Spring 6-1-2021

Object Manipulation with Modular Planar Tensegrity Robots

Maxine Perroni-Scharf
maxine.a.perroni-scharf.21@dartmouth.edu

Object Manipulation with Modular Planar Tensegrity Robots

A Thesis Submitted to The Department of Computer Science of
Dartmouth College
In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Arts in Computer Science

Maxine Perroni-Scharf
Spring 2021

Supervisor: Devin Balkcom

Object Manipulation with Modular Planar Tensegrity Robots*

Maxine Perroni-Scharf§
Dartmouth College

May 2021

Abstract

This thesis explores the creation of a novel two-dimensional tensegrity-based modular system. When individual planar modules are linked together, they form a larger tensegrity robot that can be used to achieve non-prehensile manipulation. The first half of this dissertation focuses on the study of preexisting types of tensegrity modules and proposes different possible structures and arrangements of modules. The second half describes the construction and actuation of a modular 2D robot composed of planar three-bar tensegrity structures. We conclude that tensegrity modules are suitably adapted to object manipulation and propose a future extension of the modular 2D design to a modular 3D design.

Keywords: Tensegrity structures, modular robots, re-configurable robots, object manipulation, soft robots

1 Introduction

This paper explores the design and use of modular tensegrity structures (19) to achieve object manipulation. Tensegrity structures are lightweight, durable, energy-efficient and

* I am grateful to my supervisor, Devin Balkcom, for guidance and suggestions. I am also grateful to Luyang Zhao and Julien Blanchet for their collaboration on this project.

§ Class of 2021.

compliant, giving them unique advantages over other traditional types of autonomous robots in a number of applications. As tensegrity robots are difficult to control due to their complex structure, one of the greatest challenges in the emerging field of tensegrity research is finding effective methods and uses for tensegrity automation. While tensegrity locomotion has been widely researched, tensegrity structures also hold potential for composing larger modular systems that can be used for object manipulation, where achieving effective automation becomes even more challenging.

Throughout this project, I worked together with Professor Devin Balkcom and PhD students Luyang Zhao and Julien Blanchet. I worked on the design, actuation, curve approximation and object manipulation algorithm for a planar tensegrity system, which is the main focus of this dissertation. Zhao worked on energy calculations for the planar system and designed the 3D modular system covered at the end of this paper. Blanchet helped me choose motors and materials for the planar system.

Section 1 of this thesis gives an introduction and overview of the research project. Section 2 reviews related work.

Sections 3 and 4 cover simulation and building of pre-existing tensegrity structures. We started this research project by individually simulating and building three-dimensional three-bar triangular lattice tensegrities. We explored a range of candidate formations and motions, including end-to-toe stacked and side-by-side lattice structures. Of these, a stacked formation is best for locomotion, but side-by-side lattice formations are best for object manipulation. We concluded that we wanted to design a new structure that is more suited to create regular and easily-assembled lattice formations than traditional tensegrities.

Section 5 covers the design and actuation of a new planar (2D) three-bar tensegrity structure. We proposed an end-to-end structural formation that employed these planar three-bar structures as its component modules, and proceeded to analyse the degrees of freedom in the system to determine the minimum number of motors that were needed to achieve different types of motion. We also used Lagrange Multipliers methods to find the bi-stable configurations in the system, relying on a methodology developed by Luyang Zhao. Based on this new modular structure, we developed an object manipulation algorithm that allows an object to travel along the top of the modular system by passing over several tensegrity modules, utilizing gravity and slopes to roll along dips created by the algorithm. We simulated this motion using Maya and built a working version of this sys-

tem theta employs pulleys. We determined that in order to create a more rigorous and adaptable object manipulation method than the hard-coded algorithm, we would need to find a general way to express motion along the top of the robot. To do this, we developed a method for curve approximation along the top of the structure, which enabled us to simulate peristaltic waves passing along the top of the system. We built a physical version of the structure, incorporating DC motors, which is capable of manipulating a puck across the top of the structure. We wrote code directing the motors to execute these object manipulation algorithms, and conducted trials to determine the effectiveness of our method.

Finally, Section 6 of this paper discusses an extension of this planar design to the three dimensional case, where each module is a four-bar tensegrity structure. We combined these 3D modules into a larger lattice structure and developed an object manipulation protocol for rolling a ball across the top of this structure. We then 3D printed the whole structure as a soft skeleton and actuated it using shape-memory alloy (SMA) wire. Through this work, we conclude that modular tensegrity robots are well-adapted to object manipulation.

2 Related Work

There is much prior research surrounding tensegrity robots and their applications, following on the seminal work by NASA on the Super Ball Bot in 2014. The Super Ball Bot is an all-in-one landing and mobility platform that uses a icosahedral tensegrity structure with six struts and twenty-four cables as its basis. This robot consists of compressive rigid bars connected with tensile spring-cable systems, and due to its soft nature, it can withstand being dropped from high above a planetary surface. By actuating the spring-cable tensile components of the robot, the Super Ball Bot can achieve a wide range of motions, including locomotion around a planetary surface. Due to its light weight, it is especially well-adapted to maneuvering over hazardous terrain, as it will not sink and get stuck (5; 17; 4).

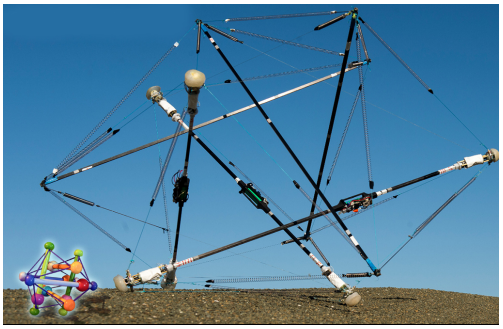

Figure 1: NASA Super Ball Bot

Following the creation of the Super Ball Bot, there have been a large amount of studies on different types of tensegrity structures composed of varying amounts of struts and cables. The mechanics of these structures have been extensively analysed, and analytical proofs have been given on the fact that most basic tensegrity structures are able to change shape substantially with little change in potential energy to the system, making them well suited for energy-efficient locomotion. Prior work includes derivation of the dynamic equations of tensegrity systems, and the development of useful tensegrity simulation and state estimation methods. (16; 14).

The robustness of tensegrity structures has also been extensively studied: their light but strong nature makes them particularly well-adapted to object manipulation with relatively large loading forces (18). Tensegrities are particularly well adapted to space exploration, as they can survive falls from a great height. Because of this, they can be deployed from above a planets surface and roam around the terrain shortly after landing (7).

The locomotion of tensegrity structures has also been widely-explored. Due to their complex structures, there are significant challenges involved with state estimation and understanding the exact motions of tensegrity robots. Therefore, the majority of tensegrity locomotion research focusing on machine-learning approaches to develop movement algorithms. For example, evolutionary algorithms have proven successful in locomoting three-bar and four-bar tensegrity robots both in simulation and real life, where the controllers adopted both static and dynamic gaits for both types of robots (15). Prior work has also explored the use of deep-reinforcement learning for producing reliable continu-

ous gaits in larger tensegrity structures including the icosahedral Super Ball Bot structure (22; 21).

