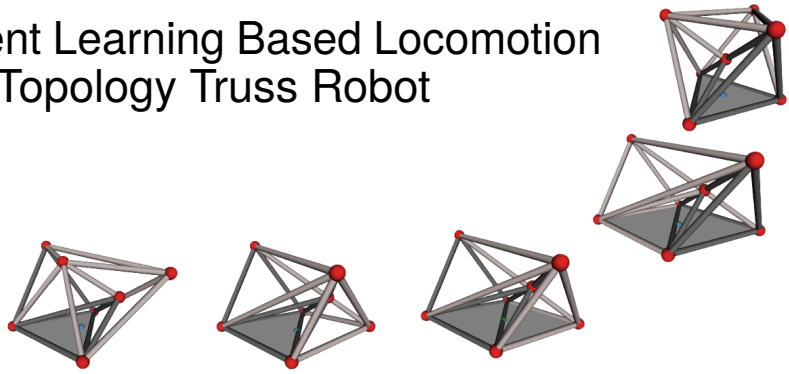
Reinforcement Learning Based Locomotion for Variable Topology Truss Robot

Ozgur Gulsuna

29 May 2024



ODTÜ METU **MMI**706 **Reinforcement** Learning

Outline

Outline

Locomotion Strategy for the VGT Robot

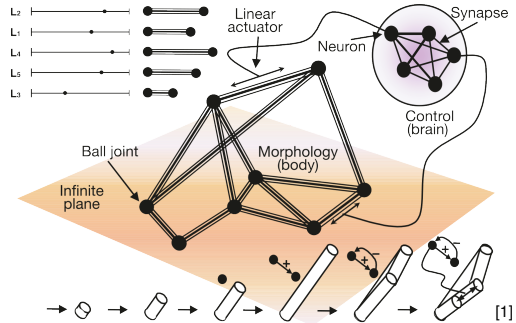
RESEARCH QUESTION

Development of a reinforcement learning based locomotion strategy for truss robots. This involves coordinating the movement of various parts in a specific sequence, similar to how serpents or mammals move, where intricate coordination is necessary.

Idea Development

The structure consists of N linear actuators, each with a single degree of freedom (DoF), interconnected with passive joints.

1. *Model Free RL:* Single topology, baseline for comparison. ✓
2. *Modular Control Policy RL:* Multiple topology, generalizability. ✓
3. *Graphing Neural Networks:* RL topology that is structured from the graph of the robot topology.



[1]

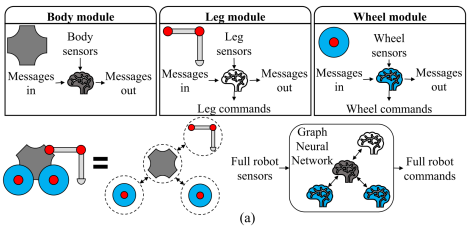
Literature Review on Truss Robots and Reinforcement Learning

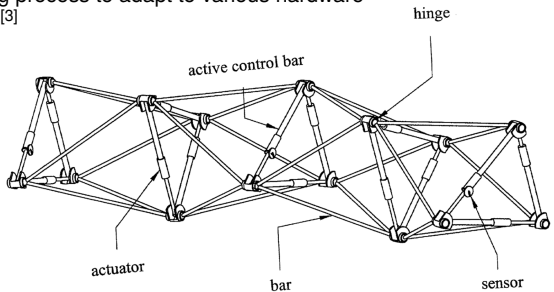
[0][Non-impact Rolling Locomotion of a Variable Geometry Truss] [2019]

The velocity of nodes is optimized to move the center of mass given the desired velocity. The member velocities are calculated and actuated accordingly.[2]

[2][Learning Modular Robot Control Policies] [2023]

Modular robots need specific control policies for each design, which becomes impractical for scalability. A modular policy framework allows for a single training process to adapt to various hardware arrangements and control different designs efficiently.[3]





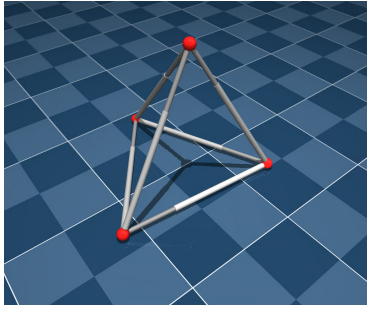
Outline

Variable Geometry Truss Robot Description

[MuJoCo]

The robots are described by an XML file, and it is structured in a tree form. This file is read by MuJoCo and it contains the actuator information, joint information, as well as the simulation parameters.

