TOWARD MACHINE METABOLISM: DESIGN OF A TRUSS RECONFIGURING ROBOT

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

David Alan Hjelle

August 2009

ABSTRACT

Biological metabolism is the process by which an organism breaks down food into its constituent modular elements (catabolism) and then uses those raw materials to create new tissue (anabolism). Metabolic processes demonstrate interesting properties that are difficult to replicate in synthetic structures, such as continual reuse of modular elements in new organisms, autonomous disassembly and assembly processes, self repair, continuous adaptation to functional requirements, and robustness to resource fluctuations. Duplicating these properties in a robotic ecology and composing, decomposing, and then recomposing items out of such modular elements could have a wide range of applications, ranging from infrastructure recovery to space exploration [36].

This is a long-term goal. Many challenges present themselves. These include: the *mechanical challenges*, such as creating appropriate modular building blocks, designing a reconfigurator robot capable of manipulating these building blocks, and the low-level control and sensing necessary for such a robot; *automated design challenges*, such as designing solutions based upon functional requirements, creating a flexible and user-friendly functional requirement language, determining the availability of current resources, methods to change and adapt solutions based upon currently available compositions of building blocks, and on-the-fly adaptation to resource fluctuations and assembly errors; and pure *algorithmic challenges*, such as path planning, efficient decomposing and recomposing, and collaboration amongst a group of reconfigurator robots. This thesis focuses on the questions of mechanical design and begins to explore questions of automated design and path planning. Collaborative and distributive questions will be

approached in future work.

Within this focus, I have designed and constructed a robotic system to serve as a testbed for exploring these ideas, including:

- A reconfigurable truss system designed for robotic manipulation. These are the basic building blocks that metabolism acts upon. Five designs are presented, including two variations of the final center-threaded strut element design.
- Four design iterations of a hinge robot capable of the following: translational and rotational movements along truss elements, and twisting of truss elements. Statistics for several movements on a truss structure are shown in section 3.8.

Additionally, I have done work regarding the algorithmic questions of machine metabolism. This includes the creation of an evolutionary algorithm that aimed to evolve a direct instruction set for the reconfiguration of a truss structure into a new structure satisfying given functional constraints.

Finally, I was a collaborator on the following related works:

- With Lobo et al. [28], a evolutionary algorithm that uses construction trees to represent the reconfiguration process and reconfigures trusses from an initial structure into a final structure, again satisfying functional constraints.
- With Yun et al. [52], a reconfiguration planning algorithm that takes as input a source and destination structure and plans an optimal path for the reconfiguration hinge robot to perform the reconfiguration.

The combination of these mechanical and algorithmic contributions will yield a demonstrated metabolistic process, and allow further exploration and discovery of the ideas of machine metabolism.

BIOGRAPHICAL SKETCH

David Alan Hjelle was born on July 1, 1980 to Dean and Ila Hjelle in Fergus Falls, Minnesota. He spent his childhood there, including his high school education at Hillcrest Lutheran Academy. In August 2009, he enrolled at Dordt College in Sioux Center, Iowa, where he studied engineering (with a mechanical emphasis) and computer science. After his graduation in May 2004, he married his wife Rita on the 22nd of the same month. In December of that year, he joined Interstates Control Systems, Inc. as a Control Systems Developer. He enrolled at Cornell in August 2007. He is expecting his Master's degree in August 2009.

To my wonderful wife, Rita: my wise woman.

ACKNOWLEDGEMENTS

I am thankful for. . .

. . . my Lord and Savior Jesus Christ. Without Him, my failings would be overwhelming; with Him, they are covered by His grace. He makes my life better than I deserve.

. . . my wonderful wife Rita, who has been with me through thick and thin.

. . . Franz Nigl for his effort in collaborating on robotic design.

. . . Stephane Constantin for his electronics design and assistance.

. . . Jon Hiller for his always-ready mechanical advice.

. . . CCSL for being a great place to work.

. . . Hod Lipson for his mentorship and guidance.

. . . the NSF-EFRI proposal authors, Daniela Rus, Eric Klavins, Mark Yim, and Hod Lipson, for many insightful discussions.

. . . and, finally the U.S. National Science Foundation's Office of Emerging Frontiers in Research and Innovation, grant #0735953, for their financial support.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . iv
Acknowledgements . . . v
Table of Contents . . . vi
List of Tables . . . ix
List of Figures . . . x

1 Introduction to Machine Metabolism **1**
1.1 Biological Metabolism: The Inspiration . . . 1
1.2 Outline of this Thesis . . . 3
1.3 Properties Useful in Engineering Problems . . . 4
1.3.1 Reuse and Reconfiguration of Basic Building Blocks . . . 4
1.3.2 Automated Design from Functional Requirements . . . 5
1.3.3 Self-repair . . . 6
1.3.4 Adaptation and Robustness . . . 7
1.4 Comparison with Traditional Modular Robotics . . . 7
1.5 Introduction to Our Approach . . . 8
1.5.1 Truss Structures . . . 8
1.5.2 Reconfiguration . . . 10
1.6 Related Work in Truss and Other Climbing Robots . . . 10
1.6.1 ROMA I . . . 10
1.6.2 ROMA II . . . 12
1.6.3 TREPA . . . 14
1.6.4 Pole Climber . . . 15
1.6.5 Shady3D . . . 17
1.6.6 Inchworm . . . 18
1.6.7 SM^2 . . . 19
1.6.8 Canadarm 2 . . . 21
1.7 Related Work in Reconfiguring Robots . . . 22
1.7.1 I-Cubes . . . 23
1.7.2 Module Assembler . . . 24
1.7.3 Robotic Assembly of Space Truss Structures . . . 24

2 Building Block Design **26**
2.1 Requirements . . . 26
2.2 Potential Designs . . . 27
2.2.1 Threaded Rod . . . 27
2.2.2 Double-Threaded Hubs . . . 28
2.2.3 Threaded End Caps . . . 29
2.2.4 Telescoping Tubes . . . 29
2.2.5 Center-Threaded Rods . . . 31
2.3 Implemented Truss Element Building Blocks . . . 32

2.3.1 Carbon Fiber Rods . . . 33
2.3.2 Threaded Pinion Wire . . . 34
2.3.3 Printed Struts . . . 35

3 Robot Design **37**
3.1 Potential Morphologies . . . 37
3.1.1 Robot Arm . . . 37
3.1.2 Node Walker . . . 38
3.2 Hinge Robot . . . 39
3.3 Geared Design . . . 43
3.4 Four Module Friction Drive . . . 45
3.5 3D Printed Gear Drive . . . 50
3.6 Acrylic Gear Drive . . . 52
3.7 Other Technical Details . . . 56
3.7.1 Gearbox Variations . . . 56
3.7.2 Electronics . . . 61
3.7.3 Servos . . . 62
3.8 Movement Statistics . . . 63

4 Reconfiguration Algorithms **66**
4.1 Construction Tree Reconfiguration Algorithm . . . 66
4.1.1 Construction Tree . . . 67
4.1.2 Structure Variation Operators . . . 69
4.1.3 Genetic Algorithm . . . 71
4.1.4 Results . . . 74
4.2 Piecewise Reconfiguration Algorithm . . . 77
4.2.1 Rotation-Based Operators . . . 81
4.2.2 Cartesian-Based Instructions . . . 85
4.2.3 Generative Encoding using Cartesian-Based Instructions . . . 91
4.2.4 Discussion and Conclusions . . . 92
4.3 Reconfiguration Planning Algorithm . . . 94
4.3.1 Optimal Matching Between Trusses . . . 95
4.3.2 Planning the Reconfiguration . . . 98

5 Future Work, Impact, and Conclusions **102**
5.1 Future Work . . . 102
5.1.1 Specific Future Work . . . 102
5.1.2 Big Picture Future Work . . . 105
5.2 Future Impact . . . 109

6 Conclusions and Contributions **112**
6.1 Conclusions . . . 112
6.2 Contributions of this Thesis . . . 112
6.3 Personal Contributions . . . 113

6.4 Lessons Learned 114

LIST OF TABLES

3.1 Specifications for the robot servos. 63
3.2 Movement statistics for the hinge robot. 63

4.1 The set of three rotation-based instructions that allows truss reconfiguration. 82
4.2 The set of nine Cartesian-based instructions that allow truss reconfiguration. 88

LIST OF FIGURES

1.1 An artist's conception of machine metabolism. 2
1.2 Robots reconfiguring a truss structure. 9
1.3 The ROMA I robot. 11
1.4 The ROMA II robot. 13
1.5 The TREPA robot. 14
1.6 The pole climber robot. 16
1.7 The Shady3D robot. 18
1.8 The Inchworm robot. 19
1.9 The SM^2 robot. 20
1.10 The Canadarm 2. 21
1.11 The I-Cubes robot. 23
1.12 The module assembler robot. 24

2.1 Random access truss elements are required. 27
2.2 Simple threaded rods. 28
2.3 Double-threaded hubs with opposing-thread rods. 29
2.4 Threaded end capped modular elements. 30
2.5 Concentric telescoping tubes. 30
2.6 Center-threaded strut element detail. 31
2.7 Center-threaded strut element insertion process. 32
2.8 Carbon fiber truss elements. 33
2.9 Threaded pinion wire. 34
2.10 Geared truss elements and gears from a 3D printer. 36

3.1 The proposed hinge design on a truss. 40
3.2 The proposed truss element holding pod. 42
3.3 An internally geared hinge robot. 43
3.4 The four-module friction drive design. 46
3.5 Planetary gear hinge design. 47
3.6 Two constructed modules of the four-module friction drive design. 49
3.7 3D printed gear drive hinge robot concept. 50
3.8 Extending a servo spline. 53
3.9 3D printed gear drive hinge robot photo. 54
3.10 The acrylic gear drive hinge robot. 54
3.11 Degree-of-freedom restrained gears. 56
3.12 Belt drive for the rotational mechanism. 57
3.13 A direct drive gearbox. 58
3.14 An 8:1 spur gearbox. 59
3.15 An 16:1 gearbox utilizing a worm gear. 60
3.16 The necessary alignment of axes in the worm gearbox. 61
3.17 Steps in a truss traversal. 64

4.1 Construction tree building blocks. 68

4.2 Construction tree representation of a structure. 70
4.3 Construction tree operators. 72
4.4 Maximizing the height of a truss structure. 76
4.5 Evolutionary fitness values over 5000 generations. 77
4.6 A reconfigured truss under resource uncertainty. 78
4.7 A evolutionary bridge, design with a physics simulator. 79
4.8 Additional evolutionary results. 80
4.9 Fitness curve for rotation-based instructions. 84
4.10 Truss produced with the rotation-based operators. 84
4.11 Fitness curve for Cartesian instructions. 89
4.12 Truss from Cartesian instructions. 90
4.13 Fitness curve for generative encoding. 92
4.14 Truss from the generative encoding. 93
4.15 2D source and target structures for reconfiguration planning. . . 95
4.16 Scanning the target structure over the source structure 97
4.17 Broken connectivity by performing the matching. 98
4.18 The adjusted matching after Algorithm 2 has been applied. . . . 99
4.19 A rendering of a reconfiguration process. 101

5.1 A simple mobile robot combination robot. 106
5.2 Hinge robots assisting each other in reconfiguration. 107
5.3 A more complicated hinge robot. 107
5.4 A tread robot composed from a metabolistic system. 108

CHAPTER 1

INTRODUCTION TO MACHINE METABOLISM

1.1 Biological Metabolism: The Inspiration

Biological metabolism is the process by which an organism breaks down food into its constituent modular elements (catabolism) and then uses those raw materials to create new tissue (anabolism). Metabolic processes demonstrate interesting properties that are difficult to replicate in synthetic structures, such as continual reuse of modular elements in new organisms, autonomous disassembly and assembly processes, self repair, continuous adaptation to functional requirements, and robustness to resource fluctuations. Duplicating these properties in a robotic ecology and composing, decomposing, and then recomposing items out of such modular elements could have a wide range of applications, ranging from infrastructure recovery to space exploration [36].

Several questions arise out of this big picture idea: First, what minimal set of modular elements would serve as a basic starting point for implementing metabolic processes in a physical system? Csete and Doyle [7] suggest that biological systems can describe large number of source and target structures through a relatively small set of building blocks, known as the *bowtie architecture*, if those building blocks are chosen judiciously. Second, what algorithms can design structures given a set of raw materials and functional requirements, and how ought those requirements be specified? Third, how does one optimally transform a source object into a new target object while maintaining both physical structural constraints and kinematic robotic constraints? In this thesis, I will present initial research into all of these areas.

Figure 1.1 – *An artist's conception of machine metabolism.* Image by Jonathan Hiller, CCSL.

1.2 Outline of this Thesis

This research will be presented in the following manner.

In the first chapter, the biological inspiration for machine metabolism is explained, as well as the useful properties that it exhibits. Then, the differences between machine metabolism and traditional conceptions of modular robotics are explained, as well an overview of this approach toward machine metabolism. Finally, the various climbing, reconfiguring, and assembling robots that give some background to this current work are discussed.

In the second chapter, the design of building blocks that are robotically manipulatable is discussed. The requirements for such building blocks and various potential morphologies that such building blocks could take are discussed, limiting the discussion to truss structures. Finally, several designs of building blocks that were fabricated and utilized are demonstrated.

Chapter three discusses the design of a robot capable of reconfiguring such truss building blocks. Various morphologies—inspired by the robots shown above—that could satisfy the requirements are discussed. Many iterations of hinge robot that were designed and constructed are presented. The details of the current design are shown, including a discussion of its capabilities.

In the fourth chapter, three algorithms useful for machine metabolism are presented. Two are evolutionary algorithms, attempting to find solutions to designing truss structures (more precisely, reconfiguring truss structures from an initial state into a goal state) based upon a given set of functional requirements. The final algorithm presents a method of optimally reconfiguring a truss structure

given an initial structure and goal structure.

Chapter five discusses briefly the future directions of work in machine metabolism, and overviews some of the potential practical impact of machine metabolism.

Finally, chapter six overviews the specific contributions of this thesis and my part in each, as well as notes on lessons learned for future explorers in this field.

1.3 Properties Useful in Engineering Problems

Several properties of biological metabolism are of particular interest in engineering applications. Each will be discussed in turn. Interestingly, these properties compare favorably with arguments by Melhuish and Holland [30] that engineers tend to make systems with characteristics that include: 1) use of simple components, 2) use of identical modules, 3) reliable operation, 4) adaptability to environmental changes, and 5) robustness to failures.

1.3.1 Reuse and Reconfiguration of Basic Building Blocks

Biological systems are able to reuse the basic building blocks of life in a sustainable way, that is, the building blocks can be reused repeatably without loss of functionality. Similarly, imagine a set of building blocks suited for engineering purposes, such as the simple truss structure building blocks presented in chapter 2. Given a robotic system capable of physically reconfiguring these building blocks, one could use it to reconfigure any given building-block object into another, perfectly

recycling the constituent materials. With the proper set of building blocks, such a process could be highly efficient (as compared to traditional recycling methods, which reduce objects into lower-level materials) and encourage greater reuse of materials. The flexibility gained in being able to reconfigure general use objects from one purpose to another with little overhead is also advantageous.

1.3.2 Automated Design from Functional Requirements

In biological systems, DNA is used to encode the information necessary for the organism to perform metabolism (among other things.) While the specific mechanisms of this process are beyond the scope of this document, it is clear that the information is stored in a very highly compressed state.

An engineering analogy can be made. In order to describe a house, one can either give a high-level description of the purpose and function of the house (seals out the elements, houses a family of six plus a dog, keeps the occupants at a comfortable temperature, allows selective entrance of sunlight, and allows the occupants freedom to enter and exit) or one could give the detailed plans. Clearly, the latter blueprints are a much lower level description of the same house. Moving from the high-level description to the blueprints requires a certain amount of creativity.

For the sake our our analogy, we propose the metabolistic systems operate in the former domain. In other words, they construct the resultant object from a functional description of the requirements. This is done autonomously.

The engineering applications here are obvious: autonomously designed

systems. By restricting the search space to a small set of modular components, we can perform autonomous design (discussed further in chapter 4) using these elements. This would allow us to feed our metabolistic system a object and other raw materials—our basic building blocks—along with a functionally-specified requirement, and it would autonomously use said building blocks to satisfy our requirements.

1.3.3 Self-repair

Biological organisms have the ability to repair themselves. Though the mechanisms for this are often low-level and small in scale, certain organisms—such as starfish—have the ability to regenerate limbs and even separate organisms. Though a more perfect robotic analogy would require active and intelligent building blocks, a similar result may be obtained if the system is able to detect damage within the passive object. Metabolistic systems—due to their automated reconfiguration and design abilities—can redesign themselves on-the-fly to re-satisfy the functional requirements once damage has occurred. This could conceivably be done on a local scale with external replacement materials or on a object-global scale, where materials from one part of the object are "grafted" into the damaged section in a way that preserves the satisfaction of functional requirements.

In the ideal machine metabolistic system, the robot performing the reconfiguration could be constructed out of the same raw modular materials as the reconfigured machine, as shown in figure 1.1. Then, in the event of damage, the modular materials of the reconfigurator could be used to repair the machine.

1.3.4 Adaptation and Robustness

Within certain limits, biological organisms are able to adapt to various external conditions and fluctuations in their natural resources. In a similar way, the dynamic design capabilities of machine metabolistic systems allow autonomous adaptation to changes in the supply of raw materials. When a change in the availability of materials occurs, the autonomous design mechanisms can be used to devise other designs with the current materials that still satisfy the functional requirements. While this is, of course, limited—design solutions may not always be found—it gives a degree of flexibility heretofore unexpected in manufacturing while evident in biology. Additionally, changes in the functional requirements may occur. Automated design decisions can then be made, and the system can reconfigure as necessary to satisfy the new requirements.

1.4 Comparison with Traditional Modular Robotics

Yim et al. [49] define modular robotics as "experimental systems made by interconnecting multiple, simple, similar units." The benefits of such modular robotics are said to be "versatility, robustness, and low cost." Typically, these units are able to rearrange themselves to change their shape, often enabling the accomplishment of different functions. Some may have a overall lattice structure (as overviewed in Yim et al. [50]), but these are typically in the nature of actuated cubes or other tessellating shapes, such as Miche [17] or the Catom concept [18]. This universal actuation is beneficial for applications such as programmable matter or quickly adapting active devices. For more stationary applications, however, this extensive actuation is unnecessary [47]. Further comparisons to

modular robotics will not be made here, but related robotic systems (other than traditional single module modular robotics) are presented in section 1.6 and section 1.7.

1.5 Introduction to Our Approach

Machine metabolism is clearly a long-term goal, with many challenges along the way. In order to address these challenges, we have made a series of simplifying assumptions to allow initial demonstrations of the concept. Figure 1.2 shows a artist's conception of our initial goal. In this image, a collection of robotic reconfigurators are at work reconfiguring a bridge-like truss structure into a tower-like truss structure.

1.5.1 Truss Structures

A wide variety of modular elements could be used in metabolistic endeavors. While this decision will be discussed in more depth in chapter 2, the decision to use cubic truss structures was made due to their relative simplicity and ability to create reasonably complex and interesting structures. We also have not considered any functional elements beyond structural elements, though one could imagine kinematic elements, actuated elements, etc. This thesis will present a robotic system that can work with cubic truss structures.

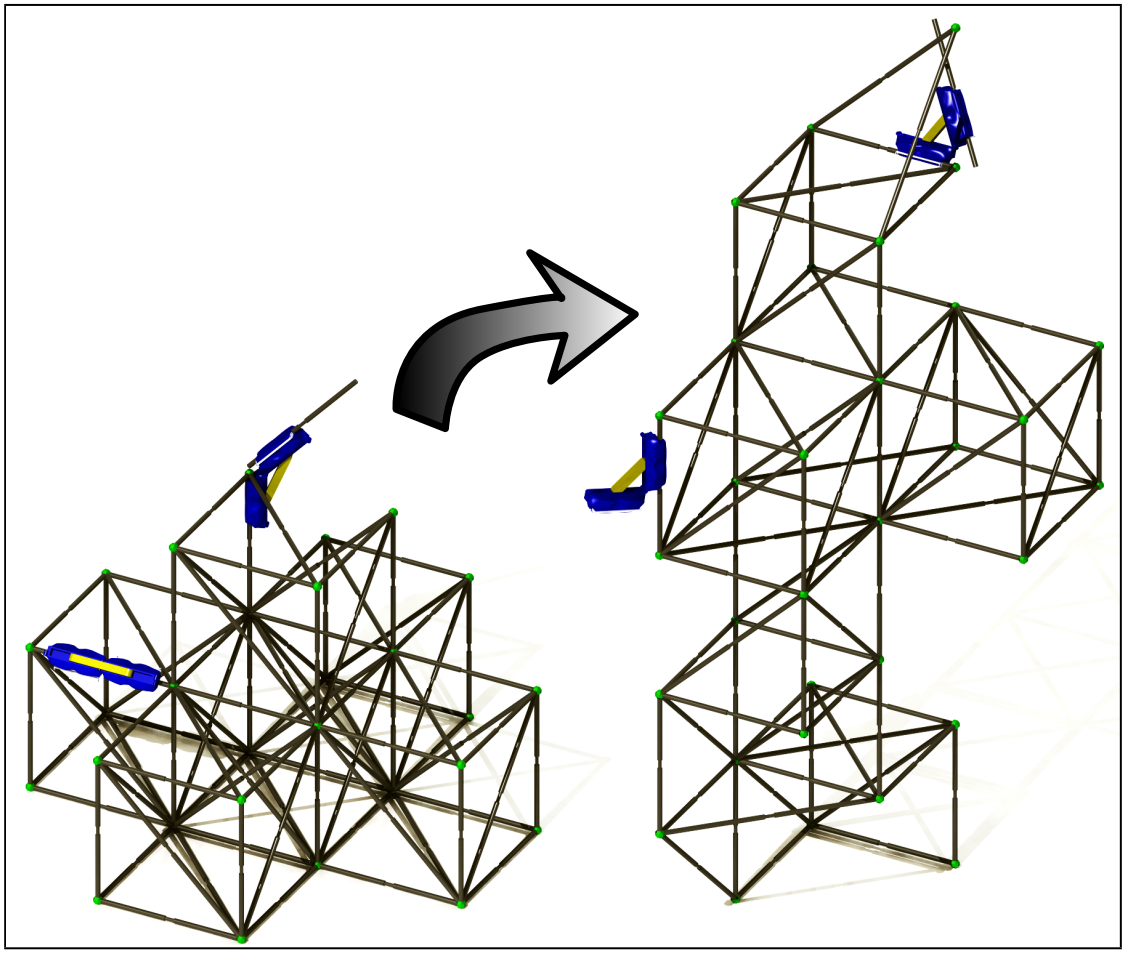

Figure 1.2 – *Robots reconfiguring a truss structure.* A set of robots autonomously reconfigures the truss structure on the left into the truss structure on the right.

1.5.2 Reconfiguration

While the automated design challenges are vast, we initially explore the problem as a reconfiguration problem: given an initial structure of some sort, what new structure can be designed that satisfies a set of functional constraints? This will be discussed more in chapter 4.

1.6 Related Work in Truss and Other Climbing Robots

This overview was published in part in Hjelle and Lipson [19].

1.6.1 ROMA I

ROMA I [2, 3] was developed at the University Carlos III of Madrid, with fully autonomous (in other words, both the control system and power supply are housed on the robot) industrial inspection and maintenance operations in mind. It is shown in figure 1.3. It has 6 degrees of freedom for movement and 2 degrees of freedom for gripper actuation: two to control each gripper's elevation, two to control each gripper's orientation, one to control the extension of the entire body, and one to rotate gripper two around the body axis. The remaining two degrees of freedom actuate the grippers. Note that the body extension and the grippers are linearly actuated, while the other degrees of freedom are rotational.

Each degree of freedom is driven by an AC motor with a PID adaptive control: due to the nonlinearities of the system, a gain table is employed to look up the appropriate PID gains for various orientations of the robot.

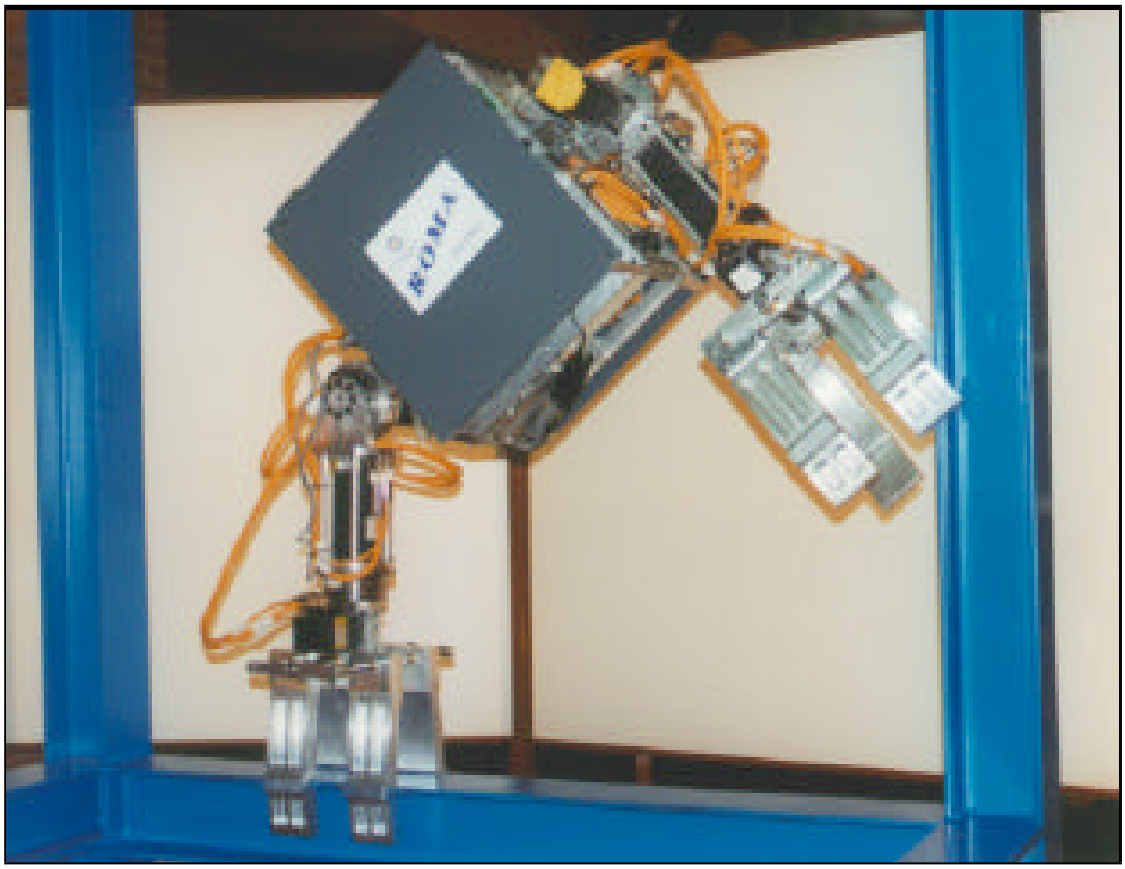

Figure 1.3 – *The ROMA I robot.* The ROMA I robot exhibiting its ability to change planes. (Image from Balaguer et al. [2].)

The movement of the robot is broken up into basic kinematic primitives that have calculated energy consumption values. Thus, for forward motion, the robot uses a "caterpillar" motion, extending and retracting its body actuator while keeping at least one gripper on the truss. Similar algorithms are used for the more complex truss motions. Additionally, possible environmental structures are divided into environment primitives, which aid in the path planning strategy.

1.6.2 ROMA II

ROMA II [3] (shown in figure 1.4) has fewer degrees of freedom (4 locomotion degrees of freedom), mixed actuation (electrical and pneumatic), and a suction gripping system to produce a robot that is 1/3 the weight of ROMA I (from ROMA I's 75 kg to 25 kg). This weight reduction is due to lighter materials and the reduced actuation. Additionally, actuation is mixed in order to utilize the precision of electrical actuation and the force of pneumatic actuation where appropriate.

The necessary degrees of freedom have been reduced by utilizing what Balaguer et al. call the *mobility criterion* and *symmetrical movement criterion*. These state that the degrees of freedom of a climbing robot can be reduced by appropriate path planning and by utilizing symmetry in the actuation of the robot. Thus, ROMA II has a single actuator to adjust the angle between the legs of the robot and the body (and therefore the distance between the feet), another actuator to adjust the pitch of the feet (both feet at once), and an actuator to rotate each foot about a nominally vertical axis.

Obviously, the suction grip of this robot does not allow truss traversal;

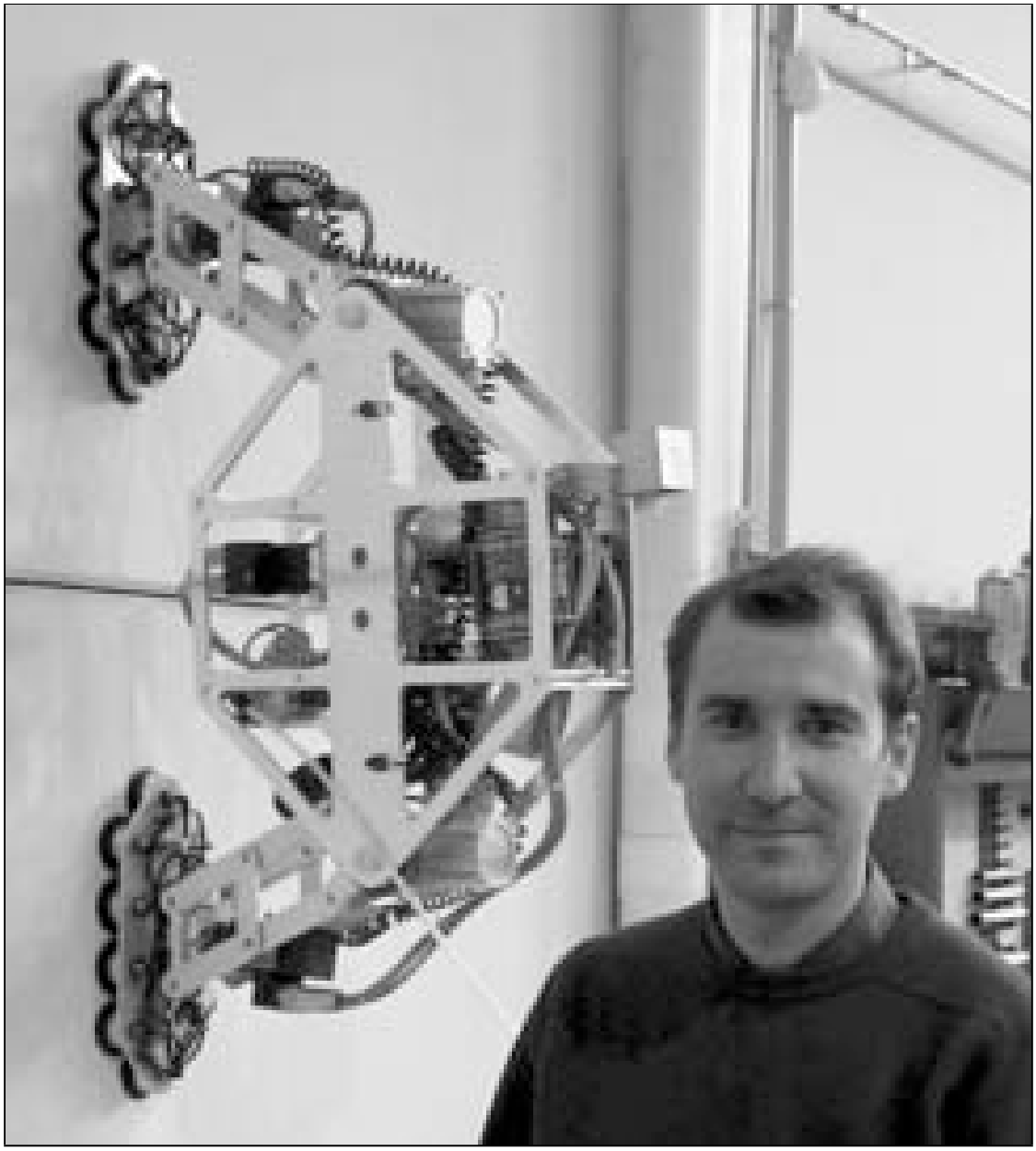

Figure 1.4 – *The ROMA II robot.* The ROMA II robot attached to a vertical wall. (Image from Balaguer et al. [3].)

