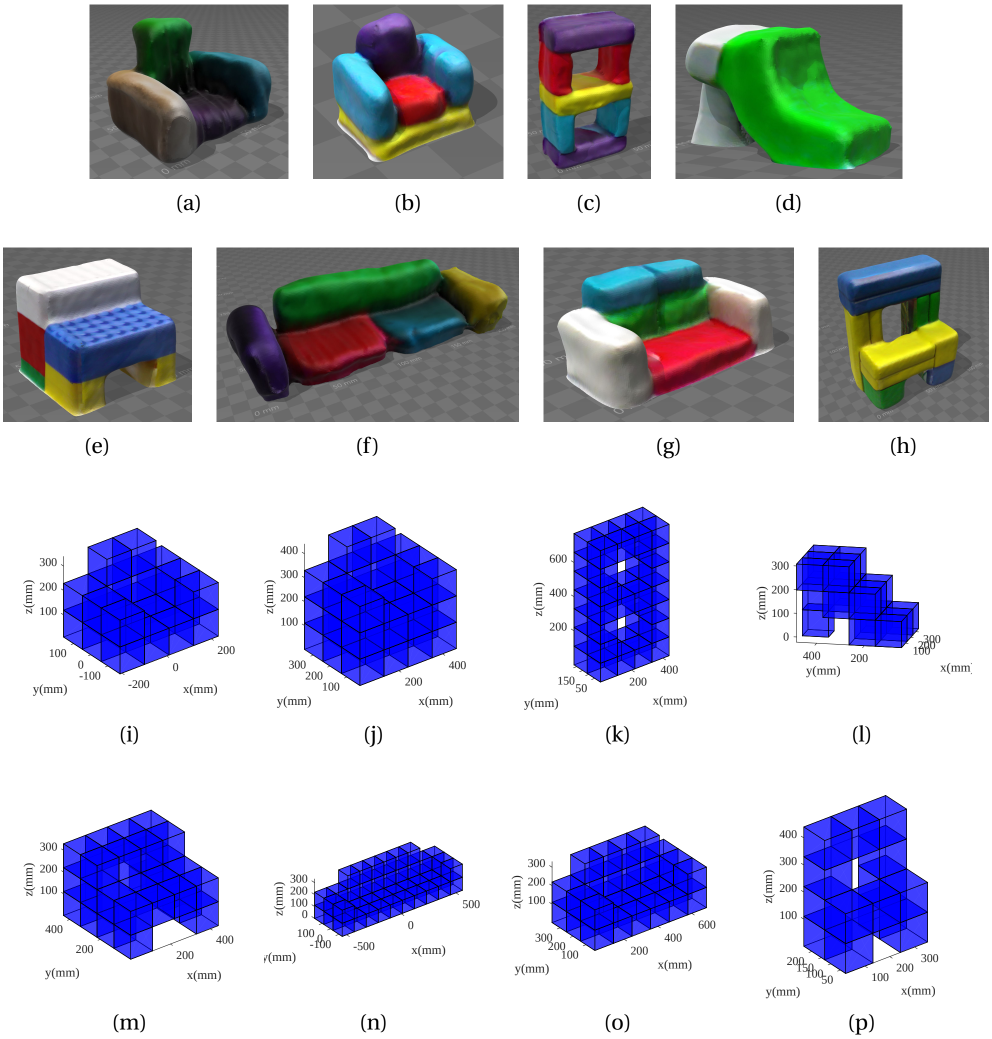
4.4.1 Effect of Initial Module Placement

The location of initial module also effects optimality. For instance when the stool example given in Fig. 4.6 is initialized with the seed module given in Fig. 4.5b, two results having highest occupancy can be seen in 4.10. Note that the second option was not available previously, due to the placement of the seed module. However, DFS already returns the first solution and global optimality of the solution relies on the user interaction where user should iterate the playdough sculpture to achieve the desired structure.

4.4.2 Different construction materials

Modeling clay is not the only option for our interface. LEGO®, wood pieces, any deformable or brick-like material can be a tangible medium in this work. The test set already has models made out of LEGO® and wood pieces. One selection criteria can be the connection capabilities. For instance, LEGO® blocks or wood pieces can only be stacked on top of each other, but, Roombots modules has docking surfaces also on the sides. In that regard, playdough can represent capabilities of Roombots better.

Even active building blocks such as Active Cubes could have been used. For instance, [Ichida et al., 2004] estimates CAD models from Voxel model made with robotic blocks. It is almost the reverse of our problem and could simplify our interface. There would be no need for the (i) scanning, (ii) initial voxelization, (iii) rotation search and (iv) large voxel offset search steps.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

Figure 4.9 – Different objects used for benchmarking and the corresponding found Roombots structures.

4.4.3 Hardware demonstrations

We have manually replicated some of the test set models with Roombots and report the *Stool* in Fig. 4.11. We also tested the reverse process by 3D scanning Roombots structure, as seen in Fig. 4.11c, and running the algorithm to replicate (or potentially by changing size scale) the Roombots structure with another set of Roombot. In that case, active Roombots blocks are replacing playdough to control yet another set of Roombots (or even another SRMR).

4.4.4 Limitations of the proposed solution

Our demonstrated system has three major limitations: (i) 3D scanning is not real time, thus we only demonstrate the proof of concept, (ii) it does not use the full potential of Roombots

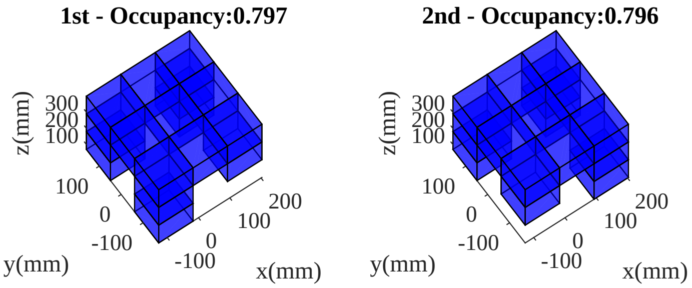
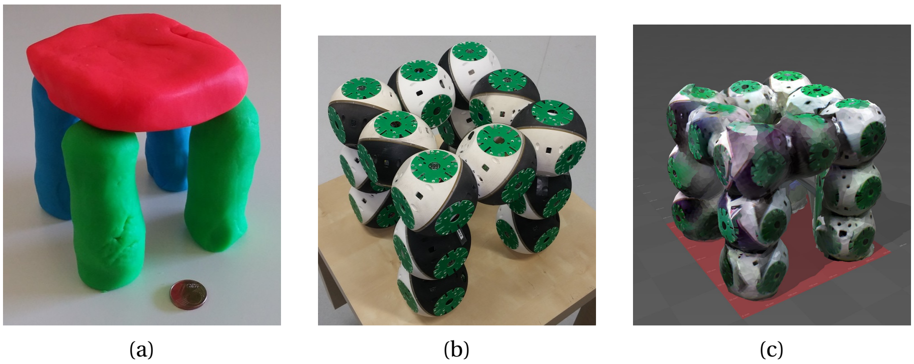


Figure 4.10 – Best two results of BFS when the seed module is taken as shown in Fig. 4.5b.


(a)

(b)

(c)

Figure 4.11 – The stool example from (a) playdough to (b) Roombots and further (c) reverse scanning the Roombots structure.

and assumes them as uniform construction bricks, (yet we can still demonstrate a large subset of the full capabilities) and (iii) scaling up the size of desired structures leads to exponential computation time.

The suggested DFS with history and the pruning check can solve a reasonably sized search space in real time which is sufficient for us since we have only 13 modules. However, scaling up to more than 40-module-structures starts suffering from computation time and getting a quick solution depends on the chance. For large free spaces, additional strategies are needed.

4.5 Conclusions and future work

In this chapter we are proposing a novel, tangible human robot interaction to control the shape of SRMRs. The proposed interface consists of multiple computation steps. The highest computation load is resulting from the search to construct the given shape optimally. The attempted solution space is highly exponential and pruning branches as soon as possible plays a crucial role to have real-time performing interface.

The proposed solution can be applied to any lattice type SRMRs. Having a SRMR with the shape of a single voxel eliminates the construction step. Thus, large scale scenarios can be be built very fast. If a SRMR has a more complex shape, the rule to get new child nodes in the construction step should be updated according to the new shape. The rest of the implementation is generic.

Properties inherent to TUIs also apply to this chapter. For example, a visually disabled person can control Roombots using the TUI. He may not make use of the suggested visual feedback, but, can always touch and feel the final structure after self-reconfiguration and modify the playdough when needed.

The future work in this line of research will focus on incorporating inverse kinematics into the proposed interface to be able to build more complex structures. Real-time 3D scanning is an enabling factor for this work. Hence, we will adopt an automated real time 3D scanning technique. The final work plan is including passive components to replace some of the robotic modules to the search algorithm which would reduce the cost of real systems and possibly boost upscaling properties of the interface.

5 Natural user interface

The previous chapter relied on 3D scanning of an external tangible interaction medium (play-dough). Although, it is very effective in conveying the shape information, it may not be very suitable for some other purposes such as conveying pose data. Humans use verbal commands and pointing gestures for such information transfer. In a previous study, [Ozgur et al., 2014] used pointing gestures to select modules in a discrete environment (grid-space) and send them to desired new locations. The study heavily relies on use of external cameras to track both the user and robots. In this chapter, we are improving the similar idea to achieve continuous control of SRMRs. More specifically, the interface is demonstrated using a metamodule of two RB modules connected in series. Metamodule has strong directional lighting capability. The user controls the location of the light cone on a desktop by using pointing gesture. The study also demonstrates further gesture capabilities for modifying light intensity.

Reference publications

- This chapter is based on ***Mehmet Mutlu***, *Stéphane Bonardi, Simon Hauser, Alexandre Bernardino and Auke Ijspeert. "Natural user interface for lighting control: Case study on desktop lighting using modular robots."* IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN) 2015.

  My original contributions

  - Defining and developing details of ideas
  - Designing the user interface
  - Most of the coding and testing
  - Partially doing the hardware experiments
  - Writing the manuscript

5.1 Introduction and motivation

The world of robotics has evolved dramatically over the last decade. Robots have seen their capabilities increasing, both in terms of mechanics and electronics but also in terms of control. A growing number of robots are no more limited to lab spaces and are being designed to be integrated in every day life environments. They should provide services, help, and support to a wide variety of end-users, ranging from young children to elderly, all of them having specific needs. These robots appear in many shapes and orders of complexity, from the very advanced humanoid robots, such as Asimo [Masato, 2001], able to walk, run, and manipulate objects, to the simpler vacuum cleaner robot Roomba [iRobot, 2018], limited to a specific task. But the complexity of these robots is often linked to their cost, which confines the most advanced ones to lab's environments. They are also often specialized into carrying out a specific set of tasks, such as manipulating objects or exploring unknown environments. More and more robots are being developed to support humans, such as the Keepon [Kozima et al., 2009] robot, used for example as an helper therapy for autistic children, or the RI-MAN robots [Odashima et al., 2006], designed to carry patients from their bed to their wheel chair. But these robots suffer from their high level of specialization into a specific domain and are lacking the ability to adapt to the task to be performed.

As opposed to this rise in complexity trend, the domain of reconfigurable modular robots has emerged as a potential solution. Reconfigurable modular robots are simple interchangeable units able to assemble to form a more complex structure to solve various more complicated tasks. Among them, Self-Reconfigurable Modular Robots (SRMRs) are equipped with active connection mechanisms allowing them to dynamically change shape to adapt to the user needs or to the task to be performed. They are different from more classical bio-inspired and anthropomorphic robots since they do not necessarily exhibit traits that would allow for an intuitive way of interaction (such has a head with cameras or hands with embedded tactile sensors).

RB are made for building reconfigurable living and working environments that adapt to the current needs of human beings. We aim at a smart assistive environment where the robotic furniture is at the service of the user and helps with important aspects of his/her daily life. The hardware technology will be modular using a novel "universal connector" such as to allow flexible use, easy plug-and-play, and gradual implementation in the house. Different research aspects linked to the Roombots project are illustrated in 5.1.

The needs for a natural way of interacting with such robots is growing, especially if we envision to deploy them in everyday life environments, as it is the case in the RB project. When considering interaction inside homes or public spaces, we have to keep in mind that the proposed interaction solution should be non-intrusive but also easy to handle for non-experts or people with disabilities. An example scenario can be seen in 5.2.

We have developed several interaction strategies to control our modular robots Roombots. The first one is a low level communication targeted towards expert users in which we send motion

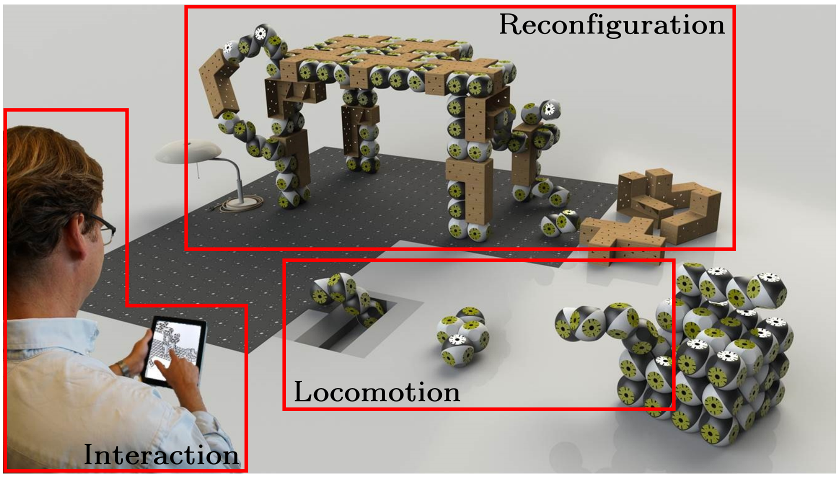


Figure 5.1 – Rendered picture representing the different aspects of the Roombots project. On this illustration, a table is being constructed out of active modules and passives elements (wooden color) evolving on a 2D grid (in dark grey). A set of modules is located out of the grid and metamodules separate from the main group to perform off-grid locomotion. They reattached to the grid using a passive alignment mechanism included in the ground. A user is controlling the process using a tablet device.

command sequences. A second interface based on a classical GUI offers to the user the ability to build pieces of furniture made of RB modules in a 3D environment. It requires the user to focus on a computer display using a keyboard and a mouse for to remotely interact with robotic units situated elsewhere. To enhance the user experience, we designed an Augmented Reality application based on a tablet display in which the user could place robots into the environment and freely move them around [Bonardi et al., 2012a]. The main limitation of this approach was the necessity for the user to carry an external device at all time. To alleviate the need for an extra device we developed a gesture based interface where the user is able to select individual RB modules and move them to target locations by pointing at the modules and at the targets, receiving visual feedbacks on LED setups on both the grid tiles composing the environment and the RB modules [Ozgur et al., 2014]. In this chapter, we proposed a new way of interacting with these non-anthropomorphic platforms, in which the end-user was placed at the center of the interaction by abstracting away the complexity of the control techniques inherent to SRMRs.

Motivated by the existing natural control interface studies [Lichtenstern et al., 2012], [McLurkin et al., 2006], [Bellmore et al., 2011], [Klompmaker et al., 2012], we decided to complement our approach to take into account the versatility of our modular system and the re-usability of its components. We propose in this chapter a novel tracking method allowing the user to interact with a set of modular robots using hand movements. We illustrate our approach with a main usage scenario using the modularity brought by the modular robotic platforms. We created a multi-directional spotlight made of two Roombots modules which is able to control light's direction and intensity based on the user arm/hand movement on a table top.

Figure 5.2 – Roombots are being designed to work in daily life environments. A rendered image that shows an example.

This chapter is organized as follows. In a first section, we describe the hardware setup and software design of our interface. We then present our case studies and the related experiments and results we carried out. Finally, we summarize our study and give potential future extension plans in the last section.

5.2 System design

Roombots is a self reconfigurable modular robot. A single module can perform locomotion on a structured grid environment. When multiple of RB come together, they can attach to each other and/or do more complex tasks by collaborating with other modules in the environment. For instance, a single module can be used as a 3 degree of freedom (DOF) manipulator when it is attached to the grid using its active connection mechanism. A second module can attach to the end of the first one. The resulting structure consisting of two RB modules in series is called metamodule. A metamodule can be used as a 6DOF manipulator on the grid or it can do controlled locomotion on or off the grid. The advantage of RB is pronounced in particular on their flexibility of use. They can form structures such as furniture, carry existing objects or manipulate surrounding environment whenever possible. The focus of this chapter is introducing an easy to use human-robot interface for controlling Roombots to illuminate desired location at controlled light intensity.

Light is the essential part of almost all visual systems. Human and machine vision is only possible with existence of either radiated or reflected light. Hence, proper lighting systems are entangled with our daily lives. Although most of the lighting systems are stationary and only on/off controlled, there exists many variations such as light intensity and color controlled ones or mobile lights as commonly seen on theater stages [Sugden, 1995] to increase comfort, give focus to certain areas or just for entertainment. Another need for local lighting arises on desktops. Small and adjustable desktop lights are commonly used while studying or working on a table. User needs to manually adjust position of the light and the table lamp occupies small space on the table. There are also intelligent and robotics solutions in the literature. A lamp design that uses a higher level of cognitive architecture to achieve fluency in human robot cooperation is presented in [Hoffman and Breazeal, 2008]. Our proposition is using more generic-use RB modules instead of table lamps since RB are mainly designed for autonomously creating furniture. Manipulation capabilities of RB for self reconfiguration purposes [Sproewitz et al., 2010b] and some native interfaces for controlling robot locomotion [Ozgur et al., 2014] were studied in previous works. Therefore, we assume a robot can carry the light to the desired location. In this chapter, the focus is on human robot interaction once the RB are on the location where light is needed.

5.2.1 Overview of the Proposed Human Robot Interface

The proposed system consists of a depth camera, light equipped RB metamodule and a computer. The depth camera can be any camera giving a depth information of image pixels. It is also possible to use stereo cameras, but, in this study Kinect is used since it is widely available and relatively cheap. One of the key requirements is having a mobile and remotely controllable light. Even though in this chapter RB are used to actuate the light, a mechanically simpler pan and tilt system [Wilson et al., 2012, Winkler et al., 2012] can also be used as light actuator. Finally, a computer is needed to implement our autonomous lighting control interface. 5.3 illustrates all components of the proposed system and 5.4 shows the real implementation of the experimental setup. The Kinect is mounted on the ceiling to have bird's eye view of the desktop. It captures both depth and color images and transfers them to PC over USB. Depth images are used to extract the desired light location by detecting the position of the user's arm. The user lifts the arm over certain height to activate the *light position control mode*. The end point of the hand is assumed to be the desired light location. Then, it is geometrically possible to calculate desired pose of the light to enlighten the desired spot, if the coordinates of Kinect and light are known. Once the desired pose of the light is known, robot is commanded to bring the light to the calculated pose. The communication between PC and RB is handled over bluetooth channel. Finally the robot directs the light to the calculated orientation. If the control of the robot is rigid enough and extrinsic calibration of robot and Kinect is done accurately, there is no need for color images. However, color is used as a feedback mechanism to ensure the desired location is really illuminated.

Using gestures for controlling hardware [Ozgur et al., 2014] or software [Kamel Boulos et al., 2011] can be one of the most intuitive control ways depending on the implementation. When a table environment is considered, a person would need to first reach the lamp and then direct the light towards a desired direction. However, with the proposed interface, simply showing the target spot is enough. The interface protocol is quite intuitive. The user needs to raise his/her arm a little upper than the usual working height and reach to the place where light is needed (5.5a). RB will immediately respond to the request by directing the light to the set direction.

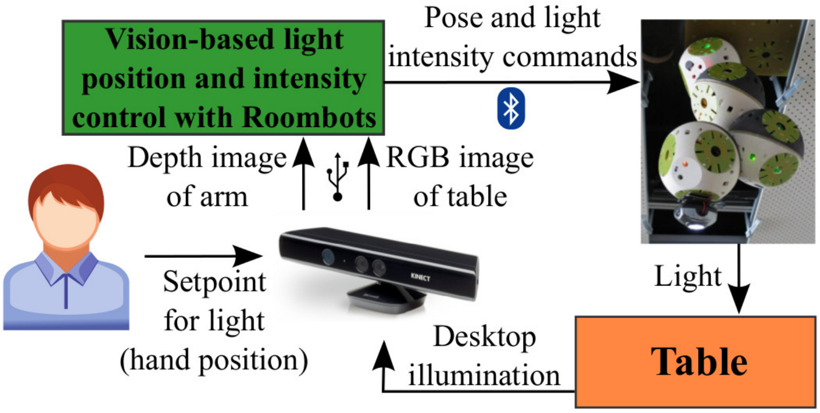

Vision-based light position and intensity control with Roombots
Pose and light intensity commands
Depth image of arm
RGB image of table
Setpoint for light (hand position)
Light
Desktop illumination
Table

Figure 5.3 – Overview of proposed user interface. The key components and relations between them are shown.

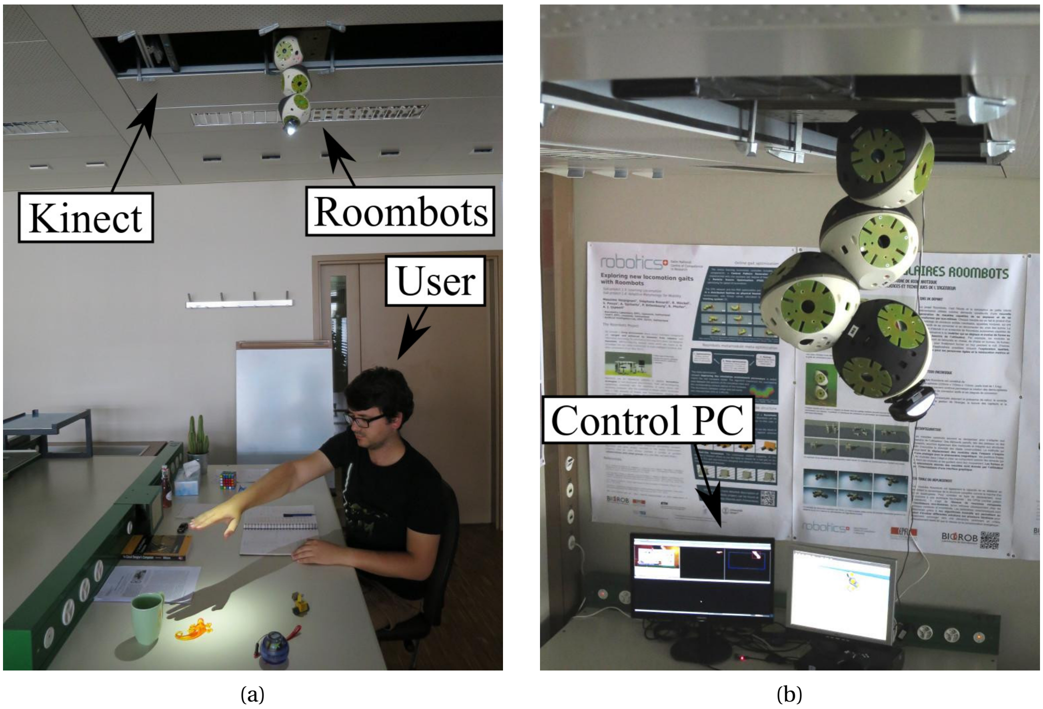
The spotlight will follow the end of the arm as long as the arm stays up, in other words in the position set mode. System continues to light the final set point when the arm is lowered to usual working height as it can be seen in 5.5b.

The second mode of the system is adjusting the brightness of the light. It is similar to setting the position. First, the user needs to raise the arm further up. When the arm is high enough, the system switches into *light brightness adjustment mode.* In this mode, the robot stops position tracking mode and stays in the final set point. At this point the user controls a virtual slider that regulates the brightness. Moving the hand towards right results in dimmer and eventually turned off the light. Similarly, moving hand toward the left side means brighter illumination as illustrated in 5.6. Once the appropriate level of brightness is set, the user can lower hand vertically and switch back to the *light position setting mode.*

A computer is needed to process the Kinect data to get input from the user and to control the robot. However, the user does not need to interact with the PC which is hidden from the user. The illustrations given in 5.7 show how the automated control part works.

5.2.2 Detecting User's Commands

The proposed interface is designed to have minimal learning for the user before starting to use the system. The fundamental information required to accomplish the task is detecting the arm, finding its end point and measuring the horizontal position and height of the end point. This information can be extracted using a depth image of the table captured from above. The depth images captured by Kinect give the distance of each pixel to the camera in millimeters. Hence, the obtained depth image is a two dimensional distance matrix of pixels in the fieldof view of the camera. The first processing step consists in converting the depth image to binary image by thresholding within position-set-mode height. If there are no objects higher than



(a) (b)

Figure 5.4 – Experimental robotic spotlight system consisting of a Roombots metamodule, Kinect (a) and computer(b).

approximately shoulder height, the obtained binary image would show only the user arm when it is raised to set the light position. The implemented case, shown in 5.4a, is an example of such scenario which simplifies the computer vision steps. If there are objects higher than the minimum range of light-position-set mode, those objects needs to be filtered out from the depth image. Using methods such as background subtraction [Piccardi, 2004]. The next step is obtaining meaningful regions in the binary image. This step requires connected component analysis on the image, also known as blob detection. Ideally, the binary image should have only a single blob which is the arm of the user. However, due to noise on the Kinect, many other small blobs can be observed on the camera. Moreover, noise can separate the arm blob into multiple blobs. In order to get rid of noise, closing operation (dilation followed by erosion) is applied on binary image and only the biggest blob is taken into account provided that it has minimum size. All the other noise generated blobs are discarded. In the end of the depth image processing, only the arm blob is obtained as seen in 5.7b. Once the arm is extracted, the tip point of it, the furthers point from user, is considered to be the target set-point for spotlight. The target point corresponding to 5.7b is marked as red circle on 5.7a.

Knowing the target position on image plane is sufficient to calculate the spotlight's desired orientation such that its light will illuminate the target point. If the intrinsic calibration of depth camera is done properly, x_{ti} and y_{ti} coordinates of the target on image plane can be mapped to actual x_{tw} and y_{tw} coordinates in the world coordinate frame. Also note that depth

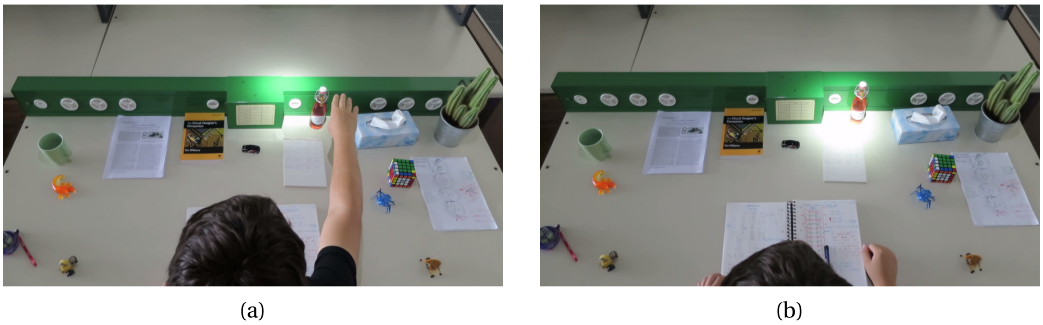

(a) (b)

Figure 5.5 – User is pointing where the light is needed (a). Once the pose of Roombots spotlight is set, it remains on the desired location (b) as long as the user does not switch to the control mode.

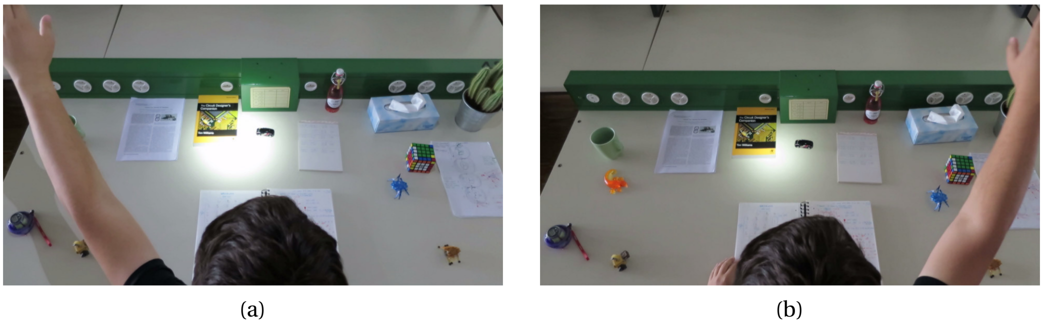

(a) (b)

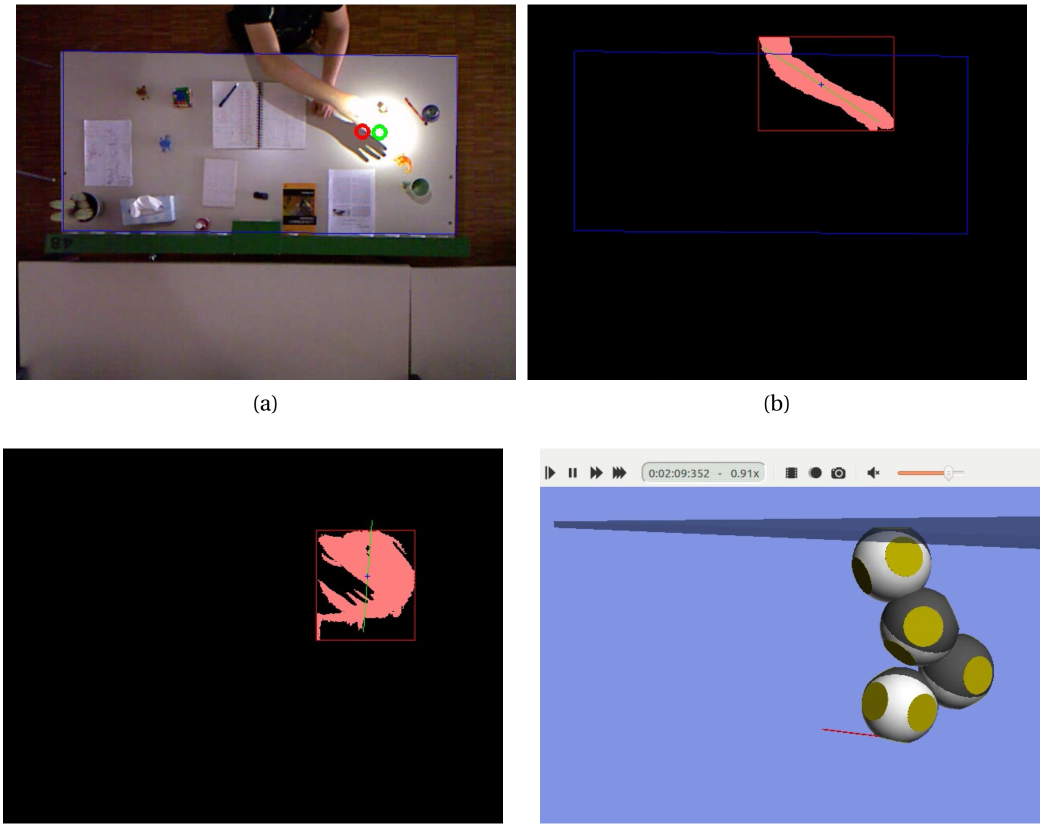
Figure 5.6 – Adjusting brightness of the light. Raising the arm further up puts the system on intensity setting mode. For higher or lower illumination hand should be on the left (a) or right (b) of the table respectively.

image gives the third coordinate, z_{tw}.

The detection of the brightness adjustment mode is almost the same as the position target setting mode. If the height of the target point is higher than the predetermined threshold, the system goes into the light intensity control mode. In this mode the lateral position of the arm tip point is mapped to the brightness of the light.

5.2.3 Controlling Roombots

Knowing the target position on the table is enough to calculate the desired orientation of the spotlight. But still Roombots need to have a low level controller for bringing the light to the desired orientation. As a design choice, the position of the end effector of the RB metamodule is kept constant at a certain point and the light orientation is changed around the pre-determined operating point. Thus, the resulting overall motion of the light is only rotation which also simplifies the control for the operator. Rotation around each axis is called pan, tilt, and roll motion. Ideally, roll has no observable result since the light going out of the source



(a) (b) (c) (d)

Figure 5.7 – The application GUI gives real time information about the detection and control. Although the user does not need to interact with the PC, this information is still useful to explain the system. The Operator can see (a) color images, (b) processed depth and (c) RGB images and (d) simulation view on the application GUI.

forms a conic shape. However, pan and tilt results in displacement of illuminated region on the desk. Even though only pan and tilt motions are required, achieving it is not trivial with RB metamodule.

Attaching two Roombots modules to each other in series results in a 6-DOF serial manipulator (Roombots metamodule). Once the Roombots metamodule fixes itself on a stationary surface (see Fig. 5.8), the forward kinematic model of the system can be written as follows,

$$ {}^{b}_{e}T = \Big(\prod_{i=1}^{6} {}^{i-1}_{\;\;i}T\Big){}^{6}_{e}T \tag{5.1} $$

where ${}^{b}_{e}T$ denotes the transformation from base (${}^{0}T = {}^{b}T$) to the end effector, ${}^{i-1}_{\;\;i}T$ adds one rotation variable for each DOF, and ${}^{6}_{e}T$ is the constant transform to reach the end effectors

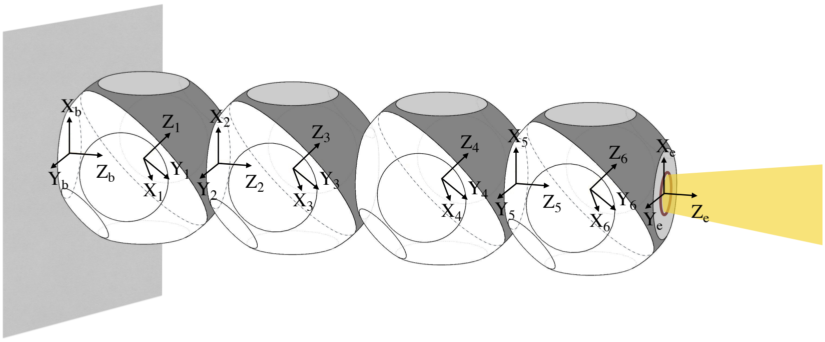


Figure 5.8 – Kinematic model of Roombots metamodule manipulator. Each module (two spheres) has three DOF with continuous rotation capability.

center which is the center of light source in this study. For each desired light orientation (i.e. end effector orientation) the inverse kinematics of the metamodule are numerically solved. The solution of inverse kinematic gives the rotation angles, Q_i, that each joint should reach. Desired light orientation can be achieved after knowing all Q_i values. However, inverse kinematics may not always have a solution or may have multiple solutions. In the multiple solution case, the solution which is closest to the previous states is used. Cases with no solution should be treated carefully or should be avoided. In order to simplify the control problem, continuous space is discretized with resolution of 0.5 degree rotation angles. When the light target orientation is calculated, it is assigned to the closest available discrete angle. Although discretization results in slight errors, the resolution is sufficient for human perception. Since the roll rotation is a free parameter, it provides some flexibility to find valid inverse kinematics solutions. When inverse kinematics could not be found in the first try, the roll angle is changed to find the solution for desired pan and tilt angles. In the end we were able to calculate inverse kinematics in real time for all possible pan and tilt angle combinations within our operation range and resolution. Each solution is tested in Webots simulation environment [Webots, 2018] to validate the results in real time. The Webots screenshot corresponding to time instance for 5.7 is shown in 5.7d.

5.2.4 Closing the Control Loop

When ideal conditions are assumed, open loop control of the RB manipulators is sufficient. However, there are many noise sources in the real world. Extrinsic calibration between RB and Kinect can be done but, slight structural misalignments are unavoidable. Furthermore, there are uncertainties arising from RB hardware such as gearbox backlash and body elasticities. Those uncertainties cause the open-loop-illuminated region to have stochastic position error. One solution to reduce the errors for different scenarios could be using simple, yet precise pan and tilt mechanism or using a rigid manipulator such as industrial ones. However, we would like to implement our proposed interface for RB and it is still possible to compensate most of the noise by implementing active control.

For the closed loop control, feedback from the environment is needed. It is obtained by processing the RGB images captured by Kinect. RGB images are first converted to grey-scale and then thresholded to get the brightest region in the binary image. To avoid noise, a similar blob filtering technique to the one explained for depth images is implemented on binary image. An example of processed RGB image can be seen in 5.7c. The highlighted region shows the blob corresponding to the brightest region. We assume that the illuminated region appears to be the brightest part in the image. Finally, the center of that blob is found by calculating,

$$S_x = \frac{1}{n}\sum_{n=1}^{n} x_i, \quad S_y = \frac{1}{n}\sum_{n=1}^{n} y_i \tag{5.2}$$

where S_x and S_y are x and y coordinates of the first moment of the area, n is the number of pixels in that region and x_i and y_i are image coordinates of blob pixels. The first moment is also known as center of mass of the area in an image. The enter of mass of that blob is assumed to be the center of the illuminated region.

Knowing the set point and the resulting light location, a conventional PID controller can be implemented to compensate errors. Only P control is implemented to have a simpler system, i.e.

$$u = K_p e \tag{5.3}$$

where u is the input to the plant that is error compensation signal, K_p is the proportional control constant and e is the error. Both u and e are two dimensional vectors since the position control of the light is in the (x,y) plane. Thus, the error signal can be written as

$$\begin{pmatrix} e_x \\ e_y \end{pmatrix} = \begin{pmatrix} x_t - x_l \\ y_t - y_l \end{pmatrix} \tag{5.4}$$

where subscripts t and l corresponds to target and actual light positions respectively. The control signal u tries to bring the actual light closer to the target point. In order to avoid overreactions to an unexpected situation u is passed through a saturation function to limit the correction signal's effect.

5.3 Results and Discussion

Proposed system has been implemented in hardware and subjective tests are conducted to evaluate the system. The supplementary material for demonstrating the performance of the interface and the whole system can be found among the media submission of this paper.

Another interesting aspect of the interface is the fact that light position gives a direct visual feedback to the user. In 5.3, we can observe one more implicit feedback loop. Since the user can directly see the location of the light, in case of an unexpected situation when the light does not reach the target position for some reason, the user can intervene the control process by giving a proxy target such that the light would end up the location the user wants. This situation actually means human in the loop control.

5.4 Conclusions and Future Work

In this study we presented an intuitive user interface to control a robotic spotlight which is actuated by a RB metamodule. We integrated a Kinect based input device to detect the arm of the user that is used to set the desired spot and brightness for illumination. The manipulation of the light is done in continuous space and in real time. Finally, we incorporated visual feedback, in the form of detecting bright regions in the environment to improve tracking performance with closed loop control.