Subsequent research influenced by these tensegrity designs has explored non-modular tensegrity robots with soft, compliant, structures inspired by real-world biological structures. For example, these robots can mimick the function of tendons or vertebral columns, a notion first alluded to in Marvin Minsky's 1981 memo on Manipulator Design Vignettes (12). Recent research has found that a tensegrity-truss is a suitable model for the spines of many different types of animals, including fish, birds and humans (10; 13).

Soft biology inspired tensegrity robots, built with rods, tension springs and motorized cables, lend themselves to many different modes of motion (8; 9). These tensegrity structures are known as "bio-tensegrity robots", and take inspiration from a variety of animals. For example, some robot designs have been based on a form of "earthworm-like" peristaltic motion to achieve locomotion through tightly constrained spaces (3). The worm robot is segmented into small tensegrity modules which are permanently attached to each other (thus it is not a modular tensegrity system). Then, this peristaltic motion is informed by a rolling-wave algorithm that continuously passes a periodic motion across the segments of the body of the worm robot by contracting the tensile connectors that occur between segments.

Simple microscopic robots have also been developed that utilize inchworm-like friction-based locomotion (6). This methodology could be used to inform the motion of microscopic bio-tensegrities, and be particularly useful in developing locomotion protocols for individual tiny tensegrity modules.

There is also much prior work on the coordination of non-tensegrity multi-robot systems and the use of these systems for object manipulation. There are many advantages to multi-robot systems (1), and they are particularly well suited for the transportation of large objects in hazardous environments. There are a variety of approaches to coordinate the motions of robots in these systems. For example cluster space control (11) relies on a pilot to use a joystick and manually coordinate four robots. These robots can then be used together for object transportation. Decentralized control methods have also been studied that allow robots to identify and approach a target object and move around to transport it together (20). The majority of such approaches rely on the robots themselves to travel distances and move around the surface, but there is less existing work on static robots that are able to pass objects between them.

3 System Simulation

Tensegrity robots can have many degrees of freedom, and are generally difficult to design and simulate for this reason. For example, a tensegrity module composed of three non-compliant struts and nine compliant cables has eighteen degrees of freedom, as its position can only be fully determined by knowledge of the 3D coordinates of the two end positions of each strut. Prior research (2) seeks to identify and analyse complex tensegrity structures with as few degrees of freedom as possible.

In order to further explore the viable motions and degrees of freedom of these complex tensegrity structures, I wrote a Python simulator that allows the user to simulate free-moving tensegrity structures given strict distance constraints on the length of each non-compliant strut. These simulations were helpful in understanding exactly how many extra constraints would be needed to develop tensegrity modules whose motions can be meaningfully actuated.

3.1 Differential-constraint Based Simulation

The Python simulator I wrote adopts a differential-constraint approach, which consists of simulating viable motions of a system while preserving linear constraints within the system.

Consider a one-bar arm anchored to the origin of 2D axes. If the length of the arm is undefined (i.e. unconstrained), there are two degrees of freedom in this system: the x-coordinate of the end of the arm, and the y-coordinate of the end of the arm.

Next, consider how to preserve the arm-length over time, i.e. how to constrain it to be of a certain length. Without loss of generality, assume this length to be 1. In order to preserve the constraint that the length of the arm must always be 1, we can write:

$$f(x, y) = 0, \tag{1}$$

where

$$f(x, y) = x^2 + y^2 - 1. \tag{2}$$

We can consider $f(x, y)$ here to be a measure of the error of the system, as whenever $f(x, y)$ is not 0, we are violating our constraint by an amount $f(x, y)$.

Now, let us consider x and y as functions of time. Then:

$$f\big(x(t), y(t)\big) = x(t)^2 + y(t)^2 - 1. \tag{3}$$

Suppose our initial state is $t = 0$, with $x(t) = 1$ and $y(t) = 0$, so $f\big(x(0), y(0)\big) = 0$ (i.e. the constraint is not violated at $t = 0$).

Now, we want this to remain at all t, as we do not want the error to change from 0 at any point. Therefore, we can differentiate f with respect to t to find out what values of $x(t)$ and $y(t)$ are needed to ensure that f does not change over time (i.e. that f has a total time gradient of zero).

By the multi-variable chain rule:

$$\frac{d}{dt} f\big(x(t), y(t)\big) = 2x\dot{x} + 2y\dot{y}. \tag{4}$$

Now, we want $(d/dt)\, f\big(x(t), y(t)\big) = 0$, so we impose $2x\dot{x} + 2y\dot{y} = 0$; which we can also express as

$$[2x\ 2y]\ [\dot{x}\ \dot{y}]^T = 0, \tag{5}$$

where $[2x\ 2y]$ is the Jacobian matrix of our system (in this case, there is one row in the matrix as there is only one constraint on the system). Then, in order to find out what values of $\dot{x}$ and $\dot{y}$ are needed to satisfy this equation, we can simply take the null space of our Jacobian matrix. If we then add $\dot{x}$ and $\dot{y}$ to x and y, we will minimize the constraint violations within the system.

Consider a three-bar triangular lattice tensegrity with three struts of length 1. Let the ends of strut 1 be (x_1, y_1, z_1) and (x_2, y_2, z_2). Let the ends of strut 2 be (x_3, y_3, z_3) and (x_4, y_4, z_4). Let the ends of strut 3 be (x_5, y_5, z_5) and (x_6, y_6, z_6). Then the constraints on this system are as follows:

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 = 1, \tag{6}$$

$$(x_3 - x_4)^2 + (y_3 - y_4)^2 + (z_3 - z_4)^2 = 1, \tag{7}$$

$$(x_3 - x_4)^2 + (y_3 - y_4)^2 + (z_3 - z_4)^2 = 1. \tag{8}$$

Therefore, the Jacobian Matrix of this system would be:

$$\begin{pmatrix} 2x_1-2x_2 & 2x_2-2x_1 & 0 & 0 & 0 & 0 & 2y_1-2y_2 & 2y_2-2y_1 & 0 & 0 & 0 & 0 & 2z_1-2z_2 & 2z_2-2z_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2x_3-2x_4 & 2x_4-2x_3 & 0 & 0 & 0 & 0 & 2y_3-2y_4 & 2y_4-2y_3 & 0 & 0 & 0 & 0 & 2z_3-2z_4 & 2z_4-2z_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2x_5-2x_6 & 2x_6-2x_5 & 0 & 0 & 0 & 0 & 2y_5-2y_6 & 2y_6-2y_5 & 0 & 0 & 0 & 0 & 2z_5-2z_6 & 2z_6-2z_5 \end{pmatrix}.$$

In order to inform the direction of our viable motion, we can take a desired movement vector and project this onto our null space. If we have multiple columns in our null space, we can alternatively choose the column of the null space that has the smallest Euclidean distance from our null space.

Therefore, in order to simulate any system containing rigid bars, we can:

1. calculate the length constraints on each bar;
2. take the derivative of the length constraints on each bar with respect to time;
3. form the Jacobian matrix, where each row of the matrix is the derivative of a constraint equation;
4. take the null space of the Jacobian, and move in the direction of one of the columns of the null space.