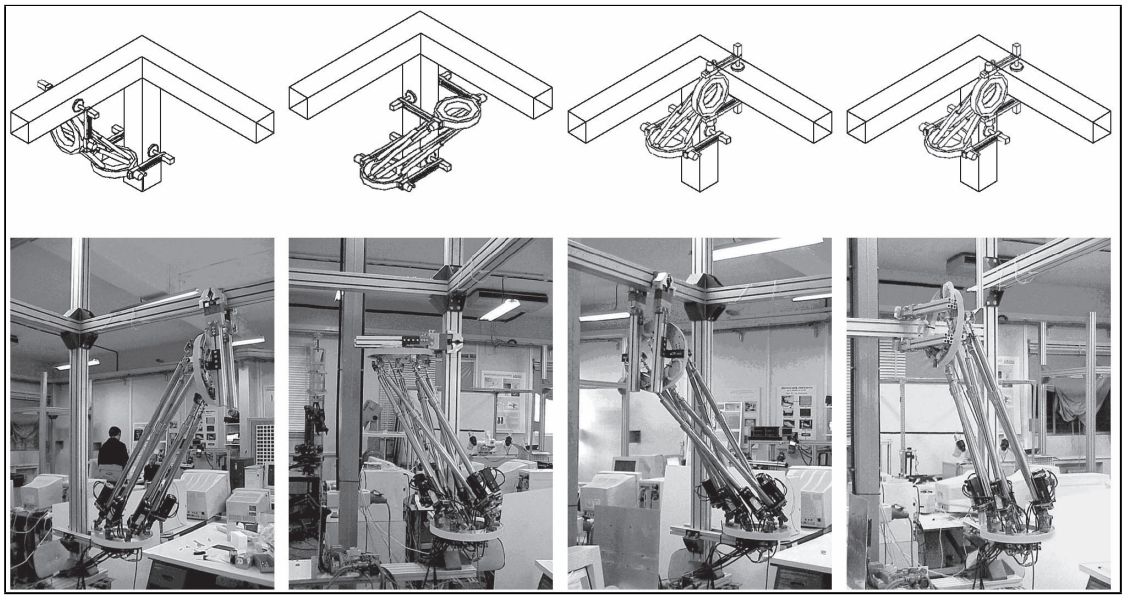

Figure 1.5 – *The TREPA robot.* The TREPA robot exhibiting some of the various possible configurations, such as those enabling bypassing of a structural node. (Image from Aracil et al. [1].)

however, replacement of the suction grip with conventional linear grippers, such as in ROMA I, would quickly allow truss traversal.

1.6.3 TREPA

A unique design amongst climbing robots, the two variations of TREPA [1] use a six-degree of freedom Stewart platform with manipulators on either end of the mechanism. (See figure 1.5.) The top and bottom faces of the platform are able to be perpendicular to each other due to specially-designed ball joints.

The first variation, designed for climbing tubular structures such as palm trees, encircles the tube and grips it with a dynamic centering algorithm. To climb the tube, the upper gripper is loosened, the structure lengthened while keep the upper ring centered, then, the upper gripper is tightened, and the lower

ring brought upwards with the same centering algorithm. The lower gripper is tightened, and the process can begin again.

A variation of this same design is used for truss climbing. Rather than encircling the structure to be climbed, the manipulators are redesigned to be a standard linear gripper capable of gripping truss structure next to the robot's body. The ability of this robot to navigate a truss structure is shown in figure 1.5.

The use of a parallel mechanism like a Stewart platform has some interesting advantages as compared to the serial mechanisms typically used in truss climbing robots. In particular, unlike many serial mechanisms, none of the actuators is required to support the entire mass of the robot. Thus, the actuators can be smaller. (Of course, this is only true for the Stewart platform actuators; the actuators in the grippers have restrictions similar to those on serial robots. Interestingly, in TREPA's case, the actuators are pneumatic, and have a high force-to-weight ratio to begin with.) There are disadvantages: the kinematics are relatively complex and the forward kinematics do not have a closed-form solution. In TREPA, a numerical method is used to derive the necessary kinematic equations.

1.6.4 Pole Climber

As if in response to TREPA's large number of degrees of freedom and strictly parallel locomotion scheme, the Sharif University of Technology in Iran developed a 4 degree of freedom pole climbing robot [40] (see figure 1.6) that is a combination of serial and parallel mechanisms. Additionally, it is electrically actuated, rather than the TREPA's pneumatic actuation.

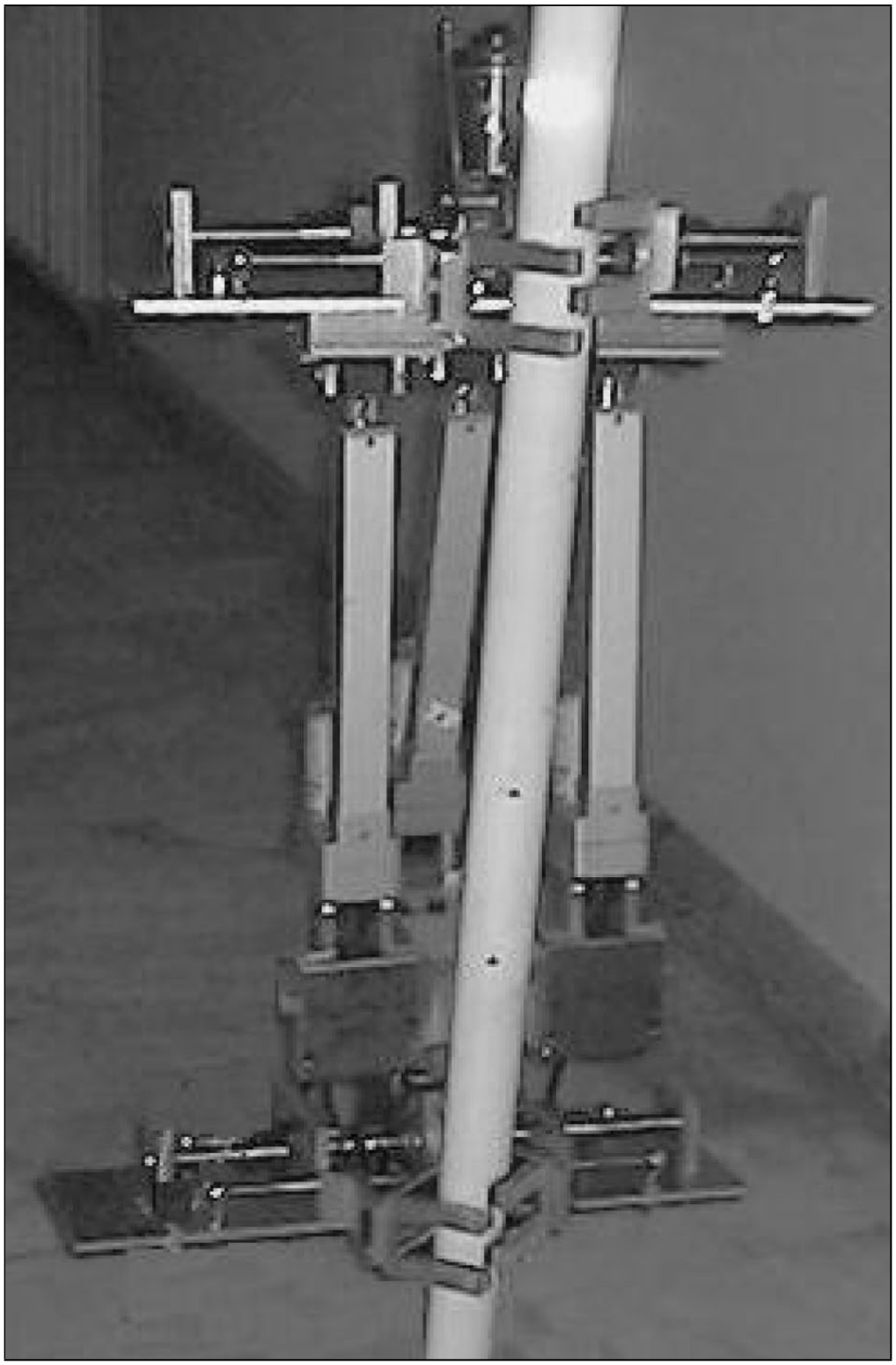

Figure 1.6 – *The pole climber robot.* The pole climber prototype with a 3 degree of freedom parallel mechanism. (Image from Tavakoli et al. [40].)

Four degrees of freedom are the minimum required for locomotion along tubular structures that have branches or bends (as was shown earlier with the ROMA II robot). In this case, a 3 degree of freedom Stewart-like platform is positioned between manipulators. In addition, one of the manipulators is able to roll with respect to the body of the robot, giving a fourth degree of freedom.

Though this design is not demonstrated climbing a truss, and it was not designed to withstand the greater moments caused on the grippers when reaching out for a perpendicular truss section, a similar kinematic design should be capable of traversing a simple cubic truss structure as described above, with a significant reduction in weight and complexity from TREPA.

1.6.5 Shady3D

Shady3D [8, 51] (as shown in figure 1.7) is another unique contribution to truss climbing robots. Rather than making a single robot with the requisite degrees of freedom, multiple robots are used: in this case, two Shady3D robots plus a passive link combine to form a robot with six independent degrees of freedom.

Each Shady3D robot has three degrees of locomotive freedom: each gripper is able to roll about its own axis and then the robot is able to twist about its center, thus changing the axis of the gripper joints from parallel to non-parallel. Provided that the two Shady3Ds are connected to the passive element such that the connected joint axis are not parallel, a 6 degree of freedom robot is created. This, then, has been demonstrated to exhibit truss climbing capabilities.

Note that the scale of Shady3D is significantly smaller than the other robots

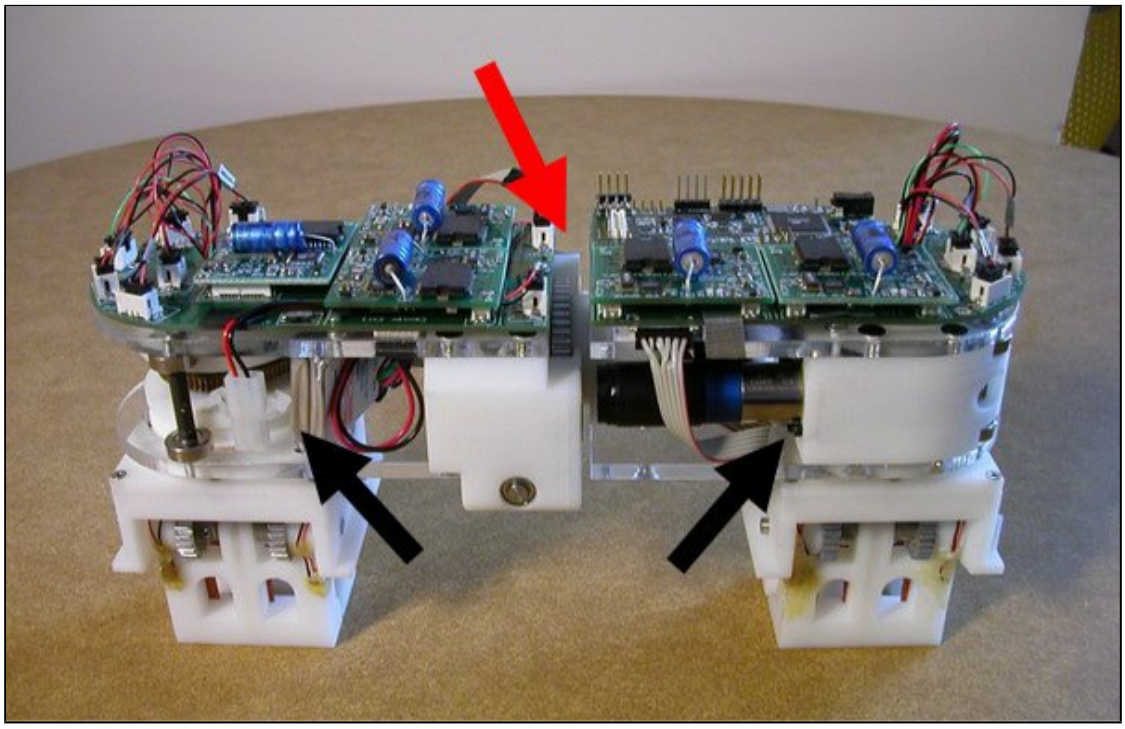

Figure 1.7 – *The Shady3D robot.* Attaching two Shady3D robots together yields a single 6 degree of freedom robot. (Image from Rus [37].)

presented thus far: one of the Shady3D robots weighs 1.34 kg and is 250 mm long.

1.6.6 Inchworm

The Inchworm [24] (as shown in figure 1.8) is another design in the interest of minimalism. It is a 4 degree of freedom robot with magnetic feet that are used to attach to ferrous surfaces. In this instance, there are three degrees of freedom as parallel-axis rotary joints in the body of the inchworm. Above the rear foot, there exists a fourth degree of freedom: a rotary joint that allows the robot to change its orientation and direction of travel.

This robot is similar in size yet notably less massive than Shady3D, at a mass of 455 grams and a fully extended length of 252 mm. Each foot contains

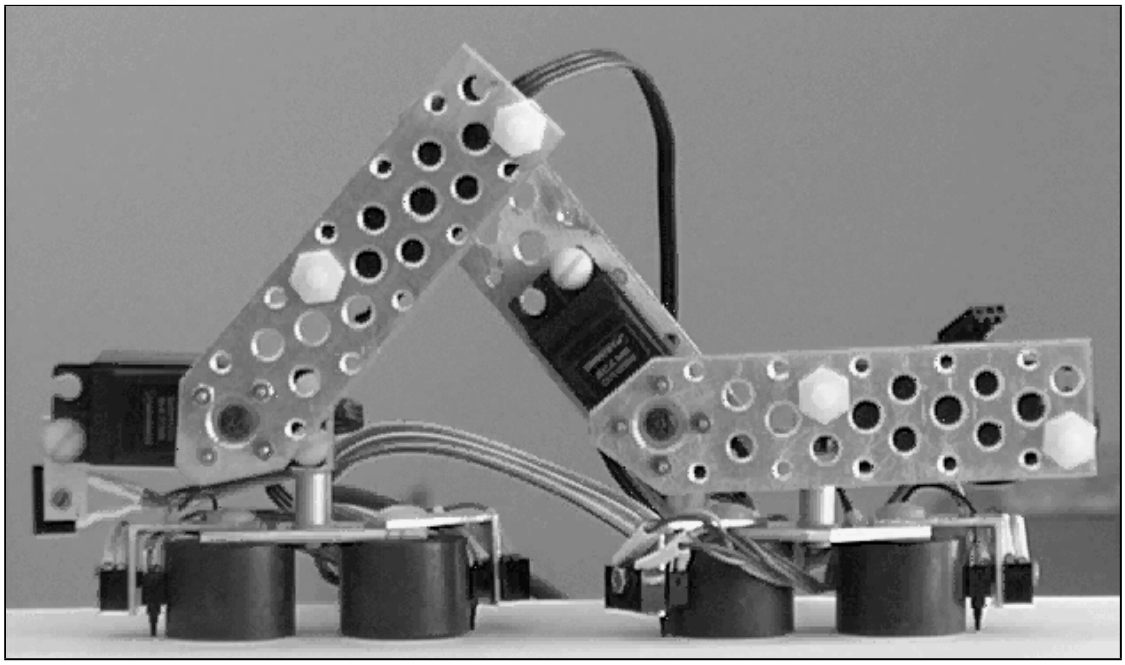

Figure 1.8 – *The Inchworm robot.* The small inchworm robot that attaches magnetically to ferrous surfaces. (Image from Kotay and Rus [24].)

two 25 mm electromagnets. It is capable of 0.25 meters per minute—one full extension/retraction cycle. Though capable of a variety of 3D motions, this robot cannot move horizontally on a vertical wall due to the limited magnetic force provided by the magnets: the robot will begin to pivot downwards during the walking motion.

1.6.7 SM^2

The SM^2 robot [33, 48], in two versions, is designed for navigating, inspecting, and maintaining the space station trusses. (To the author's knowledge, this system was declined in favor of the Canadarm2.) Two versions of SM^2 exist, corresponding to two truss designs. The first version [48] was designed to walk on the 5.00 m-spaced nodes of the space station truss structure (which is arranged

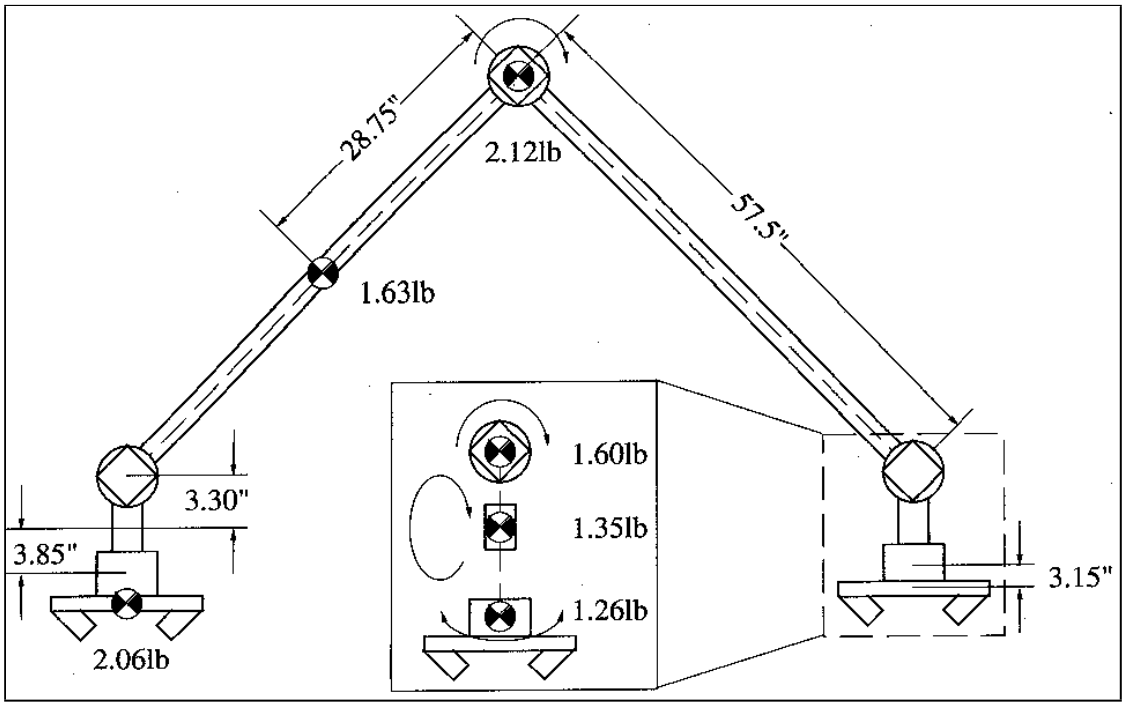

Figure 1.9 – *The SM*^{2} *robot.* A schematic of the SM^2 truss robot capable of navigating space station truss structures. (Image from Nechyba and Xu [33].)

in a regular rectangular fashion), with grippers attaching to threaded holes in the nodes. The second version [33] (shown in figure 1.9) is modified to traverse a hexagonal truss structure with I-beam members.

Apart from the degrees of freedom required for the grippers and the differences in gripping strategies, both designs are nearly identical with five degrees of freedom. Three degrees of freedom are in a four-link planar arm, with the two inner links much longer than the outer links. The final degrees of freedom are in the outer links, providing an axis of rotation axially to the link. The SM^2 walks, then, by sequentially rotating about these manipulator joints.

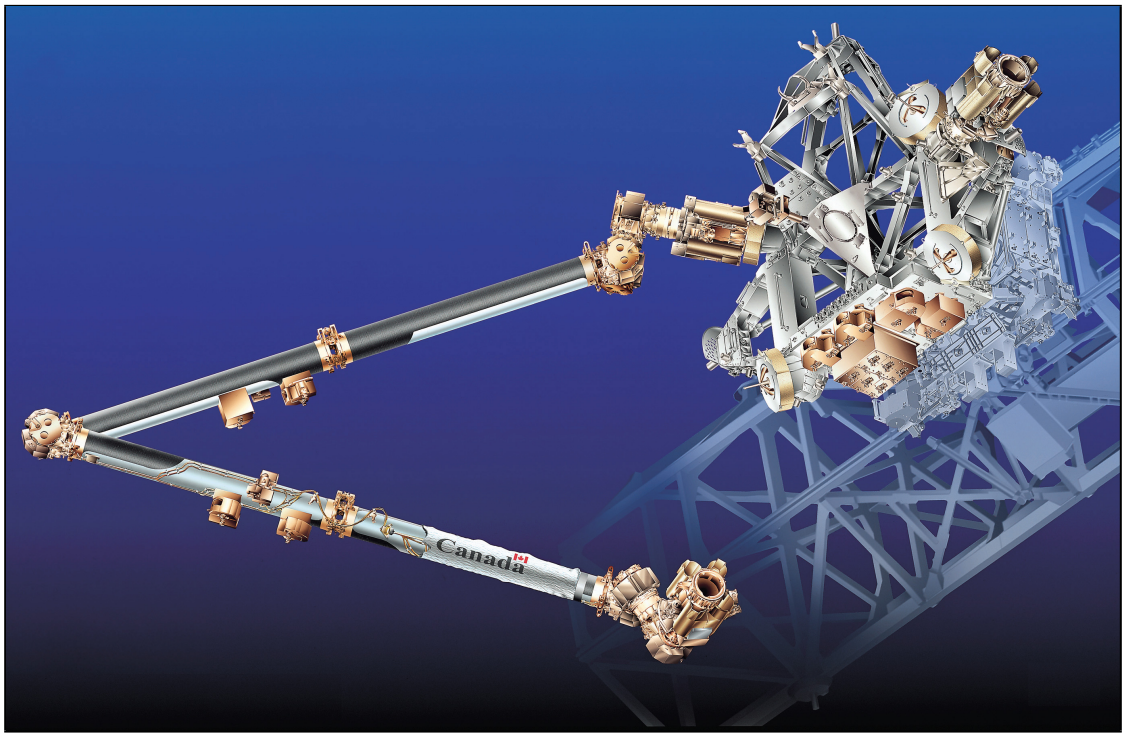

Figure 1.10 – *The Canadarm 2.* An artist's rendering of the Canadarm 2. (Image from King [22].)

1.6.8 Canadarm 2

As discussed by King [22], the Space Station Remote Manipulator System, or Canadarm 2, is a 7 degree of freedom robot very similar in configuration to SM^2. It has two arms, with a Latching End Effector (LEE) capable of attaching to the space station infrastructure on either end. Since neither end is permanently fastened to the space station, the Canadarm 2 can move around the space station using inchworm-style movements. It is used, among other things, for assembling and reconfiguring the space station. It is capable of handling payloads of up to 100,000 kg and is 15.2 m long.

1.7 Related Work in Reconfiguring Robots

Previous work has also been done on reconfiguring and assembling robots. The earliest work shows robots working with traditional construction materials, such as non-reconfigurable bricks [34] or sand for lunar bases [5]. These do not particularly meet the requirements of reconfigurable modular building blocks.

Other work shows robots reconfiguring non-structural passive elements, though not in any sort of structural sense. These works often focus on algorithmic questions, such as coordination between multiple robots while assembling a simple 2D linear structure [42] or using social behaviors to have a collection of robots move discs on a floor into a wall-like configuration [31]. These works all included physical trials of their ideas.

A variety of papers [43–46] by Werfel et al. use social behaviors to assemble a variety of solid structures beyond simple walls, and make use of passive elements that provide some structural stability via magnetic attachment.

Assembling trusses in space is an active area of research. Diftler et al. [11] show an anthropomorphic teleoperated robot for working with standard trusses in space. The system is not fully autonomous. Dubowsky and Boning [14] and Senda et al. [38] discuss algorithms for assembly of trusses in space, and showcase experiments assembling 2D trusses on an frictionless table. Robots with two arms and grippers that traverse on a low-friction table are used to manipulate the trusses, though Dubowsky and Boning [14] and Senda et al. [38] use similar but unique robotic manipulators. Everist et al. [16] uses a pair of robots tethered together for assembling triangle trusses on an air table. While these works show the ability to assemble truss structures, they focus on space structures (often

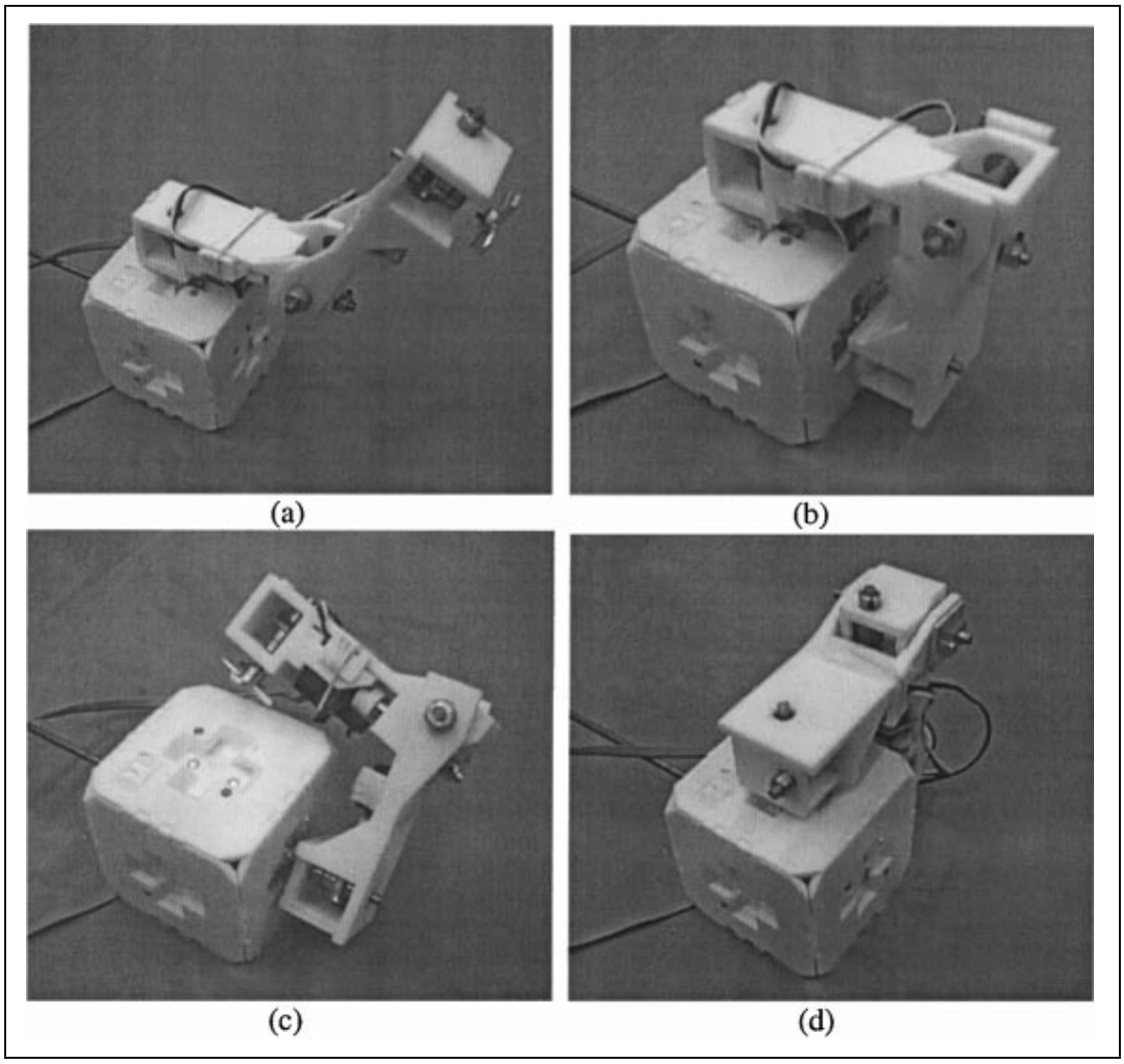

Figure 1.11 – *The I-Cubes robot.* Several frames of an active link robot moving over a passive cube. (Image from Ünsal et al. [54].)

simplified to 2D), and do not discuss reconfiguration properties vital to machine metabolism.

1.7.1 I-Cubes

Ünsal et al. [54] and Ünsal and Khosla [53] demonstrate I-Cubes, a system of 3 degree of freedom active "links," or assembler robots, and passive cubes. The

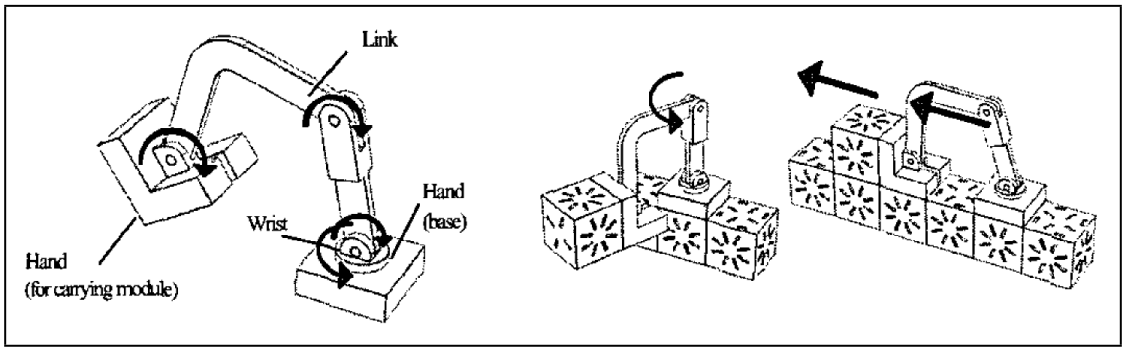

Figure 1.12 – *The module assembler robot.* The left shows a schematic of the robot and its degrees of freedom; on the right are several motion primitives. (Image from Terada and Murata [41].)

links have rotating connectors on either end, and pivot with a hinge in the middle. Thus, they can travel to visit the various faces of a single cube, or travel to other cubes. The connecting mechanisms also allow the active robots to reconfigure the blocks, though the precise mechanism of connection between blocks is not clear.

1.7.2 Module Assembler

The module assembler robot from Terada and Murata Terada and Murata [41] is a 4 degree of freedom robot built to reconfigure specialized passive cubic modules. It consists of two hands attached to a hinged joint in the middle. One of the hands tilts in relation to the body of the robot, and the other both tilts and rotates. The cubic modules can be actuated via the robot to fasten themselves to each other.

1.7.3 Robotic Assembly of Space Truss Structures

Doggett [12] uses a robot arm to assemble 3D triangular trusses. In this case, they were able to perform repeated autonomous construction and deconstruction of

102 truss members forming a 8m diameter structure. The robot arm was placed on a base capable of positioning the robot in the x and y planes, and the truss structure was built on a rotating base allowing access to all points on the outside of the truss structure. A specialized manipulator and truss elements were used, along with machine vision and other sensing, to accomplish this goal.

