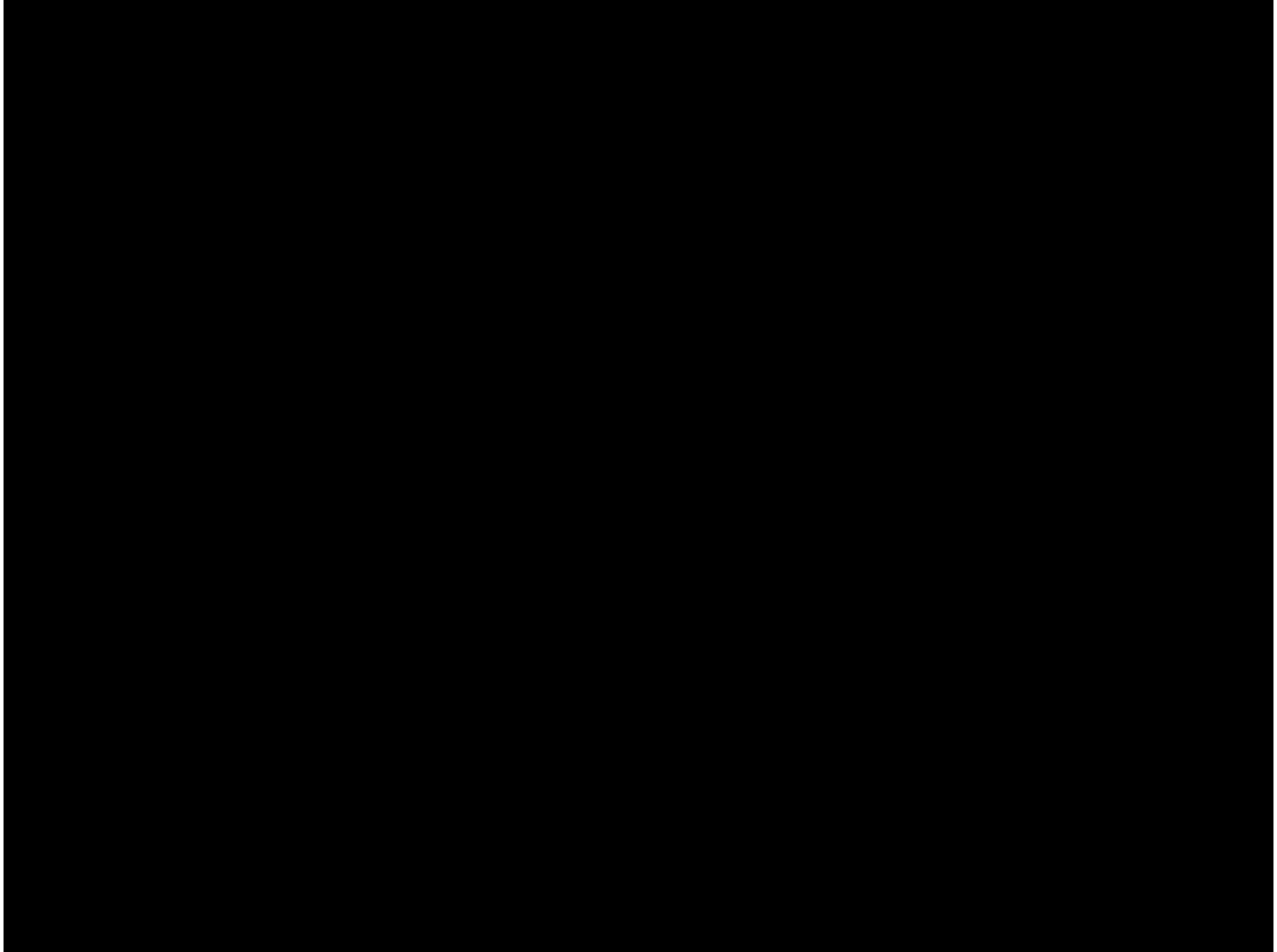
Omnidirectional Bipedal Walking in Cartesian Space through Reinforcement Learning and Optimized Quintic Splines

Marc Bestmann

Motivation

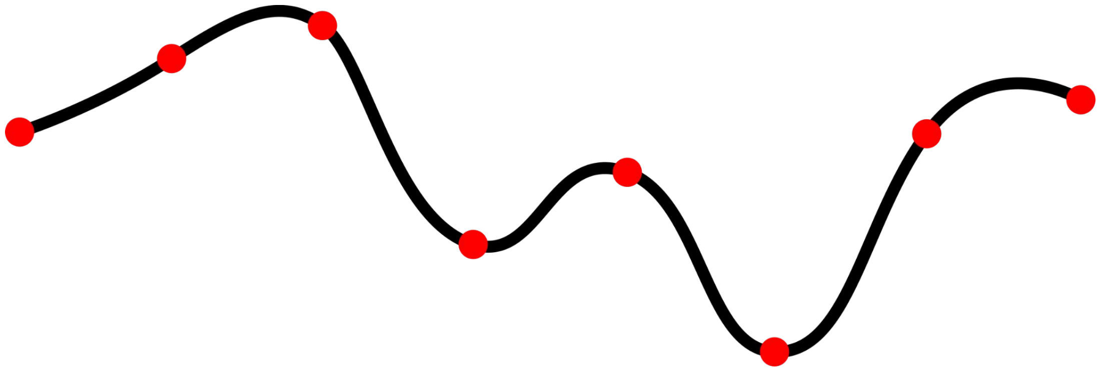
- Bipedal walking is important but difficult
- Many different approaches were tried in the past
- In recent years, deep RL became more popular
- Typically combined with reference motions from motion capture data
- I use a novel approach of generating these

