Transfer Learning for Faster Discovery of Tensegrity Robot Gaits

By

Akshay S. Kashyap

* * * * * * * * *

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Computer Science

UNION COLLEGE

March 21, 2018

Abstract

Tensegrity Robots are a class of Soft Robots gaining attention due to their ease-of-assembly, ability to preserve structural integrity under physical deformation, and ability to move with dynamic gaits. Previous work has shown how optimization techniques can be used to find the optimal gait in a given environment to move as efficiently and quickly as possible. The problem with these techniques, however, is that they suffer from the “Cold Start" problem, where they need to re-learn the optimal gait for the robot when it is deployed in new, unseen conditions. Our research explores how we Transfer Learning can be used to leverage knowledge from a previous gait discovery experience, called Source task, to a new environment, called Target task, in order to learn these gaits faster. We use Bayesian Optimization as our base optimization technique and a transfer learning framework that models the distance between the Source and Target tasks. This approach shows an improvement in both the performance of the learned gait in the Target task and the number of optimization trials it takes to learn this optimal gait.

Acknowledgements

I'd like to thank my parents for constantly comforting me, for checking up on how thesis was going, and for understanding why I could not video chat for half the term.

I'd like to thank my friends for always being encouraging and supportive during stressful times, and without whom I would have probably finished this thesis two weeks ago.

Finally, they say you succeed by standing on the shoulders of giants, so I'd like to thank one of those giants, Prof. John Rieffel, who for some reason thought I was good at research my freshman year and let me work with him, whose work I have extended here, and who was always there to help me when I was stuck with a problem.

Contents

List of Figures

List of Tables

1 Introduction

In the 1940s Kenneth Snelson, a New York city-based sculptor, began exploring the idea of tension-based structures inspired by Buckminster Fuller's geodesic dome. Snelson realized this in 1948 in the form of the X-Piece, a set of interconnected plywood struts and monofilament strings. Thus began the phenomenon of Tensegrity Structures springing up in art, architecture, and science. Since this structure appeared frequently in biology, such as in the muscular-skeletal system [2] and the animal cell's cytoskeleton [1], the field of Soft Robotics eventually collected it into it's genus of biologically-inspired robot morphologies.

Tensegrity Robots are interlocked sets of disjoint rigid struts and tensional connections such as springs or struts that give them intrinsic resilience and adaptability often lacking in conventional rigid robots. Since the components are either under compression or tension, a unique property acquired by these robots is pre-stress stability, which ables them to absorb and distribute throughout the structure even high impact crippling and deforming forces, only to spring right back into their original equilibrium state.

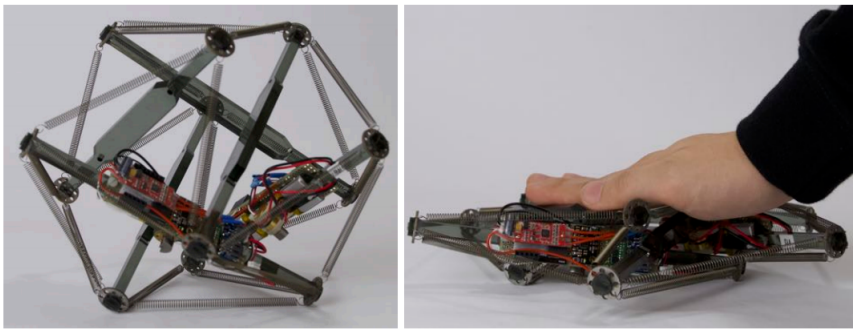


Figure 1: VVVALTR Tensegrity Robot: Regular (Left); Compressed (Right) [1]

A tensegrity robot's elasticity and structural resonance properties can be exploited to have it move by shifting its center of mass in two ways: a) by using actuators that expand and contract the soft parts, making it "crawl," or, b) by attaching motors on the struts that drive it vibrationally. Just as locomotory organisms have ***gaits***, so does a tensegrity robot. These gaits can be characterized in terms of the parameters of the machinery driving the robot, such as actuator settings or combinations of motors speeds. An ***effective gait*** moves it with superior speed and efficiency, and an ***optimal gait*** is the one that is most effective. Given the complex dynamics of the tensegrity structure, high dimensionality of the gaits, and the resulting extensive gait search space, finding the optimal, or even an effective, gait is an expensive challenge. The optimal gait varies across diverse terrains and physical conditions and trying to manually engineer these gaits is a cumber-

[1] EvoRobo Group, Union College, NY.

some effort. For these reasons, as described in the next section, data-driven statistical learning and optimization techniques can be employed to discover effective and, ideally, optimal gaits in the environment the robot is deployed in.

A problem here, however, is that current state-of-the-art gait discovery methods need to be applied to every new terrain the robots move on. This research began with the idea that being able to leverage past experiences and previously learned knowledge, much like animals so naturally do, can help the robots learn these gaits quicker and adapt better to new environments. This idea is further explored and implemented in the form of a robust Transfer Learning-based framework that can successfully transfer previously learned knowledge and enhance the gait discovery process in new, unseen environments and damaged states. Specifically, we use the Bayesian Optimization-based gait optimization, devised by Rieffel and Mouret [8], as our base gait discovery method.

1.1 Motivation

The physical properties of tensegrity robots make them well-suited for various applications that require adaptability and durability in a given environment. The modular design of these robots provides them with not only tensile strength and the ability to quickly recover from deformations but also allows them to be low-cost, reusable, and easy to build. Their physical dynamism combined with nimbleness enables locomotion in ways impossible for rigid robots. NASA is considering using Tensegrity Robots for planetary exploration, which currently costs $2.1-billion per mission [16], and researchers at UCSD are exploring deploying these in indoor air ducts maintenance, a $60-billion-a-year affair [17]. This research is motivated by the goal to provide these robots with a greater degree of mobility, adaptability, and resilience against failures to be used in such practical applications to drastically reduce financial expenditure and engineering efforts.

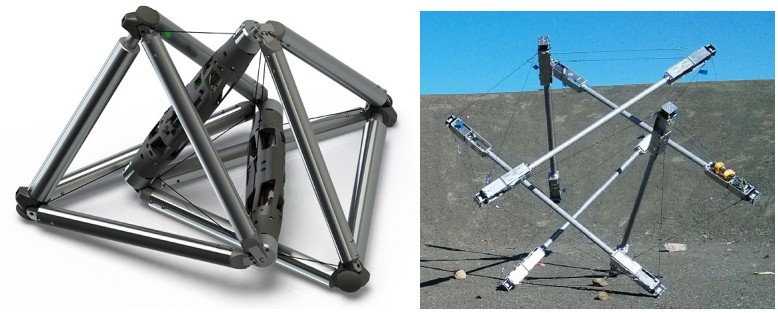


Figure 2: DucTT Robot (Left) [2]; NASA SUPERball Robot (Right) [3]

[3]UC San Diego Coordinated Robotics Lab.
[3]BEST Lab, University of California, Berkeley.

