# Simulating Biped Locomotion in Nvidia Isaac Sim

Alexander Kumar
Email: ak478@duke.edu

## I. Introduction

The subfield of humanoid robotics is an exciting frontier in robotics research. Humanoid robots fall into the legged class of robots. Like most legged robots, they can traverse over more complex terrain, displaying adaptable and robust locomotion behavior. Due to their anthropomorphic design, they will be more compatible with human environments, requiring less modification to the environment to facilitate cooperative behaviors.

Reinforcement learning typically takes a long time to train and converge. Due to the large number of training samples required, robots are initially trained in a virtual gym environment. This enables quick resets, fast iteration and modification, and faster training. When implemented in the real world, these virtually-learned policies tend to initially struggle. This is called the Sim2Real gap. This arises from complications and inaccuracies introduced in the real world that were absent during training time. To lessen the Sim2Real gap, one approach is to elevate the quality of the simulation. NVIDIA Isaac Sim is a bleeding-edge simulation software built for robotics simulation. The high-quality physics, materials, and visualizations enable more accurate simulation. It also has built-in tools for synthetic data generation to make learned policies more robust. For these reasons, NVIDIA Isaac Sim was chosen as the software to carry out the experiment.

Bipedal robots tend to be less steady, designs have a higher center of gravity with an unfixed base, and have more degrees of freedom (DOF). These two factors in conjunction make this a harder learning task. The high DOFs make the state and search space exponentially larger. The less steady design makes it difficult to find and converge on an appropriate walking policy.

## II. Problem

To get adaptable behavior, this project's approach is to train a locomotion policy using reinforcement learning.

$$\text{(joint positions, velocities, goal location)} \rightarrow \text{motor actions}$$

By not including environmental information or observations, the bipedal robot must learn a robust and reactive policy in order to successfully move to the goal location. It must also learn a meaningful internal representation of the body's state. This project investigates methods to learn a walking policy for a 12 DOF bipedal robot. Leveraging massive parallelization in Isaac Sim, this project explores reward function shaping and state space limitations to accelerate both the learning process and the policy learned.

## III. Related Work

### A. Learning to Walk

Rudin et al. (1) use massively parallel deep reinforcement learning in NVIDIA IsaacGym, the predecessor to Isaac Sim. The researchers successfully trained a policy for locomotion in minutes. They were able to achieve these results through hyperparameter tuning and parallelization. Most of their work was on the ANYmal robots, which are quadrupeds. A biped named Cassie is briefly mentioned, although not much information is extrapolated about this part of the experiment. The quadruped robots have a lower center of gravity. Although they have a similar complexity in terms of search space defined 12 degrees of freedom, promising policies may be easier to find and iterate over with four stable points of contact.

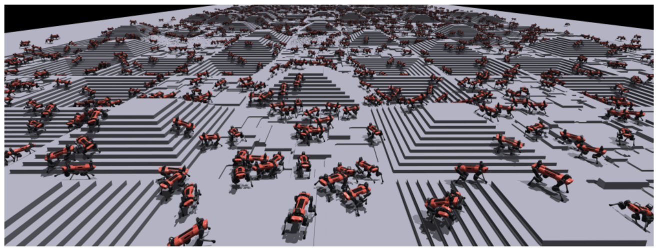
Figure 1: Thousands of robots learning to walk in simulation.

Fig. 1: (1)

### B. Humanoid Locomotion with Transformers

Radosavovic et al. (2) use a transformer architecture to learn humanoid locomotion. They also use IsaacGym's massively parallel training capabilities to train walking policies and transfer them to the real world in a zero-shot manner. The results are promising, with great performance across mild perturbations in the environment. From the fairly simple training regime, the policy is reasonably robust to small disturbances without training against adversarial examples. If trained in software with more robust physics and sample randomization, the robustness of the policies may scale accordingly.

## IV. Methodology

### A. Actor-Critic Algorithm

The reinforcement learning architecture chosen is A2C Continuous, or actor-critic continuous. A2C is a reinforcement learning architecture that combines value-based and policy-based methods (3), where the Actor and Critic interact similarly to a generator and discriminator in a GAN architecture. The actor tries to output actions based on the current state that maximizes the expected reward. The actor learns the policy. The critic evaluates the actor's actions, estimating the value of the action-state pairs and providing a feedback signal to the actor network. The critic learns the values. Both components are trained simultaneously. This method aims to reap the benefits of policy-based and value-based methods and mitigate the weaknesses of both. By doing so, training theoretically becomes more stable and learning becomes easier. A2C was originally designed for discrete state spaces. In robotics, each motor angle and velocity can have a continuous value. The A2C Continuous algorithm is the adaption for this added complexity.

