UNIVERSITY OF CALIFORNIA
SANTA CRUZ

DESIGN, BUILDING, TESTING, AND CONTROL OF SUPERball: A TENSEGRITY ROBOT TO ENABLE NEW FORMS OF PLANETARY EXPLORATION

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY
with an emphasis in ROBOTICS AND CONTROL

in

COMPUTER ENGINEERING

by

Jonathan Bruce

December 2016

Table of Contents

List of Figures **vi**

List of Tables **xiv**

Abstract **xv**

Dedication **xvii**

Acknowledgments **xviii**

1 Introduction **1**
1.1 Motivation 1
1.2 Scope of Dissertation 3

2 Literature Review **6**
2.1 Tensegrity Structures 6
2.2 Prior Work in Tensegrity Robotics Design 7
2.3 Tensegrity Robotics for Space Exploration 8
2.4 Tensegrities as Soft Robots with Morphological Computation Capabilities 9
2.4.1 Opportunities in Morphological Computation 10
2.4.2 Methods for Controlling Soft Robots 10
2.4.3 Planning for Tensegrities 12
2.5 Low-level Control for Tensegrity Robots 13
2.6 High Level Control for Tensegrity Robots 15
2.6.1 Artificial Neural Networks 15
2.6.2 Path Planning Through a Sampling-Based Motion Planner 17

3 Modeling and Model Validation **19**
3.1 Euler-Lagrange Model 21
3.2 NASA Tensegrity Robotic Toolkit Model 24

3.3 Detailed Impact Simulations and Cross Validation Using Two Simulators 25
3.4 Simulated Drop Tests and Payload Protection 27
3.5 Landing Orientation Studies 29
3.6 Conclusions from Simulation Experiments 30
3.7 Validation of NTRT and Real Hardware Prototypes 32

4 SUPERball v1 Design 33
4.1 Mechanical 34
4.1.1 Spring Holder 35
4.1.2 Battery Holder 35
4.1.3 Motor and Electronics 36
4.1.4 Cable Actuation and Routing 37
4.1.5 Ground Contact 39
4.2 Electrical 39
4.2.1 Motor Board 40
4.2.2 Sensor Board and Beagle Bone Black 40
4.2.3 Power Board 41
4.3 Communication 43
4.3.1 CAN Bus 43
4.3.2 WiFi and the Robot Operating System 44
4.4 First Step 45

5 State Estimation 47
5.1 Ranging Setup and Calibration 47
5.1.1 Sensor Operation 48
5.1.2 Ranging Setup 51
5.1.3 Calibration 52
5.2 Filter Design 53
5.2.1 Dynamic Modeling 54
5.2.2 UKF Implementation 56
5.3 Filter Evaluation 58
5.3.1 Experimental Setup 58
5.3.2 Local Trajectory Tracking 59
5.3.3 Global Trajectory Tracking 60

6 Control for SUPERball Locomotion 62
6.1 Basic Locomotion Concepts for a Icosahedron Tensegrity Robot . 62
6.1.1 SUPERball Acutation Pattern 64
6.1.2 Basic Steering Controller 66
6.2 Hand-Tuned Stepwise Controller 67
6.3 Machine Learning Enabled Controllers 68
6.3.1 Crater Escape Controller 69

6.3.2 Co-Evolutionary Learning for an Open Loop Controller . . 72
6.3.3 Mirror Descent Guided Policy Search 76

7 Conclusion and Future Work 86
7.1 Contribution 86
7.2 Future Work 87
7.2.1 Re-Design of SUPERball 87
7.2.2 Extending Controls for a Fully Actuated SUPERball v2 . . 88
7.3 Conclusion 90

Appendix A SUPERball v1 Design Evaluation 91
A.1 Mechanical Evaluation 91
A.1.1 Cable Routing 91
A.1.2 Cable Tension 94
A.1.3 Cable Material 96
A.1.4 Force Sensors 98
A.2 Electrical Evaluation 101
A.2.1 Sensor Board 101
A.2.2 Power Board 101
A.2.3 Motor Board 102
A.3 Communications Evaluation 103
A.3.1 CAN Bus 103
A.3.2 WiFi Network 103

Bibliography 105

List of Figures

1.1 Tensegrity structures are composed of pure compression and tension elements. They can be lightweight, reliable, deployable, and efficient to manipulate. **Mission Scenario** - Tightly packed set of tensegrities, expand, spread out, fall to surface of moon, then safely bounce on impact. The same tensegrity structure which cushioned the landing is then used for mobility to explore moons such as Titan and small asteroids. 2
1.2 Tensegrities are composed of pure tension and pure compression elements (e.g. cables and rods) as seen in this picture of a tensegrity robot from our physics based tensegrity simulator. They are lightweight, energy-efficient and robust to failures. 3

2.1 Overview of research topics discussed. An arrow indicates a topic that can contribute to the development of another. 11
2.2 An example of a trajectory with two forward progagations and their uncertainty. Possible future configurations are shown as transparent versions of SUPERball. (figure courtesy of Zakary Littlefield). 18

3.1 SUPERball, fully assembled, in the NASA Ames Research Center Roverscape. 19
3.2 SUPERball with a payload modeled within NTRT. 25

3.3 SUPERball simulated in Open Scene Graph. Note that this graphical display of NTRT does not support cable visualization. 26
3.4 NTRT vs EL: Vertical Position 27
3.5 NTRT vs EL Vertical Velocity 28
3.6 NTRT vs EL Vertical Acceleration 29
3.7 **Landing Forces Study**. *This shows how rod length impacts maximum deceleration of the payload and the maximum forces experienced by the tension cables. All tests were conducted with a landing velocity of 15 m/s onto a hard surface.* 30
3.8 *Heat map of the maximum acceleration that the payload encounters for all possible landing orientations. Black areas are safe, colored areas are where the payload does not meet one or both success criteria.* 31

4.1 Fully assembled Modular Tensegrity Robot images on SUPERball, except for the ground contact caps which will mount to the every end of each rod. 34
4.2 Time lapsed stills of spring compression in the MTR spring holder. Note that these stills are from a previous version of the MTR. . . 36
4.3 Cable routing roller guide within an MTR. 38
4.4 Pictures of each of the main micro-controller boards on an MTR. (a) Motor Board, (b) Sensor Board with DWM module, (c) Power Board without nRF24l01 wireless chip. 43
4.5 This is a connection diagram for power and communication for a MTR on SUPERball. 44
4.6 A simplified representation of how messages are passed within the SUPERball ROS network. 45

4.7 SUPERball performing a single face-change movement, from one equilateral triangular face to another. The robot begins with all MTRs of the red triangle touching the ground. Then, SUPERball retracts the yellow-highlighted cable on the red triangle, inducing movement. Frame 2 shows SUPERball halfway through the movement with only two points of contact on the ground. Finally, frame 3 shows SUPERball at the end, with all 3 points of the blue triangle in ground contact. 46

5.1 Basic symmetric double-side two-way ranging packet exchange. Modules 1 and 2 exchange 3 packets (*poll*, *response*, and *final*). Module 2 then estimates the distance between the modules based on the local timestamps. 48

5.2 Packet exchange between 4 modules for bidirectional pairwise and broadcast ranging. Timed broadcast messages allow for efficient ranging with a large number of modules. 51

5.3 Block diagram of data flow within the system. Red signals are passed as ROS messages and blue signals are passed using the ranging modules. Note that each rod contains two ranging sensors located at each end of the rod. The gray control strategy block represents a to-be-designed state-feedback control strategy. 57

5.4 Visualization of the UKF output. SUPERball sits in the middle of the plot surrounded by 8 ranging base stations. Lines between the robot and the base stations indicate valid ranging measures during this timestep. 58

5.5 Position plotted through time for both end cap 1 and end cap 2. The thin line represents the position output measured by the camera tracking system, and the bold line represents the position output from the UKF filter. As expected, there is a time domain lag between the measured and estimated positions. 59

5.6 Top down view of the triangular faces to which the robot transitions during the global trajectory tracking experiment for various setting of the state estimator. The small inset illustrates the movement of the robot. The line shows the estimated center of mass (CoM) using the *full* settings. Finding the initial position (origin) is hard for all settings, and without the IMUs the estimator does not find the correct initial face. After a first roll, tracking becomes more accurate. The offsets $\boldsymbol{o}$ have a minimal impact, which indicates that the calibration routine is sufficiently accurate. 60

5.7 X and Y position of end cap 12 as a function of time for the various estimator settings. The end cap was initially off the ground and touches the ground after the first roll. This is not tracked correctly when the IMUs are disabled. The system works as expected when 4 base stations ranging sensors are disabled, but with slower convergence and more noise on the robot's position. Around 60 s there's a spurious IMU value from which the state estimator recovers. . 61

6.1 This is XY data of the bottom triangle of a NTRT simulation of SUPERball performing a single face transition by changing only one side of the bottom triangle. No other cable on the system is being actuated during this simulation. The blue triangle is the bottom triangle at the start of the simulation. The centralized circle represents the center of mass (CoM) of the entire simulated SUPERball and the dots represents how the CoM moves through time. The red triangle is the configuration where the CoM moves out of the bottom triangle and the robot begins to transition to another face. 63

6.2 SUPERball performing a single face-change movement, from one equilateral triangular face to another. The robot begins with all MTRs of the red triangle touching the ground. Then, SUPERball retracts the yellow-highlighted cable on the red triangle, inducing movement. Frame 2 shows SUPERball halfway through the movement with only two points of contact on the ground. Finally, frame 3 shows SUPERball at the end, with all 3 points of the blue triangle in ground contact. 64

6.3 This is the actuation pattern used on SUPERball. There are 8 equilateral triangles, shown in either red or blue. Each red triangle represents a face with only one cable actuated. These actuated cables are denoted by the yellow double arrows. Each blue triangle represents a face with actuators on all cables. The basic forward rolling of SUPERball has the robot landing with red triangles on the ground during locomotion. Figure (a) shows each triangle highlighted on SUPERball and figure (b) shows the forward locomotion pattern, or walking pattern, of SUPERball. 65

6.4 This figure shows the tracked center of mass of a simulated SUPERball utilizing the basic steering controller. Each fully actuated equilateral triangle was squeezed the same amount for the left and right steering. This controller has no feed back and does have a slight bias toward turning right more than left. Also, it is noted that the unbiased controller tends to the left, because of no optimization on direction was used during the learning of the controller. It should be noted that this figure only shows an academic example and not a fully functional controller. 66

6.5 Time based state machine which automates the hand-tuned stepwise controller. Timers are used to allow for dynamic settling before the next action is taken. 68

6.6 Change in length of cables when one (13th) is pulled to 0.5 meters while the others are kept at the same rest length. Grey bars show original length, red show final length. While the robot is at the exact same orientation, the actual lengths of the cables change in a non-linear way. Some of the cables shorten due to the tension introduced by cable 13, and some of the cables relax. 69

6.7 (a) Generations 0 for the escape hole/crater policy. The displacements reached in 1000 independent samples using Monte Carlo generated control policies. (b) Generations 1 for the escape hole/crater policy. The displacements reached in 1100 independent samples using control policies dictated by the successful samples from Generation 0. The values in the set of sine wave parameters from Generation 1 samples were each centered around one of the successful corresponding sine wave parameters in Generation 0 control policies. These Generation 1 values were then modified to be within 0.5% of their respective Generation 0 values with the goal of optimizing Generation 0 control policies. 71

6.8 (a) 100 Samples of Tensegrity Structures with 24 Actuators. By reducing the number of functioning actuators on our simulated tensegrity, the limitations of tensegrity escape can be better explored. With all 24 actuators on the tensegrity functioning correctly, a successful escape is relatively easy (9% success rate). (b) 100 Samples of Tensegrity Structures with 12 Actuators. With just half of the original actuators functioning correctly, a successful escape is even more difficult (see Figure 7). In this generation, 4% of tensegrity rovers were still able to escape the ditch in the allotted time. 72

6.9 An example signal with 2 sub-intervals with preferred lengths of y1 and y2 and periodicity t. 73

6.10 The performance of the robots during the learning session for signals of complexity 5 and period of 4 seconds. As a side result, the percentage of the policies that were failed to stay in reasonable limits are shown with the second line. 76

6.11 This plot shows actual motor positions, target motor positions, and single axis accelerometer data over the first 40 seconds of a trial for two rod ends which are not connected via cables and not attached to the same rod. The commanded positions change based on the accelerometer feedback, showing the controller working as the robot changes orientation by rolling. The actual motor position lags behind the target motor position due to motor dynamics and network UDP packet loss. 84

7.1 Images of initial SUPERball v2 MTR designs. (a) Close up of initial design for SUPERball v2. An omni directional cable guide can be seen which eliminates cable friction. (b) Full cable routing where the cable is run over a distal pulley to increase cable length for a better effective spring constant induced by the cable stretch. 88

A.1 These figures show the hystresis effect due to friction on the SUPERball. SUPERball in the (a) figures has been lifted off the ground and gently placed back down so that all the springs in the system are allowed to reach equilibrium due to gravity. No extra force has been placed on the robot other than gravity. SUPERball in the (b) figures was arbitrarily pushed downwards on two of the top endcaps with enough force to deform the robot. It can easily been seen the that the internal friction does not allow for the robot to return to it's original state. 93

A.2 This figure shows the original exit point for the steel cable on SUPERball with the "break noodle" fix. The point of entry of the "break noodle" is can be seen where the aluminum of the bracket has been cut by the steel cable before the fix was applied. 94

A.3 This figure shows wear on the cable. The vectran cable's protective outer coating wears off during use which then allows for the individual fibers to break weakening the entire cable. Eventually this wear will degrade the max holding tension to the point of failure. The black cable is a new vectran cable and the cable below shows the black coating worn away and a breakage point. The bottom cable is from an actual failure on the system during a run. Since the system can absorb forces, when the cable breaks the stored energy gets distributed into the robot and no violent backlash is experienced. 97

A.4 This figure shows the designs for the motor mount strain gage and the spring strain gage. The motor mount is located on the left, and it functions when the arms of the cross beams deform due to the reaction torque caused by torque being applied to the motor. The right shows the original design for the spring sensors. This gage is suppose to sit between the spring and the ridged mount and compress due to spring force. However, the design never worked. 98

A.5 Motor mount torque sensor data and motor position data recorded during a square wave input position trajectory for a single motor. This plot shows measured tension from the sensor and cable length from motor encoder measurements as a function of time for this dynamic movement. 99

A.6 Global force redistribution test. Yellow marks are the means of roughly 5,000 tension sensor measurements of *cable 2* opposite that which is actuated (*cable 1.*) The black line shows the linear interpolation between points, with the red boundary as standard deviation. The pretension in the sensed cable is adjusted in each test, showing measurement sequences at increasing pretensions. 100

List of Tables

2.1 Resources for tensegrity control. 13

3.1 SUPERball and Related Robots Design Overview. 21

4.1 100W Maxon BLDC Motor Parameters 40

5.1 DWM1000 configuration 50

6.1 Average distances in meters traveled using the policies learned with varying observation representations and local policy training schemes. 82

7.1 List of companies that sell COTS actuators to be integrated into SUPERball v2. This is not an exhaustive list. 87

7.2 Comparison between the current SUPERball v1 and the intended design of SUPERball v2 89

Abstract

Design, Building, Testing, And Control Of SUPERball: A Tensegrity Robot To Enable New Forms Of Planetary Exploration

by

Jonathan Bruce

Presented in this work are the concepts to build, sense and control a completely untethered tensegrity robotic system called SUPERball (Spherical Underactuated Planetary Exploration Robot), which is a compliant icosahedron tensegrity robot designed to enable research into tensegrity robots for planetary landing and exploration as part of a NASA funded program. Tensegrity robots are structurally compliant machines, uniquely able to absorb forces and interact with unstructured environments through the use of multiple rigid bodies stabilized by a network of cables. However, instead of engineering a single new robot, a fundamentally reusable component for tensegrity robots was developed by creating a modular tensegrity robotic strut which contains an integrated system of power, sensing, actuation, and communications. SUPERball utilizes six of these modular struts, making the SUPERball system analogous to a swarm of 6 individual robots, mutually constrained by a cable network.

Since SUPERball is intended for use on planetary surfaces without the support of GPS, state estimation and control policies only utilize the sensors on board the robotic system. When external sensors are used, they must be able to account for imprecise placement and automatic calibration. Also, dynamic tensegrity systems do not exhibit continuous dynamics due to nonlinear cable conditions and interactions with the environment, thus non-traditional control development methods are implemented. In this work, control polices are developed using Monte Carlo, evolutionary algorithms, and advanced supervised learning through Guided Policy Search. Each system is evaluated in simulation, while state estimation and the Guided Policy Search method are additionally evaluated on the physical SUPER-ball robotic system.

This work is dedicated to my parents Leo and Wanda and my wife Jennifer. Without their love, dedication and support, I would not have had the foundation to make it to the end.

Acknowledgments

I would like to acknowledge Vytas SunSpiral, whom was the catalyst for my endeavors into tensegrity robotics. Without his constant hard work and support, both financially and mentally, I would not have been able to achieve this goal. He has always had my best interests at heart, and he never stopped believing in me.

My UCSC advisors, Mircea Teodorescu and Gabriel Elkaim. Without Dr. Elkaim, I would have never gotten connected with Vytas SunSpiral, and Dr. Teodorescu has supported me through the final stages of my degree.

I would also like to acknowledge all the people whom I have collaborated with. Ken Caluwaerts, Jeffrey Fresien, Atil Iscen, Andrew Sabelhaus, Adrian Agogino, Steve Lessard, Marvin Zhang, Xinyang (Young) Geng, Paul Glick, Michael Fanton, and Massimo Vespignani. Whom have given me inspiration, fueled discussion, and support over the years.

The text of this thesis includes reprints of the following previously published material: [1, 2, 3, 4, 5, 6, 7]

Chapter 1

Introduction

With the advent of NASA Mars rovers, e.g. Spirit and Opportunity [8] and MSL [9], exploration through semi-autonomous robots has been shown to be an effective way to conduct meaningful science on extraterrestrial bodies. However in order to go further into our solar system and onto other extraterrestrial bodies with more hostile or unmapped surfaces than Mars, completely new types of robotic rovers need to be developed. One such innovative system which may enable exploration on these planetary surfaces is the development of tensegrity robotics. The basics of a tensegrity structure is quickly outlined in figure 1.2 and is explained more in detail in chapter 2. Motivated by this idea, the work presented is the first step into realizing this goal.

1.1 Motivation

Tensegrity robots can facilitate an intriguing low-cost planetary exploration mission profile (see Figure 1.1) comprised of the following stages: 1) A set of tensegrity robots can be squeezed into a small launch platform; 2) After initial atmospheric entry and ejection of the heat shield, they can automatically spring

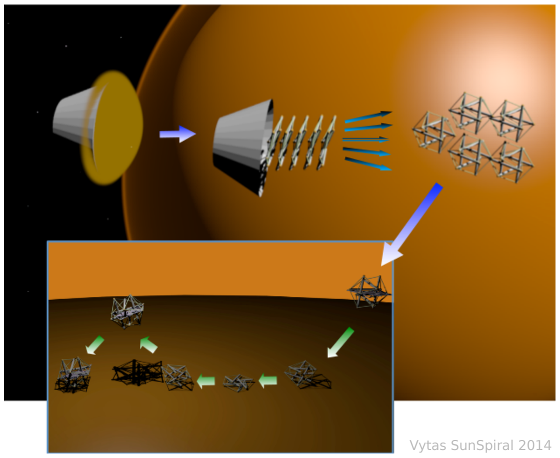


Figure 1.1: Tensegrity structures are composed of pure compression and tension elements. They can be lightweight, reliable, deployable, and efficient to manipulate. **Mission Scenario** - Tightly packed set of tensegrities, expand, spread out, fall to surface of moon, then safely bounce on impact. The same tensegrity structure which cushioned the landing is then used for mobility to explore moons such as Titan and small asteroids.

away from each other when released at their destination. 3) They bounce on impact reducing the need for final descent equipment, such as airbags; and 4) They can reorient themselves from landed position without additional reorientation hardware and efficiently move from scattered initial positions to perform sensor measurements; 5) They can survive significant falls and impacts, simplifying route planning and allowing for more aggressive exploration.

Once on the surface, tensegrity robots can perform an array of scientific analysis including soil and atmospheric composition, surface imagery and microscopic analysis. To further reduce complexity, sensors can be suspended on the interior of the tensegrity on cables attached to the nodes, or when appropriate even to the nodes themselves so that the sensors can be moved with movements of the structure itself, eliminating the need for separate sensor arms. In addition, environmental analysis can be performed in-situ at the landing site, at different local locations, or even at distant locations given a tensegrity robot's potential for efficient locomotion. The biggest advantages of this mission profile are:

1. The structure of the robot itself provides capability for deployment, Entry-Descent-Landing (EDL) scenarios, and mobility, reducing complexity, risk, and mass compared to using three separate systems.

2. Tensegrity robots are light-weight and can be packed tightly, reducing cost.

3. Tensegrity robots can scale to multiple tightly packed robots to increase scientific coverage and reduce risk.

4. Flexibility and modularity of the robot design allows design reuse, reducing mission project risk.

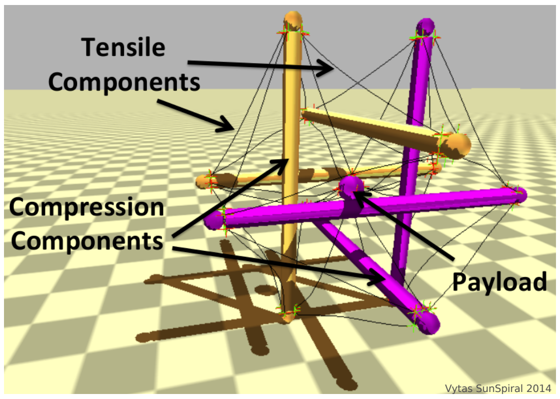


Figure 1.2: Tensegrities are composed of pure tension and pure compression elements (e.g. cables and rods) as seen in this picture of a tensegrity robot from our physics based tensegrity simulator. They are light-weight, energy-efficient and robust to failures.

1.2 Scope of Dissertation

Below is an outline of the goals achieved in this work to evaluate and develop tensegrity robotic system.

1. **Build a rolling tensegrity robot**

   A physical hardware prototype of an untethered tensegrity robot to explore locomotion. The robot will not only need to perform the basics for rolling, but will need to enable the next two goal items. To achieve this, the robot will need significant mechanical power, distributed computation, and wireless communication. This robot is presented in section 4.

2. **GPS-less State Estimation**

   In order to enable further research into path planning and system evaluations, a GPS-less state estimator will be developed. This method will track the full state of our system through sensors on board the robot as well as randomly placed ranging sensors. These randomly placed sensors could be automatically used to build a world frame relative to each other. To achieve this, research into sensor fusion and automatic calibration is needed. This work is presented in section 5.

3. **Open loop locomotion control**

   Once the hardware prototype is built, an open loop control scheme will be developed. The open loop algorithm will not change the control inputs to the system based on sensing outside of the robot. This will demonstrate the system's ability to coordinate motion between it's distributed computation and collect data. This work is presented in sections 6.2 and 6.3.2.

4. **Closed loop "gait" control**

   Once the system has proven the ability to locomote open loop, a closed loop algorithm will be developed. The closed loop control will change the "gait", or locomotion pattern, of the robot to cope with sensed variations in terrain, e.g. changes in terrain grade or climbing over an obstacles. To achieve this, research into the how much of the robot state is needed as well as various techniques to model and predict the environment by using the on board sensors. This work is presented in section 6.3.3.

Chapter 2

Literature Review

2.1 Tensegrity Structures

It is possible to design free-standing structures with axially loaded compression elements in a well crafted network of tensional elements. Such an arrangement is called a tensegrity structure (tensile integrity). Each element of the structure experiences either pure axial compression or pure tension [10][11]. The absence of bending or shear forces allows for highly efficient use of materials, resulting in lightweight, yet robust systems.

Because the struts are not directly connected, tensegrities have the unique property that externally applied forces distribute through the structure via multiple load paths. This creates a soft structure, for a soft robot, out of inherently rigid materials. Since there are no rigid connections within the structure, there are also no lever arms to magnify forces. The result is a global level of robustness and tolerance to forces applied from any direction.

This makes tensegrity robots inherently compliant and extremely well suited for physical interactions with complex and poorly modeled natural environments. Active motion in tensegrity robots can be performed by changing cable lengths in parallel, enabling the use of many small actuators that work together, rather than individual heavy actuators which work in series. There are also many indications that tensegrity properties are prevalent throughout biological systems, and the morphology of the SUPERball that we are studying, especially when carrying a payload, ends up bearing a striking resemblance to the nucleated tensegrity model of cell structure [12][13].

2.2 Prior Work in Tensegrity Robotics Design

An important advantage of tensegrity structures with respect to general pin-jointed structures is their increased mass-efficiency due to a high fraction of tensile members. Tensile members are generally more mass-efficient as they do not need to resist buckling. A further advantage from a robotics perspective is that forces diffuse in a tensegrity. There are no lever arms and torques do not accumulate at the joints as in a classic serial manipulator. Forces distribute through multiple load paths, thus increasing robustness and tolerance to mechanical failure.

The static properties of tensegrities have been thoroughly studied and some basic analysis is discussed in section 3. On the other hand, few examples are known of truly dynamic motion of these structures. Early examples of kinematic motion include the work at EPFL's IMAC laboratory [14]. Skelton and Sultan introduced algorithms for the positioning of tensegrity based telescopes and the dynamic control of a tensegrity flight simulator platform [15]. Although there were some early efforts at MIT's CSAIL lab, it wasn't until the work of Paul and Lipson at Cornell University that the concept of tensegrity robotics became widespread [16]. Paul and Lipson were the first to study the properties of dynamic tensegrity structures in hardware and simulation. A few years later Fivat and Lipson designed the IcoTens, a small actuated tensegrity icosahedron robot, but did not publish results. In recent years, the BIER lab at the University of Virginia has been studying Central Pattern Generator based control for tensegrity based fish tails, which is closely related to the control architectures proposed for SUPERball [17, 18]. Mirats-Tur has presented design and controls work on various other tensegrity morphologies that have been tethered or fixed to the ground [19, 20]. At Union College, Rieffel and colleagues are following an interesting line of work by considering vibration based actuation for small tensegrities [21]. Related work was presented by Böhm and Zimmermann, who demonstrated controlled locomotion of vibration driven tensegrity robots with a single actuator [22]. Finally, Shibata, Hirai and colleagues have developed pneumatically actuated rolling tensegrity structures [23].

Building upon these works, the SUPERball project seeks to push forward the tensegrity robotics field and develop truly untethered, highly dynamic and compliant robots exploiting the aforementioned advantages.

2.3 Tensegrity Robotics for Space Exploration

The high strength-to-weight ratio of tensegrity structures is very attractive due to the impact of mass on mission launch costs. Large tensegrity structures have been shown to be deployable from small compact configurations which enable them to fit into space constrained launch fairings. While the above qualities have inspired studies of deployable antennae and other large space structures [24], it is in the realm of planetary exploration that we see the most significant role for many of the unique force distribution qualities of tensegrity robots. The project formally funded by the NASA Innovative Advanced Concepts (NIAC) program [1] and currently NASA's Ground Changing Development (GCD) is funding this research to specifically study landing and surface mobility of tensegrities, exploiting the controllable compliance and force distribution properties which make for reliable and robust environmental interactions.

The main goal is to develop tensegrity probes with an actively controllable tensile network to enable compact stowage for launch, followed by deployment in preparation for landing. Due to their natural compliance and structural force distribution properties, tensegrity probes can safely absorb significant impact forces, enabling high speed Entry, Descent, and Landing (EDL) scenarios where the probe itself acts much like an airbag. However, unlike an airbag which must be discarded after a single use, the tensegrity probe can actively control its shape to provide compliant rolling mobility while still maintaining the ability to safely absorb impact shocks that might occur during exploration. This combination of functions from a single structure enables compact and lightweight planetary exploration missions with the capabilities of traditional wheeled rovers, but with a mass and cost similar or less than a stationary probe.

Therefore, a large fraction of the overall weight (as measured at atmospheric entry) of a tensegrity mission can be used for the scientific payload due to the dual use of the structure as a lander and a rover. This allows for cheaper missions and enables new forms of surface exploration that utilize the natural tolerance to impacts of tensegrities [2].