1.2 Contributions

Our work has two contributions:

1. The Transfer Learning-based solution significantly improves the gait optimization process by reducing the robot's control space and allowing the Bayesian Optimizer to more effectively and quickly search for an effective gait.

2. The simulator allows for analysis of tensegrity dynamics and exploration of new morphologies, behaviors, and control techniques without having to physically build these robots.

2 Background

2.1 Bayesian Optimization

Bayesian Optimization (BO) is a class of Sequential Model-based Optimization (SMBO) algorithms. Bayesian Optimization models an expensive objective function f using a regression method on previous observations of the function to determine the next optimal point to sample f at. It is called "Bayesian" since it chooses the next point by computing a posterior distribution of f using the likelihood of previously observations and a prior on the form of f. The algorithm works roughly follows:

- Use previously observed points $x_{1:n}$ to compute a posterior expectation of the shape of f.
- Sample f at a new point x' that maximizes some measure of expectation of f. This measure is called the *Acquisition Function*.

These steps are repeated until some convergence criterion is met.

2.1.1 Gaussian Processes (GP)

Bayesian Optimization uses Gaussian Processes regression to model f. A GP can be thought of as a Gaussian distribution over a function $g(x)$ at each value of x and is used in BO for constructing the posterior distribution. A probability distribution over f can be represented as follows:

$$f \sim N(\mu(x), \sigma^2(x))$$

where $x \in x_1, x_2, ..., x_n$, the set of all data points.

Since these points are also from a Gaussian distribution, each of them can have their own mean $\mu(x_i)$ and variance $\sigma^2(x_i)$. A Gaussian Process uses these to construct a mean function $m(x)$ and a covariance function $k(x_1, x_2)$:

$$x_{1:n} = [x_1, x_2, ..., x_n]$$
$$m(x_1 1 : n) = [m(x_1), m(x_2), ..., m(x_n)]$$

$$K = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \dots & \dots & \dots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix}$$

where K is an n x n covariance matrix and $K_{i,j} = k(x_i, x_j)$.

The multivariate Gaussian distribution is specified over f as follows:

$$f_{1:n} \sim N(m(x_{1:n}), K)$$

At a certain point x_i, this is gives us the mean and variance for distribution of f at x_i, i.e, $f(x_i) \sim N(m(x_i), [K_{i,1}, ..., K_{i,n}])$. This allows us to update our belief of what f looks like using the observed points and pick the next best point.

2.1.2 Acquisition Function

Instead of optimizing f itself, the GP is used to optimize a much simpler and tractable proxy function called the Acquisition Function. A common acquisition function is Expected Improvement:

$$EI(\mathrm{x}) = E[max(0, f(x_{i+1}) - f(x_i))]$$

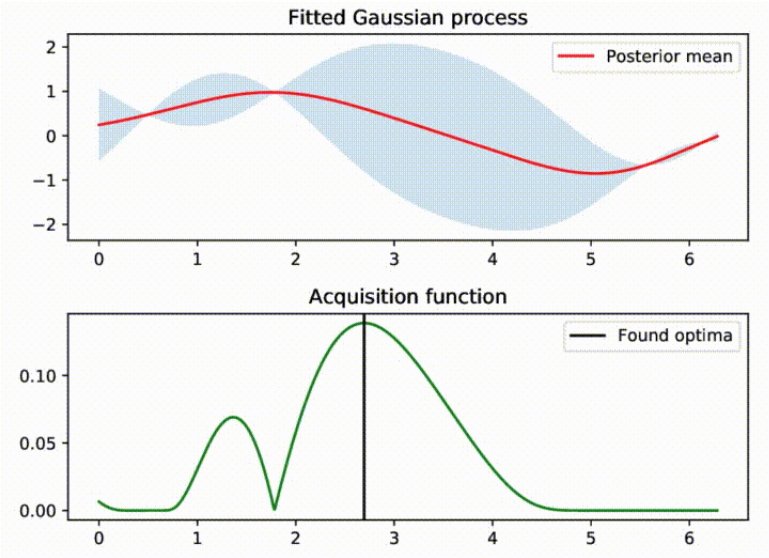


Figure 3: Bayesian Optimization Process [18]

2.2 Transfer Learning

Transfer Learning is a subfield of Machine Learning that deals with techniques to leverage past training experiences from a source task A to speed up training in new target task B. The primary objective is to alleviate the "Cold Start" problem where a learning algorithm initially explores sub-optimal regions in the parameter space before converging at an optimal solution.

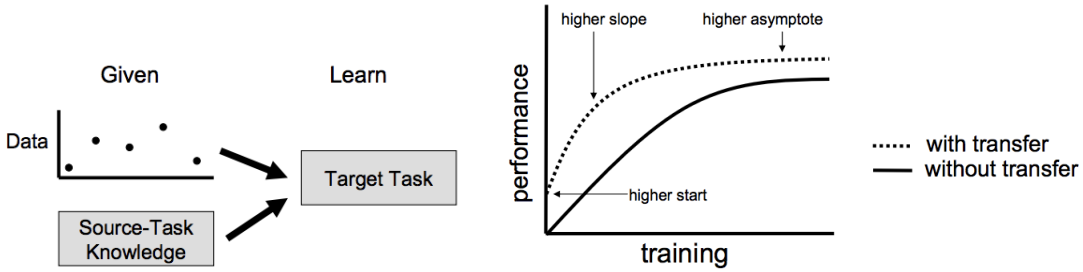


Figure 4: Transfer Learning: Overview (Left); Learning Improvement (Right) [10]

3 Related Work

3.1 Tensegrity Dynamics and Control

Much of the earlier work in tensegrity research dealt with structural analysis. Juan and Tur [3] present a mathematical framework for static analysis of tensegrity structures. They analyze rigidity and stability from the point of view of motion, force, and energy. Ali, et al. [4] propose a dynamic design criteria for tensegrity structures and perform a design optimization of a tensegrity footbridge using genetic algorithms. The induction of tensegrities into soft robotics and research on tensegrity locomotion has been more recent, using vibration- and actuation-based techniques. Paul et al. [5] present an cable-collocated actuation technique where electric motors periodically alter the resting length of the cable to shift the center of mass. In NASA's SUPERball [16], the cables are actuated in a manner that causes the robot to roll. A commonly used bio-inspired control technique for tensegrity locomotion is Central Pattern Generators (CPGs), which is a mechanism for producing periodic, rhythmic patterns in the absence of sensory feedback [6]. The previously mentioned SUPERball uses this and Bliss et al. [7] use this for a tensegrity-based swimming robot. Optimization techniques and stochastic algorithms are also being explored for locomotion control and gait generation. J. Rieffel and J.B. Mouret [8] present a Bayesian Optimization approach to discovering optimal motor speeds for robots with vibration-induced locomotion. X. Geng, et al. [11] present a Deep Reinforcement Learning approach to discover rolling gaits using Mirror Descent Guided Policy Search (MDGPS) that learns deep neural network policies.

