Generating a Dynamic Controller for a Flamingo Inspired Robot Using Deep Reinforcement Learning

Edward Lu, Nathan J. Kong, J. Joseph Payne, and Aaron M. Johnson
Carnegie Mellon University, Department of Mechanical Engineering

Overview

This work explores the process of using deep reinforcement learning (DRL) to generate a dynamic walking controller for a simulated model of a flamingo-inspired robot, enabling us to swap out different design parameters (e.g. motor models) in simulation.

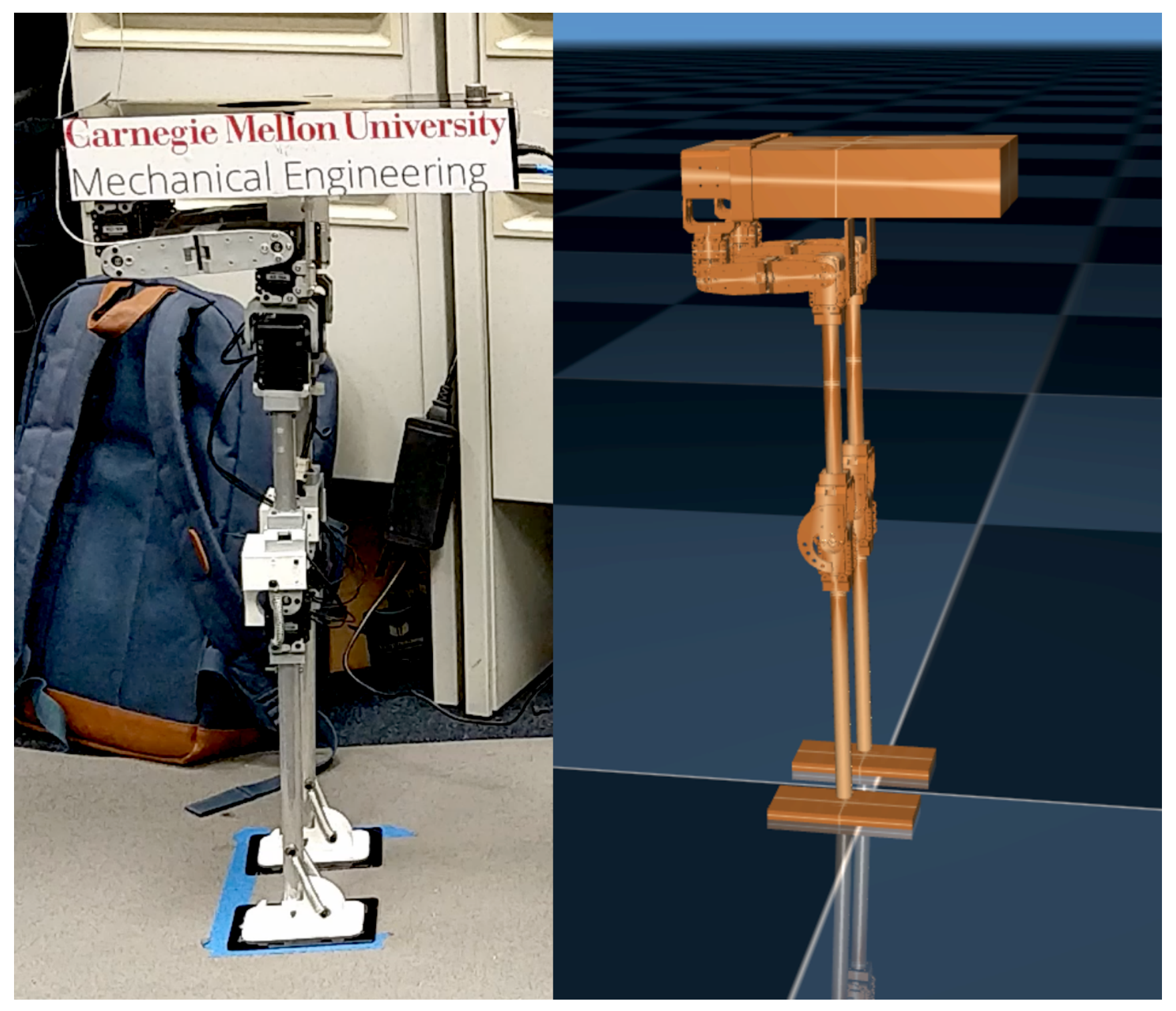

Figure 1: Real life prototype (left) and simulated model in Mujoco simulator (right).

Motivation

- In prior work, we created a passively stable bipedal robot, called "FlamingoBot," based off the skeletal structure of a flamingo (see Figure 1).
- The prototype is quite unstable when moving, making it difficult to perform a dynamic walking gait.
- DRL allows us to generate an efficient walk cycle without taking into account the robot's full dynamics, which are complex and time-consuming to optimize.