2.4 Tensegrities as Soft Robots with Morphological Computation Capabilities

Tensegrities share many of the design, fabrication, modeling, sensing, and control challenges of the broader category of soft robots [25, 26, 27], which are made out of intrinsically soft and/or extensible materials.

2.4.1 Opportunities in Morphological Computation

Both tensegrity and soft robots closely relate to the notion of embodied intelligence, where morphology and materials can take over some of the functions normally attributed to control to achieve a system that is overall simpler, more robust and adaptive than those based on the classical control paradigm. This principle is known as *morphological computation* [25, 28]. Many approaches to morphological computation [29, 28] seek to reduce the complexity of control systems through intelligent mechanism designs, which effectively exhibit complex behaviors while reducing the use of explicit control systems. An example would be a compliant, soft hand that naturally grasps a wide range of object shapes while executing the same simple control law in all cases [30]. These systems may reduce the amount of sensing, actuation, or explicit modeling and decision making associated with traditional approaches to the task.

Tensegrity robots show this quality when they passively conform to the environment and re-balance forces throughout their structure [31]. This is a very desirable property, which enables the design of robots capable of a wide range of tasks and activities. Such platforms are generally more versatile and robust in the face of noise and the unpredictability of operating in messy real-world environments.

2.4.2 Methods for Controlling Soft Robots

Soft materials can bend, twist, stretch, compress, buckle, wrinkle and so on. Such motions may involve an infinite number of Degrees of Freedom (DoF) indicating that the control if soft robots requires new approaches [32, 33]. While some progress can be made with more traditional control approaches, such as Model Predictive Control (MPC), these efforts typically require accurate models of the

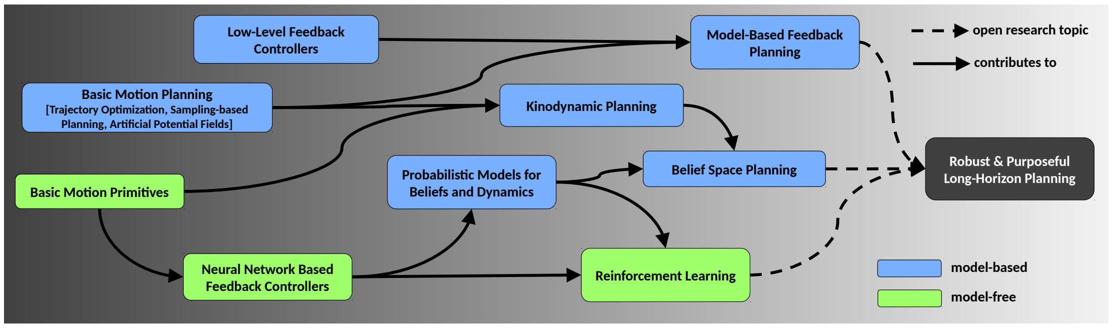

Low-Level Feedback Controllers
Model-Based Feedback Planning
open research topic
contributes to
Basic Motion Planning [Trajectory Optimization, Sampling-based Planning, Artificial Potential Fields]
Kinodynamic Planning
Belief Space Planning
Robust & Purposeful Long-Horizon Planning
Basic Motion Primitives
Probabilistic Models for Beliefs and Dynamics
Neural Network Based Feedback Controllers
Reinforcement Learning
model-based
model-free

Figure 2.1: Overview of research topics discussed. An arrow indicates a topic that can contribute to the development of another.

robot and environment, which can be difficult to acquire given the complex physical properties of soft robots. Thus, many efforts focus specifically on biomimetic systems [26], which aim to reproduce the control behaviors of their biological counterparts and often provide a new understanding of soft organisms [34].

To better understand the challenges in controlling soft and tensegrity based robots, new static, kinematic and dynamic models have been developed to capture the ability of bending and flexing [35, 36]. For many years, a popular abstraction for soft robots has been that of piece-wise constant curvature (PCC) [37], which does not capture all aspects of real mechanisms. Recently some non-constant curvature models have been proposed to better model soft mechanisms. [38]. The need for expressive models has also led to the development of simulation tools targeted to soft robots [39]. This is also an important development in tensegrity robotics [17], in which open source physics based simulation tools have recently become available [40]. Table 2.1 gives pointers to simulation tools and analytical models for tensegrity structures.

There have been both model-based and model-free approaches for the low-level control of soft robots [41]. On the model-based side, a recent effort utilizes finite elements [42], while a recent data-driven, model-free approaches utilizes graph-theory [43]. There is no consensus, however, yet regarding the appropriate methodology for control, and especially planning, for soft robots given their highly continuous, complex and intrinsically compliant deformation [32]. This has motivated efforts in proposing behavior-based control architectures for soft robots that may be applicable to tensegrities as well [44]. A critical challenge in achieving deliberative control and planning, shared between tensegrity structures and soft robots, is the difficulty of solving the inverse kinematics problem. There are solutions in certain setups, such as for semi-soft manipulators [45].

2.4.3 Planning for Tensegrities

Similar to soft robots, the very properties that make tensegrities ideal for physical interaction with the environment, such as compliance, and multi-path load distribution, present some significant challenges to traditional planning approaches and lead to uncertainty during motion execution.

For instance, compliance allows tensegrity robots to adapt their shape to workspaces of unknown or uncertain geometry. But when a force is applied, the robot can deform in a non-linear manner and will often excite oscillatory behaviors [46]. The results of contacts are therefore very hard to predict to the level of accuracy required for traditional trajectory and route planning techniques. These issues have limited investigation of planning algorithms and focused most existing efforts on controllers for the generation of local gaits [16] and quasi-static paths [47]. Yet, they have also inspired the development of approaches that go beyond the traditional control toolkit and allow adaptation to multiple different terrain types [48, 49]. Similar developments in soft robotics technologies are taking place, which address the complications of self-loading and non-linear compliance [32]. Recently, planning methods have been introduced which begin to address these needs for soft robots [50, 51].

To move this field forward for both tensegrity and soft robots, both high-level approaches of model-based and model-free planning should be investigated as highlighted in Fig. 2.1. Model-based planning approaches should be able to reason over more complex dynamical models of tensegrity systems and provide robust trajectories over probabilistic state representations that capture the diversity of possible executions given the inherent uncertainty. Another direction is to study feedback-based motion planners that provide a robust composition of controllers with performance guarantees. In the model-free domain, methods should capture the dynamics of low-level controllers and then plan over the resulting dynamics. The low-level controllers should manage the details of environmental interaction while successfully driving the system to the next waypoint.

2.5 Low-level Control for Tensegrity Robots

Early tensegrity research was mostly focused on modeling the statics [58, 60, 71] and dynamics of a structure, so as to provide effective equations of motion [72, 73, 36]. In specific cases, such as state estimation, modeling tensegrities as constrained mass-spring nets allows for highly efficient and sufficiently accurate implementations [6]. The mass-spring approach has also proven valuable in more theoretical studies of morphological computation [28]. Nevertheless, there

resource	**references**
dynamics models	[36, 52, 53, 54, 55]
kinematics & statics	[56, 57, 36, 58, 59, 60, 61]
simulation	[40, 48, 36, 17, 62]
hardware design	[63, 22, 14, 23, 17, 20, 64, 31, 65]
	[66, 16, 67, 68, 5, 3, 69]
state estimation	[70]

Table 2.1: Resources for tensegrity control.

is a trade off between computational efficiency and high dynamic model fidelity during environment interaction modeling. Table 2.1 summarizes many of these contributions that can impact tensegrity control.

Many of the tensegrity hardware robots are tendon-driven or use pneumatic actuators, which are typically a burden to accurately model analytically. A learned model might increase the computational efficiency when trying to represent real-world hardware in changing environments. Section 6.3 discusses the option of using learned models as a proxy to simulation or involved analytical models.

Given a dynamic model, it is possible to control a tensegrity structure along static equilibrium manifolds [74]. Alternatively, feedback linearization control laws [75] or Lyapunov-based controllers for 3D dynamic models [55] have also been developed. Frequently, these approaches do not account for self-collisions or environmental contact dynamics, limiting their real-world applicability. Planning processes for a real-world tensegrity structure need to utilize modeling and simulation tools that take collisions into account.

Many efforts focus on generating efficient gaits, defined as rhythmic motions, which lead to nonzero movement of the center of mass [76]. Given the high-dimensional nature of the search space, genetic algorithms are frequently applied to achieve forward locomotion gaits [16]. Evolutionary algorithms have been used for generating irregular locomotion and civil engineering structures [77, 69]. Recently, evolutionary methods have been proposed that utilize a multi-agent learning approach [78]. Other biologically-inspired approaches based on Central Pattern Generators (CPGs) have also been applied to tensegrity-based systems [63, 48, 17]. A recent overview of low-level tensegrity control approaches is available in the related literature [17, Table 2]. In this work, Monte Carlo [79] and evolutionary techniques [80] are used to learn open loop policies for locomotion.

While an Artificial Neural Network (ANN) is trained as a closed loop locomotion controller. Section 2.6.1 gives a basic overview of ANNs.

The availability of simulation tools has offered researchers the possibility to develop a wide range of controllers. Nevertheless, additional hardware validation results are needed to better support the claims regarding the efficacy of the developed solutions [68, 17]. It is crucial to determine the feasibility of each method in terms of sensing and state estimation, their aptitude for distributed implementations and the minimum number of actuators required.

Furthermore, hardware experiments have not typically utilized fundamental analytical control approaches (e.g. [54]), since they frequently depend on accurate state information, which is non-trivial to acquire. Nevertheless, there has been recent progress on actuation, sensing, and state-estimation methods that are robust to noisy sensors and environments. Thus, it may be time to revisit some of the earlier analytical control techniques. This will allow a thorough comparison with more modern methods that have reduced sensing and actuation requirements in simulation and hardware.

2.6 High Level Control for Tensegrity Robots

2.6.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a theoretical group of mathematical models loosely based on neural networks found in biology that can estimate usually unknown functions which map multiple inputs to a set of outputs. An ANN is comprised of a large set of simple functions grouped into different layers that define a mapping function $f : X \to Y$ where X is the input set and Y is the output set. In practice, layers are grouped into three main layers defined as 1) the Input layer, 2) the Hidden layer, and 3) the Output layer. The Input and Output layers transfer data to and from the Hidden layer, respectively. The Hidden layer can be comprised of one ore more sub-layers and is where the input data is converted to the output data. Each sub-layer of the Hidden layer is comprised of a defined number of simple functions H_i for $i \in n$, where n is total number of functions in a sub-layer. Changing how each function $h \in H$ is connected to other functions, by the use of weights, allows for the ANN to map its inputs to a desired output [81]. For our application, machine learning is used to estimate the connection weights through the use of a cost function. The cost function is an equation which measures the success of the learning process against a user defined parameter or set of parameters.

ANN for Locomotion

Robotic learning methods have previously produced successful policies for tasks such as locomotion for walking robots and quadrupeds [82, 83, 84, 85, 86]. These methods, however, typically require hand-engineered policy classes, such as a linear function approximator using a set of hand-designed features as input [82]. For many tensegrity systems, it is difficult to design suitable policy classes, since the structure of a successful locomotion strategy might be highly complex.

Some more recent methods learn deep neural network policies that are successful for tasks such as grasping with robotic arms and bipedal locomotion [87, 88], and such policies are more expressive and require less hand-engineering compared to policy classes used in previous methods. One such method, which is used in this work in section 6.3.3, is mirror descent guided policy search (MDGPS), an algorithm that frames the guided policy search (GPS) alternating optimization framework as approximate mirror descent [89]. MDGPS is was chosen for this work because it allows for deep neural network policies to be learned while maintaining sample efficiency, and it presents a natural extension to periodic locomotion tasks.

A key problem for locomotion tasks is the difficulty of establishing stable periodic gaits, and this is exacerbated for tensegrity robots due to their complex dynamics and unusual control mechanisms. Near-stable behavior with even small inaccuracies can lead to compounding errors over time, and will not be successful in producing a continuous periodic gait. Previous algorithms have dealt with this problem by establishing periodicity directly through the choice of policy class [83, 90, 91], utilizing a large number of samples [87], or initializing from demonstrations [92, 93]. Instead, this challenge is handled by sequentially training several simple policies that demonstrate good behavior from a wide range of states, and then learning a policy that reproduces the gait of all of the sequential policies for a successful periodic behavior. The resulting algorithm learns policies from scratch for robotic tasks that exhibit periodicity over long time horizons. Section 6.3.3 demonstrates through experimentation that a single learned policy for a tensegrity robot is capable of efficient, continuous locomotion in a range of different conditions by learning appropriate feedbacks from the robot's onboard sensors.

2.6.2 Path Planning Through a Sampling-Based Motion Planner

Recently, Littlefield et al. [94] implemented a high level sampling-based motion planner on a simulated SUPERball robot in the NASA Tensegrity Robotics Toolkit (NTRT), explained in chapter 3. This method, based on a kinodynamic planner published by Yanbo et al. [95], is able to converge to an asymptotically optimal solution on dynamic systems using only forward propagation and uses a novel parallel implementation to make the forward propagation step more computationally feasible. The whole process is done within the NTRT simulation environment, where each parallel process is a NTRT simulation of SUPERball. Figure 2.2 shows a visual representation of two forward propagation steps with their respective uncertainty shown as overlaid transparent robots.

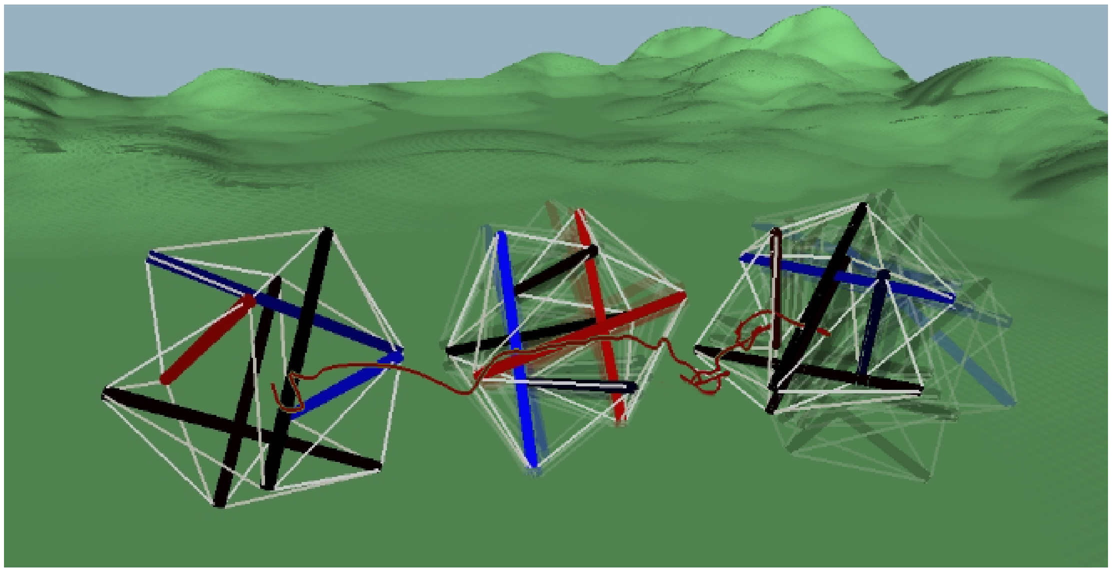


Figure 2.2: An example of a trajectory with two forward progagations and their uncertainty. Possible future configurations are shown as transparent versions of SUPERball. (figure courtesy of Zakary Littlefield).

The main control method for this process has been a random sample of individual low level motor commands to produce the desired motion. It has been shown with full actuation on the simulated SUPERball that path following over various terrains and obstacles are achievable with this method. However, this process only learns a set of open loop actions based on the simulation environment and requires 20+ processing cores to have reasonable computation times.

Chapter 3

Modeling and Model Validation

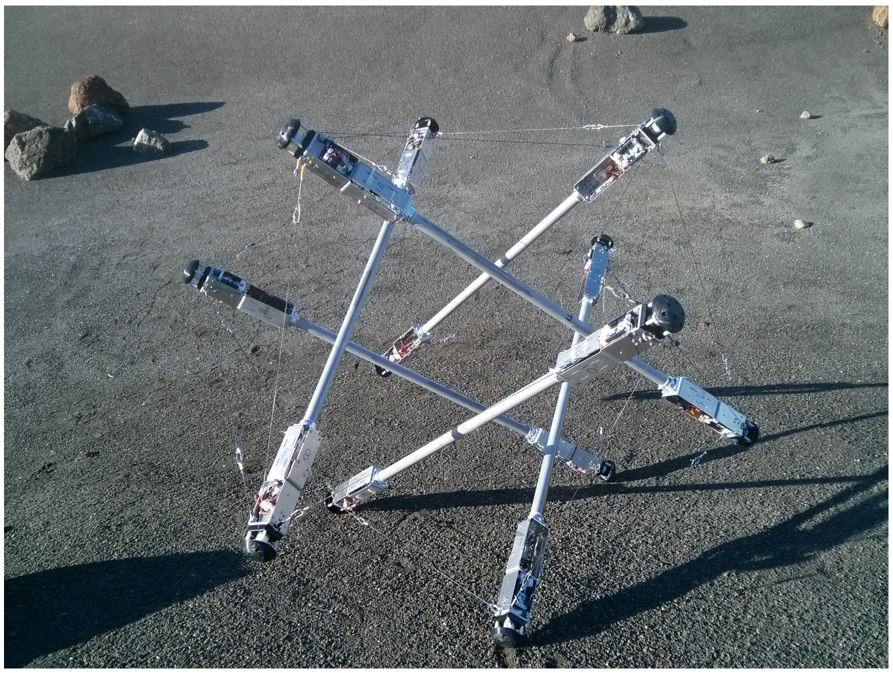


Figure 3.1: SUPERball, fully assembled, in the NASA Ames Research Center Roverscape.

As part of our research for the NASA Innovative Advanced Concepts (NIAC) program, we are developing the SUPERball (Spherical Underactuated Planetary Exploration Robot), which is a compliant icosahedron tensegrity robot designed for planetary landing and exploration, seen in figure 3.1. Tensegrity robots are soft machines which are uniquely able to compliantly absorb forces and interact with unstructured environments. However, instead of engineering a single new robot, we have chosen to develop a fundamentally reusable component for tensegrity robots by creating a modular robotic tensegrity strut which contains an integrated system of power, sensing, actuation, and communications. The purpose is to enable the exploration of the wide range of possible tensegrity robotic morphologies by simply combining the robotic struts into new systems.

Though there is much prior work in a variety of theoretical areas for tensegrities, engineering knowledge of constructing practical tensegrity robots is limited. Since a staggering variety of different tensegrity structures can be constructed from collections of simple sticks and strings, we have made it a priority to develop self-contained robotic tensegrity struts which can be used to explore and build a wide range of tensegrity robots simply by combining them into novel structures. Our designs are driven by experimental results obtained from a previous prototype, ReCTeR (Reservoir Compliant Tensegrity Robot) in combination with simulation results of our validated tensegrity simulator NTRT (NASA Tensegrity Robotics Toolkit) [96][17].

In order to develop SUPERball from ReCTeR's design limitations as well as our lab's need for rapid experimentation of various tensegrity configurations and morphologies, we came up with a modular tensegrity platform to research large scale robotic tasks; e.g. a tensegrity planetary probe to explore Saturn's moon Titan. Our lab obtained design requirements through an iterative approach with validated our NTRT simulator by experimental comparison with ReCTeR [17]. We now can quickly evaluate various tensegrity configurations in simulation to find optimal mechanical design goals. In conjunction with the NTRT solver, we also incorporated results obtained with our (open source) Euler Lagrange solver based on Skelton's work [97] and measurements on ReCTeR. The initial design requirements obtained from the NTRT simulations, refined designs after a first prototype build, and how these compare to other tensegrity robotic systems are given in Table 3.1.

Table 3.1: SUPERball and Related Robots Design Overview.

	l_{strut}	Δl_{act}	$k_{passive}$	**tethered?**	**control**	f_{act}	**#act.**	**mass**	**sensors**	**actuators**	**ref.**
Pneumatic	0.57 m	-	-	Y	open loop	800 N	24	3.3 kg	none	McKibben	[66]
ReCTeR	1 m	0.3 m	$28.4\,\mathrm{N\,m^{-1}}$	N	closed loop	12 N	6	1.1 kg	F, L, IMU	DC	[17]
Rapid Proto Kit	0.69 m	0.005 m	$1193\,\mathrm{N\,m^{-1}}$	N	open loop	<45 N	24	2.7 kg	none	linear DC	[98]
SUPERball 2014	1.5 m	0.2 m	$613\,\mathrm{N\,m^{-1}}$	N	closed loop	140 N	12	12 kg	F, L, τ, IMU	BLDC	
SUPERball 2015	1.7 m	0.42 m	$998\,\mathrm{N\,m^{-1}}$	N	closed loop	250 N	12	21 kg	R, L, IMU	BLDC	

The variable l_{strut} indicates the length of a strut, Δl_{act} is the nominal spring-cable retraction length in tension, $k_{passive}$ is the linear stiffness coefficient of a passive spring-cable (or active spring-cable if fully actuated), tethered indicates if the robot is powered externally or by internal systems, control indicates whether sensor feedback is used, f_{act} is the nominal actuated spring-cable tension and #act. is the number of actuators. In the sensors column, F represents a linear force sensor (for cables), L is cable length sensor (in the form of motor encoders), τ represents a torque sensor for motors, R represents ranging sensors for rod end cap positions, and IMU represents an accelerometer/gyroscope inertial motion sensing unit. Actuators are specified as DC motors or brushless DC (BLDC) motors. The SUPERball 2014 values are revised original design requirements based on NTRT simulations, and changed to the 2015 values after additional detail design.

Presented here is work verifying our in house tensegrity simulators. In order to achieve this, a SUPERball like structure with a center payload was used. This is believed to be closer to the proposed build profile of a real tensegrity probe, where the main science modules will be contained within the payload. Protecting this science payload is the main goal for and EDL scenario. Figure 3.2 shows a 3-D representation of SUPERball with a payload generated within NTRT.

3.1 Euler-Lagrange Model

In order to verify the simulation results produced by our NTRT simulator, we decided to compare the behavior of the NTRT to a published analytic model for tensegrity systems. We choose to use Skelton's dynamic equations because it is a well accepted and used model. It may be found in his *Tensegrity Systems* book [36] which is based on his work in [99]. In order to solve the dynamic equations with interactions with the environment, an Euler-Lagrange approach is used as well as Skelton's constrained class one structure. The lagrange equation for a constrained rod is given by

$$L = T - V - c \tag{3.1}$$

where

$$\mathbf{b} = l^{-1}(\mathbf{n}_j - \mathbf{n}_i) \tag{3.2}$$

$$c = \frac{\mathbf{J}\xi}{2}(\mathbf{b}^T\mathbf{b} - 1) \tag{3.3}$$

Equation (3.2) is the normalized vector of a rod with $\mathbf{n}_{i,j}$ the nodal positions in R^3, and equation (3.3) contains the lagrange multiplier ξ to keep (3.2) constrained. $\mathbf{J}$ is also defined as the inertia matrix for a one dimensional rod in three dimensional space. In order to define the system of k rods we need to define a combined Lagrangian as

$$\mathbf{L} = \sum_{i=1}^{k} L_i \tag{3.4}$$

where L_i is the Lagrange function for each rod. Using the approach outlined in Skelton's book for deriving the equations of motion, we can then derive the configuration matrix

$$\mathbf{Q} = \begin{bmatrix} \mathbf{R} & \mathbf{B} \end{bmatrix} \tag{3.5}$$

where $\mathbf{R}$ and $\mathbf{B}$ are matrices containing the translational and rotational vectors, respectively. They have the form

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 & \cdots & \mathbf{r}_k \end{bmatrix} \tag{3.6}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_k \end{bmatrix} \tag{3.7}$$

Also using the procedure to derive generalized forces within Skelton's book, the systems's generalized force equations are computed as

$$\mathbf{F_Q} = \begin{bmatrix} \mathbf{F_R} & \mathbf{F_B} \end{bmatrix} \tag{3.8}$$

with

$$\mathbf{F_R} = \begin{bmatrix} \mathbf{f}_{\mathbf{r}_1} & \cdots & \mathbf{f}_{\mathbf{r}_k} \end{bmatrix} \tag{3.9}$$

$$\mathbf{F_B} = \begin{bmatrix} \mathbf{f}_{\mathbf{b}_1} & \cdots & \mathbf{f}_{\mathbf{b}_k} \end{bmatrix} \tag{3.10}$$

Finally, we can define the resulting equations of motion in a compact form as

$$(\ddot{\mathbf{Q}} + \mathbf{Q}\boldsymbol{\Xi})\mathbf{M} = \mathbf{F_Q} \tag{3.11}$$

where

$$\boldsymbol{\Xi} = diag\begin{bmatrix} 0, \cdots, 0, \xi_1, \cdots, \xi_k \end{bmatrix} \tag{3.12}$$

$$\mathbf{M} = diag\begin{bmatrix} m_1, \cdots, m_k, J_1, \cdots, J_k \end{bmatrix} \tag{3.13}$$

This approach was then implemented in Python utilizing a 4th order Runge-Kutta formula for solving the system of ordinary differential equations. In order to implement a gravitational field, a force distribution function is applied along the length of each rod and calculated as a nodal force depending on the given density of the rod. This external force is then applied to the nodes during each time step, simulating a gravitational field.

3.2 NASA Tensegrity Robotic Toolkit Model

The NASA Tensegrity Robotic Toolkit (NTRT) is a software suit which enables the easy modeling and control of tensegrity robotic structures in a real-time simulation environment. NTRT utilizes the Bullet physics engine to simulate ridged body interactions, which is a Cartesian space open source numerical solver [100]. The toolkit also integrates builder tools, to simplify and standardize the construction of tensegrity structures, and a custom cable model, which more accurately simulates cable contact dynamics. Figure 3.2 shows a model of SUPERball built in NTRT and using GLUT/FreeGLUT [101] as the graphical output.

As of writing this document, there is a software bug in GLUT/FreeGLUT which affects NTRT and does not allow for the user to simultaneously display a graphical output and manually control the time step. This is not a limiting factor if the user is running thousands of Monte Carlo simulations and does not need to see each individual trial, as is the case for Mirletz and Iscen's papers [102, 103]. However for the machine learning utlizied on SUPERball, learning is quite rapid and observing the progress as the learning algorithm is working helps determine if a usable controller is being learned. To enable this, NTRT is run in non-graphical mode and a modified class was implemented that took the bar positions at each time step and displays them in the open source 3D rendering program OpenSceneGraph [104]. It was decided not to render the robot's cables in OpenSceneGraph due to the large amount of effort required to code and debug for marginal returns.

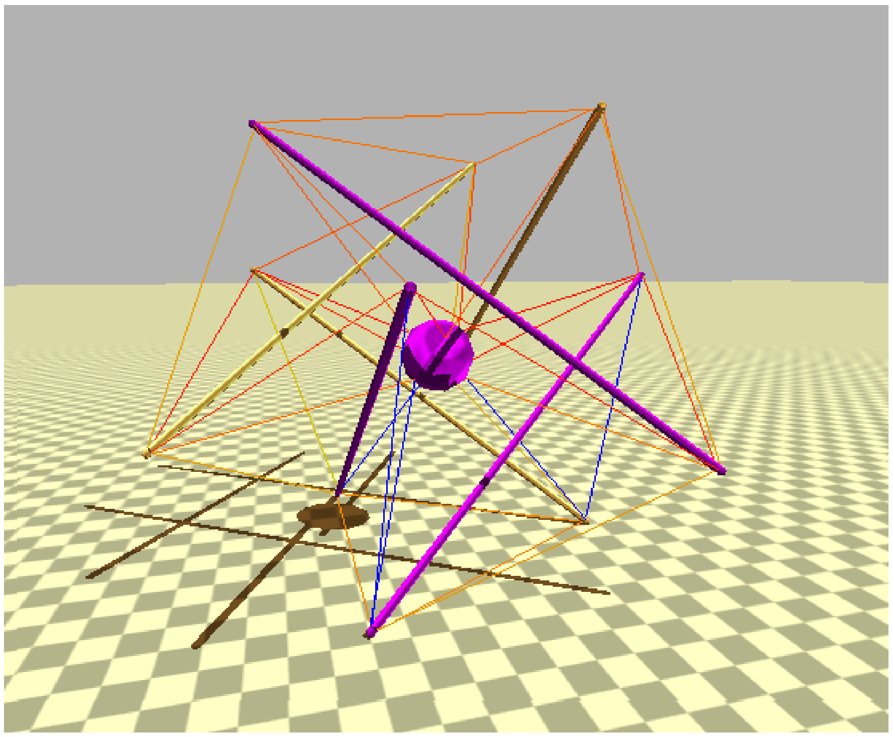


Figure 3.2: SUPERball with a payload modeled within NTRT.