- Procedural script that can create a variety of robots. ✓
- Coordinated or Dynamic Locomotion
  - Each actuator is position controlled. ✓
  - Each actuator is force controlled. ✓
- The aim is to control many different truss robots with a single training. ✓
  - *Vision*: Change in the topology, due to constraints in space or a member failure.



Improvement: Since we have the robot generation script, robots can now be initialized with random configurations, allowing for faster and more versatile training across various scenarios.

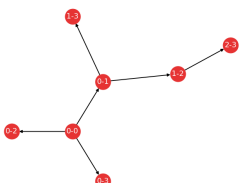
Outline

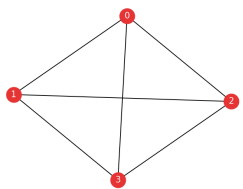
Procedural Truss Robot Generation

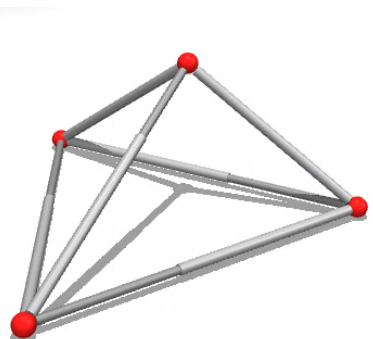
[1-TET]

The tetrahedron is first modelled, which has 4 nodes and 6 members. All of the members are active and those are constructed as linear actuators in MuJoCo.

- $N = 4$: number of nodes
- $M = 6$: number of members







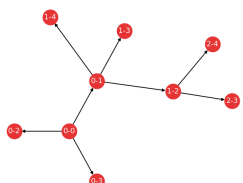
First the topology is given as connectivity diagram, then it is converted to a tree to be put in the robot definition format.

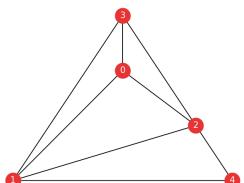
Procedural Truss Robot Generation

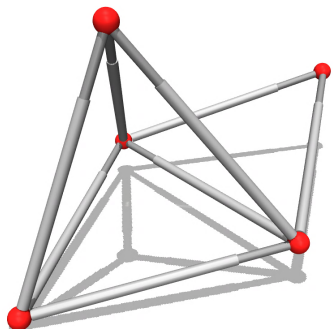
[1-TET-1]

The 1-tetrahedron-1 has the added triangle tail at one of its edges. The tail is passive, it can be used to gain momentum.

- $N = 5$: number of nodes
- $M = 8$: number of members







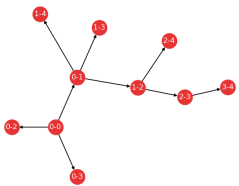
The problem is now to connect members with random lengths in accordance to the connectivity matrix. This problem has many solutions one and requires some constraints.

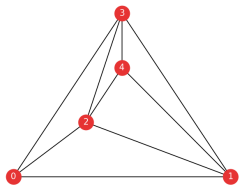
Procedural Truss Robot Generation

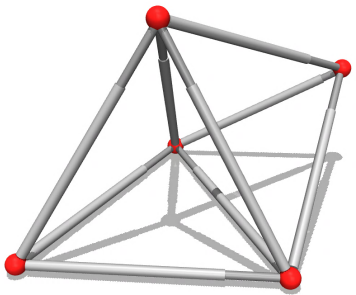
[2-TET]

The 2-tetrahedron is two tetrahedrons that are connected from a face. It is a relatively rigid body.

- $N = 5$: number of nodes
- $M = 9$: number of members







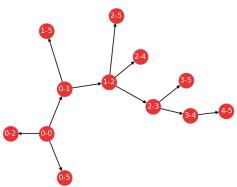
This minimization problem of defining a valid polyhedra is solved with scipy.optimize library. The result is the calculated coordinates of model geometry.

