Tiny Steps Towards Applying Reinforcement Learning to Achieve Walking Gaits on a Constrained Quadruped Robot

Harvard Edge Computing Lab

Tiny Robots Group

- Dr. Sabrina Neuman
- Brian Plancher
- Mark Mazumder
- Colby Banbury
- Shvetank Prakash
- Christopher Zhu
- Jason Jabbour

VLSI-Arch Undergraduate Research Presentations

Overview

Problem Description

Applying Reinforcement Learning to Achieve Walking Gaits

Achieving Natural-Looking Walking Gaits through Imitation Learning

Problem Description

Goal

Compensate for cheap hardware with intelligence to achieve robust locomotion

Problem Description

Goal

Compensate for cheap hardware with intelligence to achieve robust locomotion

Intermediate Goal

Train a sensor, actuator, and power impoverished robot to walk

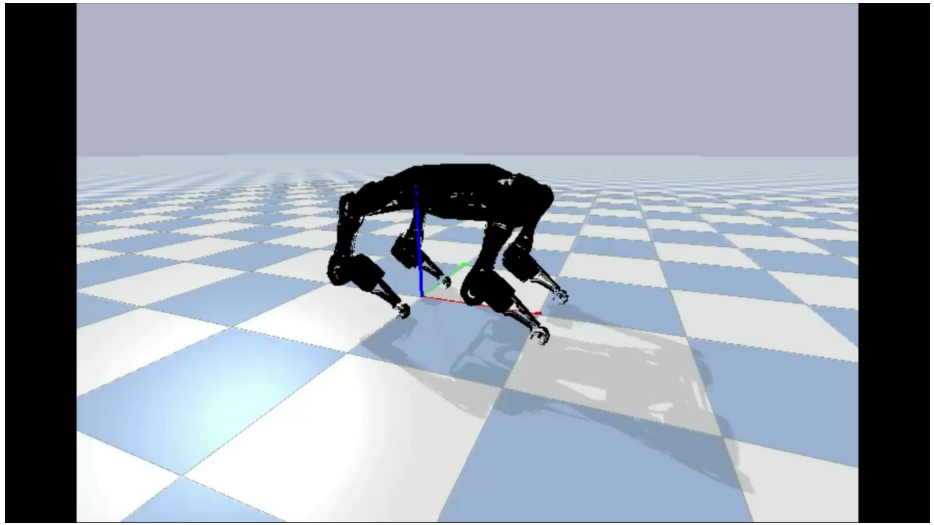
Choosing a Robot Platform

Petoi's Bittle

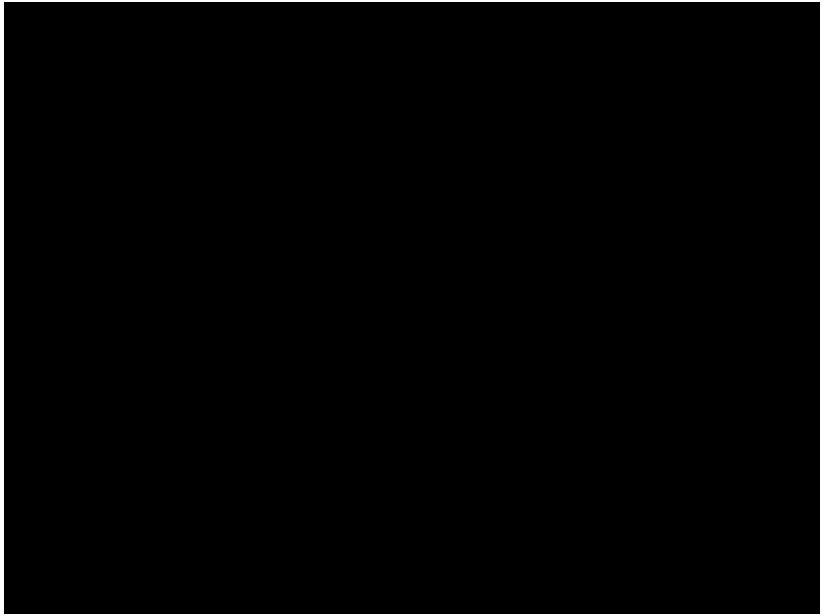
~$300

2 DoF

Problem Description

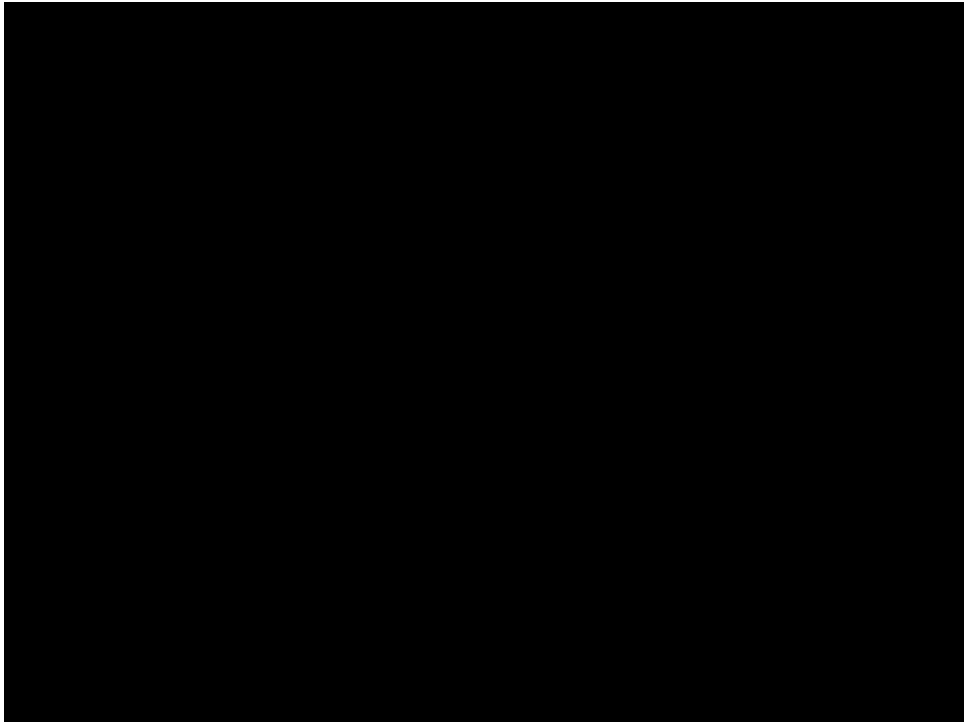


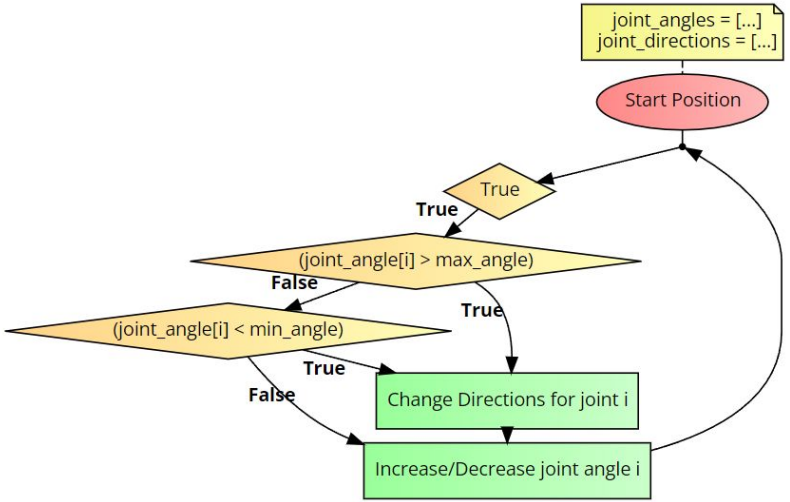
8 DoF



18 DoF

Manual Control





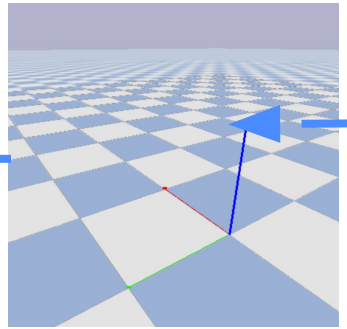
RL Overview

Agent

State
(S_t)