Figure 3.3 shows an example of SUPERball in OpenSceneGraph.

3.3 Detailed Impact Simulations and Cross Validation Using Two Simulators

The NTRT simulator is the most general purpose tensegrity simulator available, allowing users to explore control algorithms and complex environmental interactions, but it is an iterative discrete solver that has the potential of providing inaccurate answers. The Euler-Lagrange (E-L) solver, on the other hand, has a much stronger analytical basis and should provide very accurate answers, but is limited because some of the nodes (rod ends) must be constrained and locked into place. This is unrealistic for the deformation caused during landing, and makes

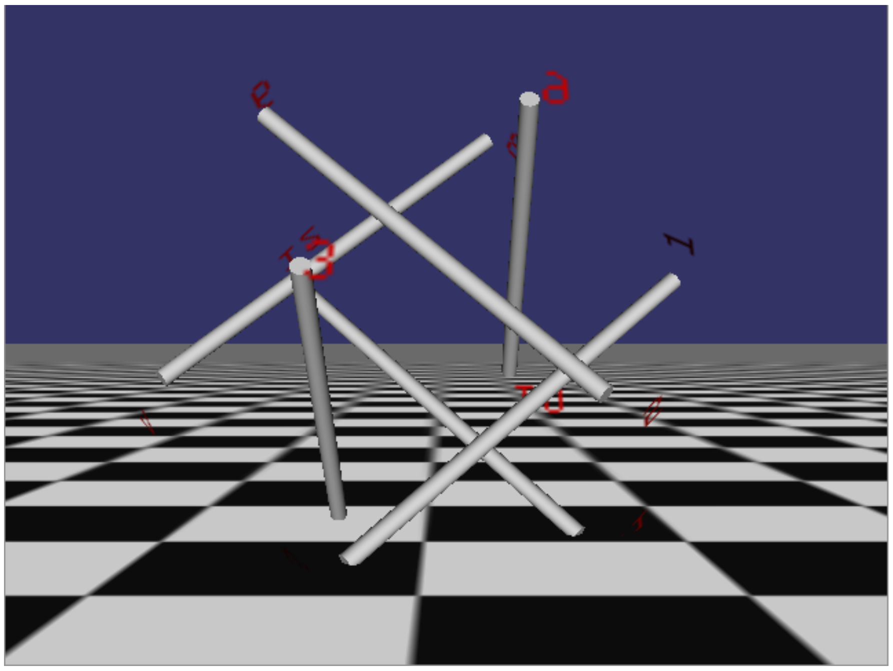


Figure 3.3: SUPERball simulated in Open Scene Graph. Note that this graphical display of NTRT does not support cable visualization.

it an inappropriate choice for mobility and controls research.

In this section, a comparison is shown between the NTRT simulator and E-L solver at the moment of impact with the ground. The simulations are compared at this moment because our implementation of the analytic E-L solver requires select nodes to be constrained. The structure is setup so that time is equal to zero at the instantaneous moment it comes into contact with the ground. In both simulations, we add an initial velocity equal to the terminal velocity of Titan, and compared each vertical trajectory, vertical velocity, and vertical acceleration of the payload. Since the structure's horizontal speed is zero at the beginning and the structure is symmetrical, the payload's horizontal components of position, velocity and acceleration are zero. As it can be seen in the Figures 3.4 and 3.5, both simulators closely match and generate the same results for position and velocity with the error margin close to zero. Comparing the accelerations generated by two simulators (Figure 3.6), it can be seen that there is a bigger difference. The reason behind this difference is the fact that NTRT uses Bullet, which is a discrete time simulator and accelerations are calculated using two point estimations from velocities at the timestep before. Yet, even with these differences in accelerations, the conclusion at the end of the comparison is that both simulators showed the same basic dynamics and their results were close enough that it is conceivable to use the more general purpose NTRT Simulator for our controls, mobility, and landing experiments.

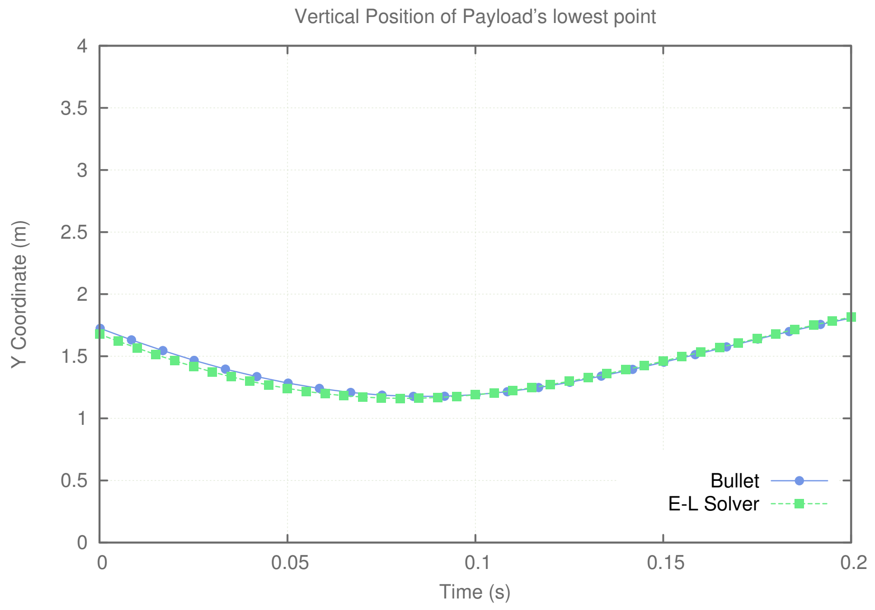


Figure 3.4: NTRT vs EL: Vertical Position

3.4 Simulated Drop Tests and Payload Protection

Finally, extensive analysis were performed on various drop tests and the protection provided to a payload. As one might expect, varying the rod lengths which

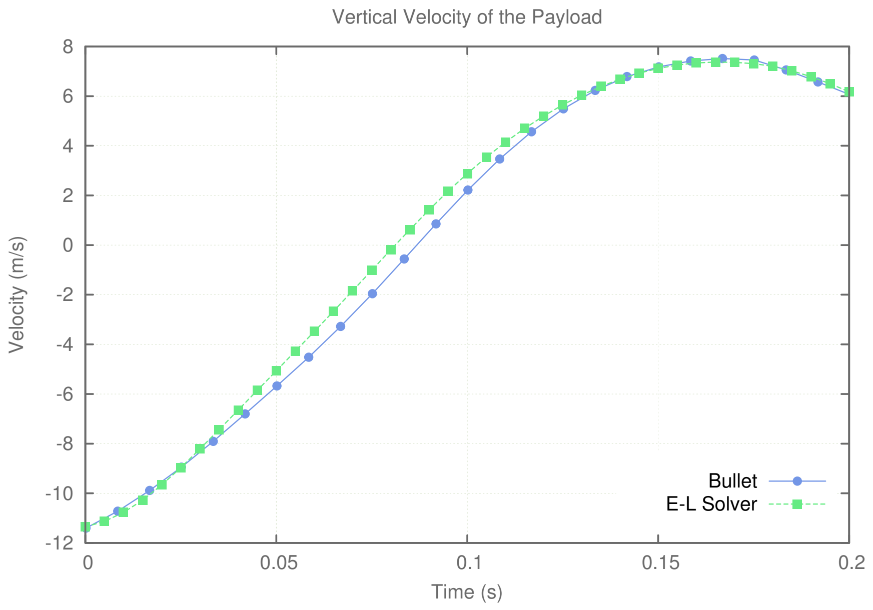


Figure 3.5: NTRT vs EL Vertical Velocity

impacts the stroke distance for the payload to decelerate, it is possible to control the maximum deceleration experienced by the payload while ensuring that it did not collide with the ground or structure. For example, with rods of 1.5 meters in length, the payload experienced a max deceleration of 21.4G when landing at 15 m/s. Figure 3.7 shows the results of a series of drop tests with different rod lengths and shows the resulting maximum deceleration and forces experienced in the tension members. As can be seen from these graphs, even for reasonable rod lengths, the maximum G's are acceptable for most instruments, and the maximum forces experienced by the cables are easily within ranges that can be engineered for. In all tests, the total system mass is kept constant at 100kg (which is 70kg for the payload and 5kg per rod) in order to highlight the impact of structural geometry and rod length. For the tension members, spring constants of 44 kN/m were used for the cables around the perimeter and 10 kN/m for the cables attached to the payload. Also, the results in Figure 3.7 were found using the landing orientation of 35 degrees around X axis and 45 degrees around Z axis, which were selected

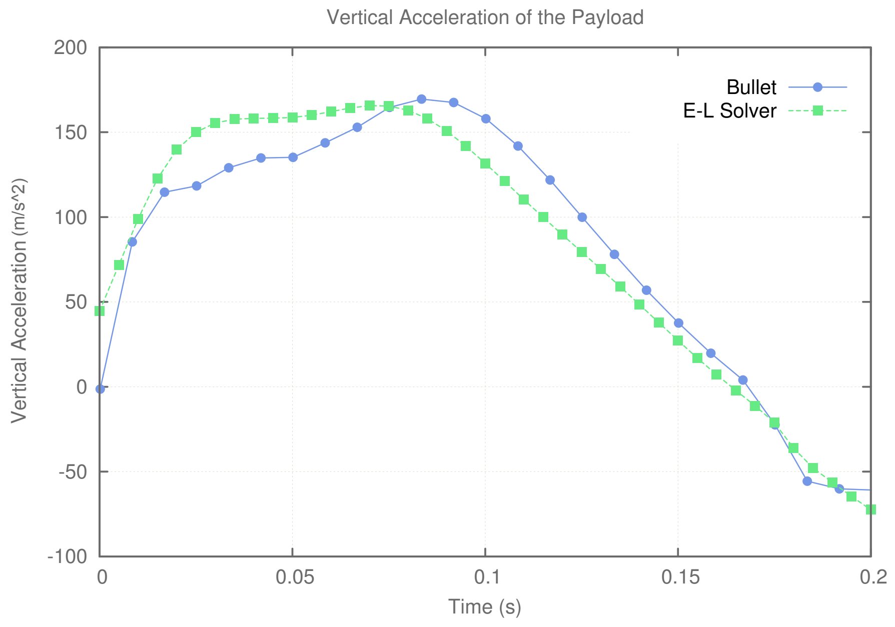


Figure 3.6: NTRT vs EL Vertical Acceleration

from the orientation studies discussed below.

A very interesting point to consider is that the mass of a SUPERball like system will grow in a linear fashion with the length in the rods, while providing increasing payload protection. On the other hand, the mass of airbags increases with the square of the radius, which is one of the reasons that the MSL rover, with its increased size and mass, had to switch from the airbag approach to the more complex Sky Crane approach. While this study has focused on small light-weight mission concepts, there could be compelling advantages for scaling up to handle larger payloads.

3.5 Landing Orientation Studies

In order to study how landing orientation affects payload decelerations and impact events, a systematic study of landing orientations was conducted. In order to get meaningful data, even for bad orientations, a larger tensegrity structure

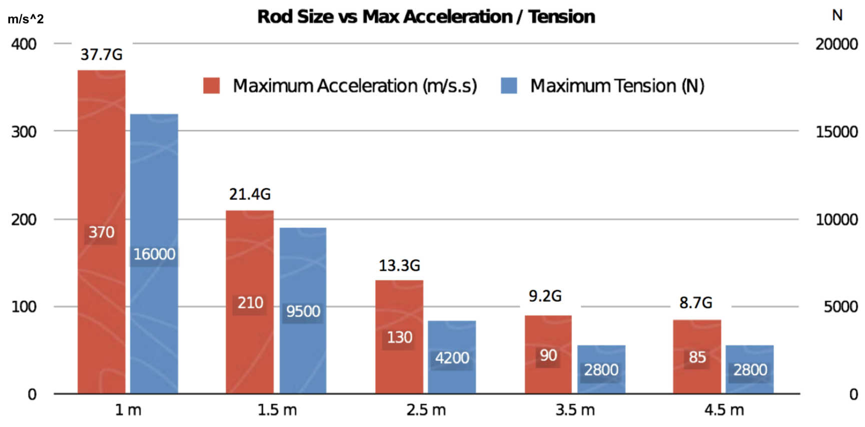


Figure 3.7: Landing Forces Study. *This shows how rod length impacts maximum deceleration of the payload and the maximum forces experienced by the tension cables. All tests were conducted with a landing velocity of 15 m/s onto a hard surface.*

with 4 meter rods was used so the data wouldn't saturate. The success criteria for this study was that the decelerations had to stay under an upper limit of 25G deceleration of the payload, and the payload had to avoid collision with the ground or parts of the tensegrity structure. Figure 3.8 shows the orientations that landed safely within these criteria (black) or failed one or both of the criteria (colored). By using a simple trailing streamer during descent it would be possible to control landing at an optimal orientation and enable the use of smaller structures with shorter rods because the orientation control would maximize the available stroke for the payload to decelerate within the structure. Conversely, these studies could be used to know what the worst possible landing scenario will be and choose a structure size which will allow safe landing at any orientation.

3.6 Conclusions from Simulation Experiments

Using the scenario of free fall impacts, two different simulation methods were developed and cross-validated which enables the exploration into the capabilities

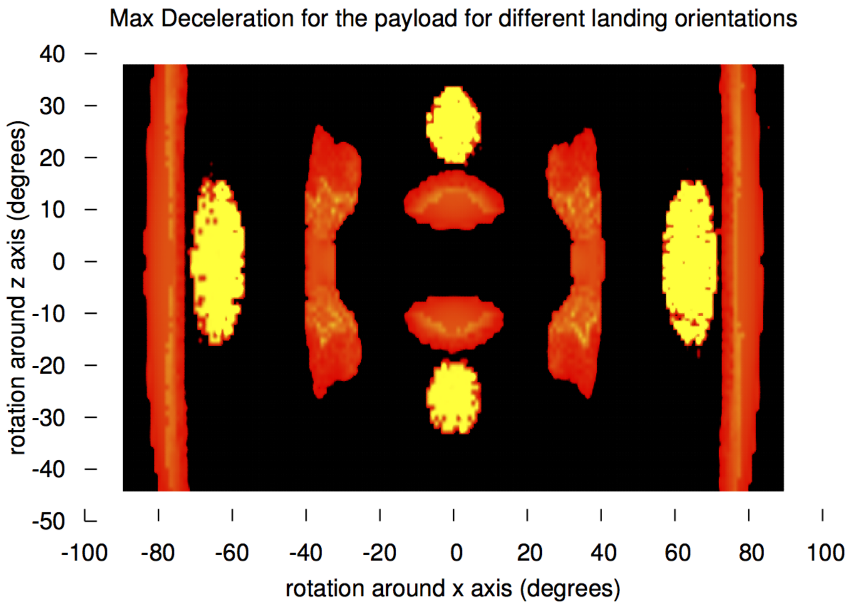


Figure 3.8: *Heat map of the maximum acceleration that the payload encounters for all possible landing orientations. Black areas are safe, colored areas are where the payload does not meet one or both success criteria.*

of a tensegrity structure to absorb the forces of landing and to simultaneously protect a delicate payload. This analysis confirmed that indeed it is possible to do so using a 6-bar tensegrity probe while maintaining maximum decelerations experienced by the instrument-containing payload to forces less than 25G, despite the structure landing at 15 m/s. Comparing this to the Huygens probe's landing acceleration of $32G$ [105], the tensegrtiy probe will have a 43% reduction in G forces experienced by the scientific payload, despite the Huygens probe's use of parachutes to land at 1/3 of the speed of our tensegrity probe.

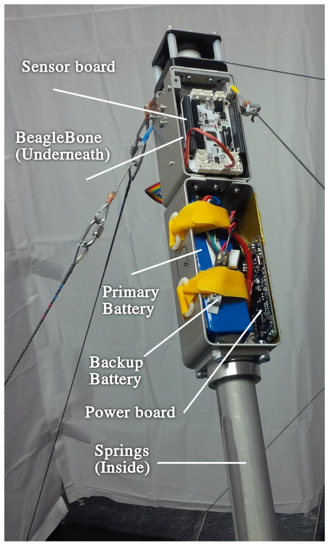
3.7 Validation of NTRT and Real Hardware Prototypes

The NASA Tesnsegrity Robotic Toolkit has also been shown to mimic real world robotic prototypes in regards to kinematics and dynamics as well as learning close loop force controls. Caluwaerts et al. demonstrated a maximum 1.3% position error when comparing rod end positions from motion capture data and NTRT [106]. For dynamics, it was shown that less than 5% time averaged error of each rod end's vertical position in relation of the robot's diameter. To further validate NTRT, Mirletz et al. used the simulator to tune Central Pattern Generator (CPG) coupled impedance controllers through Monte Carlo trials [107]. It was demonstrated that there was a maximum error of 1.6% between the forces seen on their robot vs the forces calculated in the simulator.

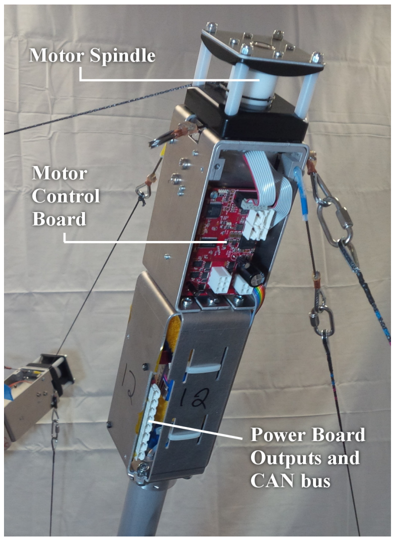
Chapter 4

SUPERball v1 Design

An ideal tensegrity system, either robotic or static, is a collection of rigid compressible elements suspended within a network of tensioned cables where none of the compressible elements are in direct contact with one another. For robotic tensegrities without a payload, the actuation and supporting electronics would be logically designed into the compressible elements. For the inception of SUPERball, this compressible element called a struct was further dissected into three parts: two identical ends called Modular Tensegrity Robots (MTR) and a section of tube stock connecting the MTRs. Connecting six of these struts into an icosahedron geometric pattern will create SUPERball. Modular Tensegrity Robot is a loose label given to the self-contained robotic element which when connected with multiple MTRs may make up a tensegrity robot. The current version of a MTR can be seen in figure 4.1 and the following subsections will explain the mechanical and electrical make up of an MTR.



(a) MTR front side.



(b) MTR back side.

Figure 4.1: Fully assembled Modular Tensegrity Robot images on SUPERball, except for the ground contact caps which will mount to the every end of each rod.

4.1 Mechanical

The main structural elements of the MTRs were kept simple to enable each MTR to be self contained so that the MTR may be removed from the connecting rod as one whole unit. The MTRs are held onto the connecting rods by a simple tube collar for easy removal. There are 5 sections to MTR: a spring holder, battery holder, motor and electronics, cable actuation and routing, and a ground contact. Design parameters are shown in table 3.1 in section 3. The supporting metallic structural elements are made from 6061-T6 aluminum and machined plastic elements are Polyoxymethylene (commercially known as Delrin) unless otherwise noted.

4.1.1 Spring Holder

A lesson learned from other tensegrity robots and the designer of ReCTeR [17] was that externally exposed springs are not ideal for a robotic system that would be interacting with a dynamic and unknown enviroment. The exposed springs get caught on objects and the assumption of near mass-less cables can no longer be applied. On the MTR, an enclosed compression spring system was developed to alleviate these issues. Compression springs were chosen so that during any unknown impact, the springs would not plastically deform before another element in the system would break. For SUPERball, a spring with a spring constant of $998Nm$ is attached to a passive cable element and a $2850Nm$ spring is attached to an actuated cable. A passive spring was chosen with a total throw of $23cm$ to allow for pretension to be instated into the passive springs as well as to allow a wide dynamic compliant range. Since the actuated cables will be able to dynamically control pretension, a smaller throw spring was chosen to conserve space. Figure 4.2 shows a closeup of how the spring holder functions. How cables are attached to the springs inside the spring holder is explained in section 4.1.4.

4.1.2 Battery Holder

From the inception of SUPERball, enabling a self-contained power source which was easily accessible per MTR was a driving design parameter. During the initial design and build of the SUPERball, it was known that the batteries would probably be 24 volt lithium polymer but optimal size and shape of the battery was unknown due to a changing power profile. Therefore, a battery holder with a simple securing mechanism which can handle a wide range of battery sizes was utilized. Two hook and loop straps were used with simple slot cutouts to enable cinching around a generic lithium polymer battery. The holder was also

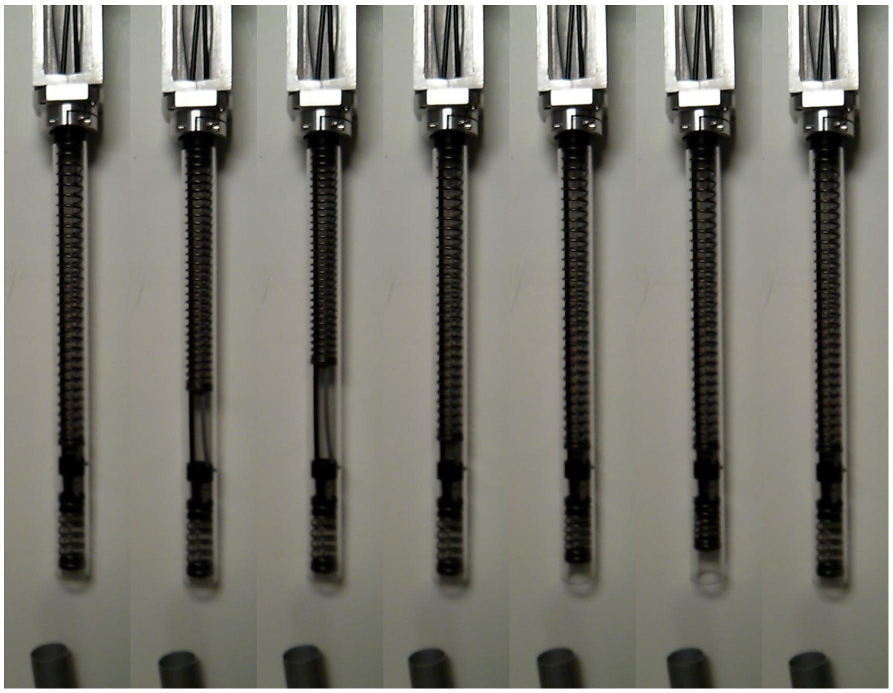


Figure 4.2: Time lapsed stills of spring compression in the MTR spring holder. Note that these stills are from a previous version of the MTR.

made large enough to hold the Power Board PCB opposite of the battery. As shown in section 4.2.3, the Power Board was designed to have a low profile in order to allow for a large battery within the holder.

4.1.3 Motor and Electronics

This section of the MTR used on SUPERball was mechanically designed around the Maxon EC-22 100 watt BLDC motor used for actuation. Each Maxon motor is $22mm$ in diameter and $108mm$ long with gearbox and encoder. The output shaft is a $6mm$ diameter D shaft of length $10.2mm$. A size requirement for how large the cross sectional diameter of the MTR could be was a limiting factor in designing the motor and electronic section. The maximum diameter for any section used in the MTR was maximally limited to double the diameter of the connecting rod. The idea for such a limitation was to keep the effective moment arm out from the center axis of any rod to a minimum. Due to the spring size and the need for a spring holder tube, the minimum diameter for the connecting rod was $35mm$ giving a maximum MTR diameter of $70mm$.

The main component in the motor and electronic section of the MTR is the cable routing support bracket. This bracket plays three roles in the mechanical design: static support for the motor, support for the supporting material, and main exit support for the internal cable routing. Figure 4.3 shows the cable routing support bracket above the pulley. The actual motor mount was designed to be mechanically floating to enable torque sensing directly on the motor mount. Thus, the cable routing support bracket sinks the reaction torque induced by the motor. There are two electronic boards, the Sensor and Motor boards, which are mounted to brackets that straddle the motor. Due to space limitations, these brackets are also load bearing components for the torsional forces induced by the motor.

4.1.4 Cable Actuation and Routing

A simple spool design was implemented to directly actuate the cable. The spool directly couples to the motor shaft by sliding onto the D shaft. A radial bearing supports the top of the spool and since the force vector applied to the spool by the actuated cable will never be just perpendicular to the spool, a thrust bearing was embedded into the bottom of the spool. The thrust bearing sinks the trust force into the motor mount and each spool has the ability to slide along the shaft's main axis. Since this thrust force is perpendicular to the torque of the motor, this force is not induced into the torque sensor built into the motor mount.

There are three other cables that connect to a MTR on SUPERball, though the device may support more with slight additions not explained here. The cables used externally from the MTR are composed of Vectran braided cable and the cables used within the MTR are braided steel cabling. Two cables are routed through the MTR to the spring housing section and the other is terminated on the outside of the MTR. Both routed cables enter the MTR through the cable routing support bracket mentioned in 4.1.3. The cables are immediately routed around a rolling guide bearing to induce an approximate 90 degree bend to guide the cables towards the spring tube holder section, seen in figure 4.3. After the rolling guide bearing, the cables enter a PTFE tube to create a bowden cable to help route the cables around components within the MTR. Once the cables reach their respective spring within the spring tube holder, the PTFE tube is terminated and the cable is routed through the spring and terminated using a copper compression sleeve.

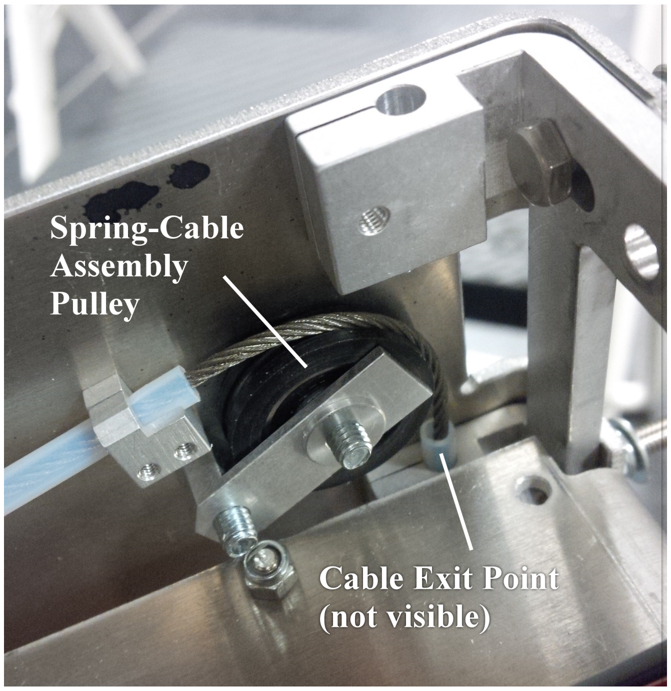


Figure 4.3: Cable routing roller guide within an MTR.

4.1.5 Ground Contact

To protect the MTR during locomotion, a 3D printed cap was manufactured to cover the end. This part is designed with a diameter of $80mm$ so that it is the only part of the MTR that contacts the ground during normal locomotion. To decrease the impact shocks as the rod contacts a surface, compliant foam sheets are place between the 3D printed cap and the MTR.

4.2 Electrical

SUPERball's electronics where developed with a focus on reliability, safety, and enabling distributed controls. Another parameter was the ability to drive the 100W BLDC Maxon motors. These main design criteria gave way to implement separate electronic boards based on their main function. Each MTR has three custom Microchip dspic33e enabled PCB boards and the ability to house one ARM based computer called a Beagle Bone Black. Each custom PCB is designed for very different purposes: A board to condition sensor data and run real-time control loops, a board to condition and distribute a 5.5V electronic power rail and a 24V motor power rail, and a board to control the 100W BLDC motor. The only requirements for each custom board is full CAN bus communication support and power conditioning for the 5.5V power rail. The boards are simply named by their main purpose, thus Sensor, Power, and Motor respectively. Though each MTR can house a Beagle Bone Black, SUPERball only has one per strut for cost saving and initial implementation simplicity.