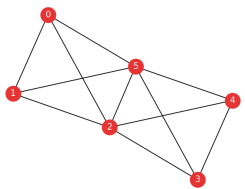
Procedural Truss Robot Generation

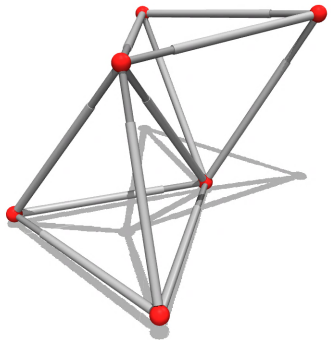
[2-TET-1]

The 2-tetrahedron-1 is two tetrahedrons that are connected from an edge. This time it is not a rigid body, hence it is similar to a large tail.

- $N = 6$: number of nodes
- $M = 11$: number of members







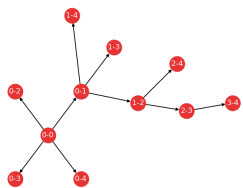
The XML file is also generated with this script. For the random member length cases, actuators initial positions should also be calculated.

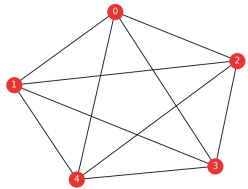
Procedural Truss Robot Generation

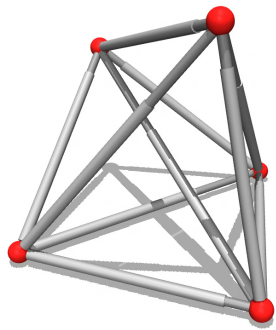
[2-TET-2]

The 2-tetrahedron-2 is two tetrahedrons that are connected from two edges, but not a face. There is the inside member that alleviates stiffness of the system.

- $N = 6$: number of nodes
- $M = 10$: number of members







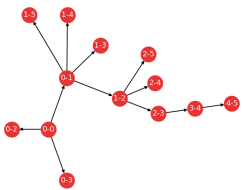
The equality constraint is used to convert the tree format of the XML file to a closed kinematic chain. This requires the calculation of anchors for the equality constraints.

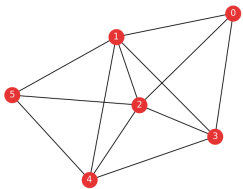
Procedural Truss Robot Generation

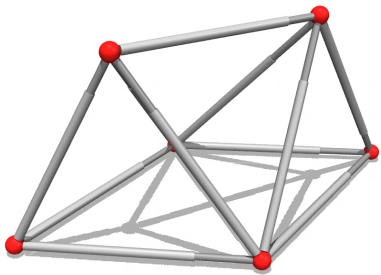
[3-TET]

The 3-tetrahedron is three tetrahedrons that are connected in-line with each other from the faces. The use of tetrahedrons bring rigidity to the final shapes.

- $N = 6$: number of nodes
- $M = 11$: number of members







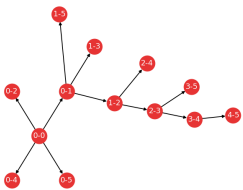
The linear actuators are used in force control mode this time. This is because the range of each member is different and it is not possible to scale them all in a single range.

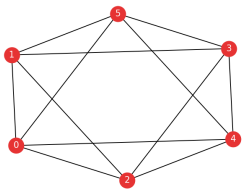
Procedural Truss Robot Generation

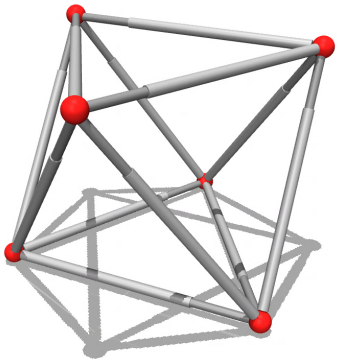
[OCTAHEDRON]

The octahedron is the platonic solid that is similar to tetrahedron, has triangles as the faces. The structure is stable and allows for large range of movements.

- $N = 6$: number of nodes
- $M = 12$: number of members







Octahedron is the main topology that is used for locomotion, it is used in optimized locomotion environment in previous research works.

