Legged Robots: Project 2

28.10.2025

Plan

Fundamentals (ungraded)

- **W1-3** (09.09, 16.09, 23.09)
  - Introduction + double pendulum kinematics and dynamics
  - Jacobian (Cartesian PD + Force Control)
  - Inverse Kinematics (compare with force control)
  - Single-leg hopping

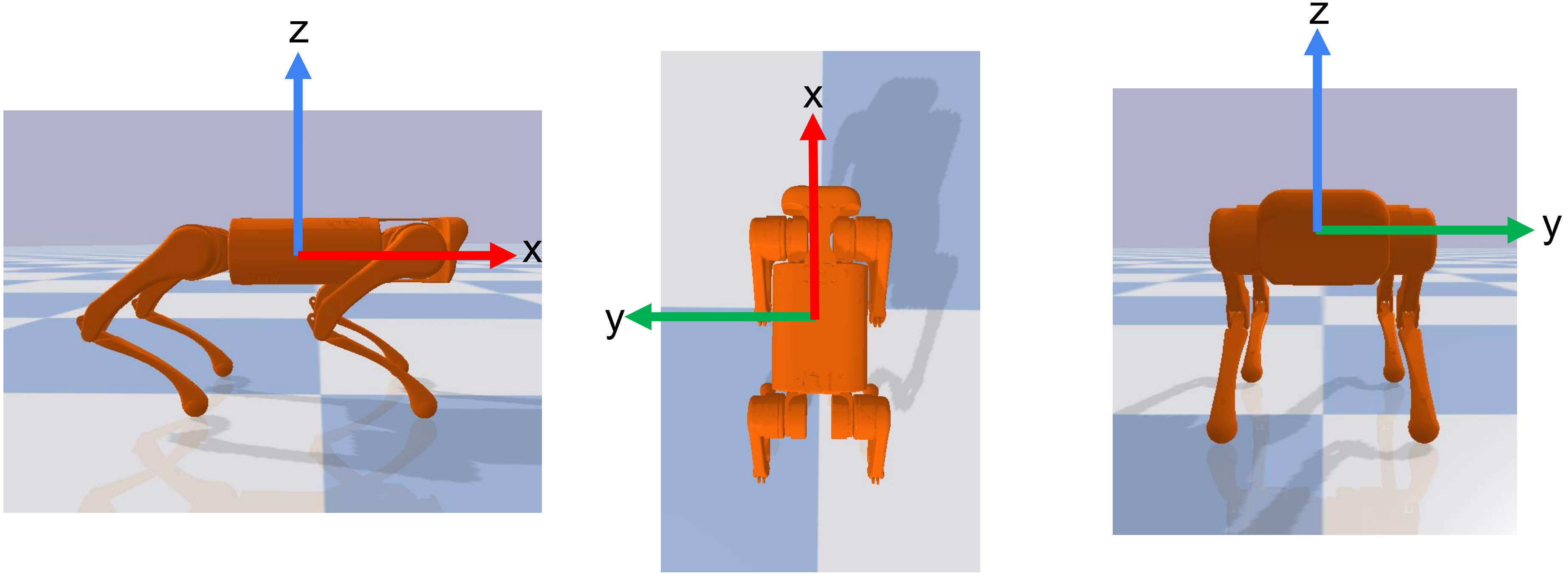
Mini-Project 1

- **W4-6** (30.09, 07.10, 14.10)
  - Model-based control of a quadruped
  - **[MP1 Report – 15% of grade]**

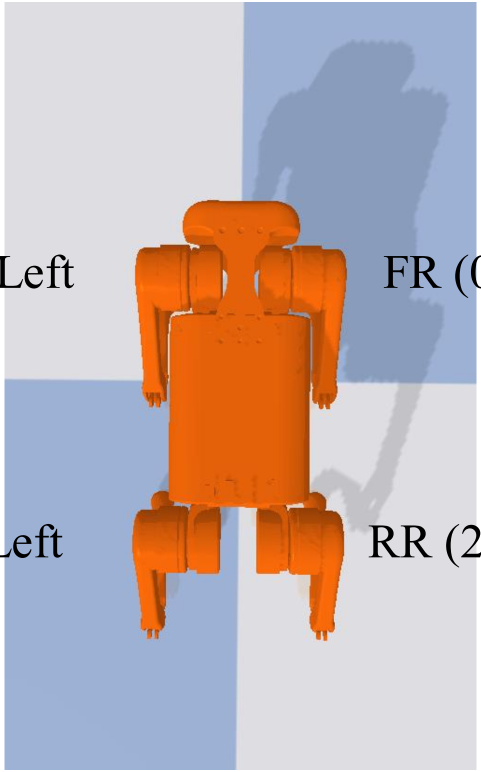
Mini-Project 2

- **W7-14** (29.10-17.12)
  - Quadruped CPG Trot
  - Quadruped Locomotion Project (CPGs, Deep RL)
  - **[MP2 Report – 35% of grade]**
  - **[Article Presentation – 20% of grade]**