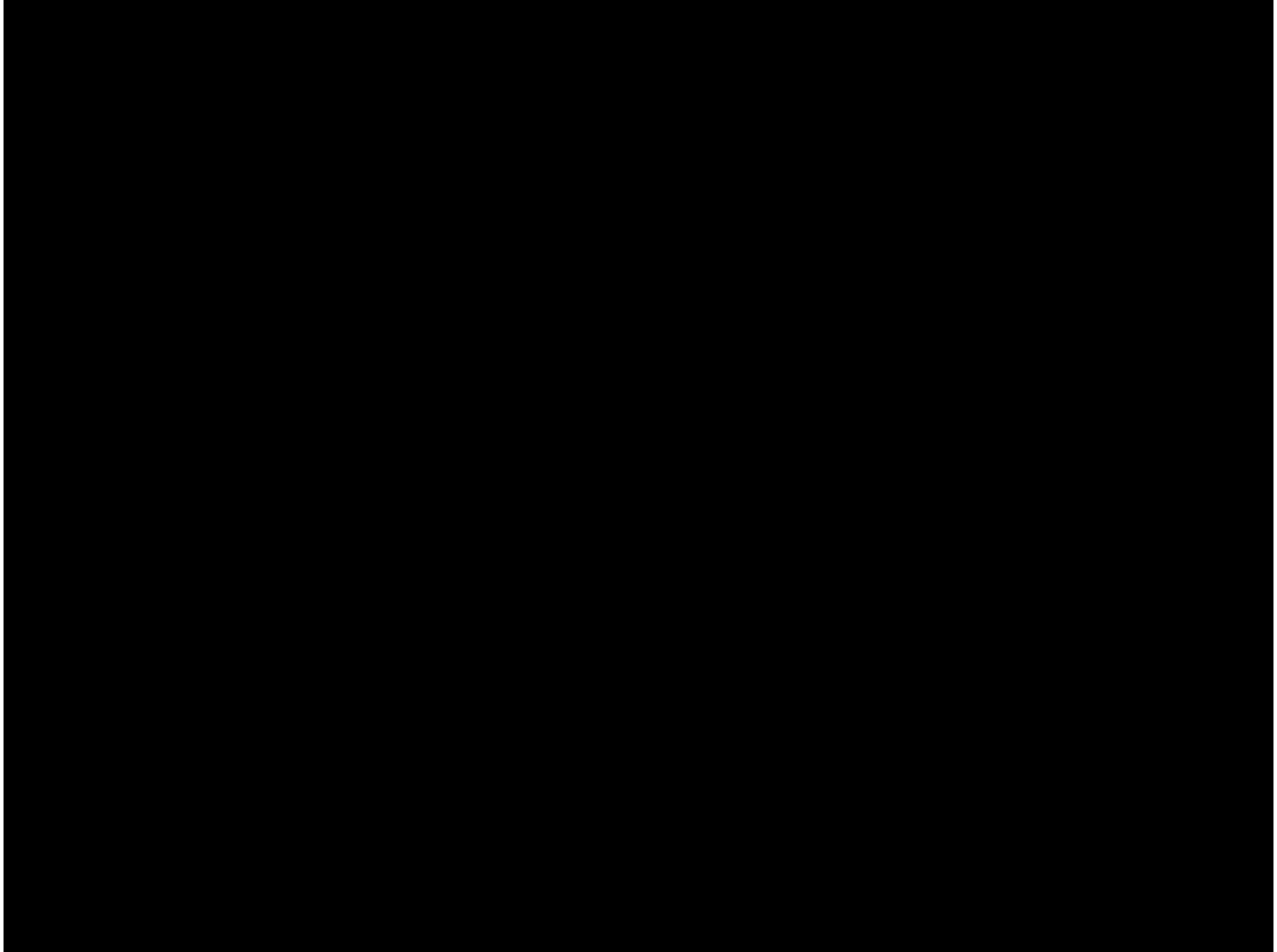


From Peng et al. “DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills”, 2018

Optimized Spline Reference Motion

- Define movement of feet in Cartesian space as quintic spline
- Dependent on walk velocity and a set of hyperparameters
- Hyperparameters are optimized with Multi-objective Tree-structured Parzen Estimator (MOTPE)
- IMU-based PD control for stabilization





Choice of Reference Motion

Motion Capture	Manual Keyframe Animation	Spline-based Engine
Only possible for human or animal like robots	Possible for any kind of robot	Possible for any kind of robot
Transfer to different kinematics needed	Designed for one specific platform	Adapts to different platforms
Reference not executable	Reference may be executable	Reference executable, provides baseline
Not optimal	Difficult to optimize	Can be optimized for the platform
One walk velocity	One walk velocity	Any walk velocity
Represents state	Represents state	Represents Action
Data recording needed	Manual creation needed	Programming needed