The main contributions of this chapter are twofold: (1) a natural interaction designed for autonomous and mobile lighting and (2) Roombots gained a new functionality and behaviour which is bringing the light to most needed spots. The proposed lighting interaction is designed to make light position and intensity control as intuitive as possible. Although, light is actuated with a complex robot, the natural human robot interaction made the control of the lighting quite user friendly. Thanks to this new interface, a relatively complex and generic purpose robot was easily controlled for lighting application.

The light used in this chapter was a commercially available LED with a proper reflective cone. It has been very useful for prototyping the system. As a future work, we will design the light from scratch to fit everything it inside RB module. Additionally, backlash in custom made gearboxes of RB has been identified as the biggest source of uncertainty for precise control of RB. We will revise the design and try to reduce the backlash as much as possible.

Another extension to this study will focus on implementing a similar interface for controlling an end effector of RB manipulator with teleoperation [Vertut, 2013]. RB manipulator can be used to locate objects, even if the system is physically in a different place. For instance, one can help a disabled person who has to stay at home to reach and fetch items using Roombots from other side of the world. In such extensions, improvements to the feedback loop will be needed.

6 Virtual reality for reconfigurable rooms

In the beginning of the Ch. 4, we argued that a user directly building the structure is inappropriate and using a screen may not be the best interaction way. In the last years, head mounted displays (HMDs) became quite popular which raised the question if there could be a possible user experience improvement via use of HMDs for controlling SRMRs. This chapter presents another user interface for SRMRs using HMDs and hand (gesture) tracking sensor. The intended task of this UI is identical to Ch. 4 which is setting a desired shape a structure to be made out of Roombots. However, in this chapter, the user assembles each module one by one in virtual environment.

Reference publications

- This chapter is based on *Valentin Nigolian, **Mehmet Mutlu**, Simon Hauser, Alexandre Bernardino and Auke Ijspeert. "Self-reconfigurable modular robot interface using virtual reality: Arrangement of furniture made out of roombots modules."* IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN) 2017.

My original contributions

  - Defining and supervising the student project
  - Designing the user interface
  - Partially doing the user study
  - Reviewing the manuscript

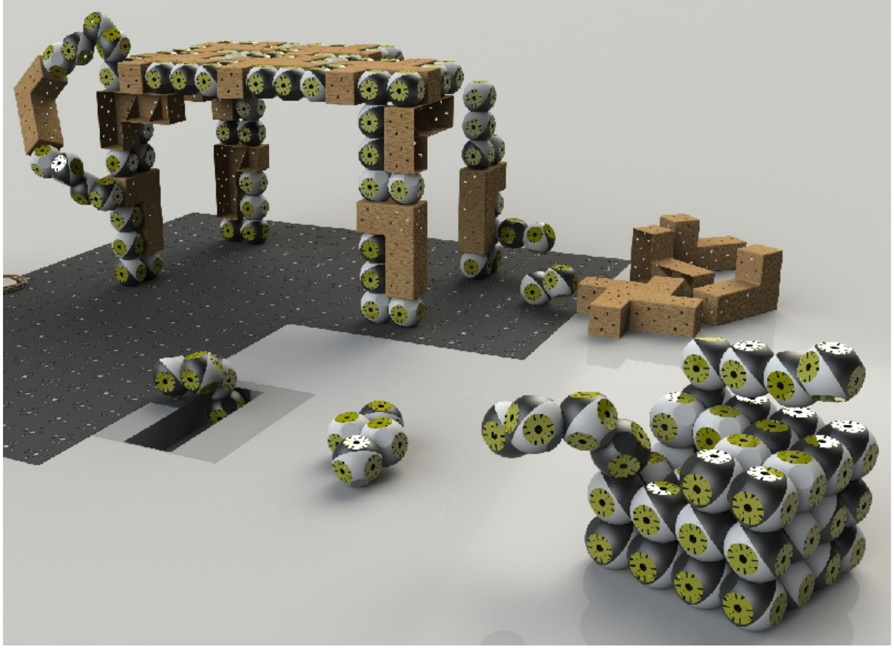


Figure 6.1 – A single Roombots module consists of four half-sphere-like parts that can rotate independently. There is a continuous rotation capability between each dark and white hemisphere within a module. They can form more complex structures through self-reconfiguration. The rendered image displays different capabilities of Roombots while reconfiguring themselves into a table by utilizing L-shaped and X-shaped passive elements.

6.1 Introduction and motivation

There are different levels and ways of achieving generic-use robots. One of the extreme branch for multi-purpose robots is called modular robots (MRs) [Yim et al., 2007]. The idea behind MRs is having relatively simple modules that are not capable of performing many tasks, but drastically enhance their capabilities when multiple modules work together in a collaborative manner. If the reconfiguration can be done autonomously by an MR, the type of MR is called a Self-Reconfigurable Modular Robot (SRMR). SRMRs can potentially offer solutions to complex problems by using many of simple building blocks, and by controlling them in a coordinated way.

[Ahmadzadeh and Masehian, 2015] and [Ahmadzadeh et al., 2015] give extensive literature surveys of MRs in terms of mechanical and control approaches respectively. There is a large literature behind challenges of SRMRs.

Our aim in this chapter is to create a user interface for assembly of structures made out of RB modules and placement of said structures. Essentially, this work focuses on the problem of assembling structures and its intricacies.The MR community already makes use of visualization tools. A nice example is given in [Zykov et al., 2007] where different structures made out of molecubes are created in a simulation environment. High DOF structures created with modular robots increase the need of user-friendly interfaces. For instance, an average remote controlled model car has two degrees of freedom (steering and throttle) and a user can fully control it. In contrast, a human cannot directly control all individual actuators of a 60-DOF SRMR structure. Such a task requires a higher level abstraction layer for a human operator control. However, no high-level interface to control and assemble SRMRs could be found in the literature. This means that although the various hardware-related aspects of SRMRs have been studied in length, very little work has been done regarding how to interact with them. As it follows, it is very difficult to find inspiration when designing such a new interface and even more difficult to evaluate it, as there is no suitable comparison. For this reason, the underlying basis of the interface we present is mostly empirical.

The contributions of this chapter are twofold. First, an interface is proposed to do assembly of structures made out of SRMRs in a seemingly natural manner. The user experience is enhanced by integrating a head mounted display and a hand tracking system to the interface. The interface was then evaluated by conducting a user study. The rest of the chapter is organized as follows. 6.2 gives a review of VR techniques used in related concepts. The proposed system is presented in 6.3 and the evaluation of the system is explained in 6.4. Finally, concluding remarks and future works are given in 6.5. Additionally, an explanatory video showcasing the interface's features was made.

6.2 Background on virtual reality

Virtual reality (VR) tools are becoming a part of daily life and they are increasingly being used. Two of the most common uses for VR are video games and online product customization. Changing color, texture or other features of a virtual product and displaying it from any angle is widely used to make the product more appealing. One of the first examples of online product customization is presented in [Nousch and Jung, 1999]. Customers can design their custom furnitures (shelves), online, within the allowed limits so that the product can be manufactured. In [Heydarian et al., 2015], the similarity of immersive virtual environments are compared with real physical mock ups for office related tasks on various architecture, engineering and construction works. Authors conclude that immersive virtual environments can successfully represent the physical world in most of the cases. Substitutionary reality is another branch similar to VR addressing the problem of differences between real and virtual objects around the user [Simeone et al., 2015]. In substitutionary reality, all real objects are associated with virtual counterparts with a certain discrepancy. A low cost VR framework is presented in [Hilfert and König, 2016] where authors focus on the framework that they present and emphasize the use of VR in training scenarios. One of the most used applications for the VR is spatial object placement. Particularly, furniture placement in a room is addressed frequently. The whole mock-up can consist of real-sized virtual objects which the user can interact with as if they are real, as shown in [Dunston et al., 2011] where authors use their system to review the design of a hospital room. Another similar VR application is given in [Beattie et al., 2015] where authors use Oculus Rift and Leap Motion in their interface to help a CAD assembly task. Moreover, virtual environments have the intrinsic feature of modifying physics. This allows for physically-impossible behaviors such as objects floating or intersecting each other. VR offers useful tools for robot control and visualization. Robots may be far away from the operator and the operator can face a challenging scenario. Indeed, analyzing the situation can be extremely difficult with only raw data. However, an appropriate use of a VR tool can ease the duty of the operator and can increase the mission performance as explained in [Nguyen et al., 2001] where VR is used while controlling remote space rovers. Another important aspect of VR in robot control is the ability to have different viewing angles. Although authors are using only simulation in [Recchiuto et al., 2015], they are analyzing the effects of the view angle in formation control tasks of robot swarms. In the real world, all viewing angles may not be possible, but recreating the robot state in a virtual environment gives more freedom to an operator.

In a previous study on the same RB hardware [Bonardi et al., 2012a], authors studied an interface with a goal similar to this current chapter. The approach in [Bonardi et al., 2012a] involves study on a human robot interface in a real-life-size virtual environment using a tablet PC. They found that being able to move in the virtual environment increases precision in furniture placement. They also concluded that augmented reality (placing virtual objects on the real video captured by the mobile device) does not have significant effect on the same spatial arrangement task. Another modular robotics interface presented a direct human robot interaction [Ozgur et al., 2014]. In that study, the operator only points towards a RB module using an arm to chose it and afterwards the operator can point a desired goal location for the chosen module. Then, the module autonomously locomotes to the goal location on the grid environment. The pointing gesture is tracked by a Kinect. This natural interface considers only re-placement of existing single modules and it does not support self-reconfiguration from one structure to a new one. If the notion of placement using VR has been extensively studied, this knowledge can be adapted to manipulate modules in a way that they are assembled to create new structures. This would be an interface which would consider modules to be brick-like objects, with the user assembling structures brick-by-brick. However, it would not consider the ability of SRMRs to change their configuration. The interface we present in this chapter is intrinsically related to the very essence of SRMRs.

6.3 System design

RB are designed to work in a shared environment with humans. In order to control many RB modules, there is a need for practical and easy-to-use interface. The demonstrative scenario considered in this chapter is building arbitrary structures with RB modules, particularly furniture.

6.3.1 Requirements of the System

The primary consideration of this work is usability. We envision a scenario where the user has many RB modules and those modules can create furniture according to needs. For instance, a table made out of RB modules should be able to self reconfigure into chairs and/or stools when needed, thus saving space and having redundant modularity at home/work. The user interface should provide a simple way to define those two different configurations with a pre-established set of modules and passive elements. VR enables also the remote control, such that the user can change the furniture configuration even if he/she is not present in the room. To sum up:

- The user should be able to create any desired structure, using RB modules.
- The user should interact with the furniture in a virtual environment which is the representative of the real room.
- The environment should support innovative input devices to provide an intuitive interface.
- The system is expected to make the user feel immersed in the virtual environment.

6.3.2 Overall system

The interaction is designed to be completely in a virtual environment in order to satisfy the design requirements of interacting with the RB furniture. The system consists of a virtual environment representing a workshop and a room and a Graphic User Interface (GUI) to interact with said environment. The minimal requirements of the interface is only a regular PC filling the system requirements of the Oculus Rift. In our system, the computer uses Microsoft© Windows 7 as an OS and acts as the central device binding the other elements together.

There is a substantial number of different areas of interaction which can potentially result in meeting our design criteria. Joystick, gamepad, inertial remote controllers, haptic devices projectors and 3D screens are only some of the most popular input-output devices that could have been used. Since the system represents the real world, devices with more real-life-like interaction would be more appropriate. Hence, two extra devices are chosen to enhance the user experience: Oculus Rift, a VR device consisting of a head mounted display for 3D vision with head tracking capability, and Leap Motion, a hand tracking and gesture recognition (GR) system. Leap Motion introduces the concept of holding items with the hand and moving them around. On the other hand, Oculus Rift not only replaces the screen with 3D vision capability, but also gives head pose tracking capability to steer the direction of sight in a life-like way.

Environment components

The environment contains various elements interacting together. We distinguish four of them:

- Roombots modules, as described in 6.1
- L-shapes, passive elements on which Roombots modules can attach. They can be seen on Fig. 6.1.
- Passive plates, simple static elements on which Roombots modules can attach. They are typically fixated on walls, the ceiling or the floor. Such plates can also be seen on Fig. 6.1 as the dark gray area on the floor.
- Structures, a set of Roombots modules and L-shapes assembled together.

Virtual environment

The virtual environment of this system is a representation of the real environment where one would assemble modules together and a room which can be any place where furniture is needed; a house, an office, a hospital, a school, a garden and so on. In our specific implementation, an indoor place is considered. This environment has two aspects: the *Workshop* and the *Room*. In the Workshop, Roombots modules and their passive companions can be manipulated to create arbitrary structures. It features a large rotating plate that can be used to consider one's current work from various angles and a grid of passive connectors on which modules can be attached. After building a structure, the user can "export" it to the "room". In the Room, the custom structures are considered to be a single object and are manipulated as such. The user can place the structures in the room along with pre-defined furnitures such as a stool or a table. On the visual side, the Room is not very refined, as this work is more of proof-of-concept than a finite product. We consider two rooms and not one to clearly distinguish the two sort of tasks they allow to do. The Workshop aims at *building* structures. Modules and passive elements are considered as distinct objects. On the other hand, the Room aims at *placing* structures as they would be in an actual room. Whole sets of modules and passive elements are considered as single objects. Screenshots from the workshop and the room can be seen in 6.2 and 6.3 respectively.

Graphical user interface (GUI)

The GUI has four main components: the holders, the pointer, the turntable and the display. First, the holder contain the Roombots modules and L-shapes. They provide a way to add new elements to the scene, as new ones will appear once an existing object has been displaced. Note that the holders are purely conceptual and are thus not shown. The second component is the pointer. It allows to pick up the structures from the holders or anywhere else. The 3D motion of this pointer can be controlled by the Leap Motion. Thirdly, the "turntable" is a large plate that can be rotated, thus giving different perspectives on the modules that are attached to it. Finally, the display shows the current state of the environment through the Oculus Rift's HMD. The Oculus Rift also acts as an input device since its orientation is interpreted by the software to define which part of the scene must be displayed. Indeed, the core functionality

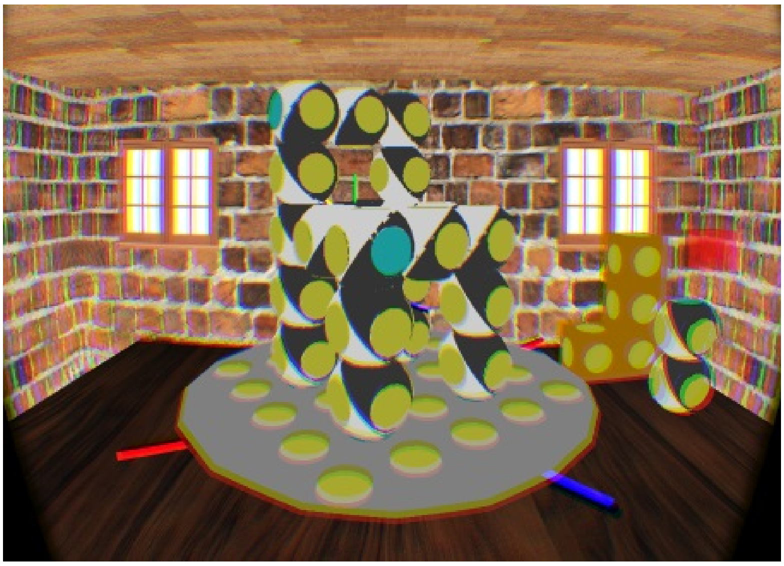


Figure 6.2 – Main components of the GUI in the Workshop. On the right, RB modules and L-shapes can be grabbed and manipulated (the red cube represents the hand-controlled pointer). The cube turns white when holding a handle, to give the user feed-back that he or she pinched it. At the bottom, the “turntable” and its handles. In the center, the structure being currently assembled.

of any VR device is to follow the user’s head movements. Furthermore, the interface uses the keyboard to perform generic tasks that are independent from the other devices. It allows to move around in the room using keys in an arrow-like configuration on the left-side of the keyboard, import and export structures and switch between the Workshop and the Room.

6.3.3 Giving more immersive feeling

Using Oculus Rift and Leap Motion devices, we aimed to give a more immersive feeling.

Directly grasping objects

Leap Motion, [Motion, 2018], is used to track hand gestures. The same function can also be achieved by other depth image sensors like Kinect. Leap Motion was chosen assuming that it can perform better since it is specifically designed to track hand gestures. Holding objects with e.g. a mouse has only two states, holding or not-holding depending on mouse button state and its real world 2D motion must be transformed in a virtual 3D one which is a common problem in 3D-based softwares. However, Leap Motion provides 3D motion and a continuous state for the pinching (grabbing) gesture depending on the closeness of fingers. The GUI gives a holding state feedback by altering the size of pointer. In other words, open hand (non-holding state) results in a big pointer and grabbing hand (holding state) displays a small pointer. Additionally, the inner cube turns white when the user is dragging an object’s

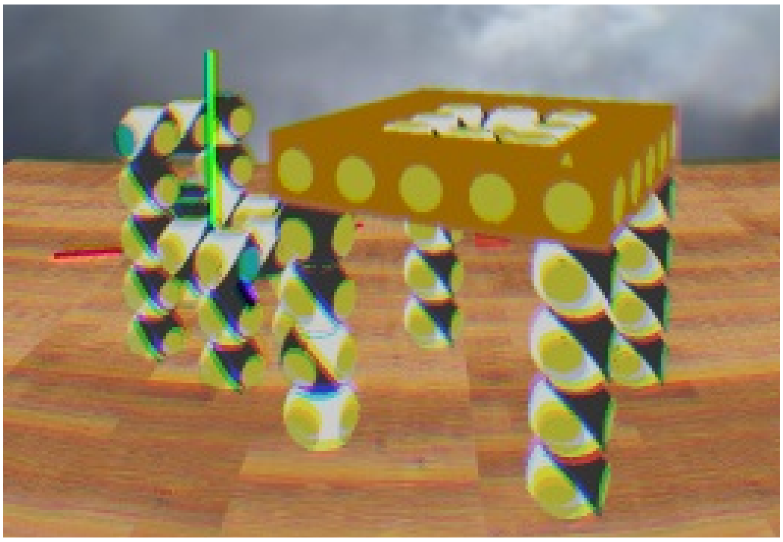


Figure 6.3 – Users can place their custom structures anywhere in the Room. Here, we can see the same chair as in fig.6.2, placed in front of a pre-built table made of Roombots modules and L-shapes.

"handles". Handles are presented in 6.3.3. Thus, the user can get feedback from the system if the grabbing attempt is not successful, e.g. when the gesture is not done correctly. 6.4 illustrates the operator while using the complete set-up to assemble RB-made structures and place them. Indeed, grabbing structures is done by pinching with the hand while bringing the pointer to an object.

Object manipulation & handles

Ideally, one would want to use the same gestures as in real life to manipulate an object. To rotate a module's components, both hands are needed, one holding the module and the other rotating another part of it. However, recognition of complex hand gestures is quite a difficult task and no current technology allows to perfectly mirror hands. For this reason and to keep the interface as simple as possible we only use one gesture, namely the "pinch". The idea is to use "handles", analogous to rods attached to the objects and their components. When an object has been grabbed, it is considered to be "selected". Once selected, an object shows handles that are used to manipulate it, rotation-wise. Pinching a handle and then dragging it would have the object rotate such that the handle follows the direction to the user's hand. Three handles are used. One follows exactly the hand's movements and the other two makes the object rotate around the first one. For a Roombots module, three more handles are added to change its configuration by rotating its components around their axis. All three correspond to rotation around its three aforementioned inner axes, i.e. the actuated degrees of freedom of the real modules. See Fig.6.5 for a visual explanation of handles.

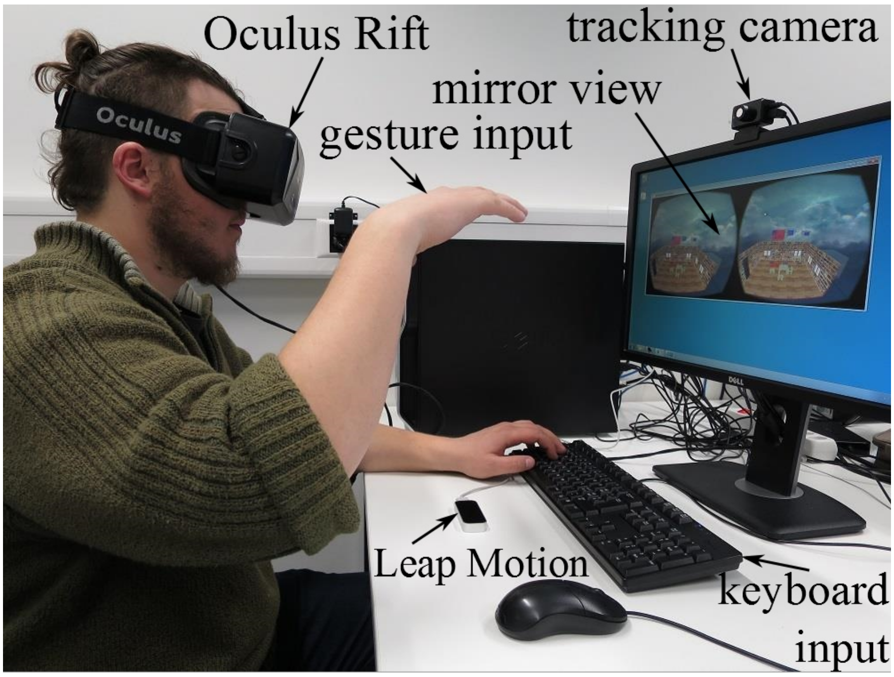


Figure 6.4 – The complete set-up required by the system during a user test.

6.3.4 Assembly

The “connector” is the main concept underlying the assembly process. All objects possess a set of connectors that are either passive or active and that are used to connect them together. Active connectors correspond to the actual ACMs present in all Roombots modules. They represent the ability to attach to other modules’ passive connectors. Two objects are connected if a connector of the first is connected to a connector of the second. More than manipulate single objects using handles, the interface aims at assembling various objects together. This task has two main components : snapping and connectivity. Snapping is changing an object orientation so their connectors match perfectly when they are brought close. It is a well-known rigid body problem and its implementation will not be explained further.

Connectivity

The connectivity problem, however, is non-trivial. Defining how the connections between the objects are represented has several aspects. In this interface, a hierarchical model is used. When two objects are connected, one is considered to be the “parent” and the other the “child”. Manipulating the parent (e.g. dragging it) results in forwarding the modification to its children in a recursive manner, thus manipulating bulks of objects as a whole. To deconstruct a bulk, one would have to grab the latest-attached modules. While this approach works well when assembling a structure piece-by-piece, it becomes more complicated when connecting bulks of objects. Indeed, when connecting two sets of objects together, each having their own

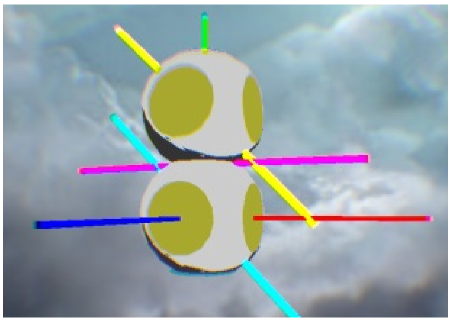


Figure 6.5 – A Roombots module in current selection, thus showing its handles. The green handle follows exactly the direction to the hand's position while the red and blue handles only rotate around the green handle. The rotation of the green handle is done around the bottom of the module. The cyan, magenta and yellow handles are used to change the Roombots module's configuration.

"parent", it is unclear which of them should become the parent of the connected structure. All relationships would have to be redefined in any case. No convenient solution was found to this problem. When two bulks are connected, modifying one will thus not forward the modification to the other. See Fig. 6.6 for a more detailed explanation of the problem.

6.4 Results and discussions

A user study was conducted to test the interface and see if it is a feasible and user-friendly approach. 21 volunteered test subjects (operators) with various VR and GR background tried the system. The main focus was making them perform a series of increasingly difficult construction tasks, thus testing the main functionalities of the interface. Subjective system appreciation was used to evaluate the proposed interface. They simply used the interface as it is. Fig. 6.7 gives more detail on the task performed by participants.

6.4.1 User study

Once the users completed their tasks, they were asked the questions in 6.1. Possible choices were: (0) not at all, (1) not really / no, (2) a little bit / neutral, (3) yes and (4) a lot / definitely depending on the question. Additionally, they were asked the following background questions: "Did you have any previous knowledge about VR?", "Did you have any previous knowledge about GR?" and "Did you have any experience in 3D environments (Games, Computer-Aided Design, etc.)?". Those questions were used to determine the prior experience of our users regarding methods used by the interface. Answers to those background question yield that our sample had sparse (high standard deviation) and overall balanced (mean close to "neutral") experience of both VR and GR, with GR being less known than VR.

The results yield an overall positive feedback, with most people enjoying their experience.

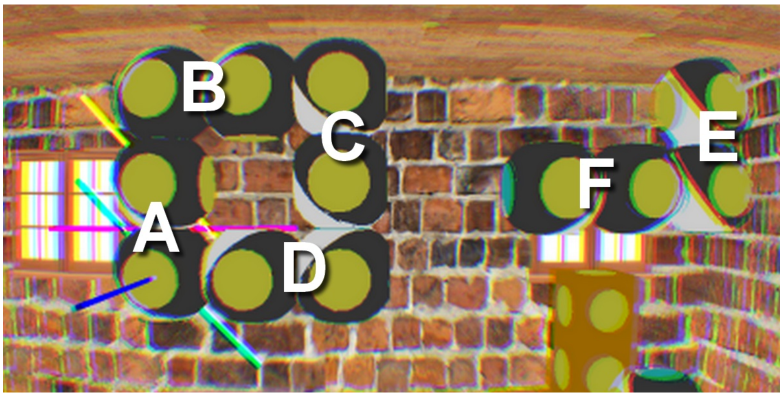


Figure 6.6 – On the left, a module loop forming a bulk. Module A is parent of B, which is parent of C which itself is parent of D. However, the chain stops here to avoid any infinite-update loop. D is thus parent of no module. Moving A would move the whole bulk, but moving C would only move D in addition. The same goes for any modification of configuration. On the right, a second bulk with E being the parent of F, hence the parent of this second bulk. If the user were to bring the left-hand bulk close-enough to F, the whole bulk would be snapped, but no connection would be made (even though the connector of F is active, as shown by its pale blue color). Indeed, after this connection, it is unclear which module becomes the parent of all modules.

This is shown by Q7, with a mean well above "yes" and little variation ($\sigma = 0.51$). In contrast, participants did not find the gesture-control particularly intuitive, as shown by Q2. However, there seems to be consensus that it becomes easier to use with time (learning effect). Indeed, answers to Q3 indicates that participants felt more comfortable after the initial adaptation time. This concurs with the examiner's observations that after a short period of confusion, mostly regarding reconfiguration, most subjects tended to rapidly gain understanding of how to use them. It must be noted that even though people *felt* they got better, a few of them still had great difficulties in mastering the proposed gesture control. Additionally, no participant was familiar with the notion of SRMRs. Q4 and Q5 strongly imply that subjects felt immersed using a VR and GR-based interface, with the Oculus Rift having very high consensual approval and the Leap Motion lower but still satisfactory approval. Q8 and Q9 show an overall positive feed-back but also distinct discrepancies regarding whether or not participants would use this interface or a similar one. Indeed, subjects were not necessarily convinced by the concept of gesture-control itself and a few of them argued they would be more efficient using a more traditional approach. For this reason, the data of those two questions are further discussed using two graphs in 6.4.2.

6.4.2 Discussions

To sum up the user study, we asked the users to perform simple tasks using the immersive interaction framework. The results are promising, yet, should be further discussed in details. Indeed, Q2 and Q8 yield disappointing results. Concerning Q2, the main problem is, as far

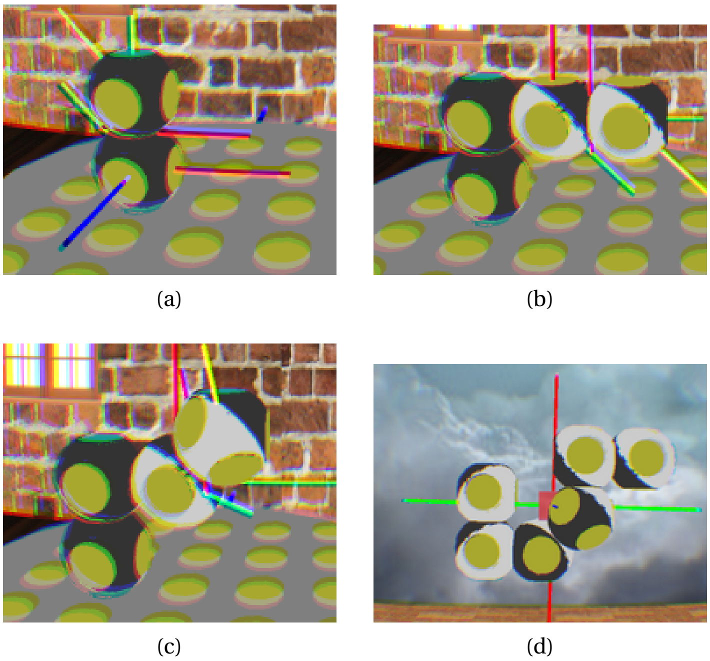


Figure 6.7 – Operators were asked to perform four successive tasks. First (a), simply connect a module to the turntable without rotating any of the joints. Second (b), rotate a module using two handles and connect it to the first one. Third (c), change the second module's configuration to match the picture. Finally (d), add a third module on top, switch to the "Room" and rotate the whole structure to match a given picture.

as we could observe, the detection of the hand due to technical limitations. For example the grabbing/pinching gesture works better when the gesture angle is similar to holding a vertical stick than a horizontal stick. In addition, the range of the Leap Motion is spatially quite limited and gets less and less precise as the hand reaches its boundary. As a result, participants were sometimes confused by why it did not work when stretching their arm too far from the sensor. This is consistent with the results of [Beattie et al., 2015], and should deserve more attention when trying to design a similar interface.

The solution offered by the interface, to let the user move in the room using the keyboard, compensates for this limitation but was observed to often be counter-intuitive. It thus requires training to learn that unexpected sensor response and our test subjects did not have enough time to fully get used to this response. It is interesting to observe that while people would use a similar interface to perform a similar assembly task, they were not particularly convinced by the one presented in the present work.

Additionally, if it did not convince subjects with no or no real experience of VR or, on the contrary, a subject with broader knowledge of it, people possessing little to moderate VR background were more convinced, as shown in Fig.6.8. This might mean that people with little VR background would be more pleased by our use of this technology than people with

Table 6.1 – Subjective evaluation of the proposed interface (Range: 0-4)

	Question	μ	σ
Q1	Did the OR help you perceive the depth better?	2.71	0.85
Q2	Is the gesture control intuitive?	2.86	0.79
Q3	After some time, did you feel the interface getting easier to use?	3.33	0.58
Q4	Did you feel immersed with the OR?	3.57	0.60
Q5	Did you feel immersed with the LM?	2.76	0.60
Q6	Overall, how would you rate this interface?	2.83	0.56
Q7	Overall, did you enjoy your experience?	3.52	0.51
Q8	Would you use this interface if you had to do a similar assembly job?	2.38	1.16
Q9	Would you use another VR-based and GR-based interface to do a similar job?	2.76	0.94

no real or no background. The latter might not see the benefits of VR to do such tasks. A similar observation can be made regarding Gesture Recognition. As shown in Fig. 6.8, the less experienced participants were, the more they were willing to use the interface again. Arguably, people with GR experience might have done so in a completely different manner and were thus forced to change their habits.

However, with little practice, one can be quite efficient at assembling structures. Indeed, the developer of this interface is able to build arbitrarily complex structures in a very natural way. Fig. 6.9 shows an example of a complex structure built by an experienced user. Finally, the results of Q7 are are positive. Which shows that even if subjects would not necessarily use such an interface to perform similar assembly tasks, they still enjoyed their experience. Indeed, VR and GR make the interfaces using those concepts more appealing.

6.5 Conclusions and future work

In this chapter, we presented a novel self-reconfigurable modular robot interface to be able to command Roombots to create desired furniture on the needed spot. The interface uses a virtual environment to represent the real world and has two different functionalities: Create structures using the Workshop and then place them using the living space. To give a more immersed control to the user, HMD and hand tracker were adopted to the interface. Although our user study is not statistically very significant due to the number of testers, we conclude that while there is still room for improvement, especially regarding to gesture-recognition, this interface and similar approaches are promising and open to novel ways to interact with robots in general and even more so with SRMRs. The “handles” approach appears to be a convincing way to achieve intuitive control over the modules considering the low complexity of gesture recognition solutions opted for (i.e. pinching). The handles have certain drawbacks (clarity,

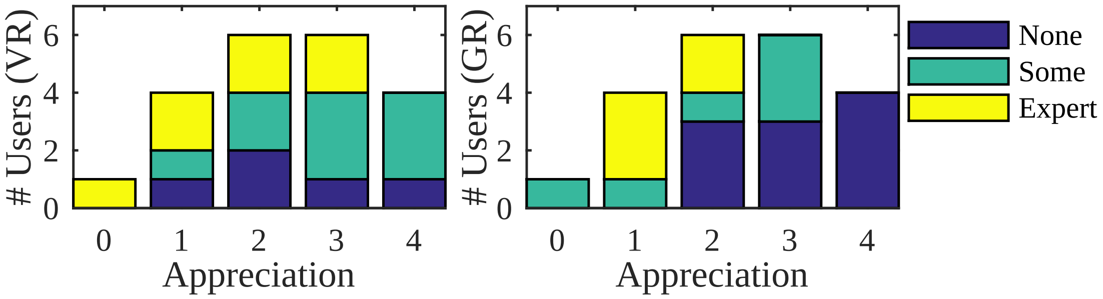


Figure 6.8 – The subjective opinions of users are asked to evaluate appreciation (Q8) distribution of users with respect to their prior experience for the proposed interface. Two plots consider VR and GR experience respectively. Three experience levels consist of (i) none or almost none, (ii) limited and (iii) high experience. Experienced VR users are hard to satisfy since they already had long enough interaction with commercial applications, whereas less experienced VR users appreciated our interface more. A similar trend can also be seen in GR experience levels. However, our users had less GR experience compared to their VR experience and they reacted with more enthusiasm.

not suited for color-blindness, crossing depending on configuration, etc.) but provide a way to create any structure in a much faster way than manually setting the rotary joint angles of each module.

It is important to note that we do not claim that the presented interface accelerates the task completion or increases precision of user control compared to any other. It is an *attempt* to give the user more immersive feeling while controlling SRMR. We evaluated its qualitative *accessibility* and not its quantitative *performance*.

The study's results clearly show the interest of participants for VR and GR. This "fun factor" is a well-known phenomenon related to VR, GR and gamification. Gamification is the subject of many studies and has a broad set of possible applications, notably teaching, as shown by [Villagrasa et al., 2014]. Incidentally, such a strong appreciation for our interface from participants implies it could be used not only to build structures but also to *learn how to* build such structures and manipulate SRMRs. Indeed, SRMRs can be quite complex and it takes time to learn how they behave since their reconfiguration might not be intuitive (efficiency and capability are the priority over practicality). Being able to "play" or "fiddle" with them virtually is very helpful to understand their mechanisms. As such, our interface might provide a more natural way to study SRMRs.

Currently, the main draw-back of the interface is that it is limited in gesture recognition. While the "pinch" is very simple to detect and intuitive, the interface would reach a different level if the hands could be used in a completely natural manner. A possibility to attain this level might be to use haptic gloves or similar gesture-recognition devices. SRMRs still present great challenges on the reconfiguration side. Indeed, it is very difficult to automatically build a target structure. The present interface could be improved to store data from the manual assembling

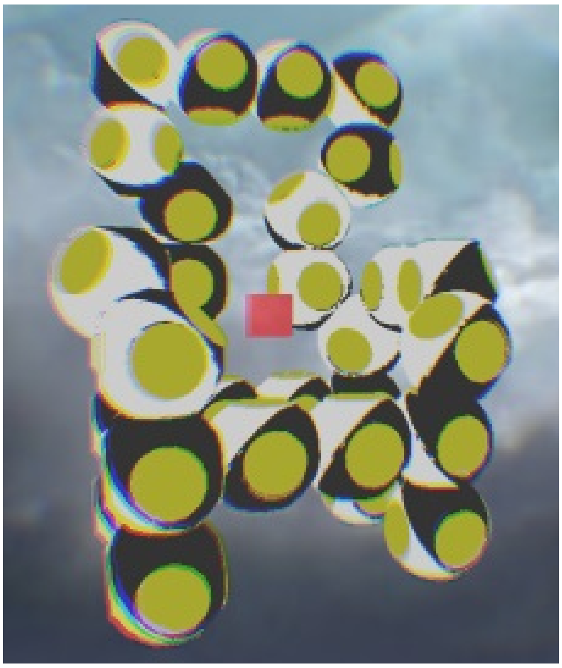


Figure 6.9 – This throne shows the possibilities of Roombots modules. It uses modules in various configurations and positions. All physically possible structures can be built by an experienced user, no matter how complex it may be.

sequence, data that could be used by the modules to self-reconfigure.

Locomotion with modular morphologies

Part III

7 Neuro-visual locomotion model for Envirobot

Self-reconfiguration is a particularly demanding task. It loads many constraints on the mechanical structure and properties of a system. Creating animal-like structures using SRMRs and expecting them to perform like real animals also becomes quite challenging. Although, it is shown in the literature [Bonardi et al., 2014], it still has certain limitations such as size, weight and weight distribution of the structure that may make it less animal-like. Furthermore, self-reconfiguration's benefit on locomotion lies in the morphology change during locomotion. If morphology is not expected to change during locomotion, the use of SRMRs becomes irrelevant. Throughout Part III, we explain three different animal-locomotion related studies where robots have modular bodies, but not self-reconfigurable. The advantage of not having self reconfiguration is the structure becomes lighter, cheaper, faster to develop and can be designed for more specific needs such as being waterproof.

This chapter reports a study of neurovisual locomotion of lamprey. The aim of this study is to create a robotic lamprey which can be used to study neurovisual locomotion model of real lampreys. Our upgraded robotic lamprey consists of a head and modular body consisting of identical and serially (chain structure) connected modules. The head module has cameras and the computation unit to execute a model. On top of revalidating results of [Manfredi et al., 2013] on a real robotic lamprey (cited study shows results in a simulated animat), we carry the study further by adapting the model to run in real time on the robot and even further by integrating more realistic (animal-like) cameras called event based cameras.