While these systems would be appropriate to use in a metabolistic process, they have one particular disadvantage as compared to the truss reconfiguring robot discussed in chapter 3: neither the cubic building blocks nor the truss elements manipulated by a robot arm are reconfigurable in random-access. (See section 2.1 for more discussion on random-access building blocks.) The robot may be able to access many elements in large structures, but—even with the truss structure—not all elements are available.

CHAPTER 2

BUILDING BLOCK DESIGN

This work was published in part in Hjelle and Lipson [19].

2.1 Requirements

- **Cheap.** We want our building blocks to be inexpensive, focusing cost on the robotic system. This also means that the part ought to be as simple as possible, for easy mass-production and for reduced cost.
- **Strong.** As expected, the building blocks need to be able to hold their own mass, the mass of a robot, and as much payload as possible (depending, of course, on the application).
- **Manipulatable.** A robotic manipulator does not often have very dextrous movement. For instance, human hands have five fingers with three or four degrees of freedom each. Only a very complex robotic manipulator would have this kind of dexterity.
- **Assemble-able.** Additionally, such dexterity is useless without the appropriate sensing and feedback. Minimizing the sensing required by judicious part design is advisable. This property is called "assemble-able."
- **Random access.** Finally, we want to enable our parts to be added or removed from any point in the structure at any given time—without affecting the rest of the structure. In other words, imagine a stick and node truss structure, as in figure 2.1. Inserting a strut into the truss requires deformation of the structure. This deformation is not necessarily possible, so we wish to avoid this situation as possible.

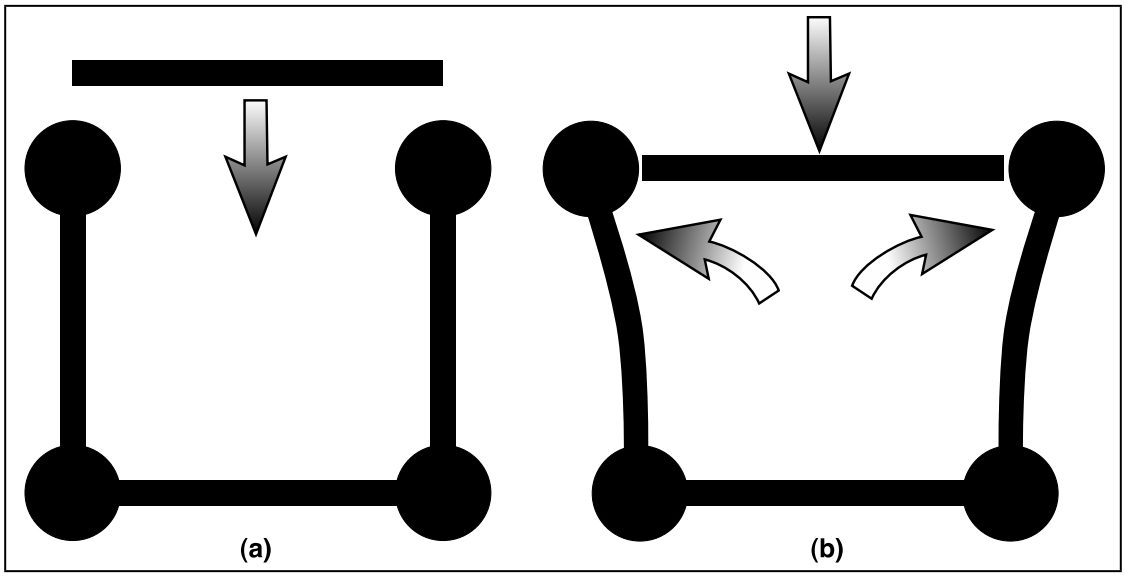

Figure 2.1 – *Random access truss elements are required.* Accessing (adding or removing) truss elements singly from arbitrary points on a cubic truss structure cannot easily be done. In order to insert a truss element into the structure in (a), the side truss elements must be distorted as in (b).

2.2 Potential Designs

Given these requirements, several designs were formulated and some prototyped. All were truss-like, as this design seems to permit a great amount of part accessibility, flexibility in end result, and strength, as well as potential inexpensiveness and manipulability. Additionally, all utilized threaded fasteners, as rotational motions are simple for robotic manipulators and threaded fasteners produce easy reliable connections.

2.2.1 Threaded Rod

One solution would be to simply use standard threaded rods. (See figure 2.2.) This, however, has the immediate disadvantage that the rod cannot be directly removed from between two nodes, nor can the connection be tightened or

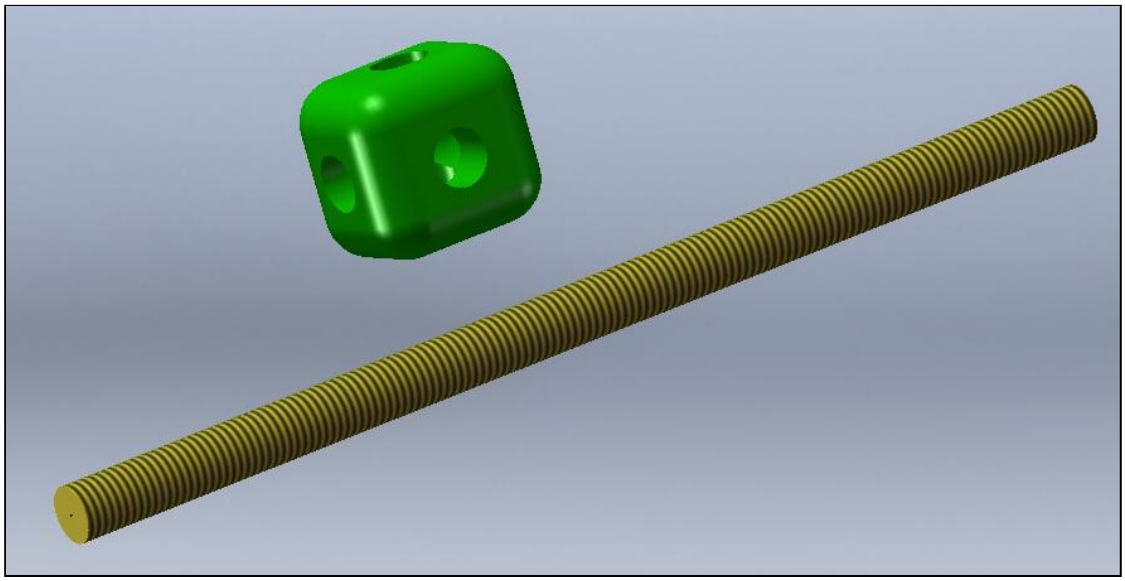

Figure 2.2 – *Simple threaded rods.* Simple threaded rods cannot be tightened or removed without removing at least one end.

loosened without manipulating a node.

2.2.2 Double-Threaded Hubs

Another solution would be to use rods that have a standard right-hand thread on one end and a left-handed thread on the other end. (See figure 2.3.) The nodes, then, have both left- and right-handed threads, allowing a turn of the rod to tighten or loosen the connection, allowing strut removal without the necessity of rotating the hubs. Accessibility is still limited, however, due to the required translation of the hubs along the axis of the strut that is necessary for removal. A similar idea for combining right-hand and left-hand threads is shown by Konzorr [23], though, in that instance, the threads were combined on the male portion of the fastener instead of the female portion.

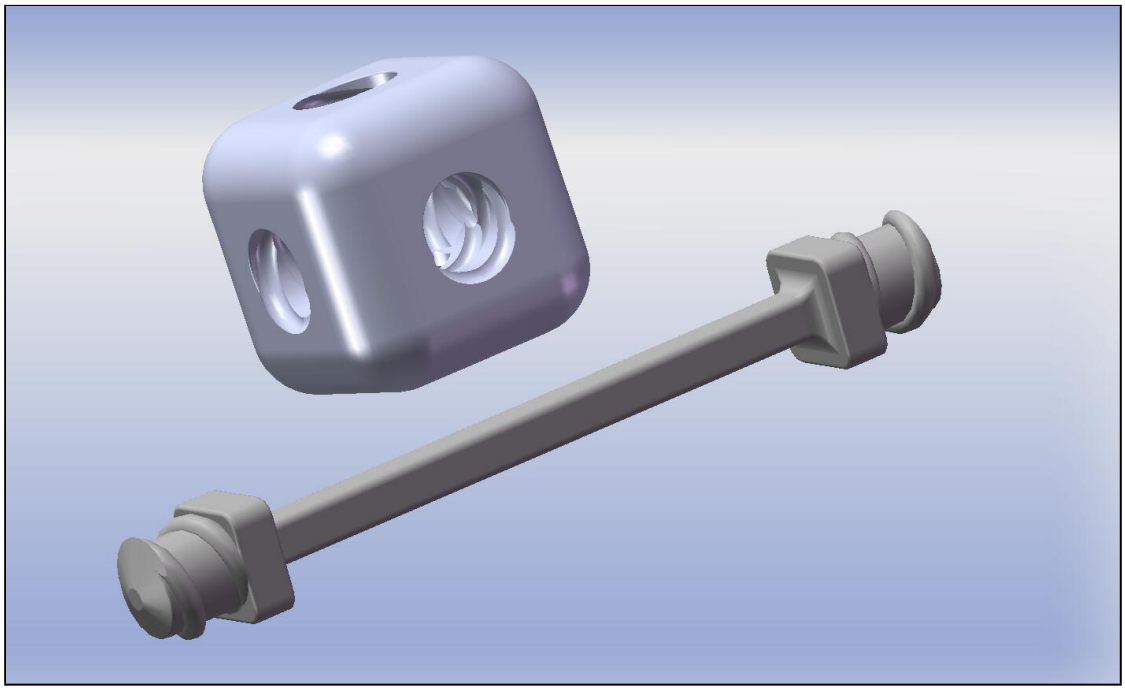

Figure 2.3 – *Double-threaded hubs with opposing-thread rods.* Double-threaded hubs allow assembling without rotation of the nodes, but still require translation of the nodes for removal.

2.2.3 Threaded End Caps

Using a rod with end caps that translate along the length of the rod and rotate about the same axis (see figure 2.4) allow the strut to be assembled and disassembled at will, in random access. Assembly of such rods, however, requires activation on both ends from a robotic system, which complicates the assembly process.

2.2.4 Telescoping Tubes

Another solution that addresses the random access problem involves a pair of concentric telescoping tubes. (See figure 2.5.) Fastening the tubes internally with a threaded fastener whose pitch matches the external threaded fasteners allows the strut's length to adapt during insertion. (For further discussion of

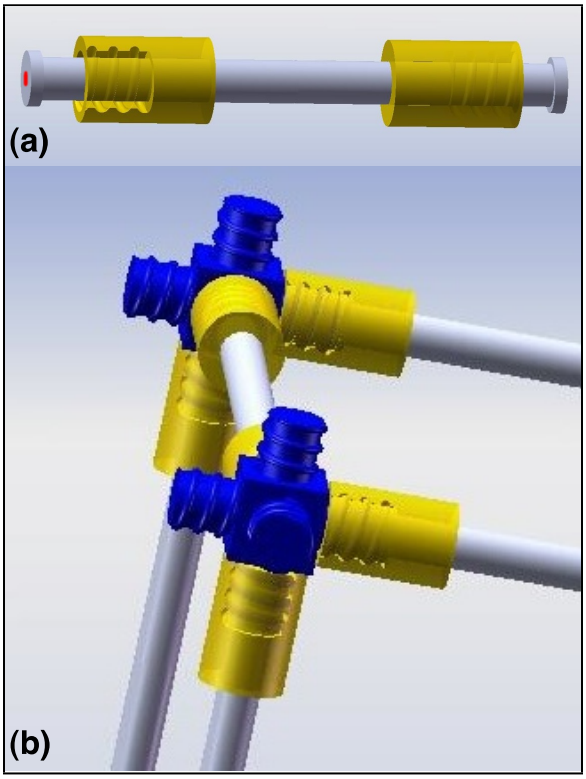

Figure 2.4 – *Threaded end capped modular elements.* Threaded end caps allow assembly and disassembly without node movement, but require activation of the end caps on either end.

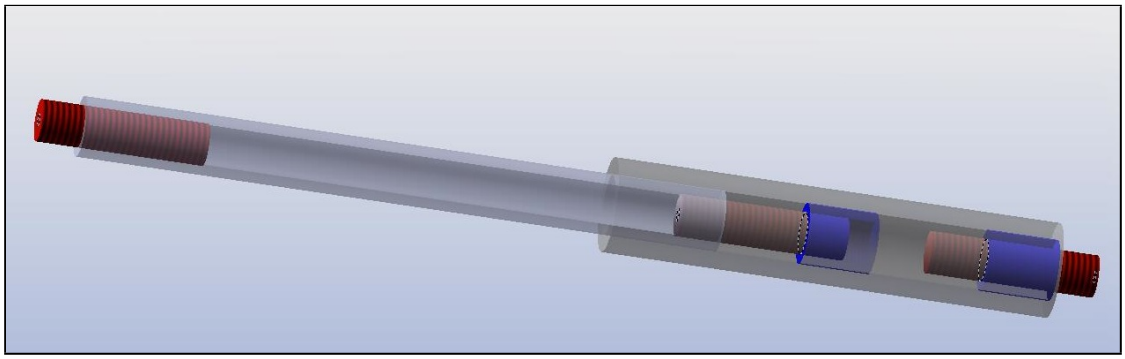

Figure 2.5 – *Concentric telescoping tubes.* Concentric telescoping tubes allow random access, and also potentially allow activation from two nearby points near the mating point.

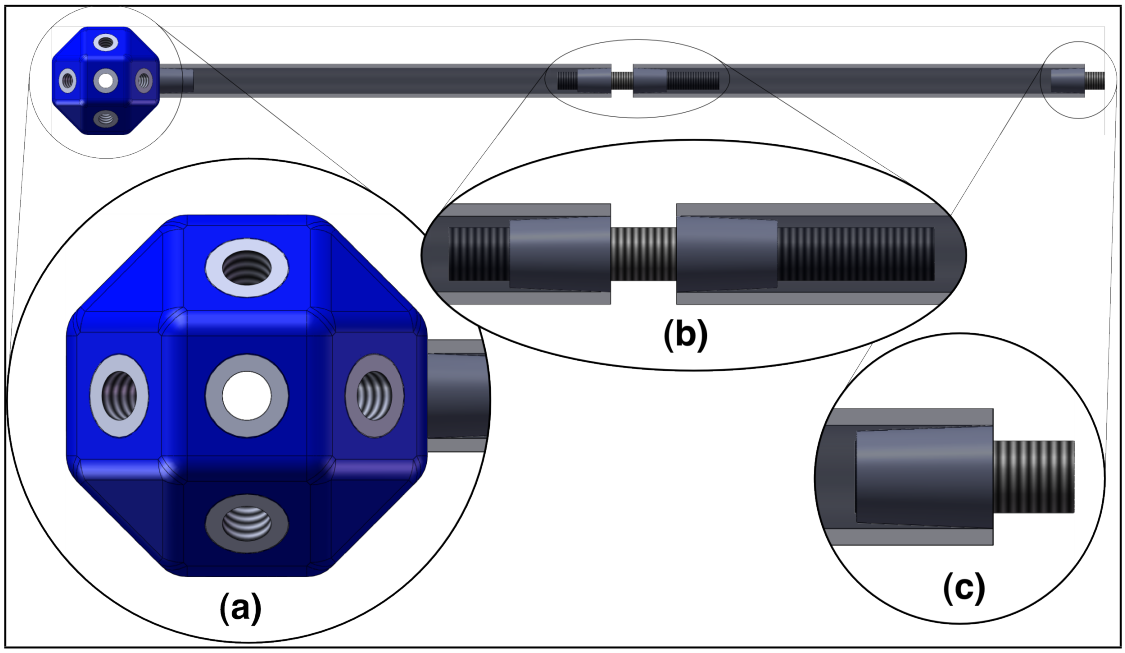

Figure 2.6 – *Center-threaded strut element detail.* Elements consist of an 18-axis threaded node (a) and a long strut. This strut is split in the center with a threading mechanism (b) and uses a threading mechanism on either side to attach to the node (c).

such auto-adaptive strut lengths, see section 2.2.5.)

2.2.5 Center-Threaded Rods

A variation on the "telescoping tubes" theme in section 2.2.4 is to remove the telescoping feature, leaving two shafts with a threaded stud on either end and a threaded connection in the center. (See figure 2.6.) Thus, if the element is place between a pair of receptacles, a simple twisting motion can be used to to fasten the strut into both nodes. The nodes do not require rotation or translation in order for a strut to be inserted or removed.

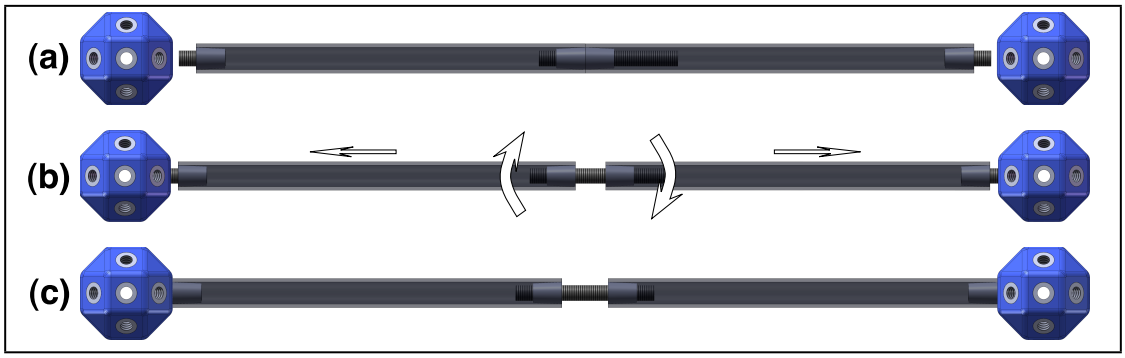

Figure 2.7 – *Center-threaded strut element insertion process.* The truss elements can be accessed in a random-access manner by inserting the contracted element between nodes (a), twisting the two element halves (b) causing the center thread to lengthen and the end threads to enter the nodes (c). Note that all threads are right-handed threads.

2.3 Implemented Truss Element Building Blocks

The center-threaded rods discussed in section 2.2.5 were chosen as the basis for the implemented rods. The design is simple and could be produced inexpensively. Built with the proper materials, they have the capacity to be quite strong. They do not require the extensive manipulation capabilities of a human truss solution such as extruded aluminum, and can be successfully assembled with minimal alignment requirements (discussed further momentarily). They also can be manipulated simply, whether at a single location near the center connection or once at each end. Finally, they can be accessed, removed, and inserted in a random access manner.

One of the fundamental problems with robotic manipulation of threaded fasteners is proper axial alignment between the male and female portions of the fastener. Work has been done on this using force and position feedback from robotic hands [9] or by "backspinning" the bolt on the nut to determine proper orientation[10]. These processes require a relatively large amount of

Figure 2.8 – *Carbon fiber truss elements.* The first implemented truss center-threaded truss elements, made from carbon fiber rods.

feedback and control to perform correctly. Several patents [4, 32, 39] list possible fastener modifications that serve to enable self-aligning. Additionally, several commercial anti-cross threading fasteners exist, such as Research Engineering & Manufacturing's Tru-Start, Camcar's Acupoint, ITW Shakeproof's Align-Rite, and MAThread from http://www.mathread.com. These options have not yet been implemented in our physical system.

Next, the various implementations of such center-threaded trusses that have been physically implemented will be discussed.

2.3.1 Carbon Fiber Rods

The first implementation of this idea was developed with carbon fiber struts, 8-32 threaded fasteners, and 3D printed nodes with 18 sockets to allow all fastening axes necessary for creating simple cubic trusses with single face diagonals. (See an example of one constructed in figure 2.8.)

The horizontal and vertical strut elements were 9 inches long, while the diagonals were $12\frac{3}{4}$ inches long. The threaded insertions on either end were approximately $\frac{3}{16}$ inches in length, with the internal thread ranging from 0 to

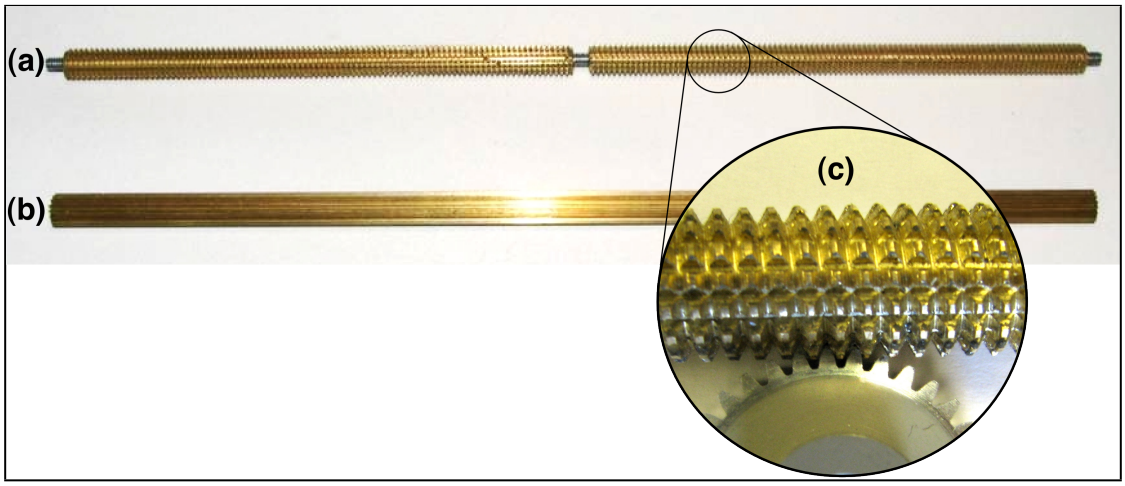

Figure 2.9 – *Threaded pinion wire.* We applied 3/8-16 threads to a 48 DP pinion wire to obtain a rod that meshed with standard 48 DP gears in both translation and rotation.

3/8 inches. The carbon fiber rods had an internal diameter of 1/4 inches and an outer diameter of 3/8 inches. Thermal inserts were used as the female part of the threaded fasteners in both the 3D printed nodes and in the carbon fiber rods. In the nodes, the inserts were thermally inserted into the 3D printed ABS plastic; in the struts, the inserts were fastened using an epoxy.

2.3.2 Threaded Pinion Wire

During development, it was discovered that friction would not transfer enough motive force between the robot and the struts to allow all the desired robot movements. It was decided to implement gear theory into the truss elements in order to provide essentially an ideal friction coefficient.

The primary problem with standard gearing, however, is that it is uni-directional. In other words, a rack and pinion would provide the ability to move laterally along a single axis, but would not assist in the ability to rotate about

that axis. Conversely, pinion wire (a long extruded spur gear) would provide the ability to rotate about its central axis, but not move laterally about the same axis.

Thus, we desired a gear form that would allow both motions. Revolving a standard rack gearform about the axis while simultaneously extruding a spur gearform along the axis would work, though this was not amenable to straightforward manufacturing in our lab. Instead, we determined that 48-pitch gears have nearly the same pitch as 16 thread-per-inch threaded fasteners. Thus, we could tap the outside of a 16 tooth pinion wire rod (an extruded spur gear) and obtain a rod with the properties we require. In figure 2.9, frame (b) shows a pre-fabrication pinion wire strut. Frame (a) shows a strut after the threading has been applied and the threaded fasteners have been inserted. Finally, frame (c) shows the detail of the mesh with a 48 pitch gear.

It is important to note that the mesh between the threaded parts and the gears is not perfect. The threads are 16 teeth per inch, while the gear teeth are $48/\pi = 15.28$ threads per inch. This is close enough for our purposes, but clearly violates gearing principles such as the Fundamental Law of Gearing.

These threaded pinion wire truss elements were the ones finally implemented in the design discussed here.

2.3.3 Printed Struts

A similar solution to that presented in section 2.3.2 is to use 3D printing to produce the two degree of freedom gears. This was originally not an acceptable solution, as the Stratasys Dimension SST 3D printer's 0.01 inch resolution was

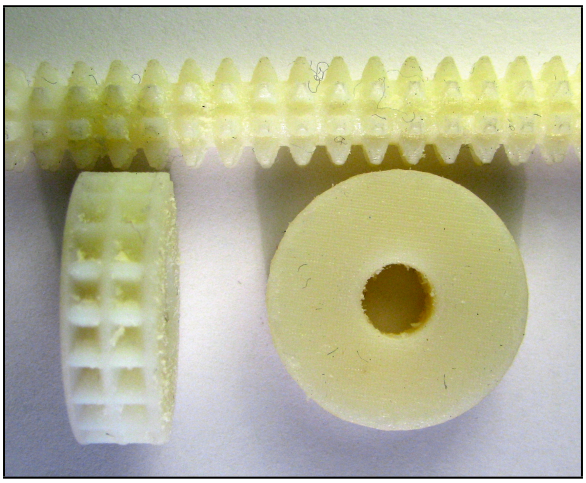

Figure 2.10 – *Geared truss elements and gears from a 3D printer.* The gears can either translate across or rotate about a truss element, while preventing motion in the other direction.

not fine enough to create reasonable gear teeth. Once our lab obtained an Objet Eden 3D printer, the resolution increased to 0.004 inch. Experimentation showed that this resolution was fine enough to properly print even 48 pitch gears. (See an photo of 27 pitch gears and truss element in figure 2.10.)

The fundamental problem, however, was the weakness of the material. Since the two-dimensional gears yielded short "spikes" on the struts, the material cross section at the base of each tooth was reduced to approximately 1 mm^2. This was sufficient to support translational motion, but not sufficient to support cantilevered rotational motion. Larger gear teeth, such as custom 27 pitch gears, showed some promise, and this is a future direction that we wish to explore further due to its flexibility.

CHAPTER 3

ROBOT DESIGN

The next fundamental problem in producing a system capable of the autonomous reconfiguration we envision in machine metabolism is the production of a robot that can traverse and reconfigure truss structures such as those described in chapter 2. Section 1.6 discusses several previous truss and other climbing robots that have potentially applicable morphologies. Here, the morphologies that present the best possibilities for machine metabolism will be discussed, and followed by the design iterations performed.

3.1 Potential Morphologies

Many morphologies are possible to obtain a robot capable of the truss reconfiguration that has been presented. These will be discussed briefly before discussing the morphology chosen for the robot.

3.1.1 Robot Arm

The most obvious potential morphology for robotic reconfiguration would be a freestanding robotic arm, as often used in manufacturing situations. While this would allow simple reconfigurations and require the least development, there are two primary reasons why this is not a acceptable solution.

First, the workspace of such a robot is limited by the size of the robotic arm. While valid and interesting reconfigurations could be done, this limits not

only our experimental range but also further practical applications of the idea. Conversely, a robot capable of traversing and reconfiguring a truss structure could scale without limit. ("Without limit" in terms of mechanical workspace; other considerations, such as time required, off-board control, and limited battery power are not considered. Collaborations of multiple robots could be one potential solution to these issues. This is not explored in this work.)

Secondarily, in addition to the workspace imposed by the lengths of the robot arms, there are workspace constraints imposed by the configurations necessary to perform reconfiguration of an entire truss structure. As stated previously, we desire to be able to insert or remove any strut in an arbitrary truss structure. In a truss structure whose outside geometry is within the boundaries of the robot arm's workspace, guaranteeing access from the outside of the truss structure to a arbitrary element within a potentially complex truss structure is non-trivial. Thus, it would be advantageous to develop a manipulator that did not depend on the actual structure of the truss.

3.1.2 Node Walker

SM^2 exemplifies a morphology of robot that solve many of the issues with a standard robot arm. As described in section 1.6.7, SM^2 is capable of "swinging" from one location on a truss to another. It has two end effectors connected by 5 degrees of freedom. With one end gripping on one section of the truss structure, the remainder of the robot can maneuver the other end effector to another point on the truss structure. This movement could allow a similar morphology of robot to traverse a truss structure completely, and could be simplified for our

regular truss structure by only allowing the end effectors to attach to the nodes of the truss structure, which would be regularly spaced. A third manipulator tool would then be required to manipulate the truss structure edges.

While this general morphology seems promising, effective implementation in a small robot under earth's gravitational forces seems difficult. The situation is compounded when a truss element needs to be transported from one location in the truss and to another. Finally, actuating such a robot is difficult: placing actuators of adequate torque and small size complicated the simple stick morphology quickly. Thus, a hinge robot was decided upon, as discussed in section 3.2.

3.2 Hinge Robot

The fundamental building block of the hinge robot is not a hinge, but is rather a module that is capable of attaching itself to a truss element, translating about a truss element, and rotating about a truss element. Such modules have two degrees of freedom, though the necessary engagement/disengagement mechanisms combined with the drive mechanisms use a total of four actuators in our design. (Future work could potentially utilize the symmetrical movement criterion discussed by Balaguer et al. [3] to reduce the number of actuators.) The rotational actuation is used for both the truss traversal by the robot and for the insertion and removal of truss elements.

Such a module can clearly not completely traverse a truss. Combining two such modules with a hinge, however, allows greatly increased mobility. The hinge would be placed such that two modules could be arranged directly in series (with the same orientation about the truss element), or the angle between the

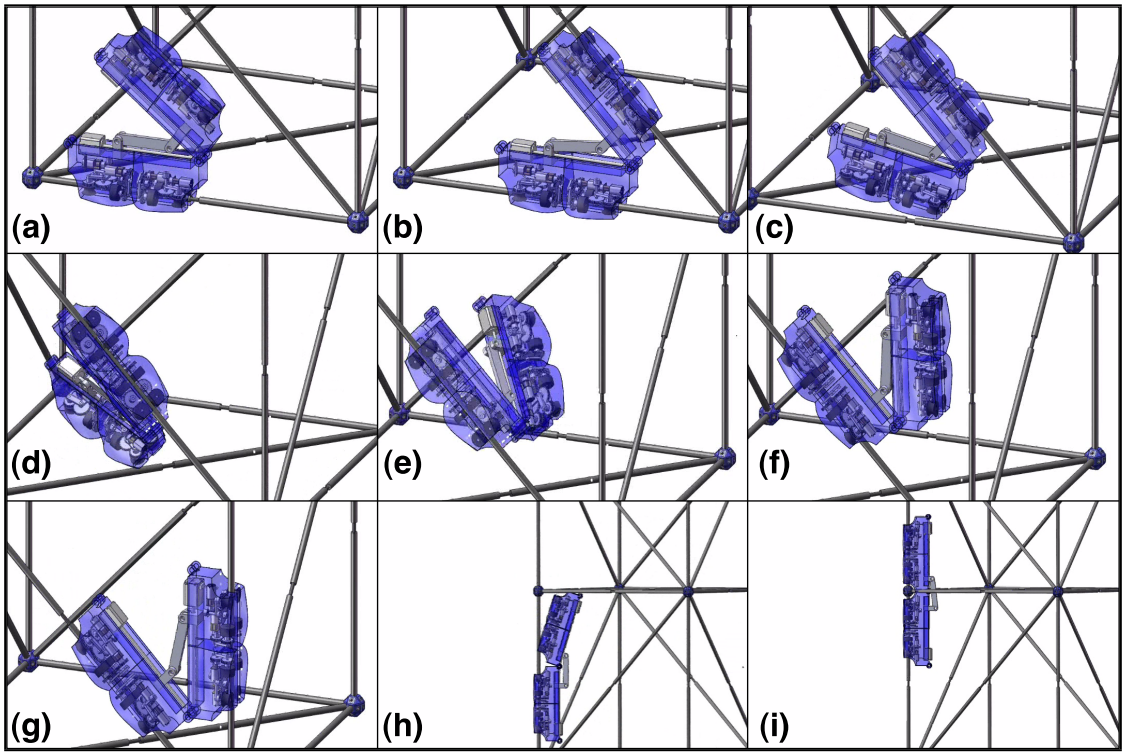

Figure 3.1 – *The proposed hinge design on a truss.* Frames (a) through (g) show a sequence of frames of a hinge robot concept moving in the way just described. The robot starts in (a) and moves rightward until the second module engages the diagonal element above (b). The robot can disengage the lower strut. Then, the robot can rotate itself about the diagonal element ((c) to (f)) until the robot is aligned with the next truss element, in this case vertical. Finally, the robot can attach to the vertical truss element (g). Another movement is described in frames (h) to (i). In this case, the robot traverses a node that lies between two vertical struts.

respective top surfaces of the module varied up to 135° from that normal position. Thus, in a cubic truss structure, the combined robot disparate but intersecting truss elements at the same time. This ability allows the robot to transfer itself from one element to another, and thus traverse a truss.

