**Development Log**

***2023.8.27 – By Qiyang***

Reward:

1. Find the reward for sliding and it can also be used in real-time for multi-shapes without any training and model.
2. Complete mathematics verification of the reward

***2023.8.28 – By Qiyang***

Minor bugs:

General:

1. Action is not completed before changing friction state, which is misleading for the agent, so always use the check\_action\_complete function before friction changing
2. Add a stuck indicator to the last action before each friction change, together with a torque self-adjust that adjusts its’ value to ensure the last action is completed before moving on.
3. Change total time steps from 50 to 100, giving the agent more time to explore to avoid having it doesn’t have time to complete all the moves needed

* Back to 50, 100 seems bit too long

1. Add a both-high-friction state before the friction changes, like what we did for the final project so that friction doesn’t change directly from -1 to 1, which is relatively unstable

Reward Engineering:

1. Remove penalty reward, it limits the exploration, **consider adding it back later**
2. Replace np.any() with array and single number classification together with a for loop to loop through the array.
3. Increase success region from 0.005 to 0.02, to make it easier to success
4. Split the success region into two parts, so the reward is higher when the agent gets closer to the middle, **consider making this region continuous instead of discrete**
5. Success now contains success reward and out-of-range=-10 penalty reward, they are merged

Algorithm:

1. Add compute\_terminated. The existing environment that the algorithm considers doesn’t have terminated, only truncated, so the episode only finishes when the time limit is reached. Now, the environment stops when the object exceeds the limit we defined.
2. Modify the sampling process for HER so it adapts to episodes with different lengths.

Other Changes:

Other:

1. Using GPUs should increase the training speed, so I’m trying to run the training with online servers, such as Colab. But ran into a problem: Colab only accepts Jupyter Notebook format, **how to convert the existing files?**
2. Add a pip install feature to the project, now we can use “pip install gymnasium-variable-friction” to download the modified gymnasium robotics library into Colab.

Next Step:

1. Learn how to implement distributed reinforcement learning.
2. How to use online GPU servo and CUDA to speed up the training process.
3. Compare SAC and PPO
4. Tune parameters
5. How to design the reward for rotation
6. Modify reward to stimulate exploration.
7. Understand 分阶段训练, 反向训练
8. Try more exploration methods.

Training:

1. VariableFriction-v0\_after15epoch\_niceProgress.pth:
2. success =\* 2 right after \_is\_success\_radii
3. self.r\_threshold = 0.02
4. radius < 0.1, \*2 again
5. VariableFriction-v0\_after46epoch\_smallerSuccessRegion.pth:
6. success =\* 2 is removed
7. self.r\_threshold = 0.015
8. radius < 0.075, \*3 again

***2023.8.29 – By Qiyang***

Training:

1. New:
2. Remove success reward.
3. Self.r\_threshold = 0.015
4. Good performance with a steady increase in success rate
5. Shift to online GPU server: Colab
6. Change device = ‘cpu’ to device = ‘cuda’

Evaluation:

1. Wrong success evaluation, forgot to change is\_success function with \_is\_success\_radi function in step function to get the success info

Colab Training:

1. Complete env setup, need to work out cuda and multi-worker training