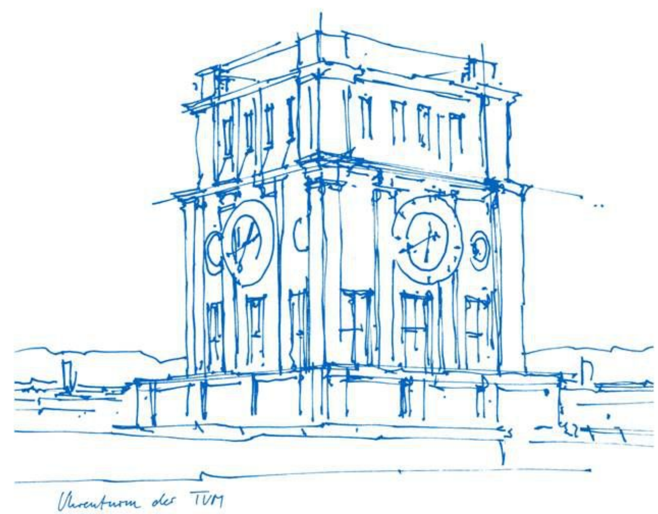
Master's Thesis in Informatics

Reinforcement Learning for Autonomous Locomotion Control of Snake-Like Robots

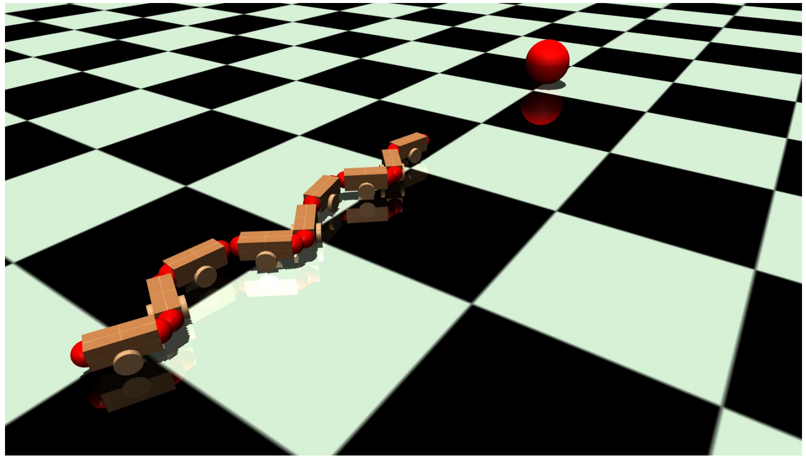
Author: Christian Lemke
Christian.Lemke@campus.lmu.de
Supervisor: Prof. Dr.-Ing. habil. Alonis Knoll
Advisor: M.Eng. Zhenshan Bing

Date: 27.07.2018



Content

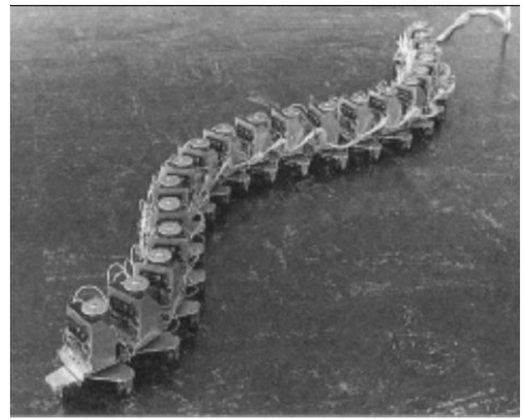
- Background of Snake-Like Robots
- Background of Reinforcement Learning
- Simulation environment
- Two control experiments
- Conclusion



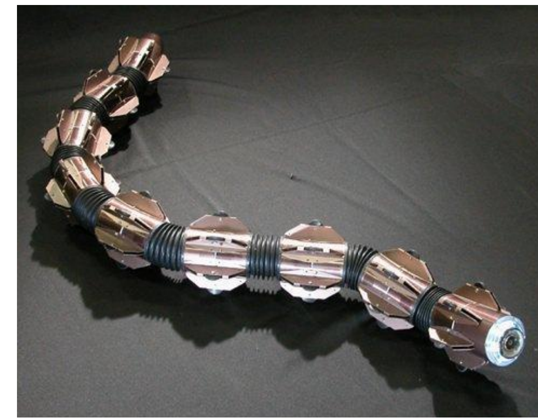
Snake-Like Robots

Mobility:
- Swim in water
- Climb stairs and poles
- Move in narrow spaces

Use cases:
- Fire fighting
- Inspection and maintenance
- Search and rescue



ACM III (1972)



ACM-R5 (2005)

Sources: Tokyo Institute of Technology | Shigeo Hirose, Peter Cave, and Charles Goulden. Biologically inspired robots: serpentile locomotors and manipulators. Oxford University Press, 1993 | LILJEBÄCK, Pål, et al. *Snake robots: modelling, mechatronics, and control*. Springer Science & Business Media, 2012. | Biorobotics Lab Carnegie Mellon University. http://biorobotics.ri.cmu.edu/projects/modsnake/

Snake-Like Robots

The modeling and control problems:

- Highly redundant degrees of freedom
- Complex interaction with the environment
- Difficult to control in real-world situations

Question:

How to solve this complex control problem?

Approach:

Reinforcement Learning

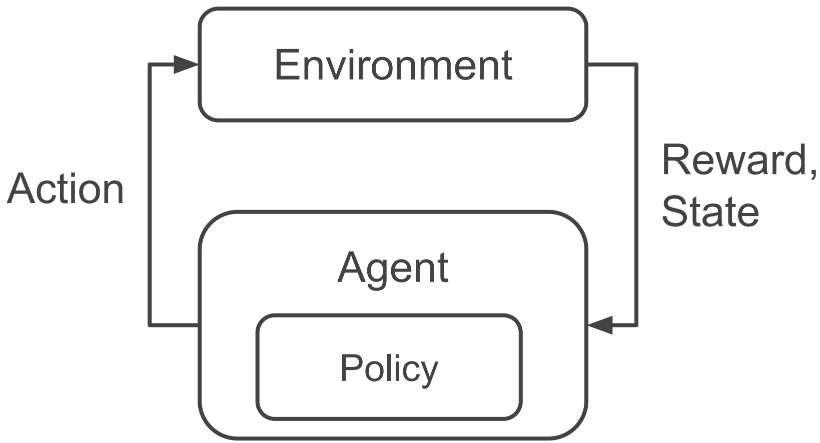
Experiments:

- Autonomous Locomotion Control
- Autonomous Target Tracking

Search and rescue scenario

Sources: LILJEBÄCK, Pål, et al. *Snake robots: modelling, mechatronics, and control*. Springer Science & Business Media, 2012.

Reinforcement Learning




Box-and-Banana Problem

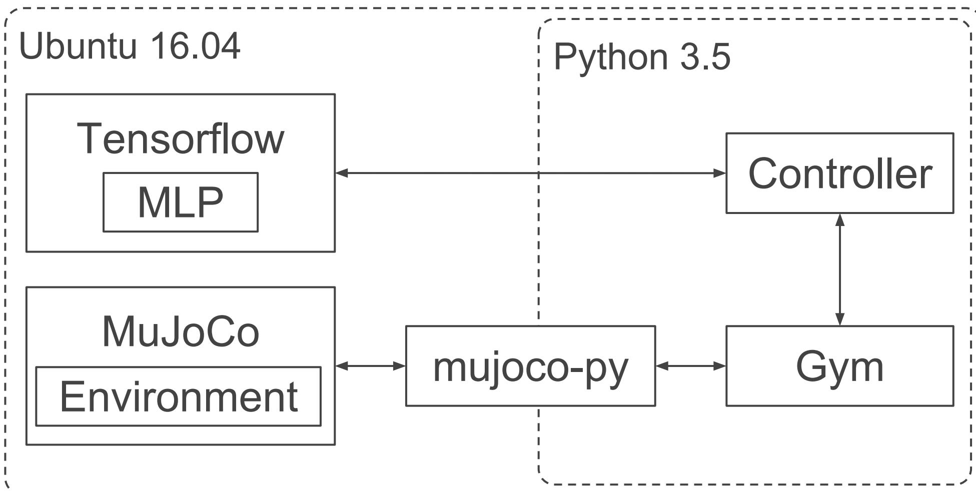
Proximal Policy Optimization

- Best performance on continuous control tasks
- Easy to implement
- Good sample efficiency
- Small hyperparameter tuning

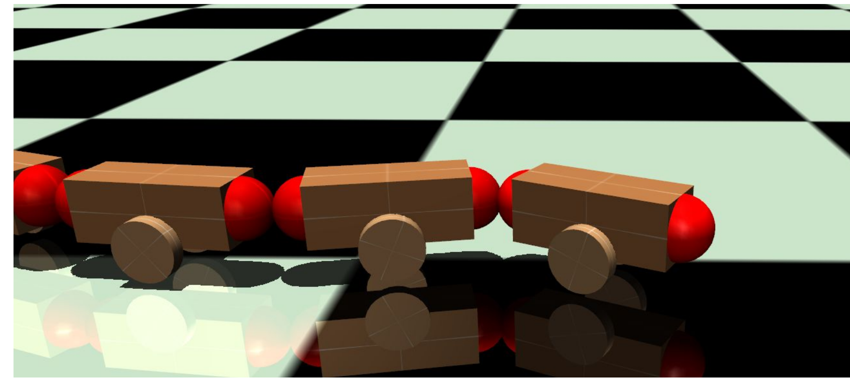
Sources: Dr. Epstein. https://www.nytimes.com/1984/04/17/science/science-watch-pigeons-show-how-to-solve-a-problem.html | Schulman et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

Environment and Robot

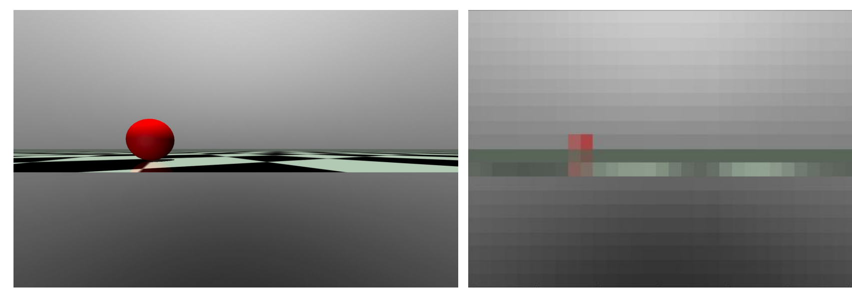
- 9 Modules and 8 Joints
- Servo position motors
- Wheels
- Vision via head camera