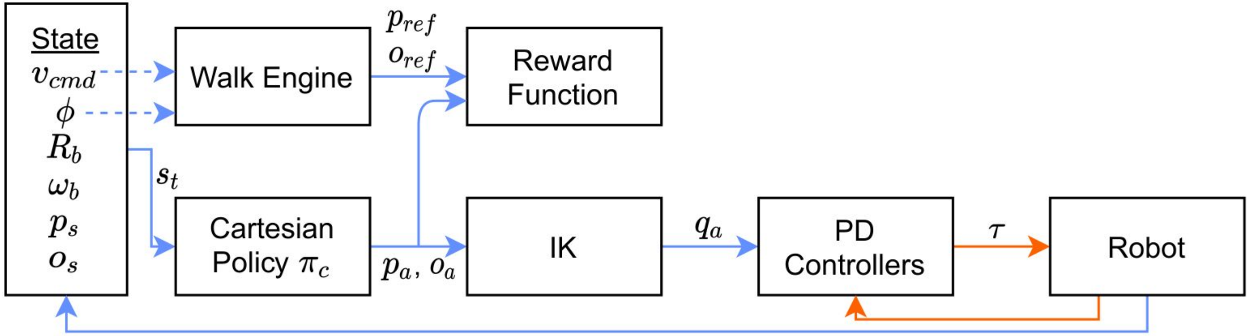
Design Choices

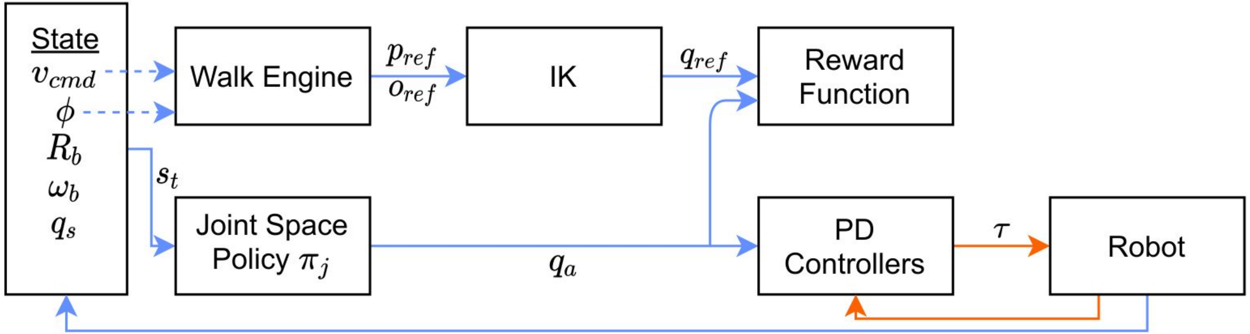
- Typical RL research uses standardized environments (OpenAi Gym)
- Then compares the different algorithms with each other
- Environment design choices were not investigated that much yet
- But have a high influence on the result
- Action and observation space
    - Cartesian / Joint space
    - Rotation representation
- Reward function
    - State or action based

Approach

- Synchronising with phase
- Reference only used for reward
- IK usage

$$r = \frac{1}{2} \cdot r_g + \frac{1}{2} \cdot r_i$$



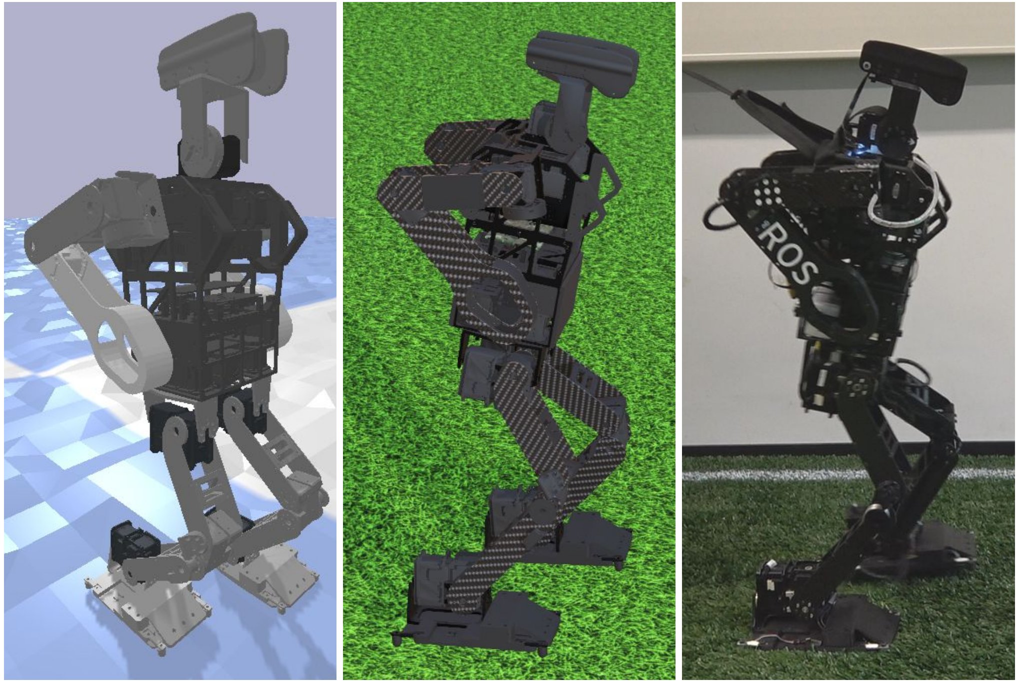


Approach

- Training by using PPO
  - 10 seconds time limit
  - Random state initialization
- Policy net
  - 2 layer each 64 fully connected neurons
  - tanh activation function
  - Fixed variance gaussian distribution
- Value function net
  - 2 layer each 64 fully connected neurons
  - tanh activation function
- PPO hyperparameter optimization using TPE

