Super Ball Bot - Structures for Planetary Landing and Exploration

Jérémie Despraz

jeremie.despraz@epfl.ch

Master Thesis

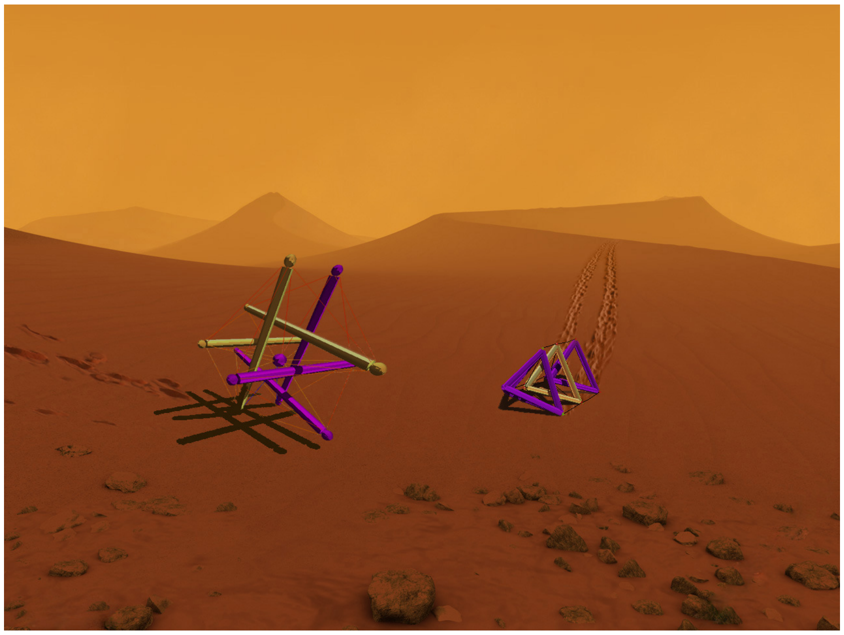
under the supervision of Professor Auke Jan Ijspeert & Vytas SunSpiral
Assistant: Mostafa Ajallooeian

NASA Ames Research Center - Intelligent Robotics Group
Ecole Polytechnique Fédérale de Lausanne - Biorobotics Laboratory

submitted on August 16, 2013

Abstract

The goal of this Master project is to develop central pattern generator (CPG) based controls for a Planetary Exploration Tensegrity Robot. Development will occur primarily in a physics-based tensegrity simulator, and will involve an exploration of various candidate tensegrity structures and associated controls to roll the robot over various terrains. Tensegrity structures are uniquely appropriate for distributed CPG based control due to their compliant tension network which mirrors the properties of the CPG network. It is believed that tensegrity robot technology can play a critical role in future planetary exploration[1].



Artist view of tensegrity robots on the surface of Titan

Acknowledgements

Foremost, I would like to thank Professor Auke Jan Ijspeert and Vytas SunSpiral for making me part of this wonderful and inspiring project. I greatly appreciated the kind supervision and support of Vytas during my stay at NASA Ames, working there has been an extraordinary experience. I would like to thank also Mostafa Ajallooeian, Atıl İşçen, Ken Caluwaerts, Jesse van den Kieboom, Ryan Adams, Jonathan Bruce, Andrew Sabelhaus, In Won Park, Brian Tietz, Alexandra Pogue, Jeffrey Friesen, Kyle Morse, George Korbel and Sophie Milam for their friendly collaboration and help throughout the whole project. Finally, last but not least, I am deeply grateful to Jessica Ryan for her constant love and encouragement, despite the distance that separated us.

Contents

1 Introduction

The aim of this work is to create, develop and test an efficient control strategy to allow a tensegrity robot to smoothly move over various type of complex terrains. The robot considered in this study is a tensegrity icosahedron made of 6 solid struts, 36 actuated springs and a central payload. Experiments are performed mainly in a physics simulator and partly on hardware to validate the simulation data and justify a further investigation. This study focuses on the feasibility and capabilities of different types of controllers and how they can be implemented and used together. We consider here mainly two types of approaches namely reactive controls and Central Pattern Generator based actuation.

Tensegrity robotics and in particular tensegrity control is a fairly new field of research. As a consequence, it was not considerably explored and a great number of tools and techniques can potentially be developed in this domain. This work is a humble step in this direction. Several research institutions such as NASA are interested in this novel area of robotics. Tensegrity robots possess many advantages that makes them very good candidates for specific space missions such as planetary landing and exploration, pushing forward the need for versatile, reliable and energy efficient control strategies.

This thesis starts by a brief introduction to tensegrity structures (section 2) and their applications in robotics (section 3). The topic of tensegrity control is then addressed and a review of the main approaches is presented (section 4). In section 5, the physics simulator used to simulate the physical world and the tensegrity structures is presented in details, with its specific features relevant to tensegrity and CPG simulation.
Sections 6, 7 and 8 contain the core of this study, namely the description of the various control strategies, their relative performances and a critical review (section 6), an analysis of the obtained gait patterns (section 7) and a validation of the physics simulator to assess the scientific value of the data obtained (section 8).
In section 9, several options are considered for a future work and different improvements and novel ideas are suggested. Section 10 summarizes and concludes the study.

2 Introduction to Tensegrity Structures

Tensegrity structures have been widely used in architecture and art but have started only recently to be studied in the field of robotics[2]. Tensegrity structures were first introduced in the 1960's by the american sculptor and photographer Kenneth D. Snelson[3]. Snelson was interested in patterns of physical forces and nature in its primary aspect. The term tensegrity itself was introduced by architect Buckminster Fuller as a combination of "tension" and "integrity"[4].

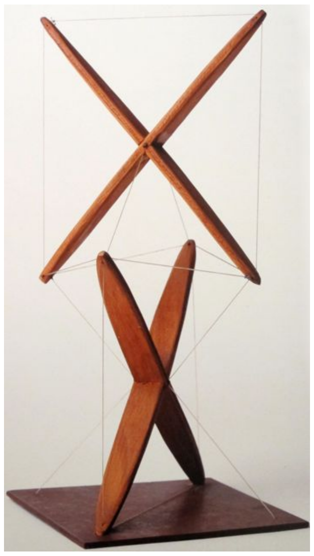


Figure 1: *Snelson artwork: "Early X piece", 1948, from [5]*

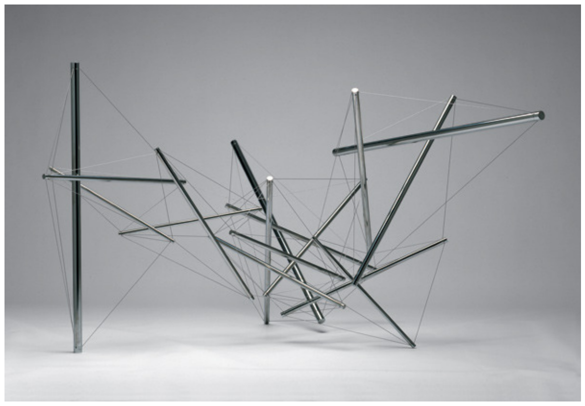


Figure 2: *Snelson artwork: "Forest Devil", 1975, from [6]*

In the field of science and engineering, Robert Skelton and Mauricio C. de Oliveira carried out a deep and complete study of tensegrity structures[7] with the purpose of improving and integrating control design with structure design. As part of their work, they developed a complete mathematical formalism to describe tensegrity structures. This formalism relies on linear algebra to describe in a very elegant manner the physical properties of tensegrity systems. They give a proper mathematical definition of a *tensegrity configuration* and a *tensegrity system* (hereafter tensegrity structure):

- *In the absence of external forces, let a set of rigid bodies in a specific configuration have torqueless connections (e.g. via frictionless ball-joints). Then this configuration forms a tensegrity configuration if the given configuration can be stabilized by some set of internal tensile members, i.e. connected between the rigid bodies. The configuration is not a tensegrity configuration if no tensile members are required and/or no set of tensile members exist to stabilize the configuration.*

- *A tensegrity system is composed of any given set of strings connected to a tensegrity configuration of rigid bodies.*

Furthermore, they define classes of tensegrity structures depending on the number of rigid body connected to each other. A tensegrity structure that has zero contact points between its rigid bodies is called a class 1 tensegrity structure (e.g. figure 2). Correspondingly, a tensegrity structure that has k contact between its rigid bodies is therefore called a $(k+1)$ class tensegrity structure. In this work, we will consider only class 1 tensegrity structures.

Tensegrity structures can very efficiently share any load that is applied to any point of the structure by distributing the forces throughout its tension network. Furthermore, every element is either in pure extension or compression and no element creates lever arms that could magnify the forces. As a result, no bending or shear forces have to be beared by the elements and a tensegrity structure is therefore much less subject to having a local point of weakness where all the load of the structure adds up and can be made of very light weight materials. Moreover, the distribution of the forces through the tension network, while making the structure very resistant, also makes it tolerant to damages. Indeed, if a tensile member had to break, the tension network would instantaneously redistribute forces to counterbalance the change and come back to a new equilibrium point. A tensegrity can thus easily respond to shocks, external stresses or to any sudden change of orientation. Another advantage resides in the large ratio load/weight that tensegrity structures exhibit. Due to the prestress that can be applied to the tension network, tensegrity structures can bear large loads without the necessity to add up weight to the structure. A detailed list of the benefits of tensegrity structures was published by Skelton et al. in[2].

In the light of all these advantageous physical properties, it is no surprise that tensegrity structures are observed in nature, where billions of years of evolution favored the emergence of the most resistant and effective designs. A recent study by Donald E. Ingber has shown that the cytoskeleton of cells are constructed as tensegrity structures[8], see figure 3. On another scale, the whole body can also be seen as a tensegrity structure where the bones are the rigid elements intricated in a tension network created by the fascia tissue[9]. Another example is spider silk fiber which is known to be very resistant to loads and extension, exhibit also a tensegrity structure on the molecular level[10].

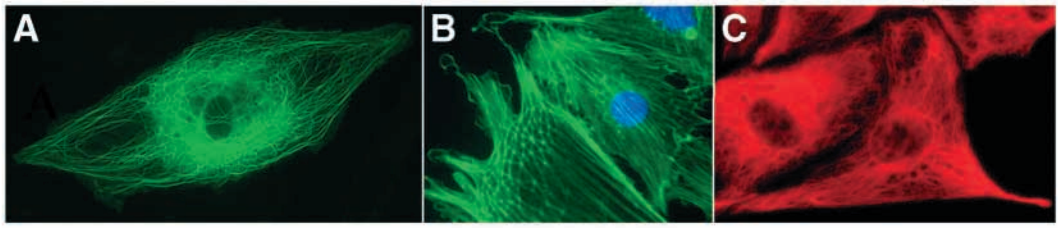


Figure 3: *Microtubules, microfilaments and intermediate filaments within the cytoskeleton of endothelial cells reveling the underlying tensegrity structure of these cells (from [8]).*

3 Tensegrity Structures in Robotics

3.1 Advantages

Using tensegrity in the field of robotics is a relatively new idea and is still an active field of research. In the light of the previous statements, the use of tensegrity structures for robotics brings a lot of advantages that are critical to robot design. In particular, tensegrity robots can easily respond to shocks, damage and loads by redistributing the force throughout their tension network, a feature that is usually not present in classical robots. For the same reason, they can also carry heavy payloads with respect to their own mass. By applying pressure on either side of the structure or by contracting a specific set of tensile elements, they can be packed in small, almost flat shapes that, added to their lightweight, makes them very easy to transport and easily deployable.

Another great advantage that tensegrity structure possess over classical robots is the intrinsic compliance of the structure that literally merges the controls with the structure. It is indeed the structure of the robot itself that is actuated to make the robot move and not any external device build on top of the original design. The control architecture is embedded in the system as it is part of the structure itself.

Another interesting feature that tensegrities possess is the ability to change their structure's stiffness. Indeed, changing the length of one or several tensile elements makes the overall structure stiffer and can therefore considerably change some of its physical properties such as natural frequency, shape, orientation, equilibrium position, etc.

Tensegrity structure, and as a consequence tensegrity robots, are by definition a network of interconnected tensile elements and rigid elements. As a result, they are by nature fully modular. Connecting a tensegrity structure to another creates a new tensegrity structure. This property can be highly exploited, on one hand for the construction of multicellular robots that can be assembled together from simple building blocks and on the other hand for the implementation of fully distributed control strategies.

3.2 Applications to Autonomous Robotics

All these advantages make tensegrity robots very good candidates for a wide variety of applications. Their ability to be packed in small spaces, absorb large shocks, their lightweight and great structure to payload mass ratio makes them very suitable as space exploration probes, as evidenced by NASA's interest in this domain[1]. As an example, a detailed study from the 2013 International Planetary Probe Workshop[11] describe the feasibility of a mission to Saturn's moon Titan. Figure 4 summarizes the proposed missions scenario where tensegrity probes are being deployed from a spacecraft and land on the satellite's surface. Simulations results from this study demonstrate that the tensegrity probes could resist a landing on the surface of Titan without any expensive and complicated landing device such as retrorockets or parachutes, the energy of the shock at impact being completely absorbed by the structure.

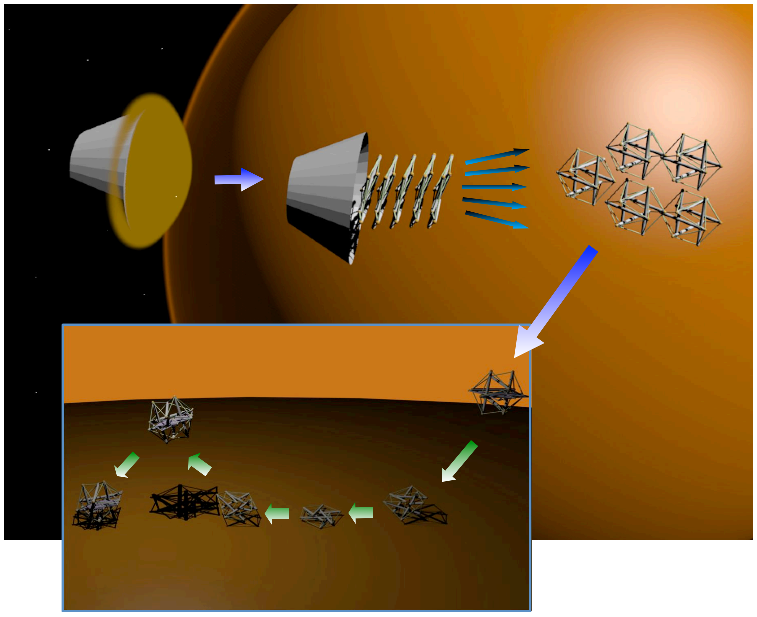


Figure 4: *Mission Scenario - Tightly packed set of tensegrities, expand, spread out, fall to surface of moon, then safely bounce on impact. The same tensegrity structure which cushioned the landing is then used for mobility to explore moons such as Titan and small asteroids (from [1]).*

Tensegrity structures being also omnipresent in Nature, biorobotics is field were tensegrity robotics finds a broad range of applications. As an example, Tietz et al. developed a modular tensegrity snake robot[12] based a the model of tensegrity spine first introduced by Tom Flemmons (see figure 5, 6 and 7 below). The robot is made of interconnected tetrahedron-shaped rigid elements interconnected via six actuated elastic strings driven by a central pattern generator. The robot is able to crawl on complex terrain and successfully overcome obstacles in simulation.

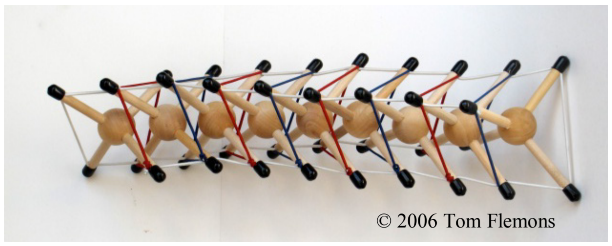


Figure 5: *Tensegrity model of the spine by Tom Flemmons (from [13])*

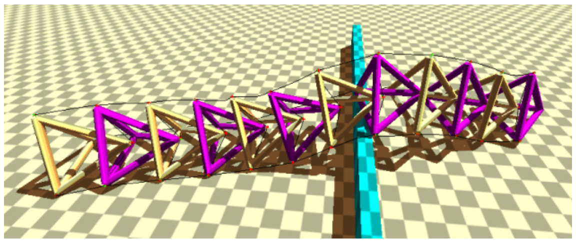


Figure 6: *Spine robot in simulation (from [12])*

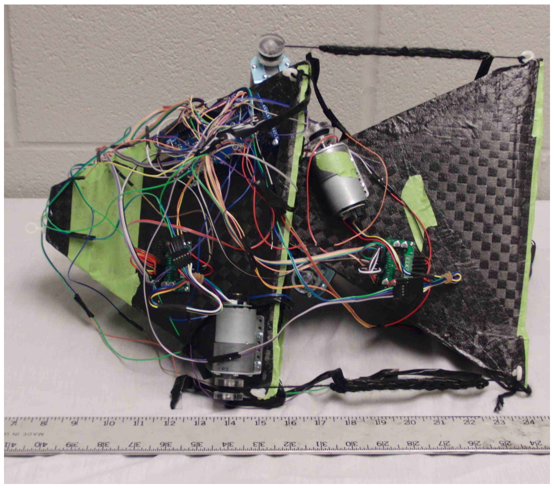


Figure 7: *Tensegrity spine hardware prototype (from [12]).*

Tensegrity robots can also play a critical role in areas where lightweight design techniques do not affect the quality of the robots functionalities and physical capabilities. Typical areas are search and rescue operations in hazardous environments, military operations, human assistive function, where the financial costs of transportability and the energetic costs of mobility are key issues in design[14].

The field of tensegrity robotics overlaps also with other domains of robotics research with a great potential of applications. As an illustration, the ability of tensegrity robots to dynamically change their stiffness can allow for shape shifting and behaviors very similar to soft robots[15]. In the same way, their high modularity can be of great interest for distributed controls robotics, autonomous modular robotics, etc.

4 Controlling Tensegrity Robots

Due to their unique structure and great compliance, tensegrity robots are highly controllable as a force applied at a point of the tensegrity will be transmitted in the whole tension network. Their response is non-linear and oscillatory and traditional control strategies are therefore not well adapted for this type of robots and new methods have to be developed.

Several control strategies have been developed in the past taking different approaches to deal with the inherent non-linearities of the system. We make here the distinction between quasi-equilibrium control where the robot is brought to a series of stable equilibrium states until the desired configuration is reached and dynamical control where the tensegrity robot is put in motion and brought out of its equilibrium position. Note that this is a non exhaustive list of control strategies, the point being here to emphasize the distinction between the two types of methods.

4.1 Quasi-equilibrium Control

Anders Wroldsen et al. proposed two control strategies for class 1 tensegrity structures using a Lyapunov based control method[16]. The idea is, given a target position, to construct a Lyapunov function for the system that ensures convergence of the structure to the target position. A demonstration in simulator is carried out using a pinned bar actuated by three string. Different target position can be reached using the control strategy. Another possibility is to compute the forces acting on each node of the tensegrity structure and try to minimize them for a specific target configuration. Such an algorithm has been tested and implemented by Burt[17].
The main disadvantage of these control strategies is the large amount of computation required to compute the values of the control parameters and the fact that the structure has to be close to an equilibrium position. Indeed, if the structure is subject to oscillations, new forces will apply on the nodes and the algorithm will not be able to converge to the target position. Similarly, if the structure is rolling on the ground, the contact points will change and thus the Lyapunov function would need to be recomputed for a different set of constraints, making it unable to converge to the desired configuration. A new type of controls is therefore needed to account for dynamic motion of the robot.

4.2 Dynamic Control

Different strategies have been explored to deal with the non-linear dynamics of the tensegrity robots. Koizumi et al. proposed an optimal gait pattern for a 6 struts icosahedron tensegrity structure where the 24 edges of the robot can be inflated with air using pneumatic soft actuators[18] (see figure 8). The authors studied different gait patterns for the robot depending on the surface in contact with the ground and found an optimal gait for the tensegrity icosahedron. An example of a stable gait pattern is presented on figure 9.

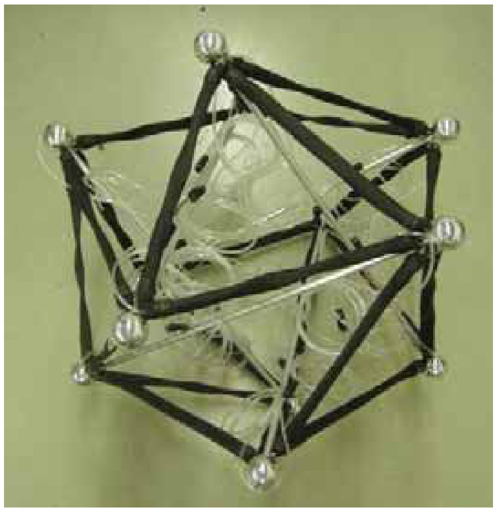


Figure 8: *Tensegrity icosahedron controlled by 24 pneumatic actuators (from [18])*

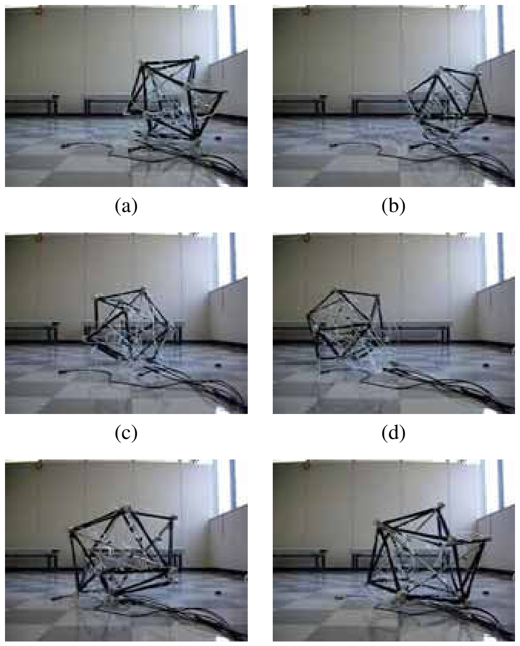


Figure 9: *Example of a gait obtained by manually actuating the edges of the tensegrity robot (from [18])*

Atıl Iscen et al. proposed an open loop control policy based on sinusoidal actuators co-evolved in order to maximize the distance traveled by the tensegrity robot[19]. The actuators act on the rest length ℓ of the springs as: $\ell(t) = A\sin(\omega t + \phi) + \bar{\ell}$, where A, ω, ϕ and $\bar{\ell}$ are evolved within an evolutionary framework embedded in a physics simulator. Results show that the tensegrity robot is able to learn a rolling gait after about 10'000 generations. The disadvantage with this method however is that the robot cannot be steered and any unexpected obstacle or bump on the ground can stop the robot and make it unable to roll for the entire simulation. Also, any different starting position will amount to a different set of evolved parameters.
Ken Caluwaerts et al. developed a closed-loop control strategy based on reservoir computing [20]. The idea is to generate a signal using a central pattern generator (CPG) and evolve parameters of a neural network that is used to tune its oscillatory output. A bias and non-linearities are added to the resulting signal by computing its hyperbolic tangent. The output signal is then used to control the robot actuators. In this paper, the authors used a Matsuoka oscillator to generate the basic oscillatory signal whose output was then compared to a teaching target signal. This control strategy has the advantage of requiring few computations, as the command signal is previously learned through a teaching phase and requires only simple matrix multiplications in the running phase. Moreover, this method uses feedback from the strings tension to estimate the robot's state, allowing it to correct the motor commands in order to come back to the target configuration and can therefore recover from shocks, obstacles and unexpected terrain configurations.