Components overview



The robot in MuJoCo



Vision of head camera

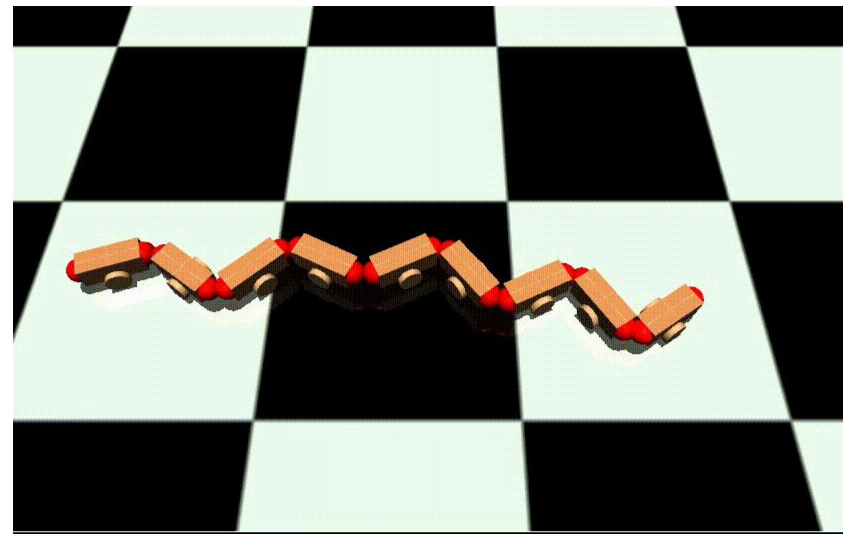
Rendering with 32x20 RGB

1. Experiment: Autonomous Locomotion Control

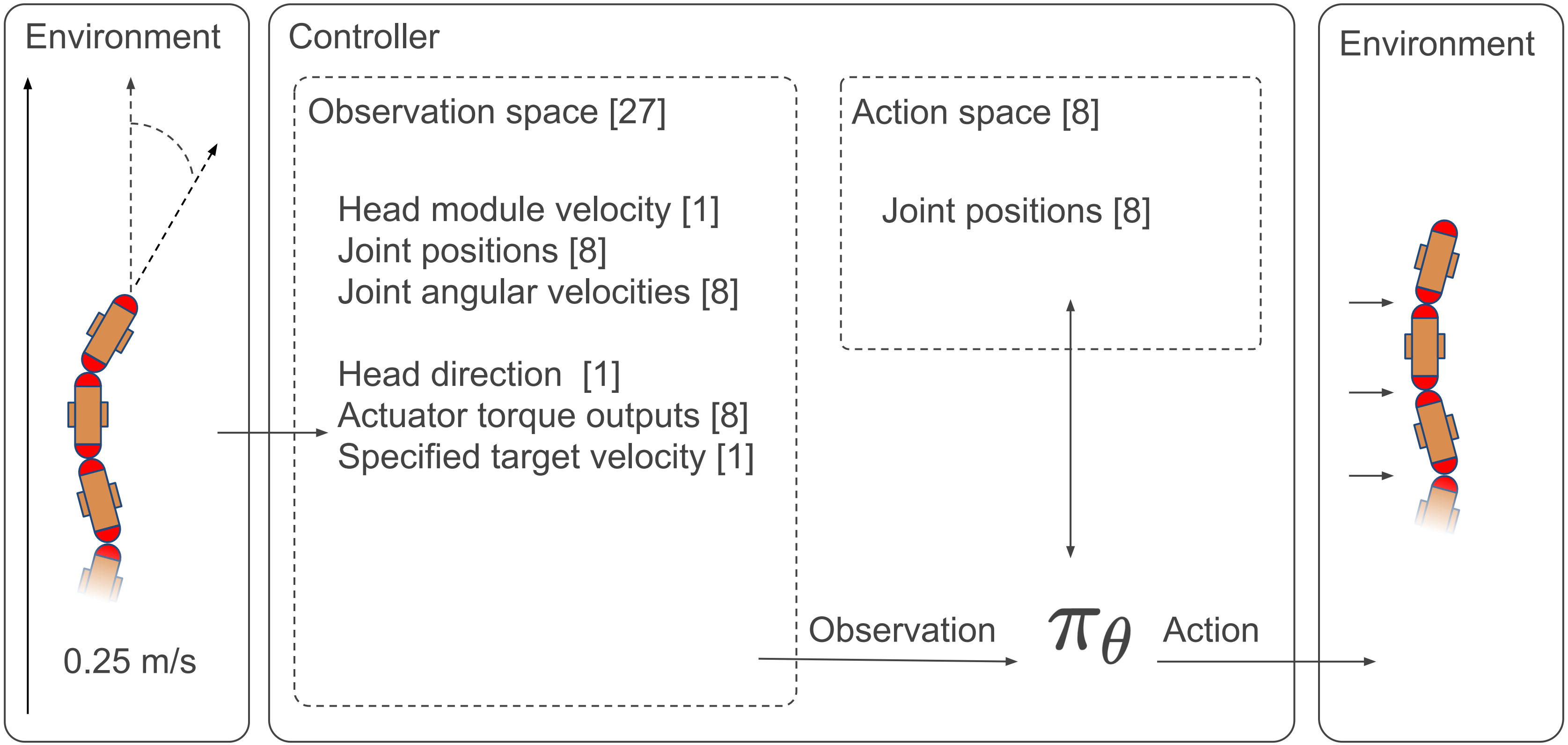
Task:
Perform a power efficient locomotion at a specified velocity.

Learn:
- Joint position commands for slithering locomotion
- Control velocity
- Power efficiency

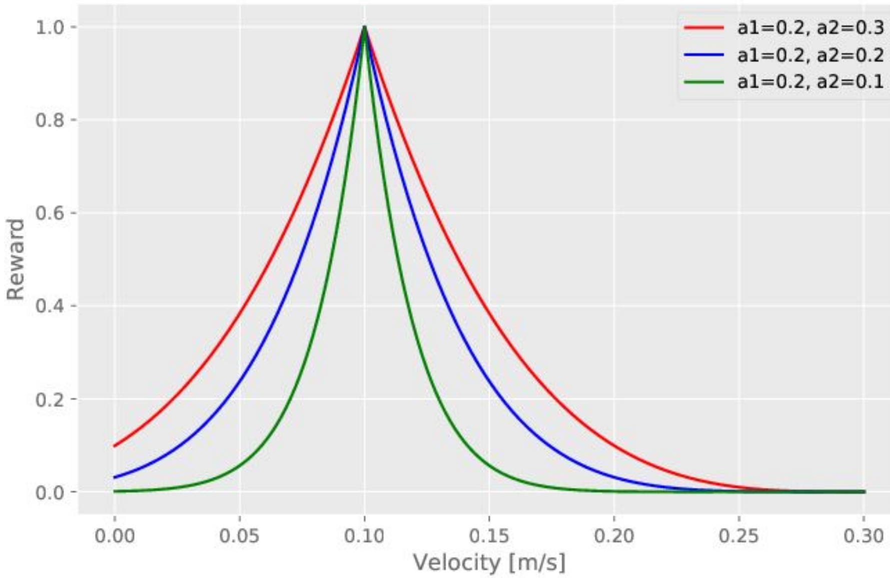
Evaluation:
Comparison to traditional equation controller

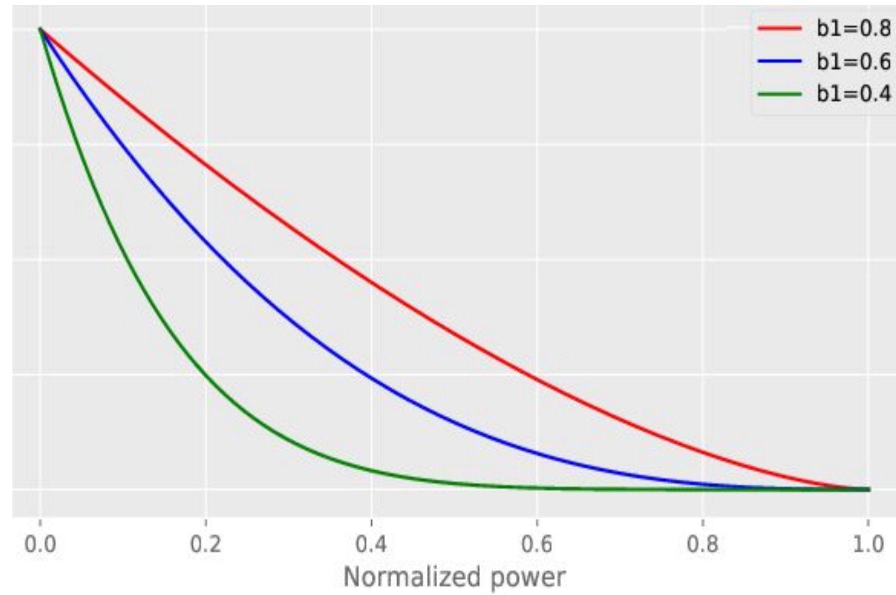


1. Experiment: Autonomous Locomotion Control
Observation Space and Action Space



1. Experiment: Autonomous Locomotion Control
Reward Function





Reward velocity:

$$r_v = \left(1 - \frac{|v_t - v|}{a_1}\right)^{\frac{1}{a_2}}$$

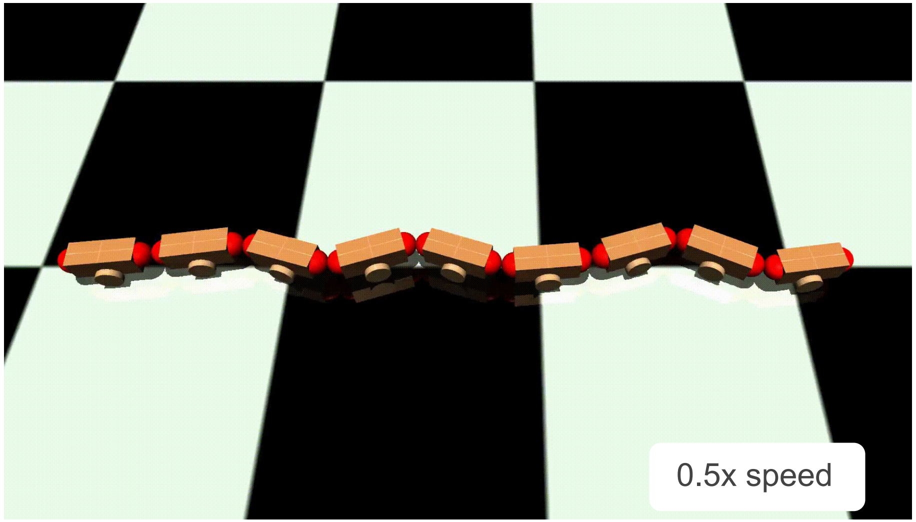
Reward power efficiency:

$$r_P = \left|1 - \hat{P}\right|^{b_1^{-2}}$$

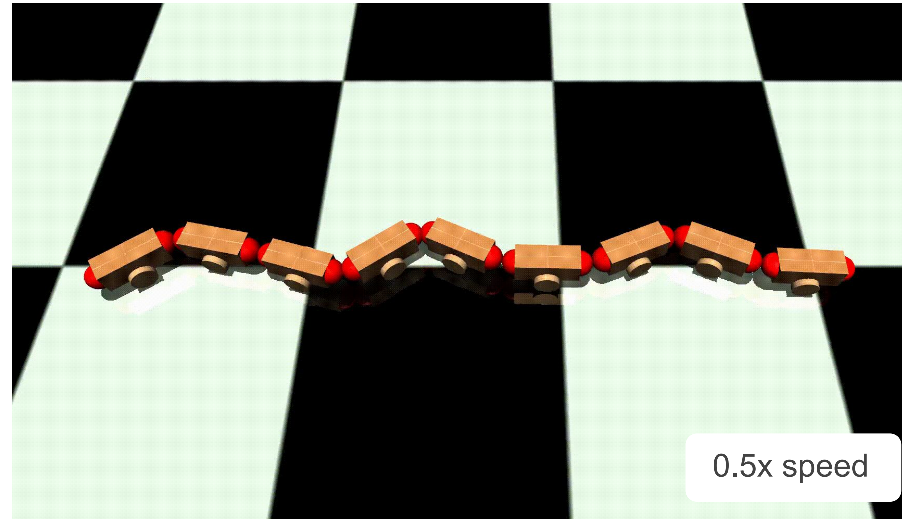
$$\hat{P} = \frac{|\tau \dot{\phi}|}{\tau_{max} \dot{\phi}_{max}}$$

- Reward is combination of velocity and power usage $r = r_v r_P$

1. Experiment: Autonomous Locomotion Control Results



- Velocity of 0.05 m/s
- “Concertina” gait pattern
- Contracts and stretches the body