4.2.1 Motor Board

An initial driving parameter used during the design of SUPERball was the BLDC motor. During the design review for SUPERball, a lightweight motor with high power and efficiency was desired. Thus, a Maxon brushless motor was a logical choice. Table 4.1 shows the electrical properties of this motor. In order to effectively drive this motor, a dedicated motor board was used on each MTR. The main development of this board was engineered by Pavlo Manovi, and certain aspects of the board where tailored for our needs [108]. The main components on the Motor board are the Microchip's 16-bit dsPIC33ep256mu506 micro-controller and the Texas Instruments DRV8303 three phase pre-driver. Figure 4.4a shows the current version of the motor board.

Table 4.1: 100W Maxon BLDC Motor Parameters

Motor without Gearbox				
Nominal Voltage (V)	No Load Speed (rpm)	No Load Current (mA)	Stall Torque	Max Efficiency (%)
24	26,500	16.8	691	92
Gearbox				
Reduction	Number of Stages	Max Continuous Torque (Nm)	Max Peak Torque (Nm)	Max Efficiency (%)
109:1	3	3	3.5	59

4.2.2 Sensor Board and Beagle Bone Black

The sensor board was originally designed as the main processing unit on a MTR. However, the design and building process has lead to the coupling of the sensor board with a Beagle Bone Black. For a detailed explanation of why the Beagle Bone Black was integrated into the system, please refer to 4.3.

The current version of the sensor board was developed as a daughter board for the Beagle Bone Black. The board mates to the Beagle Bone Black through two double row 46 pin headers and provides power and CAN communication to the ARM board. The main processing unit on the sensor board is Microchip's

16-bit dsPIC33ep128gp506 micro-controller. Environmental sensing is enabled by a 9DOF inertial measurement unit (IMU) and a 24-bit analog to digital converter (ADC) configured in a half Wheatstone bridge configuration. The IMU is comprised of Invensense's MPU6000 mastered to Freescale's MAG3110 magnetometer, and the ADC is Analog Device's AD7193.

The Beagle Bone Black is an open-source hardware single-board computer inspired by the Beagle Board, the larger predecessor developed by Texas Instruments as an educational tool. The main processor on the board is a Sitara ARM Cortex-A8 processor running at 1Ghz and capable of running a full ARM based operating system (OS). The processor is also able to interface directly with low level communication protocols such as CAN, UART, SPI, and I2C. To meet our memory and speed requirements, a custom kernel was built with only the modules needed by our system. On top of this kernel, the Beagle Bone Black is running a ROS (Robot Operating System, see section 4.3 for more detials) enabled Ubuntu ARM 14.04.1 LTS OS.

A new feature developed is the integration of DecaWave's DWM1000 module for relative distance measurements. Legacy components no longer utilized on the sensor board for mounting an XBee device are being used with a custom DWM1000 breakout board. The small breakout was designed to integrate the DWM1000 module into where the XBee device was originally mounted. Figure 4.4b shows a sensor board mounted on SUPERball.

4.2.3 Power Board

The power board was designed to enable safety, both for a person working near SUPERball and for the electronics, as well as conditioning input power to both a 5.5V and a 24V rail. The board was also designed with a minimal height profile allowing for larger batteries to be placed near the board. See section 4.1.2, to view the section which houses the power board. The main idea of safety is focused around operating the 100W BLDC motors, thus a two battery system was implemented. A small battery is used for starting the micro-controller boards but not capable of producing 24V needed by the motor, and a large battery is used during main operation of the MTR. The two batteries are a 160 milli-amp-hour 1-cell and a 3 amp-hour 6-cell lithium polymer batteries, named the back-up and the main receptively.

To enable the 24V power, multiple input conditions should be met and fed into an analog and-gate equivalent circuit. The input conditions are: a physical switch located on the MTR, a digital logic pin from the power board's micro-controller, power being applied by the back-up battery, and a signal coming from a dedicated 8-bit micro-controller monitoring a pulsed wireless 2.4Ghz signal. If any one of these conditions go false, the entire 24V rail is disabled. There are also fuses on both the 24V and 5.5V line to protect all the micro-controller circuits from shorts. Figure 4.5 shows a basic connection diagram for the power lines.

The main processing unit on the power board is Microchip's 16-bit mirco-controller, dsPIC33ep128mc506. The wireless "kill switch" monitoring mirco-controller is Mircochip's 8-bit PIC12(L)F1571/2 mirco-controller. This chip monitors a known pulse width being communicated by a Nordic Semiconductor board nRF24L01 with antenna. The pulsed signal is sent by a hand held unit off the robot. When a shut off command is sent or the PIC12's watchdog timer is triggered from a loss in wireless signal, the logic signal sent from the PIC12 is turned to false disabling the 24V power rail. 5.5V power is either supplied by the back-up battery or the main battery using a custom boost or buck switching circuit, receptively. The 24V rail is supplied directly from the main 6-cell battery when all input logic is enabled. Figure 4.4c shows the power board.

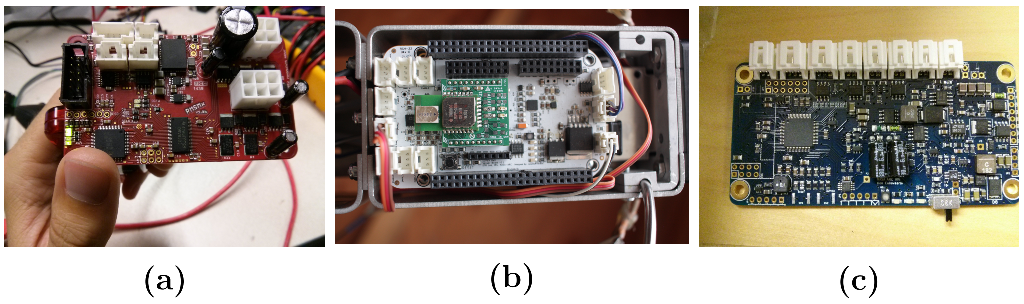


Figure 4.4: Pictures of each of the main micro-controller boards on an MTR. (a) Motor Board, (b) Sensor Board with DWM module, (c) Power Board without nRF24l01 wireless chip.

4.3 Communication

Communication on SUPERball was designed around a desire to have each rod of the tensegrity system unteathered from any other part of the system. Two wireless protocols as well as a wired Controller Area Network, or CAN, bus were implemented. The two main wireless protocols are WiFi for main data communication and a 2.4GHz channel for wireless enabling/disabling of motor power for safety. Figure 4.5 shows how power and communication are connected for a single MTR and figure 4.6 shows the connections for SUPERball's wireless communications.

4.3.1 CAN Bus

A communication design was desired that would be robust, extensible, and work over long distances. A CAN bus fits these main requirements and was implemented to be the main communication between all controllers on a single rod. Since the CAN bus is a physical layer standard, a communication protocol is

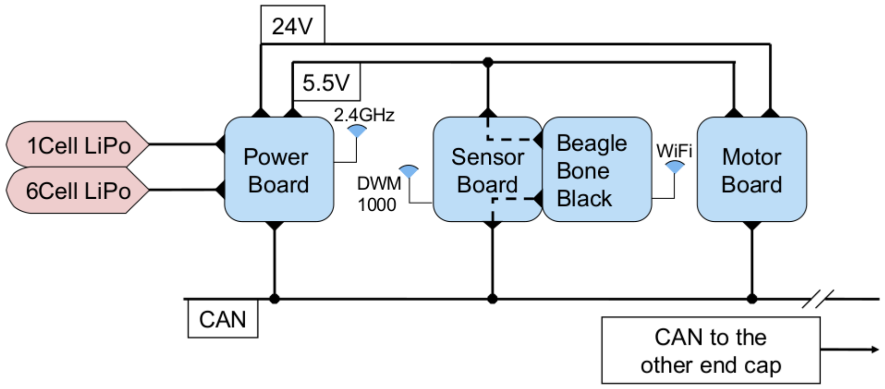


Figure 4.5: This is a connection diagram for power and communication for a MTR on SUPERball.

usually required to get a robust and extensible network. A widely accepted protocol that has been well tested and understood, is the CANOpen protocol [109]. CANOpen defines the addressing scheme, several small communication protocols and an application layer defined by a device profile. Some of the smaller communication protocols supported by CANOpen are device monitoring and communication between nodes, network management, and a simple transport layer for message processing. This open source protocol is freely distributed and has many open and closed source implementations. The CANOpen implementation used for SUPERball is the CANFestival project which focuses on implementing the basic protocol while maintaining a small code base and low computational load for embedded systems. Each mirco-controller and Bealge Bone Black are able to run the entire CANFestival project code in less than $150 \mu s$ under worst case scenarios. The physical layer CAN bus is running at $1 Mbit/s$.

4.3.2 WiFi and the Robot Operating System

The Robot Operating System, or ROS, is a collection of software to provide operating system functionality on a network linked computer cluster. Message-passing and packet management works agnostic to the network layer, allowing information to be passed from one ROS enabled node to any other ROS enabled node on a network. Figure 4.6 shows a basic representation of how this message-passing works on the SUPERball ROS network.

As explained in section 4.2.2, enabling each rod as a ROS node was the driving reason to have at least one ARM based chip on every rod. Since the Beagle Bone Black is also on the CAN bus, it's main function is to sniff the CAN network and send new information out to the ROS network. This enables for near real time data analysis and for time stamped data logging on SUPERball.

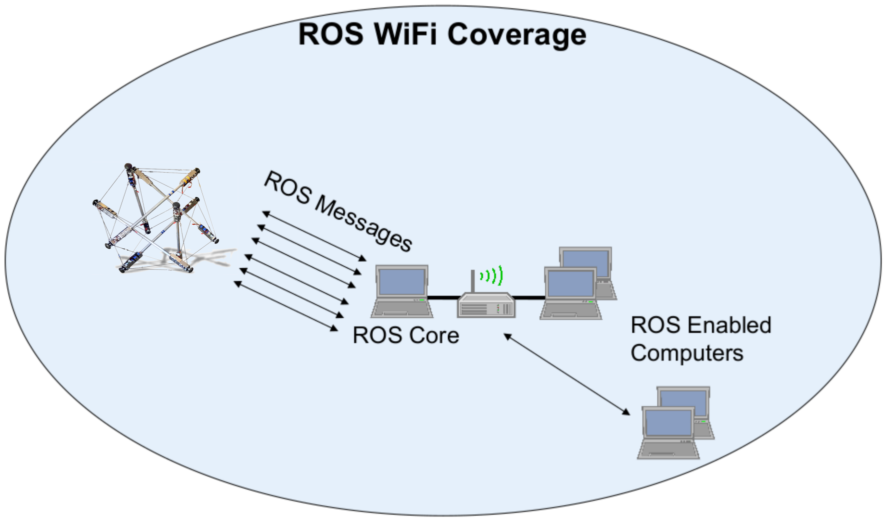


Figure 4.6: A simplified representation of how messages are passed within the SUPERball ROS network.

4.4 First Step

Using a basic step input to a single motor, SUPERball can perform a simple transition from one face of the icosahedron to another. Figure 6.2 shows a test where a motor retracts a cable, inciting a flop. The idea of this type of simple transition is to deform the base equilateral triangle such that the center of mass

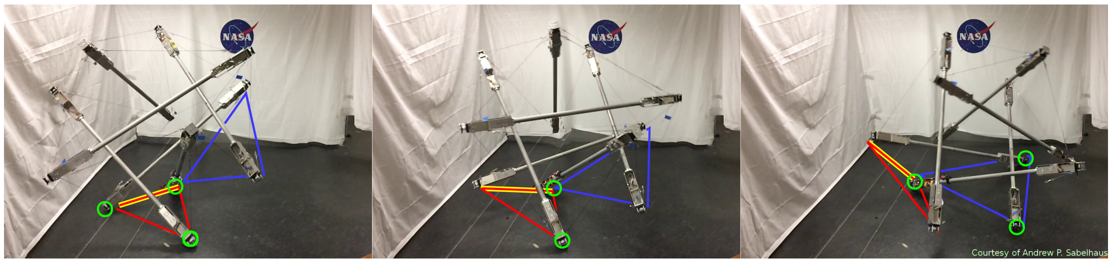


Figure 4.7: SUPERball performing a single face-change movement, from one equilateral triangular face to another. The robot begins with all MTRs of the red triangle touching the ground. Then, SUPERball retracts the yellow-highlighted cable on the red triangle, inducing movement. Frame 2 shows SUPERball halfway through the movement with only two points of contact on the ground. Finally, frame 3 shows SUPERball at the end, with all 3 points of the blue triangle in ground contact.

"moves" over the triangle's edge. The robot becomes unstable and gravity pulls the system over. The momentum of the system then rolls the robot through the adjacent isosceles triangle to the next equilateral triangle. In this test, the motor retraction was preset and experimentally derived earlier. A more in depth explanation of basic locomotion on SUPERball can be found in section 6.1.

Chapter 5

State Estimation

5.1 Ranging Setup and Calibration

This section introduces the hardware and software setup for a set of wireless ranging modules to enable position tracking of the robot both as an as internal distance measurements (end cap to end cap) an in an external (world) reference frame.

All MTRs of SUPERball are equipped with a DWM1000 ranging module from DecaWave Ltd. By employing ultra wideband technology, the low-cost DWM1000 modules provide wireless data transfer and highly accurate timestamps of transmitted and received packets. This allows the distance between two DWM1000 modules to be estimated by computing the time-of-flight of exchanged messages without the need for synchronized clocks. Using DWM1000 modules external to SUPERball as "fixed anchors" and placing them around the testing area, a world reference frame for ground truth and generation of a reward signal for the machine learning algorithms used for learning locomotion is obtained. It is intended that the final deployed version of the robot and controller will not require fixed anchors, and they are primarily used during algorithm development.

5.1.1 Sensor Operation

Bidirectional Ranging

The DWM1000 modules are operated in the so-called *symmetric double-sided two-way ranging* mode. In this mode, the modules exchange 3 packets to estimate the time-of-flight between each other. While the time-of-flight of unsynchronized modules can be estimated with the exchange of only 2 packets, the employed mode can significantly reduce measurement noise [110].

The basic ranging packet exchange is shown in Fig. 5.1. Module 1 sends out a *poll* message containing an emission timestamp (t_{SP}) using its local clock. Module 2 receives this message and timestamps the time of arrival using its local clock (t_{RP}). Then, module 2 sends out a *response* packet at time t_{SR} (module 2's clock). Module 1 receives this packet at time t_{RR} (module 1's clock). Module 1 now sends out a final message containing t_{RR} and the emission time of the final message (t_{SF}, clock of module 1). Module 2 receives this information and timestamps it (t_{RF}).

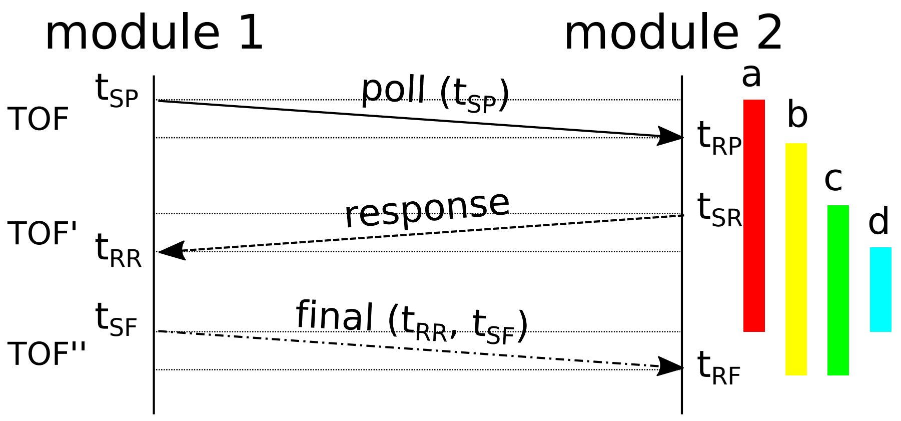


Figure 5.1: Basic symmetric double-side two-way ranging packet exchange. Modules 1 and 2 exchange 3 packets (*poll*, *response*, and *final*). Module 2 then estimates the distance between the modules based on the local timestamps.

Module 2 can now estimate the time-of-flight and the distance between itself and module 1 based on the 6 timestamps. The basic equations to estimate the distance between module i and module j (module i initiates the ranging and module j computes the distance) are given by:

$$a_i = t^i_{SF} - t^i_{SP} \tag{5.1}$$

$$b_{j,i} = t^{j,i}_{RF} - t^{j,i}_{RP} \tag{5.2}$$

$$c_{j,i} = t^{j,i}_{RF} - t^j_{SR} \tag{5.3}$$

$$d_{i,j} = t^i_{SF} - t^{i,j}_{RR} \tag{5.4}$$

$$TOF_{j,i} \approx \frac{1}{2}\left(c_{j,i} - d_{i,j}\frac{b_{j,i}}{a_i}\right) - \delta_{j,i} \tag{5.5}$$

$$\|\boldsymbol{N}_j - \boldsymbol{N}_i\| \approx \frac{1}{2\boldsymbol{C}}\left(c_{j,i} - d_{i,j}\frac{b_{j,i}}{a_i}\right) - o_{j,i} \tag{5.6}$$

$$\doteq m_{j,i} - o_{j,i}. \tag{5.7}$$

The variables a, b, c, and d are also visualized in Fig. 5.1. The time-of-flight calculation between two modules i and j ($TOF_{j,i} = TOF_{i,j}$) is hindered by a fixed measurement offset ($\delta_{j,i} = \delta_{i,j}$). This offset is due to antenna delays and other discrepancies between the timestamps and actual packet reception or emission. Whereas this offset is expected to be unique to each module, it was found that it is necessary to estimate this offset pairwise for closely located modules. The hypothesis is that the proximity of the robot's motors and the sensor's position near the end cap's metal structure influences the antenna characteristics between pairs of modules.

Eq. 5.6 estimates the distances between the modules based on the time-of-flight calculation ($\boldsymbol{C}$ is the speed of light). Rewriting the time offset $\delta_{j,i}$ as a distance offset $o_{j,i}$ (with $o_{j,i} = o_{i,j}$). Here $\boldsymbol{N}_i$ and $\boldsymbol{N}_j$ refer to the positions of nodes i and j respectively (see Section 5.2). The variables $m_{j,i}$ represent the uncorrected distance estimates.

The DWM1000 requires careful configuration for optimal performance. The main configuration settings are provided in Table 5.1. The ranging modules tend to measure non line-of-sight paths near reflective surfaces (e.g. floor, computer monitors), which may cause filter instability. Using the DWM1000's built-in signal power estimator, such suspicious packets are rejected. In practice, between 30% and 70% of packets are rejected.

Table 5.1: DWM1000 configuration

bitrate	**channel**	**preamble**	**PRF**	**preamble code**
$6.8\,\mathrm{Mbit\,s^{-1}}$	7	256	64 MHz	17

Broadcast Ranging

Due to the large number of exchanged packets (3 per pair) bidirectional ranging between pairs of modules quickly becomes inefficient when the number of modules grows. An alternative approach was developed using timed broadcast messages that scales linearly in the number of modules (3 packets per module). In this setup one module periodically initiates a measurement sequence by sending out a *poll* message. When another module receives this message it emits its own *poll* message after a fixed delay based on its ID, followed by *response* and *final* messages after additional delays. Broadcast ranging is illustrated in Fig. 5.2.

One disadvantage of the broadcasting approach is that the total measurement time between a pair of modules takes longer (up to 60 ms in the experimental setup) than a single pairwise bidirectional measurement (approx. 3 ms). However, broadcast ranging provides two measurements for each pair of modules per measurement iteration.

Note that each module now needs to keep track of the *poll* and *final* packet

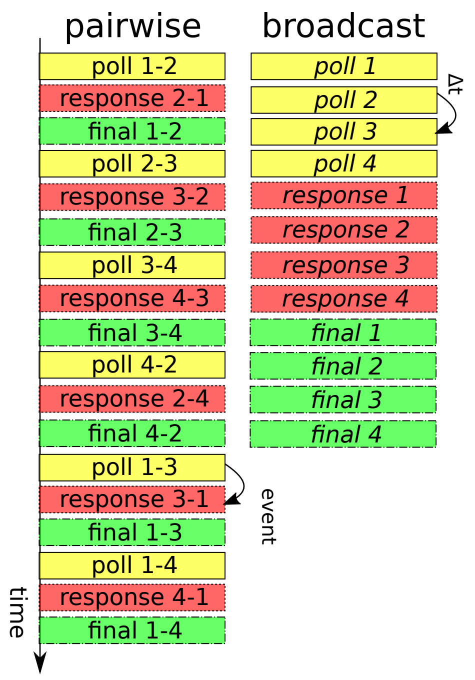


Figure 5.2: Packet exchange between 4 modules for bidirectional pairwise and broadcast ranging. Timed broadcast messages allow for efficient ranging with a large number of modules.

reception times of all other modules. The *final* packet becomes longer as each module needs to transmit the *response* reception time (t_{RR}) of all other modules.

5.1.2 Ranging Setup

Each MTR of SUPERball was fitted with a DWM1000 module located approximately 0.1 m from the end of the strut. To simplify the notation, the top of the MTRs (ends of the struts) and the position of the ranging sensor are assumed the same. In practice, this offset is taken into account in the output function of the filter (see Section 5.2).

The broadcasting algorithm runs at 15 Hz and packet transmissions are spaced 1 ms apart. This allows for over 20 modules to range. After one ranging iteration, each end cap transmits its measurements over WiFi to the ROS network. A ROS node then combines measurements from all MTRs, along with encoder and IMU data, into a single ROS message at 10 Hz.

The fixed anchors operate in a similar way to the end caps, but are not connected to a ROS node and can not directly transmit data to the ROS network. This means that two measurements are obtained (one in each direction) for each pair of modules on the robot, but only a single measurement between the fixed anchors and the modules on the robot.

5.1.3 Calibration

One of the design goals of this state estimation method is quick deployment in new environments without significant manual calibration. To achieve this, an automatic calibration procedure was implemented to jointly estimate the constellation of fixed modules (anchors, defining an external reference frame) and the pairwise sensor offsets ($o_{i,j}$). Calibration is performed - similar to common motion capture systems - by moving the robot around, while recording the uncorrected distance measurements ($m_{j,i}$).

After recording a dataset, reconstruction error is minimized L by optimizing over the offsets $\boldsymbol{o}$ ($o_{i,j}$ rearranged as a vector), the estimated anchor locations $\boldsymbol{N}^{est}$, and the estimated moving module locations $\boldsymbol{N}^{float}[1\ldots n_{samples}]$ (i.e. the module on the robot's end caps):

$$L\left(i,j,t\right) = \left(\|\boldsymbol{N}_i^{anchor} - \boldsymbol{N}_j^{float}\left[t\right]\| - o_{j,i} - m_{i,j}\left[t\right]\right)^2 \tag{5.8}$$

$$L\left(\boldsymbol{o}, \boldsymbol{N}^{anchor}, \boldsymbol{N}^{float}[1\ldots n_{samples}]\right) = \textstyle\sum_{i,j,t} \alpha_{j,t} L\left(i,j,t\right). \tag{5.9}$$

The brackets in $\boldsymbol{N}^{float}[1\ldots n_{samples}]$ indicate the moving module locations (MTR positions) at a specific timestep. For example $\boldsymbol{N}^{float}[5]$ contains the estimated end cap positions at timestep 5 in the recorded dataset. In Eq. 5.9, i iterates over anchors, j iterates over moving nodes and t iterates over samples. The indicator variables $\alpha_{j,t}$ are equal to 1 when for sample t there are at least 4 valid measurements to the fixed module for moving module j (i.e. the number of DOFs reduces).

In practice, constraints are added on the bar lengths, which take the same form as Eq. 5.8 with the offsets set to 0. BFGS [111] is used to minimize Eq. 5.9 with a dataset containing approximately 400 timesteps selected randomly from a few minutes of movement of the robot. Although the algorithm works without prior knowledge, providing the relative positions of 3 fixed nodes (3 manual measurements) significantly improves the success rate as there are no guarantees on global convergence.

Once the external offsets (between the anchors and moving nodes) and the module positions are known, the offsets can be estimated between moving nodes in a straightforward way by computing the difference between the estimated internal distances and the uncorrected distance measurements.

5.2 Filter Design

Tensegrity systems are nonlinear and exhibit hybrid dynamics due to cable slack conditions and interactions with the environment that involve collision and friction. This warrants a robust filter design to track the robot's behavior.

The commonly used Extended Kalman Filter (EKF) does not perform well on highly nonlinear systems where first-order approximations offer poor representations of the propagation of uncertainties. Additionally the EKF requires computation of time-derivatives through system dynamics and output functions which is challenging for a model with complex hybrid dynamics.

The sigma point Unscented Kalman Filter (UKF) does not require derivatives through the system dynamics and is third order accurate when propagating Gaussian Random Variables through nonlinear dynamics [112]. The computational cost of the UKF is comparable to that of the EKF, but for tensegrity systems which commonly have a large range of stiffnesses and a high number of state variables the time-update of the sigma points dominates computational cost. As such, the methods used to reduce computational cost of dynamic simulation will be described, then in the following section the outline of the specific UKF implementation for the SUPERball prototype.

5.2.1 Dynamic Modeling

The UKF requires a dynamic model which balances model fidelity and computational efficiency since it requires a large number of simulations to be run in parallel. The model implemented for the tensegrity system is a spring-mass net and the following incomplete list of simplifying assumptions where used:

- Only point masses located at each node point
- All internal and external forces are applied at nodes
- Members exert only linear stiffness and damping
- Unilateral forcing in cables
- Flat ground at a known height with Coulomb friction
- No bar or string collision modeling

For a tensegrity with n nodes and m members, the member force densities, $\boldsymbol{q} \in \mathbb{R}^m$, can be transformed into nodal forces, $\boldsymbol{F_m} \in \mathbb{R}^{n\times 3}$, by using the current

Cartesian nodal positions, $\boldsymbol{N} \in \mathbb{R}^{n\times 3}$, and the connectivity matrix, $\boldsymbol{C} \in \mathbb{R}^{m\times n}$, as described in [113]. This operation is described by the equation:

$$\boldsymbol{F_m} = \boldsymbol{C}^T diag(\boldsymbol{q})\boldsymbol{C}\boldsymbol{N},$$

where $diag(\cdot)$ represents the creation of a diagonal matrix with the vector argument along its main diagonal. First, note that $\boldsymbol{C}\boldsymbol{N}$ produces a matrix $\boldsymbol{U} \in \mathbb{R}^{m\times 3}$ where each row corresponds to a vector that points between the ith and jth nodes spanned by each member. Therefore, this first matrix multiplication can be replaced with vector indexing as $\boldsymbol{U}_k = \boldsymbol{N}_i - \boldsymbol{N}_j$, where the notation $\boldsymbol{U}_k$ is used to denote the kth row of matrix $\boldsymbol{U}$. If one then computes $\boldsymbol{V} = \boldsymbol{C}\frac{d\boldsymbol{N}}{dt}$ with the same method as $\boldsymbol{U}$, one would obtain a matrix of relative member velocities. The matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are used to calculate member lengths as $L_k = |\boldsymbol{U}_k|_2$ and member velocities as $\frac{d}{dt}(L_k) = \frac{\boldsymbol{U}_k(\boldsymbol{V}_k)^T}{L_k}$.

Member force densities, $\boldsymbol{q}$, are then calculated using Hooke's law and viscous damping as:

$$\boldsymbol{q}_k = K_k(1 - \frac{L_{0k}}{L_k}) - \frac{c_k}{L_k}\frac{d}{dt}(L_k).$$

Here K_k and c_k denote the kth member's stiffness and damping constants, respectively. Note that cables require some additional case handling to ensure unilateral forcing.