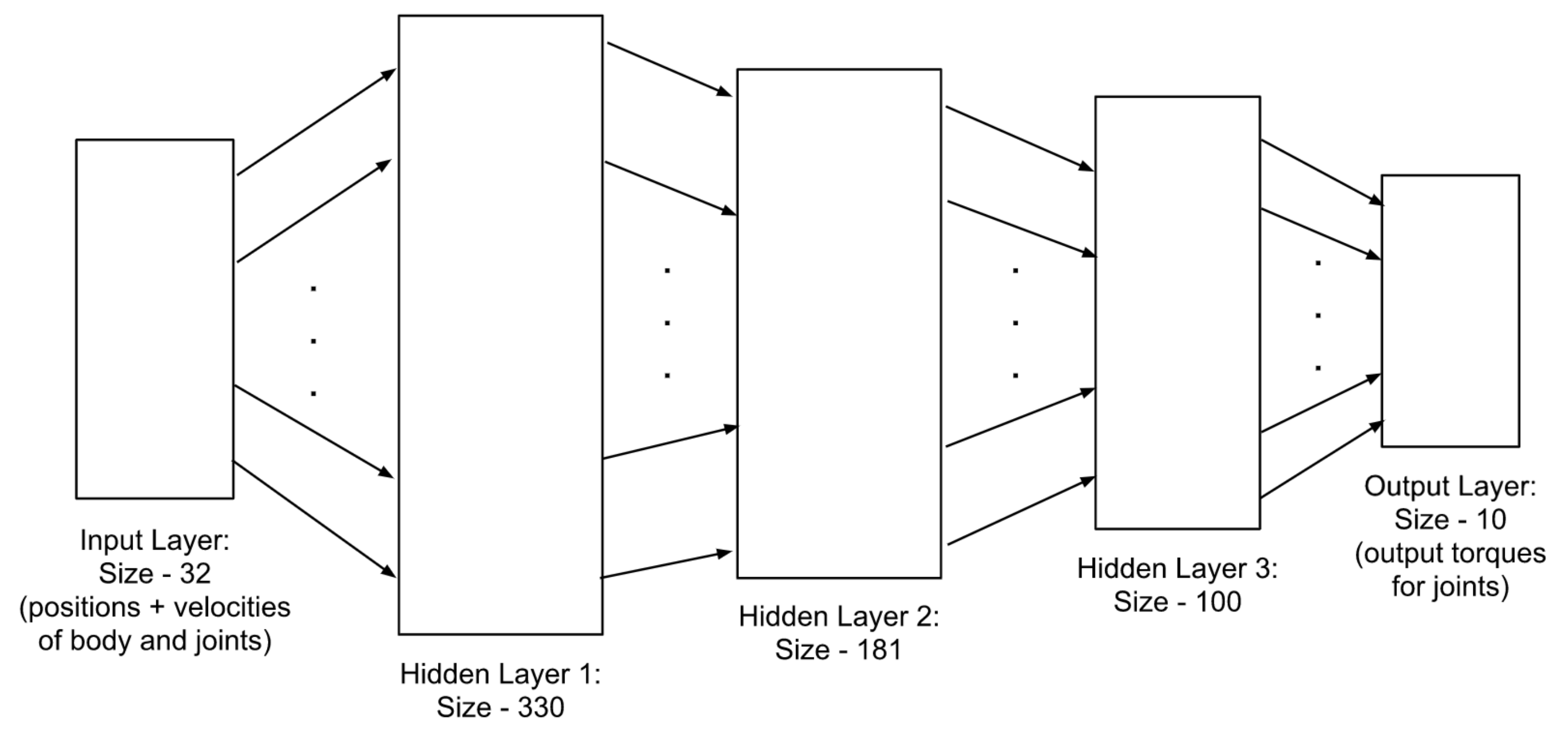

Figure 2: Neural network architecture for policy and value function in DRL framework. Inputs are body and joint positions and velocities. Outputs are torques for joints to apply.

Methods

- Created robot models in Mujoco (see Figure 3).
- Ported Mujoco models to OpenAI's Gym toolkit.
- For DRL algorithm, we used an implementation of Proximal Policy Optimization (PPO) with a generalized advantage estimation.
- Walking controller is saved as a neural network, representing the learned policy (see Figure 2 for network architecture).