3.2 Transfer Learning

Although Transfer Learning research has been active for decades, it has risen in popularity recently due to the explosive growth in the use of Machine Learning and data-driven methods for various applications. Torrey and Shavlik [9] provide an excellent overview of the various topics under this field. We will be focusing in this research on transfer techniques applicable to Bayesian Optimization. M. Gönen and A. Margolin [12] develop a kernelized Bayesian Transfer Learning framework that combines dimensionality-reduction models with task-specific projection matrices to find a shared subspace for the tasks. J. Davis and P. Domingos [13] develop a "Deep Transfer" technique that utilizes second-order Markov logic to lift first-order logic formulas in the source task domain, (or in our case, terrain,) to second-order formulas by replacing all predicate names with predicate variables, and then grouping them into cliques that are scored, ranked, and used to guide discovery of new structure in the target domain, (the new terrain). P. Zhao, et al [11] present an Online Transfer Learning framework that improves the learning model in a continuous fashion as a stream of training inputs is received, as opposed to traditional learning that

looks at batches of data. This is based on updating weights as a sequence of inputs is received from the source domain, and transferring this to the target domain task. TT. Joy, et al. [10] present a framework for transfer of Gaussian Process kernel information in Bayesian Optimization tasks that models source task as a noisy observation of the target task, which is being used primarily in this research due to its simplicity and ease of implementation.

4 Tensegrity Simulator

4.1 Motivation

Tensegrity robots exhibit complex dynamics, are nonlinear, and are of high dimensionality in terms of state and locomotion control space, making it difficult to study them. To ease the analysis of these systems, we built as Tensegrity Simulator platform to efficiently and realistically model new structures and simulate their physical interactions with the environment.

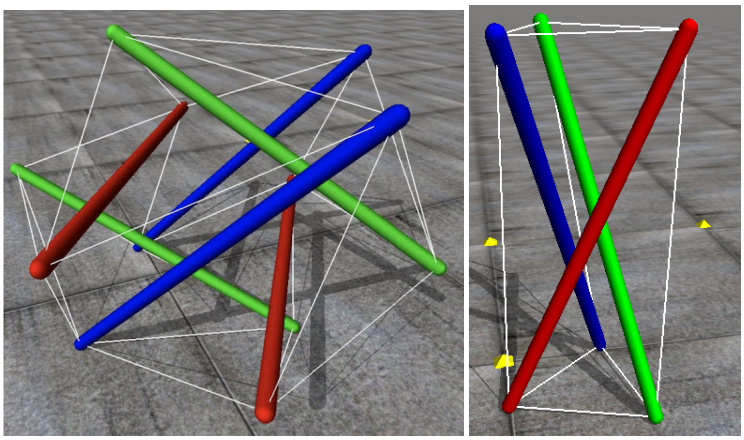


Figure 5: Tensegrity Simulator. 6-Strut (Left); 3-Strut (Right)

4.2 Tensegrity Dynamics

The simulator models tensegrity dynamics under the following specifications and assumptions.

A Tensegrity Robot is composed of N^{struts} Struts and $N^{springs}$ springs. Since we are working with a vibration-induced locomotion robot, each strut can have a motor on it, and so there are $0 < N^{motors} \leq N^{struts}$ motors.

Each strut is of length l and mass m. It can be modeled as a cylinder with radius r or a rectangular box with side a. Each strut i has a position vector $r_i = (x_i, y_i, z_i)$, with its edges at $e_i^1 = (x_i + l, y_i + l, z_i + l)$ and $e_i^2 = (x_i - l, y_i - l, z_i - l)$, and orientation vector $o_i = (\psi, \theta, \phi)$, representing yaw, roll, and pitch.

The Center-of-Mass of the system is computed as $r^{com} = \frac{\Sigma_{i=1}^{N} m_i r_i}{\Sigma_{i=1}^{N} m_i}$. A normal force of gravity acts downward on each strut in contact with the ground as $F_N = mg$, along with a force of kinetic friction against the direction of motion as $f_k = \mu_k F_N$.

Each spring is of resting length x and spring constant k, with each end attached to one of the strut edges. It exerts two equal and opposite forces on the strut edge each end is attached to, so there are $2N^{springs}$ spring forces acting in the tensegrity system. Each force obeys Hooke's Law: $F = k(d - x)$, where F is the tension/compression force and d is the length under tension/compression. Under normal circumstances, $d = ||e_i^p - e_j^q||$, the distance between the two strut edges the spring is connected to.

Each vibrating motor M_i is modeled as a perpendicular force applied periodically at points along the circumference of the strut. It is characterized by (F_{M_i}, ν_i, Φ_i), the magnitude, frequency, and phase. The motor is attached at the center of the strut j and the point at which it acts at at time t is given by the coordinates $p_t^j = (x_j sin(2\pi\nu_i t + \Phi_i), y_j cos(2\pi\nu_i t + \Phi_i), 0)$.

4.3 Implementation

The simulator was built using C++ and the Open Dynamics Engine (ODE), a rigid-body dynamics simulation framework. Each tensegrity robot component is represented as a class with necessary attributes and behavior. It makes extensive use of Factory and Compositional Software Patterns to create and bind together the components. The software simulator preservers the physical tensegrity philosophy of keeping the structure as modular and reusable as possible, making it simple to assemble or take apart.

```
Motor::Motor() {
    _step = 0;
    _frequency = PI / 8;
    _limit = (2 * PI) / _frequency;
    _speed = 1.0;
}

Motor::Motor(float initial_phase, float frequency) {
    _step = initial_phase;
    _frequency = frequency;
    _limit = 2 * PI / (_frequency);
}

void Motor::set_speed(double speed) {_speed = speed;}
double Motor::get_speed() {return _speed;}

void Motor::set_frequency(double frequency) {_frequency = frequency;}
double Motor::get_frequency() {return _frequency;}

void Motor::_step_reset() {_step = 0;}
```

```
// dSpaceID space, dWorldID world
Strut::Strut(d_vector coords, d_vector angles, double mass,
        double length, double radius, d_vector color) {

        // _space = space;
        // _world = world;
        _coords = coords;
        _angles = angles;
        _length = length;
        _mass = mass;
        _radius = radius;
        _color = color;
        _motor = NULL;
}

double Strut::get_mass() {return _mass;}
double Strut::get_radius() {return _radius;}
double Strut::get_length() {return _length;}
```

Figure 6: Code for Components

5 Proposed Solution

To quickly recap the problem being addressed, a tensegrity robot does not have the same optimal gait in different environments and physical states. Optimal gaits need to be re-discovered in new terrain or damaged state the robot operates in, which can be time consuming. We are trying to overcome this by using a Transfer Learning-based solution.