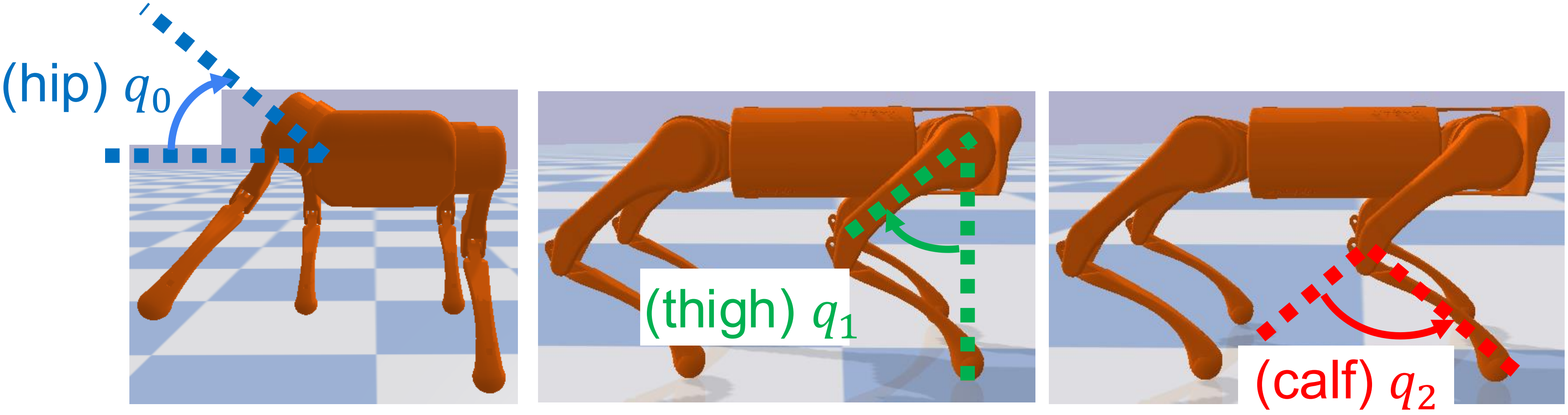
Quadruped Model Reference Frame



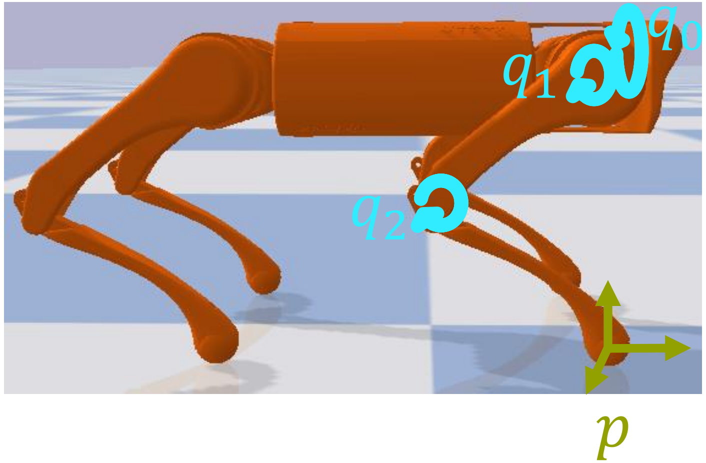
Quadruped Model Leg References



Quadruped Model Joint References



Joint angles ←→ Cartesian space (leg frame control)



$p = f(q)$	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

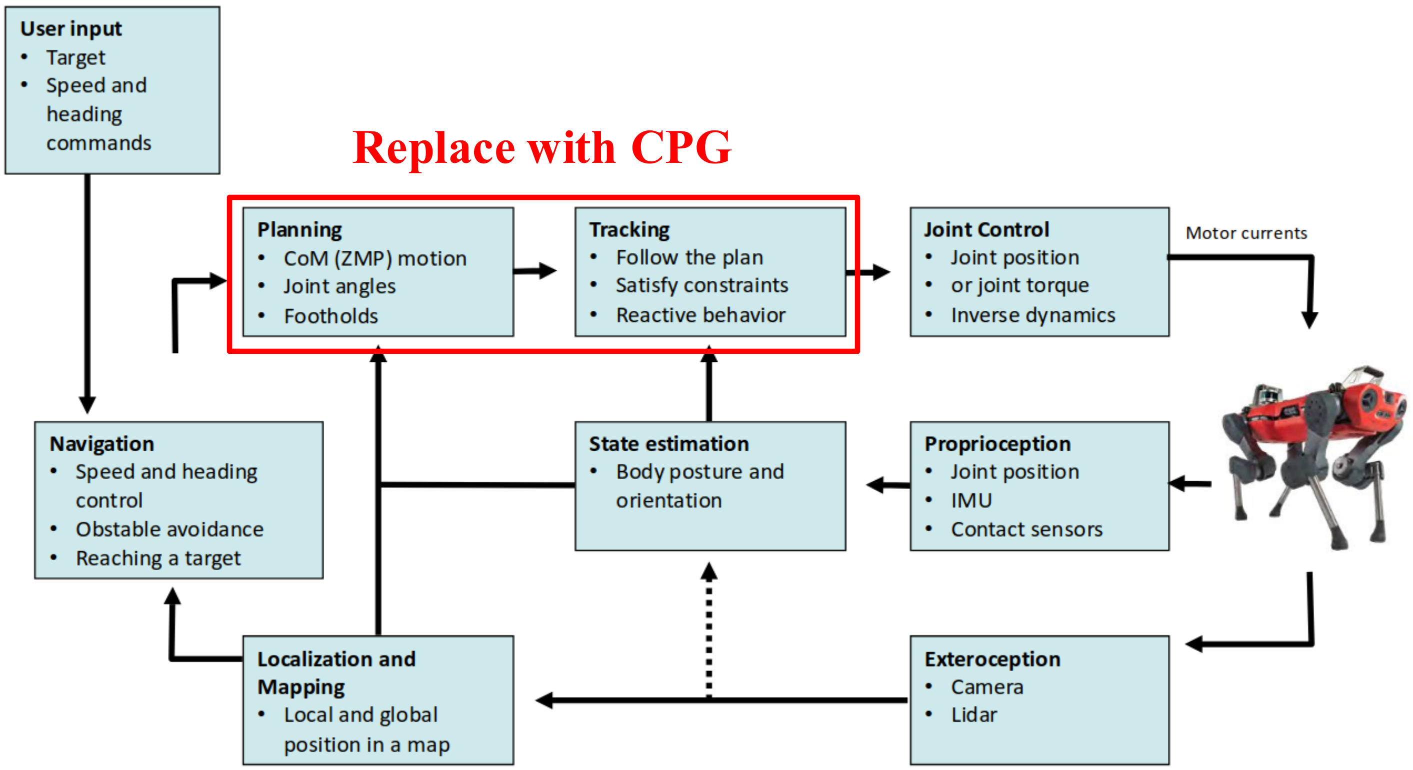
$$\tau_{joint} = K_{p,joint}(q_d - q) + K_{d,joint}(\dot{q}_d - \dot{q})$$ Joint PD