This is better explained in figure 3.1. Frames (a) through (g) show a sequence of frames of a hinge robot concept moving in the way just described. The robot starts in (a) and moves rightward until the second module engages the diagonal element above (b). The robot can disengage the lower strut. Then, the robot can rotate itself about the diagonal element ((c) to (f)) until the robot is aligned with the next truss element, in this case vertical. Finally, the robot can attach to the vertical truss element (g).

Another movement is described in frames (h) to (i). In this case, the robot traverses a node that lies between two vertical struts. Due to the gap between the modules, the node can be traversed properly.

Finally, if a holding pod is designed to hold a truss element in a secondary, non-actuated position in such a way that the robot can operate normally on truss structures, complete truss reconfiguration is possible. A sequence of holding a truss element, transferring it to the active track, and inserting the truss element is shown in figure 3.2.

This is the design morphology chosen; the various design iterations explored in this process will now be discussed. The requirements, then, for each module are to be able to translation and rotate about the truss elements. In order to perform this actuation, it was decided to have drive wheels, a single or a set for each of the two degrees of freedom. This mechanism also requires actuation to

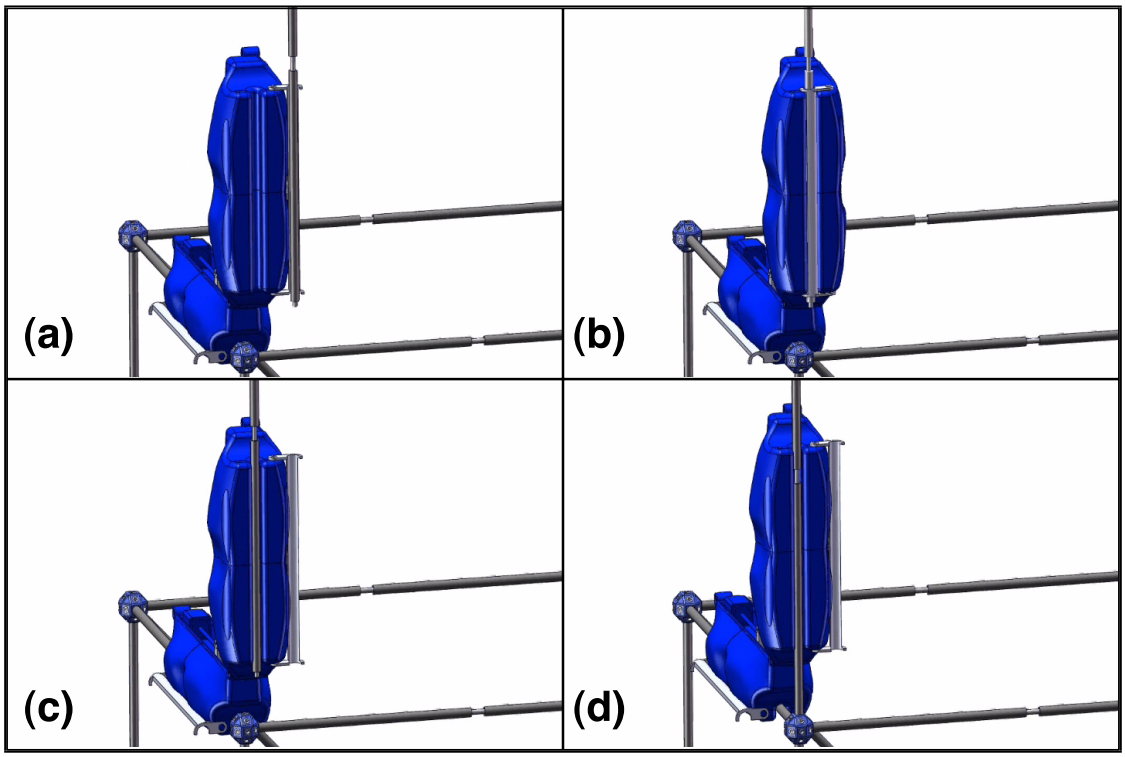

Figure 3.2 – *The proposed truss element holding pod.* This sequence of images shows the holding pod holding a passive truss element (frame (a)), transferring the truss element to the active track (frame (b)), retracting the holding pod (frame (c)), and inserting the truss element (frame (d)).

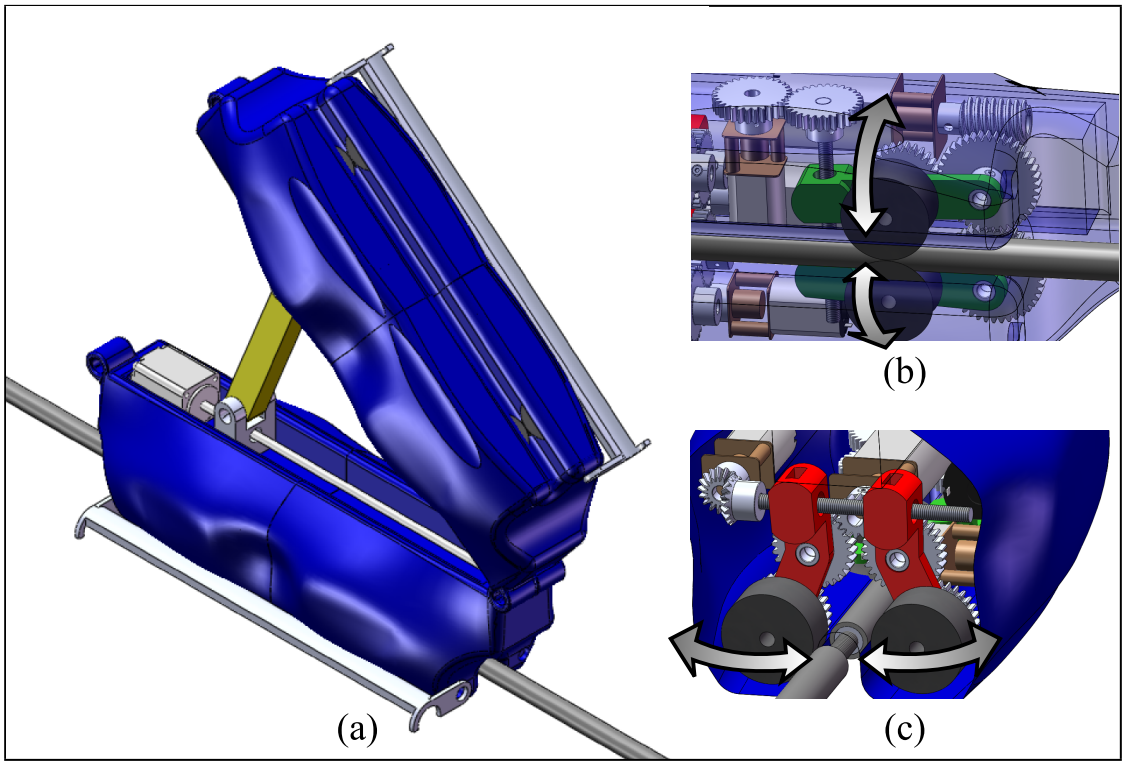

Figure 3.3 – *An internally geared hinge robot.* Frame (a) shows an overview of this design of a hinged robot. The translational mechanism, and its method of engaging and disengaging, are shown in frame (b). Frame (c) shows the same for the rotational mechanism.

engage or disengage these drive wheels. The various ways in which this was accomplished are discussed in the following sections.

3.3 Geared Design

The first conceptual design used gearmotors for actuation, friction wheels to transfer force between the motors and the struts, gearsets to transfer power between the locations of the motors and the drive wheels, pairs of drive wheels that would "pinch" the strut, and gearmotors and screws that provide for engagement and disengagement.

Figure 3.3 shows some renderings of this design. Frame (a) shows the overall hinged mechanism design. In this case, two modules were used on either side of the robot. This was for multiple purposes. First, the stability of the robot on the truss was improved by having multiple points of attachment. Second, it was desired to be able to twist the opposite ends of a strut in opposite directions with a single side. The hinge is located in the lower right of frame (a). The hinge is actuated by a linear actuator on either side. A solid link connects the actuated nuts on either linear actuator, allowing adjustment of the angle between the two sides from 35° to 180° (where 0° is where the two modules are "back to back" on top of each other). Finally, the long grey "sidecar" attachment on the side of each module is a carrying pod. It would be able to swing into the active truss element area, remove a truss element from it, and carry it passively to allow the robot to carry a truss element while moving about the truss. (Snapshots of this process are shown in figure 3.2.)

Continuing in figure 3.3, frame (b) shows the bottom of one of the modules, showing a transparent view of the translational mechanism. The arrows show the movement of the drive wheels, shown in black, between the engaged and disengaged positions. The drive wheels are "cupped" to better encircle the truss element. They are actuated via a geartrain from a single gearmotor. The green links pivot at their point of attachment to the gears on the right. This pivoting is actuated by a threaded stud with opposite-hand threads on either side so that one direction of rotation clamps the links towards the truss element (thus engaging the drive wheels) and the other direction releases the clamp. This engagement/disengagement mechanism is also actuated via a gearmotor.

Finally, in frame (c), the rotational mechanism is presented. The engagement

and disengagement mechanism is very similar to that of the translational mechanism discussed previously and is actuated in an identical fashion. The drive wheels are again actuated from a single gearmotor, and are situated so as to grip the truss element during rotation. The wheels are flat to match the profile of the truss elements. In this case, the red links pivot in their centers.

This design—particularly in its extravagant use of gears—was discarded due to its complexity and difficulty to properly construct. It was not constructed.

3.4 Four Module Friction Drive

This work was published in part in Hjelle and Lipson [19].

Due to the potential complexity of the geared design discussed above, a new design was developed to attempt to address some of these issues. This design still uses the basic principle of modules that can rotate about and translation across truss elements, and uses a series of four modules hinged in the middle. Actuation of the hinge has been replaced with a dual servo actuation. (Discussed in more detail shortly.) Drive wheels are kept, but are now singletons: only one wheel per module is actuated to provide the motive force for both translation and rotation. This has allowed the gears to be completely removed from the system. For simplicity of control, the gear motors have been replaced with standard hobby servos. (Specifications of these are found in section 3.7.) The engagement/disengagement mechanisms have been changed from the screw actuation from a gear motor to be actuated with a cam actuated by a servo. The carrying pod is the same as the previous design.

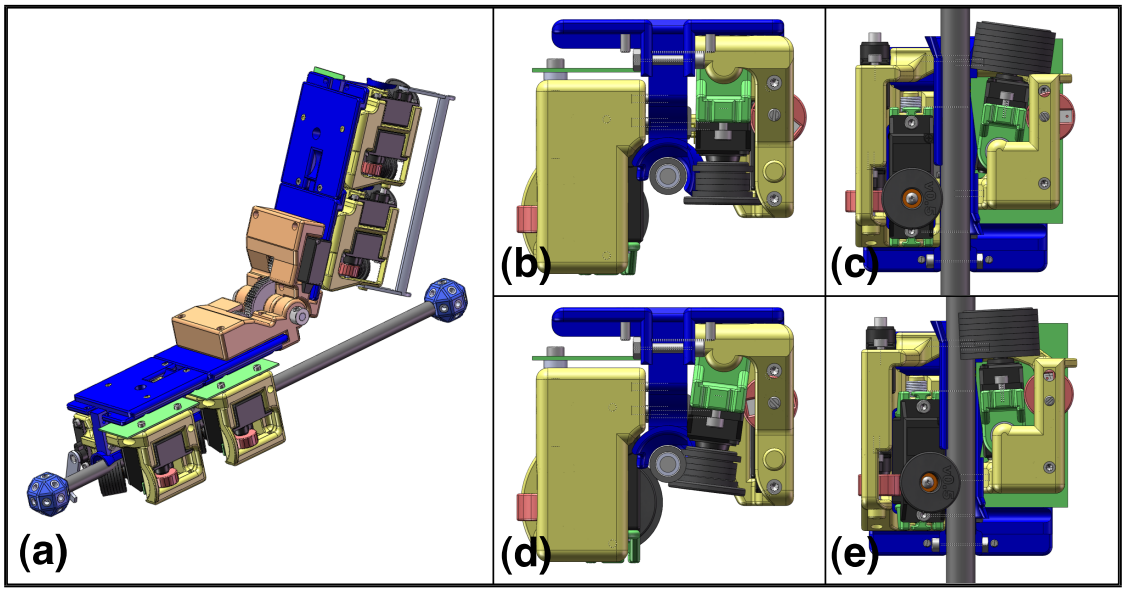

Figure 3.4 – *The four-module friction drive design.* Frame (a) shows a general overview of this robot design. The translational mechanism is shown in frames (b) and (d) as engaged and disengaged, respectively. The rotation mechanism is shown in frames (c) and (e), as engaged and disengaged, respectively.

This design is shown rendered in figure 3.4. Frame (a) shows an overview of the entire robot assembly. The initial linear actuators turned out to consume significantly more current than anticipated, so they were replaced with a mechanism that utilized a pair of servos mating with a central gear. (See figure 3.5.) The servo-attached gears serve as planetary gears to a sun gear of twice the diameter, situated so that its axis would be directly on the axis of rotation of the hinge. This increased the torque of the servos 2× while reducing their produced range of motion by 1/2 to 90°. Using two servos allows the full range of motion required.

The more important changes are shown in frames (b) to (e).

First, the translational mechanism, as shown in frames (b) and (d). Frame (b) shows the mechanism engaged, while frame (d) shows the mechanism disengaged. The servo can pivot in the top right corner, supported by the green hinge joints. This pivot action is actually accomplished via a cam (the pink

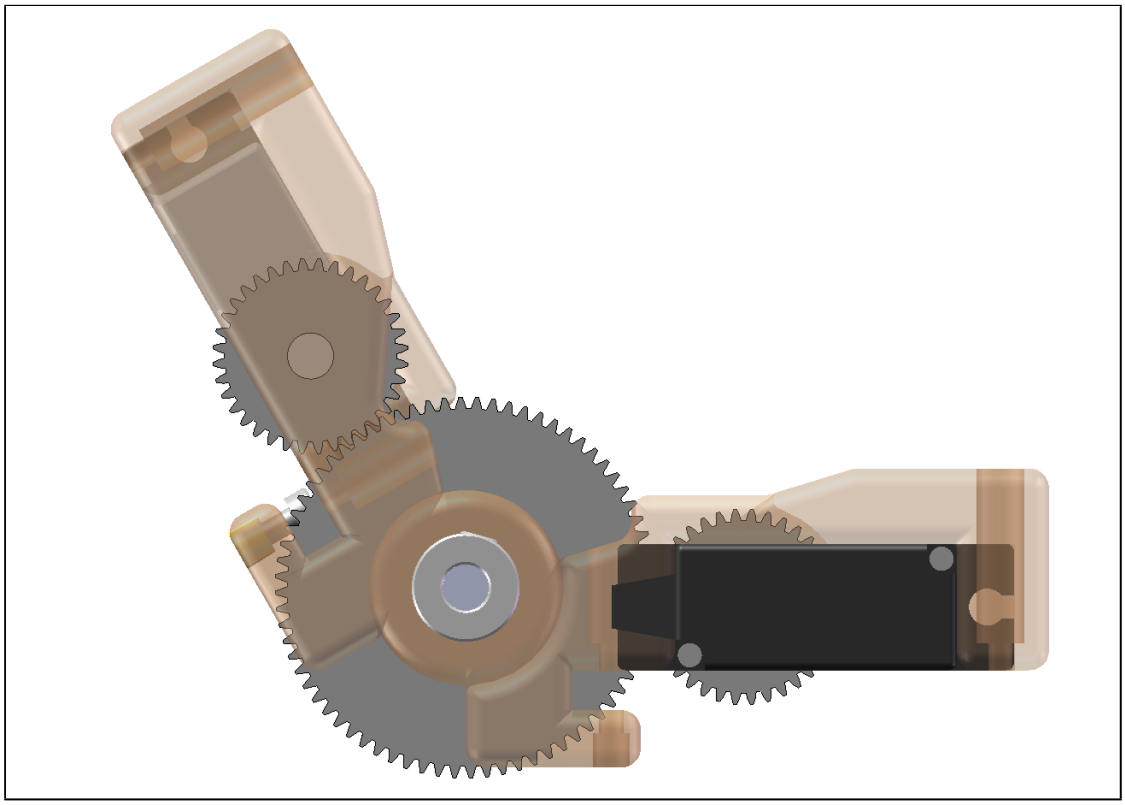

Figure 3.5 – *Planetary gear hinge design.* A central sun gear, which is free to rotate, is surrounded by two planetary gears, each fixed to the output of a servo.

circle in the far right) attached to a smaller servo. In this diagram, the cams are simply eccentric circles, though the profile can be adjusted to obtain the necessary mechanical advantage. The single drive wheel is 3D printed and coated with a friction-enhancing material.

Many materials were tested, including various durometer ratings of polyurethane, friction enhancing tapes, epoxy grit sprays, and rubber dip coats. The most effective method was using rubber Buna-N square o-rings, as depicted in the renderings. A profiled drive wheel was used to grip the truss element, providing a hanging action in addition to the translational motion. Since the drive mechanism only engages the truss element from one side, there is an central track (the blue surrounding the truss element) that keeps the strut in place during engagement and disengagement.

The drive wheels were printed via our 3D printer with a slot within which the standard servo horn (often trimmed in length) could be inserted.

The rotational mechanism, shown in frames (c) and (e) is similar in form. Since the rotational movement requires a different axis of drive wheel, however, the pivot axis is now perpendicular to the axis of the truss element and the drive axis is parallel to the axis of the truss element. The drive wheel tilts in to engage the truss element. This is, again, actuated by a cam. The drive wheel is situated in such a way to capture the truss element in the track below, fastening the module to the truss element.

The modules are connected to each other either via the hinge or via a mating connector to connect two modules together in series. All brackets (yellow, blue, and green in the renderings) are created using the Stratasys Dimension SST 3D

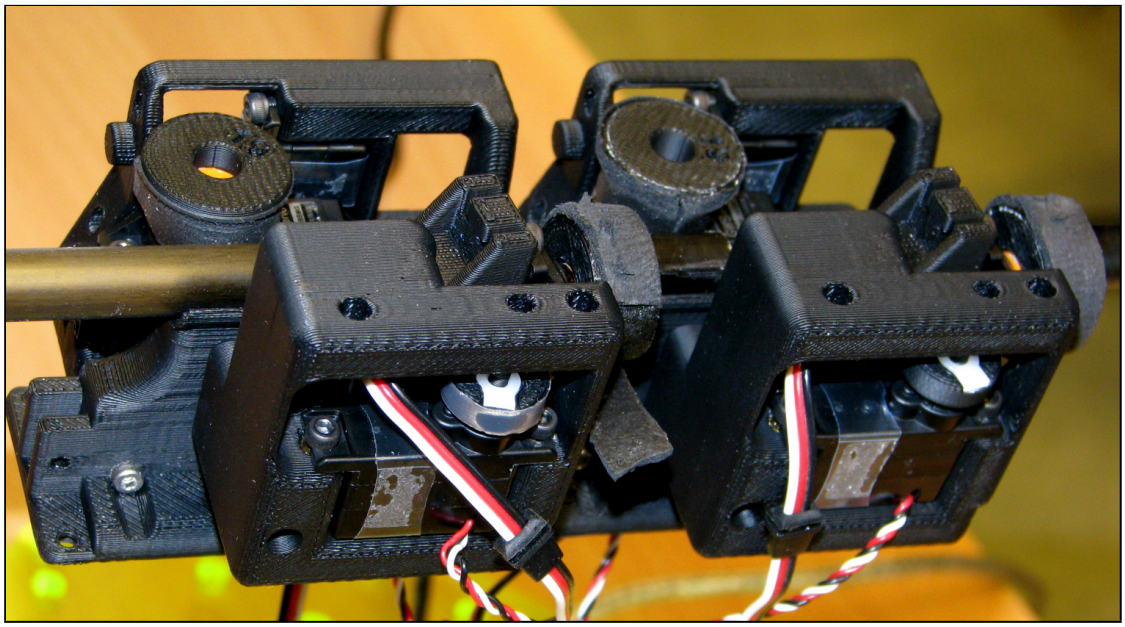

Figure 3.6 – *Two constructed modules of the four-module friction drive design.* A foam grip tape is shown on the drive wheels.

printer out of ABS plastic.

While this design was not completely constructed, several modules were constructed. These are shown in figure 3.6. As discussed earlier, several variations of drive material were tried; additionally, tweaking of drive wheel contours and cam contours was performed. This design was capable of moving about a truss element, but rotating even the two modules shown in figure 3.6 against gravity proved difficult. The design was simply too bulky, and it was found that relying on simple surface-to-surface friction alone did not produce a reliable mode of movement.

In the process of developing the rotational motion, replacing the friction drive wheel with a friction drive belt system was attempted, pictured in figure 3.12. This allowed a small drive wheel (to improve mechanical advantage) and increased the contact area between the drive belt and the carbon fiber rods significantly.

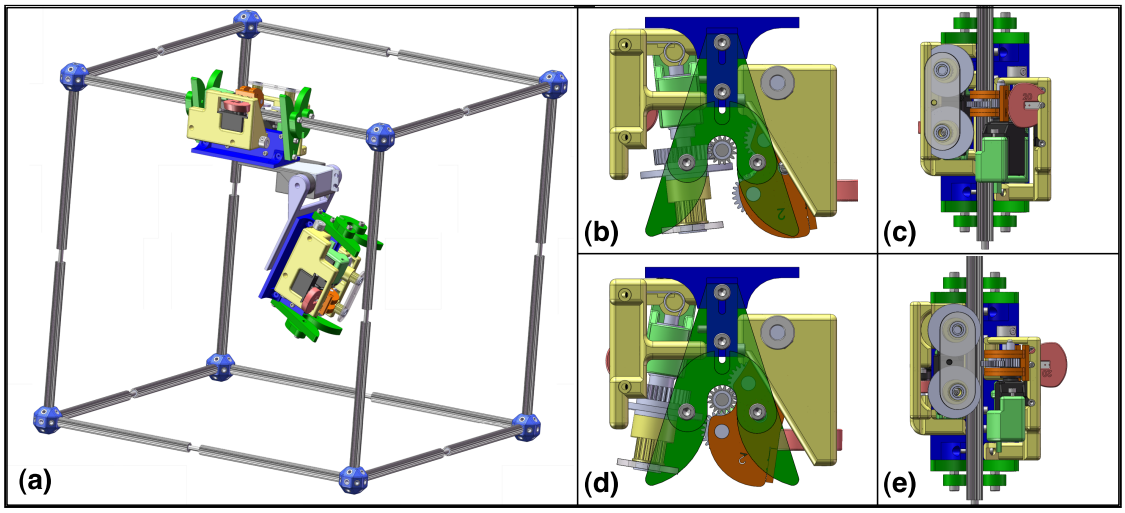

Figure 3.7 – *3D printed gear drive hinge robot concept.* Frame (a) shows an overview of this design. Frame (b) shows a side view of the translational mechanism engaged, while frame (d) shows a side view of the rotational mechanism engaged. Frames (c) and (e), respectively, show the top views of the same engagements.

3.5 3D Printed Gear Drive

In order to address the shortcomings discussed in the previous section, substantial changes to the mechanism were made. The most consequential was to replace the 3D printed, friction-based drive wheels with standard 48 pitch gears. This solved the problem of friction. (See section 2.3.2 for more discussion of the truss element gears.) To further simplify the design, the module count was reduced from four to two, with one module on either side of the hinge. Since single modules were less massive than the dual module construction, the pair of servos used for the hinge could be replaced with a single-servo direct-drive hinge. Renderings of this new design are shown in figure 3.7 and a photo of the constructed robot is shown in figure 3.9.

(Note: Franz Nigl was heavily involved in the design and construction work for the robot from this point on. Stephane Constantin did the design of the PCB and supplied other electrical advice.)

Modifications to the drive mechanisms were also done.

First, the reduction of the number of modules meant that the translational motion now only had a single drive wheel per side in contact with the truss element. This did not allow the robot to bridge the center gap in the center-threaded truss elements. Thus, a second drive gear—driven by the single servo—was powered via a timing belt. This allowed at least one drive gear to be in contact with the rods at all times. Additionally, since the drive gears were not printed and could not be contoured in order to capture the rod, discs of a larger diameter were placed on top of the drive gears in order to keep the robot attached to the rod. This can be seen in frames (c) and (e) of figure 3.9.

Second, since changing to only a single module reduced the contact points for the rotational mechanism from two to one, the rotational mechanism was modified to pivot on an axis parallel with the truss element. This allowed a secondary "hook" on the non-drive-wheel end of the servo to capture the rod and ensure the robot's stability. Additionally, the gear ratio between the truss elements and the servo depends on the ratio of the truss element diameter and servo drive gear diameter. Thus, smaller servo drive gears allow for a greater mechanical advantage. In this case, several passive gears were encased in a gear train within a laser cut acrylic bracket. Several gear combinations were experimented with, and discussed further in section 3.7.1.

Thirdly, the new geared truss elements did not work well with the 3D-printed central track. The 3D printed surface was too rough to allow smooth travel, and tended to break when the truss element and it became engaged. Thus, the entire internal track was replaced with acrylic end-pieces that hold the truss element in place. These can be seen as the transparent green end pieces in figure 3.7.

The originally envisioned manipulation method was to manipulate the rods from a single central point. Due to the problems encountered with the previous designs, and the hinge design's typical location near the nodes of the truss structure, this method was rejected. In its place, we envision the robot unfastening a single side of a truss element. Then, if the other end of the truss element is still fixed, the robot would travel to the other end to unfasten it before continuing with the reconfiguration.

Finally, a note about attaching the gears to the servos. The previous 3D printed drive wheels could easily be attached to the servos by use of the standard nylon servo "horns." The servo has a toothed spline of $^3/_{16}$ inch diameter, and the servo horn has a mating receptacle. From this receptacle, arms can branch out, typically singly, in pairs, or in a cross-shaped fashion. In 3D printed wheels, receptacles for these arms are easily made, or, alternatively, fastening the receptacle itself into a hole in the wheel via super glue. None of these methods, however, were workable for standard gears, particularly as the servo spline was not long enough to allow a gear's set screw to provide proper fastening. Our replacement solution was to take a $^3/_{16}$ inch aluminum rod and machine a tooth on the end of the shaft and a receptacle of matching dimensions into the servo horn. A machine screw was placed through the length of the aluminum rod into the threads of the servo spline. This allowed a mechanically sound extension of the servo horn, to which the gears could be reliably attached to. (See figure 3.8.)

3.6 Acrylic Gear Drive

Several further modifications to the robot to were made.

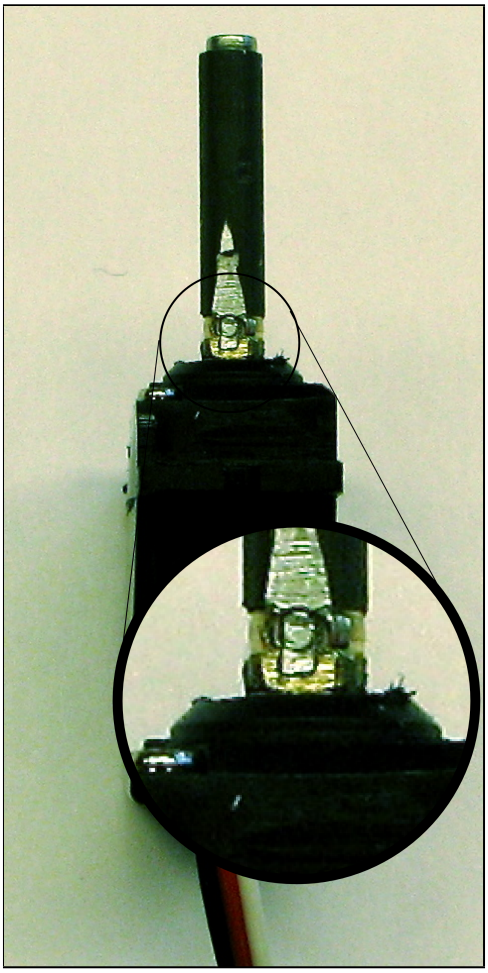

Figure 3.8 – *Extending a servo spline.* An aluminum rod was attached to a standard servo spline via a slot and shaft arrangement.

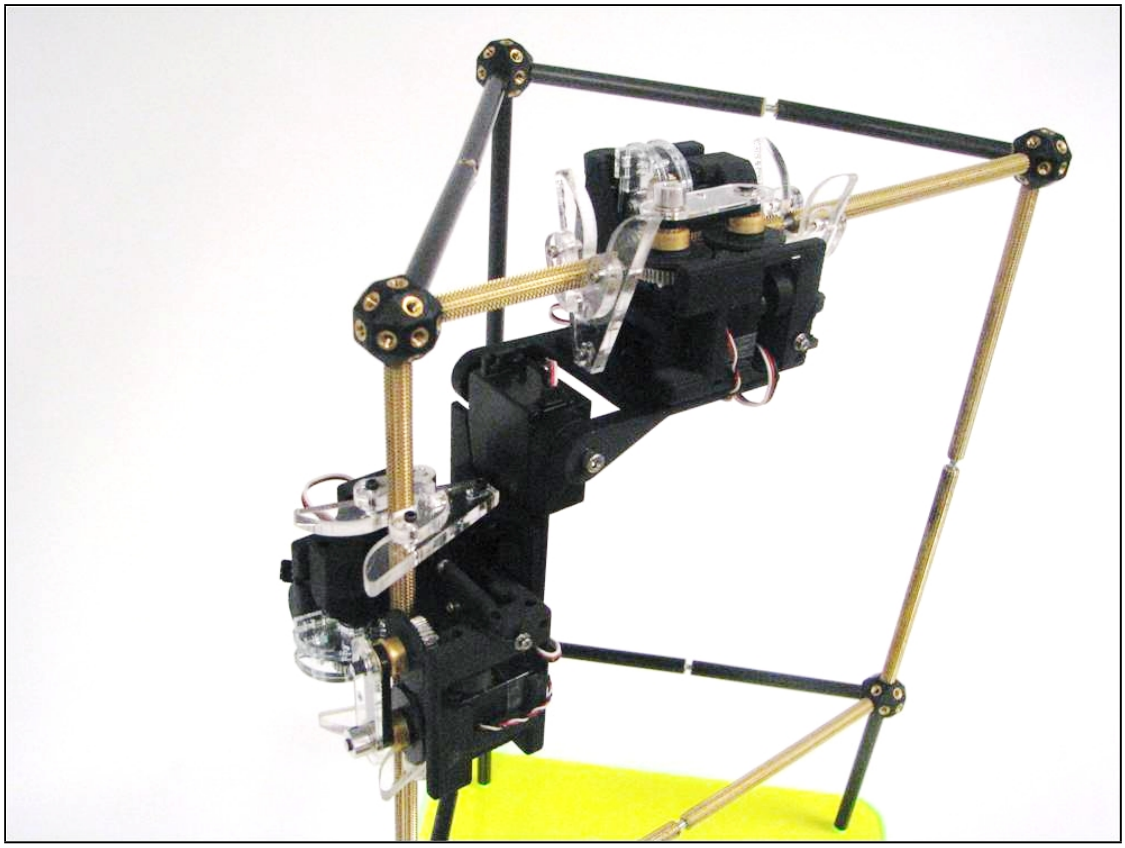

Figure 3.9 – *3D printed gear drive hinge robot photo.* A photo of the 3D printed gear drive robot on a truss structure.