4.3 Central Pattern Generators Based Control

Central pattern generators (CPGs) are neural networks that can produce rhythmic patterned outputs without rhythmic sensory or central input[21]. Rhythmic neural activity is observed in both vertebrate and invertebrate animals[22] and in particular in neural networks responsible for locomotion[23]. CPGs are well adapted to cope with non-linearities and the great compliance inherent to animal organisms. As a consequence, CPGs have been widely studied as controllers in the field of robotics, a review by Auke Ijspeert[24] gives an overview of the use of central pattern generators for animal and robot locomotion.

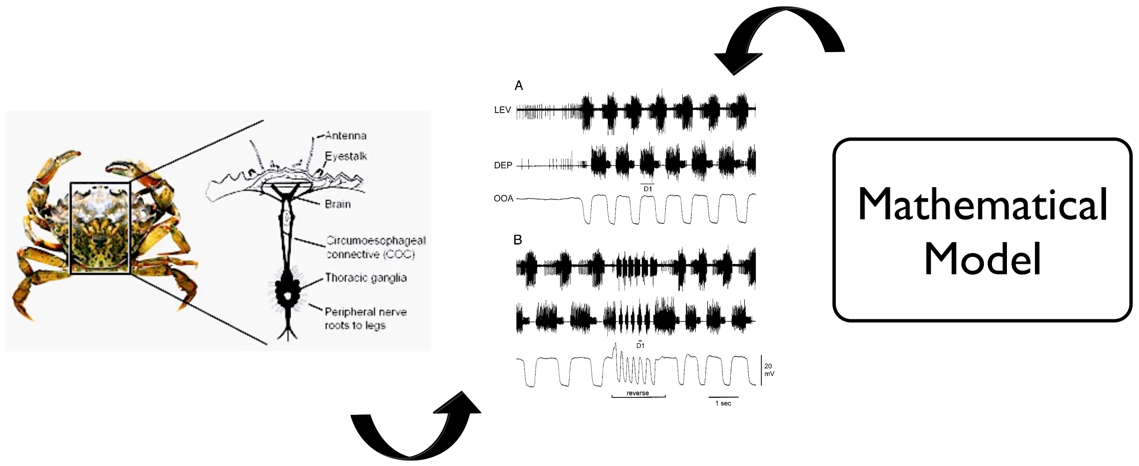


Figure 10: *Generation of rhythmic patterns by a crab's central pattern generator and a mathematical model tuned to reproduce the same type of signal (modified from [25])*

Oscillatory output of CPGs can be modeled using dynamical systems[24]. The design of these dynamical systems is usually not an easy task and require a large amount of parameter tuning. Nevertheless, they allow for a greater level of abstraction while still reflecting the main properties of central pattern generators. For instance, non-linear dynamical systems exhibiting stable limit cycles (e.g. Van der Pol oscillators, Hopf oscillators, etc.) can be created to simulate rhythmic signal generation as illustrated on figure 10. These mathematical systems have the desired property to be stable to perturbations, to easily encode multi-dimensional periodic patterns and, by tuning the right parameters, they can be easily modulated in frequency and amplitude[26][27] (see figure 11). Furthermore, they react very smoothly to any disturbance that can perturb the system.

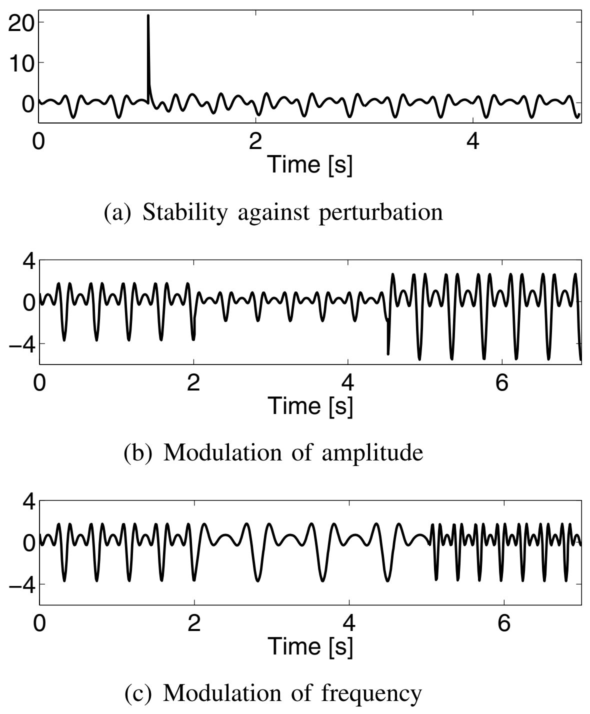


Figure 11: *Modulation of a dynamical system output modeling a CPG (from [27]).*

Another property that can be embedded in dynamical system is an adaptive frequency behavior where the system is able to synchronize to any input signal. This property enables for instance complex signal generation from a series of simple sinusoidal outputs[27] and allows for learning phases in which a CPG can be trained to accurately adapt to complex environmental constrains. This last feature has been extensively used in the development of CPG controllers for tensegrity structures as presented in section 6.6.

5 Physics Simulator

Physics simulation of tensegrity structures and their interaction with the environment was simulated using the NASA Tensegrity Robotics Toolkit (NTRT)[1]. NTRT is built on top of the open source Bullet physics engine developed mainly for gaming and animation purposes[28]. It uses an hybrid impulse and constraint-based engine to solve the equations of motions of the simulated bodies. The main advantages of the Bullet library, that justify our choice as NTRT's core physics simulator are a variable time stepping, a library modeling rigid body - soft body interaction, the possibility to run real time simulations, and the overall physically accurate performances of the engine (see study by Boing et al.[29] for a detailed analysis of Bullet's performances). Relevant experiments assessing the physical accuracy of the simulator for tensegrity robots are carried out in section 8.

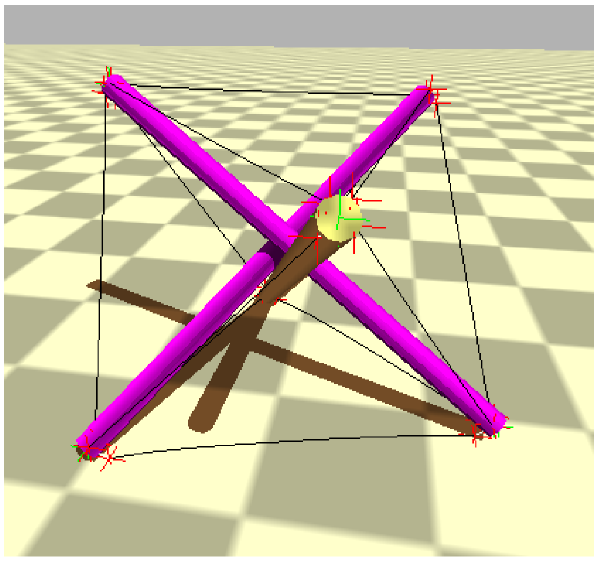


Figure 12: *Simple class 1 prism tensegrity structure simulated in the NTRT physics engine.*

In this section we will define the different tools that allow us to accurately simulate tensegrity robots in realistic environments, describe the tensegrity robot model used for the different experiments presented in the later sections and explain how the various libraries are embedded. A detailed listing of the NTRT physical properties used in this work are presented in section 12.

[1]The NASA Tensegrity Robotics Toolkit (NTRT) is soon to be released open source, latest news can be found on the official project's website http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity

5.1 Tensile Components

Bullet provides a library for the modeling of elastic components as a chain of connected rigid bodies that can be used to model tensile elements of tensegrity structures. However, this implementation has been discovered to introduce free energy in the system, allowing the tensegrity to spin on itself in a way that obviously breaks the laws of conservation of energy. In order to solve this problem, NTRT implements a new class of hookean springs. This new class models tensile elements (or muscles) as massless and shapeless elements that are attached to rigid bodies by two anchor points. These muscles have an editable rest length and can apply a force to their connected bodies when in extension. The force applied by a muscle i is computed according to the following set of equations:

$$\begin{cases} F_i = k_i(\ell_i - l_i) - \eta \dfrac{(l_i^{(t)} - l_i^{(t-1)})}{dt} & , l_i > \ell_i \\ 0 & , \text{otherwise} \end{cases}$$

Where k_i is the spring constant of muscle i, ℓ_i and l_i are its rest length and actual length respectively, η is a damping coefficient and dt is the simulation time step. With this spring model, the muscle applies a force to its rigid bodies if and only if it is in extension and goes slack otherwise. Obviously, with this implementation, two connected rigid bodies experience an equal and opposite force as expected from Newton's third law.

Note that the muscles as modeled here cannot experience contact forces or shocks as they do not have any physical shape. They can be seen as a set of physical parameters: rest length, length, anchor points and forces. As a consequence, more realistic strings models have to be implemented to deal with complex interactions with ground objects, entanglements, etc. However, it is possible to approximate massive springs by adding intermediary masses and, instead of modeling one full spring, add three distinct springs in-between as illustrated on figure 13. This type of model has also the advantage of reflecting the design of some tensegrity robots such as the one used for hardware experiments presented in section 8.

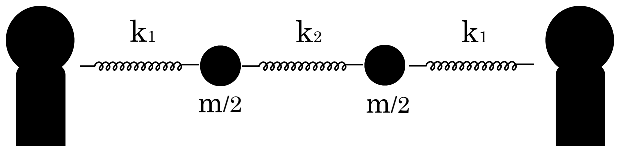


Figure 13: *Schematic view of a 3-segment muscle, allowing to account for springs mass and accurately simulate real tensegrity tensile components.*

5.2 Terrain Configurations

The NTRT simulator allows also the modeling of various terrains that can be used to test the performances of the robot's controllers. Figures 14, 15, 16 and 17 present some of these different basic terrain configurations. Any combination of these terrain can also be simulated allowing the creation of a wide variety of test platforms.

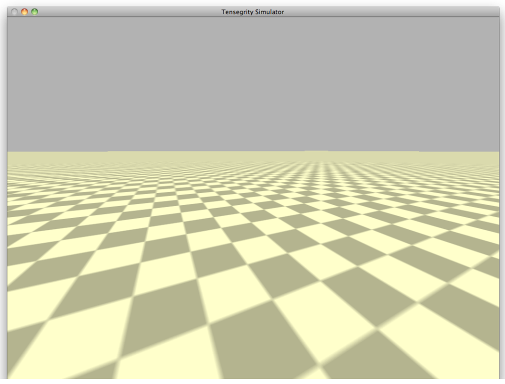


Figure 14: *flat regular terrain*

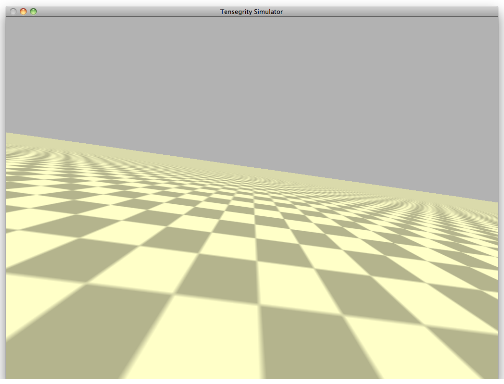


Figure 15: *8° slope*

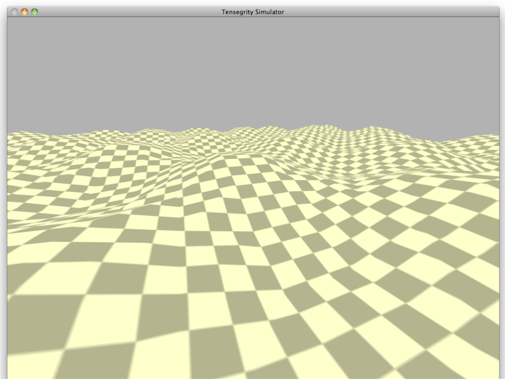


Figure 16: *regular bumps*

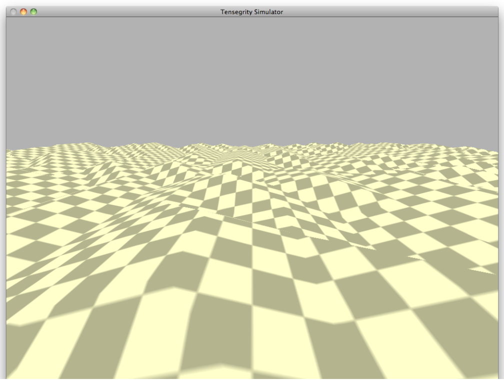


Figure 17: *random terrain configuration*

These different terrains are used as test beds for our different control algorithms. They can also be used as a reference benchmark to compare the performances of different control strategies.

5.3 The Tensegrity Icosahedron

The tensegrity robot used for the simulations is composed of 6 struts (compressive elements) connected together by 24 actuated elastic strings (tensile elements). The structure forms an icosahedron with 8 regular triangular faces and 12 non-regular isosceles triangles. A central payload is maintained in the center of the robot by 12 actuated elastic springs. Each strut is made of an homogeneous cylinder ended up by two identical spheres. The tensile elements are attached to the struts at the junction point between the spheres and the cylinder. A model of the prototype is presented on figure 18 below.

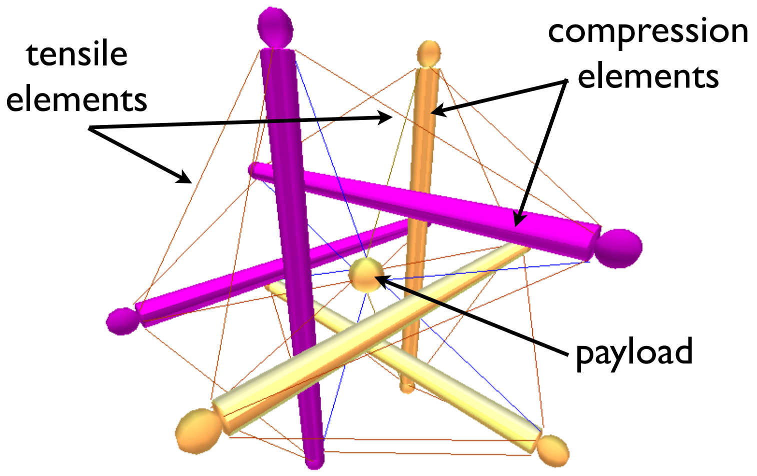


Figure 18: *The tensegrity icosahedron as modeled in the NTRT simulator. The structure consists of 6 compression elements (struts), 24 outer and 12 inner elastic springs. A payload is attached to the center of the structure, each string can be actuated independently.*

In order to model a realistic robot, we choose the physical properties of the tensegrity according to previous work by Atıl İşçen et al. [11] and according to a thorough design study for a novel tensegrity prototype (see section 8). The main physical parameters are summarized in the table below:

	mass [kg]	length [m]	k [N/m]
struts	1.0	1.0	∞
payload	5.0	0.1	∞
inner strings	0.0	0.45	1000
outer strings	0.0	0.53	1000

Note that we refer to inner strings as the strings connecting the struts to the central payload and outer strings as the strings connecting the struts together. The length of the payload is defined as the diameter of the sphere and the length of the strings refers to their initial rest length. A full detailed listing of the physical properties of rigid and tensile elements is presented in section 12.

5.4 Actuators and Controls

Actuation of the muscles is performed by either increasing or decreasing the rest length of the springs. This choice is justified to keep consistency with previous simulation experiments and results and by the following physical argument:

Imagine the simple undamped planar spring - mass system as represented on figure 19. We assume that each spring has a rest length given by ℓ and ℓ_c and we choose a coordinate system with the Ox axis pointing to the right and denote by x_1 and x_2 the horizontal position of mass m_1 and m_2 respectively.

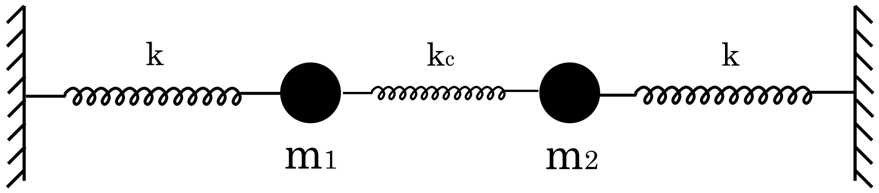


Figure 19: *planar spring - mass system*

The forces F_1 and F_2 applied on each of the masses can be easily computed:

$$F_1 = k(x_1 - \ell) + k_c(x_2 - x_1 - \ell_c)$$
$$F_2 = k(x_2 - \ell) + k_c(x_1 - x_2 - \ell_c)$$

and the equations of motion are given by Newton's second law:

$$m\ddot{x}_1 = k(x_1 - \ell) + k_c(x_2 - x_1 - \ell_c) \tag{5.1}$$
$$m\ddot{x}_2 = k(x_2 - \ell) + k_c(x_1 - x_2 - \ell_c) \tag{5.2}$$

The general solution of these two equations for $\ell = \ell_c = 0$ is of the form:

$$x_1(t) = A\cos(\omega_1 t + \phi_1) + B\cos(\omega_2 t + \phi_2)$$
$$x_2(t) = A\cos(\omega_1 t + \phi_1) - B\cos(\omega_2 t + \phi_2)$$

with $A, B, \phi_1, \phi_2 \in \mathbb{R}$ and $\omega_1, \omega_2 \in \mathbb{R}_+$. Hence, a general solution for (5.1) and (5.2) is of the form:

$$\hat{x}_1(t) = A\cos(\omega_1 t + \phi_1) + B\cos(\omega_2 t + \phi_2) + \alpha$$
$$\hat{x}_2(t) = A\cos(\omega_1 t + \phi_1) - B\cos(\omega_2 t + \phi_2) + \beta$$

with $\alpha, \beta \in \mathbb{R}$. By substitution in (5.1) and (5.2), this gives:

$$0 = k\alpha + k_c(\beta - \alpha) - k\ell - k_c\ell_c$$
$$0 = k\beta + k_c(\alpha - \beta) - k\ell - k_c\ell_c$$

and thus we find $\alpha = \beta = \ell + k_c/k\ell_c$ yielding:

$$\hat{x}_1(t) = A\cos(\omega_1 t + \phi_1) + B\cos(\omega_2 t + \phi_2) + \ell + k_c/k\ \ell_c$$
$$\hat{x}_2(t) = A\cos(\omega_1 t + \phi_1) - B\cos(\omega_2 t + \phi_2) + \ell + k_c/k\ \ell_c$$

Since $A, B, \omega_1, \omega_2, \phi_1, \phi_2$ are independent of ℓ and ℓ_c, we find that the position of the masses depends *linearly* on the rest length of the springs. This derivation justifies the choice of the spring's rest length as a control parameter. Note that, in higher dimensions, angles between struts and springs induce non-linearities that break this simple linear control law.

The robot possesses a motor connected to each string acting on the rest length. Each motor has a limit velocity and a limit acceleration. When the desired rest length is assign to the controller, the motor rotates until the new rest length is as close as possible to the assigned length. Observe that stiff strings can be approximated by springs having a very large spring constant k. In that particular case, the rest length and the actual length of the spring are indistinguishable and, as a result, a motor displacement of length d corresponds to the same actual displacement of the physical string. This result is used to accurately model motors and actuators of real tensegrity robot prototypes, as presented in section 8.

The actuation of the motors is performed via an impedance controller that imposes a desired dynamic behavior to the interaction between the robot and its environment, preventing it to induce too large forces in the tensile components of the structure. In that way, the robot can safely deal with uncertain geometric characteristics of the environments and dynamically adapt to its physical properties (e.g. stiffness, damping, friction, etc.) in a complementary way [30]. Moreover, Orki et al. used a formulation of impedance control in a two dimensional caterpillar tensegrity robot [31] and, more recently, Tietz et al. used the same impedance controller approach to control the Tetraspine robot [12] presented in section 3.2. A mathematical description of the impedance controller is introduced in section 6.2.

5.5 Codyn

We use the Codyn library[32] in order to model CPGs/dynamical systems within the physics simulation. Codyn is an open source software framework that provides objects, methods and algorithms to mathematically describe and solve the state of a dynamical system in parallel to the physical simulation. It has also the advantage of being easily embedded in an evolutionary framework.
Codyn stores the dynamical system as an object called a node that can be coupled to other nodes and therefore allows the user to create complex networks of coupled oscillators. Each node contains information on its current state, i.e., the current value of its different variables and their associated mathematical expression. The state of the network can be updated using numerical integrators that are part of the framework. In this work, we used a 4th order Runge-Kutta (RK4) method to obtain sufficient precision and stability on the dynamical systems outputs.

```
node "oscillator"
    {
        ## x state of Hopf oscillator
        ẋ = "γ_x • (μ − r2) • x − ω • y"

        ## y state of Hopf oscillator
        ẏ = "γ_y • (μ − r2) • y + ω • x"

        # Initial values outside of fixed point
        x = 0
        y = 1

        ## Desired amplitude
        target_amplitude = 1

        ## Desired frequency
        target_frequency = 1

        ## Instantaneous amplitude
        r = "hypot(x, y)"

        ## Desired amplitude². Controls the Hopf bifurcation.
        μ = "target_amplitude²"

        ## Gain on amplitude
        "γ_{x,y}" = "γ"

        ## Angular frequency
        ω = "2 • π • target_frequency"

        ## Amplitude squared
        r2 = "x² + y²"

        ## Output variable
        χ = "x"
    }
```

Figure 20: *Example of a Codyn file describing a simple Hopf oscillator represented mathematically by the system of equations* (5.3)

Dynamical systems are defined in `.cdn` files that can be directly loaded in the simulator. Example of a Hopf oscillator in Codyn syntax is presented in figure 20. The corresponding mathematical model is described by the system of equations below:

$$\begin{cases} \dot{x} = \gamma_x(\mu - r^2)x - \omega y \\ \dot{y} = \gamma_y(\mu - r^2)y + \omega x \end{cases} \tag{5.3}$$

with $r = \sqrt{x^2 + y^2}$ and initial conditions $x(0) = 0$, $y(0) = 1$.

Once the simulation is running, the dynamical system can receive inputs from the simulation and gives outputs once the new state of the system is computed. Figure 21 gives an overview of the simulation flow with all main steps of the run.

