Learning Tensegrity Locomotion using Open Loop Control Signals

Atil Iscen
Oregon State University
Corvallis, Oregon 97331
iscena@onid.orst.edu

Ken Caluwaerts
Reservoir Lab, Ghent University
Ghent, Belgium
ken.caluwaerts@ugent.be

Jonathan Bruce
USRA/University of California Santa Cruz, CA 95064, USA
jbruce@soe.ucsc.edu
Adrian Agogino
UC Santa Cruz / NASA Ames Research Center, MS 269-3, Moffett Field, CA 94035, USA
adrian.k.agogino@nasa.gov
Vytas SunSpiral
SGT Inc. / NASA Ames Research Center, MS 269-3, Moffett Field, CA 94035, USA
vytas.sunspiral@nasa.gov
Kagan Tumer
Oregon State University, Corvallis, OR, 97331, USA
kagan.tumer@oregonstate.edu

Tensegrity structures offer many advantages to mobile robots while also presenting challenges that include highly non-linear interactions and oscillatory dynamics. These challenges make tensegrity robots hard to control using classical control methods. We take the model of SUPERBall, an icosahedron tensegrity robot that is under production at NASA Ames Research Center, and study the rolling locomotion using an accurate model of the SUPERBall simulated in NASA Tensegrity Robotics Toolkit. We use a distributed and learning based controls approach using coevolutionary algorithms and different types of open loop control signals. Being simple and distributed, these controllers can be readily implemented on the SUPERBall hardware without requiring sensor information and precise coordination. We first analyze signals of different complexities and frequencies. The results show that with the right types of signals and evolutionary algorithms, the robot can learn rolling locomotion. Among the learned policies, we take one of the best ones to analyze further. We make sure that the resulting behavior is within the capabilities of the hardware by looking at different aspects of the rolling gait such as lengths, tensions and energy consumption. We also discuss the correlation between different signals controlling different parts of the tensegrity robot.

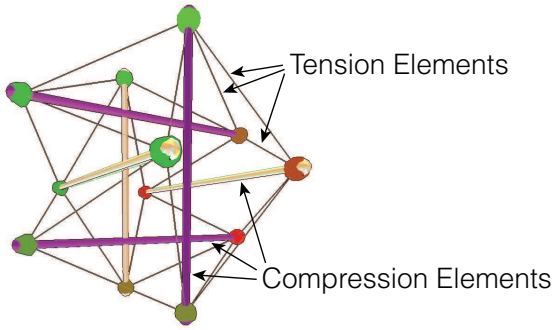


Fig. 1: An icosahedron tensegrity structure composed of 6 compression and 24 tension elements.

1 Introduction

Tensegrity robots are part of an exciting emerging field of soft-body robotics that brings many challenges from the controls perspective. The nonlinear interactions between different elements results in a challenging problem from a controls perspective. Fortunately, learning based controls combined with the distributed nature of tensegrity robots can provide a simplified controls solution to the fairly complex problem of locomotion of tensegrity robots.

Tensegrity structures are based on a simple principle: the structure is composed of pure tension and pure compression elements. Axially loaded compression elements are en-

compassed within a network of tensional elements, and thus each element experiences either pure compression or pure tension. Based on this simple principle, the structures can be arbitrarily complex and stiff by increasing the number of members and by changing their stiffness.

Since the structure does not have any bending or shear forces that must be resisted, individual elements can be extremely lightweight. Moreover, the structure is mostly composed of tension elements that are significantly lighter than the compression elements. A unique property of tensegrity structures is how they can internally distribute forces. As there are no lever arms, forces do not magnify into joints or other common points of failure. Rather, externally applied forces distribute through the structure via multiple load paths, creating a system level robustness and tolerance to forces applied from any direction. Thus tensegrity structures can be easily reoriented and are ideally suited for operation in dynamic environments where contact forces cannot always be predicted.

Tensegrities have a number of beneficial properties including:

Lightweight: Forces align axially with components and shocks distribute through the tensegrity, allowing tensegrities to be made of lightweight tubes/rods and cables/elastic lines.

Energy efficient: Through the use of elastic tensile components and dynamic gaits, efficient movement is possible.

Robust to failures: Tensegrities are naturally distributed systems and can gracefully degrade performance in the event of actuation or structural failure.

Capable of unique modes of locomotion: Tensegrities can roll, crawl, gallop, swim or flap wings depending on construction and need.

Impact tolerant and compliant: Since forces are distributed upon impact, they can fall or bump into things at moderate speed. In addition, their compliance ensures that they do minimal damage to objects they contact.

Naturally distributed control: Characteristics of force propagation in tensegrities allows effective local controllers.

The last property is the most subtle but important. In "traditional" robots, distributed controls becomes messy due to the need to communicate global state information to all the controllers with high precision, and thus often undermining the very promise of distribution. Fundamentally, this stems from the fact that a rigidly connected structure will magnify forces internally through leverage, and will accumulate force into joints. Thus, the actions of a local distributed controller can have disproportionate global consequences. These consequences can require a certain amount of global coordination and state management, undermining the value of the local controller. Tensegrity structures are different, due to the tension network, there is no leverage in the structure. Thus, forces *diffuse* through the structure, rather than accumulate in joints. As a result, actions by a local controller diffuse through the structure, integrating with all the other local controllers. While any one local controller will impact the structure globally, that impact is locality relevant and not magnified via leverage. Thus, the structure enables true distributed control, because local actions stay (predominately) local.

Despite these desirable properties, tensegrity robots have remained mostly a novelty for many years due to properties that make them hard to control with traditional control algorithms such as:

1. **Nonlinear dynamics**: A force generated on one part of the tensegrity propagates in a nonlinear way through the entire tensegrity, causing shape changes, which further change force propagations.
2. **Complex oscillatory motions**: Tensegrity robots tend to have oscillatory motions influenced by their interactions with their environment.

Fortunately the combinatorial optimization capabilities of learning based controls is a natural match to these problems. Evolutionary algorithms can learn complex control policies that maximize a performance criterion without needing to handle the oscillatory motions and distributed interactions explicitly. Cooperative Coevolutionary Algorithms (CCEAs) enable different controllers that are distributed throughout the tensegrity to learn a cooperative task such as rolling locomotion. The set of the learned policies for these controllers form a unified policy that provides locomotion of the whole robot.

Considering the positive properties of tensegrity robots discussed above, NASA is currently working on using a tensegrity robot for planetary landing and exploration missions as part of the NASA Innovative Advanced Concepts (NIAC) program [1]. For this program, SUPERBall is a close to sphere shaped, icosahedron tensegrity robot that is designed and currently under production at NASA Ames Research Center [1]. In this work, we use SUPERBall (Spherical Underactuated Planetary Exploration Robot) as our tensegrity robot model. To simulate the robot and learn the rolling locomotion, we use our highly accurate tensegrity simulator NASA Tensegrity Robotics Toolkit (NTRT).