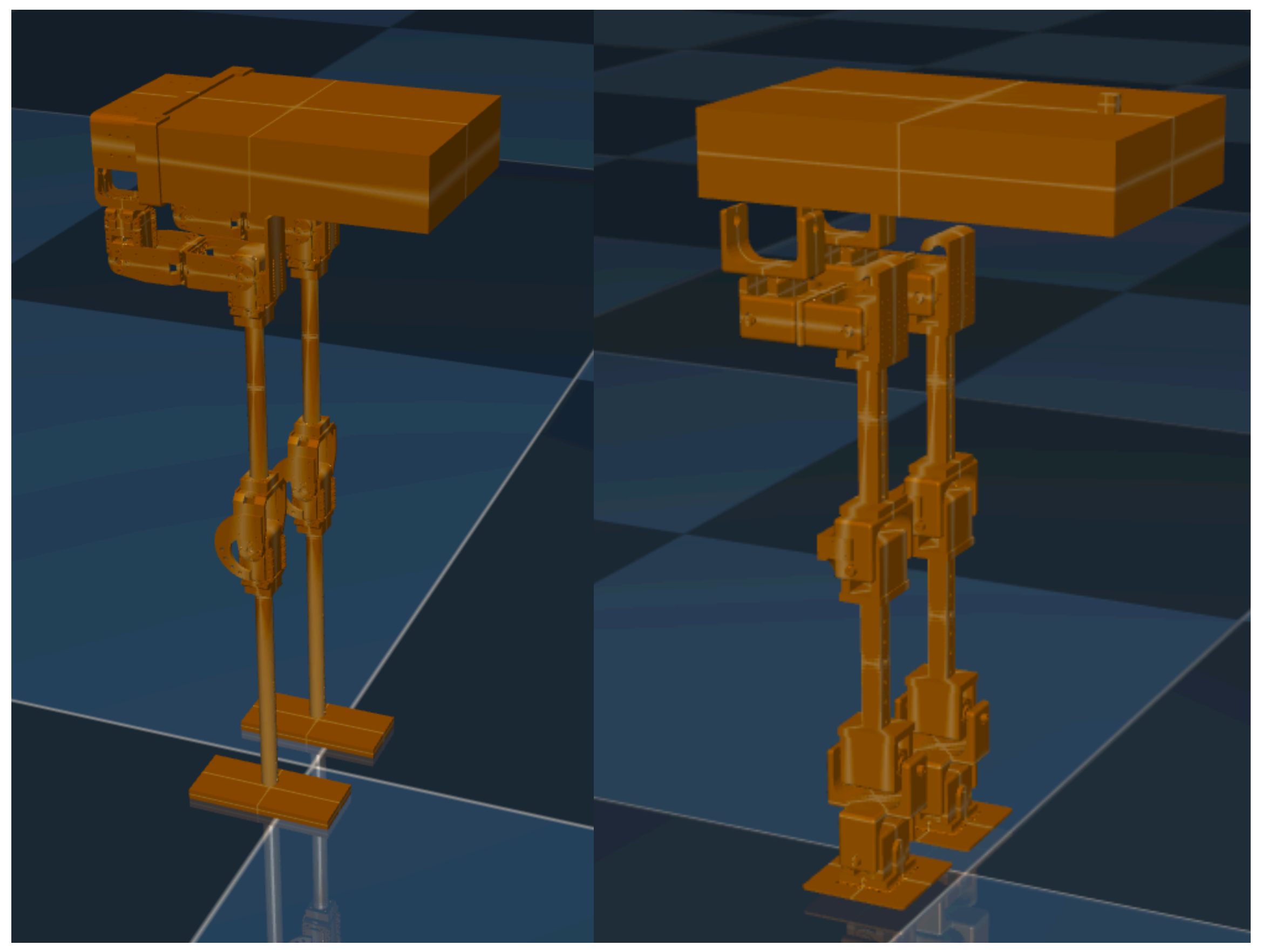

Figure 3: Two versions of simulated robot in Mujoco. Version 1 (left) is based off the original prototype. Version 2 (right) is an updated model with an extra degree of freedom at the foot.

- Models were trained on an eight-core processor with GPU acceleration for about 30,000-50,000 episodes. Training takes approximately 5-10 hours.
- Reward function is as follows:

$$R = \overrightarrow{v_x} + w_0 - w_1 \sum (\vec{\tau})^2 - w_2 \Delta \overrightarrow{f_z} - w_3 \sum (\Delta \vec{\tau})^2$$

  - R is reward at each simulation step.
  - $\overrightarrow{v}$ is the velocity at the C.O.M.
  - w is a vector of weights.
  - $\vec{\tau}$ is the action that the model took.
  - $\vec{f}$ is the position of the model's foot.
- Different motor models were swapped during every new set of training episodes. The outputted torques and velocities of these motors were logged in order to select optimal hardware for a future prototype.

