THE ART OF HUMAN MOVEMENT

Team: Haodong Ma (dryice@stanford.edu), Renke Cai (renke.cai@stanford.edu), Klee Tang (kleetang@stanford.edu)
Advisor: Łukasz Kidziński (lukasz.kidzinski@stanford.edu)

Overview

The human musculoskeletal system is an organ system that gives humans the ability to move using their muscular and skeletal systems. This system provides form, support, stability, and movement of the human body. Based on a framework supplied by Stanford Neuromuscular Biomechanics Laboratory (NIPS 2017: Learning to Run [1]), our goal is to utilize reinforcement learning techniques to study and develop a motion controller to enable a physiologically-based human model to navigate a complex obstacle course as quickly as possible. We experimented with a list of state-of-art RL algorithms such as DDPG, PPO, and Continuous DQN. Due to the nature of the high-dimensional and continuous state and action space, we discovered that training is computationally intensive, and require long training cycles. It was challenging to debug and fine tune parameters. After some state space transformation, we are able to successfully train a walkable model, and also found that DDPG is the most promising and produce better results given the slowness of the simulator.

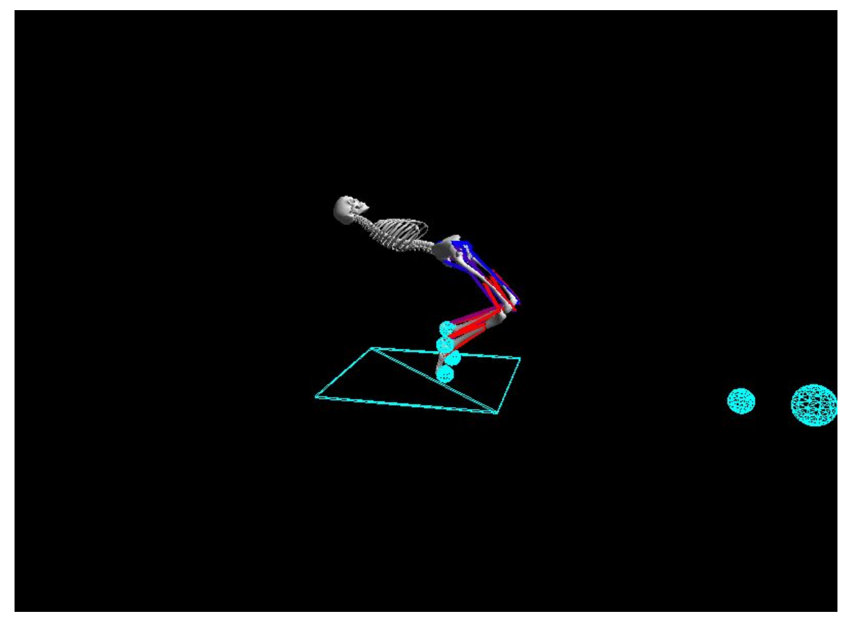


Figure 1: reward function using vx and vy

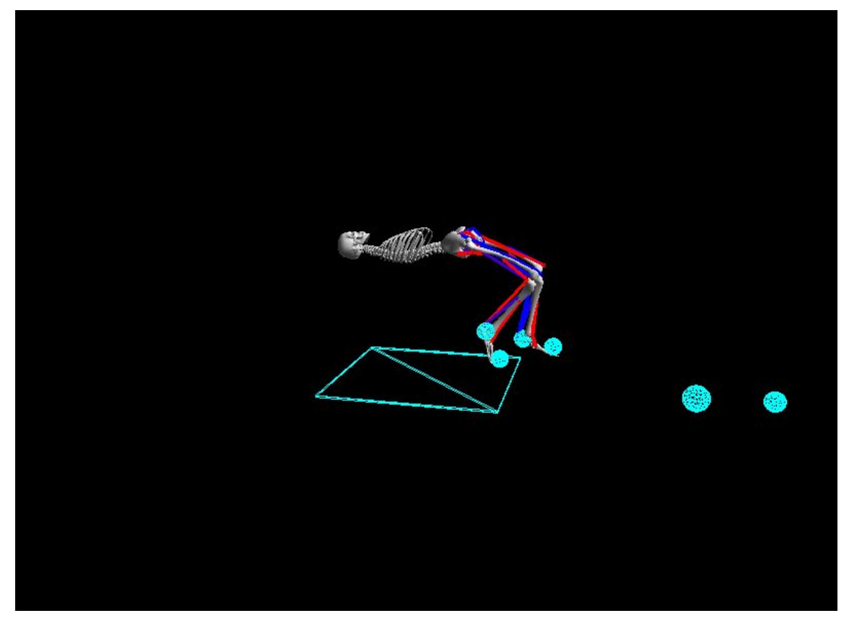


Figure 2: reward function using only vx

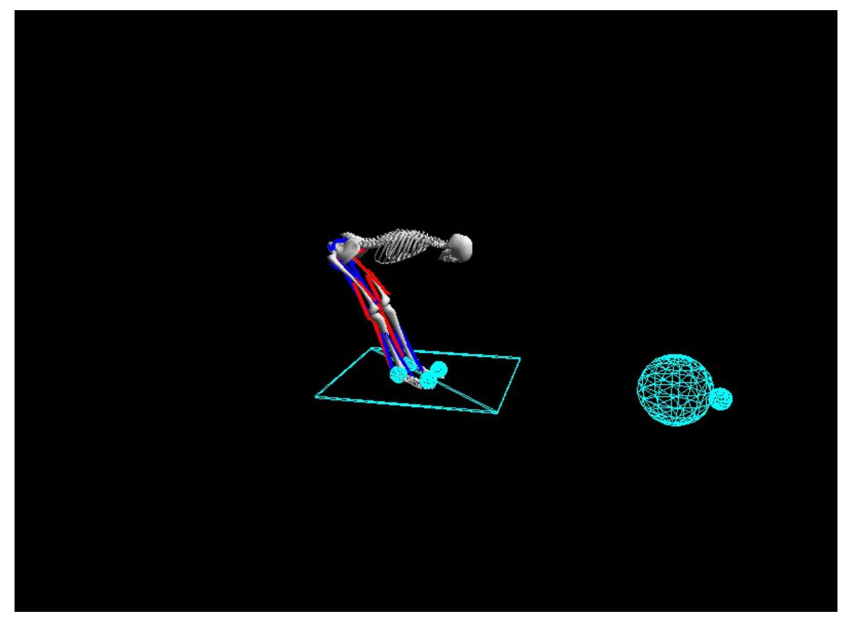


Figure 3: reward function forcing head position

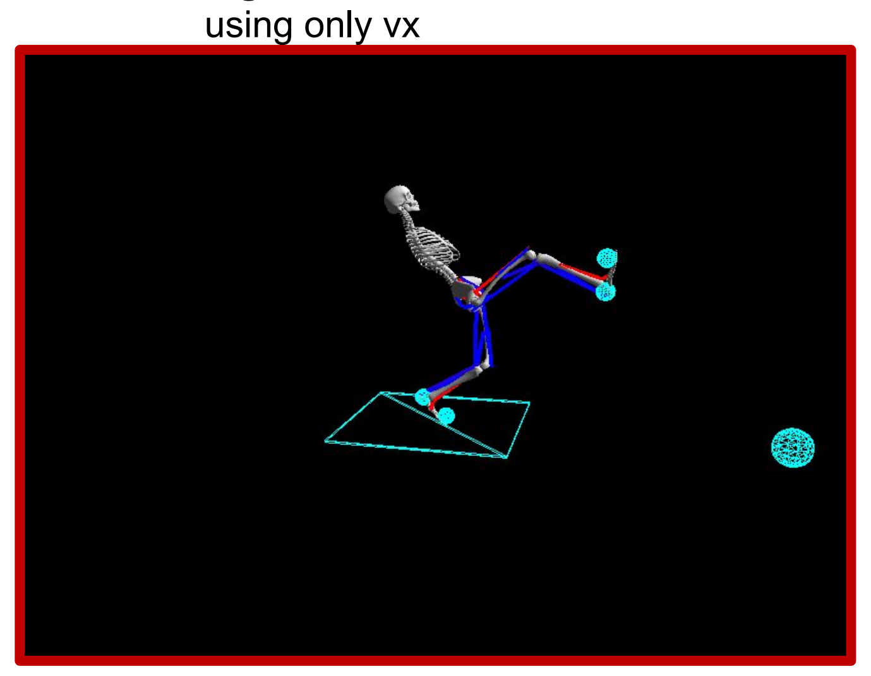


Figure 4: reward function encourage talus distance

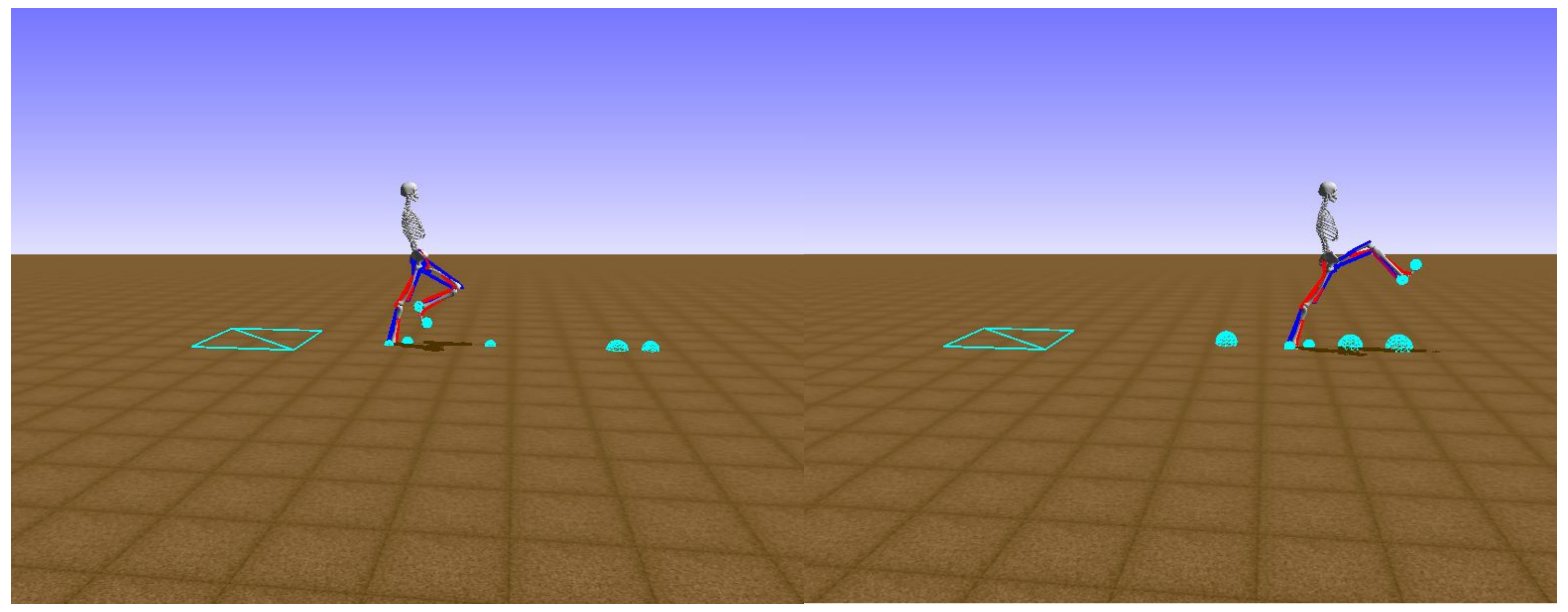


Figure 5: walking model

Figure 6: jump over obstacles

Future Plans

Continue focusing on DDPG, and dive deeper to fine-tune its parameters. Since RL is computationally intensive, we need to find better ways to distribute DDPG optimization over cheaper AWS EC2 instances and reduce the training time required. We would also like to explore ways to retrain models without having to start from scratch each time. We aim to try out different options with replay buffer to improve learning speed, and understand the impact on the learning speed related to exploration noise

Models

- **Deep Deterministic Policy Gradient (DDPG [3])** is a off-policy method and it consists of actor and critic neural networks. Critic is trained using Bellman equation and off-policy data and the actor is trained to maximize the critic's estimated Q-values through back propagation.
- **Proximal Policy Optimization (PPO [4])** is an on-policy method and a variant of TRPO that adds D_{KL} terms to training loss function, and train the policy using gradient descent like normal neural network.
- **Continuous Deep Q-learning [6]** is a continuous-action-space variant of the Q-learning algorithm and the NAF representation allows us to apply Q-learning with experience replay to continuous tasks, and substantially improves performance on robotic control tasks.