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

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

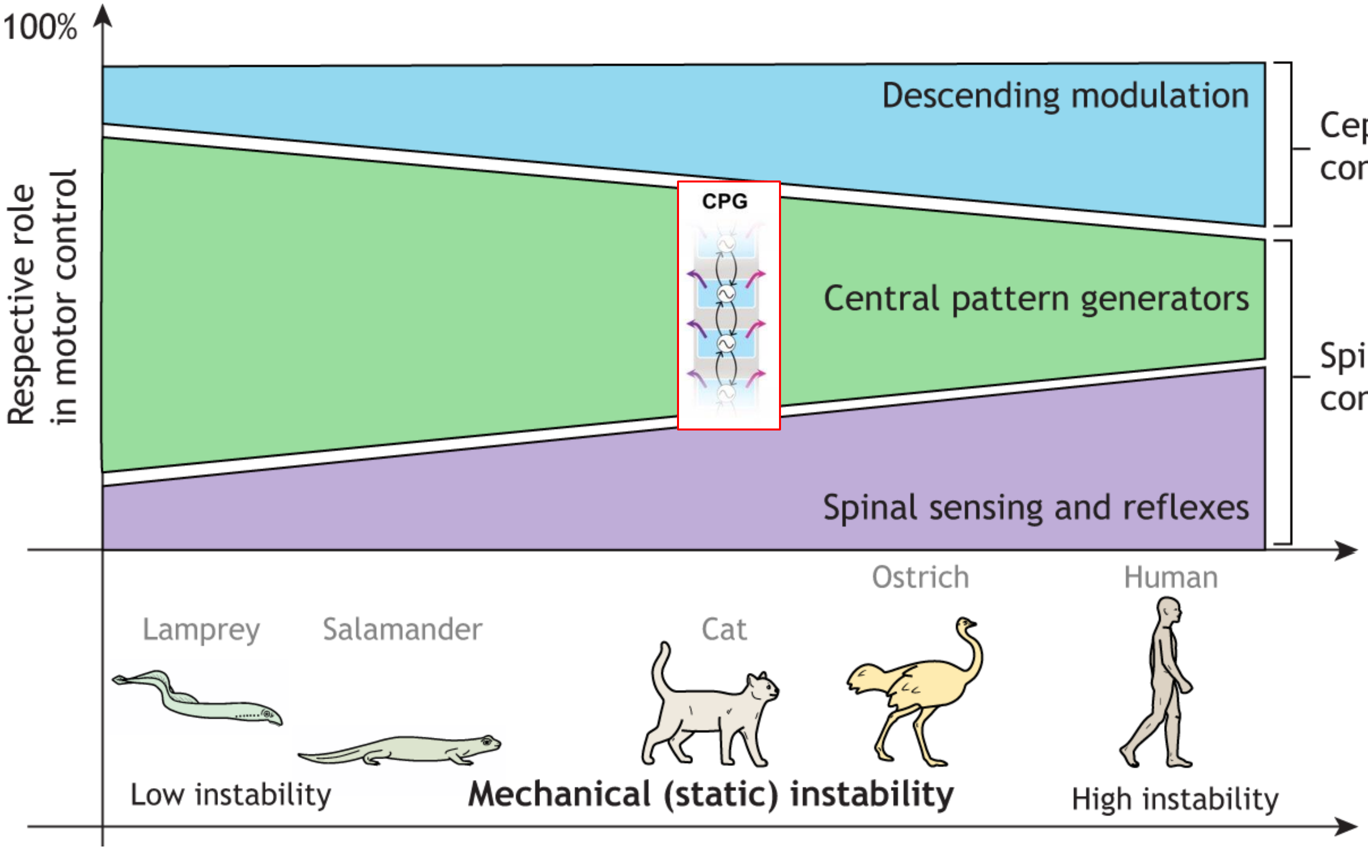
Typical control architecture

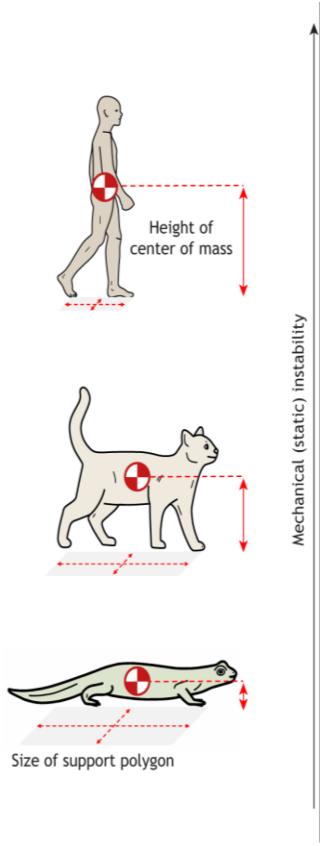
Lecture 3



Part 1: Central Pattern Generators (CPGs)

Lecture 4





Modeling the CPG with coupled oscillators (Quadruped)

Amplitude:

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

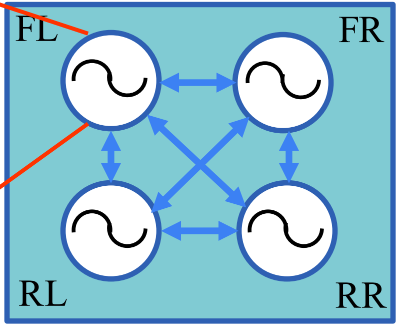
Phase:

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

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

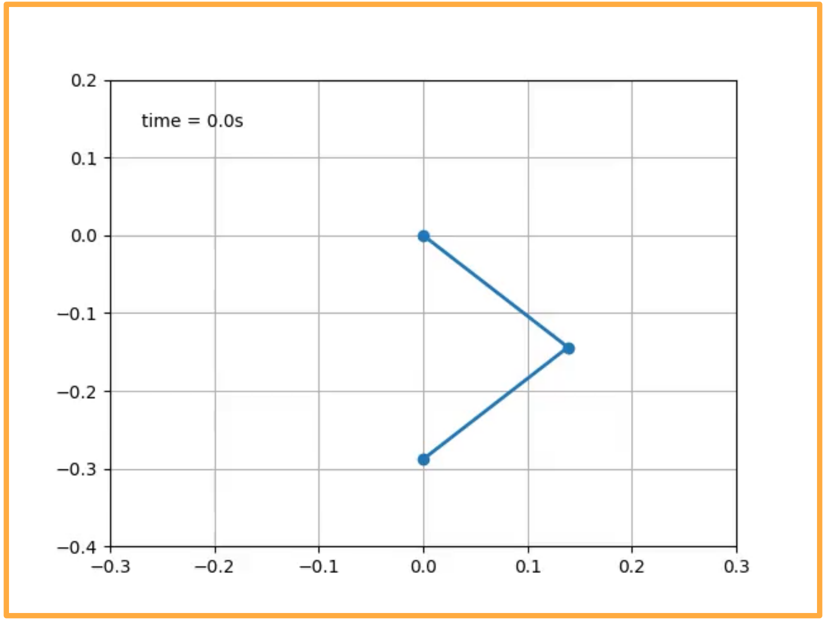
Amplitude: $$\dot{r}_i = \alpha(\mu - r_i^2)r_i$$

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

Output:

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

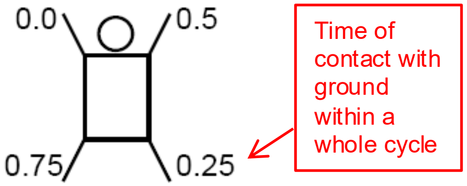
$$z_{\text{foot}} = \begin{cases} -h + g_c \sin(\theta_i) & \text{if } \sin(\theta_i) > 0 \\ -h + g_p \sin(\theta_i) & \text{otherwise} \end{cases}$$