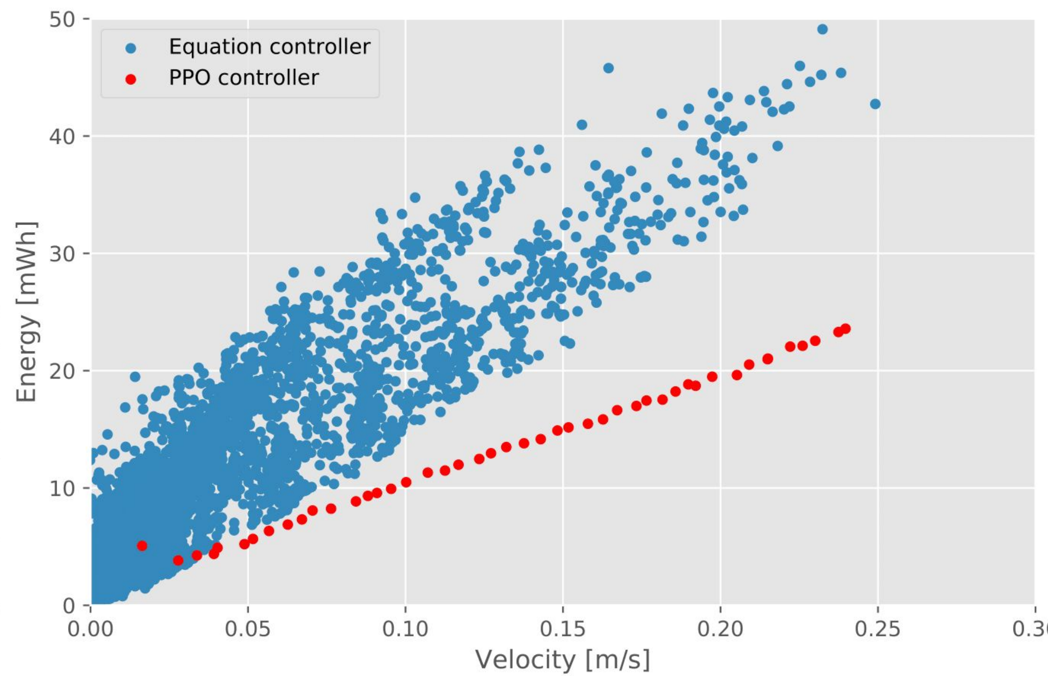
- Velocity of 0.25 m/s
- “Lateral undulation” gait pattern
- Carries waves from head to tail

1. Experiment: Autonomous Locomotion Control Comparison with traditional Equation Controller

- An efficiency comparison
- Grid search creates a variety of different gaits
- Total of 6480 gait parameter sets

Descriptions	Values
Angular frequency	0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0
Linear reduction	0.1, 0.2, 0.3, 0.4
Amplitude (in degrees)	40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180
Bending radius (in degrees)	40, 50, 60, 70, 80, 90, 100, 110, 120

Table of the equation controller parameters

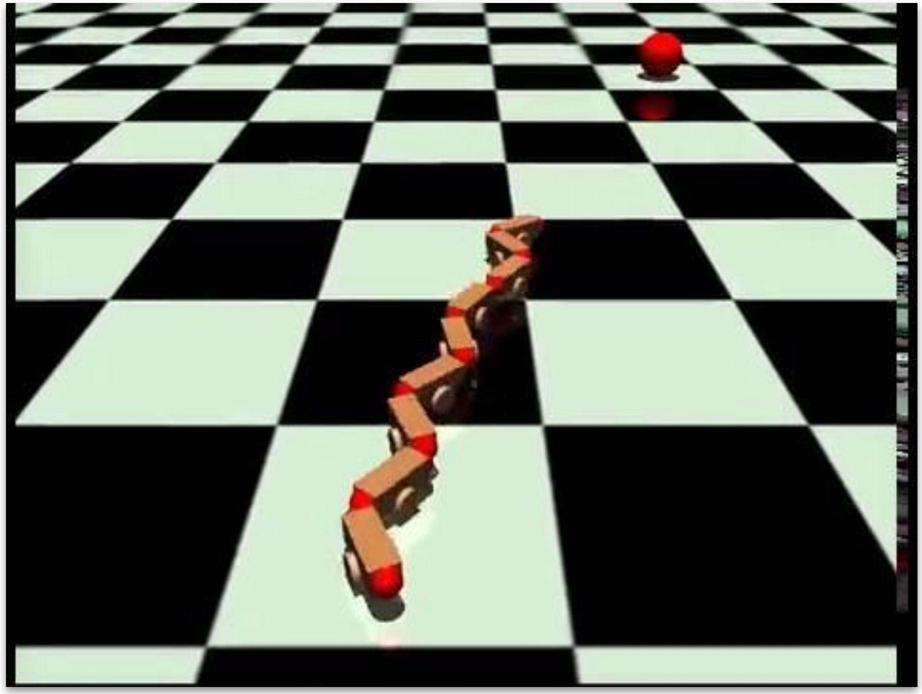


2. Experiment: Autonomous Target Tracking

Task:
Follow a moving target with a certain distance.

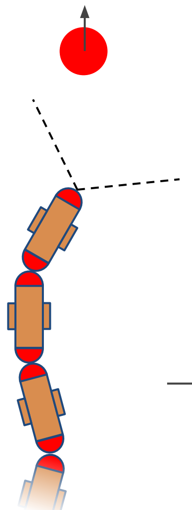
Learn:
- Control joints for locomotion
- Use head camera to track the target and estimate the distance

Evaluation:
Test on different target tracks



2. Experiment: Autonomous Target Tracking
Observation Space and Action Space

Environment



Controller

Observation space [49]

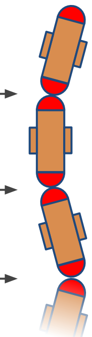
- Joint positions [8]
- Joint angular velocities [8]
- Head module velocity [1]
- Head camera Pixels [32]

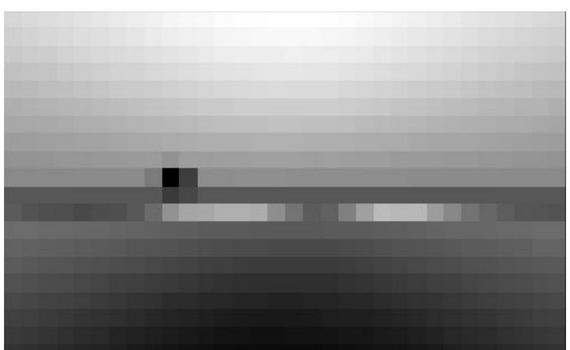
Action space [8]

- Joint positions [8]

Observation π_θ Action

Environment

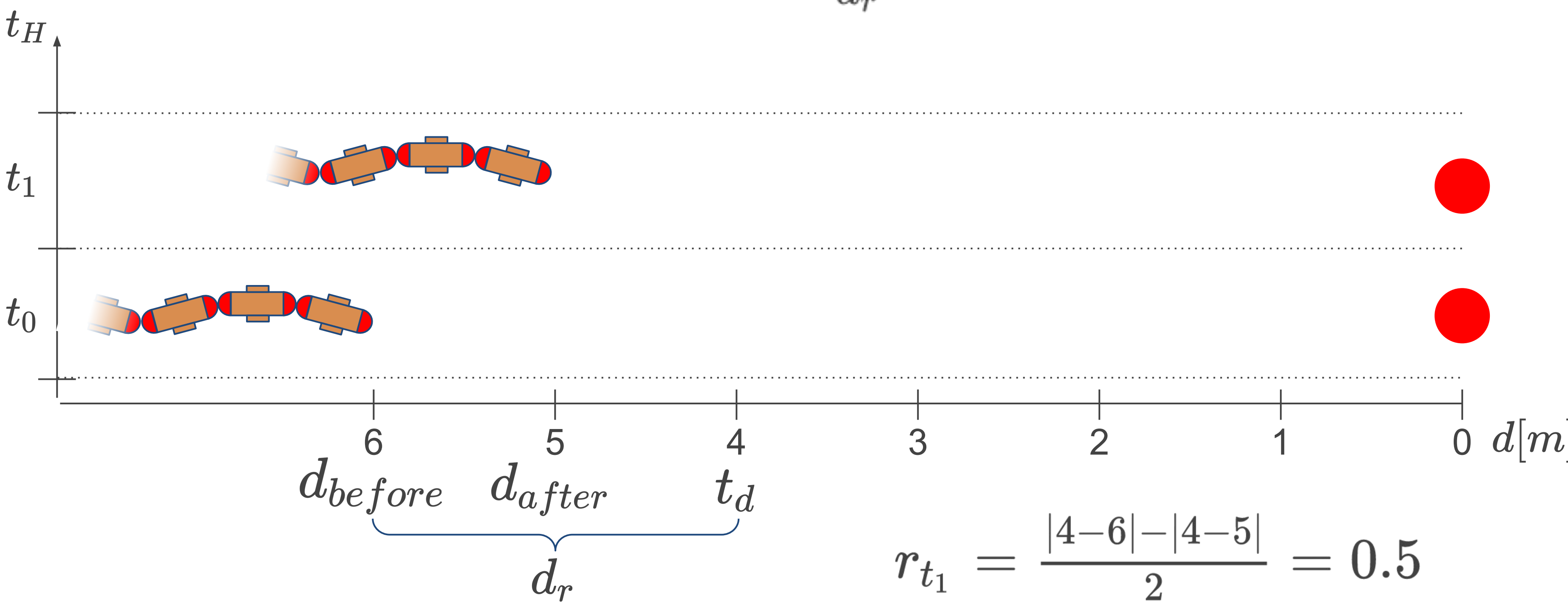




Select one row:
32x20 to 32 Pixels

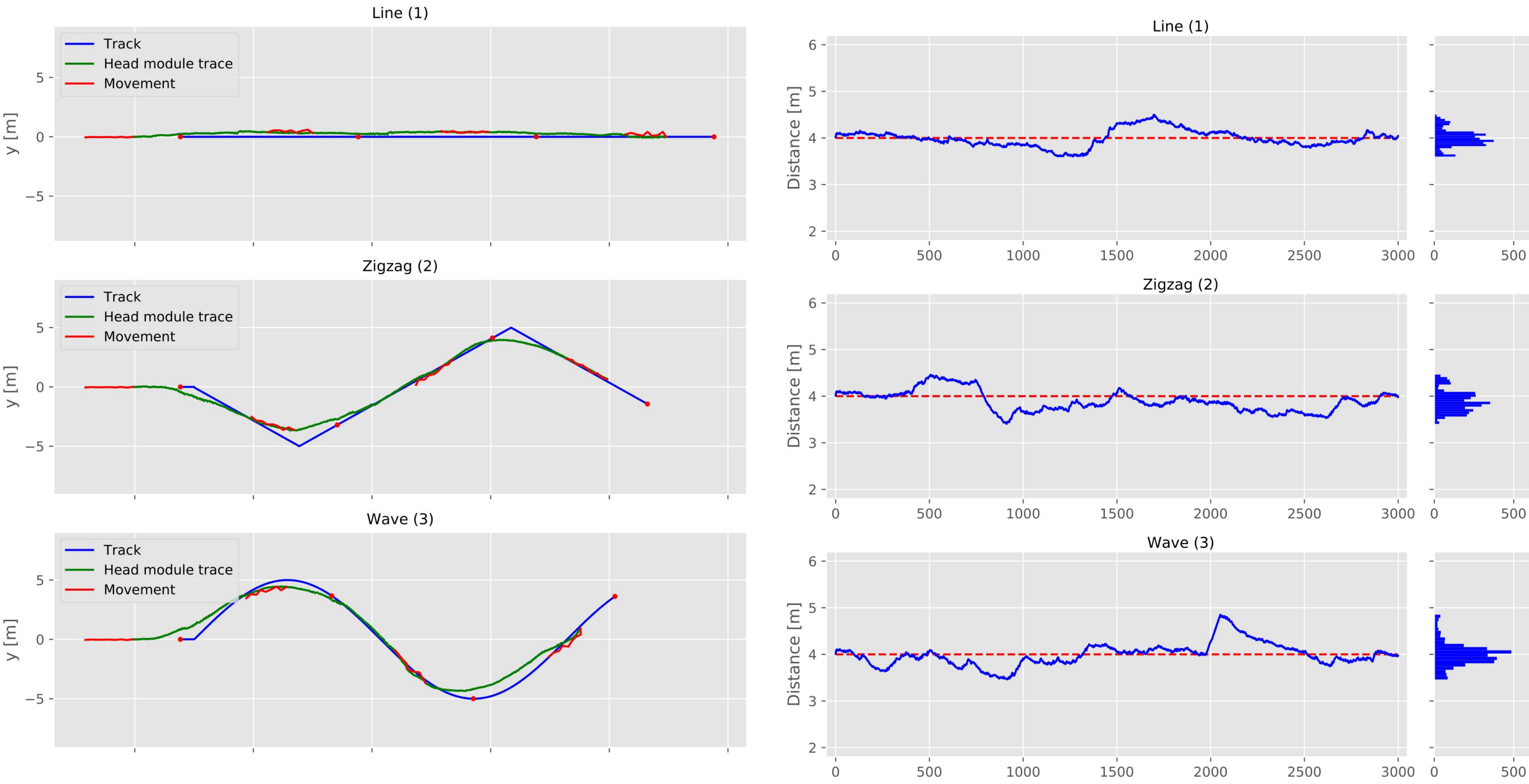
2. Experiment: Autonomous Target Tracking
Reward Function