5.1 Problem Formalization

Although a gait can be characterized by one or more motor parameters, we will be optimizing for just frequency in this research. Let $g = (f_{m_1}, f_{m_2}, f_{m_3})$ be a gait of a tensegrity robot, where m_i represents the frequency of motor i. Let $D = [f_{min}, f_{max}]$ be the domain of motor frequency values. $m_i \in D$ and $g \in D^3$.

Let S be the source task, the condition the robot has already discovered an optimal gait in using Bayesian Optimization. Let T be the target task, the new condition the robot is being deployed in and needs to discover optimal gaits in. Let $p^S(g)$ and $p^T(g)$ be the performance of the robot as functions of its gait in S and T respectively.

5.2 Flexible Transfer Learning Framework for Bayesian Optimization

TT Joy, et al., introduce a novel Transfer-Learning framework for Bayesian Optimization called **Efficient-BO**, of which a brief summary is given below. It models source observations as noisy measurements of target function:

$$y_i^S = p^T(g_i^S) + \epsilon_i^S, \forall i = 1...n$$

It does this by adding noise variance to the source observations in the Target GP kernel, which is drawn from an inverse gamma distribution:

$$\sigma_S^2 \sim InvGamma(\alpha_p, \beta_p)$$

where p is the no. of points observed so far at any given optimization trial, and α_p and β_p are parameters updated at every trial as follows:

$$\alpha_p = \alpha_0 + \frac{p}{2}$$
$$\beta_p = \beta_0 + \frac{\Sigma_{i=1}^{p}(y_i^T - \hat{y}_i^S)}{2}$$

where (y_i^T is the $i-$th observation output of the target function, $\hat{y}_i^S$) is the source value at the $i-$th observation predicted by a GP fitted on source task observations. α_0 and β_0 are hyperparameters

in the order of 10^{-3}.

The Kernel of the Target GP is updated as follows:

$$K = K + \begin{bmatrix} \sigma_S^2 I_{N^S \times N^S} & 0 \\ 0^T & \sigma_T^2 I_{N^T \times N^T} \end{bmatrix}$$

The approach is designed to handle different degrees of relatedness between the source and target functions systematically. Detailed implementation is shown in Algorithm (Section 10.2).

5.3 Adaptation

This research adapts and applies this framework for our problem. We will hereby refer to Bayesian Optimization algorithm with no Transfer Learning as **Plain-BO** and the one presented above as **Efficient-BO**.

5.3.1 Step 1: Source Task Learning

We run the BO algorithm on S for n trials to optimize p^S. Let $O^S = \{X^S, y^S\}$ be the source task observations, where $X^S = \{g_1, g_2, ..., g_n\}$ and $y^S = \{p_1, p_2, ..., p_n\}$ be the performance of the robot. The optimal gait $g_o = \text{argmax}_{g \in X^S}\ p^S(g)$.

5.3.2 Step 2: Target Task Learning

We run the Efficient-BO algorithm with O^S as initial observations and computing the source noise variance at every trial, which is added to the target GP kernel.

5.3.3 Algorithm Pseudocode

See Appendix (Section 9.2).

6 Evaluation

6.1 Simulated Tensegrity Robot

The tensegrity robot simulation used for this research is based on VVVALTR, an actual Tensegrity Robot at Union College, and has physical properties of each component in the simulation are scaled versions of and consistent.

6.2 Bayesian Optimizer

This research uses **PyGPGO** [18], an open source Bayesian Optimization library that was re-engineered to implement the proposed solution. The re-engineered code is available open source on Github: `https://github.com/kashyapakshay`

6.3 Gait Optimization Task

Each gait discovery task consists of N optimization trials. Each trial is as follows:

1. Optimizer chooses a gait to evaluate and sends it to the simulator.
2. Simulator runs the tensegrity with the sent gait for a fixed amount of time, measures the speed of the robot, and sends this back to the optimizer.

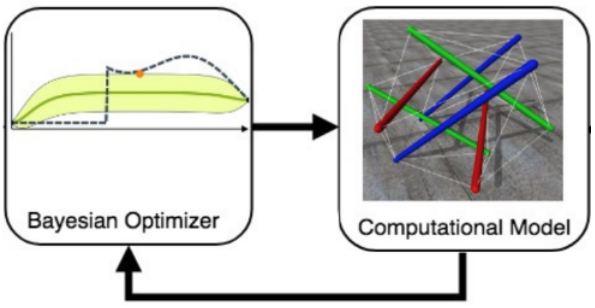


Figure 7: Gait Optimization Task[4]

6.4 Experiment Design

An experiment consists of two setups S and T, each with different environment variables such as different gravity, friction, texture, etc. or a damage in the robot such as a broken Spring (i.e., unequal/unbalanced spring forces in simulation) or a damaged strut.

1. Step 1: Run the plain Bayesian Optimization (BO) algorithm in an initial Setup S.

[4]Dr. John Rieffel, Union College, NY.

2. Step 2: Run the optimization in a modified Setup T with Plain-BO. Note the no. of trials before convergence.

3. Step 3: Run the optimization in T with Efficient-BO using knowledge transferred from S.

Ideally, in T, Efficient-BO should converge to optimal point faster than Plain-BO, since it has transferred knowledge from S.

6.5 Experimental Setups

The framework was evaluated on 3 different pairs of source and target setups. Each experiment was performed 10 times.

6.5.1 Experiment 1: Difference in Gravity and Friction

Optimization Trials: 40

Setup	S	T
Terrain	Plain Stone Ground	Plain Stone Ground
Gravity	-0.1	-0.5
Friction	0.5	0.75

Table 1: Experiment 1 Parameters

6.5.2 Experiment 2: Plain vs. Hilly Terrain

Optimization Trials: 50

Setup	S	T
Terrain	Plain Stone Ground	Hilly
Gravity	-0.1	-0.1
Friction	0.5	0.75

Table 2: Experiment 2 Parameters

6.5.3 Experiment 3: Damaged State

Optimization Trials: 40

Setup S: An undamaged tensegrity.