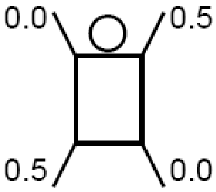
Gait Terminology

Lecture 2

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



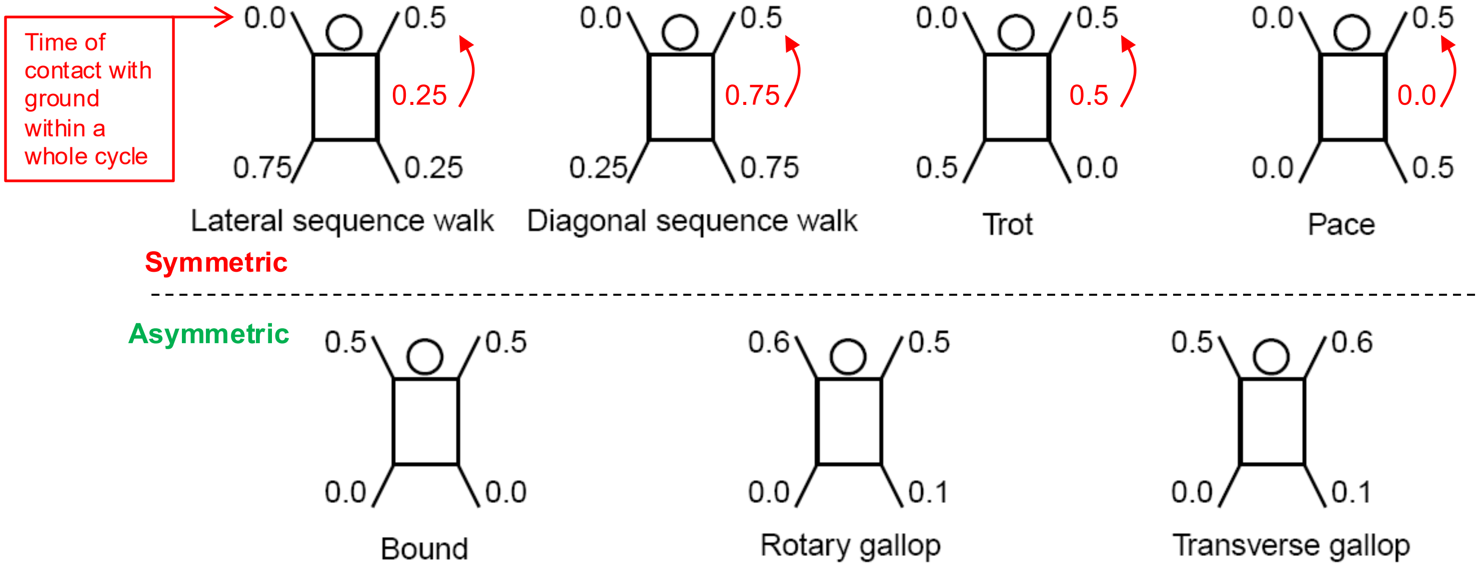
Lateral sequence walk



Trot

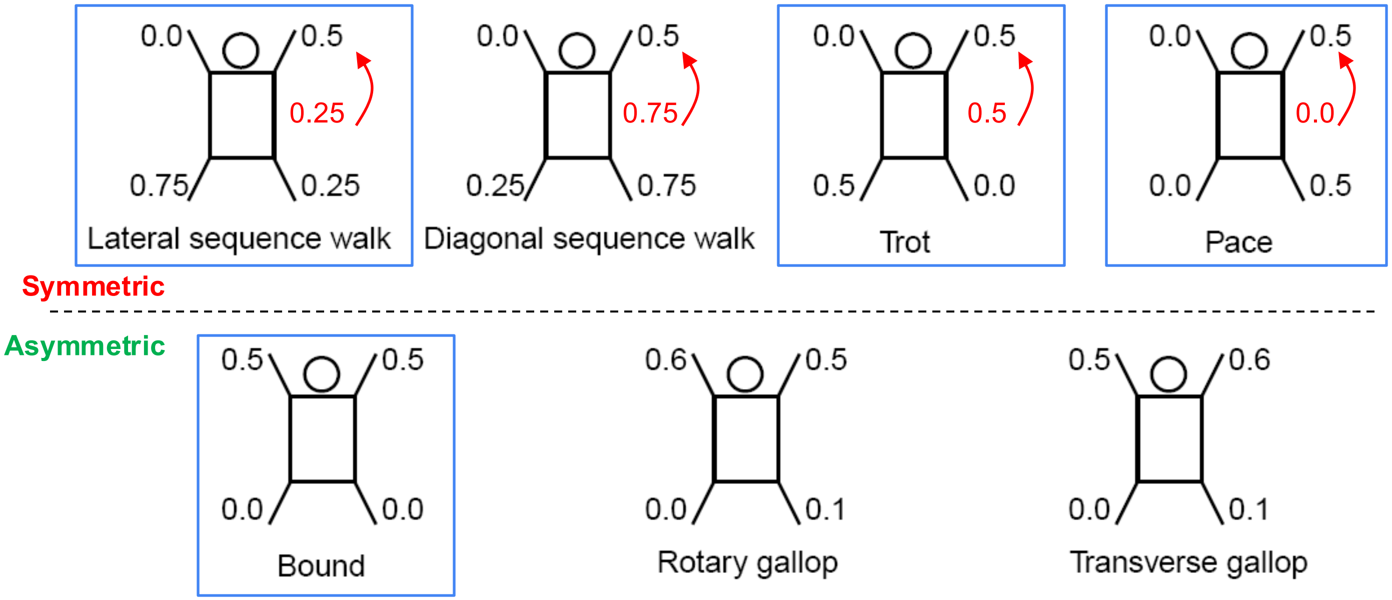
Most common quadruped gaits

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



Quadruped gaits for Project 2

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



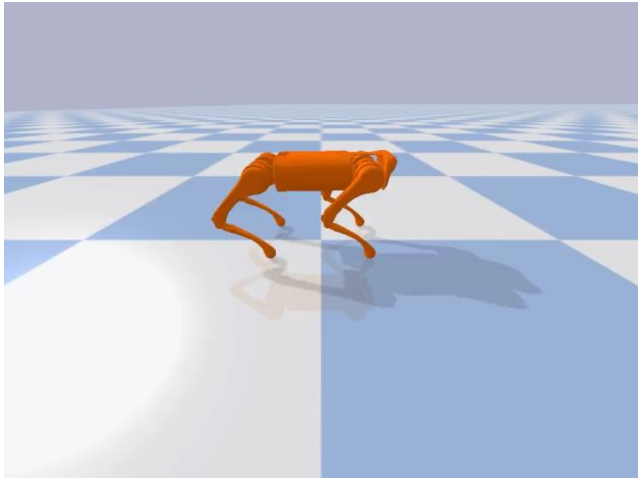
$$\dot{r}_i = \alpha(\mu - r_i^2)r_i$$

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

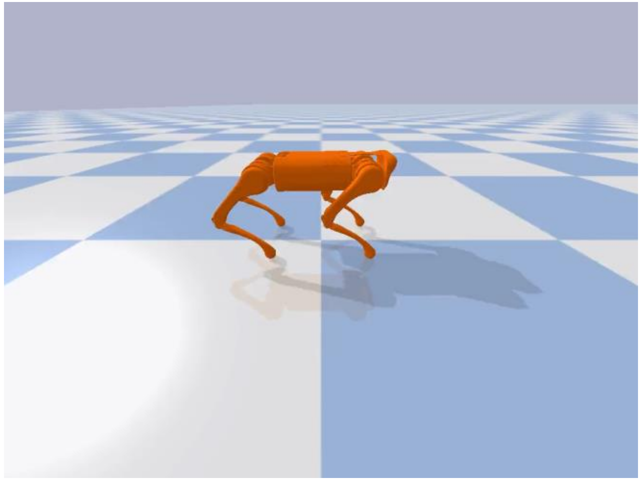
What should $\boldsymbol{\phi}$ be for each gait?

