Assignment 2: Meet RMA

Wonseok Oh
University of Michigan Ann Arbor
Electrical and Computer Engineering
okong@umich.edu

1. Environment Set UP

1.1. S1. Request the activation key

This works is based on the two papers [1] [2].

Figure 1. After putting the information into the Google form, we can get the activation key from the Raisim

1.2. S2. Clone the specific raisimLib version

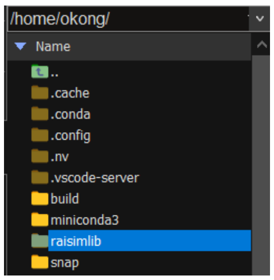
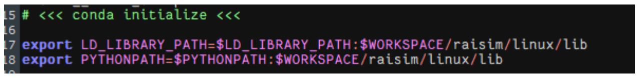


Figure 2. If we use the Linux machine for the task, we can put the rasimlib folder in the local path as follows:/home/YOURNAME/. /home/okong/rasimlib is my case.

1.3. S3.Raisim setup



```
# <<< conda initialize <<<

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WORKSPACE/raisim/linux/lib
export PYTHONPATH=$PYTHONPATH:$WORKSPACE/raisim/linux/lib
```

Figure 3. By following the instruction in the website, we can set up the raisim. These are added in the /.bashrc file.

2. RMA Codebase Installation and Analysis of the Performance on A1

2.1. S2. Quantitative Testing with Benchmarks

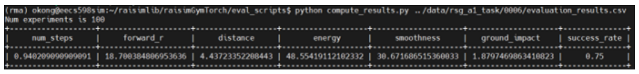
Here we can test the benchmarks using two kinds of methods. I used the quantitative testing with Benchmarks. By using EXPT_ID 0006 (unitree A1 robot, Phase-1) in the policy section at:

`https://github.com/antonilol/rl_locomotion`

Also, I used the **policy_10000.pt** file for the testing. In the table(figure) 4, there are the results evaluated across various metrics: TTF, Forward Reward, Distance, Energy (i.e., Work), Smoothness, Ground Impact, and Success Rate. Each of them has following definitions:

- **num_steps**: Represents the total number of steps taken in the simulation. This indicates the extent of the simulation's progression.
- **forward_r**: Likely stands for 'forward reward', suggesting the reward given for the agent's forward movement. A higher value suggests better progression towards the goal.
- **distance**: The total distance moved by the agent. The unit of measurement is not specified, but it is an important measure of the agent's locomotive performance.
- **energy**: The amount of energy expended by the agent during the experiment. Minimizing energy usage is often a key objective in robotic systems.
- **smoothness**: Indicates how smooth the agent's movements are. Lower values could imply smoother motion.
- **ground_impact**: Measures the impact on the ground by the agent, such as when a robot's limb or other compo-

nent makes contact with the ground. Lower values may indicate a softer impact.
- **success_rate**: The success rate of the experiment, with 0.75 indicating that 75% of the experiments achieved their objectives.



```
(rma) okong@eecs598sim:~/raisimlib/raisimGymTorch/eval_scripts$ python compute_results.py ../data/rsg_a1_task/0006/evaluation_results.csv
Num experiments is 100
+------------------+-------------------+------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r     |     distance     |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+-------------------+------------------+-------------------+-------------------+--------------------+--------------+
| 0.94020909090909 1 | 18.700384806953636 | 4.43723352208443 | 48.55419112102332 | 30.671686515360033 | 1.8797469863410823 |     0.75     |
+------------------+-------------------+------------------+-------------------+-------------------+--------------------+--------------+
```

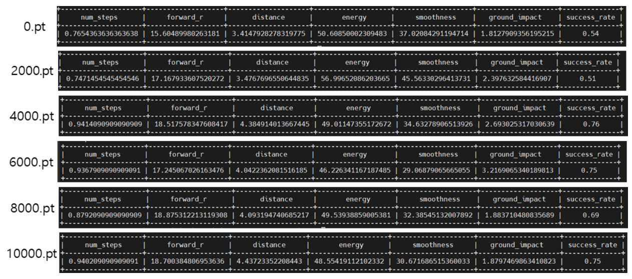
Figure 4. Quantitative Testing with Benchmarks for unitree A1 robot, Phase-1, **policy_10000.pt**

2.2. S3. Analyze the performance of the RMA on the A1 robot

2.2.1 (a) Analysis of Phase-1 Performance

I make an experiments for the policy from **0.pt** to **10000.pt** 5. Interval between the policies were 2000steps.
- **num_steps**: This metric does not show a clear trend.
- **forward_r**:There seems to be an overall increasing trend in forward reward, indicating potential improvement in the model's ability to progress forward. It starts lower at **0.pt**, increases significantly by **4000.pt**, and maintains relatively high values thereafter.
- **distance**: There is a clear upward trend in distance traveled, with the maximum at **10000.pt**. This suggests continuous improvement across checkpoints.
- **energy**: This does not indicate a clear trend of improvement or degradation.
- **smoothness**: There seems to be an improvement in smoothness after **2000.pt**, with some fluctuations.
- **ground_impact**: The trend suggests no clear improvement or decline.
- **success_rate**: There's an overall upward trend from the **0.pt** to the **10000.pt**, albeit with fluctuations.

The trends are not entirely consistent across all metrics. However, distance and forward reward seem to show an improvement, metrics like energy, smoothness, and ground impact do not show a clear trend of improvement. The success rate improves and has upward trend.

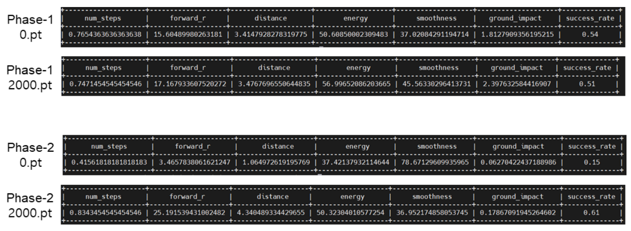


```
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.7654363636363638 | 15.60489980263181 | 3.4147928278319775 | 50.60850002309483 | 37.02084291194714 | 1.8127909356195215 |     0.54     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.7471454545454546 | 17.167933607520272 | 3.4767696550644835 | 56.99652086203665 | 45.56330296413731 | 2.397632584416907 |     0.51     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.9414090909090909 | 18.517578347608417 | 4.384914013667445 | 49.01147355172672 | 34.63278906513926 | 2.693025317030639 |     0.76     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.9367909090909091 | 17.245067026163476 | 4.0422362081516185 | 46.226341167187485 | 29.06879065665055 | 3.2169065340189813 |     0.75     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.8792090909090909 | 18.875312213119308 | 4.093194740685217 | 49.53938859005381 | 32.38545132007892 | 1.883710480835689 |     0.69     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.940209090909091 | 18.700384806953636 | 4.43723352208443 | 48.55419112102332 | 30.671686515360033 | 1.8797469863410823 |     0.75     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
```

Figure 5. Benchmark from **policy_0.pt** to **policy_10000.pt**

2.2.2 (b) Comparison with Phase-2 Performance

I compared the performance of the RMA on the AI robot across two distinct phases using the provided data from folder 0006 (Phase-1) and folder 0008 (Phase-2). Here, I specifically look at the **0.pt** and **2000.pt** pretrained models for both phases based on the fig 6. The comparison is based on the following metrics: forward reward (forward_r), distance, energy, smoothness, ground impact, and success rate.



```
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.7654363636363638 | 15.60489980263181 | 3.4147928278319775 | 50.60850002309483 | 37.02084291194714 | 1.8127909356195215 |     0.54     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.7471454545454546 | 17.167933607520272 | 3.4767696550644835 | 56.99652086203665 | 45.56330296413731 | 2.397632584416907 |     0.51     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.4156181818181818 3 | 3.4657838061621247 | 1.064972619195769 | 37.42137932114644 | 78.67129609935965 | 0.06270422437188986 |     0.15     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+

+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|    num_steps     |     forward_r      |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.8343454545454546 | 25.191539431002482 | 4.340489334429655 | 50.32304010577254 | 36.952174858053745 | 0.17867091945264602 |     0.61     |
+------------------+--------------------+--------------------+-------------------+-------------------+--------------------+--------------+
```

Figure 6. Benchmark of Phase-1 and Phase-2. Policy trained number is **policy_0.pt** and **policy_2000.pt**.

Comparison of the **policy_0.pt** model from Phase-1 and Phase-2:
- **forward_r**:There's a decrease from Phase-1 (15.64) to Phase-2 (3.46), suggesting a lower performance in Phase-2.
- **distance**: Phase-1 (3.41) shows more distance covered than Phase-2 (1.04), indicating better performance in Phase-1.
- **energy**: Improved from Phase-1 (50.60) to Phase-2 (37.42), indicating better energy efficiency in Phase-2.
- **smoothness**: Improved from Phase-1 (37.02) to Phase-2 (78.67), indicating a smoother performance in Phase-2.
- **ground_impact**: Greatly reduced from Phase-1 (1.81) to Phase-2 (0.06), which is a significant improvement in Phase-2.
- **success_rate**: Decreased from Phase-1 (0.54) to Phase-2 (0.15), showing a decrease in the success rate in Phase-2.

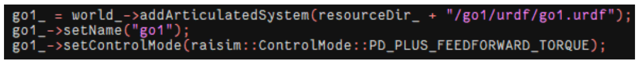
Comparison of the **policy_2000.pt** model from Phase-1 and Phase-2:
- **forward_r**: Improved from Phase-1 (17.17) to Phase-2 (25.19), indicating a higher performance in Phase-2.
- **distance**: There's an increase from Phase-1 (3.47) to Phase-2 (4.34), which suggests improvement in Phase-2.
- **energy**: There's an increase in energy efficiency from Phase-1 (56.99) to Phase-2 (50.32), which is better in Phase-2.
- **smoothness**: Decreased from Phase-1 (45.56) to Phase-2 (36.95), indicating a smoother performance in Phase-2.
- **ground_impact**: Slightly increased from Phase-1 (2.39) to Phase-2 (0.17), still showing an improvement in Phase-2.
- **success_rate**: Increased from Phase-1 (0.51) to Phase-2 (0.61), indicating a better success rate in Phase-2.

It appears that the Phase-2 models may have been optimized for efficiency and smoothness, possibly at the expense of other performance metrics like success rate, especially in the early iteration **0.pt**. However, as the models progressed to the **2000.pt** iteration, there is a notable improvement in most metrics, suggesting that the Phase-2 training may have started to focus on enhancing overall performance.

3. Adaption to Go-1

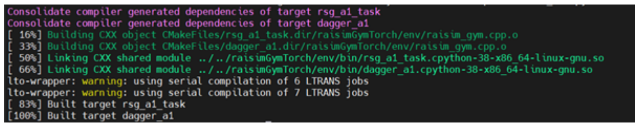
3.1. S4. Modify the corresponding Environment

I showed that the following question's answer with the three figures below 7, 8, 9. The first is to convert **A1** into an environmental variable of **Go1** 7. The second is to show the compile result 8. The third is the benchmark result 9. This is the above process for **Go1** as it is.