Reference publications

- This chapter is based on *Ibrahim Youssef, **Mehmet Mutlu**, Behzad Bayat, Alessandro Crespi, Simon Hauser, Jörg Conradt, Alexandre Bernardino and Auke Ijspeert. "Event-Based Spiking Neural Network Visual Control of an Autonomous Robotic Lamprey"* To be Submitted.

My original contributions

  - Defining the project
  - Supervising three student projects (Ibrahim Youssef, Roger Fong, Elias Klauser)
  - Helping with experiments
  - Reviewing the manuscript

7.1 Introduction

Robotics and biology benefit from a symbiotic relationship such that breakthroughs in one domain often lead to progress in the other [Gao et al., 2019]. The field of biology is a well established bastion of inspiration for robotic development, and has had an impact on subdomains ranging from sensation and artificial intelligence to mechanical design. At the same time, biologists and neuroscientists have begun employing robotic systems to validate hypotheses and computational models in their fields [Floreano et al., 2014], [Ijspeert, 2008]. The lamprey is one such organism at the center of this exchange, and has been the subject of extensive modelling and experimentation. Due to its simple locomotion strategy, well studied physiology, and relatively primitive nervous system, this organism remains fertile ground for similar studies in vision-guided locomotion.

This chapter investigated a solution to the control problem facing visually guided swimming robots. Previous research has investigated traditional feedback control strategies for manipulating aquatic robots using visual input [Sattar et al., 2005][Hu et al., 2009]. Simpler open loop control strategies have been used to implement goal-directed swimming in robotic lamprey [Manfredi et al., 2013]. More recently, a vision guided fish robot developed by Yu et al. made use of a simple body wave model to produce fish-like swimming patterns [Yu et al., 2016]. These strategies have all been used to track singular objects in the robots' fields of view. Little work has been done on robot controllers capable of resolving conflicts between competing stimuli for such aquatic robots. It would be difficult to extend existing control laws to encode and manage more complex behaviors without introducing significant complexity. The study of central pattern generators (CPGs) has gone a long way towards simplifying the problem of coordinating gaits in robotic systems [Ijspeert et al., 2007][Yu et al., 2014]. Still, effective strategies must be developed capable of bridging the gap between complex visual input and CPG drive signals.

The large computational load associated with high-frame-rate image processing remains one of the major limitations to real-time vision controlled robots. Event-based cameras offer a potential solution to overcome many of these limitations. These event driven cameras operate in a way that resembles the function of retinal photoreceptors. Rather than producing frames at fixed intervals, this sensory strategy reports events in response to light log-intensity changes at individual pixels [Lichtsteiner et al., 2008]. What is then communicated by the sensor is not a complete image, but a stream of events specifying which pixels have registered positive or negative changes in light intensity. This approach allows visual information to be transmitted at a significantly lower bandwidth since redundant information is withheld. At the same time, these systems offer a high dynamic range, making them attractive to implement high speed computer vision algorithms on-board robots. A growing body of research has shown how event driven sensors may be used to improve aggressive maneuvering with Unmanned Aerial Vehicles (UAVs) [Mueggler et al., 2015], but no work has been done to study their potential value to underwater robotics.

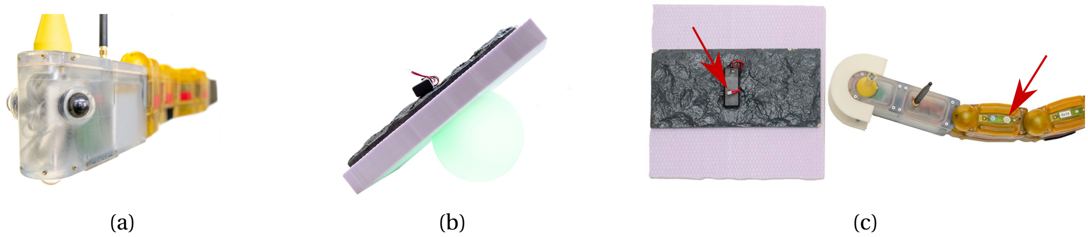
This study offers two main contributions; it adds to the growing body of research on neuron based visuomotor control, and reports on the novel use of event-driven cameras in an underwater robot. This work validated and extended the biologically inspired visuomotor control architecture proposed by Sarvestani et al. [Kamali Sarvestani et al., 2013]. In order to evaluate the control strategy, it was reproduced and used as the main high-level controller for the Envirobot platform. The Envirobot is an anguilliform swimming robot developed by Bayat et al. for the monitoring of environmental conditions in lakes [Bayat et al., 2016]. Multiple behaviors were implemented such that the robot was capable of navigating through an environment containing various attractive and repulsive visual cues. Finally, the frame based cameras used to stimulate the visuomotor controller were replaced with a pair of event based cameras, and performance was compared in a target approach task for the two modalities. The challenge with these sensors, particularly in the unexplored domain of underwater vision, lies in the development of algorithms capable of effectively processing the visual data. Neuron-based control strategies, such as that presented in this chapter, are expected to be particularly well-suited to exploiting the output properties of such event driven cameras. This marked the first instance of dynamic vision sensors being used in an aquatic environment.

7.2 Methods

7.2.1 Envirobot Design

The Envirobot platform was upgraded to include computer vision capabilities. A NanoPi NEO Plus2 [NanoPi, 2017] was used to capture images from the cameras, implement the visual network, and transmit high-level signals to the robot's CPG. The robot head was enlarged and circular apertures were introduced, allowing the cameras (either RGB or DVS interchangeable) to observe the robot's surroundings. A couple of acrylic domes were used as waterproof camera housing. A modular bumper system was introduced to protect the domes against bumping into sides of the pool. Because, water could leak into the robot if the domes would get cracked. An aluminum base plate was designed to carry and cool the robot's embedded PC

(NanoPi). This was found to appreciably reduce the CPU's working temperature from 45°C to 30°C.The finalized Envirobot can be seen in Fig. 7.1a.


(a)

(b)

(c)

Figure 7.1 – The Envirobot target bulb and overhead tracking LEDs. (a) The updated vision-ready Envirobot system. Antennas are used for remote reprogramming. The new head is connected to four modules making up the robot's spine. (b) The target bulb along with corresponding tracking LED. A styrofoam platform and weight were used to ensure that the bulb remained completely submerged in the water. (c) The minimum distance between the Envirobot tracking LED and the platform LED is 0.45 cm during collision.

7.2.2 Frame-Based Object Detection

A couple of XIMEA MU9PC-MH RGB cameras were used to validate the visuomotor control network proposed in [Kamali Sarvestani et al., 2013]. Images were captured using a pair of cameras fitted with wide-angle fish-eye lenses. Both camera allowed for an approximately 270° field of vision while having minimally overlapping region in front on the robot.

Images captured by the camera were used to drive the Envirobot's behavioral circuits. A color-based blob detection algorithm was used to discriminate between red and green blobs, which respectively represented predator and prey. All image processing was performed using functions provided by the OpenCV library (v3.3.1) [Bradski, 2000].

7.2.3 Event-Based Object Detection

A pair of iniVation mini-eDVS cameras were used as the event based sensors. As with the frame based cameras, wide-angle fish-eye lenses were fitted to the mini-eDVS devices, affording the system an approximately 240° field of vision.

With this hardware, individual pixels were directly fed into the neuron network. Since color detection was not possible with these devices, a blink-detection algorithm was used to label objects in the robot's view. A simplified version of the algorithm for flash detection in [Censi et al., 2013] was adopted that omitted the use of a particle filter. Rather than estimating the location of flashing objects, any pixels measured to be flashing at 86 Hz were used to stimulate the input layer of the neuron network.

7.2.4 Experimental Setup

In order to test the behavior and performance of the system, we constructed a water-proof target as seen in Fig. 7.1b that also carries a small LED on top to be compatible with the overhead LED tracking system. For frame based cameras, the bulb's color served as the stimulus (red for predator, green for prey). However, the dynamic vision sensors provided no color information. So, the targets were made to flash at a constant frequency of 86 Hz. Flashing pixels were used to directly stimulate the neuron network's input layer for the attractive behavior when DVS hardware was fitted. For the performance comparison, the avoidance and escape behaviors were not used. The overhead tracking system was able to record a minimum distance of 0.45 cm between the target and the Envirobot. This was due to the placements of the LEDs on the robot and target bulb as illustrated in Fig. 7.1c.

7.2.5 Neuron Model

The neuron implementation adopted for this design was based on the leaky integrate-and-fire (LIF) neuron model [Lapicque, 1907]. The number of parameters in this model were reduced by defining the membrane's time constant as $\tau_m = R_m C_m$. We also applied Ohm's law to to the input current, yielding an input membrane potential $m_i(t) = R_m I_i(t)$. The resting potential of all neurons in our network were set to $m_o = 0$, and a constant refractory period of $T_{refractory} = 1ms$ was specified. The implementation of each neuron in the network was represented by

$$\tau_m \frac{dm(t)}{dt} = -[m(t) - m_o] + m_i(t). \tag{7.1}$$

The membrane potential was computed in real-time by continuously solving the ODE using the Euler method [Griffiths and Higham, 2010].

7.2.6 Neuron Network

The visuomotor control implemented in this project was based on the work presented by Sarvestani et. all [Kamali Sarvestani et al., 2013]. The neuron network was comprised of three main units, a stimulus prioritization block, behavior arbitration block, and output generation block. These subsystems allowed the platform to filter competing signals, select the optimal behavior, and drive the robot CPG accordingly.

Stimulus Prioritization & Position Encoding

The stimulus prioritization strategy was modeled on the structure of the optic tectum found in vertebrate organisms [Bianco and Engert, 2015]. This module allowed the system to focus attention on a single stimulus from a given behavior. For example, when presented with numerous predators, this subsystem allowed the robot to identify which enemy presented the most immediate risk. The sub-network also served as a transformation between stimuli positions in the robot's field of view and system turning strength.

The stimulus prioritization sub-networks each consisted of an *"Input Layer"*, a *"Response Layer"*, and an *"Auxiliary Layer for each possible"* behavior. The neurons of this subsystem were organized into one of two possible configurations; the first arrangement drove the robot towards attractive stimuli, while the other layout was for *repulsive* stimuli.

For all types of behavioral circuits, the instantaneous lateral inhibition between response layer neurons is defined as $I(t)$ in

$$I_{b,n}(t) = w^{Resp-Resp} \sum_{k=1,k\neq n}^{N} y_{b,k}^{Resp}(t), \forall n\epsilon[1,N],\ \ \forall b\epsilon B. \tag{7.2}$$

The connections within behavioral circuits used to direct the robot towards attractive stimuli (prey) obeyed the following rules.

$$x_{b,n}^{Resp}(t) = w_{b,n}^{In-Resp} y_{b,n}^{In}(t) - I_{b,n}(t), \forall n\epsilon[1,N],\ \ \forall b\epsilon B, \tag{7.3}$$

$$x_{b,n}^{Aux}(t) = \begin{cases} \sum_{k=1}^{n} w_{b,k}^{Resp-Aux} y_{b,k}^{Resp}(t) n\epsilon[1,N/2],\ \ \forall b\epsilon B, \\ \\ \sum_{k=n}^{N} w_{b,k}^{Resp-Aux} y_{b,k}^{Resp}(t) n\epsilon(N/2,N],\ \ \forall b\epsilon B. \end{cases} \tag{7.4}$$

The connections within behavioral circuits used to direct the robot away from repulsive stimuli (predators and obstacles) were defined in

$$x_{b,n}^{Resp}(t) = \begin{cases} w_{b,n}^{In-Resp} y_{b,n-\frac{N}{2}}^{In}(t) - I_{b,n}(t) n\epsilon(N/2,N],\ \ \forall b\epsilon B \\ \\ w_{b,n}^{In-Resp} y_{b,\frac{N}{2}+n}^{In}(t) - I_{b,n}(t) n\epsilon[1,N/2],\ \ \forall b\epsilon B \end{cases} \tag{7.5}$$

The connections between neurons of the Response and Auxiliary layers are identical to Eq. (7.4).

All neuron connections in this sub-network were made according to equations Eq. (7.3)-Eq. (7.5), and are exemplified in Fig. 7.2, Fig. 7.3 and Fig. 7.4 for an 8-neuron-wide network. $x(t)$ represents neuron input potential, $y(t)$ represents neuron membrane potential, w is the connection strength between two neurons, and N is the total number of neurons in each layer. B is the set of all behaviors that the robot may perform.

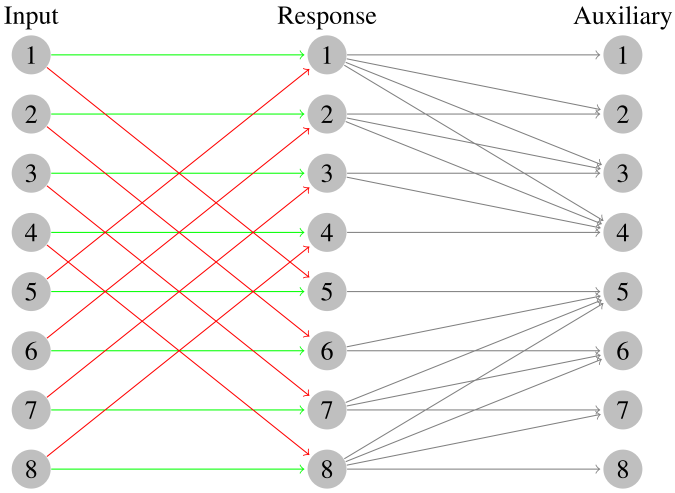


Figure 7.2 – 8-neuron wide network of the module consisting of an input layer, a response layer, and an auxiliary layer. The inhibitory connections of the response layer are not shown in this figure. Connections in green drive the robot towards attractive stimuli, while those in red drive it away from repulsive stimuli. Connections drawn in grey are common for all behaviors.

Behavior Arbitration

Behaviour arbitration sub-network allows the controller to designate a single response when faced with competing behavioral stimuli. The simplified model is comprised of three distinct neurons for each behavioral circuit. This processing block allows the system to attenuate the contribution of less pertinent behavioral cues to the system output. Drawing from the anatomy of the basal ganglia (the neural structure on which this sub-network is based), these three neurons were labeled as the subthalamic nucleus (STN) neuron, internal globus pallidus (GPi) neuron, and external globus pallidus (GPe) neuron [Kandel et al., 2013].

The STN neurons receive stimulation from the response neurons and inhibition by the GPi neuron of the same behavioral circuit. The STN neurons are also inhibited by GPe neurons of all other behaviors. It can be formulated as

$$\begin{aligned} x_b^{STN}(t) = &+ \sum_{n=1}^{N} w_{b,n}^{Resp-STN} y_{b,n}^{Resp}(t) - w_b^{GPi-STN} y_b^{GPi}(t) \\ &- \sum_{k\epsilon B, k\neq b} w_k^{GPe-STN} y_k^{GPe}(t), \ \ \forall b\epsilon B. \end{aligned} \tag{7.6}$$

Each GPe neuron is excited by the STN neurons of the same behavioral circuit.

$$x_b^{GPe}(t) = w_b^{STN-GPe} y_b^{STN}(t), \ \ \forall b\epsilon B \tag{7.7}$$

The GPi neuron of a given behavior is stimulated by the STN neurons of all other behaviors.

Each GPi neurons is also inhibited by the GPe neuron of the same behavior.

$$x_b^{GPi}(t) = + \sum_{k\epsilon B, k\neq b} w_k^{STN-GPi} y_k^{STN}(t) - w_b^{GPe-GPi} y_b^{GPe}(t),\ \forall b\epsilon B. \tag{7.8}$$

Finally, the GPi neuron of each behavior inhibits all response layer neurons of the same behavior.

$$x_{b,n}^{Resp}(t) = -w_{b,n}^{GPi-Resp} y_b^{GPi}(t),\ \forall n\epsilon[1,N],\ \forall b\epsilon B. \tag{7.9}$$

The stimulative and inhibitory connections between the neurons of this sub-network were formally represented in equations Eq. (7.6)-Eq. (7.9). The circuit is illustrated for a system capable of performing 3 different behaviors in Fig. 7.3. Although only 2 behaviors were tested in this work (approach and escape), 3 behavioral circuits are depicted in Fig. 7.3 and Fig. 7.4 to show how the architecture generalizes to perform multiple behaviors.

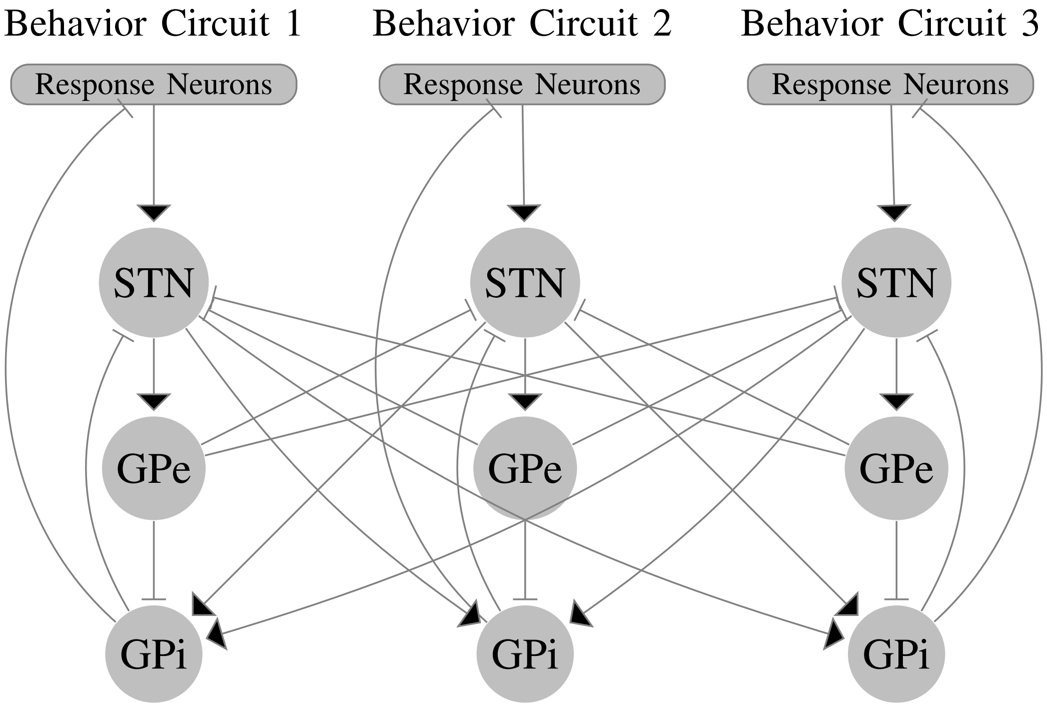


Figure 7.3 – The architecture of the behavior arbitration sub-network. Each behavior has an associated STN, GPe, and GPi neuron. The module receives input stimulation from the response layer neurons. The output of this artificial basal ganglia are the inhibitory connections between each GPi neuron and all response layer neurons of the same behavior.

Network Output

The final portion of the network generates the the robot's CPG drive signals. In this unit of the network, a slight modification to the architecture of [Kamali Sarvestani et al., 2013] is proposed; the swimming speed is controlled by the locomotor neurons' combined firing rates, while turning is dictated by the difference in reticulospinal neuron activations. Additionally, the output of each locomotor neuron is made to inhibit the reticulospinal neuron of the opposite side, further emphasizing this difference. This sub-unit has been implemented as described by

$$x_{left}^{Loco}(t) = \sum_{b\epsilon B} \sum_{n=1}^{\frac{N}{2}} w_{b,n}^{Resp-Loco} y_{b,n}^{Resp}(t). \tag{7.10}$$

$$x_{right}^{Loco}(t) = \sum_{b\epsilon B} \sum_{n=\frac{N}{2}+1}^{N} w_{b,n}^{Resp-Loco} y_{b,n}^{Resp}(t). \tag{7.11}$$

$$x_{left}^{Reticulo}(t) = +\sum_{b\epsilon B} \sum_{n=1}^{\frac{N}{2}} w_{b,n}^{Aux-Reticulo} y_{b,n}^{Aux}(t) + x_{left}^{Loco} + x_{right}^{Loco}. \tag{7.12}$$

$$x_{right}^{Reticulo}(t) = +\sum_{b\epsilon B} \sum_{n=\frac{N}{2}+1}^{N} w_{b,n}^{Aux-Reticulo} y_{b,n}^{Aux}(t) + x_{left}^{Loco} + x_{right}^{Loco}. \tag{7.13}$$

Fig. 7.4 illustrates the connections. The instantaneous estimates of speed and heading were determined using

$$speed(t) = R(x_{right}^{Loco}(t)) + R(x_{left}^{Loco}(t)) \tag{7.14}$$

$$heading(t) = R(x_{right}^{Reticulo}(t)) - R(x_{left}^{Reticulo}(t)) \tag{7.15}$$

7.3 Results

7.3.1 Behavioral Tests

To experimentally validate the visuomotor control architecture proposed in [Kamali Sarvestani et al., 2013], the controller was used to guide the Envirobot through seven different scenarios. For these experiments, only frame-based cameras were used, and behaviors produced in response to

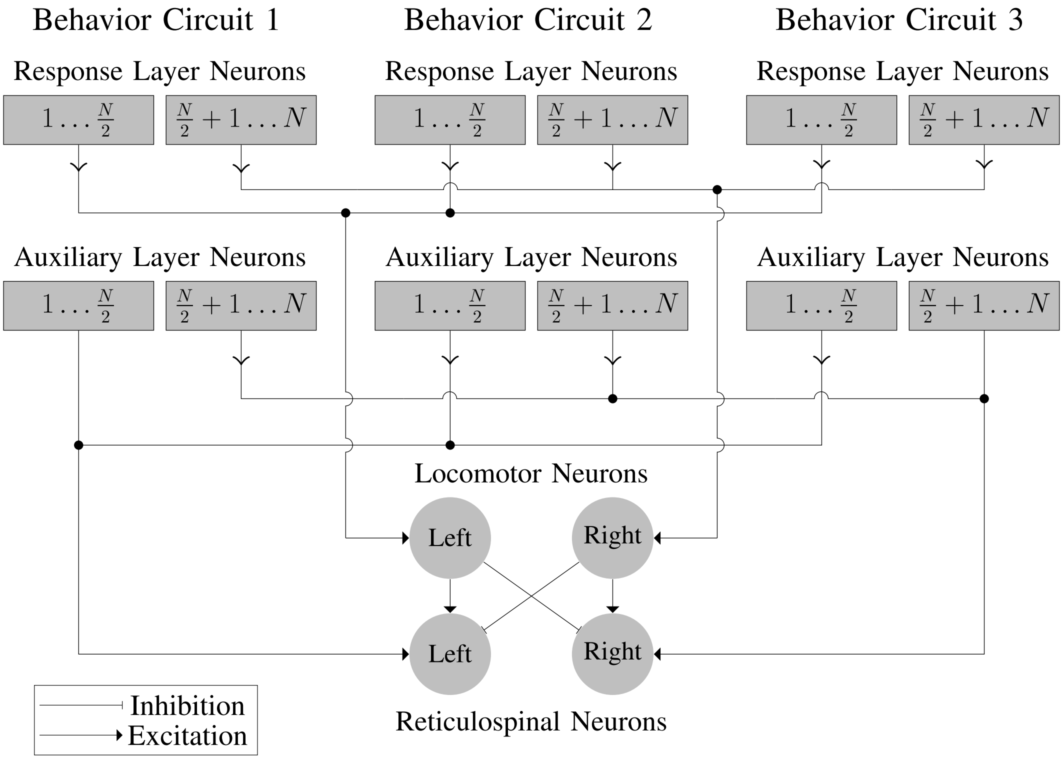
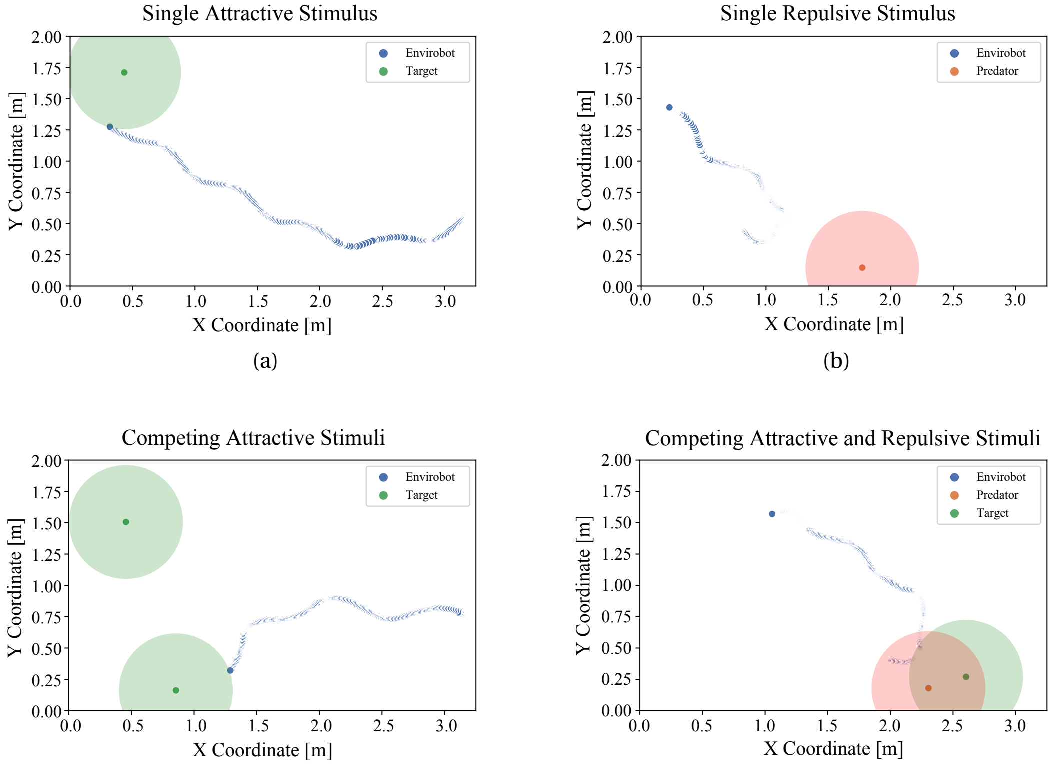


Figure 7.4 – The sub-network used to produce the system's output. Depicted are the connections between the response layer neurons and the locomotor neurons. Links between auxiliary neurons and reticulospinal neurons are also drawn.

color cues. Green circles were interpreted as prey, while red circles were considered predators. The robot was made to operate at a slow and fixed frequency in order to avoid damaging the system.

The approach behavior was tested by placing the Envirobot in a pool with a solitary attractive stimulus (Fig. 7.5a). Likewise, the escape behavior was evaluated by recording the robot's reaction to a single repulsive stimulus (Fig. 7.5b). We then studied the system's ability to prioritize between different stimuli of the same behavioral group. This was done by exposing the platform to two attractive stimuli simultaneously (Fig. 7.5c). Finally, the behavior arbitration was studied by placing attractive and repulsive stimuli in close proximity (Fig. 7.5d). The results of these four experiments are presented in Fig. 7.5.

The robot was then made to navigate through three dynamic scenarios. Exposing the system to solitary attractive and repulsive stimuli in constant motion, verified the network's ability to drive chasing and fleeing behaviors. In a more complex scenario, we studied the platform's response to sudden changes in its environment. The scenario began by initially presenting the robot with two attractive stimuli at different distances, similar to the case in Fig. 7.5c. However, upon approaching its initial target, a predator was suddenly made to appear. The Envirobot



(a) (b) (c) (d)

Figure 7.5 – The experimental Envirobot swimming trajectories in four distinct scenarios. These experiments were conducted by placing the robot in a pool with one or more stimuli, and then tracking its trajectory during locomotion. The robot was presented with (a) a single attractive stimulus (prey), (b) a single repulsive stimulus (predator), (c) two attractive stimuli (prey), but at different locations to test the controller's stimulus prioritization capabilities. Finally, (d) The robot was presented with an attractive stimulus (prey) and a neighboring repulsive stimulus (predator) to test the controller's behavior arbitration capabilities.

was capable of changing its behavior to first escape the predator, and then target the safer prey. The chasing behavior, fleeing behavior, and trajectory readjustment behavior were recorded and made available for viewing.

7.3.2 DVS Performance

We characterized the performance of the robot when driven by dynamic vision sensors instead of frame-based cameras.

The number of complete passes through the neuron network were measured using both cameras to provide an objective measure of the computational advantage introduced by the bio-inspired vision strategy. The controller was run while the robot was exposed to no stimuli, one stimulus, and two stimuli. Each scenario was maintained for 60 seconds, and repeated for both cameras. The results presented in Fig. 7.6 indicate a dramatic improvement offered by the event-based strategy. We observed an order-of-magnitude improvement when using the dynamic vision sensors.

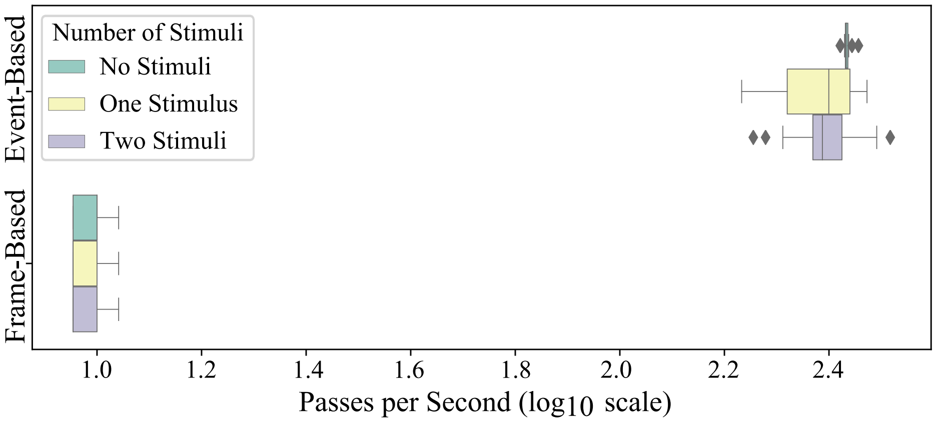


Figure 7.6 – A comparison between the neuron network computation rates achieved by the two types of image capture. The controller was run for a duration of three minutes for each type of camera. Tests began with no visual stimuli, and a stimulus was added to the robot's field of view each minute.

A final experiment studied the improvement in robot swimming precision due to the dynamic vision sensors. 20 experiments (10 with frame based cameras, and 10 with event based cameras) were performed in which the robot was made to attack a prey from a starting point approximately 3 meters away. Experiments were allowed to run until either the platform reached its target or one of the pool's walls.

A metric was developed to quantitatively evaluate robot's target approach performance. At every time instant, the angle formed by the robot's current location, the target location (the angle vertex), and its most recent past location was computed. This gaze alignment angle, $\theta_{gaze}(t)$, represented how well centered within the robot's field of view the target was at each time instant. Angles closer to 0° represented strong instantaneous alignment, while angles greater in magnitude indicated a deviation from head-on approach. Each angle was multiplied by a normalizing factor, defined as the ratio between the current distance-to-target and the initial distance-to-target. To score the robot's performance in a single trial, the gaze angles were integrated over time. The resulting value was coined the aggregate gaze alignment, A_{gaze} [$^\circ$s]. This quantity represented the amount of time spent with the target not centered within the robot's field of view. Effectively, this score served as a measure of how long the robot was able to keep the object dead-ahead.

Fig. 7.7 illustrates the trajectories for the two camera types, while Fig. 7.8a depicts the evolution of the gaze alignment for each trajectory. The statistics for the 20 experiments are reported in Fig. 7.8b. The results clearly favored usage of event-based cameras for the prey approach tasks. Use of dynamic vision sensors led to a reduction in the aggregate gaze alignment score, falling from $\bar{A}_{gaze} = 3.34\,°\text{s}$ to $\bar{A}_{gaze} = 0.57\,°\text{s}$.

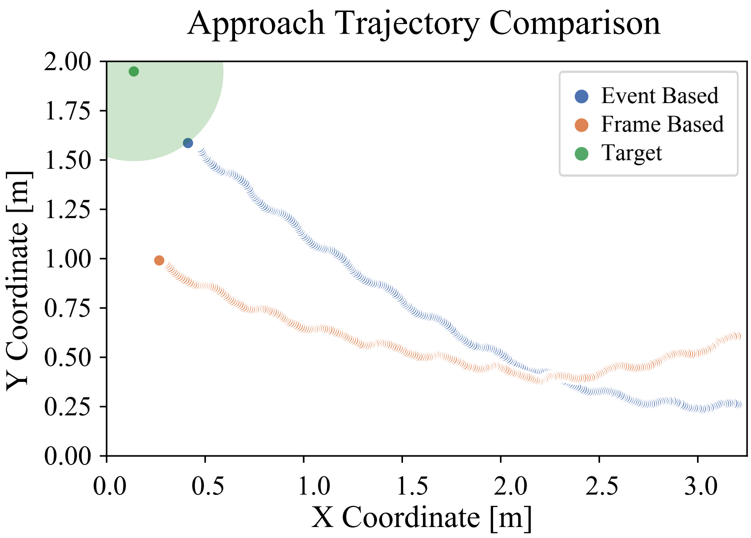


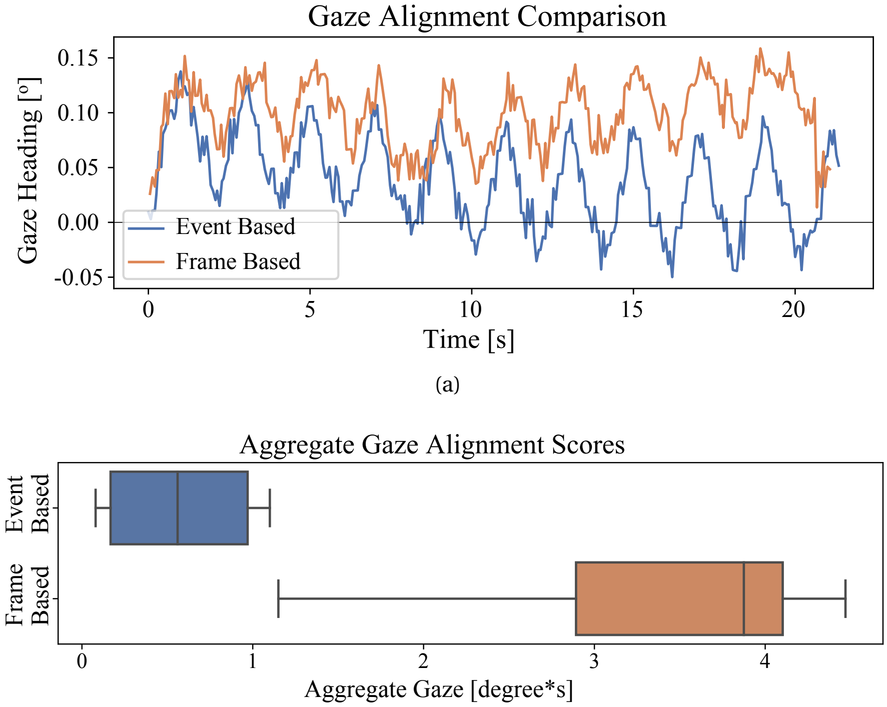
Figure 7.7 – An example of trajectories for both vision systems recorded during the experiments. The robot was placed 3 meters away from the target and allowed to perform its approach behavior. Experiments were run until the robot reached the target, or until collision with the bounds of the pool.

7.4 Discussion

7.4.1 Visual Memory

The visuomotor controller presented in this chapter was only capable of generating output signals based on instantaneous input. While introducing event based cameras to the platform was found to have improved the robot's head-on approach of attractive stimuli, the purely reactive nature of the controller made it impossible to respond to stimuli outside of its field of view. For example, the robot was found to cease performing its escape behavior upon successfully turning away from a predator such that visual contact was broken. Additionally, the controller lacked an internal model of the robot's anguilliform swimming gait. Due to stimuli rapidly shifting between the right and left eye, the platform sometimes overshot its objective.

In vertebrates, the hippocampus helps to solve the memory problem by maintaining a spatial map of its organism's surroundings [O'keefe and Nadel, 1978]. Biologically inspired vision systems seek to reproduce the functionality of such spatial maps [Rebai et al., 2012] and visual sort-term memory [Vega et al., 2013]. Kinaesthesia, on the other hand, allows organisms to estimate the relative locations and orientations of body parts. It may be possible to improve our controller by extending the visuomotor network to include a more advanced localization



(a)

(b)

Figure 7.8 – A performance comparison between event-based and frame-based cameras. (a) The gaze alignment plots corresponding to the trajectories in (a). The area under each curve was used as a final measure of how well the robot performed. An area of 4.5 °s was recorded for the frame-based trial. The event based camera experiment exhibited a significant improvement, with an aggregate gaze alignment of 0.8 °s. (b) The aggregate gaze alignment scores for the two types of camera shows a significant improvement in the robot's ability to align the target with its heading when the event based cameras were used. The mean score $\bar{A}_{gaze} = 3.34$ °s obtained using frame-based cameras, dropped to $\bar{A}_{gaze} = 0.57$ °s with event-based cameras.

and memory system. This could enhance the robot's precision while attacking prey, and allow it to retain an awareness of stimuli that have fallen outside of its field of view.

7.4.2 Behavior Adaptation & Learning

It may be interesting to study the ability of this controller to generalize its performance to previously unknown stimuli. Lamprey and other organisms have been found to be capable of adapting their behaviors based on danger cues sensed in their surrounding environments [Wagner and Bals, 2012]. Damage-released alarm cues originating from conspecific organisms under distress mark regions as high-risk, and can instigate the animals' alarm responses [Hume and Wagner, 2018]. Experiments have even shown that certain fish species are able to learn from such odor cues, which are believed to play a role in teaching organisms to recognize new organisms as predators. It has been found that combining damage released alarm cues with visual information produces learned predator recognition [Brown et al., 2011].

Using such ques as penalties or rewards, it should be possible to generalize the controller's performance when presented with novel objects. In aquatic environments chemical signals, such as the aforementioned danger cues, are dispersed by Brownian motion over time. In addition, such signals may not even be visually detectable, making them poor candidates for visual tracking (or avoidance). However, they could prove valuable as a means for providing positive reinforcement to help form new network connections. This could be done in an approach resembling the policy optimization algorithms commonly used in reinforcement learning.

Such learning algorithms could have interesting applications for environmental monitoring systems. Deploying robots capable of sensing and intelligently interpreting such chemical cues could prove to be an asset in discovering and localizing newly introduced pollutants. If the sources of such contaminants becomes visually recognizable by the system, it may even be possible to identify problems early on.

7.4.3 Underwater use of Event Based Cameras

This first use of event based cameras in an underwater environment was shown to have a positive effect on the precision of goal directed swimming in the Envirobot platform. Indeed, aquatic robot vision remains a challenge due to the effect of the medium on the dispersion and attenuation of light. These effects manifest in a significantly reduced depth of field, limiting the utility of cameras in underwater monitoring and exploration applications.