3.2 Python Simulator

Based on the above mathematical approach, I wrote a Python simulator consisting of approximately 500 lines of code that can take in any number of bars and a starting position, and simulate legal motions for this system. It can also simulate the motion of any n-bar connected arm, and move the end point of this arm in a desired direction. Additionally, the simulator can take in any current state of a tensegrity system, and derive the lengths of cables between bars that are needed to achieve this state.

The simulator defines a tensegrity class with struts, cables and initial positions. At each step in the simulation, the simulator calculates a viable direction for moving each strut's end point, and `matplotlib` draws the structure in this new position. In order to specify the direction of the bar movement in simpler cases, the simulator projects the average of the columns of the null space onto a desired direction vector. This can be used to create planar robot arms that follow a particular direction of motion.

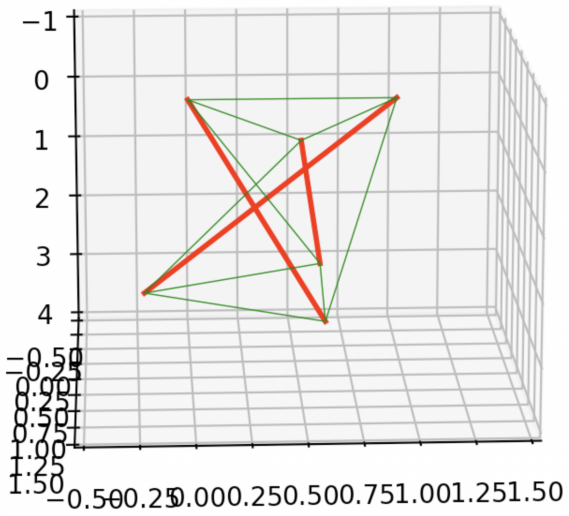

Figure 2: Screenshot of thee-bar tensegrity random simulation after ten seconds

This method has its limitations, as it assumes no constraints on the cables between bars of the system. For example, in a three-bar tensegrity module, we have constraints on the length of each of the rigid bars, but no constraints on the cables between the ends of the bars themselves. In reality, there would be maximum and minimum lengths that each actuator can take. Furthermore, this simulator does not consider external forces on the system such as gravity or friction, which must be taken into account when actuating the system.

4 Three-dimensional (3D) Triangular Prism Three-Bar Tensegrity Modular Structures

A basic and well-researched form of tensegrity structures are 3D triangular prism three-bar structures where the three bars cross each other at their midpoints. Each of these modules consists of three rigid rods and nine cables each of which connect the endpoints of the rods. In each module, six of the cables create two parallel triangles, which can be connected end-to-end to create different types of modular structures. The other three cables connect the two triangles together by stretching along the longer axis of each mod-

ule.

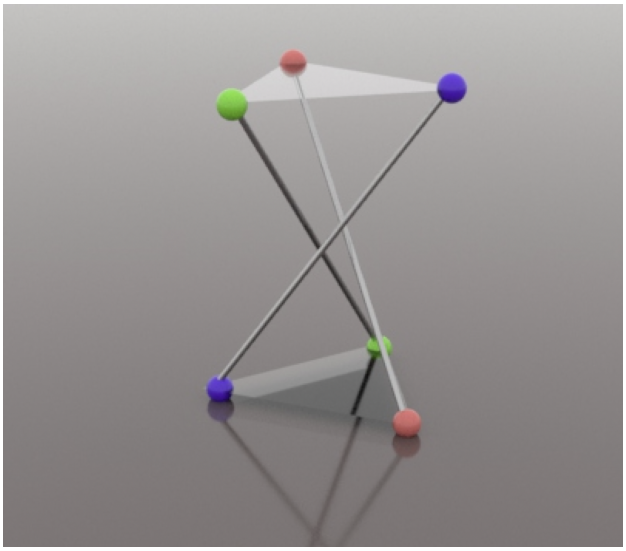

Figure 3: Maya 3D model of three-bar tensegrity structure

If the structure were completely rigid, then a three-bar tensegrity module's complete position could be determined with the (x, y) coordinates of two end points on the ground (four degrees of freedom assuming the robot is sitting on a flat plane), as a rigid triangle has four degrees of freedom. However, due to the soft nature of tensegrity robots, more degrees of freedom are introduced. We also need to consider the fact that the robot could be sitting on a tilted or bumpy piece of ground. Therefore, there is a challenge involved in determining the minimum amount of information required to fully describe the system, and the minimum amount of information required to usefully describe the system (and whether these two things are equivalent).

4.1 Construction of Three-Bar Structures

I built various three-bar physical tensegrity structures to explore their viable motions in the physical world. Each structure was composed of three struts, three elastic bands, six strings and pinholes to attach the elastic bands and string to the ends of the struts. The elastic bands are attached along the longer axis of the module, and the rigid strings are attached around the top and bottom triangles of the module.

A stepper motor was attached to the end of one strut which shortens and lengthens a single cable in this system. Through experimentation, I was able to achieve a flipping motion by shortening the actuated cable until the center of mass of the system moves over the contact pivot point of the module.

The physical models are shown in Figures 4 and 5.

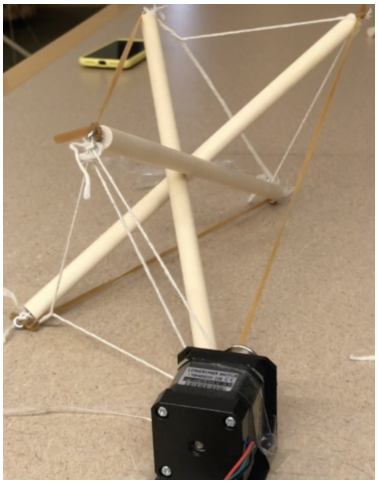

Figure 4: Actuated 3D three-bar tensegrity structure with strut length of 30 centimeters

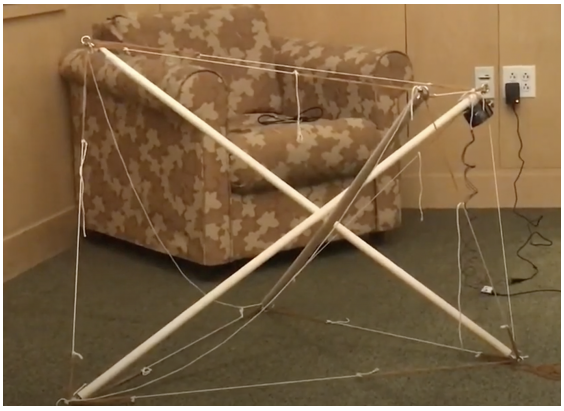

Figure 5: Actuated 3D three-bar tensegrity with strut length of 2 meters

4.2 Modular Structures

Using these three-bar structures as constituent modules, I propose a variety of three-dimensional modular structures.

We can first line up the three-bar structures end to end to create a long snake structure (Figure 5). These structures can be used to create columns or arches capable of supporting other structures or modules. Peristaltic or inchworm-like motion can be achieved along this structure to allow the system to locomote as a whole. This formation also leaves room for rolling motions as the structure is long and thin.