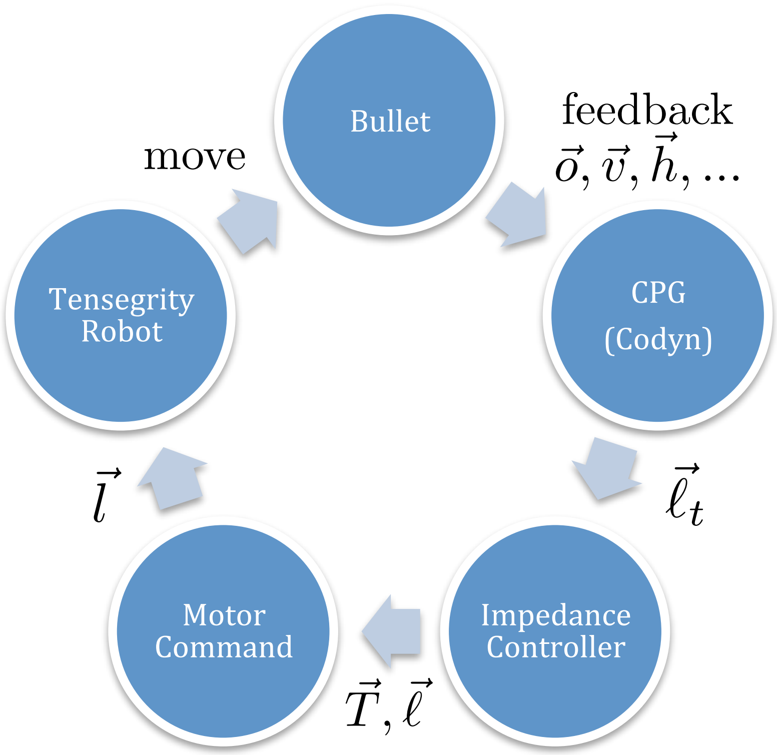


Figure 21: *Flow diagram schematizing the operating procedure of the simulator. The CPG, embedded in the Codyn library, receives feedback data from Bullet (orientation, speed, heights, etc.) and uses this information to compute the next state of the CPG and the new resulting output ℓ_t. This new target rest length is then given to the impedance controller that will compute the resulting target tension and determine the actual rest lengths ℓ that will be given to the motor command. The motor actuator compares the current rest length value to the new input and moves a distance l towards the target. The resulting changes affects the tensegrity robot that will move and interact with the Bullet environment. The cycle starts again.*

6 Control Algorithms

6.1 Challenges

Control of tensegrity structures represents a challenge in many different ways. The intrinsic compliance of the structure makes it unsuitable to most control algorithms used in classical robotics. In fact, their great compliance induces oscillatory motion and non-linearities that are usually avoided in traditional control design. The unusual structure that tensegrity robots usually exhibits, where actuators are embedded within the system itself is also another aspect of this complexity that control design has to deal with. And, ultimately, due to the relative novelty of this field of research, the scientific data addressing locomotion of tensegrity structures is rather limited. It is known however that tensegrity structures are present in Nature to a large extend, in both simple unicellular organism to complex vertebrate mammals that are able of very accurate, robust and energy efficient control. The structures of musculo-skeletal systems itself resemble closely the one of tensegrity systems. Nature can therefore be seen as an important source of inspiration for control strategies that can deal with and take advantage of these features novel to robotics. For that same reason, central pattern generators are the typical approach that can play a critical role in that direction.

Due to the fairly round shape of the tensegrity icosahedron, the most efficient way to make it move is obviously to make it tilt and, by pulling on the right strings at the right time, start storing a sufficient amount of rotational energy to make it roll smoothly. Interestingly, all known animals of size similar to the tensegrity robot, similar referring here on the scale at which identical physics phenomena occur, that use rolling as a mean of displacement achieve it in a passive way[33], for instance, as the wheel spider, by moving its limbs to give the body a circular shape and use potential energy to roll down a slope. As a result, no data coming from anatomical or biomechanical studies of rolling animals can be directly used as a source of inspiration for an active controller design.

We present in this section the different control strategies developed, the results obtained and, for each approach, we formulate a critical analysis of the performances. We start first by introducing the control principle used to move the tensegrity icosahedron, we then describe a reactive control algorithm used to move the robot over various types of terrains. We then explore different more classical approaches namely, inverse kinematics algorithms and a gyroscopic based control approach. Later on, we present two different types of CPG based on frequency adaptive oscillators used to learn a regular rolling gait and reproduce a locomotion pattern. At last, we explore the possibility of merging several control techniques to take advantage of the different approaches and improve the locomotion capabilities of the tensegrity robot.

6.2 Principle

The tensegrity structure is controlled using the torque created by the displacement of the center of mass from the ground contact surface as illustrated on figure 22. This is achieved in 2 different ways: the heading is determined by the displacement of the central payload relative to its rest position in the center of the structure and speed is determined by the actuation of the outer shell strings.

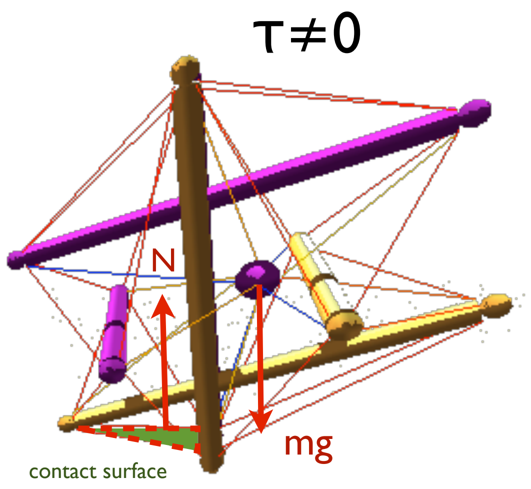


Figure 22: *torque τ applied to the whole structure by a displacement of the center of mass*

The inner strings were actuated using three different approaches, a reactive approach, an inverse kinematics approach and a CPG approach (see sections 6.3, 6.4 and 6.6 respectively). The outer strings where controlled in a different way. The goal of their dynamics is to reduce the contact surface with the ground. This affects the motion in several ways. First, it allows the creation of greater torques with the same payload displacement. Second, it enables a smoother rolling behavior of the whole structure, preventing shocks of the rods with the ground. As a consequence, playing with the reactivity of the lengths corrections can affect the global angular speed of the tensegrity structure by reducing or increasing the magnitude and direction of the ground contact forces.

The height of each string relative to the ground is computed using IR-like distance sensors located at the end of each rods (see figure 23). The height assigned to each string is computed as the average of the two end points height. These measurements are performed continuously during the simulations and the height parameter is constantly updated. If the tensegrity needs to move on uneven ground surfaces, it is important to know if the displacement occurs in the desired direction. Typically in the presence of a slope, the reduction of the ground contact surface can trigger the robot to roll down the slope without any control. In order to take this into account, we added a measure of the velocity. The velocity is computed using the global center of mass displacement between two consecutive time steps and the heading direction vector $\mathbf{v}$:

$$\text{velocity} = \mathbf{v} \cdot \frac{\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)}}{dt} \tag{6.1}$$

where $\mathbf{p}^{(n)}$ is the 3d position of the center of mass at time t_n and $dt = t_n - t_{n-1}$ is the simulation time step. With this method, the velocity is a scalar number and has a sign depending on the heading of the tensegrity (positive if heading in the desired direction and negative otherwise). It can thus be used as a feedback to influence the strings command. The strings rest lengths are computed using the following actuation rule:

$$\begin{cases} \dot{\ell}_i = w\left(\ell_0 + \min(h_i^2, h_0) - \ell_i\right) & , \text{ velocity} \geq 0 \\ \dot{\ell}_i = w\left(\bar{\ell} - \ell_i\right) & , \text{ otherwise} \end{cases} \tag{6.2}$$

where the velocity is computed according to (6.1), h_i is the height of string i as measured from the distance sensors, ℓ_i is its current rest length, ℓ_0, h_0 and $\bar{\ell}$ are constant parameters and $w \in \mathbb{R}_+$ accounts for the time scale at which lengths corrections occur. ℓ_0 and h_0 represent the offset rest length of the strings and the maximum height measurement, respectively. These parameters ensure that the value of ℓ_i does not becomes negative or higher than a certain threshold. The parameter $\bar{\ell}$ represent the default rest length of the strings that, if given as a command to all motors, puts the tensegrity in a stable position on the ground.

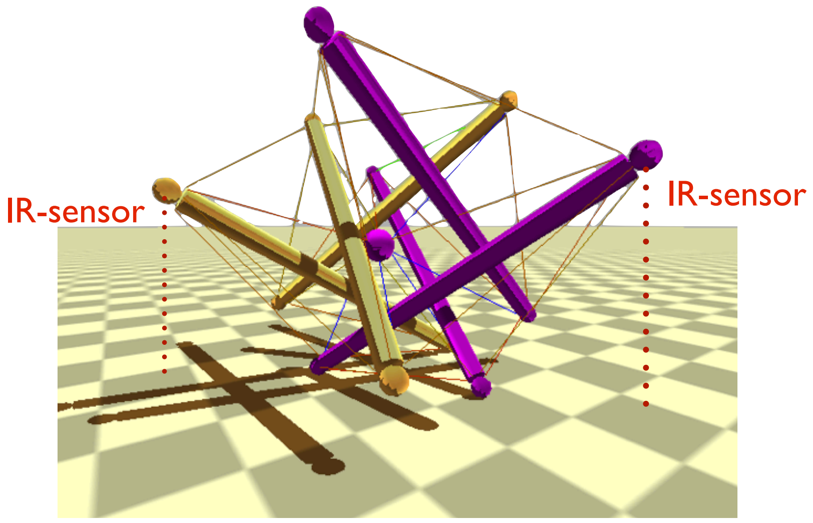


Figure 23: *Schematic view of the IR-sensors firing to the ground. Distance sensors are located at each rods end.*

With this actuation rule, two types of behavior occur, depending on the sign of the velocity measurement:

- velocity ≥ 0
  each string sees its rest length converge to a value between ℓ_0 and $\ell_0 + h_0$ depending on its height with respect to the ground. As a result, the robot will see its upper strings being relaxed while the lower strings will be more tensed, decreasing the space between the anchored struts close to the ground thus reducing the global contact surface of the tensegrity. As a consequence, the robot will be more likely to topple if a force is applied on its side.

- velocity < 0
  each string sees its rest length converge to $\bar{\ell}$ yielding to an homogeneous distribution of the forces within the structure. If $\bar{\ell}$ is chosen large enough, the structure will relax and augment its contact surface on the ground. As a result, the robot will increase its stability and become hard to tilt on either side.

Motor commands adjusting the strings rest length are performed through impedance controllers. The use of impedance controllers allows a smooth actuation of the string lengths and avoids creation of too high tensions within the string network[30][31][12]. The commands are computed using the following actuation rule:

$$T_i = T_0 + k(\ell_i - l) + \eta(V_i - V_0) \tag{6.3}$$

where T_i is the target tension, T_0 is the offset tension, k is the elastic coefficient, ℓ_i the current rest length, l the current length, η a viscosity coefficient and V_i and V_0 the actual and target velocity of the motors, respectively. In our simulation, we kept T_0, k, η and V_0 constant.

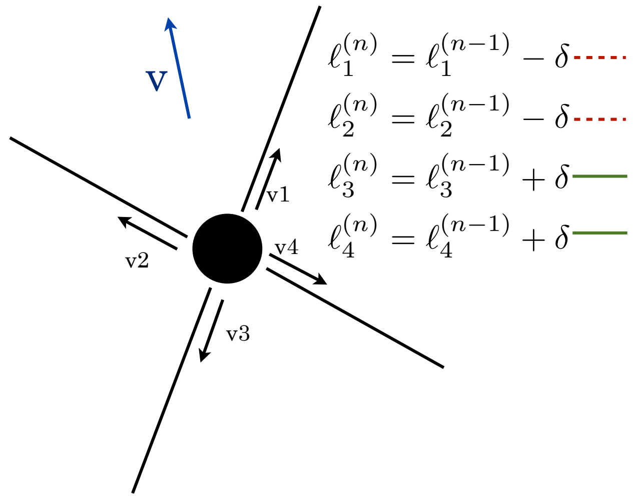
6.3 Reactive Method

Keep in mind that the only parameter that can be controlled is the rest length of each string. We denote by ℓ_i the rest length of the inner strings, l_i their actual lengths. The global heading direction is defined by the unit vector $\mathbf{v}$ and the orientation of each string is represented by the vectors $\mathbf{v}_i$. For each inner string we use the dot product $d_i = \mathbf{v} \cdot \mathbf{v}_i$ as feedback to control the position of the payload as follows:

$$\dot{\ell}_i = (\ell_0 + d_i\gamma - l_i)w \tag{6.4}$$

$$\ell_i(0) = \ell_0 \tag{6.5}$$

where the weight w determine the reactivity of the system and $\gamma < 0$ is a fixed parameter. Thus, without any external perturbation, the system has a stable equilibrium position at $\ell_0 + d_i\gamma$. With this implementation, the strings that have their orientation aligned with the global heading see their rest length reduced and the strings pointing in the opposite direction are elongated. The global result is a displacement of the payload in the direction of the heading vector. See figures 24 and 25 below for a graphical representation of the method.



V
$\ell_1^{(n)} = \ell_1^{(n-1)} - \delta$
$\ell_2^{(n)} = \ell_2^{(n-1)} - \delta$
$\ell_3^{(n)} = \ell_3^{(n-1)} + \delta$
$\ell_4^{(n)} = \ell_4^{(n-1)} + \delta$
v1
v2
v3
v4

Figure 24: *computation of the new rest lengths according to the strings individual orientations $\vec{v}_i$ (time $t^{(n-1)}$). Length corrections δ are computed according to vectors $\vec{v}_i$.*

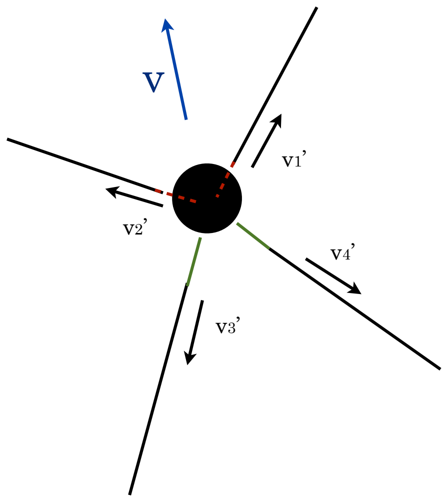


Figure 25: *the resulting effect is a displacement of the central payload in the desired direction $\vec{v}$ (time $t^{(n)} = t^{(n-1)} + dt$). The string length modification is indicated by the colored lines, dashed red if reduced and green if elongated.*

Note that the heading direction $\mathbf{v}$ can be chosen arbitrarily and can be adjusted dynamically, enabling the steering of the robot. Furthermore, it can be chosen in an absolute frame of reference or relative to the tensegrity orientation.

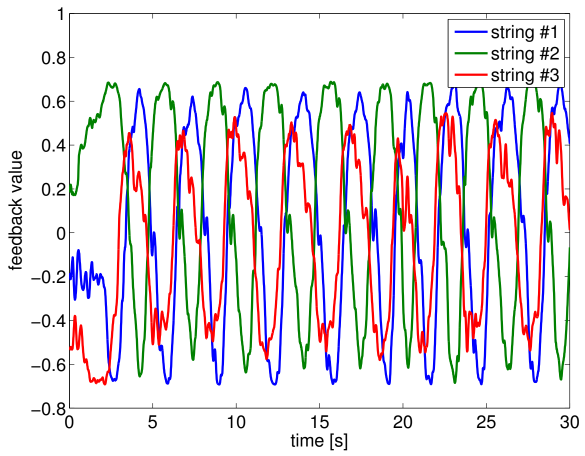


Figure 26: *feedback signal for the first 3 inner strings as a function of time. Positive values of the feedback lead to a shortening of the strings and vice versa for negative values.*

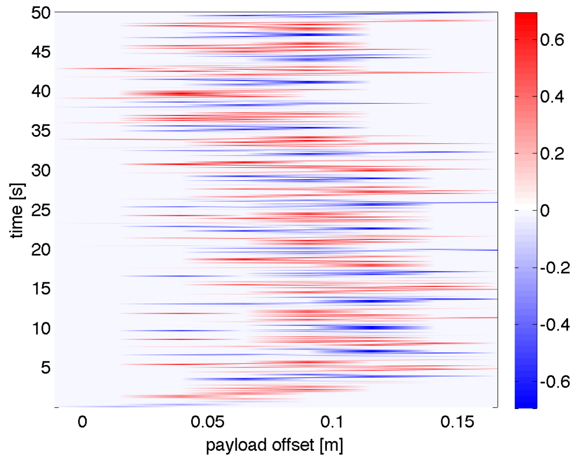


Figure 27: *intensity of the feedback signal of the second inner string as a function of the payload position relative to the center of mass and time*

Figures 26 and 27 show the value over time of the feedback signal obtained using the reactive control method. Several conclusions can be taken from these plots. First of all, it appears clearly that the feedback signal has a very stable periodicity over time, independently of the strings initial positions. This property is especially visible on figure 26. Furthermore, on figure 27, where we plotted in addition the relative position of the payload with respect to the center of mass, we can observe that the payload stays at all time within a certain distance to the center of mass, as desired to create enough torque to flip the robot. Again, the same repetitive pattern appears on the time axis.

With this implementation, we were able to obtain stable and smooth rolling gaits allowing the tensegrity to roll up to 1m/s (1 body length per second) over flat terrain. The robot could also handle slopes up to 8°, bumpy terrain, obstacles and collisions[2].

Possible improvements:
The main disadvantage of the reactive method as presented here is the large amount of feedback required to actuate the motors. This can be a serious complication when it comes to designing the real tensegrity hardware. This justifies the research for simpler methods, based on the same physical principle but requiring less feedback information.

[2]videos of the controller in action can be found on:
http://biorob.epfl.ch/files/content/users/181078/files/reactive_controller.avi

6.4 Inverse Kinematics Method

6.4.1 First order IK

The inverse kinematic (IK) method presented here can be an alternative to the reactive method. The idea is to apply the classical transpose Jacobian algorithm used in traditional robotics to control the 3d position of the central payload. Because of the intrinsic compliance of the tensegrity, the geometry of the robot is changing over time and an important modification that we have to add to this method is the dynamical computation of the Jacobian matrix to account for these changes of configuration.

The idea of this method is to define the 3d position of the central payload $\mathbf{p} = (p_1, p_2, p_3)$ as a function of the strings rest lengths $\boldsymbol{\ell} = (\ell_1, ..., \ell_n)$. Mathematically, we can write $\mathbf{p} = \mathbf{p}(\boldsymbol{\ell})$. As a consequence, in first approximation, a small displacement $\delta\mathbf{p}$ of the payload can be written as:

$$\delta p_i \approx p_i(\boldsymbol{\ell}^{(0)}) + \sum_{j=1}^{n} \frac{\partial p_i(\boldsymbol{\ell}^{(0)})}{\partial \ell_j} \delta \ell_j \qquad , \ i = 1, 2, 3$$

If we define now J the Jacobian matrix as $J(\mathbf{p}) = \left[\frac{\partial p_i}{\partial \ell_j}\right]_{ij}$, we can rewrite the above as:

$$\delta\mathbf{p} = \mathbf{p}^{(0)} + J\left(\mathbf{p}^{(0)}\right)\delta\boldsymbol{\ell}$$

This equation relates the payload displacement $\mathbf{p}^{(0)} - \delta\mathbf{p}$ to the change of rest lengths $\delta\boldsymbol{\ell}$. We can therefore determine what should be the change of the rest lengths to have the payload move in a certain direction. This can be expressed as:

$$\delta\boldsymbol{\ell} = J^{-1}(\mathbf{p}^{(0)})\left(\mathbf{p}^{(0)} - \delta\mathbf{p}\right)$$

However, computing the inverse of the Jacobian matrix is a very costly and complicated operation. Instead, we use the mathematical trick known as transpose Jacobian method where we replace the inverse of J by its transpose to obtain:

$$\delta\boldsymbol{\ell} = \alpha J^T(\mathbf{p}^{(0)})\Delta\mathbf{p} \tag{6.6}$$

with $\alpha \in \mathbb{R}_+$ and $\Delta\mathbf{p} = \left(\mathbf{p}^{(0)} - \delta\mathbf{p}\right)$. This is relation allows us to compute the motor commands (i.e., the change of rest lengths) to move the payload in any direction we want. The main issue relies in the computation of the Jacobian matrix. In a static system made of bars and joints only, this matrix is constant and can be computed very precisely using appropriate mathematical tools. But in the case of a very compliant tensegrity structure, the coefficients of the Jacobian matrix are time dependent. The idea is thus to recompute the matrix coefficients dynamically using a finite difference method:

$$J(\mathbf{p}) \approx \left[\frac{p_i^{(n)} - p_i^{(n-1)}}{\ell_j^{(n)} - \ell_j^{(n-1)}}\right]_{ij} \tag{6.7}$$

Using equation (6.6) and (6.7) we are now able to move the payload in any direction with a feedback only on the payload position in space and each string's rest length.

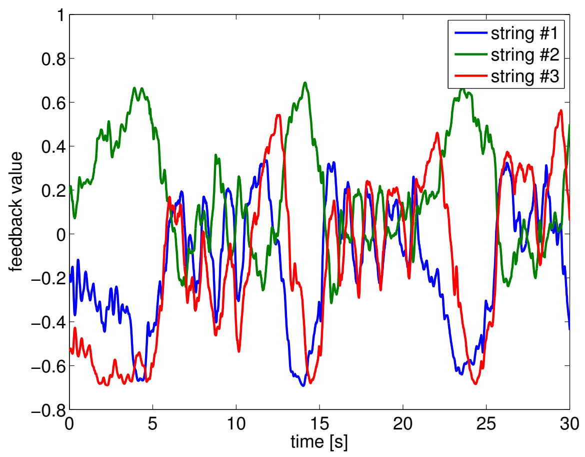


Figure 28: *feedback signal for the first 3 inner strings as a function of time. Positive values of the feedback lead to a shortening of the strings and vice versa for negative values.*

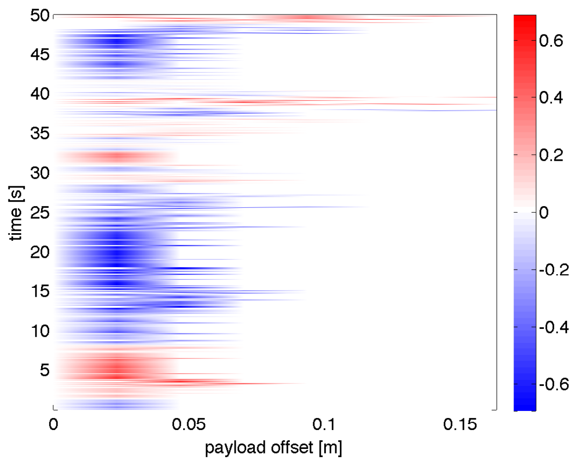


Figure 29: *intensity of the feedback signal of the second inner string as a function of the payload position relative to the center of mass and time*

Similarly to what has been presented in the previous section, figures 28 and 29 above illustrate how the feedback signal varies over time and as a function of the payload position. These plots, when compared to figures 26 and 27 emphasize the more irregular behavior of the inverse kinematics method with respect to the reactive method. The feedback signal is indeed much less regular and the distance between the payload and the center of mass is significantly smaller at any given time.

