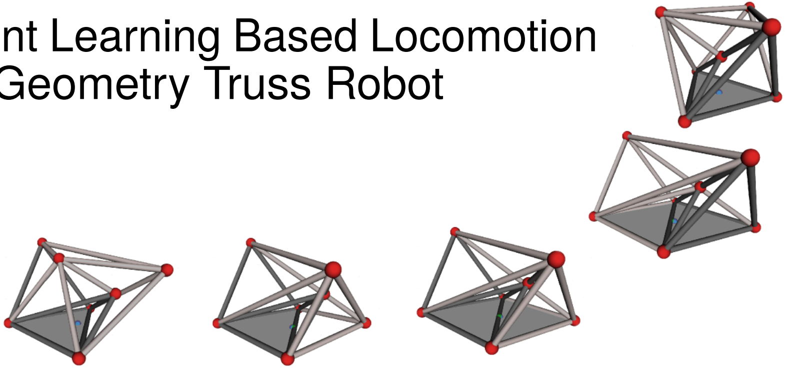
Reinforcement Learning Based Locomotion for Variable Geometry Truss Robot

Ozgur Gulsuna

25 Apr 2024



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

Outline

Outline

1. Research Question
2. Simulation Environment
3. Variable Truss Robot Model
4. Reinforcement Learning Problem
5. Reinforcement Learning Agent
6. Reinforcement Learning Training
7. Training Attempts
8. Conclusions & Future Work

Locomotion Strategy for Truss Robots

[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. *Multi-Agent Policy RL:* Each member is an individual agent.

[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]

[3]**[Distributed Coach-Based RL Controller for Snake Robot Locomotion]** [2022]

Snake robot control with RL is underexplored due to high freedom redundancy. Existing methods use asynchronous joint state representations. A new solution introduces a distributed coach-based learning approach to address these challenges.[4]

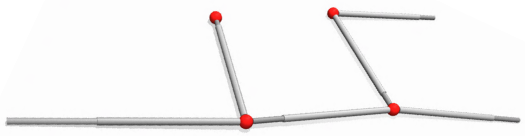
Outline

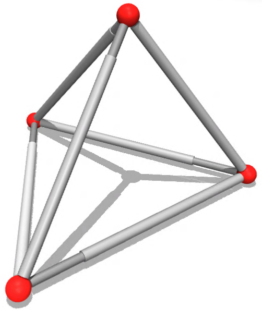
Simulation Environment and Truss Robot Model

[MuJoCo]

The initial selection of the simulation environment was the NVIDIA® Isaac Gym due to the parallel training capability, However the closed kinematic chains are not supported hence the environment is changed to MuJoCo.

The main limitation here is the simulation of closed kinematic chains are achieved through the equality constraints. The model is first described in an open tree form, then with the constraints chain is constructed.





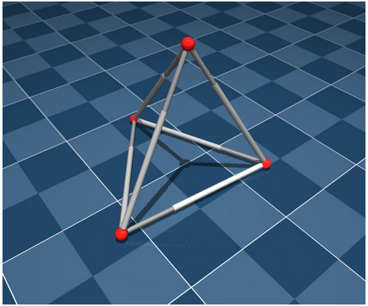
Outline

Variable Geometry Truss Robot Description

[MuJoCo]

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.

- Simple tetrahedron robot, options are to be increased.
- 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.
  - This is because a topology changing truss robot can be controlled with the same algorithm and it can work efficiently with other possible topologies.



Notes: Due to the structured nature of the robot model in MuJoCo, it is hard to initiate the robot with different configurations such as member lengths.

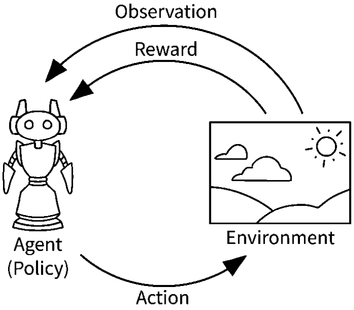
Outline

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 environment 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
0	position	0	2	[Member_1]	slide	lenght (m)
1	position	0	2	[Member_2]	slide	lenght (m)
2	position	0	2	[Member_3]	slide	lenght (m)
3	position	0	2	[Member_4]	slide	lenght (m)
4	position	0	2	[Member_5]	slide	lenght (m)
5	position	0	2	[Member_6]	slide	lenght (m)

Num	Observation	Min	Max	Name	Joint	Unit
0	x-coord of CoM	-inf	inf	coord_x	free	(m)
1	y-coord of CoM	-inf	inf	coord_y	free	(m)
2	z-coord of CoM	-inf	inf	coord_z	free	(m)
3	x-linvel of CoM	-inf	inf	vel_x	free	(m/s)
4	y-linvel of CoM	-inf	inf	vel_y	free	(m/s)
5	z-linvel of CoM	-inf	inf	vel_z	free	(m/s)
6	length of [M_1]	-inf	inf	length_1	slide	(m)
⋮	⋮			⋮		
11	length of [M_6]	-inf	inf	length_6	slide	(m)
12	x-coord of [N_1]	-inf	inf	coord_x	free	(m)
⋮	⋮			⋮		
23	z-coord of [N_4]	-inf	inf	coord_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 is used for training but it can be changed with cleanRL implementation and it is easy to try various alternatives.