Figure 6: Maya model of snake tensegrity structure

We can also arrange the modules side-by-side to form a lattice structure that could potentially be used to manipulate objects along its surface through a wave-like motion. The modules can be arranged in hexagonal structures, where each hexagon consists of six modules. Then, these hexagons can be arranged into a hexagonal lattice. This structure is not well-suited to rolling, however it could potentially be used to pass objects along the top of the lattice. The structure however is very complex and there are points where six modules connect all together (at the center of each hexagon) which would require significant inter-module coordination in terms of assembly and moving together.

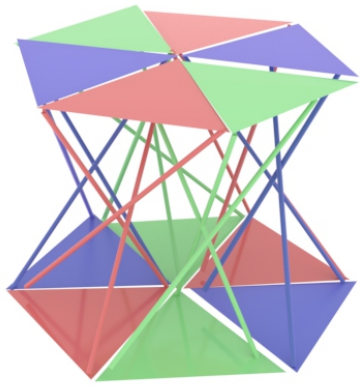

Figure 7: Maya model of lattice tensegrity structure

5 A Working Two-Dimensional Three-Bar Tensegrity System

In order to determine a more suitable modular structure for object manipulation, we designed, built and actuated a fully modular planar tensegrity system based on triangles that can be extended to a simple 3D tetrahedral design.

5.1 Design

I collaborated with Luyang Zhao on the design of this a three-bar planar tensegrity module. As shown in Figure 8, each individual module has three struts (red solid lines) connected by a revolute joint at node 0. The the position of node 3 is fixed for each module to reduce the degrees of freedom and simulate the effect of the modules sitting on a very high-friction surface. Therefore, for each module, there are four degrees of freedom (the x and y locations of node 1 and 2). We can move the system by contracting and expanding the black cable in each module. If we assume there is no gravity and that the elastic potential between nodes 1-2 and 2-3 are equal and the strut lengths of the module are the same, then we know that node 2 will move to a position such that the angles between

struts 1-2 and 2-3 are equal. Thus, without accounting for gravity, we can fully describe the position of a single module just by the length of the motorized cable.

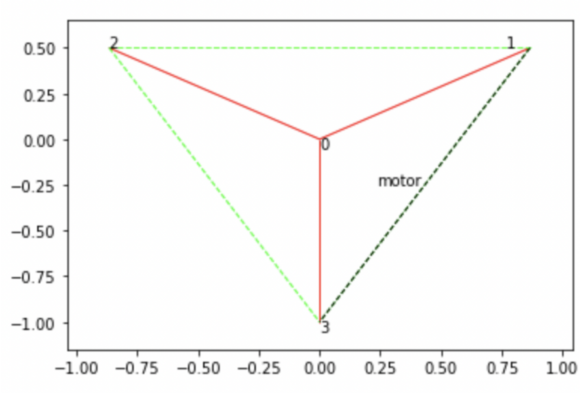

Figure 8: Design and motor locations of planar module (image courtesy of Luyang Zhao)

5.2 Basic Object Manipulation Algorithm

To achieve object manipulation, we attached each module end to end (where node 1 of each module is connected to node 2 of its right-sitting neighbor).

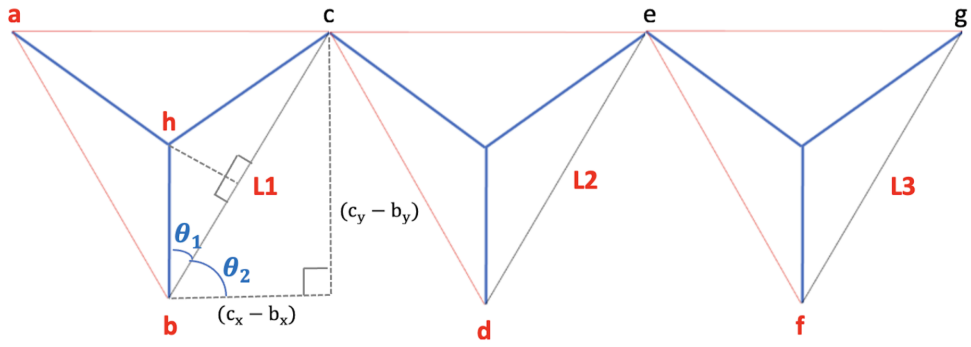

Figure 9: 3 modules

To determine the minimum number of motors needed to fully describe the positions in the system, we must analyse the degrees of freedom of this system. Assume that in Figure 9 the red values are known and all other values are all unknown.

We can derive angle θ_1 based on the known right angle triangle from h to b to halfway across the length L_1. Then:

$$\theta_2 = 90 - \theta_1. \tag{9}$$

$$c_x = b_x + L_1 \cos\theta_2. \tag{10}$$

$$c_y = b_y + L_1 \sin\theta_2. \tag{11}$$

We can similarly determine the positions of subsequent nodes e and g. Therefore to fully describe the system, the left most module needs a fixed central node and a known length from a to b which can be achieved by adding an extra motor to the module. Then the rest of the module positions can be determined with only one motor each and and an anchor at the bottom node. So the system overall has six degrees of freedom in the leftmost module node positions, plus one degree of freedom for the length of the actuator in each module. So if there are n modules, there are $6 + n$ degrees of freedom in the system.

5.3 Basic Object Manipulation Algorithm

We want to use these modules in tandem for object manipulation. In this system, object manipulation relies on utilizing gravity to pass an object along the top surface of the combined structure. The main approach for object manipulation is to create a series of one-way downwards slopes that an object can roll along in the direction of desired motion without losing net height overall or rolling backwards.

At each step in this algorithm, a downwards slope is created along two consecutive modules. The object rolls down this slope to the rightmost corner of the second module in the pair. Then, the first module re-flattens, but the object is prevented from rolling back onto the top of the first module. This process then repeats, allowing the ball to repeatedly roll to the end of the next module. The reason we only use two modules at a time in generating a slope is that we want to maintain the integrity of the structure being as close to flat as possible at any given time. If we tried instead to create broader slopes, we could run out of height in the rightmost module of the slope and end up not being

able to extend the slope any further, or placing too much external force on the system and causing it to break.

The object manipulation algorithm for this motion is

Algorithm 1 Object Manipulation via Successive Slopes

Input: list of modules M ordered in desired direction of motion
for i=0 to len(M)+1 **do**
  create a slope of 30 degrees along the top of M[i]
  create a slope of 45 degrees along the top of M[i+1]
  delay for 1 second
  create a slope of 0 degrees (flatten) along the top of M[i]
end for

For example, the sequence of commands to move a puck from the first to the fourth module is as follows:

```
module 1:  30 degree right slope
module 2:  45 degree right slope
module 1:  undo 30 degree right slope
module 2:  30 degree right slope
module 3:  45 degree right slope
module 2:  undo 30 degree right slope
module 3:  30 degree right slope
module 4:  45 degree right slope
module 3:  undo 30 degree right slope
```