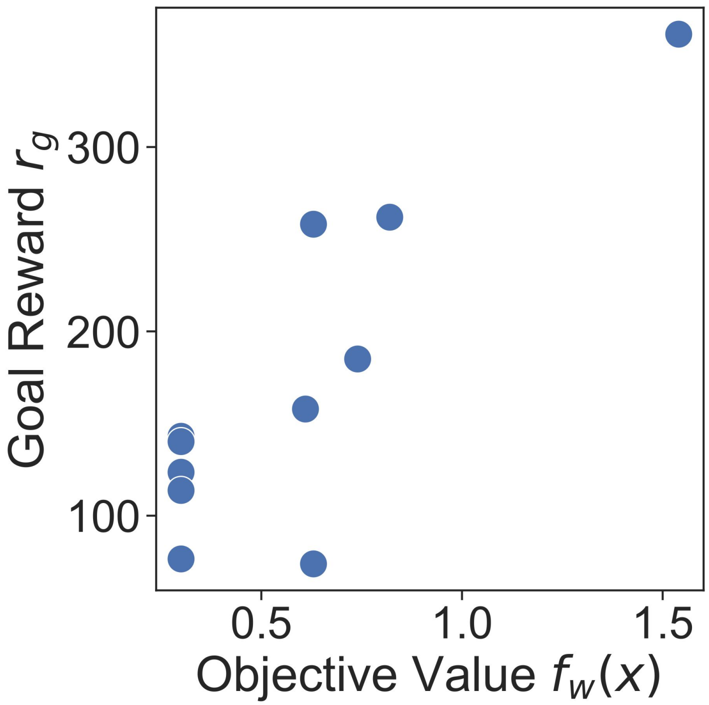
Platform

- Wolfgang-OP robot
- Learning done in either PyBullet or Webots
- Test with ROS stack in Webots or on real robot