$$r = \frac{|t_d - d_{before}| - |t_d - d_{after}|}{d_r}$$



$$r_{t_1} = \frac{|4-6|-|4-5|}{2} = 0.5$$

2. Experiment: Autonomous Target Tracking
Result

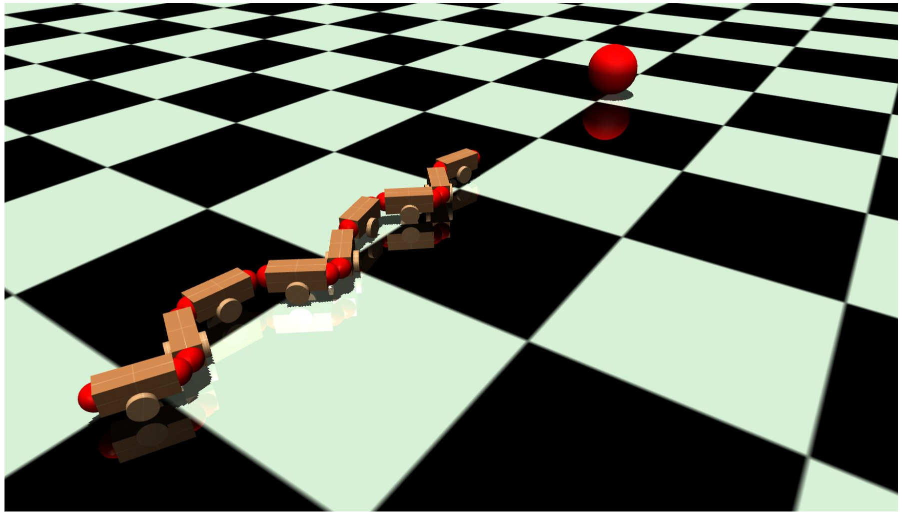


Conclusion

Advantages:
- Directly solves the problem
- No control engineering

Disadvantages:
- Challenging to develop suitable reward functions
- The policy is difficult to interpret

Future work:
- PPO on a real robot
- 3D Snake-Like Robot Model
- Explore gait adaptiveness on 3D model



Thank you

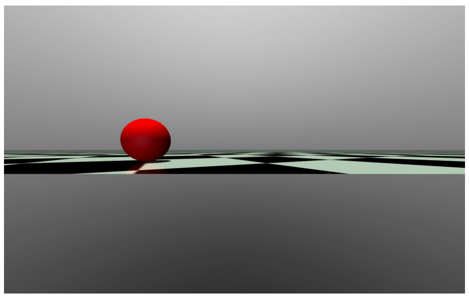
Master's Thesis in Informatics

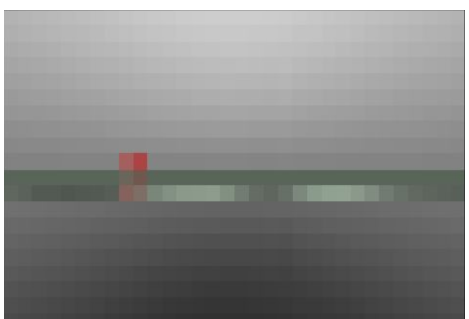
Reinforcement Learning for Autonomous Locomotion Control of Snake-Like Robots

Author: Christian Lemke
Christian.Lemke@campus.lmu.de

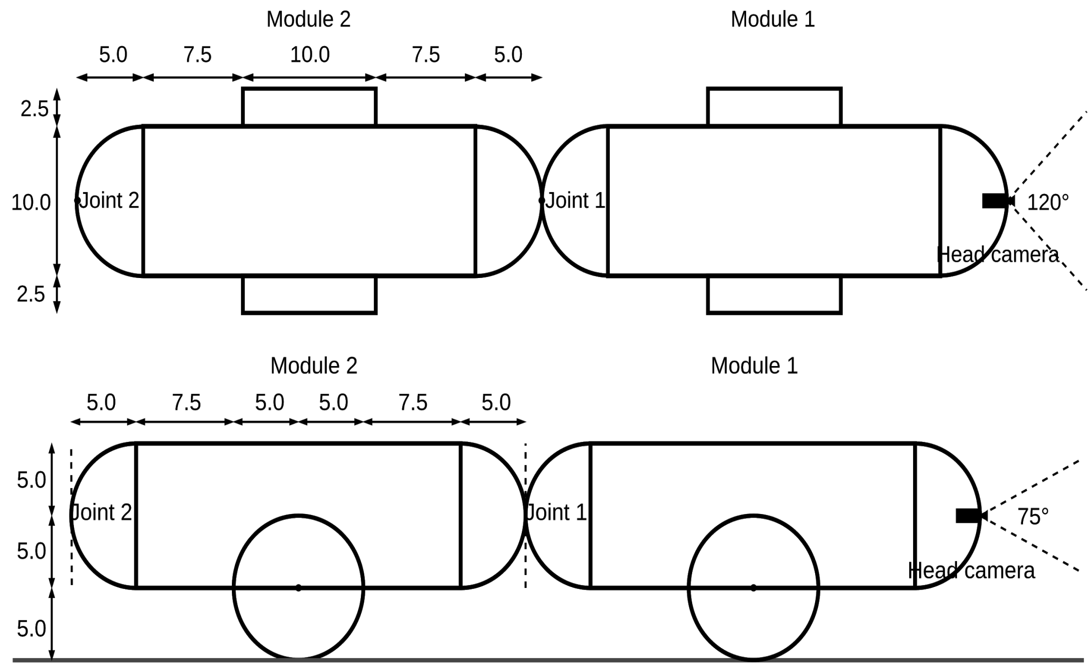
Backup

Model and Vision



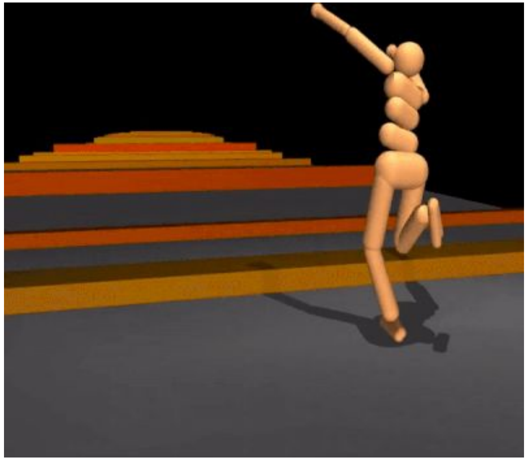


Rendering with
32x20 RGB



Proximal Policy Optimization

- Based on Policy gradient methods
- Best performance on continuous control tasks
- Simple to implement and handle
- Good sample efficiency



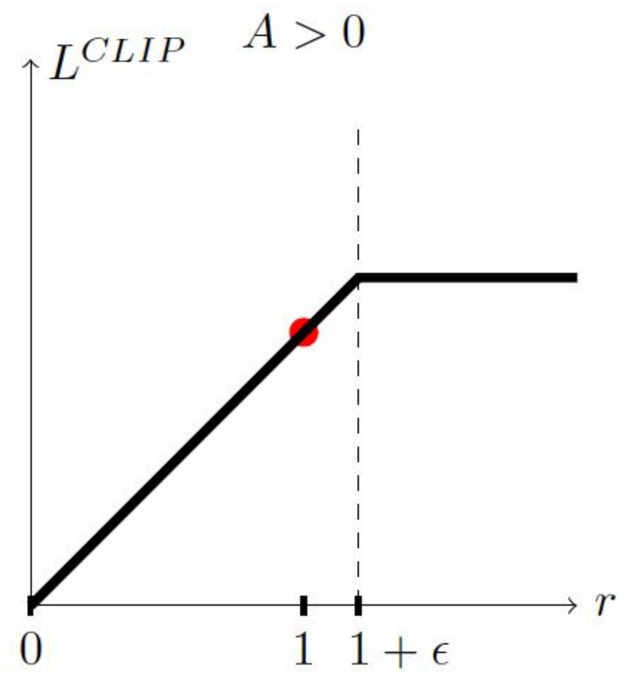
A simulated 'humanoid' walker

Sources: Schulman et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017). | https://blog.openai.com/openai-baselines-ppo/ | Nicolas Heess et al. Emergence of locomotion behaviours in rich environments. https://deepmind.com/blog/producing-flexible-behaviours-simulated-environments/

Proximal Policy Optimization

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

Components	
Objective function	$L^{CLIP}(\theta)$
Probability ratio	$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}$
Advantage function	$\hat{A}_t$
Min function	$min(value_1, value_2)$
Clip function	$clip(value, min, max)$
Clipping parameter	$1-\epsilon, 1+\epsilon$



Sources: John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

Proximal Policy Optimization

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right]$$

- θ is the policy parameter
- $\hat{E}_t$ denotes the empirical expectation over timesteps
- r_t is the ratio of the probability under the new and old policies, respectively
- $\hat{A}_t$ is the estimated advantage at time t
- ε is a hyperparameter, usually 0.1 or 0.2

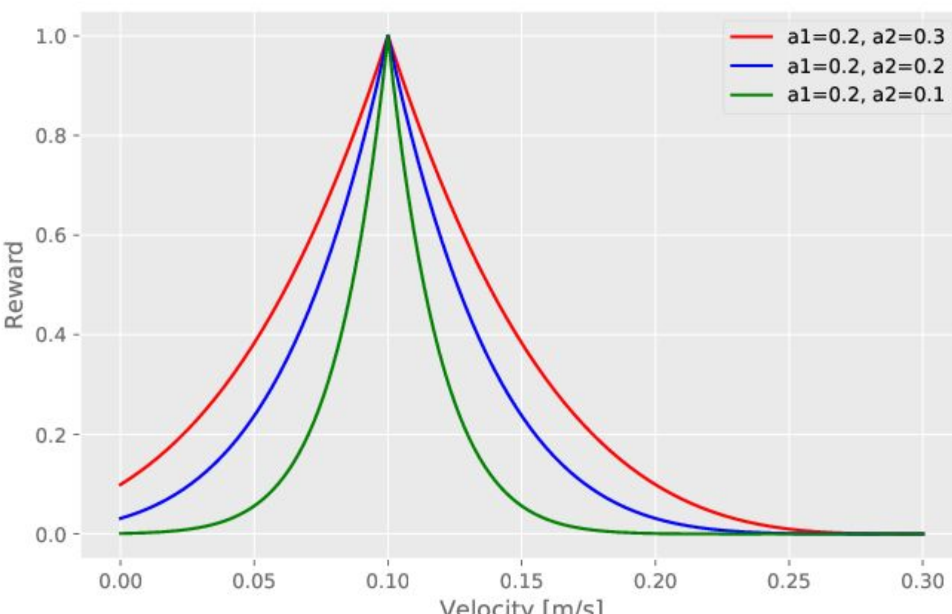
Sources: John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