The results obtained with these methods allow the tensegrity to roll up to 0.57m/s on flat terrain. However the rolling gait obtained is much more chaotic than with the reactive method. Moreover, some sudden and large corrections can occur unexpectedly. These unstable features come form the non-linearities present in the tensegrity structure that cannot be captured by this too simple first order method[3].

[3]videos of the controller in action can be found on:
http://biorob.epfl.ch/files/content/users/181078/files/inverse_kinematics_controller.avi

6.4.2 Second order IK

Again, the main drawback of the previous IK method are numerical inaccuracies due to the first order approximation to compute the relation between $\delta\mathbf{p}$ and $\delta\boldsymbol{\ell}$. Since tensegrity structures are by definition very compliant and oscillatory, higher order terms should be added in order to account for these non-linear behaviors. A possible improvement consists of extending the Taylor expansion to higher order terms, yielding (here up to 2nd order terms):

$$\delta p_i \approx p_i(\boldsymbol{\ell}^{(0)}) + \sum_{j=1}^{n} \frac{\partial p_i(\boldsymbol{\ell}^{(0)})}{\partial \ell_j}\delta\ell_j + \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^2 p_i(\boldsymbol{\ell}^{(0)})}{\partial\ell_j\partial\ell_k}\delta\ell_j\delta\ell_k \qquad , \ i = 1,2,3$$

or

$$\delta p_i \approx p_i(\boldsymbol{\ell}^{(0)}) + \left(J(\mathbf{p}^{(0)})\delta\boldsymbol{\ell}\right)_i + \frac{1}{2}\delta\boldsymbol{\ell}^T H\left(p_i(\boldsymbol{\ell}^{(0)})\right)\delta\boldsymbol{\ell} \qquad , \ i = 1,2,3 \tag{6.8}$$

where $H(p_i) = \left[\frac{\partial^2 p_i}{\partial\ell_j\partial\ell_k}\right]_{jk}$ is the Hessian matrix associated to p_i.

Considering (6.8), we define additionally $\Delta\mathbf{p} = \delta\mathbf{p} - \mathbf{p}(\boldsymbol{\ell}^{(0)})$ and $\mathbf{f}(\delta\boldsymbol{\ell};\Delta\mathbf{p})$ as

$$f_i(\delta\boldsymbol{\ell};\Delta\mathbf{p}) = \left(J(\mathbf{p}^{(0)})\delta\boldsymbol{\ell}\right)_i + \frac{1}{2}\delta\boldsymbol{\ell}^T H\left(p_i(\boldsymbol{\ell}^{(0)})\right)\delta\boldsymbol{\ell} - \Delta p_i \qquad , \ i = 1,2,3$$

The idea is, given a desired displacement of the payload in the 3d space $\Delta\mathbf{p}$, to find the corresponding lengths change $\delta\boldsymbol{\ell}$ that will lead to this displacement. This correspond to finding $\delta\boldsymbol{\ell}$ such that $\mathbf{f} = \mathbf{0}$. Note however that this system is overdetermined, non linear and might not possess a real solution. In order to bypass these difficulties, we can compute an approximation of the solution using a quasi-newtonian iterative method. The idea is to start with an arbitrary position, e.g. $\delta\boldsymbol{\ell}_0 = \mathbf{0}$, and compute compute the next iteration as:

$$\delta\boldsymbol{\ell}_{k+1} = \delta\boldsymbol{\ell}_k - J_k^{-1}\mathbf{f}(\delta\boldsymbol{\ell}_k;\Delta\mathbf{p}) \tag{6.9}$$

until convergence is achieved. J^{-1} denotes here the Moore-Penrose pseudo inverse of the Jacobian defined by $J = \left[\frac{\partial f_i}{\partial(\delta\ell_j)}\right]_{ij}$.

Note that this matrix is not the same as the one appearing in the Taylor expansion of $\mathbf{p}(\boldsymbol{\ell})$. The pseudo inverse is computed using a singular value decomposition of the Jacobian matrix $J = U\Sigma V^*$ where U is a unitary matrix, Σ is a rectangular diagonal matrix containing the singular values of J, and V^* (the conjugate transpose of V) is a second unitary matrix. Once the three matrices have been computed, the pseudo inverse can be found by a simple matrix multiplication $J^{-1} = V\Sigma^{-1}U^*$ with $\Sigma^{-1} = \mathrm{diag}(1/\sigma_i \ , i = 1, ..., n)$. An example of the convergence of the numerical method is illustrated on figure 30.

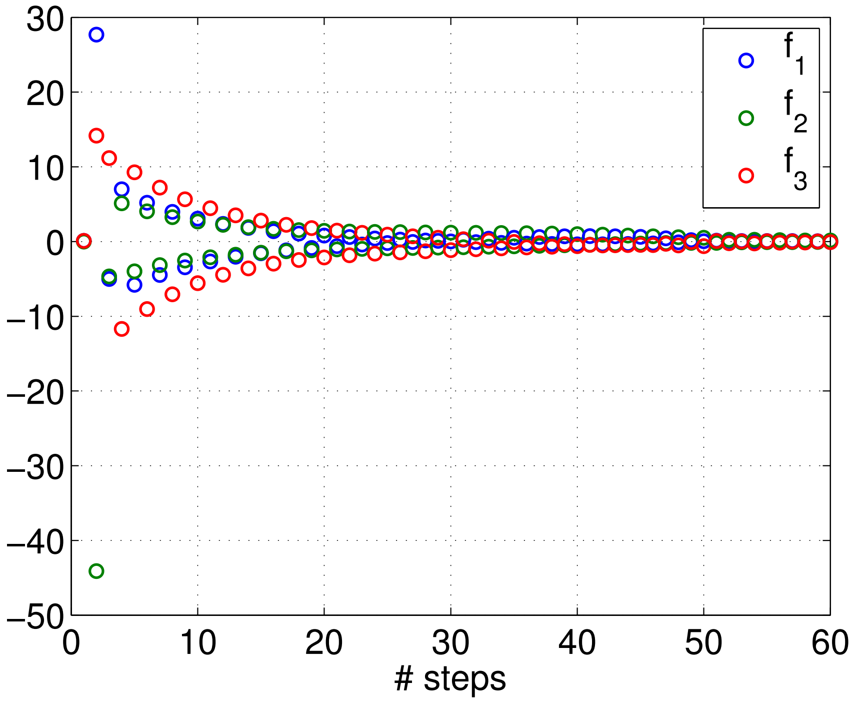


Figure 30: *Example of the convergence in ∼50 iterations of the quasi-Newton method for* $\Delta \mathbf{p} = (0.1, 0, 0.1)$*, starting from* $\delta \boldsymbol{\ell}_0 = \mathbf{0}$ *(plot of the 3 coordinates of* $\mathbf{f} = (f_1, f_2, f_3)$*).*

Note that this method can theoretically be extended to any arbitrary order. We can indeed imagine repeating the procedure and considering a kth order Taylor expansion of the position and find the values of $\delta \boldsymbol{\ell}$ such that $\mathbf{f} = \mathbf{0}$ using the same quasi-Newton method. Note however that the higher the k, the more overdetermine becomes the root finding problem.
In the same idea, we can also define the position as a function of the rest length and its derivative $\mathbf{p} = \mathbf{p}(\boldsymbol{\ell}; \dot{\boldsymbol{\ell}})$ and apply the same method to obtain the values of $\delta \boldsymbol{\ell}$. Note that in this case, the variations of the velocities $\delta \dot{\boldsymbol{\ell}}$ are considered as parameters and not variables since we supposed that only the rest length can be used as a control parameter.
The implementation of the aforementioned methods are however not presented in this work as we want to keep the focus of this study as broad as possible and not narrow down to the development of a specific inverse kinematics method.

Possible improvements:
Of course this system is then much more complicated than the first order approximation and requires therefore more complex and costly numerical techniques to be solved. Moreover, convergence cannot be guaranteed as solutions may not exist or numerical instabilities might drive the system out of the convergence region. Note also that in real hardware these additional computations may lead to higher energy consumption and would require a greater computational power in order to solve these equations in real time. Another major drawback is the fact that the payload position depends only indirectly from the strings rest lengths. As a result, the inverse kinematics might be ”mistaken” if two or more antagonist motor commands are executed at the same time. Indeed, in this particular case, a motor might reduce a given string’s rest length but still measure the payload moving away from the string’s anchor point due to the other motors actions. This effect leads to contradictory motor commands once the inverse kinematics algorithm is executed and, unfortunately, these errors cannot be solved by better numerical accuracy or even higher order methods.
A possible improvement that might address this issue would be to reduce the number of strings that can be actuated. This would reduce both the complexity of the problem and the mistakes happening when antagonist strings are actuated. However these errors would always remain present at a certain level as they are inherent to the tensegrity physical properties and to the control and feedback method we adopted.

6.5 Gyroscopic Method

An easy way to tackle the orientation issue is to use the physical properties of gyroscopes to create torques in any given direction and keep track of a given orientation. The idea of the method is to construct a gyroscope inside the central payload and change its orientation in order to create a torque on the whole structure. The torque will tilt the payload, pulling on the inner strings of the structure and thus creating a force on the ground pulling the tensegrity forward. See figures 31 and 32 for a schematic explanation.

This method has the advantage of requiring no actuation of the inner muscles, all the forces resulting from the payload twisting motion. Furthermore, the heading of the tensegrity can be chosen quite easily as the gyroscope will try to keep its orientation if no external force is applied. The control of the motion is also simplified as we reduce the problem of actuating 12 strings to choosing the pitch, roll and yaw of the payload. A detailed physical analysis of this method can be done on the model of the torsion pendulum.

With this implementation and the current physical shape (no modifications to the payload size or inner strings tensions), we were able to achieve displacements of more than 53m over 60 seconds of simulation (0.88m/s) by applying a torque of 35 Nm[4]. Note however

[4]videos of the controller in action can be found on:
http://biorob.epfl.ch/files/content/users/181078/files/gyroscopic_controller.avi

that irregular terrain configurations such as bumps and slopes cannot be handle as well as with the reactive method as the force created by the twisting of the payload is limited by the payload dimensions and the mass of the gyroscope. Furthermore, gyroscopic effects are hard to accurately simulate and, to simplify the modelization, we artificially add a constant force to twist the payload, reproducing the effect of a rotating mass being pulled or pushed away from its rotation axis. It is also worth remarking that the torque necessary to move the structure is relatively important thus requiring heavy and large gyroscopes for a real physical implementation. These issues make the gyroscopic control method less likely to be a feasible actuation method for the real icosahedron tensegrity robot.

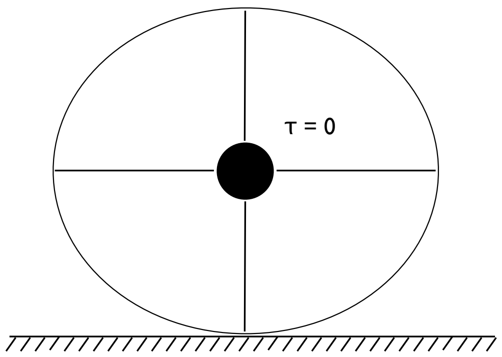


Figure 31: *Stable rest situation, no torque is applied. The tensegrity is schematized by the circular shape, the central payload is connected by 4 inner muscles.*

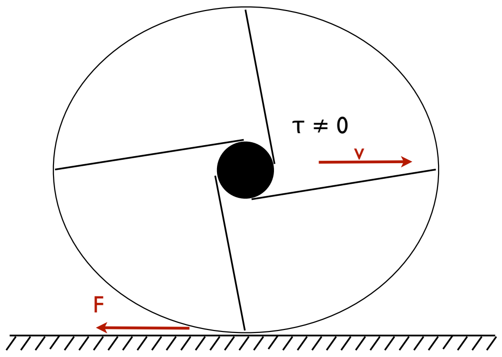


Figure 32: *Once a torque is applied, the central payload is tilted and pull on the inner strings. This motion creates a force F on the ground moving the tensegrity forward.*

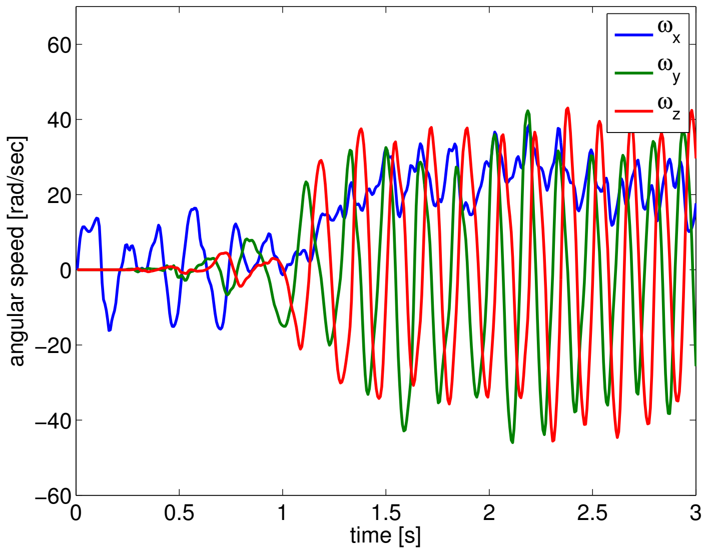


Figure 33: *Three components of the angular velocity ω of the payload in the first seconds of simulation. The maximum angular velocity corresponds to a frequency of 6.4Hz.*

Figure 33 highlights an interesting phenomenon that occurs when the torque applied to the payload exceeds a certain threshold value. In the first few seconds of simulation, the payload starts to spin around its Ox and Oy axis creating a regular movement swinging the payload back and forth at a frequency of ca. 6.4Hz. This oscillatory motion greatly improves the rolling as it enables the creation of short but large forces pulling the strings and making the whole structure tilt on its side. Nonetheless, while the gyroscope based control appears to be a potentially good actuation method, it is difficult to prove that the same type of motion would also appear in a real physical tensegrity robot, due to physical limitations within the physics simulator. This makes the gyroscope based method also highly likely to fail as a real controller.

Possible improvements:
Of course the major drawback of tis method is the requirement for relatively heavy and voluminous gyroscopes that have to be inserted in the payload structure. Also this method is heavily dependent on the diameter of the payload, the bigger the diameter, the greater the torque we can apply to the structure. As a result, the size of the payload would have to be optimized to match both the motion and security requirements. Moreover, the tension of the inner strings can be tuned to adapt to the magnitude of the force created on the ground.

6.6 CPG and Adaptive Frequency Oscillators Method

Another approach is the storage of a stable gait obtained in simulation inside a central pattern generator (CPG). The idea is to use as much feedback as needed in order to get a smooth motion and then store the resulting periodic commands as a stable limit cycle of a CPG. Once this process is done, the tensegrity can be driven with the CPG output with much less feedback than before.

6.6.1 Arbitrary Waveform Oscillator

In order to be able to store and recreate different types of signals, we use a so called arbitrary waveform oscillator (AWO)[34] which is a form of morphed oscillators developed by Ajallooeian et al.[35]. The dynamical system driving the CPG is presented below. As usual, we denote the muscle rest length by ℓ_i.

$$\dot{\ell}_i = \gamma(g(\varphi_i) - \ell_i) + \frac{dg(\varphi_i)}{d\varphi_i}\,\dot{\varphi}_i \tag{6.10}$$

$$\dot{\varphi}_i = \omega_i + \sum_j \sin(\varphi_j - \varphi_i - \phi_{ij}) \tag{6.11}$$

where the function $g(\varphi)$ is a periodic and derivable function, $\gamma \in \mathbb{R}_+$ is a parameter accounting for the systems time scale, ω_i is the pulsation of the periodic output and $\phi_{i,j}$ is the desired shift between signal i and j. Without any external perturbations, the system converges to $\ell_{i,\infty} = g(\varphi_i)$ where $\varphi_i = \omega t$ and $\varphi_i - \varphi_j = \mod(\phi_{ij}, 2\pi)\ \forall (i,j)$. In our case, for geometrical reasons and for simplicity we choose $g(\varphi) = A\sin(\varphi)$, $A \in \mathbb{R}^*$.

Furthermore, in order to synchronize to a periodic input signal $f_i(t)$ of pulsation ω_{in} and mean value $\bar{f}_i$, we add the following equation, inspired by Righetti et al. work on adaptive frequency Hopf oscillators[26], driving the time evolution of ω_i during the learning phase:

$$\dot{\omega}_i = \epsilon(f_i(t) - \bar{f}_i)g(\varphi_i + \pi/2) \tag{6.12}$$

where $\epsilon \in \mathbb{R}_+$ is a parameter accounting for the time scale. When the learning phase is completed, the value of ω_i is held constant by setting $\dot{\omega}_i = 0\ \forall i$.

On figures 34 and 35, we can observe the convergence of the CPG pulsation ω to the input signal pulsation ω_{in} in an ideal case where the input signal is a simple sinusoidal function $f(t) = \sin(2t)$.

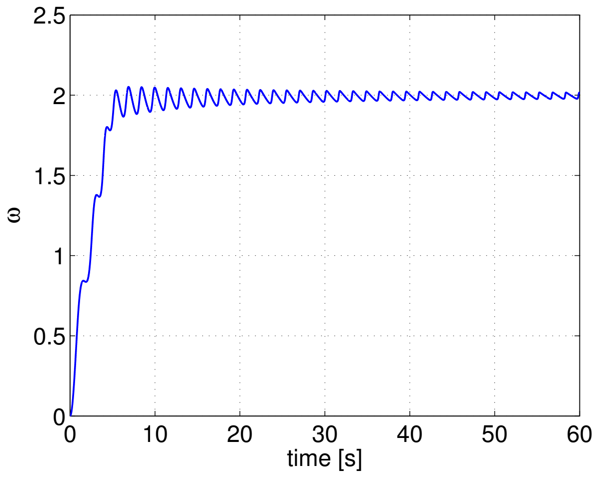


Figure 34: *Time plot showing the convergence of pulsation ω towards the input signal $\omega_{in} = 2$ during the learning phase*

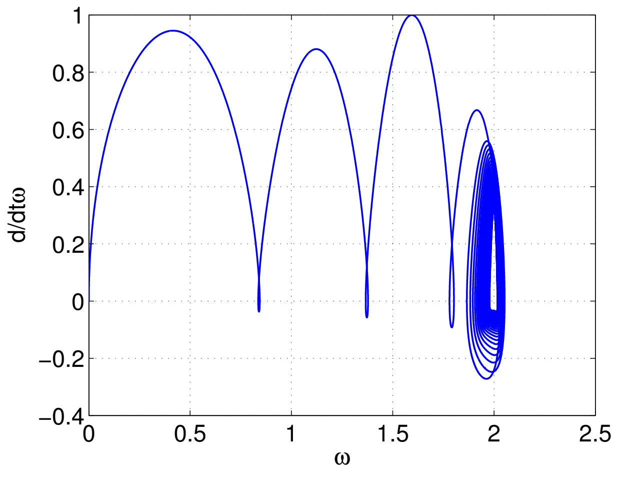


Figure 35: *Phase plot showing the convergence towards a stable fixed point at $\omega = 2$*

On figures 36 and 37, we can see the synchronization of the CPG output (red) to the learning signal (blue) during the learning phase. The signal here corresponds to the value of the dot product d_i when the tensegrity is driven using only the reactive method (see section 6.3). Each string possesses its own CPG that learns and creates its own output. The perturbations and irregularities appearing between 0 to 15 seconds are due to a chaotic motion where the tensegrity has not yet found a stable rolling pattern. After about 20 seconds, one can observe that the input signal becomes much more regular and periodic, this correspond to a stable rolling gait that can be well captured by the CPG as a sine function. Note that the mean of the signal $\bar{f}$ is dynamically updated in order to account for signals that are constantly shifted above and below 0.

Once the learning phase is completed, the parameters of the CPG are kept constant and the CPG output $g(\varphi_i)$ is then used to command the rest lengths ℓ_i. If the CPG is able to mimic perfectly the previous commands, we would expect to see the tensegrity rolling with the same gait as with the reactive method but without the need for any orientation feedbacks. Using the phase shift parameters ϕ_{ij} (see equation (6.11)), we can also force a delay between two (or more) oscillators. In some cases, for instance for diametrically opposed strings, it is obvious that a phase shift of π would be appropriate (see figure 39 for a graphical representation of the whole oscillator network). The value of the coefficients ϕ_{ij} can also be extracted by a careful analysis of the signals shift during the learning phase and then be assigned to the corresponding coefficients.