In this work, we combine a simple set controllers and evolutionary algorithms to obtain rolling locomotion for SUPERBall hardware. Being simple and distributed, these controllers can be readily implemented on the SUPERBall hardware without requiring sensor information and precise coordination. We make sure that the controllers and the rolling behavior is within the limits of the hardware by analyzing the rolling behavior from different perspectives. We also analyze the similarities of the learned signals for different parts of the robot.

The rest of the paper is organized as follows: Section 2 introduces necessary background information and related work. Section 3 defines the control problem and open loop controls schema that we use. Section 4 introduces the learning method that we use and presents the results of 'learning to roll' for different types of signals. Section 5 analyzes the learned rolling behavior from different perspectives such as movement, energy efficiency and feasibility. Section 6 analyzes the correlation between the signals that control dif-

ferent parts of the robot while providing the rolling behavior. The paper ends with Section 7, where we discuss conclusions and future research directions.

2 Background and Previous Work

Tensegrity structures are a fairly modern concept, having been initially explored in the 1960's by Buckminster Fuller [2] and the artist Kenneth Snelson [3]. The word tensegrity is formed using the words 'tensional integrity'. The structure has an internal balance while the tensile elements encounter constant tension. The early research on tensegrities was first concentrated on the design and analysis of static structures [4–6]. The tensegrity principle was used in big structures such as tower or bridges as well as small structures such as toys or furniture [7].

The concept of tensegrity is also being discovered in biological systems from individual cells to mamalian physiology [8, 9]. Emerging biomechanical theories are shifting foces from bone-centric models to fascia-centric models, where fascia is the connective tissues (muscles, ligaments, tendons, etc.). In the "bio-tensegrity" model, bones are under compression, and continuous network of fascia acts as the primary load path for the body. Inspired by this, since the cables of the robot are the tensional elements that can change length we use the term 'muscle' for the rest of the paper.

Research into control of tensegrity structures was initiated in the mid-1990's, with initial efforts at formalizing the dynamics of tensegrity structures only recently emerging [4, 10, 11]. The very properties that make tensegrities ideal for physical interaction with the environment (compliance, multi-path load distribution, nonlinear dynamics, etc.) also present significant challenges to traditional control approaches.

For the first few decades, the majority of tensegrity related research was concerned with form-finding techniques [12–18]. The problem of finding the correct configuration to deform a tensegrity robot to a specific shape is already a hard research problem due to the nature of the structure. To solve this problem, there are both algebraic studies as well as evolutionary algorithms. There have also been studies to generate different tensegrity structures using evolutionary algorithms.

In terms of locomotion, tensegrity robots are capable of providing different forms of locomotion due to the diversity of the shapes of the structures [19]. As an example, there are multiple studies that contain robots with patterns that form snake shaped robot and locomotion based on the movement of snake [20]. There are locomotion studies that use simple irregular tensegrities with 3 or 4 struts [21, 22]. Here, we are interested in a icosahedron tensegrity robots as shown in Figures 1 and 2. Icosahedron robots contain 6 struts with equal length and 24 muscles that connect the end of the struts.

The icosahedron tensegrity provides additional advantages such as rolling locomotion and deployability. The structure is the simplest tensegrity to provide an overall shape close to a sphere. It can handle external forces and impact with the ground easily by deforming. As expected, the impact is diffused through the network of tensional elements. Moreover, the structure is easily collapsible to a star pattern by loosening tensional elements. These properties make icosahedron tensegrity a great option as landing and explotary device for space missions. SunSpiral et al explains the usage of an icosahedron tensegrity robot both as an Entry, Descent and Landing (EDL) instrument and also as an mobility device for a mission to Titan [23]. This idea of development and locomotion of an icosahedron tensegrity robot as a space exploration device is part of a project funded by NASA Innovative Advanced Concepts (NIAC) program [1].

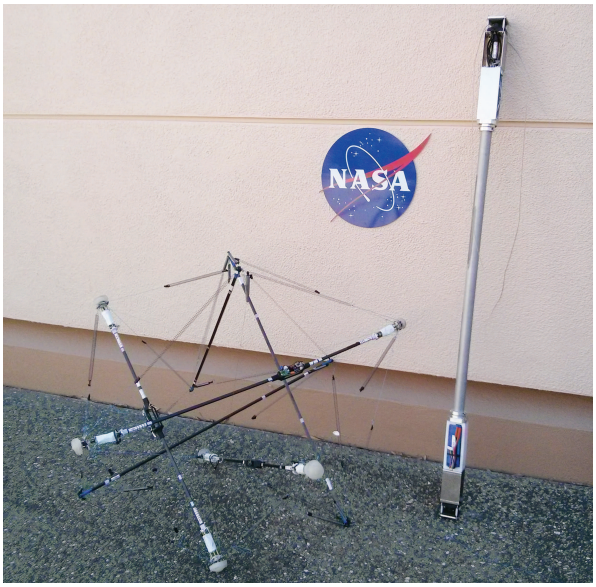


Fig. 2: ReCTeR (left), a tensegrity robot and one strut of the modular SUPERBall (right) that is under development at NASA Ames Research Center.

The literature contains few studies about active control of icosahedron tensegrities. Shibata et al analyzes different surface patterns to roll [24–26]. Rieffel at al uses vibration frequencies for locomotion without rolling [27]. Not only icosahedron tensegrities, but also the whole field of active control of tensegrities is still fairly new. A recent review [28] shows that there are still many open problems in actively controlling tensegrities.

There are few hardware implementations for tensegrity robots. Koizumi et al. use a tethered icosahedron robot with pneumatic actuators to analyze base patterns for low energy rolling [26]. Rieffel et al. uses an icosahedron robot with vibrational motors analyzing forward motion with vibration [27]. Suitable for rolling locomotion, ReCTeR is the closest to the ideal icosahedron robot [29]. It is untethered, has six motors that control the lengths of six muscles. The robot is composed of a passive shell with 24 muscles. These 24 muscles are essential for the icosahedron. In ReCTeR, these 24 muscles are passive but the structure has 6 additional active muscles that change their length for deformation and locomotion.

SUPERBall is designed in a modular way so that each

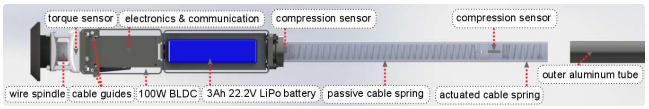


Fig. 3: Cross section of the end of the rod for the current design of the SUPERBall.

strut has 2 motors controlling two active muscles, and the struts can be used in different tensegrity robots other than icosahedron. Overall, the next prototype of SUPERBall will have an active outer shell with 12 active and 12 passive muscles, and it will be able to handle strong collisions and torque requirements for rolling locomotion in different types of terrains. In this paper, we analyze the case for 24 active muscles to use the full capabilities of a icosahedron tensegrity, but reducing the number of the controllers is part of our future work.

2.1 SUPERBall

SUPERball (Spherical Underactuated Planetary Exploration Robot) is a tensegrity icosahedron robot currently under development at the NASA Ames Research Center (Figure 3). Its main design goal is to be a more capable robot than a prior prototype called *ReCTeR*, to provide more reliable sensor equipment and to handle rougher environments.