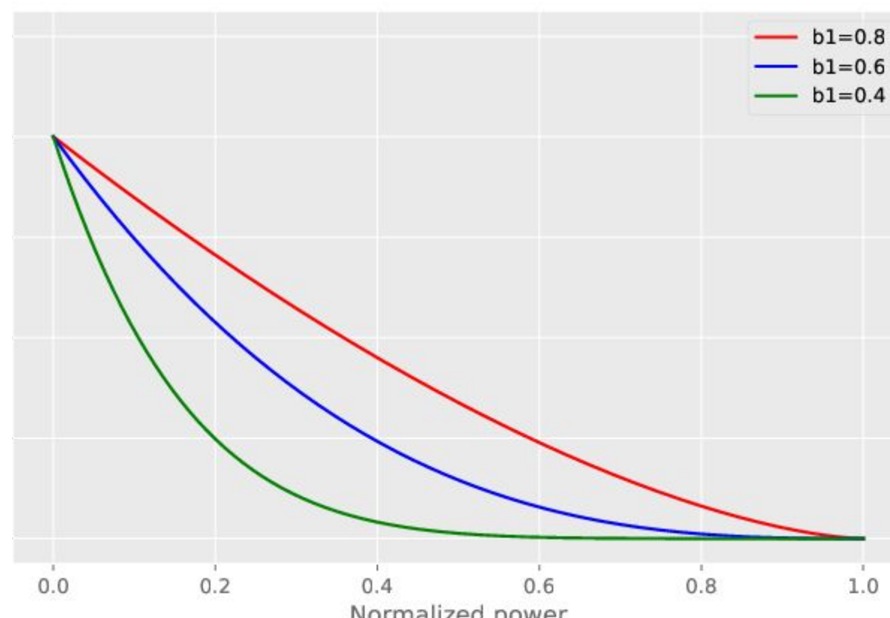
1. Experiment: Autonomous Locomotion Control
Reward Function

- Reward is combination of velocity and Power usage
- Both are normalized and multiplied together

$$r_v = \left(1 - \frac{|v_t - v|}{a_1}\right)^{\frac{1}{a_2}}$$

$$r_P = r_{max}\left|1 - \hat{P}\right|^{b_1^{-2}}$$





Power measurement

The most straightforward measure of power usage for the j-th actuator ϵ_j is the absolute value of the product of the torque τ_j and its angular velocity $\dot{\phi}_j$. The total power consumption P of all m actuators on each time step is calculated by

$$P = \sum_{j=1}^{m} |\tau_j \dot{\phi}_j| \tag{4.1}$$

where

$$\tau_j = f_j g_j \tag{4.2}$$

$$\hat{P} = \frac{1}{m} \sum_{j=1}^{m} \frac{|f_j g_j \dot{\phi}_j|}{f_{max} g_j \dot{\phi}_{max}}$$

1. Experiment: Autonomous Locomotion Control
Equation controller

Traditionally locomotion gait

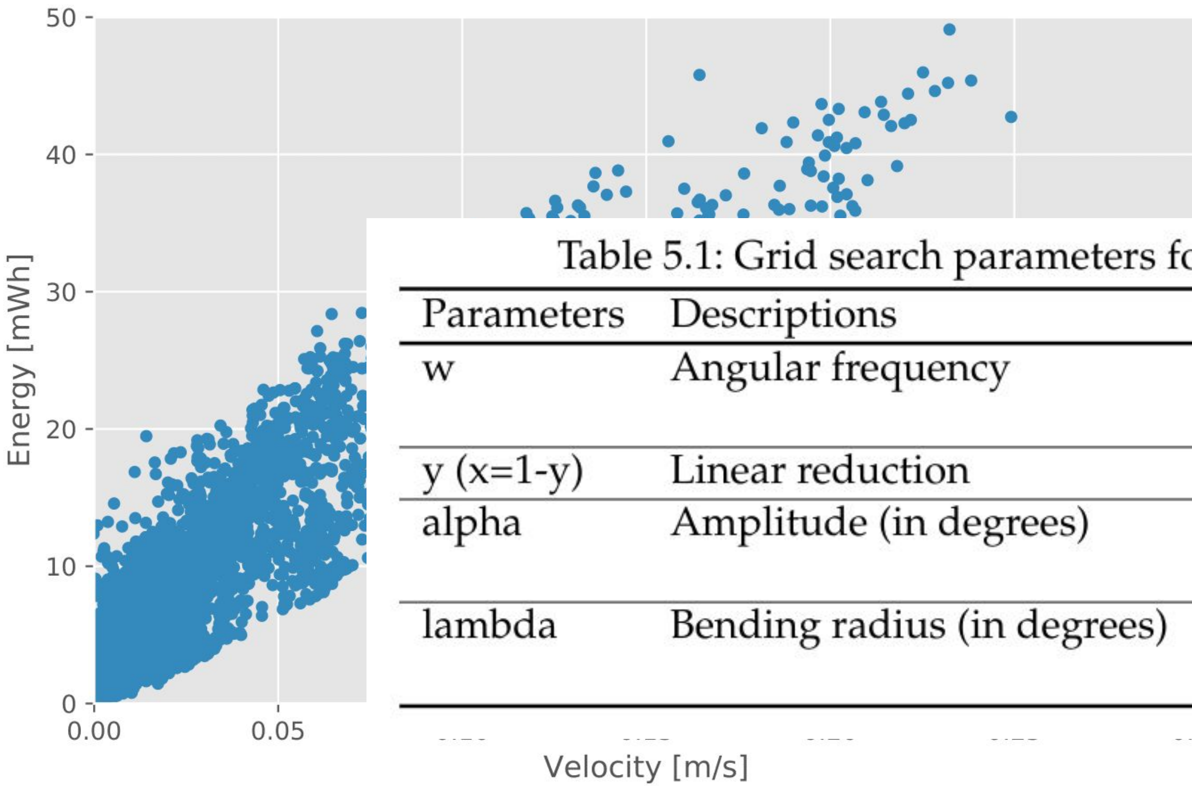


Table 5.1: Grid search parameters for the equation controller

Parameters	Descriptions	Values
w	Angular frequency	0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0
y (x=1-y)	Linear reduction	0.1, 0.2, 0.3, 0.4
alpha	Amplitude (in degrees)	40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180
lambda	Bending radius (in degrees)	40, 50, 60, 70, 80, 90, 100, 110, 120

Observation Space

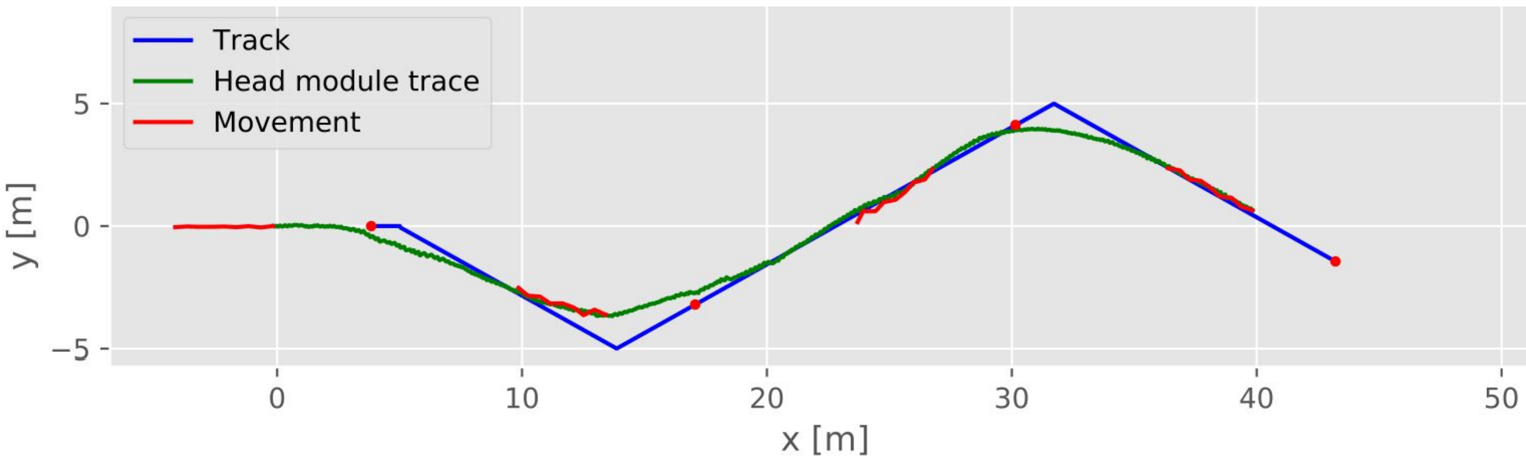
Table 5.3: Overview of the PPO controller observation space parameters

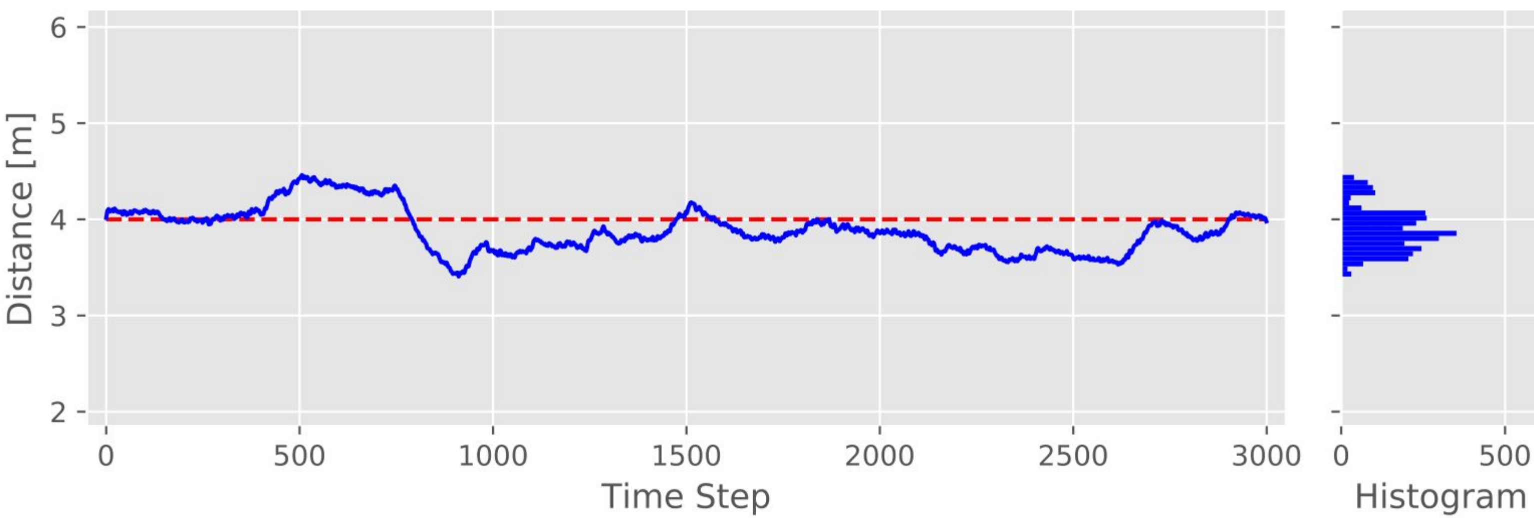
Symbols	Descriptions
ϕ_{1-8}	Relative joint angular positions
$\dot{\phi}_{1-8}$	Relative joint angular velocity
v_1	Absolute head module velocity (measured at (x_1, y_1))
τ_{1-8}	Actuator torque output
ϕ_t	Relative angle between the head direction and the target
v_t	Specified target velocity