Setup T: A tensegrity with one strut being lighter than the other and two snapped springs.

6.5.4 Visuals of Evaluation Environments

See Appendix (Section 9.1).

7 Results

7.1 Experiment 1

7.1.1 Learning Improvement

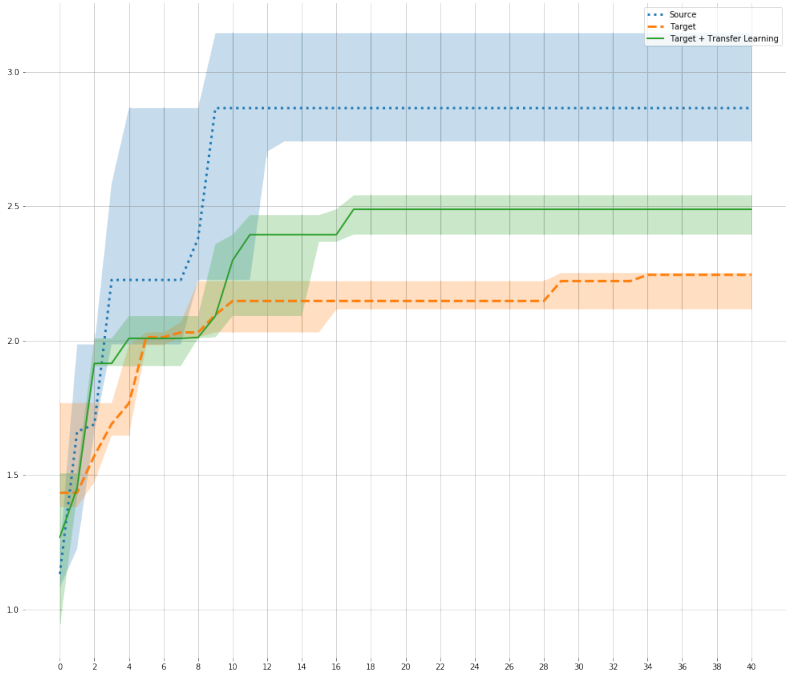


Figure 8: Experiment 1 - Speed vs. Optimization Trials

Efficient-BO achieves a **max speed 11.3% higher** than the one by Plain-BO in the same number of optimization trials. It also converges in 17 trials as opposed to 34 trials, i.e., in **50% fewer trials**.

7.1.2 Statistical Significance

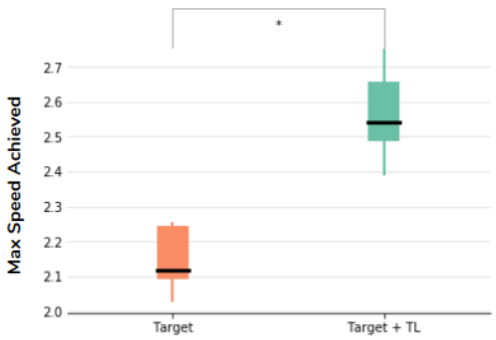


Figure 9: Experiment 1 - Statistical Significance

p-value: < 0.05

Statistically Significant

7.2 Experiment 2

7.2.1 Learning Improvement

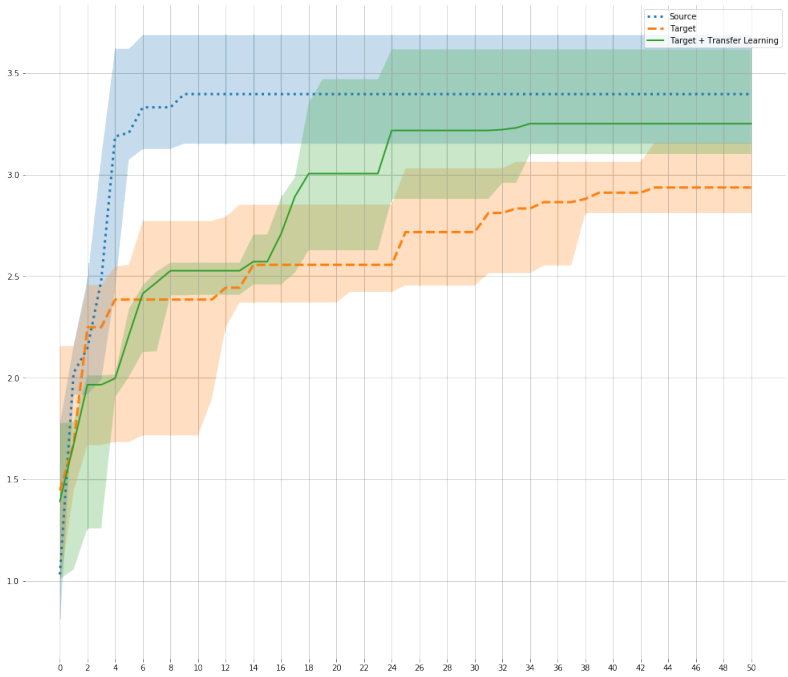


Figure 10: Experiment 2 - Speed vs. Optimization Trials

Efficient-BO achieves a **max speed 12.1% higher** than the one by Plain-BO in the same number of optimization trials. It also converges in 30 trials as opposed to 43 trials, i.e., in **23.25% fewer trials**.