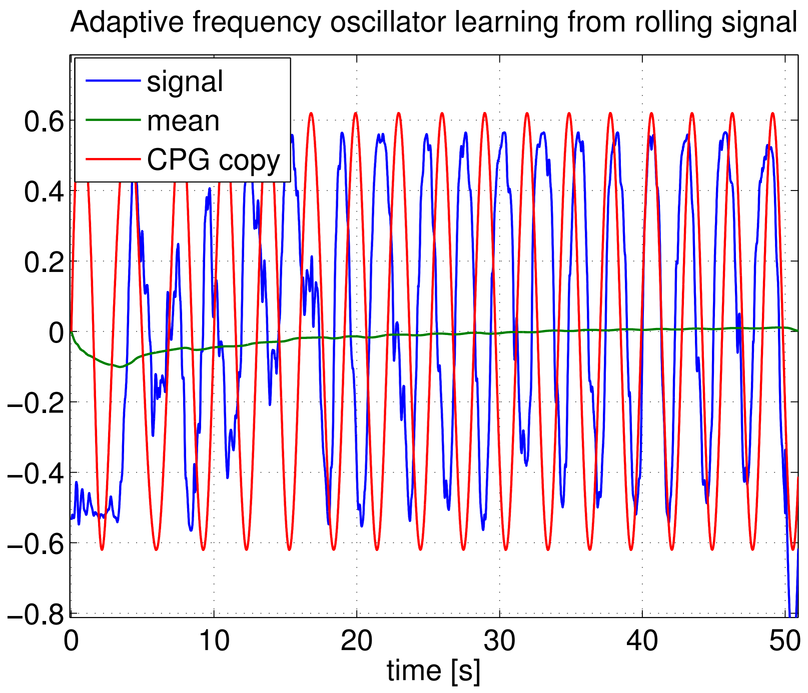


Figure 36: *dynamic synchronization of CPG signal (string #3)*

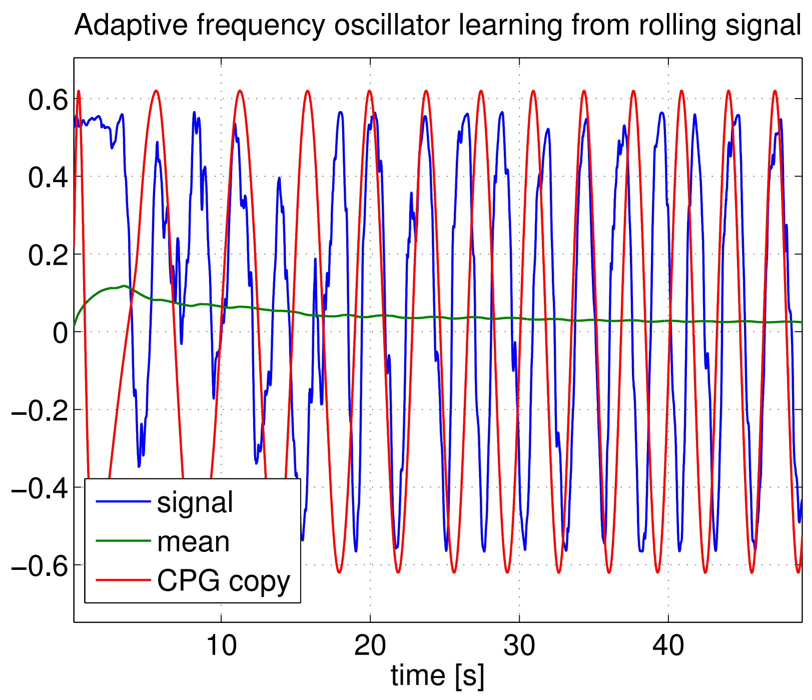


Figure 37: *dynamic synchronization of CPG signal (string #4).*

This method has the advantage of requiring no (or almost no) feedback and thus only a very small amount of computations. It can therefore be implemented easily on real hardware. However, it is important to note that the dynamical system runs on a much larger time scale than the perturbations disturbing the system. A tensegrity driven only by a CPG would then only, in the best case, have a stable rolling gait on a flat, obstacle free terrain. This effect is even accentuated by the inherent non-linear response that tensegrity structures possess. As a result, it is necessary to include also a second control method that can work on this smaller time scale and give an appropriate response to these external perturbations.

Simulations were performed where the tensegrity was driven during 50 seconds using the reactive method. This first phase was used to synchronize the CPG output with the input signal given to the robot (figures 37 and 36). Then, the controls were shifted to be driven only by the CPG output and the learning phase was stopped. No further feedback was used to drive the robot's motion.
Results showed that the tensegrity could continue rolling for a few seconds but quickly rolled on the side changing completely the orientation and being therefore unable to move smoothly using the signal learned previously. After some time and random displacements, the robot can fall back in the right position and keep rolling for some more time but eventually falls back in an unwanted position. With this type of motion, only much small travel distances were achieved compared to the reactive and inverse kinematics method.

Possible improvements:
Two main drawbacks can be pointed out from our experiments. First, the tensegrity can easily be perturbed and roll on the side without affecting the motor commands in any way and, second, the synchronization of CPG signals is very sensitive to noise in the input signal, resulting in non-negligible frequency drift in some strings actuators (see e.g. figure 36 and its effect on the robot's trajectory on figure 38). As a result, the tensegrity cannot keep rolling on a stable gait for more than a few seconds. Even on flat and obstacle free terrain, an open loop control is not sufficient to drive the robot smoothly. A good improvement would be to mix the CPG approach with one of the technique discussed above. As stated previously, the CPG could drive the global rolling motion, occurring on a large time scale, while the corrections, occurring on a much smaller time scales would be done through an inverse kinematics algorithm or a reactive control.

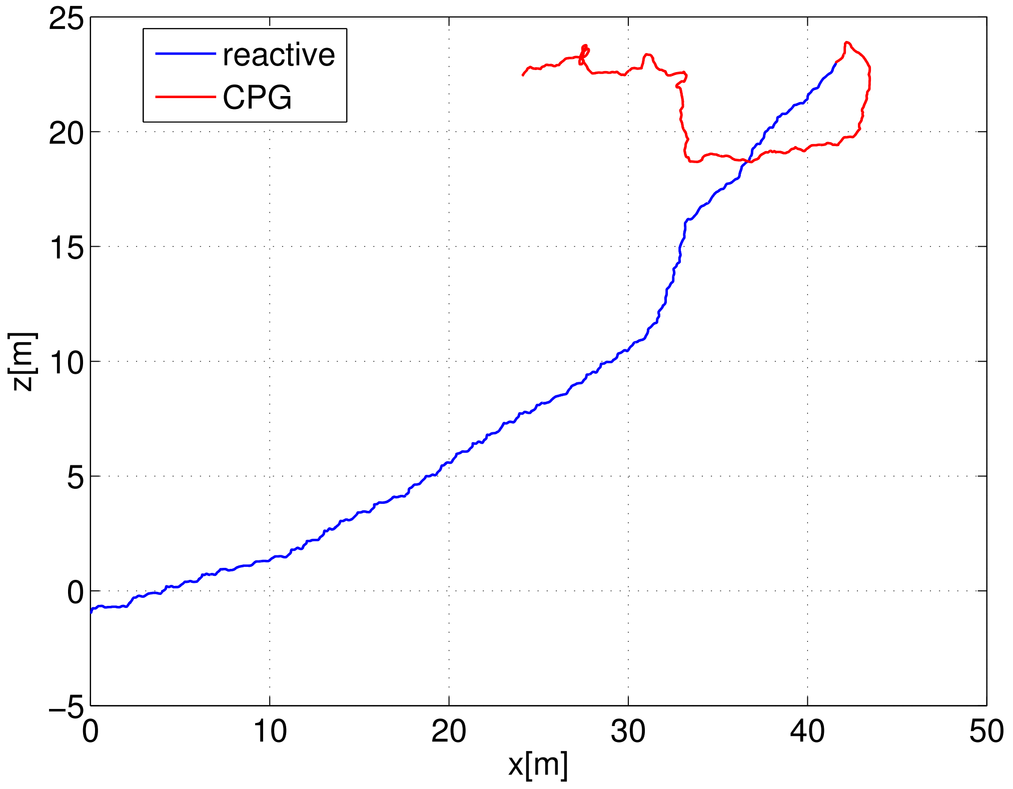


Figure 38: *2D trajectory of the tensegrity (seen from above). The blue curve represents the trajectory while the robot is driven by the reactive control algorithm and the CPG is in the learning mode (50 seconds). The motion is very regular and the heading is maintained throughout the whole period. The red trajectory represent the path traveled once the CPG controller takes over (40 seconds). Due to the open loop implementation, the heading of the robot cannot be maintained and the tensegrity ends up rolling in random directions.*

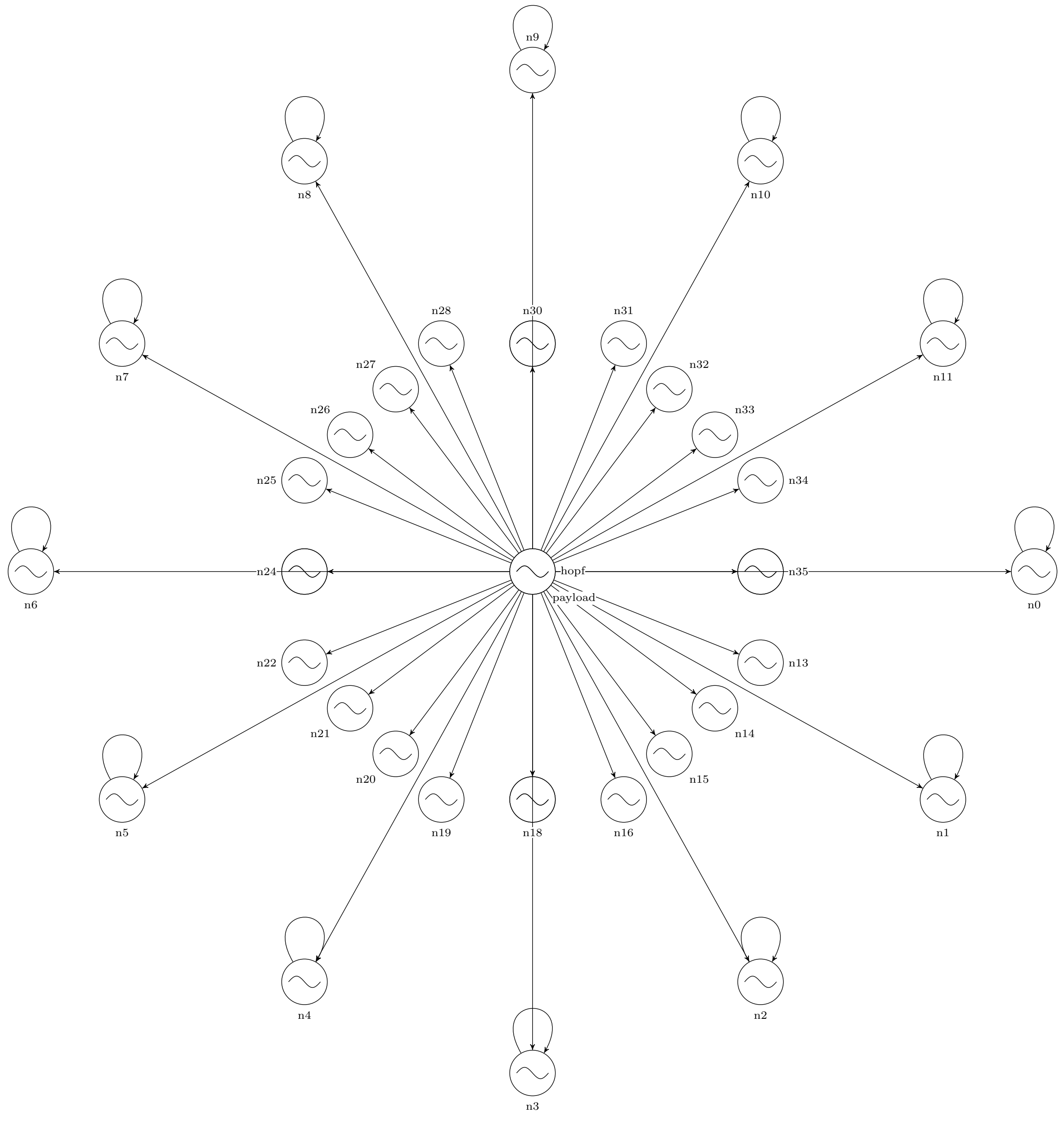


Figure 39: *Schematic view of the whole oscillators network. Nodes 0-11 represent the oscillators associated to the inner strings, nodes 12-35 represent the oscillators associated to the outer strings. The central node represents the payload, where the orientation and speed is computed and then shared to the other nodes through their common links (arrows). (Figure created using the Codyn framework[32]).*

6.6.2 Hopf Oscillator

Due to the ball-like structure of the tensegrity, the signals we want to copy during the rolling phase can be assumed as sinusoids. Thus, instead of AWOs, we can consider using Hopf oscillators[36] defined by:

$$\dot{x} = \gamma(\mu - (x^2 + y^2))x - \omega y \tag{6.13}$$

$$\dot{y} = \gamma(\mu - (x^2 + y^2))y + \omega x \tag{6.14}$$

where γ is a time constant, μ is the target frequency and ω the target pulsation of the signal. This dynamical system can be adapted to synchronize to any periodic input signal $f(t)$. The resulting adaptive frequency Hopf oscillator is defined by the following set of equations:

$$\dot{x} = \gamma(\mu - (x^2 + y^2))x - \omega y + \epsilon f(t) \tag{6.15}$$

$$\dot{y} = \gamma(\mu - (x^2 + y^2))y + \omega x \tag{6.16}$$

$$\dot{\omega} = -\epsilon f(t)\frac{y}{x^2 + y^2} \tag{6.17}$$

As with the AWO, we can now record the periodic signal during a learning phase and store it as a stable limit cycle of the Hopf oscillator. This signal can later be used to drive the tensegrity.

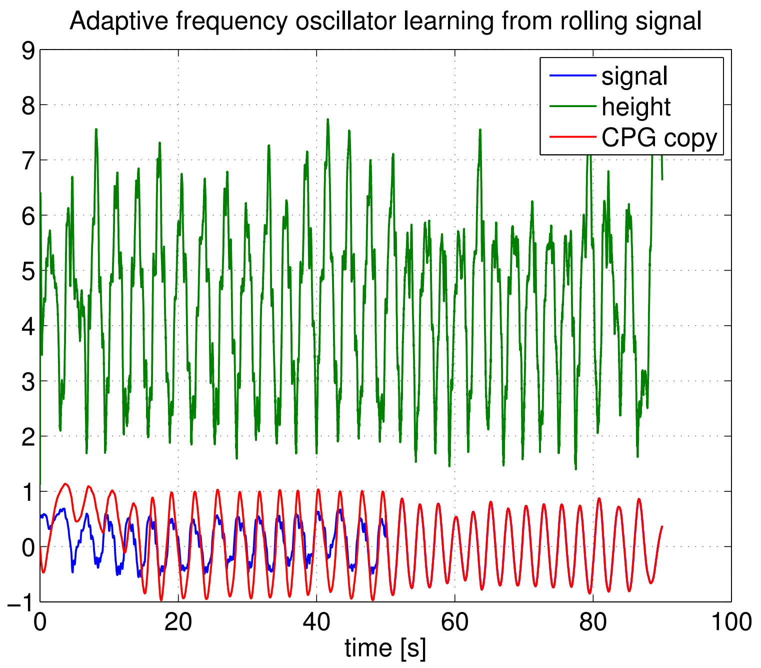


Figure 40: *Adaptive Hopf oscillator synching its output signal (red) to an periodic input signal (blue). The dynamical system can reproduce a sinusoid with the same frequency and phase once the learning signal is removed (50-90 sec).*

On figure 40 above, we can notice a coupling between the height signal (green), i.e. the height of the muscle connecting the payload with the learning signal (red). This coupling can be enforced by adding a new term to the dynamical system (6.13), (6.14). This coupling will allow the oscillator to synchronize with the real tensegrity motion and enable the system to deal with ground contact and unexpected perturbations that may occur during the rolling. We denote by $h(t)$ the height signal fed back by the IR-sensors, the resulting dynamical system used for rolling is then given by:

$$\dot{x} = \gamma(\mu - (x^2 + y^2))x - \omega y - kh(t) \tag{6.18}$$

$$\dot{y} = \gamma(\mu - (x^2 + y^2))y + \omega x \tag{6.19}$$

with $k \in \mathbb{R}_+$ a coefficient. Adding the coupling term improves significantly the distance that the tensegrity can roll, see figures 41 and 42 for a numerical example.

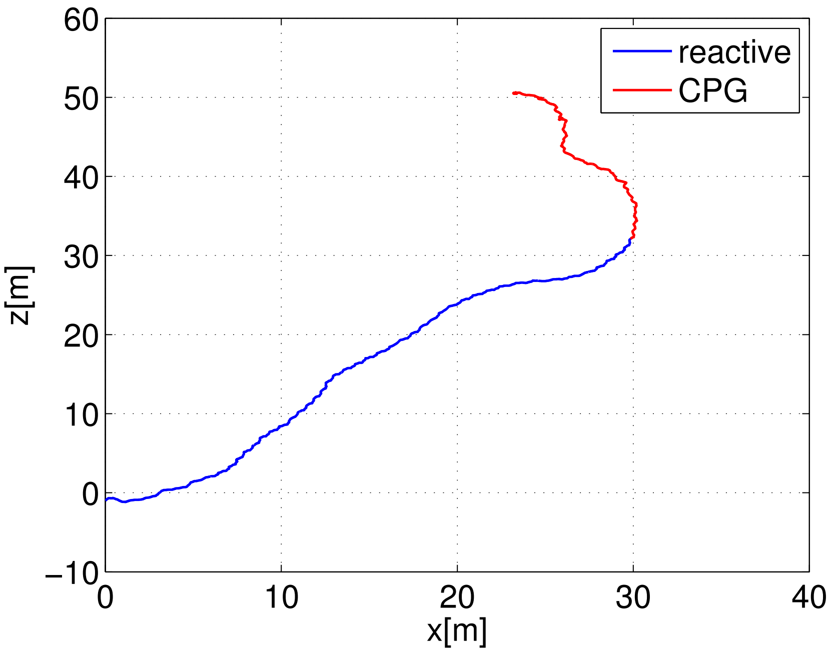


Figure 41: *2D trajectory of the tensegrity (seen from above). The blue curve represents the trajectory while the robot is driven by the reactive control algorithm and the CPG is in the learning mode (50 seconds). The motion is very regular and the heading is maintained throughout the whole period. The red trajectory represent the path traveled once the CPG controller takes over (40 seconds).*

The CPG is here driven purely open-loop and the tensegrity travels a total distance of 56.8m.

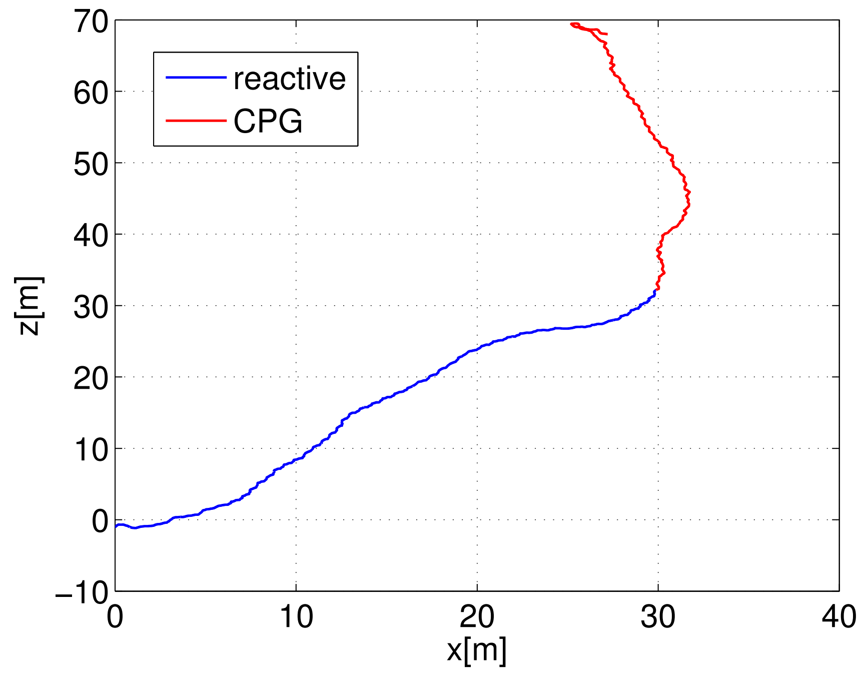


Figure 42: *2D trajectory of the tensegrity (seen from above). The blue curve represents the trajectory while the robot is driven by the reactive control algorithm and the CPG is in the learning mode (50 seconds). The motion is very regular and the heading is maintained throughout the whole period. The red trajectory represent the path traveled once the CPG controller takes over (40 seconds).*

The CPG receives here feed back from the payload strings height, the tensegrity travels a total distance of 74.3m.

Possible improvements:
Hopf oscillators can synchronize very accurately to periodic input signal and are therefore very well suited for learning motor commands for rolling tensegrities. Moreover, coupling can be added using feedback signals obtained from data measured by the robot sensors. This coupling enables the tensegrity to react to unexpected obstacles on the ground and adapts the output signal of the oscillator to the real robots position. This allows the tensegrity to roll faster and further.

However, while allowing longer rolling distances, this feedback does not control the robots heading. As we can see on figures 41 and 42, the red trajectory does not maintain a constant heading. As a consequence, additional parameters have to be added to the dynamical system or a hybrid method, such as a mix CPG - inverse kinematics can be used.

6.7 Hybrid CPG - Inverse Kinematics method

6.7.1 AWO and IK

The aim of this method is to close the loop of the AWO CPG in order to be able to control the motion of the tensegrity. This approach is inspired by a previous work by Ajallooeian et al.[37]. For this purpose, we add two terms χ_i and ξ_i to the original dynamical system equations (6.10) and (6.11):

$$\dot{\ell}_i = \gamma(g(\varphi_i) + \chi_i - \ell_i) + \frac{dg(\varphi_i)}{d\varphi_i}\,\dot{\varphi}_i + \xi_i \tag{6.20}$$

$$\dot{\varphi}_i = \omega_i + \sum_j \sin(\varphi_j - \varphi_i - \phi_{ij}) \tag{6.21}$$

These feedback terms will account for unpredicted movements of the tensegrity and perturb the limit cycle of the dynamical system to let the robot come back on the desired trajectory. Since these corrections have to be momentaneous, we impose a fading memory dynamic:

$$\dot{\chi}_i = -\lambda\chi_i$$
$$\dot{\xi}_i = -\mu\xi_i$$

with $\lambda, \mu \in \mathbb{R}_+$.

To compute the values of χ and ξ, we use the second order kinematics method as presented in section 6.4.2. The values of the coefficients are directly replaced by the output of the IK algorithm. χ and ξ are updated if and only if the payload position lies on the opposite side of the robots center of mass. In this way, the corrections are made only if the tensegrity can potentially roll in a undesired direction. Note also that in order to use this method, we need to know the position of the payload and the center of mass.

6.7.2 Hopf oscillators and IK

We can apply the same method to the adaptive frequency oscillator as presented in section 6.6.2. We add the corrective term to the output of the oscillator. The resulting dynamical system reads:

$$\dot{x} = \gamma(\mu - (x^2 + y^2))(x - \xi(t)) - \omega y - kh(t) \tag{6.22}$$

$$\dot{y} = \gamma(\mu - (x^2 + y^2))y + \omega(x - \xi(t)) \tag{6.23}$$

where $\xi(t)$ decreases exponentially with time. If the value of $\xi(t)$ is constant over time, the dynamical system converges asymptotically to $x(t) = \xi$ [38]. Figure 43, when compared to figures 41 and 42 illustrates the improvement in the robot's trajectory and steerability[5].