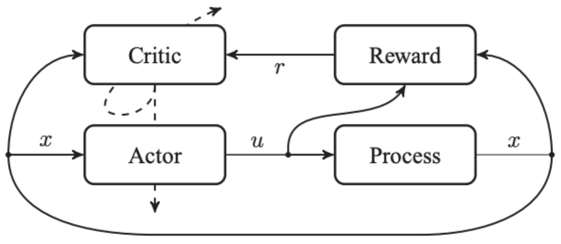

Fig. 1. Schematic overview of an actor-critic algorithm. The dashed line indicates that the critic is responsible for updating the actor and itself.

Fig. 2: (3)

### B. Crafting a Reward Function

A key component of the investigation was crafting the reward function to encourage the biped to walk. The initial formulation of the reward function was based on the Locomotion task listed in (4).

$$\begin{aligned}\text{reward} =& (\text{alive reward}) + (\text{progress reward}) + (\text{upright bonus}) \\ &+ (\text{orient bonus}) - (\text{action cost}) - (\text{motorlimit cost}) \\ &- (\text{death cost})\end{aligned}$$

This reward function encapsulates various factors contributing to the overall reinforcement signal. The components include positive aspects such as the "alive reward," "progress reward", (moving closer to the goal location) "upright bonus," (alignment angle with y-axis) and "orient bonus," (alignment angle toward goal location) which encourage behaviors associated with successful locomotion. On the contrary, negative components like "action cost," "motor limit cost," and "death cost" penalize undesired actions and states, discouraging inefficient or unsafe behavior.

Across the experiments, variables were added and removed. The reward function was stripped to its simplest version, then according to feedback from the previous experiment's logging and visualizations, the reward function was changed. Initially, changes to the reward function were determined by visual feedback from the policy run during in an inference environment. Many experiments and iterations were needed before the robot could begin to learn. The experiments shown in this paper are a limited but representational selection of how the experiments evolved.

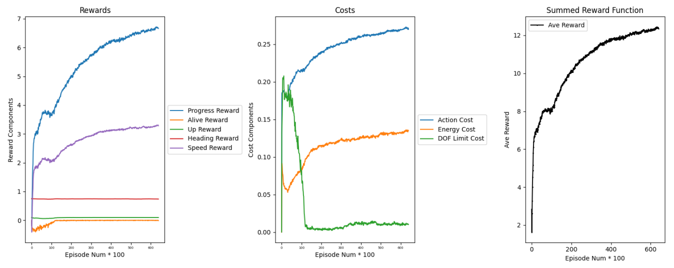

Fig. 3: Example reward plot from experiment 8

### C. Creating a Logger

To gain better insights in the experiment iteration process, a logger was created. The logger dumps reward buffer information to a text file during train time. To be space efficient, information is logged every 100 episodes. The logged values are the mean values across all parallel environments. A parser and plotter were also created to extract information from the log file and display the information in a digestible way. Additional plots were made using tensorboard, recording valuable loss and convergence data from the training A2C model.

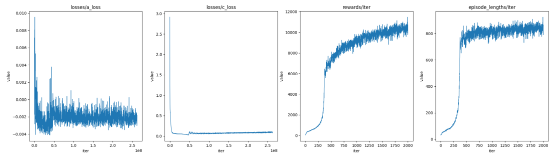

Fig. 4: Example tensorboard plot from experiment 8

### D. Isaac Sim

The experiments were run in Isaac Sim. During training, 4096 environments are run in parallel. The number of epochs is set to 2000, the maximum episode length is 1000, and the control frequency is every 2 steps. Environments are reset when conditions are triggered, such as maximum episode length or the implemented reset height. The MLP network used to learn the actor policy has three hidden dimension sizes 400, 200, 100. Experiments were run headless to speedup wall clock training time. To evaluate each model, they were run in the GUI in test mode with two environments. All task specific experiment and config files were created by the author and can be found using the README here: (5). The bipedal robot used throughout all experiments was designed by the Humanoid Team at Duke's General Robotics Lab (6).