Through the use of DVS devices, the Envirobot was able to detect objects from further away than with frame based cameras. However, to simplify the challenge of underwater object detection, these experiments were conducted with a single attractive stimulus and used a flashing target. Additional work is required to rigorously characterize the performance of event based cameras in water, and to develop algorithms capable of directing robot locomotion in more complex environments.

7.5 Conclusion

The objective of this chapter was both to validate a biologically inspired visuomotor controller, and serve as the first use case of event driven cameras underwater. To achieve this, a vision-ready head was designed for use with a modular lamprey robot. The control architecture was then evaluated on its ability to reproduce the behaviors predicted in [Kamali Sarvestani et al., 2013]. Finally, the performance benefits associated with the use of event based, rather than frame based, cameras were recorded.

The model was successful in reproducing approach behavior (attacking prey) and escape behavior (fleeing predators), while prioritizing between various stimuli of the same behavioral groups. The system was also found effective in arbitrating between competing behaviors. Based on the structure of vertebrate neurovisual circuits, this controller was capable of producing complex behaviors in response to visual cues.

Following this, event based cameras were used to drive simple goal directed swimming, and system performance was compared with that obtained using frame based hardware. It was found that the biologically inspired vision hardware improved the Envirobot's ability to approach prey with reduced overshoot. This hardware also enabled the CPU to process the neuron network more rapidly, allowing the leaky integrate-and-fire neurons to adopt more biologically accurate parameter values. The challenges associated with this hardware are an inability to detect color, a lack of algorithms for image rendering in aquatic environments, and a limited understanding of the device performance underwater.

This chapter offers contributions to both the fields of robotics and neurobiology. The vision-ready Envirobot offers an ideal platform to investigate vision-controlled locomotion in anguilliform swimmers such as salamander and lamprey. By their very nature, the dynamic vision sensors used in this study are ideal for research into optic-flows and how they are used by organisms to direct swimming. At the same time, this chapter contributes to the state-of-the art in underwater robotics. The computational and sensory advantages offered by the DVS devices may lead to improvements in applications such as marker tracking on active beacons, environment monitoring, or search and rescue systems.

8 Camera Placement on a Snake Robots

The previous chapter made use of a very simple body morphology with a chain of modules having parallel rotation axes. The challenging part in terms of hardware were waterproofing modules and integrating a vision system. In this chapter, we use a similar body morphology, but, on solid ground. Due to higher friction of terrestrial serpentine locomotion, the chain is modified from only vertical rotation axes to alternating axes (horizontal axes followed by a vertical one vice versa). That morphology change results in capabilities that can replicate serpentine locomotion modes such as sidewinding, rolling or linear progression. The aim of this chapter is to analyze advantages of various camera locations of snake robots for different applications. Such a chain morphology is one of the most commonly used ones in the literature. Almost every study, that makes use of cameras, places a camera on the head, looking straight. In this chapter we point out different possibilities.

Reference publications

- This chapter is based on ***Mehmet Mutlu***, *Kamilo Melo, Massimo Vespignani, Alexandre Bernardino and Auke Ijspeert. "Where to place cameras on a snake robot: Focus on camera trajectory and motion blur"* IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) 2015.

  My original contributions

  - Coming up with the concept
  - Simulating the snake robot
  - Writing the manuscript

8.1 Introduction

In the literature related to robotic search and rescue (SAR), it is common to find mentions to "Snake robots" as potential tools for accessing narrow cavities in places that any other platform cannot reach [Murphy, 2014]. With the exception of the Active Scope Mechanism [Huff et al., 2012], to the date, there is a lack of deployments of such technology in real missions due to either (i) the level of readiness of the snake robot platforms [Melo et al., 2013], [Hiroya Yamada, 2013], [Wright et al., 2012], their cost and complexity [Liljeback et al., 2014], [Rollinson et al., 2014], or simply because they do not comply with regulations needed to operate in certain environments [Murphy, 2014]. We consider that there is still a more relevant factor that delays the introduction of these robots as real assets for SAR missions. The definition of a task that a snake robot can perform better than other robots is still fuzzy. For instance, tasks like assessing the structural status of a collapsed building with limited access, provide video feedback of interest points (including possible victims) where a borehole is the only access through the rubble, and so on, omit many details of how the robot will perform the task with limited perception equipment on board or how capable the remote operator is (or the robot itself if it is meant to be autonomous) to control the motion and task execution.

As perception become crucial in these types of tasks, as many times mentioned in the literature, the use of a snake robot as a mobile extension of a camera [Huff et al., 2012] prevails. In fact, several researchers had come with this idea and intuitively incorporate a camera in the "head" of their snake robot designs [Hiroya Yamada, 2013, Wright et al., 2012, Liljeback et al., 2014, Rollinson et al., 2014]. Two possible ways to exploit a camera mounted on the head of the robot are for video surveillance, and for navigation. In the former, the camera acts as a simple final tool of a redundant manipulator, operated once the robot is positioned in place while guaranteeing some level of mechanical stability. The later, is more complex, as it proposes an interplay between computer vision algorithms and locomotion controllers for snake robots. To the best of our knowledge, there is scarce literature available for the mentioned topics. The references above report some robot implementations that incorporate a camera in the head module. However, there is no information on their effective use to provide data for navigation during locomotion.

Nevertheless, related work is available in the control of a snake robot, to stabilize the motion of the robot's end modules as a gait progresses [Wu and Ma, 2011, Tesch et al., 2013]. Particularly, in [Tesch et al., 2013], optimization techniques are used to the problem of moving a snake robot (an expensive system) optimizing simultaneously the gait performance as well as the ability to maintain motion of a specific module in certain way i.e. keeping the head module with minimal changes in their orientation as the gait progresses. These algorithms effectively control the robot with the expected results obeying defined policies, however their use with a camera to acquire useful video/image data for self-navigation is missing. On the other hand, in [Florez et al., 2013], the authors use the Modular Snake Robot Lola-OP™ [1], with a camera added in the head module, running standard video stabilization algorithms in order

---

[1] Developed by KM-RoBoTa s.a.s. (http://km-robota.com)

to correct the video as the robot moves. But there is no mention of neither coordination of video processing during the actual robot gaits, nor any video database useful for navigation.

All these aforementioned implementations of camera-on-head of a snake robot rely on the fact that the robot is a serial manipulator featuring a slender form and small cross section. Thus, all these robots incorporate their cameras intuitively placed pointing along the longitudinal axis of the robot. The question we want to address in this chapter is: is this camera placement in a snake robot (i.e. on the head and pointing longitudinally) in fact optimal for navigation as it seems to be for inspection and how does this depend on the type of the gait?

We want to provide a systematic approach for studying the placement of a camera in a Modular Snake Robot (MSR) since the use of a snake robot as a camera manipulator in order to acquire video data of a selected scene is straightforward. The idea behind our study is twofold. In the first place, we are simultaneously observing the natural coverage of camera with respect to absolute motion of the MSR for different gaits and for different camera locations. The second idea is tracking motion blur amount in images captured by an onboard camera during the same gaits and for the same proposed camera locations. On the other hand, our interest to find a correlation between camera trajectories and the video motion blur is because of the fact that a MSR like Lola-OP™, shown in figure 8.1, is capable of "holonomic" and repeatable motions. Additionally, the payload (weight and volume) of a MSR is not always constrained. This makes worth the analysis of different locations as well as different orientations of a fixed camera in a MSR, in order to explain quantitatively where and under which conditions a camera should be mounted, for maximizing the information gathered and ensuring quality for the processing.

The outline of this chapter is as follows. In Sec. 8.2, the description of the robot, the gaits used, the camera position and orientation are shown. Then, in Sec. 8.3 the expected motion blur of images captured by the camera while MSR is performing locomotion is reported. Further implications of motion blur and locomotion correlation is discussed in Sec. 8.4 and some real life deployment examples are given. Finally, conclusions are given in the last section.

8.2 Camera trajectory during locomotion

A common framework to control modular snake robot's locomotion is provided by [Melo and Paez, 2014].

$$Q(n,t) = \begin{cases} O_e + A_e \sin(w_e + n/\lambda_e + \delta) & \text{if } n \text{ is even} \\ O_o + A_o \sin(w_o + n/\lambda_o) & \text{if } n \text{ is odd} \end{cases} \tag{8.1}$$

In Eq. (8.1), O is the offset angle that represents the center of oscillation, A is the amplitude of oscillation, w is the oscillation frequency, λ is a body wave number and δ is the phase difference between horizontal and vertical joints.

Depending on the application, the operator may need to keep the line of sight fixed to a certain

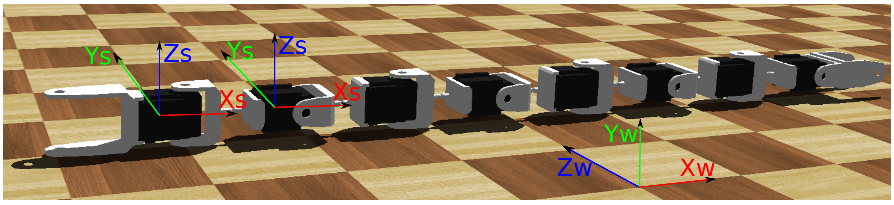


Figure 8.1 – Modular Snake Robot Lola-OP™ showing coordinate frames of global world and local sensors in a simulation environment.

area or try to observe large space during locomotion. Observing the trajectory of the camera for different types of gaits gives an idea about the coverage of the camera. There exists a trade-off in the stability of the image and covered area. Depending on the requirements of the application the definition of optimal camera placement will differ. Hence, a comparative study will be presented for camera field of view and pose.

Different gaits of the MSR are simulated in Webots 7.4.3. In Fig. 8.1, coordinate frames of the simulation world and local sensors that are placed on each module can be seen.

Rolling, sidewinding and linear progression are three main gaits of snake robots. In this chapter, one example from each of rolling, sidewinding and linear progression will be studied. Frequency of the gaits, f, is chosen to be same for even and odd modules and for all gaits and it is set to 1Hz. All gait parameters for all experiments can be seen in the Table 8.1. For each of the locomotion examples, the camera is assumed to be placed in two separate locations. Either, looking straight head camera or middle camera pointing towards sideways with respect to the snake longitudinal axis. The head camera is placed on the same location as the first sensor coordinate frame shown on the Fig. 8.1 with the orientations given in Table 8.1. Similarly, the middle camera is placed on the fourth module with the orientations given in Table 8.1.

For visualizing the camera field of view, pose of the each module of the snake robot is logged for short duration. From Fig. 8.2 to Fig. 8.7 trajectories of all 8 modules of the snake are shown. The color of a single module's path is drawn as changing from black to specific color. The very first black dot in the beginning of each module's path is captured at $t = 0sec$ and position of the module is logged for three or two (only linear progression) seconds for each run. The color change from black to a saturated color progresses linearly from $t = 0$ to end of the run. Each module's path is plotted with a different color starting with red on the head, followed by green on the second module, blue for the third one and so on. The pose of individual modules are logged in 1kHz. However, position of the modules are illustrated with dots which are drawn 50ms apart on the plots. Hence the density of the dots throughout a path gives an intuition about the linear velocity of the module. The denser the dots are, the longer the module stays around those dots which means that module moves relatively slow on that region. The overall snake velocity is marked with a blue arrow in each figure. In addition to the trajectory of modules, the orientation of the camera is also shown throughout the locomotion from Fig. 8.2 to Fig. 8.7. Orange lines starting from the camera node represent the primary axis of the

Table 8.1 – Parameters and Coordinate Transformations of Camera Placement for Examined Gaits

Rolling	
Amplitude of osc., $[A_e, A_o]$	$[40°, 40°]$
Offset of osc., $[O_e, O_o]$	$[0°, 0°]$
Delta, (δ)	$\pi/2rad$
Frequency, (f)	$1Hz$
Lambda, (λ)	100000
Camera coord., head-straight	$[x_c\ y_c\ z_c]^T = [-y_{s1}\ z_{s1}\ -x_{s1}]^T$
Camera coord., middle-side	$[x_c\ y_c\ z_c]^T = [-x_{s4}\ z_{s4}\ -y_{s4}]^T$
Sidewinding	
Amplitude of osc., $[A_e, A_o]$	$[10°, 40°]$
Offset of osc., $[O_e, O_o]$	$[0°, 0°]$
Delta, (δ)	$\pi/4rad$
Frequency, (f)	$1Hz$
Lambda, (λ)	$4/\pi$
Camera coord., head-straight	$[x_c\ y_c\ z_c]^T = [z_{s1}\ y_{s1}\ -x_{s1}]^T$
Camera coord., middle-side	$[x_c\ y_c\ z_c]^T = [x_{s4}\ y_{s4}\ z_{s4}]^T$
Linear Progression	
Amplitude of osc., $[A_e, A_o]$	$[0°, 30°]$
Offset of osc., $[O_e, O_o]$	$[0°, 0°]$
Delta, (δ)	$\pi/2rad$
Frequency, (f)	$1Hz$
Lambda, (λ)	$4/\pi$
Camera coord., head-straight	$[x_c\ y_c\ z_c]^T = [-y_{s1}\ z_{s1}\ -x_{s1}]^T$
Camera coord., middle-side	$[x_c\ y_c\ z_c]^T = [-x_{s4}\ z_{s4}\ -y_{s4}]^T$

camera (z) and are oriented along the camera's viewing direction. The actual field of view depends on lenses and is application-specific, whereas the camera direction is universal for all applications. The color gradient of camera's primary axis lines, from black to orange, has the same time progress notion with path of the snake's module position. Camera pose is plotted with fine sampling through the end and coarsely in the beginning of each run. Coarse parts also have coordinate axes plotted on them. Each plot also has zoomed versions of camera pose for the first second of locomotion.

8.2.1 Rolling motion

The rolling motion of a snake robot involves continuous rotation of the body around longitudinal axis, as can be seen in Fig. 8.2 and Fig. 8.3 for three seconds. In these figures, the robot is moving on x-z plane where y axis is the height from ground. The robot takes an arch shape to move sideways when offset terms of Eq. (8.1) are zero. This motion is known to be robust in rough terrain and slopes [Zhen et al., 2015].

Head camera looking straight

The first observation that can be made about placement of the camera on the head looking away in the longitudinal direction of the snake is that the camera is not pointing towards the motion direction. Center of the field of view of a camera is looking more than 90 degrees away from it as it can be seen in Fig. 8.2. This means the robot would not know about most of the approaching obstacles. Moreover, it is not easy to guide the robot with the information obtained from captured images if the camera is pointing away from the locomotion direction.

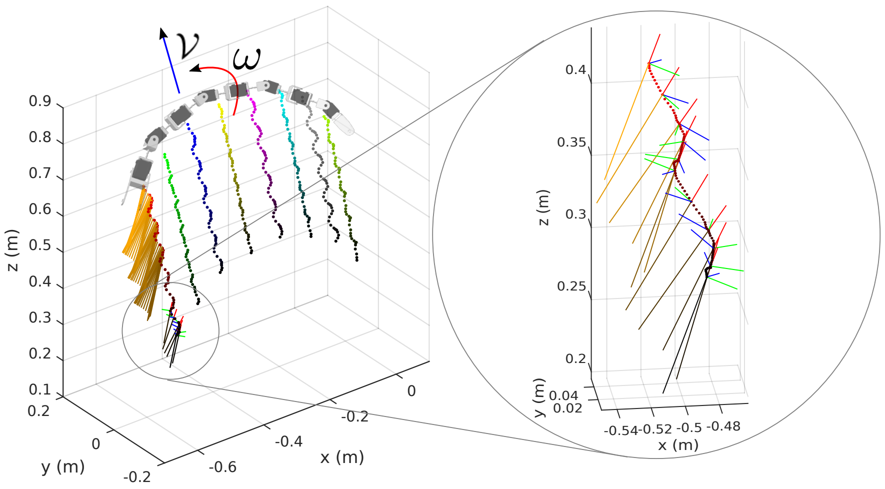


Figure 8.2 – Camera and snake trajectory during rolling gait. Camera is placed on head looking straight.

Camera field of view stays quite close to previous field of views. However, continuous rolling of the snake means rolling of the camera around itself which can be seen clearly on the zoomed part of Fig. 8.2. Hence the view would be rolling and not stay very stable while rolling.

Middle camera looking sideways

Placing the camera on the fourth module of the robot with orientation to point sideways results in very large displacement of the field of view between image frames during rolling. Although camera motion that is shown in Fig. 8.3 would be naturally covering a large area like a torus around the snake, camera rotation velocity is quite fast. Large rotational velocity results in large displacements of pixels on the image plane during the exposure period which usually has negative effect on visual SLAM algorithms and can disturb convergence of SLAM.

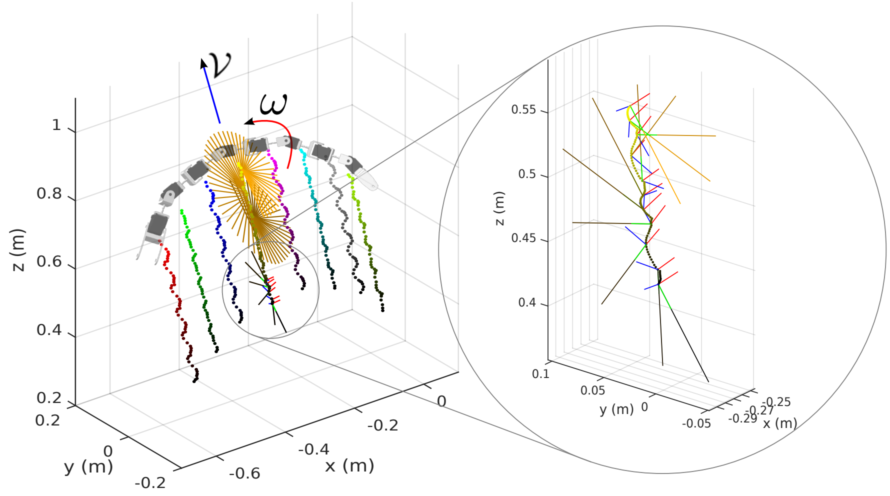


Figure 8.3 – Camera and snake trajectory during rolling gait. Camera is placed on the fourth module looking sideways.

8.2.2 Side-winding motion

Sidewinding is a very fast gait. Similar to rolling, motion of the robot is directed sideways, with a small forward component. However, the number of contact points with the ground is much smaller than rolling motion. Sidewinding is commonly performed by snakes moving on granular medium and surface contact points of the snake follows discontinuous trajectories. Slight oscillatory motion, as seen on Fig. 8.4 and Fig. 8.5 is mainly caused by the asymmetrical structure of the snake robot used in the simulations. Sidewinding plots span three seconds of experimental data.

Head camera looking straight

The camera placement in this case is similar to the rolling case. Fig. 8.4 shows camera points mostly perpendicular to the direction of locomotion. However, camera orientation does not have continuous rotation component unlike the rolling gait. Also the average speed of the sidewinding is higher than rolling. Translational progress of the camera is not uniform in sidewinding. Camera stays around ground contact point of the head module for a while and moves quite fast within the swing phase.

Middle camera looking sideways

When the camera is mounted on the direction of the locomotion in a middle module, it can be quite informative for sidewinding gait. It is not only pointing through the locomotion

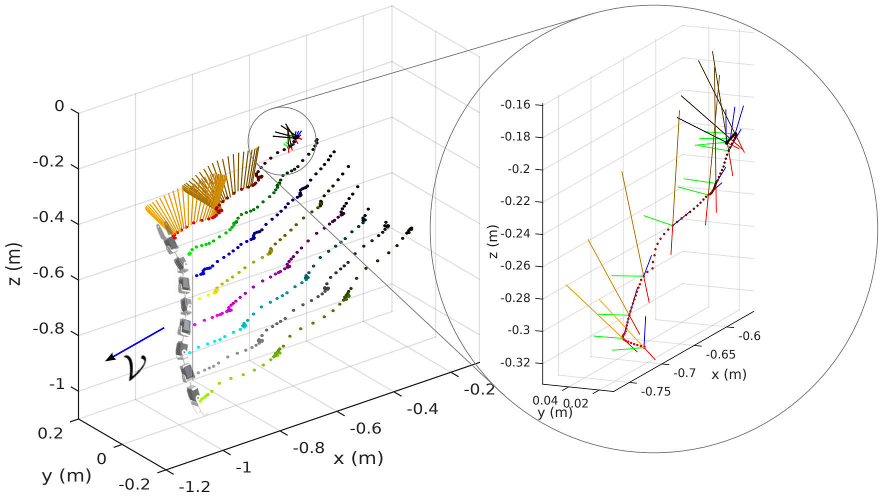


Figure 8.4 – Camera and snake trajectory during sidewinding gait. Camera is placed on head looking straight.

direction, but also exhibiting a circular exploratory motion around the snake direction, even though exploration may not cover a wide area. Furthermore, orientation of the camera always stays around the initial orientation. This type of camera placement can be useful to teleoperate the robot during locomotion in a remote place.

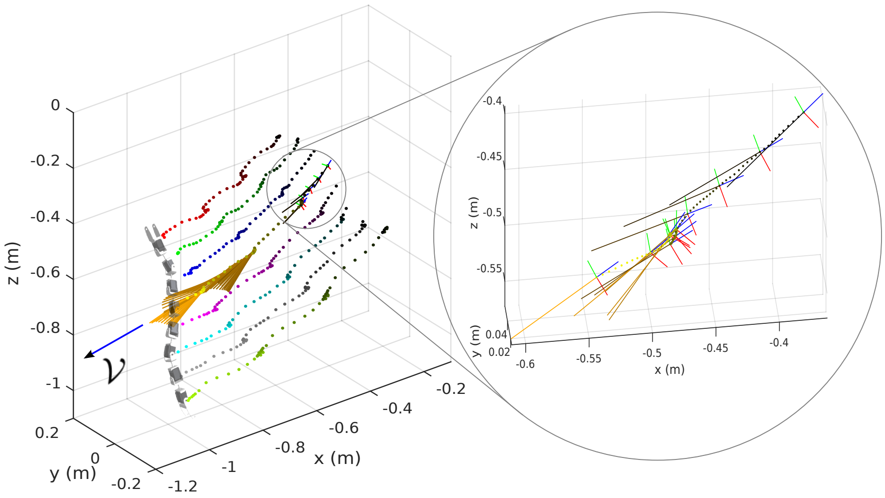


Figure 8.5 – Camera and snake trajectory during sidewinding gait. Camera is placed on the forth module looking sideways.

8.2.3 Linear progression motion

Linear progression is examined as the last type of gait in this study. Linear progression can be obtained with horizontal waves and vertical waves. However, due to high friction between ground and snake, horizontal waves cause either too much energy loss or do not result in proper locomotion. Therefore, only vertical wave generated linear progression is considered in this part. The average speed of the robot is much slower than rolling and sidewinding while performing linear progression. Fig. 8.6 and Fig. 8.7 covers only two seconds of data, because position samples of consecutive modules start to overlap for longer runs.

Head camera looking straight

The most noticeable characteristic of camera motion in this gait and placement is that there is almost no rotation with respect to two of the axes. Dominant rotation is mainly on pitch. Also the translational speed of the camera is quite low which makes the camera quite steady. The field of view is oscillatory around an offset and on the direction of motion, which helps to control the robot in teleoperated situations.

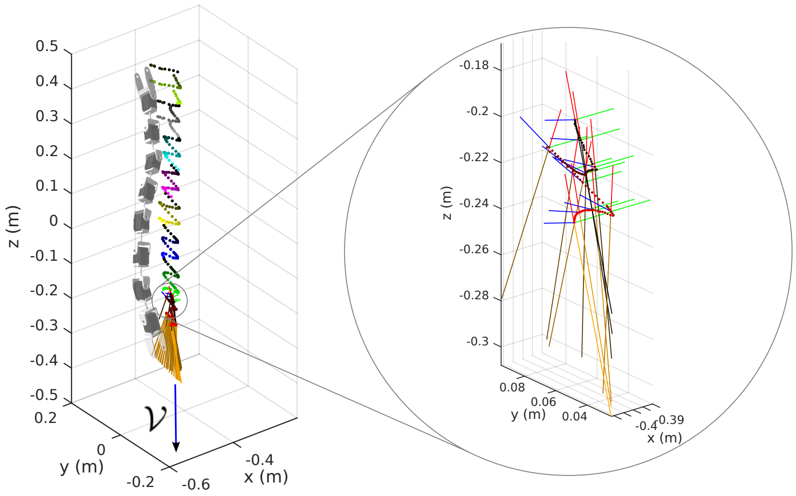


Figure 8.6 – Camera and snake trajectory during linear progression gait. Camera is placed on head looking straight.

Middle camera looking sideways

When the camera is mounted on the middle part of the snake and pointing towards sideways, camera motion is minimal and there is only slow oscillatory roll and translational motion of the camera. Also the region that the camera observes is quite stable.

8.3 Motion blur of camera during locomotion

In vision applications, the quality of captured images is equally important as the trajectory of the camera. Most of the SLAM applications require sharp images to extract features correctly, the abundance of features and low pixel displacement during locomotion. In many cases, features are extracted from the texture of surrounding environment. That stage may require

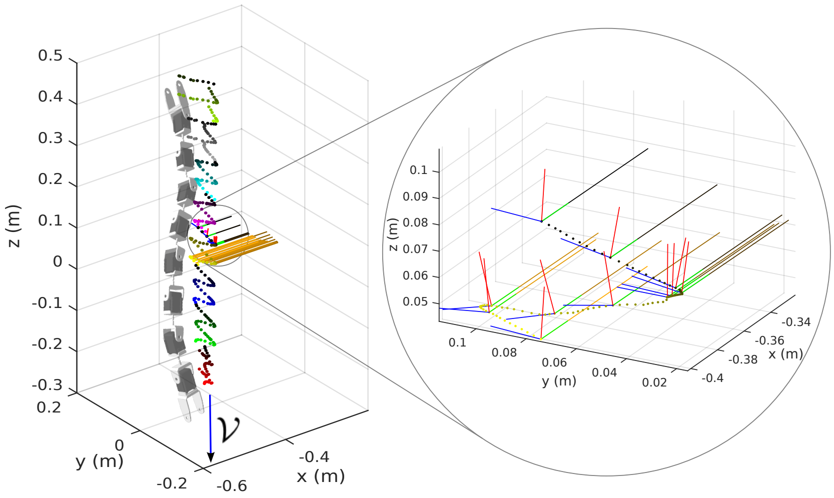


Figure 8.7 – Camera and snake trajectory during linear progression gait. Camera is placed on the fourth module looking sideways.

engineering the environment, if the natural texture is poor. In realistic search and rescue missions, it is too hard and time consuming to modify the environment. Also, there can be insufficient light in the environment that would push the camera to capture images at higher exposure time, resulting in higher motion blur. However, capturing relatively sharper images by examining motion of the robot is possible. Motion blur is one of the most pronounced causes of distortions in robot vision. Certain amount of motion blur can be recovered with proposed motion deblurring algorithms in the literature [Cho and Lee, 2009]. However, when the amount of motion blur is extreme, information loss can be irreversible. One of the most common methods to avoid motion blur is using stabilization platforms [Hilkert, 2008]. The structure and size of snake robots are usually not well suited for adding mobile stabilization platforms. On snake robots, even using the first few degrees of freedom for camera stabilization instead of locomotion can be effective up to a certain degree. However, one of the biggest challenges in snake locomotion is the state estimation due to complex ground-robot interaction. Furthermore, high vibrations and impacts observed on the body makes visual SLAM on snake robots during locomotion a very hard problem [Liljebäck et al., 2013].

In this part of the chapter, motion blur formation during snake locomotion will be analysed and motion blur characteristics of different gaits will be explained. Motion blur can be caused by the movement of objects on the scene or movement of the camera. Although there can be exceptions, the environment is expected to be mostly stationary for the robotics applications in a search and rescue area. This chapter addresses motion blur only caused by the egomotion of the camera. The effect of rotation and translation motions are different on motion blur. In particular, the amount of motion blur caused by translation depends on scene depth in addition to translational velocity, but motion blur caused by rotational motion has no scene depth dependence. Furthermore, measuring translational speed on a snake robot is more challenging than measuring rotational speed. Despite, a simple MEMS gyroscope can directly give rotational speed, translational speed can be estimated from the integration of accelerometer measurements in which calibration and bias errors accumulate. Also, the effect of translation on motion blur can be negligible when the scene depth is sufficiently large. In open fields, scene depth is large and makes the translational motion blur relatively small. Even though snake robots may need to work in cluttered environments where scene depth is small, we will only consider motion blur caused by the rotational motion of the snake. One specific observation is that rotational over translational speed ratio of the body elements on a snake robot is higher than most of the conventional wheeled or tracked robots due to natural movement of snakes. Therefore, considering only the effect of rotational motion on motion blur will still address significant amount of the motion blur source in snake robots.

A motion blur metric, Motion-Based Motion Blur Metric (MMBM), which estimates the amount of motion blur on a camera undergoing rotational motion is given in [Mutlu et al., 2014]. Higher values of the metric corresponds to higher blur. The proposed metric models optical flow of pixels out of rotational speed measurements and approximates the motion blur that would be observed on images. The definition of MMBM (μ) is given as

$$\mu := \frac{1}{\Delta u \Delta v} \iint_{u_{min}, v_{min}}^{u_{max}, v_{max}} \sqrt{\dot{u}^2 + \dot{v}^2}\, \mathrm{d}v \mathrm{d}u, \tag{8.2}$$

where u & v are image sensor coordinates, Δu & Δv are image sensor size and $\dot{u}$ & $\dot{v}$ are optical flow vectors derived from gyroscope measurements in following way,

$$\begin{aligned} \dot{u}^2 + \dot{v}^2 = & \frac{w_y^2}{f^2}\boldsymbol{u^4} + \frac{w_x^2}{f^2}\boldsymbol{v^4} - 2\frac{w_x w_y}{f^2}\boldsymbol{u^3 v} - 2\frac{w_x w_y}{f^2}\boldsymbol{u v^3} \\ & + \frac{w_x^2 + w_y^2}{f^2}\boldsymbol{u^2 v^2} + (2w_y^2 + w_z^2)\boldsymbol{u^2} \\ & + (2w_x^2 + w_z^2)\boldsymbol{v^2} - 4w_x w_y \boldsymbol{uv} - 2w_x w_z f\boldsymbol{u} \\ & - 2w_y w_z f\boldsymbol{v} + (w_x^2 + w_y^2)f^2, \end{aligned} \tag{8.3}$$

where f is the focal length of the camera and $[w_x, w_y, w_z]$ is rotational velocity of the camera. Authors also show a μ based real-time image capturing technique to reduce the average amount of motion blur on captured images from a camera mounted on a six-legged robot RHex in [Mutlu et al., 2014]. In this current chapter, μ will be used for motion blur characterization. The value of μ assumes frames captured with unit exposure time. If the value of exposure is not fixed in the application μ needs to be scaled with the exposure time. We also consider fixed exposure time. μ will be calculated for the different snake gaits examined in Sec. 8.2.

8.3.1 Rolling motion

Unlike most of the legged straight locomotion cases, snake rolling involves continuous rotation as seen in w_z and w_x parameters of Fig. 8.8 and Fig. 8.9 respectively.

Head camera looking straight

When the camera is on the head and looking straight during rolling, there is quite dominant rolling (w_z) motion of camera. Although there are also oscillations on yaw (w_z) and pitch axis w_x, the resulting motion blur has small oscillations around certain value. In roll dominated camera images, center of the images would be sharper and the periphery would be more blurred. Having some sharper region around the middle parts can be exploited in vision tasks.

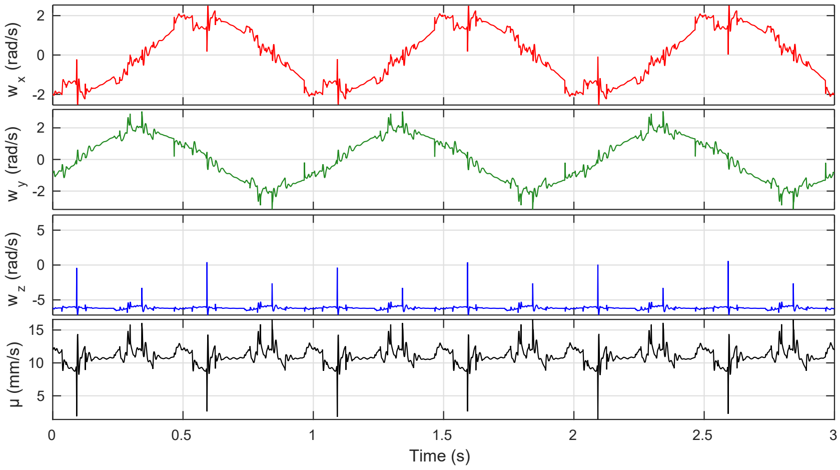


Figure 8.8 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when straight pointing camera is placed on head of a rolling snake.

Middle camera looking sideways

When Fig. 8.8 and Fig. 8.9 are compared, the similarity in the motion of head module and middle module can be easily seen. Main difference appears in the axes assignment to camera motion due to camera direction change. When high continuous rotation axis of the snake robot is assigned to the pitch axis of the camera, the amount of motion blur significantly increases. This happens because pitch motion creates stronger motion blur on whole image compared to roll of camera. Hence, this particular camera placement results in excessive motion blur during snake rolling gait.

8.3.2 Side-winding motion

Sidewinding exhibits high speed swing phases and relatively stationary stance phases for each module. Unlike the rolling gait, it is closer to the legged locomotion characteristics in terms of having rotational oscillations around zero.

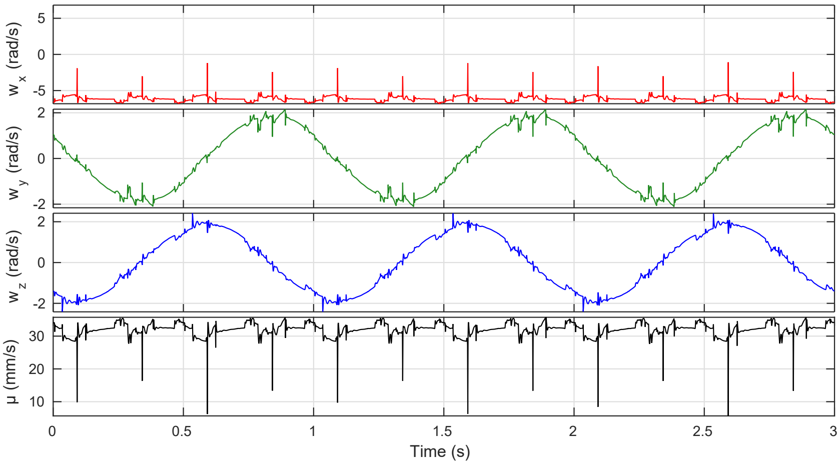


Figure 8.9 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when straight pointing camera is placed on the forth module of a rolling snake.

Head camera looking straight

The motion of camera in swing phase highly fluctuates. Although the maximum motion blur levels are close to the rolling gait, the fluctuating levels of oscillations can be exploited as explained in [Mutlu et al., 2014]. For instance camera can be triggered to capture images only when μ is sufficiently small to obtain only sharper images.

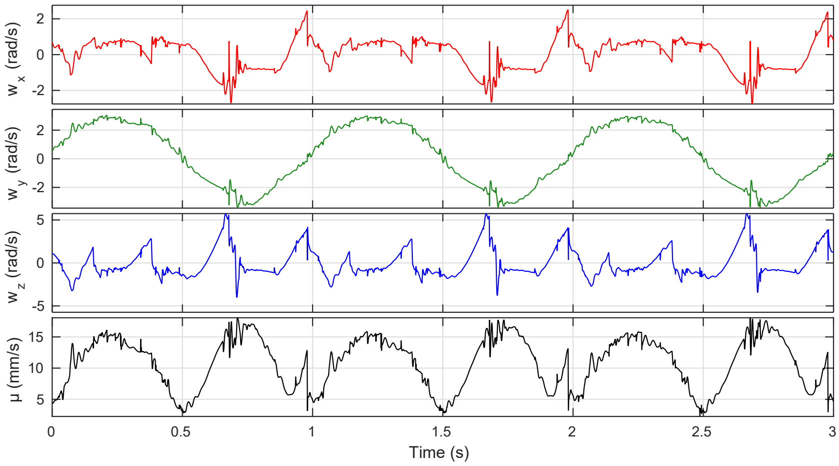


Figure 8.10 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when straight pointing camera is placed on head of a sidewinding snake.

Middle camera looking sideways

Changing the camera location to middle and changing the direction changes the motion blur fluctuations significantly. For instance, motion blur remains in lower values for longer durations, even though the maximum values go higher.

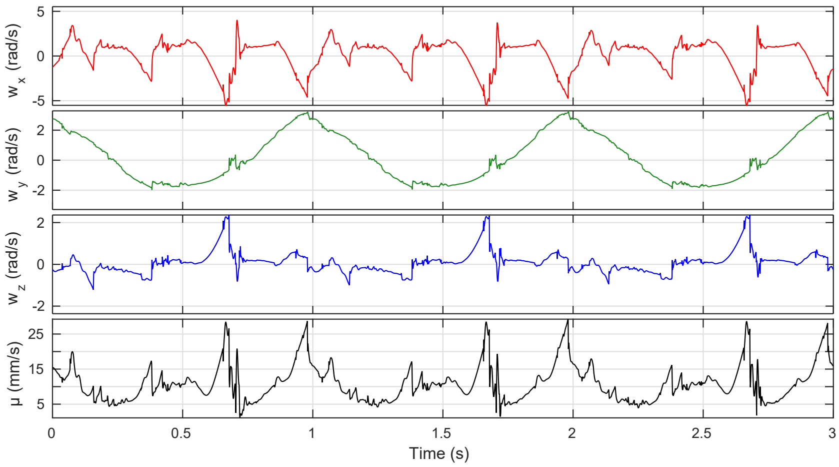


Figure 8.11 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when sideways pointing camera is placed on the forth module of a sidewinding snake.

Middle camera looking up

Placing a camera to look upward on a snake robot can be needed for certain scenarios. For example, when collaboration of ground and aerial robots is needed, upward looking camera can be useful. Moreover, for changing camera from sideways to up does not require any human intervention. Simply exchanging even and odd module amplitudes (A_e and A_o) in Eq. (8.1) is enough to do the side to up transition. Once the camera is pointing upward, mapping between camera coordinates local gyroscope axes would be $[x_c\ y_c\ z_c]^T = [x_{s4}\ \ -z_{s4}\ y_{s4}]^T$. Since, μ is calculated for the same snake module as in sideways looking camera, only μ is given in Fig. 8.12 and it appears to be quite similar to μ in Fig. 8.11

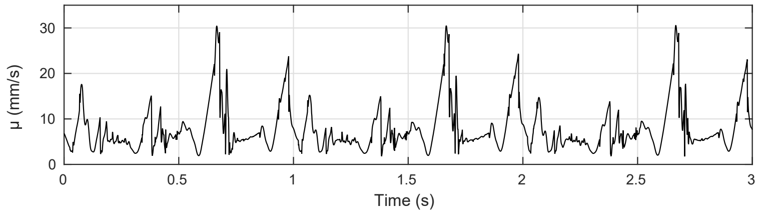


Figure 8.12 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when up pointing camera is placed on the forth module of a sidewinding snake.

