Transfer Learning for Faster Tensegrity Gait Optimization

Akshay Kashyap
Advisor: Dr. John Rieffel

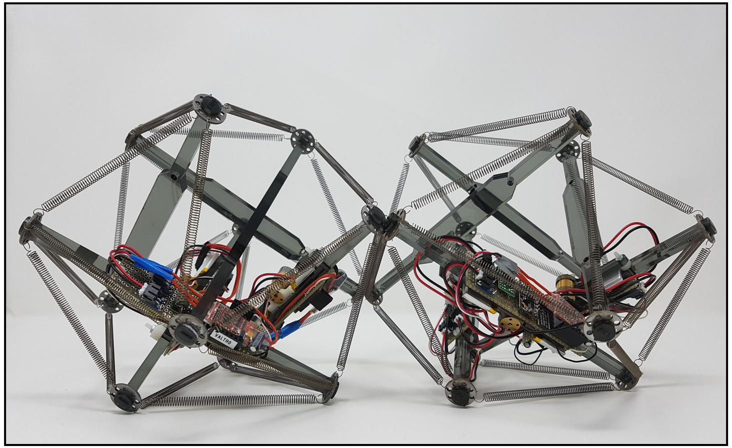
Question: Tensegrity Robots can use Machine Learning to learn how to move efficiently. Can we make them learn better and faster, especially in new, unseen conditions?

Proposed Solution: Use Transfer Learning, a subfield of Machine Learning, to have the robots use previous learning experiences to adapt better and learn faster.

0. Background

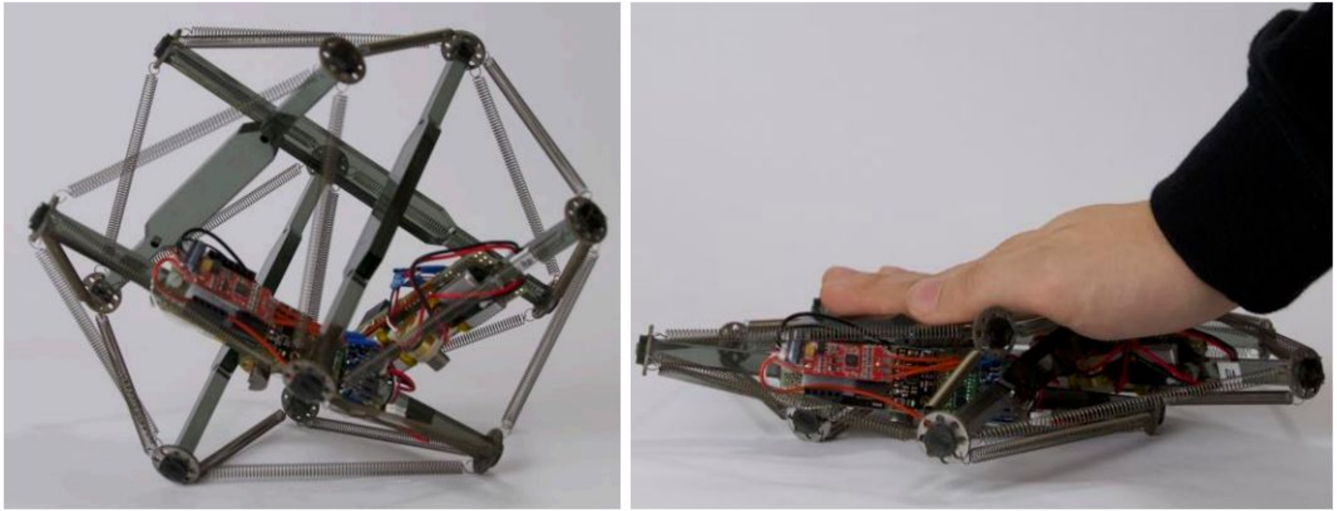
0.1 Tensegrity Robots

- A class of soft robots composed of intertwined springs and rigid struts.
- **Tensegrity** = **Tens**ile + Int**egrity**



0.1 Tensegrity Robots

- Can carry payloads in the center and can be dropped from heights.
- NASA exploring use for planetary missions.