[5]videos of the controller in action can be found on: http://biorob.epfl.ch/files/content/users/181078/files/CPG_controller.avi

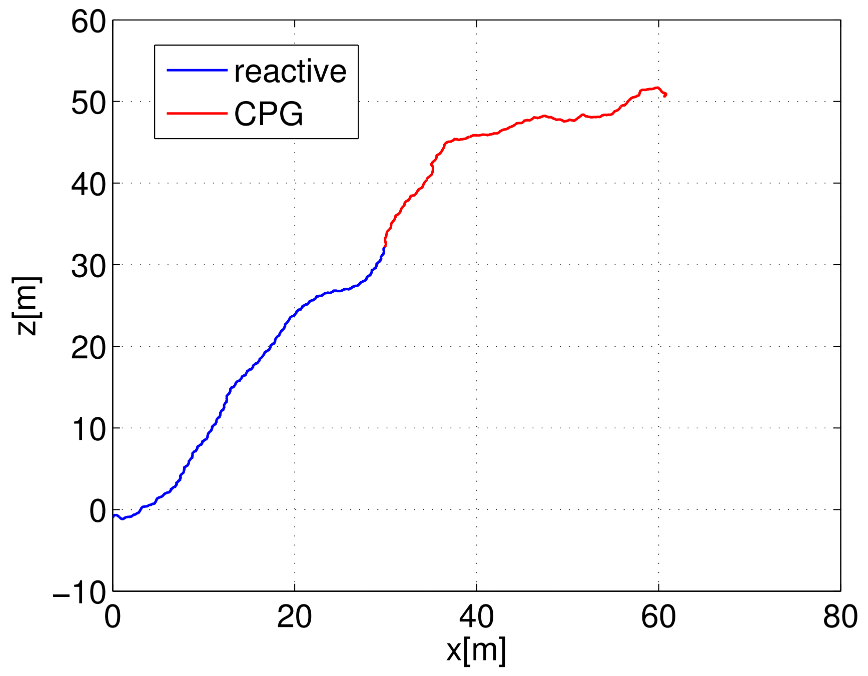


Figure 43: *2D trajectory of the tensegrity (seen from above). The blue curve represents the trajectory while the robot is driven by the reactive control algorithm and the CPG is in the learning mode (50 seconds). The motion is very regular and the heading is maintained throughout the whole period. The red trajectory represent the path traveled once the CPG controller takes over (40 seconds).*

The CPG is now coupled to the height signal and receives inputs from the second order inverse kinematics algorithm in case it starts rolling in a wrong direction. The resulting trajectory is a long and relatively straight line extending well the reactive control.

Possible improvements:
This method's efficiency relies on the quality of the inverse kinematics algorithm used. As a consequence, if we want to improve the method, we need to look into more accurate inverse kinematics algorithms. For simple tensegrity structures such as the 6-struts icosahedron considered in this paper, closed form solutions for the inverse kinematics can also be worth investigating.
Remember from the discussion of section 6.4 that the IK method has to be relatively cheap in terms of calculations and precise enough in order to deal with the dynamics of the tensegrity.

6.8 Results Summary

Summary of the results obtained with the different control strategies are plotted on figure 44. The distance is defined as the length between the starting point at rest and the position of the robot after 60 seconds of simulation. Note that the results do not take into account the trajectory of the path and, as a consequence, even if the distance traveled using the CPG controller without any trajectory control is larger than with the hybrid control, the "quality" of the path is not as good.

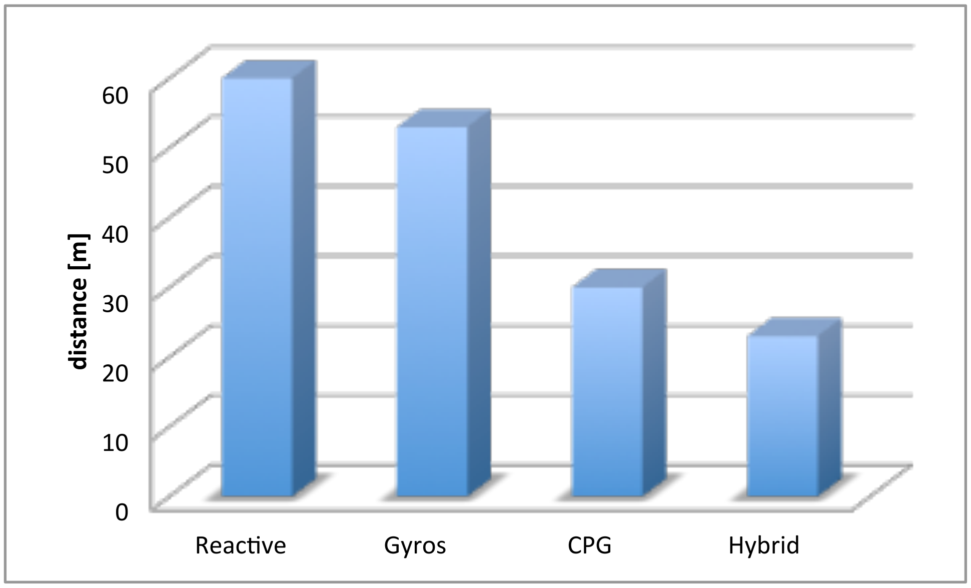


Figure 44: *Results showing the end to end distance traveled in 60 seconds on regular flat terrain by the tensegrity icosahedron using the three different control strategies developed in this section.*

With the current implementation, only the reactive controller manages to get the robot to roll in an efficient way over complex terrain such as the ones presented in section 5.2. It is also important to emphasize that the hybrid controller's performances are highly sensitive to the choice of some parameters appearing in the CPG equations such as the ones presented in equations (6.22) and (6.23). Typically, a different value of k (weight of the height feedback) can alter the performance of the controller by a factor up to 5. As a result, other methods to optimize the feedback data and to compute the corrections have to be developed to more accurately navigate in complex environments. A simple and easy way to solve this problem would be to stop the robot as soon as an unwanted displacement is measured, set the robot back to its desired position and restart the motion. However, this type of method, while easy to implement, is obviously not very efficient and a need for advanced algorithms and methods is needed in order to solve that problem. A detailed analysis of future work in that direction is presented in section 9.

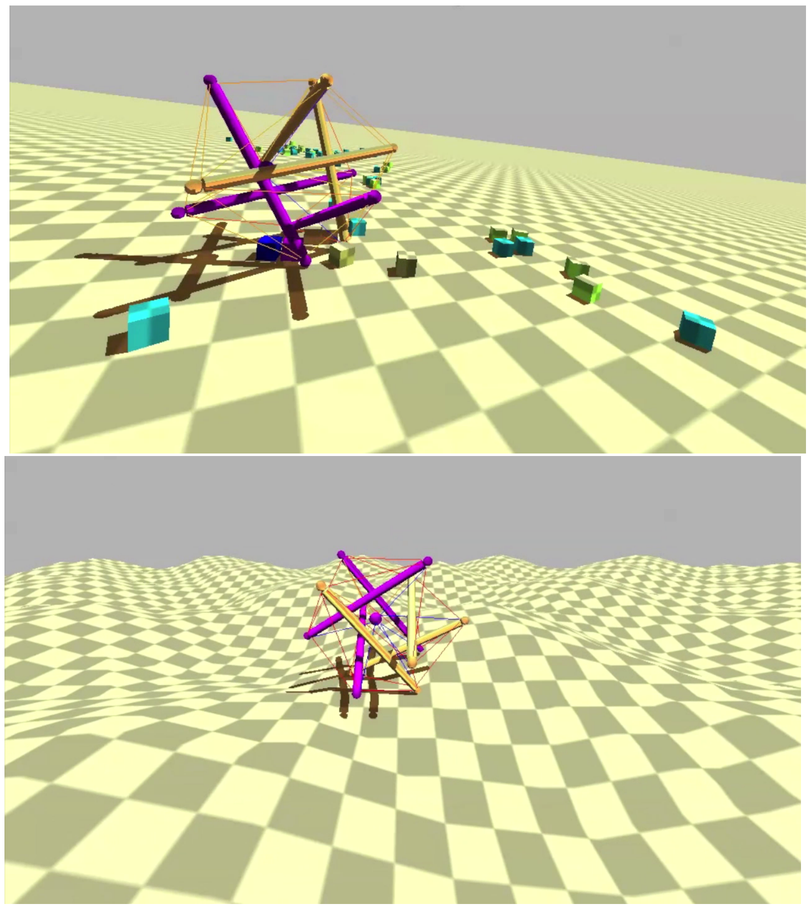


Figure 45: *Examples of successful locomotion over complex terrains such as slopes, bumps and obstacles.*

These control strategies are, to our knowledge, the only implementations of tensegrity robot controllers demonstrating advanced capabilities such as locomotion over complex terrains as well as robustness to shocks and obstacles[6]. An illustration of the robot in challenging environments as simulated in NTRT is presented on figure 45.

[6]videos of the controller in action over complex terrain can be found on: http://biorob.epfl.ch/files/content/users/181078/files/complete_controllers_demo.avi

7 Stable Gait Pattern

An interesting observation that can be made form these results has to do with the stable rolling gait pattern obtained in simulation. Using either the reactive or the CPG controllers, we observe that the rolling pattern is always the same sequence of movements. The different steps of a full roll as viewed from the front and the side are presented on figures 46 and 47, respectively. The green arrows represent the direction of locomotion.

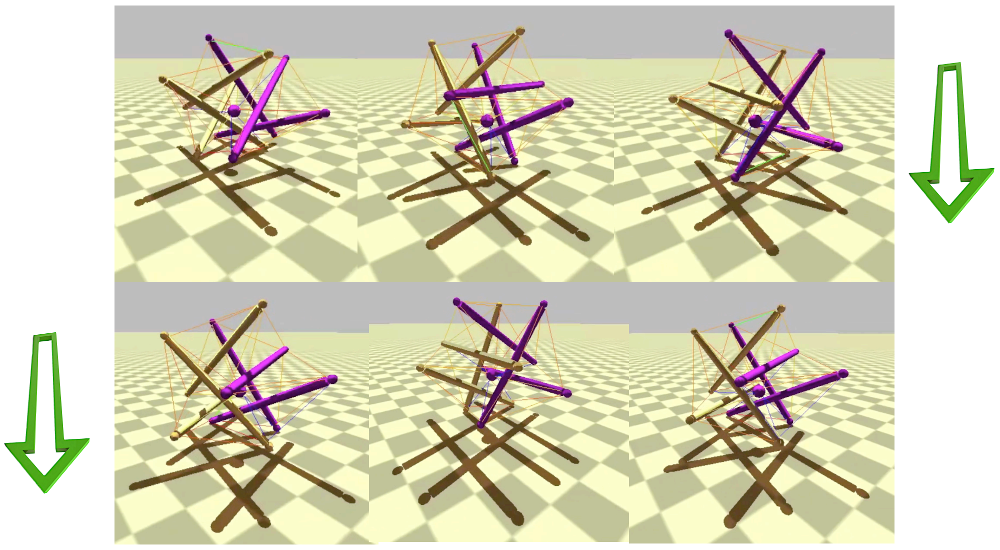


Figure 46: *Stable gait pattern, front view, 1 full roll*

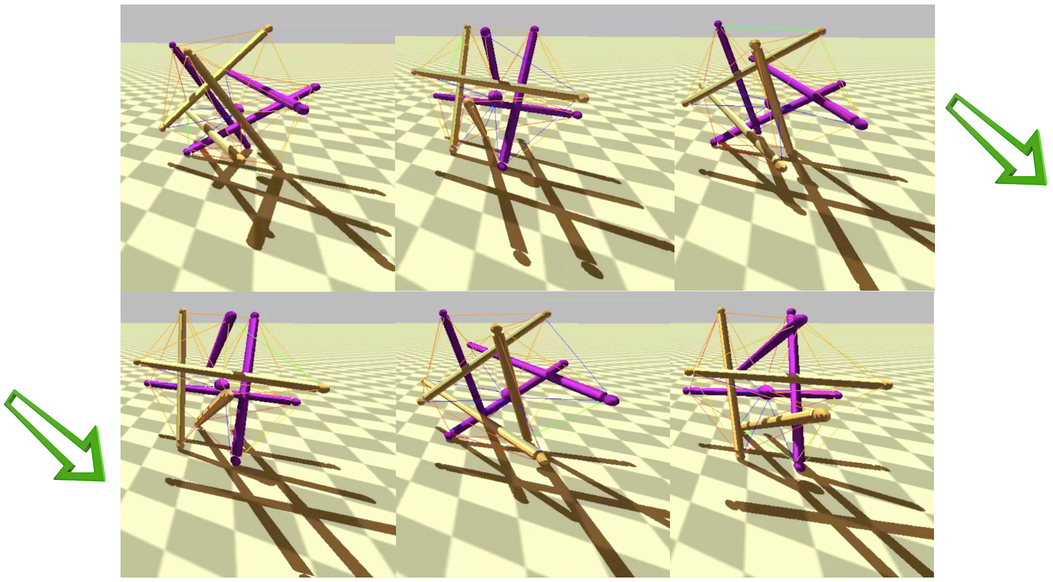


Figure 47: *Stable gait pattern, side view, 1 full roll*

At that point, it is interesting to do a parallel with a previous study of energy efficient rolling for a tensegrity robot carried out by Koizumi et al.[18]. In their paper, they studied in detail the behavior of a tensegrity icosahedron driven by pneumatic actuators located on each tensile element of the structure. They distinguish two different stable contact position where the tensegrity exhibits symmetries denoted by *axial* (**A**) and *planar* (**P**) as illustrated on figures 48 and 49.

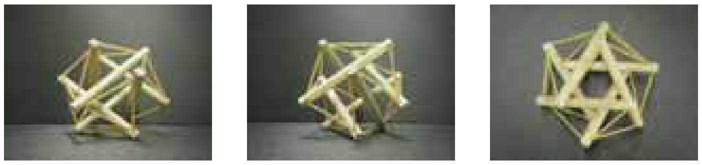

(a) front (b) side (c) top

Figure 48: *Axial symmetry contact position (**A**) (from [18])*

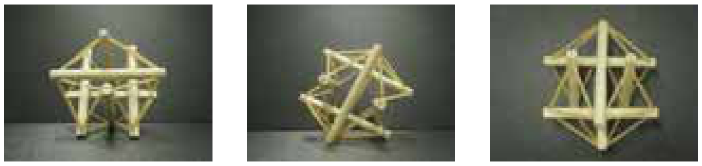

(a) front (b) side (c) top

Figure 49: *Planar symmetry contact position (**P**) (from [18])*

Possible transitions between these states are defined as follows:

- Starting from an axial contact point

  AP Axial → Planar

  AA Axial → Planar → Axial

- Starting form a planar contact point

  PA Planar → Axial

  PP Planar → Axial → Planar

Where the first state and the last state of each transition is a stable position, i.e. the robot stays in that position if no external force is applied. Experimental results show that the overall most efficient gait is obtained by transitions of type **PP** and, if the robot ends up on an axial symmetry **A**, the desired behavior would be to roll back using a transition **AP** and go back to the **PP** cycle. As quoted from Koisumi et al.:

"Rolling of a six-strut tensegrity robot should be based on transitions in PP category. In other words, the robot basically takes planar symmetric contacts alone in its stable state. If an axial symmetric contact happens due to uncertainties in the robot or the environment, we can apply any transition in category AP so that the contact between the robot and the ground returns back to any planar symmetric contact, then we can apply any transition in category PP again."

Despite the fact that the tensegrity of Koisumi et al. was controlled by pneumatic actuators and had no central payload, it is interesting to note that the stable gait obtained by the different controllers is always of type **PP** (see figures 46, 47 and 48, 49). This shows that the control strategies described in the previous section are able to converge to the most effective rolling gait. This is a crucial statement to assess for the quality of the controllers and to justify implementation on hardware prototypes.

8 Validation of the Physics Simulator

An important step of the study is the validation of the NTRT physics simulator that will on one hand confirm the quality of the results in simulation presented in the previous sections and, on the other hand, justify a further study of the same controllers in real hardware. In order to perform the validation, we carried out a series of tests both in the NTRT simulator and using a tensegrity icosahedron prototype (ReCTeR) developed by K. Caluwaerts[20][39] with a motion capture setup. Note that another detailed analysis of the results together with a description of the hardware used for these experiments is presented in [39]. We refer to this publication for technical information on the tensegrity prototype hardware details and physical properties.

The general idea of these experiments is to perform some critical sequences of actuation and movements on the prototype, record the struts position, orientation and motor commands, then position the robot in the exact same way in the simulator, replicate the same set of commands and record the positions of the struts. The data collected from both experiments is then postprocessed and compared to assess for the physical accuracy of the NTRT simulator.

8.1 Real-time Motion Capture vs Simulation

Figures 50 shows the tensegrity icosahedron prototype (ReCTeR) and figure 51 the reproduction of the same robot in the NTRT simulator. To track the state of the robot, we use an active marker motion capture setup (PhaseSpace Impulse X2 with 11 cameras). Each passive strut was fitted with 2 markers, while each active strut received 3 markers. This allows us to track the full state of the robot, except for the rotation of the thin, passive struts, which is of negligible influence.

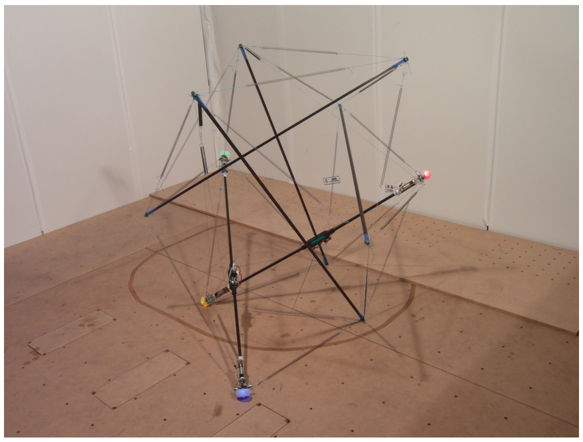
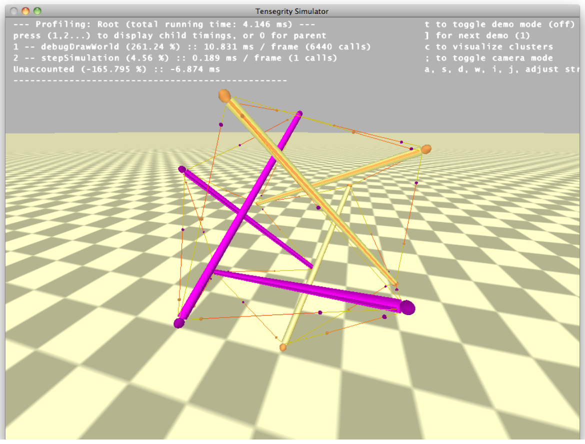


Figure 50: *Prototype tensegrity icosahedron robot (ReCTeR) used for the motion capture experiments*



```
--- Profiling: Root (total running time: 4.146 ms) ---          t to toggle demo mode (off)
press (1,2...) to display child timings, or 0 for parent        ] for next demo (1)
1 -- debugDrawWorld (261.24 %) :: 10.831 ms / frame (6440 calls) c to visualize clusters
2 -- stepSimulation (4.56 %) :: 0.189 ms / frame (1 calls)       ; to toggle camera mode
Unaccounted (-165.795 %) :: -6.874 ms                            a, s, d, w, i, j, adjust str
-------------------------------------------------
```

Figure 51: *Clone version of the prototype robot in the NTRT physics simulator*

In order to model massive springs, we use a 3 segments muscle design with intermediate masses as introduced in section 5.1. A full tensile element consists therefore of 2 massive spheres, containing each half of the spring's total mass, one inextensible massless string that can be actuated, one middle spring and is terminated by an inextensible small string. A schematic view of a 3 segment muscle is illustrated on figure 52. ReCTeR is made of 6 solid struts and 12 passive tensile components connecting each end cap to its 4 closest neighbors. 6 supplementary tensile elements are located on half of the end caps, these tensile element can be independently actuated to make the robot perform complex movements. The actuation in the real robot is performed by winding a string around an axial motor. In the simulator, this correspond to changing the actuated string's rest length since for these string $l = \ell$ if k is large enough.

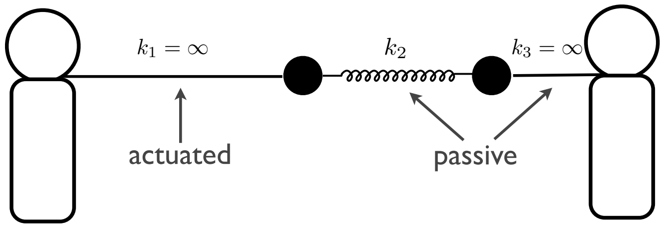


Figure 52: *ReCTeR's muscle segments scheme. The real spring mass is distributed equally among 2 intermediate masses (black spheres). The first segment is an actuated inextensible string, the second is a spring having the same spring constant than the real robot, the third and last element is a short inextensible string.*

8.1.1 Quasi-static Analysis

We start fist by investigating the static equilibrium position of the robot's end nodes in both the simulator and the motion capture. For this series of tests, we compare the equilibrium position of the nodes as a function of two actuators positions. The robot is set in an equilibrium position on the ground and the two actuators positions are updated at regular time intervals to span a wide range of values. Once the actuator has moved to a new position, the robot is remained at rest for a few second to let the system reach its equilibrium position. A schematic view of the experimental setup is represented on figure 53.

On figures 54 and 55 are plotted the vertical displacement of the node indicated by the large black dot in figure 53 as a function of the two actuated string lengths. The node of which we trace displacement is not directly actuated and its motion results from the forces distributions within the tensegrity structure. The nodal displacement as a function of the actuator position is non-linear, even for modest displacements. Note that the leftmost point (0.05, 0.05, 0) is the reference point, as the displacements are relative to this initial state. The manifold represents the interpolated average position as a function of the actuators positions.

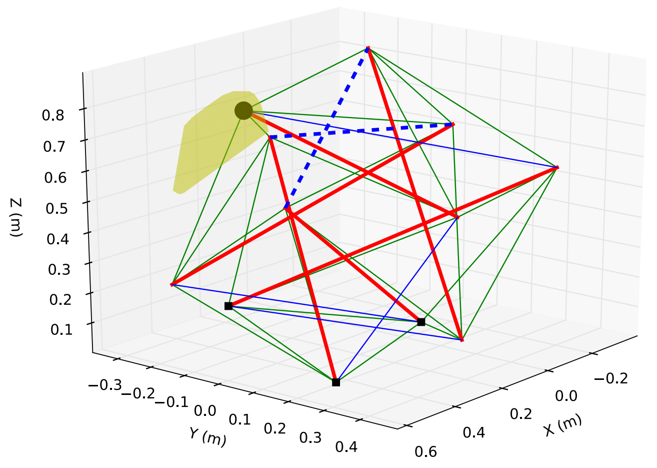


Figure 53: *Schematic view of the experimental setup. Two actuated springs (dashed lines) track a range of string lengths. The full range of motion of the tracked node during the experiment is shown in light yellow (convex hull). The nodes indicated by small black squares are on the ground.*

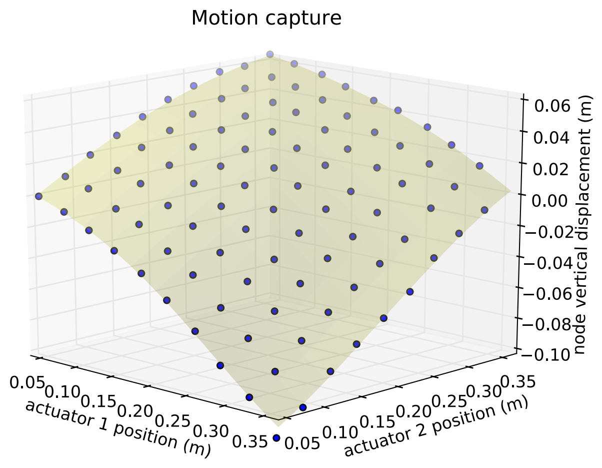


Figure 54: *Black node vertical displacement as computed from motion capture*

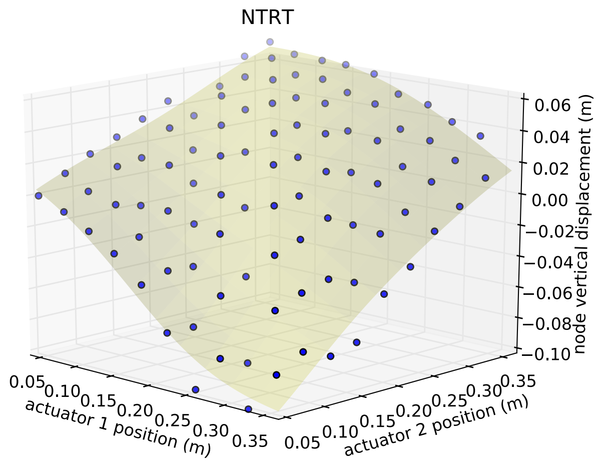


Figure 55: *Black node vertical displacement as computed from NTRT*

We observe that the manifolds are qualitatively very similar, indicating a good match and an equivalent behavior between the simulator and real physical experiments. Quantitatively, we find that the average difference on the node position is of 1.5cm which is absolutely acceptable given the order of magnitude of the robot's diameter (1.2m).