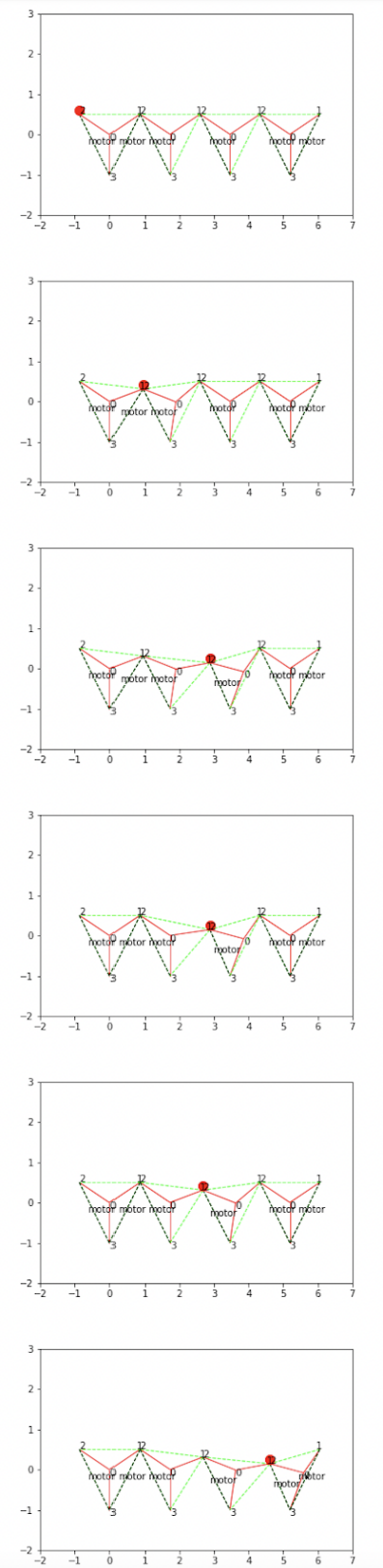

Figure 10: Several steps of the object manipulation algorithm (image courtesy of Luyang Zhao)

5.4 Curve Approximation

This basic object manipulation algorithm is effective in performing specific object manipulation tasks, but it is very dependent on the particular environment of the system. For example, if the overall tensegrity robot had an upwards slope by being placed on a hill, the angles in this object manipulation algorithm would need to be changed to become steeper for the ball to successful roll uphill. Therefore, it is necessary to develop a more general object manipulation method that could be easily adapted to different situations. To do this, we developed a continuous-motion approach inspired by biological peristaltic motion, similar to the motion of a vein pumping blood. This type of movement relies on periodic waves moving along a surface with time. We can do this by approximating a curve along the top surface of the system. These curves can then be parameterized with respect to time to create moving waves along the top of the system that could transport objects. Then, we could change the curve we are using depending on the specific task needed (for example, we could pass sine curves with varying wave heights and periods along the system).

Furthermore, curve approximation would be beneficial for creating specific shapes of structures along the top of the modular tensegrity system. For example, if we wanted to create a corrugated surface along the top of a tensegrity structure composed of these modules, we could approximate a periodic curve along the top of adjacent modules.

In this approach, we developed a way to approximate a curve along the top of the system and then retrieve the lengths of the linear actuators needed for this configuration at any point in time. These lengths are sampled at equal intervals and then smoothly interpolated between them. I developed a Python simulator that moves the endpoints of our modular tensegrities to the best possible positions for approximating a given curve while obeying the constraints between and within each module. The code (500 lines long) defines a planar tensegrity class which has instance variables for the coordinates of the endpoints of each strut in a three-bar planar tensegrity robot. The method for approximating any curve is as follows:

1. the modules are first set to default positions, where they are evenly spaced and the angle between adjacent struts is 120 degrees;

2. the curve to be approximated is defined as $f(x)$;

3. $f(x_1)$, $f(x_2)$, $f(x_3)$, $f(x_4)$ and $f(x_5)$ and are calculated, where x_1, x_2, x_3, x_4 and x_5 are the x-coordinates of the connections between modules. These values refer to the target positions to move each of the joints to;

4. constraints are defined for the lengths of each module strut, the anchor points at the bottom of each strut, and the joining ends of adjacent struts; each distance constraint is defined as a function that returns the Euclidean distance between the endpoints of a strut minus the desired Euclidean distance between these points, so when the constraint is satisfied these functions should return 0;

5. these constraints are passed into the function `scipy.optimize`, which returns the closest possible values for each joint while not violating any constraints.

In order to improve this method, one can replace the $f(x)$ values by calculating the intersection between the tangent to the curve at the midpoint between adjacent x values and the function we are approximating. Figures 10 and 11 show the results from approximating different curves overlaid onto the actual curve that is being approximated.

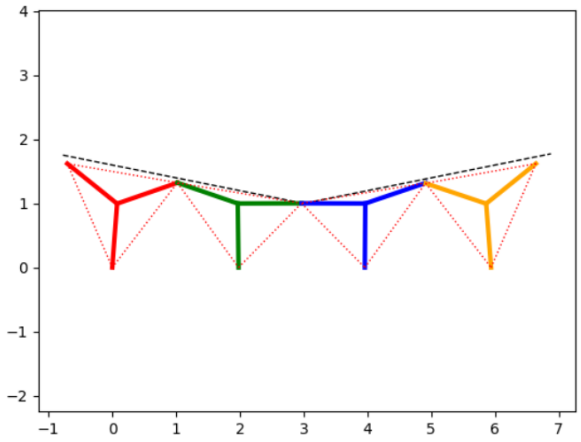

Figure 11: Approximation of $f(x) = 1 + 0.2|x - 3|$

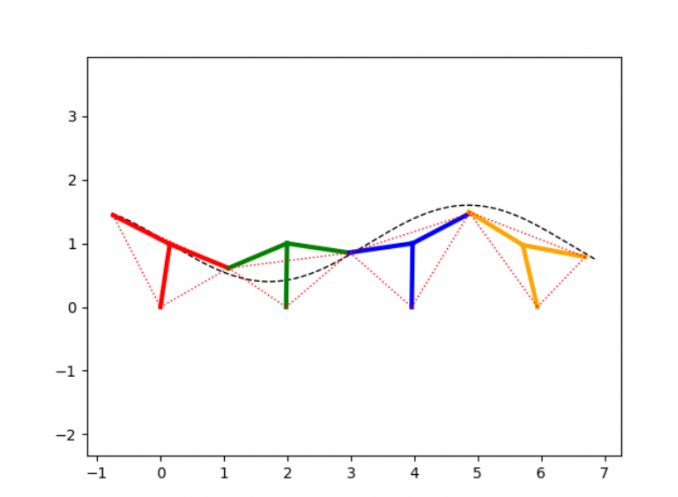

Figure 12: Approximation of $f(x) = 1 + 0.6\sin(x + 3)$

To further validate this method, I simulated how this curve changes over time while also placing circular objects in the local height minima of the system. The changing curve then pushes the objects along the top surface of the system, similarly to how a wave pushes a surfer forwards. For visualization, I animated this simulation and also plotted the system over time in a 3D space, where time is measured along the z-axis.