0.1 Tensegrity Robots

- Can be made to move using vibrations from attached motors.
- **Tensegrity Gait** = Configuration of Motors

$$Gait = (m_1, m_2, ..., m_n), \text{ where } n = \text{No. of Motors}$$

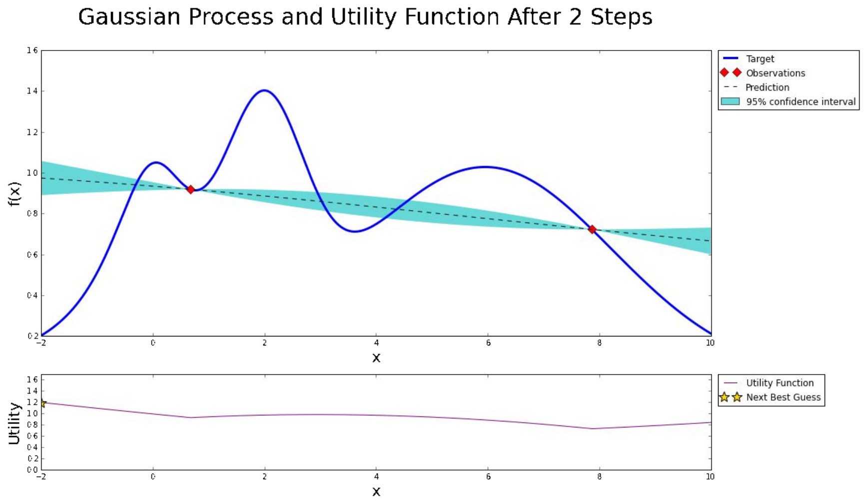
and

$$m_i = (phase, frequency, amplitude)$$

- **Gait Performance** = **Speed** / Distance Travelled / ...

0.2 Bayesian Optimization

- A Sequential Model-based Optimization (SMBO) algorithm for optimizing expensive functions using a Gaussian Process.
- Works by evaluating the function systematically at different points and trying to update it’s prediction of what the function looks like at each step.



0.2 Bayesian Optimization

- Bayesian Optimization can be used to train a Tensegrity Robot Gait to move make it move as fast as possible.

0.2 Bayesian Optimization

- Bayesian Optimization can be used to train a Tensegrity Robot Gait to move make it move as fast as possible.
- Problem: Needs to be trained in every new environment and abnormal state the robot is deployed in.

0.2 Bayesian Optimization

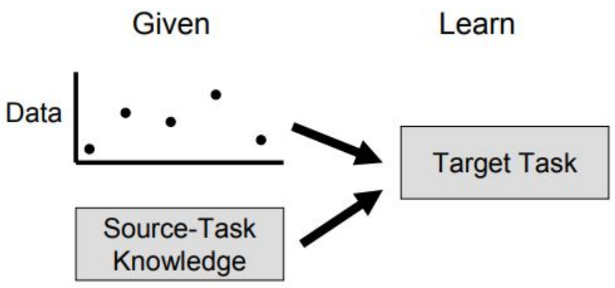
- Bayesian Optimization can be used to train a Tensegrity Robot Gait to move make it move as fast as possible.
- Problem: Needs to be trained in every new environment and abnormal state the robot is deployed in.
- (Proposed) Solution: Use Transfer Learning.

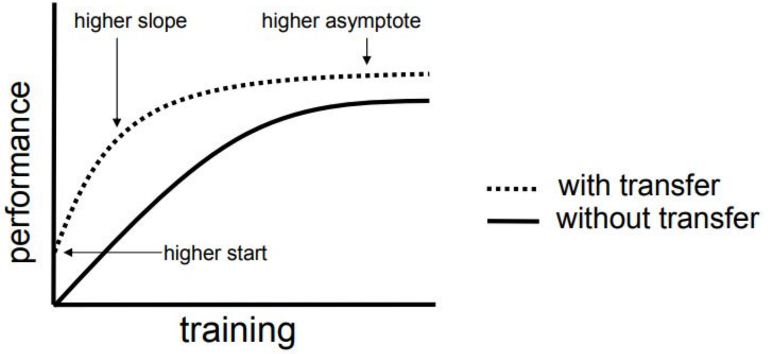
0.3 Transfer Learning

Transfer learning = Improvement of learning in a new task using knowledged from related task that has been learned previously. [2]