Feature Engineering

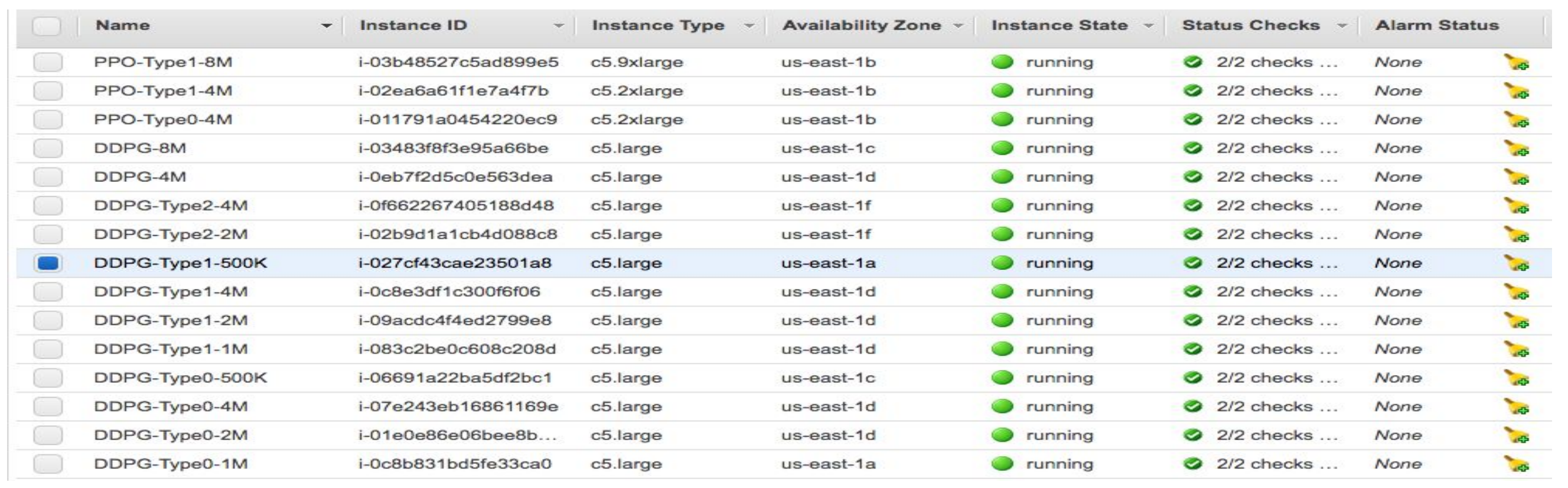
Using a simulator provided by OpenSim [2], we have an agent that is a musculoskeletal model controllable by a set of actions to actuate its muscles. At each step, a set of observations are produced by a physics-based simulation environment. Each observation contains positions and velocities of joints and bones of the lower body. The goal is to move the agent forward as fast as it can while minimizing overstretching ligaments and avoiding obstacles. The episode ends when the pelvis height is below 0.65m, or 1000 steps are reached.

The initial set of observations are 41-dimension vectors, after some trial and error, additional features such as velocity for other body parts (head, torso, etc.) are added, absolute positions and velocities are converted to the relative values based on the pelvis, and we also added information regarding 3 nearest obstacles (instead of the one right in front), in the end, our state is represented by 57-dimensional vectors in continuous space with 18-dimensional vectors for the action space.

Results

We ran various configurations on a collection of AWS EC2 instances, and were able to successfully train a model capable of walking and overcome obstacles (see figure 5, 6), after applying state transformation, DDPG agent produced the most promising results. (tested with 500 steps)

training steps	DDPG w/o state transformation	DDPG with state transformation	PPO	Continuous DQN
500,000	1.24m, 158 steps	0.37m, 500 steps	<0m, 104 steps	0.421m, 74 steps
1,000,000	2.57m, 350 steps	2.84m, 468 steps	<0m, 125 steps	0.561m, 112 steps
2,000,000	1.95m, 182 steps	4.93m, 370 steps	0.99m, 121 steps	not attempted
4,000,000	0.32m, 63 steps	7.34m, 405 steps	1.29m, 127 steps	not attempted
8,000,000	1.48m, 160 steps	not attempted	1.07m, 123 steps	not attempted