Reward
(R_t)

Action
(A_t)



Environment

Tools:

- Pybullet
- OpenAI Gym
- Stable Baselines
- URDF File

Initial Policies

Action Space = 8 Joint Angles
Observation Space = Position, Orientation, Linear Velocity, Angular Velocity

Tuning

$$\text{Reward Function} = W_1(\text{X Position}) + W_2(\text{X Velocity}) - W_3(|\text{Z Position} - \text{Bittle Height}|) - W_4(|\text{Z Velocity}|) - W_5(|\text{Roll}|) - W_6(|\text{Pitch}|)$$

Tuning

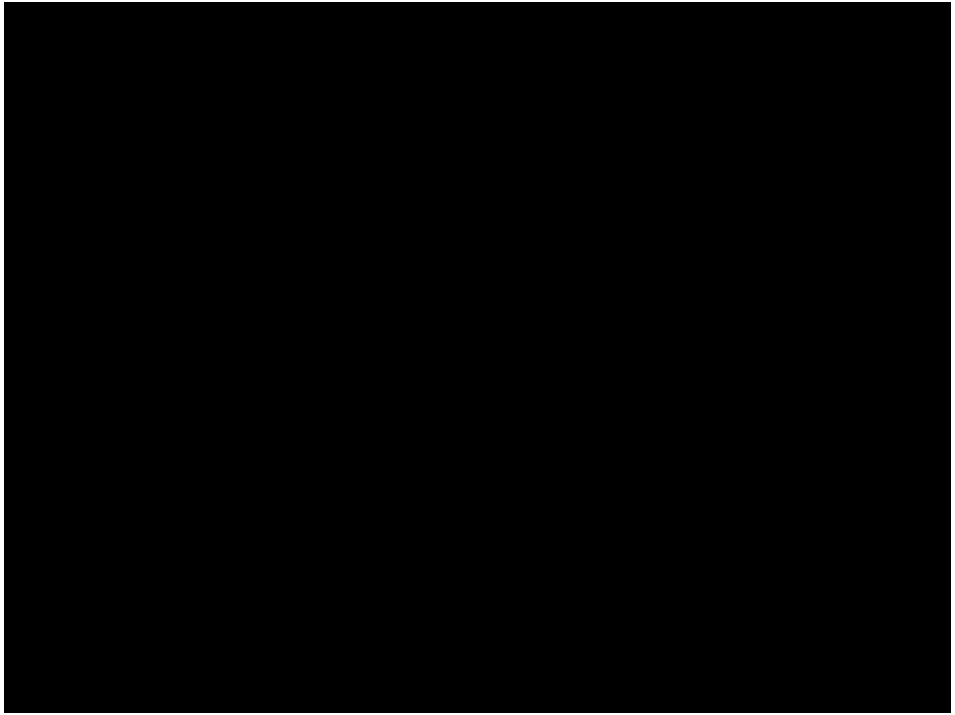
Reward Function = ~~W_1(X Position)~~ + W_2 (X Velocity) - ~~W_3(|Z Position - Bittle Height|) - W_4(|Z Velocity|) - W_5(|Roll|)~~ - W_6(|Pitch|)

