Legged Robots Practical: Project 2

16.11.2021

Plan

- **W9 16.11.2021:** Quadruped Central Pattern Generators
  + Deep Reinforcement Learning
- **W10 23.11.2021:**
- **W11 30.11.2021:**
- **W12 07.12.2021:**
- **W13 14.12.2021:** Exam
- **W14 21.12.2021:** Competition (Details to come)

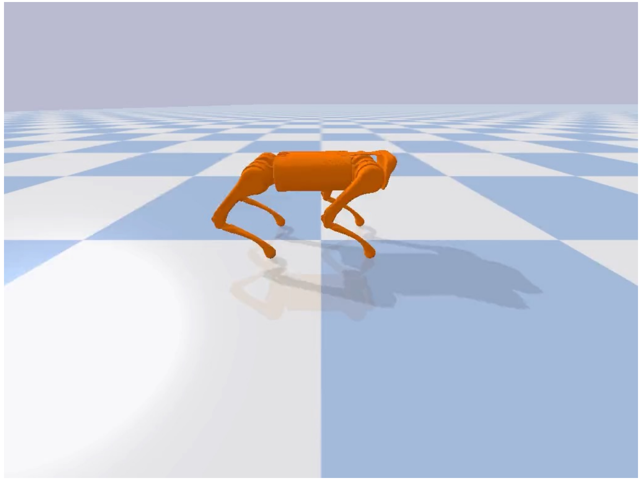
[Report 2 - Quadruped] - 30% of course grade

Quadruped Locomotion with Central Pattern Generators and Deep Reinforcement Learning