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
PPO-Type1-8M	i-03b48527c5ad899e5	c5.9xlarge	us-east-1b	running	2/2 checks ...	None
PPO-Type1-4M	i-02ea6a61f1e7a4f7b	c5.2xlarge	us-east-1b	running	2/2 checks ...	None
PPO-Type0-4M	i-011791a0454220ec9	c5.2xlarge	us-east-1b	running	2/2 checks ...	None
DDPG-8M	i-03483f8f3e95a66be	c5.large	us-east-1c	running	2/2 checks ...	None
DDPG-4M	i-0eb7f2d5c0e563dea	c5.large	us-east-1d	running	2/2 checks ...	None
DDPG-Type2-4M	i-0f662267405188d48	c5.large	us-east-1f	running	2/2 checks ...	None
DDPG-Type2-2M	i-02b9d1a1cb4d088c8	c5.large	us-east-1f	running	2/2 checks ...	None
DDPG-Type1-500K	i-027cf43cae23501a8	c5.large	us-east-1a	running	2/2 checks ...	None
DDPG-Type1-4M	i-0c8e3df1c300f6f06	c5.large	us-east-1d	running	2/2 checks ...	None
DDPG-Type1-2M	i-09acdc4f4ed2799e8	c5.large	us-east-1d	running	2/2 checks ...	None
DDPG-Type1-1M	i-083c2be0c608c208d	c5.large	us-east-1d	running	2/2 checks ...	None
DDPG-Type0-500K	i-06691a22ba5df2bc1	c5.large	us-east-1c	running	2/2 checks ...	None
DDPG-Type0-4M	i-07e243eb16861169e	c5.large	us-east-1d	running	2/2 checks ...	None
DDPG-Type0-2M	i-01e0e86e06bee8b...	c5.large	us-east-1d	running	2/2 checks ...	None
DDPG-Type0-1M	i-0c8b831bd5fe33ca0	c5.large	us-east-1a	running	2/2 checks ...	None

Discussion

Having a complete and observable state for the agent is critical, and we also discovered that RL algorithms are extremely sensitive to reward functions, designing a good and continuous reward function could significantly reduce learning time. Some early tweaks of reward functions have produced some interesting results (see figures 1-4). Overall, RL project in high dimensional space is fascinating and challenging at the same time, for us, we underestimated the demand for training time and compute power required (AWS EC2 cost). Given the set of possible paths and parameters to alter, it is essential to have a cost and time efficient strategy to debug RL problems. We would recommend the following

1) perform state data analysis independently without relying on RL algorithms which is more time consuming
2) start with a simple reward function before employing more complex versions
3) validate with known and well supported RL algorithm first, and preferably using simpler setup with small dimensions
4) start the training process gradually with easy conditions, and then move onto conditions involving more obstacles
5) bias towards distributed algorithms that lead to faster training time
6) minimize the number of changes during each test cycle
7) invest in a better ML workflow system to manage all the combinations of code, config, train results to have a more reproducible results

References

[1] Stanford Neuromuscular Biomechanics Laboratory NIPS 2017: Learning to Run https://www.crowdai.org/challenges/nips-2017-learning-to-run
[2] Reinforcement learning environments with musculoskeletal models 2017 GitHub https://github.com/stanfordnmbl/osim-rl
[3] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra 2016. Continuous control with deep reinforcement learning arXiv:1509.02971
[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov 2017. Proximal Policy Optimization Algorithms arXiv:1707.06347
[5] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, David Silver 2017. Emergence of Locomotion Behaviours in Rich Environments arXiv:1707.02286
[6] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, Sergey Levine, 2016. Continuous Deep Q-Learning with Model-based Acceleration arXiv:1603.00748
[7] Mikhail Pavlov, Sergey Kolesnikov, Sergey M. Plis, 2017. Run, skeleton, run: skeletal model in a physics-based simulation arXiv:1711.06922
[8] Dhariwal, Prafulla and Hesse, Christopher and Plappert, Matthias and Radford, Alec and Schulman, John and Sidor, Szymon and Wu, 2017 OpenAI Baselines GitHub https://github.com/openai/baselines