Action Space = [-.1, .1] for all 8 Joint Angles

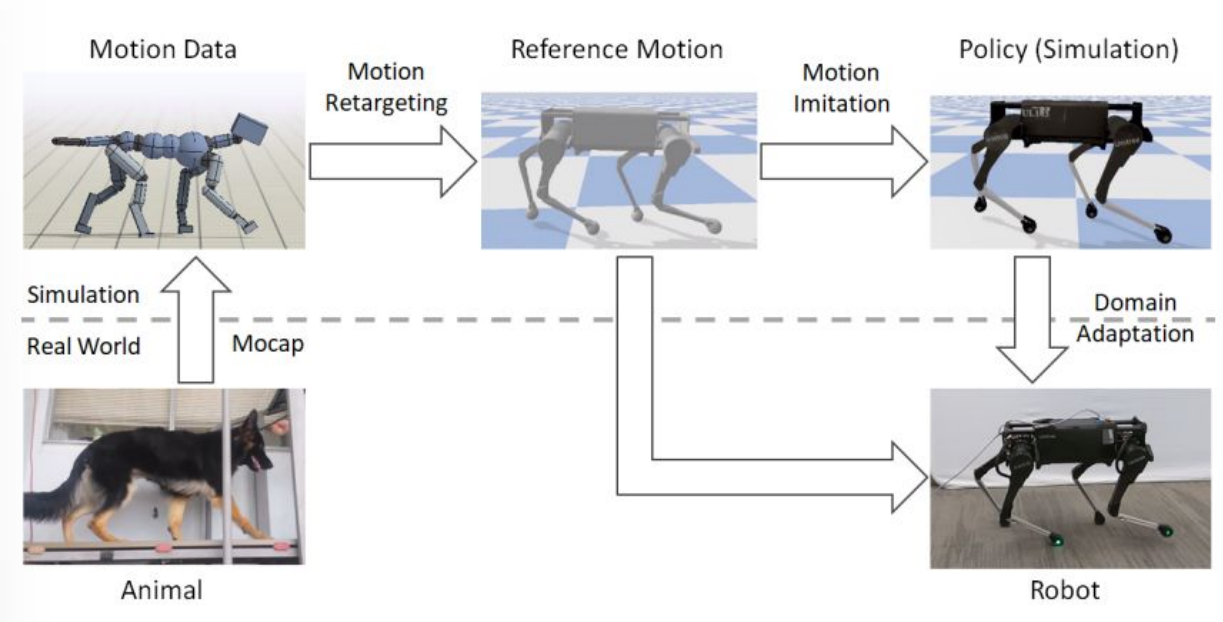
Tuning: Final Policy

```
if (X_Velocity > velocity) and self.is_upright() \
 and (Z_Position > (Torso_Position - Perturbation)):
    reward = .1
    if (X_Velcoity > velocity*2):
        reward = .2
else:
    reward = 0

if self.is_fallen():
    reward = -.1
```