7.2.2 Statistical Significance

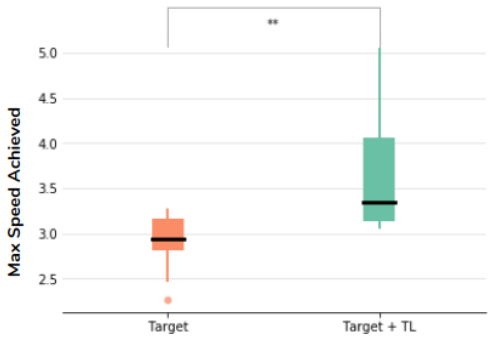


Figure 11: Experiment 2 - Statistical Significance

p-value: < 0.01

Statistically Significant

7.3 Experiment 3

7.3.1 Learning Improvement

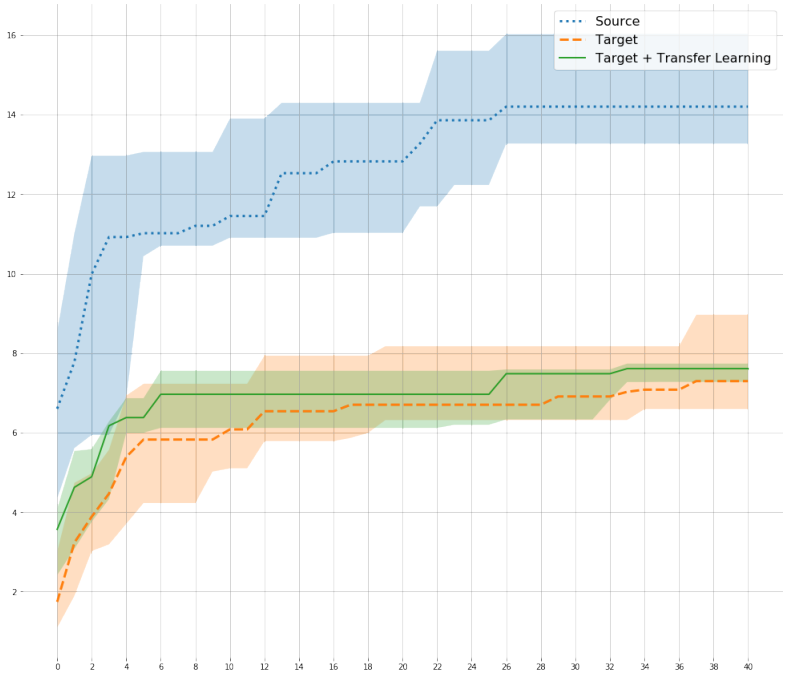


Figure 12: Experiment 3 - Speed vs. Optimization Trials

Efficient-BO achieves a **max speed 4.31% higher** than the one by Plain-BO in the same number of optimization trials. It also converges in 31 trials as opposed to 35 trials, i.e., in **11.43% fewer trials**.

7.3.2 Statistical Significance

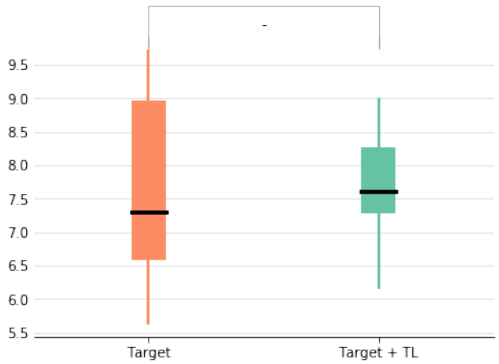


Figure 13: Experiment 3 - Statistical Significance

p-value: > 0.05

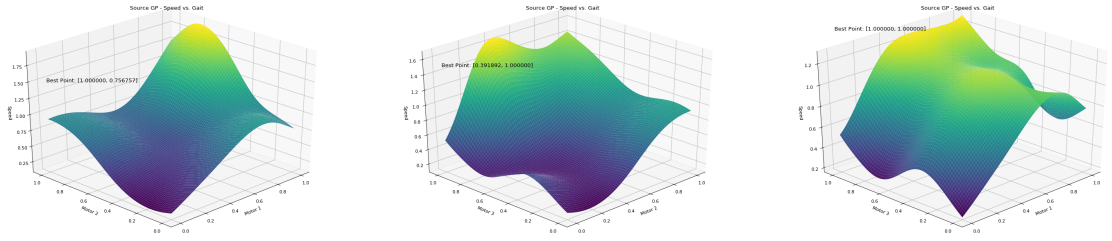
Not Statistically Significant

8 Discussion and Conclusion

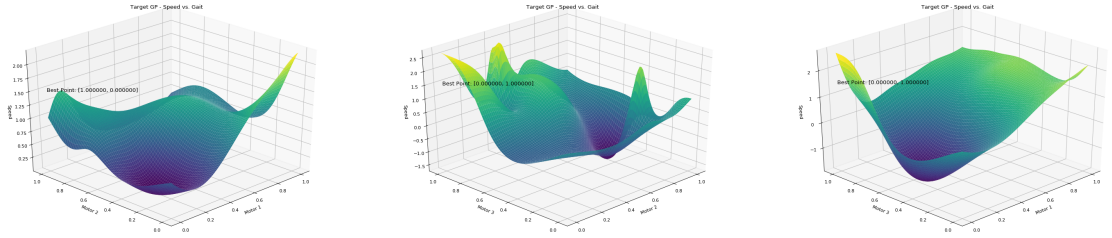
8.1 Learned Performance Functions

The performance of the robot in Experiment 1 is visualized below as a function of its gaits. Each graph represents the speed of the robot as a function of two motors with the third motor being disabled, with higher speeds in bright yellow and lower speeds in blue. It appears from these visualizations that the region of the search space the optimizer finds optimal performance in using transferred knowledge in the target task is close to where it finds optimal performance in the source task as well, indicating how the transferred knowledge biases the optimizer to explore gaits in the same regions.

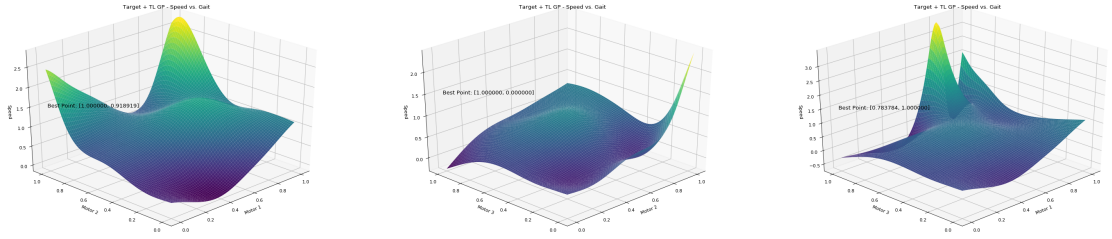
Motor 1 + Motor 2 | Motor 2 + Motor 3 | Motor 1 + Motor 3



Source Task



Target Task: Plain-BO



Target Task: Efficient-BO

8.2 Damaged State

Unfortunately, although Efficient-BO performed slightly better, the results for the damaged state turned out to be not statistically significant. This could be because the damaged tensegrity has

dynamics different from the one intact and there was therefore minimal transfer of useful knowledge. It could also be due to the damaged robot exhibiting circular motion for certain motor frequencies, in which case the displacement to reflect inaccurately for a lot of the gaits evaluated, causing a large variance in performance.