0.3 Transfer Learning

- **Source Task** = Task that has been previously learnt
- **Target Task** = New task to learn
- Use Source Task to improve learning in Target Task





0.4 Proposed Solution

- Use Transfer Learning along with Bayesian Optimization.
- Framework proposed by T.T Joy, et. al. [3]
- Model observations from source task as noisy outputs of target task:

$$y_i^s = f^t(x_i^s) + \epsilon_i^s$$

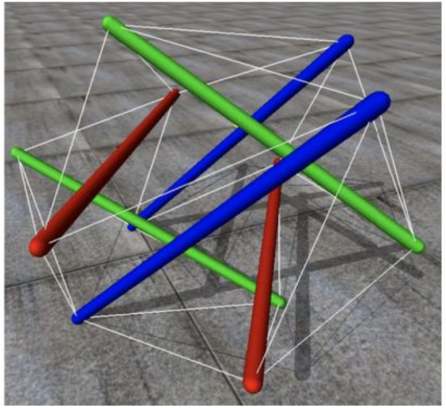
where x_i^s is the source observation, f^t the target function, and ϵ_i^s the noise.

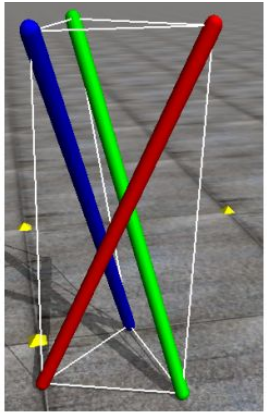
- Modify the Gaussian Process to incorporate this noise during optimization.

1. Tools

1.1 Tensegrity Simulation

- Homegrown C++ and ODE-based Tensegrity physics simulator.
- Models struts as capsule and springs as forces following Hooke's Law.
- Models each motor as a perpendicular force applied periodically at points along the circumference of the strut.

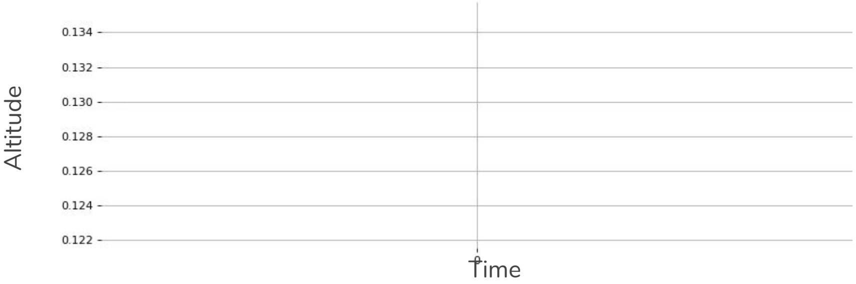




```
Motor::Motor() {
    _step = 0;
    _frequency = PI / 8;
    _limit = (2 * PI) / _frequency;
    _speed = 1.0;
}

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;}
```

1.1 Tensegrity Simulation

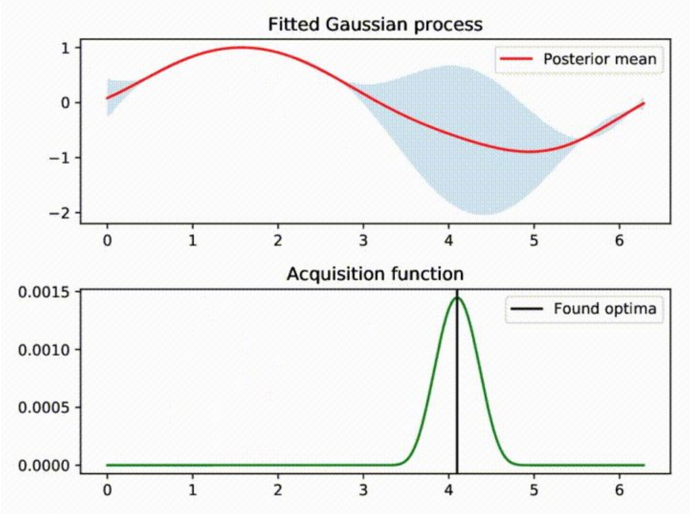


1.3 Bayesian Optimization and Transfer Learning

- Used the Python **PyGPGO** library for Bayesian Optimization.
- Re-engineered it to implement the Transfer Learning framework.