Scaling each $\boldsymbol{U}_k$ by $\boldsymbol{q}_k$ yields a matrix whose rows correspond to vector forces of the members. Denote this matrix as $\boldsymbol{U}^q \in \mathbb{R}^{m\times 3}$, and note that $\boldsymbol{U}^q = diag(\boldsymbol{q})\boldsymbol{C}\boldsymbol{N}$. Thus this matrix of member forces can be easily applied to the nodes using:

$$\boldsymbol{F_m} = \boldsymbol{C}^T\boldsymbol{U}^q.$$

A method for computing nodal forces exerted by the members is now obtained, and only ground interaction forces need to be computed, which will be denoted as $\boldsymbol{F}_g$. Ground interaction forces were computed using the numerical approach in [114]. The nodal accelerations can then be written as:

$$\frac{d^2\boldsymbol{N}}{dt^2} = \boldsymbol{M}^{-1}(\boldsymbol{F}_m + \boldsymbol{F}_g) - \boldsymbol{G},$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose diagonal entries are the masses of each node and $\boldsymbol{G} \in \mathbb{R}^{n\times 3}$ is matrix with identical rows equal to the vector acceleration due to gravity. It is then straightforward to simulate this second order ODE using traditional numerical methods.

Note also that it is possible to propagate many parallel simulations efficiently by concatenating multiple $\boldsymbol{N}$ matrices column wise to produce $\boldsymbol{N}_{\|} \in \mathbb{R}^{n\times 3l}$ for l parallel simulations. The resultant vectorization of many of the operations yields significant gains in computational speed with some careful handling of matrix dimensions.

5.2.2 UKF Implementation

A traditional UKF was implemented as outlined in [112] with additive Gaussian noise for state variables and measurements.

Several parameters are defined for tuning the behavior of the UKF, namely α, β and κ, where α determines the spread of the sigma points generated by the unscented transformation, β is used to incorporate prior knowledge of distribution, and κ is a secondary scaling parameter. Hand tuning obtained these parameters to the values $\alpha = 0.0139$, $\beta = 2$ for Gaussian distributions and $\kappa = 0$ and found this to yield an adequately stable filter.

Defining state variables as $\boldsymbol{N}$ and $\frac{d\boldsymbol{N}}{dt}$ stacked in a vector $\boldsymbol{y} \in \mathbb{R}^L$ where $L = 6n$

is the number of state variables. Also, independent state noise is assumed with variance $\lambda_y = 0.4$ thus with covariance $\boldsymbol{R} = \lambda_y \boldsymbol{I}_L$.

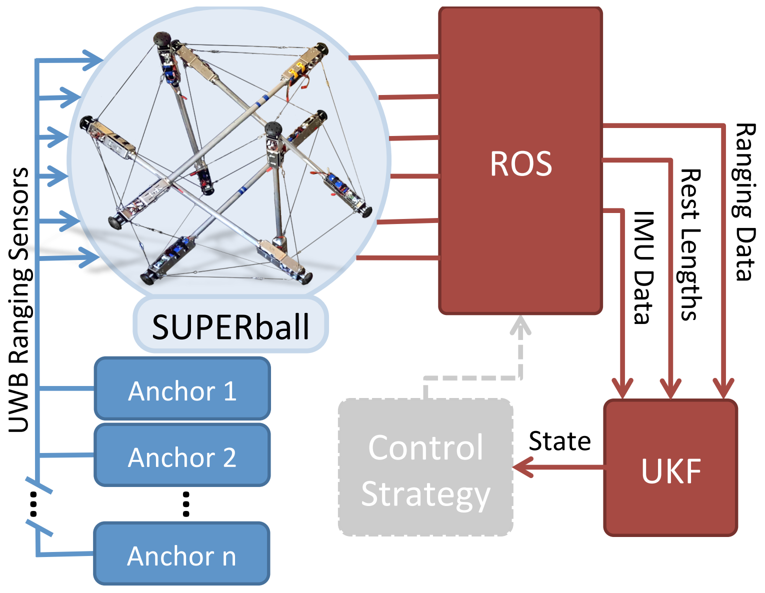


Figure 5.3: Block diagram of data flow within the system. Red signals are passed as ROS messages and blue signals are passed using the ranging modules. Note that each rod contains two ranging sensors located at each end of the rod. The gray control strategy block represents a to-be-designed state-feedback control strategy.

The measurement data used is estimated orientation data from the robot's IMUs using a gradient descent AHRS algorithm based on [115], $\theta \in \mathbb{R}^b$ where b is the number of bar angles available at the given time step and all ranging measures, $\boldsymbol{r} \in \mathbb{R}^a$, where a is the number of ranging measures available at a given time step. Independent noise is again assumed and represented by λ_θ and λ_r. The measurement covariance matrix is then defined as:

$$\boldsymbol{Q} = \begin{bmatrix} \lambda_\theta \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_r \boldsymbol{I}_a \end{bmatrix}.$$

These user defined variables are then used within the framework of the UKF to forward propagate both the current expected value of the state as well as its covariance. Fig. 5.3 shows an overview of the complete state estimation setup.

5.3 Filter Evaluation

5.3.1 Experimental Setup

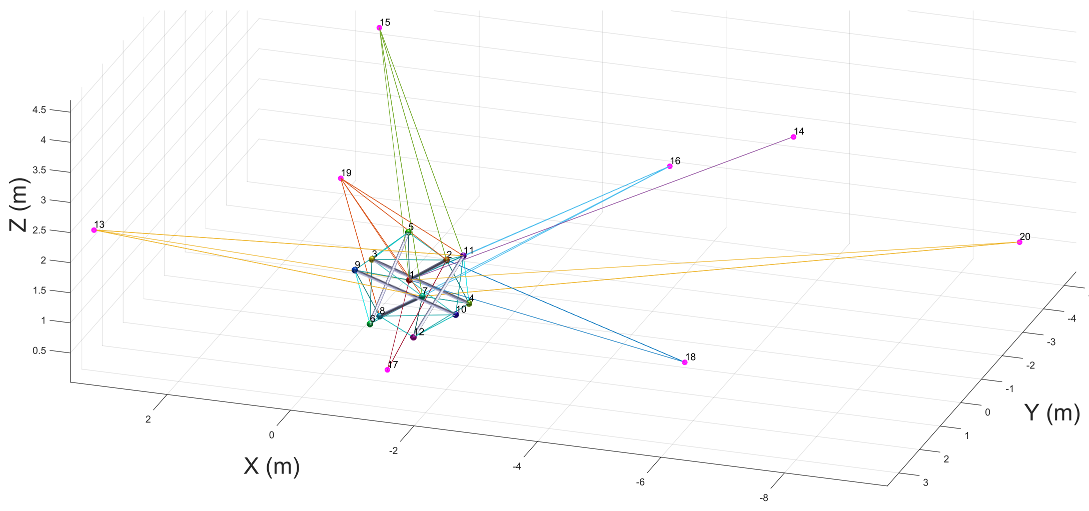


Figure 5.4: Visualization of the UKF output. SUPERball sits in the middle of the plot surrounded by 8 ranging base stations. Lines between the robot and the base stations indicate valid ranging measures during this timestep.

To evaluate the performance of the UKF, eight "fixed anchor" ranging base stations are used and calibrated as detailed in Section 5.1.3. Each end cap of SUPERball was then able to get a distance measurement to each base station. This information was sent over ROS along with IMU data (yaw,pitch,roll) and cable rest lengths to the UKF. The base stations were placed in a pattern to cover an area of approximately $91\,\mathrm{m}^2$. Each base station's relative location to each other may be seen in Fig. 5.4. SUPERball and the base stations were then used to show the UKF tracking a local trajectory of end caps and a global trajectory of the robotic system. In each of these experiments, the UKF was allowed time to settle from initial conditions upon starting the filter. This ensured that any erroneous states due to poor initial conditioning did not affect the filter's overall performance.

5.3.2 Local Trajectory Tracking

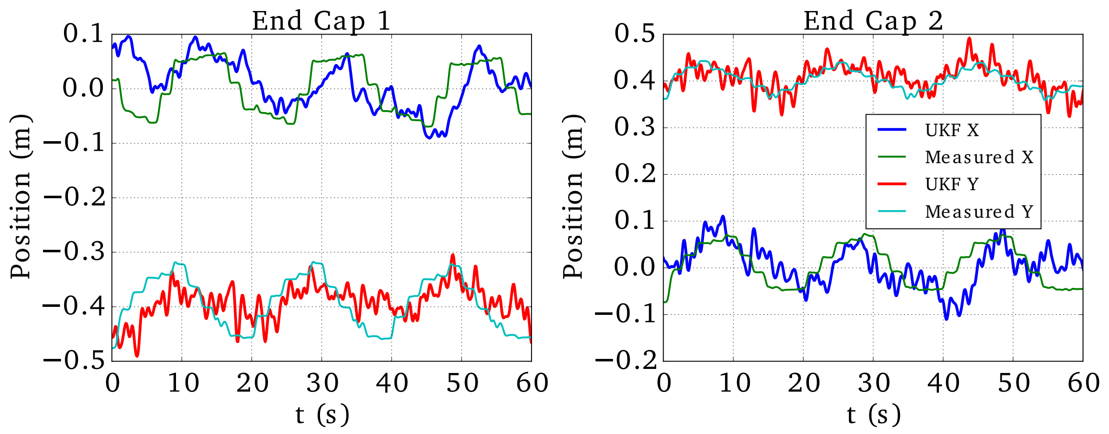


Figure 5.5: Position plotted through time for both end cap 1 and end cap 2. The thin line represents the position output measured by the camera tracking system, and the bold line represents the position output from the UKF filter. As expected, there is a time domain lag between the measured and estimated positions.

In order to track a local trajectory, SUPERball remained stationary while two of its actuators tracked phase shifted stepwise sinusoidal patterns. During the period of actuation, two end cap trajectories were tracked on SUPERball and compared to the trajectory outputs of the UKF. One end cap was directly connected to an actuated cable (end cap 2), while the other end cap had no actuated cables affixed to it (end cap 1). To obtain a ground truth for the position trajectory, a camera that measured the position of each end cap was positioned next to the robot. Both end caps started at the same relative height and the majority of movement of both fell within the plane parallel to the camera. Fig. 5.5 shows the measured and UKF global positions of the two end caps through time.

5.3.3 Global Trajectory Tracking

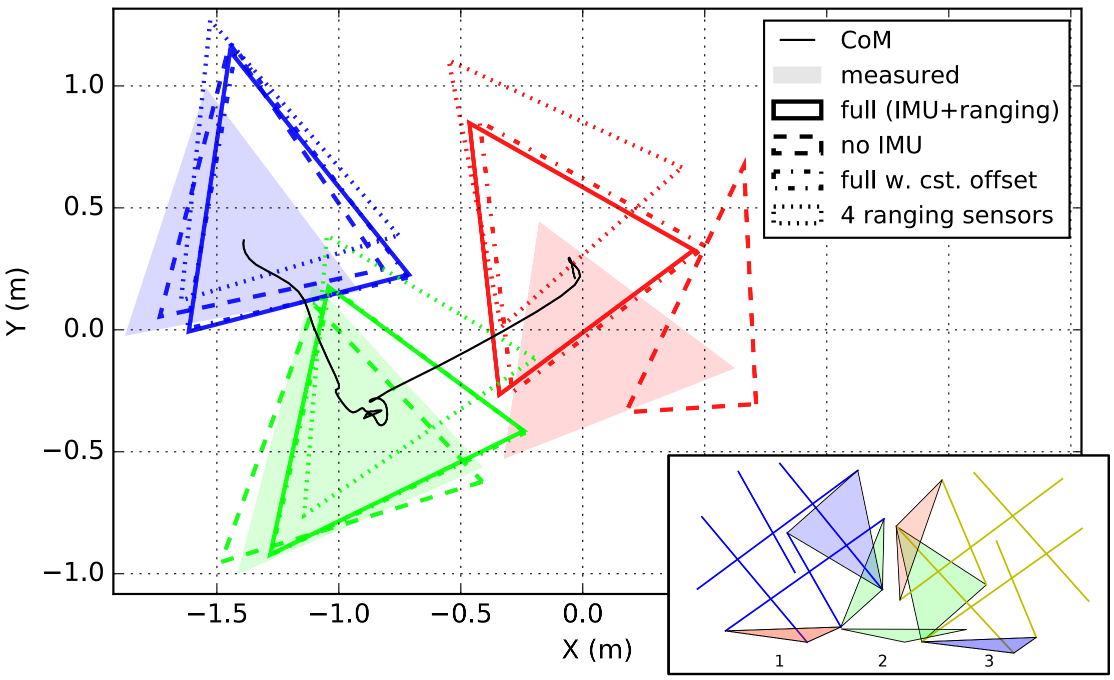


Figure 5.6: Top down view of the triangular faces to which the robot transitions during the global trajectory tracking experiment for various setting of the state estimator. The small inset illustrates the movement of the robot. The line shows the estimated center of mass (CoM) using the *full* settings. Finding the initial position (origin) is hard for all settings, and without the IMUs the estimator does not find the correct initial face. After a first roll, tracking becomes more accurate. The offsets $\boldsymbol{o}$ have a minimal impact, which indicates that the calibration routine is sufficiently accurate.

For global trajectory tracking, SUPERball was actuated to induce a transition from one base triangle rolling through to another base triangle as presented in [5]. Ground truth for this experiment was ascertained by marking and measuring the positions of each base triangle's end caps before and after a face transition. 4 settings of the state estimator were evaluated. *Full*: The state estimator as described in Section 5.2 with all IMU and ranging sensors. *no IMU*: Only the ranging sensors are enabled. *full w. cst. offset*: Same as *full*, but the offsets $\boldsymbol{o}$ are set to a constant instead of optimized individually. *4 base station ranging* *sensors*: 50% of the base station ranging sensors are disabled. The results of this experiment are presented in Fig. 5.6 and 5.7.

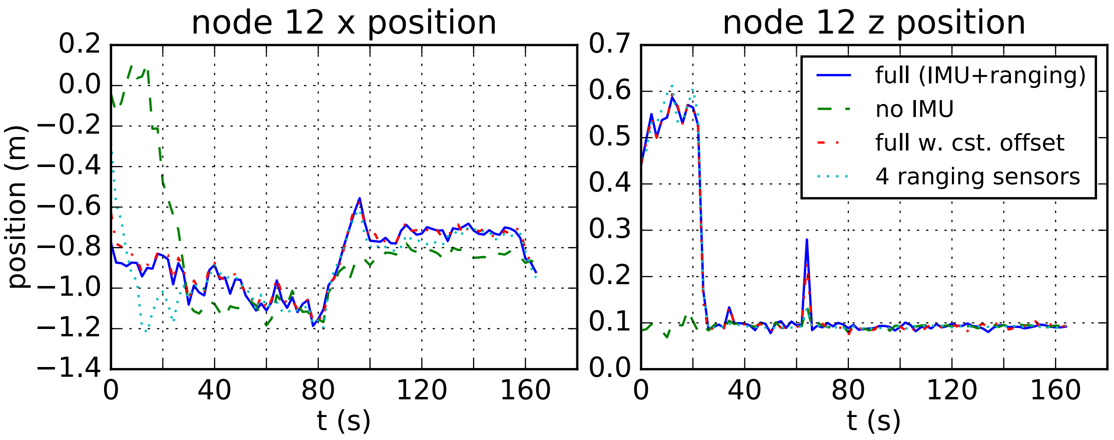


Figure 5.7: X and Y position of end cap 12 as a function of time for the various estimator settings. The end cap was initially off the ground and touches the ground after the first roll. This is not tracked correctly when the IMUs are disabled. The system works as expected when 4 base stations ranging sensors are disabled, but with slower convergence and more noise on the robot's position. Around 60 s there's a spurious IMU value from which the state estimator recovers.

Chapter 6

Control for SUPERball Locomotion

6.1 Basic Locomotion Concepts for a Icosahedron Tensegrity Robot

Locomotion for tensegrity structures like SUPERball is achieved by deforming the structure in a way in which moves the system's center of mass to an unstable configuration, tipping the robot over. This deformation is usually achieved by either changing the length of the main cable network on the outside of the robot [5, 98] or by adding additional cables which run through the structure connecting non-parallel rods [106]. For the rest of this section, deformation is assumed to be done by actuating the main cable network on the outside of the robot since this is how SUPERball is deformed.

A regular convex icosahedron is a geometric shape consisting of eight equilateral triangles interlaced with twelve isosceles triangles. In a passively stable configuration, the bottom of an icosahedron will be resting on either an equilat-

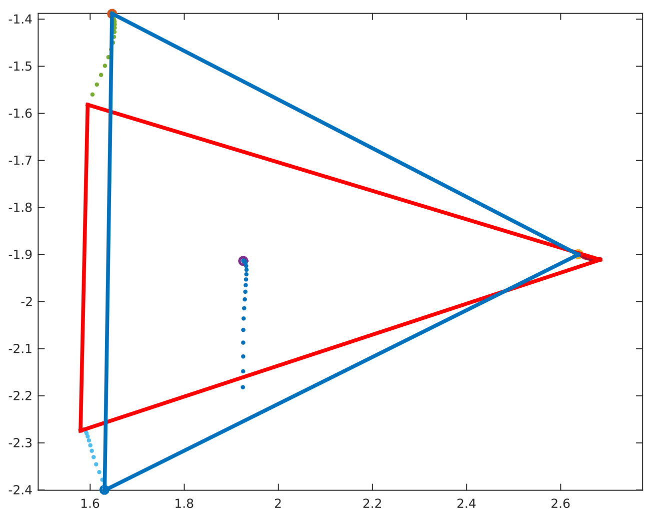


Figure 6.1: This is XY data of the bottom triangle of a NTRT simulation of SUPERball performing a single face transition by changing only one side of the bottom triangle. No other cable on the system is being actuated during this simulation. The blue triangle is the bottom triangle at the start of the simulation. The centralized circle represents the center of mass (CoM) of the entire simulated SUPERball and the dots represents how the CoM moves through time. The red triangle is the configuration where the CoM moves out of the bottom triangle and the robot begins to transition to another face.

eral or an isosceles triangle. Utilizing this knowledge, the most simple method for moving the center of mass of an icosahedron can be obtained by changing the length of one side of the bottom triangle to near zero. This will always move the center of mass to an unstable configuration by effectively reducing the bottom triangle, as seen if figure 6.1, which will cause the robot to transition to another face. However, this simple control method is not always obtainable on a real robotic system due to limitation on actuation or design methodology. For SUPERball, this control method is obtainable when the system only has the battery mounted required to actuate the single motor reducing the overall system weight. Figure 6.2 shows this single motor face transition on a weight reduced SUPERball.

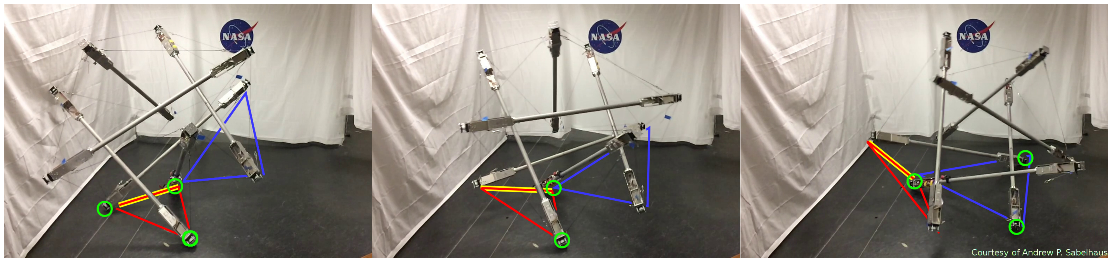


Figure 6.2: SUPERball performing a single face-change movement, from one equilateral triangular face to another. The robot begins with all MTRs of the red triangle touching the ground. Then, SUPERball retracts the yellow-highlighted cable on the red triangle, inducing movement. Frame 2 shows SUPERball halfway through the movement with only two points of contact on the ground. Finally, frame 3 shows SUPERball at the end, with all 3 points of the blue triangle in ground contact.

6.1.1 SUPERball Acutation Pattern

SUPERball is an underactuated icosahedron tensegrity robot. Of the 24 connection cables, SUPERball only has 12 cables which are actively actuated and the other cables are passive as discussed in section 4. Since each MTR is manufactured with the same elements, each of the four cables attached to it are one of four types: an actuated cable attached to a motor, an actuated cable from an adjacent MTR terminating at this MTR, a passive cable attached to a spring, or a passive cable from an adjacent MTR terminating at this MTR. Therefore, there is a unique pattern of cables, and care is needed when choosing this pattern for locomotion.

For SUPERball, a symmetric pattern is used where each equilateral triangle has at least one actuator associated with one of its sides. As stated in section 4, SUPERball has eight equilateral triangles, so at most three triangles will have more than one actuated side. These triangles are evenly spaced round the surface of the robot such that there are "rings" of six equilateral triangles with the other two triangles flanking this ring. Forward locomotion is then achieved by transi-

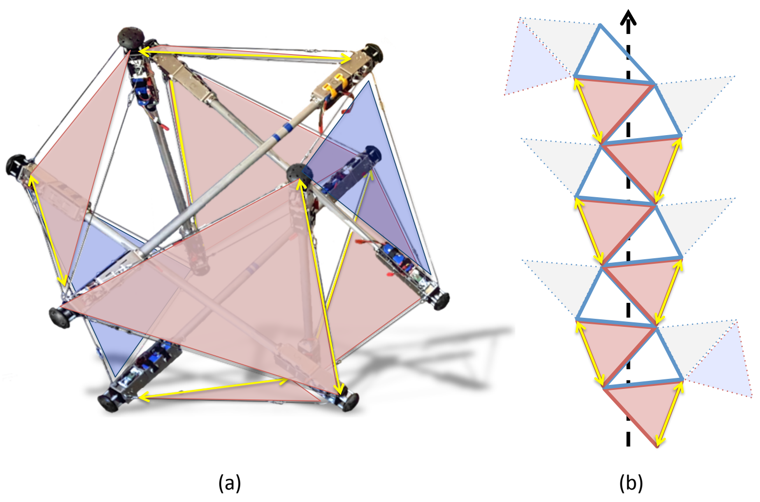


Figure 6.3: This is the actuation pattern used on SUPERball. There are 8 equilateral triangles, shown in either red or blue. Each red triangle represents a face with only one cable actuated. These actuated cables are denoted by the yellow double arrows. Each blue triangle represents a face with actuators on all cables. The basic forward rolling of SUPERball has the robot landing with red triangles on the ground during locomotion. Figure (a) shows each triangle highlighted on SUPERball and figure (b) shows the forward locomotion pattern, or walking pattern, of SUPERball.

tioning the structure such that each of the six equilateral triangles on this "ring" comes in complete contact with the ground at some instantaneous point in time. Since a symmetric pattern was desired, it was chosen to place one actuator per "ring" triangle in such a way where the theoretical full actuation length change of that triangle side would cause the robot to transition towards the next sequential equilateral triangle on that "ring". The other six motors were then attached at all the side of the remaining two equilateral triangles, those not associated with the "ring". Figure 6.3 (a) shows a graphical overlay of the actuation "ring" and the fully actuated triangles marked in red and blue, respectively. Figure 6.3 (b) is a icosahedron rolled out to show the "walking" gait pattern achieved by rolling about the actuation "ring".

6.1.2 Basic Steering Controller

The majority of this chapter deals with how to achieve a continuous forward locomotion gait for SUPERball. However, having the ability to turn makes navigation a bit more interesting and this section will discuss some preliminary results in achieving a simple left and right turning gait. It has been shown in literature that a simulated fully actuated (24 actuators) SUPERball like robot can achieve left and right turning gaits [98] as well as goal direction navigation [103].

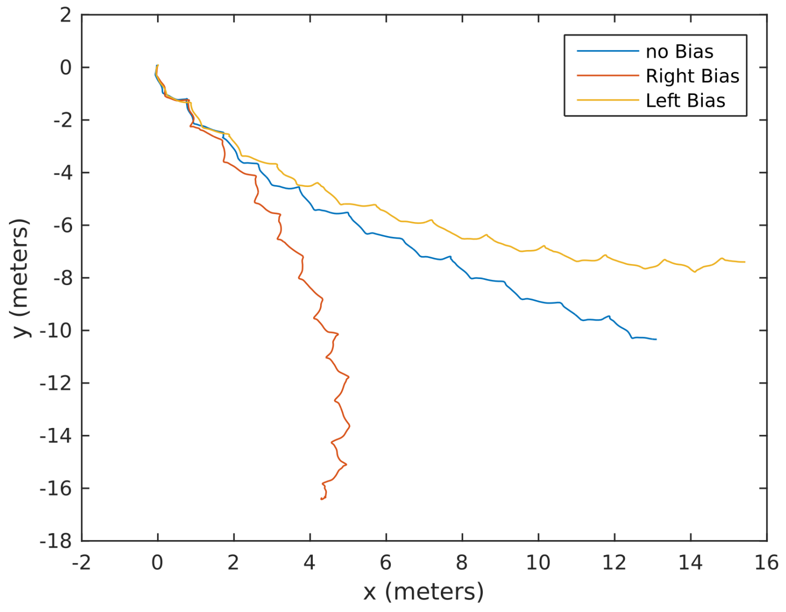


Figure 6.4: This figure shows the tracked center of mass of a simulated SUPERball utilizing the basic steering controller. Each fully actuated equilateral triangle was squeezed the same amount for the left and right steering. This controller has no feed back and does have a slight bias toward turning right more than left. Also, it is noted that the unbiased controller tends to the left, because of no optimization on direction was used during the learning of the controller. It should be noted that this figure only shows an academic example and not a fully functional controller.

The current version of SUPERball has limited actuation as stated in section 6.1 and can not perform the published turning gaits. A simple solution which can be implemented with any type of forward locomotion controller involves the use of the two fully actuated equilateral triangles outlined in section 6.1.1 and shown in figure 6.3. Actuating all three motors for a given triangle equal amounts will bias the gait such that the roll performs a wide turn. This equal actuation of the triangle on the left side biases the robot left, and conversely the right triangle biases the robot right. This behavior can be seen in figure 6.4.

6.2 Hand-Tuned Stepwise Controller

The initial controller developed for SUPERball was a basic open loop, hand tuned controller. Motor position commands were systematically found through experimentation which moved the robot into a kinematically unstable configuration for each of the six faces mentioned in section 6.1. Under normal conditions on flat ground, when the system starts on an equilateral triangle the forward momentum of the structure after deformation will push it through the isosceles triangle and come to rest on another equilateral triangle. Using this assumption, only six different kinematic configurations were implemented. To automate this process, enabling the system to detect which face it resided on was necessary. A simple K-nearest neighbor algorithm was implemented on recorded IMU data for each equilateral triangle face of SUPERball. Since the faces are discrete enough, one hundred percent classification was found. With this information, a basic open loop controller was written that used the detected face as an input and commanded the correct motor commands for the kinematically unstable configuration to transition the robot to the next face.

Once the robot acted the kinematically unstable configuration, all motor commands were set back to their starting configuration. This ensured correct detection and transition time for the next cycle. The hand-tuned stepwise controller has two main states, a detect and move state and a relax state as seen in 6.5. The

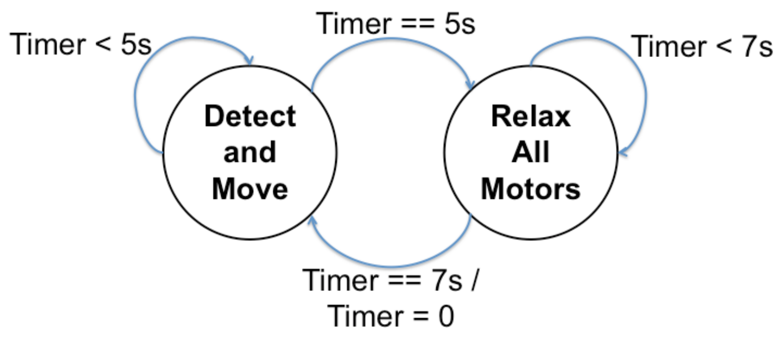


Figure 6.5: Time based state machine which automates the hand-tuned stepwise controller. Timers are used to allow for dynamic settling before the next action is taken.

transition between each state is time based, where the timing between states was empirically obtained to ensure transition and dynamic settling. It should be noted that for SUPERball, a single face transition requires more than the single motor command as stated in section 6.1. This non ideal behavior is caused by many factors, and these factors mainly affect the maximum tension a single cable should experience during actuation. Limiting actuation to a value less than the required to move the CoM outside the bottom triangle.