Deployed CPG locomotion

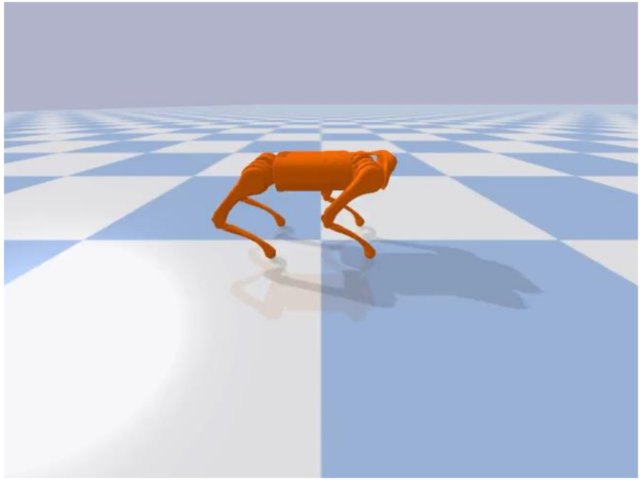
Trot



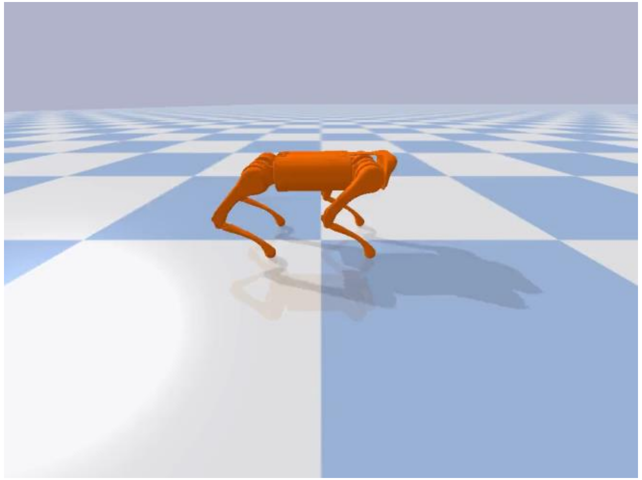
Bound



Pace



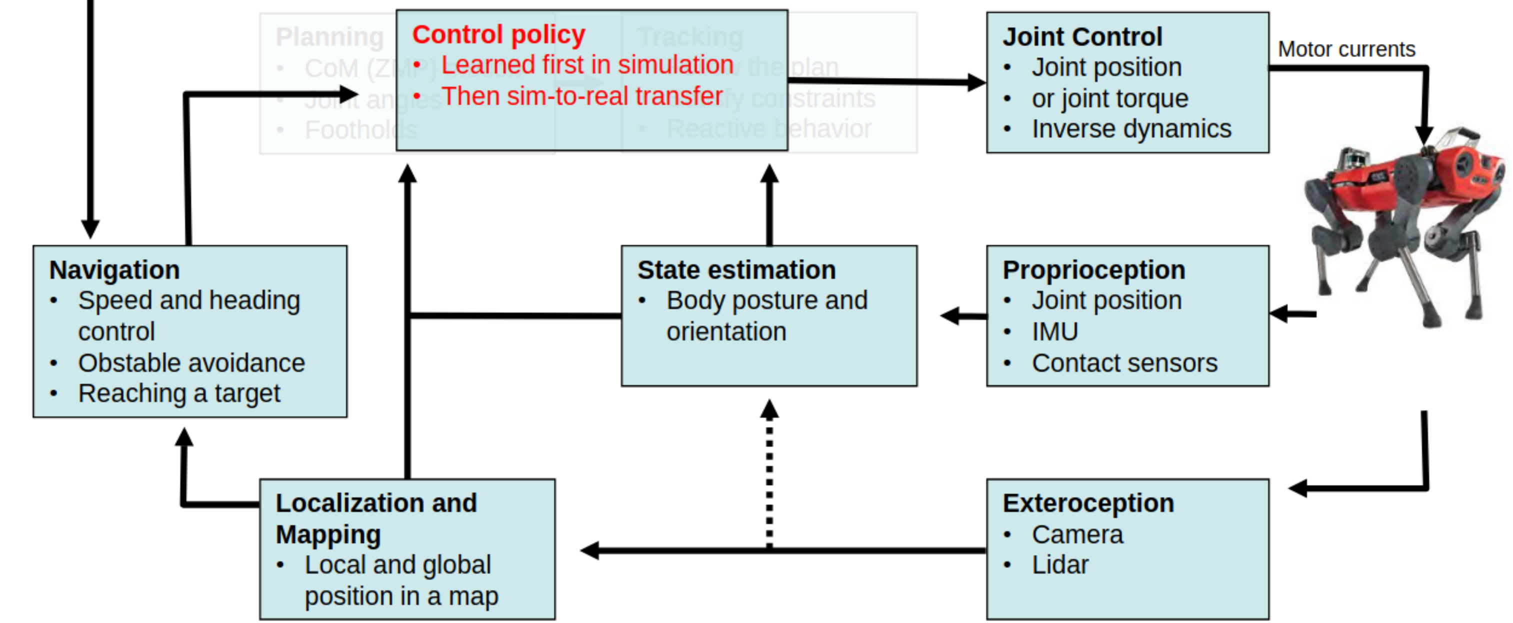
Walk



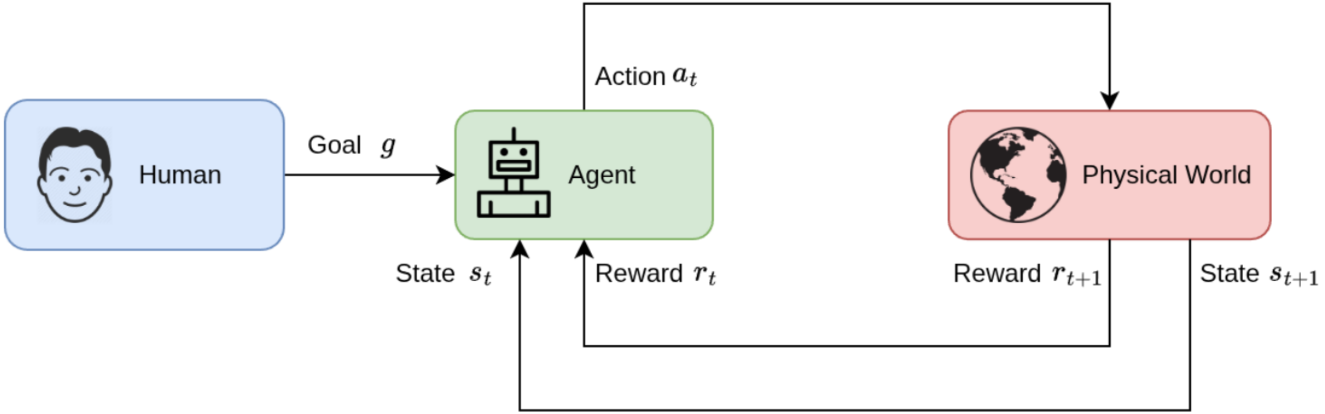
Part 2: Deep Reinforcement Learning (DRL)

Lecture 3

Typically a control policy implemented as a multi-layered neural network replaces the planning and tracking.