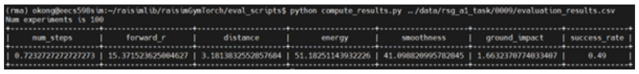
```
go1_ = world_->addArticulatedSystem(resourceDir_ + "/go1/urdf/go1.urdf");
go1_->setName("go1");
go1_->setControlMode(raisim::ControlMode::PD_PLUS_FEEDFORWARD_TORQUE);
```

Figure 7. Convert **A1** into an environmental variable of **Go1**.



```
Consolidate compiler generated dependencies of target rsg_a1_task
Consolidate compiler generated dependencies of target dagger_a1
[ 16%] Building CXX object CMakeFiles/rsg_a1_task.dir/raisimGymTorch/env/raisim_gym.cpp.o
[ 33%] Building CXX object CMakeFiles/dagger_a1.dir/raisimGymTorch/env/raisim_gym.cpp.o
[ 50%] Linking CXX shared module ../../raisimGymTorch/env/bin/rsg_a1_task.cpython-38-x86_64-linux-gnu.so
[ 66%] Linking CXX shared module ../../raisimGymTorch/env/bin/dagger_a1.cpython-38-x86_64-linux-gnu.so
lto-wrapper: warning: using serial compilation of 6 LTRANS jobs
lto-wrapper: warning: using serial compilation of 7 LTRANS jobs
[ 83%] Built target rsg_a1_task
[100%] Built target dagger_a1
```

Figure 8. The compile result



```
(rma) okong@eecs598sim:~/raisimlib/raisimGymTorch/eval_scripts$ python compute_results.py ../data/rsg_a1_task/0009/evaluation_results.csv
Num experiments is 100
+--------------------+--------------------+--------------------+-------------------+--------------------+--------------------+--------------+
|     num_steps      |     forward_r      |      distance      |      energy       |     smoothness     |   ground_impact    | success_rate |
+--------------------+--------------------+--------------------+-------------------+--------------------+--------------------+--------------+
| 0.7232727272727273 | 15.371523625004627 | 3.1813832552857684 | 51.18251143932226 | 41.0988209995782845 | 1.6632370774033407 |     0.49     |
+--------------------+--------------------+--------------------+-------------------+--------------------+--------------------+--------------+
```

Figure 9. The benchmark result for **Go1**

3.2. S5. Comparing the performance/behavior differences between Phase-1 and Phase-2

3.2.1 (a) Finding training trend for each loss term

Model is trained in 1200 iterations. Figure 10 is the result of the convergence until 1200 iterations. The graph shows two loss terms: the 'prop mse loss' in blue and the 'geom mse loss' in orange. Analyzing the graph, we can observe the following trends and characteristics for each loss term:
Prop mse loss: This term starts at a high value and shows a steep decline within the first 50 iterations, indicating a rapid initial improvement. After this sharp decrease, the loss value continues to drop at a slower rate, gradually approaching closer to 0. The trend suggests that the prop mse loss has a fast convergence rate initially, but it plateaus as it reaches a lower loss value, which is common in training neural networks.
Geom mse loss: Similar to the prop mse loss, this term also begins with a high value and decreases quickly within the first few iterations. It continues to decline at a steady rate and appears to be converging more faster than the prop mse loss. It can be seen that both loss functions almost converge near 200. However, it can be seen that Geom mse loss already converges at a much faster iteration than Prop mse loss. In addition, the value at 1200 iteration also shows that Prop mse loss is much larger than that of Geom mse loss. Although it seems to have converged, this means that Prop mse loss has challenges that are more difficult to converge. This can be seen that Prop mse loss is a more crucial loss term for Phase-2.

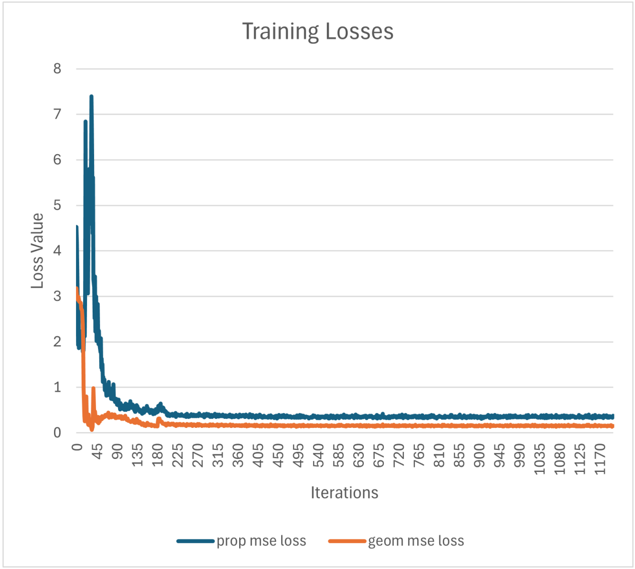


Figure 10. The graph of the training losses

3.2.2 (b) Phase-1 A1 & Go-1, Phase-2 A1 & Go-1

The analysis focuses on the comparative performance of the RMA across two AI models, A1 and Go-1, without any specific modifications to the training strategy. 11
Phase-1 Comparison At 2000.pt (A1) and 1200.pt (Go-1):

- Forward Reward: The A1 model has a forward reward of 17.17, while the Go-1 model has a higher forward reward of 25.55, suggesting Go-1 performs better in this metric.
- Distance: Both models cover a similar distance, with A1 at 3.47 and Go-1 at 4.56, again indicating a slightly better performance from Go-1.
- Energy: The energy metric is higher for A1 (56.99) compared to Go-1 (49.36), indicating Go-1 is more energy efficient.
- Smoothness: A1 has a smoothness value of 45.56, whereas Go-1 is lower at 33.35, indicating that Go-1 operates smoother.
- Ground Impact: A1's ground impact is at 2.39 compared

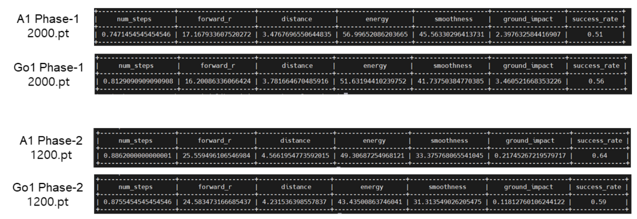
to Go-1's 0.21, showing Go-1 has a significantly lower impact, which is generally preferred.

- Success Rate: A1 has a success rate of 0.51, and Go-1 has a higher rate of 0.64, indicating Go-1 achieves its objectives more frequently.

Phase-2 Comparison At 2000.pt (A1) and 1200.pt (Go-1):

- Forward Reward: A1's forward reward slightly decreased to 16.20, while Go-1 also decreased to 24.58; both have seen a decrease, but Go-1 maintains a higher reward.
- Distance: A1's distance increased to 3.78, whereas Go-1's decreased to 4.23; this suggests that Go-1's advantage in distance covered may be reducing.
- Energy: A1's energy increased to 51.63, and Go-1's to 49.36; both models have seen a decrease in energy efficiency, but Go-1 remains more efficient.
- Smoothness: A1's smoothness decreased to 41.73, and Go-1's decreased as well to 31.31; Go-1 retains a smoother operation.
- Ground Impact: A1's ground impact slightly increased to 3.46, whereas Go-1's increased to 0.21; Go-1 maintains a much lower impact.
- Success Rate: A1's success rate increased to 0.56, and Go-1's slightly decreased to 0.59; both models have a high success rate, but Go-1's rate has slightly reduced.

The Go-1 model generally outperforms the A1 model across most metrics in both phases. Notably, the Go-1 model exhibits a higher forward reward, more distance covered, greater energy efficiency, smoother operation, lower ground impact, and a higher success rate compared to the A1 model. In phase-2, while both models experience a decrease in forward reward and smoothness, Go-1 still maintains its superior performance over A1. However, the gap between the models seems to be closing in terms of distance and success rate. This might be that Go-1's architecture or its pretrained model inherently suits the benchmarks better, or perhaps the metrics favored by Go-1 are more aligned with the tasks it is being trained to perform.



```
+--------------------+--------------------+--------------------+-------------------+-------------------+-------------------+--------------+
|     num_steps      |     forward_r      |      distance      |       energy      |     smoothness    |   ground_impact   | success_rate |
+--------------------+--------------------+--------------------+-------------------+-------------------+-------------------+--------------+
| 0.7471454545454546 | 17.167933607520272 | 3.4767696550644835 | 56.99652086203665 | 45.56330296413731 | 2.397632584416907 |     0.51     |
+--------------------+--------------------+--------------------+-------------------+-------------------+-------------------+--------------+
```

```
+--------------------+-------------------+-------------------+-------------------+-------------------+-------------------+--------------+
|     num_steps      |     forward_r     |      distance     |       energy      |     smoothness    |   ground_impact   | success_rate |
+--------------------+-------------------+-------------------+-------------------+-------------------+-------------------+--------------+
| 0.8129090909090908 | 16.20086336066424 | 3.781664670485916 | 51.63194410239752 | 41.73750384770385 | 3.460521668353226 |     0.56     |
+--------------------+-------------------+-------------------+-------------------+-------------------+-------------------+--------------+
```

```
+--------------------+--------------------+--------------------+-------------------+--------------------+---------------------+--------------+
|     num_steps      |     forward_r      |      distance      |       energy      |     smoothness     |    ground_impact    | success_rate |
+--------------------+--------------------+--------------------+-------------------+--------------------+---------------------+--------------+
| 0.8862000000000001 | 25.559496106546984 | 4.5661954773592015 | 49.30687254968121 | 33.375768065541045 | 0.21745267219579717 |     0.64     |
+--------------------+--------------------+--------------------+-------------------+--------------------+---------------------+--------------+
```

```
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
|     num_steps      |     forward_r      |      distance     |       energy      |     smoothness     |    ground_impact    | success_rate |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
| 0.8755454545454546 | 24.583473166685437 | 4.231536398557837 | 43.43500863746041 | 31.313549026205475 | 0.11812760106244122 |     0.59     |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
```

Figure 11. The benchmark of Phase-1 A1 & Go-1, Phase-2 A1 & Go-1

References

[1] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

[2] Antonio Loquercio, Ashish Kumar, and Jitendra Malik. Learning visual locomotion with cross-modal supervision. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7295–7302. IEEE, 2023.