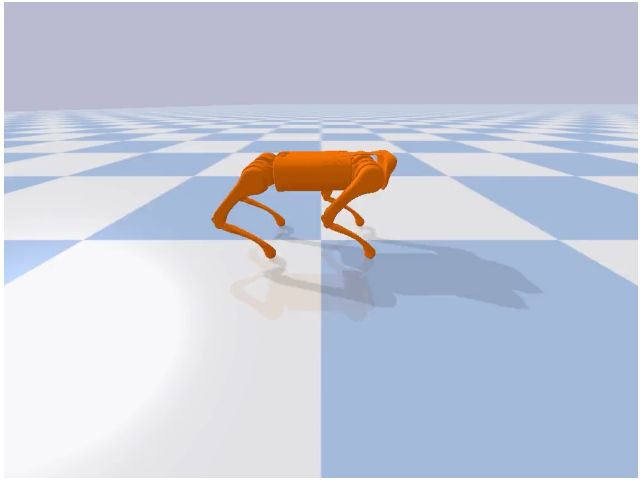
Legged Robots

Part 1: Central Pattern Generators

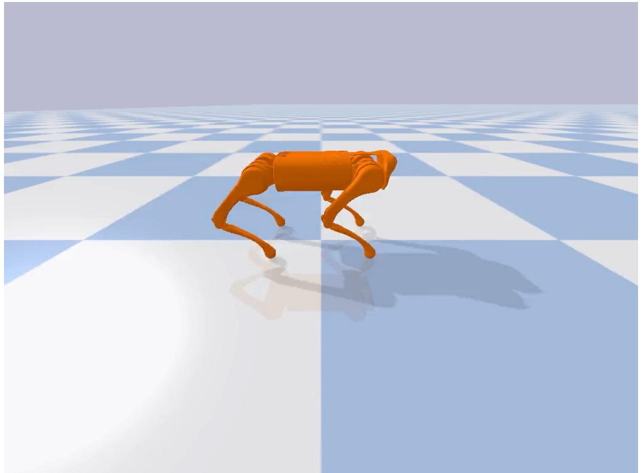
Trot



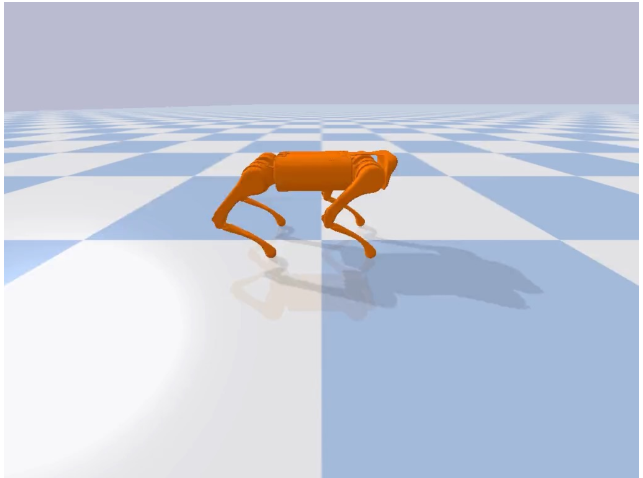
Bound



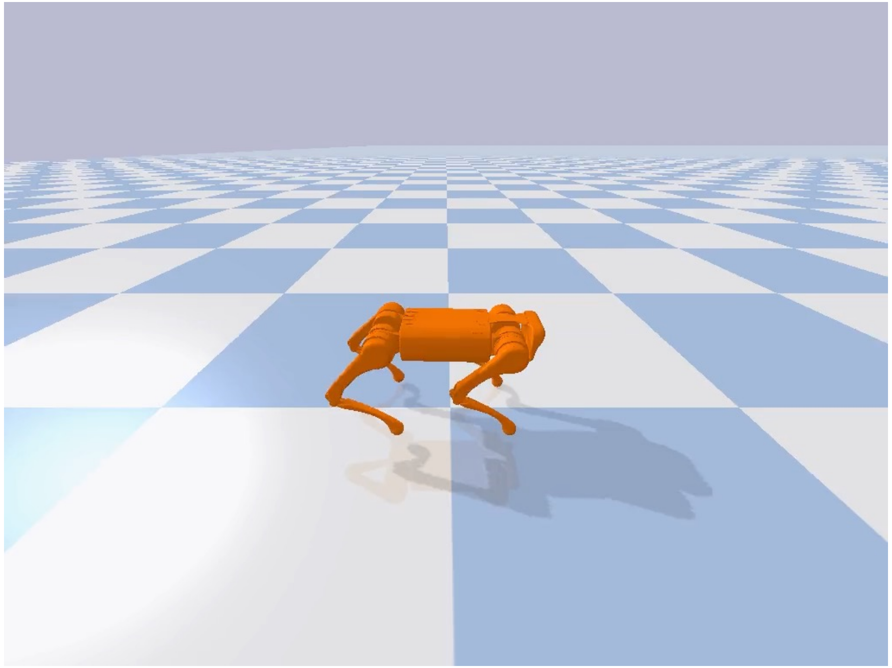
Pace

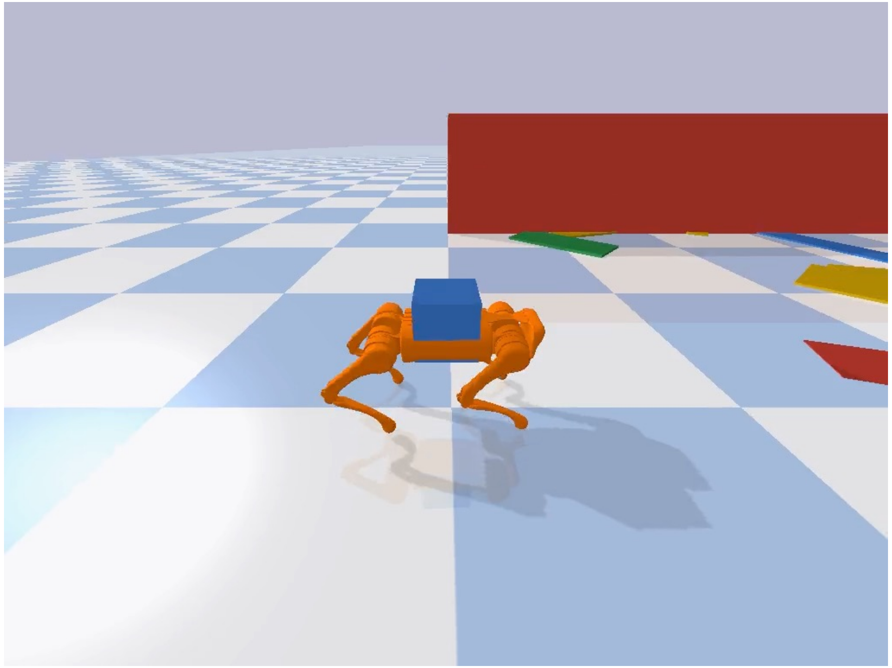


Walk

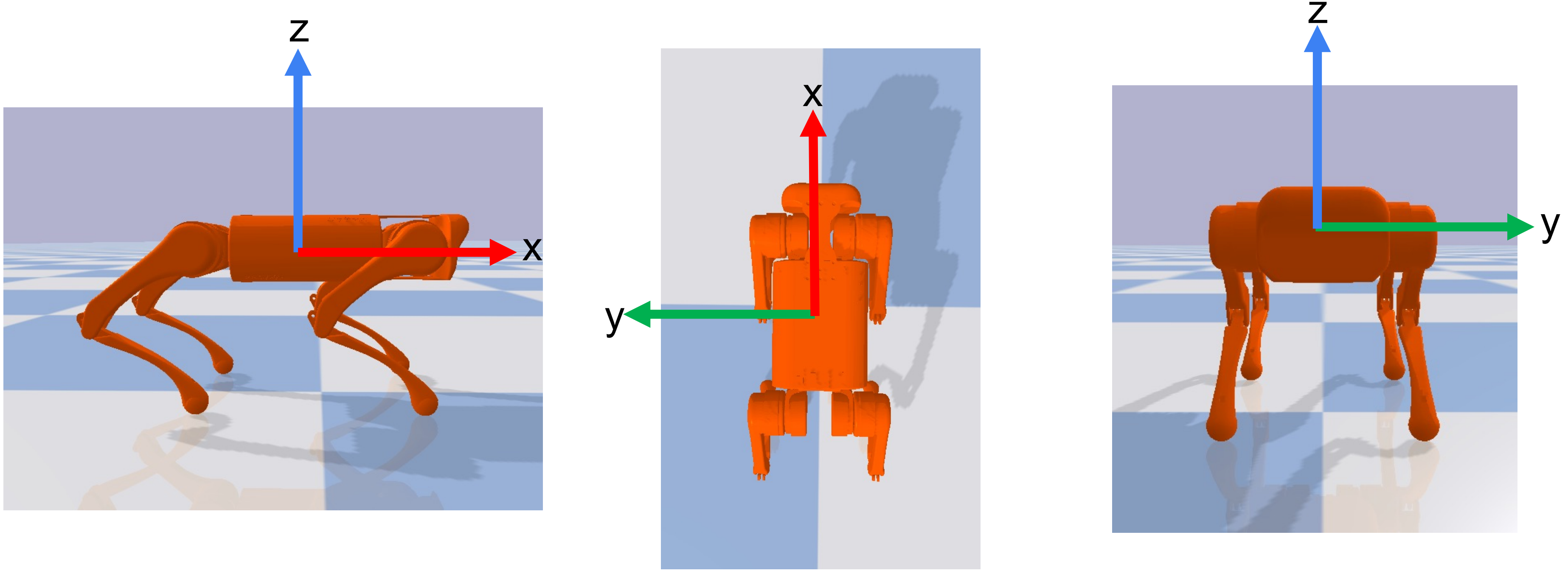


Part 2: Deep Reinforcement Learning





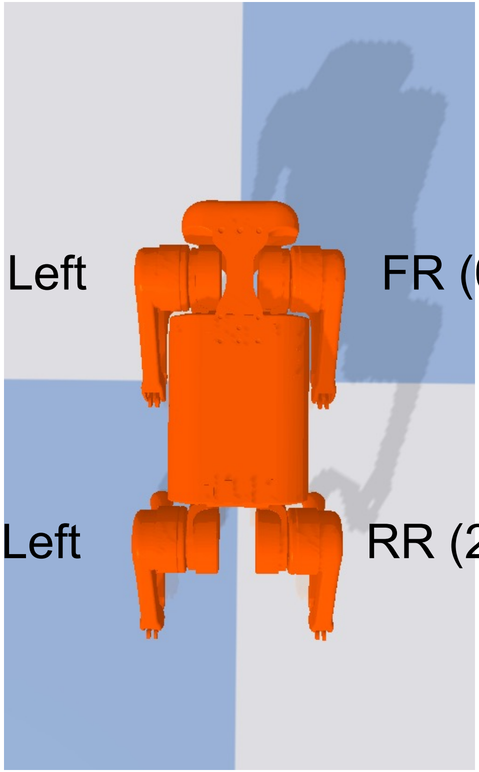
Quadruped Model Reference Frame



Quadruped Model Leg References

FL (1): Front Left

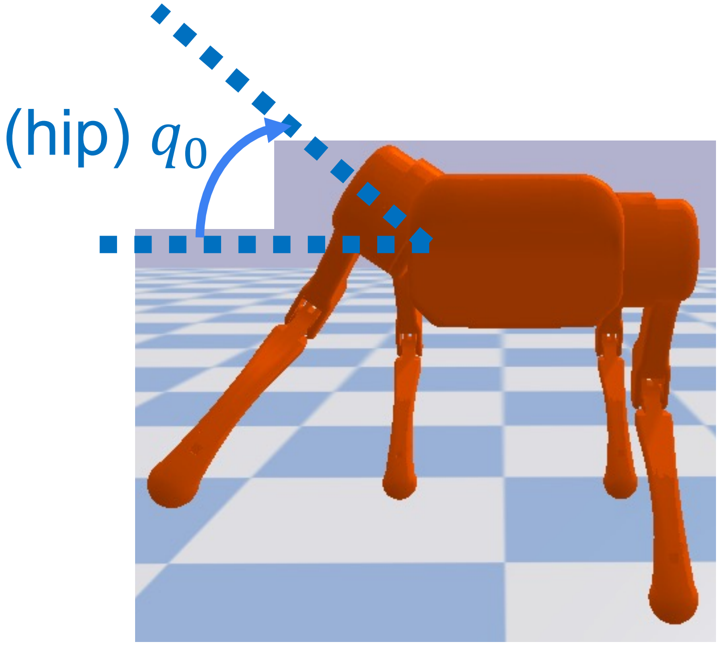
FR (0): Front Right

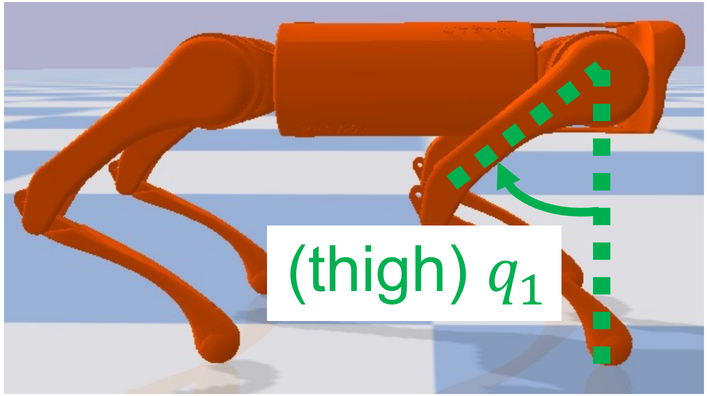


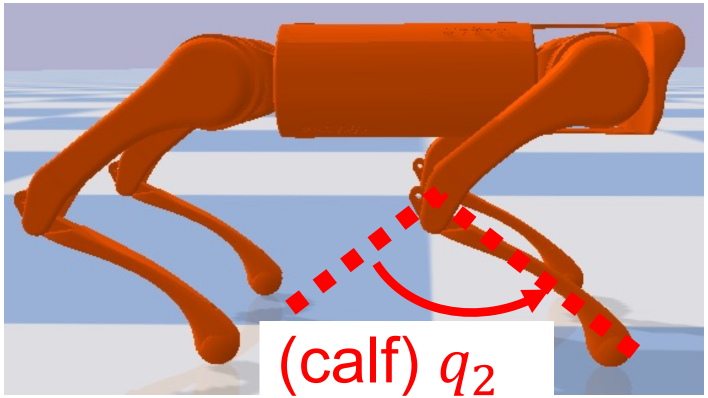
RL (3): Rear Left

RR (2): Rear Right