6.3 Machine Learning Enabled Controllers

The previous sections in this chapter dealt with locomotion controllers that are idealistically simple as in section 6.1 or completely derived by human experimentation as in section 6.2. Thus, these controllers do not leverage any inherent properties within the tenesgrity structure or have the ability to optimize around any limitations in the structure's design. One property that will be leveraged in the following sections is global force distribution. Figure 6.6 shows how the tensioning on a single cable affects all other cables on SUPERball.

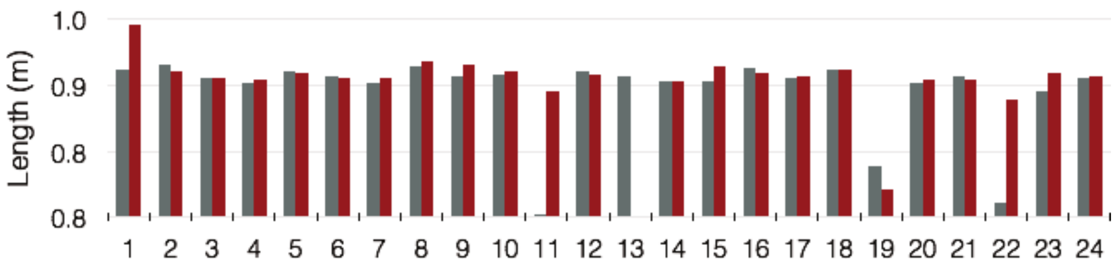


Figure 6.6: Change in length of cables when one (13th) is pulled to 0.5 meters while the others are kept at the same rest length. Grey bars show original length, red show final length. While the robot is at the exact same orientation, the actual lengths of the cables change in a non-linear way. Some of the cables shorten due to the tension introduced by cable 13, and some of the cables relax.

The majority of the learned controllers discussed in this section will be implemented in the NTRT simulation environment with the final locomotion controller implemented on the SUPERball hardware. The first section will discuss a two stage Monte Carlo method for maneuvering SUPERball out of craters/holes. This if followed by an open loops learned controller for forward navigation. Finally, the last section discusses a closed loop locomotion controller developed in simulation and implemented on the physical SUPERball hardware.

6.3.1 Crater Escape Controller

This section explains a locomotion controller that actuates a simulated SUPERball like structure out of a hole/crater. In order to achieve a solution to this task, a two stage Monte Carlo technique is used on a simplified control space on a simulated fully actuated SUPERball in NTRT. The simplified control space consists of clustering each equilateral triangle, shown in section 6.1, such that all three actuators in that cluster follow a sine wave dictated by equation 6.1

$$y = Asin(\omega t + \varphi) + D \tag{6.1}$$

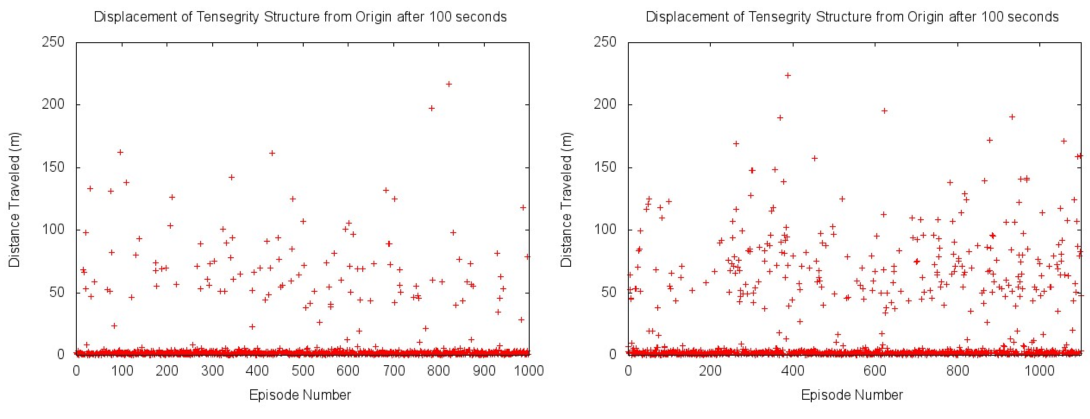
Where A is amplitude, ω is angular frequency, φ is phase change, and D is the DC offset. This makes 32 parameters for a policy to learn and search over (4 parameters $\times$ 8 clusters). A successful policy is one where the robot moves a linear distance greater than the maximum radius of the hole/crater it starts in.

There are two stages to this process, where the first stage run 1000 samples with evenly distributed random parameters for each samples (generation 0). The second stage (generation 1) filters out the successful samples from the first stage and runs 10 evenly distributed samples around each of the previously successful parameters such that each value is no more than $\pm 0.05\%$ from the initial value. Each sample is let run for a simulation time of 60 s. This approach makes it feasible to run this simulation process in a relatively short time frame (less than 40 minutes on a 2014 or later quad core i7 or equivalent processor), enabling the system to learn a new policy based on a real time estimate of its current state. It should be noted that this is a simplified open loop control policy of the one used in the open loop locomotion controller found in section 6.3.2.

Experiment and Results

Following the escape criterion set in the previous section, the robot escapes the hole/creater when a linear distance of 25 meter is reached in the simulation environment.

Figure 6.7a shows the simulated robot's CoM displacement for each of the 1000 samples in generation 0 of the learning process. This generation had 110 samples which reached a distance of 25 meters or greater. For generation 1, 1100 samples



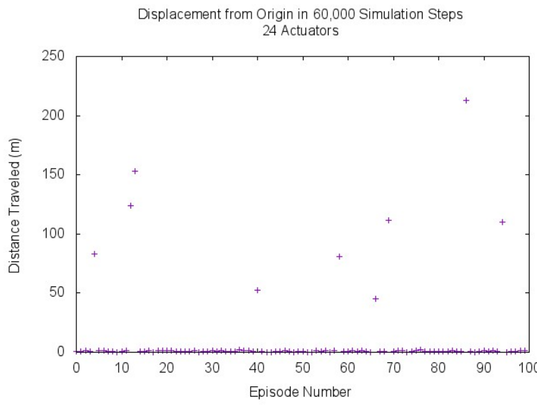
(a) **(b)**

Figure 6.7: (a) Generations 0 for the escape hole/crater policy. The displacements reached in 1000 independent samples using Monte Carlo generated control policies. (b) Generations 1 for the escape hole/crater policy. The displacements reached in 1100 independent samples using control policies dictated by the successful samples from Generation 0. The values in the set of sine wave parameters from Generation 1 samples were each centered around one of the successful corresponding sine wave parameters in Generation 0 control policies. These Generation 1 values were then modified to be within 0.5% of their respective Generation 0 values with the goal of optimizing Generation 0 control policies.

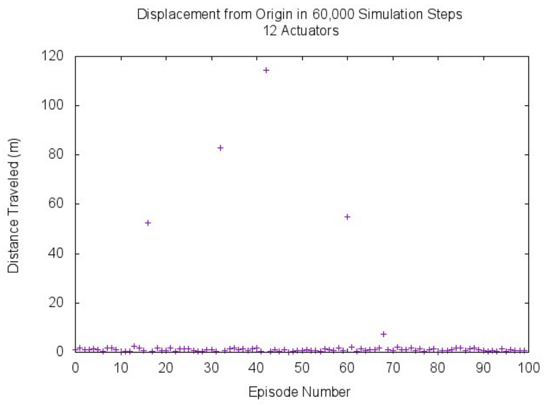
were taken, and its results are shown in figure 6.7b.

To further explore this policy, a simple comparison is conducted to evaluate how well the policy controls the robot out of a hole/crater with only 12 usable actuators. As a control, 100 samples are taken with all actuators functioning, shown in figure 6.8a. Then another set of 100 samples are taken where only 12 of the actuators are responding to the sine wave command, shown in figure 6.8b. The other actuators are modeled as passive linear springs.

The result of this policy only showed limited success with the second generation only achieving 24% success. However, the policy is still able to achieve success even when half of the actuators are turned off. These preliminary results demonstrate that it is possible to actuate a SUPERball like system out of hole/craters it may be stuck within.



(a)



(b)

Figure 6.8: (a) 100 Samples of Tensegrity Structures with 24 Actuators. By reducing the number of functioning actuators on our simulated tensegrity, the limitations of tensegrity escape can be better explored. With all 24 actuators on the tensegrity functioning correctly, a successful escape is relatively easy (9% success rate). (b) 100 Samples of Tensegrity Structures with 12 Actuators. With just half of the original actuators functioning correctly, a successful escape is even more difficult (see Figure 7). In this generation, 4% of tensegrity rovers were still able to escape the ditch in the allotted time.

6.3.2 Co-Evolutionary Learning for an Open Loop Controller

This section explores an evolutionary controller on a fully actuated (24 actuators) SUPERball like robot, and all the results are based on a simulated robot in NTRT. This work shows how a controller learned through machine learning can utilize the dynamics of such a structure to find optimally consistent locomotion gaits for goal directed behavior. The work presented in this section are summarized sections of collaborative work found in Iscen et al. 's work [4].

The overall goal of this controller is to have the tensegrity robot roll smoothly within the limitations of the simulated actuation and hardware parameters. A periodic open loop controller is used with parameters that are set by an evolutionary algorithm [116]. During rolling locomotion, the controllers will repeat the

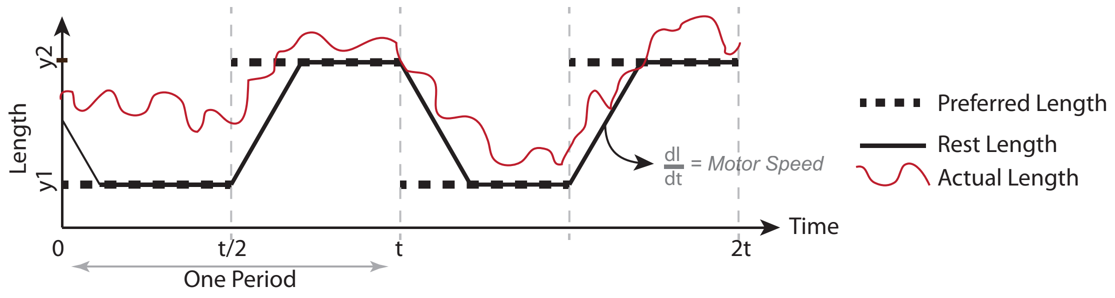


Figure 6.9: An example signal with 2 sub-intervals with preferred lengths of y1 and y2 and periodicity t.

same actuation motion. Considering that the rolling locomotion is a repetitive behavior, the signals produced by the controllers will be periodic. The key to making this system work is determining the shape of this periodic signal.

For this work, a signal of periodicity t has a function $F(t)$ where the function is a discrete number of step functions n. Thus, a simplistic example where $n = 2$ would have the form

$$F(t') = \begin{cases} y1 & \text{for } t' \in [0, t_1] \\ y2 & \text{for } t' \in (t_1, t] \end{cases} \tag{6.2}$$

where $t_1 < t$ and $y1$ and $y2$ are motor position values and this simple case can be seen in figure 6.9.

When the function is expanded to $n = k$ it takes the form of

$$F(t') = \begin{cases} y1 & \text{for } t' \in [0, t_1] \\ y2 & \text{for } t' \in (t_1, t_2] \\ \vdots & \vdots \\ yk & \text{for } t' \in (t_{k-1}, t] \end{cases} \tag{6.3}$$

and it can easily be shown that as $n \rightarrow \infty$ any arbitrary signal may be generated. To generate a signal, the only parameters needed are number of sub-intervals and rest length values for each sub interval. For the specific example given in Figure 6.9, the number of subintervals is 2 and y1 and y2 are the values of preferred rest lengths for those intervals.

Algorithm and Learning Method

The problem is episodic, the agents have 60 seconds to test their policies. At the end of each episode these candidates are evaluated according to their performance. Performance is then measured as the distance covered in 60 seconds. Formally, the evaluation is defined as

$$f = d(y_{0,0}, y_{0,1}, ..., y_{0,n}, y_{1,0}, ..., y_{24,n}) \ , \tag{6.4}$$

where, $y_{i,j}$ is the rest length for the ith controller and jth subinterval. Depending on the complexity of the signals (n) selected, there are $24 * n$ parameters to learn. In order to learn these parameters, this method uses a historical average co-evolutionary algorithm. In historical average, each member receives its fitness according to the average of their performances. If a member survives for the next generation (is not eliminated or mutated) the member keeps its previous experiences. At each generation, the fitness assignment is the average of this growing history of past evaluations. The algorithm used can be found in Algorithm 1.

Algorithm 1: Cooperative coevolutionary Algorithm with Historical Average

```
Data: Population of $n$ elements for each agent
for $i$=1..$k$ do
    $random_{team} \leftarrow \emptyset$ ;
    forall the Populations do
        $random_{team} \leftarrow random_{agent}$;
    end
    score = evaluate($random_{team}$) ;
    forall the $agents \in random_{team}$ do
        agent.history ← score ;
    end
end
forall the Populations do
    forall the agents do
        agent.fitness = average(agent.history) ;
    end
    order population according to the fitness;
    eliminate the last $z$ members;
    copy the first $z$ to the last $z$;
    mutate the last $z$;
    clear history for the last $z$;
end
```

Learning Results

An example learning session is shown using signals with complexity (n) of 5 and period (t) of 4 seconds. Figure 6.10 illustrates the distance rolled by the robots over the course of learning. Starting with 0 meters, the robots converge to rolling over 32 meters in 60 seconds. This result shows that successful learning of rolling locomotion using this method is possible. The second line at the same Figure (Figure 6.10) shows the rate of unfeasible policies that are tried while learning to

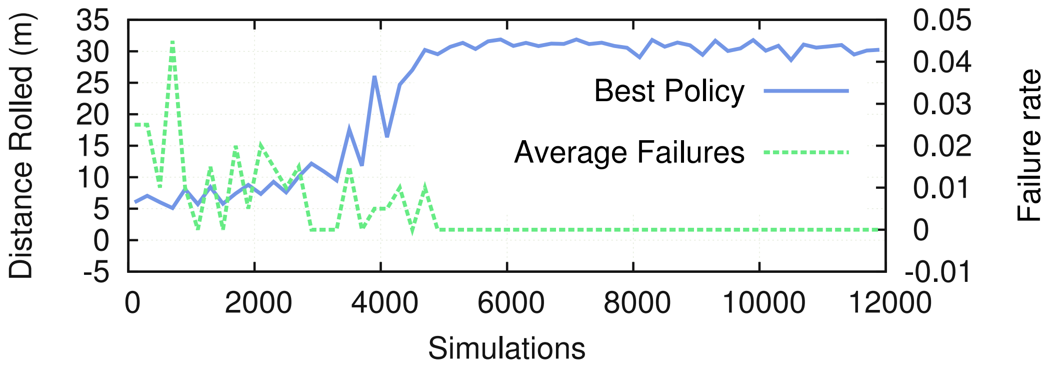


Figure 6.10: The performance of the robots during the learning session for signals of complexity 5 and period of 4 seconds. As a side result, the percentage of the policies that were failed to stay in reasonable limits are shown with the second line.

roll. While converging to rolling locomotion, unfeasible policies drop to 0. This shows that the learned policy lies within simulated parameters set by the user.

6.3.3 Mirror Descent Guided Policy Search

Guided Policy Search utilizes supervised policy learning to leverage a series of non-generalized optimized local polices to learning a generalizable global policy [92]. These non-generalized optimized local policies, $p_i(\mathbf{u}_t|\mathbf{x}_t)$, only successfully work from specific initial states and require full state information. Guided Policy Search allows for the use of simple and efficient methods for training the local policies, such as trajectory optimization methods when there is a known model, or trajectory-centric reinforcement learning methods [93].

In this work, a modified version of Guided Policy Search is used based on mirror descent [89], called Mirror Descent Guided Policy Search (MDGPS). This version optimizes the global policy by sampling the current iteration's local polices and approximates the minimum divergence between the global policy and the local policies. To optimize the local polices, $J(\theta)$ is minimized such that there

Algorithm 2: Mirror descent guided policy search (MDGPS)

```
1: for iteration $k = 1$ to $K$ do
2:     Run either each $p_i$ or $\pi_\theta$ to generate samples $\{\tau\}$
3:     Set $p_i \leftarrow \arg\min_{\hat{p}_i} \mathbb{E}_{\hat{p}_i}[\ell(\tau)]$ $s.t.$ $D_{KL}(\hat{p}_i \| \bar{\pi}_{\theta i}) \leq \epsilon$
4:     Train $\pi_\theta$ using supervised learning on $\{\tau\}$
5: end for
```

is a bound on the Kullback-Leibler divergence (KL-divergence) between the local policy and the linearized global policy $\bar{\pi}_{\theta i}$ [117, 118, 119, 87]. For clarification, the KL-divergence is a measure of how much information is lost when using a probability distribution to approximate another distribution. A generic MDGPS algorithm is shown in Algorithm 2.

Policy learning machine learning algorithms, commonly called policy search algorithms, are used to directly search the policy parameter space and are alternatives to value function based reinforcement learning algorithms [120]. This algorithm tries to find a set of policy parameters θ which optimizes the policy $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ with respect to the expected cost. With a finite set of episodes, the expected cost under the policy is given by $J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\pi_\theta}[\ell(\mathbf{x}_t, \mathbf{u}_t)]$, where $\ell(\mathbf{x}_t, \mathbf{u}_t)$ is the cost function. $\mathbf{x}_t$ is the state of the system at time t, $\mathbf{o}_t$ is the observation of the state at time t, and $\mathbf{u}_t$ is the action at time t.

Optimizing Periodic Gaits with MDGPS

Using the MDGPS method stated in section 6.3.3, a periodic rolling gait for SUPERball was learned using single transitions between faces mentioned in section 6.1 as the local policies and how to transition between them using limited sensor data as the global policy. In order to obtain this stable periodic rolling gait, the task is split across several policies, each optimized over a small time segment. After establishing a desired behavior across the states seen by the policies, a global policy is learned that can generalized the behavior of the local polices based on the Guided Policy Search framework.

GPS algorithms such as MDGPS use supervised learning to learn a global policy, where the supervision comes from several local policies $p_i(\mathbf{u}_t|\mathbf{x}_t)$, $i \in \{1, \ldots, C\}$. Each local policy is trained from a different initial state, where C is the chosen number of initial states. Each local policy is optimized over T^p time steps, and we wish to learn a global policy $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ that can succeed by generalizing the behavior of these local policies over an episode of length T^π.

In locomotion tasks, we ideally want the global policy to exhibit continuous successful behavior, i.e., $T^\pi = \infty$, and we can empirically determine T^p based on the amount of supervision the global policy needs to learn a continuous periodic gait. For SUPERball, T^p is initialized to a short horizon, and continually increased until the global policy learns a successful locomotion gait.

If the required T^p is long, as is the case for the SUPERball locomotion task, it is difficult to optimize a local policy over this time horizon due to the accumulation of uncertainty and errors. However, L local policies $p_i^1, \ldots, p_i^L$ can be learned for each initial state i, each optimized for T^p/L time steps. For the local policies p_i^j, $j \in \{2, \ldots, L\}$, we set the initial state $\mathbf{x}_0^j$ to be the final state of the preceding local policy, i.e. $\mathbf{x}_{T^p/L}^{j-1}$. This amounts to training local policies in a sequential fashion, where the L local policies together are optimized over T^p time steps. The algorithmic details using in learning a periodic stable gait for SUPERball can be seen in Algorithm 3. Note that on line 7, samples can be collected from either the local polices or the global policy. Initially, these samples are taken from the local polices, but are switched to the global policy based on a user's expert knowledge for how the local polices are performing.

Algorithm 3: MDGPS with sequential local policies

```
1: for iteration $k = 1$ to $K$ do
2:     for $i = 1$ to $C$ do
3:         $S_i \leftarrow \{\}$
4:         for the desired number of samples do
5:             $x_0 \leftarrow$ initial state $i$
6:             for $l = 1$ to $L$ do
7:                 Run either $p_i^l$ or $\pi_\theta$ to generate sample $\tau$
8:                 $S_i$, $x_0 \leftarrow S_i \cup \{\tau\}$, end state of $\tau$
9:             end for
10:        end for
11:        for $l = 1$ to $L$ do
12:            $p_i^l \leftarrow \arg\min_{\hat{p}_i^l} \mathbb{E}_{\hat{p}_i^l}[\ell(\tau)]$ $s.t.$ $D_{KL}(\hat{p}_i^l \| \bar{\pi}_{\theta i}) \leq \epsilon$
13:        end for
14:    end for
15:    Train $\pi_\theta$ using supervised learning on $\bigcup_i S_i$
16: end for
```

Kinematic Constraints for Safe Actions

A challenge for machine learning techniques is that policy requirements are usually encoded into the techniques' unique cost function. This function not only needs to guide task level objectives, such as movement, but the cost function must also guide hard constraints like safety without any other external limits. Due to the fact that SUPERball only has position control, as outlined in section A.1.2, a random sampling of motor position might cause the system to tension a cable beyond it's mechanical limit breaking the cable, the motor, or both. These configurations of tension limits are difficult to to embed analytically into the cost function and even harder to optimally balance with task level objectives. Thus, a simple global motor position safety constraint method was implemented that interfaces outside of the machine learning framework.

Specifically, the cable tensions are estimated for a particular set of actuator positions using a simple forward kinematic model of SUPERball. This model is based on the model outlined in section 5.2.1 with the simplification of no gravity or ground contact. The limits are preset as a maximum tension value exerted on any cable based on a user defined maximum value. For SUPERball, this maximum value was experimentally found to be 250 N. Computing the cable tensions for a given set of motor positions takes a few milliseconds using forward kinematics. As this is easily parallelized, a database was constructed containing about 100 million motor positions deemed safe in a few hours. Then, when the policy outputs an action, an efficient look up method called Fast Library for Approximate Nearest Neighbors (FLANN) [121] is used to compute and command the nearest (ℓ_1 norm) safe action. This ensures that even if the exact action isn't located in the table, an approximate action set is chosen. At runtime, finding the nearest neighbor action takes roughly 200 µs and is easily embedded into both training and testing without disrupting the command frequency of 10 Hz.

Generalization Across Domains

Local policies are trained with full state information $\mathbf{x}_t$, but the global policy is learned such that only observations of the state $\mathbf{o}_t$ are used as inputs. This separation between the local policies and global policy reflects prior work on tasks involving partial observability, where the intuition is that the local policies are trained in a controlled environment but the global policy must be able to adapt to a more general setting [122]. For SUPERball, the full state $\mathbf{x}_t$ can only be obtained through simulation or the use of an external state estimator system as described in section 5. In contrast, an observation $\mathbf{o}_t$ is used that can be calculated directly from the sensors on the robot. This can greatly simplify the transfer from simulation to the real robot, as the learned policy is less prone to overfit to the simulation and takes actions directly based on the sensor measurements from the physical robot. Furthermore, because the goal of SUPERball and many other robots is deployment to unfamiliar, remote environments, the choice of an observation that relies only on the robot's onboard sensors is very important, as it is unrealistic to expect the level of information and reliability that an external state estimator can provide.

Because real-world sensors and actuators are noisy and imperfect, noise is introduced on the input to the policy during training. Gaussian noise of mean 0 and variance equal to 10% of the observation range is added to all sensors. Since SUPERball uses a WiFi network, network drop out is modeled by randomly selecting 10% of all sensor measurements as dropped measurements. When the current observation is dropped, the previous observation is used as the input to the policy. Adding noise improves the generalization capabilities of the learned policy across conditions such as terrain, gravity, and motor failure. These conditions are evaluated in Section 6.3.3.

Experimental Setup

The state of the system $\mathbf{x}_t$ is set to be the position and velocity of each of the 12 bar endpoints of SUPERball, and the position and velocity of each of the 12 motors, measured in radians, for a total dimensionality of 96. There are two different representations for the observation $\mathbf{o}_t$, "full" and "limited". The "full" 36-dimensional observation includes motor positions, and also uses elevation and rotation angles calculated from the accelerometer and magnetometer sensors on the robot. The "limited" observation is 12-dimensional and only uses the acceleration measurement along the bar axis from each of the accelerometers. It was found that interfering magnetic fields near the testing grounds at NASA Ames cause the magnetometers to be unreliable and difficult to calibrate. Therefore, the policy using the limited observation is much easier to transfer on to the real robot.

Table 6.1: Average distances in meters traveled using the policies learned with varying observation representations and local policy training schemes.

		Our Method			Open-Loop	
		Full Observation, Six Initial States	Limited Observation, Six Initial States	Limited Observation, One Initial State	Mean Actions from Best Learned Policy	Hand-Engineered Punctuated Rolling
Normal Conditions		$\mathbf{25.307 \pm 0.309}$	24.141 ± 0.352	20.008 ± 0.871	*25.076 ± 0.078*	10.266 ± 0.071
Terrain	Rocky	*6.025 ± 2.835*	$\mathbf{9.568 \pm 5.197}$	3.124 ± 1.083	3.069 ± 2.201	1.734 ± 0.411
	Uphill	$\mathbf{18.547 \pm 0.231}$	*16.107 ± 0.809*	13.573 ± 0.174	7.721 ± 0.236	8.136 ± 0.026
	Downhill	$\mathbf{32.896 \pm 0.275}$	*29.970 ± 0.858*	21.963 ± 2.403	27.661 ± 0.136	11.264 ± 0.091
Gravity	10%	$\mathbf{19.505 \pm 0.746}$	16.966 ± 0.362	13.927 ± 0.516	*18.024 ± 2.356*	11.044 ± 0.054
	50%	$\mathbf{23.331 \pm 0.871}$	*21.220 ± 0.202*	17.766 ± 0.490	19.673 ± 3.244	10.310 ± 0.010
	200%	$\mathbf{27.600 \pm 2.307}$	*26.715 ± 0.566*	21.680 ± 1.330	24.865 ± 0.190	9.845 ± 0.009
Robot	Heavy	12.521 ± 1.710	$\mathbf{14.561 \pm 0.079}$	*12.972 ± 0.110*	1.081 ± 0.019	10.550 ± 0.003
	End Cap Failure	*21.890*	**22.100**		10.291	10.247
Added Noise	0%	**26.494**	*25.828*		N/A	N/A
	20%	*7.725*	**19.212**			

The open-loop mean actions from the learned policy that performs best under training conditions, and the hand-engineered open-loop policy. Results are averaged across five trials of one minute each for a variety of terrain, gravity, noise, and robot settings. "Normal Conditions" are the training conditions, which are flat terrain, 100% gravity, 10% added noise to the input, and normal robot parameters. When varying one setting, all other settings remain the same as during training time. The open-loop controllers are not shown with varying input noise, because these controllers do not have any input. Bolded numbers indicate the farthest distance traveled for any given condition, and italicized numbers are the second farthest. Note that the first two learned policies generally outperform all other controllers, demonstrating the benefits of this method and using multiple initial states in learning efficient and generalizable locomotion.

The action $\mathbf{u}_t$ is the instantaneous desired position of each motor.

For the rolling task, each local policy reliably learned in about 200 samples. Simultaneously during training of the local policies, a global policy is learned, which for this work is a deep neural network. The deep neural network has three hidden layers of 64 rectified linear units (ReLU) each, using the same samples. The cost function $l(\mathbf{x}_t, \mathbf{u}_t)$ is simply the negative average velocity of the bar endpoints of the robot. This trains the policy to favor faster rolling behavior.

Results in Simulation

To show that this method of training sequential local policies is effective, the results of training two sequential local policies for 5 s each against training one local policy for the full 10 s were compared, both using the trajectory-centric reinforcement learning method detailed in [93]. The average distance traveled over five trials for the two 5 s local polices and the single 10 s policy were 3.15 m and 1.23 m, respectively. These results demonstrate that, by training sequences of local policies over shorter horizons, more efficient locomotion can be achieved with fewer samples by decreasing the accumulation of error over time.

To demonstrate the benefit of multiple initial states, a global policy was learned by using one long sequence of six local policies, trained over 5 s each, starting from only one initial state. These six polices encapsulated a full rotation of the robot. Due to the build-up in variance in the starting states of the local polices and the divergence in the behavior of the local policies from the desired periodic gait, training a rolling gait was not achievable. Results for testing the learned policies against a range of environmental and robot parameters are presented in Table 6.1.