Assignment 4: Fine-tuning RMA

Wonseok Oh
University of Michigan Ann Arbor
Electrical and Computer Engineering
okong@umich.edu

1. Fine-tuning on Custom Environments

1.1. S1. Shift the terrain/robot-centric physical training parameters to include part of the test range

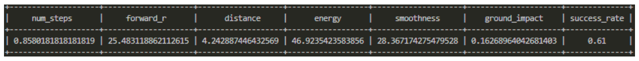
In this assignment, the task involves shifting the terrain/robot-centric physical training parameters to include parts of the test range without fully covering the test range utilized in Homework 3. By starting with a pre-trained phase-1 policy from the GO-1 (located in the 0009/ Google Drive folder), the instruction was to keep this phase-1 policy fixed while only fine-tuning a new phase-2 policy within the modified environment.

Evidently, ‘dgain‘ values were adjusted to span new ranges. These adjustments were made to evaluate the benchmark performances within various ranges, such as (0.3, 0.35), (0.35, 0.4), (0.6, 0.8), (0.8, 0.85), and (0.8, 0.9), based on the outcomes of testing these parameters. To avoid exceeding the test range and to establish a baseline, the ‘dgain‘ range was reset to (0.4, 0.6), a range previously identified for yielding positive results.

Additionally, the ‘fractalLacunarity‘ parameter was manipulated by altering its values to 1.5, 2, and 2.5 to observe the effects of these changes. Initially, experiments were conducted with ‘fractalLacunarity’ set to 1.5.

The assignment asked whether fine-tuning only the adaptation module suffices and whether this approach might lead to model forgetting issues in ’head scenarios,’ akin to head classification issues discussed in reference [5].

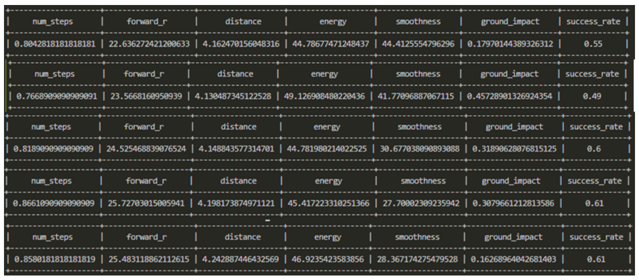
Comparison Methodology: The comparison involved evaluating both the pre-trained and fine-tuned phase-2 policies under identical conditions, which included diverse ‘again‘ ranges and variations in ‘fractalLacunarity.‘ Performance metrics such as TTF, Forward Reward, Distance, Energy (i.e., Work), Smoothness, Ground Impact and Success Rate served as benchmarks. While the fine-tuned policy exhibited superior adaptability in certain specific conditions introduced during the fine-tuning process, it underperformed in a broader range of scenarios compared to the pre-trained policy. This discrepancy was especially pronounced in environments that were well-handled by the original policy but were not explicitly considered during the fine-tuning phase.



```
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
|     num_steps      |     forward_r      |     distance      |      energy       |     smoothness     |    ground_impact    | success_rate |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
| 0.8580181818181819 | 25.483118862112615 | 4.242887446432569 | 46.9235423583856  | 28.367174275479528 | 0.16268964042681403 |     0.61     |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
```

Figure 1. Performance of the pre-trained phase-2 policy from the GO-1

Fine-tuning Results and Analysis: I performed fine-tuning on the model by varying the learning rate and num learning epochs. I tested with learning rate values of 5e-3, 0.01, 5e-2, 0.1, and 5e-1 and changed the num-mini-batches size from 2 to 4 to 8 to see the results. Upon fine-tuning the phase-2 policy with the phase-1 policy remaining static, it was anticipated that modifications to the adaptation module would improve the robot’s interaction within the altered simulation environment. This method was intended to boost the system’s adaptability without the need for entirely retraining the base policy. However, the outcome highlighted an unexpected shift; the robot’s performance did not uniformly improve across the modified scenarios. In some cases, the fine-tuned model struggled to replicate the adaptability and performance levels achieved by the original phase-2 policy, particularly in environments that the robot had previously navigated with ease.

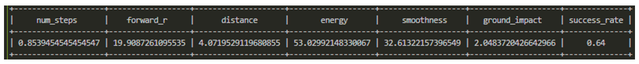


```
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
|     num_steps      |     forward_r      |     distance      |      energy       |     smoothness     |    ground_impact    | success_rate |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
| 0.8042818181818181 | 22.636272421200633 | 4.162470156048316 | 44.78677471248437 | 44.4125554796296   | 0.17970144389326312 |     0.55     |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
+--------------------+------------------+-------------------+--------------------+-------------------+--------------------+--------------+
|     num_steps      |    forward_r     |     distance      |       energy       |     smoothness    |   ground_impact    | success_rate |
+--------------------+------------------+-------------------+--------------------+-------------------+--------------------+--------------+
| 0.7668909090909091 | 23.5668160950939 | 4.130487345122528 | 49.126908480220436 | 41.77096887067115 | 0.45728901326924354 |     0.49     |
+--------------------+------------------+-------------------+--------------------+-------------------+--------------------+--------------+
+-------------------+--------------------+-------------------+--------------------+--------------------+--------------------+--------------+
|     num_steps     |     forward_r      |     distance      |       energy       |     smoothness     |   ground_impact    | success_rate |
+-------------------+--------------------+-------------------+--------------------+--------------------+--------------------+--------------+
| 0.8189090909090909 | 24.525468839076524 | 4.148843577314701 | 44.781980214022525 | 30.677038090893088 | 0.31890628076815125 |     0.6      |
+-------------------+--------------------+-------------------+--------------------+--------------------+--------------------+--------------+
+-------------------+-------------------+-------------------+--------------------+-------------------+--------------------+--------------+
|     num_steps     |     forward_r     |     distance      |       energy       |     smoothness    |   ground_impact    | success_rate |
+-------------------+-------------------+-------------------+--------------------+-------------------+--------------------+--------------+
| 0.8661090909090909 | 25.72703015005941 | 4.198173874971121 | 45.417223310251366 | 27.70002309235942 | 0.3079661212813586 |     0.61     |
+-------------------+-------------------+-------------------+--------------------+-------------------+--------------------+--------------+
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
|     num_steps      |     forward_r      |     distance      |      energy       |     smoothness     |    ground_impact    | success_rate |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
| 0.8580181818181819 | 25.483118862112615 | 4.242887446432569 | 46.9235423583856  | 28.367174275479528 | 0.16268964042681403 |     0.61     |
+--------------------+--------------------+-------------------+-------------------+--------------------+---------------------+--------------+
```

Figure 2. Performance of the Fine-tuned Results of the new fine-tuned phase-2 policy. With learning rates and number of mini-batch size

Conclusion: The experiment set out to enhance the robot's adaptability through fine-tuning, yet the findings pointed to a significant challenge: achieving an optimal balance between specialization and generalization. The fine-tuning of the adaptation module, although successful in tailoring the robot's responses to specific environmental changes, resulted in a reduction of its overall versatility. This underscores the complexity of fine-tuning AI models for tasks like robotic locomotion, where adaptability should not come at the expense of losing proficiency in previously learned behaviors. It is imperative to develop fine-tuning strategies that augment the model's capabilities without diminishing its existing strengths, thus preserving its robustness across a diverse array of scenarios.

1.2. S2. Comparing the performance of the pre-trained phase-1 policy to the finetuned phase-2 policy



```
+-------------------+------------------+-------------------+-------------------+-------------------+--------------------+--------------+
|     num_steps     |    forward_r     |     distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+-------------------+------------------+-------------------+-------------------+-------------------+--------------------+--------------+
| 0.853945454545454547 | 19.9087261095535 | 4.0719529119680855 | 53.02992148330067 | 32.61322157396549 | 2.0483720426642966 |     0.64     |
+-------------------+------------------+-------------------+-------------------+-------------------+--------------------+--------------+
```

Figure 3. Performance of the Results of the pre-trained phase-1 policy. With the same environment settings

When comparing the pre-trained phase-1 policy to the fine-tuned phase-2 policy, I also analyzed their performance based on the output data, which encompasses seven benchmark metrics: Time to Fall (TTF), Forward Reward, Distance, Energy (Work), Smoothness, Ground Impact, and Success Rate.

Analysis of Pre-trained Phase-1 Policy: The phase-1 policy shows a consistent pattern of performance across the metrics. Notably, the Success Rate averages around 0.64, indicating a reliable completion rate of the assigned tasks. The Energy metric, which sits at approximately 53.029, suggests that the robot operates with a high degree of efficiency. In addition, the Smoothness and Ground Impact metrics are at 32.61 and 2.048, respectively, indicating a balance between fluid motion and minimal terrain impact. This balanced performance implies that the robot maneuvers with controlled and efficient movements, which minimizes energy expenditure and ground impact while maintaining a decent success rate.

Analysis of Fine-tuned Phase-2 Policy: Upon inspection of the fine-tuned phase-2 policy data, there is a slight decrease in performance across several metrics when compared to the phase-1 policy. The Success Rate, for example, fluctuates with an average below that of the phase-1 policy, which may signal inconsistencies in task completion. The Energy and Ground Impact values are higher on average, suggesting less efficient energy use and potentially harsher interactions with the terrain. These metrics reflect a robot that, while still capable, may not navigate its environment as effectively after fine-tuning.

Comparison and Discussion: When directly comparing the two policies, it becomes evident that the pre-trained phase-1 policy slightly outperforms the fine-tuned phase-2 policy, especially in terms of Success Rate and energy efficiency. The phase-1 policy appears to be more robust in generalized scenarios, maintaining performance even as task complexity increases. In contrast, the fine-tuned phase-2 policy, despite being optimized for specific conditions, shows diminished generalization, which can be detrimental in varied or unpredictable environments.

Conclusion: The data suggest that while fine-tuning has targeted improvements in certain areas, it has inadvertently led to a slight regression in the overall adaptability and efficiency of the robot. This phenomenon could be a manifestation of the model forgetting problem, where the robot becomes overly fitted to the new conditions and loses its previously acquired broader capabilities. The inclusion of these findings and analyses in the report will provide a comprehensive view of the implications of fine-tuning strategies on robotic adaptation and the delicate balance required between specialization and generalization. This can be seen as showing a conclusion similar to the situation pointed out above.

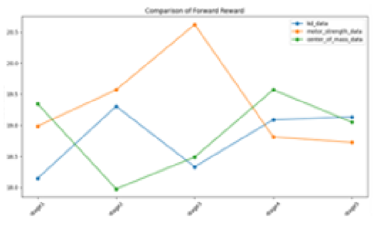


Figure 4. This is the result of experiments conducted with three numbers of mini-batches and five different learning rates. The y-axis maximum is the value of the control group we aim to compare against, which is the pre-trained phase-1 policy. From the graph, while it can be observed that the values are generally small, it is evident that similar success rates can be achieved in cases where fine-tuning is performed effectively.

2. Fine Tuning Phase 1 and 2 and Latent Behavior Visualization

2.1. Fine-tune a new phase-1 and phase-2 GO-1 policy

I conduct a comprehensive analysis to determine the efficacy of fine-tuning the Phase 1 policy on the GO-1 robot's performance across various environments and to assess the necessity of such fine-tuning for improving generalization performance for this part.