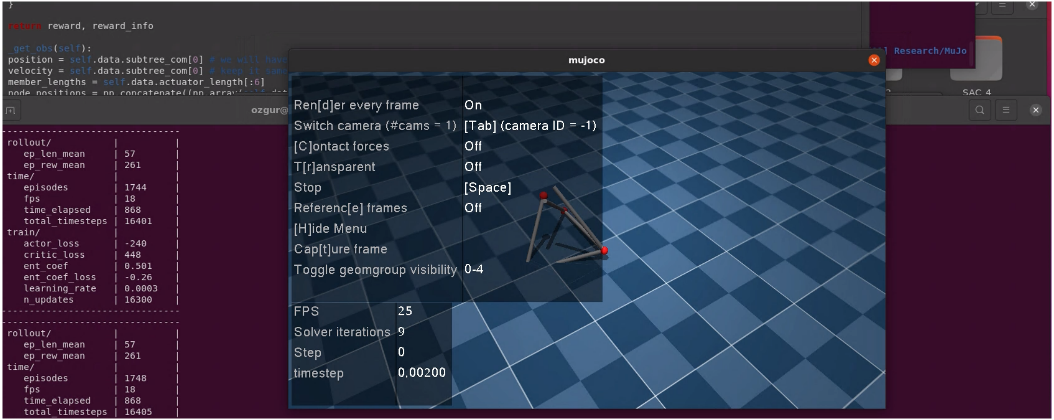
- $forward\_reward\_weight$: Multiplier for rewarding forward movement.
- $healthy\_reward$: Reward earned per time step when the robot is healthy.
- $size\_cost\_weight$: Penalty for increasing the size of the robot.
- $episode\_horizon$: Length of each episode; longer episodes allow for more learning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

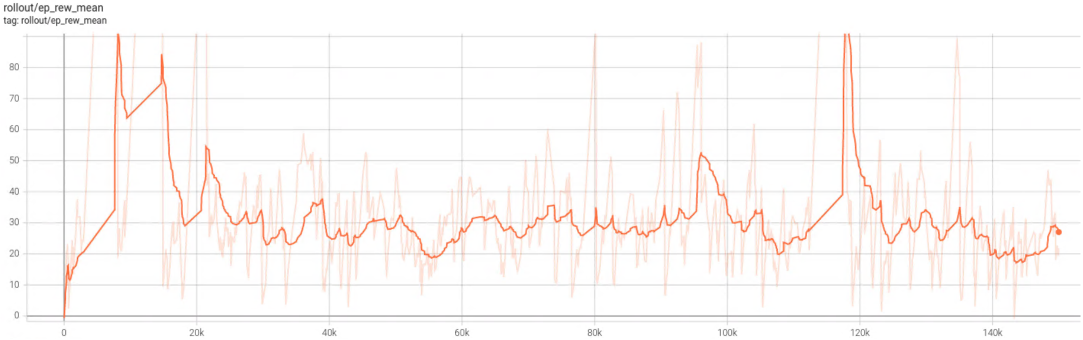
Outline

[Training Attempt][00]

In the 0^{th} experiment, glitches occur initially, causing the robot to jump with random velocity. However, it still manages to extend its members, resulting in increased rewards. This observation is noteworthy.



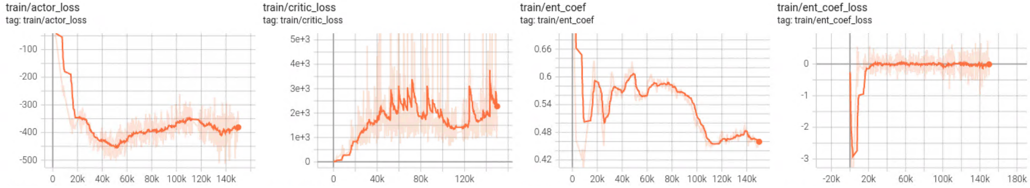
[Training Attempt][00]



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

[Training Attempt][00]