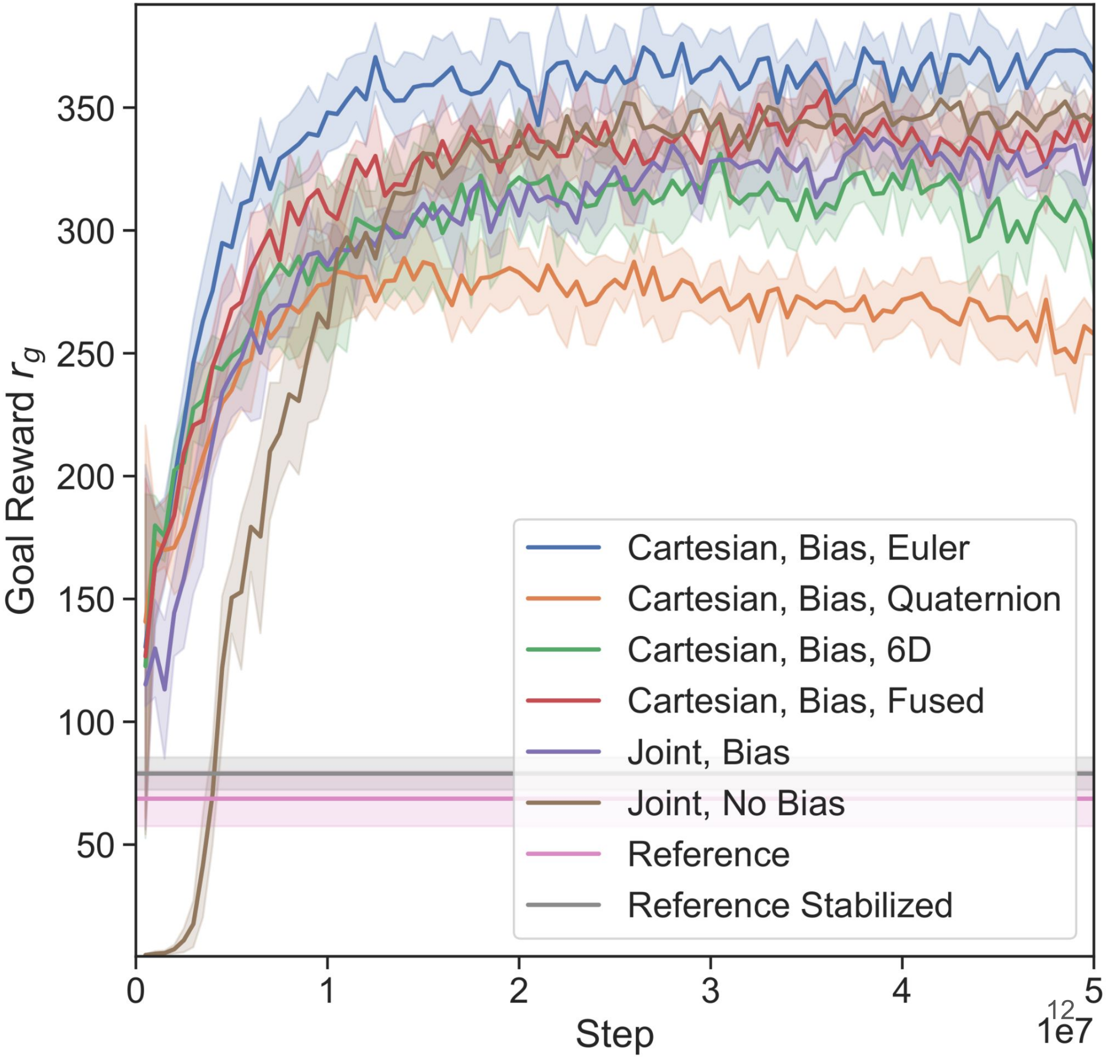
Reference Motion Quality

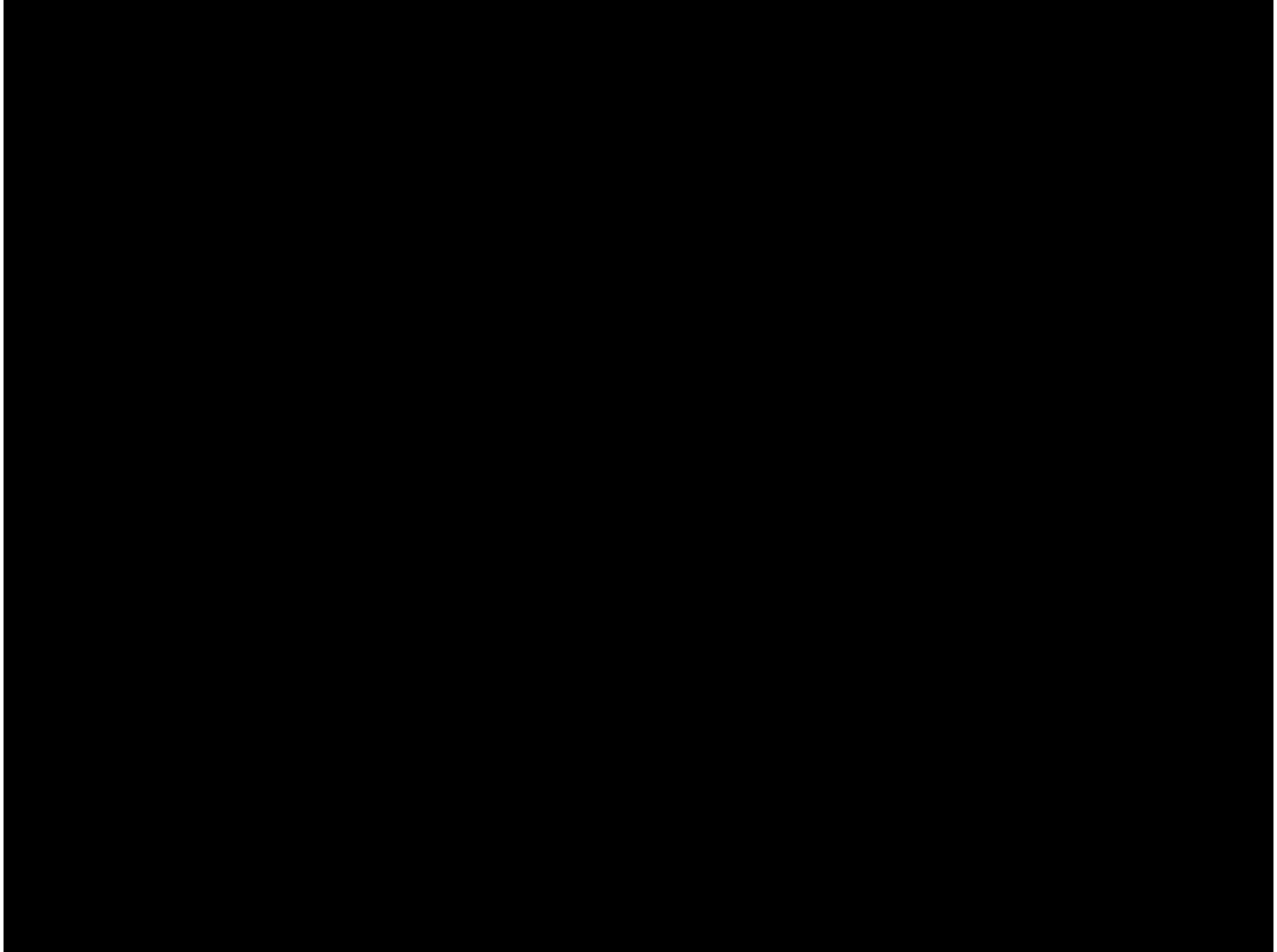
- Learn multiple times using different hyperparameter for reference motion
- Achieved reward is proportional to the quality of the reference motion
- Confirms hypothesis that optimized reference motion is desirable