In summary, these results show that all learned policies substantially outperform the hand-engineered rolling controller, and the closed-loop neural network policies outperform the open-loop baselines in almost all conditions, indicating the benefits of both learning and feedback in SUPERball locomotion. This method is able to learn successful and efficient policies even with the limited sensory observations provided by only SUPERball's accelerometers, and the learned policies demonstrate generalization to unseen conditions representative of what a planetary exploration rover might encounter, such as changing terrains, unstable levels of noise, and hardware failure. The addition of input noise during training encourages this generalization, and results in learned policies with similar levels of reliability as the hand-engineered controller, though significantly faster and less likely to cause hardware failure on the real robot.

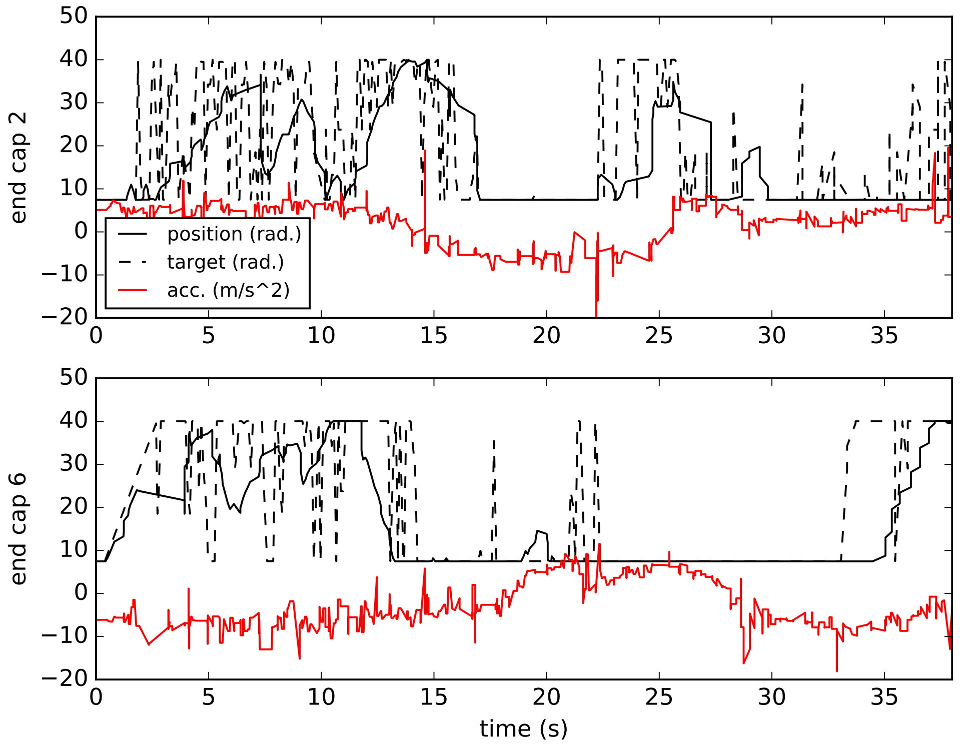


Figure 6.11: This plot shows actual motor positions, target motor positions, and single axis accelerometer data over the first 40 seconds of a trial for two rod ends which are not connected via cables and not attached to the same rod. The commanded positions change based on the accelerometer feedback, showing the controller working as the robot changes orientation by rolling. The actual motor position lags behind the target motor position due to motor dynamics and network UDP packet loss.

Results in the Real World

We compared the learned policy with limited observation, trained in simulation, against an open-loop policy that outputs the mean actions from this learned policy under training conditions. Both policies were run on the physical SUPERball robot on flat terrain. Over three trials of 100 seconds each, using the learned policy, SUPERball rolled approximately 12 m, 9 m, and 8 m. 12 m is about the maximum distance allowed during the trials, as the robot rolled out side the limited network range and could not roll any further. Also, a cable malfunction cut the last trial short by about 20 seconds, which was on track to reach the 12 m limit. Despite these issues and the differences between the simulated and physical robot, the policy was able to successfully produce a gait on SUPERball that is more reliable, and less risky for the hardware, than any previous locomotion controller. The learned policy is able to adapt to the physical SUPERball robot by using feedback from the accelerometers, as seen in figure 6.11.

The open-loop policy was not able to produce any reasonable behavior on the real robot, and we ran it only once due to concerns about hardware safety.

Chapter 7

Conclusion and Future Work

7.1 Contribution

The contributions of this dissertation to the field of tensegrity robotics are effective state estimation from a large number of noisy senors in near real time, algorithms and methods that will control underactuated tensegrity robotic systems with limited sensor data, and implemented a control policy learned entirely through simulation with no prior knowledge of system dynamics on physical hardware. Another smaller contribution to tensegrity robotics was the creation of the world's first unteathered underactuated tensegrity robot capable of complex controls. Artificial neural networks (ANN) where the main control methodology used due to the inherent nonlinear system dynamics, since ANN are data driven controllers which do not require dynamic models. Several learning methods were used to train a ANN to accomplish locomotion as well as escaping from simulated craters. In completing these tasks, contributions were also made to the fields of machine learning and sensor fusion.

7.2 Future Work

7.2.1 Re-Design of SUPERball

SUPERball was the initial prototype for this project. It succeeded in performing and showcasing a tensegrity robot built for locomotion, though it does have its limitations. An evaluation of SUPERball and its limitations may be found in appendix A.

To overcome these limitations and to further expand the types of control polices for a SUPERball like robot, a re-design is proposed which has full cable actuation and will be more robust to large environmental impacts. The largest limitation for the current design of SUPERball is the limited low level motor control, force sensing, and actuation stroke. With the advent of robotic companies producing commercial off-the-shelf (COTS) motor solutions, using a COTS motor with integrated force sensing and control would be a viable option. Table 7.1 show a limited list of companies which offer products that could be consider for a SUPERball re-design.

Table 7.1: List of companies that sell COTS actuators to be integrated into SUPERball v2. This is not an exhaustive list.

Company	Dynamixel	Kinova	Muse Robotics	Hebi Robotics
Product	MX-106	K-58	COTS motor control boards	X8-3

From table 7.1, a likely candidate to succeed is the X8-3 from Hebi Robotics. This COTS actuator has position, velocity, and output torque control which utilizes a series elastic element to directly sense the output torque. It also supports tunable control parameters and closed loops control parameters, which all run at 1000 Hz and can be tuned while the system is operating. Thus, an initial prototype has been designed around this X8-3 motor, where single rod and end cap

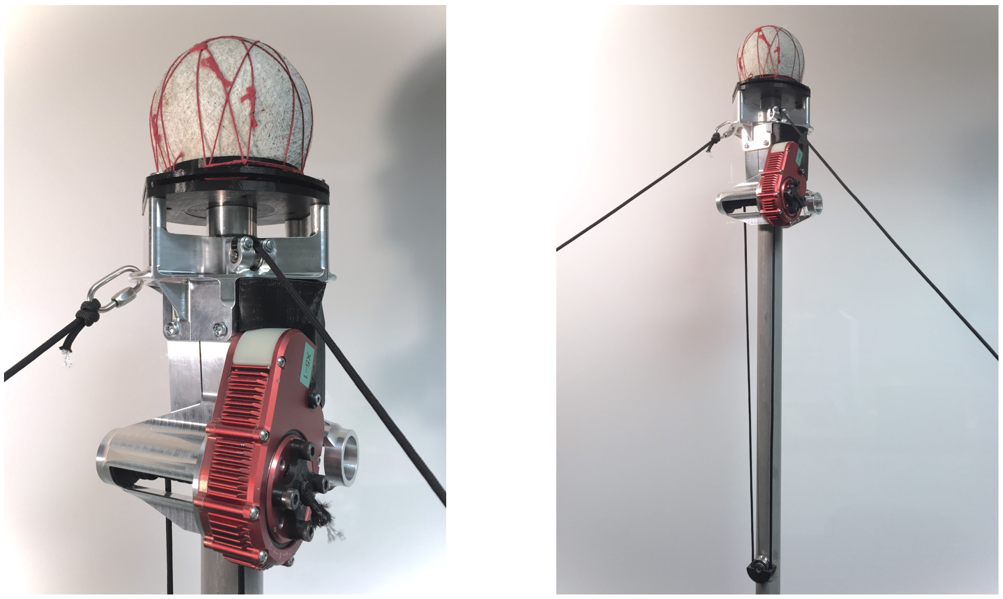

(a) (b)

Figure 7.1: Images of initial SUPERball v2 MTR designs. (a) Close up of initial design for SUPERball v2. An omni directional cable guide can be seen which eliminates cable friction. (b) Full cable routing where the cable is run over a distal pulley to increase cable length for a better effective spring constant induced by the cable stretch.

testing still needs to be done. This prototype eliminates the internal springs and relies on cable stretch and motor elastic element for system passive compliance. Figure 7.1 shows what the new design concept looks like.

The redesign of SUPERball is still in very early stages of development, however table 7.2 shows some of the differences in base physical properties and features between the two versions.

7.2.2 Extending Controls for a Fully Actuated SUPERball v2

In chapter 6, there was a focus on learning a forward locomotion gait for a hardware system that had very limited actuation. SUPERball v2 aims to address

Table 7.2: Comparison between the current SUPERball v1 and the intended design of SUPERball v2

	Number of Actuators	Size / Weight	Motor Controls	Sensors	Maximum Length for Change in Cable Length*
SUPERball v1	12	1.75 m/21 kg	Position	Motor Encoders Accelerometer, Gyro, Magnetometer Ranging / localization Sensors	0.4 m
SUPERball v2	24	1.8 m/30 kg	Position Velocity Torque	Motor Encoders Motor Torque Sensor Accelerometer, Gyro Ranging / localization Sensors Linear Cable Sensors	1.5 m

* This is the total length a single motor can pull starting with no cable wound on the spool. For SUPERball v2 this means the start length of cable will mean the robot will be in a collapsed state initially.

these limitations and expand the robustness of the system, which will open up new areas of research. This system will allow for forward locomotion to be achieved through the "basic" gait as outlined in section 6.1. Using this control method as well as the basic steering from section 6.1.2 as priors for the learning method outlined in section 6.3.3, highly optimized gaits for navigation should be rapidly achieved.

Decentralized Control

One limitation with the controllers developed so far for SUPERball is their reliance on centralized controllers for basic locomotion. A computer connects to SUPERball's network in order to receive and send data over a standard 802.11 [123] wireless protocol using the ROS UDP messaging protocol. This allows for easy communication between the robot and the control computer, however large amounts of data can be lost from this network structure. High data loss dramatically affects the performance of a controller and may even cause the controller to fail. Utilizing a decentralized control method will allow the individual actuators to adapt without this wireless network layer. Mirletz et al. developed a control scheme which uses a Central Pattern Generator on each actuator to achieve decentralized locomotion for a snake like tensgerity structure [102]. Taking this approach and replacing the learning method with the guided policy search method shown in section 6.3.3, a controller could be learned in a fraction of the time that may require no hand tuning to implement on hardware. This will enable SUPERball to learn gaits optimized for various terrains which requires no external centralized control input.

Path Planning

Having basic locomotion controls achieved, high level navigation and path planning techniques can be researched and demonstrated. The path planning method discussed in section 2.6.2, currently relies on random actuator commands to actuate the simulated SUPERball like robot along a given path. The author directly states that faster and more computationally efficient paths will be generated if the algorithm can assume some level of confidence when selecting a control input [94]. Using a method like the guided policy search method mentioned above, a list of control inputs for various scenarios may be compiled. It may also be possible to integrate the GPS algorithm into the path planning method, enabling the creation of efficient gaits when new environments are encountered.

7.3 Conclusion

This works hopes to inform and further future research into tensegrity and highly compliant cable driven robotics, and their applications in terrestrial exploration. The future of terrestrial robotics will utilize passive compliance to adapt and perform in scenarios such as rocky and icy terrain, surviving high falls, and navigating through extreme conditions where it will be difficult for non-passively compliant robots to perform.

Appendix A

SUPERball v1 Design Evaluation

Once SUPERball v1 was built and evaluated, the system performed mostly as designed. Since this was an initial venture into building an untethered tensegrity robot, there were several design choices that ultimately impacted the system's performance negatively. In the following sections, I will try to summarize and evaluate how the mechanical, electrical, and communication subsystems performed. I will then try to make suggestions to improve upon these systems.

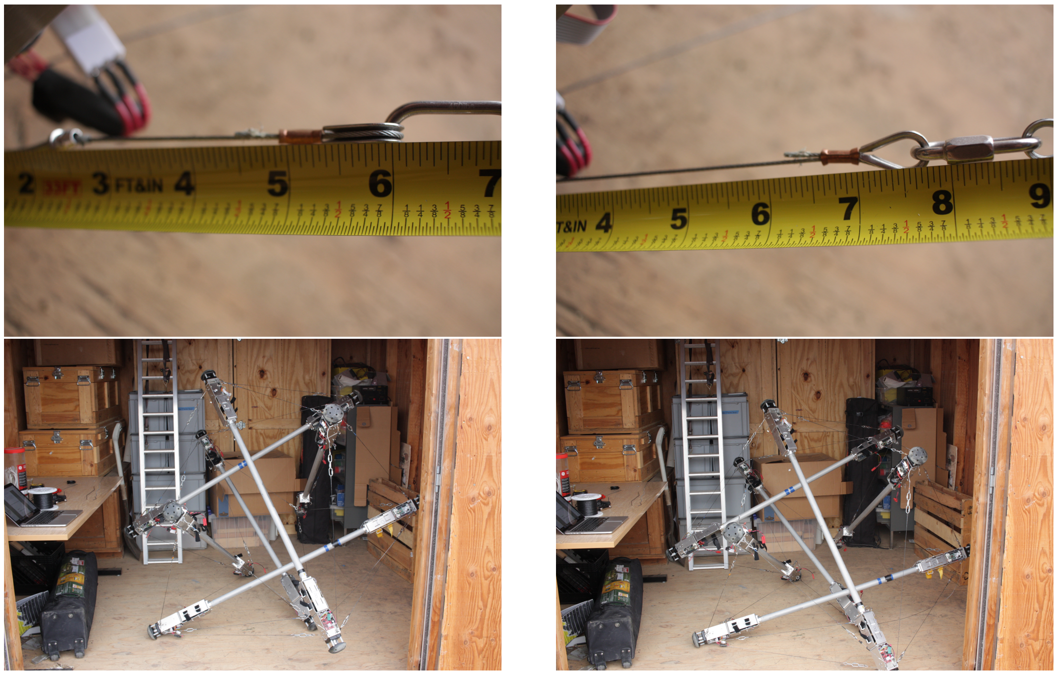
A.1 Mechanical Evaluation

In this section, an evaluation of the SUPERball will be taken based on the three main subsystems outlined above.

A.1.1 Cable Routing

Upon testing and using SUPERball, the limitations on the mechanical design became the largest hindrance to achieving consistent performance. A large majority of the mechanical limitations stemmed from the bowden cable routing, management, and material choices. The design of the MTR was focused around using a bowden system to route cables around components and features within the housing. This allowed for the cable routing to be secondary to the placement of components within the MTR, making the overall design easier, but increasing the number of bends in the bowden cable housing. As explained in section 4.1.4, the internal cable material used was braided steel cable. The steel cable allowed for easy assembly and good wear resistance, however it has a minimum bend radius to keep the cable from plastically deforming. Our initial design took into account this limitation with a correctly sized roller guide (see figure 4.3), though in practice the combination of the induced tension and the wrong type of groove in the roller guide leads to a slight plastic deformation of the steel cable in the form of kinks. Extra friction is then imparted into the bowden system as theses kinks try to slide inside the bowden housing. Friction between the cable and housing element was a known factor, however the amount of friction it induced is much larger than expected. Figure A.1 shows a qualitative example of hysteresis in cable length due to the effect friction has on the system.

Another limitation imposed by the cable routing system was the lack of design effort that went into the cable exit system. Originally, it was designed to be a section of the nylon bowden cable housing sticking out of the robot to isolate the steel cable as it exited the MTR aluminum housing. However, the angle in which the cable exits the system was never considered in the original SUPERball design. This failure meant that an exit angle of 90 degrees perpendicular to the rod was used for the cable exit. Comparing this to the actual cable exit angle of approximately 30 degrees from perpendicular during normal operation meant that more than 60% of the tension force is imparted into the nylon cable and the aluminum housing exit hole. When the cable would slide in and out of the exit point, the nylon tube quickly would sheer apart. This then caused the steel



(a) (b)

Figure A.1: These figures show the hystresis effect due to friction on the SUPERball. SUPERball in the (a) figures has been lifted off the ground and gently placed back down so that all the springs in the system are allowed to reach equilibrium due to gravity. No extra force has been placed on the robot other than gravity. SUPERball in the (b) figures was arbitrarily pushed downwards on two of the top endcaps with enough force to deform the robot. It can easily been seen the that the internal friction does not allow for the robot to return to it's original state.

cable to rub on the aluminum housing, which then would slowly saw into the housing. Since there was not budget to redesign the whole routing system, a drop in replacement fix was required. The solution was to use a "break noodle" steel cable router typically used in bicycle braking systems. This prevented the steel cable from cutting into to MTR housing, but it causes a bit more friction to get imparted into the whole cable system. Figure A.2 shows the cable exit point on SUPERball with the "break noodle" fix.

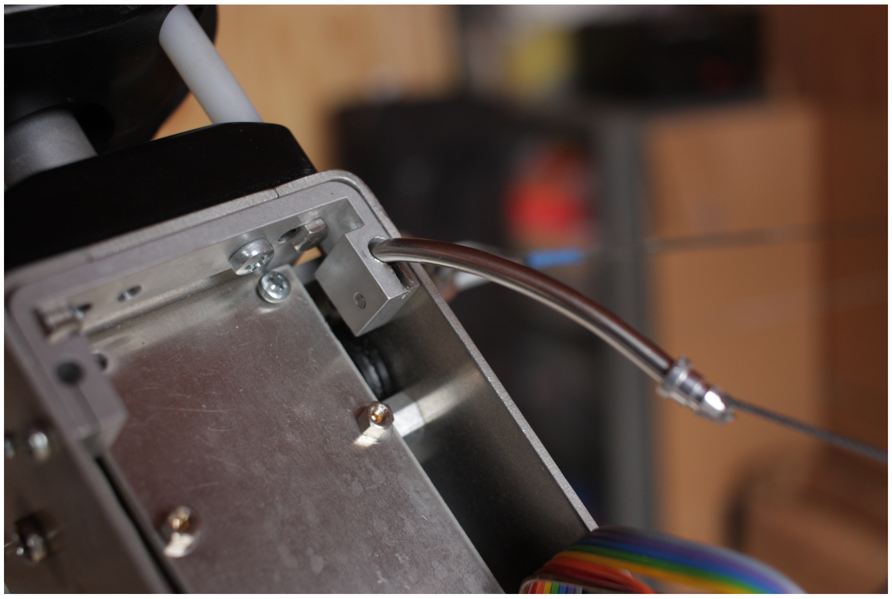


Figure A.2: This figure shows the original exit point for the steel cable on SUPERball with the "break noodle" fix. The point of entry of the "break noodle" is can be seen where the aluminum of the bracket has been cut by the steel cable before the fix was applied.

A.1.2 Cable Tension

Another limitation on the system's performance is the larger than designed tensions seen on the individual cables. This caused a limit on the maximum length each actuator imposes on a cable. Max cable tension on SUPERball was designed around a maximum continuous operating tension force of $200N$ per cable. The system was also designed to take intermittent forces $50 - 75N$ higher than this maximum that would be caused by rolling. These values were obtained through evaluating the tension forces derived from Iscen et al's control work within the NASA Tensegrity Robotics Toolkit [103]. Iscen used a simulated robot based on the initial design of SUPERball, which consisted of a total mass of $18kg$ and each rod being $1.5m$ in length. Comparing these values to those in table 3.1, the final SUPERball parameters were different than the ones used to determine the operating tensions. This was due to Iscen's work was published quite early in the design process and not all the mechanical design aspects of the MTRs were finalized. This discrepancy in weight caused the actual nominal operating tension to be around $240N$ and the maximum operational tension to be above $400N$.

The motors used in the final design of SUPERball were 100 watt Maxon motors as specified in table 4.1. Each motor has a Maxon gearbox with a maximum continuous output torque of $3Nm$ originally designed to be coupled to a spool with a diameter of $30mm$. Using the spool's radius and the robot's operating tensions listed above, the reaction torques applied to the gearboxes are calculated in equations A.1 and A.2.

$$\tau_{nominal,30mm} = 240N \times 0.015m = 3.6Nm \tag{A.1}$$

$$\tau_{maximum,30mm} = 400N \times 0.015m = 6.0Nm \tag{A.2}$$

These values are well above the continuous and peak torques that the motor's gearbox is rated to handle. To keep from breaking all the motors when operating the robot, smaller diameter spools were designed. The trade off is that the cable actuation velocity is decreased, forcing the robot to move slower. Decreasing the cable actuation velocity too far will result in a robot which will be incapable of achieving a continuous average forward velocity. At the time, the only controller which moved a simulated SUPERball like robot with a continuous velocity was Iscen's work, which required a cable actuation velocity of $30cm/s$ [103]. Thus reducing the speed below 50% could potentially be too slow to achieve continuous forward velocity. The new spindles were then reduced to a spool diameter of $18mm$ and the new torques are shown in equations A.3 and A.4.

$$\tau_{nominal,18mm} = 240N \times 0.009m = 2.7Nm \tag{A.3}$$

$$\tau_{maximum,18mm} = 400N \times 0.009m = 3.6Nm \tag{A.4}$$

This design still is not ideal, but is a compromise between reducing the reaction torque on the gearboxes and not reducing the actuation velocity too low. The torques induced onto the gearbox could still go over the rated value during maximum tension scenarios, but reduces the actuation speed to 47% of the original value set by Iscen. To further reduce the reaction torques applied to the gearboxes, a maximum position of cable change was set to keep the reaction torque applied by any one motor to be within the nominal spool reaction torque in equation A.3. Though experimentation, the limit was found to be $40cm$ of cable change from a pretension value in each cable of $100N$.

A.1.3 Cable Material

The choice of external cable material used on SUPERball made some complications in the long term use of the system. The material used is a $1.3mm$ diameter braided hollow core Vectran cable. This cable has excellent material properties for a system which requires light and strong cables with near zero creep. However, the cable has a high coefficient friction and will undergo tensile fractures when exposed to large amounts of stress. This causes the many thin fibers to fray, eventually leading to loss in overall tensile strength and eventual failure. In SUPERball, this usually occurs at the cabling closest to the spool. Since this small length of cable always experiences high stress due to wrapping around the spool, the cables fray quite rapidly and break. The frequency of this happening depends on how many times that particular cable is used. On average it has become an expectation that a single cable will break after at least three hours of actuation on that single cable. Figure A.3 shows different stages of the vectran cable as it wears to breakage.

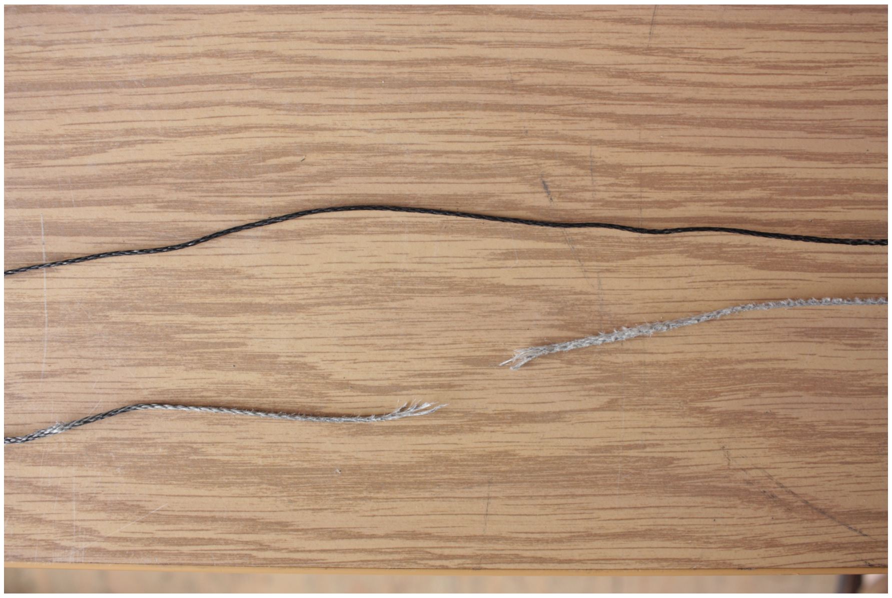


Figure A.3: This figure shows wear on the cable. The vectran cable's protective outer coating wears off during use which then allows for the individual fibers to break weakening the entire cable. Eventually this wear will degrade the max holding tension to the point of failure. The black cable is a new vectran cable and the cable below shows the black coating worn away and a breakage point. The bottom cable is from an actual failure on the system during a run. Since the system can absorb forces, when the cable breaks the stored energy gets distributed into the robot and no violent backlash is experienced.

A.1.4 Force Sensors

SUPERball was designed to have three force sensors per MTR system to sense forces being applied by the motor, the distal actuated cable, and the distal passive cable. Since the system had a constrained budget, purchasing off the shelf force sensors was out of the budget and custom sensors were implemented. Figure A.4 shows the designs for each force sensor.

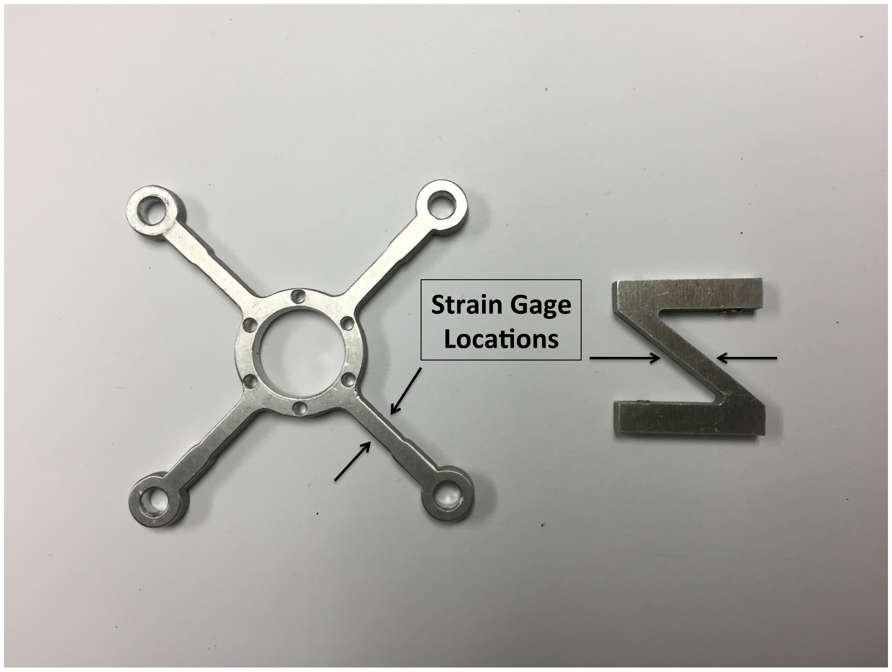


Figure A.4: This figure shows the designs for the motor mount strain gage and the spring strain gage. The motor mount is located on the left, and it functions when the arms of the cross beams deform due to the reaction torque caused by torque being applied to the motor. The right shows the original design for the spring sensors. This gage is suppose to sit between the spring and the ridged mount and compress due to spring force. However, the design never worked.