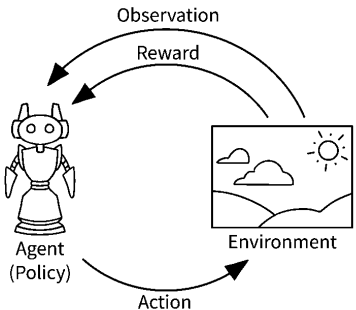
Outline

Reinforcement Learning Problem

[OpenAI-Gym]

For an initial test of the simulation tools and the robot model, a simple training environment and problem needed to be devised. The initial problem selected for the truss robot locomotion is a simple one, go right as fast as possible.

- Planar terrain, no ground sensing.
- Traverse in the positive x-axis, without a target point.
- Reward for maintaining health and moving to the right.
- In the next experiments, try:
  - Adding a target destination point.
  - Introducing a moving target point.
  - Aiming for specific foot nodes.
- Proper reward mechanism design leads to more advanced locomotion.



Outline

Reinforcement Learning Agent

[OpenAI-Gym]

This agent consist of a tetrahedron robot where the each edge (**member**) is a linear actuator and the vertices (**nodes**) are the passive joints. The robot is controlled by changing the length of the linear actuators either dynamically or by changing the position of the center of mass of the robot.

Num	Action	Min	Max	Name	Joint	Unit
01	force	-1	1	[Member_0-1]	slide	newtons (N)
02	force	-1	1	[Member_0-2]	slide	newtons (N)
03	force	-1	1	[Member_0-3]	slide	newtons (N)
⋮				⋮		⋮
14	force	-1	1	[Member_3-5]	slide	newtons (N)
15	force	-1	1	[Member_4-5]	slide	newtons (N)

Num	Observation	Min	Max	Name	Joint	Unit
01	x-coord of CoM	-inf	inf	coord_x	free	(m)
02	y-coord of CoM	-inf	inf	coord_y	free	(m)
03	z-coord of CoM	-inf	inf	coord_z	free	(m)
04	x-vel of CoM	-inf	inf	vel_x	free	(m/s)
05	y-vel of CoM	-inf	inf	vel_y	free	(m/s)
06	z-vel of CoM	-inf	inf	vel_z	free	(m/s)
07	length of [0-1]	-inf	inf	length_1	slide	(m)
⋮	⋮			⋮		⋮
21	length of [4-5]	-inf	inf	length_15	slide	(m)
22	x-coord of [0]	-inf	inf	node_1_x	free	(m)
⋮	⋮			⋮		⋮
39	z-coord of [5]	-inf	inf	node_6_z	free	(m)

Outline

Reinforcement Learning Training

[Stable-Baselines3]

Training requires the use of specific parameters, which have a significant impact on the algorithm's performance. Therefore, tuning is generally necessary. The SAC algorithm was previously used, this time PPO is also tested and compared with the prior.

- $forward\_reward$: Multiplier for rewarding forward movement.
- $healthy\_reward$: Reward earned per time step when the robot is healthy.
- $size\_cost\_weight$: Penalty for increasing the size of the robot.
- $episode\_horizon$: Length of each episode; longer episodes allow for more learning and adaptation time but increase training duration.

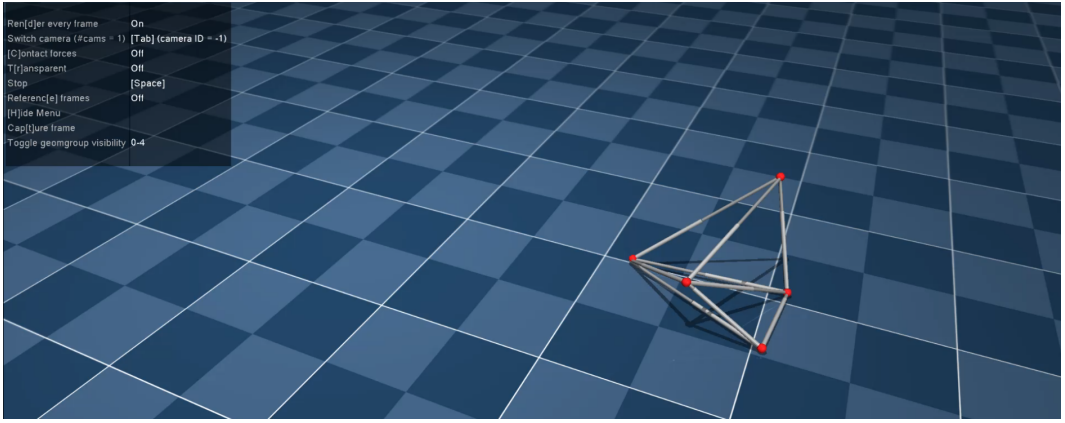
Note: Randomizing the starting configuration of the robot offers advantages. However, our current robot configuration complicates the calculation of various member length tetrahedron geometries and their conversion to qpos.
Note: Terminating unhealthy or glitchy episodes is crucial because the algorithm might inadvertently exploit glitches to propel itself in a particular direction.