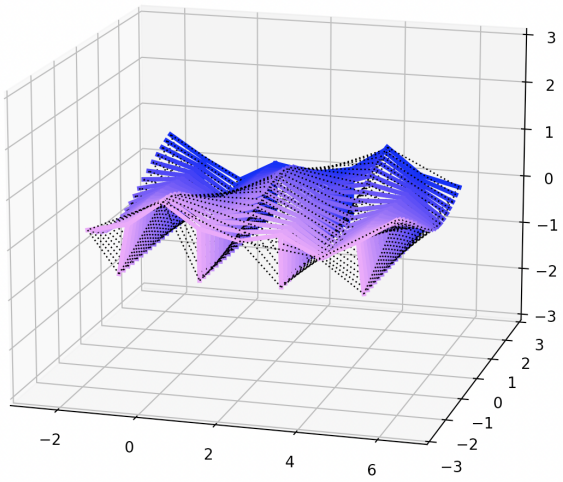

Figure 13: Approximation of $f(x) = 1 + 0.3\sin(x + t)$ visualized over time

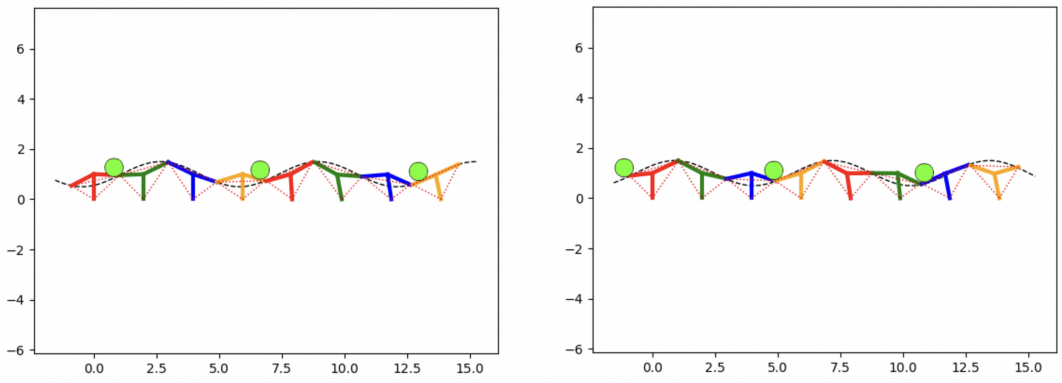

Figure 14: Pucks at two different times in curve approximation animation

5.5 Construction

Each module consists of three bars connected by a central revolute joint, with two elastic cables and one linear actuator connecting the ends of the cables. As the actuating cable becomes shorter or longer, the endpoints of the arms of the modules move to accommodate the tension from this actuator.

In order to validate and compare different approaches to object manipulation in a real environment with gravity and friction, I built and actuated a planar system containing four planar modules. Several variables needed to be considered in the design of this system, including how to support and constraint the modules, how to build a 2D system in 3D without adding any significant volume in the third dimension, how to connect the components together and how to mount and control the linear actuators in the system.

Each module is comprised of three flat struts which are hinged together in the middle with a wide bolt and cap. The struts are designed to be flat and wide to create a large contact surface at the central hinge and prevent the struts from bending forwards or backwards outside of the two dimensions of intended motion, while still allowing the struts to pivot around a central joint in 2D. The struts are 7.5 inches long to make sure the system was large enough to mount motors onto for actuation. On the endpoints of each strut there are two holes: one for connecting modules and one for rigging up each module with elastic bands. I designed these struts and bolts in the 3D modelling software Rhino

and 3D printed them. I then used these parts to assemble the skeletons of each module.

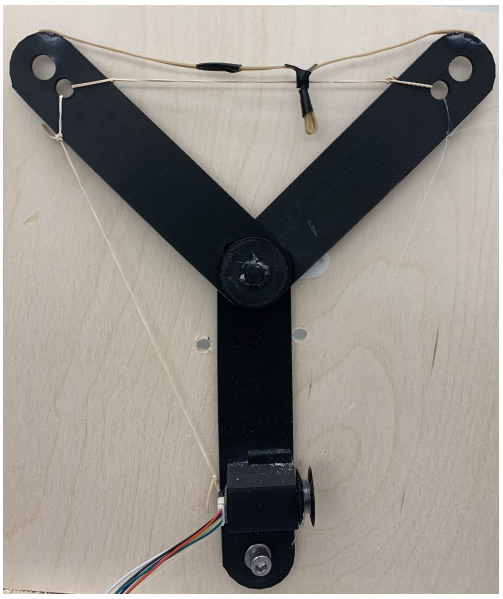

Figure 15: Single module

After the three struts were hinged together in the center, I rigged each module with two elastic bands and one non-elastic wire, which is attached to a DC motor mounted at the ends of one of the struts. These wires and bands are tied around the built into the ends of the struts. In order to mount the motors, I needed to design a 3D mount that allows for the motor to be held firmly in place by placing the spoke of the motor through a hole in the mount and attaching a spindle to the other side of the hole. This spindle then winds the wire in order to change its length and actuate each module.

In order to move the physically constructed system, there needs to be a way to shorten and lengthen the linear actuators in each module. This is done via DC motors which have spindles on the ends, allowing the wire wrapped around them to change length by winding and unwinding.

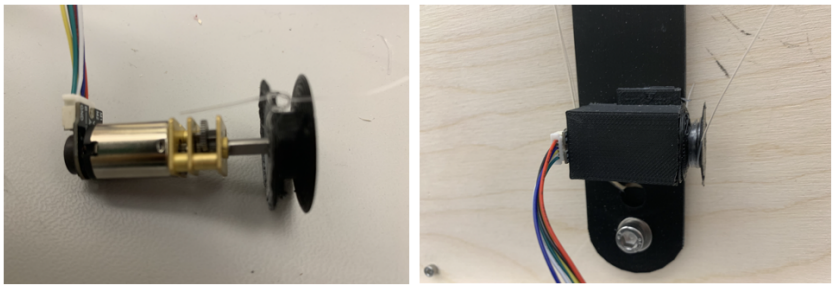

Figure 16: Motor, spindle and motor mount

These motors are then connected to an Arduino and C++ code is uploaded to the Arduino to actuate the motors using the Adafruit motor shield C++ library. I wrote a 100 line driver for these motors.

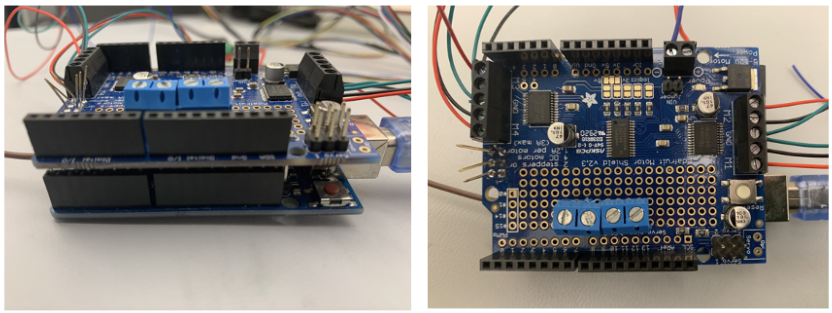

Figure 17: Arduino and Adafruit motor shield

Finally, the modules are attached to a wooden board by constraining one endpoint of each module to the board with a nut and bolt. This is to ensure the system remains planar. The modules are also attached end to end with each other using more 3D printed bolts that allow the modules to hook together. A final set of loose elastic bands is attached to the very top of the entire system to provide a surface for a puck to roll over. Then the system is tilted back 30 degrees, which allows downwards gravity to prevent the puck from falling forwards off of the system while still making the puck roll along slopes created at the top of the system.