ReCTeR is underactuated and can roll by actuating six tensile elements running through the center of the robot. A passive tensegrity icosahedron (24 tensile elements) forms ReCTeR's outer shell, the six actuated members connect opposite sides of the shell. Hence, only 20% of ReCTeRs tensile members are actuated.

For SUPERball, the goal is to have the possibility to actuate all tensile elements. The current prototype is going to have 12 actuated tensile elements out of 24, but considering our future goal, this paper studies the fully actuated robot. The main rationale for full actuation is that we aim to explore various control strategies (e.g. the one presented in this work) without hardware limitations and have precise manipulation capabilities (e.g. for end effector or payload positioning).

With a total mass of $1.1kg$ (batteries included), ReCTeR cannot carry any scientific payload without a significant impact on the robot dynamics. This is another reason to develop a new platform. SUPERball's mass with 50% actuated tensile members is around $15kg$. This larger mass will allow us to explore the behavior of tensegrity robots with a small payload suspended by tensile elements in the center of the robot [23]. Another important improvement is SUPERball's high power-to-weight ratio. ReCTeR's small brushed DC motors are often maxed out ($25W/kg$), while SUPERball has significant headroom ($> 80W/kg$, final design $> 100W/kg$). This allows for dynamic motion even in energetically suboptimal regimes.

Typical spring tensions are $5N$ to $20N$ for ReCTeR, while the average tension in SUPERball's current configuration is $50N$. The SUPERball hardware is designed to handle tensile forces up to $500N$. One interesting aspect of the tensegrity icosahedron configuration is that the design can conveniently be scaled up or down. Forces scale approximately linear as a function of the robot's mass. For practical purposes, SUPERball is deployed in its minimum diameter configuration (struts of approx. $1.5m$ in length).

Increasing the diameter of the robot does not significantly increase its total mass, because lightweight hollow aluminum tubes are used to connect the end caps. More precisely, we are using $35mm$ diameter tubes with a $1.25mm$ wall thickness. These tubes currently account for 15% of the robot's mass. Tripling the robot's diameter ($4.5m$ struts) with the same end caps, would only increase this fraction to 41% (not considering buckling). This is an advantage over an airbag design, for which the surface area and thus its mass scales quadratically as a function of the diameter. It can easily be seen that the payload volume scales cubically as a function of the strut length. This trivially defeats the square-cube law, because the density of the structure drops as one increases the strut length.

To simplify our designs and to allow us to explore various morphologies, SUPERball is a highly modular platform. Its basic element is the end cap, each of which is independent. Every end cap provides actuation, battery power, various sensors and wireless communication. In the current design, each SUPERball end cap houses a single $100W$ brushless DC motor and approx. $70Wh$ of battery power. Thus, we can currently control twelve tensile members in the icosahedron configuration (50%). The sensory equipment of each end cap consists of 2 tensile force transducers, a torque sensor on the motor and a 9 degree of freedom inertial measurement unit.

To make the robot compliant, SUPERball has compression springs inside the tube connecting the end caps. The tensile elements are cable-spring assemblies. Each cable runs between two end caps on different struts. The cable runs from the opposing end cap to the spring end cap, passes through a cable housing assembly inside the spring end cap, and connects to a compression spring located within the spring end cap's tube. The opposing end cap of the cable connects to a motor in the case of an actuated tensile member or is simply fixed to the opposing end cap for passive elements. The motor assembly is a wire spindle that winds or unwinds the cable to change the rest length of the tensile member.

Passive and actuated cable-spring assemblies behave as linear springs with a given (possibly varying) rest length. This can be implemented in hardware in various ways. Active tensile elements are actuated by changing their rest length and we will assume constant velocity motors in this paper. While not entirely accurate from a hardware perspective, we will show in this work that the required motor power for our controllers is significantly below the motors' power rating, which validates this simplification.

2.2 Nasa Tensegrity Robotics Toolkit

Simulation of Tensegrity robots is critical for application of intelligent controls. Simulation provides multiple advantages such as being able train controllers in a faster than training on actual hardware, or figuring out design requirements for the hardware. In our paper, we use NTRT that is developed on top of the Bullet Physics Engine [30]. Bullet Physics Engine is a discrete timestep, iterative physics simulator that handles collisions and interactions between different types of objects [31]. Bullet Physics Engine is successfully used in many games and it is shown to be capable of simulating interaction of massive amounts of objects.

NTRT simulator relies on Bullet Physics Engine to handle movement and collisions of rigid body objects. On the other hand, to simulate the tensional elements, we needed precise elastic components that we can change their length to simulate active controls of the robot. For this purpose, we implemented our own tensional elements that apply tension according to their stretch. Through the rest of the paper, these elements are called 'Muscle'. The name 'muscle' is inspired from the biotensegrity studies that describe the bones as compressional elements and the muscles as tensional elements.

A muscle is attached to two different rigid bodies from specific points. We assume that muscles are abstract elements that apply force to these two rigid bodies according to their tensions. They have basic properties such as rest length (length without stretch) and elasticity coefficient. The tension of a muscle is computed by looking at the current distance between two attachment points and the rest length of the muscle. For simulation purposes, we assume that the muscles have negligeable weight compared to the rest of the structure. Moreover, we also ignore the interactions of the muscles with the rest of the environment. These two assumptions are understandable because the tensional elements stay inside the structure and they do not interact with other objects except extreme deformations. Also, the weight of the physical cables vs the weight of the rest of the robot is extremely low.

The active controls of the muscles are done by changing the rest length of the muscle using a constant speed (motor speed). This is modeled after the motors that pull the elastic cables to increase the tension of a specific elastic component.

NTRT is previously validated with a physical tensegrity robot (ReCTeR) and precise motion capture. The results showed that the simulator can correctly simulate tensegrity structures with a small margin of error (less than 1 %) for static structure with active controls and a semi static structure that moves step by step.

3 Controls Problem

Controlling a tensegrity robot brings multiple challenges such as distributed controls, nonlinear interactions between components and handling difficult to model dynamics such as oscillations. For this reason traditional centralized controllers and centralized designs are not a good match for a tensegrity robot. In contrast we present a decidedly distributed approach for controlling a tensegrity robot. On the hardware side, the core of our design is an independently controllable rod containing two independent "end-cap" controllers on each side of the rod. This model naturally matches the distributed yet holistic nature of a tensegrity. The controllers act independently of each other but they interact through the system, where changing lengths of one of the muscles affects the whole structure in a non-linear way. This behavior can be seen in Figure 4. When we pull only one of the muscles (muscle 13), all the muscles change their length while some of them get shorter and some of them get longer.

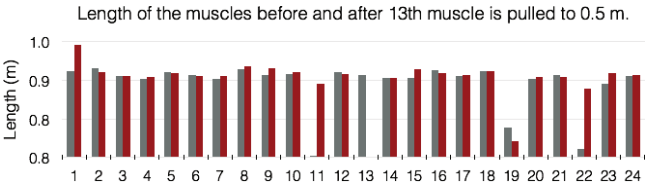


Fig. 4: Change in length of the muscles when one of them (13rd) is pulled to 0.5 meters while other muscles keep the same rest length as before. While the robot is at the exact same orientation, the actual lengths of the muscles change in a non-linear way. Some of the muscles shorten due to the tension introduced by muscle 13, some of the muscles relax.