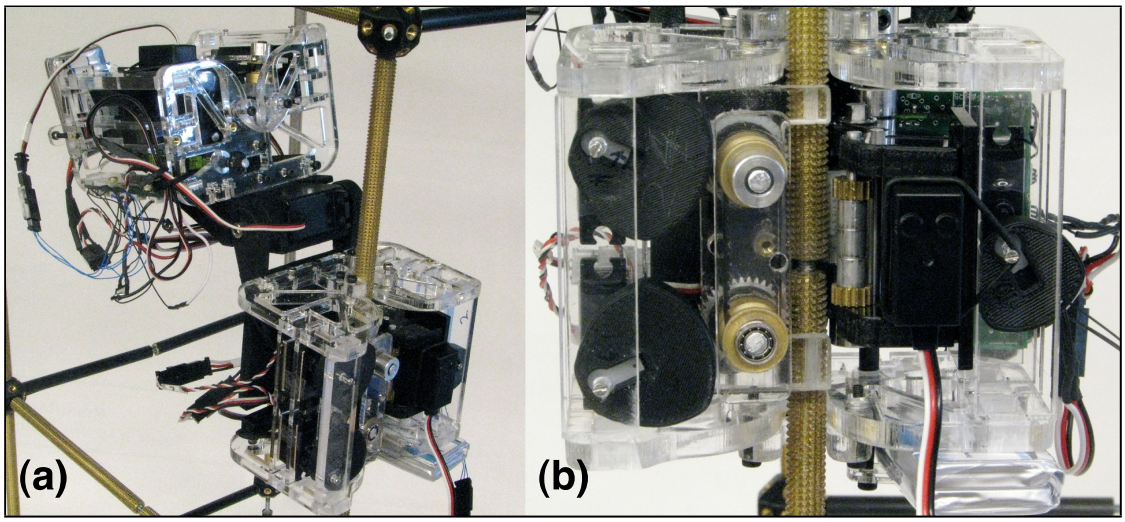

Figure 3.10 – *The acrylic gear drive hinge robot.* Frame (a) shows the robot on a truss structure, and frame (b) shows a view of the internal mechanism, which is similar to the mechanisms of the previous robots.

First, the 3D printed ABS parts were not very stiff, and problems were often experienced with the parts flexing undesirably, particularly on the rotational mechanism. Changing the overall structure of the robot to acrylic increased its stiffness (and, unfortunately, its weight) significantly. Some of the supporting parts are still 3D printed, depending on their required shape.

Second, several design variations of the rotational gearbox were performed, as discussed further in section 3.7.1. The worm gear drive bracket of this robot design was chosen as the final solution.

Third, previous designs did not necessarily allow the rotational and translation mechanisms to engage the truss simultaneously. This was especially true with the worm gear drive bracket. Thus, the discs of the translational mechanism used to hold the robot onto the truss element was replaced with a "["-shaped acrylic bracket. This allowed both mechanisms to engage simultaneously.

Fourth, the translation mechanism did not fasten to the truss elements reliably, so a second cam servo was added.

Fifth, the robot would not hold its translational position while rotation or hold its rotational position while translating. This made open loop control very difficult. 3D-printed gears with gear teeth in the other orientation were used to prevent this motion. Gears for both translation and rotational motion are shown in figure 3.11. Several design iterations were performed to solve this problem, including attempting to place sheet metal between gears to serve as a tooth or putting wire in a gear with slots cut in it. These proved to be difficult to manufacture, whereas the 3D printed gears were straightforward. The 3D printed SLA material needs to be frequently replaced, but is acceptable for short-term

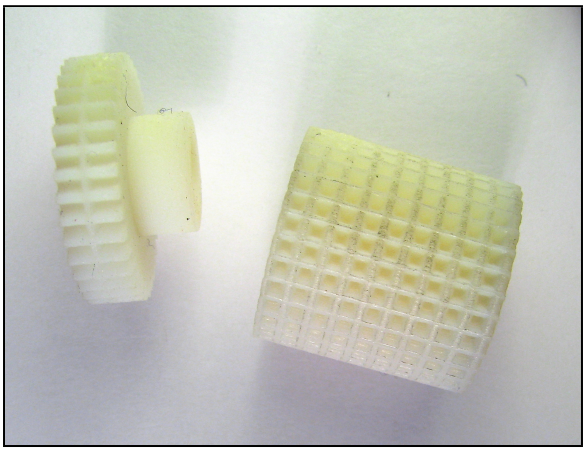

Figure 3.11 – *Degree-of-freedom restrained gears.* These gears include teeth in both directions to prevent slippage. The gear on the left is a translational gear with a single tooth through the middle. The gear on the left is a rotational gear with many crosswise teeth.

experimentation.

Finally, since the holding pod had not yet been developed, it has been relegated to future work.

This robot was capable of performing the movements discussed in section 3.8. It is shown in figure 3.10.

3.7 Other Technical Details

3.7.1 Gearbox Variations

Several variations of the rotational drive mechanisms and gearbox were developed and tested. The first, a belt drive, was used with the carbon fiber truss elements. The three last were developed for the geared truss elements.

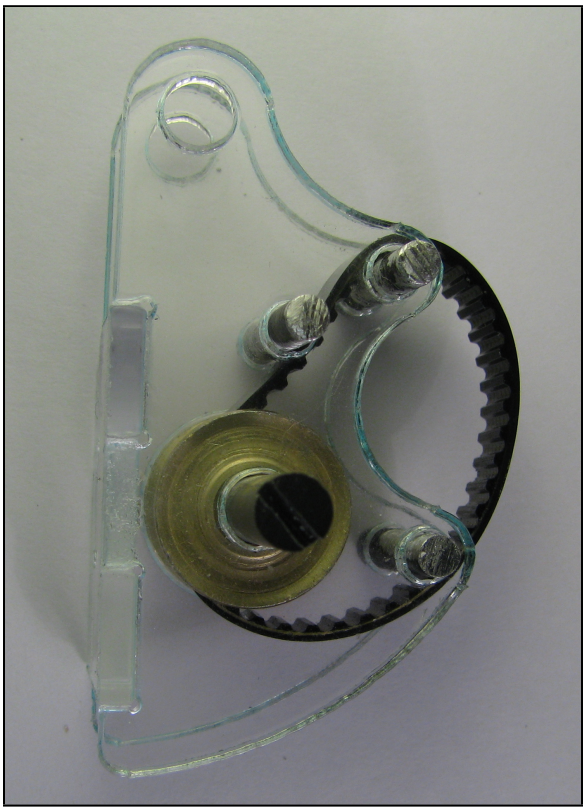

Figure 3.12 – *Belt drive for the rotational mechanism.* The truss element fits in the concave portion on the acrylic bracket on the right. This tensions the belt, and approximately 1/3 of the circumference of the rod is in contact with the belt.

Servo Belt Drive

This belt drive (shown in figure 3.12) was developed for use with the carbon fiber rods, in order to increase the force transferred between the drive servo and the truss element. The belt wrapped around approximately 1/3 of the circumference of the rod. Though the torque applied about the rod was increased, it was determined that the geared truss elements (discussed in section 2.3.2) were a better option.

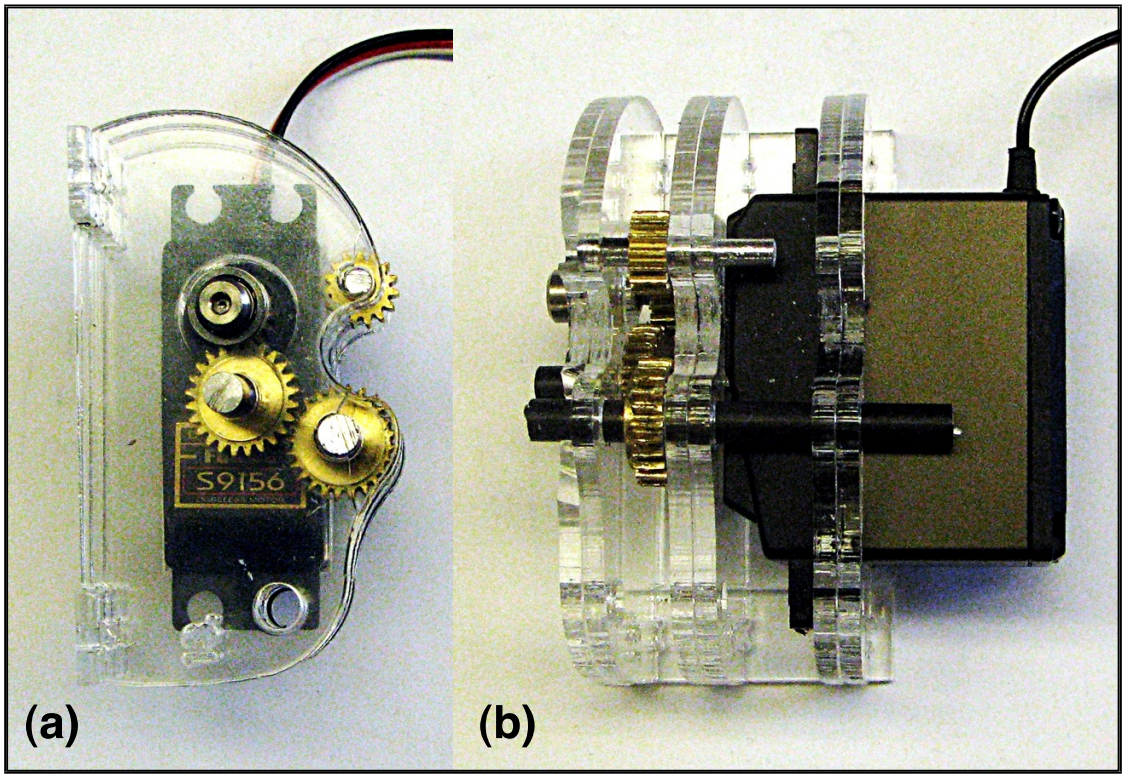

Figure 3.13 – *A direct drive gearbox.* There is one passive spur gear at the top. The other gears transmit torque from the servo to the truss element without any further gear reduction. Frames (a) and (b) show two views of the same mechanism.

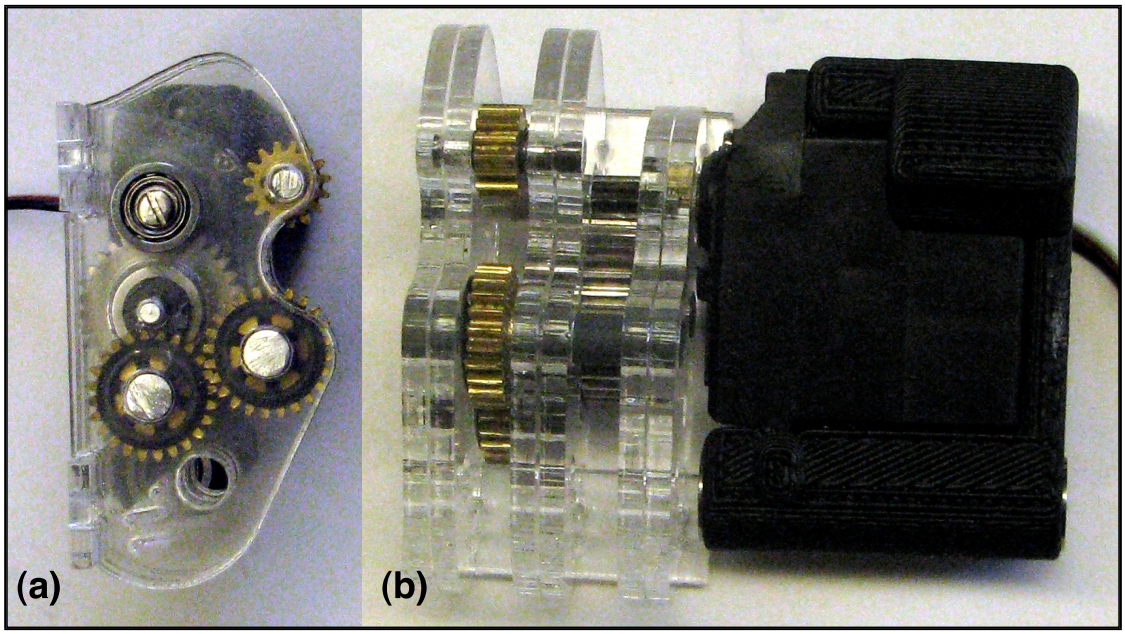

Figure 3.14 – *An 8:1 spur gearbox.* This was developed to increase the mechanical advantage from the gearbox shown in section 3.7.1. Frames (a) and (b) show different views of the same mechanism.

Direct Gear Drive

The belt drive shown in section 3.7.1 was adapted for the new geared truss elements. Since the gear ratio between the servo and the rod is determined (barring an internal gear train) by the tooth ratio between the gear connected directly to the servo and the truss element itself, the smaller servo drive gears increase the mechanical advantage. Then, since the small gears cannot contact the truss element, passive gears are required to transmit torque. This is shown in figure 3.13.

8:1 Gearbox Drive

In order to increase the available torque, a gearbox composed of spur gears was developed with an 8:1 gear ratio between servo and truss element. This required

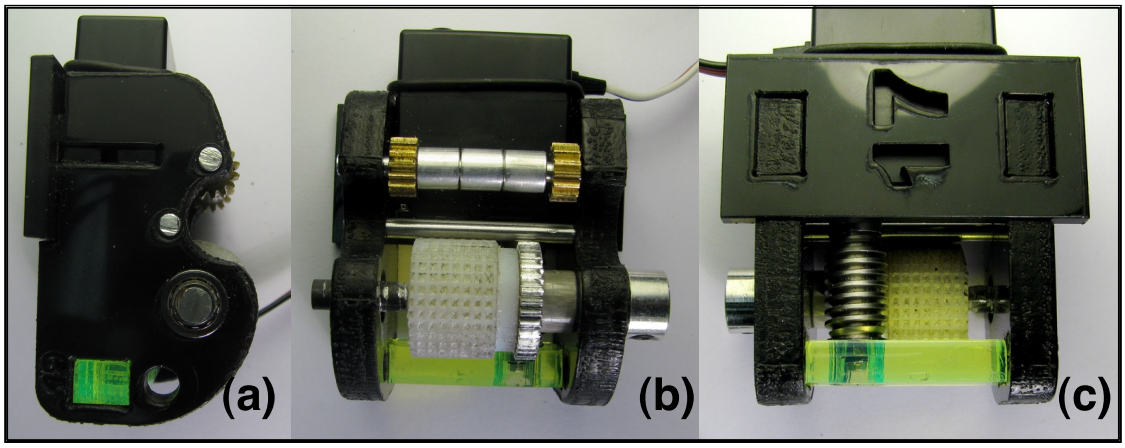

Figure 3.15 – *An 16:1 gearbox utilizing a worm gear.* Use of a worm gear decreased the friction of the system in section 3.7.1 while providing an increased mechanical advantage. Frames (a-c) show different views of the same mechanism. The worm gear is visible in figure (c), and the printed gear is visible in frame (b).

a series of gears with 8 (drive), 36, 8 and 16 (rod) teeth in series. Several gears were used to transmit torque from the final 8 tooth gear to the truss element. The primary disadvantage in this system was the increased amount of friction inherent with the large number of gears.

16:1 Worm Gearbox

In order to decrease the friction in the gearbox described in section 3.7.1, a worm gear attached to the servo was utilized, and transmitted the torque via a intermediate gear to the truss element. This provided a 16:1 mechanical advantage.

One difficult issue was keeping the drive gear in engagement with the truss element. First, the stiffness of the system was increaded with the acrylic body assembly. Second, the gear that contacted the truss element was located as far away from the engagement cam as possible, which increased the force with which engagement could be maintined. Third, the material of the acrylic brackets was changed to Delrin, which is significantly less brittle. Finally, the intermediate

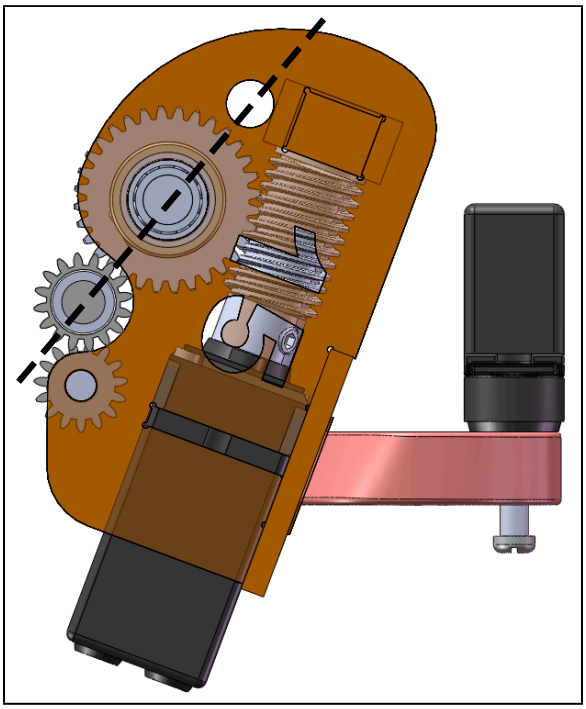

Figure 3.16 – *The necessary alignment of axes in the worm gearbox.* The dotted line crosses through the axes of the rod (lower left), the intermediate drive gear (middle), and the mechanism axis (top).

gears' axis was moved inline with both the truss element and the pivot point of the rotational mechanism. (See figure 3.16.) The normal forces created during the rotation of the drive gears were directed along this axis. Thus, the fixed mechanisms rotation point and truss element could restrain the mechanism from disengaging, rather than the (relatively weak) engagement cam.

The fundamental remaining issue is that the geared rod tends to have a large amount of friction between it and the friction points on the robot.

3.7.2 Electronics

Initial versions of the robot were powered via a LynxMotion SSC-32 servo controller, controlled via a serial interface. LynxMotion's control was used

software originally, but we soon wrote a simple C++ program to send commands to the control board in response to keyboard input. A similar command protocol was used once the tethered control board was replaced with on-board wireless electronics. This protocol consisted of sending a `Pnxx` string over a serial interface to the wireless board. The `n` is an integer from 1 to 6 that represents the number of the servo that to be addressed. (Each half of the robot is controlled by a separate control board; including the hinge servo, there are a maximum of six servos on either side.) The `xx` is a two-byte sequence that represents the high byte and low byte (in that order) of the pulse width of the servo. This pulse width ranges from 500 to 2500, with 1500 the center, as in standard servo control. In the end, our control software gave the user the option of performing low-level actions (such as engage the translation mechanism on side one) or higher-level open loop programmed actions (such as switch the side of the robot that is attached to the structure from one side to the other). The lower-level primitives could be easily called by a programmer wishing to open-loop automate part of the robot.

The microcontroller used was a Atmel ATmega32L-8AU and the wireless controller was a Radiotronix Wi.232FHSS-25-R that provided a serial interface.

Power is provided by a pair of 730 mAh, 7.4 v lithium-ion batteries by ThunderPower. It is estimated that these batteries would provide roughly 30 minutes of typical use in the robot.

3.7.3 Servos

Three kinds of servos were used in the robot: servos for the hinge, for the drive motions, and for the engagement/disengagement cams. Their specifications are

Table 3.1 – Specifications for the robot servos.

Purpose	Model	Max. Torque at 6v	Dimensions (mm)	Mass (g)
cam	GWS Pico Std	0.84 kg · cm	22.8 × 9.5 × 16.5	5.4
hinge	Futaba RS404PD	8.9 kg · cm	40 × 20 × 39	47.5
drive	GWS Micro 2BB MG	6.4 kg · cm	29 × 14 × 29.8	28

Table 3.2 – Movement statistics for the hinge robot.

Movement	Success	Tries	Success %
Vertical to vertical	4	10	40%
Horizontal to vertical rotate	9	10	90%
Vertical node traversal	6	10	60%

shown in table 3.1.

3.8 Movement Statistics

Data has been collected as to various movements of the robot under open-loop control. This is presented in table 3.2. The movements tested are as follows:

Vertical to vertical. The robot is placed in a square of truss elements, arranged with one half attached to a vertical element, and the other half facing up directly under the top horizontal element. It travels to the other side of the structure via the top horizontal element. The failures varied in cause. Approximately half of the failures were open-loop failures (such as incorrect alignment with a truss element to be latched on to) which would be solved with closed-loop control. The other half were equipment failures: servos breaking, batteries wearing out, etc. See figure 3.17 for more detail.

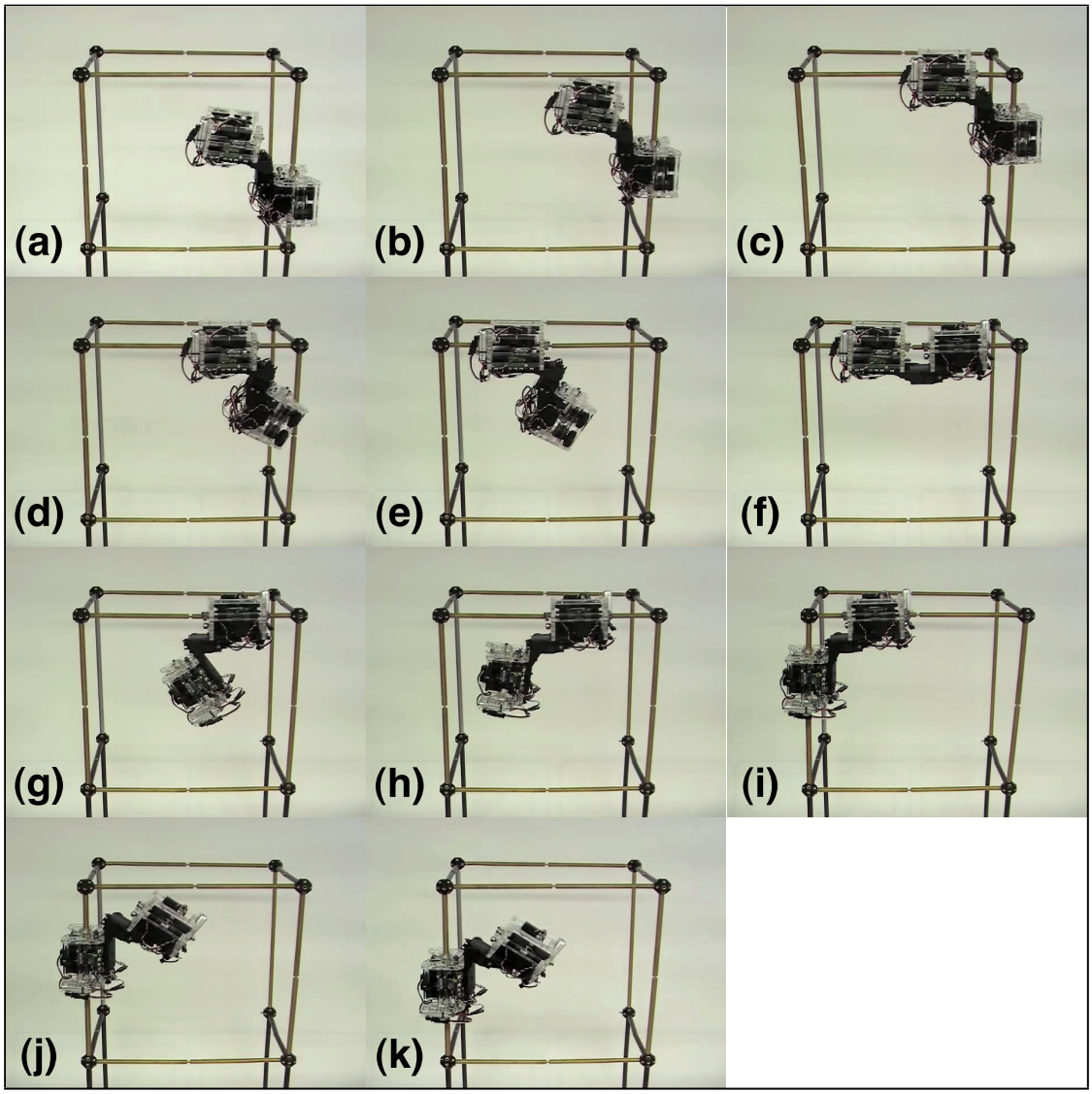

Figure 3.17 – *Steps in a truss traversal.* These frames show steps in a successful vertical to vertical motion.

Horizontal to vertical rotate. The robot is placed on the bottom of the square of truss elements. Similar to the beginning of vertical to vertical, it connects to the vertical truss element below the square—now, however, with the rotational mechanism. The rotational mechanism then rotates the robot outwards in preparation for the next movement: vertical node traversal.

Vertical node traversal. The robot is placed below a node, travels up to it, and adjusts the hinge from 45° to 180°. It then attaches above the nodes, adjusts the hinge appropriately, and moves up past the node.

CHAPTER 4

RECONFIGURATION ALGORITHMS

The mechanical reconfiguration aspect is only one portion of the overall machine metabolism problem. Once a robot exists that can perform autonomous truss reconfigurations given an instruction set (say, a list of truss elements to move, their destinations, and the intra-truss paths to use in the reconfiguration process), what algorithms can be used to transform a given truss structure (a collection of raw truss element materials) into a destination structure that is determined solely by a specified set of functional requirements? Also, what algorithms can be used to plan a path between transformations once the final structure is determined?

This section will describe three algorithms that approach different parts of these questions.

4.1 Construction Tree Reconfiguration Algorithm

The first question to be discussed is a question of automated reconfiguration design. If given a initial truss structure and a functional description of the tasks to perform (e.g. a tall tower or a bridge with maximum length, etc.), can our system autonomously create a design to satisfy those conditions?

Others have considered autonomous truss optimization problems before. The first approach was performed by Dorn et al. [13] in 1964. Genetic algorithms as proposed by Holland [20] have been used for truss topology optimization in several places [6, 26, 35]. That work, however, did not approach the reconfiguration problem, it rather optimized a truss structure given a non-constant supply of raw

materials. It is desired to obtain a set of reconfiguration instructions that perform the changes desired.

This work was done in collaboration with Daniel Lobo [28].

4.1.1 Construction Tree

A truss structure could be represented in a computer in any number of different ways. A representation that is conducive to algorithmic manipulation, that can be determined from a given truss structure, and which conserves the number of truss elements during manipulative processes is required. While a grammar or an L-system would allow representation of a wide variety of interesting truss structures, there is no straightforward way of determining the representation from an arbitrary given truss structure. A simple graph, in which the struts could be represented by edges in the graph and nodes represented by nodes in the graph, is easy to construct from a give truss structure, but ensuring that the physical constraints of the truss are met while reconfiguring the graph is also not straightforward. Instead, a construction tree representation was developed that is conducive to our requirements. This will be explained next.

A construction tree is a deterministic representation of a truss structure formed from a series of assembly operations stringing together primitive building blocks of truss elements into a tree structure. Traversing this tree structure allows one to reconstruct the original structure. Additionally, a valid construction tree (often non-unique) can be generated from a given structure.

The choice of building blocks used in a construction tree has a direct bearing on

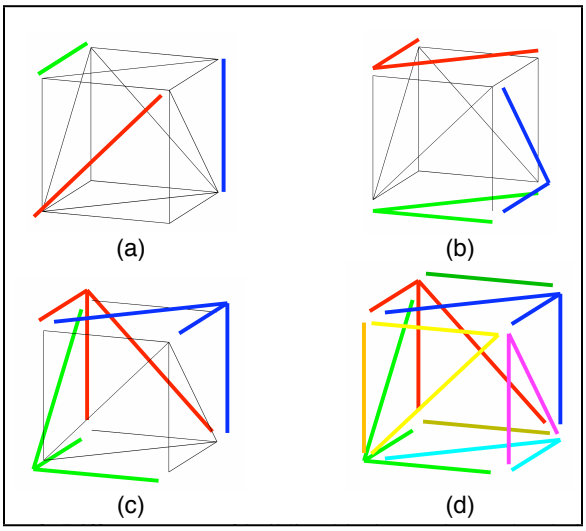

Figure 4.1 – *Construction tree building blocks.* Different types of building blocks can be used to build a construction tree from a truss structure. They are highlighted in color. (a) Three building blocks of only one edge each. (b) Three building blocks of two edges each. (c) Three building blocks of three edges each. (d) The example structure completely composed of building blocks of one, two, and three edges.

the solution for the problem. In this case, a building block refers to a construction of one or more truss elements plus the necessary joints to connect them together. (This algorithm does not conserve the number of joints; that will be a consideration for future work.) Figure 4.1 shows an example of building blocks composed of one, two, or three truss elements, and shows how they can be combined to construct a simple truss structure. More complex structures can be easily constructed from similar building blocks. In this application, the building blocks are limited to be chosen from this list, in decreasing order of preference: a pair of 90° truss elements, a pair of 45° truss elements, a pair of either 135° or 180° truss elements, and, finally, a single edge.

To obtain a construction tree representation of a structure, one first picks a initial joint of the structure. (Different initial joints yield different construction trees for the same structure.) This joint is added to a queue. Iteratively, the joints

in this queue are then processed. For each joint, the algorithm determines if a building block (with edges not already used in the construction tree) exists that uses that joint. The first building block that matches is stored in a new node in the construction tree (as a child of the node that contains the joint being processed), and the joints in the building block itself are added to the queue to be processed. If no building blocks match, the next joint in the queue is processed. Provided that a single-edged building block is included, the structure can be described completely. Figure 4.2 shows a step-by-step example of this process with a simplified 2D truss structure. The construction tree can be transformed back into a structure by a simple preorder traversal, connecting the joints as specified in each node.

4.1.2 Structure Variation Operators

Construction tree representations have the distinct advantage that reconfigurations of the construction tree correspond directly to physical reconstructions. For instance, removing a branch of a construction tree and moving it to a different location on the tree corresponds to making a complete split between two portions of a physical truss structure and reattaching one portion to another location.

For the purposes of our algorithm, four operations are devised to alter a construction tree.