Dynamic Torque Sensor Testing This test was performed to demonstrate the force sensors' ability to capture data under dynamic motion. Figure A.5 shows a plot of sensor data from one end cap whose motor is commanded in a square-wave position trajectory. The position trajectory had a period of 13 s, and oscillated

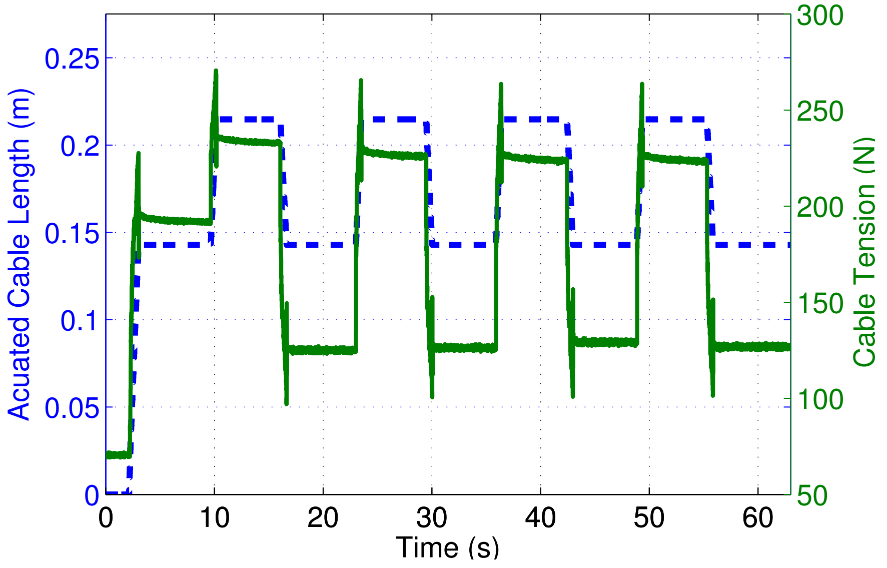


Figure A.5: Motor mount torque sensor data and motor position data recorded during a square wave input position trajectory for a single motor. This plot shows measured tension from the sensor and cable length from motor encoder measurements as a function of time for this dynamic movement.

between 10 rad and 15 rad of the output shaft measured before the gearbox, by the encoder. The trajectory of sensor torque values reasonably tracks the position square wave: the commanded position trajectory starts at 10 seconds and ends at 62 seconds, as does the sensed tension square wave. The overshoot on the torque sensor measurements is due to the system inertia and spring dynamics.

Global Force Redistribution Sensor Testing A test was performed to validate the distribution of tension throughout the system, and to show that all sensors can work in conjunction simultaneously. Figure A.6 shows tension readings from a different motor-mount torque sensor on the opposite side of SUPERball (Cable 2) from a cable which is being retracted (Cable 1.) Cable 2 was not actively actuated during each test. For each plot in Figure A.6, the actuated cable was retracted with various step inputs marked in the figure. Each data point in this figure (yellow) was collected by averaging data from the sensor board for a total of 5 seconds at 1 kHz, after waiting 2 seconds after the step input actuation to avoid dynamic effects. These tests were done with different levels of pretension on the sensed cable: this pretension was adjusted by changing the length of the sensed cable. Though the lower-pretension tests show smaller changes in readings, the higher pretensions show increasing readings which demonstrate the ability to sense forces throughout the tension network in pseudo-equilibrium states, as well as SUPERball's passive force redistribution properties.

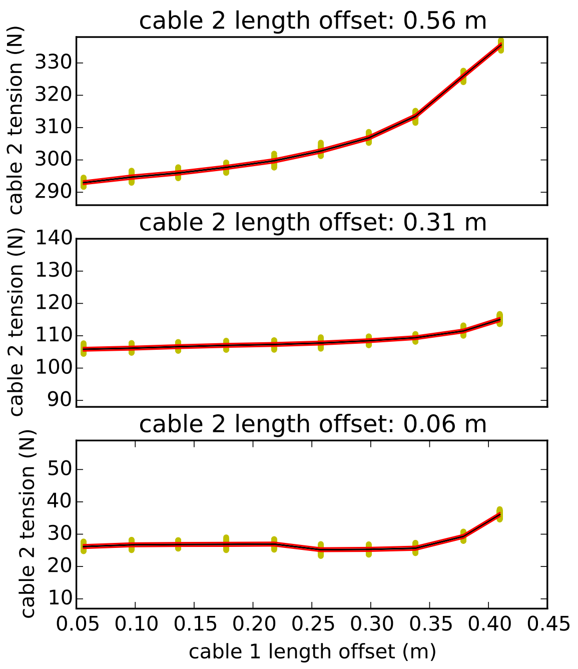


Figure A.6: Global force redistribution test. Yellow marks are the means of roughly 5,000 tension sensor measurements of *cable 2* opposite that which is actuated (*cable 1*.) The black line shows the linear interpolation between points, with the red boundary as standard deviation. The pretension in the sensed cable is adjusted in each test, showing measurement sequences at increasing pretensions.

Force Sensing Viability The force reaction sensor was the only custom sensor to accurately sense the forces applied. However, inconsistencies in manufacturing this sensor made calibration of each sensor extremely difficult. A single sensor could be made, calibrated and functional for about a day or two after which the sensor's strain gages would shift. This either would make the calibration no longer valid or be such a shift, that the sensor would no longer be functional. Due to these set backs, the force sensors were disregarded as viable sensors to be used on the system.

A.2 Electrical Evaluation

A.2.1 Sensor Board

The sensor board eventually functioned as designed. All major functions that were set out in its inception were implemented. It was able to deliver the CAN physical layer for the Beagle Bone Black, interpret and condition sensor data, and enable the addition of the DWM module (which was not in the initial design). The only caveat in its sound function was driver code developed for the DWM module. If the DWM module crashed, usually due to the inability to handle soft system resets, the SPI communication line would block the sensor board code. This was a rare occurrence, but would require a full power cycle of the sensor board to recover. The correct fix is to implement the DWM module's SPI code to be non-blocking, however this was not a higher priority than finishing other higher level control experiments.

A.2.2 Power Board

This board was the "heart" of SUPERball and functioned extremely well. It distributed conditioned power to all electronics and sensors on SUPERball as well as managed many of the safety features on SUPERball. One down side to this board was its size. Since it has many complex features implemented in discrete logic, the board has 136 components. This made the board expensive and large, with some extra features which were never utilized in the final implementation of SUPERball. To improve upon this, designing the board as a two sided PCB would reduce the size dramatically. Furthering reducing the size and complexity would be to not support 6 power lines. These were originally implemented to support expansion boards. However, this feature was never utilized and in hind sight would never need to support that many extra boards.

A.2.3 Motor Board

For the most part, all the electronic boards on SUPERball functioned as designed with only the motor control board having lasting issues that hindered the performance of the system. The motor board was supposed to be a low cost solution to achieve position, velocity, and torque control utilizing the Field Oriented Control (FOC) method to commutate the brushless DC motors. A third party start-up company was tasked with the design of the motor board. It took the company many attempts to get a working version for just position control, with the initial delivery causing motors to generate excessive heat due to shorting the windings during commutation. After a quick redesign and manufacturing new boards, position control was achieved with pretty good results. However, the company went out of business after their delivery of working motor control boards and never completed their implementation of the other control methods. This in conjunction with the issues stated in section A.1.4, torque control was never implemented.

A.3 Communications Evaluation

There were two main types of data communication on SUPERball, a CAN bus and the wireless WiFi network enabled on each rod.

A.3.1 CAN Bus

Since SUPERball was designed to have multiple boards communicating on a single network over distances greater than one meter, a CAN bus was implemented to achieve these goals. For the most part, this network along with the CANOpen standard met all of the initial design criteria. However, sending a small number of messages at 1 kHz would over flow the bus at the maximum network speed of $1 Mbit/s$. This is in part due to the large CANOpen message header and the large data types bing transmitted. The protocol also uses a lot of system resources to manage. It was measured that approximately 10% of the micro controllers' total computation went into handling CAN messages.

A.3.2 WiFi Network

SUPERball relied upon standard WiFi communication through ROS to transmit data from the robot to an external computer on the network. ROS messaging is extremely convenient for data logging, management, and conditioning all shared over a common network. The main issue experienced with this implementation on SUPERball is due to the ARM based network driver support and WiFi dongles used. These issues result in unexpected network lags and connection issues to the network. Several driver and module addition were required to get the WiFi communication on the Beagle Bone Blacks to be stable and with a relative low network latency. However, these fixes never solved the network connection issue.

The maximum range of the system was also limited due to the budget home router used. Since purchasing though a government entity put limits on computer hardware, it was difficult to find a good wireless router which fit the design needs and budget. Therefore, a budget router was purchased and the routing feature on the wireless router was disabled. In its place, a Linux computer was setup as the router. The budget wireless router (now a wireless access point), had a very weak signal and limited the systems range to about 12 m in an open space.

Bibliography

[1] Adrian K. Agogino, Vytas SunSpiral, and David Atkinson. Super Ball Bot - structures for planetary landing and exploration. *NASA Innovative Advanced Concepts (NIAC) Program, Final Report*, 2013.

[2] Vytas SunSpiral, George Gorospe, Jonathan Bruce, Atil Iscen, George Korbel, Sophie Milam, Adrian K. Agogino, and David Atkinson. Tensegrity based probes for planetary exploration: Entry, descent and landing (EDL) and surface mobility analysis. In *10th International Planetary Probe Workshop (IPPW)*, July 2013.

[3] Jonathan Bruce, Ken Caluwaerts, Atil Iscen, Andrew P. Sabelhaus, and Vytas SunSpiral. Design and evolution of a modular tensegrity robot platform. In *ICRA*, pages 3483–3489, May 2014.

[4] Atil Iscen, Ken Caluwaerts, Jonathan Bruce, Adrian Agogino, Vytas SunSpiral, and Kagan Tumer. Learning tensegrity locomotion using open-loop control signals and coevolutionary algorithms. *Artificial life*, 2015.

[5] Andrew P Sabelhaus, Jonathan Bruce, Ken Caluwaerts, Pavlo Manovi, Roya Fallah Firoozi, Sarah Dobi, Alice M Agogino, and Vytas SunSpiral. System design and locomotion of SUPERball, an untethered tensegrity robot. In *ICRA*, pages 2867–2873, 2015.

[6] K. Caluwaerts, J. Bruce, J. M. Friesen, and V. SunSpiral. State estimation for tensegrity robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1860–1865, May 2016.

[7] Xinyang Geng, Marvin Zhang, Jonathan Bruce, Ken Caluwaerts, Massimo Vespignani, Vytas SunSpiral, Pieter Abbeel, and Sergey Levine. Deep reinforcement learning for tensegrity robot locomotion. *arXiv preprint arXiv:1609.09049*, 2016.

[8] Mark W Maimone, P Chris Leger, and Jeffrey J Biesiadecki. Overview of the mars exploration rovers' autonomous mobility and vision capabilities. In *IEEE international conference on robotics and automation (ICRA) space robotics workshop*, 2007.

[9] John P Grotzinger, Joy Crisp, Ashwin R Vasavada, Robert C Anderson, Charles J Baker, Robert Barry, David F Blake, Pamela Conrad, Kenneth S Edgett, Bobak Ferdowski, et al. Mars science laboratory mission and science investigation. *Space science reviews*, 170(1-4):5–56, 2012.

[10] R. B. Fuller. *Synergetics: Explorations in the Geometry of Thinking.* Scribner, January 1975.

[11] Kenneth Snelson. Continuous tension, discontinuous compression structures. United States patent 3169611, February 1965.

[12] Ning Wang, Jessica D Tytell, and Donald E Ingber. Mechanotransduction at a distance: Mechanically coupling the extracellular matrix with the nucleus. *Nature Reviews Molecular Cell Biology*, 10(1):75–82, 2009.

[13] N. Wang, K. Naruse, D. Stamenović, J. J. Fredberg, S. M. Mijailovich, I. M. Tolić-Nørrelykke, T. Polte, R. Mannix, and D. E. Ingber. Mechan-

ical behavior in living cells consistent with the tensegrity model. *PNAS*, 98(14):7765–7770, Jul 2001.

[14] Etienne Fest, Kristina Shea, Ian F C Smith, and M Asce. Active Tensegrity Structure. *Journal of Structural Engineering*, 130(10):1454–1465, 2004.

[15] Cornel Sultan, Martin Corless, and Robert Skelton. Tensegrity flight simulator. *Journal of Guidance, Control, and Dynamics*, 23(6):1055–1064, 2000.

[16] C. Paul, F. J. Valero-Cuevas, and H. Lipson. Design and control of tensegrity robots for locomotion. *IEEE Transactions on Robotics*, 22(5), October 2006.

[17] Ken Caluwaerts, Jérémie Despraz, Atıl Işçen, Andrew P. Sabelhaus, Jonathan Bruce, Benjamin Schrauwen, and Vytas SunSpiral. Design and control of compliant tensegrity robots through simulation and hardware validation. *Journal of The Royal Society Interface*, 11(98), 2014.

[18] T. Bliss, J. Werly, T. Iwasaki, and H. Bart-Smith. Experimental validation of robust resonance entrainment for CPG-controlled tensegrity structures. *IEEE Transactions On Control Systems Technology*, 21:666–678, 2012.

[19] A. Graells Rovira and J. M. Mirats-Tur. Control and simulation of a tensegrity-based mobile robot. *Robotics and Autonomous Systems*, 57(5):526–535, May 2009.

[20] J.M. Mirats-Tur and J. Camps. A three-DoF actuated robot. *Robotics Automation Magazine, IEEE*, 18(3):96–103, Sept 2011.

[21] Mark Khazanov, Julian Jocque, and John Rieffel. Developing morphological computation in tensegrity robots for controllable actuation. In *GECCO*, pages 1049–1052, 2014.

[22] V. Bohm and K. Zimmermann. Vibration-driven mobile robots based on single actuated tensegrity structures. In *ICRA*, pages 5475–5480, May 2013.

[23] M. Shibata, F. Saijyo, and S. Hirai. Crawling by body deformation of tensegrity structure robots. In *ICRA*, pages 4375 –4380, may 2009.

[24] Gunnar Tibert. *Deployable Tensegrity Structures for Space Applications*. PhD thesis, Royal Institute of Technology, 2002.

[25] R. Pfeifer, M. Lungarella, and F. Iida. The Challenges Ahead for Bio-Inspired "Soft" Robotics. *Comm. of the ACM*, 55(11):76–87, 2012.

[26] S. Kim, C. Laschi, and B. Trimmer. Soft Robotics: A Bioinspired Evolution in Robotics. *Trends in Biotechnology*, 31(5), 2013.

[27] C. Majidi. Soft Robotics: A Perspective - Current Trends and Prospects for the Future. *Soft Robotics*, 1(1):5–11, 2014.

[28] Helmut Hauser, Auke J Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer, and Wolfgang Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105:355–370, January 2012.

[29] D. Zambrano, M. Cianchetti, and C. Laschi. Morphological Computation Principles as a New Paradigm for Robotic Design. In *Morphological Computation*, pages 214–225. 2014.

[30] R. Deimel and O. Brock. Soft Hands for Reliable Grasping Strategies. In A. Verl, A. Albu-Schaffer, O. Brock, and A. Raatz, editors, *Soft Robotics*. Springer, 2015.

[31] M. Khazanov, J. Jocque, and J. Rieffel. Developing Morphological Computation in Tensegrity Robots for Controllable Actuation. In *GECCO*, pages 1049–1052, 2014.

[32] D. Rus and M. T. Tolley. Design, Fabrication and Control of Soft Robots. *Nature*, 521:467–475, 2015.

[33] B. Trimmer. Soft Robot Control Systems: A New Grand Challenge? *Soft Robotics*, 1(4):231–232, 2014.

[34] H.-T. Lin, C. G. Leisk, and B. Trimmer. GoQBot: a Caterpillar Soft-bodied Rolling Robot. *Bioinsp. and Biomim.*, 6(2), 2011.

[35] F. Saunders, B. Trimmer, and J. Rife. Modeling Locomotion of a Soft Anthropod using Inverse Dynamics. *Bioinsp. and Biomim.*, 6(1), 2011.

[36] R. E. Skelton and M. C. De Oliveira. *Tensegrity Systems*. Springer, 2009 edition, June 2009.

[37] R. J. Webster and B. A. Jones. Design and Kinematic Modeling of Constant Curvature Continuum Robots. *IJRR*, 29, 2010.

[38] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi. Dynamic Model of a Multi-Bending Soft Robot Arm Driven by Cables. *TRO*, 30:1109–1122, 2014.

[39] J. M. Germann, A. Maesani, M. Stockli, and D. Floreano. Soft Cell Simulator: A Tool to Study Multi-Cellular Robots. In *ROBIO*, 2013.

[40] NASA Tensegrity Robotics Toolkit. `ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity/ntrt`.

[41] G. G. Rigatos. Model-based and Model-free Control of Flexible-link Robots: A Comparison between Representative Methodsœ. *Applied Mathematical Modelling*, 33:3906–3925, Oct. 2009.

[42] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez. Real-time Control of Soft-Robots using Asynchronous Finite Element Modeling. In *ICRA*, 2015.

[43] V. Vikas, P. Grover, and B. Trimmer. Model-free Control Framework for Multi-Limb Soft Robots. In *IROS*, 2015.

[44] C. Armbrust, L. Kiekbusch, T. Ropertz, and K. Berns. Soft Robot Control with a Behaviour Architecture. In A. Verl, A. Albu-Schaffer, O. Brock, and A. Raatz, editors, *Soft Robotics*. Springer, 2015.

[45] S. Neppalli, M. A. Csenscits, B. A. Jones, and I. D. Walker. Closed-From Inverse Kinematics for Continuum Manipulators. *Advanced Robotics Journal*, 23(2077-2091), 2009.

[46] V. SunSpiral, G. Gorospe, J. Bruce, A. Iscen, G. Korbel, et al. Tensegrity Based Probes for Planetary Exploration: Entry, Descent and Landing (EDL) and Surface Mobility Analysis. In *IPPW*, 2013.

[47] J. M. Porta and S. Hernández Juan. Path planning for active tensegrity structures. *IJSS*, 78–79:47 – 56, 2016.

[48] B. Mirletz, P. Bhandal, R. D. Adams, A. K. Agogino, R. D. Quinn, and V. SunSpiral. Goal directed CPG based control for high DOF tensegrity spines traversing irregular terrain. *Soft Robotics*, 2015.

[49] Jeroen Burms, Ken Caluwaerts, and Joni Dambre. Online unsupervised terrain classification for a compliant tensegrity robot using a mixture of echo

state networks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4252–4257. IEEE, 2015.

[50] M. Bonilla, E. Farnioli, L. Pallotino, and A. Bicchi. Sample-Based Planning for Soft Arms Under Task Constraints. In *ICRA*, 2015.

[51] A. D. Marchese, R. Tedrake, and D. Rus. Trajectory Optimization for a Soft Spatial Fluidic Elastomer Manipulator. *IJRR*, 2016.

[52] S. Hernandez Juan, R.E. Skelton, and J.M. Mirats Tur. Dynamically stable collision avoidance for tensegrity based robots. In *Reconfigurable Mechanisms and Robots, 2009. ReMAR 2009. ASME/IFToMM International Conference on*, pages 315–322, June 2009.

[53] Josep M Mirats Tur and S Juan. Tensegrity frameworks: Dynamic analysis review and open problems. *Mechanism and Machine Theory*, 44(1):1–18, 2009.

[54] C. Sultan, M. Corless, and R. E. Skelton. Linear dynamics of tensegrity structures. *Engineering Structures*, 24(6):671–685, 2002.

[55] A. S. Wroldsen, M. C. de Oliveira, and R. E. Skelton. A Discussion on Control of Tensegrity Systems. In *CDC*, 2006.

[56] Calladine C.R. Buckminster Fuller's Tensegrity structures and Clerk Maxwell's rules for the construction of stiff frames. *International Journal of Solids and Structures*, 14(2):161–172, 1978.

[57] J. Friesen, A. Pogue, T. Bewley, M. C. de Oliveira, R. E. Skelton, and V. SunSpiral. DuCTT: A tensegrity robot for exploring duct systems. In *ICRA*, pages 4222–4228, May 2014.

[58] R. Motro. *Tensegrity: Structural Systems of the Future.* Kogan, 2003.

[59] S D Guest. The stiffness of tensegrity structures. *IMA Journal of Applied Mathematics*, 76(1):57–66, 2010.

[60] Sergi HernÃăndez Juan and Josep M. Mirats Tur. Tensegrity frameworks: Static analysis review. *Mechanism and Machine Theory*, 43(7):859 – 881, 2008.

[61] Sergio Pellegrino. *Deployable structures*, volume 412. Springer, 2014.

[62] A. G. Rovira and J. M. Mirats Tur. Control and Simulation of a Tensegrity-based Mobile Robot. *RAS*, 57(5):526–535, 2009.

[63] T. Bliss, T. Iwasaki, and H. Bart-Smith. Central Pattern Generator Control of a Tensegrity Swimmer. *Trans. on Mech.*, 18(2), 2013.

[64] Kyunam Kim, Adrian K Agogino, Aliakbar Toghyan, Deaho Moon, Laqshya Taneja, and Alice M Agogino. Robust learning of tensegrity robot control for locomotion through form-finding. In *IROS*, 2015.

[65] W.L. Chan, F. Bossens D. Arbelaez, and R.E. Skelton. Active vibration control of a three-stage tensegrity structure. In *SPIE 11th Annual International Symposium on Smart Structures and Materials*, San Diego, March 2004.

[66] Y. Koizumi, M. Shibata, and S. Hirai. Rolling tensegrity driven by pneumatic soft actuators. In *ICRA*, pages 1988–1993, 2012.

[67] John Rieffel, Franciso Valero-Cuevas, and Hod Lipson. Morphological communication: exploiting coupled dynamics in a complex mechanical structure

to achieve locomotion. *Journal of the Royal Society Interface*, 7:613–621, 2009.

[68] B. T. Mirletz, I. W. Park, R. D. Quinn, and V. SunSpiral. Towards bridging the reality gap between tensegrity simulation and robotic hardware. In *IROS*, 2015.

[69] Nicolas Veuve, Seif Dalil Safaei, and Ian F. C. Smith. Deployment of a Tensegrity Footbridge. *JSE*, 2015.

[70] K. Caluwaerts, J. M. Friesen, J. Bruce, and V. SunSpiral. State Estimation for Tensegrity Robots. In *(arXiv:1510.01240)*, 2016.

[71] M. Arsenault and C. M. Gosselin. Kinematic and Static Analysis of a 3 DoF Spatial Modular Tensegrity Mechanism. *IJRR*, 27(8), 2008.

[72] N. Kanchanasaratool and D. Williamson. Motion control of a tensegrity platform. *CIS*, 2(3):299–324, 2002.

[73] M. C. de Oliveira. Dynamics of systems with rods. In *CDC*, 2006.

[74] R. E. Skelton and C. Sultan. Controllable Tensegrity: A New Class of Smart Structures. In *Proc. of the SPIE*, 1997.

[75] J. B. Aldrich, R. E. Skelton, and K. K. Delgado. Control Synthesis for Light and Agile Robotic Tensegrity Structures. In *ACC*, 2003.

[76] K. A. McIsaac and J. P. Ostrowski. Motion Planning for Anguilliform Locomotion. *TRO*, 19(4):637–652, 2003.

[77] J. Rieffel, F. Valero-Cuevas, and H. Lipson. Automated Optimization of Large Irregular Tensegrity Structures. *Comp. and Structures*, 2009.

[78] A. Iscen, A. K. Agogino, V. SunSpiral, and K. Tumer. Controlling Tensegrity Robots Through Evolution. In *GECCO*, 2013.

[79] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[80] R Paul Wiegand, William C Liles, and Kenneth A De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the genetic and evolutionary computation conference (GECCO)*, volume 2611, pages 1235–1245, 2001.

[81] Richard Lippmann. An introduction to computing with neural nets. *IEEE Assp magazine*, 4(2):4–22, 1987.

[82] R. Tedrake, T. Zhang, and H. Seung. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *IROS*, 2004.

[83] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IROS*, 2004.

[84] R. Calendra, A. Seyfarth, J. Peters, and M. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 76:5–23, 2016.

[85] J. Kolter and A. Ng. The Stanford LittleDog: A learning and rapid replanning approach to quadruped locomotion. *IJRR*, 30(2):150–174, 2011.

[86] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal. Learning locomotion over rough terrain using terrain templates. In *IROS*, 2009.

[87] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel. Trust region policy optimization. In *ICML*, 2015.

[88] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[89] W. Montgomery and S. Levine. Guided policy search as approximate mirror descent. In *NIPS*. 2016.

[90] T. Geng, B. Porr, and F. Wörgötter. Fast biped walking with a reflexive controller and realtime policy searching. In *NIPS*, 2006.

[91] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng. Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot. *IJRR*, 27(2), 2008.

[92] S. Levine and V. Koltun. Guided policy search. In *ICML*, 2013.

[93] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, 2014.

[94] Zakary Littlefield, Ken Caluwaerts, Jonathan Bruce, Vytas SunSpiral, and Kostas E Bekris. Integrating simulated tensegrity models with efficient motion planning for planetary navigation. June, 2016.

[95] Yanbo Li, Zakary Littlefield, and Kostas E Bekris. Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5):528–564, 2016.

[96] Ken Caluwaerts, Michiel D'Haene, David Verstraeten, and Benjamin

Schrauwen. Locomotion without a brain: physical reservoir computing in tensegrity structures. *Artificial Life*, 19(1):35–66, 2013.

[97] R. E. Skelton and M. C. De Oliveira. *Tensegrity Systems*. Springer, 2009 edition, June 2009.

[98] Kyunam Kim, Adrian K. Agogino, Deaho Moon, Laqshya Taneja, Aliakbar Toghyan, Borna Dehghani, Vytas SunSpiral, and Alice M. Agogino. Rapid prototyping design and control of tensegrity soft robot for locomotion. In *To Appear in ROBIO*, 2014.

[99] Robert Skelton. Dynamics and control of tensegrity systems. In H. Ulbrich and W. GÃIJnthner, editors, *IUTAM Symposium on Vibration Control of Nonlinear Mechanisms and Structures*, volume 130 of *Solid Mechanics and its Applications*, pages 309–318. Springer Netherlands, 2005.

[100] Erwin Coumans. Bullet physics engine, 2012.

[101] S Baker. The freeglut project, 2008.

[102] Brian T Mirletz, Perry Bhandal, Ryan D Adams, Adrian K Agogino, Roger D Quinn, and Vytas SunSpiral. Goal-directed cpg-based control for tensegrity spines with many degrees of freedom traversing irregular terrain. *Soft Robotics*, 2(4):165–176, 2015.

[103] Atil Iscen, Adrian Agogino, Vytas SunSpiral, and Kagan Tumer. Flop and roll: Learning robust goal-directed locomotion for a tensegrity robot. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2236–2243. IEEE, 2014.

[104] Robert Osfield, Don Burns, et al. Open scene graph, 2004.

[105] RD Lorenz. Huygens probe impact dynamics. *ESA Journal*, 18:93–117, 1994.

[106] Ken Caluwaerts, Jérémie Despraz, Atıl Işçen, Andrew P Sabelhaus, Jonathan Bruce, Benjamin Schrauwen, and Vytas SunSpiral. Design and control of compliant tensegrity robots through simulation and hardware validation. *Journal of The Royal Society Interface*, 11(98):20140520, 2014.

[107] Brian T Mirletz, In-Won Park, Roger D Quinn, and Vytas SunSpiral. Towards bridging the reality gap between tensegrity simulation and robotic hardware. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5357–5363. IEEE, 2015.

[108] Pavlo Manovi. Pmsm. `https://github.com/FrauBluher/PMSM`, 2014.

[109] H Boterenbrood. Canopen high-level protocol for can-bus. *Nikhef, Amsterdam*, 2000.

[110] DecaWave. APS011 application note: Sources of error in DW1000 based two-way ranging (TWR) schemes.

[111] Roberto Battiti and Francesco Masulli. Bfgs optimization for faster and automated supervised learning. In *International neural network conference*, pages 757–760. Springer, 1990.

[112] Eric Wan, Ronell Van Der Merwe, et al. The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE, 2000.

[113] Robert E Skelton and Mauricio C Oliveira. *Tensegrity systems.* Springer, 2009.

[114] Katsu Yamane and Yoshihiko Nakamura. Stable penalty-based model of frictional contacts. In *ICRA*, pages 1904–1909, 2006.

[115] Sebastian OH Madgwick, Andrew JL Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *IEEE International Conference on Rehabilitation Robotics (ICORR2011)*, pages 1–7, 2011.

[116] Thomas Back, David B Fogel, and Zbigniew Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[117] J. Bagnell and J. Schneider. Covariant policy search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[118] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21:682–697, 2008.

[119] J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

[120] J Andrew Bagnell, Sham M Kakade, Jeff G Schneider, and Andrew Y Ng. Policy search by dynamic programming. In *Advances in neural information processing systems*, page None, 2003.

[121] FLANN: Fast Library for Approximate Nearest Neighbors. `http://www.cs.ubc.ca/research/flann/`.

[122] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 17:1334–1373, 2016.

[123] IEEE Computer Society LAN MAN Standards Committee et al. Wireless

lan medium access control (mac) and physical layer (phy) specifications, 1997.