To facilitate distributed assembly, the controllers communicate via wifi wireless network. This design allows for simplified construction, and reduces cabling problems that could arise if the tensegrity robot needs to roll through adverse conditions. This design is not only distributed, but modular. For a simple six-bar tensegrity, we can simply assemble six identical rods to form the tensegrity robot. In addition for more complex designs additional rods can be used without changing the design of the rods themselves.

Our next challenge is to control a set of assembled rods into a high-performance tensegrity robot. To do this, we need controls that are able to work in a distributed control environment, and also work when wireless communication may not be high-bandwidth or reliable. To overcome this problem, we use distributed controls and distributed learning, where each controller learns its policy, but the overall behavior requires coordination of these controllers to make the tensegrity robot move. The setup described is a coordination learning problem where we have independent learners working towards a shared goal. The details of the learning distributed controls for this setup is described at the Section 4.

An additional control challenge is how to handle the physical hardware limitations of each actuation system. Ideally we would like our controller to be able to simply dictate the actual lengths of each muscle it is responsible for. However, due to the overall tension caused by the rest of the structure, the controllers can only provide the rest length of the muscles. Since the muscles are flexible, the actual lengths change out of the control of the controller.

In addition, the hardware limitations also play an important role when tensions get higher. Since all the motors in the robot pull against each other, it is easily possible to reach tensions that the motors cannot handle. After reaching

a certain threshold of tension, the motors cannot pull the cables anymore. Moreover, since the rest of the structure can force the cable more from the other hand, there is a chance that higher force can be applied causing the motors to loose some of the cable stored in the pulley. In addition, the cables are assumed to be shorter than 1.1 meters. If the motors still cannot pull the muscle (because tension is at its limit) and the length still reaches this amount, the simulation is stopped and the policy is considered as not feasible.

To stay within the bounds of the physical hardware, we simulate the motors with high level controllers that has constant speed (0.2 m/s) while the tensions stay within reasonable limits. Indeed, the physical motors on the SUPERBall can pull with 0.5 m/s within the tension range that we are dealing with, but we select a 0.2 m/s both to not to push the limits and also to lower power consumption. While the motors move with constant velocity, the controllers dictate preferred positions for the motors. Dictating preferred position is exactly same as dictating preferred rest length if the cables do not slip. Every timestep, the motors pull or release their cables with constant speed to get closer to their goal. While staying in reasonable tensions, this setup is feasible on the real robot. The assumption is that there is an intermediate controller layer that regulates the voltage vs torque to provide constant speed rotation.

The overall goal of the controller is to have the tensegrity robot roll smoothly within the limitations of the actuation and communications hardware. To accomplish this, we use a periodic open loop controller with parameters that are set by an evolutionary algorithm (note that we have also performed research on closed loop systems, but due to sensor feedback difficulties and overall increased complexity, we are focusing on open loop controllers in this paper). During rolling locomotion, the robot (and the controllers) will repeat the same motion (one full revolution) over and over. 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.

Let's assume that the periodicity of the signal is t and we represent the signal as $F(x)$, x being time within interval $[0,t]$. There are many possible ways to represent this control function. For instance a natural choice would be a sine wave, or a series of overlapping waves to form more complex control policies. To reduce complexity, in this paper we use an even simpler control model: we break down each control interval into sub-intervals and assign different preferred rest lengths for each sub-interval. Considering the limited velocity of the motors, the motor will slowly move to reach these selected points during those sub-intervals. The control model is essentially a set of overlapping square waves. As an example, we can divide one period to 2 sub-intervals, where the motor will have a preferred length of $y1$ for the first half of the signal, and $y2$ for the second half of the signal. With the motor moving towards $y1$ and $y2$, the resulting signal will be similar to the Figure 5.

To generate a signal, the only parameters needed are number of sub-intervals and rest length values for each sub

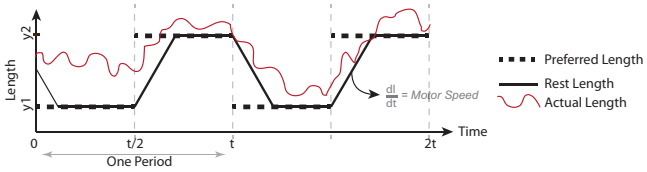


Fig. 5: An example signal with 2 sub-intervals with preferred lengths of y1 and y2 and periodicity t.

interval. For the specific example given in Figure 5, the number of subintervals is 2 and y1 and y2 are the values of preferred rest lengths for those intervals. The example given in the Figure 5 is a simple signal, and while number of subintervals is low, the complexity of the signals that can be generated is limited. On the other hand, the complexity of the possible signals increases with the number of sub-intervals. Due to this, from now on, we will refer to this parameter as the complexity degree (n) of the signal. Depending on the complexity degree and the values of $y_1, y_2, ...y_n$, the shape of the signal can change between typical trapezoid, zigzags, stairs or combination of those.

To summarize, the complexity of the signal depends on n and each controller has n number of inputs depending on the complexity selected. The rest lengths of the signal follows this signal, and actual lengths of the muscles change according to other muscles and interaction of the robot with the environment. Each controller has a separate signal and it controls only one of the motors. 24 motors controls 24 muscles independently but affecting each other with the common goal of rolling locomotion.

4 Learning to Roll

While the control parameters to generate the signal are straightforward, the interaction between these signals to reach a rolling behavior is highly complex. As explained before, the nonlinear and oscilattory nature of the problem makes the tensegrity hard to control with classical control methods. The consequences of specific signal combinations can be simulated, but finding the correct signal parameters for a specific behavior is not possible. In this section we explore how we can use the simulation combined with a fitness evaluation to implement an evolutionary algorithm that can evolve a set of control parameters that leads to the desired behavior.

Evolutionary algorithms are a family of biologically inspired learning methods, where new candidate solutions for a problem are generated, evaluated and eliminated repeatedly [32]. Evolutionary algorithms consists of the cycle of forming new members, assigning fitness and selecting the most fit members. This cycle is called a generation and it is repeated until desired behavior is obtained.

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. In the rolling tensegrity problem, we measure the performance as the distance covered in 60 seconds. Formally, the evaluation

is defined as:

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

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. Note that the decomposition of the distance function d is not readily obtainable in closed form. Instead it must be computed from observing simulations or measured from a physical implementation. Also note that our evaluation does not explicitly take any behavior into account besides the distance moved (final position - initial position). Tensegrities can exhibit many different gaits, ranging from hopping to rolling, and many different paths, ranging from spirals to straight lines. However, tensegrities that maximize final vs initial position tend to roll towards one direction. Deviations from this pattern tend to hurt performance.