- **Branch move.** A node is selected from the tree, and its parent is changed to another node in the tree.
- **Partial branch move.** This is similar to the previous branch move, except

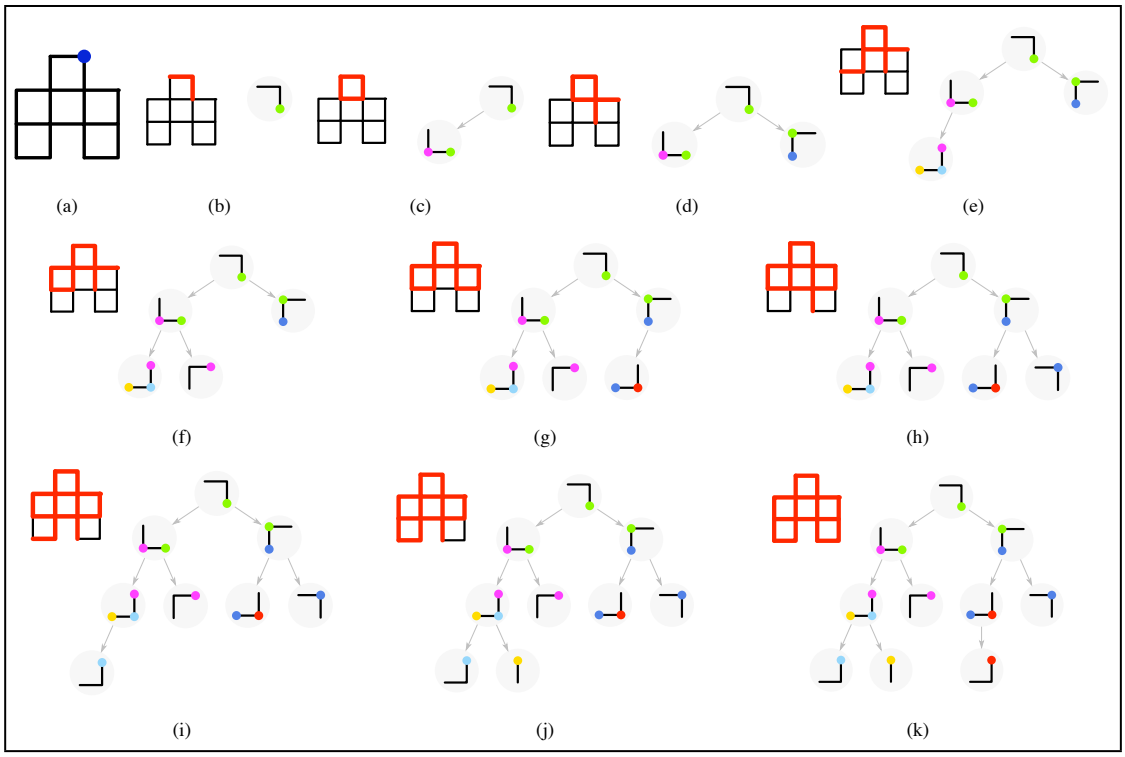

Figure 4.2 – *Construction tree representation of a structure.* Construction tree representation of a structure. A step-by-step example of the generation of a construction tree simplified using a 2D structure. (a) The given structure with the initial joint highlight in blue. (b-k) Steps of building the construction tree. In each frame, the given structure with the edges process so far is on the left, and, on the right, the built construction tree.

that only a part of the branch will move to the new location. The remaining section of the branch is connected to the original node's parent.

- **Branch swap.** Two nodes (and their sub-branches) are swapped between parents.
- **Branch node rotation.** A node is selected from the tree, and all subnodes are rotated by a constant angle, compatible with the physical capabilities of the joints. Physically, this causes a rotation of the section of truss structure around a node.

Figure 4.3 shows an example of various operators on a simple 2D truss. Frame (a) shows the initial structure and its construction tree. Frame (b) highlights a branch of the construction tree and the corresponding portion of the truss. In frame (c), the branch moved from one parent to another in the construction tree, and the corresponding change in the physical structure. Finally, frame (d) shows a rotation operator of 90° applied to the subtree.

These operators refer to nodes using an indirect method, so that they are not broken during mutations or crossover. Integral values between 0 and 1000 are split between the nodes, assigning ranges of values to each node. Thus, when a instruction refers to node 87, it is really referring to the node whose range includes 87.

4.1.3 Genetic Algorithm

A genetic algorithm is an optimization algorithm that utilizes procedures such as reproduction, mutation, selection, and crossover to optimize a measure called

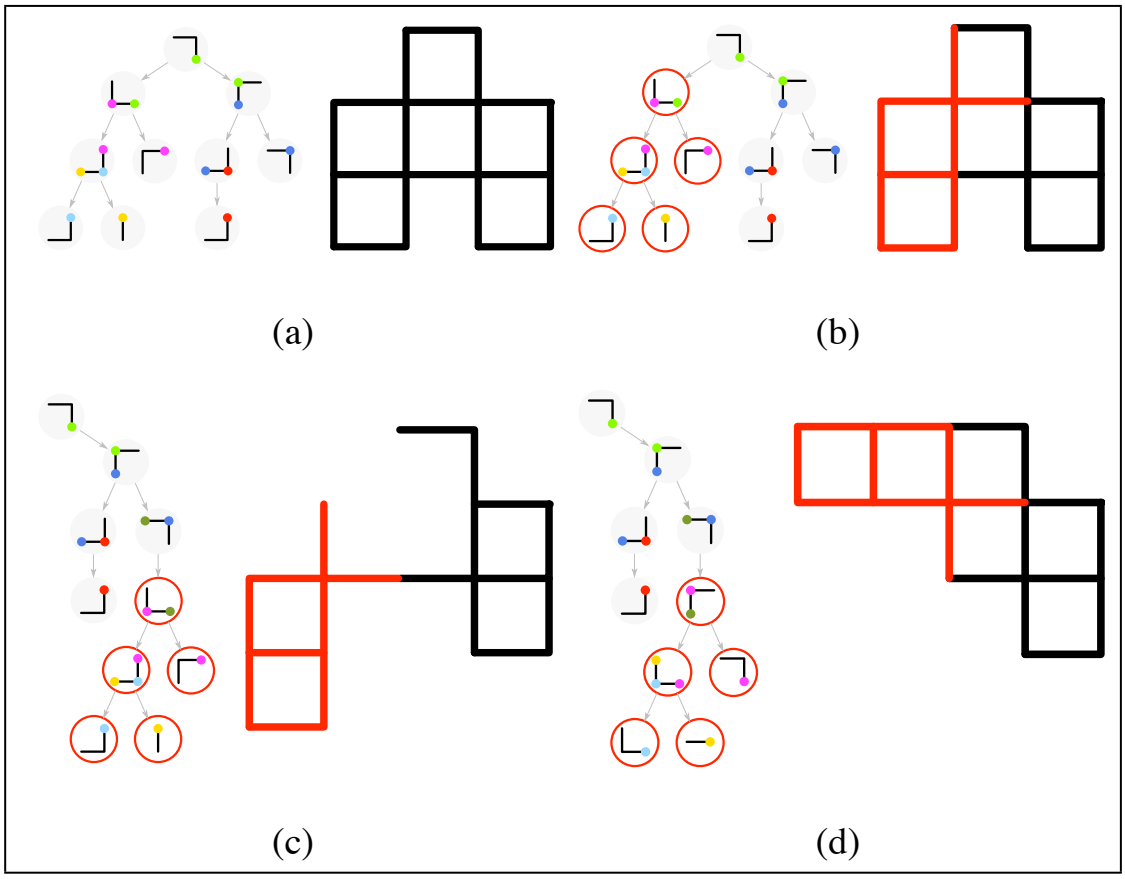

Figure 4.3 – *Construction tree operators.* An example of the branch move and rotation operators applied to a construction tree. (a) An initial structure and its construction tree. (b) The branch selected by the move branch operator and the structure edges codified by that branch are highlighted. (c) The branch has moved according to the branch move operator by changing the parent of the root node of the branch. The resulting change in the structure is shown on the right. (d) A rotation operator is applied to the same subtree.

a fitness. Typically, a population of n individuals is randomly created. Various methods of selecting individuals from that population can be performed, and the selected individuals then "reproduce" using crossover techniques to produce a new generation of population. Mutations may also occur. The individuals in the new population are evaluated for their fitness, and the process begins again.

The internal representation of the truss structure and the allowable operations discussed above can be combined into a genome that can be evolved within a genetic algorithm. This genome consists of, first, a gene that encodes the starting joint of the construction tree, so that the identical construction trees can be deterministically reproduced. This is the only gene in the initial population, and is selected at random for each initial individual. Secondly, the genome consists of a variable-length list of the operators from section 4.1.2 and their parameters. Thus, an entire reconfiguration process is encoded in the genome.

Deterministic crowding [29] is used for the selection criteria, where offspring replace their parents in the population only if they have equal or better fitness. The population size of 50, mutation probability of 0.5, and the crossover probability of 0.5 have been empirically determined. The genetic operators used between generations are one-point crossover and mutations. A variety of mutations could be performed to a genome, including:

- **Mutate the initial joint.** Randomly change the initial joint recorded in the genome to another joint.
- **Mutate a gene's parameter.** A parameter of a gene, such as rotation angle, affected node(s), etc., are mutated to another valid value.
- **Delete a gene.** An operation gene is randomly removed.

- **Add a gene.** A randomly constructed new gene is inserted randomly into the genome.

To evaluate fitness, the functional requirements are evaluated (for instance, the height of the resultant structure), and are linearly combined with the number of genes in the genome in such a way so as to "reward" shorter instruction sets. If a instruction set yields overlapping edges at any point of the reconfiguration, that is punished with a very low fitness value as it is not physically realizable. Finally, the vertex connectivity is evaluated. This is a shortcut method to evaluate the structural robustness of the resultant structure. The minimum number of joints whose removal would completely separate the structure (analogous to the vertex connectivity of a graph) is calculated and, if that value is below a threshold (typically 3), the individual is punished with a very low fitness.

4.1.4 Results

Several variations of fitness evaluations were used, corresponding to different functional requirements.

Maximizing Height

The first objective for the algorithm was to transform a truss structure as in figure 4.4, frame (1), into a structure that maximized its height. The result can be seen in frame (9) of figure 4.4. Frames (2) through (9) show the actual steps of the reconfiguration as performed physically with elements described in section 2.3.1. Figure 4.5 shows the fitness values over time during a run on 100 generations.

You can see that the algorithm successfully resulted in a truss structure eight units high.

Robust Reconfiguration

In order to demonstrate the algorithm's robustness to resource uncertainty, the algorithm was run again, using the same initial structure as before (in figure 4.4 frame (1)) except with a random 10% of the initial edges deleted. The average height of 20 runs yielded the same 10 unit height that the algorithm did under the same vertex connectivity criterion except with no edges removed. A sample of the initial structure and final structure are shown in figure 4.6. The vertex connectivity of 2 was chosen because the height differential between the initial and final structures was greater.

Physics Simulator Results

Another test was performed using a physical simulator based on springs [27] was used to test the performance of a truss structure based. In this case, the initial structure was still figure 4.4, frame (1). The objective was to maximize the horizontal length of the structure while minimizing the tension of the edges of structure and the deflection of the structure under it's own weight. Length, tension, and deflection were equally weighted. Figure 4.7 shows the result of reconfiguration produced by the algorithm.

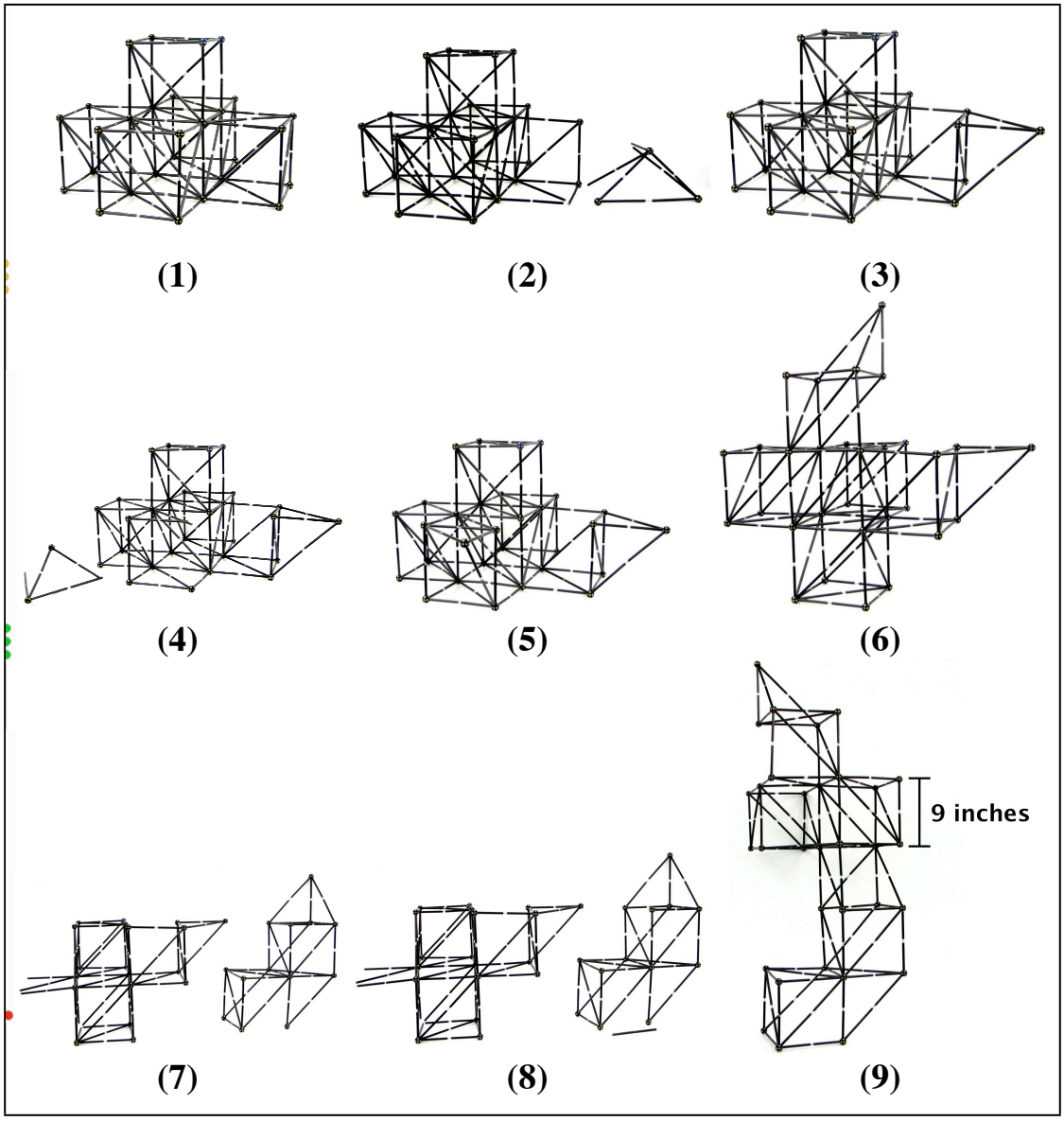

Figure 4.4 – *Maximizing the height of a truss structure.* A reconfiguration result, with frame (1) as the original structure and frame (9) as the resultant structure, with the functional specification of maximizing height. Frames (2) through (9) represent the actual reconfiguration steps performed on a physical truss structure—in this case, the structure described in section 2.3.1.

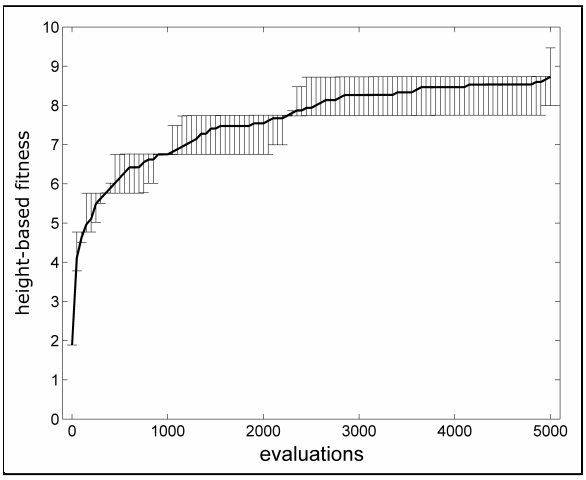

Figure 4.5 – *Evolutionary fitness values over 5000 generations.* Average maximum population fitness values over 15 evolutionary runs, with error bars indicating the first and third quartiles.

Other Results

Finally, other reconfigurations were performed, and are shown in figure 4.8. These include a bridge (frame (b)) and a tower (frame (c)) formed from a more complex initial structure (in frame (a)), but also a tower (frame (f)) and a shelter structure (frame (e)), designed to have the maximum above-ground area, from an initial hand-designed bridge structure (shown in frame (d)).

4.2 Piecewise Reconfiguration Algorithm

The previous reconfiguration work had several limitations to be applied directly to machine metabolism. The primary limitation is that the reconfiguration steps that were produced did not yield physical reconfiguration steps that could be performed by the hinge robot discussed in chapter 3: reconfiguring the structure in large chunks is efficient in terms of operations necessary, but is simply not possible for the hinge robot to do physically. Thus, a new genetic algorithm was

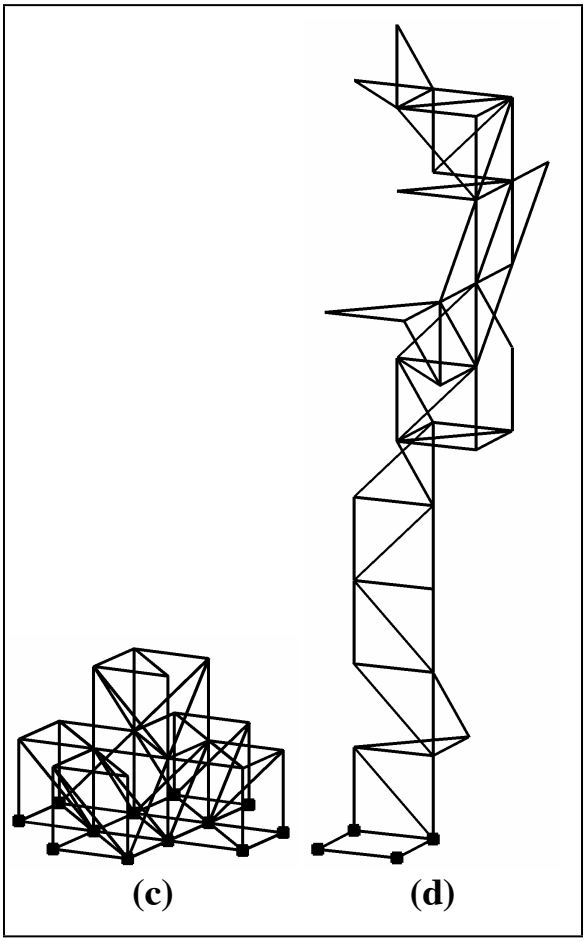

Figure 4.6 – *A reconfigured truss under resource uncertainty.* Reconfigured structure found by the algorithm from the given structure in frame (c). The resulting structure does not degrade (in terms of height or vertex connectivity) with the loss of edges of the given structure.

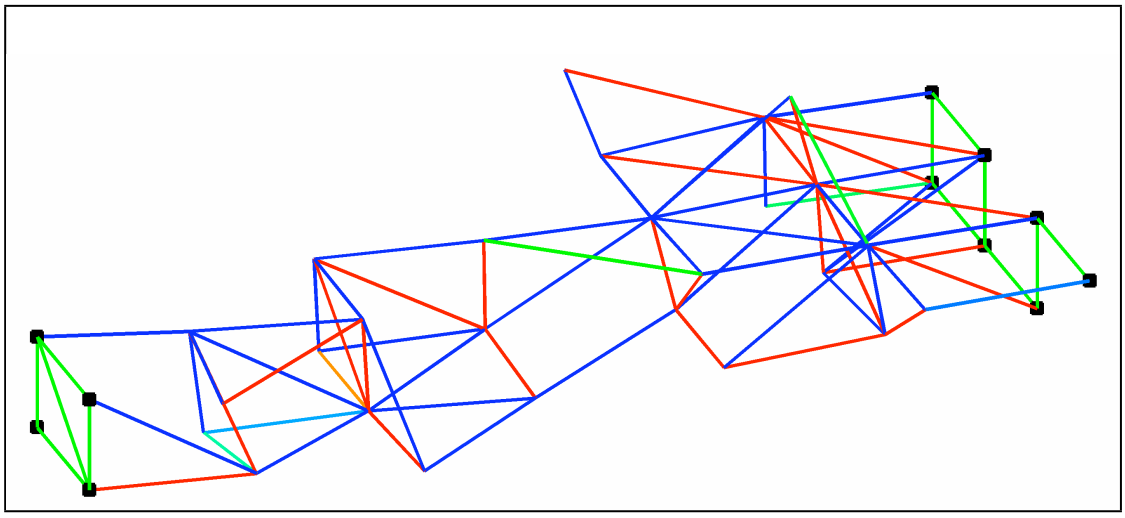

Figure 4.7 – *A evolutionary bridge, design with a physics simulator.* Evolutionary result using a physics simulator to maximize length while minimizing internal stresses and overall deflection.

created to perform this task.

Three primary attempts at accomplishing this goal were made, using three different representations at the heart of the genetic algorithm. Each of the representations, at its base, consists of a single list of parametric instructions that could be fed to a robotic system to perform. The first two, the rotation-based instructions (of section 4.2.1) and the Cartesian-based instructions (of section 4.2.2), are simple lists. The third uses the same Cartesian-based instructions, but instead arranges them via a parametric l-system for its representation. Each of these and the results achieved with each representation are discussed below.

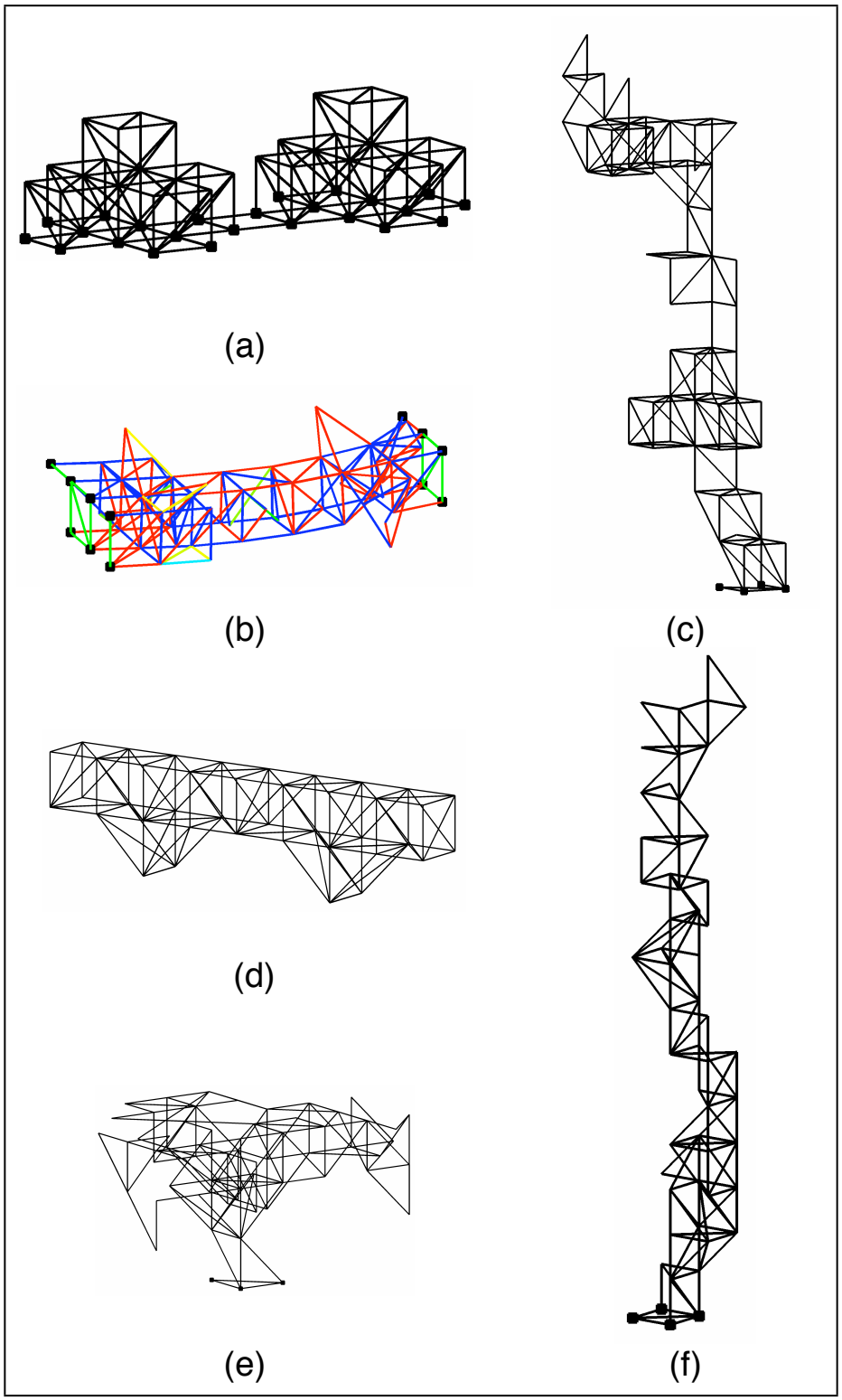

Figure 4.8 – *Additional evolutionary results.* Additional evolutionary results. From the two tower structure of (a) a bridge using the physics simulator (b) and a tower that maximizes its height (c) are evolved separately. In (d) is shown a hand-designed bridge used to evolve both (e) a shelter structure, covering the maximum area above the ground, and (f) a tower that maximizes its height.

4.2.1 Rotation-Based Operators

Instruction Description

The first set of instructions (see table 4.1) used were based on a rotation-based reference, explained shortly. All rotations are based off of a current location, which is a reference to the location of a virtual robot on the structure. This *current location* consists of two parts: the *current edge*, which corresponds to one of the edges in the structure, and the *current node*, which corresponds to either end of the current edge. This node does not necessarily correspond to a physical connector in the truss structure; it simply represents one of the ends of the edge. The rotations, then, are relative to this current position.

The first attempt used a three-instruction set of instructions, summarized in table 4.1. Each instruction is relative to the current position.

The `toggleNode` instruction modifies the current position by changing the current node to the other node on the current edge.

`movePosition` changes the current edge, but keeps the current node in the same location. The new edge is specified by two parameters: an axis (x, y, or z), and a rotational direction, indicated by *clockwise* or *counterclockwise*. To determine the new position, a coordinate axes, parallel to the coordinate axes at the origin, is placed at the current node. Then, a standard rotation (about the specified axis and in the specified direction) is performed on the vector that describes the current edge. The potential new position is then examined. For the `movePosition` instruction, it is first verified that a edge exists in the position. If so, the current position is set to the edge just computed. If not, the rotations continue in the

Table 4.1 – The set of three rotation-based instructions that allows truss reconfiguration.

Instruction	Parameters	Description
`movePosition`	*x*-, *y*-, or *z*-axis; CW or CCW	Moves the current position to the nearest truss element.
`toggleNode`	-	Switches the current node to the other node on the edge.
`pickPlaceEdge`	*x*-, *y*-, or *z*-axis; CW or CCW	Either "picks up" or "places" an edge.

specified direction until another edge is found. If no new edges are found, the current position does not change.

The `pickPlaceEdge` instruction either "picks up" an existing edge, relative to the current position, or "places" an edge at a empty position in the structure. Since only one edge can be "stored" at a time by the robot, picking occurs by this instruction only when the "holding pod" is empty, and placing occurs only when the "holding pod" contains an edge. Additionally, the instruction differentiates between the long (diagonal) and short edges in the structure, and ensures that they are not placed in incorrect locations. To pick an edge, the exact same heuristic (using the same parameters) is used as in the `movePosition` instruction, except that the current position is not changed; rather, the edge referred to is listed as "picked." When placing an edge, the location is chosen with the same method (and, again, the same axis and directional parameters), except that only locations that are able to accept an edge are chosen.

Evolutionary Parameters

Once these instructions were chosen, evolution was performed. In this instance, 15 individuals per generation were used, with deterministic crowding [29] used

for selection and repopulation. Each individual is initialized with a random number of random instructions, with a length of 20 to 40 instructions. (The low population number was chosen via hand-tuning; larger populations did not yield notably improved results. This is sensible since there are a total of 13 different instruction and parameter combinations, which would be well represented within the individual population.) Four populations were ran simultaneously, plus a baseline population of a random mutation hillclimber. Fitness was measured by directly measuring the height of the structure after the instructions were performed on the initial truss structure, plus one. (The additional one is a constant offset; it serves no particular purpose. It was an artifact of my truss measuring procedure.) The mutation rate was 0.10 and the crossover rate was 1.0; crossover was a single-point crossover. A random point in the instruction list was chosen from both parents; the left of one parent was combined with the right of the other. Thus, the length of the instruction set could change as the algorithm progressed. Mutation was unsubtle and unintelligent; mutating an instruction replaced it with a new completely random instruction.

Results

After 300 generations, the average best height among the four populations was 4, with the best height amongst all populations being 5. This is an improvement of 3 units above the initial truss height of 2. The baseline population stayed constant after the first generation, with a height of 3. The fitness curve for this run is shown in figure 4.9 and the final truss is shown in figure 4.10.

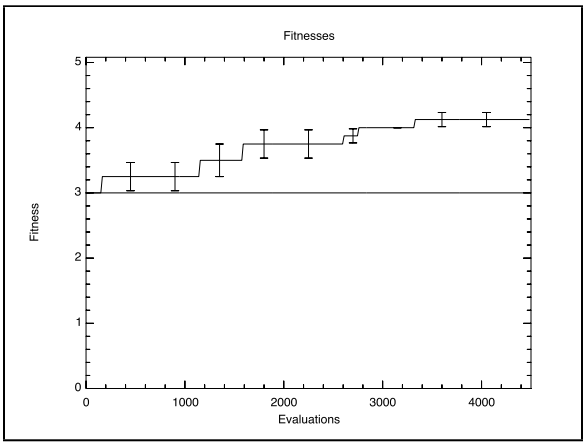

Figure 4.9 – *Fitness curve for rotation-based instructions.* The top line is evolution, while the bottom line is a random mutation hillclimber.

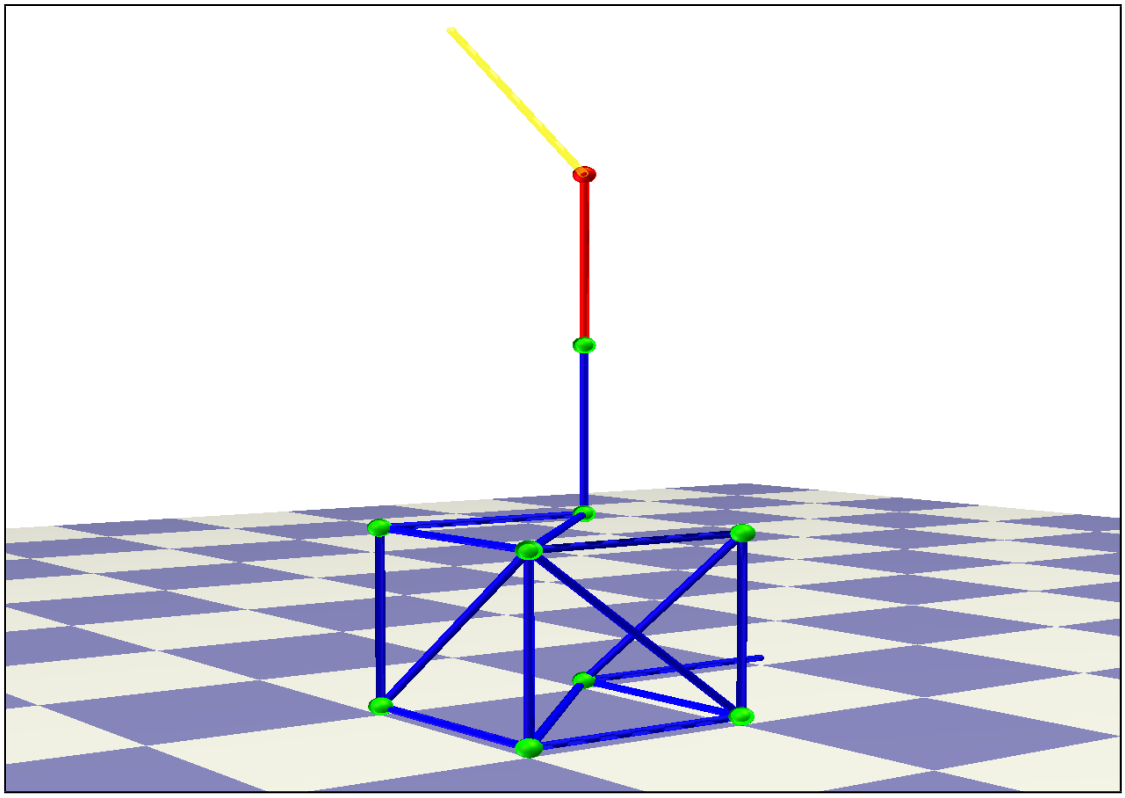

Figure 4.10 – *Truss produced with the rotation-based operators.* The final truss produce by an evolutionary run with the rotation-based operators.