8.3 Future Work

Future work includes exploring Transfer Learning for control schemes other than just those using Bayesian Optimization. Further analysis needs to be done for the case of damaged state. A wider variety of terrains could also be experimented on, with different materials for the robot and textures for the terrain. Experiments could also include the robot being in different under the effects of water and heat. The optimization can be done for parameters other than just frequency, which would be a simple extension of the same framework. Although preliminary tests were done using an actual tensegrity robot, more thorough research could be conducted into applying this approach to real-world conditions.

8.4 Conclusion

> “Look deep into nature, and then you will understand everything better."
> - Albert Einstein

Biological systems are products of billions of years of evolution and natural selection, having shown resilience against and adaptability to permutations of the harsh natural conditions. The field of Bio-inspired Soft Robotics has the luxury of borrowing the best features of these systems to overcome some of the hardest problems faced by conventional robotics. In contributing to this endeavor, this thesis presents a solution to make tensegrity robots much more adaptable and resilient to change. The solution is easily reproducible and implementable in simulation and physical tensegrity robots. In conclusion, this research hopes to have to a positive impact on the growth of the field of Tensegrity Robotics and in making them usable for solving problems in the real world, (or as NASA hopes, on the Moon.)

9 Appendix

9.1 Evaluation Environments

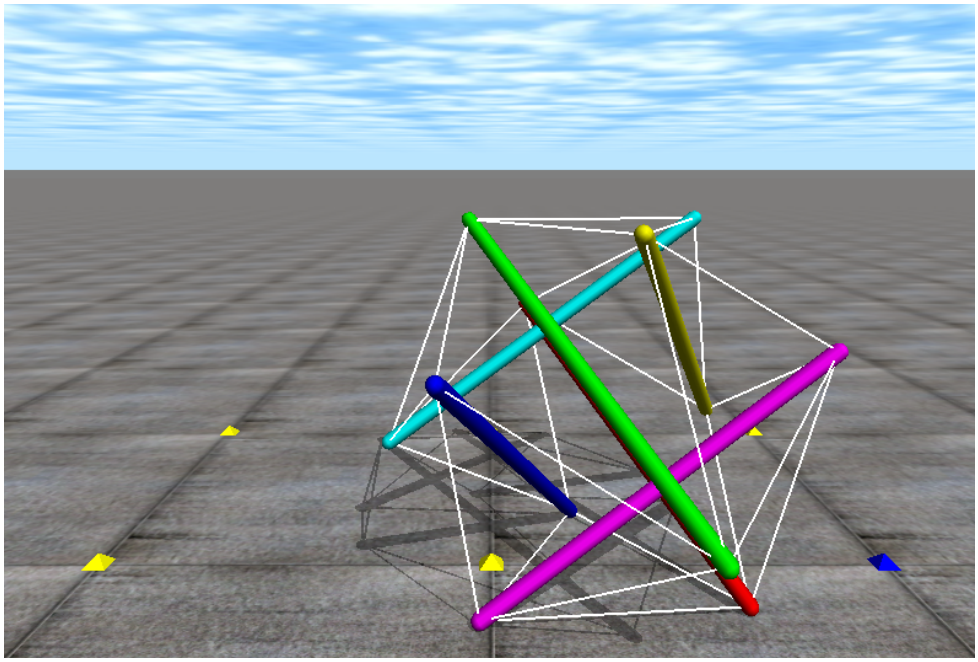


Figure 14: Plain Ground Terrain

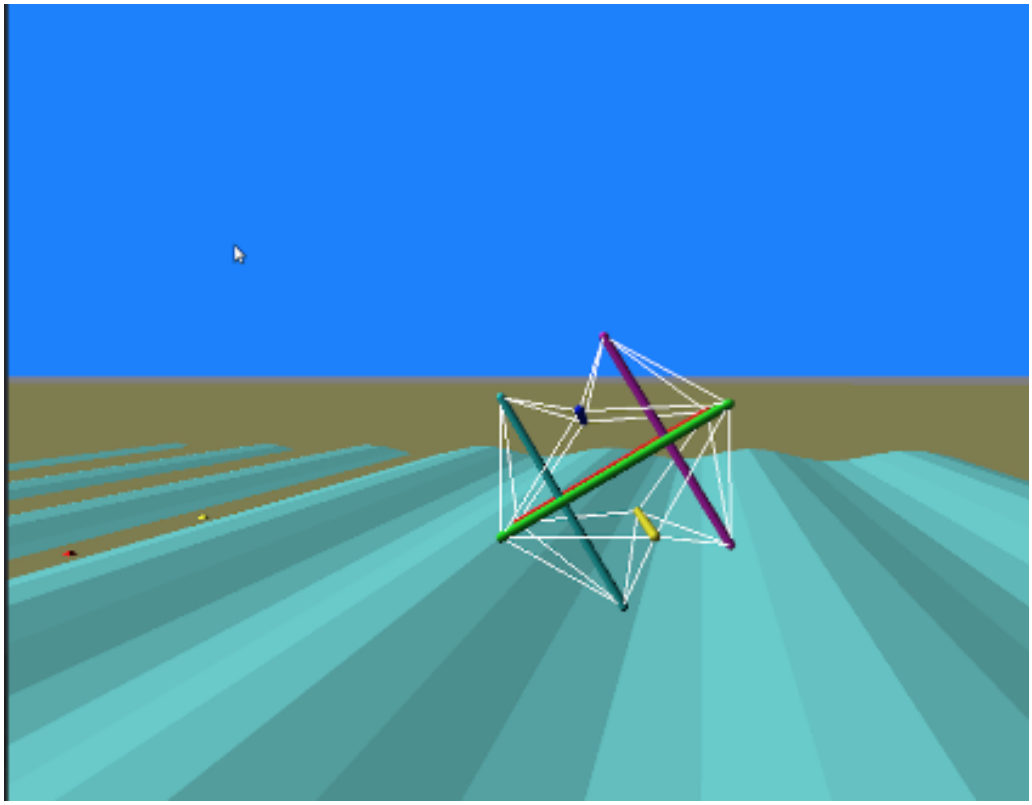


Figure 15: Rough Hilly Terrain

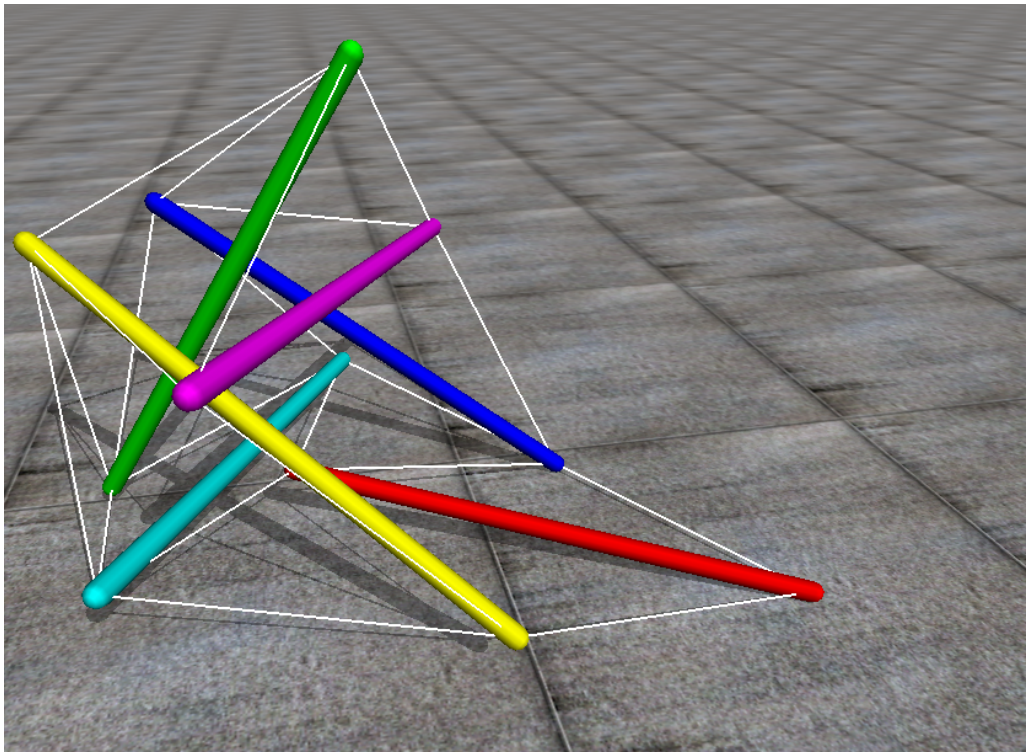


Figure 16: Broken Tensegrity Robot

9.2 Algorithm

Algorithm 1 Transfer Learning Framework

procedure RUN-TENSEGRITY(T, g, s)
    T.SET-GAIT(g)

    start_time ← GET-CURRENT-TIME( )
    start_location ← T.GET-CURRENT-LOCATION( )

    while GET-CURRENT-TIME( ) < (start_time + s) **do**
        T.MOVE-FORWARD( )
    end while

    end_location ← T.GET-CURRENT-LOCATION( )
    speed ← (end_location - start_location) / s

    return speed
end procedure

procedure EFFICIENT-BO(X^S, Y^S, N)
    GP^S ← CREATE-NEW-GP( )
    GP^T ← CREATE-NEW-GP( )

    GP^S.FIT-DATA(X^S, Y^S)

    $X^T = \{g_{random}\}$
    $Y^T = \{$RUN-TENSEGRITY(g_{random})$\}$

    for i = 1 **to** N **do**
        $g_i \leftarrow \text{argmax}_{g \in D^3} EI(g) | \{X^T, Y^T\}$
        $y_i \leftarrow$ RUN-TENSEGRITY(g_i)

        $\hat{y}_i^S \leftarrow GP^T$.PREDICT($x_i$)
        $\sigma_S^2 \leftarrow$ COMPUTE-SOURCE-NOISE-VARIANCE( )
        GP^S.K ← GP^S.K + σ_S^2