Algorithm 1: Cooperative coevolutionary Algorithm with Historical Average

```
Data: Population of n elements for each agent
for i=1..k do
    randomteam ← ∅ ;
    forall the Populations do
        random_team ← random_agent;
    end
    score = evaluate(random_team) ;
    forall the agents ∈ 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
```

The problem that we are working on is distributed by nature. The controllers are independent and there is not a centralized mechanism. To work on such problems, cooperative coevolutionary algorithms (CCEA) is a class of algorithms that extends evolutionary algorithms. Instead one particular solution for the problem, there are multiple agents. Multiple agents have to coordinate to solve a problem or to optimize an outcome [33]. Each agent has a separate population that evolves independently. On the other hand, to evaluate the members of these separate populations, the agents have to form a team for fitness assignment, because the task needs

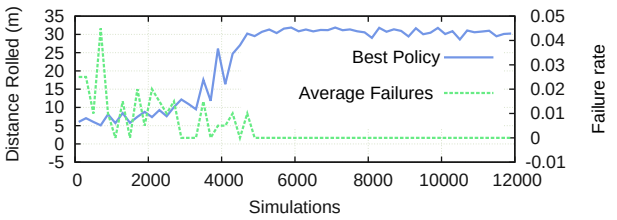


Fig. 6: The performance of the robots during the learning session for signals of complexity 5 and period of 4 seconds. As a side result, the percentage of the policies that were failed to stay in reasonable limits are shown with the second line.

cooperation of the agents to maximize the rolling behavior. At each experiment 1 member from each population are chosen randomly to form a team. This set of policies form the team for that experiment. In our problem, we have 24 agents which means 24 populations.

Each agent optimizes n parameters while they are judged based on their performances within different teams formed during evaluation phase. At each evaluation phase, each member is evaluated multiple times. To move on to the elimination phase, each member has to be assigned a single fitness number using these values. The literature contains multiple approaches to this problem such as taking the maximum score (leniency), testing each member with best team so far (hall of fame) or average score [33–35]

For this study, we used the historical average method that we previously developed and tested with earlier versions of NTRT to obtain rolling behavior using sine wave signals (REF). In historical average, each member receives its fitness according to the average of their performances, moreover, 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 overall cooperative coevolutionary algorithm with historical average fitness assignment can be found in Algorithm 1.

First, we show an example learning session using signals with complexity (n) of 5 and period (t) of 4 seconds. Figure 6 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 CCEA is possible. In Section 3 we discussed when a policy is labeled as unfeasible during learning. The second line at the same Figure (Figure 6) shows the rate of unfeasible policies that are tried while learning to roll. While converging to rolling locomotion, unfeasible policies drop to 0. This shows that the learned policy lies within reasonable lengths and tensions, moreover it is also far from the limits since small mutations that are tried during evolution are also feasible.

Considering that the robot has a shape that is similar to a sphere with a diameter of 1.5 meters, rolling 32 meters means approximately 7 revolutions in a minute. This results in 8

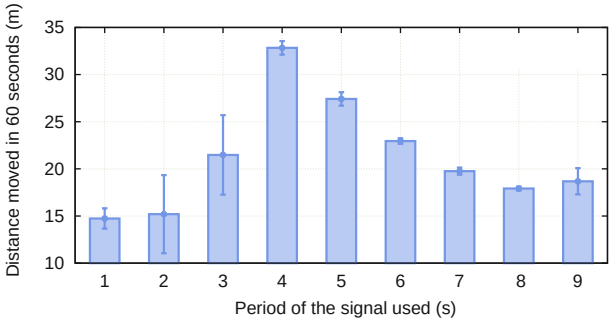


Fig. 7: The performance of the converged policies after learning for signals with periods of different lengths, while the complexity is fixed to 5 points. The best performance is reached with signals that are repeated every 4 seconds. Signals with longer periods have a decreasing performance proportional to the inverse of the periodicity.

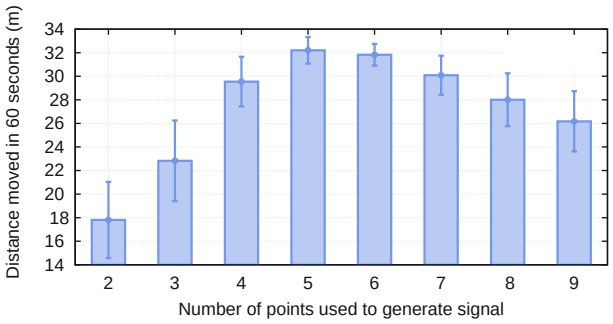


Fig. 8: The performance of the converged policies after learning for signals with different complexity levels, while the periodicity is fixed to 4 seconds. The best performance is reached with signals that use 5 points. Less complex signals cannot generate rolling locomotion, and more complex signals are hard to learn.

seconds per revolution. Considering that we selected 4 seconds as periodicity of the signal, the learned signals provide half revolution, and applying same signals also results in the other half of the one complete revolution. This supports the reasoning behind selecting periodic signals to obtain rolling locomotion as a periodic movement of the robot.

The first set of experiments illustrated when n and t are selected as 5 and 4 (4 seconds with complexity of 5). Next, we investigate the results of learning using signals with different complexity and periodicity. Figure 7 shows the converged behaviors when we fix n to 5 and learn using variable t. Note that the signal used for different values of t is not the same. The signals are learned from scratch for each value of t.

Clearly, the peak is when the signals have periods of 4 seconds (frequency of 0.25 Hz). When we shorten the period below 4 seconds, the robot cannot learn to roll. One can think that providing the same signal with a higher frequency can provide the same rolling behavior, but when the tensegrity deforms to start rolling with a higher speed, the contact forces from the ground and the reaction of the structure changes completely. When the increase the periodicity to longer than 4 seconds, the frequency drops and the performance gradually decreases as expected. Moreover, the rolled distance is linearly proportional to the frequency. For the values of 4 to 8 seconds, the performance divided by the frequency give the same value ($\frac{33}{1/4} \simeq \frac{27}{1/5} \simeq \frac{22.5}{1/6} \simeq \frac{19}{1/7} \simeq 132$).

Next, we investigate the effects of different signal complexity level to the learning. The period is fixed at 4 seconds, because it gave the best score combined with the complexity of 5 in previous set of experiments (Figure 7). We started with the complexity of 2, where the signal is simplest possible. It alters between one high value for the first 2 seconds, and one low value for the last 2 seconds. We increase the complexity up to 9 points. The result is illustrated at Figure 8. The first conclusion is that signals with complexity 2 cannot succeed learning to roll, but the performance increases with higher complexity. Clearly the controllers need more complex signals to provide rolling locomotion. The peak performance is reached at complexity 5, where the preferred length alters between 5 different points during 5 intervals of 0.8 seconds each.

The second conclusion of this experiment is seen when the complexity increases even further. The learned behavior gradually decreases and error bars show that statistical significance goes down. The reason behind this behavior is that the parameters to learn increase linearly, and the problem to learn becomes linearly harder for each controller. Since all the controllers learn simultaneously while interacting with each other, overall hardness of the problem is increased even further. The error bars show that in more complex problems, some statistical tests reach good results while some of them fail completely due to the hardness of the problem.