Table 6.2: Overview of the target tracking controller observation space parameters

Symbols	Descriptions
ϕ_{1-8}	Relative joint angular positions
$\dot{\phi}_{1-8}$	Relative joint angular velocity
v_1	Head link velocity (measured at (x_1, y_1))
$p_{10,1-32}$	Pixel 1 to 32 of the 10th row of the gray camera image

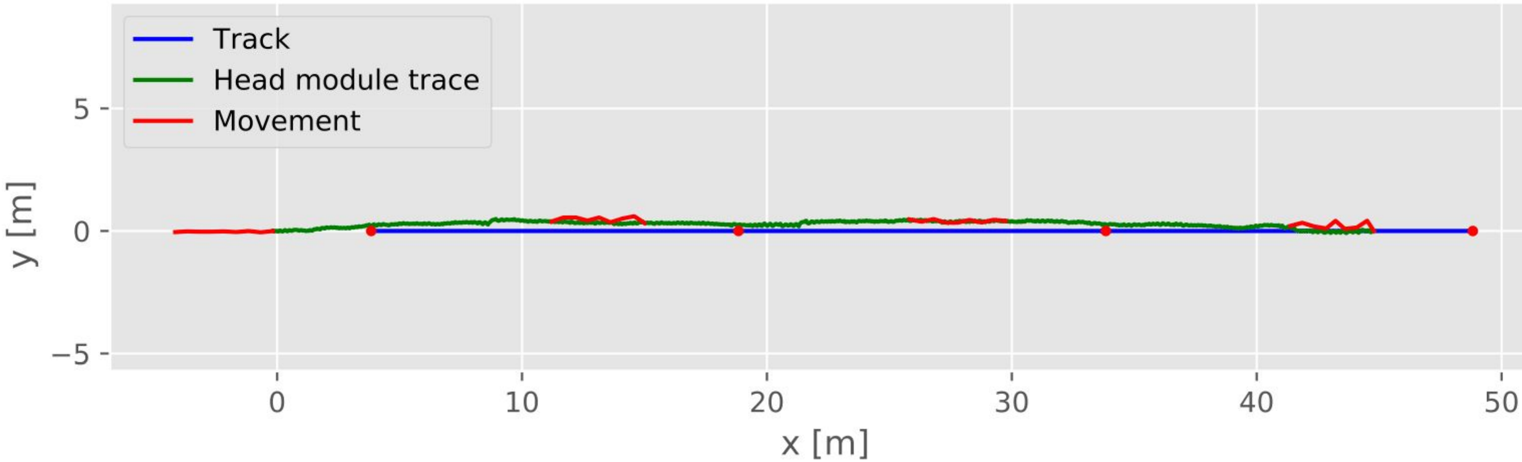
2. Experiment: Autonomous Target Tracking
Result: Zigzag-Track (1/4)

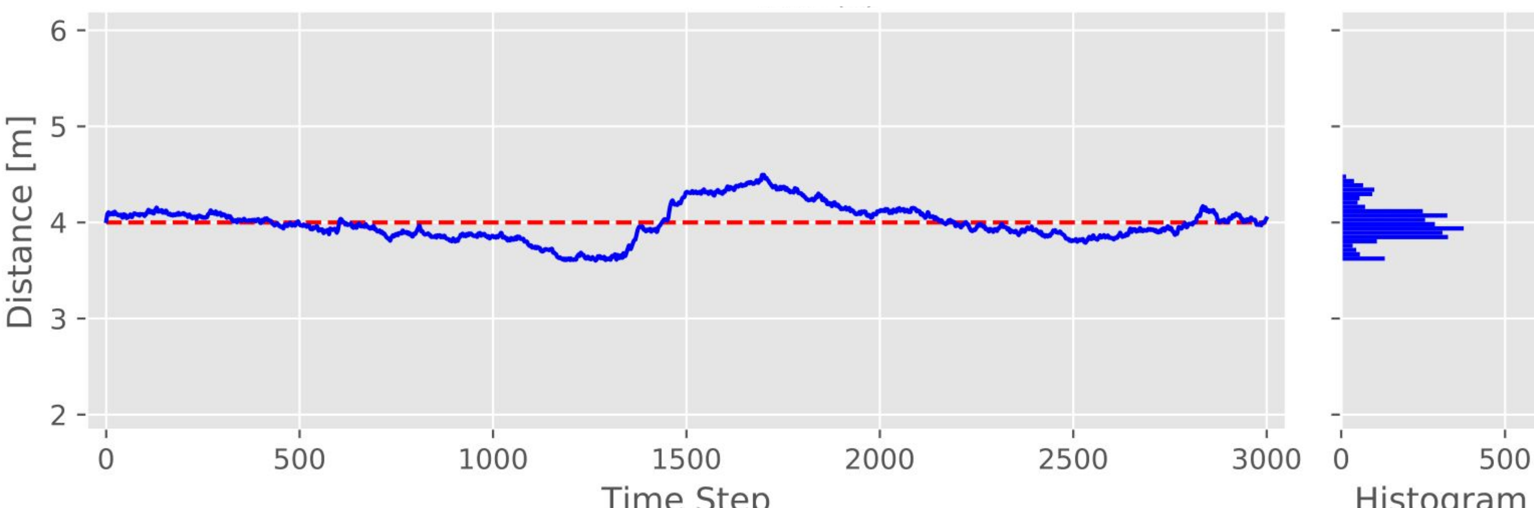




Mean:	3.96 m
Std:	0.24 m

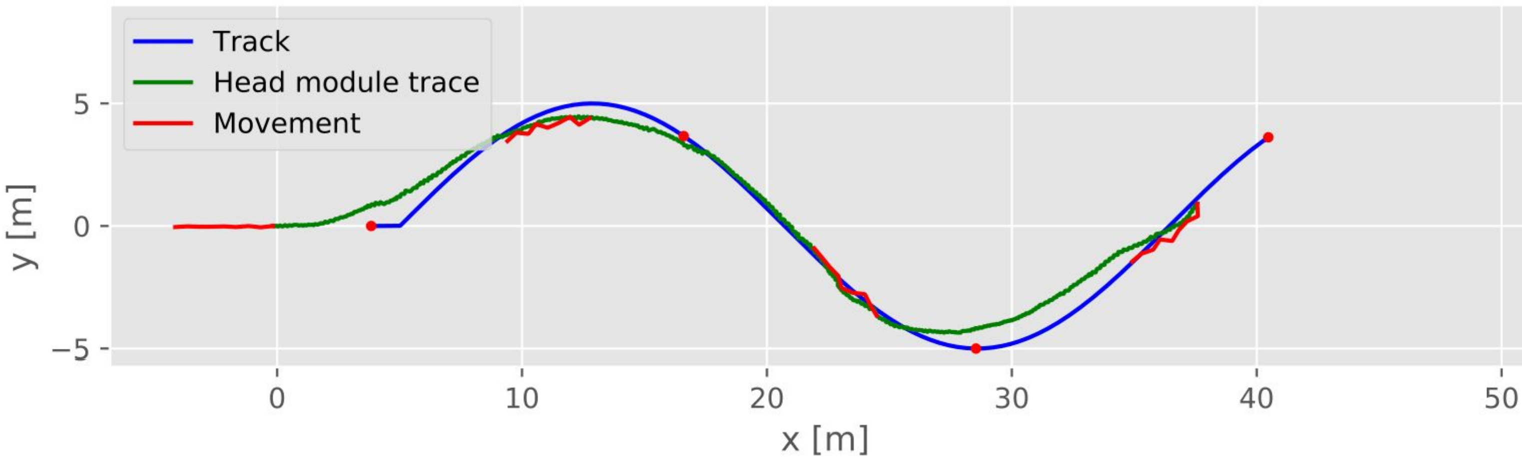
2. Experiment: Autonomous Target Tracking
Result: Line-Track (2/4)

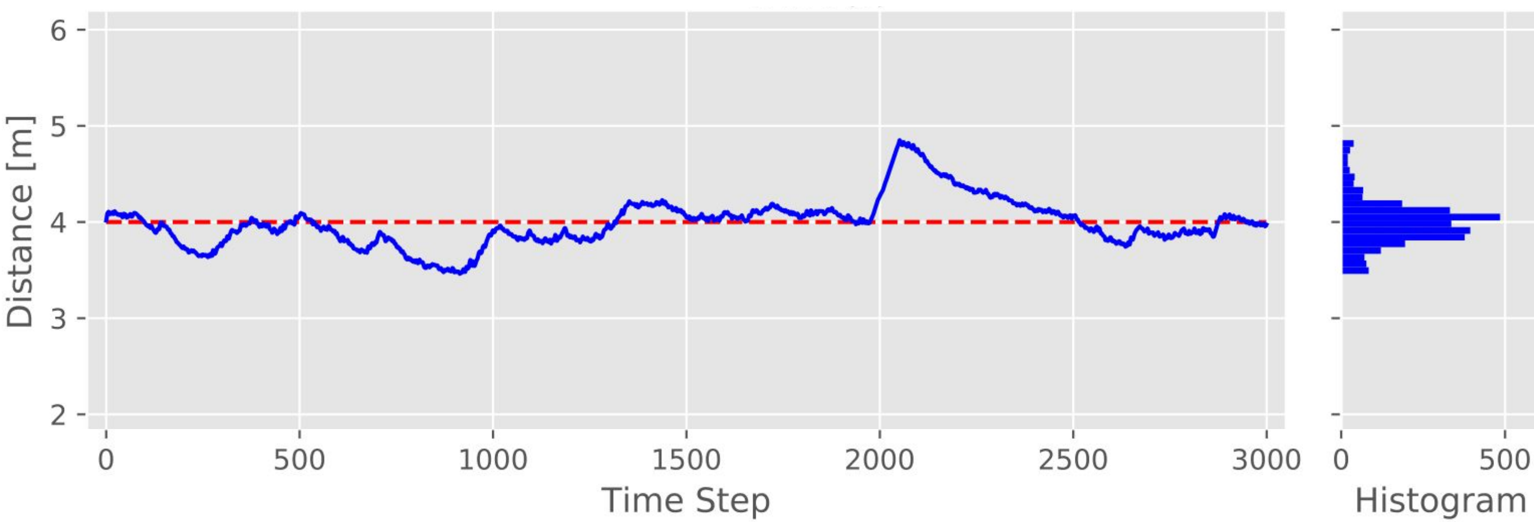




Mean:	3.99 m
Std:	0.18 m

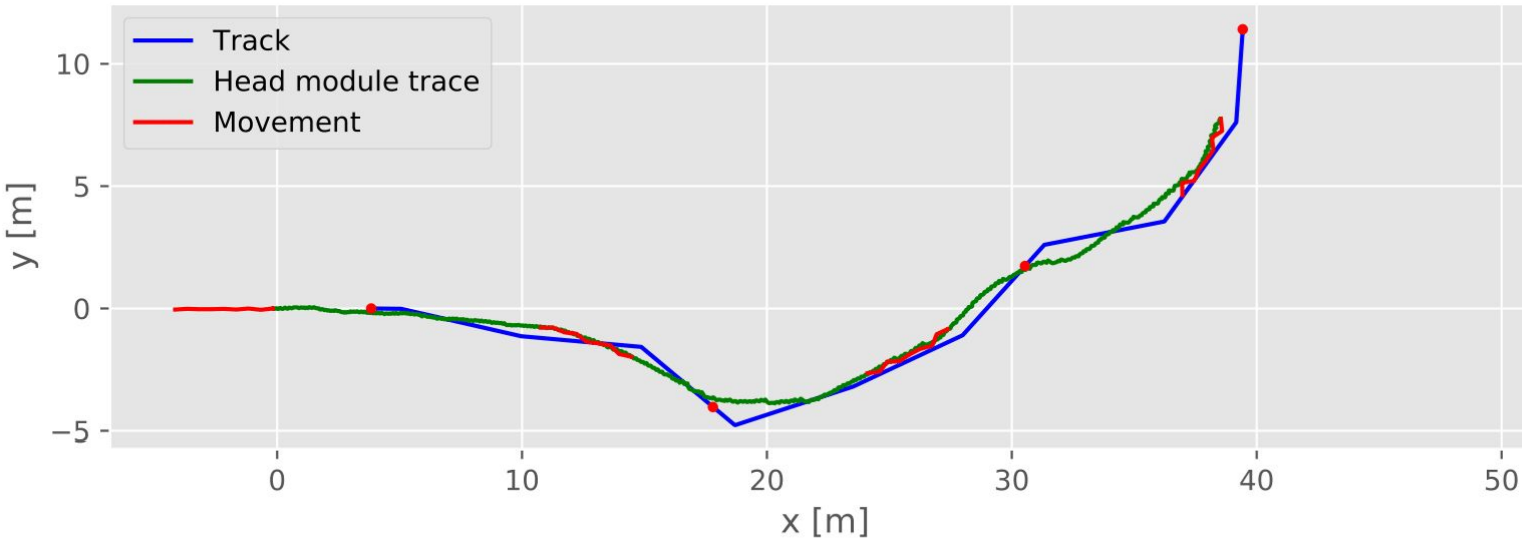
2. Experiment: Autonomous Target Tracking
Result: Wave-Track (3/4)

