MMI706 METU
Reinforcement Learning Phase-II

29/04/2024

1. Idea Summary

Variable Geometry Truss (VGT) robots are modular, deformable robotic truss structures. They consist of nodes and linear actuators connecting those, they offer a means of movement despite their typically rigid nature. They come in various topologies, from simple to complex. For instance an octahedral robot consist of twelve actuators and six passive joint nodes, can move in complex ways. While optimization based approaches are prevalent in the literature, their adoptibİlity to more complex structures remains a concern. Exploring reinforcement learning promises to unlock new avenues in truss robot locomotion.

The research focuses on creating a way for truss robots to move. This involves using reinforcement learning, a type of artificial intelligence, as the main tool. The goal is to coordinate the movement of many parts, much like muscles and joints work together in living creatures. The robot design includes several linear actuators, each able to move in one direction, connected by passive joints. Three strategies are proposed: A model-free reinforcement learning approach for a single robot, a modular approach where different typologies are learned together, and a modular approach where each actuator is an agent and they coordinate with each-other to construct the overall movement.

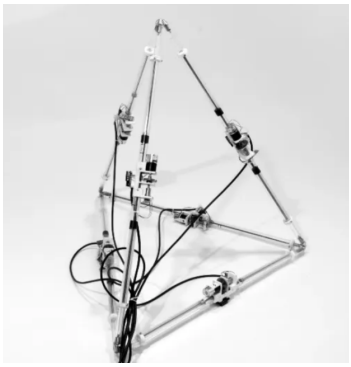

Fig. S1. Physical Tetrahedron Robot (Morphs).

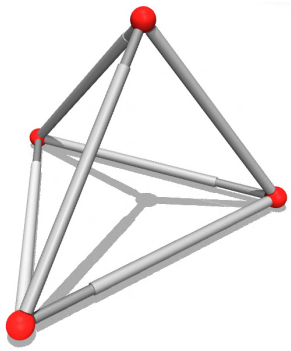

Fig. S2. MuJoCo Tetrahedron Model.

It's essential to view three proposed strategies as iterative milestones rather than hierarchical steps. The initial step, model-free RL control, serves as a foundational stage but requires thorough development and optimization in algorithm selection and parameter tuning. The second step delves deeper into learning algorithms, aims to train for various robots and apply the algorithm to a complete different robot. The third step follows a similar logic, aimed at a similar result with the second step. The project's objective is to extend the first milestone to diverse robot topologies, comparing various algorithms and training on more challenging tasks. The second and third milestones are advanced research domains demanding intricate algorithm designs and these designs can indeed fail. Consequently, they are approached as experimental endeavors, with progress documented throughout development.

There are three main papers that will help in development, first one is **Non-impact Rolling Locomotion of a Variable Geometry Truss**, which is the main analytical approach behind the truss robot modeling. Up to the phase two, this paper is primarily used in implementation. The second (**Learning Modular Robot Control Policies**) and third (**Distributed Coach-Based RL Controller for Snake Robot Locomotion**) papers are about modular learning approach and they will be used in the upcoming trials.

2. Simulation Environment & Robot Model

Initially, the simulation environment chosen was NVIDIA® Isaac Gym because of its parallel training capabilities. However, since it doesn't support closed kinematic chains, the environment was switched to MuJoCo. The primary constraint encountered is the inability to simulate closed kinematic chains directly, which are crucial for the robot's movement. To work around this limitation, the model is initially represented in an open tree form, and then the closed chains are constructed using equality constraints within the simulation. However, these constraints act as force fields, attempting to gather nodes together, resulting in displacement when other forces become too strong.

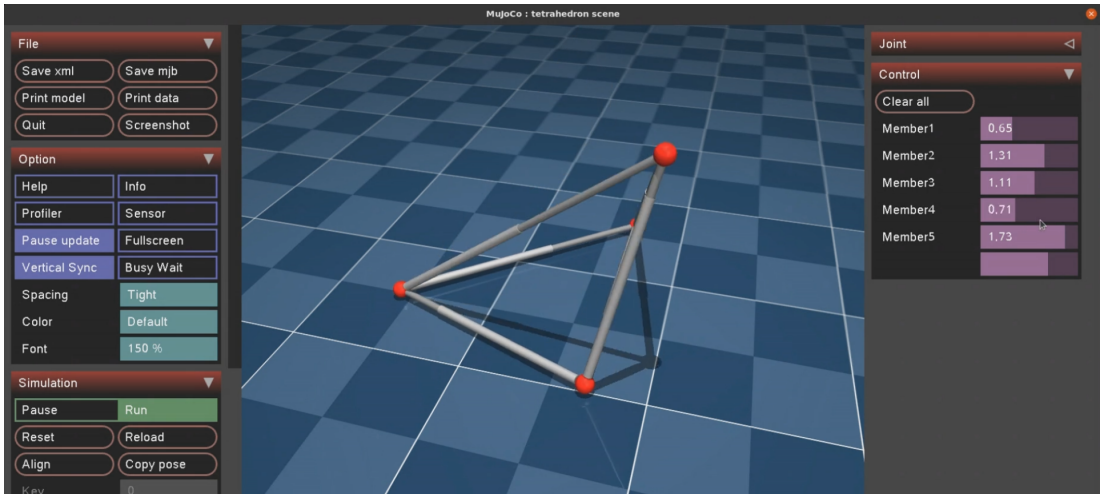

Fig. S3. Tetrahedron robot in MuJoCo environment, member lenght controls are on the right.