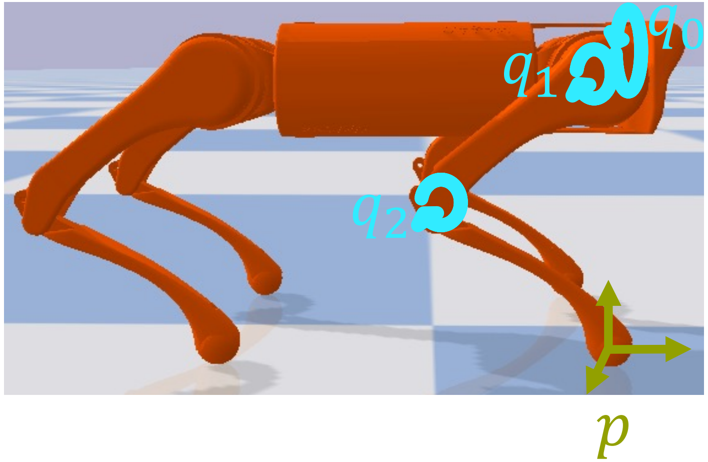
Quadruped Model Joint References





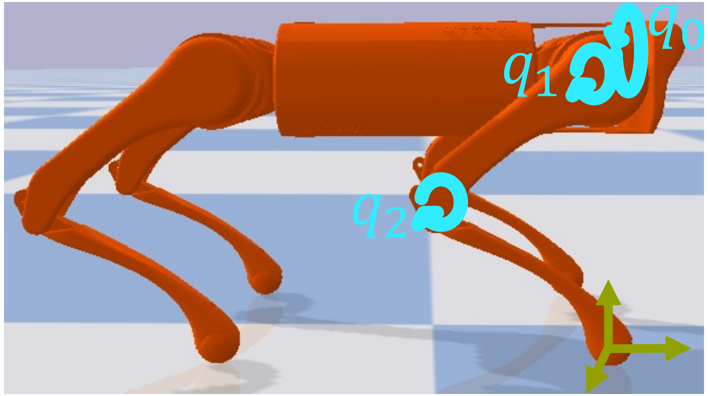


Joint angles ←→ Cartesian space (in leg frame)



$p = f(q)$	Forward kinematics
$q = f^{-1}(p)$	Inverse kinematics
$\dot{p} = v = J(q)\dot{q}$	Foot linear velocity
$\tau = J^T(q)F$	Map desired end effector force to torques

Joint angles ←→ Cartesian space (leg frame control)



p

$p = f(q)$ Forward kinematics

$q = f^{-1}(p)$ Inverse kinematics

$\dot{p} = v = J(q)\dot{q}$ Foot linear velocity

$\tau = J^T(q)F$ Map desired end effector force to torques

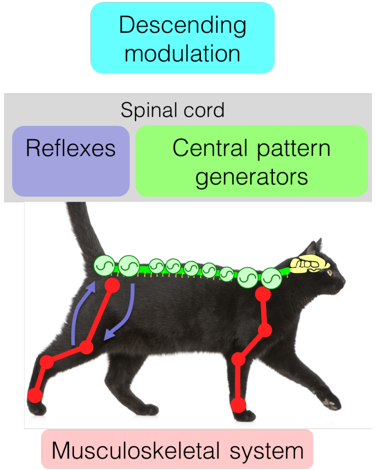
$$\tau_{joint} = K_{p,joint}(q_d - q) + K_{d,joint}(\dot{q}_d - \dot{q})$$ Joint PD

