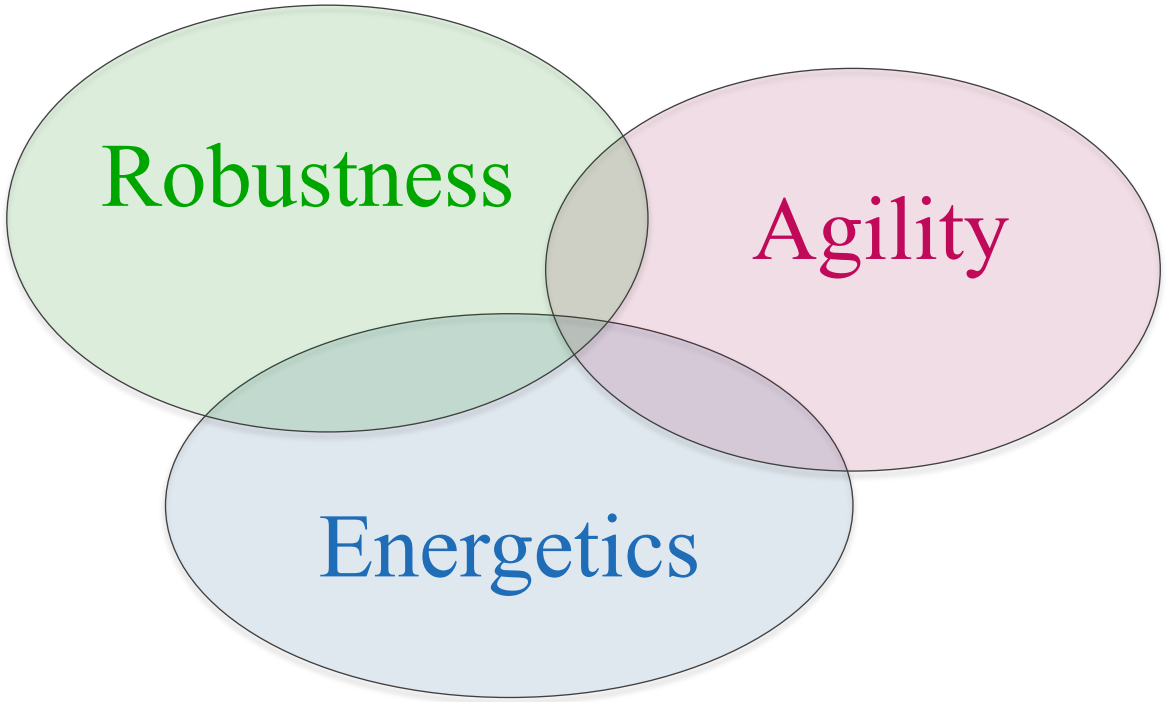
Legged Locomotion: Trajectory Optimization, Machine Learning, and Bio-Inspired Control

Guillaume Bellegarda

08.10.2024

EPFL Biorobotics Laboratory

Locomotion Goals



…in stochastic environments (variability)

Robustness

unknown variability

Agility

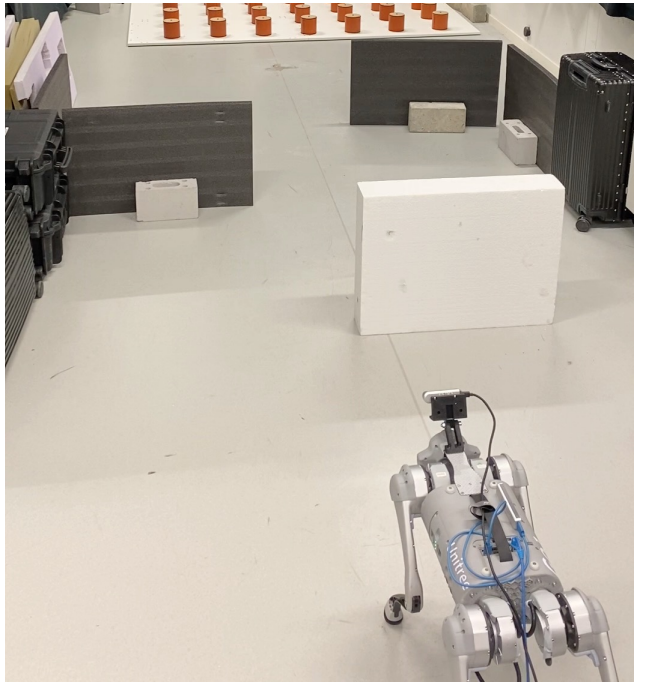
Energetics

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

G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. “Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning,” IROS 2022

Robustness

unknown variability

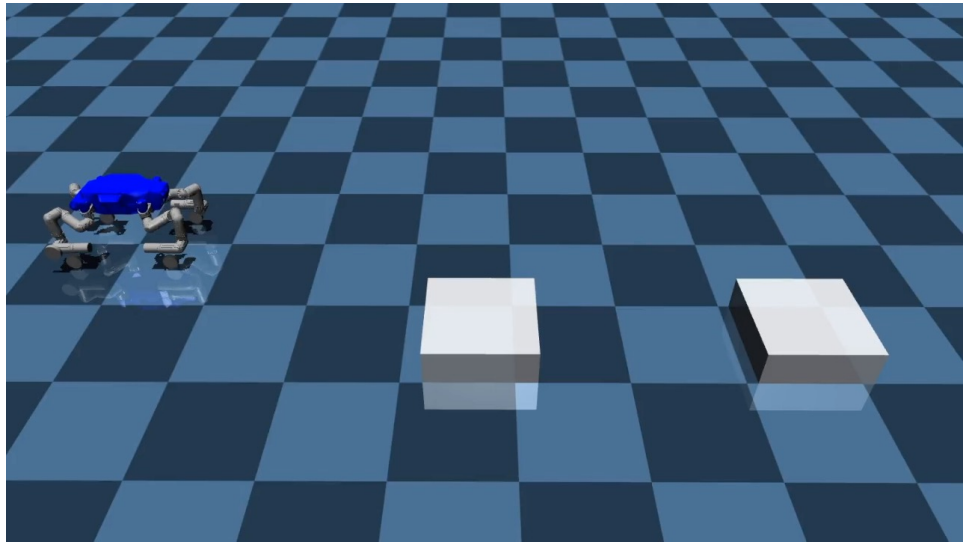


Agility

known variability

Energetics

M. Shafiee, **G. Bellegarda**, A. Ijspeert. “Viability Leads to the Emergence of Gait Transitions in Learning Agile Quadrupedal Locomotion on Challenging Terrains,” Nature Communications 2024.
G. Bellegarda, M. Shafiee, A. Ijspeert. “Visual CPG-RL: Learning Central Pattern Generators for Visually-Guided Quadruped Locomotion,” ICRA 2024.
G. Bellegarda, K. Byl. “Versatile Trajectory Optimization Using a LCP Wheel Model for Dynamic Vehicle Maneuvers,” ICRA 2020.



Robustness

unknown variability

Agility

known variability

Energetics

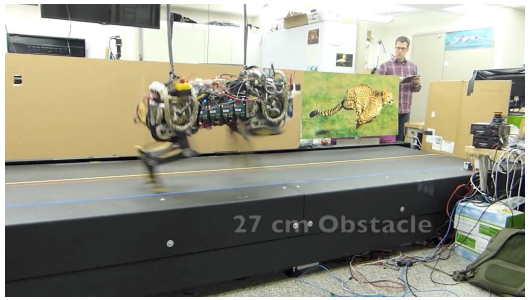
as efficient as practical

G. Bellegarda, K. Byl. “Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip,” CDC 2019.
G. Bellegarda, K. Byl. “Design and Evaluation of Skating Motions for a Dexterous Quadruped,” ICRA 2018.

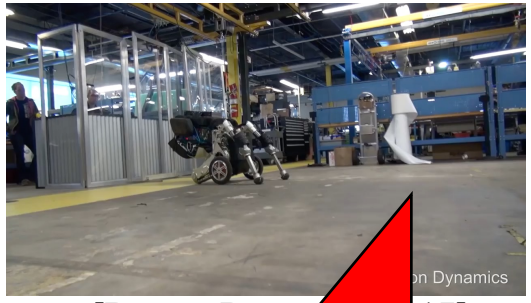
Robotics Approaches

Model-based methods

i.e. trajectory optimization, model-predictive control (MPC), Central Pattern Generators (CPGs)



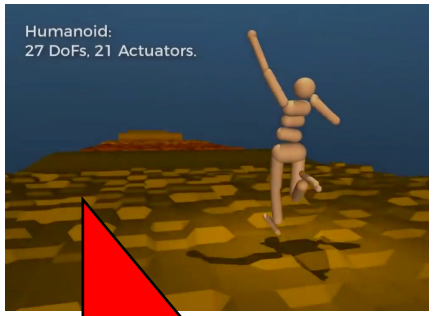
[Park et al. 2017]



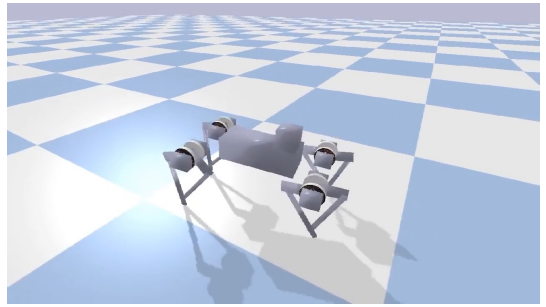
[Boston Dyna 017]

Learning

i.e. reinforcement learning, imitation learning

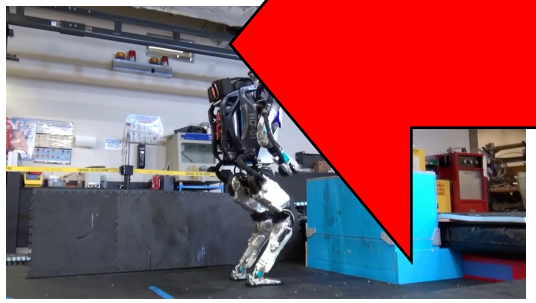


t al. 2017]

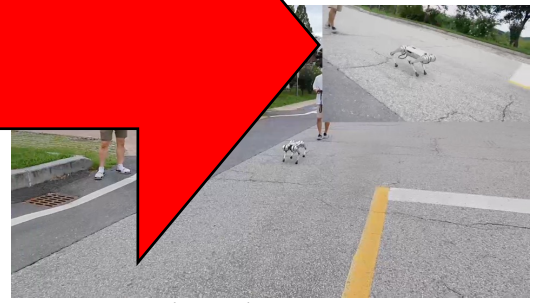


[Tan et al. 2018]

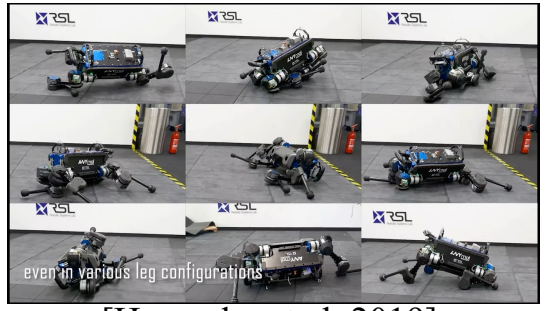
[Kim et al. 2019]



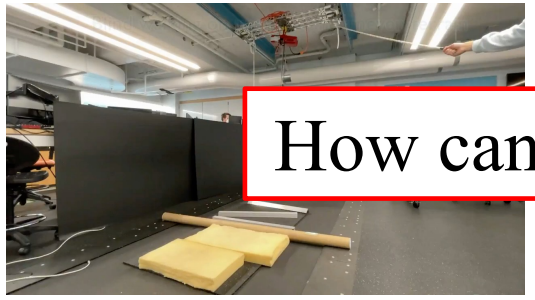
[Boston Dynamics 2018]



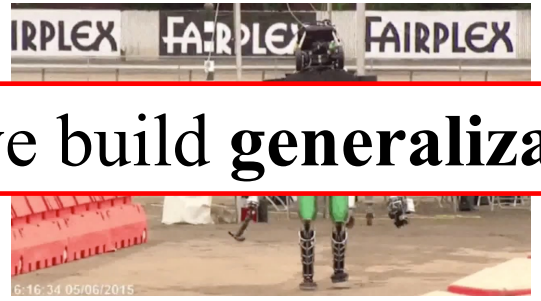
[Ji et al. 2022]



[Hwangbo et al. 2019]



[Xiong et al. 2021]



[DRC 2015]

[Lee et al. 2020]

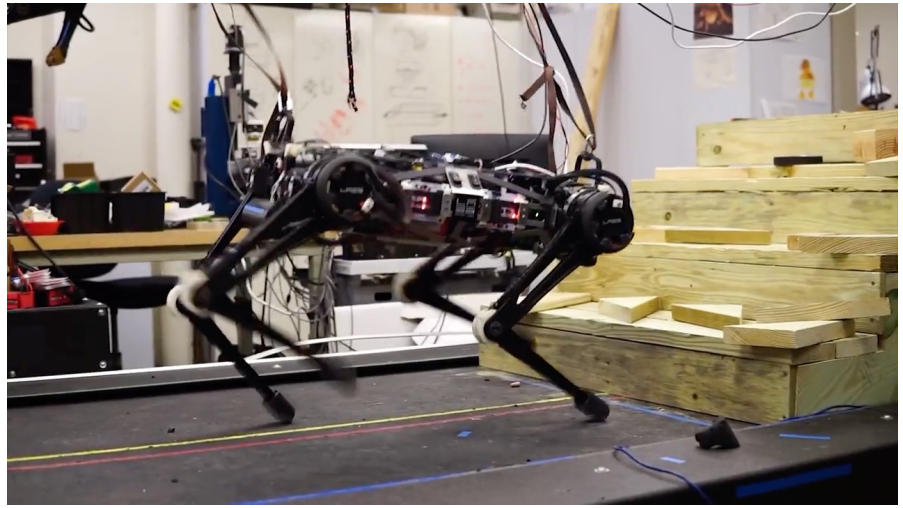
[Yang et al. 2021]

How can we build **generalizable**, agile, robust, and safe systems?

Goals for Today

1. Insights into formulating trajectory optimization (model-predictive control) problems

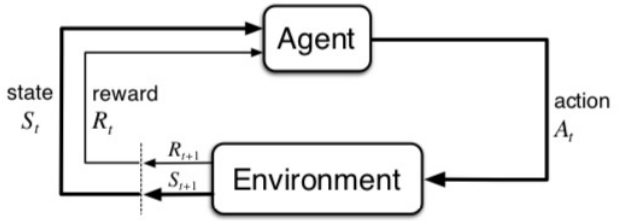
$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=0}^{k-1} ||\mathbf{x}_{i+1} - \mathbf{x}_{i+1,\text{ref}}||_{\mathbf{Q}_i} + ||\mathbf{u}_i||_{\mathbf{R}_i} \\
\text{subject to} \quad & \mathbf{x}_{i+1} = \mathbf{A}_i \mathbf{x}_i + \mathbf{B}_i \mathbf{u}_i, i = 0 \dots k-1 \\
& \underline{\mathbf{c}}_i \leq \mathbf{C}_i \mathbf{u}_i \leq \overline{\mathbf{c}}_i, i = 0 \dots k-1 \\
& \mathbf{D}_i \mathbf{u}_i = 0, i = 0 \dots k-1
\end{aligned}
$$

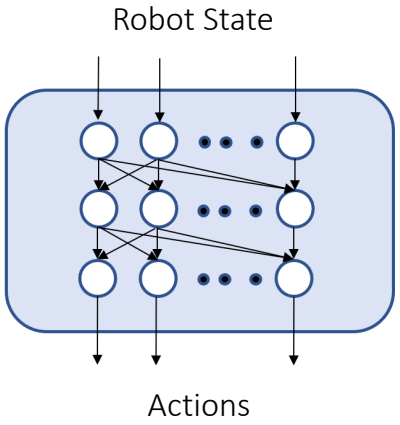


G. Bledt, P. Wensing, S. Ingersoll, S. Kim. *Contact Model Fusion for Event-Based Locomotion in Unstructured Terrains.* ICRA 2018
G. Bledt, M. Powell, B. Katz, J. D. Carlo, P. Wensing, S. Kim. *MIT Cheetah 3: Design and Control of a Robust, Dynamics Quadruped Robot.* IROS 2018

Goals for Today

1. Insights into formulating trajectory optimization (model-predictive control) problems
2. Insights into training control policies with deep reinforcement learning





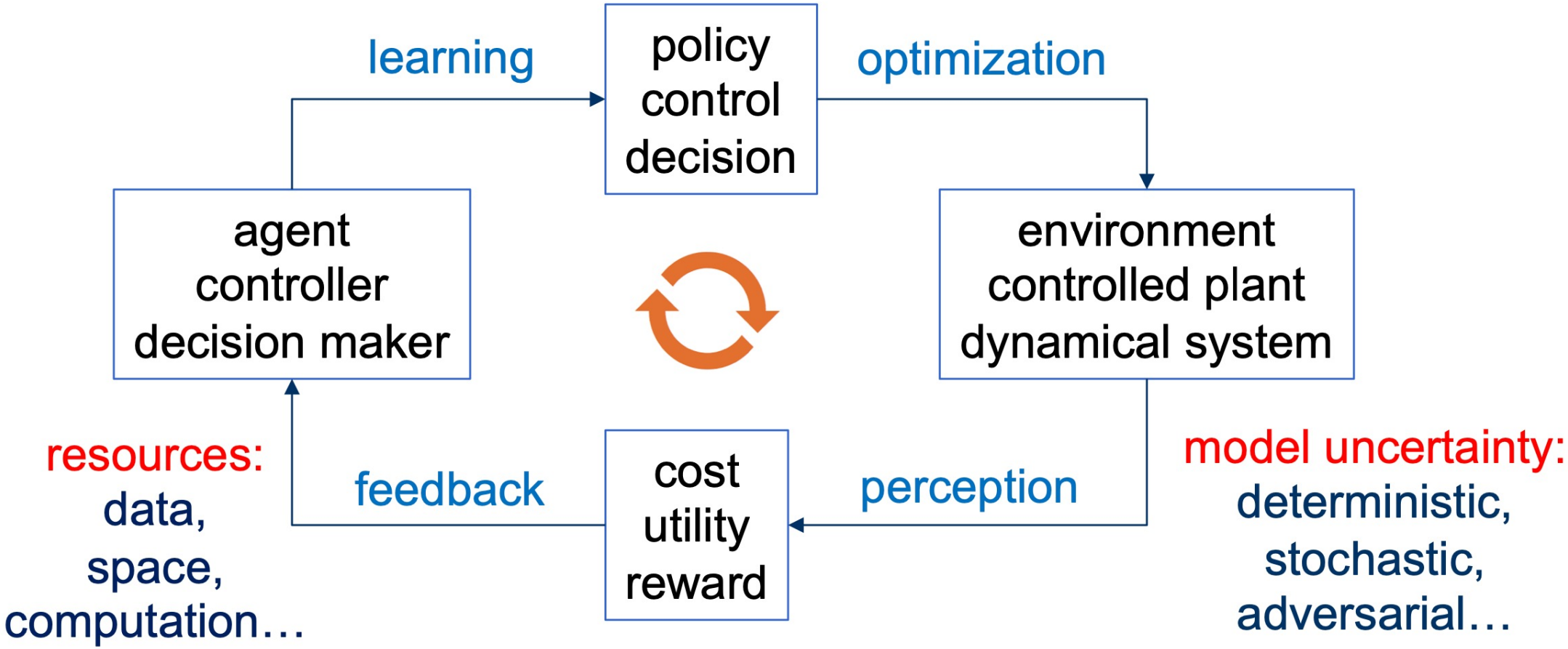
R. Sutton and A. Barto. Introduction to Reinforcement Learning. MIT Press 1998

J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. Science Robotics 2020

Outline

I. Motivation and Goals
II. Model-Predictive Control vs. Reinforcement Learning (same?)
III. Model-based methods
    I. Background on RoboSimian and designing skating motions
    II. Trajectory optimization allowing wheel slip
IV. Learning-based methods
    I. Action space in reinforcement learning [RoboSimian, A1]
    II. Augmenting motion planning with deep reinforcement learning [A1]
V. Bio-inspired learning
VI. Conclusion

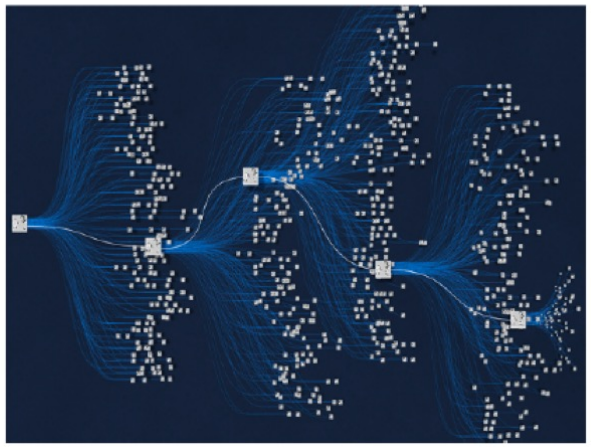
Common Setting: Closed-Loop Autonomous System



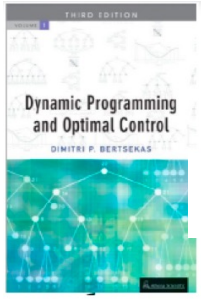
Reinforcement Learning Challenges

In RL, an agent learns by interacting with an environment

- unknown or changing environments
- delayed rewards or feedback
- enormous state and action space
- nonconvexity



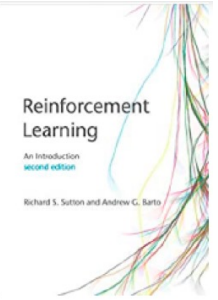
Terminology: State Space Model



Optimal Control

environment
controlled plant
dynamical system

Reinforcement Learning



State and Control space: $\mathcal{S}, \mathcal{U}$

State: $x_k \in \mathcal{S}, k = 0, 1, \ldots$

Control: $u_k \in \mathcal{U}, k = 0, 1, \ldots$

Dynamical System:

$$x_{k+1} = f(x_k, u_k)$$

$$x_{k+1} = f(x_k, u_k, w_k)$$ stochastic

Output/observation (feature):

$$y_k = h(x_k, u_k) + n_k$$ stochastic

State and Action space: $\mathcal{S}, \mathcal{A}$

State: $s_t \in \mathcal{S}, t = 0, 1, \ldots$

Action: $a_t \in \mathcal{A}, t = 0, 1, \ldots$

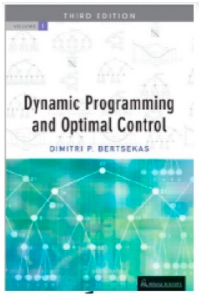
MDP Transition (or simulation):

$$\mathcal{T}_{ijk} = p(s_{t+1} = i \mid s_t = j, a_t = k)$$

Observation (feature):

$$p(o_t \mid s_t)$$

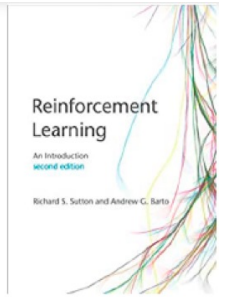
Terminology: Optimization Objective



Optimal Control

policy
control
decision

Reinforcement Learning



Cost: $g(x_k, u_k) \in \mathbb{R}$

Total cost function:

$$J(x_0; u_0, \ldots, u_N) = \sum_{k=0}^{N} g(x_k, u_k)$$

Control law: $u(x_k);\ u^*(x_k)$

$$x_{k+1} = f(x_k, u(x_k))$$
$$u_{k+1} = u(x_{k+1})$$

Value function (minimal cost to go):

$$J^*(x_0) = \min_{u(\cdot)} \sum_{k=0}^{N} g(x_k, u(x_k, k))$$

Reward: $r(s_t, a_t) \in \mathbb{R}$

Total reward (return):

$$J(s_1; a_1, \ldots, a_T) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[r(s_t, a_t)]$$

Policy: $\pi(a_t \mid s_t);\ \pi^\star(a_t \mid s_t)$

$$p((s_{t+1}, a_{t+1}) \mid (s_t, a_t)) =$$
$$p(s_{t+1} \mid s_t, a_t)\pi(a_{t+1} \mid s_{t+1})$$

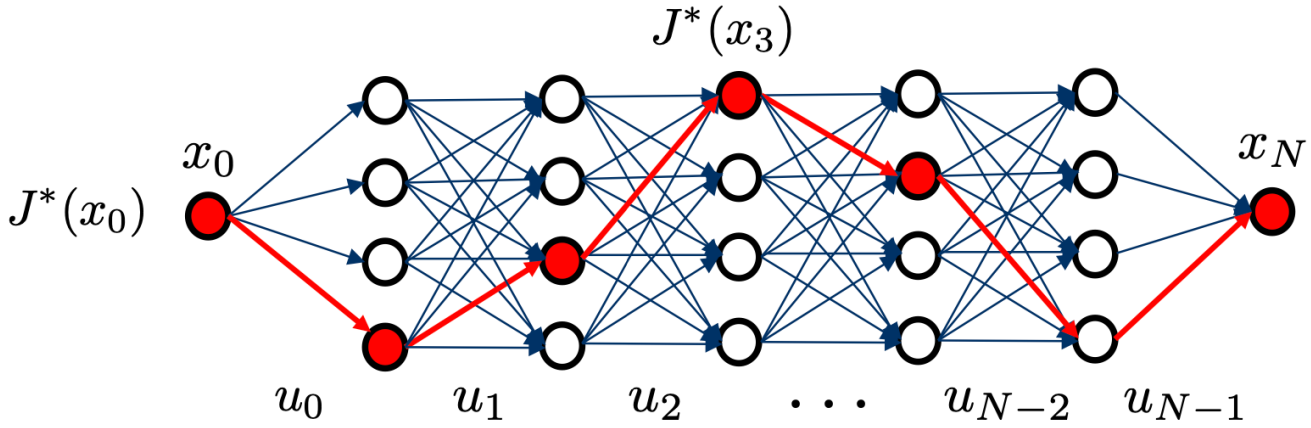
Value function (maximal return):

$$V^\star(s_1) = \max_{\pi(\cdot)} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_\pi[r(s_t, a_t)]$$

Principle of (Path) Optimality

Dido of Carthage…, Euler, Lagrange, Newton, Hamilton, Jacobi, Pontryagin, Bellman, Ford, Kalman

850 BC → 1960 AC

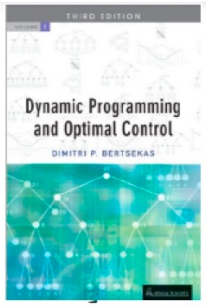


Principle of Optimality (***Richard Bellman'54***):
An optimal path has the property that any subsequent portion is optimal.