8.1.2 Dynamic Motion Analysis

To validate the dynamic behavior of the tensegrity in simulation, on which all the results presented previously in this work rely, we analyze and compare a flopping of the robot in motion capture and in simulation. For this experiment, we compress a spring and at a given time release it from 32cm to 53.5cm at 0.6m/s, causing the robot to topple on its side. Results are presented on figures 56, 57 and 58 below.

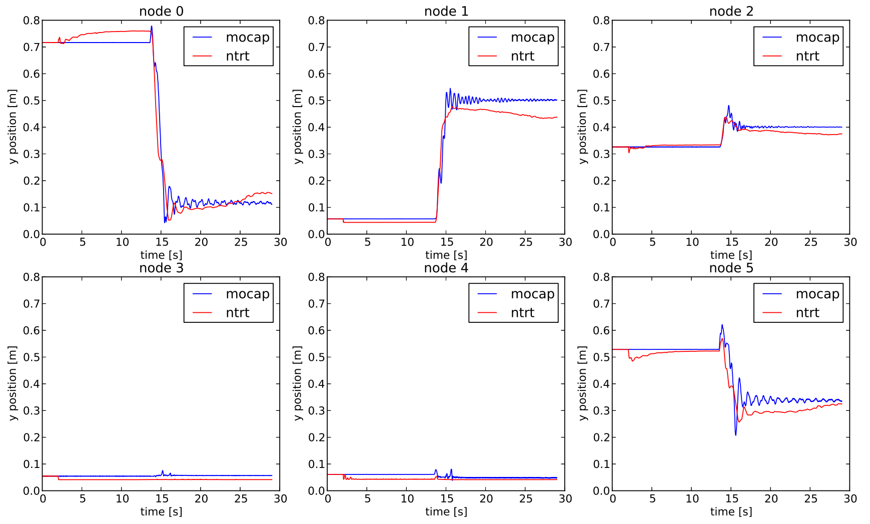


Figure 56: *Nodes 0-5 vertical position during a flop of the robot. The motion capture and NTRT simulation data are represented in blue and red, respectively.*

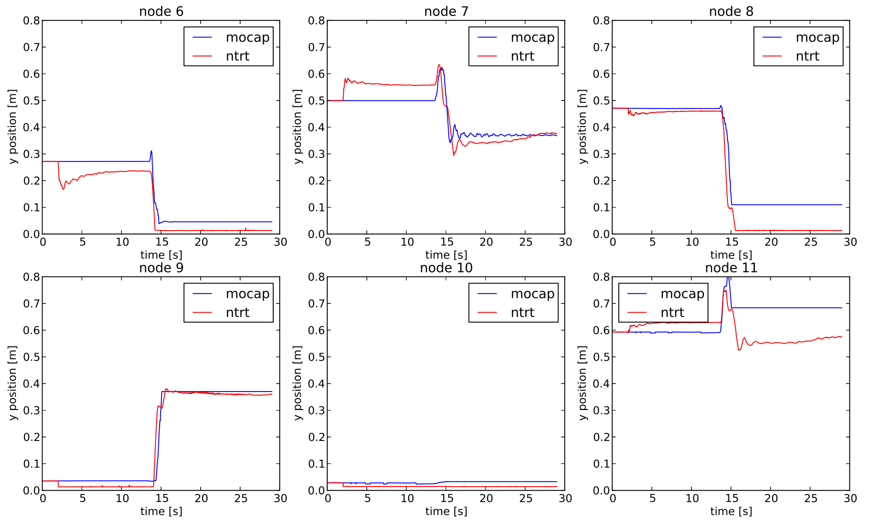


Figure 57: *Nodes 6-11 vertical position during a flop of the robot. The motion capture and NTRT simulation data are represented in blue and red, respectively.*

On figures 56 and 57, are represented the 12 nodes vertical position at each time step both for the motion capture and the simulation. We observe that the timing of the flop is very accurately reproduced in the simulator and that the general behavior is very similar in the two experiments. We can notice that the initial position is slightly offset from the motion capture, this is mainly due to the mass distribution of the rods that are in reality less regular than assumed in NTRT and the friction on the ground that is not perfectly reproduced within the simulator. This also explains the offset once the robot has toppled on its side. Other experiments on different types of ground surfaces have shown that friction can indeed significantly affect the end position of the nodes.

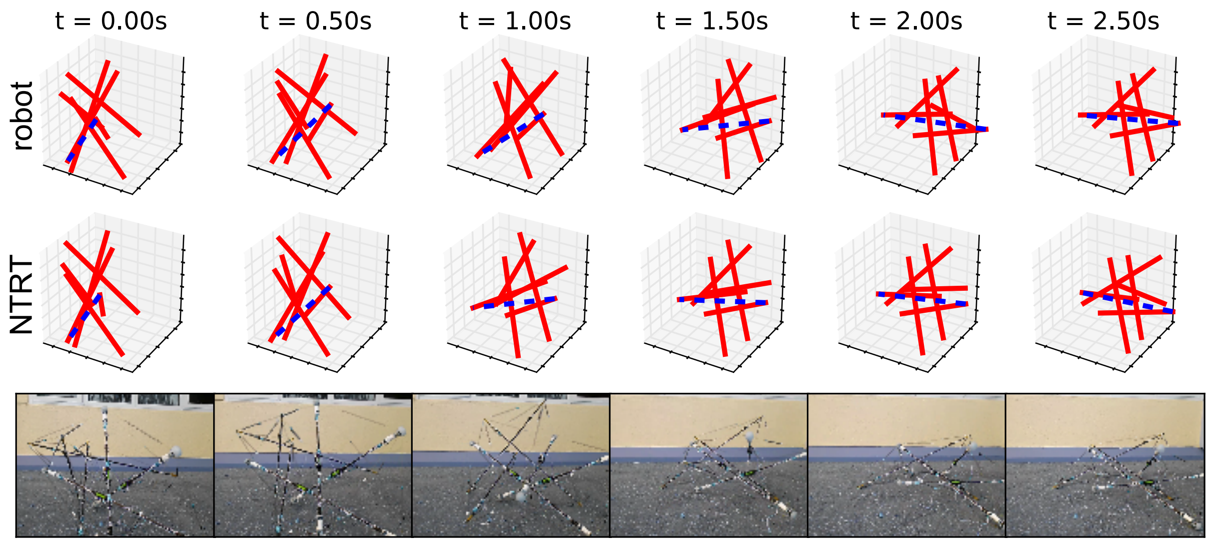


Figure 58: *Comparing robot and NTRT dynamics. The tensioned spring indicated by the dashed blue line is released (32cm to 53.5cm at 0.6m/s), causing the robot to topple. Two other actuated springs running through the center of the icosahedron are also tensioned. The other three actuated springs are at their initial lengths, resulting in two slack springs. The robot sometimes fails to roll on slippery terrains, which we also observed when reducing the NTRT friction parameters (not shown).*

On figure 58, we plot the actual position of the robot in the motion capture and in NTRT for different times during a toppling motion. The top two rows show a simplified representation of the robot's state, the red segments indicate the struts positions while the dashed blue line indicate the location of the actuated spring. During the experiment, two of the remaining 5 strings are stretched to increase the structure's stiffness and facilitate the roll. Finally, the last row of the figure shows the actual robot performing the same motion[7]. We find that the error on the struts vertical position averaged over time is of maximum 5% of the robot's diameter.

[7]a video presenting simultaneously the real and simulated behavior can be found on: http://biorob.epfl.ch/files/content/users/181078/files/flopping_simulation_vs_reality.avi

8.1.3 Results Analysis

Results of section 8.1.1 and 8.1.2 prove that the NTRT simulator is accurate enough to successfully replicate different static equilibrium positions of the robot as a function of the motor positions as well as reproduce its dynamic behavior when the tensegrity structure is not in an equilibrium/minimum of energy state. Dynamic experiments highlighted also the importance of mass repartition along the struts as well as ground contact friction and forces modeling. The end position of the robot is indeed greatly influenced by ground interaction and the whole toppling motion can be modified or even prevented by changing the struts moment of inertia. This leads us to the conclusion that similar tensegrity robots using rolling as a mean of locomotion should as far as possible try to increase their moment of inertia along the struts, enabling it to increase the amplitude of its movements. Note however that increasing the moment of inertia also increases the amplitude of the oscillation appearing inside the structure and these larger oscillations will have to be compatible with the control design of the robot.

9 Future Work

9.1 Neural Networks and Reservoir Computing

9.1.1 Principle

Some physical quantities or robot states can sometimes not be computed with a good accuracy or are sometimes not directly available by the sensors data. This problem becomes even of greater importance in tensegrity structures where the sensors might be subject to large noise to signal ratios as the high compliance of the structure introduces unexpected oscillatory responses. In real hardware, the same complications arise with the limited number of sensory information and the inherent errors associated to their measurements.
A solution to this issue is the preprocessing of the sensory information before its use. This can be achieved in different ways and we will focus here on the use of neural networks and reservoir computing. The new data flow is schematized on figure 59. A successful example of a similar implementation can be seen in a study by Gay et al.[40] where a neural network weights the sensory information from a gyroscope or a camera used as a feedback for a CPG controller.

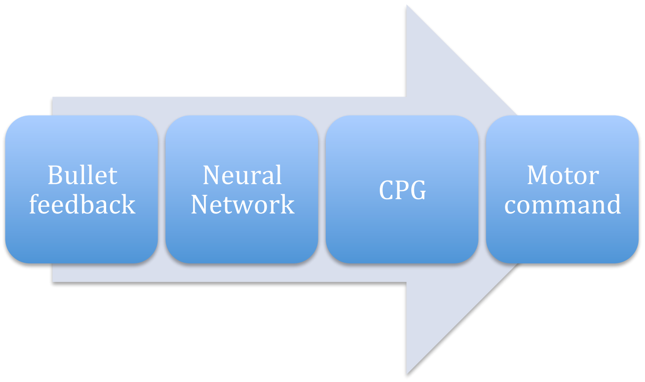


Figure 59: *New implementation where the sensor data is preprocessed by a neural network before being fed back to the CPG controller. The neural network can simulate artificial sensors, remove noise, cut frequencies or approximate certain physical parameters such as center of mass, speed, etc.*

This new preprocessing step can be used to remove noise of the sensor data, for instance by remove high frequency components of a signal, or to simulate an artificial sensor. Indeed, some physical quantities such as speed, orientation, etc. can sometimes be deduced with a relatively good accuracy from other sensor information (pressure, force, etc.). Theoretically, if there exists a mathematical relationship linking a group of sensor measurements to a physical quantity, then it is possible to artificially simulate a sensor measuring this quantity.
This mathematical relationship can be a simple linear function in which case a linear regression on a series of measurements is well adapted. However, the relation is usually more complex and a non-linear system such as the ones provided by reservoir computing methods need to be train to best approximate the desired signal.

In the case of tensegrity structures, this approach is justified as some physical parameters are not well defined. Imagine for example the measurement of the orientation of a tensegrity robot. For a rigid structure, any vector connecting two points can be used to deduce the global orientation of the robot itself. For a tensegrity however, the position of two distinct points (e.g. the two ends of a rod) can correspond to an infinite number of different orientations. Preprocessing the sensory information can in this case resolve the problem by computing a new global orientation that is a combination of the different elements orientation and possibly some past data.
This method possesses the advantage of being computationally very efficient, as no complex mathematical systems have to be solved to compute the new data, making it very easy to embed in a physics simulator or on real hardware. Note however that a training set has to be available to correctly tune the weights of the neural network, requiring an a priori knowledge of the signal we want to learn. A detailed example is presented below to better illustrate how these techniques can be used to control a tensegrity robot.

9.1.2 Example - Center of mass speed estimation from tension sensors

As an example, we compute an estimation of the center of mass speed from the tensions in the 36 springs of the tensegrity icosahedron. We compare two different approaches: a simple linear regression and a reservoir computing approach. In both of these methods, we apply a regression on a data set from a recording of 90 seconds where the center of mass speed is recorded from the physics simulator.

Linear regression:
For the training set of size n, we denote by $\vec{y} = (y_1, y_2, ..., y_n)^T$ the scalar speed of the center of mass and X the matrix defined as:

$$X = \begin{pmatrix} T_1^{(1)} & T_2^{(1)} & ... & T_{36}^{(1)} & 1 \\ T_1^{(2)} & T_2^{(2)} & ... & T_{36}^{(2)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ T_1^{(n)} & T_2^{(n)} & ... & T_{36}^{(n)} & 1 \end{pmatrix} \in \mathbb{R}^{n\times 37}$$

where $T_i^{(k)} \in [0,1]$ is the tension of spring i at time t_k divided by the maximal tension. We want to find $\vec{w} = (w_1, w_2, ..., w_{37})^T$ the set of weights minimizing the euclidian distance:

$$d_1 = \|X\vec{w} - \vec{y}\|^2 \tag{9.1}$$

The optimal set of weights $\vec{w}_{\text{opt}}$ satisfying $\|X\vec{w}_{\text{opt}} - \vec{y}\|^2 \leq \|X\vec{w}_{\text{opt}} - \vec{y}\|^2 \ \forall \vec{w} \in \mathbb{R}^{37}$ can be obtained easily by an optimization algorithm from Eigen3[41] or NumPy[42] linear algebra libraries. If the mathematical relationship between the spring's tensions and the center of mass speed is indeed linear, the speed is then simply given by:

$$\text{speed}_{\text{lin\_reg}} = \sum_{i=1}^{36} T_i w_{\text{opt}_i} + w_{\text{opt}_{37}} \quad , \ \forall \ T_i \in \mathbb{R} \tag{9.2}$$

Reservoir computing:
The reservoir computing or echo state network (ESN) approach is a more elaborate supervised learning method that can approximate a large variety of signals[43]. Examples of a successful application of reservoir computing to robotics are numerous in the literature, see e.g. [44] and [45]. The idea of the method is to use a so-called reservoir that consist of a neural network with a random connectivity matrix and to train this network with a data set. The output of the network is then computed and weighted to best fit the target set. A schematic representation of the training flow is depicted on figure 60.
Following this procedure, we start first by creating a random squared matrix $W \in \mathbb{R}^{m \times m}$ representing the random connectivity weights of a neural network of m neurons. W is constructed such that $\max_\lambda \{\lambda \in |(\text{Spec}(W))|\} = r < 1$ where r is spectral radius of W. This condition enforces the fading memory property of the neural network. We then define the entering connections to the network. Again, we construct a random matrix W_{in} such that 90% of its entries are zeros (i.e., 10% of the neurons receive the input signal). We then define a series of m states $\vec{s}_j \in \mathbb{R}^n$ with $\vec{s}_0$ being a random vector with entries in the interval [-1,+1] and $\vec{s}_{k+1}$ being computed recursively by the rule:

$$\vec{s}_{k+1} = \tanh(W\vec{s}_k + W_{\text{in}}\vec{x}_k) \quad , \; k = 1, ..., n \tag{9.3}$$

where $\vec{x}_k = (T_1^{(k)}, T_2^{(k)}, ..., T_{36}^{(k)})^T$.

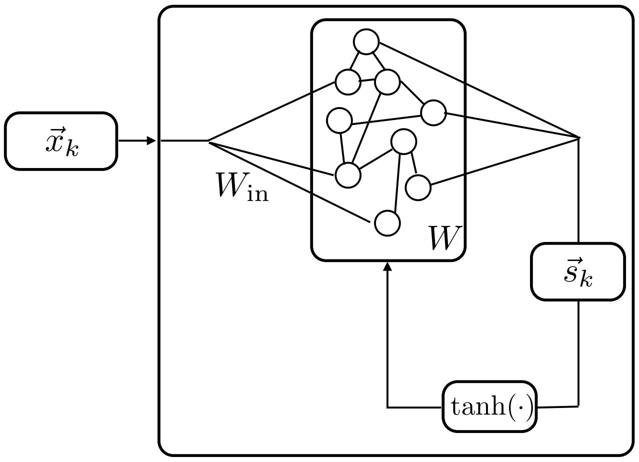


Figure 60: *Schematic representation of the recursive neural network: the input signal $\vec{x}_k$ is fed to the neural network through the connexion matrix W_{in}, the neural connexion weights are stored in the matrix W, the state of the network at time t_k is represented by the vector $\vec{s}_k$, non-linearities are added by an hyperbolic tangent function acting on the system output at each time step.*
The neural network is constructed such that W_{in} and W are random matrices with $w_{i,j}, w_{in_{i,j}} \in [0,1]$, $\max_\lambda \{\lambda \in |Spec(W)|\} < 1$ (ensuring the fading memory property) and W_{in} sparse.

We then construct the states matrix S as:

$$S = \begin{bmatrix} \vec{s}_1 & \vec{s}_2 & \dots & \vec{s}_n & 1 \end{bmatrix}$$

and, using a linear regression algorithm, compute the optimal set of weights $\vec{w}_r$ minimizing the Euclidian distance:

$$d_2 = \|S\vec{w} - \vec{y}\|^2 \tag{9.4}$$

The final signal estimation is then given by:

$$\text{speed}_{\text{res}} = \sum_{i=1}^{36} T_i w_{r_i} + w_{r_{37}} \quad , \; \forall \; T_i \in \mathbb{R} \tag{9.5}$$

As a mean of comparison, results obtained by the linear regression method and the reservoir computing are plotted on figure 61. The target signal representing the center of mass velocity (green) is approximated by a linear regression method on the data set XW (red) and using a reservoir computing method (blue). The approximation errors obtained by the two methods with respect to the train data set are plotted on figure 62.

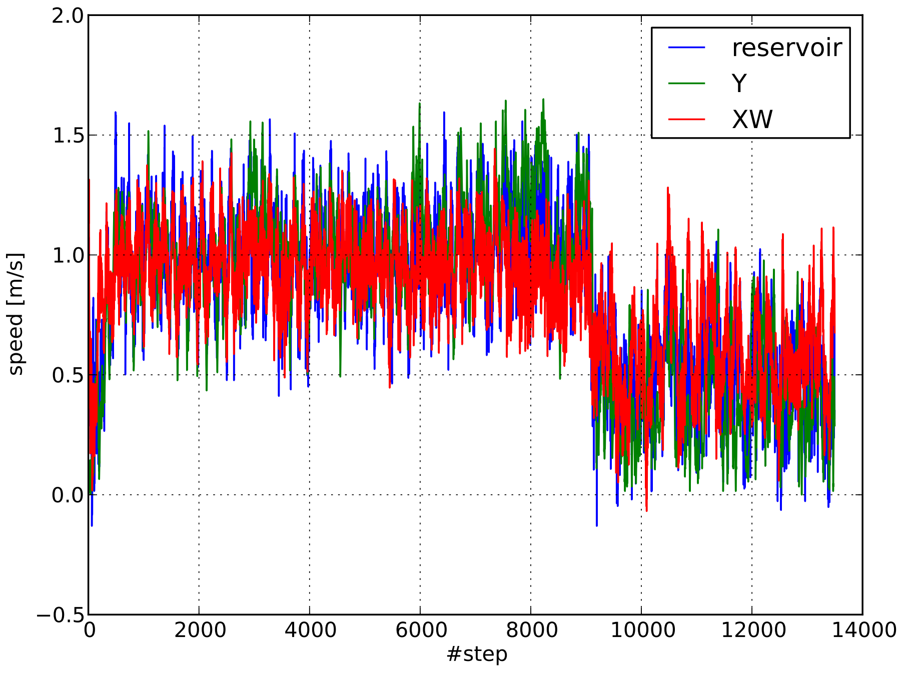


Figure 61: *Estimation of the tensegrity's center of mass velocity from tensions values in the 36 muscles, learning over a period of 90 seconds. The exact value (from NTRT) is plotted as Y (green), the estimation using linear regression XV (red) and the estimation using reservoir computing (reservoir, in blue).*

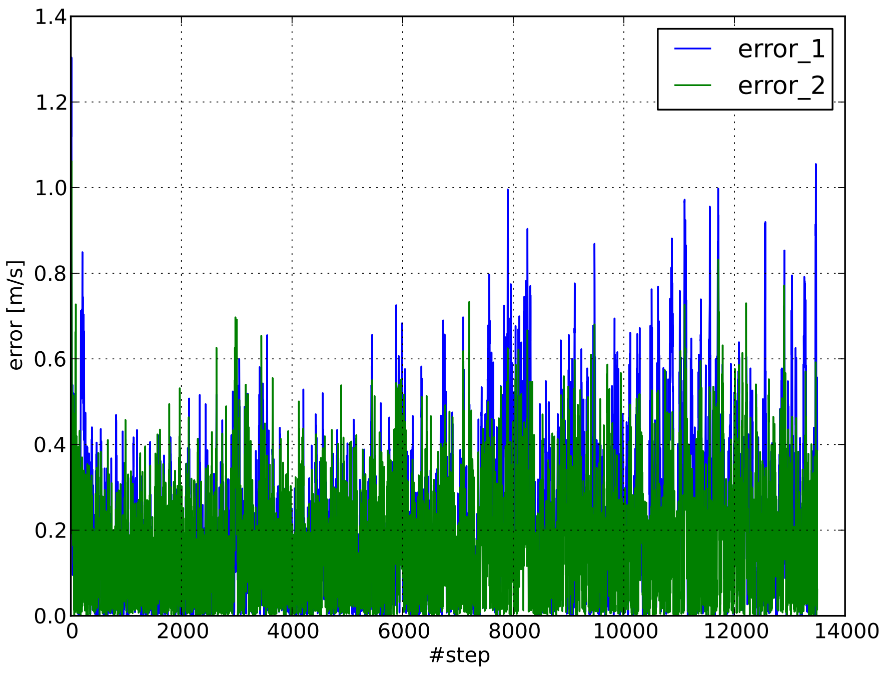


Figure 62: *Error on the speed estimation from sensor data. Error_1 represents the error computed by linear regression (RMS value of 0.2889 m/s) and Error_2 represents the error for the reservoir computing method (RMS value of 0.2085 m/s).*

We observe that the center of mass speed can indeed be deduced with a reasonable precision from the string's tension data, using simple linear regression or reservoir computing, with a relatively better performance from the reservoir computing method. It is important to note that if this method were to be used to compute a given physical quantity later used in simulation or on a real platform, it would need further tests to validate the quality of the measurements, as there is always a risk of over fitting on training data sets. This example has to be seen here as a proof-of-concept towards such an implementation.