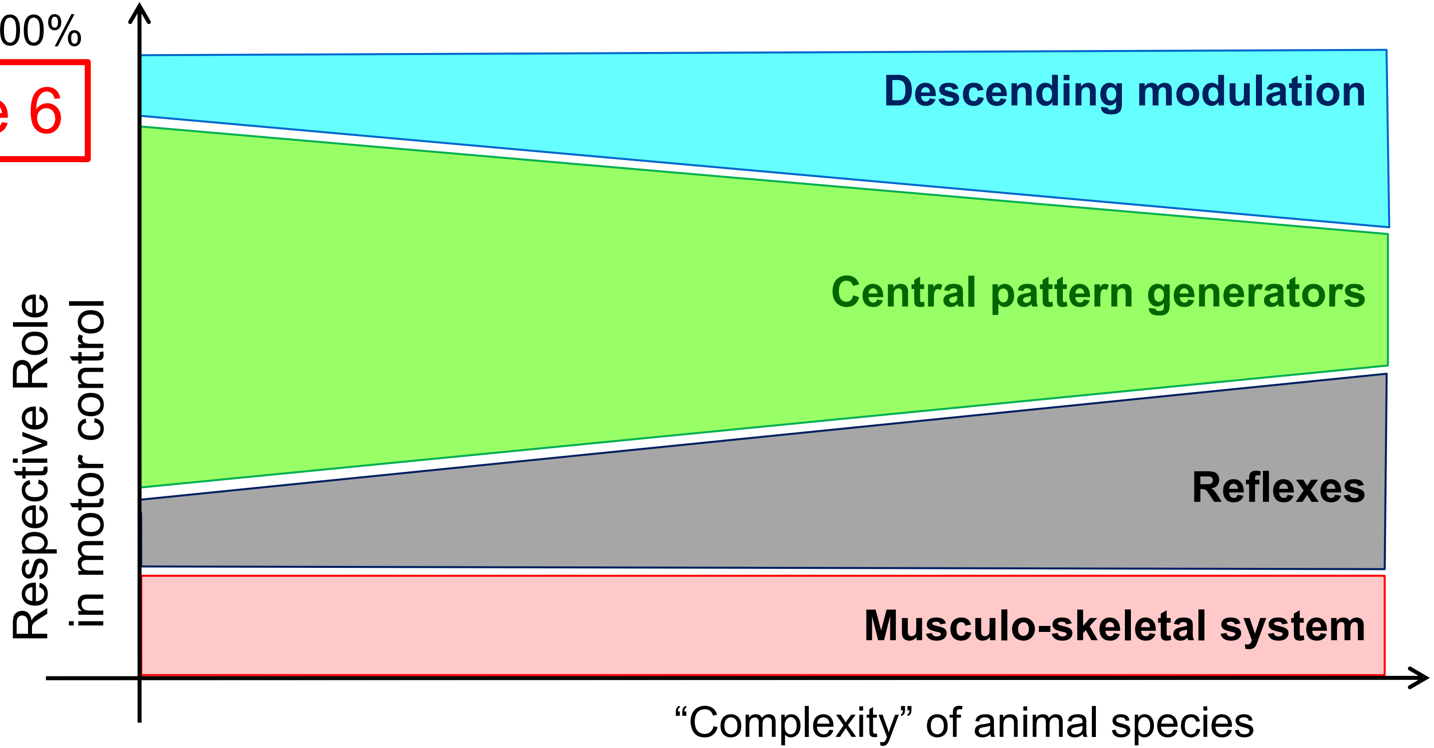
$$\tau_{Cartesian} = J^T(q)\left[K_{p,Cartesian}(p_d - p) + K_{d,Cartesian}(v_d - v)\right]$$ Cartesian PD

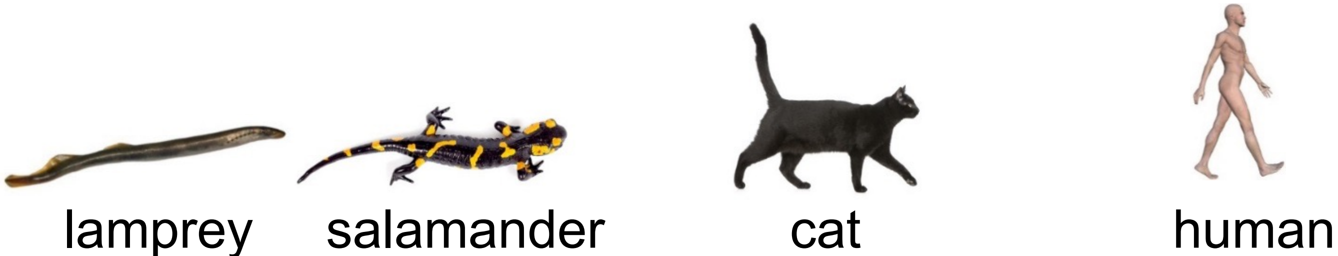
$$\tau_{final} = \tau_{joint} + \tau_{Cartesian}$$ Contributions from both joint PD and Cartesian PD