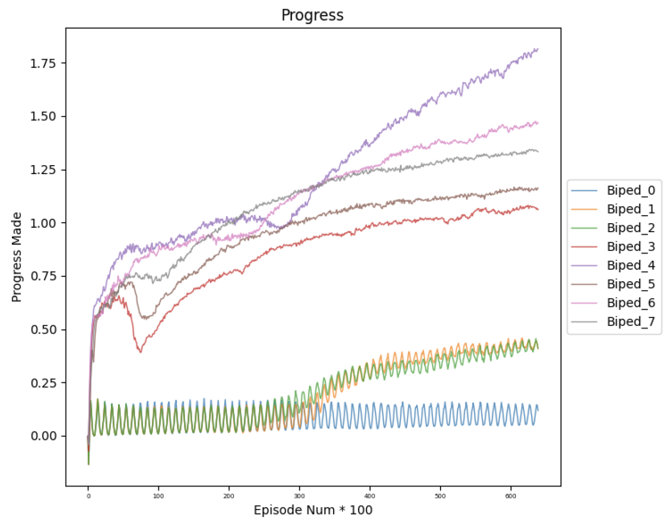

Fig. 5: Comparing Progress (Distance Towards Goal)

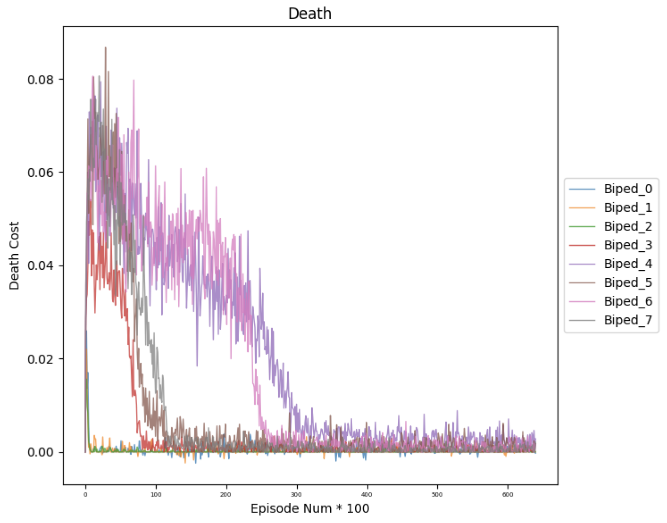

Fig. 6: Comparing Death Costs Across Experiments

## V. Experiments

1) learning rate = 5e-4:
default values / baseline experiment
2) learning rate = 5e-5:
decreased lr due to noise in a2c convergence
3) death cost = $-5.0 \leftarrow -1.0$:
robot doesn't stand
4) reset height = $0.7 \leftarrow 0.0$:
robot keeps falling / crawling
5) progress multiplier = $5.0 \leftarrow 1.0$:
reweight to encourage progress
6) reset upright = $0.7 \leftarrow 0.0$:
robot dolphin dives when unstable
7) reset orientation = $0.5 \leftarrow 0.0$:
robot side / backward shuffles
8) action cost, energy cost = $0.03 \leftarrow 0.01$, $0.1 \leftarrow 0.05$:
encourage smoother locomotion

The eight experiments selected and shown above give insight into the experimental process. Between each experiment, a key takeaway was formed based on the previous experiment, a hypothesis was made on how to best mitigate the issue found, and the modification was made to the experimental setup. It should be noted that there are many failed experiments that have been excluded for brevity, such as this. (It should also be noted that the video experiment naming conventions do not match up with the naming conventions used in this paper).

By analyzing the comparison plot in [5], we can see that with the initially higher learning rate of 5e-4, the policy was unable to learn how to move in any meaningful way. In the first three experiments, the robot falls straight forward or stands straight up. In experiment 3, the robot learns to stand. It also gradually increases the progress reward by oscillating its feet in a manner that allows it to inch forward.

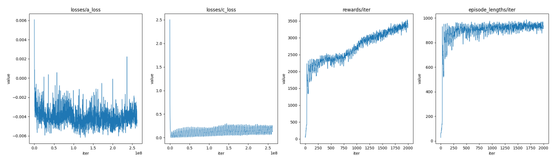

Fig. 7: Experiment 3 Tensorboard

This is not sufficient walking behavior. These first experiments displayed noisy actor and critic loss, signaling that the model was having difficulties converging, and when it is converging, it converges on a poor solution. This signals weak exploration in the exploration-exploitation tradeoff. The next approach was to limit the search and state space. As the A2C continuous architecture was struggling to find high performing policies within a reasonable amount of time, introducing new reset conditions would force the policy in the desired direction. The first new reset condition introduced was reset height due to crawling behavior seen here. This height strictly enforces "feet-walking" policies, prohibiting "knee-walking." The progress multiplier is added in the next experiment to encourage stronger walking gait behavior.