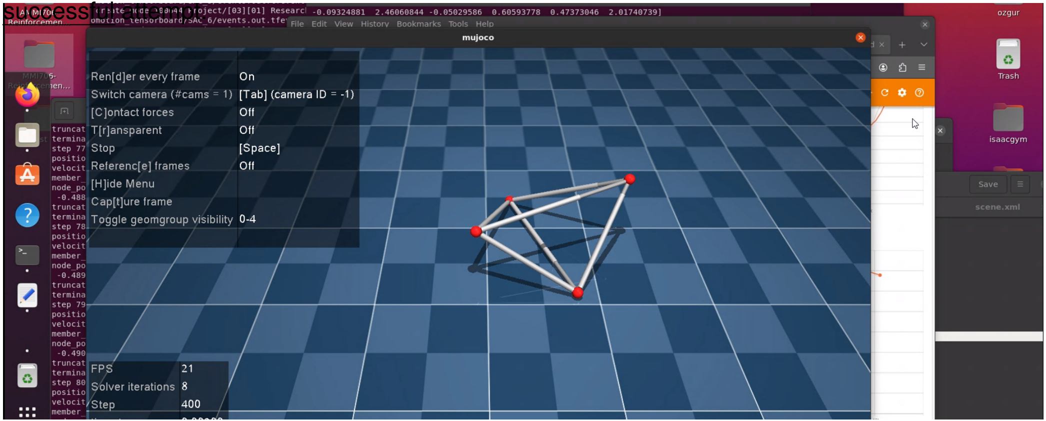
[MuJoCo]



The actor and critic losses exhibit an expected behavior initially, providing another soft indication of a leaning towards learning.

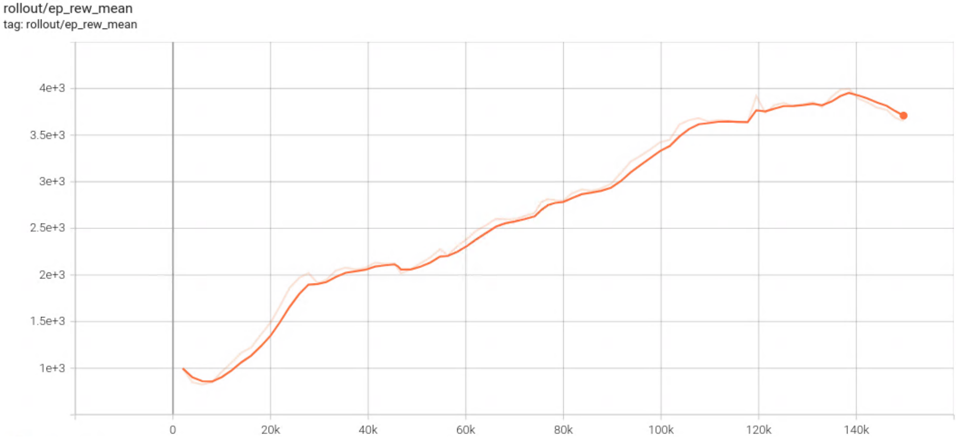
[Training Attempt][01]

The glitch problem has been resolved, and the agent demonstrates trained behavior by placing a foot towards the intended direction and actively locomoting towards it most of the time. This marks a successful attempt



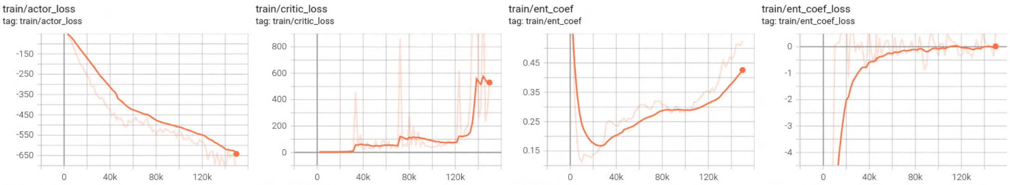
[Training Attempt][01]

[MuJoCo]



[Training Attempt][01]

[MuJoCo]

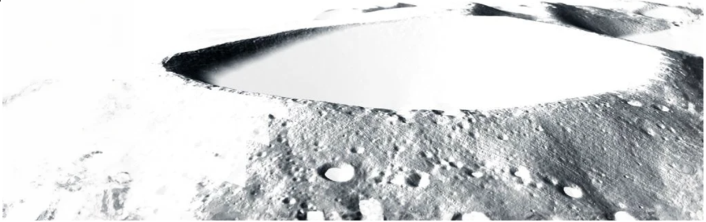


The reward has increased up to three times the initial value, primarily due to the reward for staying healthy, which serves as the baseline reward. This clearly indicates that the soft actor-critic method has effectively increased the reward, resulting in behavior that appears trained. The loss waveforms for actor-critic methods once again demonstrate expected behavior.

Outline

Conclusions & Future Works

- The baseline training has been successful, indicating that the agent is ready for more complex tasks.
- Experimenting with different algorithms such as PPO is anticipated to yield superior results.
- For more in-depth learning experiments, additional robot models should be created, and algorithm modifications should be explored thoroughly.
- Action and observation spaces should be determined to accommodate robots with varying dimensions and topologies.



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.

[4] Y. Jia and S. Ma, “Distributed coach-based reinforcement learning controller for snake robot locomotion,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1231–1238, 2022.

Thank you for listening