Outline

[Single Agent Training][00]

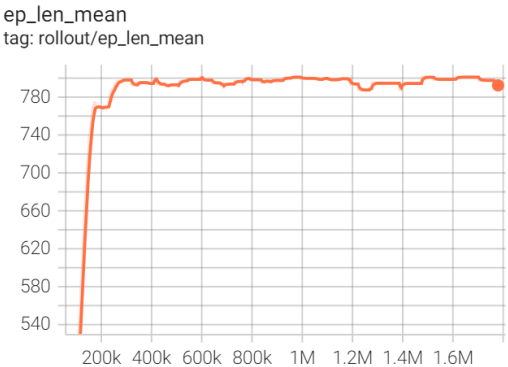
[2-TET]

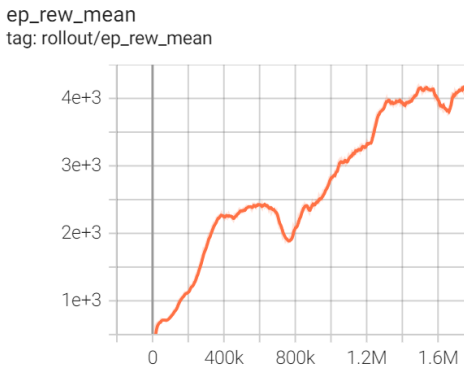
Here we have the 2-tetrahedron training, this is more complicated than compared with tetrahedron. In this training, the model is more dynamic, allowing for more fluent rotary tumbling motion of the agent.



[Single Agent Training][00]

[2-TET]

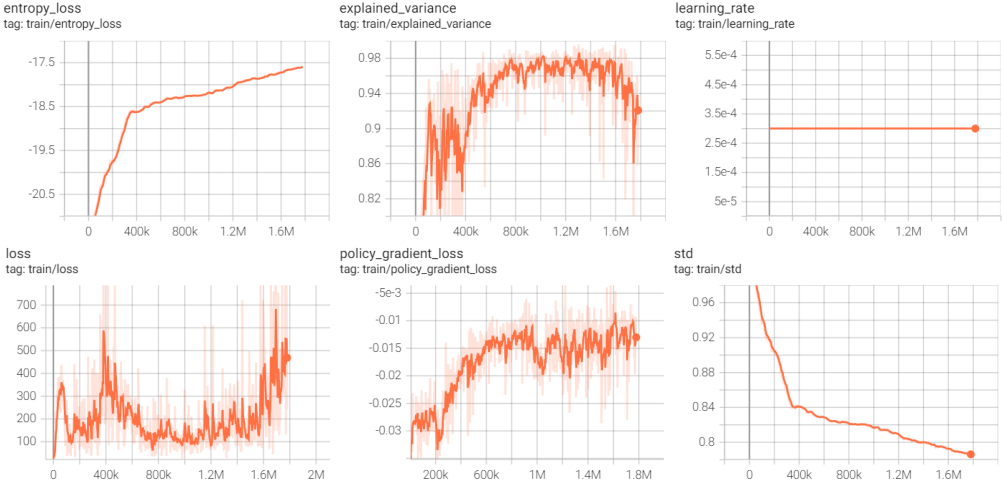




PPO algorithm was successful in comparison with SAC in terms of both faster converging and results in a better reward, even-though the model is more complex this time.