5 Analyzing the Rolling Behavior

In previous section, we showed that learning to roll for a tensegrity robot succeeds with signals of right periodicity and complexity. In this section, we look at the learned behavior and analyze the feasibility of the behavior, lengths and tensions during rolling, converged signals and robustness of the behavior.

As a sample learning behavior, we select one of the learned behaviors with periods of 4 seconds and complexity of 5. The learning process for this behavior was illustrated in Section 4 and Figure 6. For each simulation, the robot tests different policies for 60 seconds, and the distance moved is marked as the score. The policies that are tested are updated according to the Cooperative Coevolutionary Algorithm with Historical Average (Algorithm 1). For this particular experiment, the robots reach the performance of rolling 32 meters around 5000 simulations. As a side result, the percentage of failed policies (due to high tension reaches 0.

First, we take this learned behavior and look at the learned policy. Figure 9 shows the intervals that each muscle's length lies within. The muscles' lengths vary from 0.7 m to 1.1 m. While some of the muscles such as 1 and 21 have minimal change in length, some of them (such as 13 and 23) changes bigger intervals. Another remark is that, the

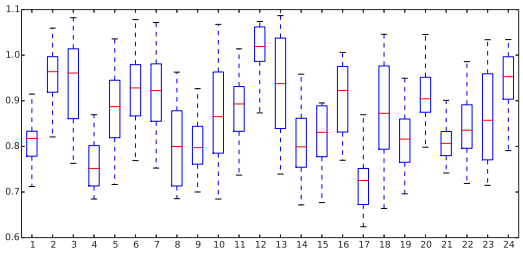


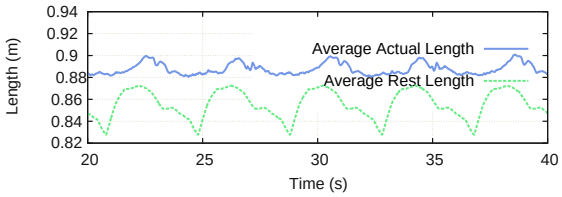
Fig. 9: A sample learned policy for 24 motors is illustrated. For each signal, the red line at the center shows the mean of the signal and the box and the dashed lines show the interval that the signal lies in.

mean of the signal (that is noted by the red horizontal line) is not necessarily at the center of the interval that the signal lies in. This is one difference that more complex signals provide. For example, the length of the muscle number 1 is mostly around 0.82 m, but it reaches as low as 0.7 once in a while.

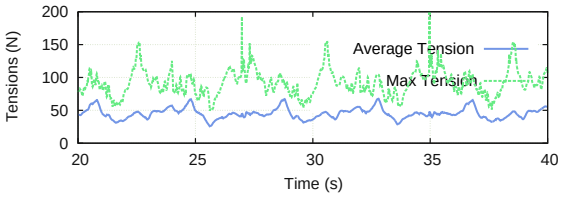
One way to analyze rolling behavior is looking at the average lengths and tensions of the muscles in addition to the potential and kinetic energy of the structure. First, we look at how the actual lengths of the muscles change compared to the signals provided. Figure 10a shows the average rest length of the muscles (signal provided) and the average actual length of the muscles. The area between the two lines shows the stretch of the muscles due to the tension. The most interesting the fact about this graph is the difference of frequencies for two lines. Although the signals that are provided to the muscles repeat themselves every 4 seconds, the actual lengths repeat themselves every 8 seconds. This supports our previous conclusion about using signals that has periodicity of 4 seconds can conclude in revolutions that take 8 seconds. The first half of the roll and second half of the roll use same signal, but the ground interactions make the actual lengths differ.

During the rolling, the average and maximum tensions of the muscles are illustrated in Figure 10b. The average tension is low, stays around 60 N. The second line shows the the tension of the muscle with longest stretch at each particular time of the simulation. The value goes up to 200N staying in values that our hardware design can handle. The maximum tension graph also repeats itself every 8 seconds as expected.

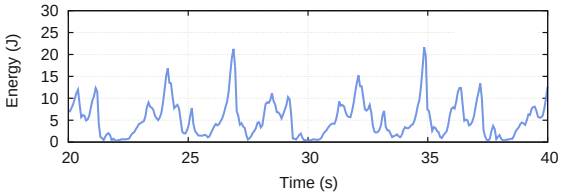
When we observed the gait learned using our simulator, we see the rolling locomotion does not have a constant speed. Instead, it slows down and speeds up periodically during each revolution. To illustrate this behavior, we look at the total kinetic energy of all the rods over time in Figure 10c. If we take the interval between two peak points (when t=27 and t=35), the kinetic energy stays at zero for 1 second around t=30s. Moreover, the repetitive acceleration and deceleration can be clearly seen. This behavior creates an inefficiency in terms of energy for the gait. There are two main reasons for this behavior: First, the learning algorithm only optimizes the distance rolled, not the energy spend during learning. Optimizing more than one criteria is actually part



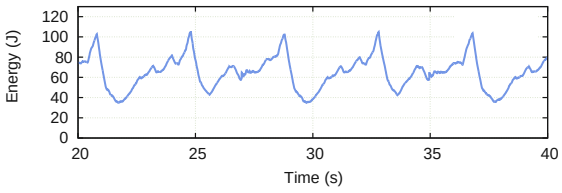
(a) Average Rest Lengths and Actual Lengths of The Muscles



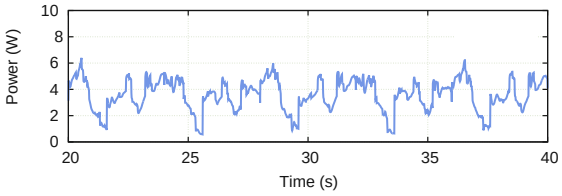
(b) Average and Maximum Tensions of the Muscles



(c) Kinetic Energy of the Tensegrity Robot



(d) Total Potential Energy Stored in Muscles



(e) Power Used by the Motors to Roll

Fig. 10: Illustration of different aspects of the Tensegrity Robot over time, during rolling locomotion. The used signals for muscles repeat themselves every 4 seconds. The tensegrity robot completes one revolution in 8 seconds. Tensions, Lengths and Power usage of the robot stays in our defined limits for the hardware.

of our future work. Second, in this work, we are testing open loop controllers. Using some feedback from the robot (such as lengths, tensions or orientation) having a smoother rolling experience can be possible. This is also another future work that we explain in Section 7.

Next, we observe the potential energy stored in the muscles. Figure 10d shows the pattern that repeats itself every 8 seconds as expected. First 4 seconds and second 4 seconds are similar but they slightly differ due to different environ-

ment reaction during the second half of a complete roll. The pattern shows an overall behavior of increasing the potential energy slowly over time, and releasing it. This matches the kinetic energy behavior that we observed in Figure 10c. The kinetic energy of the structure increases during following few seconds whenever the potential energy is released (i.e. t=25s).