The initial model is tetrahedron, featuring 4 nodes and 6 active members represented as linear actuators in MuJoCo. This simple tetrahedron robot serves as a starting point, with plans to expand options in the future. For locomotion, two approaches are considered: position control and force control. Position control relies on slower linear actuators, focusing on shifting the center of mass to facilitate robot movement. Force control, on the other hand, harnesses the masses of the actuators to leverage the robot's inertia for dynamic locomotion. In this project, our emphasis will be on position control, aiming to develop a locomotion strategy centered on manipulating the center of mass rather than dynamic movements. However, in environments with lower gravity, dynamic locomotion may prove more efficient and require less effort.

3. Reinforcement Learning Environment & Training

Since we're at the first stage and have a functional robot model in our chosen simulation environment, we can begin formulating a robot control problem. To test the simulation tools and the robot model, starting with a simple training environment and problem are necessary. The initial problem chosen for the truss robot locomotion is straightforward: move to the right as quickly as possible. The environment features planar terrain with no ground sensing, and the objective is to move in the positive x-axis direction without a specific target point. Rewards are provided for maintaining the robot's health and progress to the right. In subsequent experiments, additional complexities can be introduced, such as incorporating a target destination or a moving target point, and aiming for specific foot nodes. Properly designing the reward mechanism is crucial for advancing locomotion capabilities.

With the problem defined, the next step is constructing the agent. The agent's sensory capabilities are critical, offering various options. Additionally, the agent have the ability to modify its member lengths.

A. Action Space

The action space is a Box(0,2,(6,), float32). An action is a vector of 6 elements where each element is the length of the member. The length of the member is between 0 and 2 meters.

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)

Fig. S4. Action space of a single agent

B. Observation Space

Observations capture the positional values of the center of mass and the respective time derivative, as known as, velocity of the center of mass. Additionally, the measured member lengths and node positions are captured. The observation space is a Box(-inf, inf, (24,), float64). We might need to alter the observation space to have a better learning performance. Although the reward does not seem to be saturated.

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)

Fig. S5. Observation space of a single agent

Training requires the use of specific parameters such as rewards, costs, episode lengths, simulation time steps, and etc., which have a significant impact on the algorithm's performance. Therefore, tuning is generally necessary. The algorithms for the training are various. The soft actor-critic (SAC) algorithm is used for training initially but it will be changed to try various alternatives such as proximal policy optimization (PPO) .

- **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.

Limitation: 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.

4. Outcomes and Results

A. [Training][01]

Initial training, which comprised 150 thousand episodes, took approximately 45 minutes on an average CPU. The episode horizon was set to 2500 time steps with each step is 0.002 seconds, during which the robot was trained to move towards the right. The training proved successful, yielding a locomotion strategy wherein the robot gradually places its feet in the direction of the reward. What is interesting is that, while it pushes a node to right after a certain point two of the other actuators also take action which effectively shifts the center of mass to the right without causing a singularity trap to the robot.

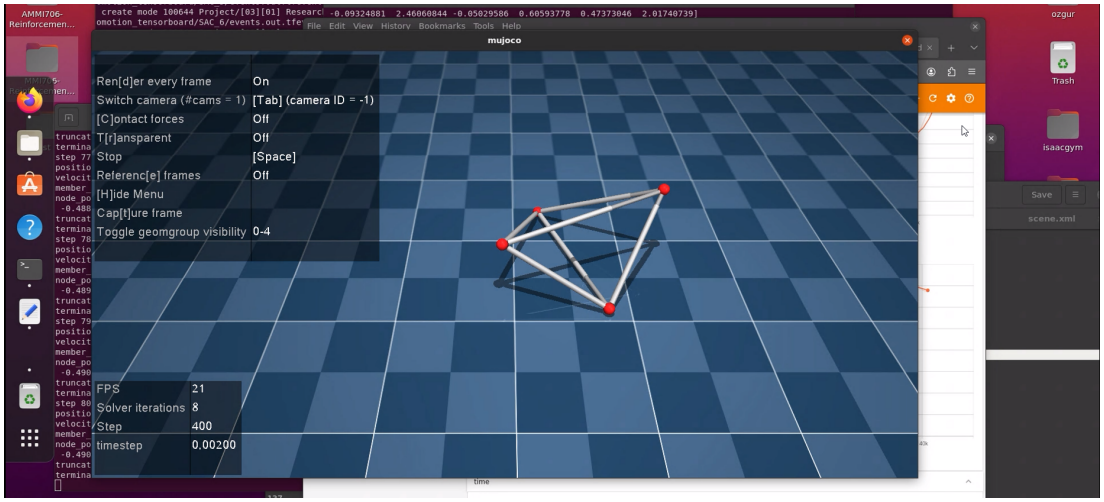

Fig. S6. Tetrahedron locomotion to the right.