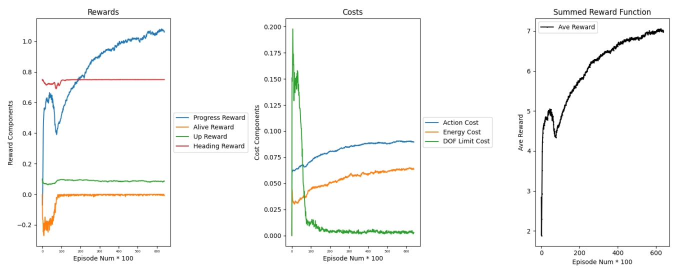

Fig. 8: Experiment 4 Rewards

From [5], it is clear that the models are learning to walk as progress reward increases semi-steadily over training. This indicates a dominant stable strategy. This is also reflected by the lower number of reset deaths in the neighboring plot [6]. In experiment 5, this is exhibited in the form of crab walking behavior. It is also observed that when the model believes it is in an unstable state, the biped dolphin dives to gain extra progress reward before dying. To discourage this myopic and unstable behavior, the upright reset was added. This strictly prohibits diving as the environment resets when the projected upward vector deviates from the global Z vector excessively.

This resulted in the behavior seen in experiment 6, where the robot walks quickly and stably. However, it does so oriented backwards. This is probably due to the robot's design where knees protrude in the positive X direction, rather than in the negative X direction like in Cassie from (1). To enforce a more human-like style, the orientation reset was implemented. This strictly prohibits walking backwards as the environment resets when the projected heading vector deviates from the global X vector excessively. This resulted in the behavior seen in experiment 7, where the robot finally learns a policy that acts and looks like a human walking motion. However, there is a notable amount of attitude in its stride.

## VI. Conclusions

### A. Challenges

There were many issues getting the project off the ground. Isaac Sim requires Linux. Up to this point, my machines only ran MacOSX and Windows 10. According to forums, Isaac Sim also struggles in WSL. I downloaded Ubuntu and ran into several issues, including memory allocation and driver compatibility. After Ubuntu was working, I then had to figure out how to setup and configure the Isaac Sim software. The program is relatively new with limited tutorials and outstanding issues. It took a while to launch the program and then learn how to work within it, as this was my first robotics simulation project.

### B. Key Takeaways

In the process of carrying out experiments for this project, challenges central to robotics research became clearer. Robotics research is difficult because of the interdisciplinary nature of the work. Many projects require expertise in mechanical engineering, electrical engineering, and computer science. The particular design of the bipedal robot used for this experiment affected the way locomotion was learned. If the knees had the opposite range of motion, the added orientation reset might not be necessary. Perhaps if the center of mass was lower the policy would be able to converge on stable configurations more quickly. The approaches and outcomes are extremely multifaceted.

The practice of reward shaping is time-consuming and inefficient. Crafting a reward function that effectively guides the learning process demands a substantial investment of time and expertise. Researchers must meticulously design rewards that not only capture the nuances of desired behaviors but also navigate the trade-offs between immediate task success and broader learning objectives. Additionally, the more a reward function is custom-shaped for a particular downstream task, the less generalizable it is to future alternative downstream tasks. This emphasizes the importance of meta-learning techniques. Adaptive algorithms that do not necessitate excessive reward shaping are essential for generalizing knowledge from one task and applying it to novel challenges.

By artificially restricting the search space, researchers can set up guide rails for agents to learn desired policies. By establishing height, orientation, and upright resets, the valid state space was restricted in a way that made effective and stylistic bipedal locomotion simpler to learn. However, this shaping is antithetical to the philosophy behind reinforcement learning - learning from raw inputs and allowing agents to explore and discover optimal strategies autonomously.

### C. Future Work

A future avenue for exploration is in incorporating domain randomization techniques in the environments. This would entail variations in goal points and terrains during training. By training on diverse scenarios, the model can develop a more comprehensive policy, bolstering its adaptability to real-world conditions and lessening the Sim2Real gap.

Another path forward is in meta-learning and early stopping mechanisms. Meta-learning techniques, reducing researcher reward-shaping, can significantly expedite the learning process. Moreover, the usage of early stopping methodologies allows for the fine-tuning of training durations, preventing excessive training on unpromising experimental configurations and training longer on promising configurations.

## References

[1] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," 2021.

[2] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Learning humanoid locomotion with transformers," *arXiv:2303.03381*, 2023.

[3] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[4] "Omniisaacgymenvs github homepage," https://github.com/NVIDIA-Omniverse/OmniIsaacGymEnvs/tree/main/omniisaacgymenvs.

[5] "Forked biped branch of the omniisaacgymenvs github," https://github.com/ACK101101/OmniIsaacGymEnvs/tree/biped.

[6] "General robotics lab," http://generalroboticslab.com/index.html.