Dynamical Programming: A "Fixed-Point" Type Algorithm

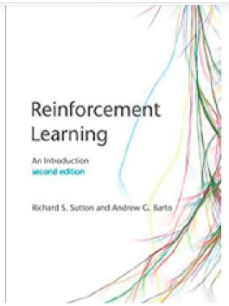
$$J^*(x_k) = \min_{u_k} \left[g(x_k, u_k) + J^*(\underbrace{f(x_k, u_k)}_{x_{k+1}}) \right], \ \forall x_k$$

Terminology: Optimization Objective



Optimal Control

Reinforcement Learning



Value function and Q-function:

$$J^*(x_k) = \min_{u_k} \underbrace{\left[g(x_k, u_k) + J^*(f(x_k, u_k))\right]}_{Q(x_k, u_k)}, \ \forall x_k \qquad V^\star(s_t) = \max_{\pi} \mathbb{E}_\pi \Big[\underbrace{p(s_{t+1} \mid s_t, a_t)}_{\mathcal{T}} \underbrace{\left[r(s_t, a_t) + V^\star(s_{t+1})\right]}_{Q(s_t, a_t)} \Big], \ \forall s_t$$

(many, many, many different ways to learn and solve them, depending on…)

Given the value or Q-function, the optimal control/policy and path:

$$u_k^* = \arg\min_{u_k} \underbrace{\left[g(x_k^*, u_k) + J^*(f(x_k^*, u_k))\right]}_{Q(x_k^*, u_k)},$$

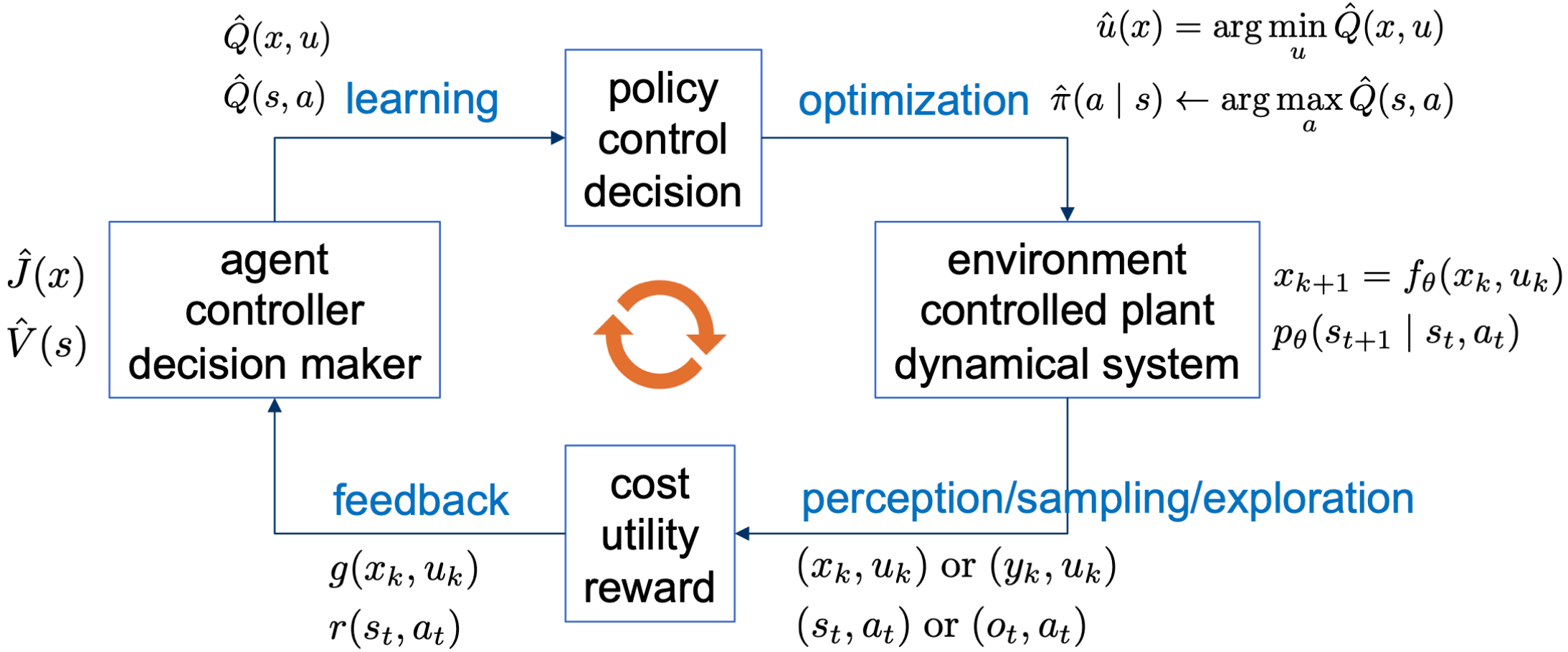
$$x_{k+1}^* = f(x_k^*, u_k^*)$$

$$\pi^\star(s_t) = \arg\max_{a_t} Q(s_t, a_t)$$

$$p(s_{t+1} \mid s_t, \pi^\star(s_t))$$

*In practice, states can be replaced by observations or "**features**" to relate to control or action.*

Closed-Loop Autonomous System (Formal)



From Principle to Computation

What to Compute, and How?

	Optimal Control	Reinforcement Learning
Optimal value function:	$J^*(x),$	$V^\star(s)$
Optimal Q-function:	$Q^*(x,u),$	$Q^\star(s,a)$
Optimal control/policy:	$u^*(x),$	$\pi^\star(a \mid s)$ (or $u^\star(y),\ \pi^\star(a \mid o)$)
System/model identification:	$f^*(x,u),$	$p^\star(s_{t+1} \mid s_t, a_t)$

Closed-form versus numerical solution (simulation & optimization)

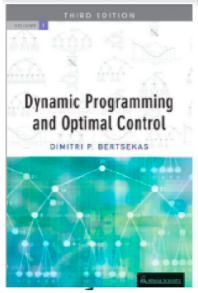
LQR: $J^*(x_k) = \min_{u_k} \left[x_k^T Q x_k + u_k^T R u_k + J^*(Ax_{k+1} + Bu_k)\right]$

The Riccati equation (Kalman Filter '60):

$$K_k = -(\bar{R} + \bar{B}^\intercal V_{k+1} \bar{B})^{-1} \bar{B}^\intercal V_{k+1} \bar{A} \tag{48}$$

$$V_k = \bar{Q} + \bar{A}^\intercal V_{k+1} \bar{A} - \bar{A}^\intercal V_{k+1} \bar{B} (\bar{R} + \bar{B}^\intercal V_{k+1} \bar{B})^{-1} \bar{B}^\intercal V_{k+1} \bar{A} \tag{49}$$

Control vs. Learning



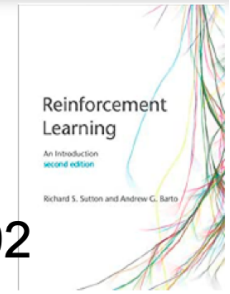
Optimal Control

- LQR
- Parallel parking
- Chained form systems
- Mechanical systems…

Conditions & Assumptions

- clear model class/uncertainty
- clear cost function
- low to moderate dimension
- continuous state/time…

Reinforcement Learning



- Backgammon: Tesauro, 1992
- Chess: Deep Blue, 1997
- Go: Alpha Go, 2017
- Video games, robots…

Conditions & Assumptions

- unknown models (but can sample)
- uncertain, long-horizon return
- large-scale, high-dimensional
- discrete state/time…

Solutions that work for a broad class of problems v.s. a few (important) instances

Outline

I. Motivation and Goals
II. Model-Predictive Control vs. Reinforcement Learning (same?)
III. Model-based methods
    I. Background on RoboSimian and designing skating motions
    II. Trajectory optimization allowing wheel slip
IV. Learning-based methods
    I. Action space in reinforcement learning [RoboSimian, A1]
    II. Augmenting motion planning with deep reinforcement learning [A1]
V. Bio-inspired learning
VI. Conclusion

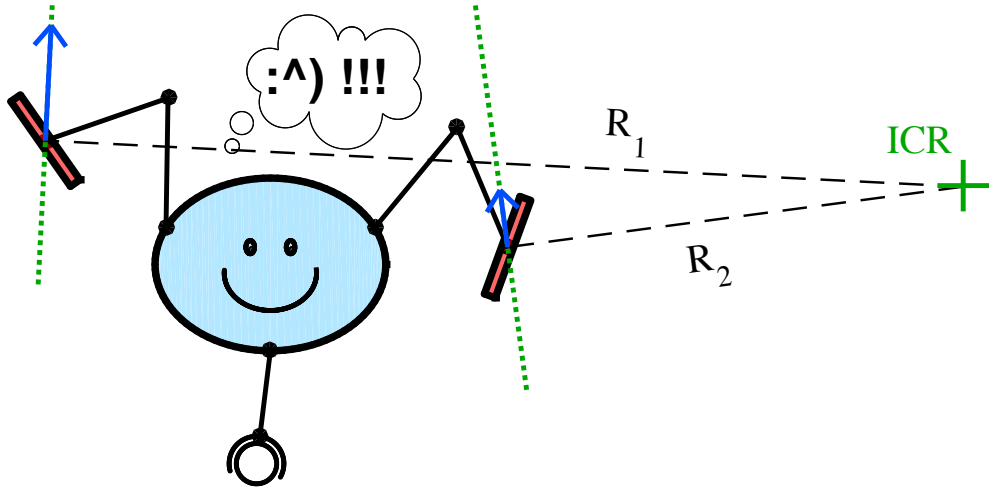
JPL's RoboSimian: Dexterous Quadruped

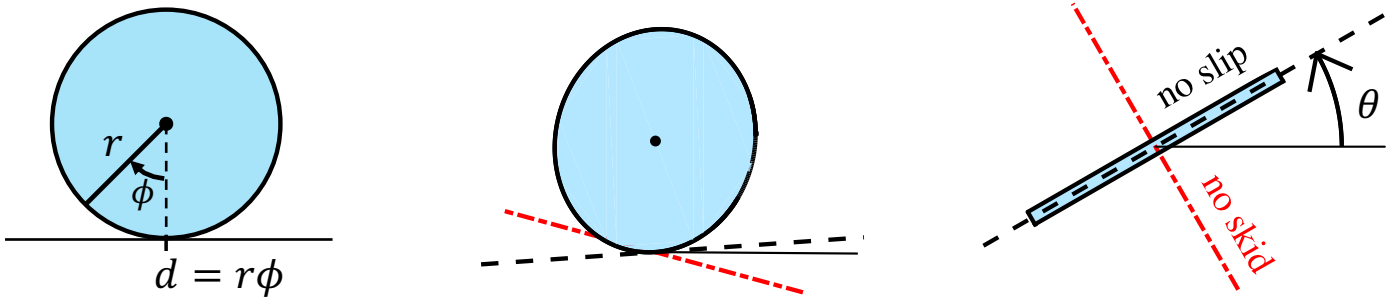
- Four identical limbs, each with 7 degrees of freedom (DOFs)
- Joint motor speeds peak at 1 (rad/s)
  - RoboSimian is stable, but very slow
- Passive single-wheel skate mounted at each forearm
  - Increase efficiency, speed
- 6 actuated DOFs available to set 6-DOF pose of each skate
- No sensing available at skate contact

G. Bellegarda, K. van Teeffelen, K. Byl. "Design and Evaluation of Skating Motions for a Dexterous Quadruped," ICRA 2018

Modeling: Skating Kinematics

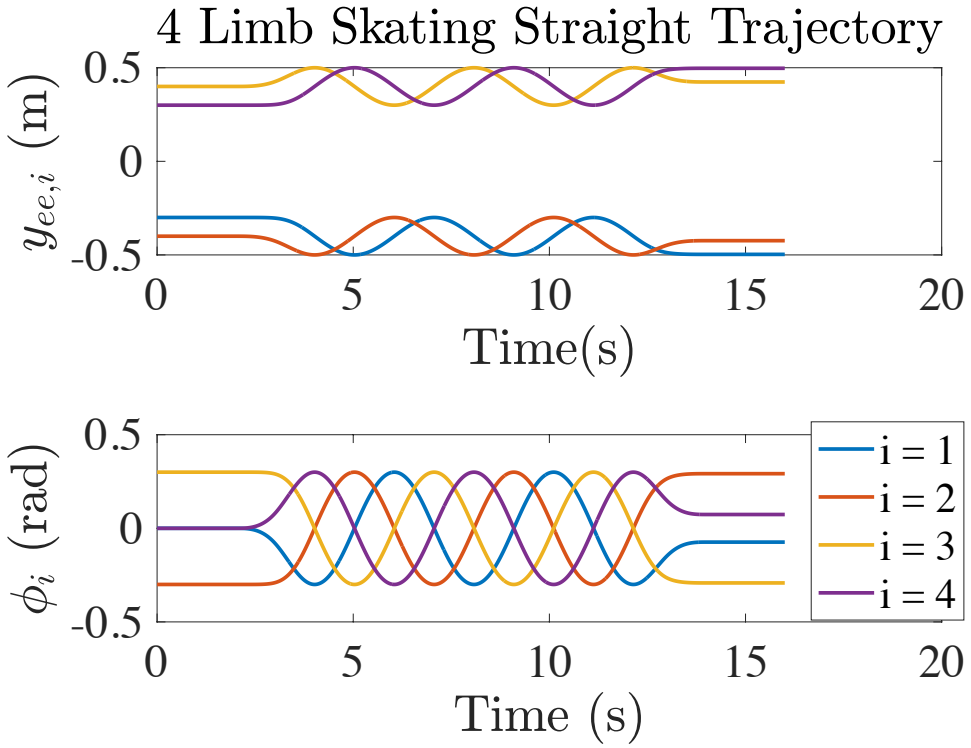
- Assumptions:
  - No skid at wheel axis
  - Free rolling perpendicular to no-skid axis (no-slip)
  - Wheel can rotate freely about a point contact with the ground

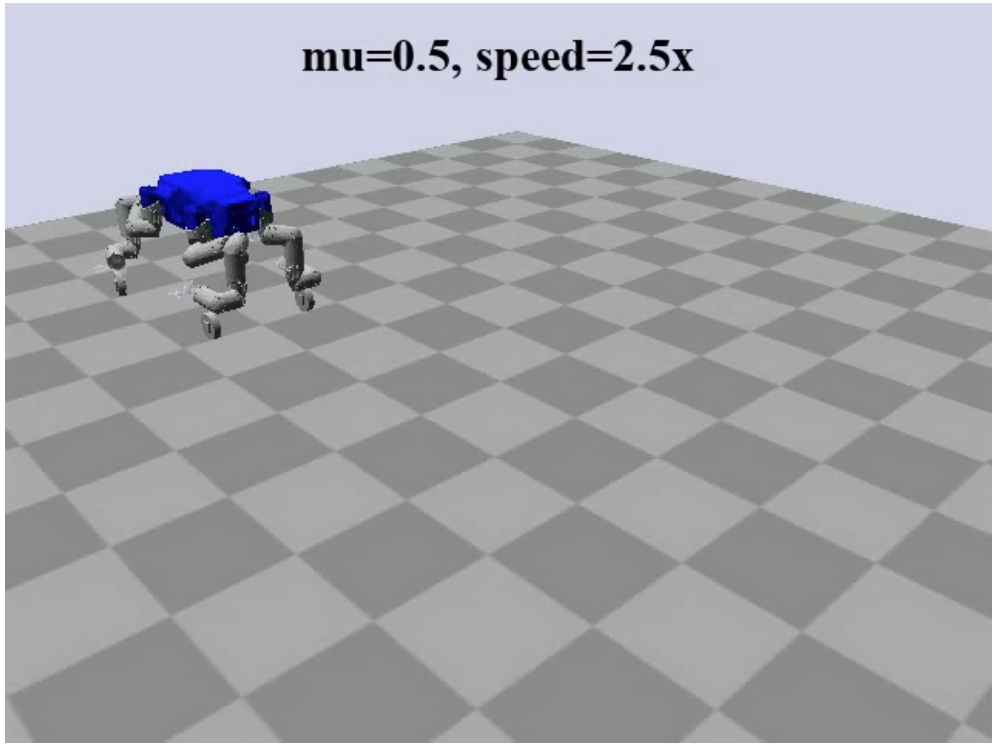




G. Bellegarda, K. van Teeffelen, K. Byl. "Design and Evaluation of Skating Motions for a Dexterous Quadruped," ICRA 2018

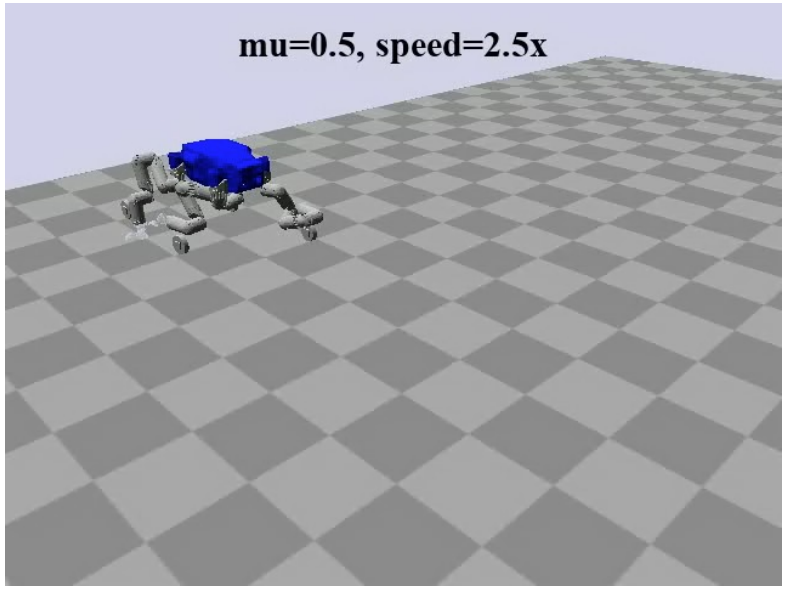
Designing Skating Motions for a Dexterous Quadruped

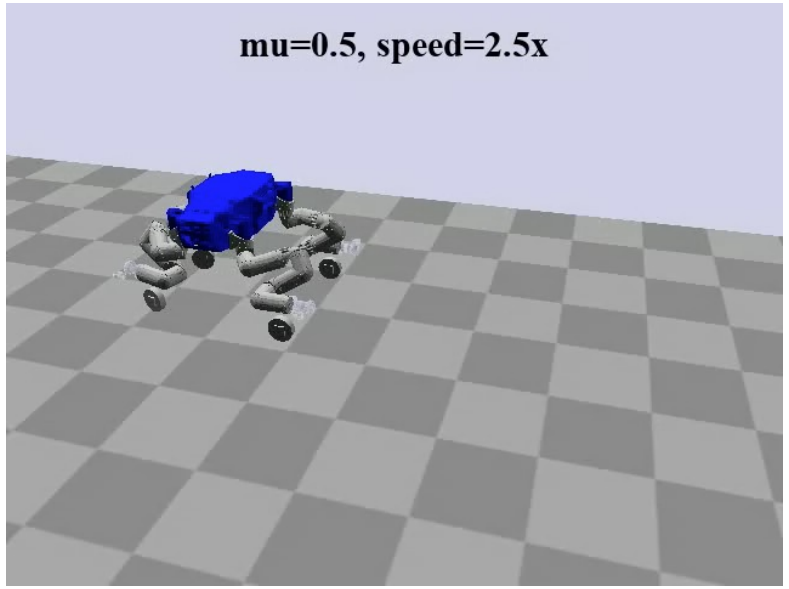


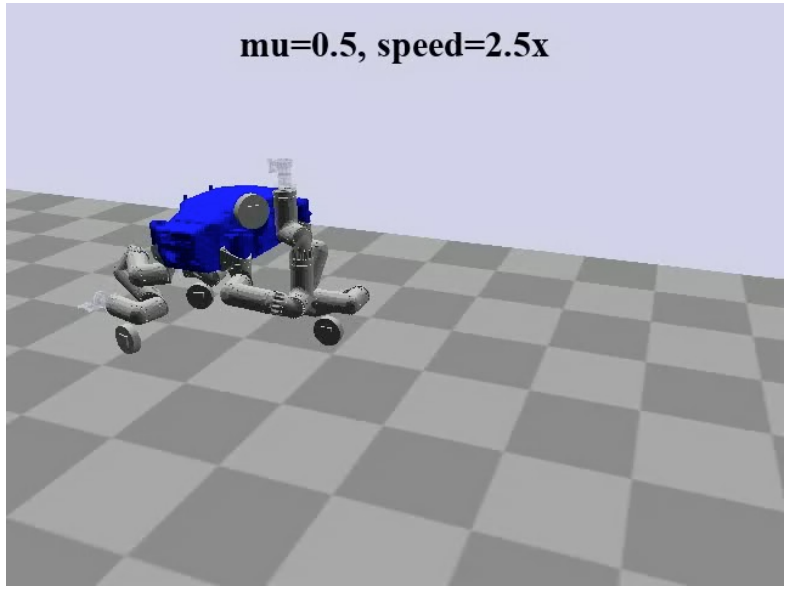


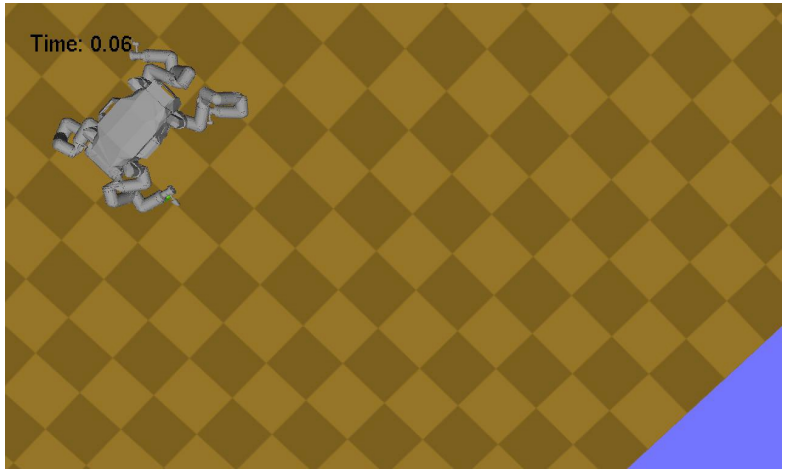
G. Bellegarda, K. van Teeffelen, K. Byl. "Design and Evaluation of Skating Motions for a Dexterous Quadruped," ICRA 2018

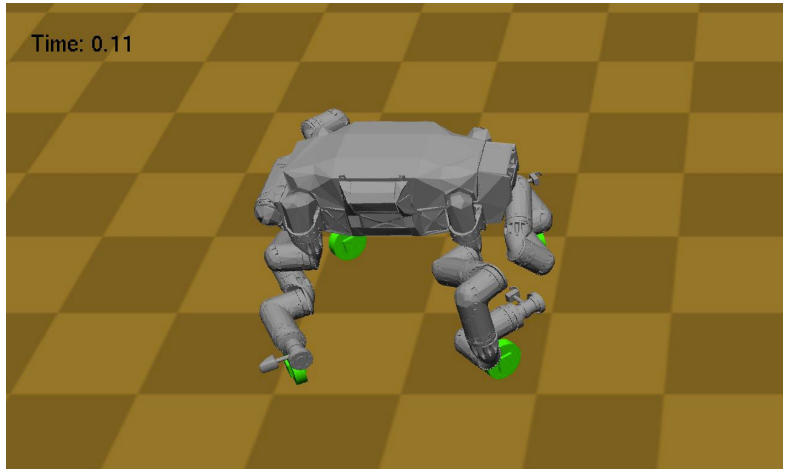
Other Agile Trajectories

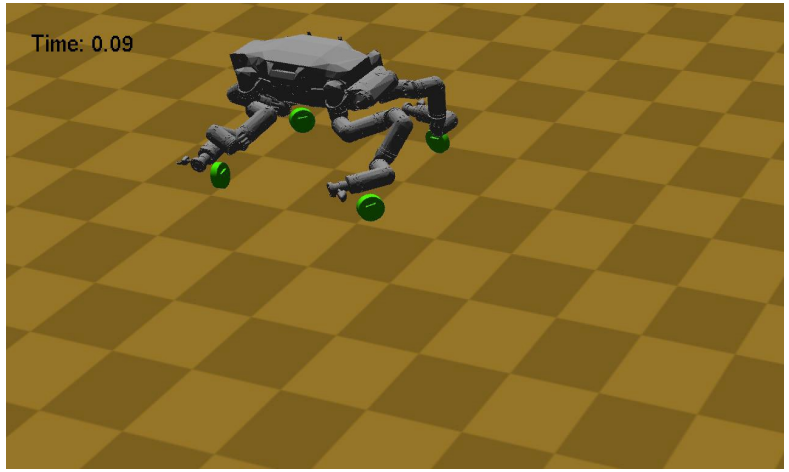






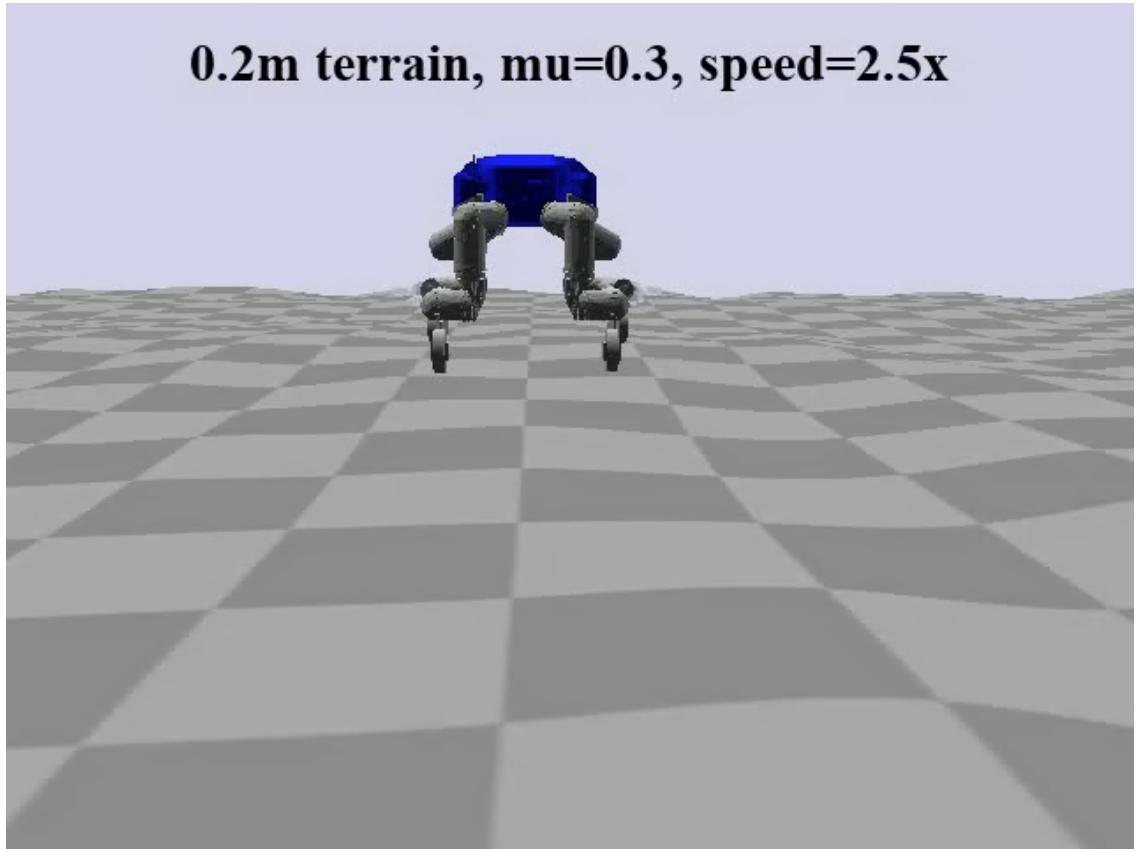


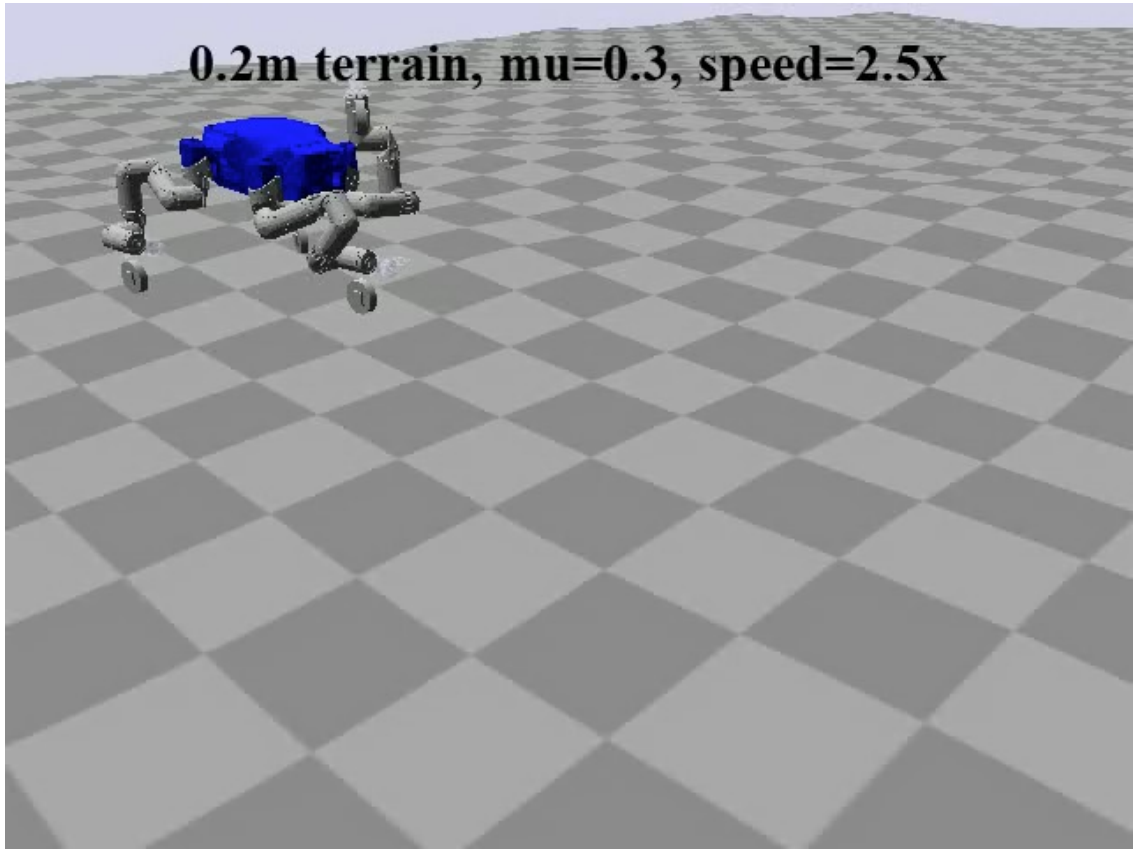




G. Bellegarda, K. van Teeffelen, K. Byl. “Design and Evaluation of Skating Motions for a Dexterous Quadruped,” ICRA 2018

Skating on 0.2 (m) “Rough” Terrain





G. Bellegarda, K. van Teeffelen, K. Byl. “Design and Evaluation of Skating Motions for a Dexterous Quadruped,” ICRA 2018

Outline

I. Motivation and Goals
II. Model-Predictive Control vs. Reinforcement Learning (same?)
III. Model-based methods
    I. Background on RoboSimian and designing skating motions
    II. Trajectory optimization allowing wheel slip
IV. Learning-based methods
    I. Action space in reinforcement learning [RoboSimian, A1]
    II. Augmenting motion planning with deep reinforcement learning [A1]
V. Bio-inspired learning
VI. Conclusion