8.3.3 Linear progression motion

Finally the linear progression gives relatively lower motion blur compared to rolling and sidewinding when Fig. 8.13 and Fig. 8.14 are examined. The appealing feature of linear progression is that it has significant rotation only in a single axis and small impact related disturbances in the other axes.

Head camera looking straight

Rotation of the head corresponds to pitch motion when the camera is looking straight. Hence, the amount of motion blur is still considerable although it is smaller than other gaits. Also the fluctuating behavior allows the acquisition of sharper images at lower values of μ during locomotion.

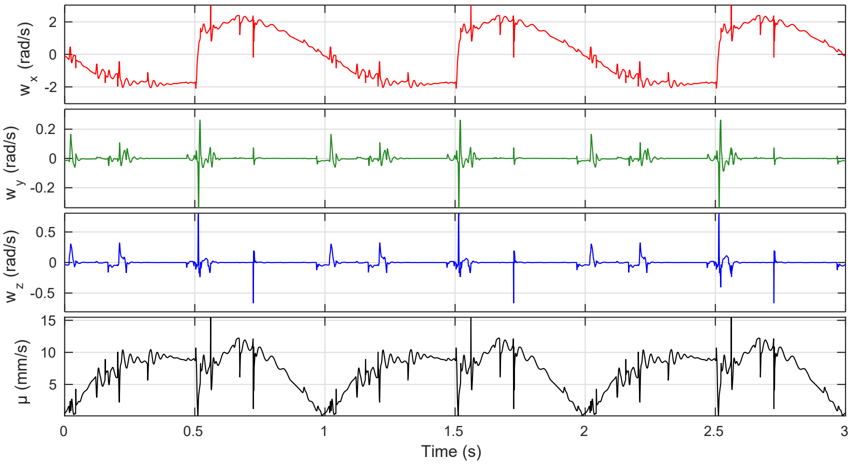


Figure 8.13 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when straight pointing camera is placed on head of a snake doing linear progression.

Middle camera looking sideways

The sideways looking camera exhibits the least amount of motion blur among all examined snake gaits since the main rotation is small and it corresponds to roll of camera.

Middle camera looking up

When the camera is oriented upwards, camera axes changes to $[x_c\ y_c\ z_c]^T = [-y_{s4}\ x_{s4}\ z_{s4}]^T$. Snake can also switch camera direction from side to up without external intervention for linear progression. μ automatically increases since the single rotating axis now corresponds to pitch motion of camera.

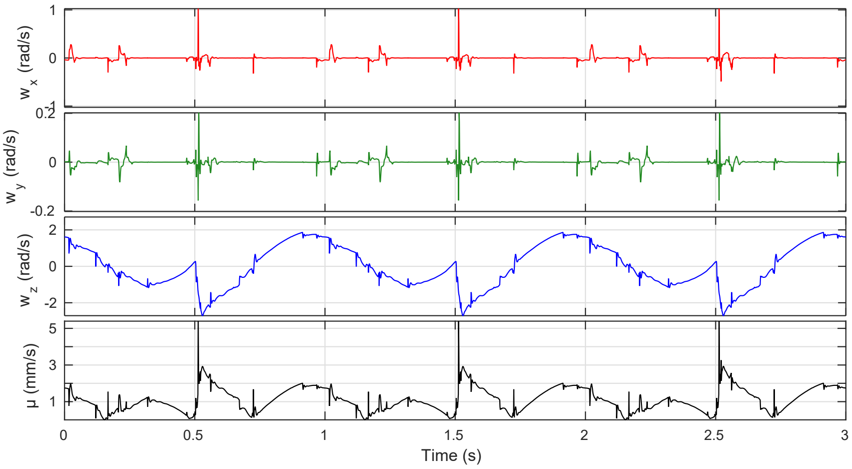


Figure 8.14 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when sideways pointing camera is placed on the forth module of a snake doing linear progression.

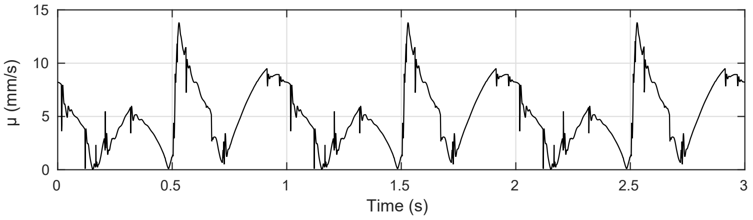


Figure 8.15 – Roll(w_z), pitch(w_x) and yaw(w_y) rotational speeds of camera and corresponding μ when up pointing camera is placed on the forth module of a snake doing linear progression.

8.4 Discussions and demonstrations

Visual data during the snake locomotion can be highly corrupted due to body oscillations and impacts especially when there is not enough light in the environment and exposure time is high. With μ we have an estimate of motion blur amount on images captured by camera on a snake robot during locomotion. Lower values of μ shows sharper images if image is captured at that moment. Table 8.2 shows that linear progression is very advantageous in terms of average motion blur. On the other hand, having fluctuations on μ can be exploited to capture relatively sharper images during locomotion. For instance, even if sidewinding and rolling head cameras have similar average motion blur, more variation of sidewinding can be exploited.

In order to give concrete illustration of the motion blur levels and camera motion of examined gaits, consecutive images captured during rolling, sidewinding and linear progression are provided in Fig. 8.16, Fig. 8.17 and Fig. 8.18 respectively. The sharper parts around the middle of rolling images can be seen in Fig. 8.16. Fig. 8.17 shows behavior of fluctuating μ although

Table 8.2 – MMBM statistics for different snake gaits

Locomotion	**Camera**	**Min.**	**Max.**	**Avg.**	**Std. Dev.**
Rolling	**Head**	1.41	16.60	11.05	1.22
	Side	5.67	35.74	32.07	2.19
Sidewinding	**Head**	2.26	18.07	10.87	4.07
	Side	1.10	29.24	10.99	5.14
	Up	1.75	30.56	7.78	5.12
Linear Progression	**Head**	0.098	15.47	7.18	3.04
	Side	0.0072	5.39	1.12	0.65
	Up	0.0033	13.81	5.17	3.01

the illustrated part is mainly located around smaller μ region. Images can get sharper or more blurry during locomotion. Finally, low to moderate level of blur of linear progression is shown in Fig. 8.18.

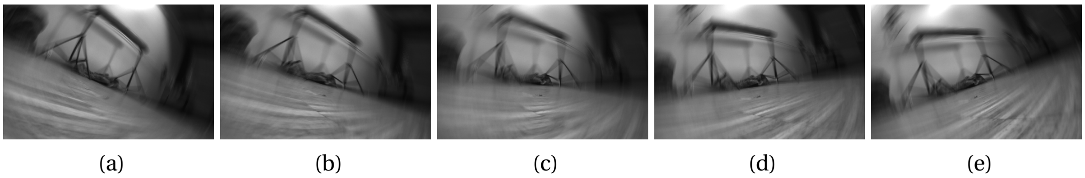

(a) (b) (c) (d) (e)

Figure 8.16 – Consecutive images captured during rolling gait by camera mounted on head of the robot pointing straight.

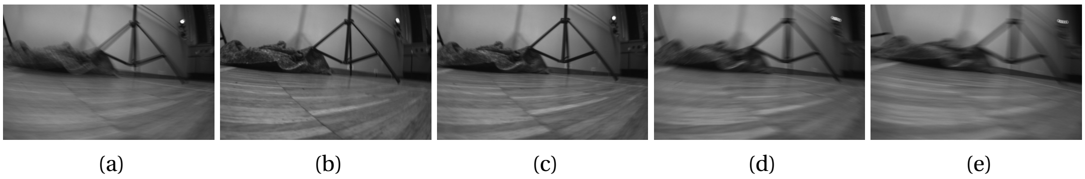
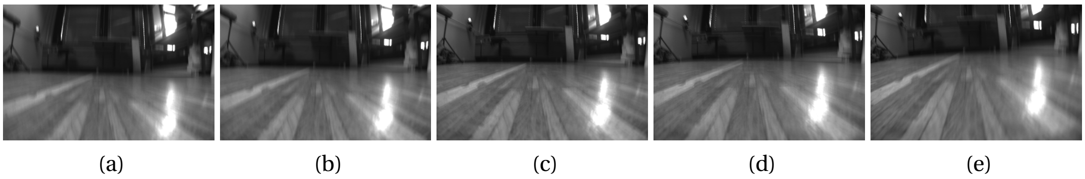

(a) (b) (c) (d) (e)

Figure 8.17 – Consecutive images captured during sidewinding gait by camera mounted on head of the robot pointing straight.

8.5 Conclusions

We have presented a systematic analysis of camera trajectories during three different types of snake locomotion when the camera is placed on the head pointing straight and in the middle pointing sideways. Camera pose is explicitly computed during different gaits and we were



(a) (b) (c) (d) (e)

Figure 8.18 – Consecutive images captured during linear progression gait by camera mounted on head of the robot pointing straight.

able to analyze implications of different camera locations on field of view coverage during the natural locomotion. We also relate snake motion to motion blur on captured images.

Although linear progression gives sharper images on average, there is no globally optimal location for camera location for all tasks. But, knowing the implications of different camera locations during different gaits, information input in visual tasks can be maximized. For example, if the camera information will be used to steer the robot during sidewinding or rolling, placing the camera on the side is more useful than placing it on the head since head camera will not be pointing to the direction of locomotion. If having multiple onboard cameras is a possibility, we can place them as to obtain maximal benefits. The proposed analysis can be used for designing new robots for a given desired gait or to choose suitable gaits for already existing robots that are equipped with cameras. Ultimately, it will contribute to the quality of visual SLAM during snake locomotion.

9 Locomotion: effects of local passive and active compliance

In this final study of locomotion studies on modular structures, we step up to legged locomotion. We analyze the effects of both hard (mechanical) and soft (control-based) compliance in legged locomotion. We choose a quadrupedal structure as it is both dynamic, quasi stable and exhibits various gaits. We create easy-to-swap compliant parts on the close to joints. On top of that, we use Tegotae-controller. Tegotae rule is based on uncoupled oscillators for each leg and a force feedback from feet to modulate the phase of oscillators. Although oscillators are uncoupled, they still converge to a gait such as trot, pace or gallop due to mechanical interaction via being part of the same body. We aim to gain insights on emerging gaits under changing compliance and changing surface type in this experimental study. The insights of this study can yield potential use in SRMRs in general.

Reference publications

- This chapter is based on ***Mehmet Mutlu***, *Simon Hauser, Alexandre Bernardino and Auke Ijspeert. "Effects of passive and active joint compliance in quadrupedal locomotion."* Advanced Robotics 2018.

My original contributions

  - Coming up with the concept
  - Developing electronic parts of the robot (bringing components together and testing the hardware)
  - Developing and performing hardware experiments
  - Writing the manuscript

9.1 Introduction

Quadrupedal animals exhibit great adaptability to changing environment conditions during locomotion. Adaptation can be in the form of reflexive actions, gait change or muscle stiffness modulation. While reflexes and gait transitions drastically alters locomotion characteristics, muscle stiffness modulation usually improves performance without completely changing the locomotion type. The performance criteria can include energy efficiency, speed, precision, accuracy etc.. For instance, real animals move in an energy efficient way [Dickinson et al., 2000]. Different performance demands yield different optimal muscle stiffnesses.

Compliance is a crucial characteristic in legged animal locomotion. All parts of the musculoskeletal system of a body, muscles, tendons, tissue, skin, bones etc., exhibit different levels of compliance. The effects of compliance greatly vary on the momentary task or activity. It has the potential to add robustness to stiff/brittle structures, is able to store and release energy and can help to reduce peak forces e.g. when an impact is experienced. In order to profit from such properties, it is important to note that in most cases compliance needs to be well-tuned to obtain a desired effect.

In locomotion, compliance is thought to play a key-role in many aspects from safety and gait stabilization to energy efficiency and dynamic gaits (e.g. [Alexander, 1984]). It is unclear however, which kind of compliance acts on which aspects of locomotion and how to quantify potential benefits. For instance, according to [Geyer et al., 2006], compliant legs are essential to obtain the basic walking mechanics in bipedal human locomotion. A widely used approach is adding a passive compliance as demonstrated in [Lasa and Buehler, 2001, Iida et al., 2007, Hutter et al., 2013, Sproewitz et al., 2013]. A pragmatic extension to passive compliance is the capability of tuning stiffness using hardware approaches either during the locomotion or even during a single step cycle [Takuma et al., 2008, Hurst et al., 2010, Galloway et al., 2013] with variable stiffness actuators.

Elastic, spring-like materials are not the only way to achieve compliance. Using proximal sensors and an active control, it is possible to model virtual spring effects and integrate into motor servo control, described as proxy-based sliding mode in [Kikuuwe et al., 2010]. Another use of virtual springs during quadrupedal locomotion is explained in [Ajallooeian et al., 2013]. Impedance control through controlling torque is also widely used to adjust compliance, e.g. [Focchi et al., 2012] shows an implementation on quadrupedal locomotion. Force sensors on the feet can be incorporated to achieve actively compliant locomotion [Ugurlu et al., 2013] in a morphologically rigid robot. In a recent study [Fukuhara et al., 2016], the authors propose compliant locomotion through Tegotae-based, sensory feedback driven control where phase coupling of leg oscillators emerge from robot-environment interaction. By incorporating machine learning to footstep planning [Kalakrishnan et al., 2011, Buchli et al., 2009] it is also possible to achieve adaptation to highly rough terrain.

The robots that use the Tegotae-based control scheme have usually been reported to have series elastic elements on legs [Owaki et al., 2017]. The Tegotae scheme can clearly generate different gaits (trot, bound, gallop etc.) and the passive compliance of the leg has a modulating role. [Fukuhara et al., 2016] declares that a lower level of active-stiffness results in less Tegotae (good/useful feedback) in robot-environment interaction and steady state locomotion may be severely degraded for very low stiffness values.

Despite all previous works emphasize the importance of compliant legs, they cover only limited aspects. In particular, they lack the analysis of compliant locomotion in rough terrains. Moreover, there are limited previous studies on the combined effect of passive and active compliance on locomotion performance. We could not find any study qualitatively evaluating the relationship of different levels of passive compliance and the Tegotae based control.

The driving force this chapter is the increasing need to understand role of both passive and active compliance better in quadrupedal locomotion. To this end, a compliant modular quadruped robot has been designed using low-budget off-the-shelf components. The robot is highly customizable and fast to reconfigure which enables a wide set of experiment possibilities involving morphological changes. The goal of the chapter is to answer a set of questions which are

1. How does the passive compliance of legs affect quadrupedal robot locomotion?
2. Can having asymmetric passive compliance on fore and hind limbs increase the performance of locomotion?
3. Is it possible to boost adaptation of the robot to its environment using active compliance?
4. Does active compliance and passive compliance cooperate well or destruct each other's contributions?
5. Do results scale up to different terrains?

Contributions of the chapter are three-fold: (i) a comprehensive and systematic experimental analysis of quadrupedal locomotion on various surfaces with passive leg compliance; (ii) a quantitative analysis on combination of Tegotae-based control and passive leg compliance under changing compliance levels; (iii) the first study of Tegotae-based control of a quadruped robot on rough terrain.

The rest of the chapter is organized as follow. Details of the implementation and experiments are given in Sec. 9.2 and Sec. 9.3 respectively. Analysis and findings from the data are explained in the Sec. 9.4. Finally, the chapter is concluded in Sec. 9.5.

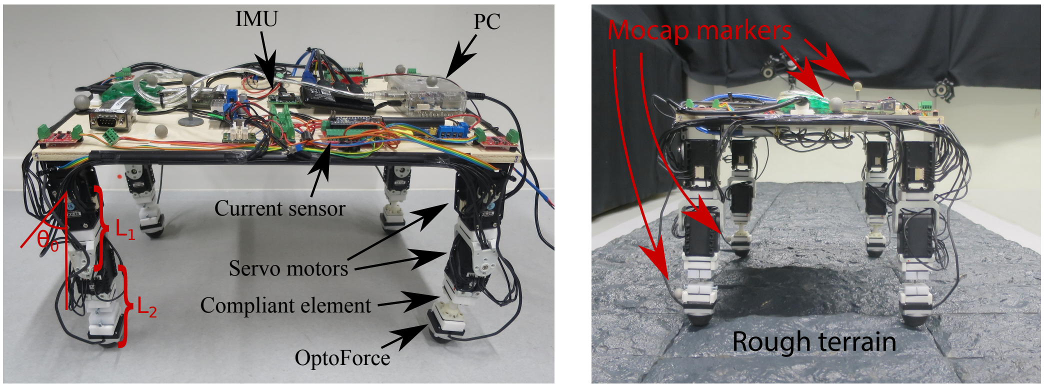


Figure 9.1 – (a) The quadrupedal robot consists of various on-board sensors and has an embedded PC to collect all the data and handle the high level control. (b) The control PC can get the data from the external tracking system through Wi-Fi. All data is collected on the same PC to ensure synchronization across various sensors.

9.2 Methods

9.2.1 Hardware platform

This study is conducted on a simple yet dynamically rich quadrupedal morphology. The robot consists of a rectangular body (39 cm x 23.5 cm). The overall structure of the robot can be seen in Fig. 9.1 a). Each limb has 2 degrees of freedom (DOF) where the upper leg (L1) is 79 mm and the lower leg (L2) is 110 mm. The motion of the limbs is constrained to the sagittal plane. Each hip joint is powered by a Dynamixel RX-28 servo motor and each knee joint is powered by a Dynamixel AX-12A. In a preliminary study [Mutlu et al., 2017], it is shown that the proximal parts of the limb should have higher compliance compared to the distal parts. Hence, the proximal hip part of the leg is directly connected to the lower (distal) leg without any compliant elements in between. However, lower limbs are extended with easily changeable “compliant elements” which can have different mechanical properties such as height, spring stiffness, weight etc.. In this study the different tested elements are designed to have the same dimensions and only the spring stiffness is varied. They are rigid elements made out of Polyoxymethylene (POM) rods and two types of compliant elements made out of super-elastic Nitinol wire with diameters of d = 1.5 mm (further called “soft”) and 2 mm (further called “hard”) with corresponding flexural stiffnesses 2.3 Nm/rad and 7.3 Nm/rad; torsional stiffnesses 1.75 Nm/rad and 5.54 Nm/rad [Vespignani et al., 2015]. The three different elements and the quick locking mechanism are shown in Fig. 9.2. The quick lock mechanism eliminates the need of extra tools to exchange compliant elements and considerably accelerates the exchange process.

The robot is equipped with an embedded PC to collect sensor data and control servo motors. The detailed list of on-board components is as follows:

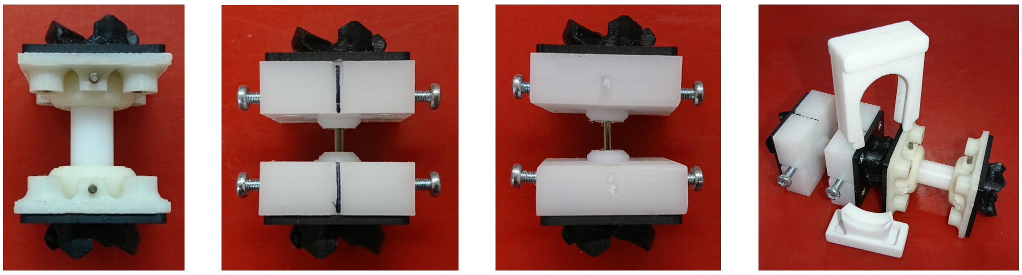


Figure 9.2 – Three different elements with the same dimensions have been used in this study: (a) rigid Polyoxymethylene (POM), (b) super elastic nitinol wire with 2 mm and (c) 1.5 mm diameter. (d) A locking mechanism has been designed which eliminates need for tools for the change of elements and enables quick change of elements. The mechanism consists of two interlocking pieces that wrap around the connection joint.

- *Dynamixel RX-28* servo motors (4x)
- Interchangeable passive elements (4x)
- *Dynamixel AX-12* servo motors (4x)
- *Optoforce OMD-30-SE-100N* 3D-force sensors (4x)
- *ODROID-XU4* embedded control pc
- *INA169* DC current sensor
- *Xsens MTi-3 AHRS* IMU
- *USB2Dynamixel* communication bus converter (2x)
- *LM2596S* (12V) DC Voltage regulator

The servo motors are used for joint angle control and have encoders that feed back their position. The current sensor measures the total current going to the motors at 1500 Hz. The current sensor has an intermediary Arduino board to send the data to the PC. The robot is powered externally through a tether from a DC voltage source. Since the source voltage is the same, the current reading directly correlates to the power demanded by motors during locomotion. Force sensors give 3D ground reaction force information on each foot which is essential for the Tegotae control scheme. The IMU has a 3-axis accelerometer, a 3-axis gyro and a 3-axis magnetometer. It is used for recording acceleration and rotational velocity of the robot's body at 100 Hz during locomotion. Due to drifts in the IMU, the global pose of the robot is tracked with an external motion capture (MoCap) system (Fig. 9.1 b)). The MoCap data is also streamed to the robot's on-board PC to ensure synchronization across different data sources. Thus, all data frames are timestamped with the same clock and they are inherently synchronized.

9.2.2 Gait generation

Quadrupedal animals can walk or run in a great variety of ways. In this study we picked the "trot" gait for our analysis since it is one of the most energy efficient quadrupedal gaits that is observed on many quadrupedal animals during long journeys [Bramble and Lieberman, 2004]. Moreover, trot is a switching gait from walking to fast galloping. Hence, it has rich dynamical characteristics, but does not demand very high actuator power. In this study, two different types of controllers will be tested: (i) open loop control (forced to trot) and (ii) Tegotae-based closed loop control.

9.2.3 Foot trajectory

In this study, the role of locomotion controller is coordinating the phase of a gait cycle. However, the desired trajectory that a foot tracks is designed as a predefined cyclic motion. The parameters of the foot trajectory, swing amplitude and swing height, are optimized using Particle Swarm Optimization (PSO) in a simulation environment to yield faster speed for a given constant locomotion frequency. The shape of the foot trajectory is set to be two different elliptic arcs during stand and swing phases with maximum 1.5 cm foot clearance ($h_{sw} - h_{st}$) as illustrated in Fig. 9.3. The mid stance phase is marked as p_2, mid swing phase is marked as p_4 and stance-swing switching moments are marked as p_1 and p_3. For the inclined surface locomotion, an offset $\theta_0 = 0.1$ rad (5.7°) is added to the hip angle to shift the center of mass slightly forward. Details of the implemented foot trajectory are further explained in [Vasconcelos et al., 2017].

9.2.4 Open loop CPG-based control

Legged locomotion can be achieved to some extend in an open loop control scheme, although the expected robustness of the gait would be low. Open loop control is chosen to form a baseline in this study. The steady-state locomotion is a cyclic motion such that each limb i is at a specific phase $\phi_i(t)$ of the cycle at the given time t.

In this particular implementation, each leg's motion is modeled as a phase-oscillator to steer the system into a desired limit cycle [Sproewitz et al., 2013], [Ijspeert, 2008], [Collins and Stewart, 1993], [Cohen et al., 1982]. Leg oscillators are coupled to each other in lateral and axial fashion. The phase of each limb i is described by the set of coupled differential equations given as

$$\dot{\phi}_i = 2\pi f + \sum_j w_{ij} \sin(\phi_j - \phi_i - \psi_{ij}) \tag{9.1}$$

where ϕ denotes the phase of an oscillator, f is the gait frequency and ψ is the desired phase difference between two oscillators. The coupling terms adjust the phase update of each

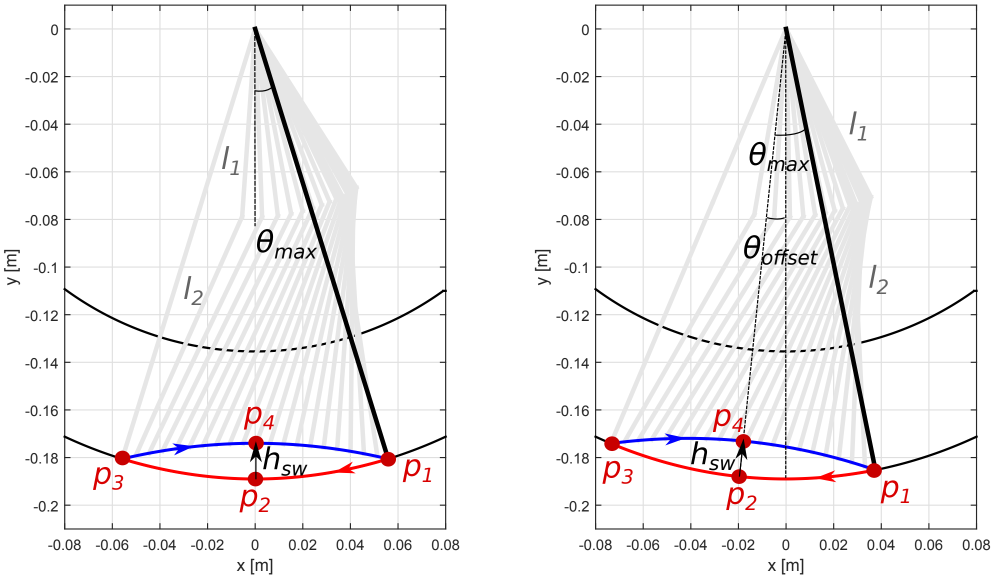


Figure 9.3 – (a) The designed foot trajectory when the hip is fixed and foot freely moves in the air. Blue elliptic arc shows the swing phase and the red one shows the stance phase. Limits of the workspace is shown with black arcs. Parameters defining the gait are hip angle maximum extension ($\theta_{max} = 0.3$ rad), height difference from swing phase mid point to fully stretched leg ($h_{sw} = 15mm$) and height difference from stance phase mid point to fully stretched leg ($h_{st} = 0$ mm). (b) As an inclination compensation, an offset angle ($\theta_{offset} = 0.1$ rad) is added to the hip angle.

oscillator, according to the phase of the neighbors ϕ_j, desired phase shift ψ_{ij} between limbs i and j, and the weight of the coupling w_{ij}. The phase update rule given in Eq. 9.1 brings the system - starting from any arbitrary initial conditions - to a desired limit cycle after an initial transient phase. At steady state, the locomotion frequency is constant. First, the phase is propagated which then is used to calculate the actual joint commands using the desired Cartesian foot trajectory and inverse kinematics.In other words, ϕ is given to hip and knee local PID controllers as the set-point after conversion to joint angles. ϕ is integrated within the controller and it is independent of the real leg phase measured by encoders. A separate feedback term that measures real leg angles could have been added to the Eq. 9.1. But, open loop control is chosen as a baseline to keep compliance experiments isolated from effects of feedback. Due to bandwidth limitations of the servo motors, open loop gait frequency is set to 0.5 Hz throughout all experiments.

9.2.5 Tegotae-based control

In the open loop control, coordination of the legs depends on hard-coded couplings between the oscillators. On the other hand, Tegotae-based control has no explicit leg coordination and oscillators are coupled indirectly through sensory feedback. All phase oscillators are independent. However, the Tegotae method is making use of local force feedback and the phase oscillators still naturally converge to a synchronized behavior due to the dynamic robot-environment interaction. In this study the implemented Tegotae scheme is similar to the one explained in [Owaki et al., 2012]. The phase equations of limb oscillators are given as

$$\dot{\phi}_i = 2\pi f + s\,N_i\,\cos(\phi_i) \tag{9.2}$$

where N_i is the z axis reading of the force sensor which corresponds to the axial normal force applied to ith leg by the ground and s is the Tegotae attraction coefficient. The equation implies that whenever there is a normal ground reaction force on a foot, a phase attraction is created towards the mid-stance phase. More visually, ϕ gets attracted towards p_2 during the stance phase shown as red arc from p_1 to p_3 in Fig. 9.3. Amount of attraction depends on the normal ground reaction force. Another expectation is that on a rough terrain, robot will not exhibit periodic body oscillations which will lead to slightly different phase changes at each gait cycle and the Tegotae-based control is expected to introduce an exploratory stepping behavior.

Hypothesis 1 (H1) *Tegotae-based control will improve rough terrain locomotion performance thanks to its exploratory nature.*

In [Fukuhara et al., 2016], there are two Tegotae rules; one modulating the phase and the other one modulating the proportional constant (K_P) of the local PID controller. Changing K_P is not always easy in real hardware. Most of the local PID controllers are not designed for seamless operation in constantly changing parameters especially due to involved setting delays. It is also the case for the hardware presented in this chapter. Hence, only the conventional Tegotae rule is considered. Nevertheless, the conventional Tegotae rule can be considered as a set-point control scheme and it has dampening and exploratory characteristics and imitates a compliant control.

9.3 Experiments

This chapter is primarily based on an experimental study. Main research questions of this chapter were posed in Sec. 9.1. Our strategy to tackle each question can be summarized as follows:

1. Add series passive compliant elements to all of the legs.
2. Use elements having different compliance on fore and hind limbs.
3. Implement Tegotae-based control and observe if it introduces any advantages over passive compliance.
4. Test closed loop active controller with different leg compliances.
5. Repeat experiments for different surface conditions.

To address all of these questions, experiments need to involve analyzing various hardware configurations using open and closed loop controllers on different surface conditions. Our approach is a comprehensive exhaustive systematic search where only one parameter is changed at a time. At each different scenario, the robot runs more than 10 steps in steady-state and the last 10 steps of each run are taken for analysis. The rest of this section elaborates the details of the followed procedures.

9.3.1 Passive compliance

The passive elements tested in this study are the same as the ones used in [Mutlu et al., 2017]. The rigid element sets a baseline for the other compliant elements. The only upgrade is the locking mechanism which does not get loose during the locomotion and has considerably less backlash compared to the one used in [Mutlu et al., 2017]. It is important to note that the robot itself has an intrinsic compliance arising from body elasticities, motor and connector backlashes and low level servo control errors. Hence, even the case with the rigid elements has a hard-to-model parasitic compliance. The other elements introduce significant compliance on the lower limb only.

The selection of compliance distribution is empirical. The main aim is to observe effects of relative compliance. There exists $3^4 = 81$ different distributions using only 3 compliance levels on 4 limbs. Left-right of the robot is always kept symmetric as in most healthy quadrupedal animals. When the left/right symmetry is considered, number of possible compliance distributions reduce to $3^2 = 9$. Most of the quadrupedal animals also have stronger and bigger hind limbs then fore limbs. Thus, the distributions where hind limbs are softer than fore limbs (3 cases) are discarded. In this study 5 different compliant element distributions are tested: (i) “all limbs rigid” (practically no bending), (ii) “all limbs hard” (low to moderate bending), (iii) “all limbs soft” (moderate to high bending), (iv) “fore limbs hard / hind limbs rigid, (v) “fore limbs soft / hind limbs rigid”. “Fore limbs soft / hind limbs hard” distribution is intentionally left out. Because, “fore limbs complaint / hind limbs rigid” cases (4^{th} and 5^{th} distributions) are expected to give similar trend with the excluded case. Moreover, initial tests revealed that robot has more troubles with soft leg compliances compared to harder/moderate distributions. Hence one of the visually not very promising distributions is not selected for further experiments to reduce the total number of runs.

9.3.2 Tegotae-based active compliance

Tegotae control is demonstrated to have the capability to generate rich set of locomotion modes without setting any coupling between leg oscillators by modulating only a single variable (s). Moreover, Tegotae can adapt the gait to asymmetrical morphology changes too [Vasconcelos et al., 2017]. Hence it may not be a perfectly matching condition to compare open loop and Tegotae control since Tegotae may not always stay in trot mode. In order to close the difference gap, we have initialized the gaits controlled by Tegotae in trot. We also set a single value of Tegotae attraction coefficient ($s = 0.3$) which is experimentally checked to converge to trot gait at steady-state when locomoting on flat surface and most of the other cases. Due to its randomness, it is not possible to guarantee the steady-state trot convergence on the rough terrain. However, rich response characteristics of the Tegotae control make it even more interesting and worthy to study as an active compliance source.

9.3.3 Experiment terrain

Since Tegotae control is fundamentally based on the robot-environment interaction, different surface conditions are included in the study.

Flat surface, no inclination:

The flat surface with no inclination constitutes the baseline for the different surface types, because gait cycles are consistently repetitive and it is easier to observe steady state behavior particularly for the Tegotae control. This surface type is the most commonly used one for the Tegotae experiments. Hence, it can be used as a bridge between other studies and our study.

Rough surface, no inclination:

A rough surface of 3 m x 1 m size is made out of house decoration tiles. The tiles are painted with spray paint to obtain the surface seen on Fig. 9.1.b because the original white color was too reflective under the motion capture system. The roughness consists of valleys and peaks having approximately 1 cm and maximum 2 cm height difference. Such roughness introduces stochastic perturbations to the robot's locomotion in the form of slippage or getting stuck. A robust locomotion is expected to perform well on the rough terrain.

Rough surface, 3° uphill inclination:

The final experiment surface is the 3° inclined version of the same rough terrain. The robot walked uphill during the tests. Uphill conditions are even more challenging since the gravity is against the locomotion direction.

In summary, we exhaustively tested all different conditions. Therefore, the experiment set consists of

$$5 \textit{ (compliant element distribution)} * 2 \textit{ (open and closed loop control)} * 3 \textit{ (surface types)} = 30 \tag{9.3}$$

runs.

9.4 Results and Discussions

This section presents the analysis of the data collected during the experiments as well as our observations and comments on the data. Quantifying the locomotion performance is done by introducing various metrics calculated by using logged data. Furthermore, qualitative gait symmetry analysis is presented to illustrate locomotion modes emerging from different leg compliance levels as well as the Tegotae-based rule. Finally, more insights are given about the Tegotae-based control on various surfaces. The answers to the questions posed in Sec. 9.1 can be found distributed throughout sections 9.4.2, 9.4.3, 9.4.4 and 9.4.5. Answers are not given in a list to keep the coherence of the text. However, visual marks such as (*A#*) are added around the relevant text to clarify our answers for the reader.

9.4.1 Performance metrics

Quantifying performance of locomotion is a challenging task as there are different perspectives to look at the same data and some of those approaches can be biased or not as important as others. In order to be as fair and rigorous as possible, many different performance metrics have been proposed. They can be grouped in two subgroups: (i) conventional metrics such as speed, cost of transport, power consumption etc., and (ii) stability of locomotion metrics to evaluate how much the body oscillated during the locomotion, i.e. how much it deviates from the horizontal plane.

Stride length (l_s):

The foot trajectory is the same for all of the experiments. However, the stride length is expected to change for different configurations because of slippage and the robot getting stuck. Having long strides without getting stuck is a desired locomotion criteria. This metric is calculated as

$$l_s = d_t / N_s \tag{9.4}$$

where d_t is the total distance taken in 10 steps and N_s is the number of steps (fixed to 10 in this study).

Experiment time for 10 steps (t_e):

By the definition of Tegotae, it has power to suppress or advance the gait phase. So, taking 10 steps always takes the same amount of time in open loop locomotion whereas the actual time needed to perform 10 steps with the Tegotae control varies around the gait period ($1/f$). Hence, we report the time Tegotae-based control needs to take 10 steps. This metric is reported for the completeness and further used to calculate average speed. It is important to note that taking 10 steps within less time does not necessarily mean faster locomotion since actual stride length can change due to foot slipping or getting stuck even though the desired foot trajectory is the same for all experiments.

Average speed (v_a):

Although the stride length l_s is correlated with the average speed v_a, there can be differences since the experiment time for the Tegotae control is not fixed, i.e.:

$$v_a = d_t / t_e \tag{9.5}$$

Average speed is one of the most reported locomotion metrics in the literature since many researchers are trying to make faster robots.

Average power consumption (P_a):

The current demand for the locomotion is among the logged measurements. Motors are powered using a fixed DC voltage source. Hence the total power consumption for the experiment is

$$P_a = \frac{V_{DC}}{t_e} \cdot \int_0^{t_e} I(t)dt \tag{9.6}$$

where V_{DC} = 18 V. Power consumption is a widely used metric in robotics because it has implications on the battery size and operation time of a robot.

Cost of transport (CoT):

The cost of transport evaluates the power efficiency of the locomotion and is calculated as

$$CoT = \frac{P_a}{m \cdot g \cdot v_a} \tag{9.7}$$

where m is the mass of the robot (m = 2.25 kg) and g is the gravitational constant. CoT is also one of the most common locomotion metrics in the literature and it gives the operation cost of the robot to move from point A to B.

Control tracking error (e):

We log the actual motor angles read from encoder and desired joint angles. The control error e is simply the mean (per actuator) of the absolute value of the difference of the setpoint and the actual motor angle. Setpoint tracking capability is a feature of the local controller, hence expected to be invariant throughout experiments since weight does not change. However, when the high level control input is very high or when motors are blocked, actuators may have higher setpoint tracking error. Lower local control errors indicate the robot is at least moving in a desired way and not getting stuck.

Average acceleration (a_a):

This metric and the following ones are related with the smoothness (oscillation amount) of the body motion during locomotion. A more oscillating locomotion may not necessarily be less stable than a more flat one. However, it is an indicator of the energy efficiency. We consider lower accelerations to be potentially better gaits. The average acceleration is simply the mean value of all acceleration vector's Euclidean norms logged during the locomotion.

Average rotational velocity (ω_a):

The IMU gives rotational velocity with respect to each axis in 3D. Total rotational velocity ω_a is calculated the same way as the acceleration and it relates to the stability of the body too.

Force fluctuations (σ_f):

The axial force fluctuation is calculated as

$$\sigma_f = \sum_{i=1}^{4} \sigma_i \tag{9.8}$$

where σ_i is the standard deviation of the norm of (3D) data collected by the OptoForce sensors during the experiment. This metric is correlated with the amplitude of foot touchdown impacts. Higher impacts will usually result in higher σ_f values. High impacts can also deteriorate sensor readings and usually they are undesired in robotics unless high impacts are needed for a specific task.