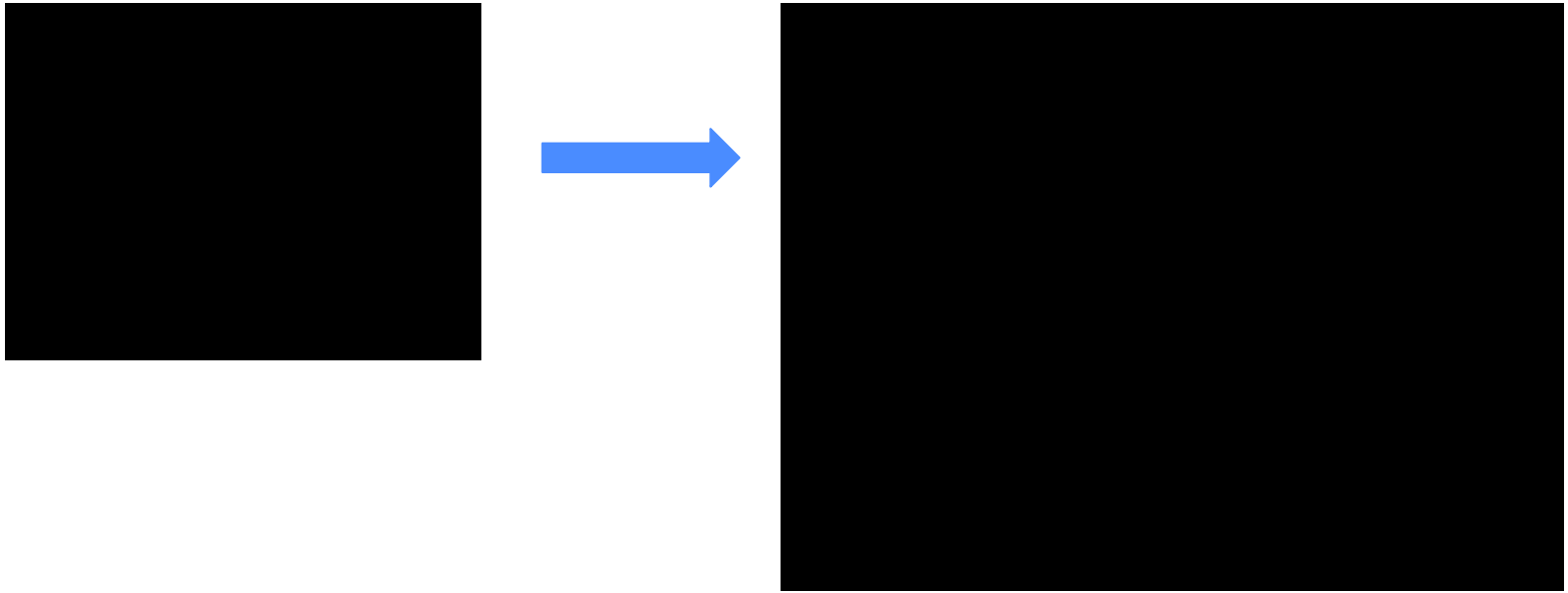
Imitation Learning



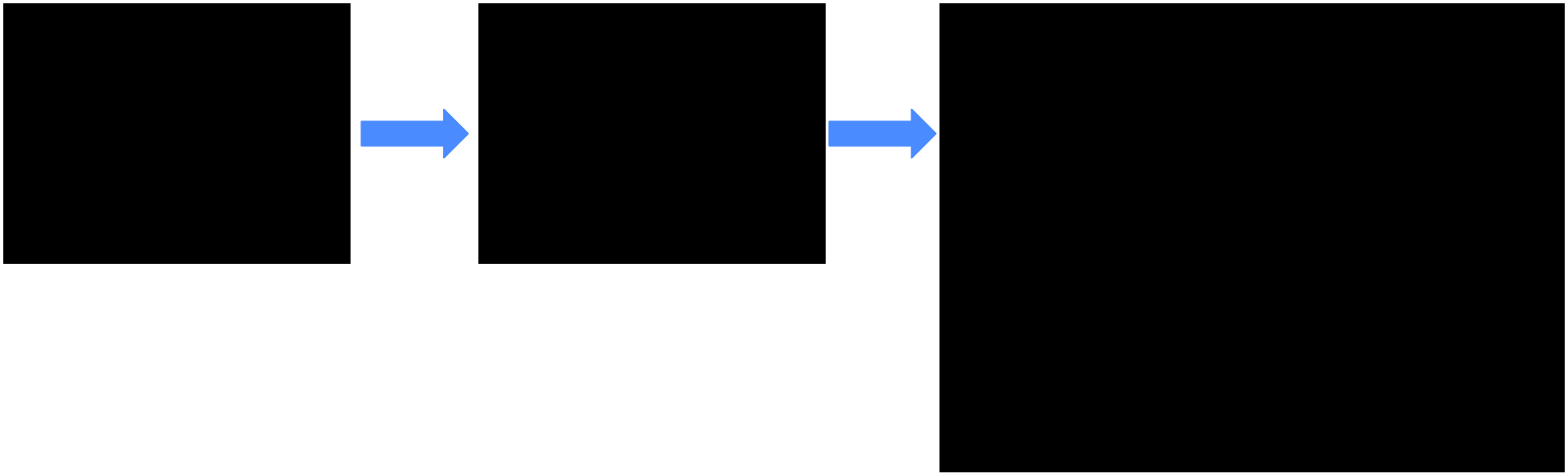
(Peng et al., 2020)