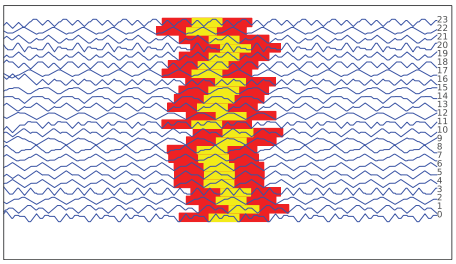
The last set of experiments analyzes the approximated power usage by the motors during the rolling locomotion. In simulation, the power consumption is approximated using the current tension of the element and the constant speed that the motors shortens the muscles. As we explained earlier, the learned behavior is not optimized to be power efficient for this study. On the other hand, we want to make sure that it is within limits of motors and batteries that will be used in the hardware. Figure 10e illustrates the average power consumption of the muscles that varies between 2 to 6 W per muscle. Considering that the motors always pull against the tension (and all the muscles are tight all the time) this value is considerably low. Moreover, it is always possible to lower this value by using a feedback controller in future work.

6 Analyzing the Roles of Different Muscles

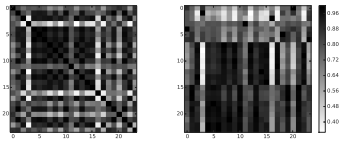
Next, we analyze the signals further by looking at their shapes and correlation between them. The top half of the Figure 11a shows all 24 signals when they are normalized between 0 and 1. The purpose of this experiment is to show similarities of signals and different types of signals learned at the end of coevolution. First, we look at the correlation between signals. For each signal, red-yellow-red area highlights the interval that maximizes correlation with other signals. At this ordering, it is hard to see similarities between signals. In Figure 11d, we shift the signals so that their selected intervals match, then we use hierarchical clustering (Figure 11c) to group the signals according to the similarity metric.

After reordering the signals, Figure 11d shows that half of the signals have one peak high and one bottom, but the other half has 2 signals that are more complex with 2 peak points. This result gives us multiple conclusions. First, the learning algorithm makes use of the complexity provided. Although a subset of the signals is simple, but another subset has more complex signals with multiple peak points that can only be generated with complexity coefficients that are higher than 3. The subsets of the signals that are similar can be regenerated using different parameters but the same formula. Let's consider a normalized signal as $f(x)$. The formula $g(x) = A + B * f(x + C)$ can produce similar signals for different values of A, B and C. Using this idea, all 24 signals can be reproduced using 3 to 4 base functions and different parameters. This gives us a hint about why many papers in literature proposes to use central pattern generators to control tensegrity robots.

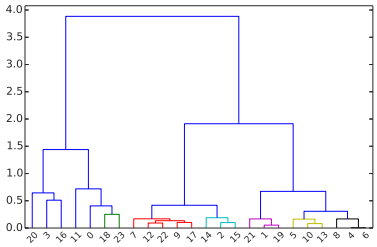
The last set of results show how critical each muscle is for a given rolling locomotion. We take the tensegrity robot with the learned policy and disable of the muscles and observe the effect of such a failure to overall learning behavior. The figure 12 shows that performance depends on which muscle fails. For a significant number of muscles, using the same algorithm still provides rolling behavior with similar performance. On the other hand some muscles play a critical role for that specific policy.



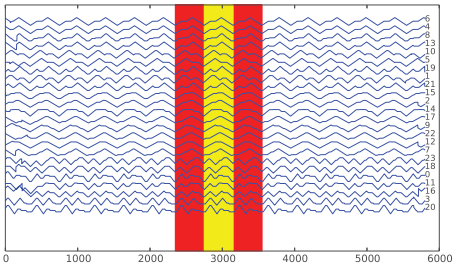
(a) The signals of the 24 muscles normalized between [i,i+1] for muscle i. After correlation using time offsets, the highest correlated intervals of 4 seconds are marked with red,yellow,red pattern.



(b) 24x24 Correlation matrix of signals used for each muscle before and after reordering using hierarchical clustering.



(c) Result of Hierarchical clustering to cluster and reorder similar signals



(d) The signals for the muscles when they are shifted according to the highest correlation and reordered according to the hierarchical clustering.

Fig. 11: The process of analyzing the signals used for the muscles. The signals are shifted and reordered to show similarities. Subfigure (d) shows that groups of signals have similar patterns.

7 Conclusions and Future Work

Tensegrity robots have a great potential with their flexibility, deployability, robustness and different forms of loco-

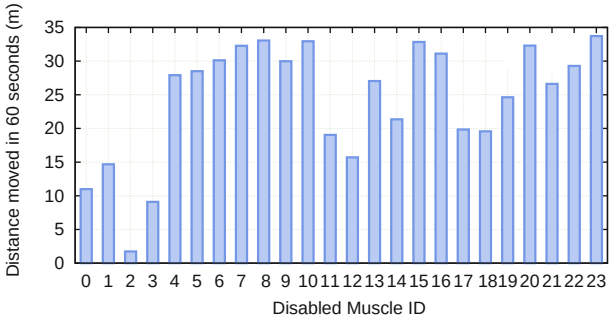


Fig. 12: The performance of the learned policy, when we disable one of the muscles. Learned policy is partially robust to failures of some of the muscles.

motion. Due to these advantages, it is considered as a good candidate for space exploration and mobility missions. On the other hand, controlling a tensegrity robot brings multiple challenges due to the distributed nature and nonlinear interactions between the components. Controlling a tensegrity robot to reach a rolling locomotion is hard to figure out with classical control methods.

In this work, we used coevolutionary algorithms combined with open loop signals to learn distributed rolling locomotion for icosahedron tensegrity robot. We used the model of the robot that is currently under production at NASA Ames Research Center. As a simulator, we used NTRT a simulator for tensegrity robots that closely matches reality. For learning, we analyzed the results of using signals with different complexity levels and different frequencies. The results show that by selecting the right type of signals, learned behavior can reach decent rolling locomotion of 32 meters per minute.

To study the learned behavior, we used one of the learned policies and analyzed the signals used vs the actual lengths of the muscles, the tensions and energy stored in the structure. The results shows that one complete revolution takes 8 seconds. The muscles use the same signal for the first half of the revolution and second half of the revolution. Moreover the rolling behavior is not smooth, the robot repeatedly accelerates and decelerates during rolling. This causes inefficiency but that is understandable since the learning algorithm does not optimize energy consumption.

Finally, we analyzed the signals of different muscles, and concluded that while some muscles can function with simple signals, others require complex activation patterns. In addition, there is significant similarity and overlap among sets signals, implying interchability fo the muscles. This observation leads us to conclude that in terms of robustness, while some muscles play a critical role for rolling, the presented algorithm can handle failures in some muscle groups.

In conclusion, we show that the coevolution combined with open loop signals can provide rolling locomotion. Moreover the learned behavior is feasible for the designed hardware that is under production. Currently, there are open research directions for rolling locomotion for tensegrity robots. First, using multi-objective optimization techniques, we are currently investigating minimizing energy consumption and obtaining a smoother rolling behavior. Second, using feedback for the controllers can increase the overall ability of the locomotion. The algorithm can make use of the information about the interaction of the structure. This future research direction can address the problems such as rolling on uneven terrains, rolling uphill and rolling when there are external forces to disturb or interrupt the locomotion.