This system was actuated by DC motors which wind and unwind the actuator cables in each module. First, the desired length sequence of the actuators are determined via

the object manipulation algorithms. Each motor is controlled by setting a speed and a time delay to rotate at this speed, so through experimentation I created a map from the desired actuator lengths to the motor speeds and delays.

In the basic object manipulation algorithm approach, eight functions are created for each module: creating a 30 degree left slope, creating a 30 degree right slope, creating a 45 degree left slope, creating a 45 degree right slope, undoing the 30 degree left slope, undoing the 30 degree right slope, undoing the 45 degree left slope and undoing the 45 degree right slope.

In the curve approximation approach, lengths of the actuated cables at different points in the moving time-parameterized curve are sampled using the curve approximation code. Then, these lengths are mapped to functions to be sent in sequence to the motors.

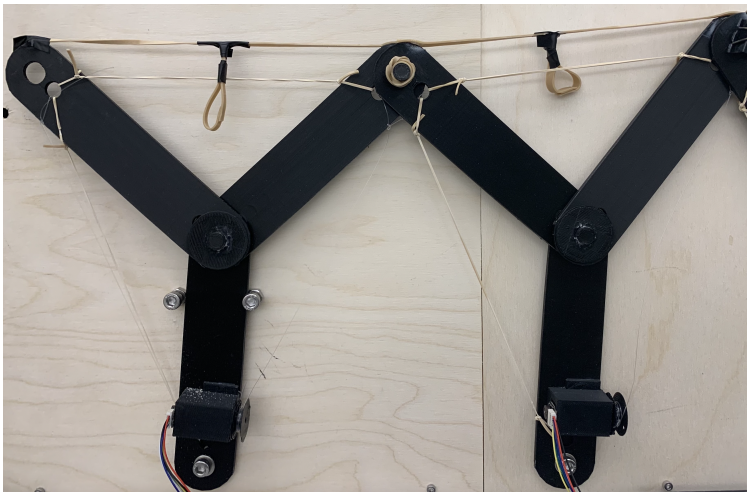

Figure 18: Two connected modules

The engineering of the system had its limitations. For example, as it is planar and the elastic skin along the stop of the system is thin, the puck will fall forwards unless the entire board is tilted backwards to keep the puck flush to the board. Furthermore, the endpoints of modules are connected by bolts rather than electromagnets, meaning the modules need to be manually attached and detached. As the system is planar, it is also possible the elastic skin falls directly below the struts of a module when the slope is very high. In this case, the puck rides along the struts of the module rather than the elastic.

However, this is inconsequential, as in this scenario the struts will also be sloped in the desired direction of motion for the puck to move correctly.

5.6 Results

In order to validate my algorithm, I carried out a series of twenty trials. In each trial I would place a puck on the leftmost module and run the Arduino code from scratch. Out of these trials, seventeen were successful, two failed from the puck falling forwards off of the board and one failed from the puck not rolling down one of the elastic bands and becoming stuck from friction. Still shots of several stages in a successful trial are shown in Figures 18-21.

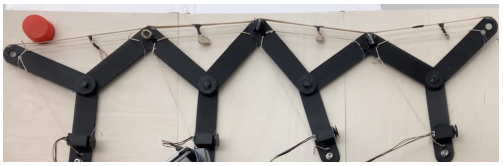

Figure 19: Stage 1

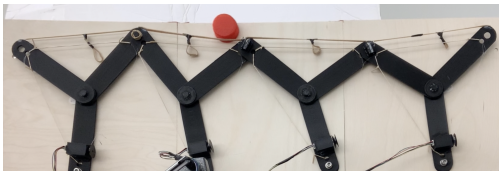

Figure 20: Stage 2

Figure 21: Stage 3

Figure 22: Stage 4

I also used this method to roll the ball to one end and back along the modules to show the bi-directionality of the method and demonstrate that it can be used to transport objects in two directions. I also created curves along the top surface of the robot based on values from the curve approximation code. One limitation of this test-bed is that there are only four modules, which is not enough to show multiple objects being transported, as at any point during the basic algorithm there would need to be a gap of at least two modules between consecutive slopes.

6 Extension to 3D Tetrahedron Module Designs

The planar tensegrity system we have designed and tested demonstrates the functionality of modular tensegrity robots in object manipulation. However, these modules are planar and need to be mounted on a board in order to stand upright, and could only be used to transport "planar" objects of limited thickness (such as the puck we have used in our design). In order to devise a usable tensegrity system for the manipulation of generic 3D objects, I collaborated with Dartmouth PhD student Luyang Zhao to design a 3D extension of the planar system. Zhao also analyzed and built a fully actuated version of this system.

In the 3D version of the modular system, each module consists of a four-bar tensegrity structure. The modules include four compressive components and six tensile components, with the struts are freely hinged at the center point of the module.

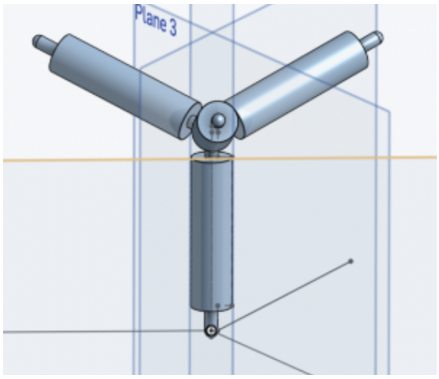

Figure 23: Single 4-bar 3D module (image courtesy of Luyang Zhao)

This structure is then duplicated and flipped and placed directly under the first tetrahedron. This creates a symmetrical module composing of two tetrahedrons.

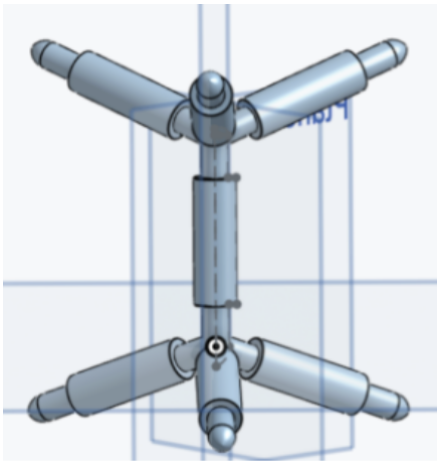

Figure 24: Symmetric design based on tetrahedral module (image courtesy of Luyang Zhao)

These modules are then placed into a lattice structure where the triangles formed by the top and bottom of the modules create two triangular tiled surfaces. These layers can then be stacked to form a tall lattice, or left as a single layer.

This entire structure of struts is printed out as a single component using soft malleable plastic, which allows the structure to flex (rather than having separate joints).The

modules are connected via magnets at the ends of the struts. Individual modules can then locomote using SMA wire, and the lattice structure as a whole can manipulate a ball across the top.