Results

- Robot model was able to walk consistently during training for a maximum of about 7 minutes before falling.
- The DRL workflow allows us to easily swap motor parameters and adjust the walking gait. We are able to find several brushless motors that fit the parameters of the best walk cycle.

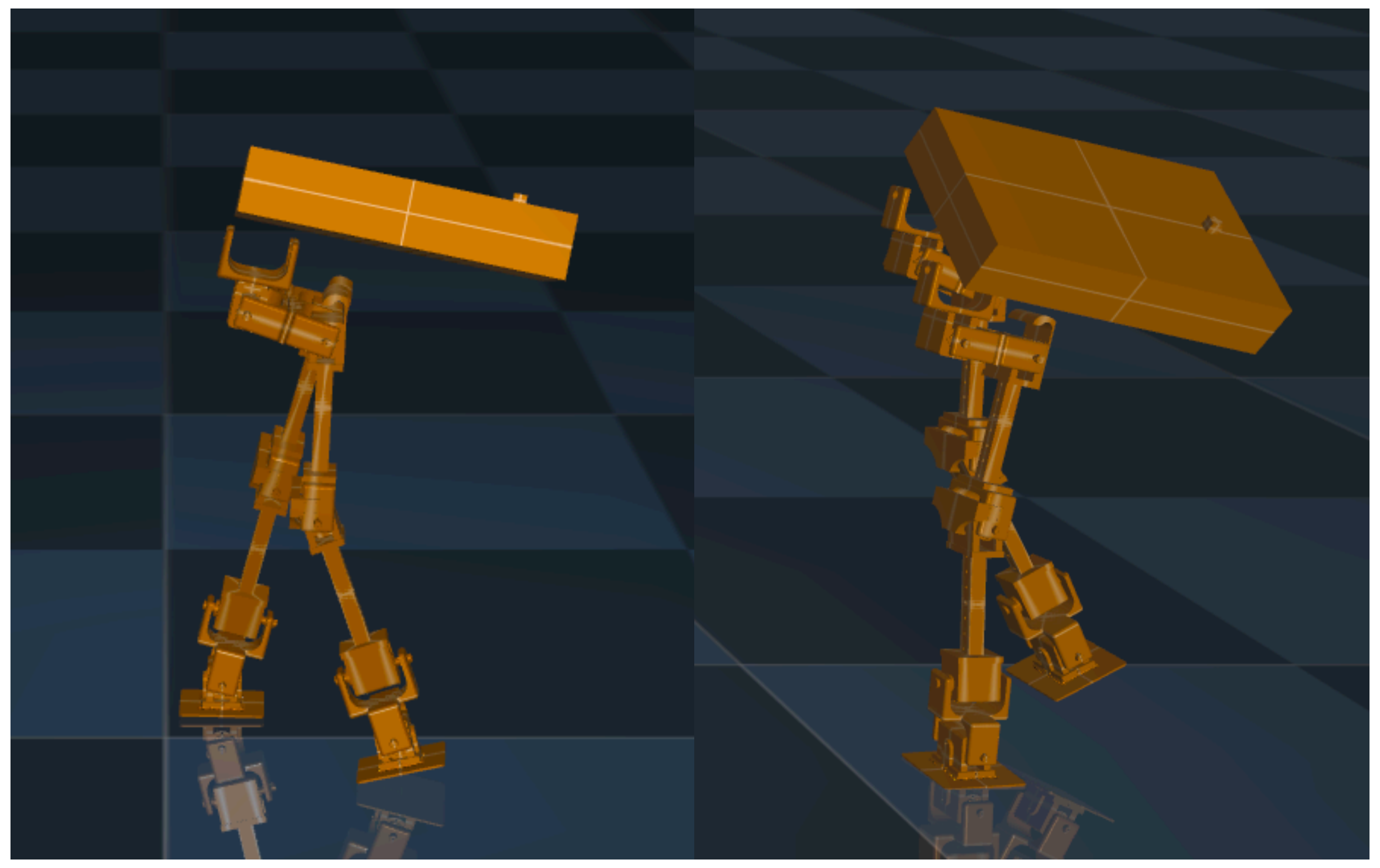

Figure 4: FlamingoBot model Version 2 walking.

- After saving and restoring the DRL policy, the model was able to walk for about 2000 simulation steps, each 0.005 seconds long.

Videos

- FlamingoBot V1 Walk
- FlamingoBot V2 Walk
- FlamingoBot V2 Standing After Walk

Future Work

- We plan to make our models and policies more robust by implementing dynamic randomization and adding noise to observations in order to account for real world unpredictability and fluctuations.
- We are in the process of generating a policy for stopping after walking for a while.
- We will eventually train policies on a real-life robot once we have an updated prototype.