Trajectory Optimization

$$\underset{t_0, t_F, x(t), u(t)}{\text{minimize}} \quad J\big(t_0, t_F, x(t_0), x(t_F)\big) + \int_{t_0}^{t_F} w\big(\tau, x(\tau), u(\tau)\big) d\tau$$

subject to	$\dot{x}(t) = f\big(t, x(t), u(t)\big)$	System Dynamics
	$h\big(t, x(t), u(t)\big) \leq 0$	Path Constraint
	$g\big(t_0, t_F, x(t_0), x(t_F)\big) \leq 0$	Boundary Constraint
	$x_{low} \leq x(t) \leq x_{upp}$	Path Bound on State
	$u_{low} \leq u(t) \leq u_{upp}$	Path Bound on Control
	$t_{low} \leq t_0 < t_F \leq t_{upp}$	Bounds on Initial and Final Times
	$x_{0,low} \leq x(t_0) \leq x_{0,upp}$	Bound on Initial State
	$x_{F,low} \leq x(t_F) \leq x_{F,upp}$	Bound on Final State

M. Kelly. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. SIAM 2017

Direct Collocation

$$\underset{t_0, t_F, x(t), u(t)}{\text{minimize}} \quad J\big(t_0, t_F, x(t_0), x(t_F)\big) + \int_{t_0}^{t_F} w\big(\tau, x(\tau), u(\tau)\big) d\tau$$

subject to	$\dot{x}(t) = f\big(t, x(t), u(t)\big)$	System Dynamics
	$h\big(t, x(t), u(t)\big) \leq 0$	Path Constraint
	$g\big(t_0, t_F, x(t_0), x(t_F)\big) \leq 0$	Boundary Constraint
	$x_{low} \leq x(t) \leq x_{upp}$	Path Bound on State
	$u_{low} \leq u(t) \leq u_{upp}$	Path Bound on Control
	$t_{low} \leq t_0 < t_F \leq t_{upp}$	Bounds on Initial and Final Times
	$x_{0,low} \leq x(t_0) \leq x_{0,upp}$	Bound on Initial State
	$x_{F,low} \leq x(t_F) \leq x_{F,upp}$	Bound on Final State

$$\underset{x_k, u_k;\ k=1\ldots N}{\text{minimize}} \quad J\big(x_N\big) + h \sum_{k=1}^{N} w\big(x_k, u_k\big)$$

$$\text{subject to} \quad d\big(x_k, u_k, x_{k+1}, u_{k+1}\big) = 0,\ k = 1, \ldots, N-1$$

$$\phi\big(x_k, u_k\big) = 0,\ k = 1, \ldots, N$$

$$\psi\big(x_k, u_k\big) \geq 0,\ k = 1, \ldots, N$$

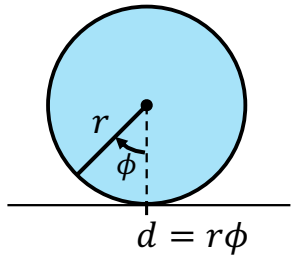
Backward Euler integration:

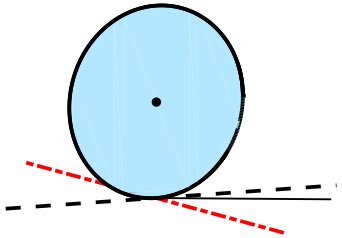
$$d\big(x_k, u_k, x_{k+1}, u_{k+1}\big) := x_{k+1} - \big(x_k + h f(x_{k+1}, u_{k+1})\big)$$

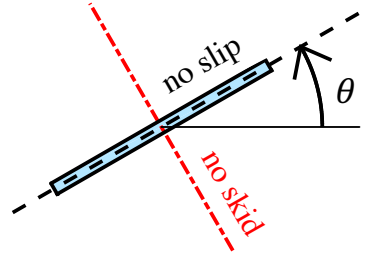
M. Posa, C. Cantu, R. Tedrake. A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact. IJRR 2013
M. Kelly. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. SIAM 2017

Wheel Model?

- Non-slip constraints can be violated!
- Account for slip, plan agile motions
- Existing wheel/tire models
  - i.e. Pacejka, Fiala, LuGre, Linear, Combined Slip, CarSim (?)





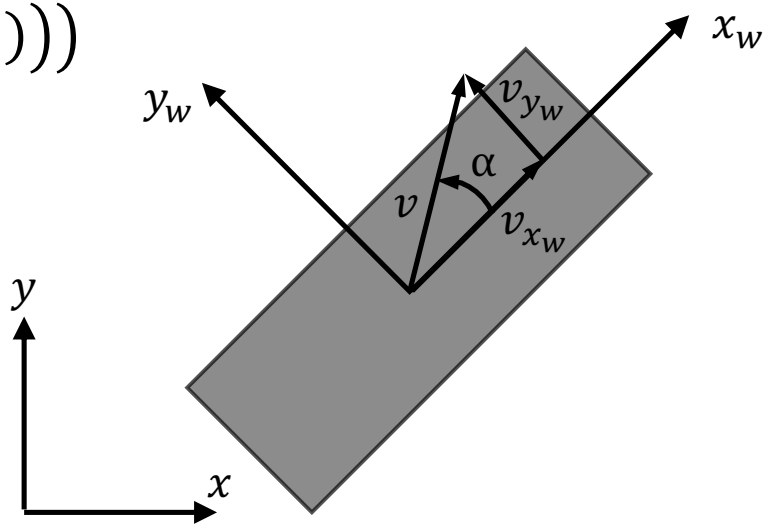


$$F_y = D \sin\left(C \tan^{-1}\left(B\alpha - E\left(B\alpha - \tan^{-1}(B\alpha)\right)\right)\right)$$

  - Slip angle α is difficult for optimization

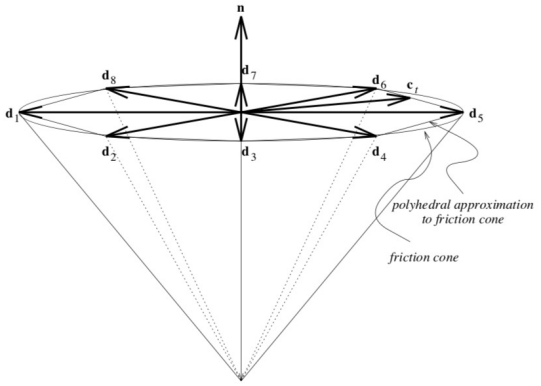
$$\alpha = \tan^{-1}\left(\frac{v_{y_w}}{v_{x_w}}\right)$$

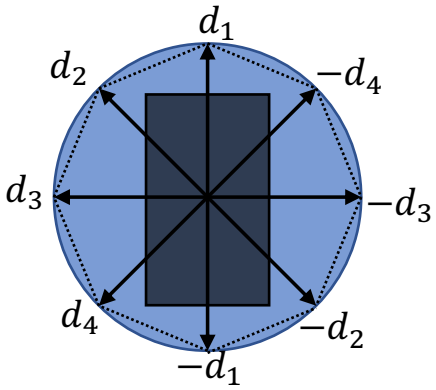
  - Many works thus use different models for planning and control

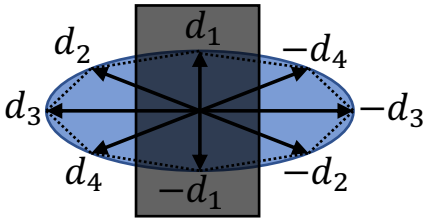


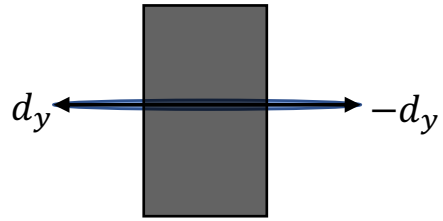
G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019
G. Bellegarda and K. Byl. "Versatile Trajectory Optimization Using a LCP Wheel Model for Dynamic Vehicle Maneuvers," ICRA 2020

Friction as a Linear Complementarity Problem (LCP)









- Polyhedral friction cone/triangle: $\mathcal{F}(q) = \{ F_n n + D\beta \mid F_n \geq 0,\ \beta \geq 0,\ e^T\beta \leq \mu F_n \}$

- Inequalities:

$$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$$
$$\mu F_n - e^T \beta \geq 0, \quad \gamma \geq 0$$

- Complementarity conditions:

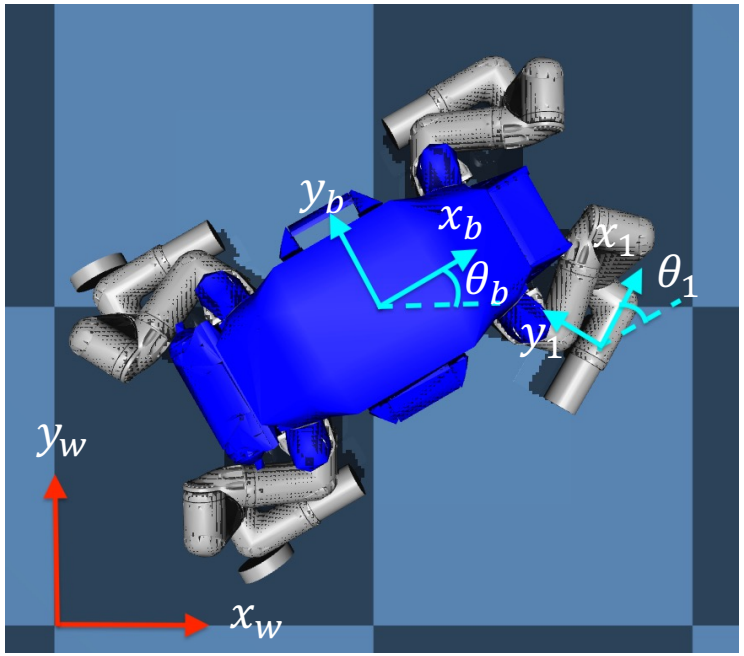
$$(\gamma e + D^T v^{k+1})^T \beta = 0$$
$$(\mu F_n - e^T \beta)\gamma = 0$$

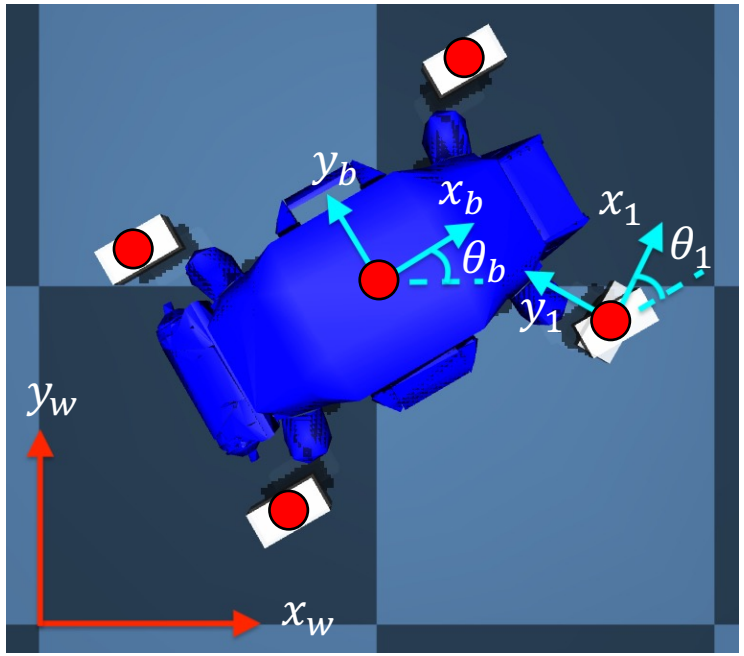
- F_n is the magnitude of the normal contact force
- D is a matrix of direction vectors
- $\beta \in \mathbb{R}^j$ is a vector of weights
- $e = [1, 1, ..., 1]^T \in \mathbb{R}^j$
- v^{k+1} is the global 2D planar velocity vector of the contact point at the end of the next time step
- γ is a scalar roughly equal to the magnitude of the relative tangential velocity at a contact

D. Stewart and J.C. Trinkle. An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction. ICRA 2000

G. Bellegarda and K. Byl. "Versatile Trajectory Optimization Using a LCP Wheel Model for Dynamic Vehicle Maneuvers," ICRA 2020

Additional Modeling Details





- Body has point mass m_b and inertia J_b, and each skate has point mass m_i and inertia J_i
- Each floating skate's coordinates are relative to the body frame
- $q = [x_b, y_b, z_b, \theta_b, x_i, y_i, z_i, \theta_i]^T \in \mathbb{R}^{20}$ where $i \in \{1, 2, 3, 4\}$
- $u = [u_{x_i}, u_{y_i}, u_{z_i}, u_{\theta_i}]^T \in \mathbb{R}^{16}$ where $i \in \{1, 2, 3, 4\}$, since we can set arbitrary configurations with inverse kinematics

G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

Trajectory Optimization Framework

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$$\phi(q, \dot{q}, u, F_n) = 0$$

$$\psi(q, \dot{q}, u, F_n) \geq 0$$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$$

$$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$$

$$(\gamma e + D^T v^{k+1})^T \beta = 0$$

$$(\mu F_{n_i} - e^T\beta)\gamma = 0$$

$$F_{fric_i} = D\beta$$

- ZMP Constraints (RoboSimian Only)
- Gait Constraints (RoboSimian Only)

G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

Trajectory Optimization Framework

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$$\phi(q, \dot{q}, u, F_n) = 0$$

$$\psi(q, \dot{q}, u, F_n) \geq 0$$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$$

$$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$$

$$(\gamma e + D^T v^{k+1})^T \beta = 0$$

$$(\mu F_{n_i} - e^T\beta)\gamma = 0$$

$$F_{fric_i} = D\beta$$

- ZMP Constraints (RoboSimian Only)
- Gait Constraints (RoboSimian Only)

End Location Power

$$J = J_F(q_N, \dot{q}_N) + h\sum_{k=1}^{N}\sqrt{\left(\dot{q}_k^T U_k\right)^2 + \epsilon}$$

with

- $U = [0, 0, 0, 0, u_{x_i}, u_{y_i}, u_{z_i}, u_{\theta_i}]^T \in \mathbb{R}^{20}$ for $i \in \{1, 2, 3, 4\}$
- $\epsilon = 10^{-6}$ is a regularization term to help smooth the cost function
- $h = \Delta t$ is the time interval between time steps
- $J_F(q_N, \dot{q}_N)$ is a distance measure to a set of goal coordinates, such as:

$$J_F = \alpha_x(x_g - x_N)^2 + \alpha_y(y_g - y_N)^2 + \alpha_\theta(\theta_g - \theta_N)^2$$

M. Srinivasan, *Why walk and run: energetic costs and energetic optimality in simple mechanics-based models of a bipedal animal.* Cornell University, Ithaca, NY, 2006.

G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

Trajectory Optimization Framework

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$\phi(q, \dot{q}, u, F_n) = 0$

$\psi(q, \dot{q}, u, F_n) \geq 0$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$

$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$

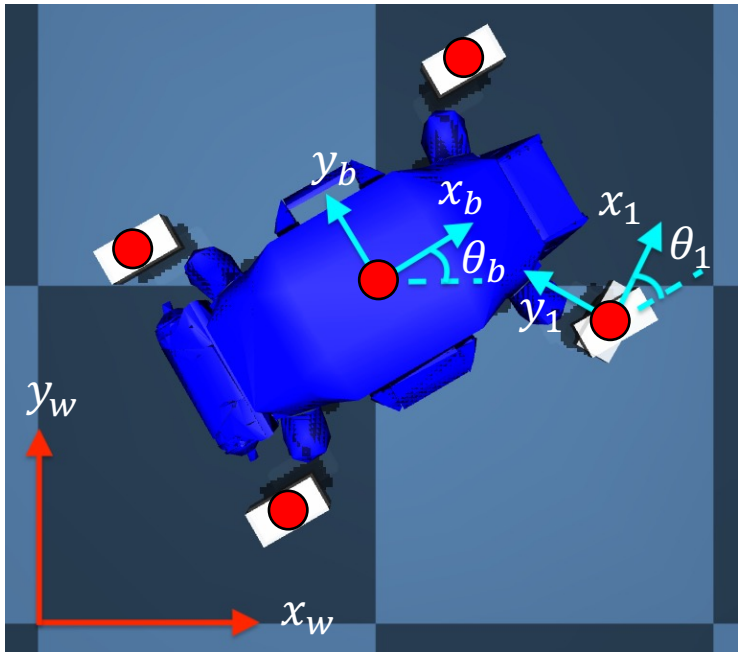
$(\gamma e + D^T v^{k+1})^T \beta = 0$

$(\mu F_{n_i} - e^T\beta)\gamma = 0$

$F_{fric_i} = D\beta$

- ZMP Constraints (RoboSimian Only)
- Gait Constraints (RoboSimian Only)

- $q, \dot{q}$ are bounded by ranges for placing each skate with inverse kinematics, as well as by joint limits
- u is bounded explicitly, as well as implicitly by $\dot{q}$ ranges
- Known contact sequence, where $F_n \geq 0$ for each skate in contact, and $F_n = 0$ for skates not in contact



G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

Trajectory Optimization Framework

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$\phi(q, \dot{q}, u, F_n) = 0$

$\psi(q, \dot{q}, u, F_n) \geq 0$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$

$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$

$(\gamma e + D^T v^{k+1})^T \beta = 0$

$(\mu F_{n_i} - e^T\beta)\gamma = 0$

$F_{fric_i} = D\beta$

- ZMP Constraints (RoboSimian Only)

- Gait Constraints (RoboSimian Only)

$$q_{k+1} = q_k + h\dot{q}_{k+1}$$
$$\dot{q}_{k+1} = \dot{q}_k + h\ddot{q}_{k+1}$$

with

$$\ddot{q}_{k+1} = M_{k+1}^{-1}(B_{k+1}u_{k+1} + F_{n_{k+1}} + J_{k+1}^T F_{fric_{k+1}} - C_{k+1}\dot{q}_{k+1} - G_{k+1})$$

G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

Trajectory Optimization Framework

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$\phi(q, \dot{q}, u, F_n) = 0$

$\psi(q, \dot{q}, u, F_n) \geq 0$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$

$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$

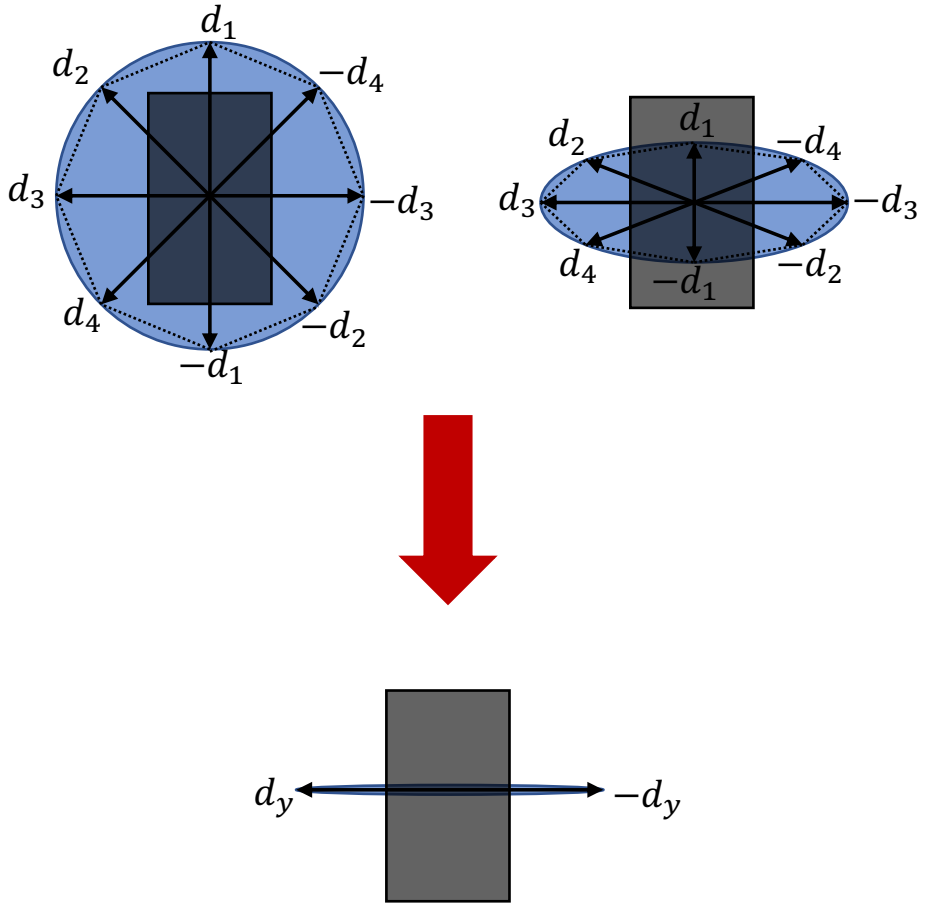
$(\gamma e + D^T v^{k+1})^T \beta = 0$

$(\mu F_{n_i} - e^T\beta)\gamma = 0$

$F_{fric_i} = D\beta$

- ZMP Constraints (RoboSimian Only)

- Gait Constraints (RoboSimian Only)



G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

G. Bellegarda and K. Byl. "Versatile Trajectory Optimization Using a LCP Wheel Model for Dynamic Vehicle Maneuvers," ICRA 2020

Trajectory Optimization Framework

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$\phi(q, \dot{q}, u, F_n) = 0$

$\psi(q, \dot{q}, u, F_n) \geq 0$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$

$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$

$(\gamma e + D^T v^{k+1})^T \beta = 0$

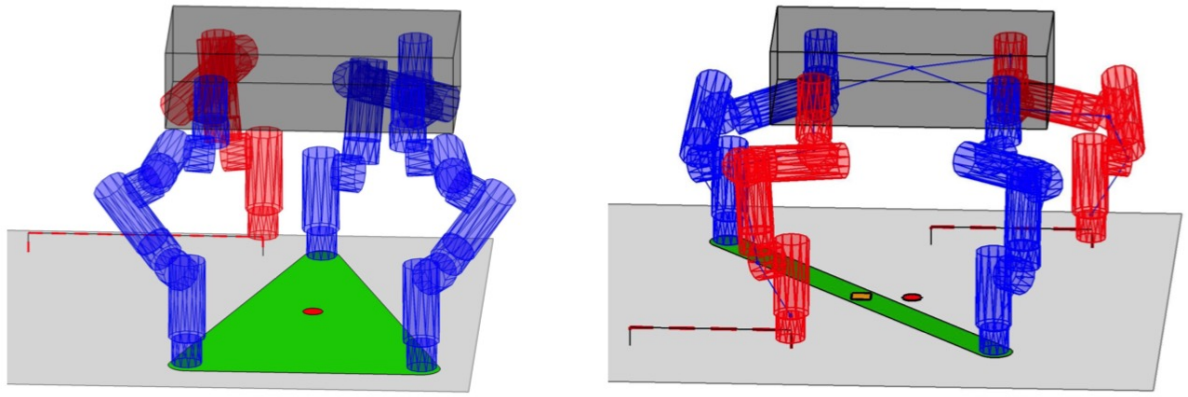
$(\mu F_{n_i} - e^T\beta)\gamma = 0$

$F_{fric_i} = D\beta$

- ZMP Constraints (RoboSimian Only)

- Gait Constraints (RoboSimian Only)

Zero-Moment Point (ZMP): point on ground where the sum of all moments of the active forces is equal to zero



M. Vukobratovic, B. A. Borovac. Zero-Moment Point – Thirty Five Years of its Life. IJHR 2004

P. Ha and K. Byl. Feasibility and Optimization of Fast Quadruped Walking with One- Versus Two-at-a-Time Swing Leg Motions for RoboSimian. 2014

Trajectory Optimization Framework

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$$\phi(q, \dot{q}, u, F_n) = 0$$

$$\psi(q, \dot{q}, u, F_n) \geq 0$$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$$

$$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$$

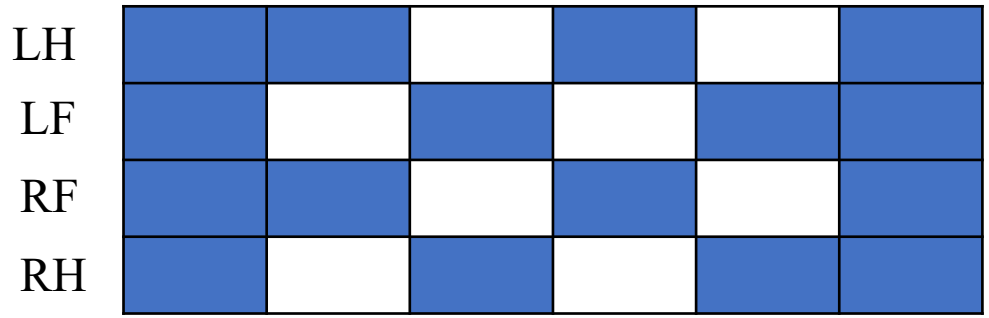
$$(\gamma e + D^T v^{k+1})^T \beta = 0$$

$$(\mu F_{n_i} - e^T\beta)\gamma = 0$$

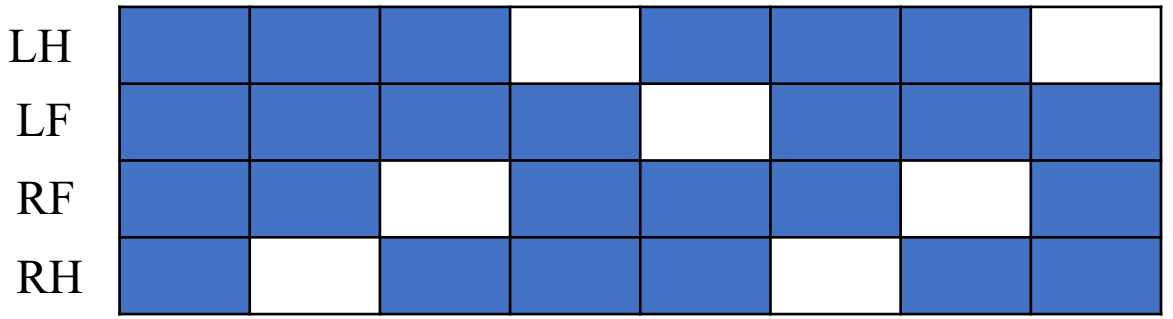
$$F_{fric_i} = D\beta$$

- ZMP Constraints (RoboSimian Only)
- Gait Constraints (RoboSimian Only)