pyGPGO: Bayesian Optimization for Python

build passing | codecov 80% | docs passing | DOI 10.5281/zenodo.1040676 | JOSS 10.21105/joss.00431

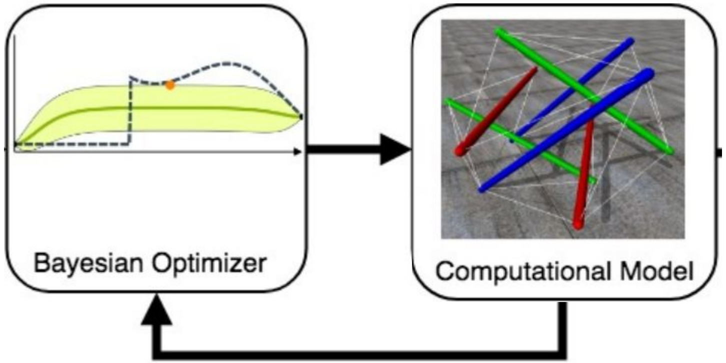


pyGPGO is a simple and modular Python (>3.5) package for bayesian optimization.

2. Methodology

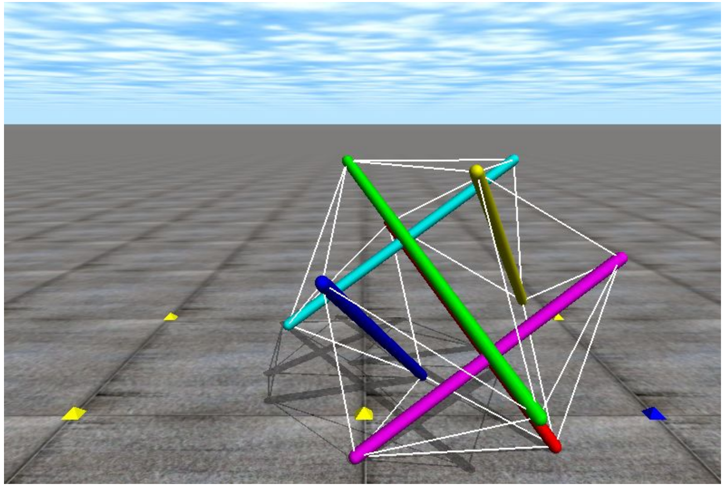
2.1 Simulator + Optimizer System

- The optimizer program communicates with the simulator using **Sockets**.
- Optimizer sends out gaits to evaluate. Simulator evaluates and sends back performance of the gait.

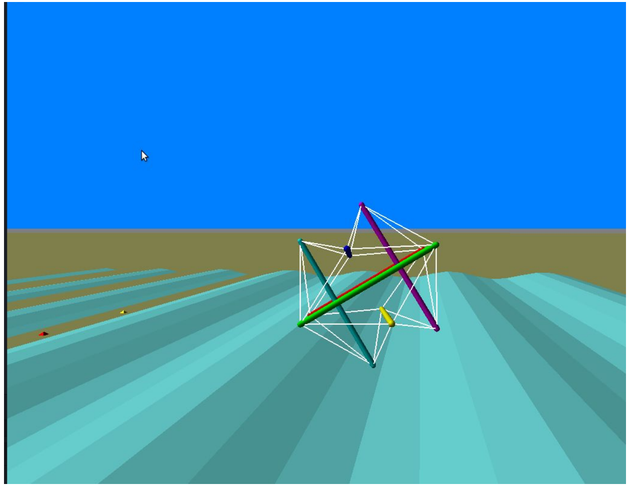


2.2 Experiment

- Perform the optimization process for **Source Task**, **Target Task without Transfer Learning**, and **Target Task with Transfer Learning**.
- Perform $n = 50$ optimization cycles for each task. Perform 10 experiments.