The mean plot of the reward illustrates the improvement over time. Initially, the reward starts from a positive value, the healthy reward total. For a standard episode, the healthy reward reaches 1000 points. After 150 thousand episodes, the mean reward triples that of the healthy reward, indicating an improvement in performance. Here, since the episode length quite short the robot tries to learn a faster locomotion. A longer episode length also mean that there is a lot more time and different situations that robot will come into, and each of those situations it will learn what action to take.

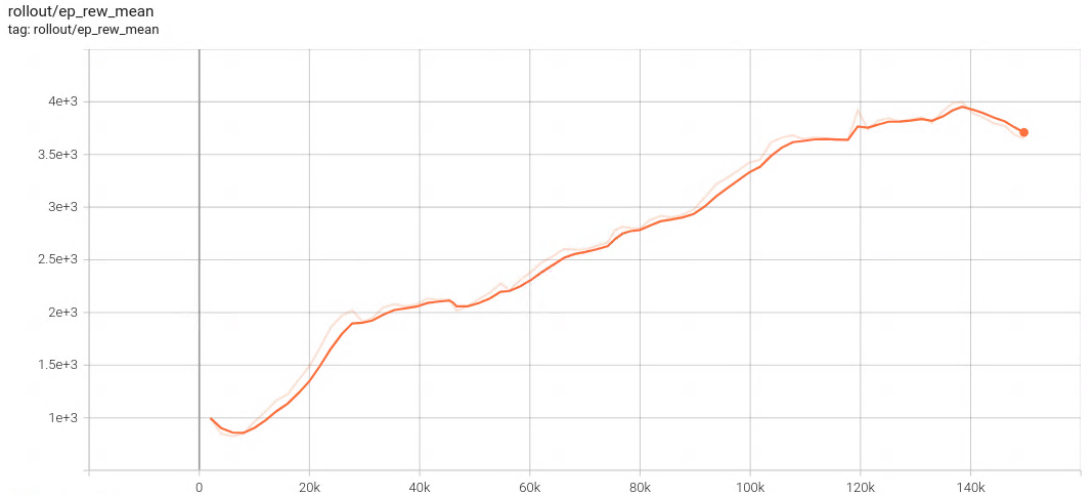

Fig. S7. Mean reward versus episode count

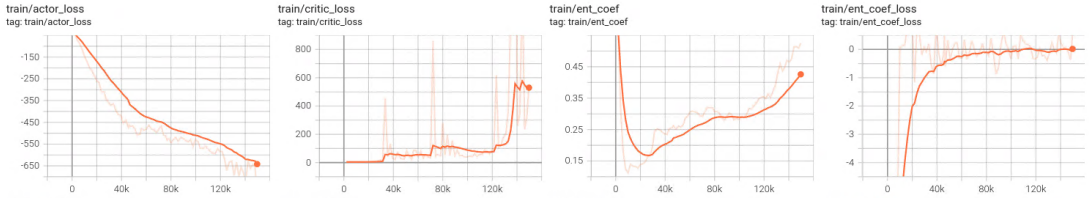

Fig. S8. Various actor critic dependent results of the training

5. Completion and Future Plans

As of now, the completion status stands at approximately 40% for all objectives. When focusing on the primary objective, which aligns with the course's aim, it's about 75% complete. However, the secondary objective of achieving locomotion towards a goal position, as outlined in phase two, is currently missing from the progress.

The final phase involves several critical tasks. Firstly, we require at least four additional robot topologies. While manually creating these topologies is challenging, writing a script to generate robot models in XML format is an option, albeit time-consuming. Alternatively, we could create simplified robot models and let the simulation compute the angles and node positions. However, this approach would require simulating the initial few seconds for every simulation, which could slow down the training process.

Once different robot topologies are created, we need to design observations and actions that can be uniformly applied to each robot. In the later stages, a classic algorithm such as PPO or SAC will be modified based on key ideas from the second paper. Training will then be conducted for multiple robots after applying the modified policy to a robot that hasn't been trained before. This iterative approach aims to generalize the learning across various robot configurations and enhance overall performance.

Ozgur Gulsuna
2307668