Static Trotting Gait

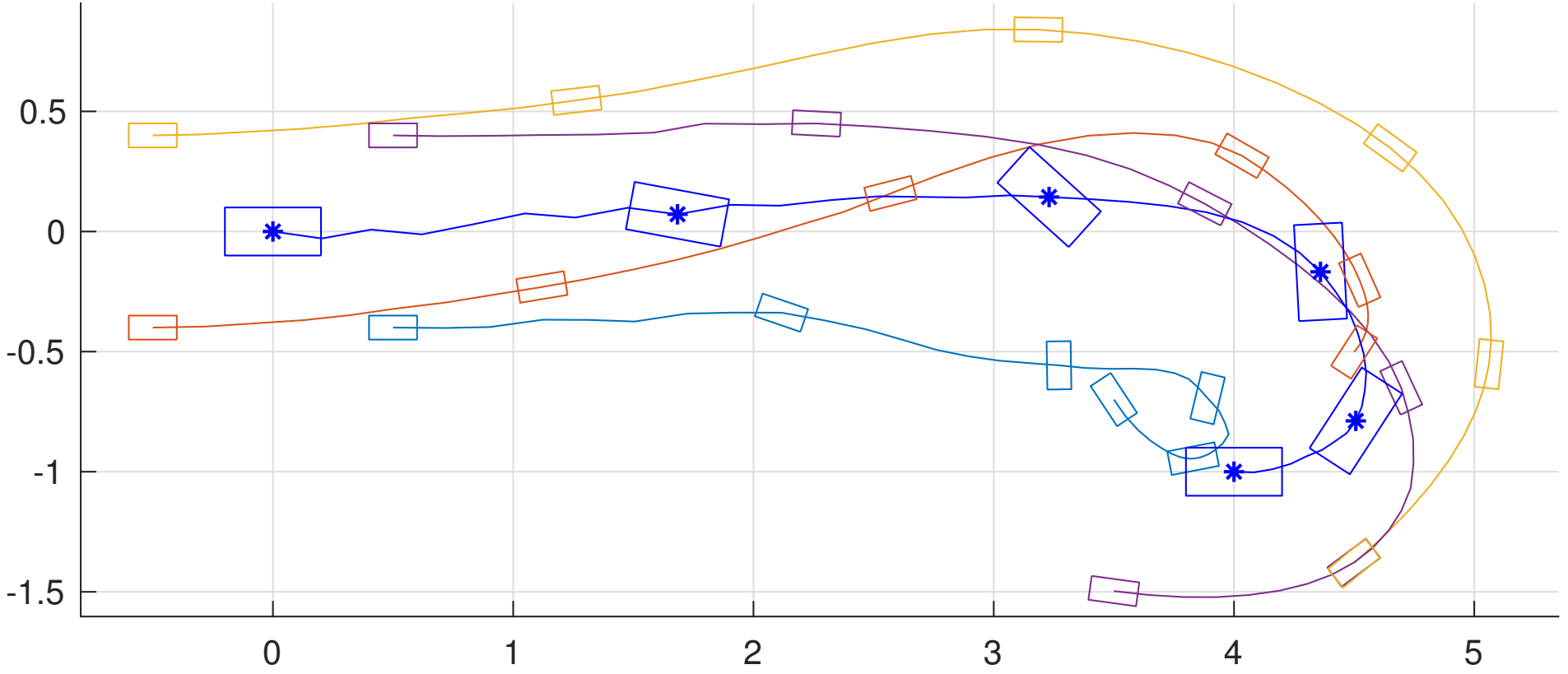


Static Walking Gait



Dynamic Parking Maneuver

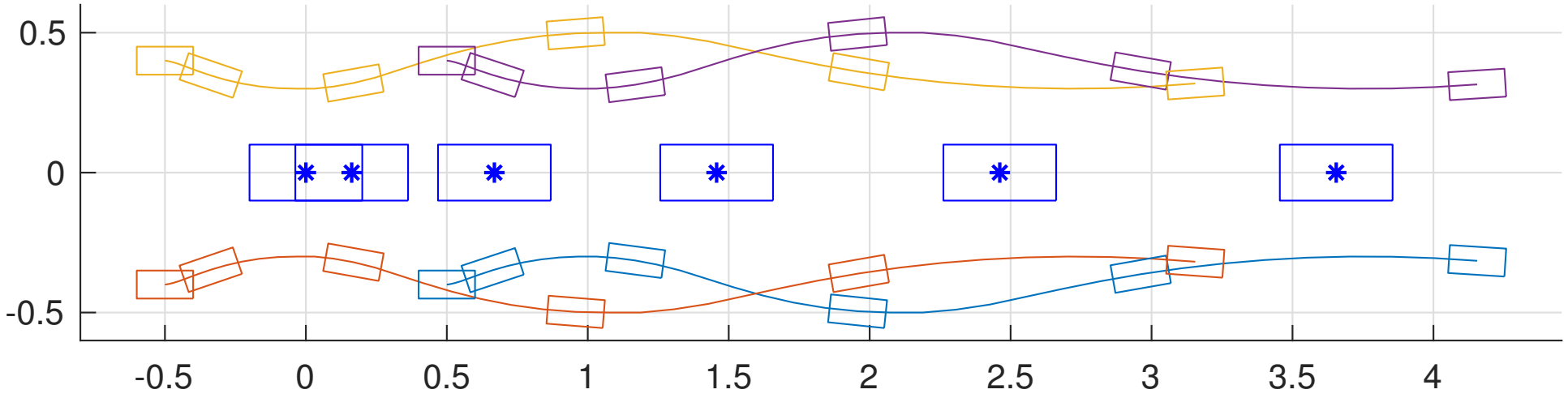
- Goal: "park" at $(x_b, y_b, \theta_b) = (4, -1, -\pi)$, initialized at (0,0,0) with $\dot{x}_b = 4$ (m/s)



G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

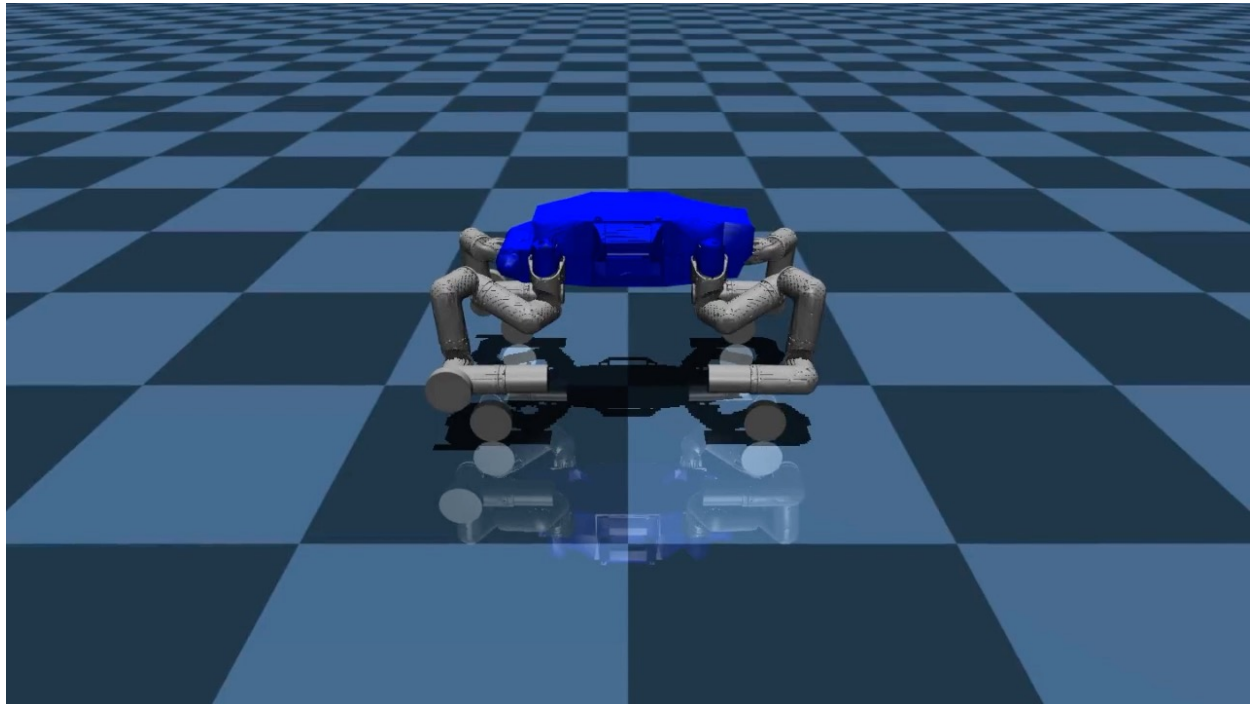
Energy-Efficient Forward Locomotion

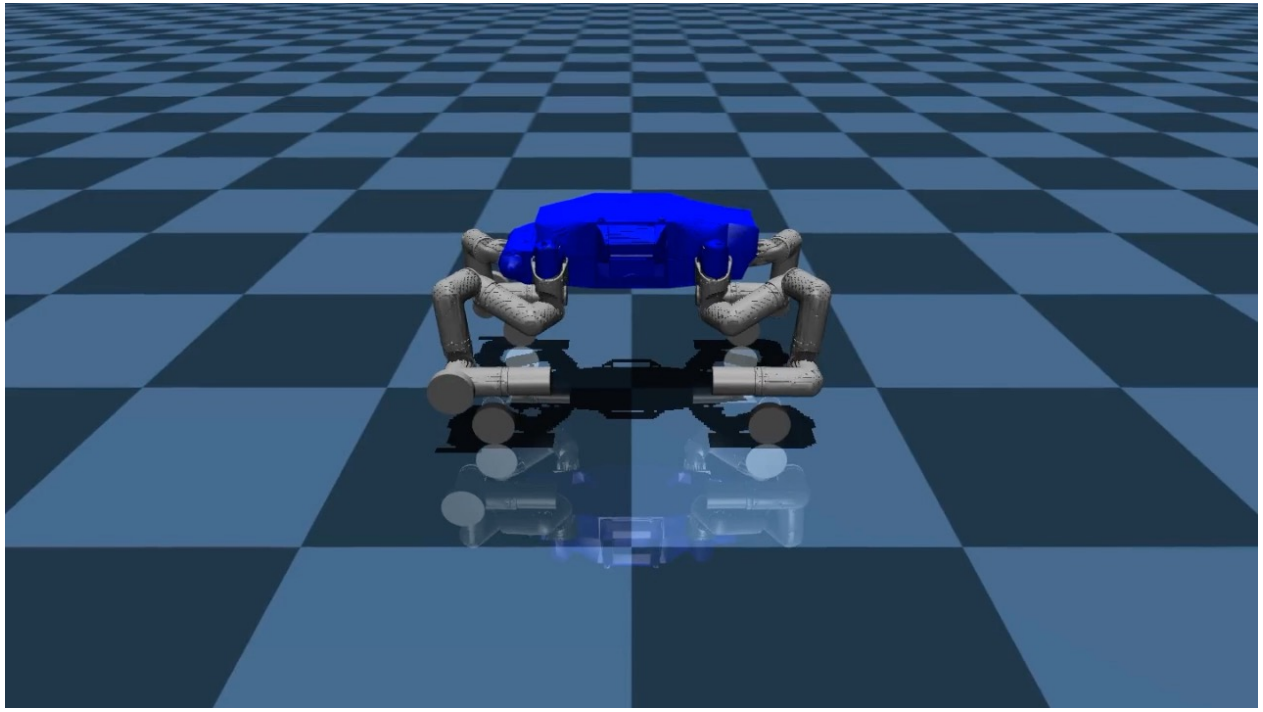
- Maximize final body position in the x-direction, while minimizing energy use



G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

Hybrid Rolling-Walking/Trotting (Animation!)

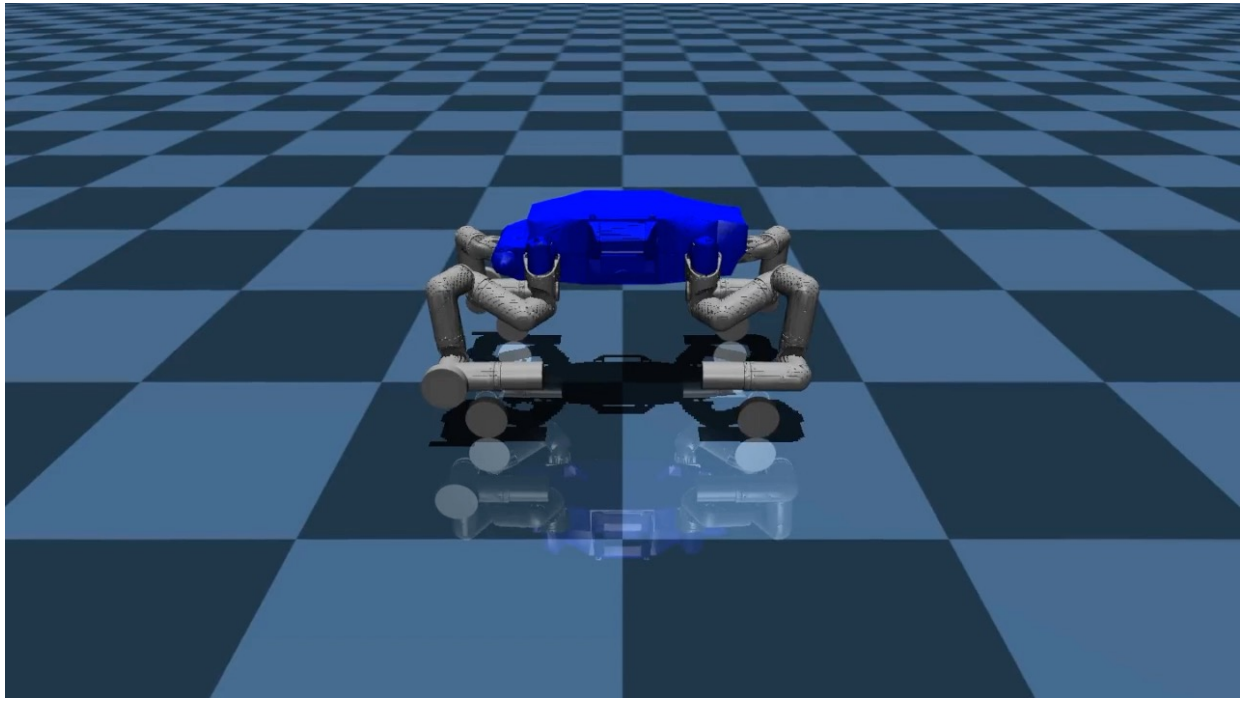


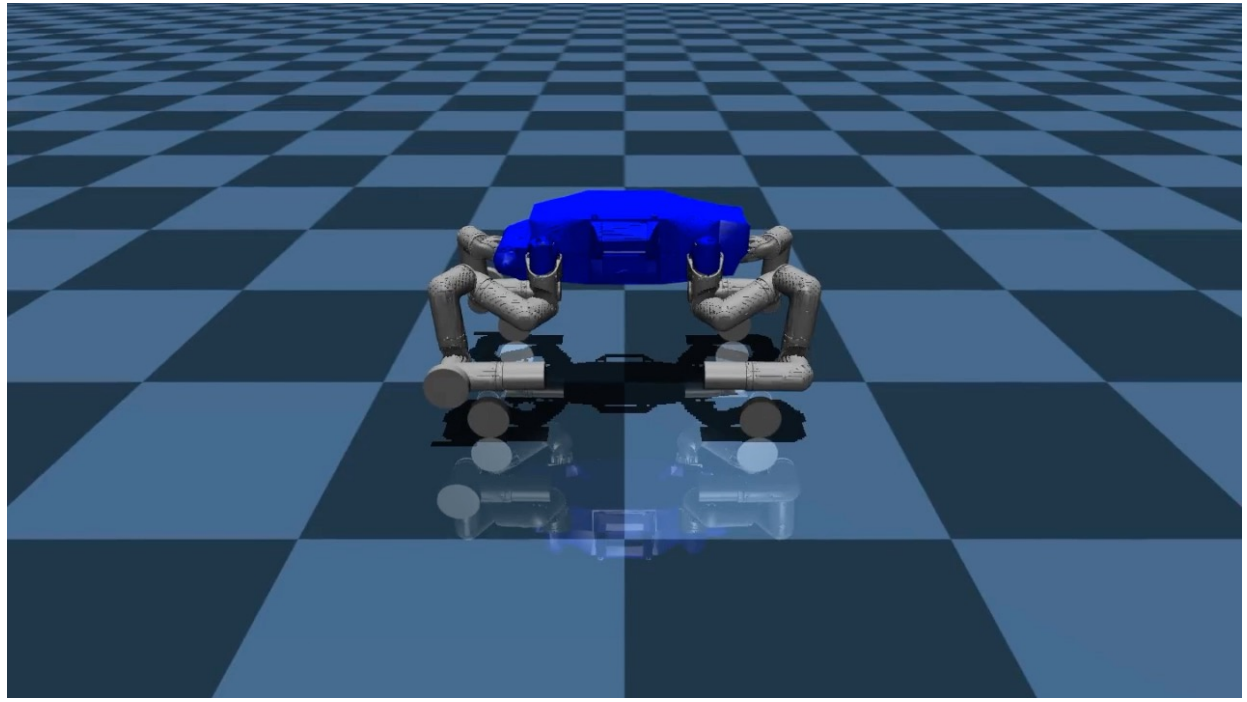


G. Bellegarda and K. Byl. “Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip,” CDC 2019

Tracking Hybrid Rolling-Walking/Trotting

- Currently: calculate inverse kinematics along trajectories, joint PD control
- Dynamics mismatch due to RoboSimian's heavy limbs
- Need a whole-body controller for better tracking





G. Bellegarda and K. Byl. "Trajectory Optimization for a Wheel-Legged System for Dynamic Maneuvers that Allow for Wheel Slip," CDC 2019

Applying the Optimization Framework to a Model Car

find $q, \dot{q}, u, F_n, F_{fric}$ at discrete timesteps $k = 1...N$

subject to minimize cost J

- State Constraints:

$\phi(q, \dot{q}, u, F_n) = 0$

$\psi(q, \dot{q}, u, F_n) \geq 0$

- Dynamics Constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + A(q)^T\lambda = B(q)u + F_n + J(q)^T F_{fric}$$

- Friction Constraints (for each contact i):

$\gamma e + D^T v^{k+1} \geq 0, \quad \beta \geq 0$

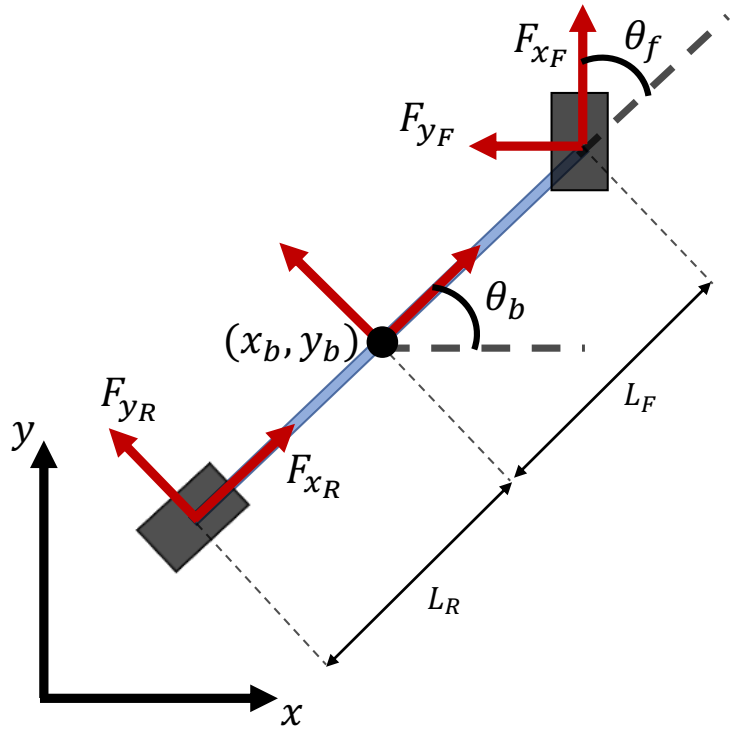
$\mu F_{n_i} - e^T\beta \geq 0, \quad \gamma \geq 0$

$(\gamma e + D^T v^{k+1})^T \beta = 0$

$(\mu F_{n_i} - e^T\beta)\gamma = 0$

$F_{fric_i} = D\beta$

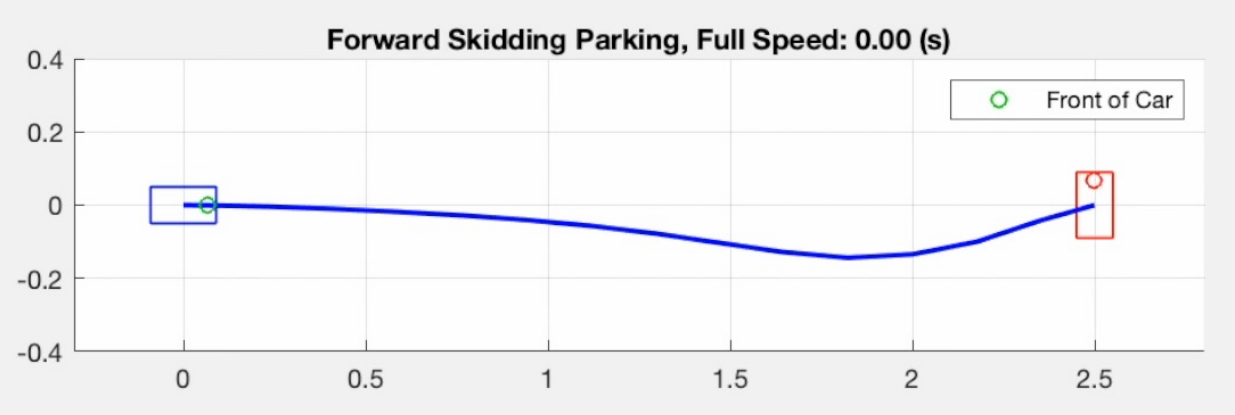
- ZMP Constraints (RoboSimian Only)
- Gait Constraints (RoboSimian Only)



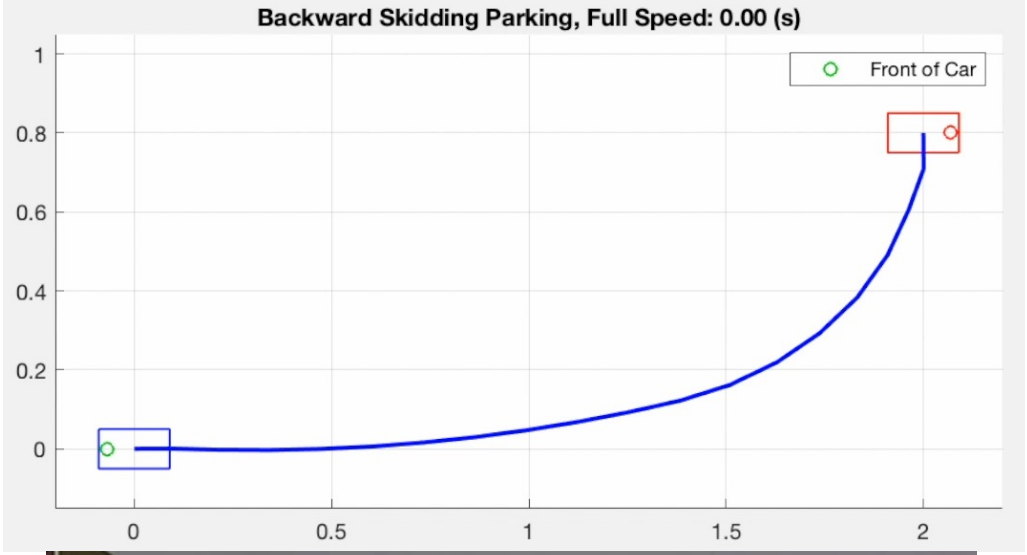
G. Bellegarda and K. Byl. "Versatile Trajectory Optimization Using a LCP Wheel Model for Dynamic Vehicle Maneuvers," ICRA 2020

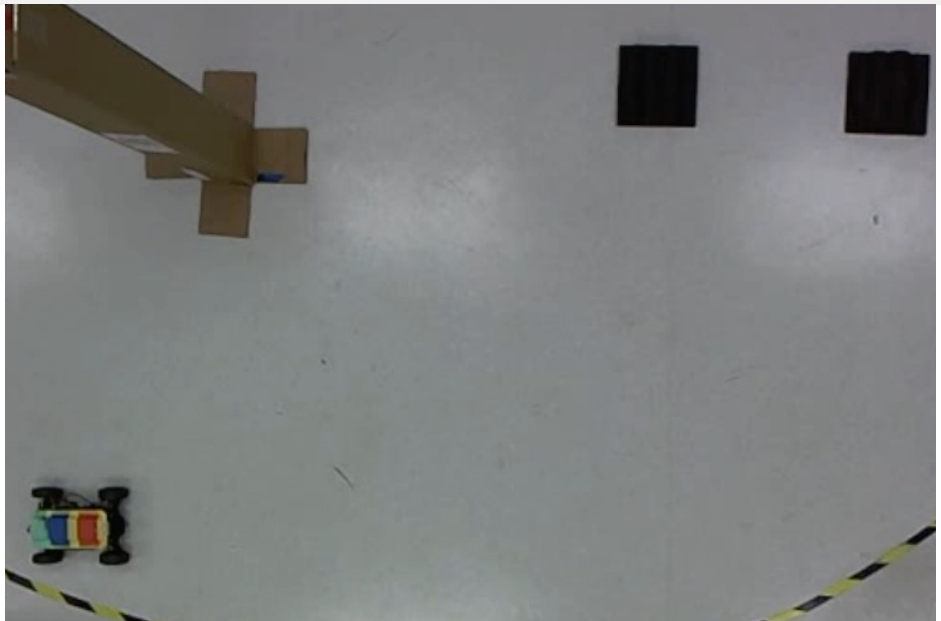
Dynamic Parking Maneuvers

- Park at $(x_b, y_b, \theta_b) = (2.5,0,\frac{\pi}{2})$, in 0.75 (s)