Central Pattern Generators: Review

From Lecture 6



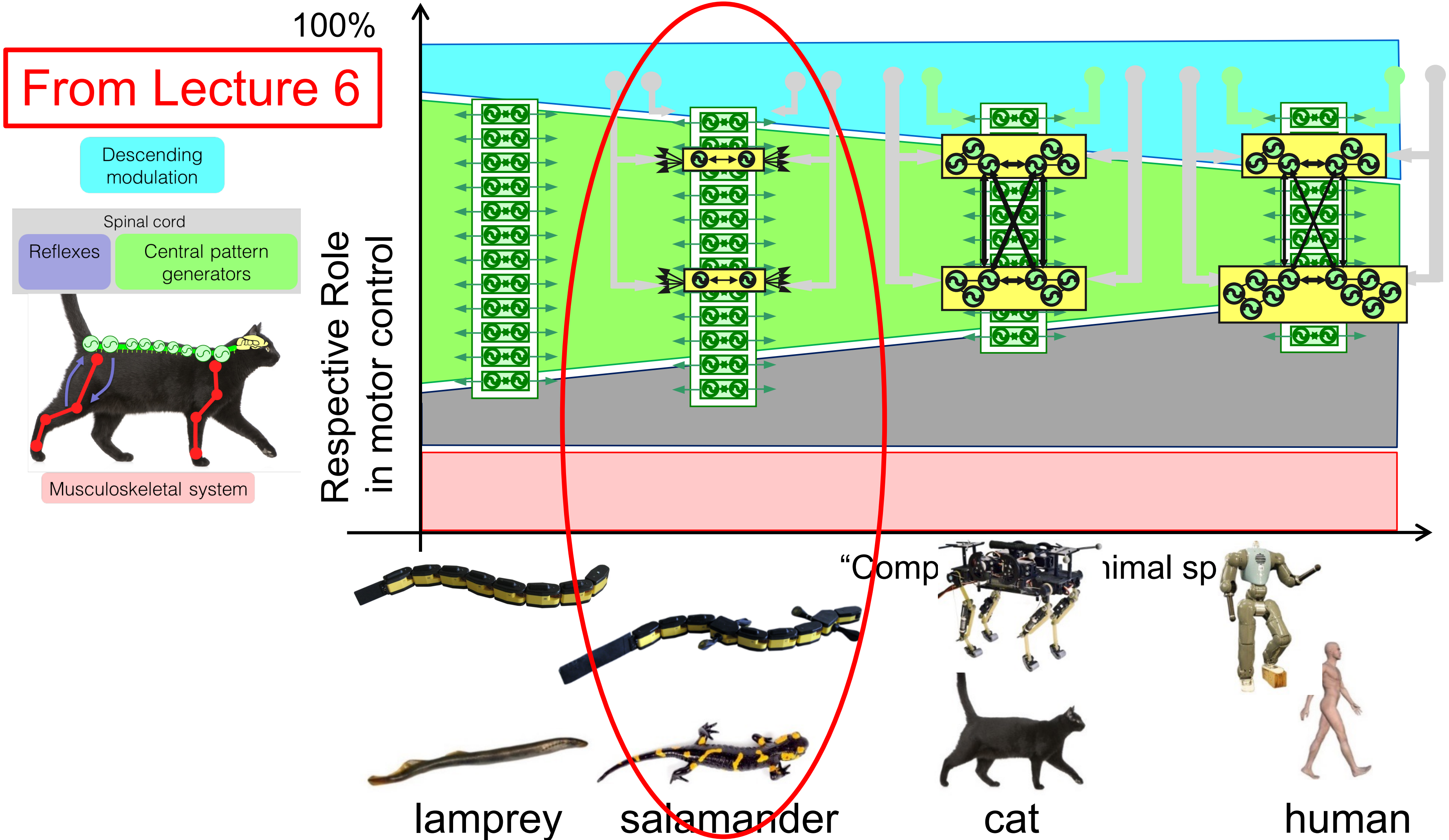




Ryczko, Simon, Ijspeert, Trends in Neuroscience, 2020

Minassian et al Neuroscientist 2017

From Lecture 6



Modeling the CPG with coupled oscillators

A segmental oscillator is modeled as an amplitude-controlled phase oscillator as used in (Cohen, Holmes and Rand 1982, Kopell, Ermentrout, and Williams 1990) :

Phase:

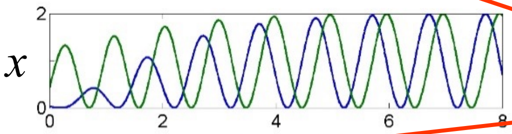
$$\dot{\theta}_i = 2\pi \nu_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij})$$

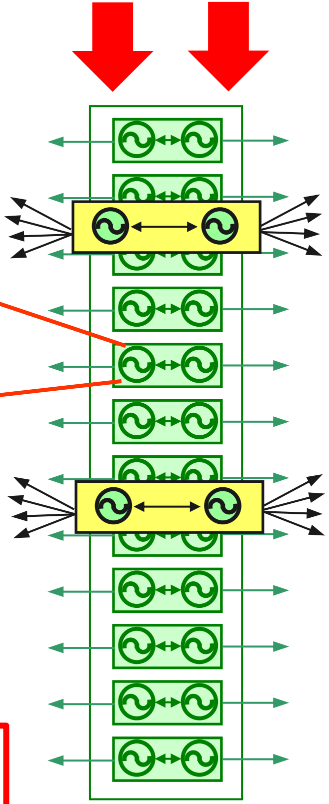
Amplitude:

$$\ddot{r}_i = a_i \left(\frac{a_i}{4} (R_i - r_i) - \dot{r}_i \right)$$

Output:

$$x_i = r_i (1 + \cos(\theta_i))$$





Setpoints:

$$\varphi_i = x_i - x_{N+i} \quad \textit{for the axial motors}$$

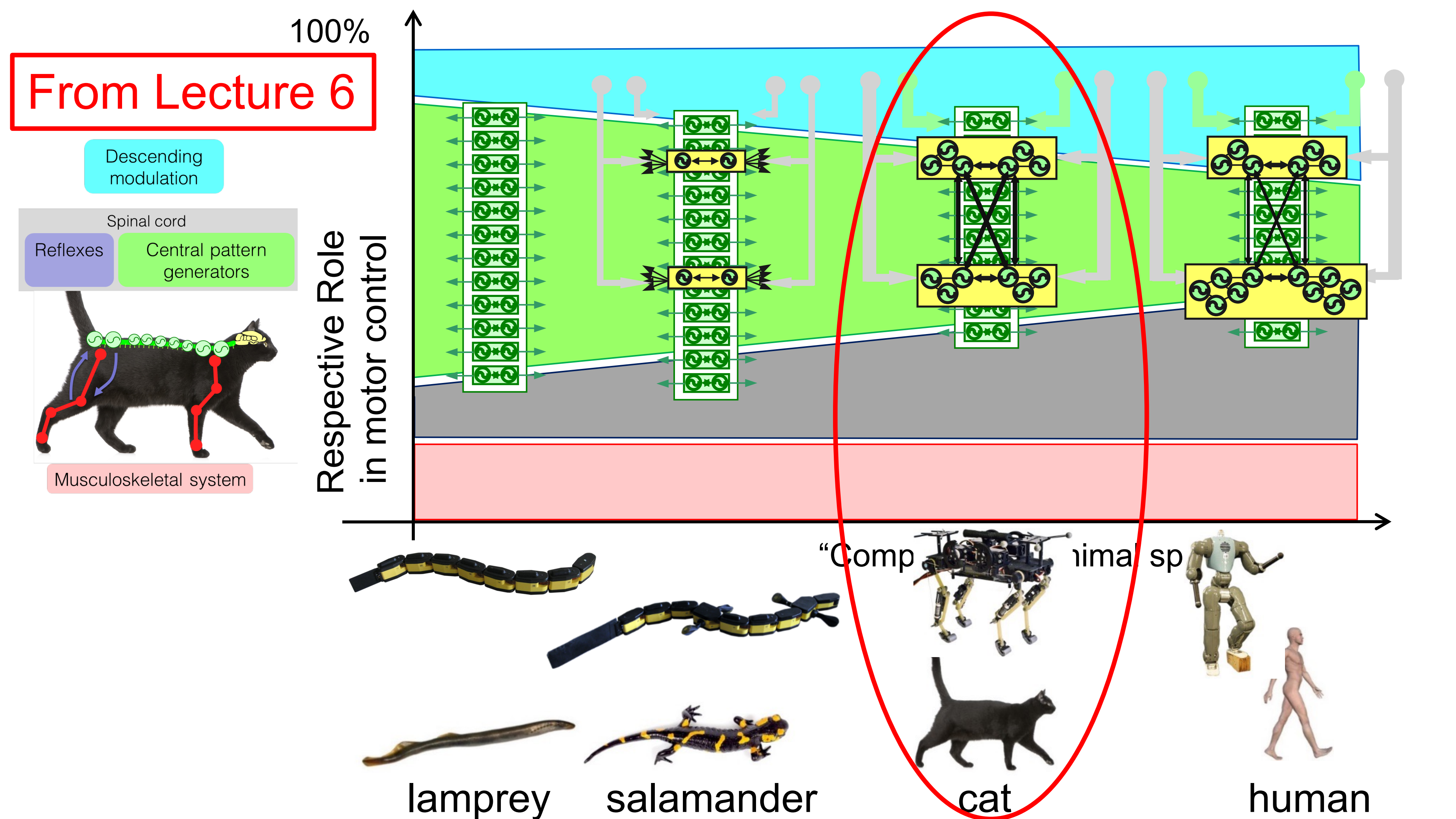
$$\varphi_i = f(\theta_i) \quad \textit{for the (rotational) limb motors}$$