[Single Agent Training][00]

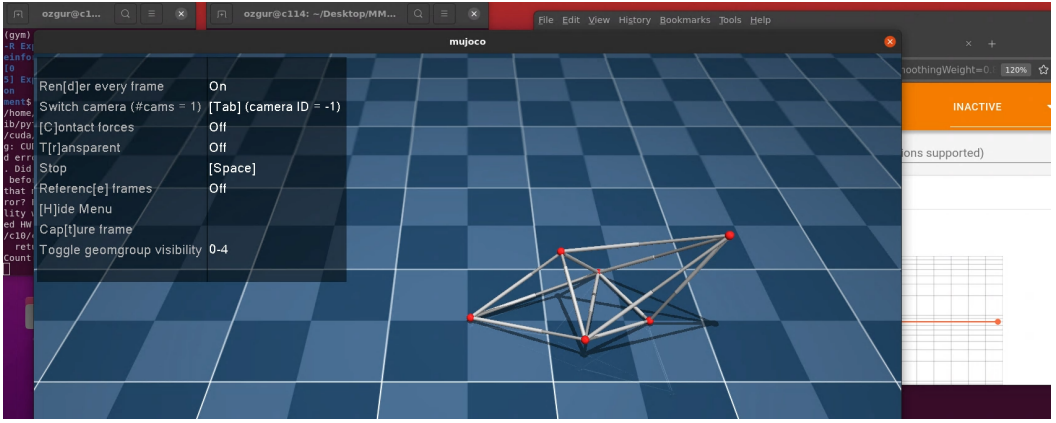
[2-TET]



[Single Agent Training][01]

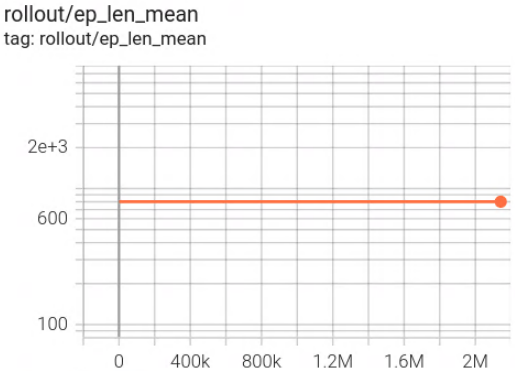
[3-TET]

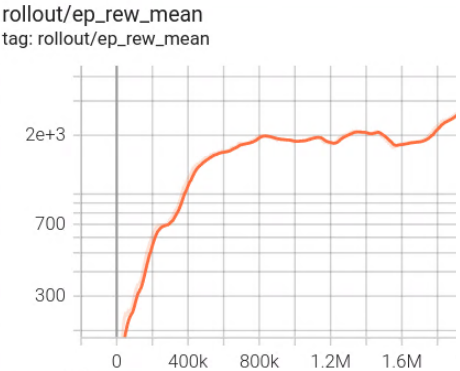
3-Tetrahedron is more challenging due to the increased number of members and more complicated geometry. The agent is not as successful as 2-tetrahedron. A longer training is necessary.



[Single Agent Training][01]

[3-TET]

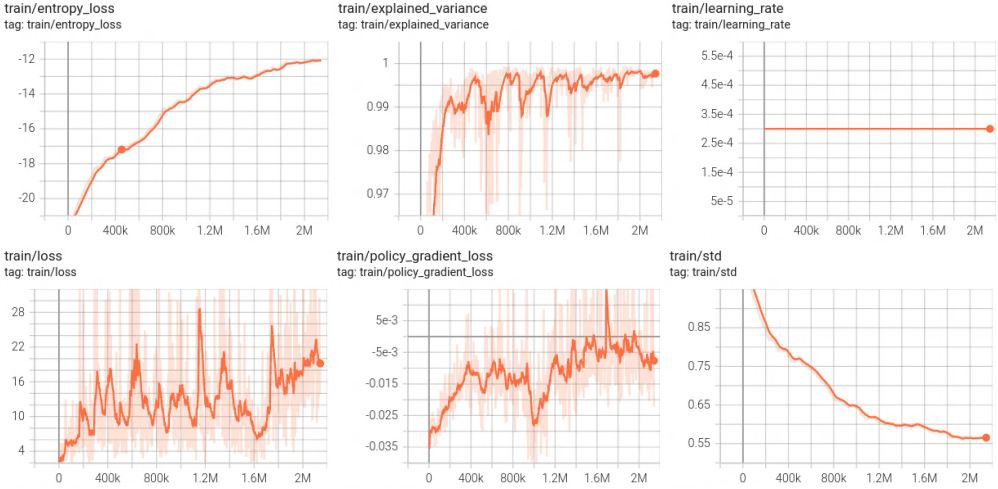




In about 1.6 M total time-steps, the reward is about half of the 2-tetrahedron case. However, results indicate the learning continues and there is a room for improvement.