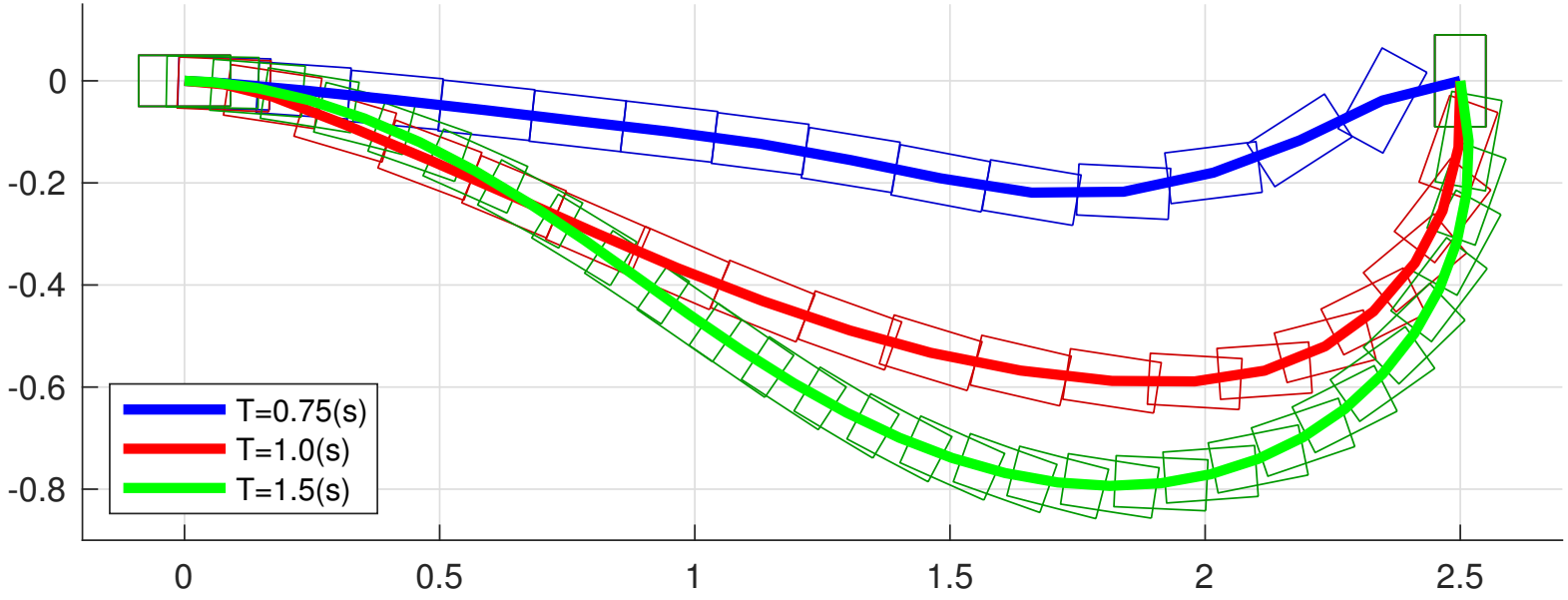
- Park at $(x_b, y_b, \theta_b) = (2,0.8,0)$, in 1.0 (s)





Framework Versatility

- Task is still to "park" at $(x_b, y_b, \theta_b) = \left(2.5, 0, \frac{\pi}{2}\right)$
- By varying the time horizon, the framework naturally discovers trajectories that either exploit (short time horizons), or avoid slipping



G. Bellegarda and K. Byl. "Versatile Trajectory Optimization Using a LCP Wheel Model for Dynamic Vehicle Maneuvers," ICRA 2020

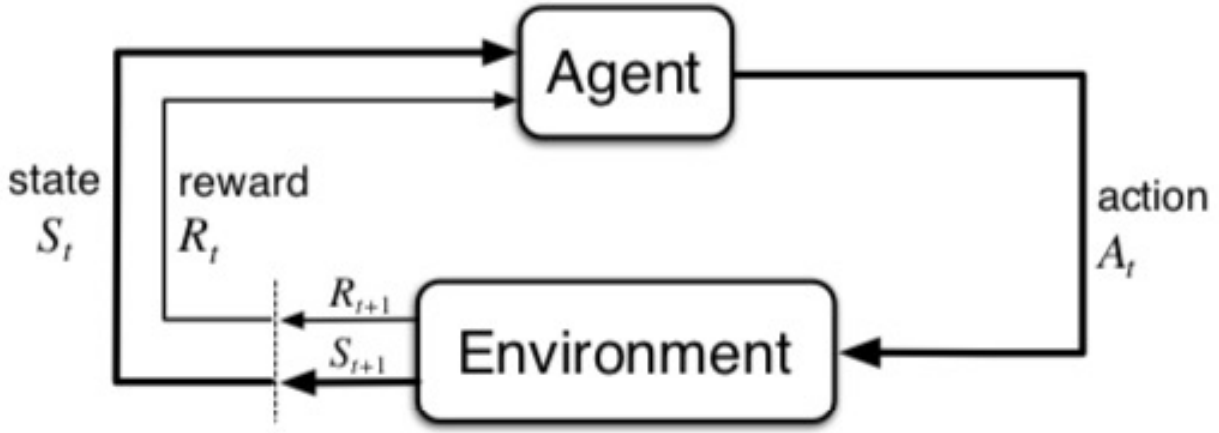
Outline

I. Motivation and Goals
II. Model-Predictive Control vs. Reinforcement Learning (same?)
III. Model-based methods
    I. Background on RoboSimian and designing skating motions
    II. Trajectory optimization allowing wheel slip
IV. Learning-based methods
    I. Action space in reinforcement learning [RoboSimian, A1]
    II. Augmenting motion planning with deep reinforcement learning [A1]
V. Bio-inspired learning
VI. Conclusion

Reinforcement Learning as a 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

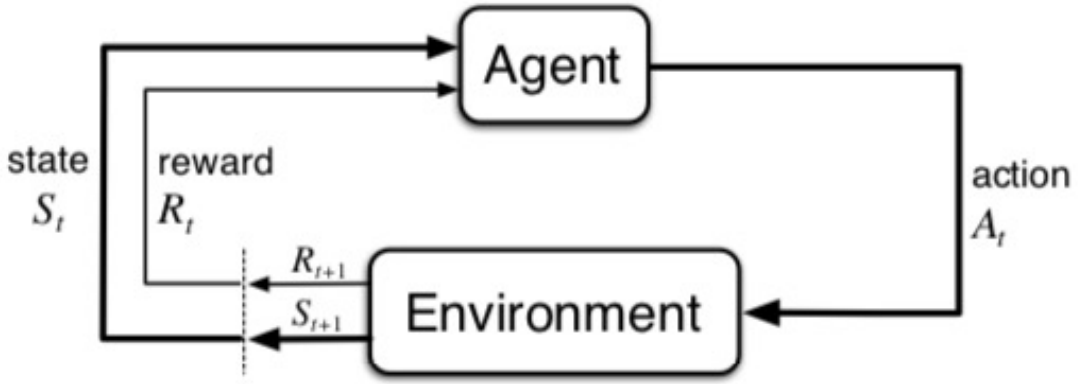


R. Sutton and A. Barto. Introduction to Reinforcement Learning. MIT Press 1998
Deep RL Bootcamp. Berkeley CA, August 2017
S. Levine. Deep Reinforcement Learning Course. https://rail.eecs.berkeley.edu/deeprlcourse/

Reinforcement Learning as a 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



- 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

Many Existing Tools for Reinforcement Learning

- RL algorithm implementations
  - 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/
  - … 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!

Reinforcement Learning Considerations

Algorithm

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

…implementation dependent!

MDP Design Decisions

- Observation space
- Action space
- Reward function

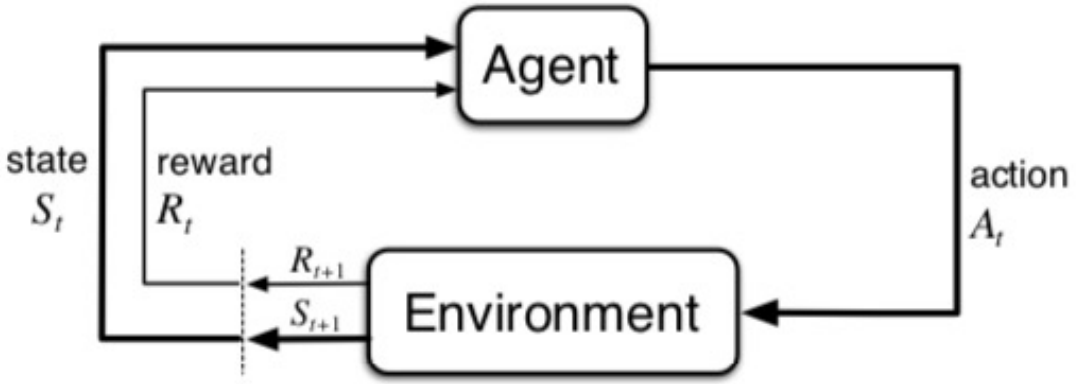
Environment Parameters

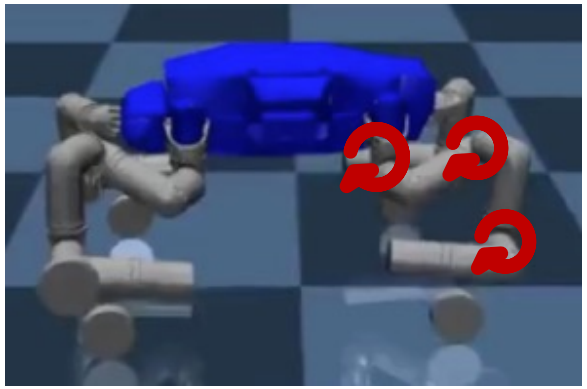
- Simulator dynamics
- Control gains – joint/Cartesian
- Control/environment time step
- Noise, latency

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

State/Action/Reward Space?

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





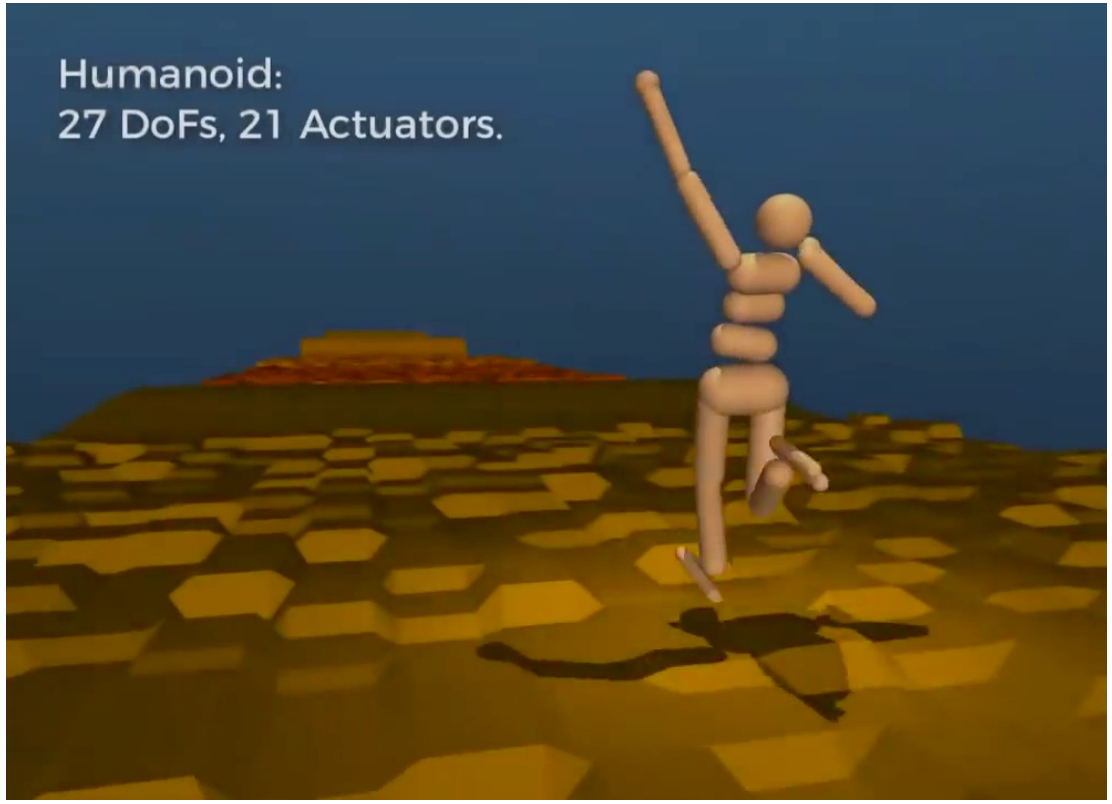
a_t ?
-motor positions/torques

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

G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019

Training in Joint Space (PPO)

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

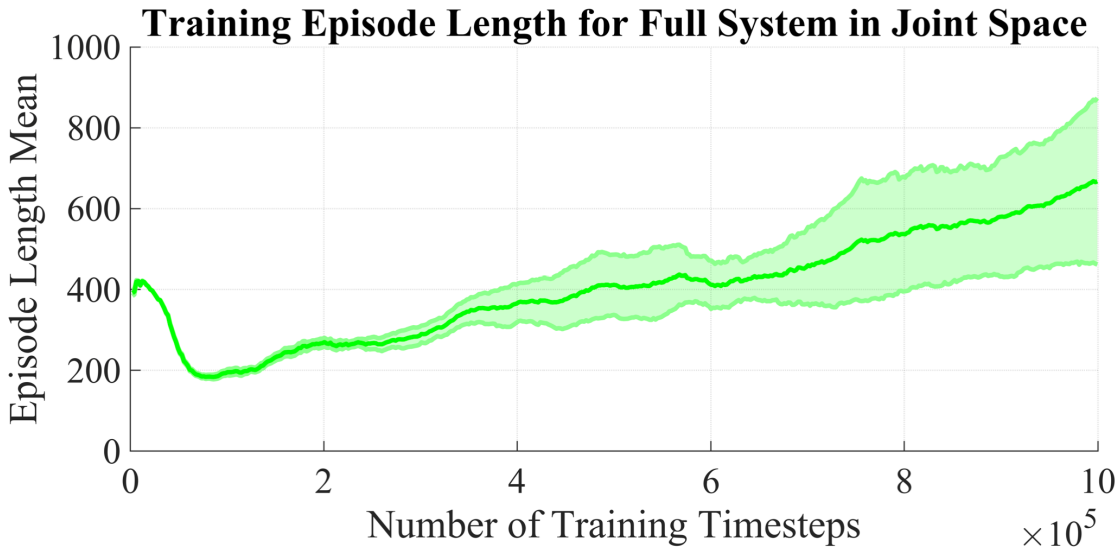


N. Heess et al. Emergence of Locomotion Behaviours in Rich Environments. *arXiv:1707.02286*, 2017
J. Schulman et al. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017
G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019

Training in Joint Space (PPO)

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

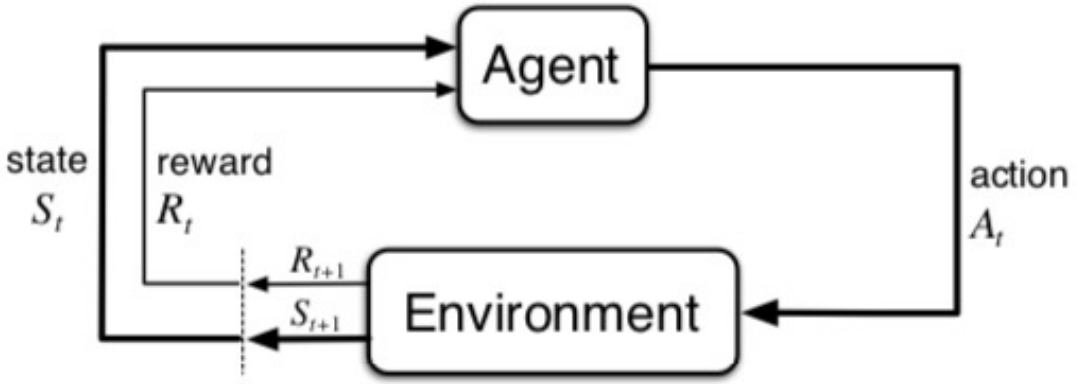
- Agent is essentially forced to learn forward and inverse kinematics!

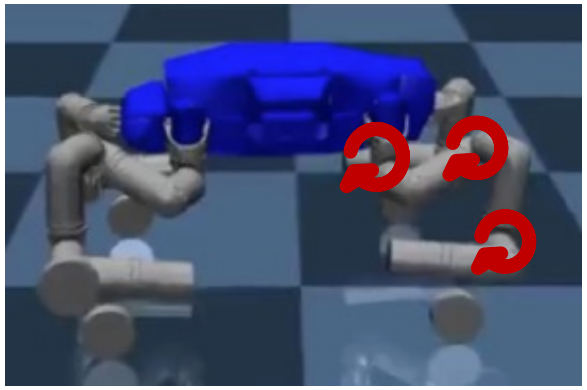


N. Heess et al. Emergence of Locomotion Behaviours in RichEnvironments.*arXiv:1707.02286*, 2017
J. Schulman et al. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017
G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019

State/Action/Reward Space?

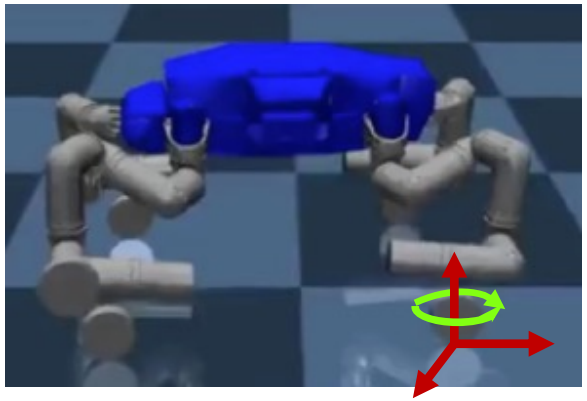
s_t ? i.e.
-body (z, r, p, y)
-body velocities
-joint states



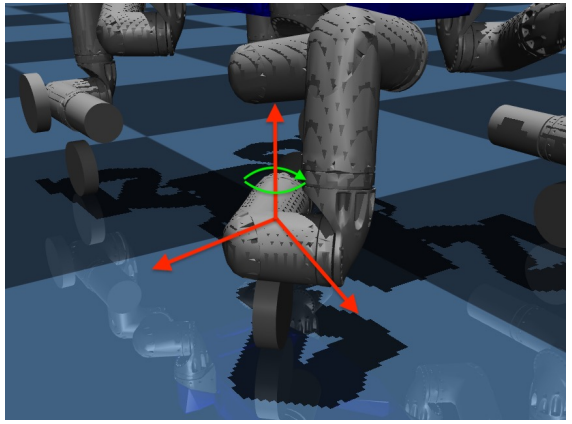


a_t ?
-motor positions/torques
-Cartesian coordinates

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

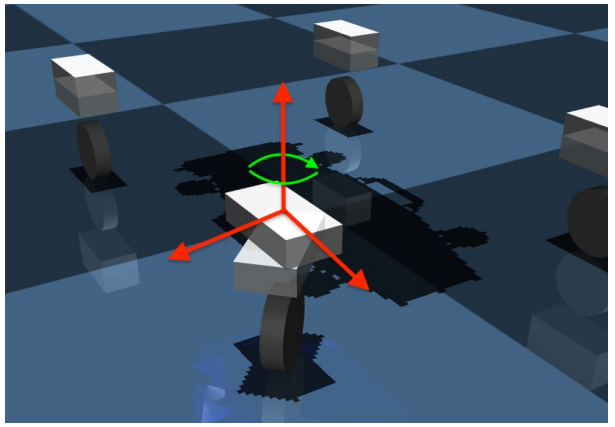


G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019



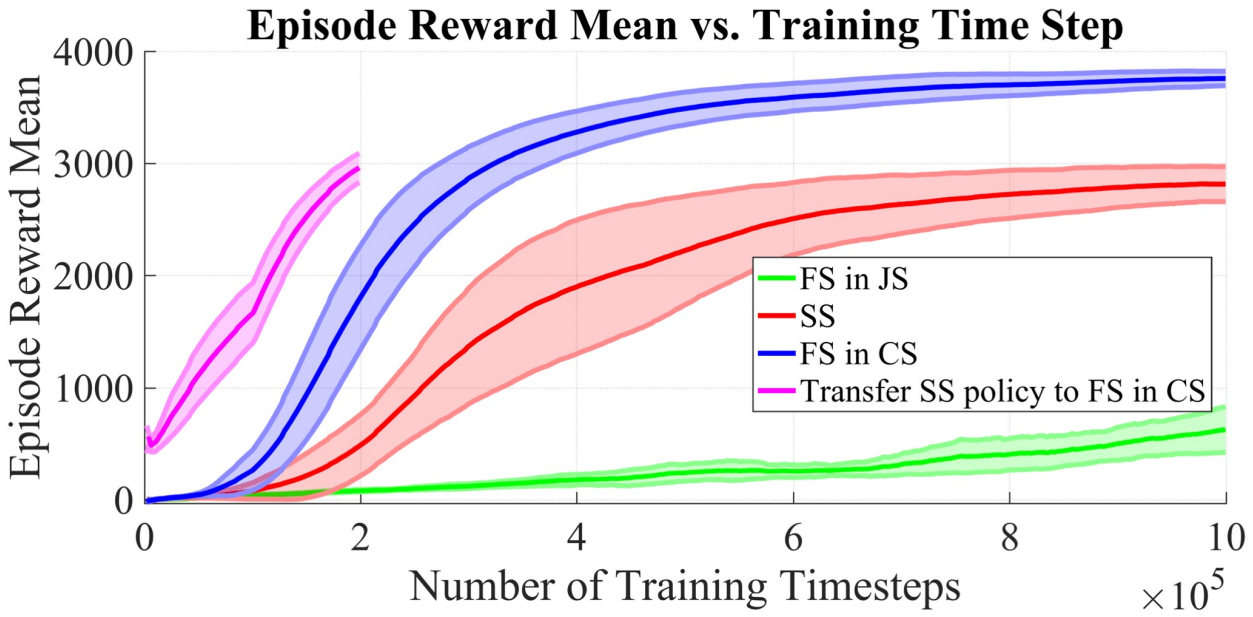
Training in Task Space (PPO)

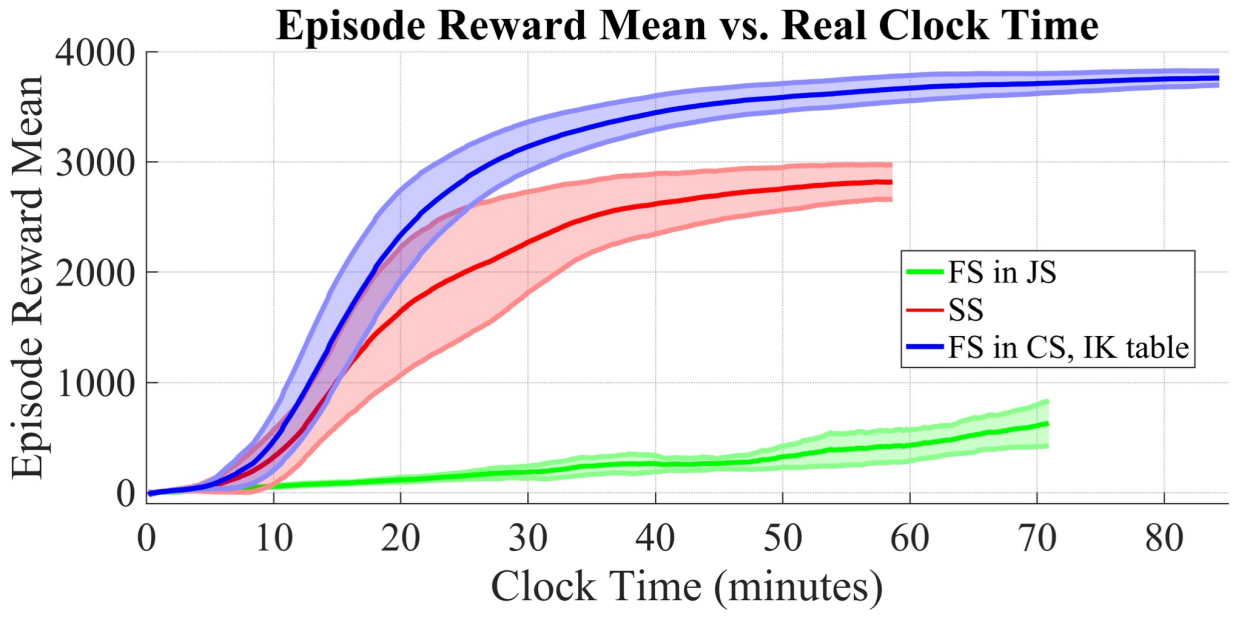
Action Space: $a_t = [y_{ee_i}, \phi_{ee_i}]$ for $i \in \{1,2,3,4\}$