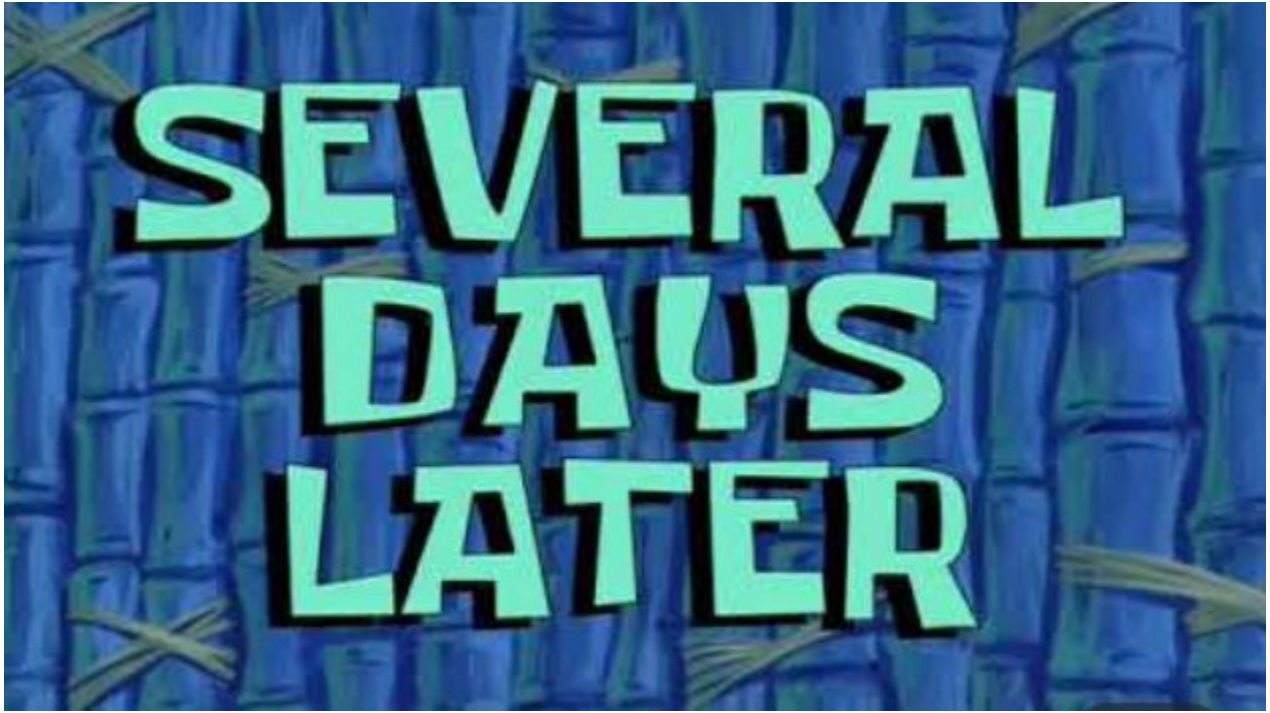
Imitation Learning: **Motion Capture**

Imitation Learning: **Motion Retargeting**



Imitation Learning: Using Reference Motion to Train Policy




SEVERAL
DAYS
LATER

Imitation Learning: **Final Imitation Learning Policy**

Future Work

Domain Adaptation

TinyML Shrink our Policy

Special Thanks to

Dr. Vijay Janapa Reddi, Dr. Sabrina Neuman, Brian Plancher, Mark Mazumder, and Colby Banbury