4.2.2 Cartesian-Based Instructions

Clearly, though the above evolutionary run was successful in terms of functionality, it was not successful in terms of height gained per number of instructions, nor in terms of height gained versus maximum possible height from the given truss elements. Further consideration indicated that the rotation-based representation was fragile, i.e. instructions depended heavily on the instructions that preceded them, such that crossover or mutations could completely negate any progress. Thus, a representation that was less fragile was attempted: a representation based upon absolute Cartesian-based movements.

Rather than having a current position that consists of both a current edge and a current node, this representation made use of only a current node. This node is moved via movement instructions, and all instructions base off of the current node. If an instruction is issued to move the node in a direction in which a node does not exist, the instruction is simply ignored.

Clearly, these instructions are less prone to the fragility of the previous instructions. The rotation-based instructions depended heavily on the current topology of the truss around the current location. The position which was reached via a rotation depended entirely on the truss edges that were around the current location. The Cartesian-based instructions, however, only depend on whether a truss element exists in the position that they reference.

Instruction Description

The particulars of each of the instructions will now be discussed, which are summarized in table 4.2. Note that in all cases, the repetition parameter n can be

any integer between 1 and 20.

The `moveNode` instruction changes the position of the current node. This command takes two parameters: a direction parallel to the x-, y-, or z-axes, positive or negative, as well as a repetition parameter n that indicates how many units in that direction the movement ought to occur. If the structure contains a node at this new location, the structure's current node is changed correspondingly. Otherwise, nothing is done.

The `moveNodeDiagonal` instruction also changes the position of the current node in a similar way to `moveNode`. However, instead of a direction parameter describing a direction parallel to the coordinate axes, the direction parameter describes one of the diagonal, by using a combination of two of the basis vectors and + or − signs. In addition to the direction, a repetition parameter n indicates how many units in the direction movement ought to occur. Similarly to the `moveNode` instruction, the current node of the structure is only changed if a structural node actually exists at the location referenced by the instruction.

`pickFromNode` picks a "short" edge located near the current node in the direction specified by the instruction's parameters. ("Pick" here means the same thing as the previous set of parameters; the strut is moved into a virtual holding pod for eventual placement elsewhere.) If an edge does not exist at that location or if the removal of that edge would disconnect the structure, the instruction is ignored. If an edge has already been picked, the new picked edge replaces it.

`pickFromNodeDiagonal` is nearly identical to the `pickFromNode` instruction, except it will exclusively pick diagonal (or "long") edges. The same criteria for ignoring or replacing edges applies.

The `placeFromNode` instruction places any currently picked edge at a location based on the current node. It takes three parameters: a direction parallel to one of the axes, a diagonal direction, and a repetition parameter n. First, it will move the current node in the non-diagonal direction for n units. Then, if the currently picked edge is a regular edge, it will use the non-diagonal direction to place the currently picked edge. Otherwise, it will use the diagonal direction parameter to place the edge. If a placement is not possible (such as if a truss element already exists in that location or a diagonal strut is in a conflicting location), the instruction is ignored.

`placeFromNodeDiagonal` is nearly identical to the `placeFromNode` instruction, except that the initial movement is in a diagonal direction. Otherwise, the placement procedure is identical.

`pushPosition` and `popPosition` serve as a storage mechanism for locations on the truss. `pushPosition` takes the current position and places it into a stack. `popPosition` removes the most recently added position from the stack and sets the current position to it. If there is no location on the stack, `popPosition` has no effect.

Finally, the `instructionBlock` command allows a group of commands to be grouped and repeated as a group. In this case, up to 5 other instructions may be stored in an `instructionBlock` and repeated n times.

Evolutionary Parameters

Once these instructions were chosen, evolution was performed. In this instance, 15 individuals per generation were used, with deterministic crowding used

Table 4.2 – The set of nine Cartesian-based instructions that allow truss reconfiguration.

Instruction	Parameters	Description
`moveNode`	± x-, ± y-, ± z-axis; repetition n	Move current node n times in direction.
`moveNodeDiagonal`	Diagonal direction; repetition n	Move current node n times in direction.
`pickFromNode`	± x-, ± y-, ± z-axis	Pick the edge in direction.
`pickFromNodeDiagonal`	Diagonal direction.	Pick the edge in direction.
`placeFromNode`	± x-, ± y-, ± z-axis; diagonal direction; repetition n	Move in direction n times, and place edge.
`placeFromNodeDiagonal`	± x-, ± y-, ± z-axis; diagonal direction; repetition n	Move in direction n times, and place edge.
`pushPosition`	-	Put current node position onto a stack.
`popPosition`	-	Pop the position off of the stack, if exists.
`instructionBlock`	-	Repeats up to 5 instructions up to 20 times.

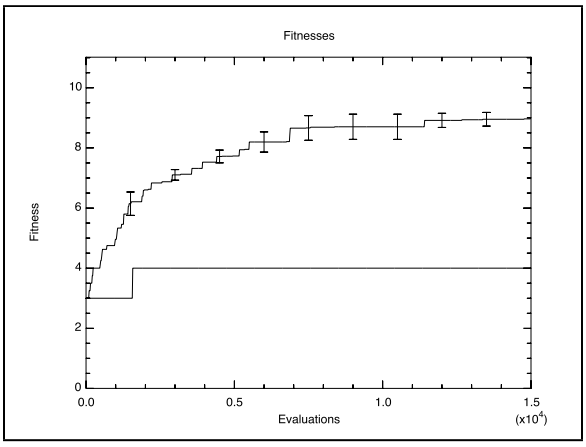

Figure 4.11 – *Fitness curve for Cartesian instructions.* The top line is evolution, while the bottom line is a random mutation hillclimber.

for selection and repopulation. Each individual is initialized with a random number of random instructions, with a length of 10 to 40 instructions. (The low population number was chosen via hand-tuning; larger populations did not yield notably improved results.) Four populations ran simultaneously, plus a baseline population of a random mutation hillclimber. Fitness was measured as before. The mutation rate was 0.10 and the crossover rate was 1.0; crossover was a single-point crossover as before. Mutation was also as before. As mentioned in the instruction description, `instructionBlock`s could contain up to 5 instructions, and the repetition parameter could be any integer between 1 and 20 inclusive.

Results

After 1000 generations, the average best height among the four populations was 9, with the best height amongst all populations being 10. This is an improvement of 8 units above the initial truss height of 2. The fitness curve for this run is shown in figure 4.11 and the final truss is shown in figure 4.12.

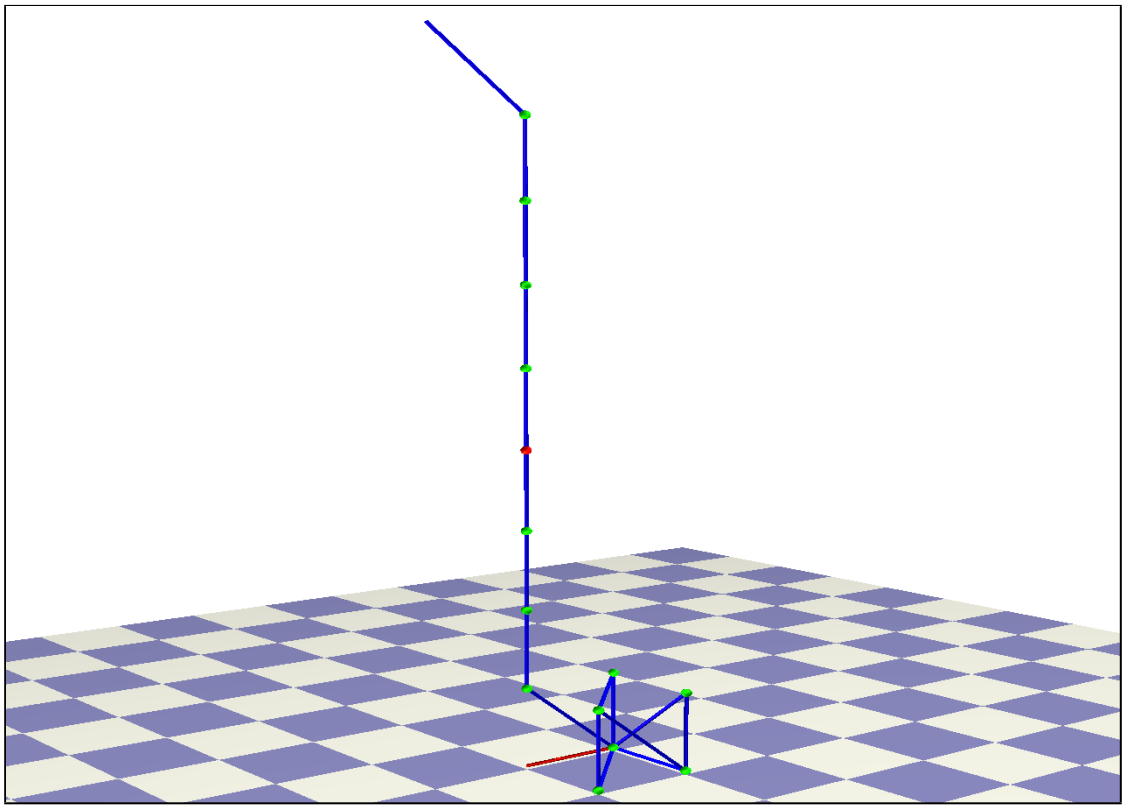

Figure 4.12 – *Truss from Cartesian instructions.* The final truss produce by an evolutionary run with the Cartesian instructions.

4.2.3 Generative Encoding using Cartesian-Based Instructions

Instruction Description

Finally, an evolutionary run using a simple generative encoding, a context-free non-parametric l-system, such as that presented by Hornby and Pollack [21], was tried. In this instance, 5 rules were created, each of which could have 3 to 20 of the instructions listed in table 4.2 or instructions that referenced any of the 5 rules. Starting with a seed instruction, rule references were replaced with the context of that rule. Thus, a short rule set could generate complex modular behavior.

Evolutionary Parameters

In this instance, 100 individuals per generation were used, with deterministic crowding used for selection and repopulation. Each individual is initialized with 5 rules, each of which contains from 3 to 20 instructions (including references to the other rules). Four populations ran simultaneously, plus a baseline population of a random mutation hillclimber. Fitness was measured as before. The mutation rate was 0.10 and the crossover rate was 1.0; crossover was a single-point crossover. This crossover picked a rule number, between 1 and 5, and combined two parents via splitting the rulesets at that point such that the children maintained the same number of rules. Mutation was also as before. As mentioned in the instruction description, `instructionBlock`s could contain up to 5 instructions, and the repetition parameter could be any integer between 1 and 20 inclusive. The system was always seeded with a single instance of the first rule, and 5 iterations were performed to obtain the final instruction set to perform.

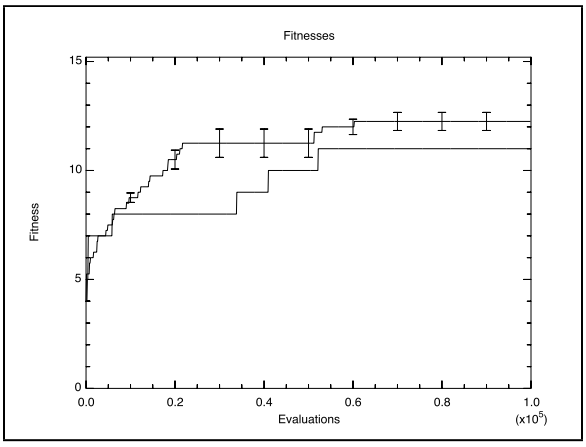

Figure 4.13 – *Fitness curve for generative encoding.* The fitness curve for an evolutionary run with the generative encoding. The top line is evolution, while the bottom line is a random mutation hillclimber.

Results

After 1000 generations, the average best height among the four populations was 12.25, with the best height amongst all populations being 13. The fitness curve for this run is shown in figure 4.13 and the final truss is shown in figure 4.14.

4.2.4 Discussion and Conclusions

As expected, the Cartesian instruction set outperformed the fragile rotation-based instruction set, and using these instructions within a simple generative encoding scheme outperformed either of the other two options. Several observations can be made from this data.

First, none of the approaches made successful use of the diagonal elements. While they were occasionally used to increase height, no more than one diagonal element was ever used in this fashion. We expect that the reason is due to the nature of the instruction sets chosen, and that it is difficult to build upon a

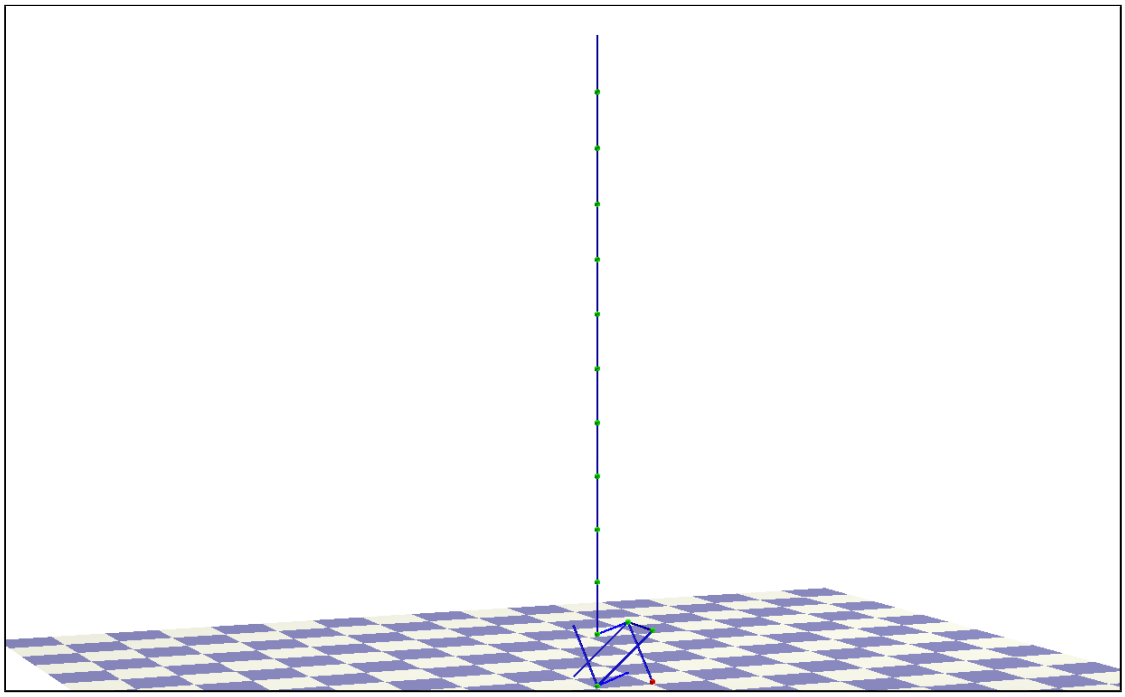

Figure 4.14 – *Truss from the generative encoding.* The final truss produce by an evolutionary run with the generative encoding.

diagonal element. If several truss elements are progressing directly upward, and a diagonal strut is attached to the top, two complementary instructions are required to add a new element above that: a instruction to move directly below the diagonal, and an instruction to move onto or across the diagonal. This is significantly more work than a single movement instruction. We hypothesize that a movement instruction that could avoid this complication would make more effective use of the diagonal elements.

Second, it is clear that none of the evolved structures are physically robust. Implementing a physical simulator as part of the fitness function would likely solve this problem.

Third, one can note that the generative encoding evolutionary function was nearly matched by the performance of the random mutation hillclimber. While this speaks to the effectiveness of the generative encoding approach in this setup,

it also indicates that improvements could be made to the evolutionary process. Parameterizing the l-system rules would be one way to do this. Another option would be to improve the mutation operations to act more gently, tending to adjust parameters rather than completely replacing instructions. Additionally, more work could be done if one wished to improve the very simplistic crossover scheme implemented for the l-system.

Fourth, it is interesting to note that the generative encoding was capable of removing all non-diagonal elements from the initial truss structure except the one that was necessary to to keep the ground structure from disconnecting. If the instructions were more conducive to using the diagonal elements, we expect that it would demonstrate the ability to use all truss elements to maximize the height.

As a whole, however, the experiment to evolve instruction sets capable of being performed by a hinge robot was successful. The structures produced satisfied the conditions set forth, and the algorithm output the desired instruction sets.

4.3 Reconfiguration Planning Algorithm

Another solution to the problem of translating the reconfigurations shown in section 4.1 into instructions executable by the hinge robot was devised in collaboration with MIT [52]. This algorithm takes the initial structure and the desired structure as input, and outputs an optimal way to reconfigure the structures in a way that the hinge robot can perform. As with the previous algorithms, the nodes are not yet considered during computation.

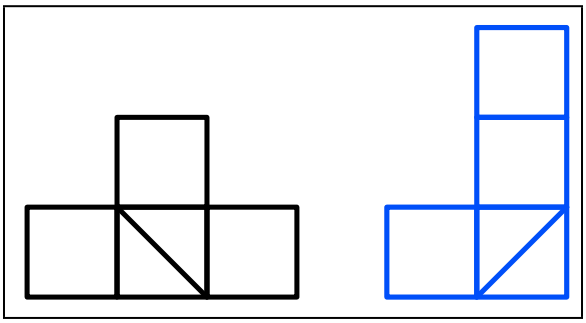

Figure 4.15 – *2D source and target structures for reconfiguration planning.* Source and target structure: G_1 (left) and G_2 (right).

In order to explain the algorithm, a 2D source and destination structure analogous to the 3D reconfiguration case is used. This is shown in figure 4.15.

Several preliminaries also need to be explained. First, as shown in the truss structures in chapter 2, there are two types of truss elements: *sides* and *diagonals*. A truss structure is represented by a weighted graph, G, such that the edges designate truss elements, the nodes represent joints, and the weights represent whether the element is a side or a diagonal. In this instance, as shown in figure 4.15, G_1 represents the initial, or source, structure, and G_2 represents the destination structure. Then, the edges that require moving are described by $G_1 - G_2$, and the final locations for these elements are described by $G_2 - G_1$. First, a graph $G_m = G_1 \cap G_2$ is found that requires the least reconfiguration between the two truss structures. Then, the way in which this reconfiguration can be performed optimally while maintaining the connectivity of the structure is computed.

4.3.1 Optimal Matching Between Trusses

The first problem is to determine the optimal matching between the source and destination truss structures. This is done via algorithm 1. This algorithm

is explained using the 2D example shown in figure 4.15, and comment on its three-dimensional extension.

The essence of the algorithm is this: G_2 is superimposed over G_1 in every possible orientation and position. In the 3D case, that means rotation G_2 about the vertical z-axis at intervals of 90°, and translating each of those rotated trusses on the x- and y-axes to all positions that overlap. This is done by the `for` loops on lines 2 and 6 of algorithm 1. A sample of this scanning process for the 2D trusses originally shown in figure 4.15 is shown in figure 4.16. In the 2D case, of course, there is only translation along the horizontal y-axis and the only allowed rotation is a flip about the vertical x-axis. Then, for each of those possible superpositions, the optimal cost is evaluated for moving the truss elements in G_1 to their corresponding location in G_2. The truss elements that need to be moved are expressed by $G_{source} = G_1 - G_2$, and the locations to which they will be moved is expressed by $G_{destination} = G_2 - G_1$. Computing the optimal cost is done by first computing the costs to move each of the elements in G_{source} to any of the locations in $G_{destination}$ using Dijkstra's algorithm. The sides and diagonals are considered separately. Then, the Hungarian algorithm [25] is used to compute the optimal matching between the elements in G_{source} and G_{dest}. (Again, diagonals and sides are considered separately.) Thus, for each possible superposition, $G_3 = G_1 \cup G_2$, the superimposed truss, and M, the list of "which element goes where" for that particular superposition, are obtained. The total cost for each superposition can be computed, and the one with the lowest cost chosen.

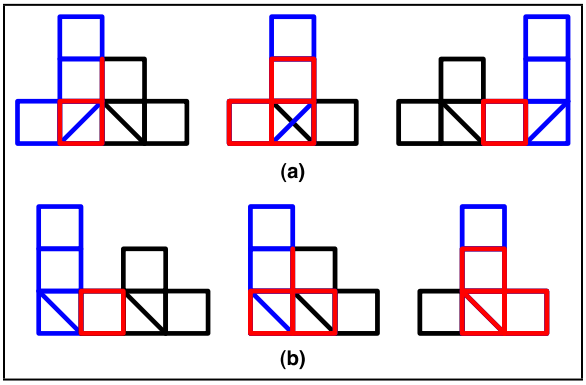

Figure 4.16 – *Scanning the target structure over the source structure* Black edges are struts that belong to only G_1 and blue edges are only for G_2. The overlapped edges are highlighted by the red lines. In frame (a), G_2 is scanned from left to right over G_1. In frame (b), the same occurs, except G_2 is first changed to its other possible 2D orientation by flipping about the x-axis.

Algorithm 1 Scanning algorithm for the optimal matching. The algorithm returns the optimally merged graph G_m and the optimal matching M_m in G_m.

Make G_1 and G_2 cornered at the origin
for orientation $O_{G_2} \in \left[0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right]$ **do**
  Rotate G_2 by O_{G_2} w.r.t z-axis
  $X_1 = \max\{x(G_1)\}$, $Y_1 = \max\{y(G_1)\}$
  $X_2 = \max\{x(G_2)\}$, $Y_2 = \max\{y(G_2)\}$
  for $x_t \in [-X_2 \dots X_1 + X_2]$, $y_t \in [-Y_2 \dots Y_1 + Y_2]$ **do**
    Transform G_2 by (x_t, y_t)
    n = number of the overlapped elements in $G_1 \cap G_2$
    $G_3 = G_1 \cup G_2$
    $M = \text{OptimalMatching}_{side}\ (\ G_1 - G_2, G_2 - G_1\ ) + \text{OptimalMatching}_{diagonal}\ (\ G_1 - G_2, G_2 - G_1\ )$
  end for
end for
$M_m = \text{argmin}_M(cost(M))$
$G_m = \text{argmin}_{G_3}(cost(M))$

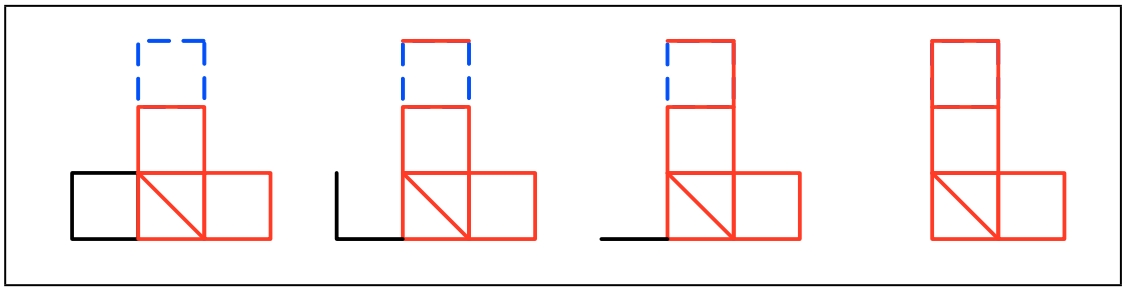

Figure 4.17 – *Broken connectivity by performing the matching.* The black elements are $G_{source} = G_1 - G_2$ and the blue ones are $G_{destination} = G_2 - G_1$. The black elements are moved to the locations shown by blue, at which point they join the destination structure in red. In the computed optimal matching, however, the first element is moved into a location that is disconnected from the rest of the truss.

4.3.2 Planning the Reconfiguration

A problem still exists with this optimal matching, however. This is shown in figure 4.17. In this sequence, the black elements are $G_{source} = G_1 - G_2$ and the blue ones are $G_{destination} = G_2 - G_1$. The black elements are moved to the locations shown by blue, at which point they join the destination structure in red. In the computed optimal matching, however, the first element is moved into a location that is disconnected from the rest of the truss. While optimal in terms of the algorithm stated above, this is clearly not a physically acceptable solution.

Algorithm 2 is our solution to this problem. First, S is defined as the set of truss elements in G_1 that will be moved, and T as the set of destinations. P is the set of paths, such that p_i describes the path computed in algorithm 1 from the original location s_i to the destination local t_i. The process begins with no elements moved from their location in S to their location in T and concludes when all of the elements in S have been successfully moved to T.

To begin the moving process, a truss element s_i is picked to move such that its destination location t_i is connected to the current truss structure and every point

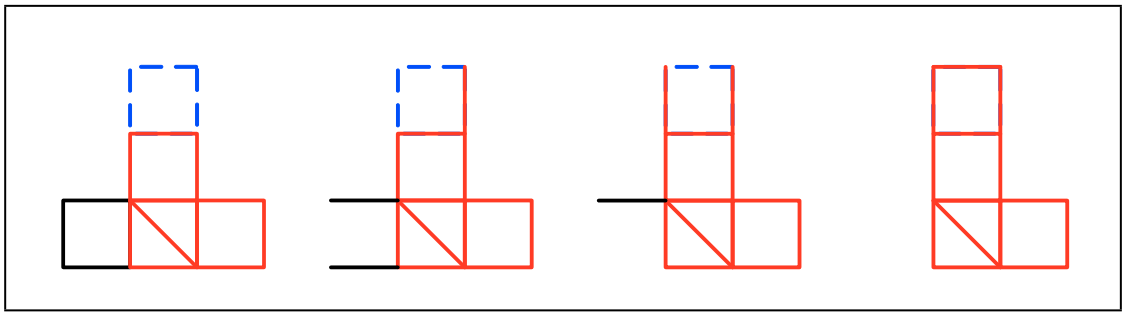

Figure 4.18 – *The adjusted matching after Algorithm 2 has been applied.* The truss elements are now no longer disconnected during the reconfiguration.

along its reconfiguration path p_i is also on the current truss. If this element is not in the reconfiguration path of another element (in other words, no p_k exists that utilizes this position; truss elements used in later reconfiguration instructions are not moved) or if it has been placed in the element storage queue Q (discussed later), s_i is moved along p_i to its destination t_i. p_i is removed from the list of paths, and remove s_i from Q if it originated in the queue.

If s_i did not satisfy these conditions, it can not be moved, as it will block a later reconfiguration. Thus, the source element s_j whose path p_j was blocked by the original element s_i is determined. If this element is of the same type (in other words, $\text{type}(s_i) = \text{type}(s_j)$), then the targets are swapped (since the truss elements are equal in function) and the reconfiguration proceeds. If that is not the case, the deepest predecessor of s_j is chosen, and moved as far as possible. (To find the *deepest predecessor*, elements whose reconfiguration path includes the current source element are found recursively. The deepest predecessor is found when previous elements satisfying this condition can no longer be found.) If, for some reason, a truss element cannot be moved all the way to its destination on the specified optimal path, it is stored there temporarily and added to the queue Q.

This process repeats until the reconfiguration is complete. The results on the

Algorithm 2 Exchange algorithm for trajectories to maintain the connectivity and the shortest paths

$S = truss(G_1 - G_2)$
$T = edge(G_2 - G_1)$
$P = path(S \rightarrow T)$ in G_m
deactivate T in G_m
$Q = \varnothing$
while $S \neq T$ **do**
  pick s_i such that $t_i \in T - S$ is connected to $S - T$ and $p_i \in S - T$
  trussSelected = *false*
  while *not* trussSelected **do**
    if $s_i \notin p_j$ $(j \neq i,\ j \in S - T)$ *or* $s_i \in Q$ **then**
      move s_i along p_i
      delete p_i
      activate the edges that are connected to t_i
      pull out s_i from Q (if $s_i \in Q$)
      trussSelected = *true*
    else
      choose s_j such that $s_i \in p_j$
      if $\exists s_j$ such that $s_i \in p_j$ and type(s_i)=type(s_j) **then**
        exchange t_i and t_j
        $p_i \leftarrow p_j(s_i \rightarrow t_j)$
        $p_j \leftarrow p_j(s_j \rightarrow s_i) + p_i(s_i \rightarrow t_i)$
        $i \leftarrow j$
      else
        pick s_k, the deepest predecessor of s_j
        move s_k along p_k
        if $s_k \neq t_k$ **then**
          $s_k \rightarrow Q$
        end if
        trussSelected = *true*
      end if
    end if
  end while
end while

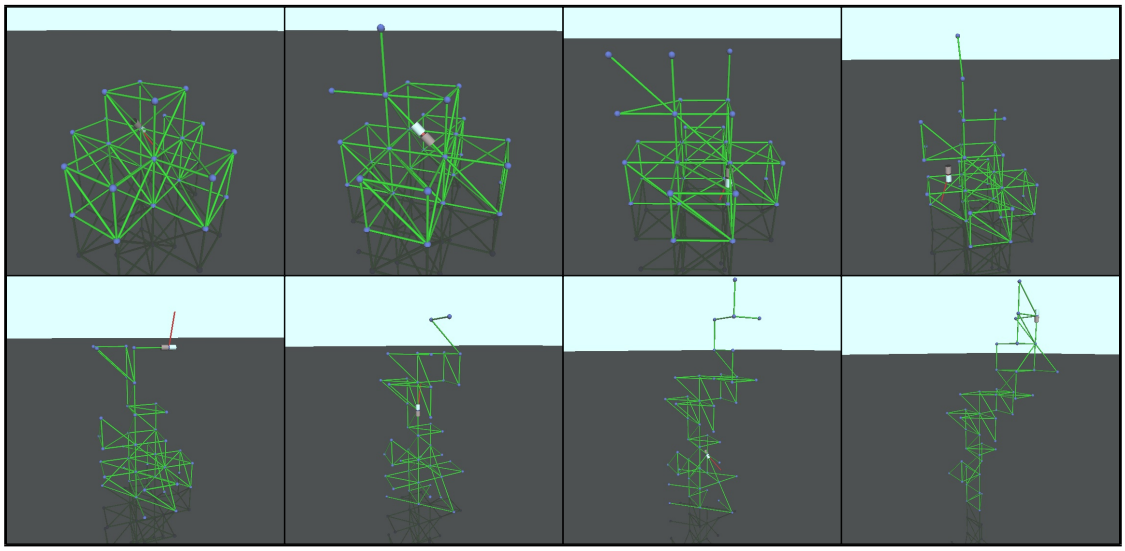

Figure 4.19 – *A rendering of a reconfiguration process.* These images show various steps of a simulated robot performing a reconfiguration process.

reconfiguration originally shown in figure 4.17 is shown in figure 4.18.

This algorithm has been demonstrated on a variety of structures [52]. One example reconfiguration process is shown via the artists rendering in figure 4.19. The algorithm does not yet take in to account structural soundness of the structure during reconfiguration or of the physical costs associated with various actual robot movements.

CHAPTER 5

FUTURE WORK, IMPACT, AND CONCLUSIONS

Before I conclude, I would like to discuss several future-facing aspects of machine metabolism. First, specific future work that can be done directly, based off of the work presented in this thesis. Second, big-picture future work that applies the work in this thesis, but may be many years off in development. Finally, I wish to discuss a few items that look philosophically at the possible future impacts from the ideas behind machine metabolism.

5.1 Future Work

5.1.1 Specific Future Work

Robot and Building Block Future Work

There is much room for future work on the robot, though the final acrylic-based design shows much promise of being a final solution.

First, the robot is more massive than is desired, and the rotation of the robot against gravity while in a cantilevered position only works occasionally. I believe that a reworking of the specifics of the robot design, particularly using the Objet 3D printer, could yield some great improvements.

Second, the printed gears discussed in section 2.3.3 yield some very interesting and elegant possibilities. The current robot is again too massive for the material

strength in the printed gears. Being able to replace all internal metal gears with printed ones gives a great deal of flexibility in gearing.

Thirdly, sensing and control. The current system is entirely open loop without any feedback. Simple contact sensors installed on the inside of the track would allow us to ensure proper truss engagement, and similar sensors on either end would allow us to ensure our position on the truss. Including on-board data acquisition in terms of an accelerometer, gyroscope, proximity sensors, and perhaps optical position sensors trained on the truss would allow us to progress down the road towards autonomy. Using external optical 3D position sensors, such as OptiTrack, would also assist in this direction.