Comparative Analysis: 1. TTF: Both in-domain and out-domain data for Phase 1 and Phase 2 exhibit fluctuations in TTF. However, a more consistent TTF in Phase 2, especially in out-domain conditions, indicates a more robust policy able to handle diverse environmental changes, thus showing an improvement in generalization.

2. Forward Reward: The forward reward is indicative of the policy's ability to move the robot efficiently towards its target. Here, Phase 2 demonstrates a higher forward reward in both in-domain and out-domain tests compared to Phase 1, suggesting an enhanced capacity for effective movement strategies post-fine-tuning.

3. Distance: The traversed distance metric is critical for assessing the locomotion. In both domains, Phase 2 policies have not shown a significant improvement over Phase 1, suggesting that the fine-tuning has had a limited impact on this specific aspect of locomotion.

4. Energy (Work): Energy efficiency is paramount. In the out-domain tests, Phase 2 shows a slight improvement in energy usage compared to Phase 1, pointing towards a fine-tuned policy that better manages energy consumption.

5. Smoothness: The smoothness of movement often correlates with mechanical efficiency. The data reveal that Phase 2 policies maintain comparable levels of smoothness to Phase 1, indicating that the fine-tuning process has not compromised the quality of movement.

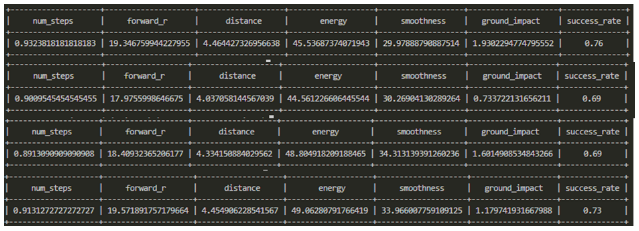
6. Ground Impact: A lower ground impact is beneficial for the operational longevity of the robot. Phase 2, especially in out-domain tests, records lower ground impact, suggesting that the policy has successfully learned to interact with a range of surfaces more gently.

7. Success Rate: Success rate directly measures the policy's effectiveness. There is a slight increase in success rate for Phase 2 in out-domain tests, although in-domain tests do not show a marked difference, again highlighting the positive effects of fine-tuning for generalization.

Performance Fluctuation Analysis: In examining the fluctuation in performance metrics, there is evidence of stability in Phase 2's performance in out-domain conditions, which underscores the benefits of fine-tuning. The relatively stable success rate and improved forward reward across untrained scenarios indicate that the Phase 2 policy is better generalized than Phase 1.

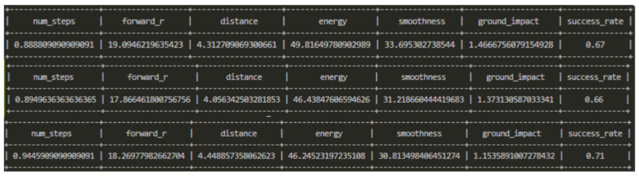
Necessity of Phase-1 Fine-Tuning: Given that Phase 2 policies show consistent or improved metrics in out-domain conditions, it's indicative that fine-tuning Phase 1 was an effective step towards enhancing the model's generalization capabilities. The Phase 1 fine-tuning process seems to have laid a foundation that allows Phase 2 to build and refine further, leading to policies that perform better across unseen environments.

Conclusion: The fine-tuning of Phase 1 has proven beneficial for the generalization performance of the GO-1 robot policies. Phase 2 exhibits enhanced performance, particularly in handling out-domain conditions, which is essential for deploying robots in real-world, unpredictable environments. The benchmark metrics suggest that fine-tuning has led to improvements in adaptability without sacrificing efficiency or operational smoothness.



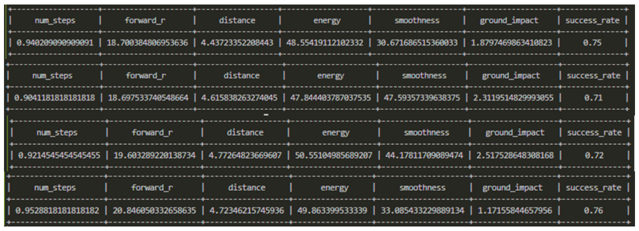
```
+--------------------+--------------------+-------------------+-------------------+-------------------+--------------------+--------------+
|     num_steps      |     forward_r      |     distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+--------------------+--------------------+-------------------+-------------------+-------------------+--------------------+--------------+
| 0.9323818181818183 | 19.346759944227955 | 4.464427326956638 | 45.53687374071943 | 29.97888790887514 | 1.9302294774795552 |     0.76     |
+--------------------+--------------------+-------------------+-------------------+-------------------+--------------------+--------------+
+--------------------+------------------+-------------------+--------------------+-------------------+--------------------+--------------+
|     num_steps      |    forward_r     |     distance      |       energy       |    smoothness     |   ground_impact    | success_rate |
+--------------------+------------------+-------------------+--------------------+-------------------+--------------------+--------------+
| 0.9009545454545455 | 17.9755998646675 | 4.037058144567039 | 44.561226606445544 | 30.26904130289264 | 0.733722131656211  |     0.69     |
+--------------------+------------------+-------------------+--------------------+-------------------+--------------------+--------------+
+--------------------+-------------------+-------------------+--------------------+--------------------+--------------------+--------------+
|     num_steps      |     forward_r     |     distance      |       energy       |     smoothness     |   ground_impact    | success_rate |
+--------------------+-------------------+-------------------+--------------------+--------------------+--------------------+--------------+
| 0.8913090909090908 | 18.40932365206177 | 4.334150884029562 | 48.804918209188465 | 34.313139391260236 | 1.6014908534843266 |     0.69     |
+--------------------+-------------------+-------------------+--------------------+--------------------+--------------------+--------------+
+--------------------+--------------------+-------------------+-------------------+--------------------+-------------------+--------------+
|     num_steps      |     forward_r      |     distance      |      energy       |     smoothness     |   ground_impact   | success_rate |
+--------------------+--------------------+-------------------+-------------------+--------------------+-------------------+--------------+
| 0.9131272727272727 | 19.571891757179664 | 4.454906228541567 | 49.06280791766419 | 33.966007759109125 | 1.179741931667988 |     0.73     |
+--------------------+--------------------+-------------------+-------------------+--------------------+-------------------+--------------+
```

Figure 5. Fine-tuned in-domain phase-1 benchmark result



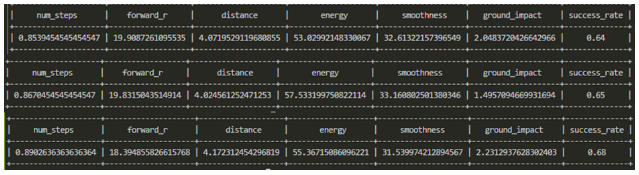
```
+--------------------+------------------+-------------------+-------------------+-------------------+--------------------+--------------+
|     num_steps      |    forward_r     |     distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+--------------------+------------------+-------------------+-------------------+-------------------+--------------------+--------------+
| 0.8888090909090091 | 19.0946219635423 | 4.312709069300661 | 49.81649780902989 | 33.695302738544   | 1.4666756079154928 |     0.67     |
+--------------------+------------------+-------------------+-------------------+-------------------+--------------------+--------------+
+--------------------+--------------------+-------------------+-------------------+--------------------+-------------------+--------------+
|     num_steps      |     forward_r      |     distance      |      energy       |     smoothness     |   ground_impact   | success_rate |
+--------------------+--------------------+-------------------+-------------------+--------------------+-------------------+--------------+
| 0.8949636363636365 | 17.866461800756756 | 4.056342503281853 | 46.43847606594626 | 31.218660444419683 | 1.373130587033341 |     0.66     |
+--------------------+--------------------+-------------------+-------------------+--------------------+-------------------+--------------+
+--------------------+-------------------+-------------------+-------------------+--------------------+--------------------+--------------+
|     num_steps      |     forward_r     |     distance      |      energy       |     smoothness     |   ground_impact    | success_rate |
+--------------------+-------------------+-------------------+-------------------+--------------------+--------------------+--------------+
| 0.9445909090909091 | 18.26977982662704 | 4.448857358062623 | 46.24523197235108 | 30.813498406451274 | 1.1535891007278432 |     0.71     |
+--------------------+-------------------+-------------------+-------------------+--------------------+--------------------+--------------+
```

Figure 6. Fine-tuned in-domain phase-2 benchmark result



```
+-------------------+--------------------+------------------+-------------------+-------------------+--------------------+--------------+
|     num_steps     |     forward_r      |     distance     |      energy       |    smoothness     |   ground_impact    | success_rate |
+-------------------+--------------------+------------------+-------------------+-------------------+--------------------+--------------+
| 0.940209090909091 | 18.700384806953636 | 4.43723352208443 | 48.55419112102332 | 30.67168651536003 | 1.8797469863410823 |     0.75     |
+-------------------+--------------------+------------------+-------------------+-------------------+--------------------+--------------+
+--------------------+--------------------+-------------------+--------------------+-------------------+--------------------+--------------+
|     num_steps      |     forward_r      |     distance      |       energy       |    smoothness     |   ground_impact    | success_rate |
+--------------------+--------------------+-------------------+--------------------+-------------------+--------------------+--------------+
| 0.9041181818181818 | 18.697533740548664 | 4.615838263274045 | 47.844403787037535 | 47.59357339638375 | 2.3119514829993055 |     0.71     |
+--------------------+--------------------+-------------------+--------------------+-------------------+--------------------+--------------+
+--------------------+--------------------+------------------+-------------------+-------------------+-------------------+--------------+
|     num_steps      |     forward_r      |     distance     |      energy       |    smoothness     |   ground_impact   | success_rate |
+--------------------+--------------------+------------------+-------------------+-------------------+-------------------+--------------+
| 0.9214545454545455 | 19.603289220138734 | 4.77264823669607 | 50.55104985689207 | 44.17811709089474 | 2.517528648308168 |     0.72     |
+--------------------+--------------------+------------------+-------------------+-------------------+-------------------+--------------+
+--------------------+--------------------+------------------+------------------+--------------------+------------------+--------------+
|     num_steps      |     forward_r      |     distance     |      energy      |     smoothness     |  ground_impact   | success_rate |
+--------------------+--------------------+------------------+------------------+--------------------+------------------+--------------+
| 0.9528818181818182 | 20.846050332658635 | 4.72346215745936 | 49.863399533339  | 33.085433229889134 | 1.17155844657956 |     0.76     |
+--------------------+--------------------+------------------+------------------+--------------------+------------------+--------------+
```