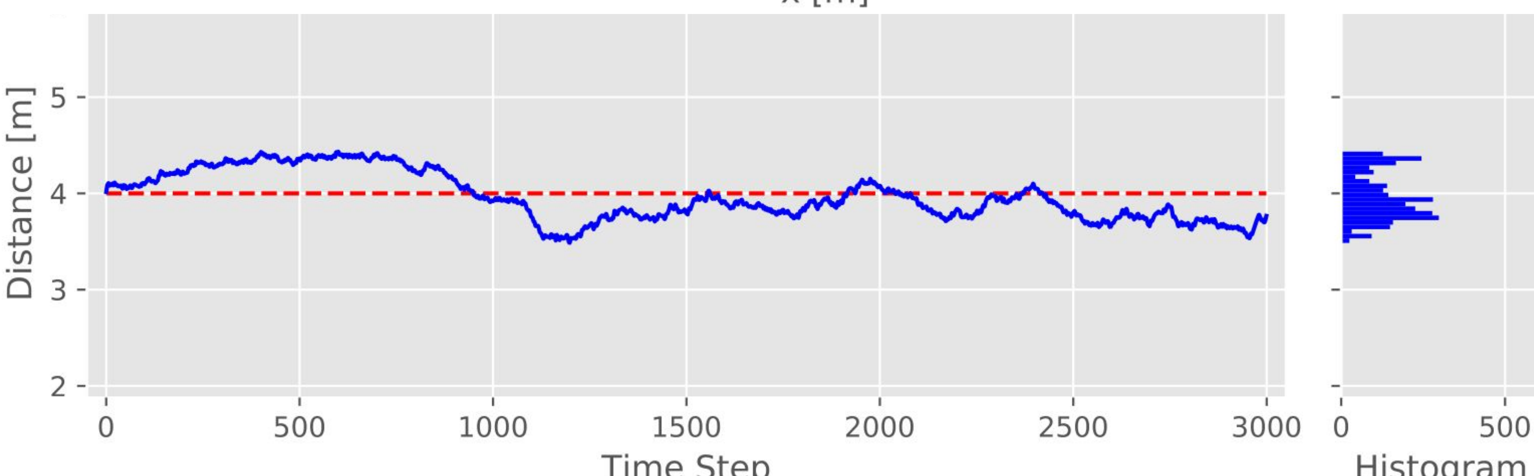




Mean:	3.99 m
Std:	0.24 m

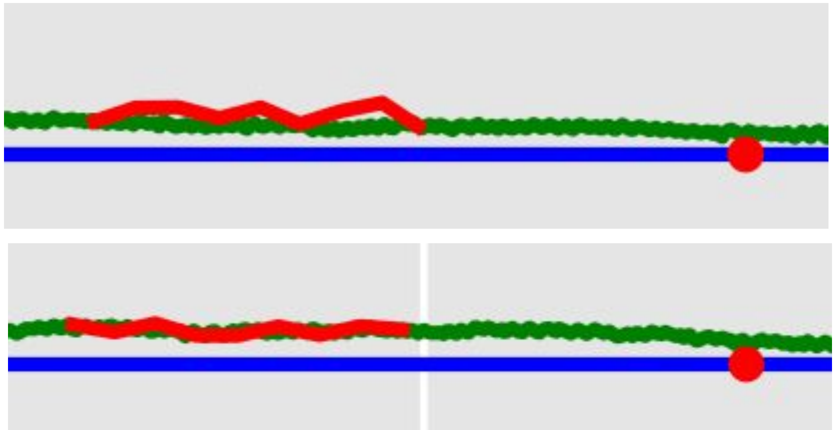
2. Experiment: Autonomous Target Tracking
Result: Random-Track (4/4)



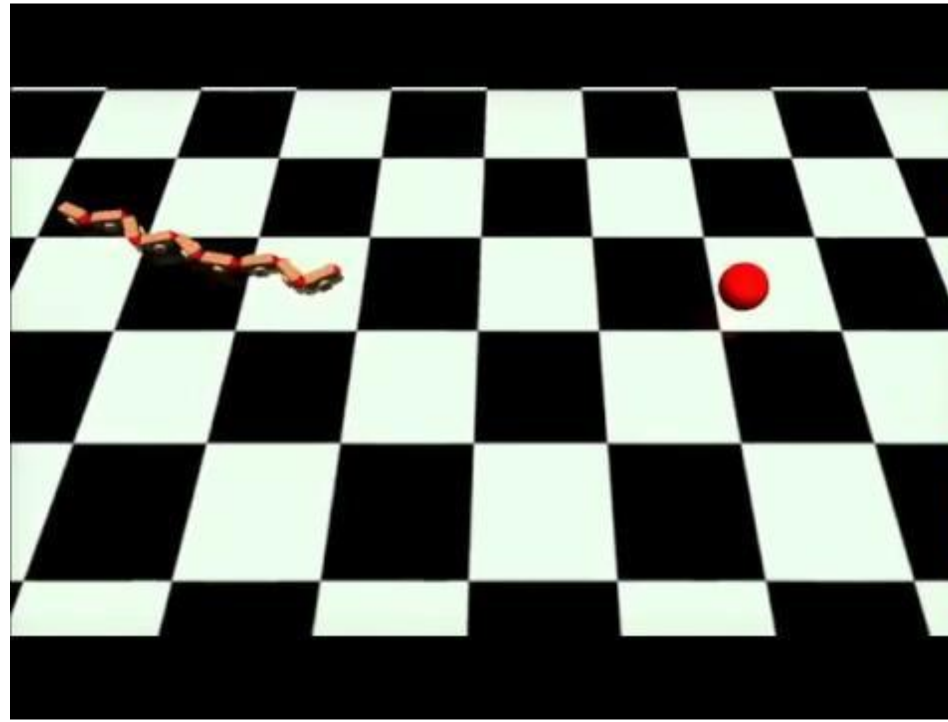


Mean:	3.88 m
Std:	0.22 m

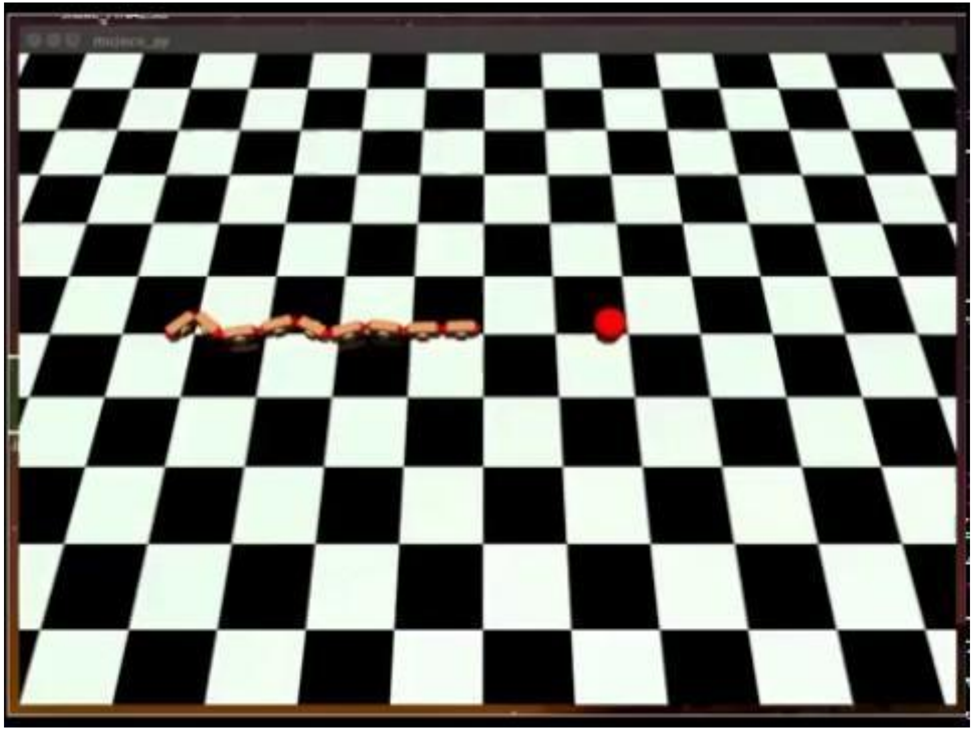
2. Experiment: Autonomous Target Tracking Result



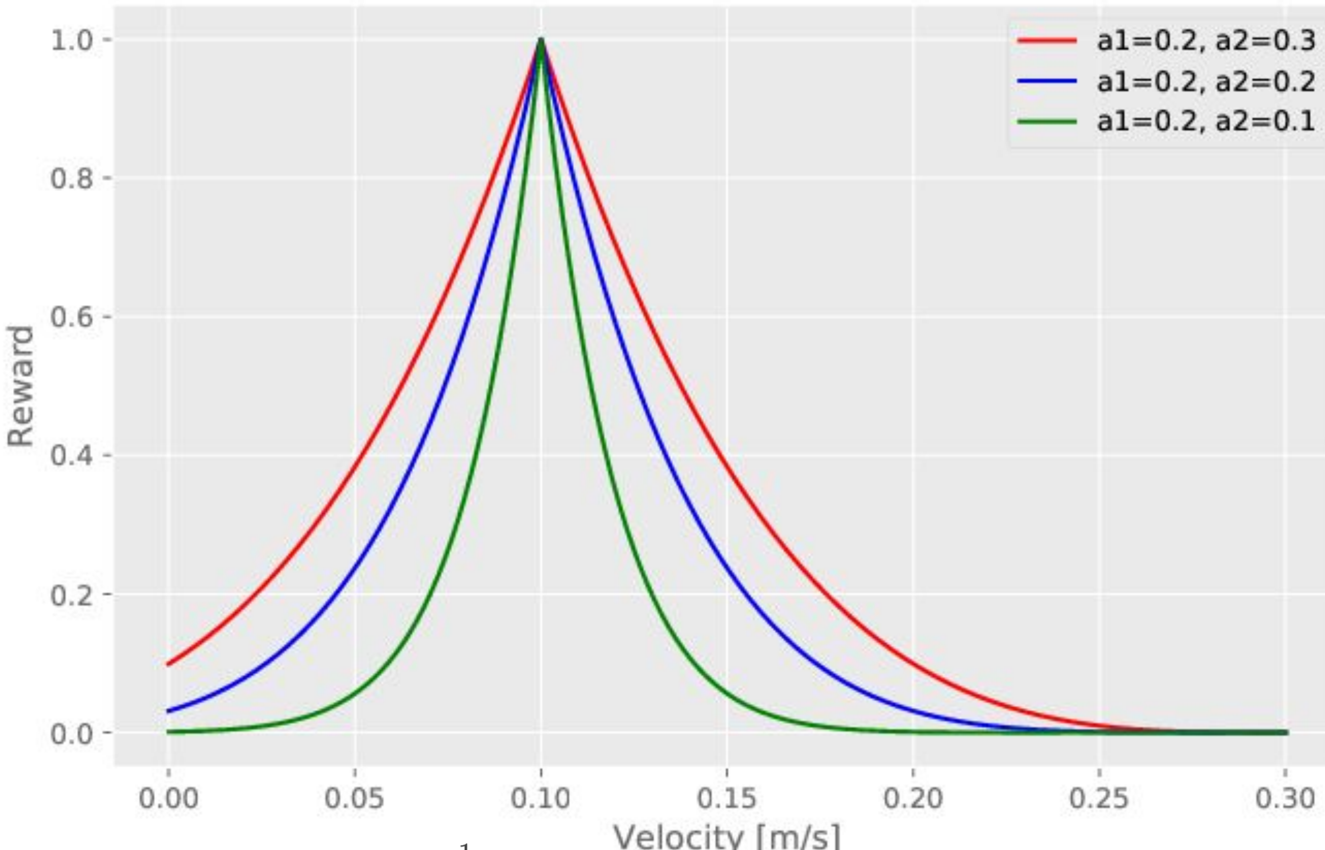
- Stops to keep distance
- Uses different
- Path following behavior
- Does not face the target directly