Fourth, control program integration. The current control program is command-line driven and rudimentary. Completing high-level autonomous control and being able to include some of the reconfiguration algorithms discussed in chapter 4 would see the conclusion of an excellent testbed.

Finally, actual reconfiguration consisting of several truss elements should be achieved.

Additionally, some work is required regarding the building blocks. Section 2.3.3 discusses the 3D printed struts which have been preliminarily developed. Additionally, the final development of the connecting mechanism and design of the nodes needs to be done.

Algorithmic Future Work

The primary algorithmic goal is to combine the physical robotic testbed with the algorithmic results. Currently, several limitations remain before this is complete.

First, none of the algorithms—reconfiguration or planning—take in to account the nodes of the truss structure. This was done to simplify computation, but is clearly not possible to perform this way in the physical system. Additionally, since the robot cannot manipulate or transport nodes separately, the planning algorithms need to determine how to transport nodes attached to truss struts.

Second, little work—apart from a few results designing bridges with a physics simulator—has been done to consider the physical characteristics of the truss, particularly during the process of reconfiguration. This is clearly an area that needs to be explored further.

Third, related, the planning algorithms need to be changed to reflect physical characteristics of the robot, such as power consumption in various modes, preferential types of movements, limitations in carrying truss elements through a truss structure, etc.

Fourth, a functional language specification could be developed to allow end users to build functional requirements from an easy-to-use library. Combined with a system to input initial truss structures, very interesting reconfiguration results could be obtained with a minimum amount of work.

Fifth, though the results of the generative encoding for truss structure reconfiguration did not yield superior results in my analysis, I expect that there is a real possibility for generative encodings to yield interesting and effective results,

as they have been shown to in other structural contexts.

Finally, the algorithms can be expanded to consider both communities of robots (as has often been considered in modular robotics) and low-level local rules, such as "if a 45-degree joint is found in a certain orientation, add a truss element vertically to the top of it if one does not exist." (Similar ideas have been considered in other contexts, such as in Estévez [15].) Finding local rules such as this that would produce a globally desired behavior would be particularly reminiscent of biological metabolism.

5.1.2 Big Picture Future Work

Other considerations of the ideas presented here are possible and should be considered. Some of these ideas are long-term, and unlikely to be accomplished in the near-term. Others are different conceptions and different analogies for the same capabilities we propose here.

Alternative Analogies

Alternative analogies to "metabolism" could also be applied for other inspiration. For instance, we could consider the robots as enzymes or catalysts, necessary for reactions between the constituent truss elements that would not or could not happen otherwise. Physiological respiration is the process by which oxygen is brought to various organs from the outside air; if we consider the basic building blocks of our process as oxygen and their destination in some reconfiguration a tissue, we could imagine a set of robots stochastically carrying building blocks

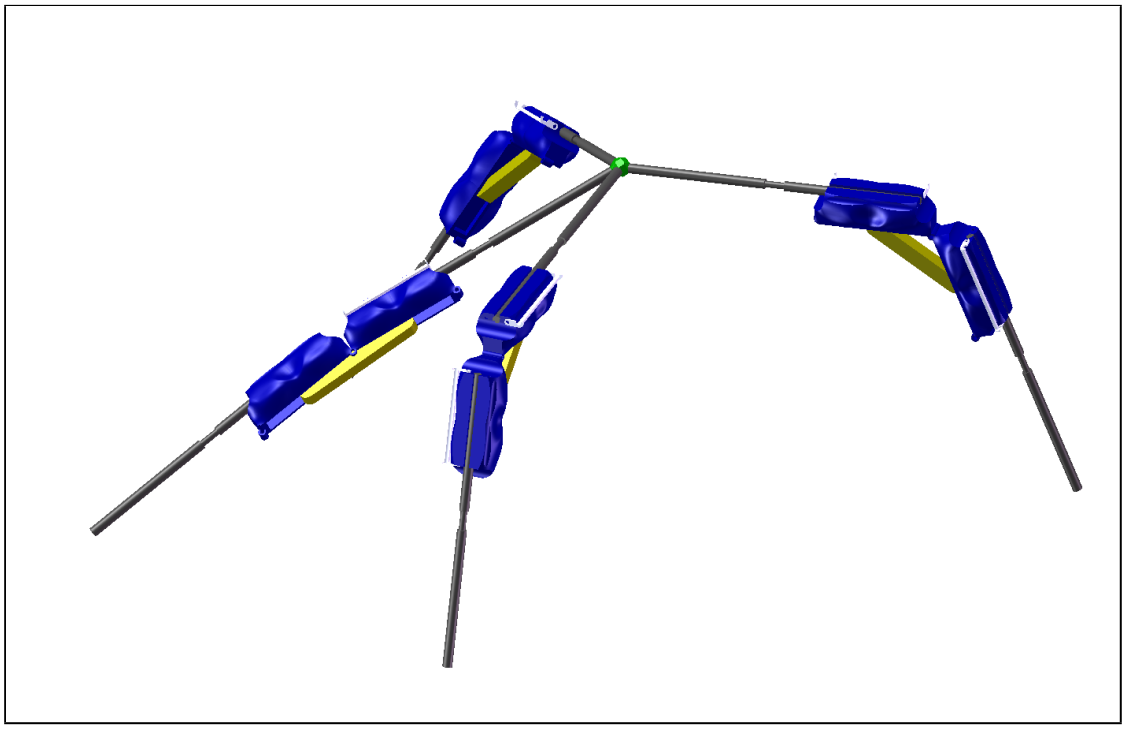

Figure 5.1 – *A simple mobile robot combination robot.* A small number of hinge robot and passive elements could become an active robot of their own.

about a structure in search of locations in which to place them, in a very similar way to the respiration process. Finally, higher level analogies, such as that of a colony of bees using basic building blocks to create, repair, and reconfigure their hive may also be insightful.

Robot-Structure Integrations

These images demonstrate some ways in which the active robot can be combined with passive structures to form a larger, active robot. Such applications could dramatically increase the functional specifications available to a metabolistic system.

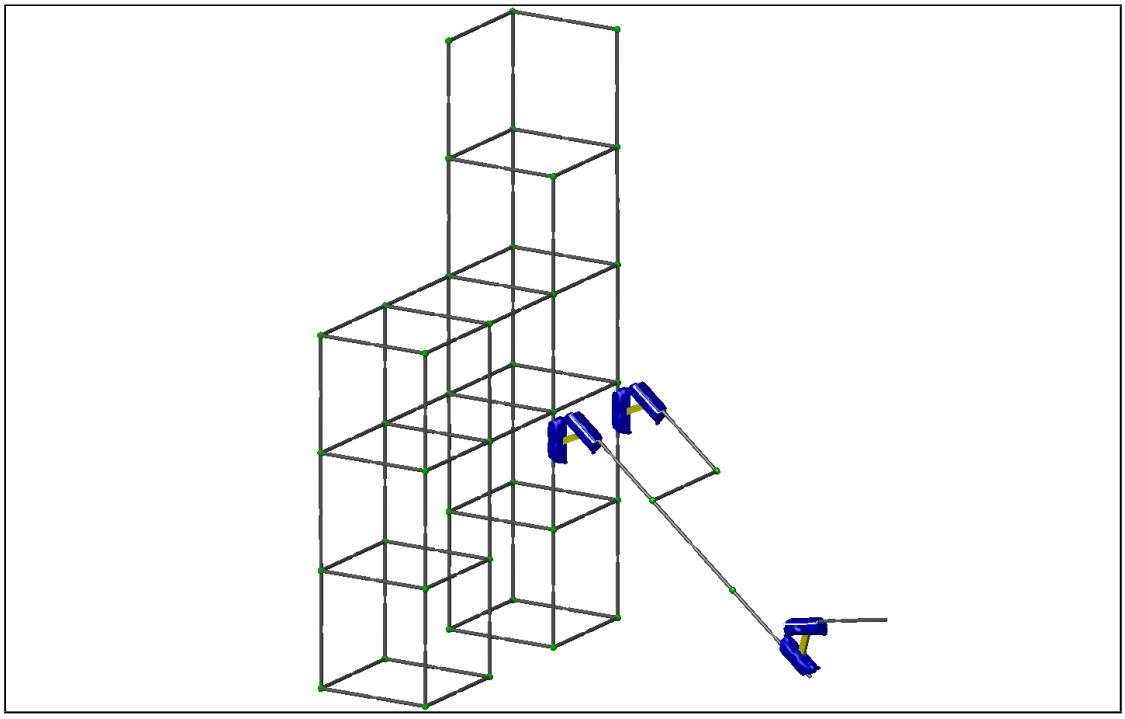

Figure 5.2 – *Hinge robots assisting each other in reconfiguration.* The reconfiguration process could be completed more quickly with hinge robots cooperating and creating shortcuts. Here, two hinge robots assist a third to the top of a structure.

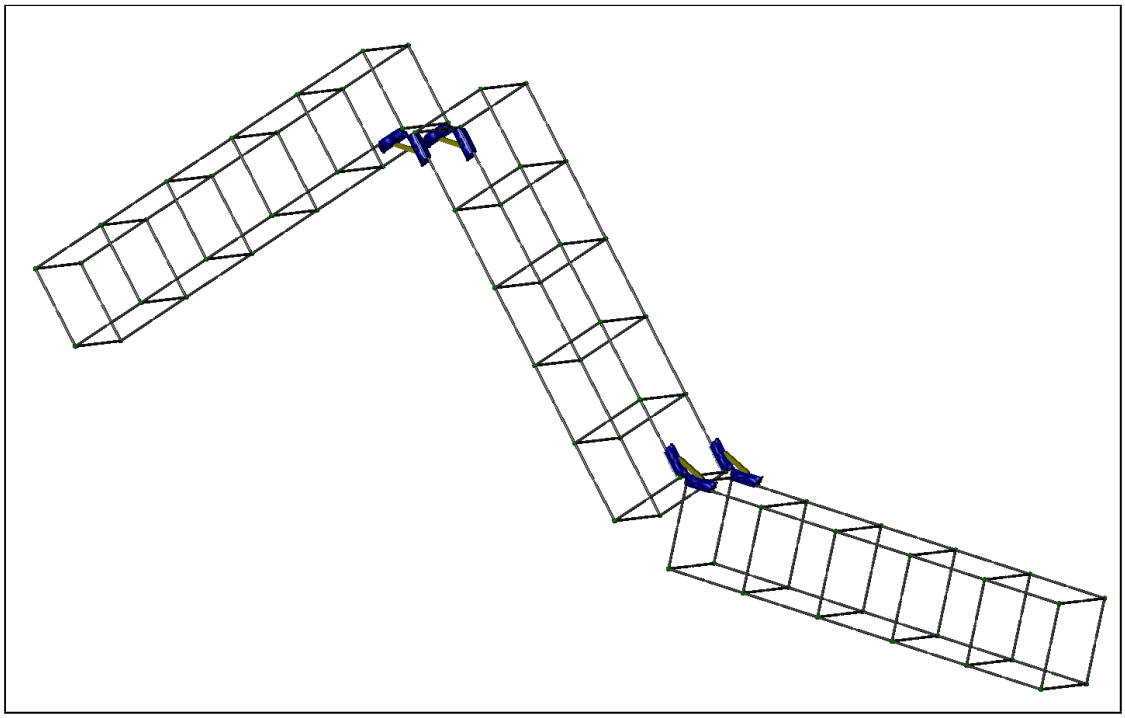

Figure 5.3 – *A more complicated hinge robot.* Even larger active structures could be produced.

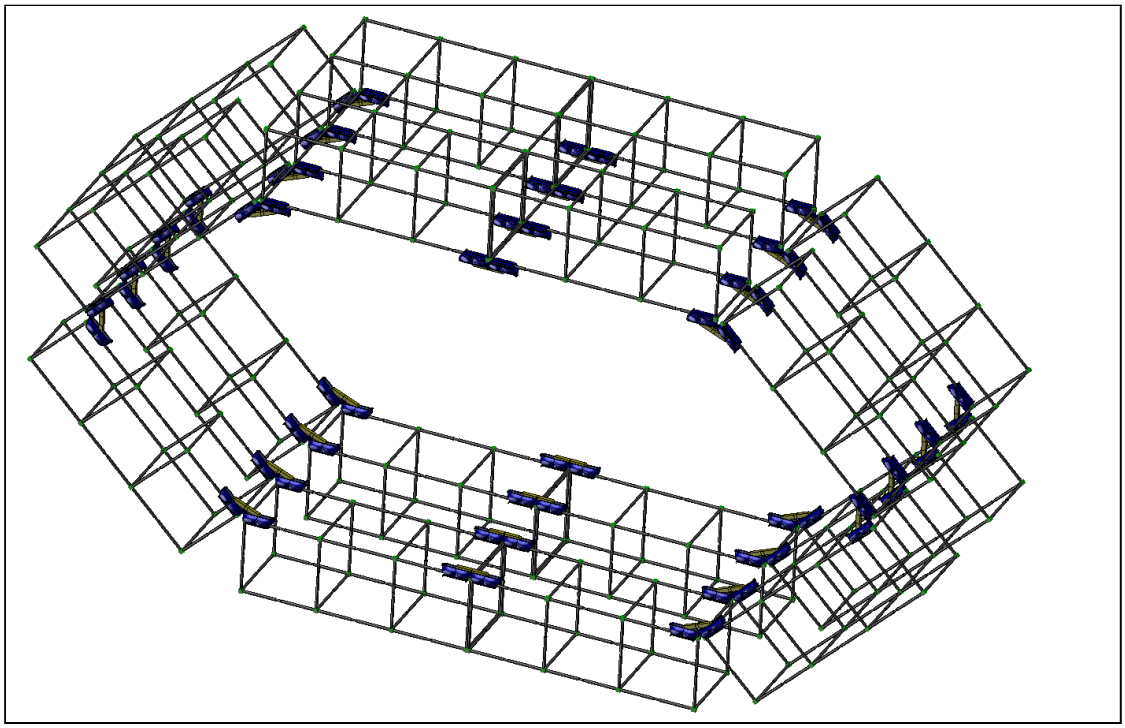

Figure 5.4 – *A tread robot composed from a metabolistic system.* More complex systems such as a metabolic tread may allow rolling motion from hinge robots and passive elements.

5.2 Future Impact

Section 1.3 discussed some of the properties of biological metabolism that are useful in engineering applications. These properties, if fully realized, could affect a variety of industries, technological applications, and event ethical considerations in a variety of ways.

Manufacturing

Within the manufacturing world, automated design capabilities could allow manufacturers to have products redesigned on-the-fly to variations in available raw materials. This would help prevent factory shutdowns due to supply chain problems—without necessitating extra human input.

Rather than replace human engineers, automated design can augment engineering design processes and stimulate creative thinking. Automated systems have been used as tools to augment human invention for decades, and new approaches (such as machine metabolism) can hopefully continue to do so.

Recycling

Making more items out of reusable modular elements—using a basic set of building blocks with various capabilities—would allow unused items to be re-structured into new and more useful things. Broken parts could be found and replaced more easily. This would allow greater and more efficient material conservation, and promote reuse as much as possible.

Additionally, designs accomplishing new things with these basic building blocks could be distributed digitally, not only reducing the waste of raw materials that is currently prevalent in our culture but also reducing distribution costs for many everyday items.

Disaster recovery

Constructing dwellings and other infrastructure with robotically reconfigurable building blocks would allow not only more dynamic infrastructure but also the ability to reconstruct infrastructure autonomously. Robotic systems could analyze the debris in the wake of natural (and other) disasters and determine what building blocks are still viable. Using the available raw building blocks, then, autonomous design and building could be performed to help communities faced with disaster begin to operate—at least on a minimal level—more quickly than ever before possible.

Space exploration

Clearly, the ability to reconfigure into many functional configurations out of a small supply of raw materials would be a great boon in space exploration, where weight and space are severely constrained. The ability to perform a variety of functions out of a reduced amount of raw materials—especially if it could be adapted to new and unforeseen functional requirements—could make many forms of space exploration possible.

Ethics and philosophy

Machine metabolism also brings its share of ethical questions and considerations.

The first ethical question will likely be how automated design affects human engineers. Likely, automated design will serve as another tool to complement and stimulate the human engineer's creativity. However, there may need to be steps taken to ensure that activities vital and central to human existence—such as *creating*—remain viable human activities in the face of increasing automation.

Questions of the nature of creativity will also need to be explored, as robotic systems become more and more capable of exhibiting what is arguably creative behavior.

Finally, on a completely different note, a few practical ethical considerations will need to be dealt with before widespread adoption of metabolistic systems for everyday times. Due to the ease of reconfiguring and reusing basic building blocks, theft becomes a completely new proposition: not only can the thieves hide their evidence, but they can obtain something more useful to them in return.

CHAPTER 6

CONCLUSIONS AND CONTRIBUTIONS

6.1 Conclusions

As mentioned in the introduction, machine metabolism, consisting of robotically reconfigurable building blocks with a reconfigurator robot capable of autonomous design and reconstruction based off of functional requirements, is a long-term goal. Much work has yet to be done before its promise can be fulfilled. Nonetheless, notable progress has been made. A robot capable of some open-loop truss traversal has been developed, and shows promise of being able to autonomously reconfigure trusses with future work. Truss elements were designed specifically for this robotic reconfiguration task. Finally, several algorithms discussing the reconfiguration design and planning were presented.

6.2 Contributions of this Thesis

The contributions of the work described in this thesis are the following:

- Designed and constructed a truss structure designed specifically for robotic manipulation and random-access.
- Designed and constructed a hinge robot for traversal of this truss structure and preliminary manipulation.
- Demonstrated two evolutionary algorithms capable of reconfiguring a source truss structure into a destination truss structure based purely upon

functional requirements.

- Demonstrated a optimal path planning algorithm for performing the reconfiguration of one truss structure into a destination structure.

6.3 Personal Contributions

I have included and mentioned the work of several others in this thesis; I will now explain my specific contributions to each section and idea discussed herein. At all points in the process, Hod Lipson provided helpful feedback and advice; the EFRI collaborators Daniela Rus, Eric Klavins, and Mark Yim also provided insightful discussions.

The overview of machine metabolism and related previous work in chapter 1 were done by me, as was the overview of building block morphologies and the final designs discussed in chapter 2. Franz Nigl contributed to the development of the geared truss elements discussed in section 2.3.2.

In chapter 3, the work presented in sections section 3.2 to section 3.4 were mine, with helpful discussions with Hod Lipson and Jonathan Hiller. For section 3.5 through the completion of the chapter, Franz Nigl and I worked closed on the development of the robot. I focused primarily on the rotational mechanism and computer programming aspects, and Franz focused primarily on the translational mechanism and hinge, though there was much overlap. Stephane Constantin designed and programmed the PCB and microcontroller, and gave much helpful advice on electronics questions.

Chapter 4 demonstrated three algorithms. The algorithm described in sec-

tion 4.1 was done by Daniel Lobo and Hod Lipson, with my input on the physical properties of the truss structure. The algorithm described in section 4.3 was done by Seung-kook Yim and Daniela Rus, with input from me regarding physical properties of the truss structure and the hinge robot.

Finally, the ideas discussed in chapter 5 are mine.

6.4 Lessons Learned

Finally, I would like to call out a selection of lessons learned in the development of this robot, as an aim to assist future development. These include:

- Consider carefully the construction of the modular elements. Manufacturing carbon fiber is difficult, as is attaching threaded fasteners to it. If the strength is sufficient, 3D printing the elements is a good solution, or perhaps a good prelude to producing injection-molded nylon parts.
- 3D printing the module elements also raises interesting possibilities: custom threaded fasteners that only require 1/2 a turn to fasten, self-alignment possibilities (both during insertion and rotationally between halves of truss element), and custom gear teeth as discussed in section 2.3.3. Perhaps a "threaded" fastening mechanism that would not require gaps in the truss elements would be a possibility.
- Attempting to reduce weight on the robot is critical, as the ability of the servos to perform is directly related to weight.
- Keeping gears properly engaged on the truss elements is difficult, due to the flexibility of the robot materials. I suspect that varying the actuation

(re-arranging the cams, using screws to engage/disengage, changing the position of contact between the drive gears and the truss elements, etc.) could yield benefits. Also, making larger and deeper gear teeth would help alleviate this issue.

- In addition to the above two points, I expect that re-designing the robot with the capabilities of the Objet 3D printer in mind would be fruitful, allowing a smaller, lighter, and simpler robot.
- Friction drive in this situation is a poor solution. It is not worth the effort necessary to get it working.
- Internal friction is a continuing challenge, especially in the rotational mechanism. The rotational mechanism produces many times more torque than theoretically necessary to rotate a cantilevered robot about the truss against gravity, but still struggles with that motion. This is at least partially due to the truss elements being in contact with the acrylic and Delrin robot parts.
- Delrin is a laser cut-able material that is similar to acrylic but is less brittle. This is helpful when acrylic parts are tending to break.
- Futaba servos have a 6v internal cutoff circuitry, and require a voltage regulator to work reliably. For instance, with the 7.4v batteries we used, they would tend to turn off in high load situations.
- The attachments shafts and gears that ServoCity sells for Futaba and HiTec servos are worth exploring; much time was spent attaching mechanical devices to the non-standard GWS splines.
- Controlling the continuous rotation servos reliably has been difficult. Perhaps replacing the potentiometer with a soldered-in resistor, gluing the

potentiometer, or bypassing the electronics and controlling it as a gearmotor would be more appropriate.

- I believe a mechanism that would allow the hinge to pivot such that the pivot point is located at a node could be devised. This would simplify several alignment issues in truss traversal.

BIBLIOGRAPHY

1: R. Aracil, RJ Salateru, and O. Reinoso. A climbing parallel robot. *Robotics & Automation Magazine, IEEE*, 13(1):16–22, 2006.

2: C. Balaguer, A. Giménez, JM Pastor, VM Padrón, and M. Abderrahim. A climbing autonomous robot for inspection applications in 3D complex environments. *Robotica*, 18(03):287–297, 2000.

3: C. Balaguer, A. Gimenez, and A. Jardon. Climbing Robots' Mobility for Inspection and Maintenance of 3D Complex Environments. *Autonomous Robots*, 18(2):157–169, 2005.

4: H. Bilimoria, D. Wynne, I. Ralph, and A. Topolcsanyi. Self aligning screw fasteners and engagement thereof, 2000. Patent WO 00/63566.

5: R.A. Brooks, P. Maes, M.J. Mataric, and G. More. Lunar base construction robots. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, pages 389–392 vol.1, Jul 1990. doi: 10.1109/IROS.1990.262415.

6: Carlos and Alan Christiansen. Multiobjective optimization of trusses using genetic algorithms. *Computers and Structures*, 75(6):647–660, 2000.

7: M. Csete and J. Doyle. Bow ties, metabolism and disease. *Trends in Biotechnology*, 22(9):446–450, 2004.

8: Carrick Detweiler, Mersette Vona, Yeoreum Yoon, Seung-Kook Yun, and D. Rus. Self-assembling mobile linkages. *Robotics & Automation Magazine, IEEE*, 14(4):45–55, 2007.

9: M.A. Diftler and I.D. Walker. Determining alignment between threaded parts using force and position data from a robot hand. *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, 2:1503–1510 vol.2, Apr 1997. doi: 10.1109/ROBOT.1997.614352.

10: M.A. Diftler and I.D. Walker. Experiments in aligning threaded parts using a robot hand. *Robotics and Automation, IEEE Transactions on*, 15(5):858–868, Oct 1999. ISSN 1042-296X. doi: 10.1109/70.795791.

11: Myron A. Diftler, William R. Doggett, Joshua S. Mehling, and Bruce D. King. Reconfiguration of eva modular truss assemblies using an anthropomorphic robot. In Mohamed S. El-Genk, editor, *SPACE TECH.&APPLIC.INT.FORUM-STAIF 2006: 10th Conf Thermophys Applic Microgravity; 23rd Symp Space Nucl Pwr 4th Conf Human/Robotic Tech &Nat'l Vision for Space Explor.; 4th Symp Space Coloniz.; 3rd Symp on New Frontiers &Future Concepts*, volume 813, pages 992–999. AIP, 2006. doi: 10.1063/1.2169280. URL `http://link.aip.org/link/?APC/813/992/1`.

12: W. Doggett. Robotic assembly of truss structures for space systems and future research plans. *Aerospace Conference Proceedings, 2002. IEEE*, 7, 2002.

13: W. S. Dorn, R. E. Gomory, and H. J. Greenberg. Automatic design of optimal structures. *Journal de Mecanique*, 3:25–52, 1964.

14: S. Dubowsky and P. Boning. The coordinated control of space robot teams for the on-orbit construction of large flexible space structures. In *Proceedings of the 2007 IEEE International Conference Robotics and Automation, Special Workshop on Space Robotics*, April 2007.

15: Nicholas Estévez. Functional blueprints: A dynamical systems approach

to structure representation. Master thesis, Mechanical Engineering Dept., Cornell University, 2007.

16: J. Everist, K. Mogharei, H. Suri, N. Ranasinghe, B. Khoshnevis, P. Will, and Wei-Min Shen. A system for in-space assembly. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2356–2361 vol.3, 2004.

17: K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu. Miche: Modular shape formation by self-disassembly. *The International Journal of Robotics Research*, 27(3-4):345, 2008.

18: SC Goldstein, JD Campbell, and TC Mowry. Programmable matter. *Computer*, 38(6):99–101, 2005.

19: David Alan Hjelle and Hod Lipson. A robotically reconfigurable truss. In *In Proceedings of ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots (ReMAR 2009)*, June 2009.

20: John Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975. ISBN 0472084607.

21: GS Hornby and JB Pollack. The advantages of generative grammatical encodings for physical design. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, 2001.

22: D. King. Space servicing: Past, present and future. In *Proc. of the 6 th International Symposium on Artificial Intelligence and Robotics & Automation in Space (i-SAIRAS 2001), Montreal, Canada*, 2001.

23: Hubert Konzorr. Pedal for bicycles and the like. US Patent 3798997, May 1973.

24: Keith D. Kotay and Daniella L. Rus. Navigating 3d steel web structures with an inchworm robot. *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, 1:1178–1185 vol.2, 1996. doi: 10.1109/IROS.1994.407465.

25: HW Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics*, 52(1), 2005.

26: Wei Lingyun, Zhao Mei, Wu Guangming, and Meng Guang. Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. *Computational Mechanics*, 35(5):361–368, 2005.

27: Daniel Lobo. Matsprings: Matlab 3d spring tuss simulator, 2008. URL `http://www.geb.uma.es/matsprings.html`.

28: Daniel Lobo, David Alan Hjelle, and Hod Lipson. Reconfiguration algorithms for robotically manipulatable structures. In *In Proceedings of ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots (ReMAR 2009)*, June 2009.

29: S.W. Mahfoud. Crowding and preselection revisited. In *Proceedings of the 1992 Parallel Problem Solving from Nature conference*, 1992.

30: C. Melhuish and O. Holland. Getting the most from the least: lessons for the nanoscale from minimal mobile agents. In *Artificial Life V*, volume 100, Nara, Japan, 1996.

31: C. Melhuish, J. Welsby, and C. Edwards. Using templates for defensive wall

building with autonomous mobile ant-like robots. In *Proceedings of Towards Intelligent Autonomous Mobile Robots*, volume 99, 1999.

32: Y. Murase, Y. Ito, Y. Mitsui, et al. Self-aligning bolt, September 19 2000. US Patent 6,120,227.

33: MC Nechyba and Y. Xu. Human-robot cooperation in space: SM^2 for new spacestation structure. In *Robotics & Automation Magazine, IEEE* Xu et al. [48], pages 4–11.

34: G. Pritschow, M. Dalacker, J. Kurz, and J. Zeiher. A mobile robot for on-site construction of masonry. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 3, pages 1701–1707 vol.3, Sep 1994. doi: 10.1109/IROS.1994.407628.

35: S. Rajeev and C. S. Krishnamoorthy. Genetic algorithm-based methdologies for design optimization of trusses. *Journal of Structural Engineering*, 123(3): 350–358, 1997.

36: Daniela Rus, Eric Klavins, Hod Lipson, and Mark Yim. Controlling the autonomously reconfiguring stochastic factory. NSF EFRI-ARESCI Grant #0735953, August 27 2007.

37: Daniella L. Rus. Drl - shady3d, 2009. URL `http://groups.csail.mit.edu/drl/wiki/index.php/Shady3D`.

38: K. Senda, Y. Murotsu, A. Mitsuya, H. Adachi, S. Ito, J. Shitakubo, and T. Matsumoto. Hardware experiments of a truss assembly by an autonomous space learning robot. *Journal of Spacecraft and Rockets*, 39(2):267–273, 2002.

39: L. Simons. Self aligning screw, December 23 1986. US Patent 4,630,985.

40: M. Tavakoli, MR Zakerzadeh, GR Vossoughi, and S. Bagheri. A hybrid pole climbing and manipulating robot with minimum DOFs for construction and service applications. *Industrial Robot: An International Journal*, 32(2):171–178, 2005.

41: Y. Terada and S. Murata. Automatic assembly system for a large-scale modular structure - hardware design of module and assembler robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2349–2355 vol.3, 2004.

42: J. Wawerla, G.S. Sukhatme, and M.J. Mataric. Collective construction with multiple robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, pages 2696–2701, 2002.

43: J. Werfel. Building blocks for multi-robot construction. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*. Springer, 2004.

44: J. Werfel, Y. Bar-Yam, and R. Nagpal. Building patterned structures with robot swarms. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1495. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005.

45: J. Werfel, Y. Bar-Yam, and R. Nagpal. Construction by robot swarms using extended stigmergy. *AI Memo AIM-2005-011, MIT CSAIL, Cambridge, MA, 2005*, 2005.

46: J. Werfel, Y. Bar-Yam, D. Rus, and R. Nagpal. Distributed construction by mobile robots with enhanced building blocks. In *Proceedings of 2006 IEEE International Conference on Robotics and Automation, Orlando, USA*, 2006.

47: Justin Werfel and Radhika Nagpal. Three-dimensional construction with mobile robots and modular blocks. *International Journal of Robotics Research*, 27(3-4):463–479, 2008.

48: Y. Xu, HB Brown Jr, M. Friedman, and T. Kanade. Control system of the self-mobile space manipulator. *Control Systems Technology, IEEE Transactions on*, 2(3):207–219, 1994.

49: M. Yim, Y. Zhang, and D. Duff. Modular robots. *Spectrum, IEEE*, 39(2):30–34, 2002.

50: M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, Hod Lipson, E. Klavins, and G.S. Chirikjian. Self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, page 44, 2007.

51: Yeoreum Yoon and D. Rus. Shady3d: A robot that climbs 3d trusses. *Robotics and Automation, 2007 IEEE International Conference on*, pages 4071–4076, April 10-14 2007. ISSN 1050-4729. doi: 10.1109/ROBOT.2007.364104.

52: Seungkook Yun, David Alan Hjelle, Eric Schweikardt, Hod Lipson, and Daniela Rus. Planning the reconfiguration of grounded truss structures with truss climbing robots that carry truss elements. *Robotics and Automation, 2009. ICRA 2009. Proceedings 2009 IEEE International Conference on*, May 2009.

53: Cem Ünsal and Pradeep K. Khosla. Mechatronic design of a modular self-reconfiguring robotic system. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 1742–1747 vol.2, 2000. doi: 10.1109/ROBOT.2000.844847.

54: Cem Ünsal, Han Kiliççöte, and Pradeep Khosla. A modular self-reconfigurable bipartite robotic system: Implementation and motion planning.

Autonomous Robots, 10(1):23–40, 01 2001. URL http://dx.doi.org/10.1023/A:1026592302259.