Average motion blur (μ_a):

In order to enrich our analysis, we are adding a (Self) Motion-based Motion Blur Metric (MMBM, in short μ) which estimates the average motion blur of images caused by rotational motion of the robot's body if there was a fixed camera mounted on the robot's body. High levels of motion blur causes a loss of high frequency information on images and it is usually considered as an undesired artifact for further processing of images. The detailed explanation of the metric is given in [Mutlu et al., 2014]. This metric is calculated using only rotational velocity and camera parameters. As a result, it is highly correlated with the gyroscope data, but takes into account the image formation and the camera model. The variation of μ is similar to ω_a, but, they are different. The difference of the two metrics is more pronounced when the camera roll becomes the dominant motion. A MMBM value is calculated for each gyroscope data. An illustrative example is given in Fig. 9.4 where MMBM values during 2.5 gait cycles are depicted. Higher values of the metric corresponds to higher motion blur levels if an image is captured at that time instance. Hence lower average MMBM value is considered to be better for legged locomotion. μ_a is the mean of all MMBM samples collected during the locomotion. As seen in the plot, Tegotae-based control yields lower expected motion blur levels at peak points.

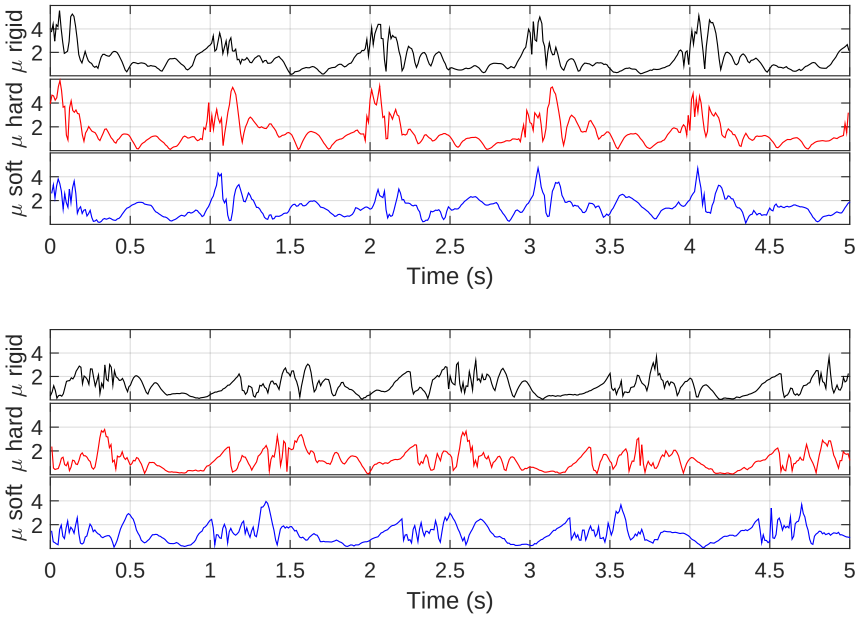


Figure 9.4 – Motion based motion blur metric (MMBM, or shortly μ) values during 2,5 gait cycles when the robot is equipped with different compliant elements and walking on flat surface with (a) open loop and (b) Tegotae control. Higher metric values correspond to more blurry images. Hence, lower values are considered to be better for the locomotion.

Chance to capture sharp images ($\mu_{\%}$):

MMBM can detect very blurry images before any image processing. This information can be used to selectively capture images when MMBM is low and discard images when MMBM is high. Even though the images can get extremely blurry at some phases of the gait and has higher MMBM average (μ_a), if there are long periods of low MMBM value, such a gait can also be useful for computer vision applications. This final metric is defined as

$$\mu_{\%} = \frac{n_s \cdot 100}{n \cdot N_s} \tag{9.9}$$

where n_s is the number of MMBM samples smaller than a predefined threshold (selected as 1 in our case) and n is the total number of MMBM samples. Camera motion blur related metrics are not very commonly used in the literature of robotic locomotion and they are quite task specific since they relate to image capturing quality on mobile cameras. However, we believe that it is another way to look at the locomotion performance and they enrich the locomotion analysis.

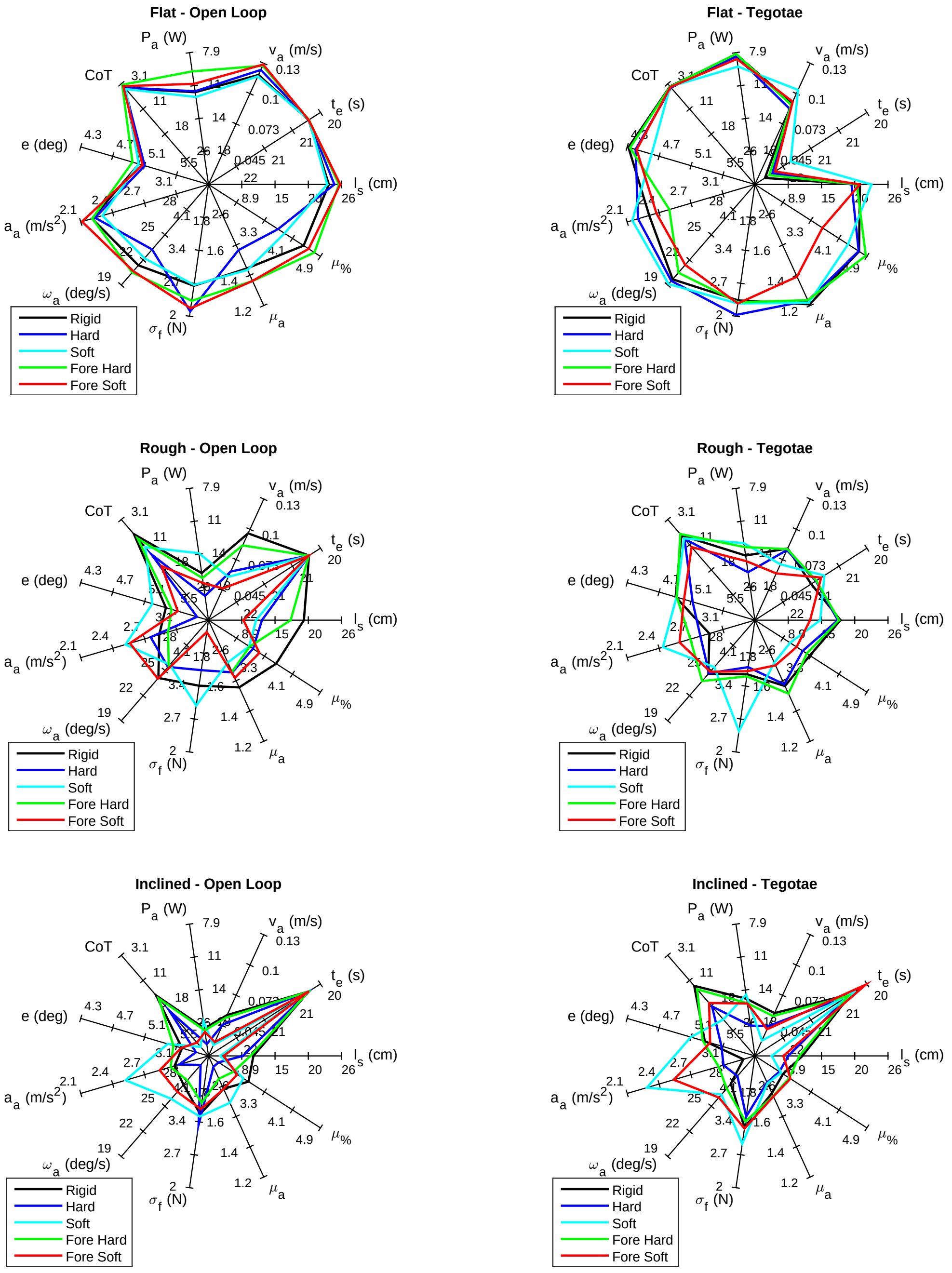
9.4.2 Overall results

The performance metrics explained in the previous subsection are calculated for all of the experimental data and the results are presented in Fig. 9.5. The axes in the spider plots have been reversed in necessary cases such that outer (from center) values indicate better locomotion performance in accordance with the defined metrics. Furthermore, all of the axes have the same range across different plots and their range is scaled and normalized such that the minimum metric value is always very close to the center and the maximum metric value is at the outer limits of the spider plot. Therefore, the area of each spider plot gives an idea about how good the performance is. However, it is important to note that under real conditions, different axes have different importances, thus the area of the spider plot is not an absolute classification criteria for any locomotion task. It is rather up to the reader's interests and intentions to choose and weight desired metrics for a performance evaluation. It is also important to note that very small metric value differences between different experiments may not be absolute indicator. The experiment time was limited to 10 gait cycles and especially on rough surface, error margins are expected to be higher due to randomness of the contact points.

The flat and rough terrain results show two major differences between open loop and Tegotae control: (i) Tegotae control takes longer t_e (experiment time for 10 steps) than the open loop control, but, (ii) the tracking error e resulting from the Tegotae control is less than the open loop e. ($A5$) Similar observations also hold for e on inclined surface locomotion. But, the difference of e between open loop and Tegotae control is less pronounced during inclined surface locomotion. An interesting observation is that t_e in Tegotae control is getting less than the open loop case on the inclined surface. The shortest t_e case occurs with the soft fore / rigid hind limb compliance on inclined surface. The reason is that the force on the z axis of the force sensor is changing direction. Fig. 9.6 shows right fore leg, the ground reaction forces, force sensor orientation as well as the sensor tip point during the mid-stance instance in rigid and soft fore/rigid hind compliance configurations on inclined rough terrain. The leg can be bent beyond the natural range when the compliance is too soft.

Another interesting observation is that the Tegotae control seems to have multiple benefits on both rough 0 deg and 3 deg inclined terrains. ($A3$ & $A4$) First of all, it has a tendency to decrease the cost of transport. Additionally, the metrics related with the stability of the robot (a_a, ω_a, e, μ_a etc.) are indicating the performance increase. Finally, the whole area that is covered by the spider plots also increase from open loop gaits to Tegotae control. That is a significant outcome indicating the Tegotae rule can boost locomotion performance on rough terrain, thus hypothesis **H1** is validated. It can also be concluded that introducing passive compliance to a robot having Tegotae-based control improves locomotion performance as long as compliance is within favorable range.

In terms of the compliance distribution, there is no single outstanding observation dominating for all runs. But, some of the distributions perform better under specific conditions. (A1) This


Flat - Open Loop

Flat - Tegotae

Rough - Open Loop

Rough - Tegotae

Inclined - Open Loop

Inclined - Tegotae

Figure 9.5 – Evaluation of the proposed metrics for all of the experiments. The data is divided into 6 spider plots illustrating (a) flat terrain / open loop control, (b) flat terrain / Tegotae control, (c) rough terrain / open loop control, (d) rough terrain / Tegotae control, (e) inclined rough terrain / open loop control, (f) inclined rough terrain / Tegotae control. Each plot is showing all of the tested compliance distributions: (i) all rigid elements (practically no bending), (ii) all hard elements (low to moderate bending), (iii) all soft elements (moderate to high bending), (iv) fore limbs hard / hind limbs rigid elements, (v) fore limbs soft / hind limbs rigid elements.

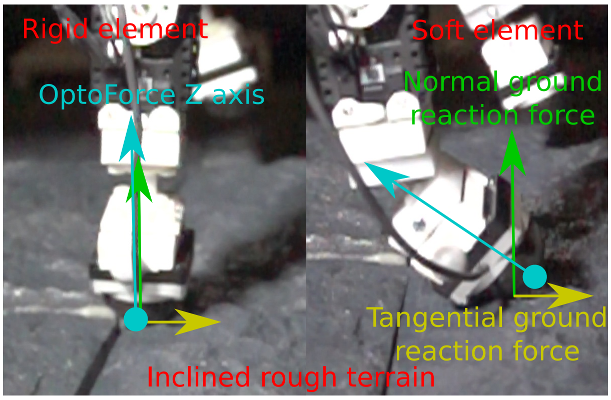


Figure 9.6 – The ground reaction force and the orientation of the force sensor mounted on the foot. When the compliance is very low the bending can reach quite high and unnatural looking levels.

is an expected result since this work tries to answer a broad range of questions. As there is no single algorithm to solve all problems, there is no perfect compliance distribution to satisfy different goals. It is still possible to infer that fore hard / hind rigid case has better performance over all hard distribution. Which points out that asymmetric fore and hind limb compliance has potential to improve locomotion performance

The experimental data is presented as a comprehensive set without hiding some aspects that may seem insignificant in the first look. However, we believe that demonstrating invariance of certain metrics is as important as finding drastic correlations. Thanks to the completeness of the experiments, a wide range of readers can find even a subset of all results useful for their own designs.

9.4.3 Feet trajectories

During the trot, two legs should be in the stance phase while the other two should be in the swing phase. However, our robot does not have active feet controlled from the ankle and only has passive spherical feet. As a result, the robot has only two points rather than 2 surfaces in real quadrupedal animals, touching the ground at most of the times. Thus, the robot dynamics exhibit unstable, inverted-pendulum-like, behavior and one of the swing phase legs sometimes prematurely touch to the ground to form dynamically stable tripod (unnatural for animals). Even though, the center of mass is approximately at the geometric center of the robot, for all of the open loop gaits the tripod is formed by two hind limbs and a fore limb. Even when one of the swing phase legs touches the ground, the majority of the weight is still carried by the stance phase legs. In order to capture such behaviors, right feet trajectories are recorded during flat terrain experiments. Fig. 9.7 gives the fore and hind limb trajectories for all uniform compliance distributions when locomoting on the flat surface. It is clear that the hind leg is always dragging. ($A2$) However, Tegotae control is trying to balance that instability and trying to bring the system to more uniform swing/stance phases. Previously, it was reported that

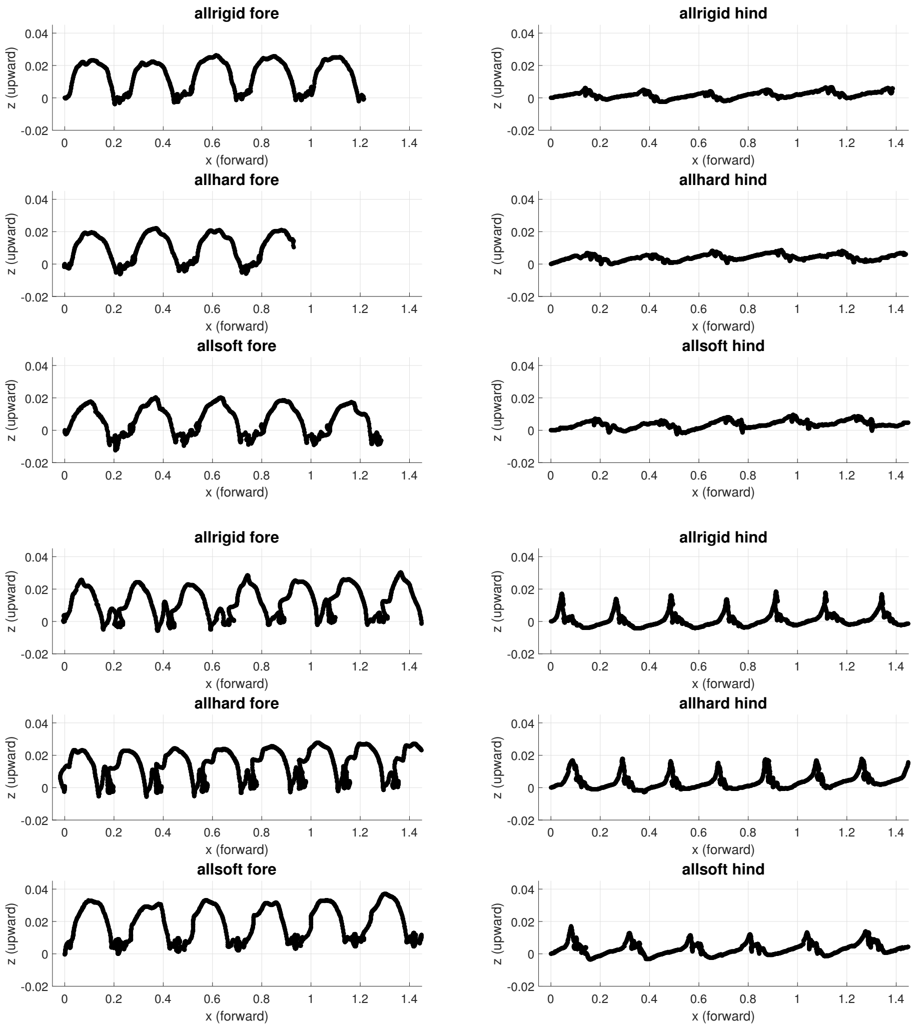


Figure 9.7 – Foot trajectories during locomotion on the flat surface, using different uniform compliance when the robot is controlled with (a) open loop or (b) Tegotae based controllers.

Tegotae rule can fight against introduced morphological asymmetries and tries to bring the locomotion to more symmetrical regime [Vasconcelos et al., 2017]. Similarly, we re-validate that Tegotae rule tries to overcome locomotion asymmetries. Furthermore, we observe cues to extend that hypothesis to include asymmetries arising from the locomotion surface.

9.4.4 Phase vs compliance in Tegotae-based control

The time evaluation of the phase gives important clues about the locomotion in Tegotae control. The phase of the limb oscillators when the robot is controlled with the Tegotae rule is given in Fig. 9.8. The figure displays three different runs with different uniform compliant elements. For each run, both right (fore and hind) limbs' phases are plotted. More stiff legs can have higher interaction forces and we observe that phase delay is increased. This observation

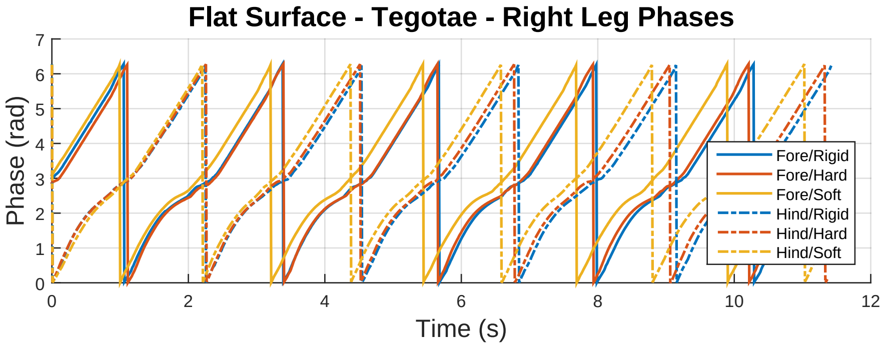


Figure 9.8 – Phase evaluation of fore and hind right limbs when the robot is locomoting on flat surface with Tegotae control. Three different uniform compliance tested. When the limbs are more stiff, the robot-environment interaction is higher and phases get longer delay.

is also consistent with the literature.

9.4.5 Emerging gaits from Tegotae-based control

Open loop runs were always forced to be trot. However, Tegotae control does not have any direct coupling between phase oscillators of limbs. Nevertheless, on flat surface Tegotae always converges to trot (for our selected $s = 0.3$) no matter which compliant element is used. However, the rough surface has peaks and valleys resulting in unexpected touchdown moments. Even on the rough terrain, the gait mostly remains a trot. But, in some cases deviations from trot have been observed. We believe those deviations towards different gait modes help the robot move more efficiently on the rough terrain. Hence we associate the decrease on the cost of transport (under aforementioned scenarios) to the gait adaptation capability of Tegotae. Indeed, open loop control can sometimes get stuck (repeating same steps blindly with no actual forward motion) on some unfavorable spots on the rough terrain whereas Tegotae control explores different locomotion modes on the same spots and has higher likelihood of passing those points. All of the recorded gait patterns are given in the appendix of this chapter.

9.4.6 Limitations of the platform:

Even though it is an animal-like quadruped structure, unlike real animals, legs have proportionally more mass than the body. Moreover, the outer layer of force sensors is covered with rubber which has high friction coefficient which leads to sticktion during the locomotion. The sticktion of force sensors at feet causes energy accumulation in compliant elements during the locomotion. At takeoff, the compliant leg starts oscillating because of the lack of damping. Such oscillations become significant distortions when the leg mass is relatively high compared to the body mass and can be noticed especially in MMBM in the hard case of Fig. 9.4.

Although Optoforce OMD-30-SE-100N gives 3 axis force information, the readings are well calibrated only when the force is applied on the tip of the sensor dome. During the locomotion the touchdown may occur on any spot of the the sensor and sensor gives less accurate readings when the touchdown point is further away from the tip of the dome. Nevertheless, there are not many commercially available solutions in the market with the similar size and weight. Even though the force measurement may have some numerical errors, their relative response to the changing force is quite acceptable.

9.5 Conclusions and future work

In this chapter the effects of leg compliance is investigated. An economical quadrupedal robot is built to test different passive and active compliance characteristics. The robot is a biologically inspired one, but, the morphological characteristics does not strictly imitate any real animal. Instead, it has a modular structure to allow quick morphological changes. Three different passive compliance levels are tested in five different compliance distribution. For each compliance distribution is tested on three different surface (flat, rough and rough/inclined). Furthermore, Tegotae based method is implemented as an active (soft) compliance source and its performance is compared to open loop gait for all proposed scenarios.

Various sensor data has been collected during each trial and later on analyzed based on defined metrics. The problem has many dimensions to consider and there is no single compliance distribution outperforming all the others. Rather there appears to be some favorable combinations for certain criteria in specific scenarios. The relation of compliance to the gait is highly nonlinear and is affected by many parameters. Readers with various different applications in mind can select a subset of findings presented in this chapter to guide their compliant robot and controller designs.

Tegotae based control is presenting a great value as an active compliance source. In this study only one level of Tegotae attraction coefficient, s, tested. s is set to a fixed value which has tendency to converge to trot in all test cases. When the limbs are physically more stiff, the effects of Tegotae are more pronounced. One of the most prominent effects of Tegotae is observed to be reducing the cost of transport when locomotion on rough terrain. The most likely reason for such performance increase is the adaptation capability of the Tegotae control. By exploring slight gait changes around trot, it overcomes certain local minima where open loop gaits can exhibit significant performance drops.

The open loop gait parameters are optimized with PSO to yield fast and efficient locomotion, so the performance difference between the open loop control and the Tegotae control is not very significant. We expect that Tegotae control has a bigger potential to improve a sub-optimal open loop gait performance. In the future studies, we will include a non-optimal open loop gait to our comparisons to observe if Tegotae would perform proportionally more significant than the open loop control. The feet are completely passive in the current robot. However, feet play a crucial role in locomotion performance too. Our future work will also include integration of active feet to the same platform. Finally, we focused only on a fixed level of Tegotae attraction coefficient, s in this chapter. Exploring a wider spectrum of s is also among our future aims.

10 Conclusions

This thesis explains what we did to bring MRs one step closer to wide daily use. We envision SRMRs to be Swiss army knives of robotics: Have a SRMR and use it for every possible task. However, the vision of *"one robot to rule them all"* has not been realized yet. The studies conducted in the scope of this thesis do not find *"the Holy Grail"* neither. Nevertheless, we have advanced the SRMR front towards that direction.

The majority of the literature on *"hardware"* of SRMRs explains *"how to create a novel/new SRMR"*. Number of persistent studies that research the same SRMR platform throughout the years (roughly in the scale of a decade or more) and explain incremental steps is just a small portion. After a few years of study, number of novel and publishable outcomes on the same hardware becomes smaller. Many promising SRMRs can in principle cope with a lot of tasks that can be expected from SRMRs. However, all of the demonstrated platforms are bound to saturate in novelty due to technological limitations. At the point of saturation, a specific platform can be abandoned and a significantly different SRMR design emerges to address some of the uncovered issues. On the other hand, we can also keep pushing further on the same platform. This thesis falls into the latter category. By the time the studies of this thesis started, major hardware and control papers on the Roombots (RB) platform had already been published. Instead of starting over with a new SRMR design, we decided to walk on the steep path of the same platform and try to reach to the end. After five years of continual research on RB, the end is nowhere within reach and there is still more that it can contribute to the literature.

This thesis addresses several research questions. Next, the answers are summarized.

- **How far can we go on the path of creating the perfect SRMR that can tackle every possible task?**

We started with improving the existing **hardware** by re-engineering existing functionalities of RB. For instance, the custom designed gear boxes are upgraded from plastic to brass in order to reduce the backlash on each DOF. Similarly, active connection mechanisms (ACMs) are redesigned to reduce static friction to avoid getting stuck. On top of upgrading existing features, we have also added new ones such as absolute encoders to enable faster and more flexible experiment initialization by removing the need for calibration after each power cycle. New lighting systems are added to RB both for illumination (high power directional spotlight) and user feedback (low power LED ring) purposes. A small scale universal gripper is designed and integrated within RB modules to enable manipulation of objects with arbitrary shapes. New proximity sensors are added to each ACM surface to validate alignment during docking process. Thus, RB can avoid significant amount of uncertainties during self reconfiguration. Furthermore, the development of a GUI has significantly simplified use of RB. Details of all hardware changes are explained in Ch. 2.

Complementary to hardware of RB, we have worked on scalable **control** too. Starting from the core, control of each DOF is re-visited. In addition to tuning local PID parameters for the new RB configuration (new gearbox, weight increase, adding absolute encoder), some new safety and security features have been added. On higher level control, a new optimal planning framework has been developed for autonomous self reconfiguration tasks in grid environment as explained in Ch. 3. The new framework is a centralized algorithm that finds optimal set of actions to self reconfigure from an initial state to a desired goal state. Completeness and optimality in SRMR come with a high computation burden that limits scalability of the system. In other words, the framework cannot work for very large number of modules in very complex tasks. However, the framework is still extremely useful as it enables the very core functionality of self reconfiguration. Because, the same method can be integrated in a higher level motion planning scheme to plan small sub-tasks in a very complex scenario by compromising optimality to gain scalability. Prior planning methods of RB had certain assumptions (simplifications) such as allocating space for certain (meta-)modules and using artificial potential field vectors. In extremely nonlinear systems such as RB, potential navigation fields are doomed to fail in complex scenarios. Modules can easily get stuck in local minima and never reach to goal. The new framework does not have any restrictions on it and it can plan for all scenarios as long as it is physically possible. Additionally, we have ant-inspired local control attempt which is highly scalable, but, designed for solving a specific task: Forming bridges and crossing large gaps. Details of the distributed-local control attempt can be found in Ch. A.

Thanks to **both hardware and control upgrades**, we can demonstrate capabilities and effectiveness of RB system. The largest demo consists of 12 modules (making 36 DOF system). Modules are randomly initialized and are all disconnected in the beginning. They can form a complex shape such as chair almost fully autonomously. Unfortunately, the self reconfiguration on some of the challenging and complex scenarios is still not 100% reliable. Minimal user interference is still needed once in a while. We already have preliminary studies to improve the docking success rate even further by adding a camera and visual servo control. However, there are no concrete results on that front by the time this thesis is submitted.

- **What kind of user interfaces are suitable for using SRMRs in daily lives?**

The ultimate goal of SRMR research is creating the system that can do every task. It can be tasks in dangerous places, space applications, industrial applications or simple households. Bringing SRMRs to daily lives means a demand for human-robot interaction. Since such robots are not used as a commercial product, there are only very few research studies on interaction methods with SRMRs. Nobody truly know what is the best way to control SRMRs yet. However, opening a PC terminal every time user wants to control the system would be tremendously inefficient and boring. In the search for better user experiences, we study three different user interfaces (UIs) in the scope of this thesis. Each three UIs focus on different applications and offer their own niches.

The first UI is addressing the *"easy shape information transfer"* need. User may need a new structure being formed which has a novel (not pre-defined) shape. Verbal description could have been possible to use, but instructions can get quite complicated and lengthy. So, we propose a novel interface using a **tangible** and malleable interaction medium in Ch. 4. In this UI, the user creates a draft sculpture (model) of the desired shape. Then the sculpture is 3D scanned into digital environment where it is processed further to autonomously create the same shape using RB. In the complete realization, this UI depends on a real time 3D scanning solution which was not available during the study. Nevertheless, the UI is demonstrated with offline (lengthy) 3D scanning method. So far, the capabilities and the concept of the UI has been presented. Non-real time interaction due to lengthy 3D scanning is the next obstacle before enabling the full potential of this UI.

The second UI study is focusing on eliminating all middle interaction tools and commanding SRMRs directly with gestures. Such direct and invisible interfaces are also called as ***"natural user interface"***. Gestures have been the oldest communication method. They existed even before formation of languages. Direct physical interaction is a very powerful method. A simple pointing gesture can be much more clear than many other interaction methods. Inspired by the nature, we propose a direct human-SRMR interaction. User can simply point a module or group of modules and send them to a location by simply pointing to a desired goal location. Such an interface relies on visual tracking system to track both user, modules or both in the environment. The UI can manifest in different embodiments. In a plausible scenario we demonstrate it with a (meta-)module carrying a directed spotlight to illuminate a desired location. User simply shows where the light is needed and RB direct the light on the desired spot. We believe that the direct interaction with SRMRs in future intelligent environments (spaces augmented with SRMRs) is a much more exciting and efficient way of interacting with such robots.

The final UI envisions a teleoperation possibility. User may not be in a certain environment, yet may need to control SRMRs remotely. In such scenario, there is a need for **virtual environment**. It can be a simple PC terminal. But, we take the study one step further add head mounted displays to enhance 3D perception in virtual environment. Additionally, we also add a hand gesture tracking device to manipulate modules in the virtual environment. As a result, we can assemble any 3D structure with this UI in Ch. 6. The aim of the UI is to create a desired shape similar to the first proposed tangible UI. Additionally, we can also command the pose of the final structure in the environment.

There is a great number of ways to control tasks with SRMRs. The three novel UIs we proposed are merely surface of a deep ocean. We believe that UI studies for SRMRs will create a big hype, if SRMRs ever reach to the consumer market.

- **How closely can we recreate animal locomotion in laboratories using (SR)MRs?**

All of Part III contributes to the answer of this question as we consider lamprey swimming, terrestrial serpentine gaits and quadrupedal locomotion. Bodies of robots used in these studies consists of blocks of modules which does not look very realistic when stationary. However, when they start moving, they highly resemble real animals. Particularly Ch. 7 deeply elaborates the goal oriented neuro-visual locomotion (swimming) model of lamprey. It is using CPGs as the main motion mechanism. However, there is a closed loop control on top of the CPG based of what lamprey sees. The visual input is fed into a neuron network consisting of multiple layers replicating the real animal. Envirobot is chosen as the hardware platform. Envirobot did not have visual capabilities. Hence, a new head is designed for Envirobot to accommodate cameras and a more powerful computation unit. In the hardware level, we integrated event-based cameras as an alternative to conventional RGB cameras. Event-based cameras replicate the real structure of eye and it reports only events rather than full images. Our cameras consider a significant intensity change of a pixel as an event. Event based cameras are much more suitable for driving biological neural control structures compared to conventional cameras. First of all they respond much faster to changes in the environment. Then, we can feed each event to the neuron network, creating a more direct interface between the input layer and the controller. Thus, such biological models can be implemented for real-time robot control applications.

- **How does the visual system correlate with bio-inspired locomotion of modular robots?**

Motion of eyes is extremely crucial part of animal vision system. Even with a single eye, it is possible to get 3D environment information through lighting, occlusions, perspective projection or motion. Moving an eye (camera) in a stationary environment can yield 3D information. That is the main reason why some birds have exaggerated head motion while examining an object on the ground, because they have a narrow overlapping field of view for stereo vision. Another interesting eye feature is saccadic motion to quickly scan many points on the field of view to help 3D reconstruction (that happens in the brain). Without the saccadic motion, physiology of eyes permit to see only a very narrow conical angle with color. Furthermore there is a direct coupling between eye motion and the rest of the body to stabilize the eye during bumpy locomotion of the body. Ch. 8 explores the relation between locomotion and vision. More specifically, snake robot is chosen as the application hardware.

In the literature, we encounter snake robots equipped with camera(s). However, in all of cases, camera is placed on the head and pointing roughly forward. Although this common camera placement has certain advantages, it is not the only option. We compare different camera location possibilities and compare a few numerical metrics to quantify the quality of vision during different serpentine gaits. Even though there is no single the best solution, we report that different camera locations can be more promising for different applications depending on the gait.

- **What is the role of compliance in bio-inspired locomotion of modular robots?**

Animals consist of many different tissues spanning a wide range of compliance such as bones and fat. Moreover, there are active components such as muscles that can change compliance during the locomotion. In conventional SRMRs there is only inherent structural compliance. Compliance definitely has an important role in locomotion to dampen high impacts and increase energy efficiency by storing and releasing the kinetic energy at favorable periods during the locomotion. Ch. 9 is a comparative study understand insights of compliance in MRs during locomotion. It considers two different compliance causes: passive (structural) and active (control-based). The passive compliance is obtained by adding extra elements near joints. The active compliance comes through Tegotae-based control. Tegotae-rule is actually used gait emergence studies. When each leg's phase is controlled with decoupled oscillator, Tegotae rule modified the rate of phase depending on ground reaction forces applied on the foot. Thanks to physical interaction of oscillators through robot's body, different gaits emerge in multi legged systems. The way the Tegotae rule is formulated can be considered as an active spring and damper. Divers combinations of passive and active compliance has been tested on a quadrupedal MR while locomoting on different terrains. Each run is quantified with many metrics to evaluate locomotion performance. Results do not yield a linear trend as the compliance in legged locomotion is a highly nonlinear phenomenon, but, confirm that compliance should be well tuned for better locomotion performance. Another significant finding is that active compliance base on Tegotae rule can considerably improve the locomotion performance on rough terrain.

10.1 Future work

Although this thesis did improve the SRMRs on multiple sides, we are still quite far from bringing SRMRs into daily lives. Focusing on niche applications such as in space domain is still the more plausible use for SRMRs. Because existing solutions are quite expensive for the limited functionality they provide.

In the RB side, the hardware of the platform is almost saturated. It is getting exponentially difficult to add new features to extend current capabilities due to space constraints. On the other hand, there is also definitely room for improvement. For instance, RB are still using Bluetooth to connect to a centralized PC to get user commands. Bluetooth is limited to 7 (or 8 depending on the version) modules per Bluetooth dongle. Even though Linux accepts more than one dongles at a time, some OS (e.g. Windows) don't allow more than one dongle at a time. Hence, RB is not super scalable at the moment. There is an increasing need for large scale wireless communication solution such as Zigbee or variants. To our best knowledge very high number of modules (e.g. 1000) has never been tested in a very compact volume. Another improvement can be on proper integration of computer vision systems in SRMRs. Especially powerful computation units are getting smaller by the day. It is only a matter of time until we see better computer vision integrations in SRMRs.

However, there is still a lot of room for research in the control, planning and UI domains. There is a great potential space for expansion particularly for UIs. One of the directions in RB project can be obtaining a real-time 3D scanner to enable the full potential of the proposed tangible UI.

A Cooperative bridge building by SRMR based on ants' stigmergic behaviour

Ch. 3 gave an optimal centralized planning framework for SRMRs. However, its power is limited with number of modules and complexity of the planning since the search space is exponentially increasing with that. Another method of controlling SRMR is distributed control using local rules. Each module follows simple rules and decides on the next action by observing the close proximity of itself. The advantage of this method is scalability. Such models can in principle be scaled to much higher number of modules by compromising optimality and sometimes not finding a solution depending on the task. Each local controller must have task specific set of rules. This chapter explains our ant inspired approach to the particular problem of forming bridges with SRMRs and crossing gaps a single module cannot pass otherwise.

Reference publications

- This chapter is based on *Jasmine Nguyen-Duc, **Mehmet Mutlu**, Simon Hauser, Alexandre Bernardino and Auke Ijspeert. "Cooperative bridge building by self-reconfigurable modular robots based on ants' stigmergic behaviour."* Submitted to Adaptive Motion of Animals and Machines 2019.

  My original contributions

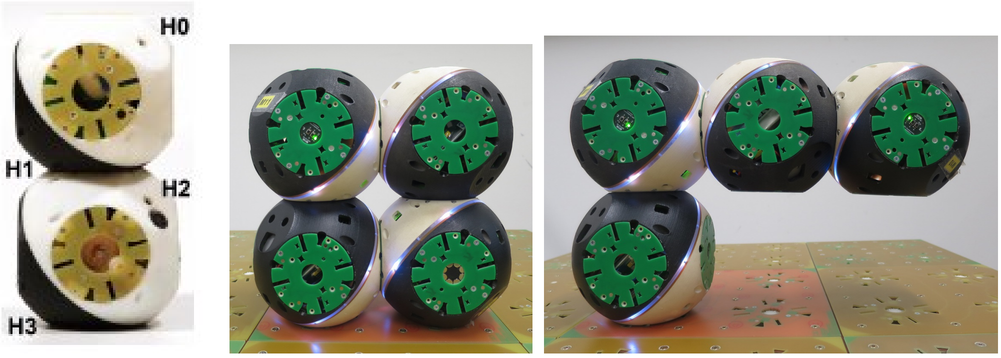
  - Defining and supervising the semester project of Jasmine Nguyen-Duc
  - Providing a base code for the framework
  - Guiding and planning of simulations
  - Partially writing and fully reviewing the manuscript

A.1 Introduction

Insect societies have particular ways of self-organising. To improve their efficiency in task managing, ants are capable of forming a bridge-like self assemblage. "Army ant bridges are remarkably strong and adaptive; the insects begin to build them as soon as they sense a gap in their path and disassemble them once traffic has cleared" [Reid et al., 2015].

This research presents a method for robots to accomplish the construction of a bridge similarly to ants. For this, the concept of stigmergy must be studied. The use of stigmergy in the robotics field has lead to the creation of an artificial swarm intelligence. The particularity of stigmergy is mainly the use of indirect communication and the absence of a central leader. So the robots take initiatives without needing an external command or any direct communication with their "teammates". The decision making of each agent is based on a set of rules and on the time varying environment.

Roombots are self-reconfigurable modular robots (SRMRs) designed in Biorobotics Laboratory of EPFL. Each module has two adjacent cube-like shapes and has three rotary degrees of freedom (DOF) as seen in Fig. A.1a. Roombots are capable of self reconfiguration and locomotion by gripping onto specific surfaces. Active Connection Mechanism (ACM) and proximity sensors are positioned at the extremities of each module.



(a) (b) (c)

Figure A.1 – (a) Single RB module [Bonardi et al., 2012b] and (b) an agent composed of two modules in standing position and (c) during a step forward.

The most common way of controlling SRMRs is centralized where all SRMRs are connected to a central computation unit, which decides how each module should behave. However, there exists distributed control ways too [Ahmadzadeh and Masehian, 2015], [Cucu et al., 2015]. Centralized methods usually suffer from high dimentionality of SRMRs and distributed methods may fail to find a solution for a desired task. Creating an algorithm inspired by a biological method used by ants would enable Roombots to achieve a certain primitive task, namely, crossing a wide gap that requires collaboration of multiple modules.