From Lecture 6

[Ijspeert *et al*, *Science*, March 2007].

CPGs can modulate speed, heading, and type of gait under the modulation of a few drive signals

Ijspeert *et al*, *Science*, 2007, Crespi et al, *IEEE TRO*, 2013.



From Lecture 6
100%
Descending modulation
Spinal cord
Reflexes
Central pattern generators
Musculoskeletal system
Respective Role in motor control
lamprey
salamander
cat
human

Modeling the CPG with coupled oscillators (Quadruped)

Amplitude:

$$\dot{r}_i = \alpha(\mu - r_i^2) r_i$$

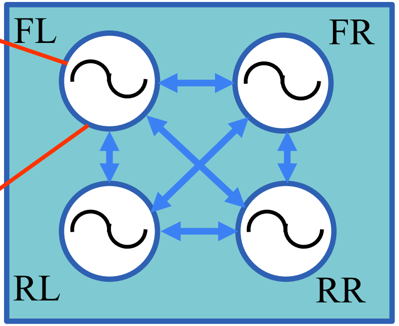
Phase:

$$\dot{\theta}_i = \omega_i + \sum_{j=0}^{3} r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij})$$

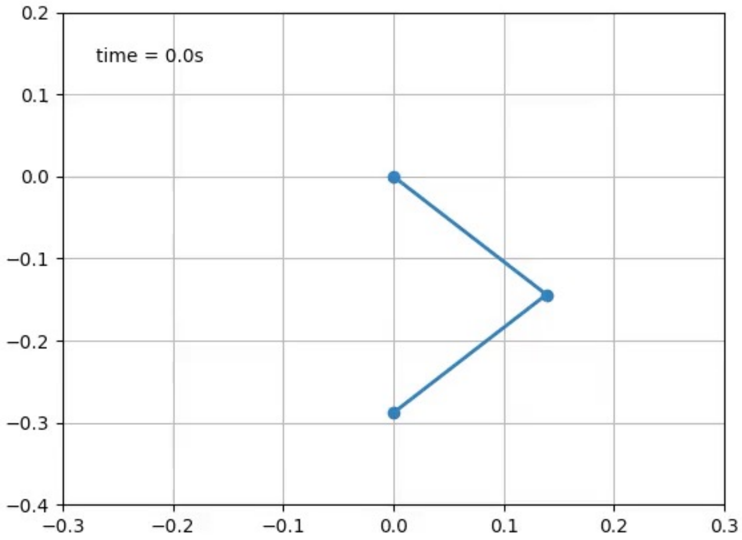
Output:

$$x_{\text{foot}} = -d_{step} r_i \cos(\theta_i)$$

$$z_{\text{foot}} = \begin{cases} -h + g_c \sin(\theta_i) & \text{if } \sin(\theta_i) > 0 \\ -h + g_p \sin(\theta_i) & \text{otherwise} \end{cases}$$



Mapping CPG States to Foot Positions with Inverse Kinematics



$$\dot{r}_i = \alpha(\mu - r_i^2)r_i$$

$$\dot{\theta}_i = \omega_i$$

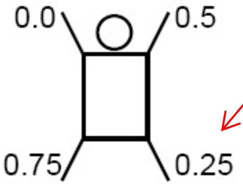
$$x_{\text{foot}} = -d_{step} r_i \cos(\theta_i)$$

$$z_{\text{foot}} = \begin{cases} -h + g_c \sin(\theta_i) & \text{if } \sin(\theta_i) > 0 \\ -h + g_p \sin(\theta_i) & \text{otherwise} \end{cases}$$

Gait Terminology

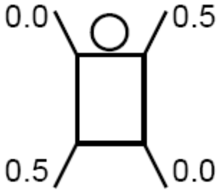
From Lecture 2

- *Stride duration* = the duration of a complete cycle (the period)
- *Swing phase* of a limb (period during which the limb is off the ground)
- *Stance phase* (period during which the limb touches the ground)
- *Duty factor* = Stance duration / Stride duration