Learning process

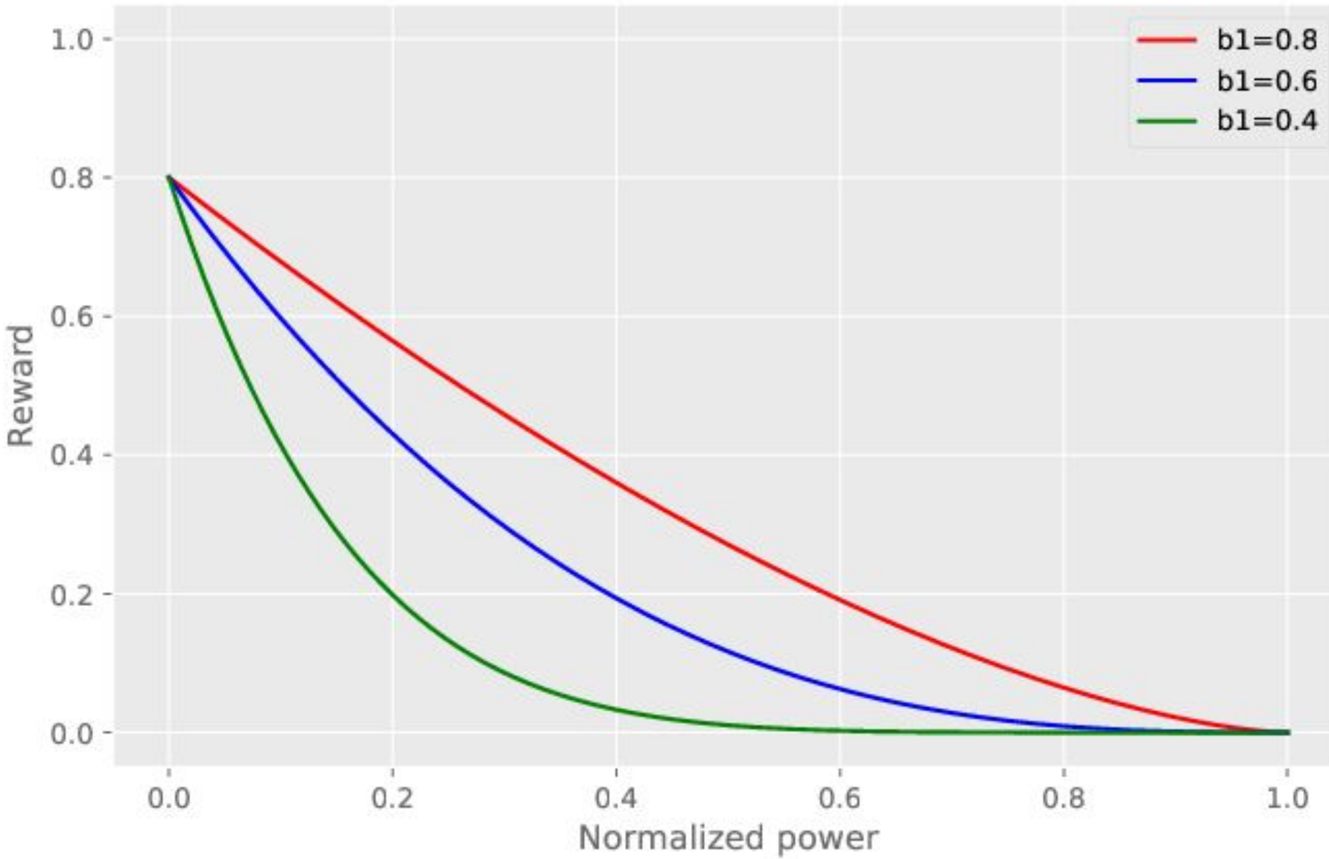


1. Experiment: Autonomous Locomotion Control
Reward Function



$$r_v = \left(1 - \frac{|v_t - v|}{a_1}\right)^{\frac{1}{a_2}}$$

1. Experiment: Autonomous Locomotion Control
Reward Function



$$r_P = r_{max} \left|1 - \hat{P}\right|^{b_1^{-2}}$$

Training 1

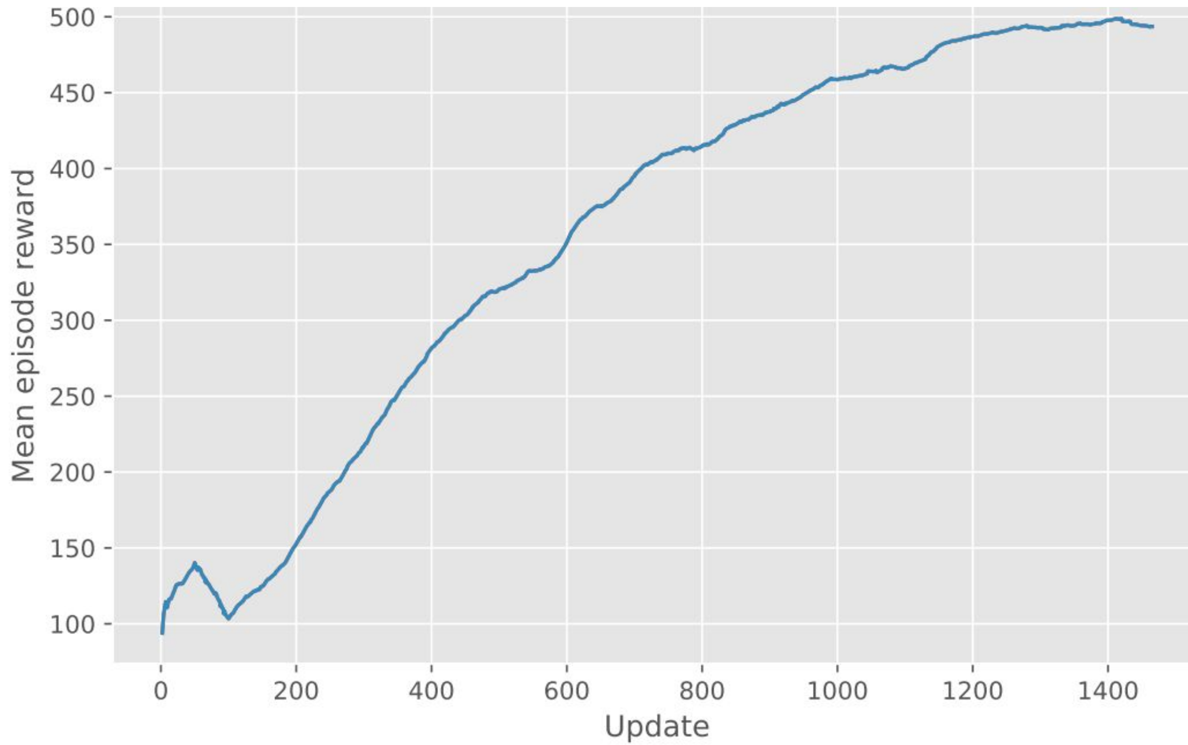


Figure 5.5: This plot shows the learning curve of the PPO controller. Mean episode reward over 3 million time steps. The x-axes represent the number of network updates and the y-axis the achieved mean episode reward per update. Note that an update contains 2048 time steps.

Training 2

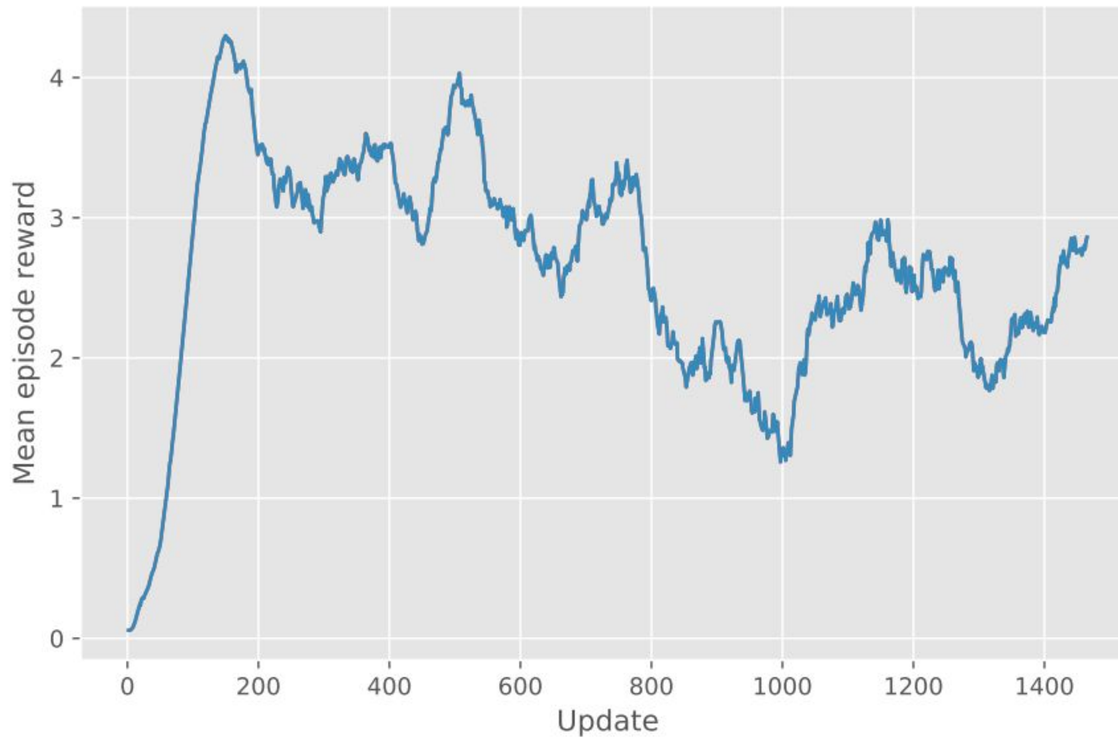


Figure 6.4: The learning curve of the autonomous target tracking model. It is trained with 3 million time steps with 1000 time steps per episode and 1024 time steps per update. The mean episode reward does not converge to a specific reward.