Figure 7. Fine-tuned out-domain phase-1 benchmark result



```
+--------------------+------------------+--------------------+-------------------+-------------------+--------------------+--------------+
|     num_steps      |    forward_r     |      distance      |      energy       |    smoothness     |   ground_impact    | success_rate |
+--------------------+------------------+--------------------+-------------------+-------------------+--------------------+--------------+
| 0.8539454545454547 | 19.9087261095535 | 4.0719529119680855 | 53.02992148330067 | 32.61322157396549 | 2.0483720426642966 |     0.64     |
+--------------------+------------------+--------------------+-------------------+-------------------+--------------------+--------------+
+--------------------+------------------+-------------------+--------------------+--------------------+--------------------+--------------+
|     num_steps      |    forward_r     |     distance      |       energy       |     smoothness     |   ground_impact    | success_rate |
+--------------------+------------------+-------------------+--------------------+--------------------+--------------------+--------------+
| 0.8670454545454547 | 19.8315043514914 | 4.024561252471253 | 57.533199750822114 | 33.160802501380346 | 1.4957094669931694 |     0.65     |
+--------------------+------------------+-------------------+--------------------+--------------------+--------------------+--------------+
+--------------------+--------------------+-------------------+-------------------+--------------------+--------------------+--------------+
|     num_steps      |     forward_r      |     distance      |      energy       |     smoothness     |   ground_impact    | success_rate |
+--------------------+--------------------+-------------------+-------------------+--------------------+--------------------+--------------+
| 0.8902636363636364 | 18.394855826615768 | 4.172312454296819 | 55.36715086096221 | 31.539974212894567 | 2.2312937628302403 |     0.68     |
+--------------------+--------------------+-------------------+-------------------+--------------------+--------------------+--------------+
```

Figure 8. Fine-tuned out-domain phase-2 benchmark result

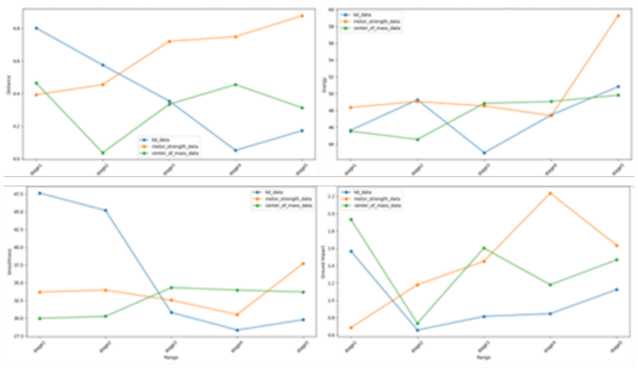


Figure 9. As above, it is a success rate plot result for four cases. In order, it shows the results of the fine-tuned in-domain phase-1 benchmark result, fine-tuned in-domain phase-2 benchmark result, fine-tuned out-domain phase-1 benchmark result, and fine-tuned out-domain phase-2 benchmark result.

2.2. Visualize the z-latent representation when behaving in in-domain/out-of-domain terrains

In Assignment 3, the results were obtained as follows. These results include A1 w/o adapt (blue), A1 w/ adapt (green), Gol w/o adapt (red), and Gol w/ adapt (purple) TSNE plots. The overlaps suggest that Gol w/o adapt features tend to be dissimilar to the other quadruped models, which could be due to the morphology, making it difficult to estimate the environmental parameter at that point. From Assignment 3, it was confirmed that adjusting the variable fractalLacunarity yields results as seen (the results of hw3 were received from the gsi).

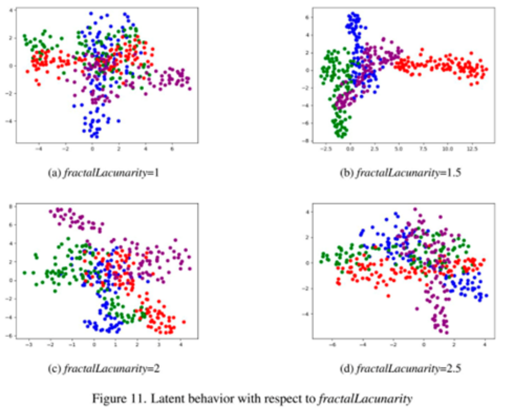
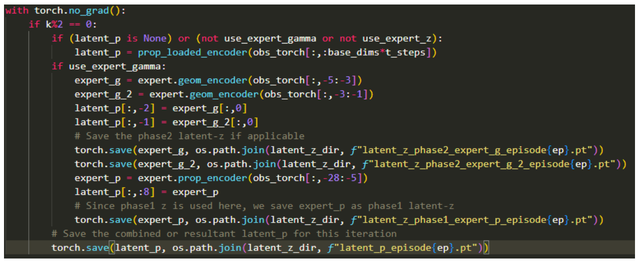


Figure 10. Visualize the z-latent representation when behaving in in-domain/out-of-domain terrains in the pre-trained results

I add the following code to find the values of the latent-z.



```
with torch.no_grad():
    if k%2 == 0:
        if (latent_p is None) or (not use_expert_gamma or not use_expert_z):
            latent_p = prop_loaded_encoder(obs_torch[:,:base_dims*t_steps])
        if use_expert_gamma:
            expert_g = expert.geom_encoder(obs_torch[:,-5:-3])
            expert_g_2 = expert.geom_encoder(obs_torch[:,-3:-1])
            latent_p[:,-2] = expert_g[:,0]
            latent_p[:,-1] = expert_g_2[:,0]
            # Save the phase2 latent-z if applicable
            torch.save(expert_g, os.path.join(latent_z_dir, f"latent_z_phase2_expert_g_episode{ep}.pt"))
            torch.save(expert_g_2, os.path.join(latent_z_dir, f"latent_z_phase2_expert_g_2_episode{ep}.pt"))
            expert_p = expert.prop_encoder(obs_torch[:,-28:-5])
            latent_p[:,:8] = expert_p
            # Since phase1 z is used here, we save expert_p as phase1 latent-z
            torch.save(expert_p, os.path.join(latent_z_dir, f"latent_z_phase1_expert_p_episode{ep}.pt"))
        # Save the combined or resultant latent_p for this iteration
        torch.save(latent_p, os.path.join(latent_z_dir, f"latent_p_episode{ep}.pt"))
```

Figure 11. Added code to find the latent-z

3. Explore another Fine-tuning Strategy

Explicit Inductive Bias for Transfer Learning with Convolutional Networks Concept: The idea is to introduce prior knowledge into the convolutional layers that make the network predisposed to learn certain features that are beneficial for the task at hand.

Integration Approach: For terrains adaptation, an inductive bias might be to focus on textural and shape features that are indicative of different terrains. This can be done by pre-training the convolutional layers on datasets that contain a variety of textures and shapes similar to the terrains the robot will encounter, then freezing these layers during fine-tuning.

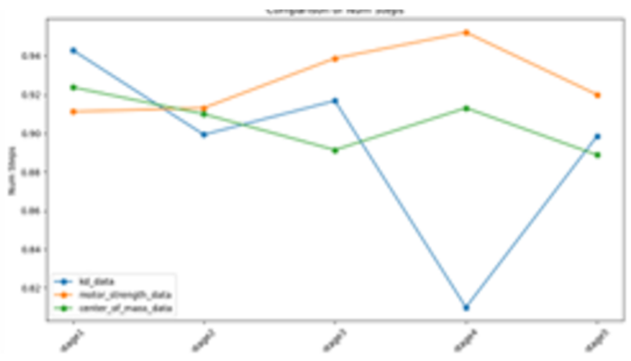


Figure 12. Fine-tuned plot result of Explicit Inductive Bias for Transfer Learning with Convolutional Networks

Overcoming Catastrophic Forgetting in Neural Networks Concept: Catastrophic forgetting occurs when a network forgets previously learned information upon learning new information. This is especially problematic in continual learning settings.

Integration Approach: A common technique to mitigate this issue is Elastic Weight Consolidation (EWC), which slows down learning on certain weights based on their importance to previously learned tasks.

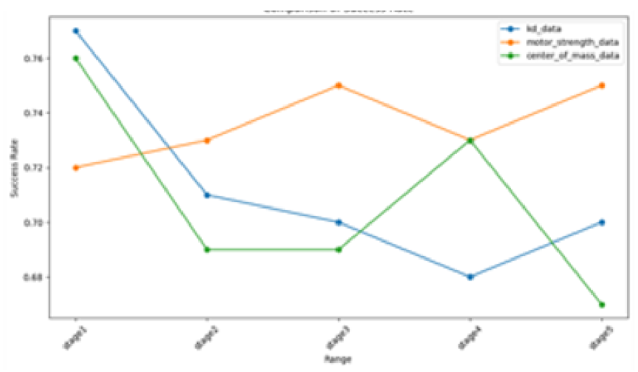


Figure 13. Fine-tuned plot result of Overcoming Catastrophic Forgetting in Neural Networks

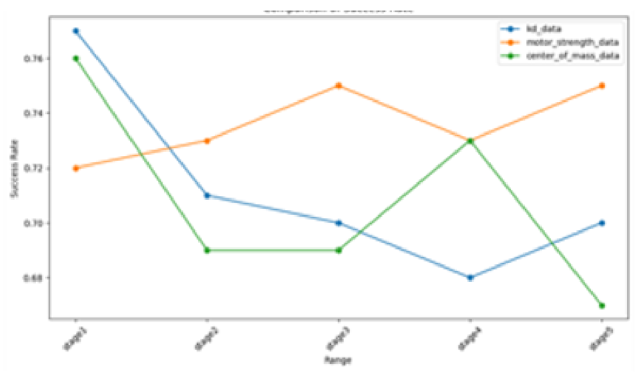


Figure 14. Fine-tuned out-domain phase-2 benchmark result

In the first method, we specifically tailor the network to be sensitive to terrain-related features by pre-training on a texture and shape-rich dataset. The inductive bias is introduced by freezing the weights of the convolutional layers, ensuring that these layers act as feature extractors during fine-tuning.

In the second method, the EWC technique is used to preserve the performance on the previous tasks (e.g., walking on known terrains) while continuing to learn new tasks (e.g., new terrains). The ewc-loss term penalizes changes to important weights, preventing catastrophic forgetting.