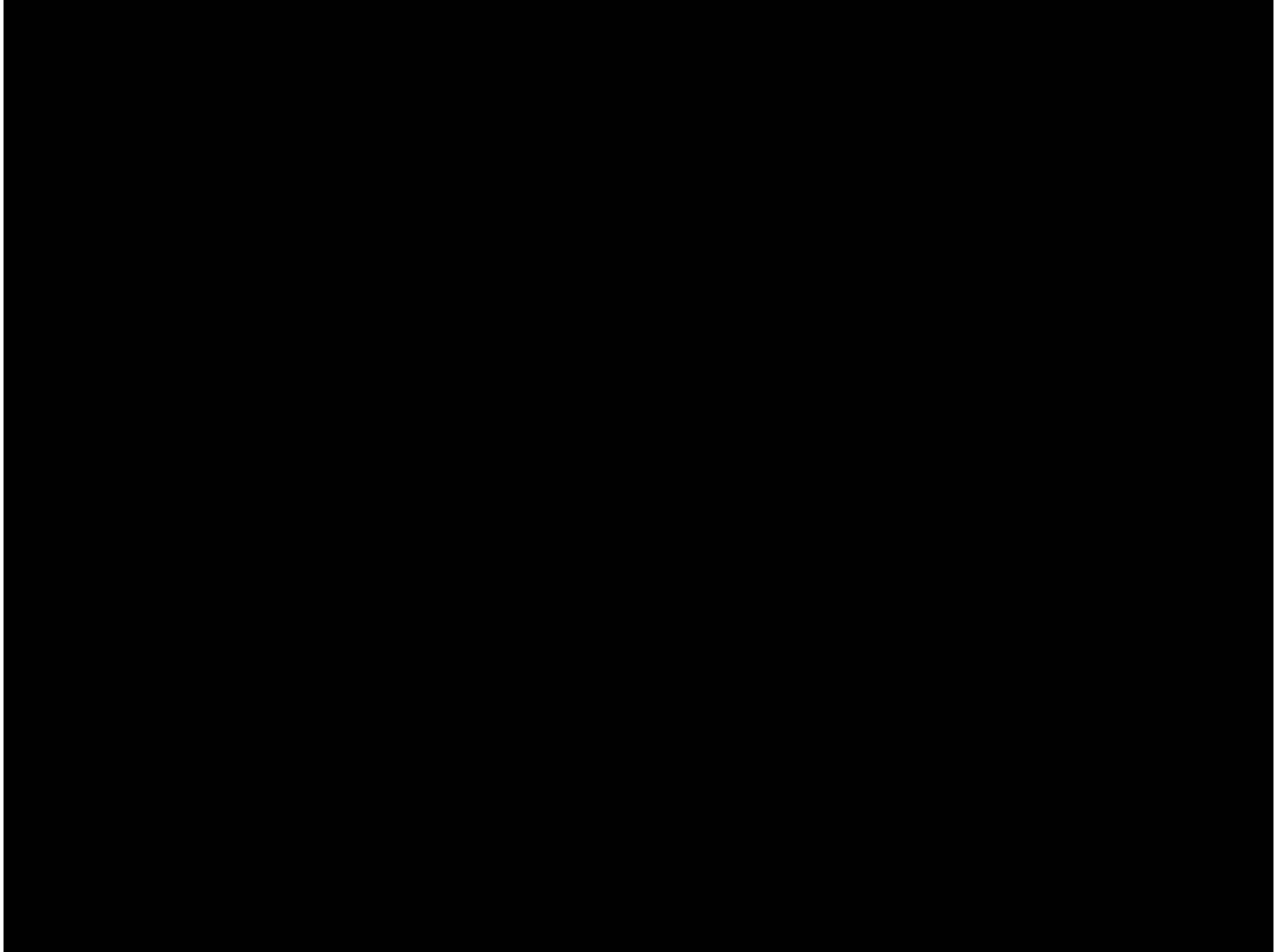


Policy Space

- Choice of rotation type matters
- Cartesian space learns faster and achieves higher rewards
- RL approach surpases performance of simple stabilization methods
- Bias reduces number of falls

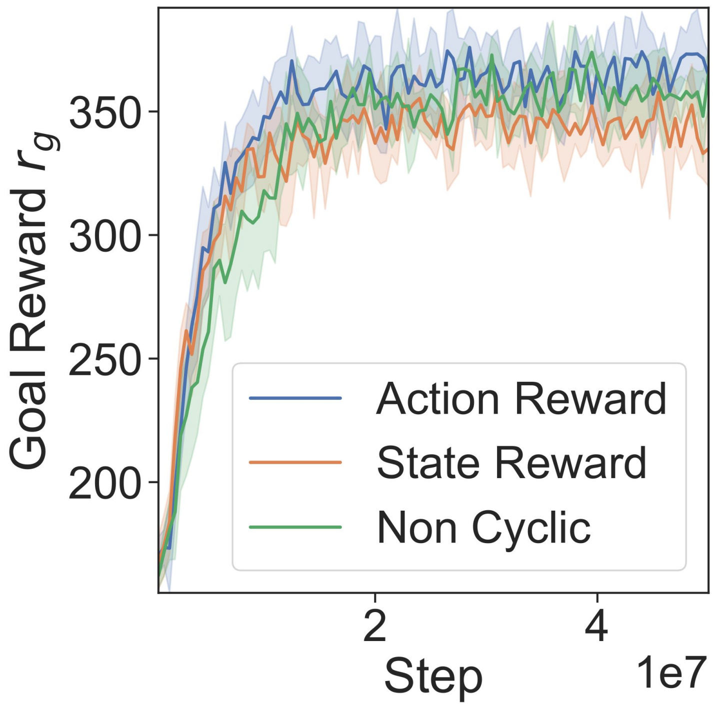






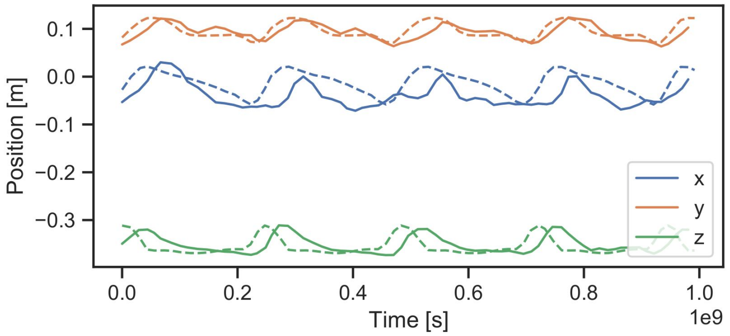
Reward

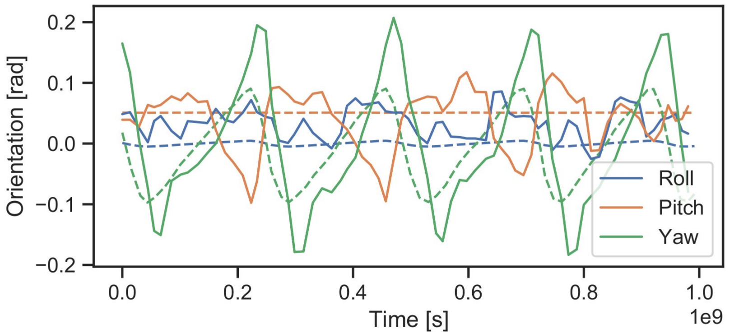
- Action based reward achieves higher reward
- Difference between cyclic and non cyclic phase is small