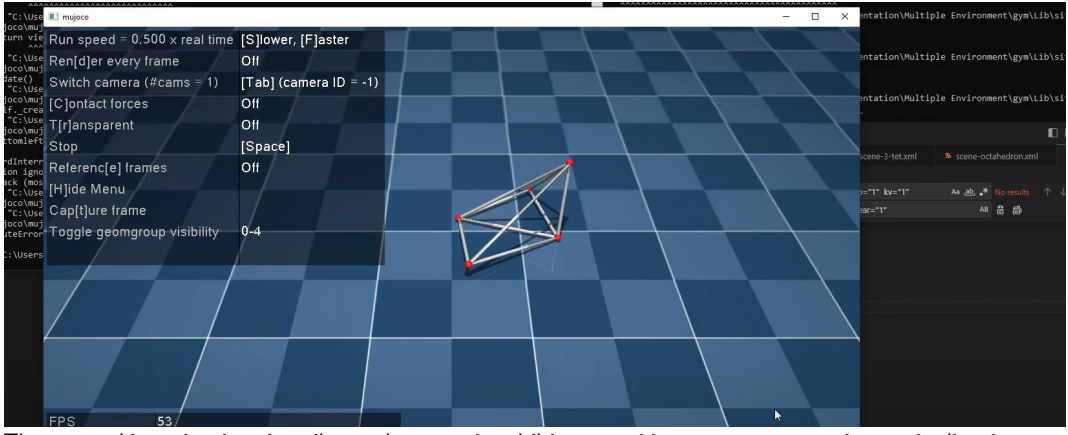
[Single Agent Training][01]

[3-TET]



[Multi Agent Training][02]

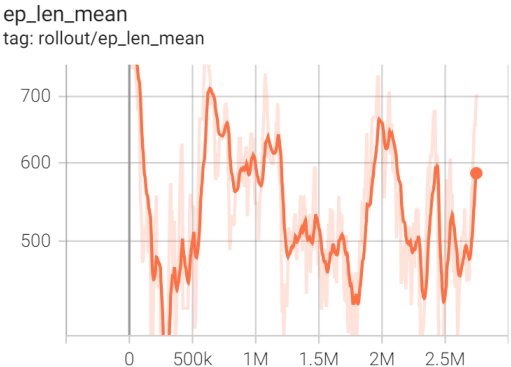
[1-TET | 2-TET | 3-TET | OCTA]

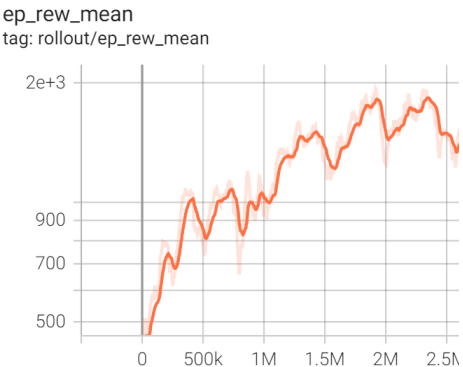


The reward iteration is primarily random, yet it exhibits a positive mean, suggesting an inclination towards the intended direction.