Plain Ground



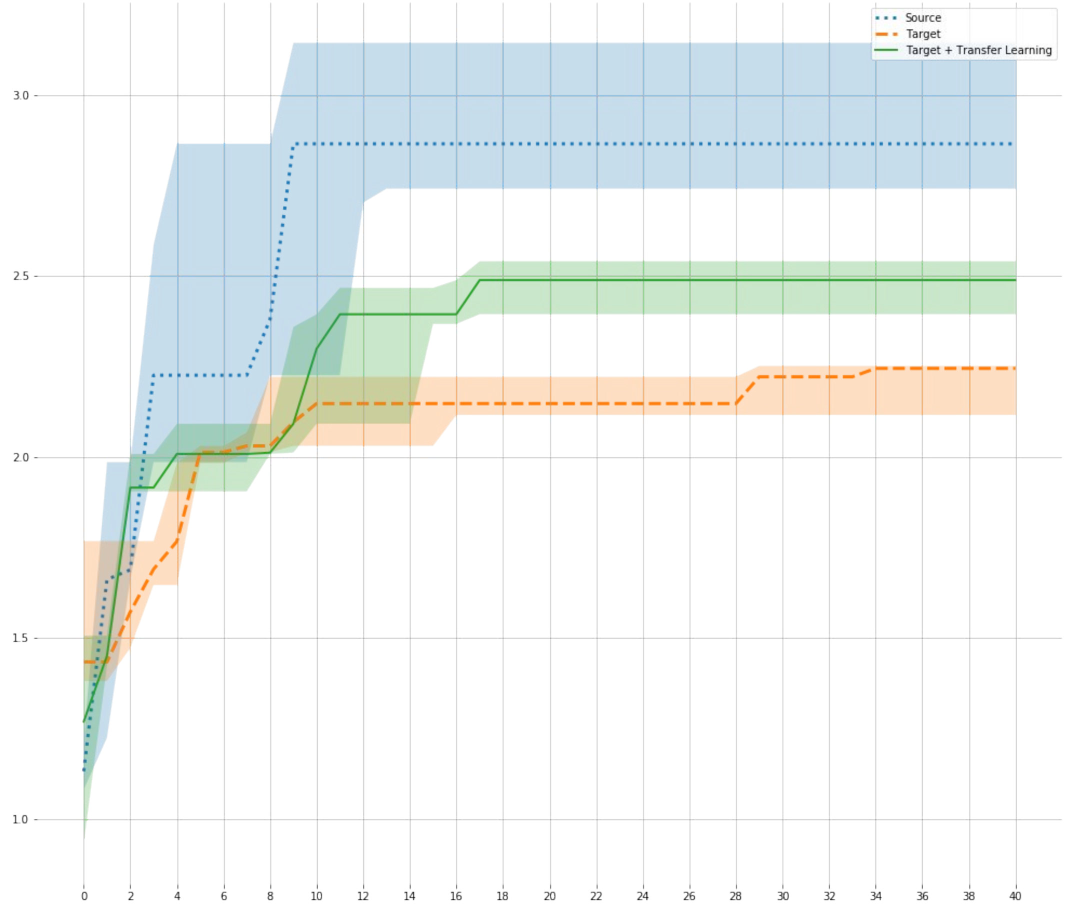
Hilly Terrain

3. Results

3.1 Learning Improvement

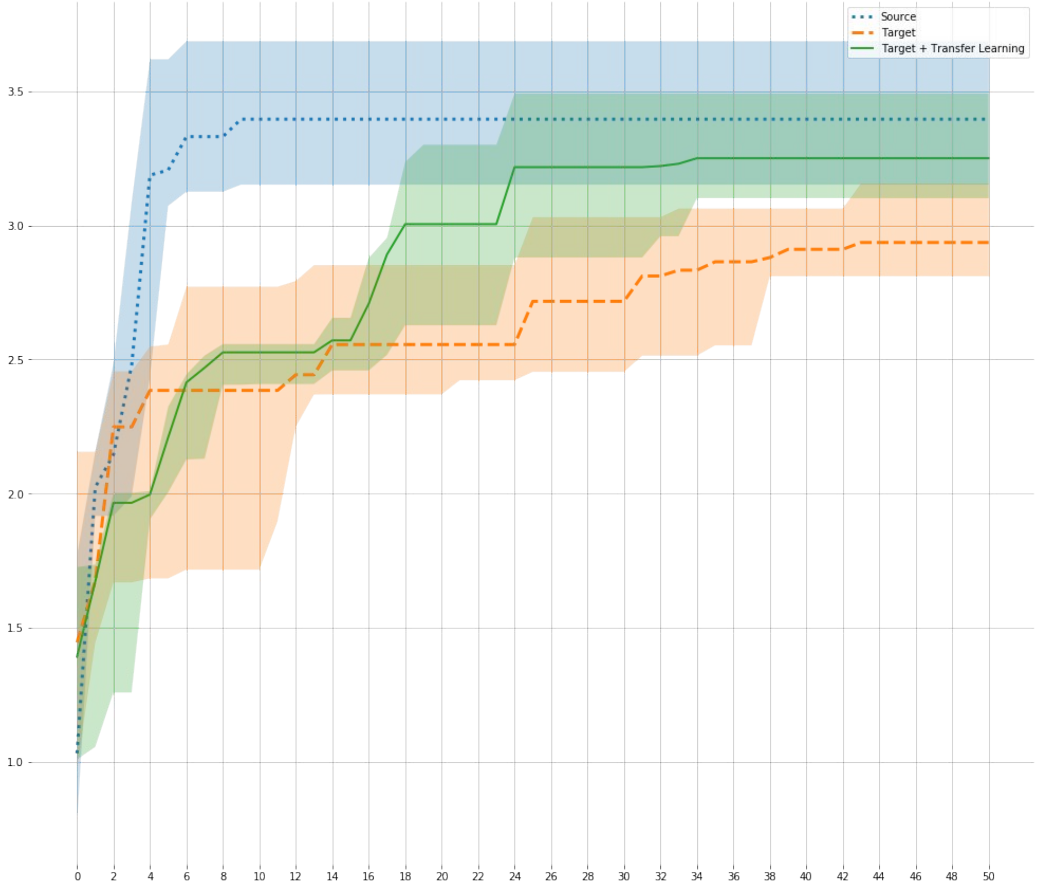
3.1.1 Experiment 1 - Difference in Gravity and Friction

- **Source Task**:
  - Gravity = -0.1
  - Friction = 0.5
- **Target Task**:
  - Gravity = -0.5
  - Friction = 0.75
- **40** Optimization Trials
- **Metric:** Max Speed Achieved
- **10% Improvement**



3.1.1 Experiment 2 - Flat vs. Hilly Terrain

- **Source Task**:
  - Flat Surface
  - Gravity = -0.1
  - Friction = 0.5
- **Target Task**:
  - Hilly Surface
  - Gravity = -0.1
  - Friction = 0.75
- **60** Optimization Trials
- **Metric:** Max Speed Achieved
- **12.1% Improvement**