References

Assignment 5: From RMA to CMS – Tailoring Visual info

Wonseok Oh
University of Michigan Ann Arbor
Electrical and Computer Engineering
okong@umich.edu

1. Workspace Set Up

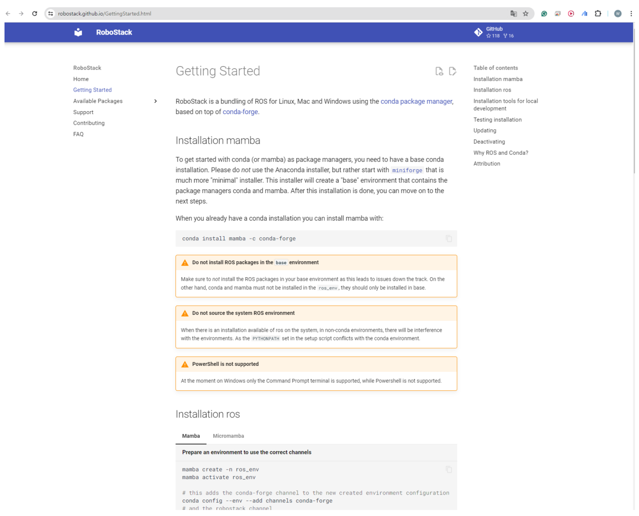
1.1. ROS Noetic Installation

1.1.1 S1. Ubuntu version in 20.04

The shared desktop is in Ubuntu version 22.04

1.1.2 S2. Use Conda to install RoboStack

Use Conda to install RoboStack. It is a bundling of the ROS for Linux, Mac, and Windows using the Conda package manager. Please refer to https://robostack.github.io/ GettingStarted.html for more conceptual details and instructions for installing ROS-1.



Getting Started

RoboStack is a bundling of ROS for Linux, Mac and Windows using the conda package manager, based on top of conda-forge.

Installation mamba

To get started with conda (or mamba) as package managers, you need to have a base conda installation. Please do *not* use the Anaconda installer, but rather start with `miniforge` that is much more "minimal" installer. This installer will create a "base" environment that contains the package managers conda and mamba. After this installation is done, you can move on to the next steps.

When you already have a conda installation you can install mamba with:

```
conda install mamba -c conda-forge
```

Do not install ROS packages in the `base` environment

Make sure to *not* install the ROS packages in your base environment as this leads to issues down the track. On the other hand, conda and mamba must not be installed in the `ros_env`, they should only be installed in base.

Do not source the system ROS environment

When there is an installation available of ros on the system, in non-conda environments, there will be interference with the environments. As the `PYTHONPATH` set in the setup script conflicts with the conda environment.

PowerShell is not supported

At the moment on Windows only the Command Prompt terminal is supported, while Powershell is not supported.

Installation ros

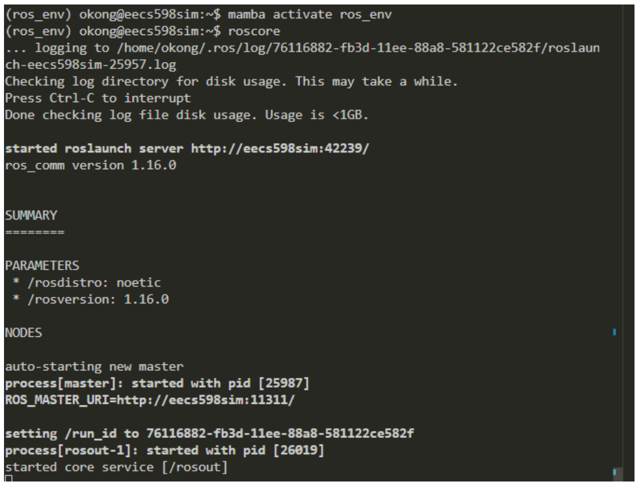
Prepare an environment to use the correct channels

```
mamba create -n ros_env
mamba activate ros_env

# this adds the conda-forge channel to the new created environment configuration
conda config --env --add channels conda-forge
# and the robostack channel
```

Figure 1. Website and the installation guide of Robostack

(a): I activate the installed environment (for example, your environment is named ros env) via mamba. The result can be found in Fig 2.

(b): For executing rivs, we need a different desktop. By using the existing desktop, we cannot get a suitable result for (b). (I used the 141.212.106.149)



```
(ros_env) okong@eecs598sim:~$ mamba activate ros_env
(ros_env) okong@eecs598sim:~$ roscore
... logging to /home/okong/.ros/log/76116882-fb3d-11ee-88a8-581122ce582f/roslaun
ch-eecs598sim-25957.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://eecs598sim:42239/
ros_comm version 1.16.0


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [25987]
ROS_MASTER_URI=http://eecs598sim:11311/

setting /run_id to 76116882-fb3d-11ee-88a8-581122ce582f
process[rosout-1]: started with pid [26019]
started core service [/rosout]
```

Figure 2. Run result of the roscore

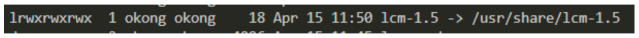
1.2. LCM Installation

1.2.1 S1. Installing dependencies. Before setting up LCM, install Lua, Java, and Go via conda

I used (a) For Lua: conda install conda-forge::lua, (b) For Go: conda install anaconda::go, (c) For Java: conda install conda-forge::openjdk these three commands to install the dependencies.

1.2.2 S2. Installing LCM

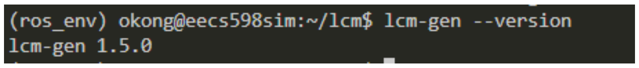
We need to follow the steps as follows. (a) Download and place lcm package via: cd git clone https://github.com/lcm-proj/lcm cd lcm (b) Compile lcm: cmake . sudo make install (c) Test if lcm is correctly installed via running: lcm-gen –version. After that, we should link lcm-1.5 from /usr/share like Fig 3



```
lrwxrwxrwx  1 okong okong    18 Apr 15 11:50 lcm-1.5 -> /usr/share/lcm-1.5
```

Figure 3. Linked picture of lcm-1.5 from /usr/share

If we execute the final result, then we can get the output as Fig 14.



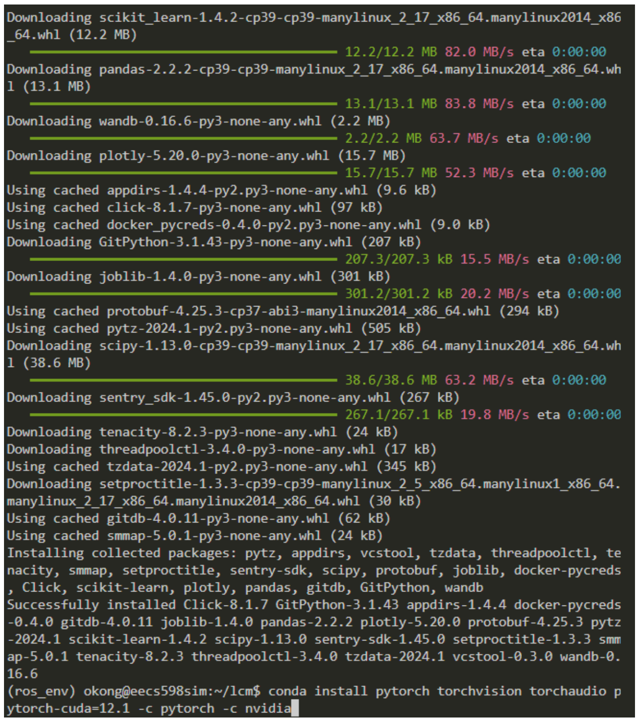
```
(ros_env) okong@eecs598sim:~/lcm$ lcm-gen --version
lcm-gen 1.5.0
```

Figure 4. Output image of lcm-gen –version. We can test lcm's version like this figure

1.3. CMS Set Up

1.3.1 S1. Pytorch and development packages installation

Based on your computing platform, install a suitable Pytorch version. We install Pytorch 2.2.1 with CUDA 12.1. Refer to Section 1 in README-1 for concrete commands.

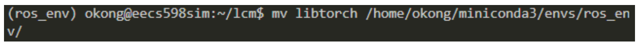


```
Downloading scikit_learn-1.4.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl (12.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 82.0 MB/s eta 0:00:00
Downloading pandas-2.2.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.wh
l (13.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 83.8 MB/s eta 0:00:00
Downloading wandb-0.16.6-py3-none-any.whl (2.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 63.7 MB/s eta 0:00:00
Downloading plotly-5.20.0-py3-none-any.whl (15.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.7/15.7 MB 52.3 MB/s eta 0:00:00
Using cached appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Using cached click-8.1.7-py3-none-any.whl (97 kB)
Using cached docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
Downloading GitPython-3.1.43-py3-none-any.whl (207 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.3/207.3 kB 15.5 MB/s eta 0:00:00
Downloading joblib-1.4.0-py3-none-any.whl (301 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 301.2/301.2 kB 20.2 MB/s eta 0:00:00
Using cached protobuf-4.25.3-cp37-abi3-manylinux2014_x86_64.whl (294 kB)
Using cached pytz-2024.1-py2.py3-none-any.whl (505 kB)
Downloading scipy-1.13.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.wh
l (38.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 38.6/38.6 MB 63.2 MB/s eta 0:00:00
Downloading sentry_sdk-1.45.0-py2.py3-none-any.whl (267 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 267.1/267.1 kB 19.8 MB/s eta 0:00:00
Downloading tenacity-8.2.3-py3-none-any.whl (24 kB)
Downloading threadpoolctl-3.4.0-py3-none-any.whl (17 kB)
Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Downloading setproctitle-1.3.3-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (30 kB)
Using cached gitdb-4.0.11-py3-none-any.whl (62 kB)
Using cached smmap-5.0.1-py3-none-any.whl (24 kB)
Installing collected packages: pytz, appdirs, vcstool, tzdata, threadpoolctl, te
nacity, smmap, setproctitle, sentry-sdk, scipy, protobuf, joblib, docker-pycreds
, Click, scikit-learn, plotly, pandas, gitdb, GitPython, wandb
Successfully installed Click-8.1.7 GitPython-3.1.43 appdirs-1.4.4 docker-pycreds
-0.4.0 gitdb-4.0.11 joblib-1.4.0 pandas-2.2.2 plotly-5.20.0 protobuf-4.25.3 pytz
-2024.1 scikit-learn-1.4.2 scipy-1.13.0 sentry-sdk-1.45.0 setproctitle-1.3.3 smm
ap-5.0.1 tenacity-8.2.3 threadpoolctl-3.4.0 tzdata-2024.1 vcstool-0.3.0 wandb-0.
16.6
(ros_env) okong@eecs598sim:~/lcm$ conda install pytorch torchvision torchaudio p
ytorch-cuda=12.1 -c pytorch -c nvidia
```

Figure 5. This figure is the figure that displays the installed result of the pytorch and development packages

1.3.2 S2. Libtorch installization

Download a suitable zip file from https://pytorch.org/get-started/locally/. Unzip the package and place the libtorch under your conda-ros environment

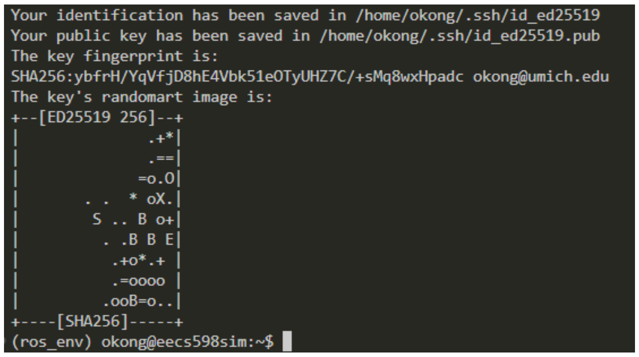


```
(ros_env) okong@eecs598sim:~/lcm$ mv libtorch /home/okong/miniconda3/envs/ros_en
v/
```

Figure 6. Placing the libtorch under the conda-ros env

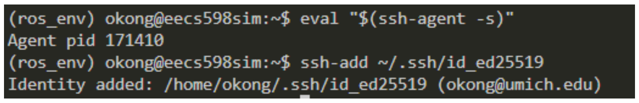
1.3.3 S3. Libtorch installization

Use catkin tool to compile CMS environment. Refer to Section-2 in README-1 for concrete commands and modifications.