Robot Learning

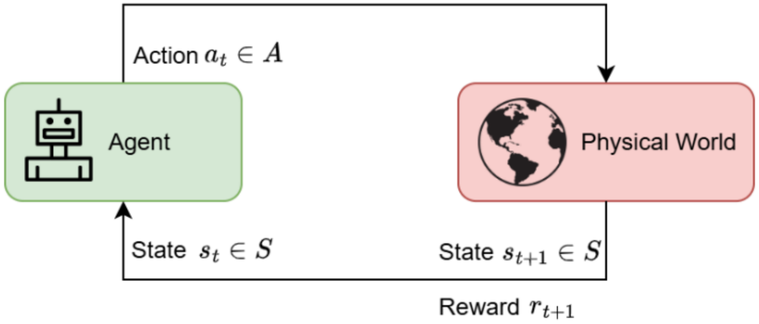


Slides from Karam Daaboul, FZI Forschungszentrum Informatik

Markov Decision Process (MDP)

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



<u>RL components</u>

- 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: $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$

Reinforcement Learning Tools

- **RL algorithm libraries**
  - stable-baselines3 https://github.com/DLR-RM/stable-baselines3
  - ray[rllib] https://github.com/ray-project/ray
  - spinningup https://github.com/openai/spinningup
  - tianshou https://github.com/thu-ml/tianshou/
  - rslrl https://github.com/leggedrobotics/rsl_rl
  - … many others!

Common algos: PPO, SAC

- **Physics simulators**
  - pybullet https://github.com/bulletphysics/bullet3
  - MuJoCo https://mujoco.org
  - RaiSim https://raisim.com
  - Isaac-Gym https://developer.nvidia.com/isaac-gym
  - Isaac-Sim https://github.com/isaac-sim/IsaacSim
  - … many others!

RL Considerations

Algorithm

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

MDP Design Decisions

- Observation space
- Action space
- Reward function

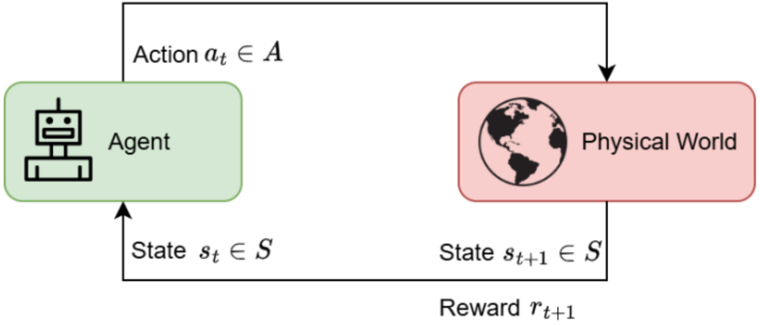
Task specific!

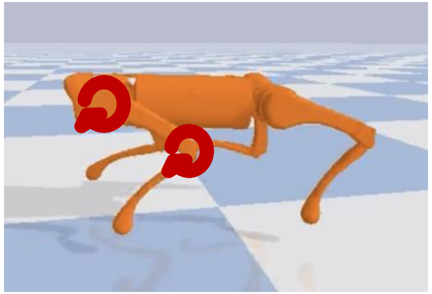
Environment Parameters

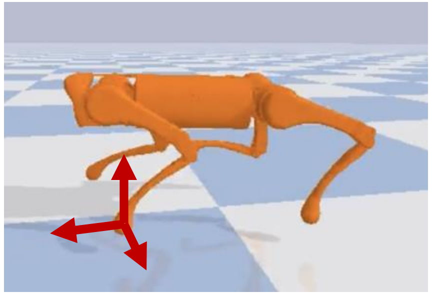
- Simulator dynamics
- Control gains – Joint/cartesian
- Control/environment time step
- Noise, latency

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

Modeling State/Action/Reward Spaces







How to model the MDP?

State/observation space?

- Body states (z, r, p, y)
- Body velocities
- Joint states

Action space?

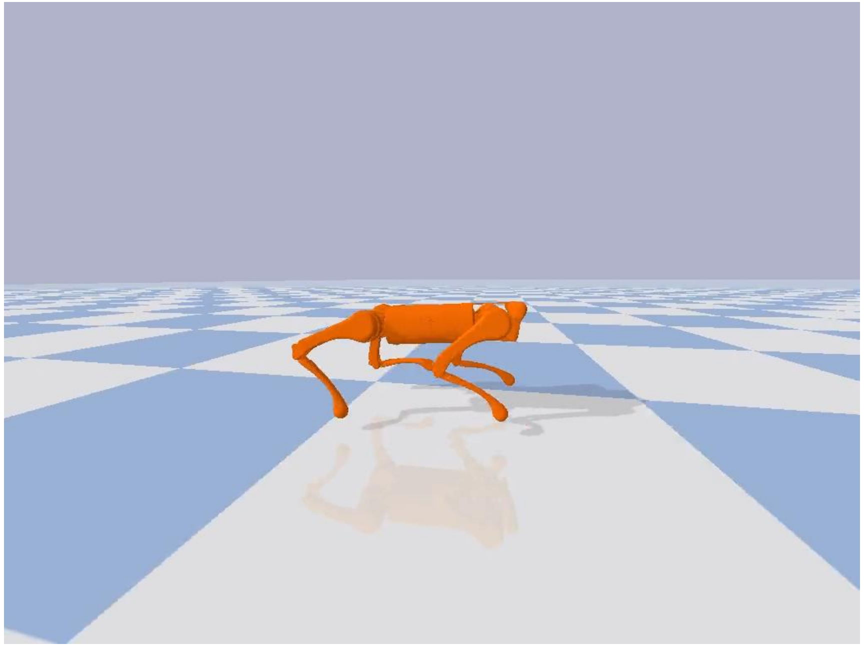
- Motor positions/torques
- Cartesian PD
- CPG state modulations

Rewards?

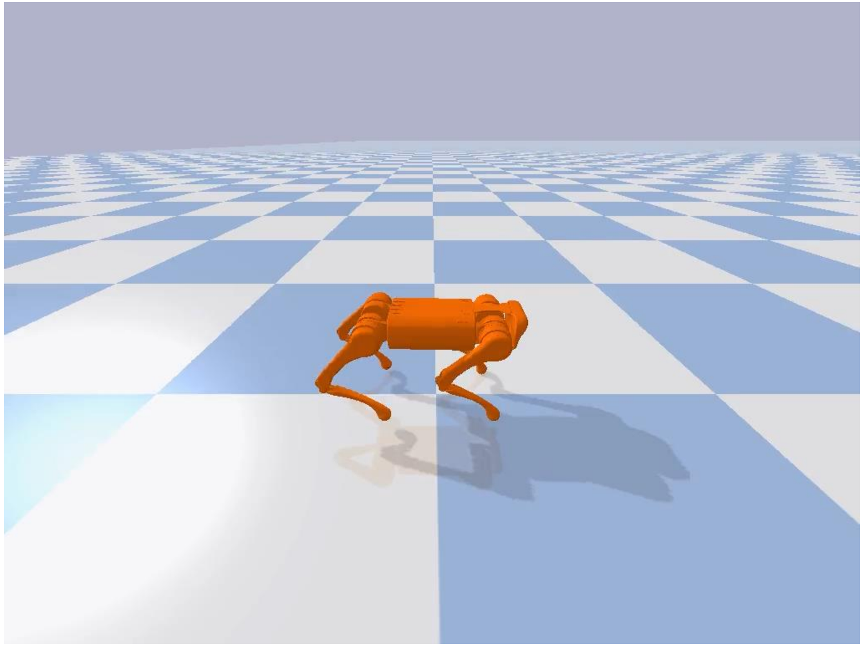
- Body linear velocity tracking
- Energy penalty
- Action rate