[Multi Agent Training][02]

[1-TET | 2-TET | 3-TET | OCTA]

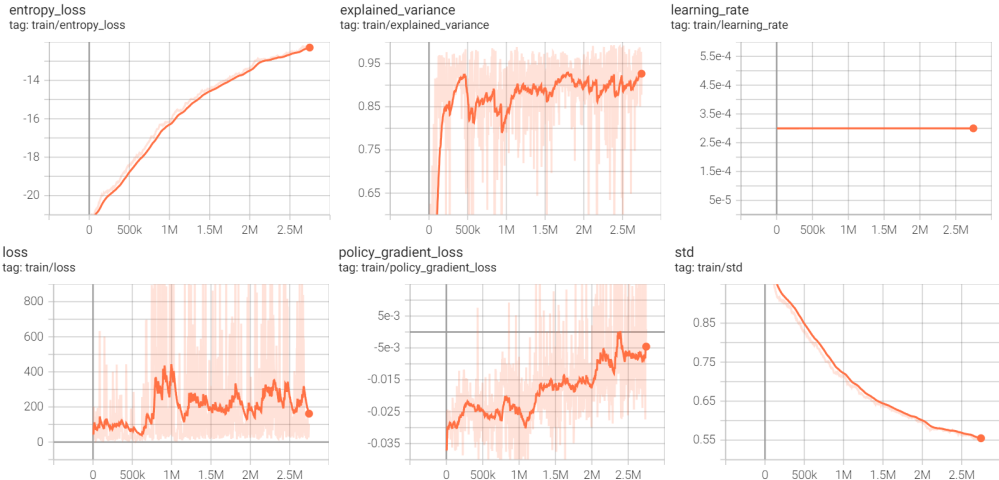




The reward iteration is primarily random, yet it exhibits a positive mean, suggesting an inclination towards the intended direction.

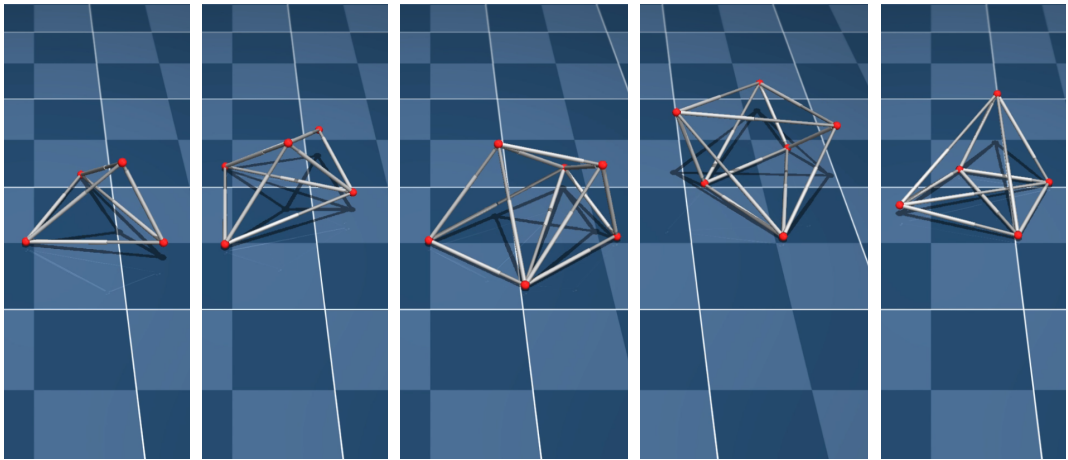
[Multi Agent Training][02]

[1-TET | 2-TET | 3-TET | OCTA]



[Multi Agent Training][02]

[1-TET | 2-TET | 3-TET | OCTA]



Outline

Conclusions & Future Works

- The possible robot topology space is quite vast and in terms of varying the topology, a resilient unified controller is critical.
- The individual training with PPO algorithm performed better than SAC in terms of robots with higher dimensions.
- For more in-depth learning experiments, additional robot models should be created, and algorithm modifications should be explored thoroughly.
- The vision is failure resilient controller for a topology changing size altering truss robot.
- Graph neural networks and topology architectures can be investigated.

Bibliography I

[1] H. Lipson and J. B. Pollack, “Automatic design and manufacture of robotic lifeforms,” *Nature*, vol. 406, p. 974–978, Aug. 2000.

[2] N. Usevitch, Z. Hammond, S. Follmer, and M. Schwager, “Linear actuator robots: Differential kinematics, controllability, and algorithms for locomotion and shape morphing,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5361–5367, 2017.

[3] J. Whitman, M. Travers, and H. Choset, “Learning modular robot control policies,” *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 4095–4113, 2023.

ODTÜ
METU

Thank you for listening