G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019

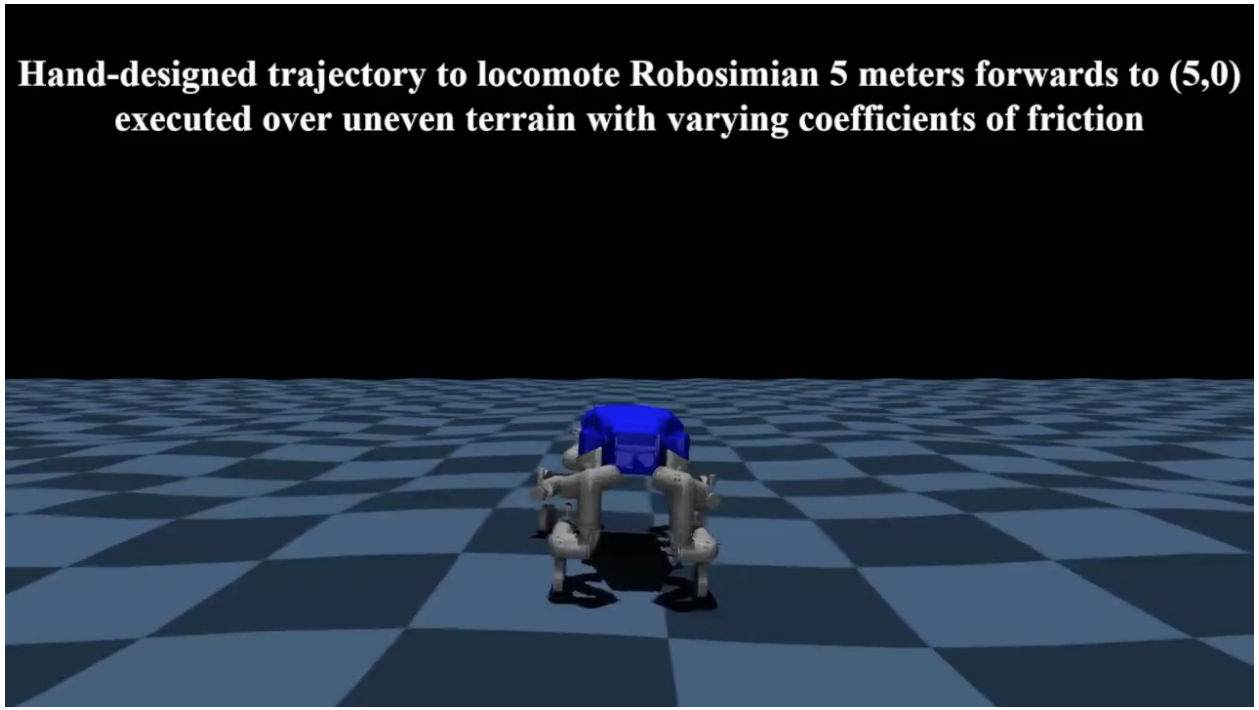
Training in Joint Space vs. Task Space

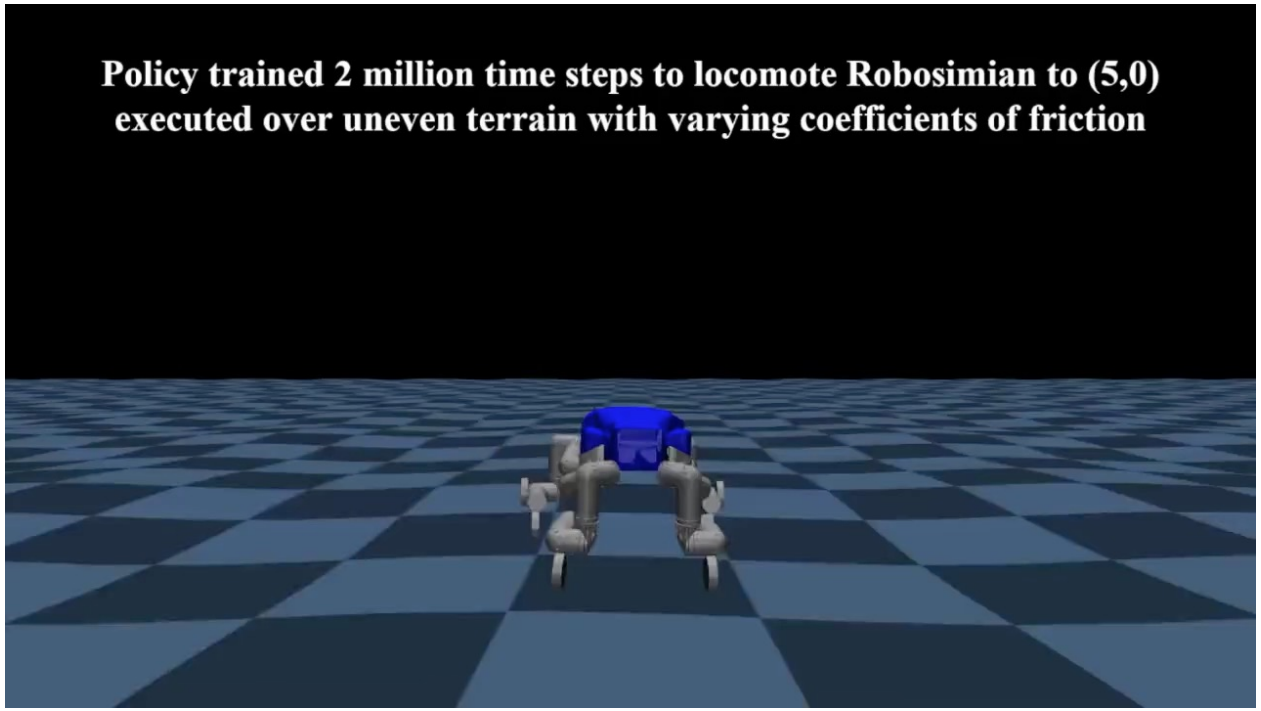




G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019

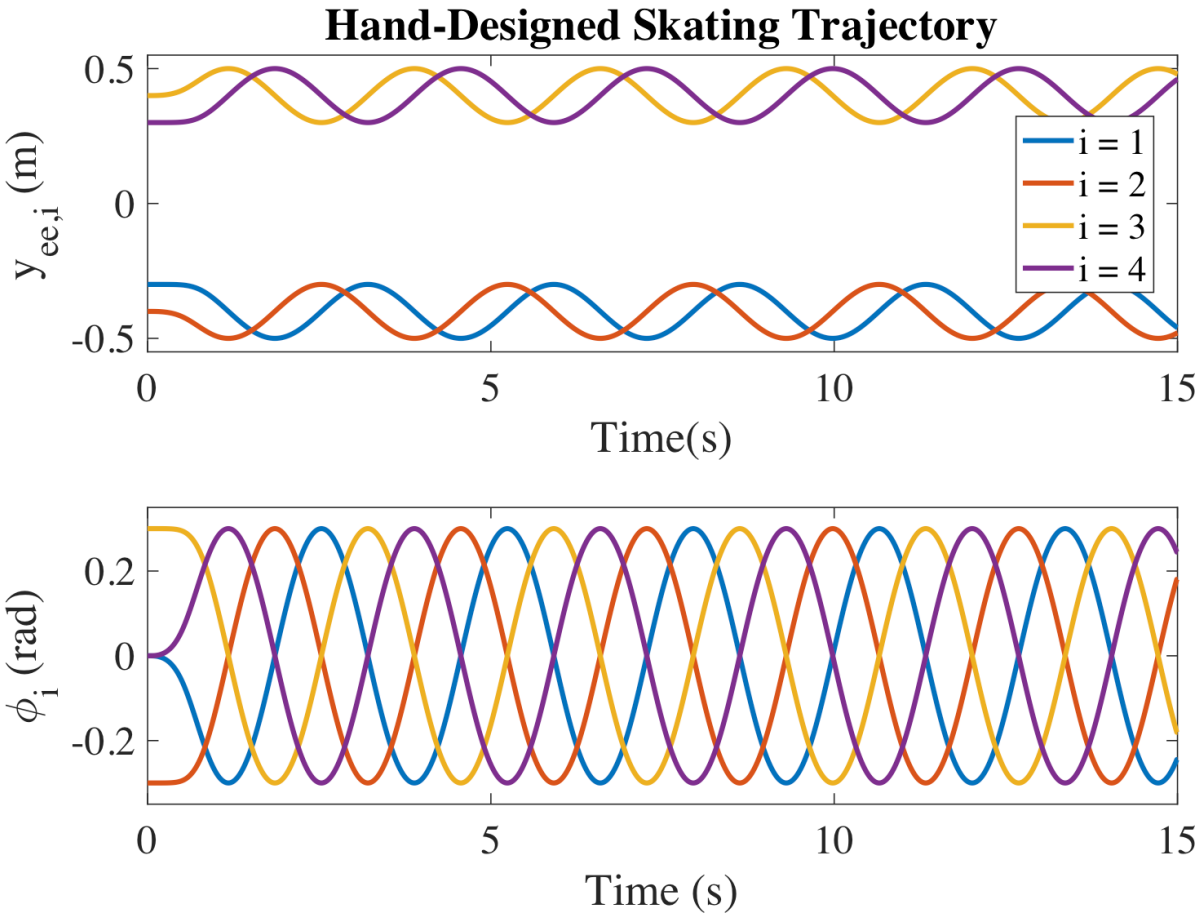
Hand-designed Trajectory vs. Trained Policy over uneven terrain with varying coefficients of friction

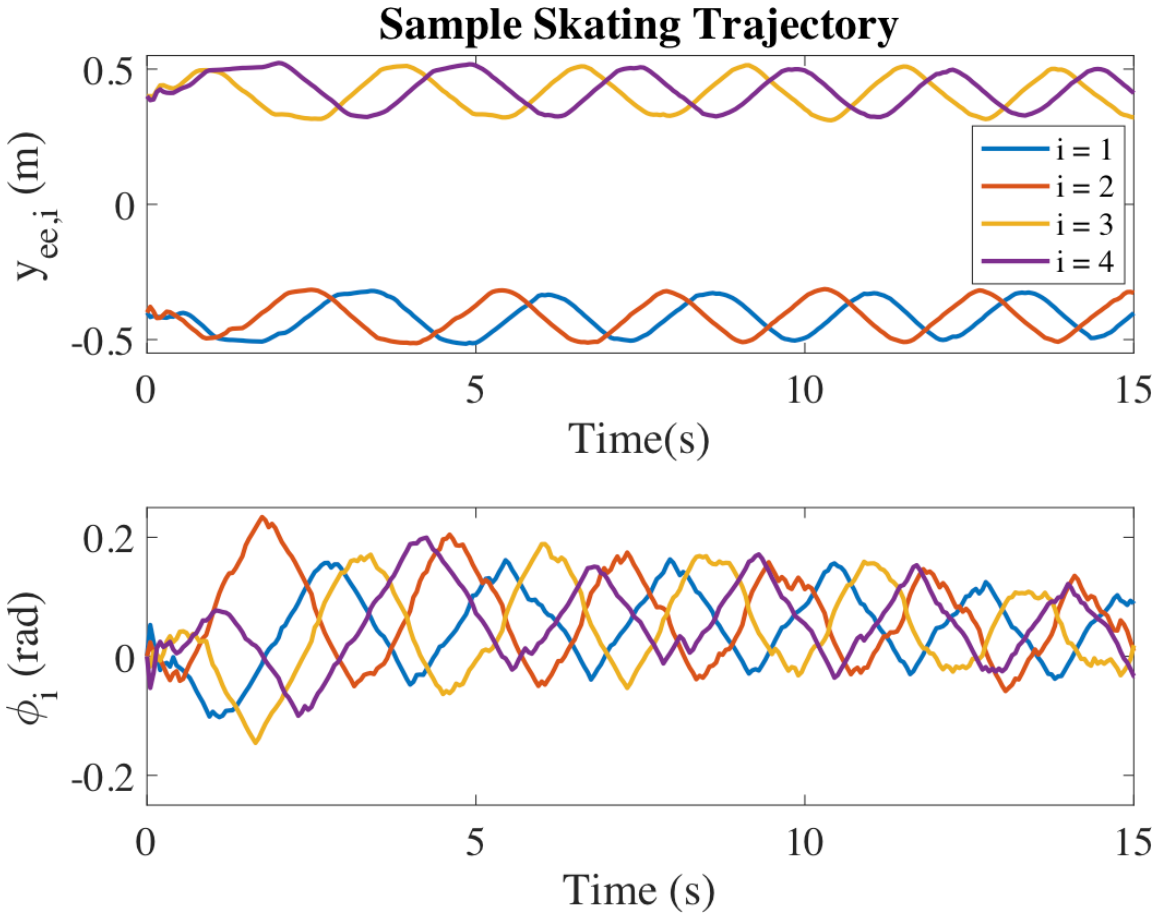




G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019

Hand-designed Trajectory vs. Trained Policy

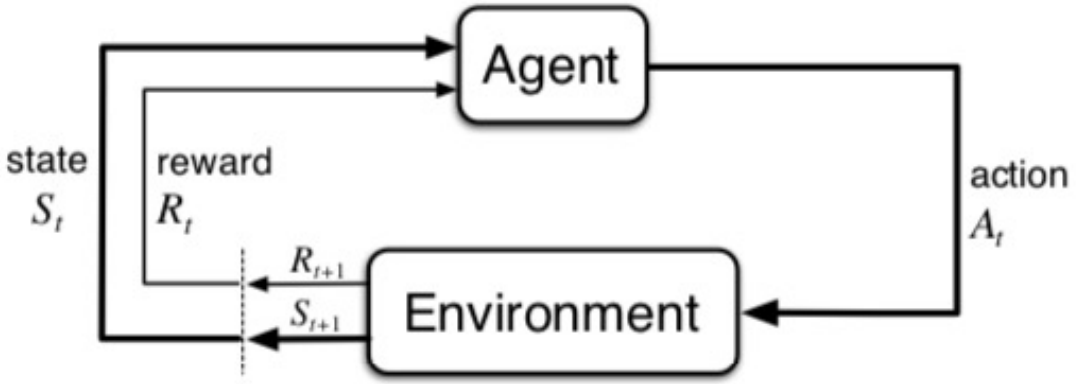


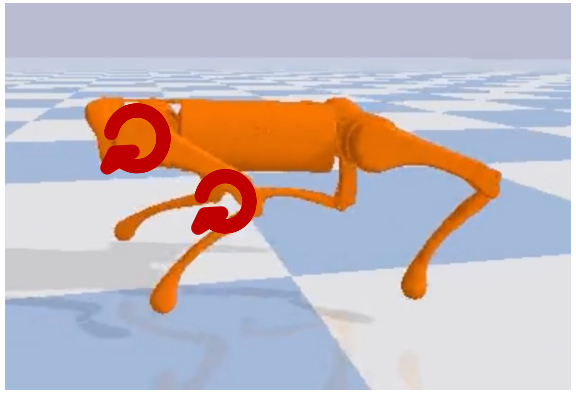


G. Bellegarda and K. Byl. "Training in Task Space to Speed Up and Guide Reinforcement Learning," IROS 2019

State/Action/Reward Space: A1

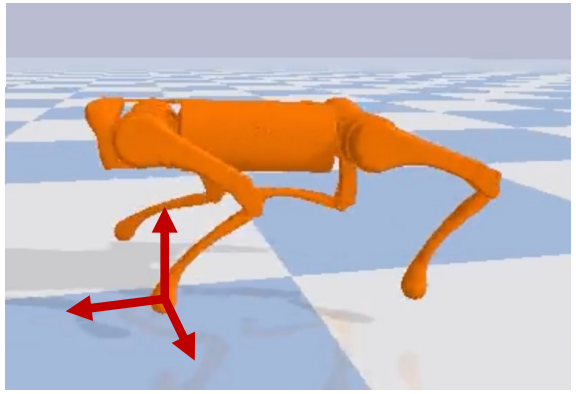
s_t ? i.e.
-body (z, r, p, y)
-body velocities
-joint states





a_t ?
-motor positions/torques
-Cartesian PD

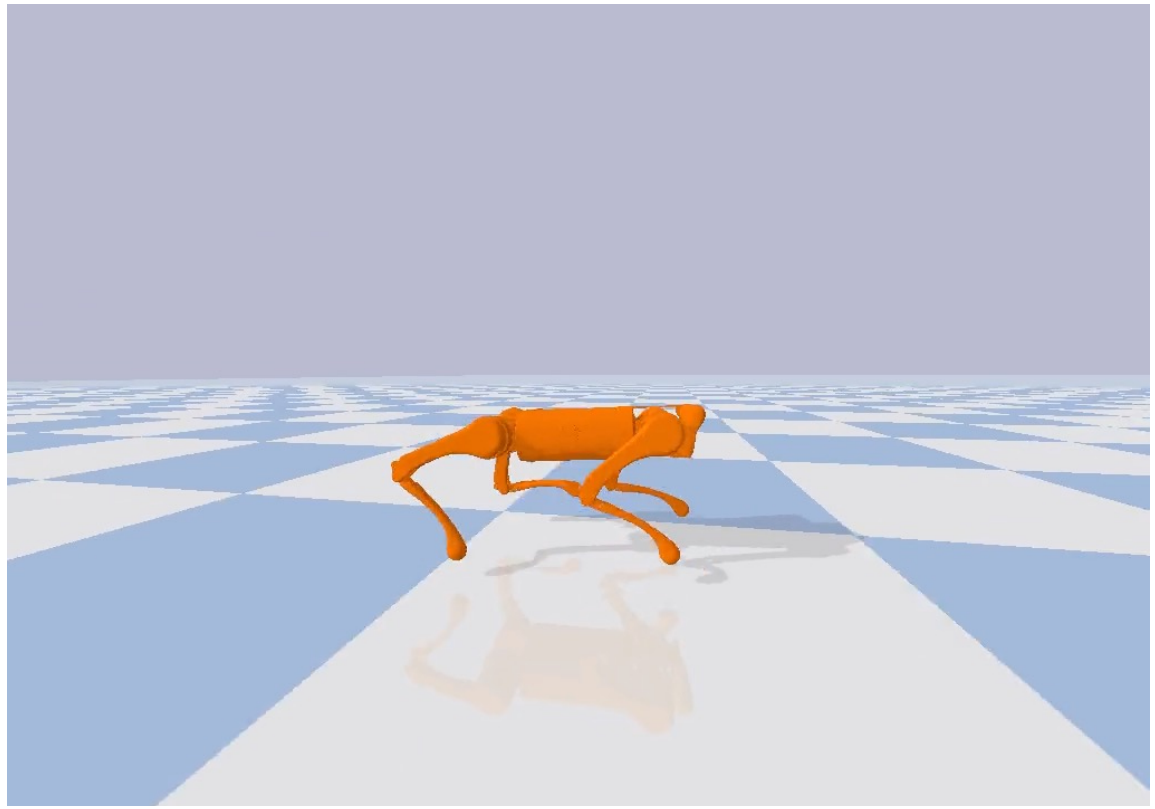
r_t ? i.e.
-body linear velocity
-energy penalty

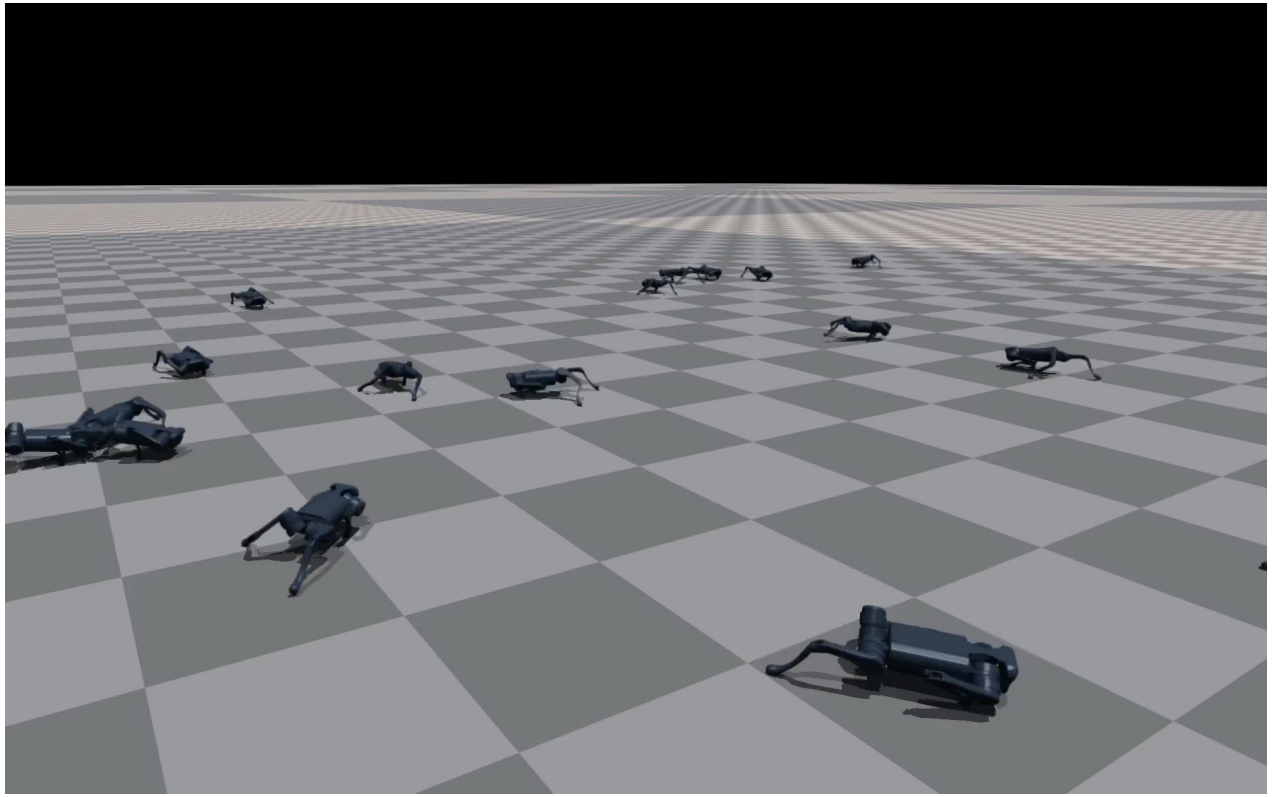


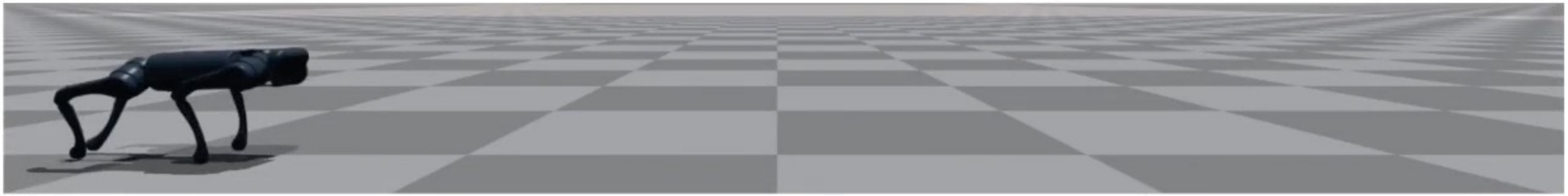
G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning," IROS 2022

Learning in Joint Space (PPO)

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

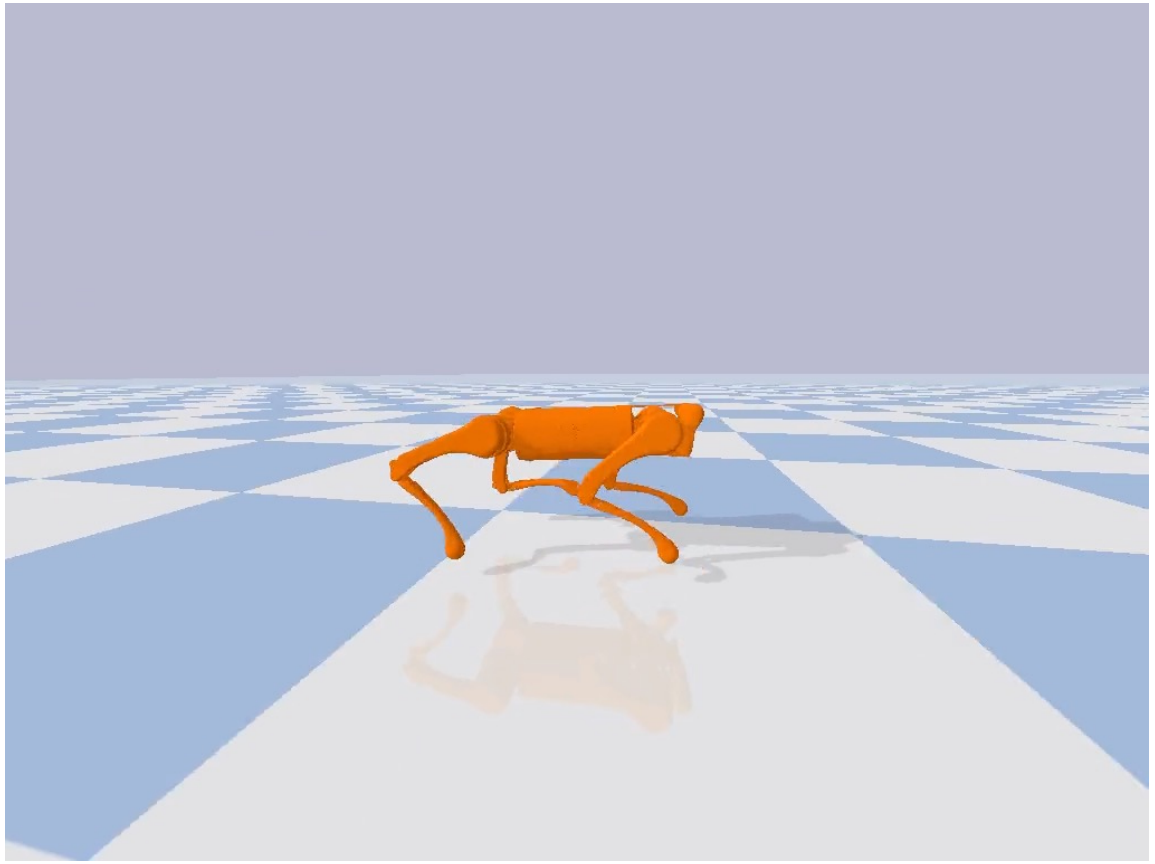




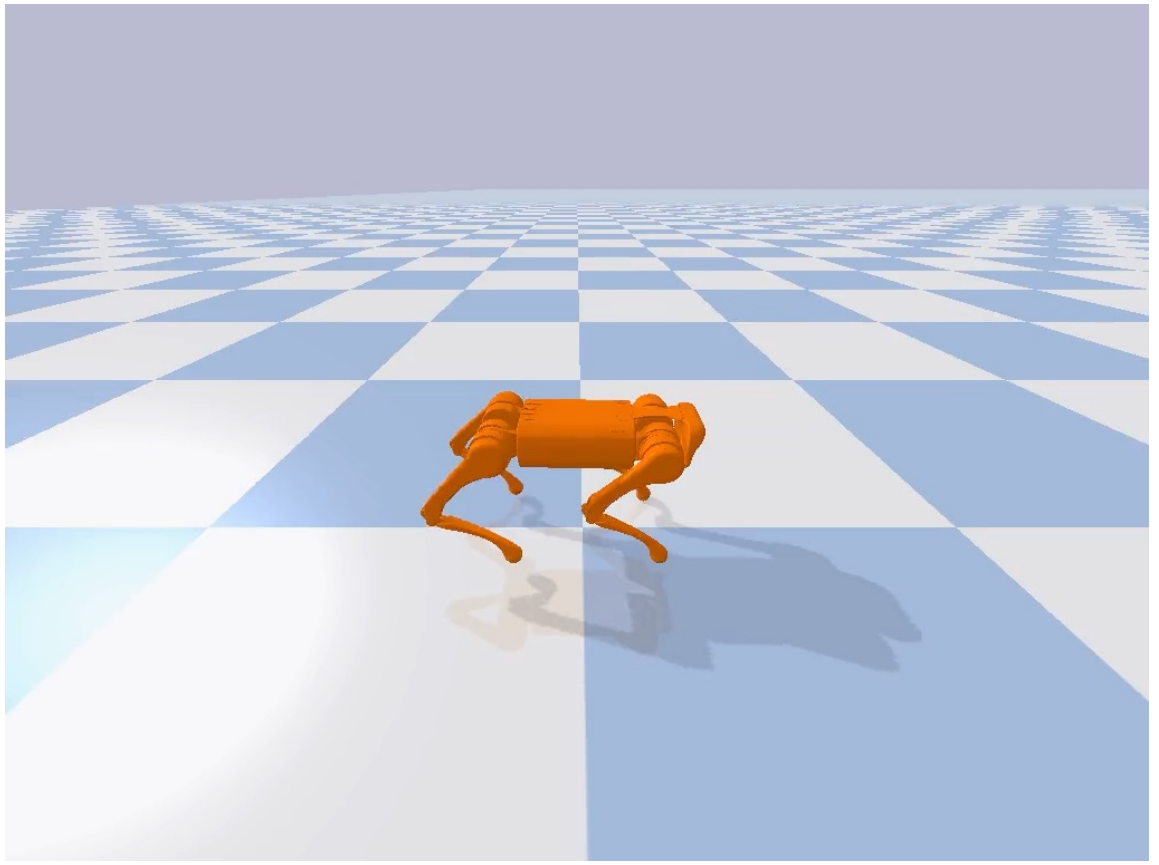


Position Control vs. Cartesian PD Control (PPO)

Action Space: $a_t = q_{1...N}$

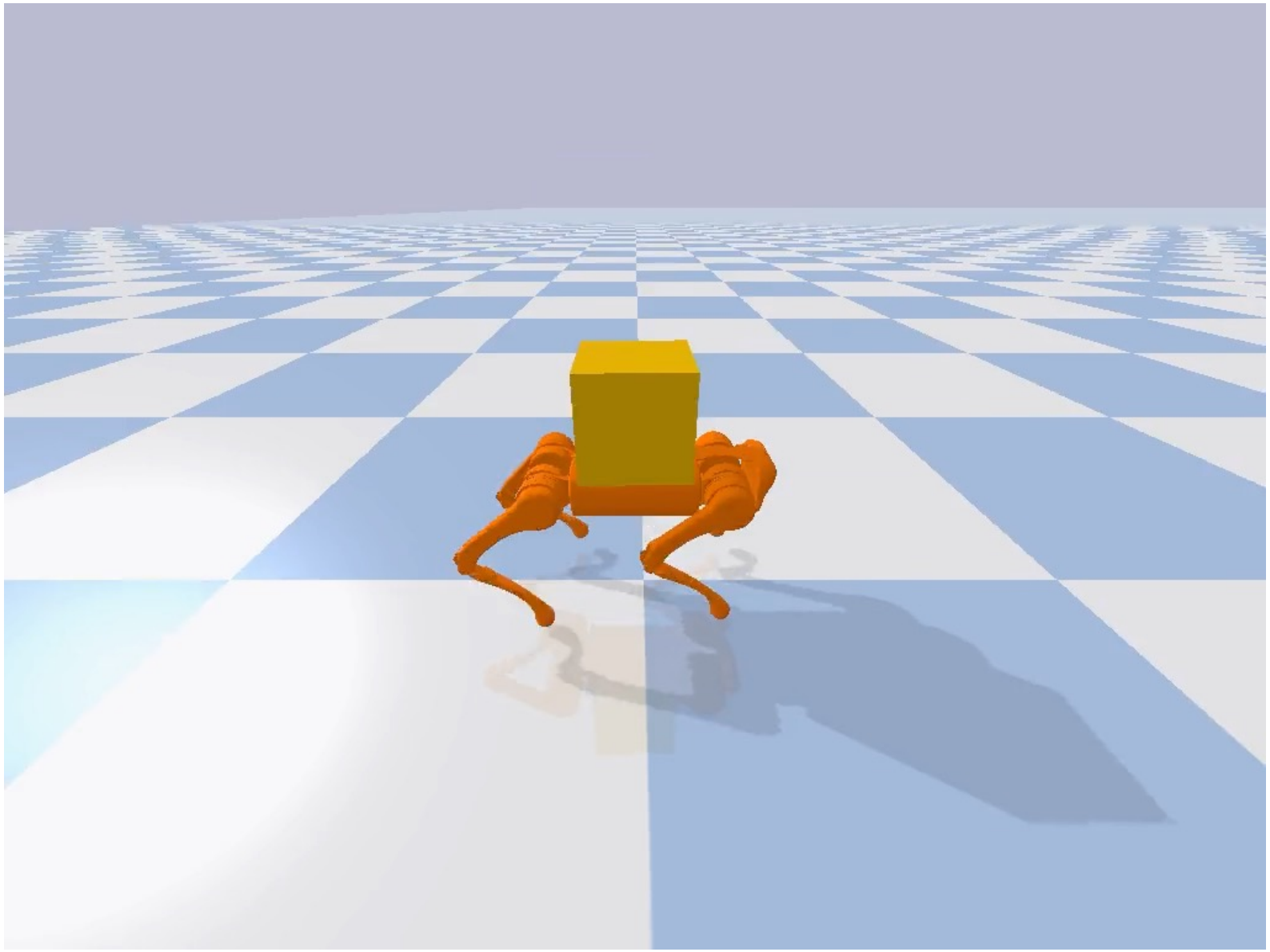


Action Space: $a_t = [x_{ee_i}, y_{ee_i}, z_{ee_i}]$



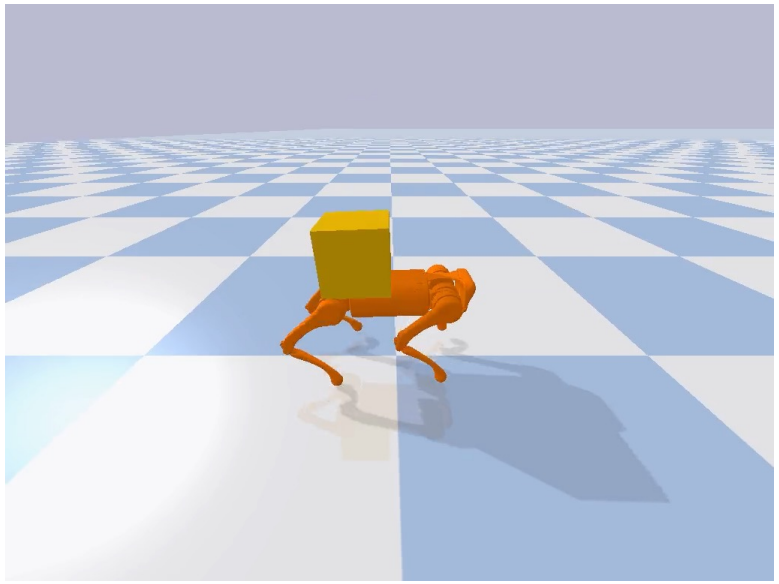
G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning," IROS 2022