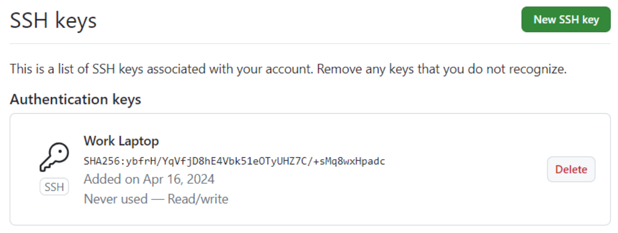
```
Your identification has been saved in /home/okong/.ssh/id_ed25519
Your public key has been saved in /home/okong/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ybfrH/YqVfjD8hE4Vbk51eOTyUHZ7C/+sMq8wxHpadc okong@umich.edu
The key's randomart image is:
+--[ED25519 256]--+
|              .+*|
|              .==|
|             =o.O|
|        . .  * oX.|
|         S .. B o+|
|           . .B B E|
|            .+o*.+ |
|            .=oooo |
|           .ooB=o..|
+----[SHA256]-----+
(ros_env) okong@eecs598sim:~$
```

Figure 7. Identification Key of my id_ed25519



```
(ros_env) okong@eecs598sim:~$ eval "$(ssh-agent -s)"
Agent pid 171410
(ros_env) okong@eecs598sim:~$ ssh-add ~/.ssh/id_ed25519
Identity added: /home/okong/.ssh/id_ed25519 (okong@umich.edu)
```

Figure 8. Identity added result



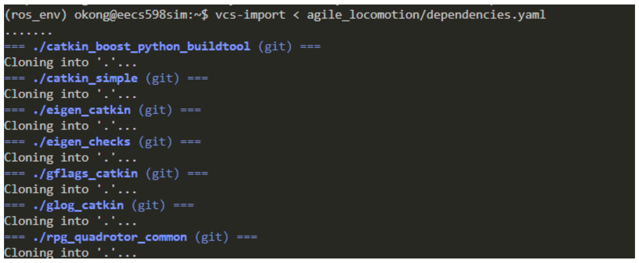
SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys

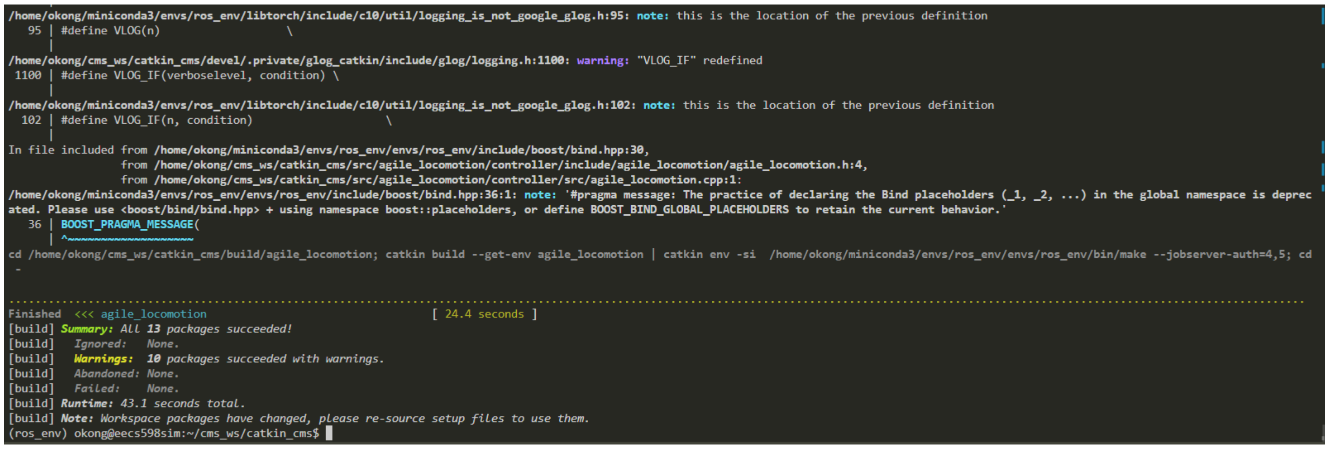
Work Laptop
SHA256:ybfrH/YqVfjD8hE4Vbk51eOTyUHZ7C/+sMq8wxHpadc
Added on Apr 16, 2024
Never used — Read/write

Figure 9. I registered my ssh key in my GitHub account



```
(ros_env) okong@eecs598sim:~$ vcs-import < agile_locomotion/dependencies.yaml
.......
=== ./catkin_boost_python_buildtool (git) ===
Cloning into '.'...
=== ./catkin_simple (git) ===
Cloning into '.'...
=== ./eigen_catkin (git) ===
Cloning into '.'...
=== ./eigen_checks (git) ===
Cloning into '.'...
=== ./gflags_catkin (git) ===
Cloning into '.'...
=== ./glog_catkin (git) ===
Cloning into '.'...
=== ./rpg_quadrotor_common (git) ===
Cloning into '.'...
```

Figure 10. Cloning the dependencies which is agile_locomotion/dependencies.yaml



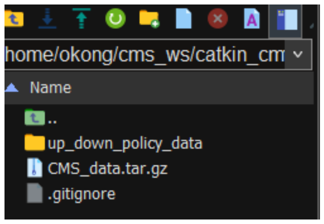
```
/home/okong/miniconda3/envs/ros_env/libtorch/include/c10/util/logging_is_not_google_glog.h:95: note: this is the location of the previous definition
   95 | #define VLOG(n)                   \
      |
/home/okong/cms_ws/catkin_cms/devel/.private/glog_catkin/include/glog/logging.h:1100: warning: "VLOG_IF" redefined
 1100 | #define VLOG_IF(verboselevel, condition) \
      |
/home/okong/miniconda3/envs/ros_env/libtorch/include/c10/util/logging_is_not_google_glog.h:102: note: this is the location of the previous definition
  102 | #define VLOG_IF(n, condition)                   \
      |
In file included from /home/okong/miniconda3/envs/ros_env/envs/ros_env/include/boost/bind.hpp:30,
                 from /home/okong/cms_ws/catkin_cms/src/agile_locomotion/controller/include/agile_locomotion/agile_locomotion.h:4,
                 from /home/okong/cms_ws/catkin_cms/src/agile_locomotion/controller/src/agile_locomotion.cpp:1:
/home/okong/miniconda3/envs/ros_env/envs/ros_env/include/boost/bind.hpp:36:1: note: '#pragma message: The practice of declaring the Bind placeholders (_1, _2, ...) in the global namespace is deprec
ated. Please use <boost/bind/bind.hpp> + using namespace boost::placeholders, or define BOOST_BIND_GLOBAL_PLACEHOLDERS to retain the current behavior.'
   36 | BOOST_PRAGMA_MESSAGE(
      | ^~~~~~~~~~~~~~~~~~~~
cd /home/okong/cms_ws/catkin_cms/build/agile_locomotion; catkin build --get-env agile_locomotion | catkin env -si  /home/okong/miniconda3/envs/ros_env/envs/ros_env/bin/make --jobserver-auth=4,5; cd
 -

.......................................................................................................................................................
Finished  <<< agile_locomotion                            [ 24.4 seconds ]
[build] Summary: All 13 packages succeeded!
[build]   Ignored:   None.
[build]   Warnings:  10 packages succeeded with warnings.
[build]   Abandoned: None.
[build]   Failed:    None.
[build] Runtime: 43.1 seconds total.
[build] Note: Workspace packages have changed, please re-source setup files to use them.
(ros_env) okong@eecs598sim:~/cms_ws/catkin_cms$
```

Figure 11. Final output of the complied CMS environment

2. Fine-tune the Vision Predictor

2.1. Download the dataset and move it

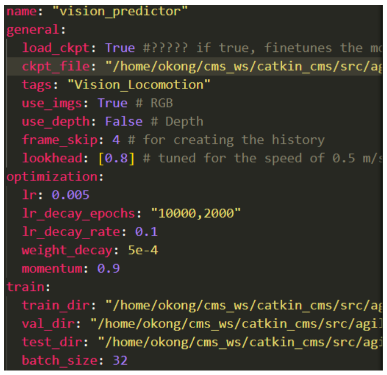
Download the dataset and put it under your /cms_ws/catkin_cms/src/agile locomotion/data/ path.



```
home/okong/cms_ws/catkin_cm
Name
..
up_down_policy_data
CMS_data.tar.gz
.gitignore
```

Figure 12. Put the download data into /cms ws/catkin cms/src/agile locomotion/data/ path.

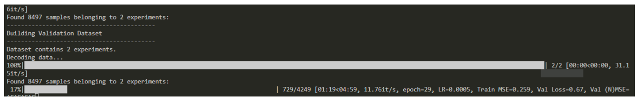
2.2. Modify the training config



```
name: "vision_predictor"
general:
  load_ckpt: True #????? if true, finetunes the mo
  ckpt_file: "/home/okong/cms_ws/catkin_cms/src/ag
  tags: "Vision_Locomotion"
  use_imgs: True # RGB
  use_depth: False # Depth
  frame_skip: 4 # for creating the history
  lookhead: [0.8] # tuned for the speed of 0.5 m/
optimization:
  lr: 0.005
  lr_decay_epochs: "10000,2000"
  lr_decay_rate: 0.1
  weight_decay: 5e-4
  momentum: 0.9
train:
  train_dir: "/home/okong/cms_ws/catkin_cms/src/ag
  val_dir: "/home/okong/cms_ws/catkin_cms/src/agil
  test_dir: "/home/okong/cms_ws/catkin_cms/src/agi
  batch_size: 32
```

Figure 13. Modify the training config under your 'catkin cms/src/cms/visual locomotion/learning' path with changing load ckpt to True, ckpt file to model's path

2.3. Final training of the model



```
6it/s]
Found 8497 samples belonging to 2 experiments:
-----------------------------------------
Building Validation Dataset
-----------------------------------------
Dataset contains 2 experiments.
Decoding data...
100%|                                                  | 2/2 [00:00<00:00, 31.1
5it/s]
Found 8497 samples belonging to 2 experiments:
 17%|                               | 729/4249 [01:19<04:59, 11.76it/s, epoch=29, LR=0.0005, Train MSE=0.259, Val Loss=0.67, Val (N)MSE=
```

Figure 14. Training process of the model in provided dataset

The last part of the report is to make the converging plot of the loss. I used Validation loss to get the plot of the training model.