Exemplary Plots

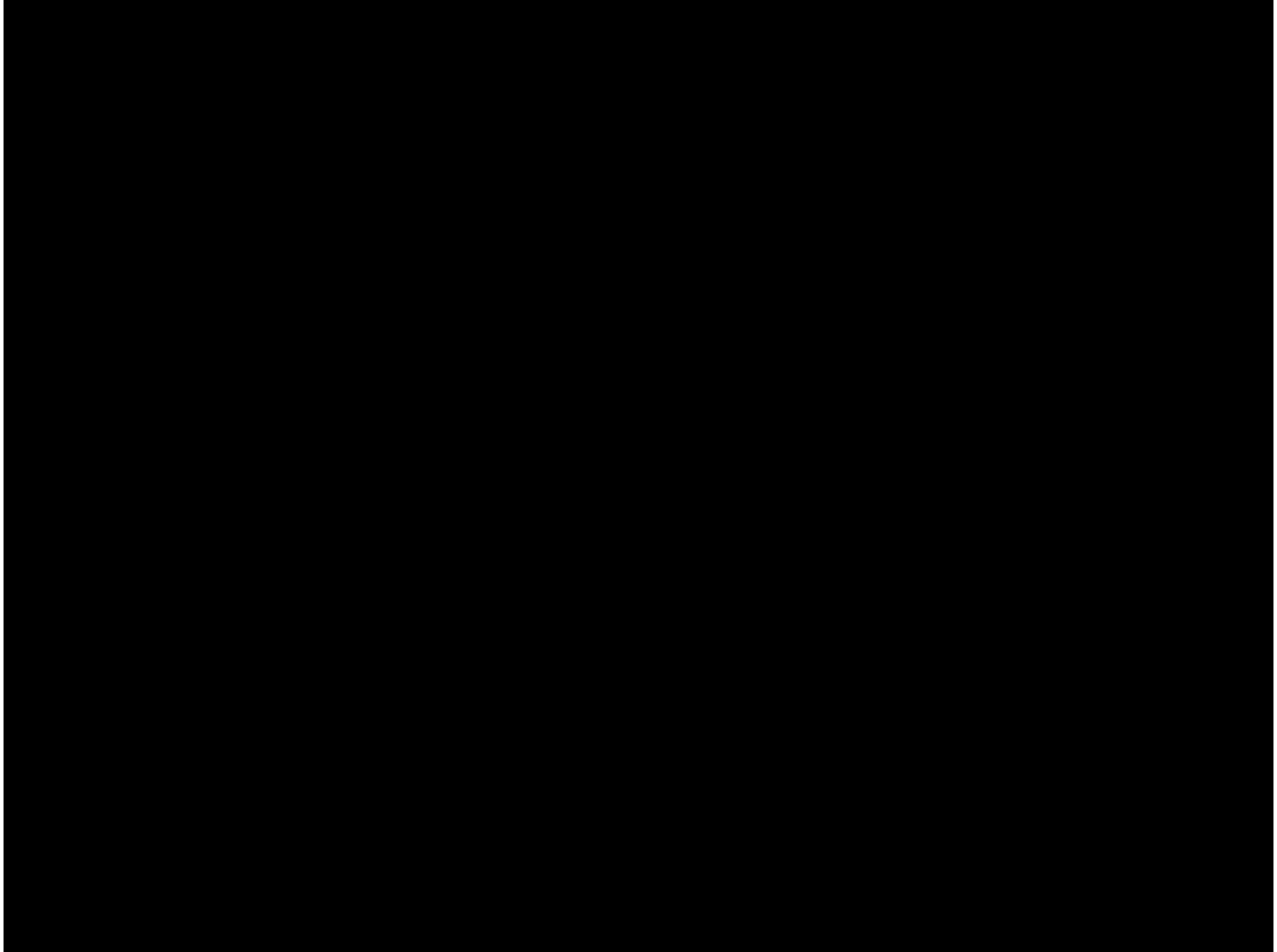
- Dashed lines are reference motion
- Policy not only reproduces the reference motion