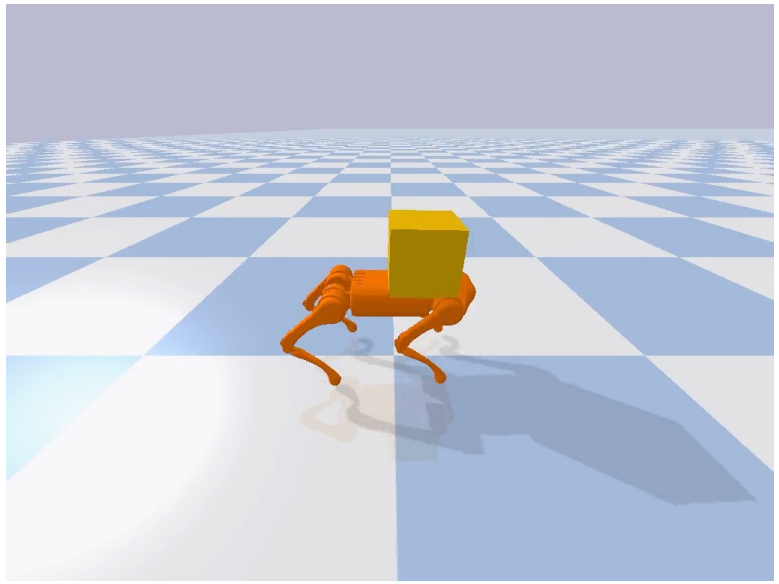
Robust to disturbances (10kg load, and 20% body mass/inertia variability)



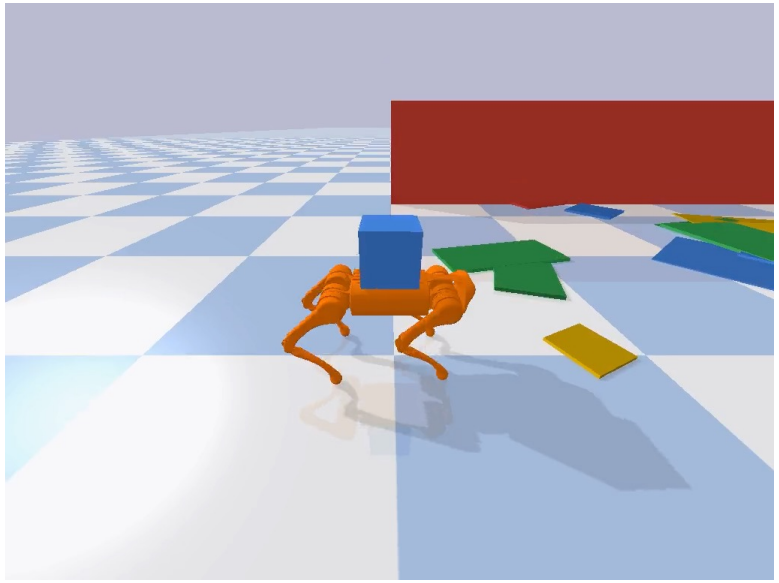
G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning," IROS 2022

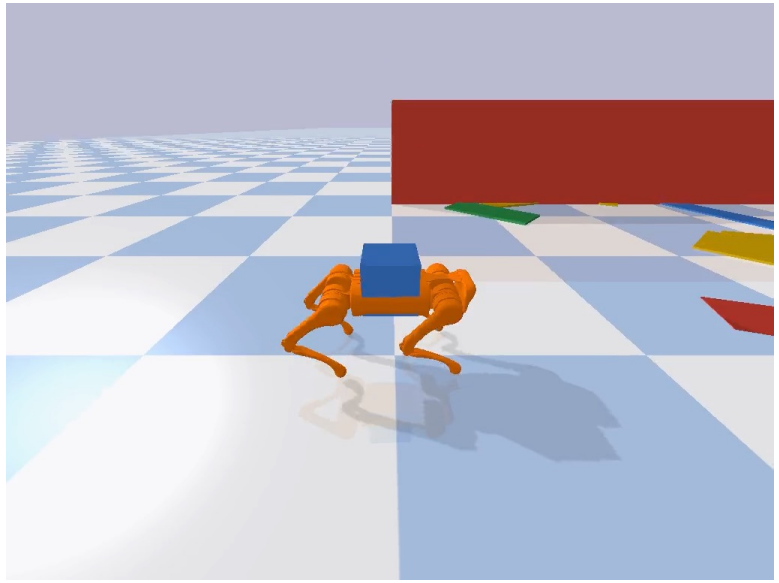
Robust to disturbances (10kg load, and 20% body mass/inertia variability)





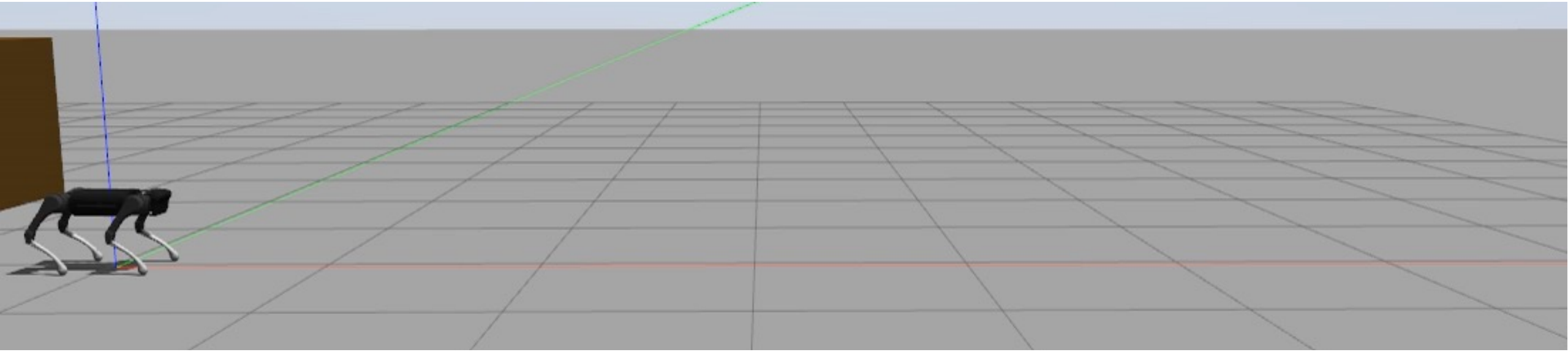
Robust to disturbances (6kg load, rough terrain up to 0.04m in height)



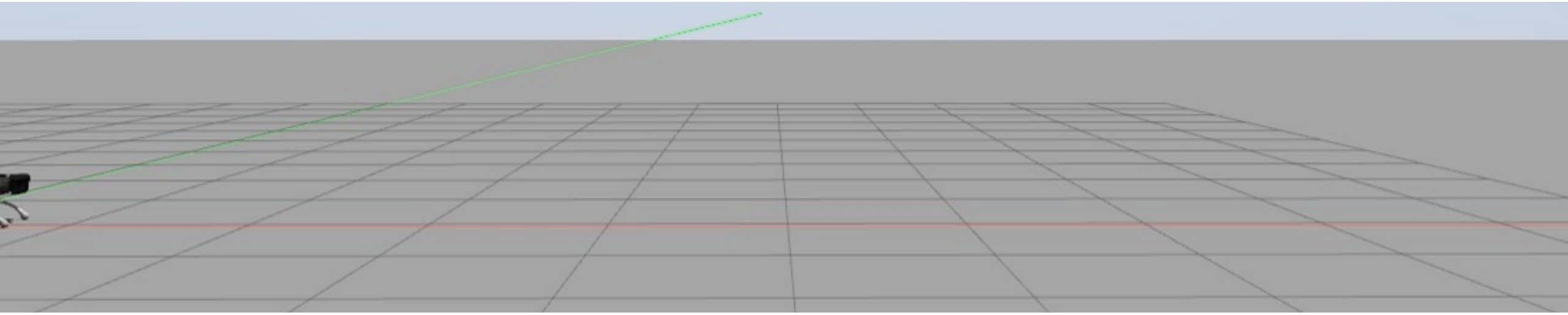


G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning," IROS 2022

Running at 4 m/s



Running at 3.5 m/s with a 10kg load



G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. “Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning,” IROS 2022

Sim-to-Real Additions

- Reward function terms:

Name	Formula	Weight
Velocity reward	$1 - \lvert v_{base} - v_{des} \rvert$	0.1
Feet swing reward	$\sum_{i=0}^{3}(t_i^k - t_i^{k-1} - 0.5)$	0.2
Energy penalty	$\int_t^{t+1} \lvert \boldsymbol{\tau} \cdot \dot{\boldsymbol{q}} \rvert dt$	-0.008
Orientation penalty	$\lVert \boldsymbol{\omega} - (0,0,0,1) \rVert$	-0.1
Lateral drift penalty	$\lvert y \rvert$	-0.1
Height penalty	$\lvert z - 0.3 \rvert$	-0.1

- Dynamics randomization:
  - Terrain randomization, observation noise

Parameter	Lower Bound	Upper Bound
Mass (each body link)	80%	120%
Added mass	0 kg	5 kg
Added mass base offset	[-0.15,-0.05,-0.05] m	[0.15,0.05,0.05] m
Coefficient of friction	0.5	1

G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning," IROS 2022

Experimental Results

G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. “Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning,” IROS 2022

Running with a 5kg Load

G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. “Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning,” IROS 2022

Running Over Different Terrains

G. Bellegarda, Y. Chen, Z. Liu, Q. Nguyen. "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning," IROS 2022

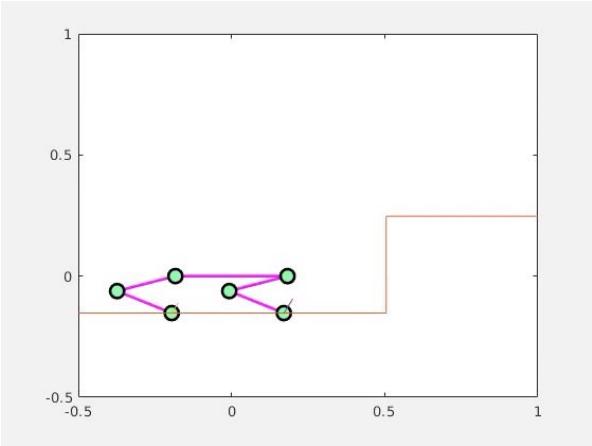
Outline

I. Motivation and Goals
II. Model-Predictive Control vs. Reinforcement Learning (same?)
III. Model-based methods
    I. Background on RoboSimian and designing skating motions
    II. Trajectory optimization allowing wheel slip
IV. Learning-based methods
    I. Action space in reinforcement learning [RoboSimian, A1]
    II. Augmenting motion planning with deep reinforcement learning [A1]
V. Bio-inspired learning
VI. Conclusion

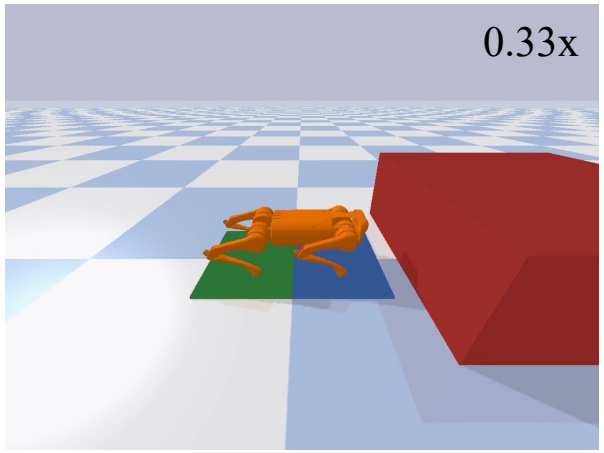
How can we combine trajectory optimization and deep learning?

- Use learning to improve trajectory optimization

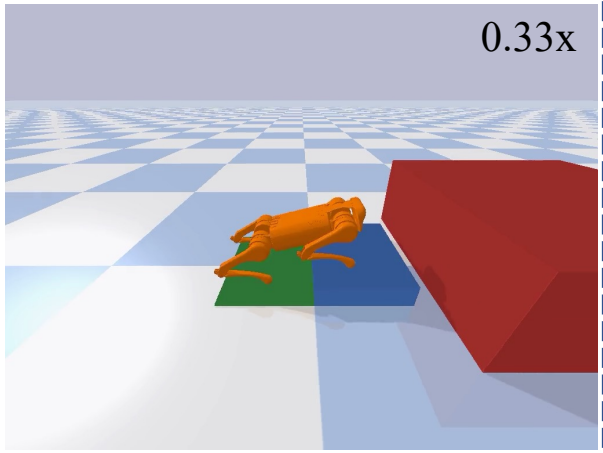
Feedforward trajectory optimization



Nonlinear trajectory optimization generates jumping motions

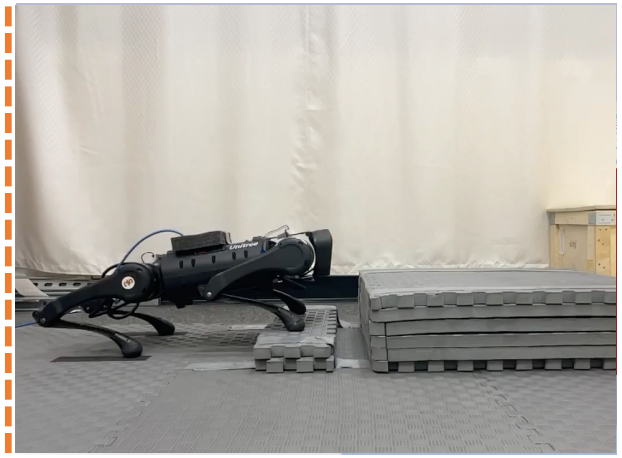


Feedforward execution



Feedforward execution **with disturbances** – doesn't work!

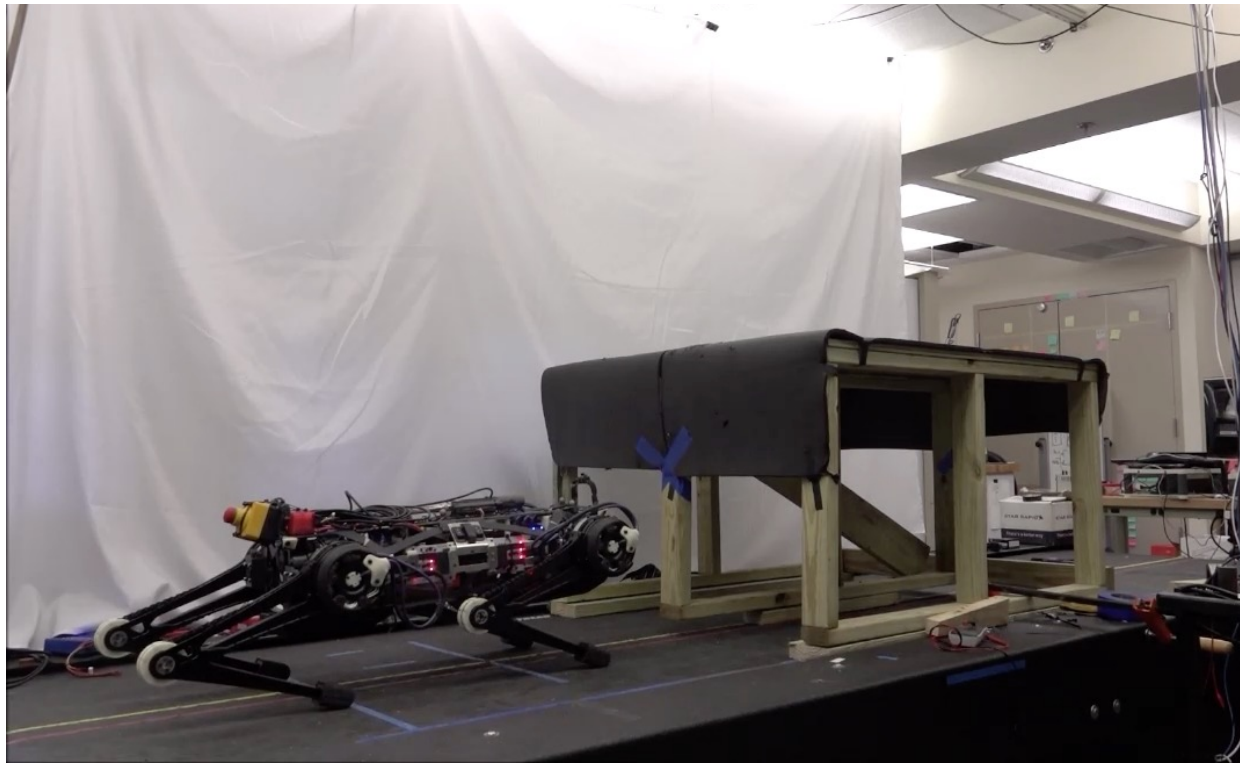
TO + Learning

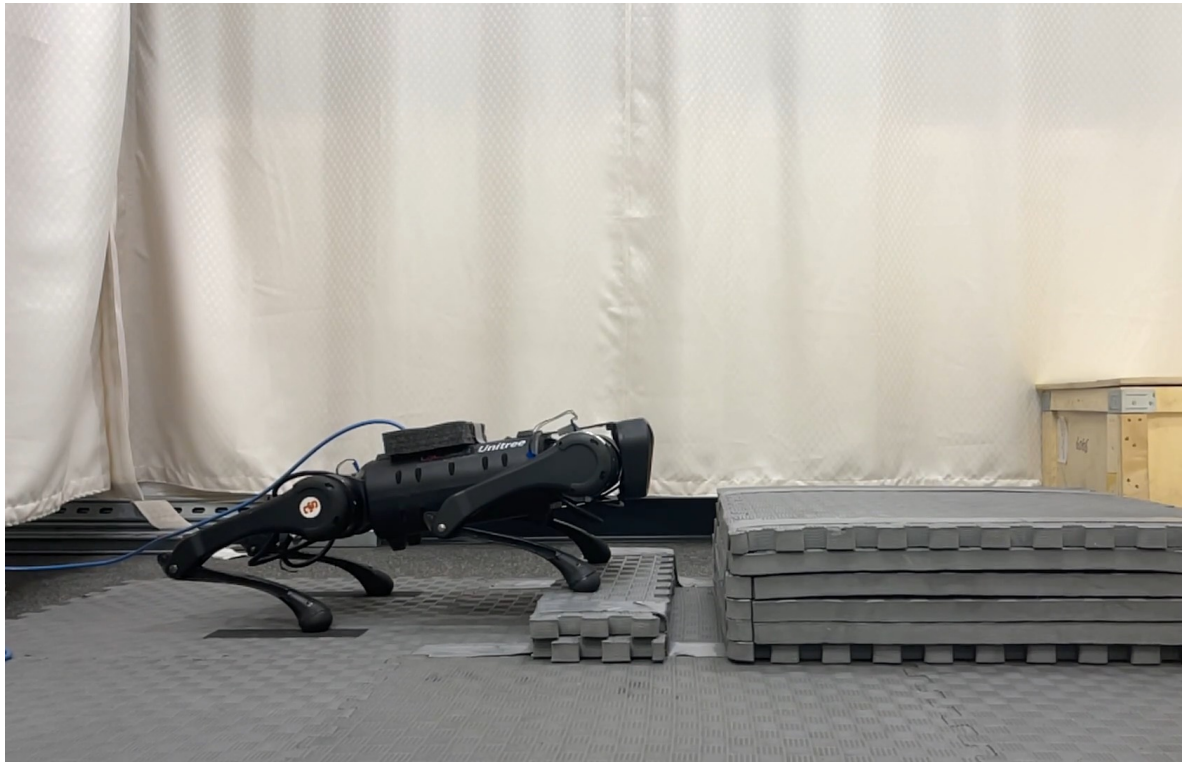


Our method:
Trajectory Optimization + DRL

G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

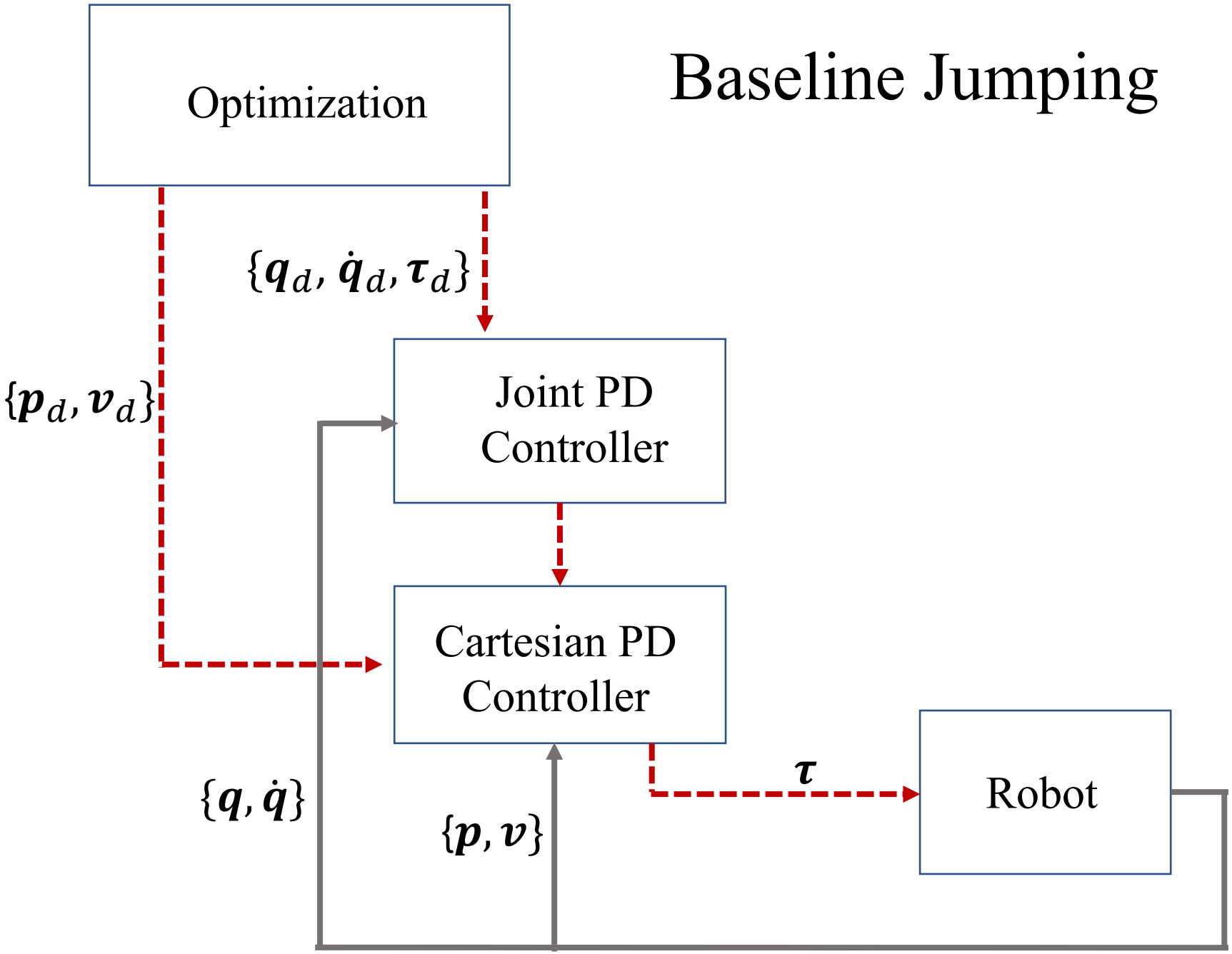
Jumping Task (Robust?)