3.2 Learnt Gaits

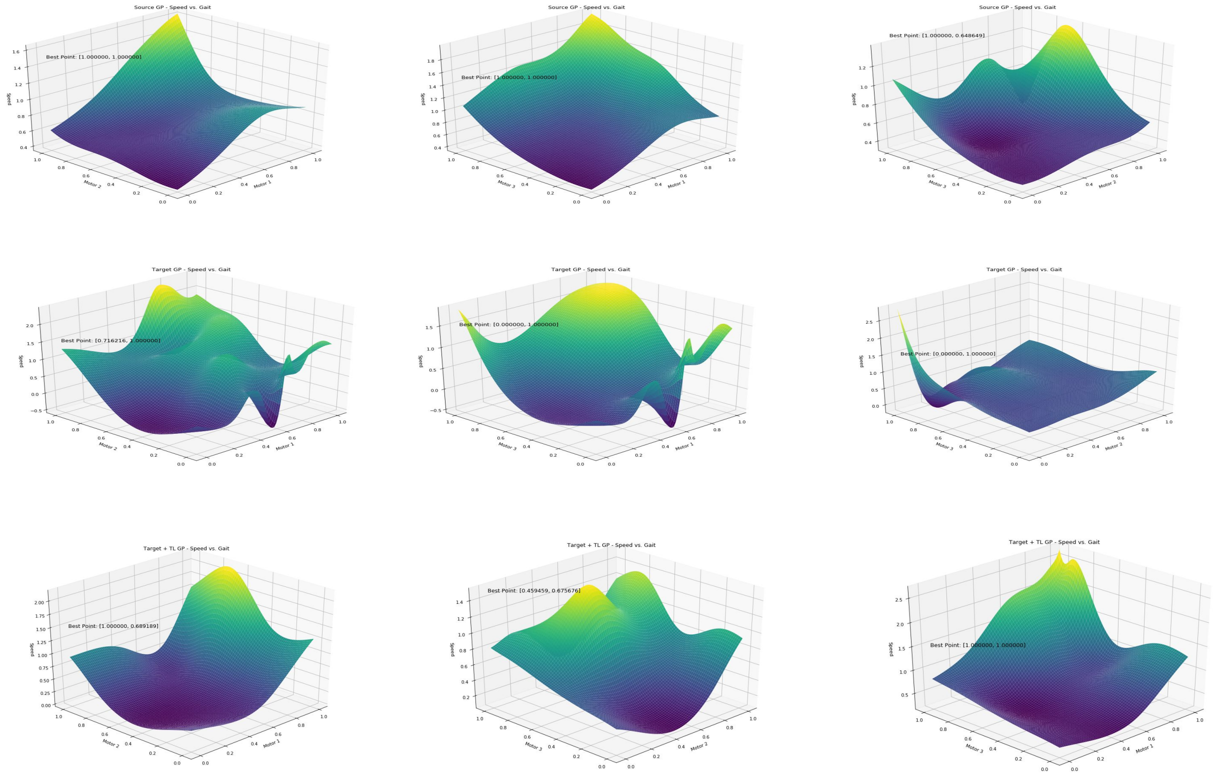
3.2.1 Experiment 1 - Difference in Gravity and Friction