Acknowledgments

This research was supported by the NASA Innovative Advanced Concepts program. Ken Caluwaerts was supported by a Ph.D. fellowship of the Research Foundation - Flanders (FWO). Support also came from NSF Graduate Research Fellowship No. DGE1106400, and by NASA Prime Contract Number NAS2-03144 awarded to the University of California, Santa Cruz, University Affiliated Research Center.

The authors would like to thank Andrew P. Sabelhaus, Alice Agogino, Brian Tietz, Terry Fong and the NASA Ames Intelligent Robotics Group.

References

[1] Agogino, A., SunSpiral, V., and Atkinson, D., 2013. "Super Ball Bot - structures for planetary landing and exploration". *NASA Innovative Advanced Concepts (NIAC) Program, Final Report*.

[2] Fuller, B., 1961. "Tensegrity". *Portfolio and Art News Annual,* **4**, pp. 112–127.

[3] Snelson, K., Feburary 1965. Continuous tension, discontinuous compression structures. united states patent 3169611.

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

[5] Bel Hadj Ali, N., Rhode-Barbarigos, L., Pascual Albi, A., and Smith, I., 2010. "Design optimization and dynamic analysis of a tensegrity-based footbridge". *Engineering Structures,* **32**(11), pp. 3650–3659.

[6] Juan, S. H., and Tur, J. M. M., 2008. "Tensegrity frameworks: Static analysis review". *Mechanism and Machine Theory,* **43**(7), pp. 859 – 881.

[7] Klimke, H., and Stephan, S., 2004. "The making of a tensegrity tower". In IASS Symposium.

[8] Ingber, D. E., 1993. "Cellular tensegrity: defining new rules of biologic design that govern cytoskeleton". *Journal of Cell Science,* **104**, pp. 613–627.

[9] Levin, S., 2002. "The tensegrity-truss as a model for spine mechanics". *Journal of Mechanics in Medicine and Biology,* **2**, pp. 375–388.

[10] Motro, R., 2003. *Tensegrity: structural systems for the future*. Butterworth-Heinemann.

[11] Wroldsen, A., de Oliveira, M., and Skelton, R., 2006. "A discussion on control of tensegrity systems". In Decision and Control, 2006 45th IEEE Conference on, IEEE, pp. 2307–2313.

[12] Zhang, J. Y., and Ohsaki, M., 2006. "Adaptive force

density method for form-finding problem of tensegrity structures". *International Journal of Solids and Structures,* **43**, pp. 5658–5673.

[13] Masic, M., and et al., 2005. "Algebraic tensegrity form-finding". *International Journal of Solids and Structures,* **42**, pp. 4833–4858.

[14] Tibert, and et al., 2003. "Review of form-finding methods for tensegrity structures". *International Journal of Space Structures,* **18**, pp. 209–223.

[15] Rieffel, J., Valero-Cuevas, F., and Lipson, H., 2009. "Automated discovery and optimization of large irregular tensegrity structures". *Comput. Struct.,* **87**(5-6), Mar., pp. 368–379.

[16] Zhang, L., and et al., 2006. "Form-finding of nonregular tensegrity systems". *Journal of Structural Engineering,* **132**, pp. 1435–1440.

[17] Paul, C., Lipson, H., and Cuevas, F. J. V., 2005. "Evolutionary form-finding of tensegrity structures". In Proceedings of the 2005 conference on Genetic and evolutionary computation, GECCO '05, ACM, pp. 3–10.

[18] Pugh, A., 1976. *An introduction to tensegrity*. Univ of California Press.

[19] Fujiia, M., Yoshiia, S., and Kakazub, Y., 2006. "Movement control of tensegrity robot". *Intelligent Autonomous Systems 9: IAS-9,* **9**, p. 290.

[20] Tietz, B. R., Carnahan, R. W., Bachmann, R. J., Quinn, R. D., and SunSpiral, V., 2013. "Tetraspine: Robust terrain handling on a tensegrity robot using central pattern generators". In AIM, pp. 261–267.

[21] Paul, C., Valero-Cuevas, F., and Lipson, H., 2006. "Design and control of tensegrity robots for locomotion". *Robotics, IEEE Transactions on,* **22**(5), pp. 944–957.

[22] Paul, C., Roberts, J., Lipson, H., and Cuevas, F., 2005. "Gait production in a tensegrity based robot". In Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on, IEEE, pp. 216–222.

[23] SunSpiral, V., Gorospe, G., Bruce, J., Iscen, A., Korbel, G., Milam, S., Agogino, A., and Atkinson, D., 2013. "Tensegrity based probes for planetary exploration: Entry, descent and landing (EDL) and surface mobility analysis.". *International Journal of Planetary Probes*, July.

[24] Shibata, M., Saijyo, F., and Hirai, S., 2009. "Crawling by body deformation of tensegrity structure robots". In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE, pp. 4375–4380.

[25] Shibata, M., and Hirai, S., 2010. "Moving strategy of tensegrity robots with semiregular polyhedral body". In Proc. of the 13th Int. Conf. Climbing and Walking Robots (CLAWAR 2010), Nagoya, pp. 359–366.

[26] Koizumi, Y., Shibata, M., and Hirai, S., 2012. "Rolling tensegrity driven by pneumatic soft actuators". In Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, pp. 1988–1993.

[27] Rieffel, J. A., Valero-Cuevas, F. J., and Lipson, H., 2009. "Morphological communication: exploiting coupled dynamics in a complex mechanical structure to achieve locomotion". *J R Soc Interface,* **7**, pp. 613–621.

[28] Tur, J. M. M., and Juan, S. H., 2009. "Tensegrity frameworks: Dynamic analysis review and open problems". *Mechanism and Machine Theory,* **44**, pp. 1–18.

[29] Caluwaerts, K., D'Haene, M., Verstraeten, D., and Schrauwen, B., 2013. "Locomotion without a brain: physical reservoir computing in tensegrity structures". *Artificial Life,* **19**(1), pp. 35–66.

[30] NasaTensegrityRoboticsToolkit. http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity/.

[31] BulletPhysicsEngine. http://www.bulletphysics.org/.

[32] Back, T., Fogel, D. B., and Michalewicz, Z., 1997. *Handbook of evolutionary computation*. IOP Publishing Ltd.

[33] Wiegand, R. P., 2003. "An analysis of cooperative coevolutionary algorithms". PhD thesis, Citeseer.

[34] Panait, L., Tuyls, K., and Luke, S., 2008. "Theoretical advantages of lenient learners: An evolutionary game theoretic perspective". *The Journal of Machine Learning Research,* **9**, pp. 423–457.

[35] Iscen, A., Agogino, A. K., SunSpiral, V., and Tumer, K., 2013. "Controlling tensegrity robots through evolution". In GECCO, pp. 1293–1300.