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

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



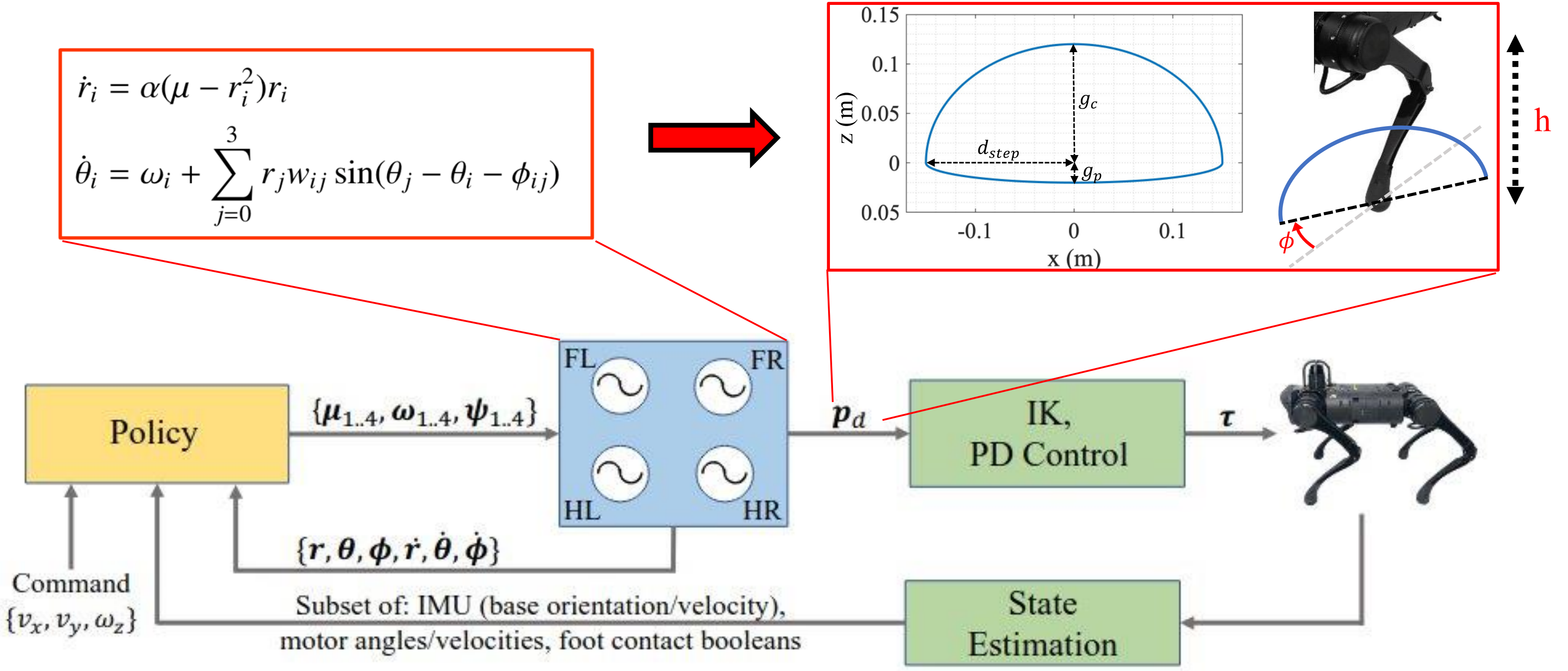
Action Space: $a_t = [x_{ee_i}, y_{ee_i}, z_{ee_i}]$



CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion

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

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



G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

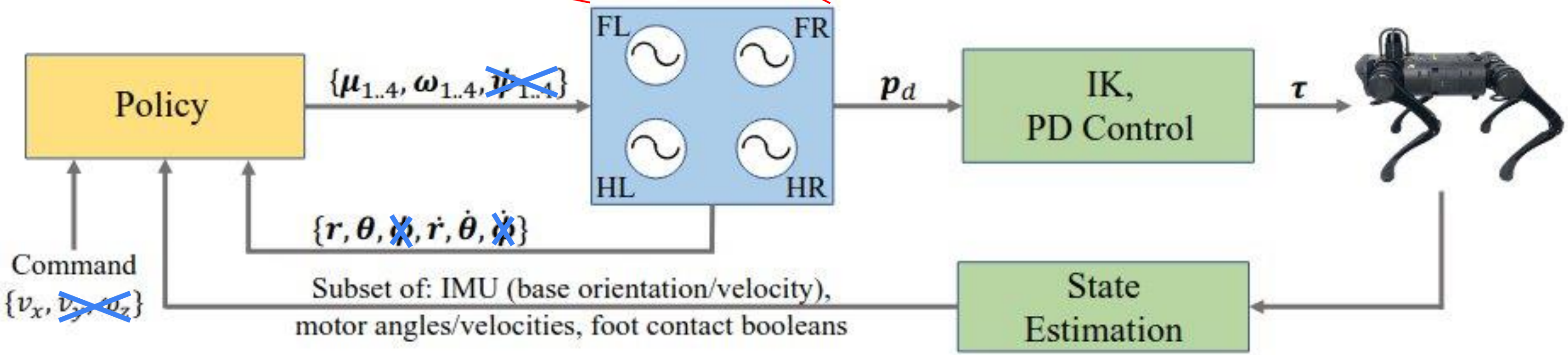
Simplification of problem formulation

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

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

1. No coupling between limb oscillators (Orange cross)

2. X locomotion direction only (Blue crosses)



G. Bellegarda, A. Ijspeert. “CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion,” RA-L 2022

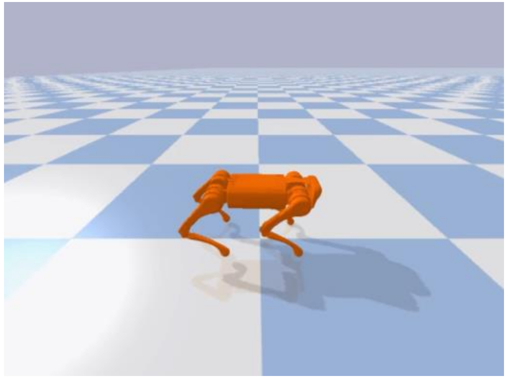
Notations to note for CPG-RL

- Notations that may be confusing, please read the CPG-RL paper for more details!