A.2 Methodology

A.2.1 An algorithm from ants

In previous studies, ants have been noticed to follow a set of simple rules while forming bridges: (1) if a gap is ahead, stop moving or slow down; (2) if an ant is immobile in front, climb on top of it and continue walking; (3) as long as an ant is on top of another, freeze. These are morphology-free commands that can be converted to an algorithm.

In our Roombots adaptation, each agent randomly explores the environment. The exploration is modelled as: ant continues walking in the same direction with 80% chance, changes direction with 15% chance and stops (without detecting gap) with 5% chance. The formation of a bridge can take a very long time since the agents just walk around randomly in the environment. However, agents can be attracted towards a gap, for example, driven by smell or light. We considered that there exists a slight attractor towards the gap. Thus, ants are more likely to walk towards the gap and start forming bridges during the exploration.

When a gap is detected while exploring, an agent leans forward and freezes for a fixed amount of time. If no other robot climbs over it in during the frozen period, the leaning robot stands up again and resumes exploration. If on the contrary, another robot does climb onto it, it stays frozen independent of time. Thus, the foraging robot has enough time to walk across the leaning robot's body. Once the other side of the gap is reached by a climbing robot, the bridge is formed and allows others to forage across it. As long as there are foraging agents on top of the "bridge-builders", they cannot stand up. Once there are no more foraging ants on the bridge, the first agent that formed the bridge stands up first and crosses the bridge while disassembling it.

A.2.2 Roombot's mechanics

Each Roombots module has only 3 DOF that is quite low compared to capabilities of a real ant. In order to have more capable robotic units, metamodules of two modules (attached in series) have been considered as the smallest robotic agent (ant) as shown in Fig. 3.8j and Fig. A.1c. Together, the two modules form a biped that is capable of walking in all four directions in the grid environment. Locomotion sequence is defined as follows: (1) Front module grips the ground and (2) back module releases connection. Then, (3) back module is first lifted up and (4) brought to the front side by a rotation of the gripping (former-front) module.

To detect a gap or an obstacle, the agent uses an IR sensor located at the tip of each "leg". Before dropping the leg onto the ground, the IR sensor is orientated downwards, checks for obstacles and gaps. In the absence of obstacles or gaps, it drops the leg to the ground and resumes exploration.

The presented method is essentially a distributed control framework. However, detection of collisions is a challenging task to be implemented in hardware. Even though, Roombots hardware allows rough torque measurements to detect unexpected collisions, we prefer not to execute actions that would result in collisions. Collisions could be avoided with distal sensors which are not yet integrated. Therefore, we tackle the collision problem in a centralised manner. As a compromise, only one agent moves at a time.

A.2.3 Logic-level simulation

MATLAB is used to simulate the proposed method. Robots and the environment is abstracted into 3D voxel grid. Environment voxels are fixed and agent voxels are updated at each step according to the decision agent takes. Although, distributed agents can all move at the same time, in this work, only one agent moves one step at a time to simplify collision problems. Each step in the MATLAB simulation is a predefined set of actions (motion primitives) in real robots.

A.2.4 Experimental set-up

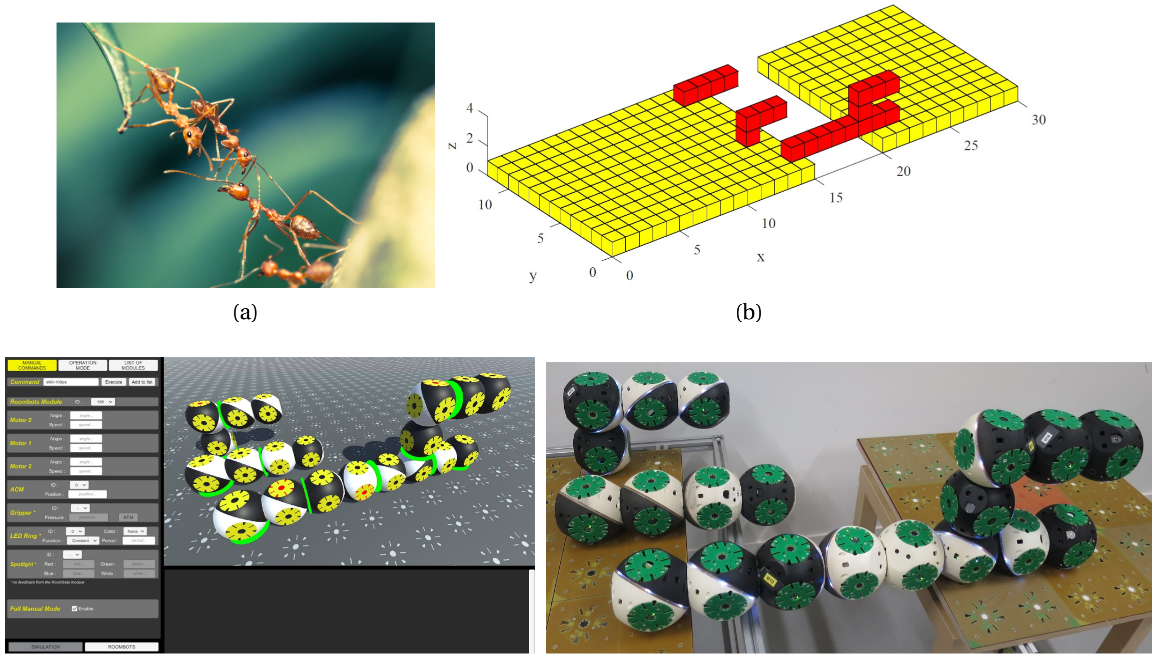
The simulation environment has a flat floor with a (four-unit wide) uniform gap that a single agent (ant) cannot cross. All agents are initially placed to one side of the gap and the experiments are concluded when all agents cross the gap or a single one is left behind. Different number of agents was tested varying from two to seven. Although, we envision a fully autonomous framework including the real hardware as shown in Fig. A.2, for this work the the main focus has been the logic level simulation as illustrated in Fig. A.2b.

A.3 Results

The most obvious difference between this simulation and the biological system is related to the conformation of the bridge. The bridges formed in the simulation are much simpler in shape (only linear) and shorter. There are three major reasons for that. (1) Ants are very strong and can carry up to 50 times of their body weight. For Roombots, that number is slightly more than one. (2) Ants can hold on to each other in many different ways, whereas Roombots need to be well aligned and have much less connection possibilities. (3) Only a limited set of motion capabilities of Roombots have been considered for simplicity. Each action of an agent is a predefined set of motion primitives. In other words, the command “Move one step forward" translates into a set of consecutive actions such as “DOF 2 of Module 1 goes to 120 degrees", “ACM 2 of Module 1 disconnect"... It is possible to enrich considered motion primitives to enable formation of more realistic bridges.

A.4 Conclusions

We have discussed cooperative behaviour for Roombots that cross a gap of a certain distance by self-assembling into a bridge. It is inspired by stigmergic foraging in ant colonies. In the



(a) (b)

(c) (d)

Figure A.2 – Whole envisioned pipeline of the work consists of (a) Inspiration from real ants (credit iStock/lirtlon) (b) modeling the behaviour for distributed SRMR control using predefined motion primitives, (c) obtaining real action sequence and (d) execution on the real robotic system.

simulation, the agents modify the environment by leaning over a gap when it is detected, thus decreasing its length. Other agents were shown successful in climbing over the ones leaning to extend the bridge until the other side is reached. The bridge becomes a shortcut which allows ants to reduce the distance to reach their target. This work will be extended with richer set of motion primitives to enable formation of larger and stronger bridges.

B Leg touchdown patterns of the modular quadruped

The leg touchdown in Ch. 9 is detected by thresholding taking the norm of the force vector obtained from OptoForce sensors. The blue regions in Figs. B.1, B.2, B.3, B.4, B.5 indicate the foot touchdown and white regions indicate the swing phase. Each plot has a three letter title describing the experiment:

- **1st letter.** R: all rigid elements, H: all hard elements, S: all soft elements, A: fore hard, hind rigid elements, F: fore soft, hind rigid elements,
- **2nd letter.** F: flat surface, R: rough surface, I: rough inclined surface
- **3rd letter.** O: open loop control, T: Tegotae based control

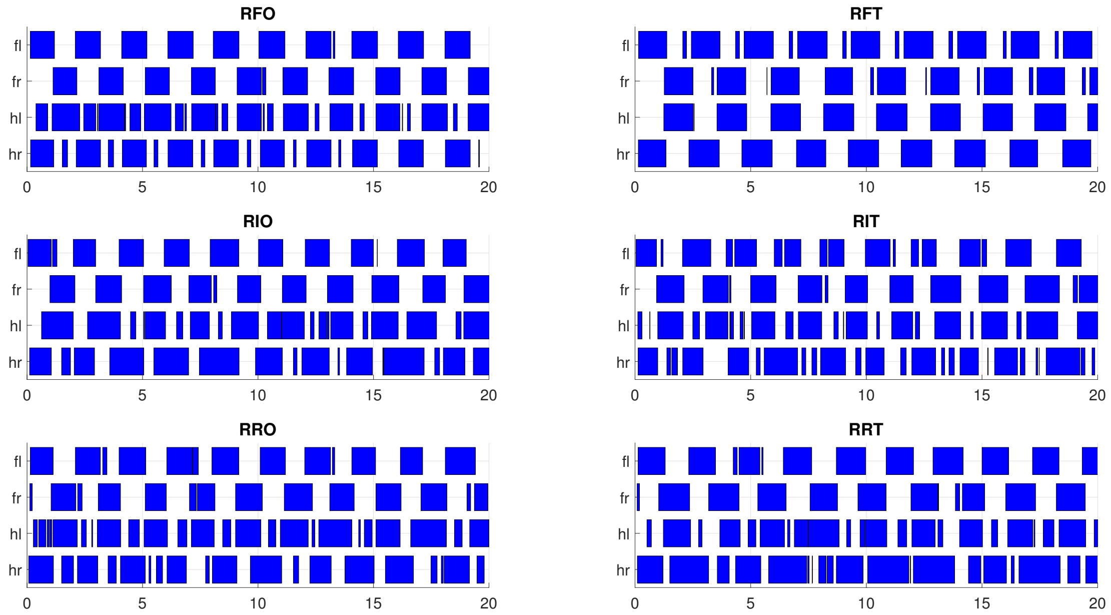


Figure B.1 – Leg touchdown patterns when all element are rigid.

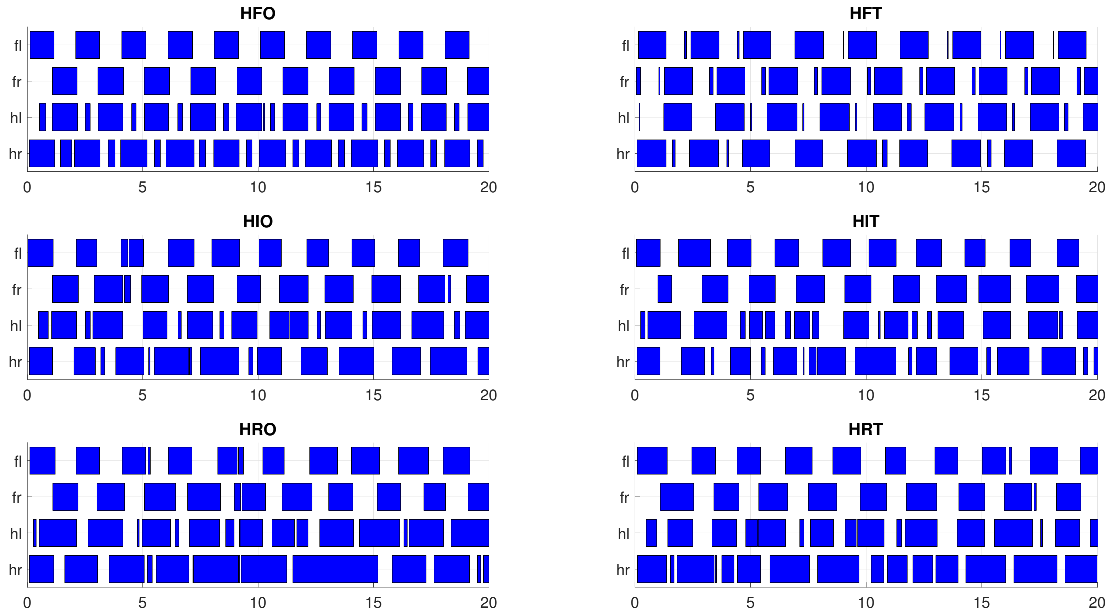


Figure B.2 – Leg touchdown patterns when all element are hard.

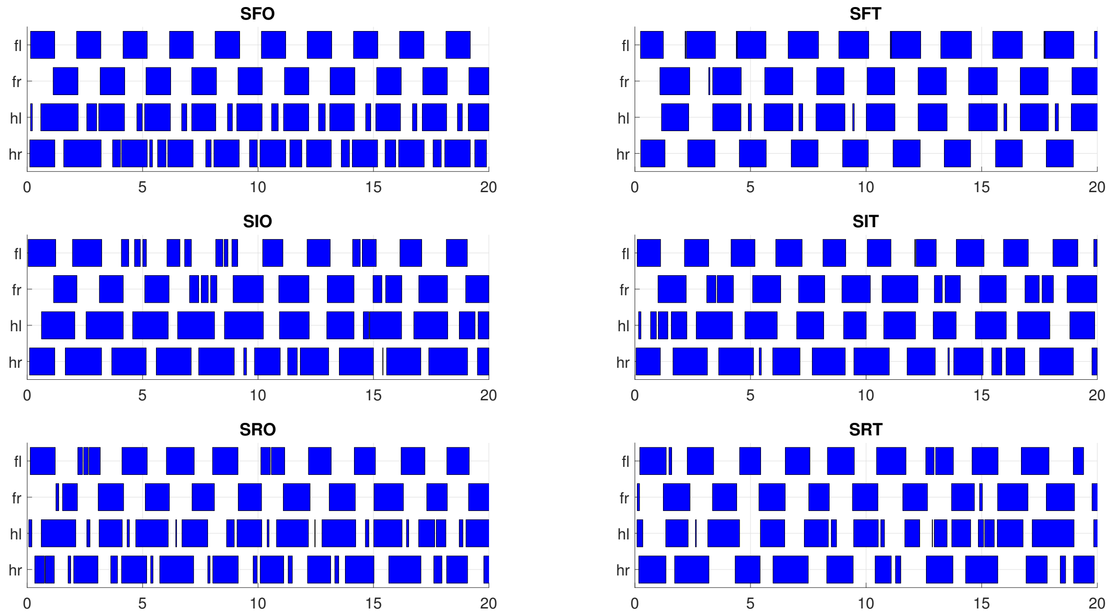


Figure B.3 – Leg touchdown patterns when all element are soft.

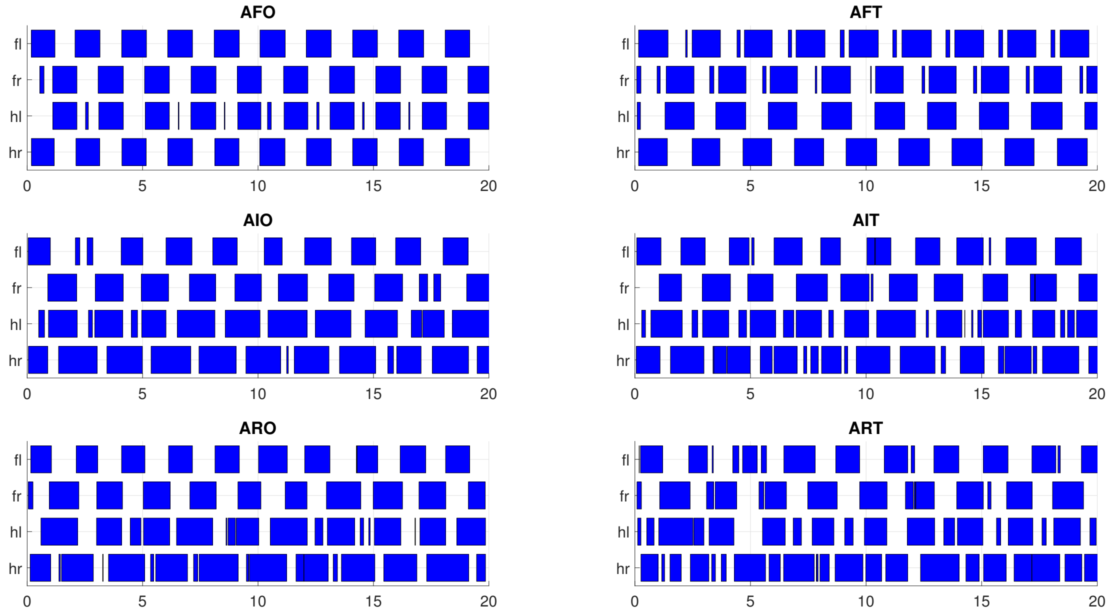


Figure B.4 – Leg touchdown patterns when fore limb elements are hard and hind limb elements are rigid.

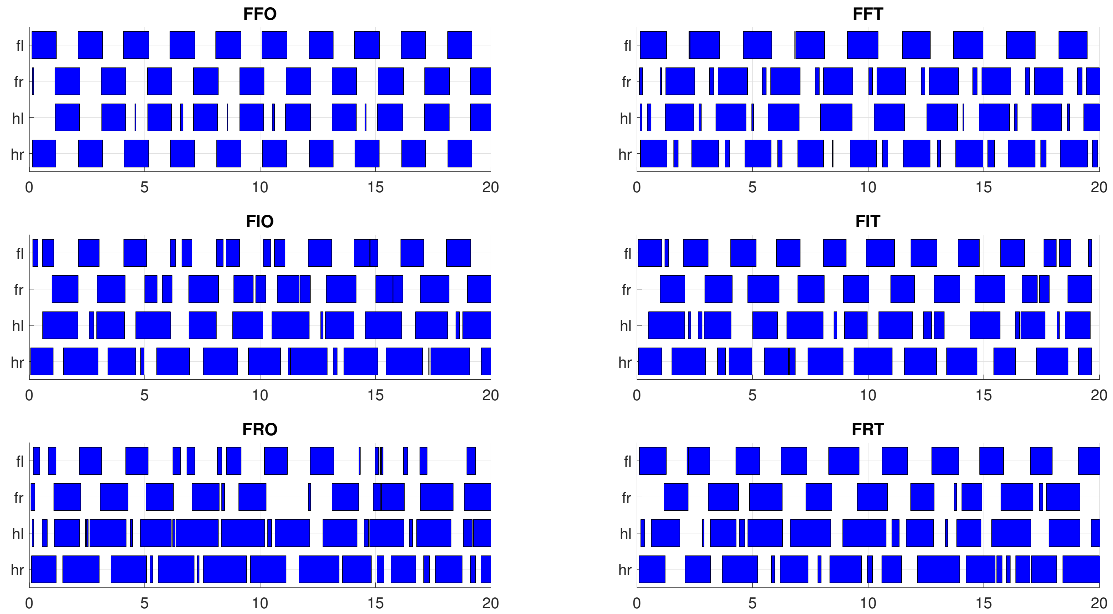


Figure B.5 – Leg touchdown patterns when fore limb elements are soft and hind limb elements are rigid.

Bibliography

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

[Ahmadzadeh et al., 2015] Ahmadzadeh, H., Masehian, E., and Asadpour, M. (2015). Modular Robotic Systems: Characteristics and Applications. *Journal of Intelligent & Robotic Systems*, pages 1–41.

[Ajallooeian et al., 2013] Ajallooeian, M., Pouya, S., Sproewitz, A., and Ijspeert, A. J. (2013). Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion. In *2013 IEEE International Conference on Robotics and Automation*, pages 3321–3328.

[Alexander, 1984] Alexander, R. M. (1984). Elastic Energy Stores in Running Vertebrates. *American Zoologist*, 24(1):85–94.

[Amend et al., 2012] Amend, J., Brown, E., Rodenberg, N., Jaeger, H., and Lipson, H. (2012). A Positive Pressure Universal Gripper Based on the Jamming of Granular Material. *IEEE Transactions on Robotics*, 28(2):341–350.

[Amend et al., 2016] Amend, J., Cheng, N., Fakhouri, S., and Culley, B. (2016). Soft Robotics Commercialization: Jamming Grippers from Research to Product. *Soft Robotics*, 3(4):213–222.

[Anderson et al., 2000] Anderson, D., Frankel, J. L., Marks, J., Agarwala, A., Beardsley, P., Hodgins, J., Leigh, D., Ryall, K., Sullivan, E., and Yedidia, J. S. (2000). Tangible interaction + graphical interpretation: a new approach to 3d modeling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 393–402. ACM Press/Addison-Wesley Publishing Co.

[Arancha Casal, 1999] Arancha Casal, M. H. Y. (1999). Self-reconfiguration planning for a class of modular robots. volume 3839, pages 3839 – 3839 – 12.

[Asadpour et al., 2009] Asadpour, M., Ashtiani, M. H. Z., Sproewitz, A., and Ijspeert, A. (2009). Graph signature for self-reconfiguration planning of modules with symmetry. In *2009*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5295–5300.

[Asfour et al., 2013] Asfour, T., Vahrenkamp, N., Schiebener, D., Do, M., Przybylski, M., Welke, K., Schill, J., and Dillmann, R. (2013). ARMAR-III: Advances in Humanoid Grasping and Manipulation. *Journal of the Robotics Society of Japan*, 31(4):341–346.

[Barrios et al., 2016] Barrios, L., Collins, T., Kovac, R., and Shen, W. M. (2016). Autonomous 6d-docking and manipulation with non-stationary-base using self-reconfigurable modular robots. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2913–2919.

[Bayat et al., 2016] Bayat, B., Crespi, A., and Ijspeert, A. (2016). Envirobot: A bio-inspired environmental monitoring platform. In *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 381–386.

[Beattie et al., 2015] Beattie, N., Horan, B., and McKenzie, S. (2015). Taking the LEAP with the Oculus HMD and CAD - Plucking at thin Air? *Procedia Technology*, 20:149–154.

[Beetz et al., 2012] Beetz, M., Jain, D., Mosenlechner, L., Tenorth, M., Kunze, L., Blodow, N., and Pangercic, D. (2012). Cognition-Enabled Autonomous Robot Control for the Realization of Home Chore Task Intelligence. *Proceedings of the IEEE*, 100(8):2454–2471.

[Bellmore et al., 2011] Bellmore, C., Ptucha, R., and Savakis, A. (2011). Interactive display using depth and RGB sensors for face and gesture control. In *2011 Western New York Image Processing Workshop*, pages 1–4.

[Bianco and Engert, 2015] Bianco, I. H. and Engert, F. (2015). Visuomotor transformations underlying hunting behavior in zebrafish. *Current Biology*, 25(7):831–846.

[Bonardi et al., 2012a] Bonardi, S., Blatter, J., Fink, J., Moeckel, R., Jermann, P., Dillenbourg, P., and Ijspeert, A. (2012a). Design and evaluation of a graphical iPad application for arranging adaptive furniture. In *2012 IEEE RO-MAN*, pages 290–297.

[Bonardi et al., 2012b] Bonardi, S., Moeckel, R., Sproewitz, A., Vespignani, M., and Ijspeert, A. J. (2012b). Locomotion through Reconfiguration based on Motor Primitives for Roombots Self-Reconfigurable Modular Robots. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6. VDE.

[Bonardi et al., 2013] Bonardi, S., Vespignani, M., Moeckel, R., and Ijspeert, A. J. (2013). Collaborative manipulation and transport of passive pieces using the self-reconfigurable modular robots roombots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2406–2412. Ieee.

[Bonardi et al., 2014] Bonardi, S., Vespignani, M., Moeckel, R., Van den Kieboom, J., Pouya, S., Sproewitz, A., and Ijspeert, A. (2014). Automatic generation of reduced CPG control networks for locomotion of arbitrary modular robot structures. In *Proceedings of Robotics: Science and Systems.*

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[Bramble and Lieberman, 2004] Bramble, D. M. and Lieberman, D. E. (2004). Endurance running and the evolution of homo. *Nature*, 432(7015):345–352.

[Brandt, 2006] Brandt, D. (2006). Comparison of a and rrt-connect motion planning techniques for self-reconfiguration planning. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 892–897.

[Brandt et al., 2007] Brandt, D., Christensen, D. J., and Lund, H. H. (2007). ATRON Robots: Versatility from Self-Reconfigurable Modules. In *2007 International Conference on Mechatronics and Automation*, pages 26–32.

[Broekens et al., 2009] Broekens, J., Heerink, M., and Rosendal, H. (2009). Assistive social robots in elderly care: a review. *Gerontechnology*, 8(2).

[Brown et al., 2010] Brown, E., Rodenberg, N., Amend, J., Mozeika, A., Steltz, E., Zakin, M. R., Lipson, H., and Jaeger, H. M. (2010). Universal robotic gripper based on the jamming of granular material. *Proceedings of the National Academy of Sciences*, 107(44):18809–18814.

[Brown et al., 2011] Brown, G. E., Ferrari, M. C. O., and Chivers, D. P. (2011). *Learning about Danger: Chemical Alarm Cues and Threat-Sensitive Assessment of Predation Risk by Fishes*, chapter 4, pages 59–80. John Wiley & Sons, Ltd.

[Buchli et al., 2009] Buchli, J., Kalakrishnan, M., Mistry, M., Pastor, P., and Schaal, S. (2009). Compliant quadruped locomotion over rough terrain. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 814–820.

[Camplani et al., 2017] Camplani, M., Paiement, A., Mirmehdi, M., Damen, D., Hannuna, S., Burghardt, T., and Tao, L. (2017). Multiple human tracking in rgb-depth data: a survey. *IET Computer Vision*, 11:265–285(20).

[Carlson and Demiris, 2012] Carlson, T. and Demiris, Y. (2012). Collaborative Control for a Robotic Wheelchair: Evaluation of Performance, Attention, and Workload. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):876–888.

[Censi et al., 2013] Censi, A., Strubel, J., Brandli, C., Delbruck, T., and Scaramuzza, D. (2013). Low-latency localization by active led markers tracking using a dynamic vision sensor. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 891–898.

[Cho and Lee, 2009] Cho, S. and Lee, S. (2009). Fast motion deblurring. *ACM Trans. Graph.*, 28(5):145:1–145:8.

[Cohen et al., 1982] Cohen, A. H., Holmes, P. J., and Rand, R. H. (1982). The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: A mathematical model. *Journal of Mathematical Biology*, 13(3):345–369.

[Collins and Stewart, 1993] Collins, J. J. and Stewart, I. N. (1993). Coupled nonlinear oscillators and the symmetries of animal gaits. *Journal of Nonlinear Science*, 3(1):349–392.

[Cucu et al., 2015] Cucu, L., Rubenstein, M., and Nagpal, R. (2015). Towards self-assembled structures with mobile climbing robots. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1955–1961.

[Cui et al., 2012] Cui, X., Zhao, J., Zhu, Y., and Tang, S. (2012). UBot: a new reconfigurable modular robotic system with multimode locomotion ability. *Industrial Robot: the international journal of robotics research and application*, 39(2):178–190.

[Cully et al., 2015] Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521(7553):503–507.

[Daudelin et al., 2018] Daudelin, J., Jing, G., Tosun, T., Yim, M., Kress-Gazit, H., and Campbell, M. (2018). An integrated system for perception-driven autonomy with modular robots. *Science Robotics*, 3(23).

[Davey et al., 2012] Davey, J., Kwok, N., and Yim, M. (2012). Emulating self-reconfigurable robots - design of the SMORES system. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4464–4469.

[Demiris, 2009] Demiris, Y. (2009). Knowing when to assist: Developmental issues in lifelong assistive robotics. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3357–3360.

[Dickinson et al., 2000] Dickinson, M. H., Farley, C. T., Full, R. J., Koehl, M. a. R., Kram, R., and Lehman, S. (2000). How Animals Move: An Integrative View. *Science*, 288(5463):100–106.

[Dunston et al., 2011] Dunston, P. S., Arns, L. L., Mcglothlin, J. D., Lasker, G. C., and Kushner, A. G. (2011). An Immersive Virtual Reality Mock-Up for Design Review of Hospital Patient Rooms. In *Collaborative Design in Virtual Environments*, Intelligent Systems, Control and Automation: Science and Engineering, pages 167–176. Springer, Dordrecht.

[Edmonds and Karp, 1972] Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264.

[Fitch and Butler, 2008] Fitch, R. and Butler, Z. (2008). Million module march: Scalable locomotion for large self-reconfiguring robots. *The International Journal of Robotics Research*, 27(3-4):331–343.

[Fitch and McAllister, 2013] Fitch, R. and McAllister, R. (2013). *Hierarchical Planning for Self-reconfiguring Robots Using Module Kinematics*, pages 477–490. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Floreano et al., 2014] Floreano, D., Ijspeert, A., and Schaal, S. (2014). Robotics and neuroscience. *Current Biology*, 24(18):R910 – R920.

[Florez et al., 2013] Florez, J., Calderon, F., and Parra, C. (2013). Video stabilization taken with a snake robot. In *Image, Signal Processing, and Artificial Vision (STSIVA), 2013 XVIII Symposium of*, pages 1–5.

[Focchi et al., 2012] Focchi, M., Boaventura, T., Semini, C., Frigerio, M., Buchli, J., and Caldwell, D. G. (2012). Torque-control based compliant actuation of a quadruped robot. In *2012 12th IEEE International Workshop on Advanced Motion Control (AMC)*, pages 1–6.

[Follmer and Ishii, 2012] Follmer, S. and Ishii, H. (2012). KidCAD: Digitally Remixing Toys Through Tangible Tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2401–2410, New York, NY, USA. ACM.

[Forlizzi and DiSalvo, 2006] Forlizzi, J. and DiSalvo, C. (2006). Service robots in the domestic environment: A study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, HRI '06, pages 258–265, New York, NY, USA. ACM.

[Fukuhara et al., 2016] Fukuhara, A., Owaki, D., Kano, T., and Ishiguro, A. (2016). Leg Stiffness Control Based on "TEGOTAE" for Quadruped Locomotion. In *Biomimetic and Biohybrid Systems*, Lecture Notes in Computer Science, pages 79–84. Springer, Cham.

[Galloway et al., 2013] Galloway, K. C., Clark, J. E., and Koditschek, D. E. (2013). Variable Stiffness Legs for Robust, Efficient, and Stable Dynamic Running. *Journal of Mechanisms and Robotics*, 5(1):011009–011009–11.

[Gao et al., 2015] Gao, W., Zhang, Y., Nazzetta, D. C., Ramani, K., and Cipra, R. J. (2015). RevoMaker: Enabling Multi-directional and Functionally-embedded 3d Printing Using a Rotational Cuboidal Platform. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 437–446, New York, NY, USA. ACM.

[Gao et al., 2019] Gao, Z., Shi, Q., Fukuda, T., Li, C., and Huang, Q. (2019). An overview of biomimetic robots with animal behaviors. *Neurocomputing*, 332:339 – 350.

[Georgoulas et al., 2014] Georgoulas, C., Linner, T., and Bock, T. (2014). Towards a vision controlled robotic home environment. *Automation in Construction*, 39:106 – 116.

[Geyer et al., 2006] Geyer, H., Seyfarth, A., and Blickhan, R. (2006). Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the Royal Society of London B: Biological Sciences*, 273(1603):2861–2867.

[Gilpin and Rus, 2010] Gilpin, K. and Rus, D. (2010). Modular Robot Systems. *IEEE Robotics Automation Magazine*, 17(3):38–55.

[Griffiths and Higham, 2010] Griffiths, D. F. and Higham, D. J. (2010). *Euler's Method*, pages 19–31. Springer London, London.

[Gross and Green, 2012] Gross, M. D. and Green, K. E. (2012). Architectural robotics, inevitably. *interactions*, 19(1):28–33.

[Gross et al., 2006] Gross, R., Tuci, E., Dorigo, M., Bonani, M., and Mondada, F. (2006). Object transport by modular robots that self-assemble. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2558–2564.

[Han et al., 2012] Han, J., Pauwels, E. J., de Zeeuw, P. M., and de With, P. H. N. (2012). Employing a rgb-d sensor for real-time tracking of humans across multiple re-entries in a smart environment. *IEEE Transactions on Consumer Electronics*, 58(2):255–263.

[Heydarian et al., 2015] Heydarian, A., Carneiro, J. P., Gerber, D., Becerik-Gerber, B., Hayes, T., and Wood, W. (2015). Immersive virtual environments versus physical built environments: A benchmarking study for building design and user-built environment explorations. *Automation in Construction*, 54:116–126.

[Hilfert and König, 2016] Hilfert, T. and König, M. (2016). Low-cost virtual reality environment for engineering and construction. *Visualization in Engineering*, 4(1):2.

[Hilkert, 2008] Hilkert, J. (2008). Inertially stabilized platform technology concepts and principles. *Control Systems, IEEE*, 28(1):26–46.

[Hiroya Yamada, 2013] Hiroya Yamada, Shunichi Takaoka, S. H. (2013). A snake-like robot for real-world inspection applications (the design and control of a practical active cord mechanism). *Advanced Robotics*, 27(1):47–60.

[Hoffman and Breazeal, 2008] Hoffman, G. and Breazeal, C. (2008). Achieving Fluency Through Perceptual-symbol Practice in Human-robot Collaboration. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, HRI '08, pages 1–8, New York, NY, USA. ACM.

[Hou and Shen, 2010] Hou, F. and Shen, W. (2010). On the complexity of optimal reconfiguration planning for modular reconfigurable robots. In *2010 IEEE International Conference on Robotics and Automation*, pages 2791–2796.

[Hou and Shen, 2014] Hou, F. and Shen, W.-M. (2014). Graph-based optimal reconfiguration planning for self-reconfigurable robots. *Robotics and Autonomous Systems*, 62(7):1047 – 1059. Reconfigurable Modular Robotics.

[Hu et al., 2009] Hu, Y., Zhao, W., Wang, L., and Jia, Y. (2009). Underwater target following with a vision-based autonomous robotic fish. In *2009 American Control Conference*, pages 5265–5270.

[Huff et al., 2012] Huff, J., Voyles, R., and Tadokoro, S. (2012). Explosion proof active scope camera. In *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, pages 1–3.

[Hume and Wagner, 2018] Hume, J. B. and Wagner, M. (2018). A death in the family: Sea lamprey (petromyzon marinus) avoidance of confamilial alarm cues diminishes with phylogenetic distance. *Ecology and Evolution*, 8(7):3751–3762.

[Hurst et al., 2010] Hurst, J. W., Chestnutt, J. E., and Rizzi, A. A. (2010). The Actuator With Mechanically Adjustable Series Compliance. *IEEE Transactions on Robotics*, 26(4):597–606.

[Hutter et al., 2013] Hutter, M., Remy, C. D., Hoepflinger, M. A., and Siegwart, R. (2013). Efficient and Versatile Locomotion With Highly Compliant Legs. *IEEE/ASME Transactions on Mechatronics*, 18(2):449–458.

[Ichida et al., 2004] Ichida, H., Itoh, Y., Kitamura, Y., and Kishino, F. (2004). Interactive Retrieval of 3d Shape Models Using Physical Objects. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, MULTIMEDIA '04, pages 692–699, New York, NY, USA. ACM.

[Iida et al., 2007] Iida, F., Rummel, J., and Seyfarth, A. (2007). Bipedal Walking and Running with Compliant Legs. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3970–3975.

[Ijspeert, 2008] Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653.

[Ijspeert et al., 2007] Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420.

[iRobot, 2018] iRobot (2018). Roomba.

[Ishii et al., 2012] Ishii, H., Lakatos, D., Bonanni, L., and Labrune, J.-B. (2012). Radical Atoms: Beyond Tangible Bits, Toward Transformable Materials. *interactions*, 19(1):38–51.

[Ishii et al., 2004] Ishii, H., Ratti, C., Piper, B., Wang, Y., Biderman, A., and Ben-Joseph, E. (2004). Bringing Clay and Sand into Digital Design — Continuous Tangible user Interfaces. *BT Technology Journal*, 22(4):287–299.

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. (2011). KinectFusion: Real-time 3d Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA. ACM.

[Jabbari Arfaee et al., 2011] Jabbari Arfaee, S., Zilles, S., and Holte, R. C. (2011). Learning heuristic functions for large state spaces. *Artif. Intell.*, 175(16-17):2075–2098.

[Jing et al., 2016] Jing, G., Tosun, T., Yim, M., and Kress-Gazit, H. (2016). An end-to-end system for accomplishing tasks with modular robots. In *Robotics: Science and Systems*, pages 1–5.

[Johnson and Thomas, 2010] Johnson, S. and Thomas, A. P. (2010). Squishy Circuits: A Tangible Medium for Electronics Education. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 4099–4104, New York, NY, USA. ACM.

[Kalakrishnan et al., 2011] Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., and Schaal, S. (2011). Learning, planning, and control for quadruped locomotion over challenging terrain , Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258.

[Kamali Sarvestani et al., 2013] Kamali Sarvestani, I., Kozlov, A., Harischandra, N., Grillner, S., and Ekeberg, Ö. (2013). A computational model of visually guided locomotion in lamprey. *Biological Cybernetics*, 107(5):497–512.

[Kamel Boulos et al., 2011] Kamel Boulos, M. N., Blanchard, B. J., Walker, C., Montero, J., Tripathy, A., and Gutierrez-Osuna, R. (2011). Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation. *International Journal of Health Geographics*, 10(1):45.

[Kandel et al., 2013] Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., and Hudspeth, A. J. (2013). *Principles of Neural Science, Fifth Edition*. McGraw-Hill.

[Khodr et al., 2019] Khodr, H., Mutlu, M., Hauser, S., Bernardino, A., and Ijspeert, A. (2019). An optimal planning framework to deploy self-reconfigurable modular robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page (Submitted).

[Kikuuwe et al., 2010] Kikuuwe, R., Yasukouchi, S., Fujimoto, H., and Yamamoto, M. (2010). Proxy-Based Sliding Mode Control: A Safer Extension of PID Position Control. *IEEE Transactions on Robotics*, 26(4):670–683.

[Klompmaker et al., 2012] Klompmaker, F., Nebe, K., and Fast, A. (2012). dSensingNI: A Framework for Advanced Tangible Interaction Using a Depth Camera. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, TEI '12, pages 217–224, New York, NY, USA. ACM.

[Koppula et al., 2013] Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970.

[Kotay and Rus, 2000] Kotay, K. D. and Rus, D. L. (2000). 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 vol.3.

[Kozima et al., 2009] Kozima, H., Michalowski, M. P., and Nakagawa, C. (2009). Keepon. *International Journal of Social Robotics*, 1(1):3–18.

[Krupke et al., 2012] Krupke, D., Li, G., Zhang, J., Zhang, H., and Hildre, H. P. (2012). Flexible modular robotic simulation environment for research and education. In *European Conference on Modelling and Simulation (ECMS)*, pages 243–249.