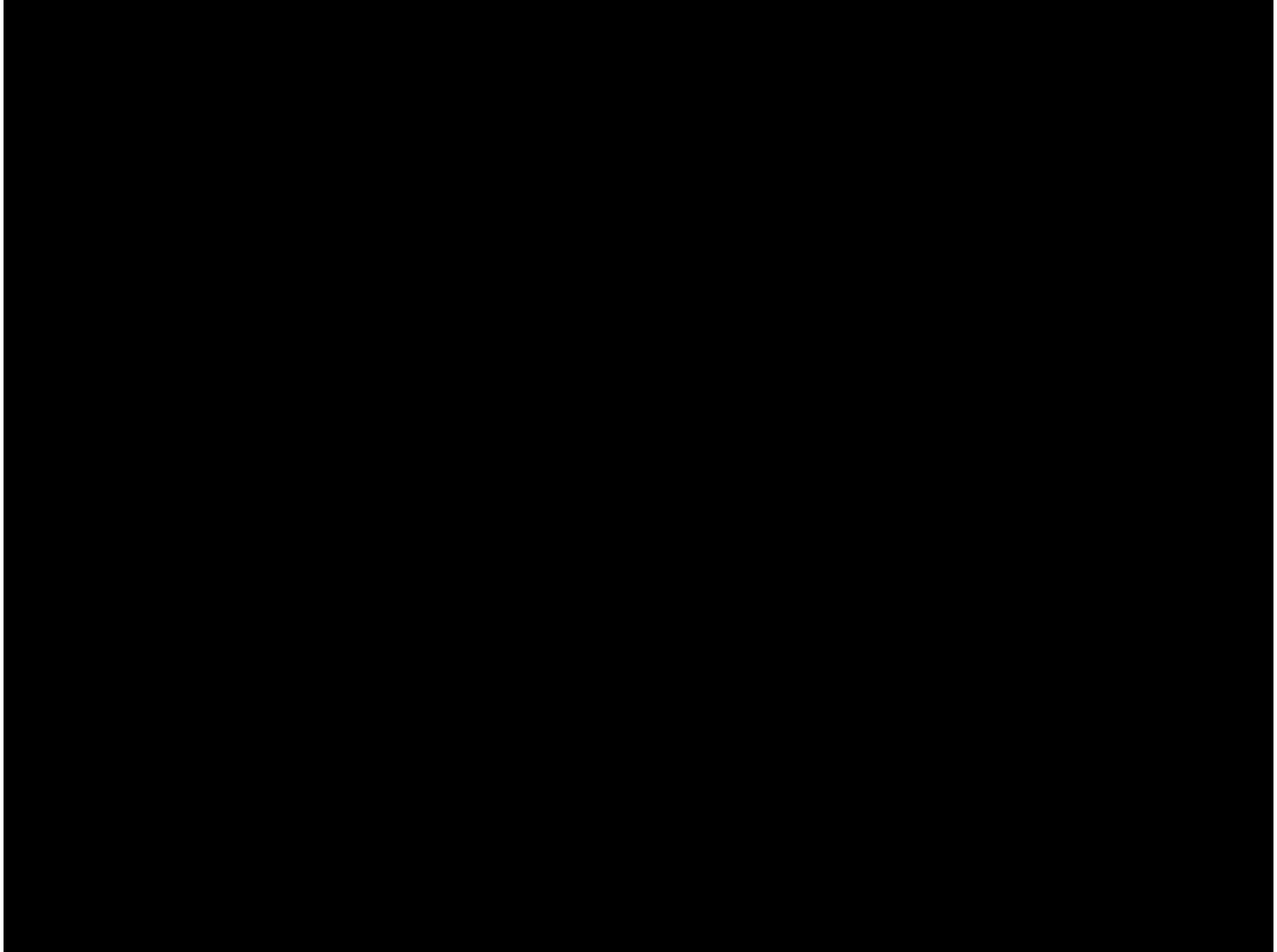




Sim2Real

- Domain randomization in PyBullet
    - Links: mass, inertia
    - Joints: torque, velocity
    - Simulation: restitution, lateral friction, spinning friction, rolling friction
- Transfer to Webots
    - More accurate robot model including backlash
    - Soft (grass-like) floor
    - Different physics engine (ODE instead of Bullet)
- Transfer to actual hardware

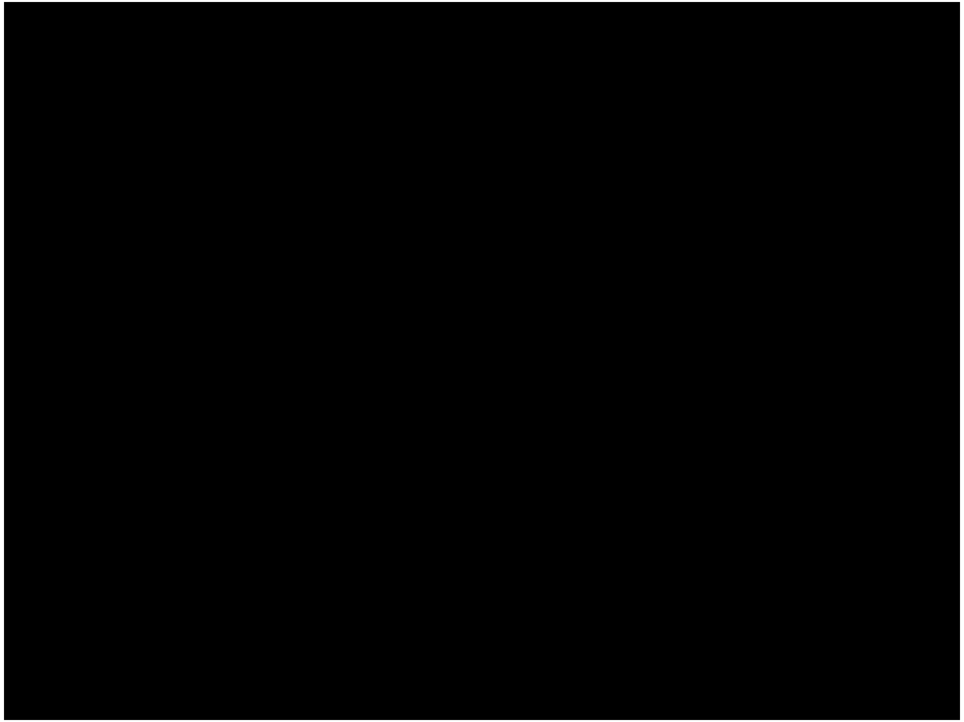




Current Issues

- Not completely stable on real hardware
- Trained policies sometimes work and sometimes not
    - Based on choices like using a terrain
    - But also differences between two trainings
- Domain randomization maybe not diverse enough
- Delay maybe needs to be modelled

Remaining Issues



Future Work

- Improve performance on actual hardware
- Run further experiments
  - Adaptive phase
  - Beta distribution
  - Different network structure
  - Other reward functions
  - Try on different robot models
- (Implement in Mujoco)
- Do same approach with stand-up motion
  - Together with Sebastian Stelter
- Path planning
  - Master thesis of Jasper Güldenstein

Questions?