Lateral sequence walk

Time of contact with ground within a whole cycle

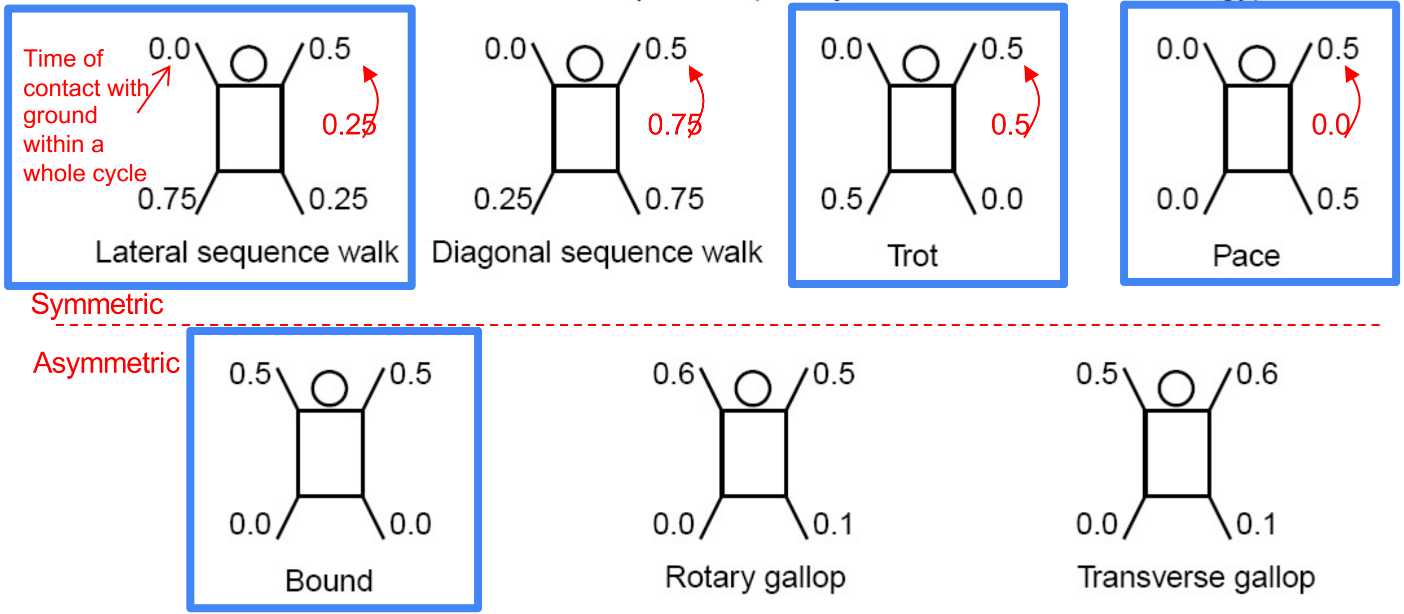


Trot

Most common quadruped gaits

From Lecture 2

Classification in terms of the footfall sequences (mainly used in mathematical biology)



This project

$$\dot{r}_i = \alpha(\mu - r_i^2) r_i$$

$$\dot{\theta}_i = \omega_i + \sum_{j=0}^{3} r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij})$$

What should $\boldsymbol{\phi}$ be for each gait?

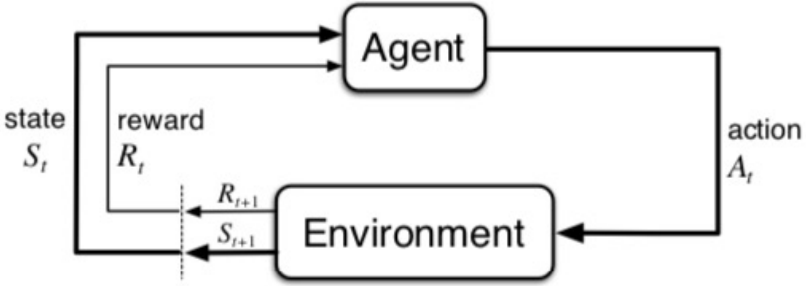
Deep Reinforcement Learning: Review

From Lecture 7

Reinforcement Learning

An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



- Return over a trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots)$

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

- Policy $\pi(a_t|s_t)$ maps from states s_t to actions a_t (Goal: find policy maximizing above return)
- Value function: $V^\pi(s) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s]$
- Action-value function: $Q^\pi(s,a) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a]$
- Advantage function: $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

R. Sutton and A. Barto. Introduction to Reinforcement Learning. MIT Press 1998
Deep RL Bootcamp. Berkeley CA, August 2017

From Lecture 7

Many Existing Tools for Reinforcement Learning

- RL algorithm implementations
  - stable-baselines3 https://github.com/DLR-RM/stable-baselines3 PPO, SAC
  - ray[rllib] https://github.com/ray-project/ray
  - spinningup https://github.com/openai/spinningup
  - tianshou https://github.com/thu-ml/tianshou/
  - … many others!
- Physics simulators
  - pybullet https://github.com/bulletphysics/bullet3
  - MuJoCo https://mujoco.org
  - RaiSim https://raisim.com
  - Isaac-Gym https://developer.nvidia.com/isaac-gym
  - … and others!

From Lecture 7

RL Considerations

Algorithm

- On/off policy
- Hyperparameters
- Network architecture
- Random seeds/trials

…implementation dependent!

MDP Design Decisions

- Observation space
- Action space
- Reward function

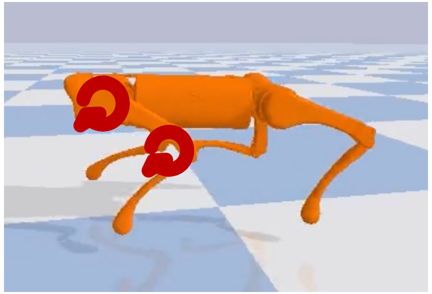
Environment Parameters

- Simulator dynamics
- Control gains – joint/Cartesian
- Control/environment time step
- Noise, latency

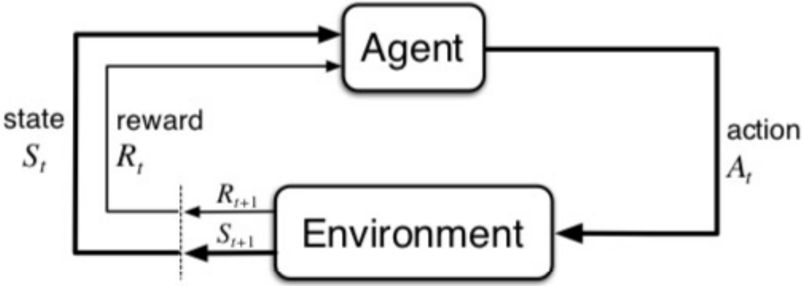
P. Henderson et al. *Deep Reinforcement Learning that Matters*. arXiv:1709.06560, 2017