M1 - M2 | M2 - M3 | M1 - M3

Source Task

Target Task

Target + TL



3.2.2 Experiment 2 - Flat vs. Hilly Terrain

M1 - M2

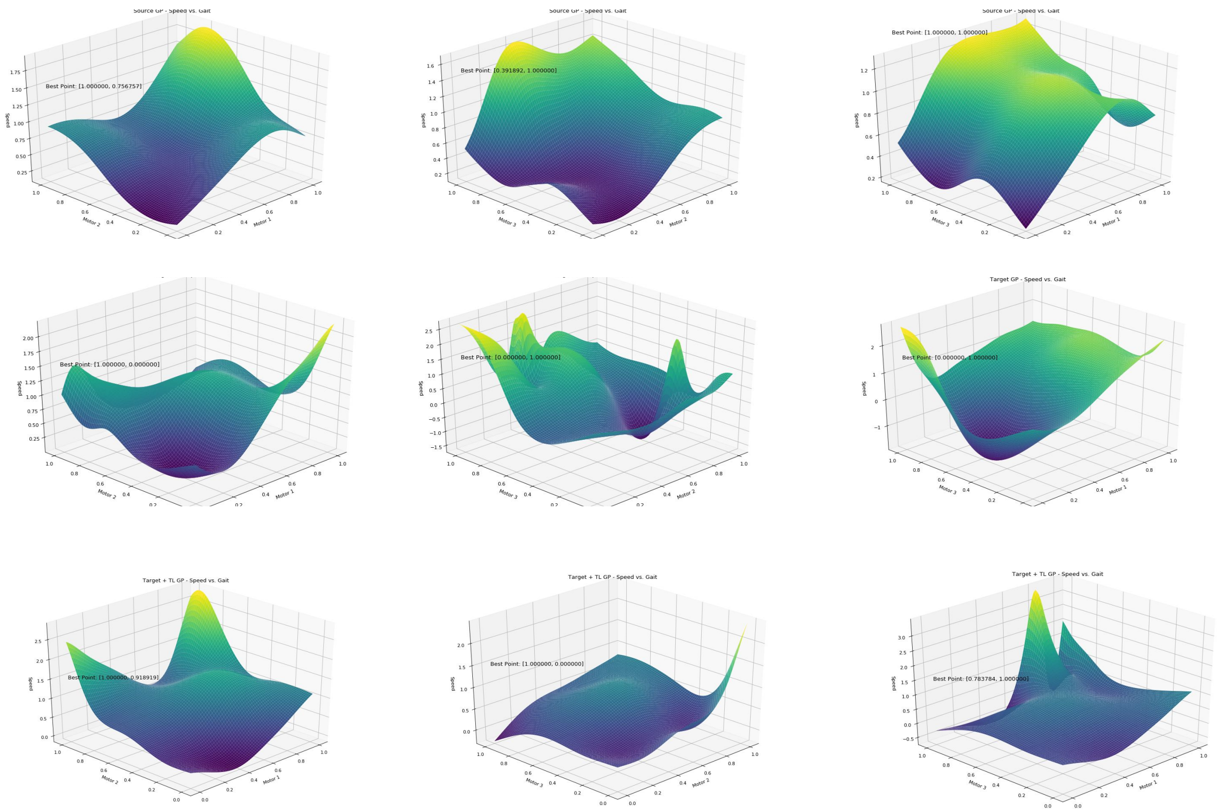
M2 - M3

M1 - M3

Source Task

Target Task

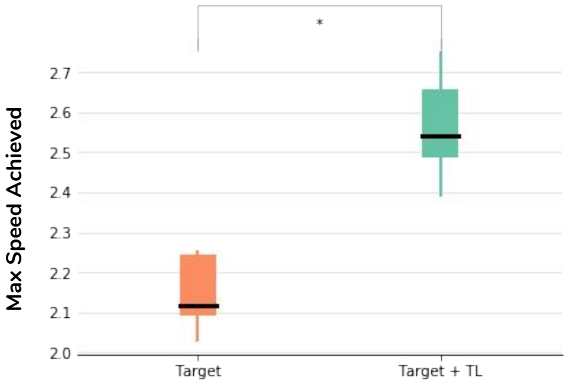
Target + TL



3.3 Statistical Significance

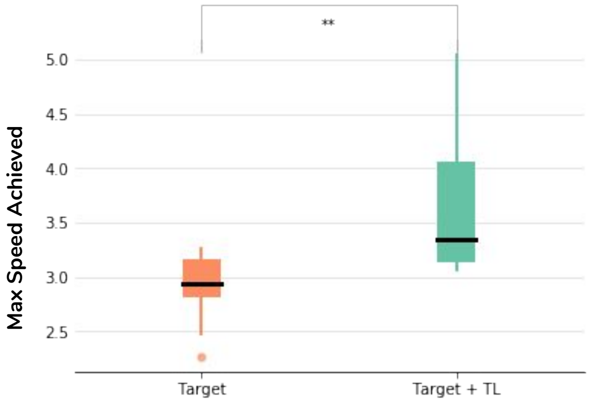
Using **Mann–Whitney U test**

Experiment 1



p-value < 0.05
95% Confidence Interval

Experiment 2



p-value < 0.01
99% Confidence Interval

4. Conclusion

4.1 Conclusion

In conclusion, my research shows that previous learning experiences indeed can be leveraged to improve new learning tasks for Tensegrity Robots in the context of locomotion.

4.2 References

[1] John Rieffel and Jean-Baptiste Mouret. Adaptive and resilient soft tensegrity robots. *Soft Robotics*, page (to appear), 2018. arXiv preprint arXiv:1702.03258.

[2] Lisa Torrey and Jude Shavlik (2009). Transfer Learning. In *Handbook of Research on Machine Learning Applications*. 2009.

[3] Joy T.T., 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.

May the force be with you!