        $X^T \leftarrow X^T \cup g_i$
        $Y^T \leftarrow Y^T \cup y_i$
    end for

    $g^{opt} \leftarrow \text{argmax}_g$ RUN-TENSEGRITY(g)
end procedure

References

[1] Ingber, D.E., “Cellular tensegrity: defining new rules of biological design that govern the cytoskeleton,” In *Journal of Cell Science*, Vol. 104, 1993.

[2] M.S. Levin, “The tensegrity-truss as a model for spinal mechanics: Biotensegrity", In *Journal of Mechanics in Medicine and Biology*, Vol. 2, No. 3, pp 375-388, 2002.

[3] S. H. Juan and J. M. M. Tur, “Tensegrity frameworks: Static analysis review,” In *Mechanism and Machine Theory*, vol. 43, no. 7, pp. 859 – 881, 2008.

[4] N. Bel Hadj Ali, L. Rhode-Barbarigos, A. Pascual Albi, and I. Smith, “Design optimization and dynamic analysis of a tensegrity-based footbridge,” Engineering Structures, vol. 32, no. 11, pp. 3650–3659, 2010.

[5] C. Paul, F. J. Valero-Cuevas, and H. Lipson, “Design and control of tensegrity robots for locomotion,” IEEE Transactions on Robotics, vol. 22, no. 5, 2006

[6] E. Marder, D. Bucher, “Central pattern generators and the control of rhythmic movements," In *Current Biology*, Volume 11, Issue 23. 2001.

[7] T. Bliss, T. Iwasaki, and H. Bart-Smith, “Central pattern generator control of a tensegrity swimmer,” IEEE/ASME Transactions on Mecha- tronics, vol. 18, no. 2, pp. 586–597, 2013.

[8] J. Rieffel and JB. Mouret. “Soft Tensegrity Robots." *Awaiting Publication.* 2016.

[9] L. Torrey and J. Shavlik (2009). Transfer Learning. In *Handbook of Research on Machine Learning Applications*. 2009.

[10] T.T. Joy, Rana S., Gupta S.K., Venkatesh S. (2016) Flexible Transfer Learning Framework for Bayesian Optimisation. In *Advances in Knowledge Discovery and Data Mining*. PAKDD 2016.

[11] X. Geng, M. Zhang, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine. “Deep Reinforcement Learning for Tensegrity Robot Locomotion." In Review at *International Conference on Robotics and Automation (ICRA)*. 2017.

[12] M. Gönen and A.A. Margolin. “Kernelized Bayesian transfer learning." In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[13] J. Davis and P. Domingos. “Deep transfer via second-order markov logic." In *Proceedings of the 26th annual ACM International Conference on Machine Learning.* 2009.

[14] C. Finn, T. Yu, J. Fu, P. Abbeel, and S. Levine. "Generalizing Skills with Semi-Supervised Reinforcement Learning." In *Proceedings of the 5th annual International Conference on Learning Representation.* 2017.

[15] P. Zhao, S. Hoi, J. Wang, B. Li. "Online Transfer Learning." In *Artificial Intelligence*. Vol 216. Pg. 76-102. 2014.

[16] https://www.nasa.gov/sites/default/files/files/Agogino_SuperBallBot.pdf

[17] https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140010029.pdf

[18] https://github.com/hawk31/pyGPGO