From Lecture 7

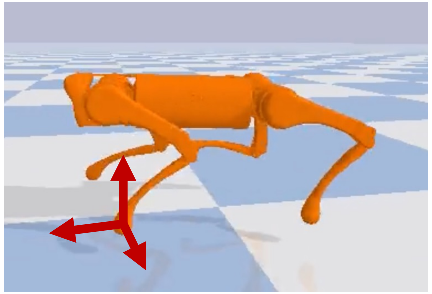
State/Action/Reward Space: A1



s_t ? i.e.
-body (z, r, p, y)
-body velocities
-joint states



a_t ?
-motor positions/torques
-Cartesian PD



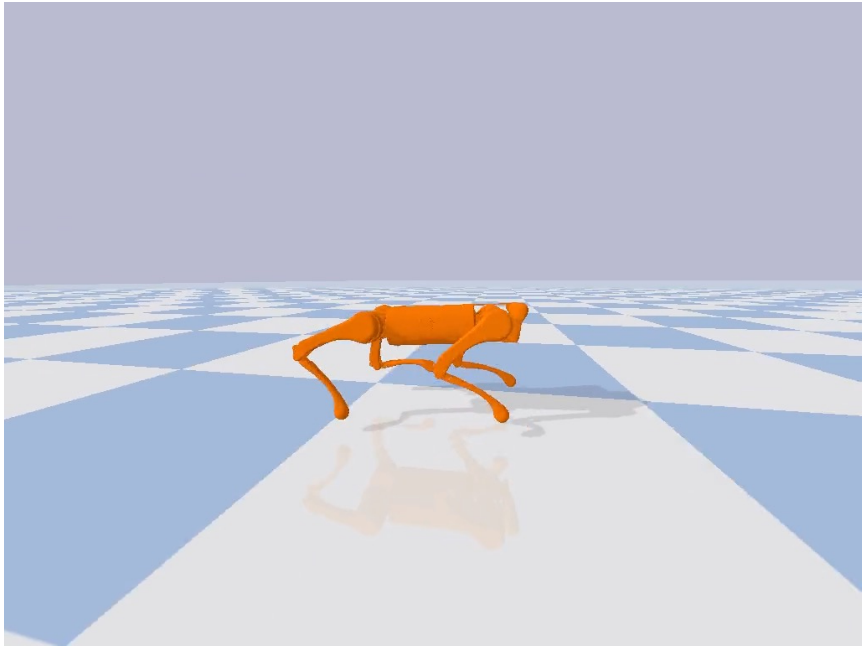
r_t ? i.e.
-body linear velocity
-energy penalty

This project: construct the MDP

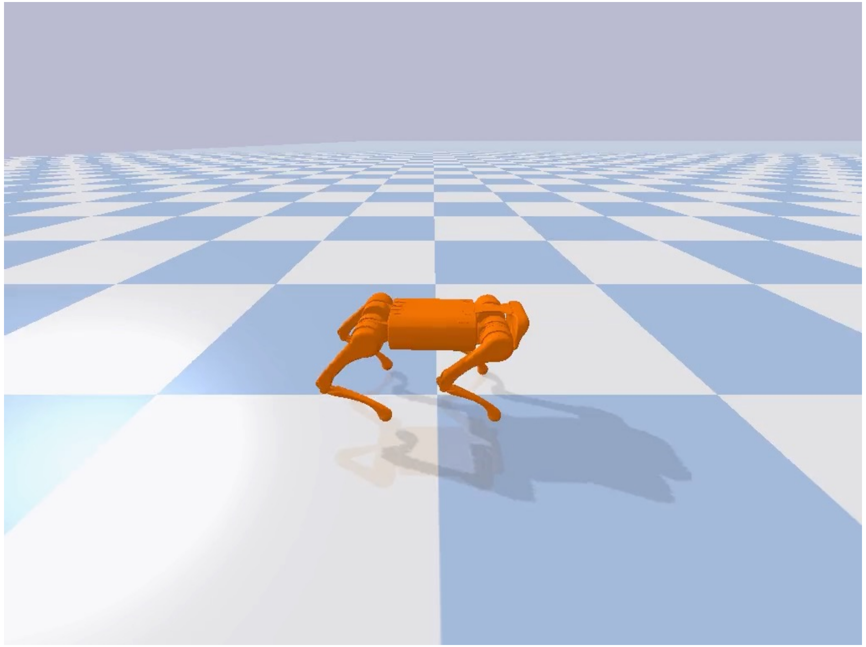
From Lecture 7

Joint Position Control vs. Cartesian PD Control (PPO/SAC)

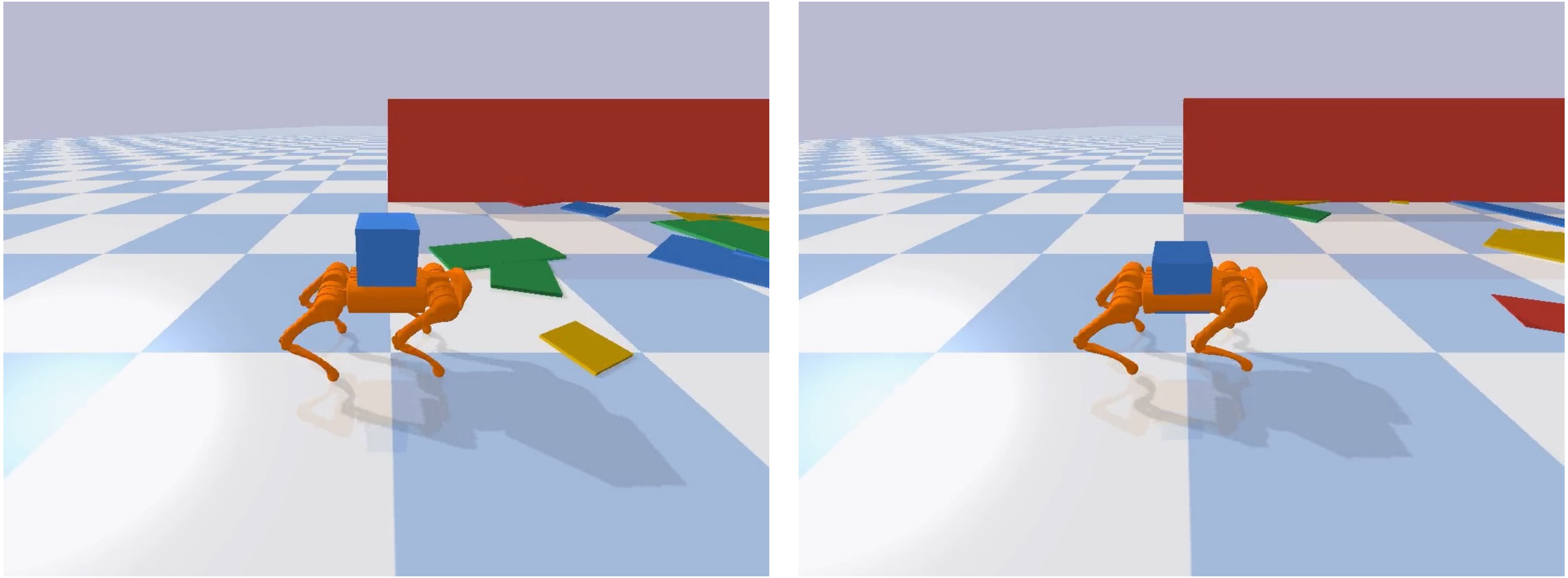
Action Space: $a_t = q_{1 \ldots N}$



Action Space: $a_t = [x_{ee_i}, y_{ee_i}, z_{ee_i}]$



How robust is your approach? To be determined at the 21.12.2021 competition



Tips

- Monitor episode length and reward mean during training
- Training should complete within 1 million timesteps for reasonable observation space, action space, and reward function choices (with no noise in the environment)
- No training on test environment (used for competition)
- Start training early!