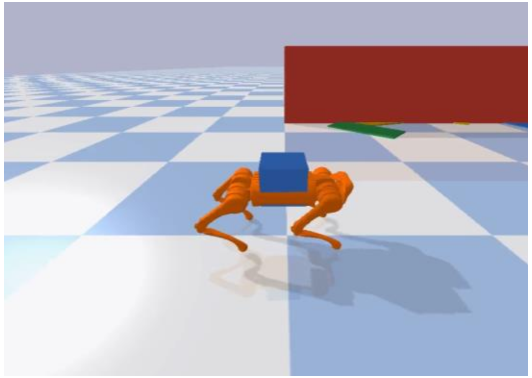
- **u (mu):** Desired oscillator amplitude
- **ω (omega):** Desired oscillator frequency
- **Ψ (psi):** Rate of change of rotation about z axis
- **r (r):** Current oscillator amplitude
- **Θ (theta):** Current oscillator phase
- **Φ (phi):** Rotation about z axis
- **$\dot{\Phi}$ (phi dot):** Rate of change of rotation about z axis
- **φ_{ij} (phi_ij):** Fixed phase offsets between oscillators
- **ω_{ij} (omega_ij):** Weights between oscillators / coupling strength

Deployed DRL locomotion

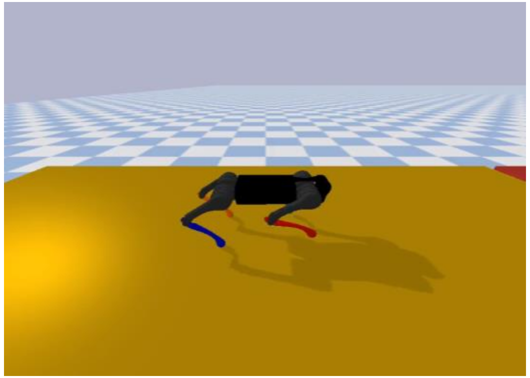
Forward locomotion



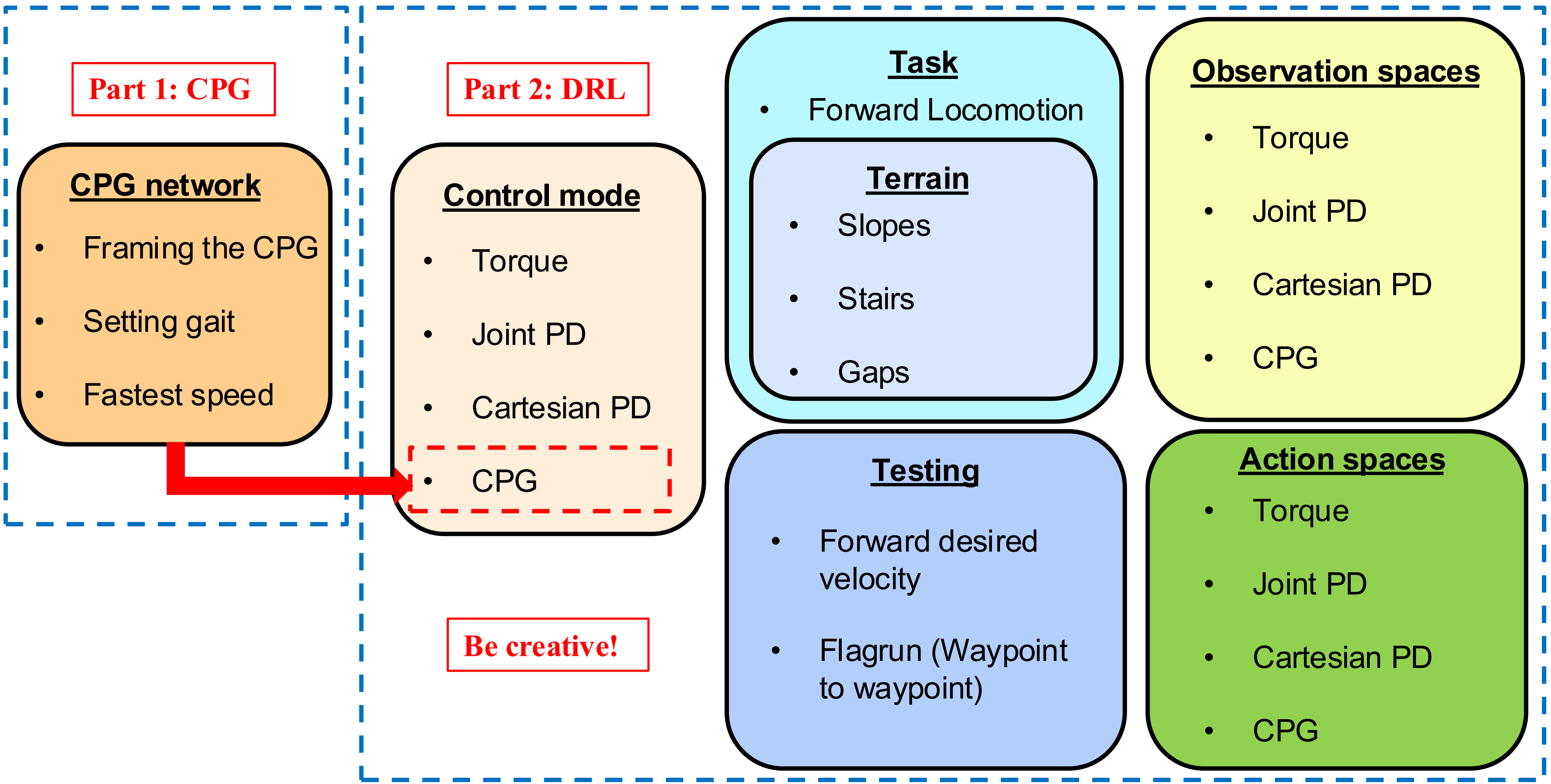
Weight carrying +
Unstructured terrain



Gap crossing



Project 2


Part 1: CPG
CPG network
Framing the CPG
Setting gait
Fastest speed
Part 2: DRL
Control mode
Torque
Joint PD
Cartesian PD
CPG
Be creative!
Task
Forward Locomotion
Terrain
Slopes
Stairs
Gaps
Testing
Forward desired velocity
Flagrun (Waypoint to waypoint)
Observation spaces
Torque
Joint PD
Cartesian PD
CPG
Action spaces
Torque
Joint PD
Cartesian PD
CPG

Tips

- Always start ***simple***! Don't tune stuff unnecessarily before getting a simple working example.
- ***Monitor*** episode length and reward mean during training, you can always replay intermediate weights to check performance.
- Training should complete within a couple million timesteps for simple tasks with ***reasonable*** observation space, action space, and reward function choices (with no noise in the environment).
- Start training ***early***! Explore deeply ***one configuration*** than everything (Eg Joint PD + Gaps, CPG + Slopes)
- Make sure the thought flow is ***logical*** instead of just blindly tuning/hacking the parameters, this can be never ending :(. A good way is to check how the ***research articles*** on Moodle or ***literature online*** model their MDP.

Assignment

- Project 2 instructions are on Moodle
- Submit 1 zip file per group before 19th December 23:59
- Download code at https://gitlab.epfl.ch/pey/lr-miniproject-2.git