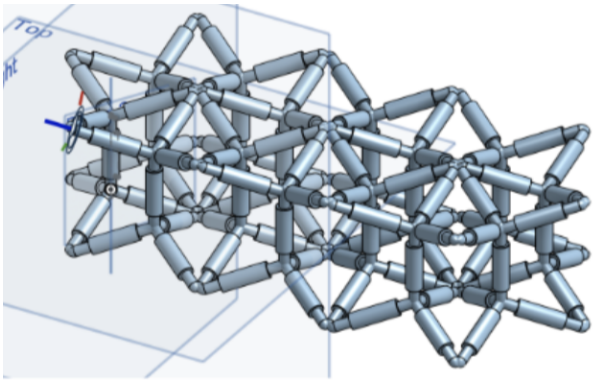

Figure 25: Tetrahedron tensegrity lattice (image courtesy of Luyang Zhao)

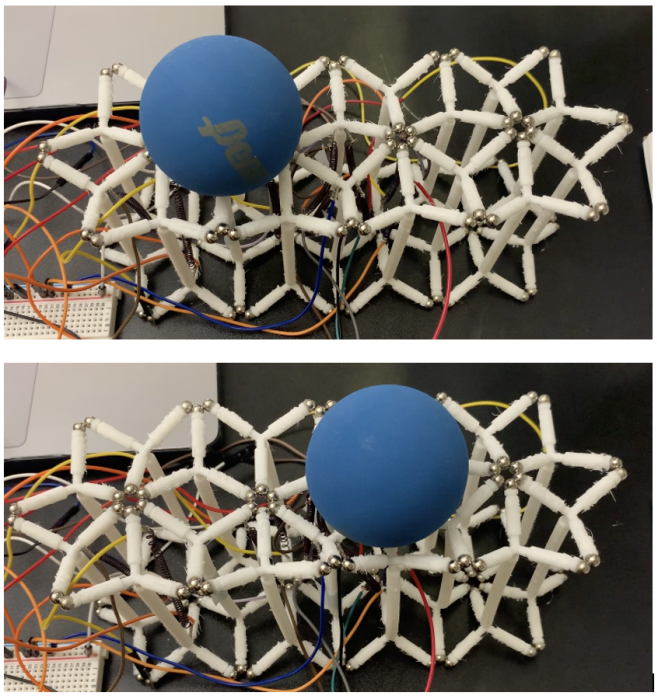

Figure 26: Two steps in object manipulation of 3D design (image courtesy of Luyang Zhao)

References

[1] ARAI, T., PAGELLO, E., PARKER, L. E., ET AL. Advances in multi-robot systems. *IEEE Transactions on robotics and automation 18*, 5 (2002), 655–661.

[2] BAKER, A., AND CRANE III, C. Analysis of three degree of freedom 6× 6 tensegrity platform. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2006), vol. 42568, pp. 31–37.

[3] BOXERBAUM, A., SHAW, K., CHIEL, H., AND QUINN, R. Continuous wave peristaltic motion in a robot. *The International Journal of Robotics Research 31*, 3 (2012), 302–318.

[4] BRUCE, J., CALUWAERTS, K., ISCEN, A., SABELHAUS, A. P., AND SUNSPIRAL, V. Design and evolution of a modular tensegrity robot platform. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), IEEE, pp. 3483–3489.

[5] BRUCE, J., SABELHAUS, A., CHEN, Y., LU, D., MORSE, K., MILAM, S., CALUWAERTS, K., AGOGINO, A., AND SUNSPIRAL, V. SUPERball: Exploring tensegrities for planetary probes. In *12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)* (2014).

[6] DONALD, B., GROSS, S., ET AL. Self-organizing microrobots. *Advanced Manufacturing Technology 29*, 7 (2008), 4–6.

[7] FURUYA, H. Concept of deployable tensegrity structures in space application. *International Journal of Space Structures 7*, 2 (1992), 143–151.

[8] HE, Q., AND POST, M. An adaptable robotic snake using a compliant actuated tensegrity structure for locomotion. In *Annual Conference Towards Autonomous Robotic Systems* (2020), Springer, pp. 70–74.

[9] LESSARD, S., CASTRO, D., ASPER, W., CHOPRA, S. D., BALTAXE-ADMONY, L. B., TEODORESCU, M., SUNSPIRAL, V., AND AGOGINO, A. A bio-inspired tensegrity manipulator with multi-dof, structurally compliant joints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), IEEE, pp. 5515–5520.

[10] LEVIN, S. M. The tensegrity-truss as a model for spine mechanics: biotensegrity. *Journal of mechanics in medicine and biology 2*, 03n04 (2002), 375–388.

[11] MAS, I., AND KITTS, C. Object manipulation using cooperative mobile multi-robot systems. In *Proceedings of the World Congress on Engineering and Computer Science*

(2012), vol. 1.

[12] MINSKY, M. Manipulator design vignettes, 1981 dspace@mit memo.

[13] MIRLETZ, B. T., PARK, I.-W., FLEMONS, T. E., AGOGINO, A. K., QUINN, R. D., AND SUNSPIRAL, V. Design and control of modular spine-like tensegrity structures. In *The 6th World Conference of the International Association for Structural Control and Monitoring (6WCSCM)* (2014), pp. 4–10.

[14] PAGITZ, M., AND TUR, J. M. Finite element based form-finding algorithm for tensegrity structures. *International Journal of Solids and Structures 46*, 17 (2009), 3235–3240.

[15] PAUL, C., VALERO-CUEVAS, F., AND LIPSON, H. Design and control of tensegrity robots for locomotion. *IEEE Transactions on Robotics 22*, 5 (2006), 944–957.

[16] ROVIRA, A. G., AND TUR, J. M. M. Control and simulation of a tensegrity-based mobile robot. *Robotics and Autonomous Systems 57*, 5 (2009), 526–535.

[17] SABELHAUS, A., BRUCE, J., CALUWAERTS, K., MANOVI, P., FIROOZI, R., DOBI, S., AGOGINO, A., AND SUNSPIRAL, V. System design and locomotion of SUPERball, an untethered tensegrity robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), IEEE, pp. 2867–2873.

[18] SKELTON, R., HELTON, J., ADHIKARI, R., PINAUD, J.-P., AND CHAN, W. An introduction to the mechanics of tensegrity structures. In *The Mechanical Systems Design Handbook*. CRC Press, 2017, pp. 315–388.

[19] SKELTON, R. E., AND DE OLIVEIRA, M. C. *Tensegrity systems*, vol. 1. Springer, 2009.

[20] SONG, P., AND KUMAR, V. A potential field based approach to multi-robot manipulation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)* (2002), vol. 2, IEEE, pp. 1217–1222.

[21] SUROVIK, D., WANG, K., VESPIGNANI, M., BRUCE, J., AND BEKRIS, K. E. Adaptive tensegrity locomotion: Controlling a compliant icosahedron with symmetry-reduced reinforcement learning. *The International Journal of Robotics Research* (2019), 0278364919859443.

[22] ZHANG, M., GENG, X., BRUCE, J., CALUWAERTS, K., VESPIGNANI, M., SUNSPIRAL, V., ABBEEL, P., AND LEVINE, S. Deep reinforcement learning for tensegrity robot locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), IEEE, pp. 634–641.