9.2 Evolutionary Optimization

Evolutionary algorithms (EAs) are meta-heuristics search and optimization algorithms that are based on natural evolutionary principles (survival of the fittest). EAs can be used to discover complex tensegrity control strategies for example by evolving sinusoidal control parameters and optimize the traveled distance[19] and also to fine tune CPG parameters. In the NTRT physics simulator, coevolution and centralized evolution optimization tools were build and can be used to improve target performances of any CPG controller. The principle of the algorithm used in our simulator is to create a population of tensegrity robots having each their version of a reference CPG controller. Evolved parameters can be specified and have a fixed range of possible values. Each robot is then simulated during a fixed amount of time and its fitness function is computed. The fitness function can typically be the start to end distance, the total length of the path traveled, etc. Once the simulations are over, the best individuals, i.e. the individuals with the largest fitness function, are kept for the following generation. Random mutations and crossovers are used to generate the full new population. A scheme of the general flow of an evolutionary algorithm is presented on figure 63

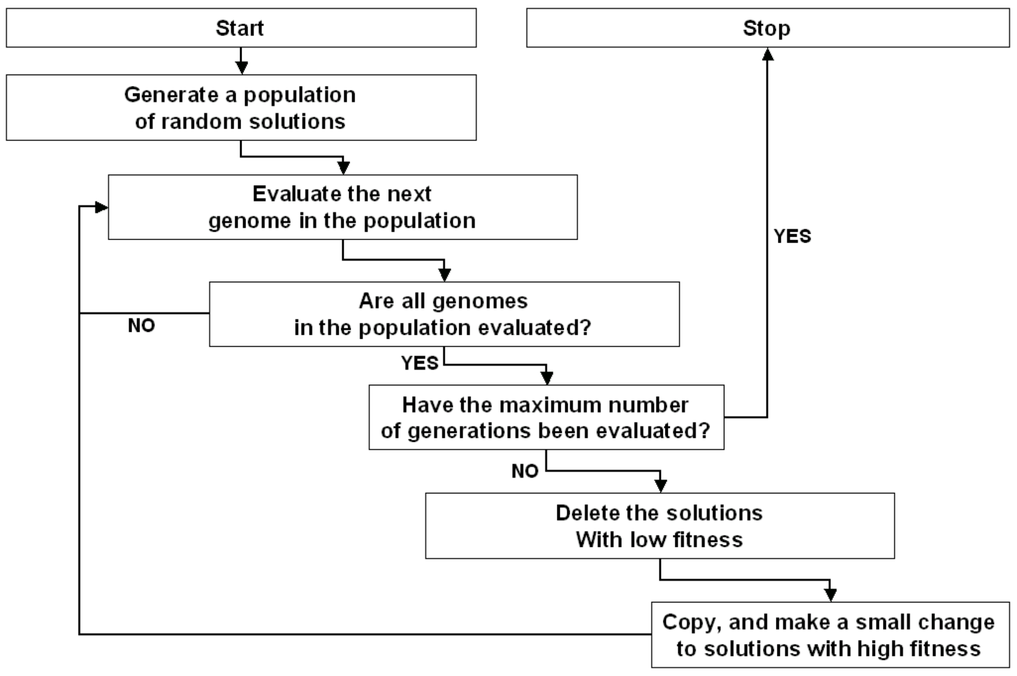

Start
Stop
Generate a population of random solutions
Evaluate the next genome in the population
Are all genomes in the population evaluated?
NO
YES
Have the maximum number of generations been evaluated?
YES
NO
Delete the solutions With low fitness
Copy, and make a small change to solutions with high fitness

Figure 63: *Schematic representation of an evolutionary algorithm (from [46])*

The use of EAs can be brought even further to a higher level of abstraction. Instead of tuning parameters of a dynamical system, we can create sequences of instructions or sets of oscillators that can be combined according to a well defined set of mathematical rules. Imagine for example a controller that generate motor signals as a combination of mathematical functions chosen from a reference set (see figure 64). The reference set contains typically simple mathematical functions that have desirable mathematical properties, e.g. symmetry, periodicity, discontinuities, etc. The set of functions that are chosen by each controller can be use as its genotype and evolved in a series of simulations. In the end, complex motor commands can be sent to the different robot's controllers allowing for a broad range of sophisticated motions.
Similarly, Ijspeert et al. presented a study in which neural networks are evolved to control the swimming of a lamprey[47]. In this study, the neural network is produced according to a set of construction rules. This set of rule is then evolved using a genetic algorithm as showed on figure 65. Once the evolutionary process is over, the lamprey is able to swim and turn efficiently.

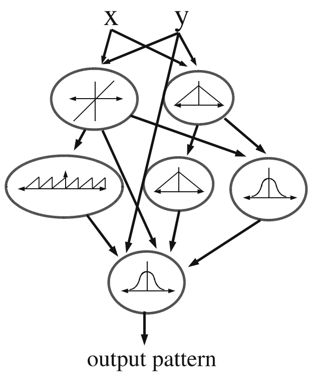


Figure 64: *Illustration of a complex function generation by composition of simple mathematical function chosen from a reference set (from [48])*

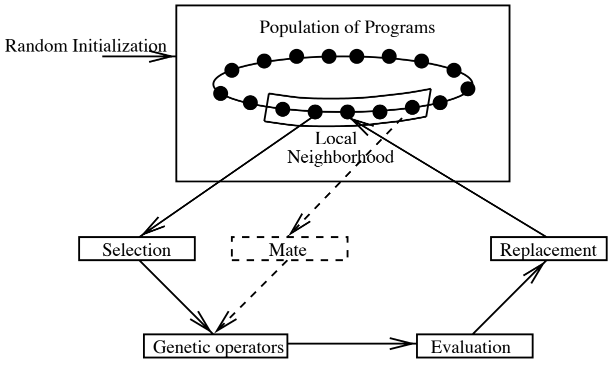


Figure 65: *Sketch of the evolutionary algorithm used to create a robust neural network from a set of construction rules to control a lamprey in simulation (from [47])*

Similar evolutionary algorithms could be used to evolve tensegrity controllers, together with their CPG implementation. The great advantage offered by this technique is that no a priori knowledge of the controller or the neural network is required, the physical constraints fixed by the user being the only limits to the search space.

9.3 Exploring new Geometries and Soft-Robotics

The current NTRT physics simulator supports a feature that allows the use of `.spr` files used in the Java/VRML tensegrity simulator Springie[49]. These files are part of a database of tensegrity models, some of them having a large number of components (see e.g. figure 66). As a result, these complex geometries can be loaded, simulated and actuated using the same principle as described for the tensegrity icosahedron. This brings the search for control strategies to a larger extend and enable a deeper study of shape matching and soft robotics control strategies for tensegrity.

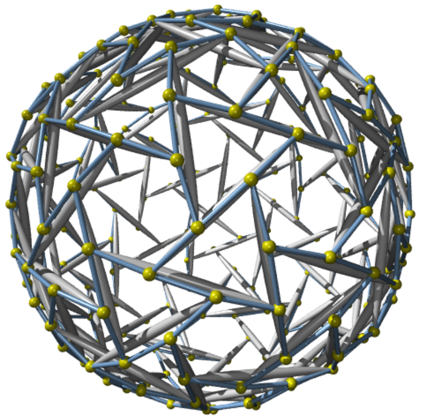
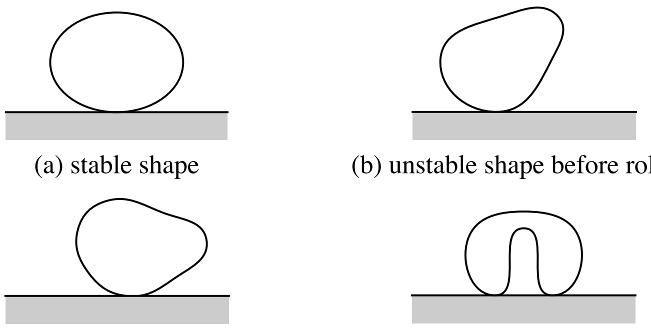


Figure 66: *Example of a complex tensegrity structure modeled using Springie[49]*

To illustrate how the study can be expanded, we make here a parallel with a study made by Sugiyama et al.[50]. In this study, a controller strategy was developed to make a spherical deformable wheel shaped robot crawl and jump over complex terrains. The wheel was made of a rubber material and was actuated using shape memory alloys connected to the wheel and tied together in its center. When heated by an electric current, the material retracts, pulling on the attached wheel segment. The principle used to make the wheel spin was to change the shape of the whole robot to bring it in an unstable state, forcing it to tilt in one direction. To make it jump, potential energy was stored in the structure as the robot was deforming. See figure 67 for illustrations.


(a) stable shape

(b) unstable shape before rolling

(c) stable shape after rolling

(d) stable shape with high potential energy

Figure 67: *Illustration of the possible ways to control a compliant robot's shape to enable rolling or to store potential energy within the structure (from [50])*

This type of structural deformations could be used in the same way for tensegrity structures having enough components to simulate a flexible material. We could for example imagine a controller generating undulations in the robot depending on the height of each of the actuator, using for instance the IR-distance sensors described in section 6.2. If the spring length is set according to the height measurement (e.g. $\ell(t) = |\sin(h+\varphi(t))|+\varepsilon$, where ε is an offset value and $\varphi(t)$ is a linear function of the time), the whole tensegrity will oscillate above the ground. This type of behavior could potentially be exploited for locomotion, jellyfishes for example use similar types of body deformation to move their body in the water[51] that could be reproduced by oscillators[52].
Going even closer towards to the field of soft robotics, Umedachi et al.[53] developed a decentralized control strategy for soft-bodies that could potentially be imitated by large scale tensegrity structures.

10 Conclusion

Tensegrity structures and their control for robotics application is still a novel topic and only a few studies have been carried out in this direction. The main difficulty, consisting of dealing with non-linearities and the high compliance of the physical system, is usually avoided in classical robotics control. In this work, we showed that CPG controllers could be a suitable and realistic way to deal with these features inseparable form tensegrity structures. The first step of the study was to find a way to move the tensegrity over simple terrain. A reactive controller was developed to this end. This controller has demonstrated very good performances in simulation environment, both over simple and complex terrains. However, due to the large sensing information required, real hardware implementation for this kind of controller is quite unrealistic. On the other hand, we have been able to show that adaptive frequency oscillators were be able to learn and reproduce the same gaits while requiring much less sensory information. Furthermore, it has been showed that the gait obtained is the most energy efficient. We then proposed and tested several improvements in order to enable the new CPG controller to deal with more complex terrains. For this purpose, first and second order inverse kinematics methods were implemented. These methods have been able, when combined with a simple CPG, to provide enough feedback to control the motion of the robot on simple flat terrain. In order to justify the results obtained in simulation, we assessed the correctness and accuracy of the physics simulator using a motion capture setup and a prototype tensegrity robot. Finally, we proposed several new ideas and further thoughts on a future development of these control strategies.

While contributing to tensegrity robotics, this work participates also to central pattern generator research, assessing their suitability to control highly compliant structures and demonstrating notably the usefulness of adaptive frequency oscillators.

August 18, 2013
Moffett Field, United States

Jérémie Despraz

11 Glossary

Tensegrity Structure

Tensegrity structures are composed of axially loaded compression elements encompassed within a network of tensional elements, and thus each element experiences either pure linear compression or pure tension.

Reactive controls

Reactive controls are a type of controller that computes actuators commands according only to the sensor information fed back to the robot. In that sense, the controller reacts directly to the environment through the sensory feedback and does not require any other type of stimulation. Obviously, this type of implementation requires lots of measurements in order to perceive the robot's environment accurately.

Central Pattern Generator (CPG)

Central pattern generators are neural circuits found in both invertebrate and vertebrate animals that can produce rhythmic patterns of neural activity without receiving rhythmic inputs. CPGs have been studied from a biological perspective and have been used extensively in robotics especially for walking and other locomotion research.

Inverse Kinematics (IK)

Inverse kinematics refers to the use of the kinematics equations of a robot to determine the motor commands that provide a desired position of the robot or some of its elements.

NASA Tensegrity Robotics Toolkit (NTRT)

NTRT is a set of tools built over the open source physics engine Bullet. This toolkit allow a physically accurate modeling of tensegrity structures.

Reservoir Computing (RC)

Reservoir Computing is an approach to design, train, and analyze recurrent neural networks. More specifically, RC offers methods for designing and training artificial neural networks, and yields computational and sometimes analytical models for biological neural networks.

Evolutionary Algorithms (EAs)

The evolutionary framework or evolutionary algorithm (EA) is a family of search meta-heuristic and optimization algorithms that mimics the process of natural evolution (survival of the fittest) using technics inspired by biological evolution such as reproduction, mutation, recombination and selection. They operate on a population of candidate solutions which performances within the framework are measured by a fitness function.

12 Annexes

- Physical parameters of the simulation:

Property	Value	Units
Scaling factor	10	-
Speed factor	0.4	-
Gravity	-10	m/s^2
Ground friction coeff.	2.7	-
Ground restitution coeff.	0.2	-
Time step (fixed)	1/60	s

- Physical parameter of rigid bodies (solid struts):

Property	Value	Units
Total mass	1	kg
Cylinder length	1	m
Cylinder radius	0.3	m
End-caps radius	0.3	m
Linear velocity damping	0	-
Angular velocity damping	0	-
Inertia along symmetry axis	$1/12ml^2$	m·kg^2

- Physical parameter of rigid bodies (payload):

Property	Value	Units
Total mass	5	kg
Radius	0.5	m
Linear velocity damping	0	-
Angular velocity damping	0	-
Inertia along symmetry axis	$2/5mr^2$	m·kg^2

- Physical parameter of tensile elements (strings/springs):

Property	Value	Units
Spring constant	1'000	N/m
Velocity damping coeff.	1	-
Mass	0	kg

References

[1] Adrian Agogino, Vytas SunSpiral, and David Atkinson. Niac phase i final report. *NASA Innovative Advanced Concepts Program*, 2013.

[2] Robert E Skelton, R Adhikari, J-P Pinaud, Waileung Chan, and JW Helton. An introduction to the mechanics of tensegrity structures. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 5, pages 4254–4259. IEEE, 2001.

[3] Kenneth D Snelson. Continuous tension, February 16 1965. US Patent 3,169,611.

[4] D. E. Ingber and M. Landau. Tensegrity. 7(2):8344, 2012.

[5] http://mondo-blogo.blogspot.ch/2012/01/genius-of-kenneth-snelson.html.

[6] http://www.marlboroughgallery.com/galleries/chelsea/artists/kenneth-snelson.

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

[8] Donald E Ingber. Tensegrity i. cell structure and hierarchical systems biology. *Journal of Cell Science*, 116(7):1157–1173, 2003.

[9] Donald E Ingber. Tensegrity and mechanotransduction. *Journal of bodywork and movement therapies*, 12(3):198–200, 2008.

[10] Yves Termonia. Molecular modeling of spider silk elasticity. *Macromolecules*, 27(25):7378–7381, 1994.

[11] Vytas SunSpiral, George Gorospe, Jonathan Bruce, Atil Iscen, George Korbel, Sophie Milam, Adrian Agogino, and David Atkinson. Tensegrity based probes for planetary exploration: Entry, descent and landing (edl) and surface mobility analysis. To appear in. *International Journal of Planetary Probes*, July 2013.

[12] Brian R Tietz, Ross W Carnahan, Richard J Bachmann, Roger D Quinn, and Vytas Sunspiral. Tetraspine : Robust Terrain Handling on a Tensegrity Robot Using Central Pattern Generators. In *2013 IEEE/ASME Advanced Intelligent Mechatronics*, pages 261–267. IEEE, July 2013.

[13] Tom Flemmons. The bones of tensegrity. http://www.intensiondesigns.com/bonesoftensegrity.html, 2012.

[14] Chandana Paul, John William Roberts, Hod Lipson, and FJ Valero Cuevas. Gait production in a tensegrity based robot. In *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on*, pages 216–222. IEEE, 2005.

[15] John Rieffel, Barry Trimmer, and Hod Lipson. Mechanism as mind-what tensegrities and caterpillars can teach us about soft robotics. In *ALIFE*, pages 506–512, 2008.

[16] Anders S Wroldsen, Mauricio C de Oliveira, and Robert E Skelton. A discussion on control of tensegrity systems. In *Decision and Control, 2006 45th IEEE Conference on*, pages 2307–2313. IEEE, 2006.

[17] Steven James Burt. Kinematics algorithms for tensegrity structures. 2013.

[18] Yuusuke Koizumi, Mizuho Shibata, and Shinichi Hirai. Rolling tensegrity driven by pneumatic soft actuators. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1988–1993. IEEE, 2012.

[19] Atil Iscen, Adrian Agogino, Vytas SunSpiral, and Kagan Tumer. Controlling tensegrity robots through evolution. 2013.

[20] Ken Caluwaerts, Michiel D'Haene, David Verstraeten, and Benjamin Schrauwen. Locomotion without a brain: Physical reservoir computing in tensegrity structures. *Artificial life*, 19(1):35–66, 2013.

[21] SL Hooper. Central pattern generators. encyclopedia of life sciences, 2001.

[22] Fred Delcomyn. Neural basis of rhythmic behavior in animals. *Science*, 210(4469):492–498, 1980.

[23] Sten Grillner and Peter Wallen. Central pattern generators for locomotion, with special reference to vertebrates. *Annual review of neuroscience*, 8(1):233–261, 1985.

[24] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

[25] Ralph A DiCaprio. Gating of afferent input by a central pattern generator. *Journal of neurophysiology*, 81(2):950–953, 1999.

[26] L. Righetti, J. Buchli, and A.J. Ijspeert. From dynamic hebbian learning for oscillators to adaptive central pattern generators. In *Proceedings of 3rd International Symposium on Adaptive Motion in Animals and Machines – AMAM 2005*. Verlag ISLE, Ilmenau, 2005. Full paper on CD.

[27] L. Righetti and Ijspeert A.J. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006.

[28] Erwin Coumans. Bullet physics engine, 2005.

[29] Adrian Boeing and Thomas Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 281–288. ACM, 2007.

[30] Neville Hogan. Impedance control: An approach to manipulation. In *American Control Conference, 1984*, pages 304–313. IEEE, 1984.

[31] Omer Orki. *A Model of Caterpillar Locomotion Based on Assur Tensegrity Structres*. PhD thesis, TEL AVIV UNIVERSITY, 2012.

[32] Jesse van den Kieboom. http://www.codyn.net/.

[33] Rhodri H Armour and Julian FV Vincent. Rolling in nature and robotics: A review. *Journal of Bionic Engineering*, 3(4):195–208, 2006.

[34] Jesse Van Den Kieboom. Arbitrary wave-form oscillator. *Biorob Laboratory, EPFL*.

[35] Mostafa Ajallooeian, Albert Mukovskiy, Jesse van den Kieboom, Martin Giese, Auke Ijspeert, et al. A general family of morphed nonlinear phase oscillators with arbitrary limit cycle shape. Technical report, 2013.

[36] Ludovic Righetti, Jonas Buchli, and Auke Jan Ijspeert. From dynamic hebbian learning for oscillators to adaptive central pattern generators. In *Proceedings of 3rd International Symposium on Adaptive Motion in Animals and Machines–AMAM 2005*, page 45, 2005.

[37] Mostafa Ajallooeian, Sébastien Gay, Alexandre Tuleu, Alexander Sproewitz, and Auke Ispeert. Modular control of limit cycle locomotion over unperceived rough terrain. In To appear in. *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013.

[38] Sébastien Gay, Sarah Dégallier, Ugo Pattacini, Auke Ijspeert, and José Santos Victor. Integration of vision and central pattern generator based locomotion for path planning of a non-holonomic crawling humanoid robot. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 183–189. IEEE, 2010.

[39] Ken Caluwaerts, Jérémie Despraz, Atil Iscen, Andrew Sabelhaus, Jonathan Bruce, and Vytas SunSpiral. Design and control of compliant tensegrity robots through simulation and hardware validation. In To appear in. *Journal of the Royal Society Interface*, 2013.

[40] Sébastien Gay, José Santos-Victor, and Auke Ijspeert. Learning robot gait stability using neural networks as sensory feedback function for central pattern generators. In To appear in. *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013.

[41] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[42] Travis Oliphant et al. Numpy, 2007.

[43] Herbert Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007.

[44] Eric A Antonelo and Benjamin Schrauwen. Supervised learning of internal models for autonomous goal-oriented robot navigation using reservoir computing. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2959–2964. IEEE, 2010.

[45] Eric A Antonelo, Benjamin Schrauwen, and Jan Van Campenhout. Generative modeling of autonomous robots and their environments using reservoir computing. *Neural Processing Letters*, 26(3):233–249, 2007.

[46] J.C. Bongard, R. Pfeifer, E. Hafen, and I. Harvey. Incremental approaches to the combined evolution of a robot's body and brain.

[47] Auke Jan Ijspeert and Jérome Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial Life*, 5(3):247–269, 1999.

[48] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[49] T Tyler. Springie: A tensegrity simulator using java, vrml and pov ray.

[50] Yuuta Sugiyama and Shinichi Hirai. Crawling and jumping by a deformable robot. *The International Journal of Robotics Research*, 25(5-6):603–620, 2006.

[51] Matthew J McHenry and Jason Jed. The ontogenetic scaling of hydrodynamics and swimming performance in jellyfish (aurelia aurita). *Journal of Experimental Biology*, 206(22):4125–4137, 2003.

[52] Lesley Ann Low, Per G Reinhall, Duane W Storti, and Erica B Goldman. Coupled van der pol oscillators as a simplified model for generation of neural patterns for jellyfish locomotion. *Structural Control and Health Monitoring*, 13(1):417–429, 2006.

[53] Takuya Umedachi, Koichi Takeda, Toshiyuki Nakagaki, Ryo Kobayashi, and Akio Ishiguro. Fully decentralized control of a soft-bodied robot inspired by true slime mold. *Biological cybernetics*, 102(3):261–269, 2010.