[Kurokawa et al., 2008] Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., and Murata, S. (2008). Distributed Self-Reconfiguration of M-TRAN III Modular Robotic System. *The International Journal of Robotics Research*, 27(3-4):373–386.

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

[Lapicque, 1907] Lapicque, L. (1907). Recherches quantitatives sur l'excitation electrique des nerfs traitée comme une polarization. *J. Physiol. Pathol.*, 9:620–635.

[Larkworthy and Ramamoorthy, 2010] Larkworthy, T. and Ramamoorthy, S. (2010). An efficient algorithm for self-reconfiguration planning in a modular robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 5139–5146.

[Lasa and Buehler, 2001] Lasa, M. d. and Buehler, M. (2001). Dynamic compliant quadruped walking. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 3153–3158 vol.3.

[Lichtenstern et al., 2012] Lichtenstern, M., Frassl, M., Perun, B., and Angermann, M. (2012). A Prototyping Environment for Interaction Between a Human and a Robotic Multi-agent System. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, pages 185–186, New York, NY, USA. ACM.

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128×128 120 db 15μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576.

[Liedke et al., 2013] Liedke, J., Matthias, R., Winkler, L., and Wörn, H. (2013). The Collective Self-reconfigurable Modular Organism (CoSMO). In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1–6.

[Liljebäck et al., 2013] Liljebäck, P., Pettersen, K. Y., Stavdahl, Ø., and Gravdahl, J. T. (2013). Future research challenges of snake robot locomotion. In *Snake Robots*, pages 287–291. Springer.

[Liljeback et al., 2014] Liljeback, P., Stavdahl, O., Pettersen, K., and Gravdahl, J. (2014). Mamba - a waterproof snake robot with tactile sensing. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 294–301.

[Liu et al., 2015] Liu, J., Liu, Y., Zhang, G., Zhu, P., and Chen, Y. Q. (2015). Detecting and tracking people in real time with rgb-d camera. *Pattern Recognition Letters*, 53:16 – 23.

[Magnússon et al., 2013] Magnússon, A., Pacheco, M., Moghadam, M., Lund, H. H., and Christensen, D. J. (2013). Fable: Socially Interactive Modular Robot. *The Eighteenth International Symposium on Artificial Life and Robotics*, page 9.

[Maitin-Shepard et al., 2010] Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., and Abbeel, P. (2010). Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315.

[Manfredi et al., 2013] Manfredi, L., Assaf, T., Mintchev, S., Marrazza, S., Capantini, L., Orofino, S., Ascari, L., Grillner, S., Wallén, P., Ekeberg, Ö., Stefanini, C., and Dario, P. (2013). A bioinspired autonomous swimming robot as a tool for studying goal-directed locomotion. *Biological Cybernetics*, 107(5):513–527.

[Martin et al., 2009] Martin, S., Kelly, G., Kernohan, W. G., McCreight, B., and Nugent, C. (2009). Smart home technologies for health and social care support. *Cochrane Database of Systematic Reviews 2008*, 4(2):13.

[Masato, 2001] Masato, H. (2001). Development of humanoid robot ASIMO.

[McLurkin et al., 2006] McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., and Schmidt, B. (2006). Speaking Swarmish: Human-Robot Interface Design for Large Swarms of Autonomous Mobile Robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 72–75.

[Melo et al., 2013] Melo, K., Leon, J., di Zeo, A., Rueda, V., Roa, D., Parraga, M., Gonzalez, D., and Paez, L. (2013). The Modular Snake Robot Open Project: Turning animal functions into engineering tools. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6.

[Melo and Paez, 2014] Melo, K. and Paez, L. (2014). Experimental determination of control parameter intervals for repeatable gaits in modular snake robots. In *Safety, Security, and Rescue Robotics (SSRR), 2014 IEEE International Symposium on*, pages 1–7.

[Mondada et al., 2005] Mondada, F., Gambardella, L. M., Floreano, D., Nolfi, S., Deneuborg, J. ., and Dorigo, M. (2005). The cooperation of swarm-bots: physical interactions in collective robotics. *IEEE Robotics Automation Magazine*, 12(2):21–28.

[Motion, 2018] Motion, L. (2018). Leap Motion.

[Mueggler et al., 2015] Mueggler, E., Baumli, N., Fontana, F., and Scaramuzza, D. (2015). Towards evasive maneuvers with quadrotors using dynamic vision sensors. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–8.

[Murata et al., 2007] Murata, S., Kakomura, K., and Kurokawa, H. (2007). Toward a scalable modular robotic system. *IEEE Robotics & Automation Magazine*, 14(4):56–63.

[Murphy, 2014] Murphy, R. R. (2014). *Disaster Robotics*. The MIT Press.

[Mutlu et al., 2016] Mutlu, M., Bonardi, S., Vespignani, M., Hauser, S., Bernardino, A., and Ijspeert, A. J. (2016). Natural user interface for lighting control: Case study on desktop lighting using modular robots. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 288–293.

[Mutlu et al., 2017] Mutlu, M., Hauser, S., Bernardino, A., and Ijspeert, A. (2017). Effects of joint compliance in quadrupedal locomotion. page 2.

[Mutlu et al., 2018] Mutlu, M., Hauser, S., Bernardino, A., and Ijspeert, A. (2018). Playdough to Roombots: Towards a novel tangible user interface for self-reconfigurable modular robots. page in press. IEEE.

[Mutlu et al., 2014] Mutlu, M., Saranli, A., and Saranli, U. (2014). A real-time inertial motion blur metric: Application to frame triggering based motion blur minimization. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 671–676.

[Nakagaki et al., 2016] Nakagaki, K., Vink, L., Counts, J., Windham, D., Leithinger, D., Follmer, S., and Ishii, H. (2016). Materiable: Rendering Dynamic Material Properties in Response to Direct Physical Touch with Shape Changing Interfaces. pages 2764–2772. ACM Press.

[NanoPi, 2017] NanoPi (2017). *NEO Plus2*. FriendlyElec.

[Neubert and Lipson, 2015] Neubert, J. and Lipson, H. (2015). Soldercubes: a self-soldering self-reconfiguring modular robot system. *Autonomous Robots*.

[Nguyen et al., 2001] Nguyen, L. A., Bualat, M., Edwards, L. J., Flueckiger, L., Neveu, C., Schwehr, K., Wagner, M. D., and Zbinden, E. (2001). Virtual Reality Interfaces for Visualization and Control of Remote Vehicles. *Autonomous Robots*, 11(1):59–68.

[Nigolian et al., 2017] Nigolian, V., Mutlu, M., Hauser, S., Bernardino, A., and Ijspeert, A. (2017). Self-reconfigurable modular robot interface using virtual reality: Arrangement of furniture made out of roombots modules. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 772–778.

[Nousch and Jung, 1999] Nousch, M. and Jung, B. (1999). CAD on the World Wide Web: Virtual Assembly of Furniture with BEAVER. In *Proceedings of the Fourth Symposium on Virtual Reality Modeling Language*, VRML '99, pages 113–119, New York, NY, USA. ACM.

[Odashima et al., 2006] Odashima, T., Onishi, M., Tahara, K., Takagi, K., Asano, F., Kato, Y., Nakashima, H., Kobayashi, Y., Mukai, T., Luo, Z., and Hosoe, S. (2006). A Soft Human-Interactive Robot RI-MAN. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–1.

[O'keefe and Nadel, 1978] O'keefe, J. and Nadel, L. (1978). *The hippocampus as a cognitive map.* Oxford: Clarendon Press.

[OriLiving, 2019] OriLiving (2019). Robotic wardrobes.

[Owaki et al., 2017] Owaki, D., Goda, M., Miyazawa, S., and Ishiguro, A. (2017). A Minimal Model Describing Hexapedal Interlimb Coordination: The Tegotae-Based Approach. *Frontiers in Neurorobotics*, 11.

[Owaki et al., 2012] Owaki, D., Morikawa, L., and Ishiguro, A. (2012). Listen to body's message: Quadruped robot that fully exploits physical interaction between legs. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1950–1955.

[Ozgur et al., 2014] Ozgur, A., Bonardi, S., Vespignani, M., Mockel, R., and Ijspeert, A. J. (2014). Natural user interface for Roombots. In *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, pages 12–17. IEEE.

[Pamecha et al., 1997] Pamecha, A., Ebert-Uphoff, I., and Chirikjian, G. S. (1997). Useful metrics for modular robot motion planning. *IEEE Transactions on Robotics and Automation*, 13(4):531–545.

[Patil and Ravi, 2005] Patil, S. and Ravi, B. (2005). Voxel-based representation, display and thickness analysis of intricate shapes. In *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, pages 6 pp.–.

[Piccardi, 2004] Piccardi, M. (2004). Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 4, pages 3099–3104 vol.4.

[Piper et al., 2002] Piper, B., Ratti, C., and Ishii, H. (2002). Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 355–362, New York, NY, USA. ACM.

[Raffle et al., 2004] Raffle, H. S., Parkes, A. J., and Ishii, H. (2004). Topobo: A Constructive Assembly System with Kinetic Memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 647–654, New York, NY, USA. ACM.

[Ramey et al., 2011] Ramey, A., Gonzalez-Pacheco, V., and Salichs, M. A. (2011). Integration of a low-cost rgb-d sensor in a social robot for gesture recognition. In *2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 229–230.

[Rebai et al., 2012] Rebai, K., Azouaoui, O., and Achour, N. (2012). Bio-inspired visual memory for robot cognitive map building and scene recognition. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2985–2990.

[Recchiuto et al., 2015] Recchiuto, C. T., Sgorbissa, A., and Zaccaria, R. (2015). Usability evaluation with different viewpoints of a Human-Swarm interface for UAVs control in formation. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 523–528.

[Reid et al., 2015] Reid, C. R., Lutz, M. J., Powell, S., Kao, A. B., Couzin, I. D., and Garnier, S. (2015). Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. *Proceedings of the National Academy of Sciences*.

[Rollinson et al., 2014] Rollinson, D., Bilgen, Y., Brown, B., Enner, F., Ford, S., Layton, C., Rembisz, J., Schwerin, M., Willig, A., Velagapudi, P., and Choset, H. (2014). Design and architecture of a series elastic snake robot. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4630–4636.

[Romanishin et al., 2015] Romanishin, J. W., Gilpin, K., Claici, S., and Rus, D. (2015). 3d M-Blocks: Self-Reconfiguring robots capable of locomotion via pivoting in three dimensions. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1925–1932. IEEE.

[Romanishin et al., 2013] Romanishin, J. W., Gilpin, K., and Rus, D. (2013). M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295.

[Rubenstein et al., 2012] Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3293–3298.

[Rusinkiewicz et al., 2002] Rusinkiewicz, S., Hall-Holt, O., and Levoy, M. (2002). Real-time 3d Model Acquisition. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 438–446, New York, NY, USA. ACM.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, Upper Saddle River, NJ, USA.

[Salemi et al., 2006] Salemi, B., Moll, M., and Shen, W.-M. (2006). 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.

[Salvador et al., 2002] Salvador, F., Forza, C., and Rungtusanatham, M. (2002). Modularity, product variety, production volume, and component sourcing: theorizing beyond generic prescriptions. *Journal of Operations Management*, 20(5):549 – 575.

[Sattar et al., 2005] Sattar, J., Giguere, P., Dudek, G., and Prahacs, C. (2005). A visual servoing system for an aquatic swimming robot. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1483–1488.

[Simeone et al., 2015] Simeone, A. L., Velloso, E., and Gellersen, H. (2015). Substitutional Reality: Using the Physical Environment to Design Virtual Reality Experiences. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3307–3316, New York, NY, USA. ACM.

[Sirkin et al., 2015] Sirkin, D., Mok, B., Yang, S., and Ju, W. (2015). Mechanical ottoman: How robotic furniture offers and withdraws support. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, pages 11–18, New York, NY, USA. ACM.

[Sproewitz, 2010] Sproewitz, A. (2010). *Roombots: Design and Implementation of a Modular Robot for Reconfiguration and Locomotion.* PhD thesis, EPFL.

[Sproewitz et al., 2009] Sproewitz, A., Billard, A., Dillenbourg, P., and Ijspeert, A. (2009). Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture. In *IEEE International Conference on Robotics and Automation, 2009. ICRA '09*, pages 4259–4264.

[Sproewitz et al., 2010a] Sproewitz, A., Laprade, P., Bonardi, S., Mayer, M., Moeckel, R., Mudry, P.-A., and Ijspeert, A. J. (2010a). Roombots - Towards decentralized reconfiguration with self-reconfiguring modular robotic metamodules. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1126–1132, Taipei. IEEE.

[Sproewitz et al., 2008] Sproewitz, A., Moeckel, R., Maye, J., and Ijspeert, A. J. (2008). Learning to Move in Modular Robots using Central Pattern Generators and Online Optimization. *The International Journal of Robotics Research*, 27(3-4):423–443.

[Sproewitz et al., 2014] Sproewitz, A., Moeckel, R., Vespignani, M., Bonardi, S., and Ijspeert, A. (2014). Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot. *Robotics and Autonomous Systems*, 62(7):1016–1033.

[Sproewitz et al., 2010b] Sproewitz, A., Pouya, S., Bonardi, S., Kieboom, J. V. D., Mockel, R., Billard, A., Dillenbourg, P., and Ijspeert, A. J. (2010b). Roombots: Reconfigurable Robots for Adaptive Furniture. *IEEE Computational Intelligence Magazine*, 5(3):20–32.

[Sproewitz et al., 2013] Sproewitz, A., Tuleu, A., Vespignani, M., Ajallooeian, M., Badri, E., and Ijspeert, A. J. (2013). Towards dynamic trot gait locomotion: Design, control, and experiments with Cheetah-cub, a compliant quadruped robot. *The International Journal of Robotics Research*, 32(8):932–950.

[Srinivasa et al., 2012] Srinivasa, S. S., Berenson, D., Cakmak, M., Collet, A., Dogar, M. R., Dragan, A. D., Knepper, R. A., Niemueller, T., Strabala, K., Weghe, M. V., and Ziegler, J. (2012). Herb 2.0: Lessons Learned From Developing a Mobile Manipulator for the Home. *Proceedings of the IEEE*, 100(8):2410–2428.

[Stoy, 2006] Stoy, K. (2006). Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems*, 54(2):135–141.

[Stoy et al., 2010] Stoy, K., Brandt, D., and Christensen, D. (2010). *Self-Reconfigurable Robots: An Introduction.* MIT Press.

[Stoy and Nagpal, 2004] Stoy, K. and Nagpal, R. (2004). Self-repair through scale independent self-reconfiguration. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 2062–2067 vol.2.

[Straub et al., 2015] Straub, J., Kading, B., Mohammad, A., and Kerlin, S. (2015). Characterization of a Large, Low-Cost 3d Scanner. *Technologies*, 3(1):19–36.

[Sugahara et al., 2004] Sugahara, Y., Hosobata, T., Mikuriya, Y., Sunazuka, H., Lim, H.-o., and Takanishi, A. (2004). Realization of dynamic human-carrying walking by a biped locomotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 3, pages 3055–3060 Vol.3.

[Sugden, 1995] Sugden, W. H. (1995). Computer controlled stage lighting system.

[Takuma et al., 2008] Takuma, T., Hayashi, S., and Hosoda, K. (2008). 3d bipedal robot with tunable leg compliance mechanism for multi-modal locomotion. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1097–1102.

[Terada and Murata, 2008] Terada, Y. and Murata, S. (2008). Automatic Modular Assembly System and its Distributed Control. *The International Journal of Robotics Research*, 27(3-4):445–462.

[Tesch et al., 2013] Tesch, M., Schneider, J., and Choset, H. (2013). Expensive multiobjective optimization for robotics. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 973–980.

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

[Tosun et al., 2018] Tosun, T., Jing, G., Kress-Gazit, H., and Yim, M. (2018). *Computer-Aided Compositional Design and Verification for Modular Robots*, pages 237–252. Springer International Publishing, Cham.

[Ugurlu et al., 2013] Ugurlu, B., Kotaka, K., and Narikiyo, T. (2013). Actively-compliant locomotion control on rough terrain: Cyclic jumping and trotting experiments on a stiff-by-nature quadruped. In *2013 IEEE International Conference on Robotics and Automation*, pages 3313–3320.

[Unsal et al., 2001] Unsal, C., Kiliccote, H., and Khosla, P. K. (2001). A modular self-reconfigurable bipartite robotic system: Implementation and motion planning. *Autonomous Robots*, 10(1):23–40.

[van den Berg et al., 2011] van den Berg, J., Miller, S., Goldberg, K., and Abbeel, P. (2011). *Gravity-Based Robotic Cloth Folding*, chapter 6, pages 409–424. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Vasconcelos et al., 2017] Vasconcelos, R., Hauser, S., Dzeladini, F., Mutlu, M., Horvat, T., Melo, K., Oliveira, P., and Ijspeert, A. (2017). Active stabilization of a stiff quadruped robot using local feedback. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4903–4910.

[Vega et al., 2013] Vega, J., Perdices, E., and Cañas, J. M. (2013). Robot evolutionary localization based on attentive visual short-term memory. *Sensors (Basel)*, 13(1):1268–1299.

[Vertut, 2013] Vertut, J. (2013). *Teleoperation and Robotics: Applications and Technology*. Springer Science & Business Media. Google-Books-ID: ZEfqCAAAQBAJ.

[Vespignani, 2015] Vespignani, M. (2015). *Challenges in the Locomotion of Self-Reconfigurable Modular Robots*. PhD thesis.

[Vespignani et al., 2015] Vespignani, M., Melo, K., Mutlu, M., and Ijspeert, A. J. (2015). Compliant snake robot locomotion on horizontal pipes. In *Safety, Security, and Rescue Robotics (SSRR), 2015 IEEE International Symposium on*, pages 1–8. IEEE.

[Vespignani et al., 2013] Vespignani, M., Senft, E., Bonardi, S., Moeckel, R., and Ijspeert, A. J. (2013). An experimental study on the role of compliant elements on the locomotion of the self-reconfigurable modular robots Roombots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4308–4313. Ieee.

[Villagrasa et al., 2014] Villagrasa, S., Fonseca, D., and Durán, J. (2014). Teaching Case: Applying Gamification Techniques and Virtual Reality for Learning Building Engineering 3d Arts. In *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM '14, pages 171–177, New York, NY, USA. ACM.

[Wagner and Bals, 2012] Wagner, C. M. and Bals, J. D. (2012). Behavioral responses of sea lamprey (petromyzon marinus) to a putative alarm cue derived from conspecific and heterospecific sources. *Behaviour*, 149(9):901 – 923.

[Watanabe et al., 2004] Watanabe, R., Itoh, Y., Asai, M., Kitamura, Y., Kishino, F., and Kikuchi, H. (2004). The Soul of ActiveCube: Implementing a Flexible, Multimodal, Three-dimensional Spatial Tangible Interface. *Comput. Entertain.*, 2(4):15–15.

[Webots, 2018] Webots (2018). Robotics simulator.

[Wei et al., 2011] Wei, H., Chen, Y., Tan, J., and Wang, T. (2011). Sambot: A self-assembly modular robot system. *IEEE/ASME Transactions on Mechatronics*, 16(4):745–757.

[Weller et al., 2011] Weller, M. P., Gross, M. D., and Goldstein, S. C. (2011). Hyperform Specification: Designing and Interacting with Self-reconfiguring Materials. *Personal Ubiquitous Comput.*, 15(2):133–149.

[Wilson et al., 2012] Wilson, A., Benko, H., Izadi, S., and Hilliges, O. (2012). Steerable Augmented Reality with the Beamatron. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 413–422, New York, NY, USA. ACM.

[Winkler et al., 2012] Winkler, M. B., Höver, K. M., Hadjakos, A., and Mühlhäuser, M. (2012). Automatic Camera Control for Tracking a Presenter during a Talk. In *2012 IEEE International Symposium on Multimedia*, pages 471–476.

[Wright et al., 2012] Wright, C., Buchan, A., Brown, B., Geist, J., Schwerin, M., Rollinson, D., Tesch, M., and Choset, H. (2012). Design and architecture of the unified modular snake robot. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4347–4354.

[Wu and Ma, 2011] Wu, X. and Ma, S. (2011). Sensor-driven neural controller for self-adaptive collision-free behavior of a snake-like robot. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 191–196.

[Ye et al., 2013] Ye, M., Zhang, Q., Wang, L., Zhu, J., Yang, R., and Gall, J. (2013). *A Survey on Human Motion Analysis from Depth Data*, chapter 2, pages 149–187. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Yim et al., 2000] Yim, M., Duff, D., and Roufas, K. (2000). PolyBot: a modular reconfigurable robot. In *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*, volume 1, pages 514–520 vol.1.

[Yim et al., 2007] Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., and Chirikjian, G. (2007). Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]. *IEEE Robotics Automation Magazine*, 14(1):43–52.

[Yoshida et al., 2002] Yoshida, E., Murata, S., Kamimura, A., Tomita, K., Kurokawa, H., and Kokaji, S. (2002). A self-reconfigurable modular robot: Reconfiguration planning and experiments. *The International Journal of Robotics Research*, 21(10-11):903–915.

[Yu et al., 2007] Yu, C.-H., Willems, F., Ingber, D., and Nagpal, R. (2007). Self-organization of environmentally-adaptive shapes on a modular robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2353–2360.

[Yu et al., 2016] Yu, J., Sun, F., Xu, D., and Tan, M. (2016). Embedded vision-guided 3-d tracking control for robotic fish. *IEEE Transactions on Industrial Electronics*, 63(1):355–363.

[Yu et al., 2014] Yu, J., Tan, M., Chen, J., and Zhang, J. (2014). A survey on cpg-inspired control models and system implementation. *IEEE Transactions on Neural Networks and Learning Systems*, 25(3):441–456.

[Zhen et al., 2015] Zhen, W., Gong, C., and Choset, H. (2015). Modeling rolling gaits of a snake robot. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3741–3746. IEEE.

[Zhu et al., 2014] Zhu, Y., Jin, H., Zhang, X., Yin, J., Liu, P., and Zhao, J. (2014). A multi-sensory autonomous docking approach for a self-reconfigurable robot without mechanical guidance. *International Journal of Advanced Robotic Systems*, 11(9):146.

[Zykov et al., 2007] Zykov, V., Chan, A., and Lipson, H. (2007). Molecubes: An open-source modular robotics kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, pages 3–6. Citeseer.

Mehmet Mutlu

Route de la Plaine 5, Apt. 46
1022, Chavannes-Près-Renens, Vaud, Switzerland
Phone: +41-76-675-5500
E-mail: mehmet.mutlu@epfl.ch
me@mutlumehmet.net
Skype: mehmutlu
LinkedIn: https://www.linkedin.com/in/mehmutlu/
Web: https://people.epfl.ch/mehmet.mutlu
https://www.mutlumehmet.net

23 March 1988
Turkish & Bulgarian
Single
B residence permit until 2023
A, B, C, D driving licence

Strengths

PCB design **Firmware development** **Control of dynamic robots**

Education

Ph.D. (Joint)	École Polytechnique Fédérale de Lausanne (EPFL)	Robotics, Control & Intelligent Systems	2019
	Instituto Superior Técnico - Lisboa (IST)	Robotics, Brain & Cognition	
M.Sc.	Middle East Technical University (METU)	Electrical & Electronics Eng. w/ Robotics	2014
B.Sc.	Middle East Technical University (METU)	Electrical & Electronics Eng. w/ Control	2011
B.Sc. Minor	Middle East Technical University (METU)	Mechatronics	2011

Professional Experience

EPFL Biorobotics Laboratory Switzerland Oct 2014 - Present

Designing, prototyping, producing and maintaining all electronic boards in 13 Roombots modules (18 boards distributed throughout a module), Arbiter and Envirobot's head. Integrating higher level sensors such as camera, force sensor, IMU and associated computational hardware on robots. Developing firmware, communication protocols (on SPI, I2C, UART, RS-485), user interfaces and higher level motion planning. Modeling bio-inspired terrestrial and aquatic locomotion for various morphologies of modular robots. Data analysis on experiments.

IST Computer and Robot Vision Laboratory Portugal Jul 2016 - Sep 2017

Developing visual servo control for manipulators made out of Roombots. Developing novel user interfaces for SRMR.

METU Robotics Laboratory Turkey Dec 2010 - Sep 2014

Motion modeling of dynamical mechanisms, design of stabilization platform controllers, implementing hardware and software approaches for compensating motion blur, gait optimization and hardware improvements on bio-inspired, RHex variant, hexapod robot SensoRHex.

ASELSAN REHIS Turkey Jun 2010 - Jul 2010

Internship. Array antenna design, measurements and tests for military radar and electronic warfare systems.

MedSav R&D Department Turkey Jun 2009 - Jul 2009

Internship. RS485 communication board design for medical drug storage systems.

Technical Skills

PCB design	Altium Designer (primary), Ares, Eagle
Electronic simulation	Multisim, LT Spice, Isis
Microcontroller programming	Microchip PIC (XC16 and PIC-C), Arduino (C++), Motorola 68HC11 (Assembly)
Embedded PCs	NanoPi, RaspberryPi, Intel Joule, Odroid
Electronics debugging	Oscilloscope, logic analyzer, spectrum analyser, multimeter, firmware debugging tools
Programming languages	C, C++, C#, MATLAB, Simulink, Java, Python
FPGA programming	Verilog and schematic level design with Xilinx Spartan 3
VLSI chip design	Cadence
Technical drawing	Inventor, KeyCreator, Solidworks
Physical simulation	Webots, Robotics Library, Ansys Workbench
OS	Ubuntu, Windows
Other tools	Git, SVN, Inkscape, Gimp, Blender, Premiere Pro, Microsoft Office, Latex

Language Skills

	Written Proficiency	Spoken Proficiency
English	Fluent (C2)	Fluent (C1)
Turkish	Native (C2)	Native (C2)
French	(Pre-)Intermediate (A2-B1)	(Pre-)Intermediate (A2-B1)
German	Beginner (A1)	Beginner (A1)
Russian	Beginner (A0)	N/A

Scholarships & Awards

The Foundation for Science and Technology in Portugal (FCT)	Ph.D. grant	2014-19
Masters' Regatta Lisbon - amateur rowing competition, 8+1 category	2nd out of 4 teams	2018
The Scientific and Technological Research Council of Turkey	Graduate project scholarship	2011-14
METU Foundation	Undergraduate scholarship	2006-11
Coşkunöz Holding Foundation	Undergraduate scholarship	2006-11

Extracurricular Training

Business Concept - entrepreneurship course	InnoSuisse	Sep-Dec 2018
Safety Security and Rescue Robotics Summer School	IEEE-RAS	Sep 2012

Teaching Experience

EPFL	Teaching/laboratory assistant for "*Computational Motor Control*"	Spring 2017-18 Spring 2015-16
IST-Lisbon	Laboratory assistant for "*Distributed Real-Time Control*"	Fall 2016-17
EPFL	Teaching assistant for "*Electrotechnic I*"	Fall 2015-16
METU	Laboratory head-assistant for "*Laboratory of Feedback Control Systems*"	Spring 2013-14 Spring 2012-13
METU	Laboratory assistant for "*Electrical Circuits Laboratory I*"	Fall 2013-14
METU	Organizing assistant for "*Laboratory of Feedback Control Systems*" Course was given for the first time in Spring 2012 period	Fall 2012-13
METU	Teaching assistant for "*Engineering Design I*"	Fall 2012-13
METU	Laboratory assistant for "*Electrical Circuits Laboratory II*"	Spring 2011-12
METU	Laboratory assistant for "*Analog Electronics Laboratory*"	Fall 2011-12

Selected Public Events

Exhibiting Roombots as an art piece in Ars Electronica (technological art exhibition)	Sep 2016
Promoting Swiss robotics in Bay Area Science Festival	Oct 2015

Interests & Activities

Motorcycle enthusiast	2017 - Present
Rowing (amateur level)	2016 - Present
Skiing (amateur level)	2014 - Present
Collecting plastic bottle caps to be exchanged for a wheelchair in a recycling plant	Sep 2012-13
IEEE METU Student Branch, Robotics & Automation Society chairman	May 2009-10
Preparing and giving 30 hours, theoretical and practical robotics lessons for RAS	May 2009 - Jan 2010
IEEE METU Student Branch, Robotics & Automation Society vice chairman	May 2008-09
InterRail backpacking Europe tour	2009 Summer
Work & Travel Program in Ohio, the USA (First salary in an unqualified job)	2007 Summer

References

Prof. Auke Ijspeert	Ph.D. supervisor	auke.ijspeert@epfl.ch	(+41) 21 693 26 58
Prof. Alexandre Bernardino	Ph.D. co-supervisor	alexandre.bernardino@tecnico.ulisboa.pt	(+351) 21 841 82 93
Dr. Massimo Vespignani	Postdoc	massimo.vespignani@nasa.gov	(+1) 312 468 28 37
Prof. Afşar Saranlı	M.Sc. supervisor	afsars@metu.edu.tr	(+90) 312 210 45 29

Publications

Ph.D. Thesis

- **M. Mutlu**, "*Vision Based Control and Perception Methods in Self Reconfigurable Modular Robots*", École Polytechnique Fédérale de Lausanne (EPFL), Expected: June 2019, supervised by Auke Ijspeert and Alexandre Bernardino, **(In Preparation)**

M.Sc. Thesis

- **M. Mutlu**, "*A Novel Real-Time Inertial Motion Blur Metric with Applications to Motion Blur Compensation*", Middle East Technical University (METU), August 2014, supervised by Afşar Saranlı and Uluç Saranlı

Journal Articles

1. S. Hauser, **M. Mutlu**, P-A Leziart, H. Khodr, A. Bernardino, A. Ijspeert, "*Roombots Extended: Challenges of the Next Generation of Self-Reconfigurable Modular Robots and Their Application in Adaptive and Assistive Furniture*" **(To be Submitted)**
2. H. Khodr, **M. Mutlu**, S. Hauser, A. Bernardino, A. Ijspeert, "*An Optimal Planning Framework to Deploy Self-Reconfigurable Modular Robots*", IEEE, Robotics and Automation Letters, 2019. **(Submitted)**
3. **M. Mutlu**, S. Hauser, A. Bernardino, A. Ijspeert, "*Effects of Passive and Active Joint Compliance in Quadrupedal Locomotion*", Advanced Robotics, 19 July 2018
4. S. Hauser, **M. Mutlu**, P. Banzet, A. Ijspeert, "*Compliant Universal Grippers as Adaptive Feet in Legged Robots*", Advanced Robotics, 27 July 2018
5. B. Rohani, Y. Yazicioglu, **M. Mutlu**, O. Ogucu, E. Akgul and A. Saranli, "*Lagrangian Based Mathematical Modeling and Experimental Validation of a Planar Stabilized Platform for Mobile Systems*", Journal of Computational and Applied Mathematics, 21 October 2013

International Conference Proceedings

1. **M. Mutlu**, S. Hauser, M. Rickert, J. Santos-Victor, A. Ijspeert, A. Bernardino, "*Vision Based Autonomous Docking with Self-Reconfigurable Modular Robots*" **(In Preparation)**
2. **M. Mutlu**, S. Hauser, A. Bernardino and A. Ijspeert, "*Playdough to Roombots: Towards a Novel Tangible User Interface for Self-Reconfigurable Modular Robots*", IEEE, ICRA, Brisbane, May 2018.
3. S. Hauser, **M. Mutlu**, F. Freundler and A. Ijspeert, "*Stiffness Variability in Jamming of Compliant Granules and a Case Study Application in Climbing Vertical Shafts*", IEEE, ICRA, Brisbane, May 2018.
4. R. Vasconcelos, S. Hauser, F. Dzeladini, **M. Mutlu**, T. Horvat, K. Melo, O. Paulo and A. Ijspeert, "*Active Stabilization of a Stiff Quadruped Robot Using Local Feedback*", IEEE, IROS, Vancouver, September 2017.
5. V. Nigolian, **M. Mutlu**, S. Hauser, A. Bernardino and A. Ijspeert, "*Self-Reconfigurable Modular Robot Interface Using Virtual Reality: Arrangement of Furniture Made Out of Roombots Modules*", IEEE, RO-MAN, Lisbon, August 2017.
6. **M. Mutlu**, S. Bonardi, M. Vespignani, S. Hauser, A. Bernardino and A. Ijspeert, "*Natural User Interface for Lighting Control: Case Study on Desktop Lighting Using Modular Robots*", IEEE, RO-MAN, New York City, August 2016.
7. **M. Mutlu**, K. Melo, M. Vespignani, A. Bernardino and A. Jan Ijspeert, "*Where to Place Camera on a Snake Robot: Focus on Camera Trajectory and Motion Blur*", IEEE, SSRR, Chicago, October 2015.
8. M. Vespignani, K. Melo, **M. Mutlu** and A. Ijspeert, "*Compliant Snake Robot Locomotion on Horizontal Pipes*", IEEE, SSRR, Chicago, October 2015.
9. **M. Mutlu**, A. Saranli and U. Saranli, "*A Real-Time Inertial Motion Blur Metric: Application to Frame Triggering Based Motion Blur Minimization*", IEEE, ICRA, Hong Kong, June 2014.
10. E. Akgul, **M. Mutlu**, A. Saranli and Y. Yazicioglu, "*A Comparative Evaluation of Adaptive and Non-Adaptive Sliding Mode, LQR & PID Control for Platform Stabilization*", IEEE, MSC, Dubrovnik, October 2012.
11. B. Rohani, E. Akgul, **M. Mutlu**, A. Saranli and Y. Yazicioglu, "*A Nonlinear Dynamic Strategy for Mathematical Modeling and Simulation of Stabilized Platform in Planar Motion in One Body and Three Bodies*", IAM, ICACM, Ankara, October 2012.

Poster Presentations

1. J. Nguyen-Duc, **M. Mutlu**, S. Hauser, A. Bernardino and A. Ijspeert, "*Cooperative Bridge Building by Self-Reconfigurable Modular Robots Based on Ants' Stigmergic Behaviour*, AMAM, Lausanne, August 2019 **(Submitted)**
2. S. Hauser, M. Dujany, M. van der Saar, **M. Mutlu** and A. Ijspeert, "*Learning to Walk in Arbitrary Morphologies*, AMAM, Lausanne, August 2019 **(Submitted)**
3. **M. Mutlu**, S. Hauser, A. Bernardino and A. Ijspeert, "*Effects of Joint Compliance in Quadrupedal Locomotion*", AMAM, Sapporo, June 2017.
4. S. Hauser, K. Melo, **M. Mutlu** and A. Ijspeert, "*Fast State-Switching of a Jamming-Based Foot*", AMAM, Sapporo, June 2017.

APPENDIX

(Co-)Supervised Student Projects

I. Youssef	(M.Sc. Thesis) A Biologically Inspired Visuo-Motor Controller for Robotic Lamprey	2018-19
H. Khodr	(M.Sc. Thesis) Collaborative Locomotion in Self-Reconfigurable Modular Robots	2018-19
M. Riou A. Chassignet S. Montadon	Supervising a robot competition team	2018-19
F. Efremov	A Novel Self-Reconfigurable Modular Robot Concept	2017-18
M. Dujany	Localization of an Underwater Swimming Robot	2017-18
E. Klauser	EnVision, a Vision System for Envirobot: Hardware Aspects	2017-18
R. Fong	EnVision, a Vision System for Envirobot: Control	2017-18
H. Kohli	Stereo Vision in Self-Reconfigurable Modular Robotics	2017-18
E. Clément W. Gilles Y. Jian	Supervising a robot competition team	2017-18
R. Vasconcelos	(M.Sc. Thesis) CPG & Tegotae-Based Locomotion Control of Quadrupedal Modular Robots	2016-17
T-T Denisart	Hardware Integration of a Universal Gripper to the Roombot Module	2016-17
Pol Banzet	Integration of Variable Stiffness Granular Feet in a Quadruped Robot	2016-17
Q. Golay	Passing Objects: Robot-Robot Interaction with Universal Grippers	2016-17
F. Freundler	Young's Modulus Variation of a Variable Stiffness Element Based on Jamming of Compliant Granules	2016-17
V. Nigolian	User Interface for Virtual Assembly of Self-Reconfigurable Modular Robots	2016-17
A. Häfliger	Autonomous Vision Based Docking of Roombots	2015-16
A. Vardi	Self-Reconfigurable Robots For Space Exploration	2015-16
R. Dryzner	From Play-Doh to Roombots	2015-16
V. Nigolian	Immersive Interaction Framework for Self-Reconfigurable Modular Robots	2015-16
S. Bussier	Multi-Sensory Autonomous Docking Approach for a Self-Reconfigurable Robots	2015-16
A. Öztürk	Universal Gripper Controller Design with Visual Feedback	2015-16
M. Félix M. Jean C. Marie	Supervising a robot competition team	2015-16
M. Moret	Hybrid Brain Computer Interface to Control Modular Robots	2014-15

Selected Educational & Hobby Projects

Case study on discrete time multi-variable control of a quadcopter. Simulation in Matlab.	2016
3D swimming and terrestrial locomotion with snake & lamprey-like robot in Webots simulation.	2016
Vector field based formation control with E-Pucks (differential drive mobile robots) in Webots.	2015
Implementing an autonomous navigation on SensoRHex based on the first order logic planner.	2013
Implementing an information metric based exploration algorithm to enhance SLAM performance.	2013
Statistical Robotics project. Comparing open source SLAM algorithms.	2013
AI project. Ms. Pacman vs. Ghosts Competition - Ghosts' AI.	2012
Robot Vision project. Improving watershed segmentation with resolution pyramids.	2012
Computer Graphics project. Custom design CAD program.	2012
Implementing Pong game on a FPGA board.	2012
Pattern Recognition project. Combining classifiers.	2011
Intelligent Control project. Simulating proxy based sliding mode controller for a hip node of RHex.	2011
B.S. final project. A real-life Sokoban solving and playing robot.	2010-11
Mechatronics final project. Emco CNC mill & lathe renovation.	2011
Mechatronics project. Image processing based color-target shooting robotic mechanism with laser.	2010
Mechatronics project. Building DC brushed motor using scrap material from scratch.	2010
Sokoban variant game design and implementing a solver for it.	2010
Digital Electronics Laboratory project. Sudoku solver on FPGA board.	2010
Analog Electronics Laboratory project. Automatic gain controller.	2010
Sensor based light emitting target shooting robotic mechanism with laser.	2009
Mechaboard design. An education board that is used to teach microcontroller programming in ME220, ME461 and ME462 mechatronics courses for 3 years in METU. All design by myself.	2009
RoboCup SSL. First prototypes for robots.	2008-09
Mini projects. Sumo and line following robots to participate in local robotics competitions.	2007-∞
Rotating led signs, basic room entrance logging system and many basic microcontroller experiments.	

Last update: April 17, 2019