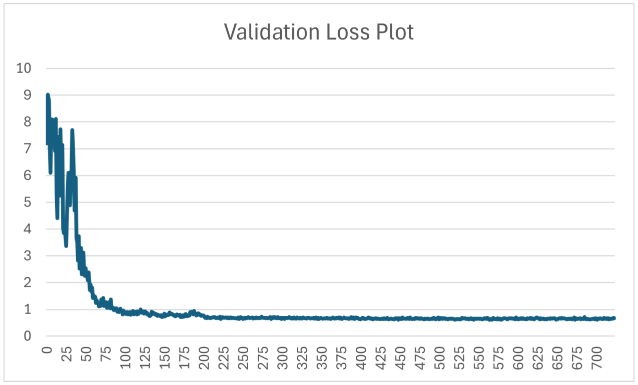


Figure 15. Validation loss plot of training

Due to CUDA's characteristics not aligning well with my needs, I opted to run training on CPU. During this process, I encountered a recurring issue where the training would halt midway. To address this, I saved the validation loss data into a CSV file and subsequently generated a graph from this data. Through this method, I was able to carry out approximately 720 training iterations, and it was evident that convergence occurred within the first 100 steps. This is displayed in the Fig 15. This suggests that the model is capable of rapidly adapting to new data, a desirable characteristic when fine-tuning for performance in novel visual environments. Moreover, the rapid convergence indicates a potentially robust integration of proprioceptive and visual information, allowing the model to generalize well from the training data. Further analysis would be necessary to evaluate how this translates to practical applications and whether the model's performance is consistent across diverse scenarios. This is to be investigated using the actual quadruped robot in the last assistance.

EECS598 Final HW

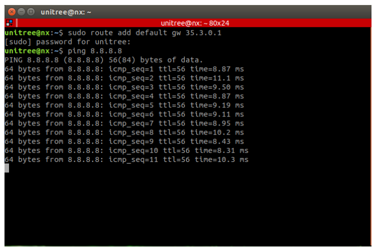
Zhengjie Xu, Hanxi Wan, Wonseok Oh, Elvin Yang

April 2024

1 Introduction

1.1 Workspace Set Up

1.1.1 Connecting GO-1 to the Internet



```
unitree@nx:~$ sudo route add default gw 35.3.0.1
[sudo] password for unitree:
unitree@nx:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=56 time=8.87 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=56 time=11.1 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=56 time=9.50 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=56 time=8.87 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=56 time=9.19 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=56 time=9.11 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=56 time=8.95 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=56 time=10.2 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=56 time=8.43 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=56 time=8.31 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=56 time=10.3 ms
```

Figure 1: Successful connecting to the internet

1.1.2 Workspace Installation

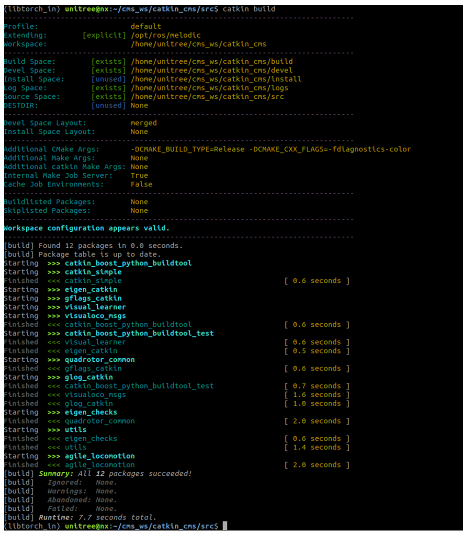
Please refer to Figure 2. As we can see, the workspace is configured and built correctly.

2 Real-world Evaluation

In this section, our trained policy is tested in various real-world environments. To evaluate the robot's adaptability and robustness, we qualitatively compared our policy's results with Go-1's built-in MPC+IK algorithm.

We designed a series of different terrains to test the robot's abilities. These terrains include:

- Sand
- Sand with rocks



```
(libtorch_in) unitree@nx:~/cms_ws/catkin_cms/src$ catkin build
--------------------------------------------------------------------------------
Profile:                     default
Extending:        [explicit] /opt/ros/melodic
Workspace:                   /home/unitree/cms_ws/catkin_cms
--------------------------------------------------------------------------------
Build Space:        [exists] /home/unitree/cms_ws/catkin_cms/build
Devel Space:        [exists] /home/unitree/cms_ws/catkin_cms/devel
Install Space:      [unused] /home/unitree/cms_ws/catkin_cms/install
Log Space:          [exists] /home/unitree/cms_ws/catkin_cms/logs
Source Space:       [exists] /home/unitree/cms_ws/catkin_cms/src
DESTDIR:            [unused] None
--------------------------------------------------------------------------------
Devel Space Layout:          merged
Install Space Layout:        None
--------------------------------------------------------------------------------
Additional CMake Args:       -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS=-fdiagnostics-color
Additional Make Args:        None
Additional catkin Make Args: None
Internal Make Job Server:    True
Cache Job Environments:      False
--------------------------------------------------------------------------------
Buildlisted Packages:        None
Skiplisted Packages:         None
--------------------------------------------------------------------------------
Workspace configuration appears valid.
--------------------------------------------------------------------------------
[build] Found 12 packages in 0.0 seconds.
[build] Package table is up to date.
Starting  >>> catkin_boost_python_buildtool
Starting  >>> catkin_simple
Finished  <<< catkin_simple                                    [ 0.6 seconds ]
Starting  >>> eigen_catkin
Starting  >>> gflags_catkin
Starting  >>> visual_learner
Starting  >>> visualoco_msgs
Finished  <<< catkin_boost_python_buildtool                    [ 0.6 seconds ]
Starting  >>> catkin_boost_python_buildtool_test
Finished  <<< visual_learner                                   [ 0.6 seconds ]
Finished  <<< eigen_catkin                                     [ 0.5 seconds ]
Starting  >>> quadrotor_common
Finished  <<< gflags_catkin                                    [ 0.6 seconds ]
Starting  >>> glog_catkin
Finished  <<< catkin_boost_python_buildtool_test               [ 0.7 seconds ]
Finished  <<< visualoco_msgs                                   [ 1.6 seconds ]
Finished  <<< glog_catkin                                      [ 1.0 seconds ]
Starting  >>> eigen_checks
Finished  <<< quadrotor_common                                 [ 2.0 seconds ]
Starting  >>> utils
Finished  <<< eigen_checks                                     [ 0.6 seconds ]
Finished  <<< utils                                            [ 1.4 seconds ]
Starting  >>> agile_locomotion
Finished  <<< agile_locomotion                                 [ 2.0 seconds ]
[build] Summary: All 12 packages succeeded!
[build]   Ignored:   None.
[build]   Warnings:  None.
[build]   Abandoned: None.
[build]   Failed:    None.
[build] Runtime: 7.7 seconds total.
(libtorch_in) unitree@nx:~/cms_ws/catkin_cms/src$
```

Figure 2: Successful catkin build of the workspace

- Half and half: where the robot's left and right legs are on different terrains, e.g., grass and sand.
- Uphill: where the robot climbs a hill with slippy sand on the surface.
- Up/Down stairs.

To test the robot's performance in these environments, we manually controlled its velocity and yaw angle using a joystick.

The evaluation videos can be found in the following link: `https://docs.google.com/presentation/d/1C3ed4iqJyF-YOh46ZnzuxhPuRl4O1pEmbMjD_HdgLDc`.Subsequently, we will analyze the performance on each terrain.

- Sand: Comparatively, the built-in controller achieves higher speeds than our policy. While sand presents a slippery surface, the terrain remains mostly flat, ensuring stability for both policies. The order of terrains doesn't significantly influence the robot's performance.
- Sand with rocks: The built-in controller is not able to go over the rock and we have to go around the rock, while our policy is able to navigate through them directly. However, for some large rocks, our policy encounters difficulty. The order of terrains doesn't significantly influence the robot's performance.
- Half and Half: Both the built-in controller and our policy are able to adapt to different terrains. The order of terrains doesn't have a significant influence on the robot's performance.

- Uphill: The hill is so steep and slippy that none of the policy is able to climb to the top. Our policy goes further compared to the built-in controller, showing its better performance in adapting to different terrains. The order of terrains doesn't significantly influence the robot's performance.

- Up/Down stairs: Both policies successfully go up or down stairs, yet our policy exhibits notably faster and smoother traversal compared to the built-in policy. The order of terrains doesn't significantly influence the robot's performance. Figure 3 and 4 show the motion of robot when going up the stairs.

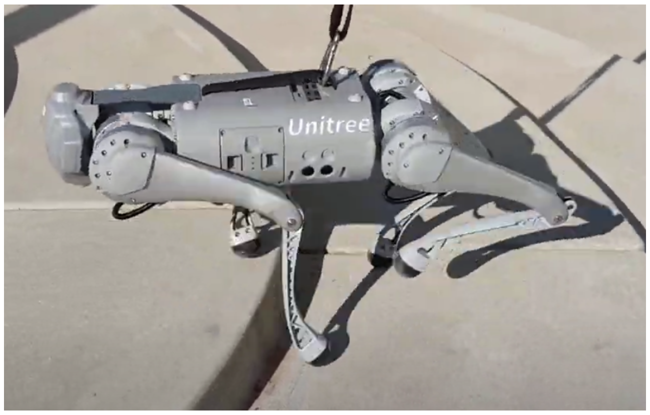


Figure 3: Up Stairs w/ Built-in Controller.

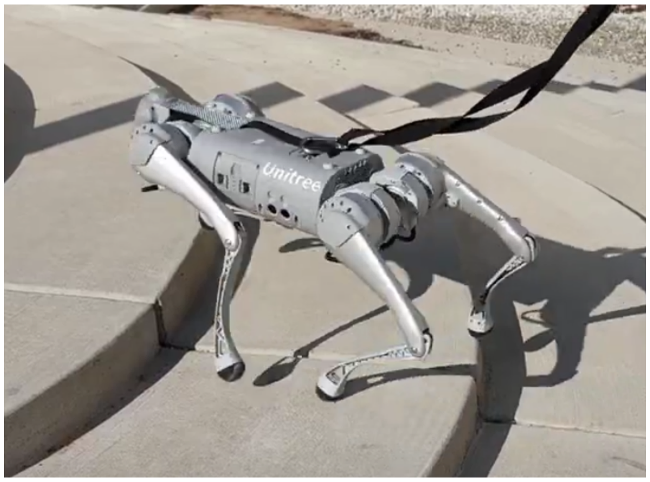


Figure 4: Up Stairs w/ Our Policy.