Q. Nguyen, M. Powell, B. Katz, J. D. Carlo, S. Kim. *Optimized Jumping on the MIT Cheetah 3 Robot.* ICRA 2019
G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

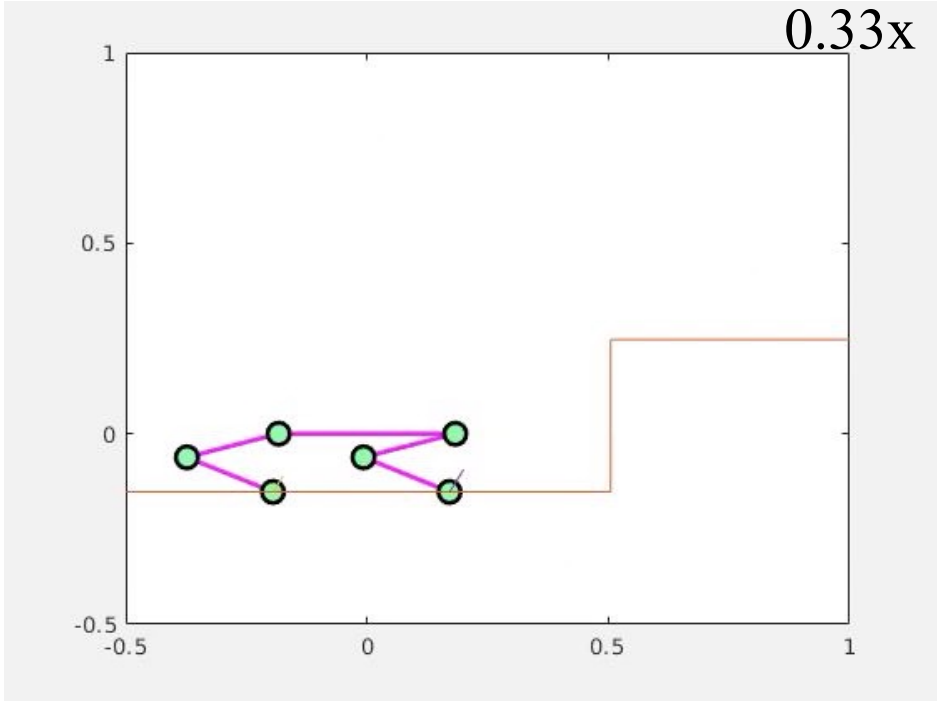
Baseline Jumping



G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

Optimization

Nonlinear trajectory optimization generates jumping motions



$\{\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \boldsymbol{\tau}_d\}$

$\{\boldsymbol{p}_d, \boldsymbol{v}_d\}$

Joint PD Controller

Cartesian PD Controller

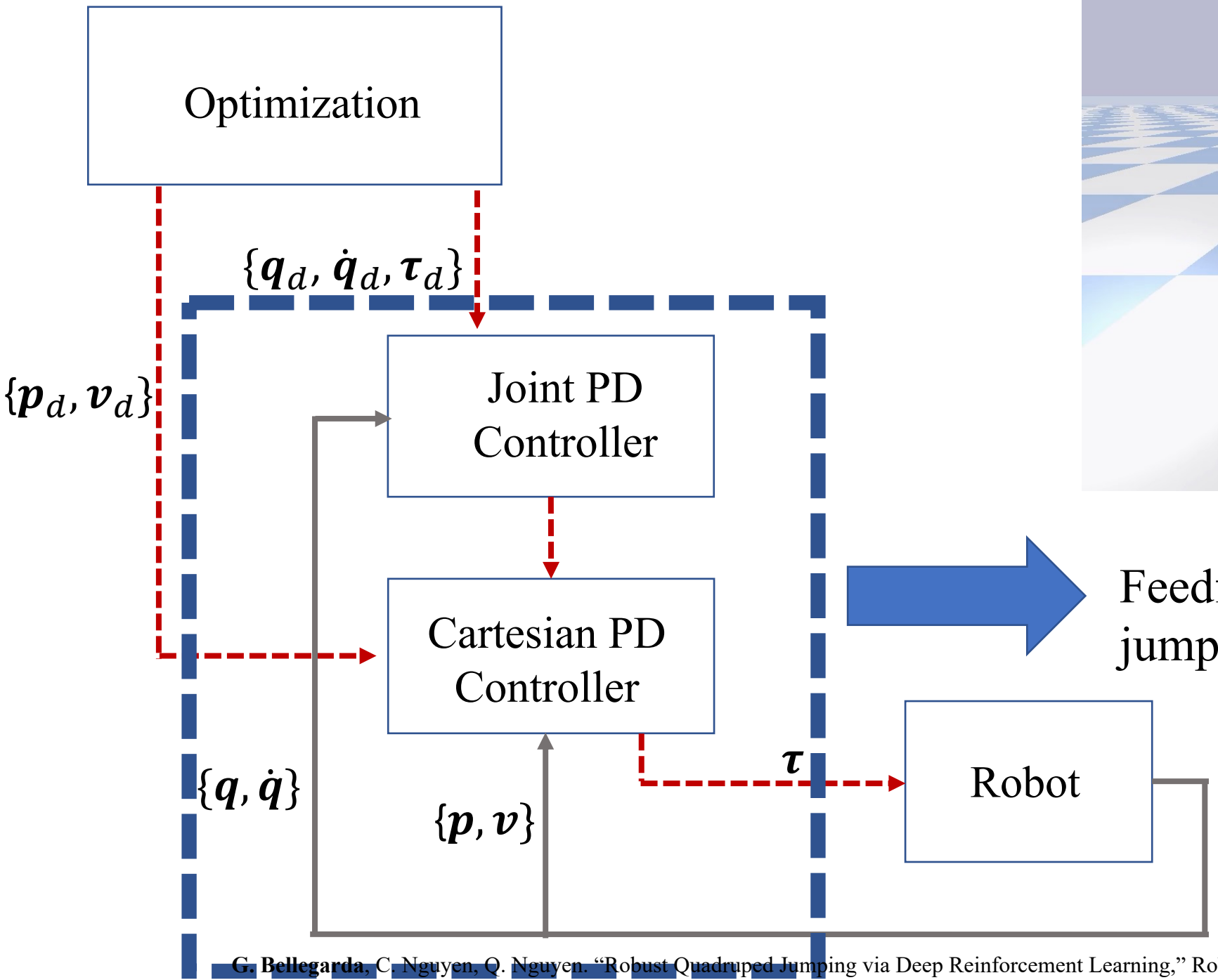
$\boldsymbol{\tau}$

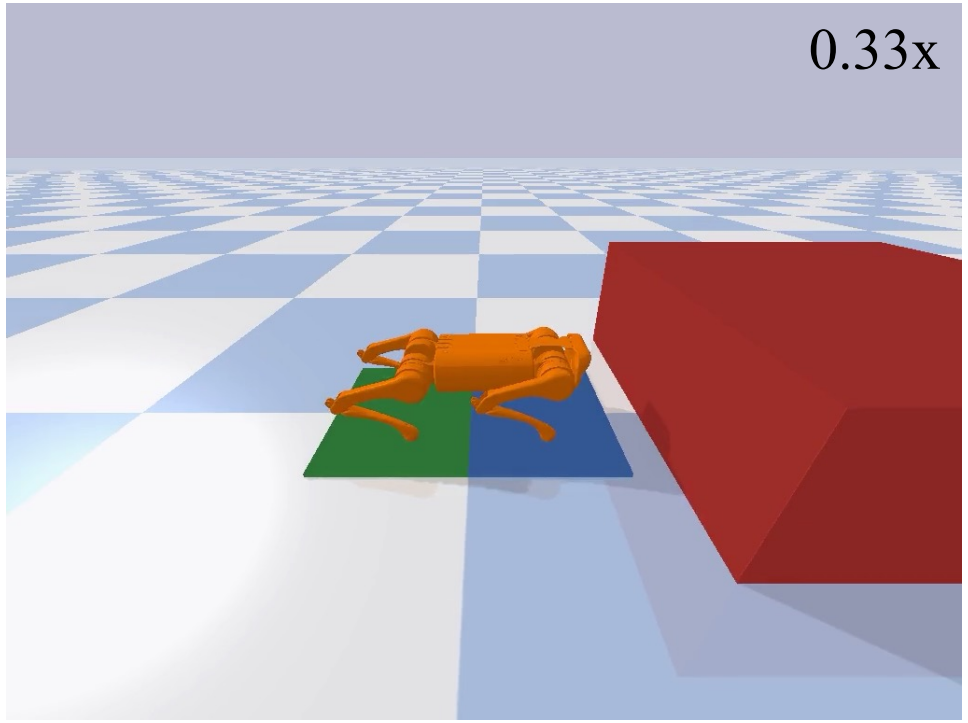
Robot

$\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$

$\{\boldsymbol{p}, \boldsymbol{v}\}$

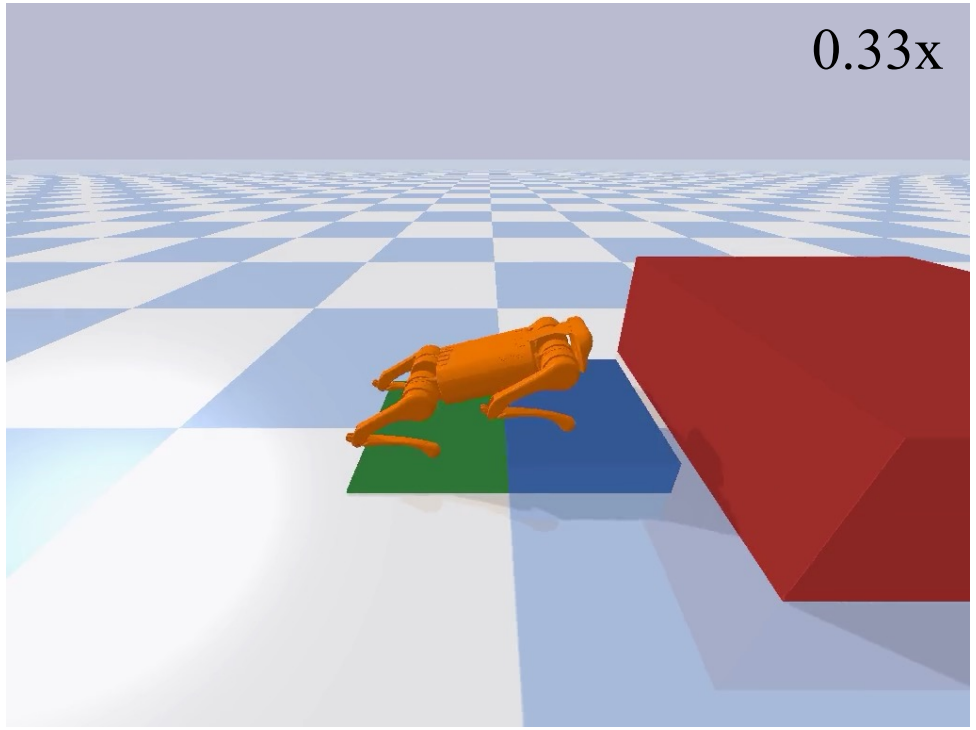
G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.





Feedforward controller tracks jumping motions…

G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.



Optimization

$\{\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \boldsymbol{\tau}_d\}$

$\{\boldsymbol{p}_d, \boldsymbol{v}_d\}$

Joint PD Controller

Cartesian PD Controller

$\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$

$\{\boldsymbol{p}, \boldsymbol{v}\}$

$\boldsymbol{\tau}$

Robot

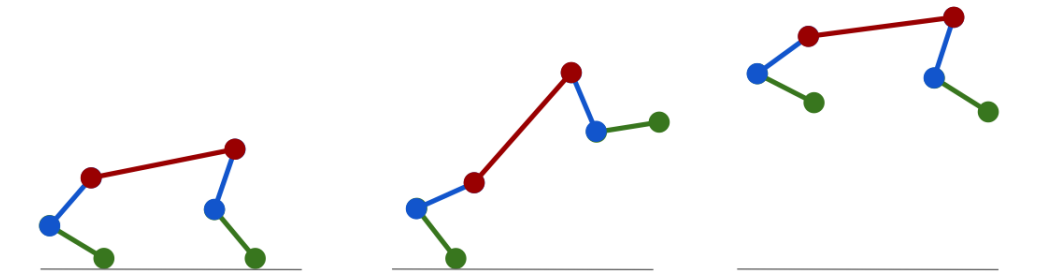
Feedforward controller tracks jumping motions…

…but only works under ideal conditions!

G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

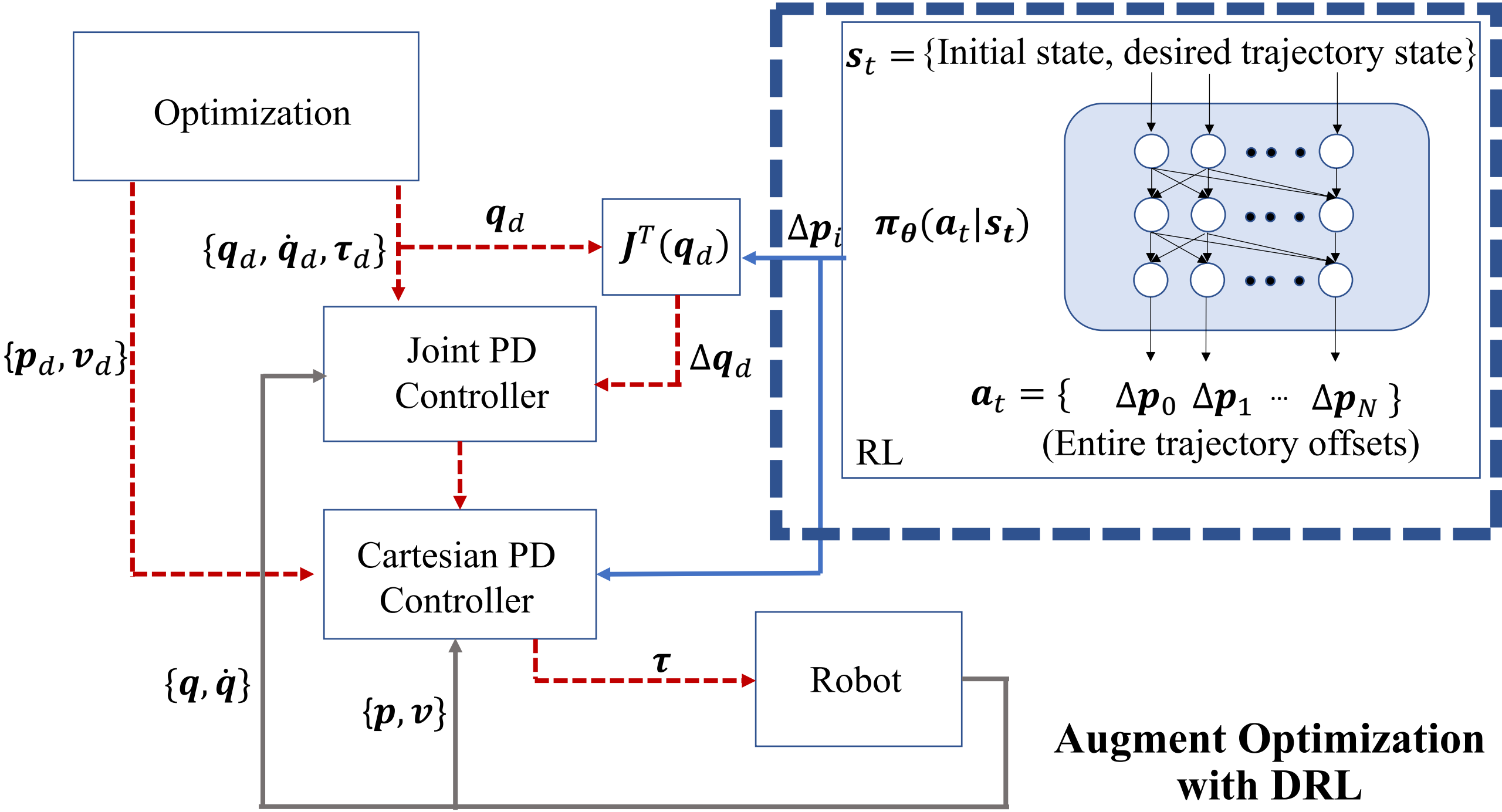
How to improve jumping with learning?

- Jumping involves very specific joint trajectories

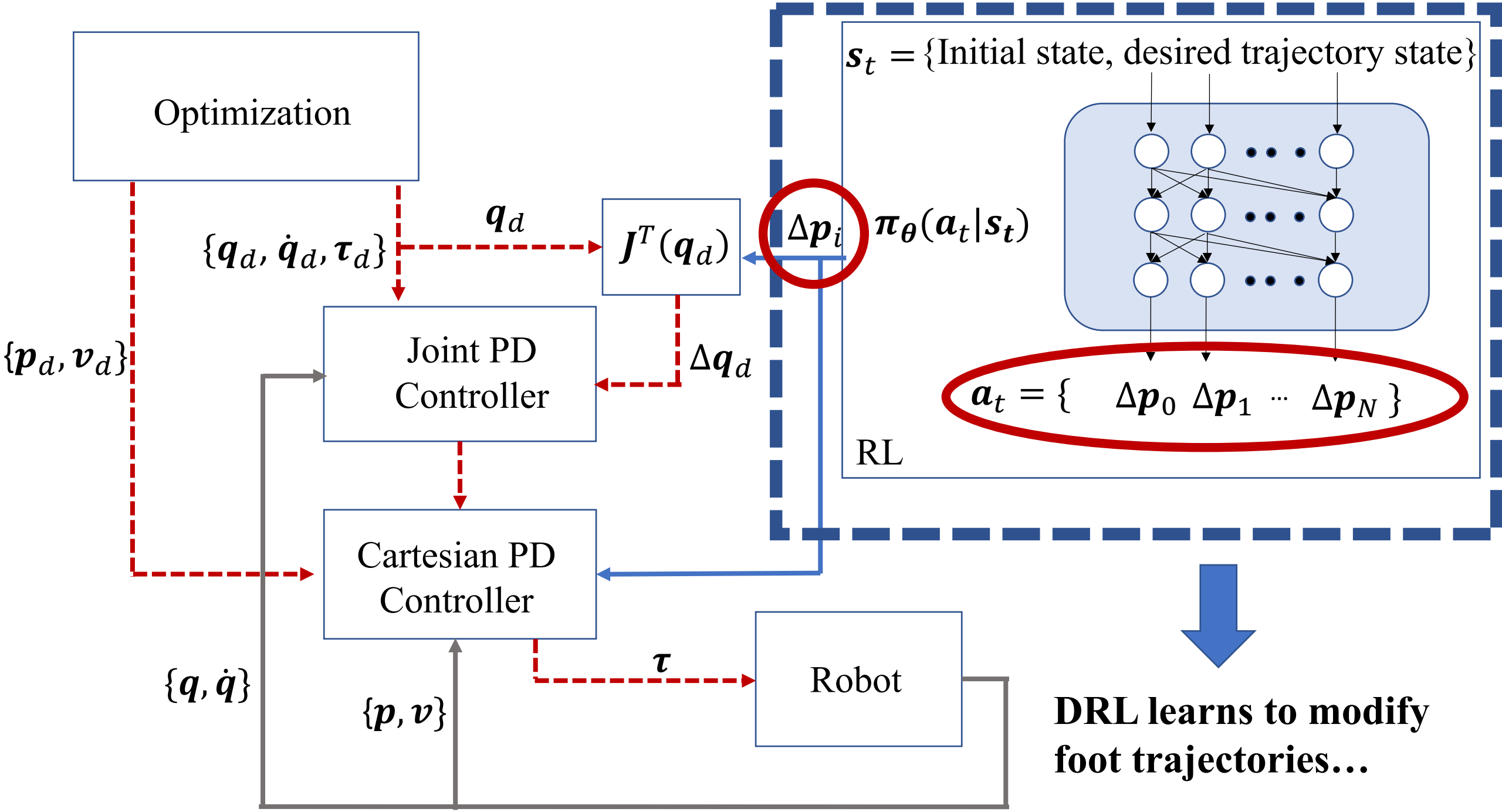


- Difficult to learn from scratch..
  - Imitate trajectory?
  - Reward function tuning..
- Already have an approximate solution in ideal case
  - Learn to modify existing trajectories for robustness (residual learning)

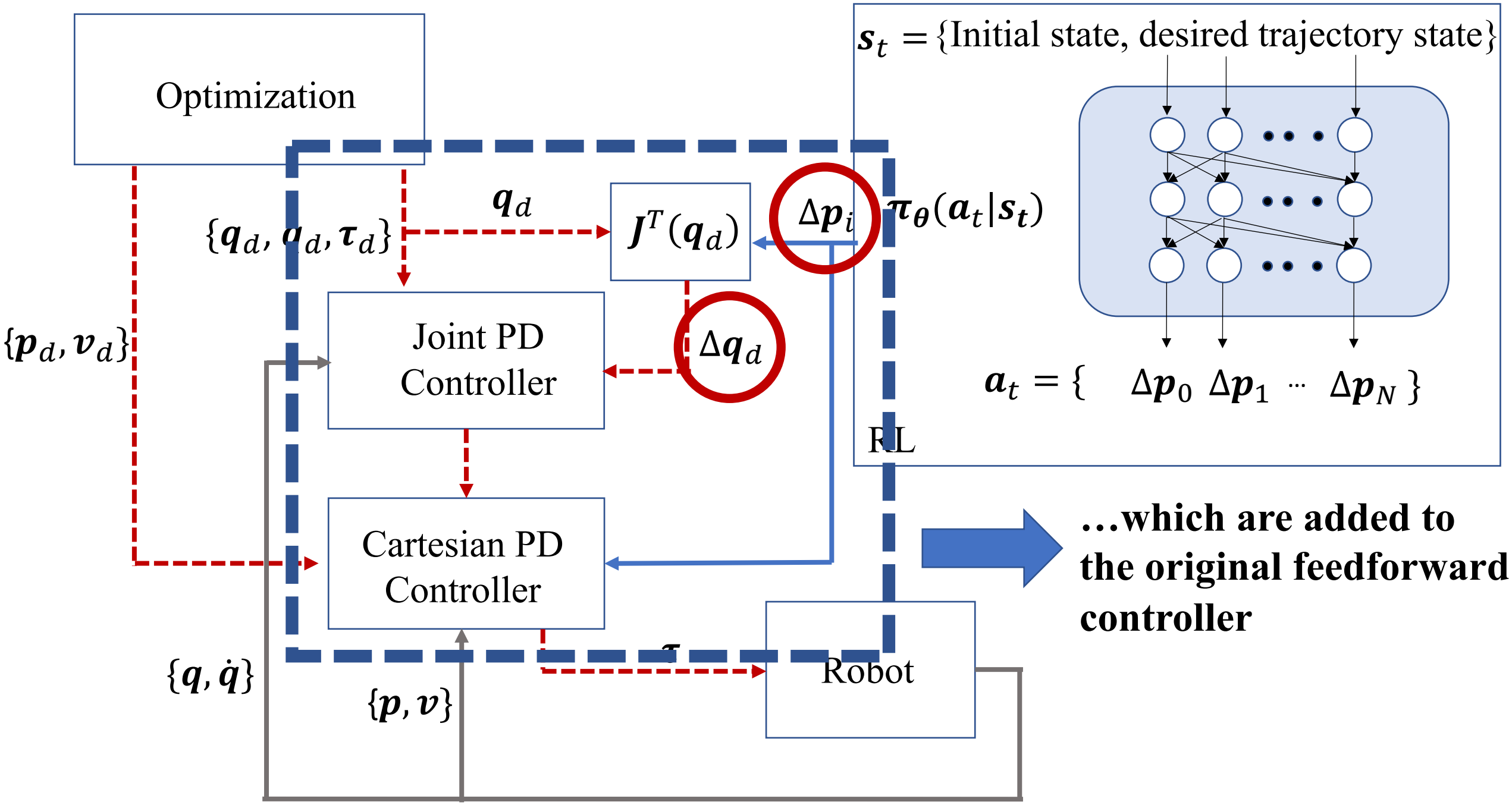
G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.



G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.



G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

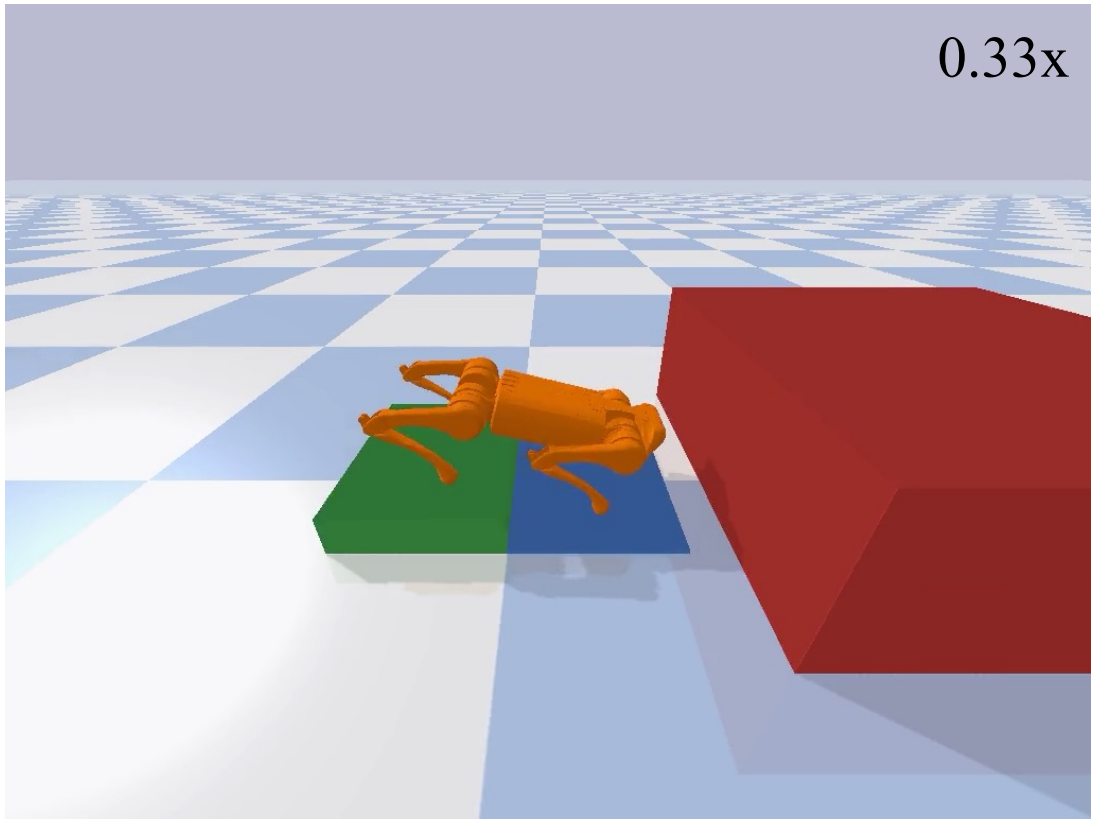


G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

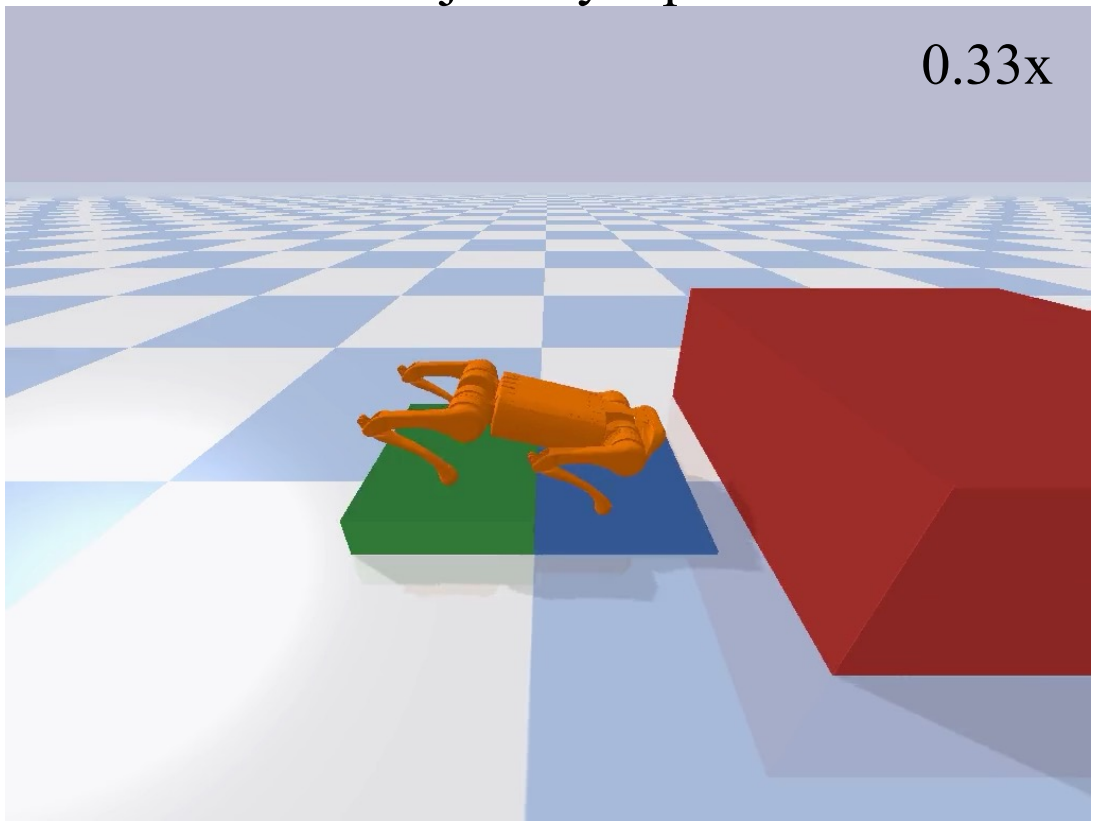
Example Simulation Result

- Jumping height and distance: **(0.4, 0.7) m**
- Mass and inertia uncertainty: **5%**
- Ground uncertainty: **0.1 m** block under rear feet

Baseline - Pure Trajectory Optimization



Our method – Trajectory Optimization + DRL

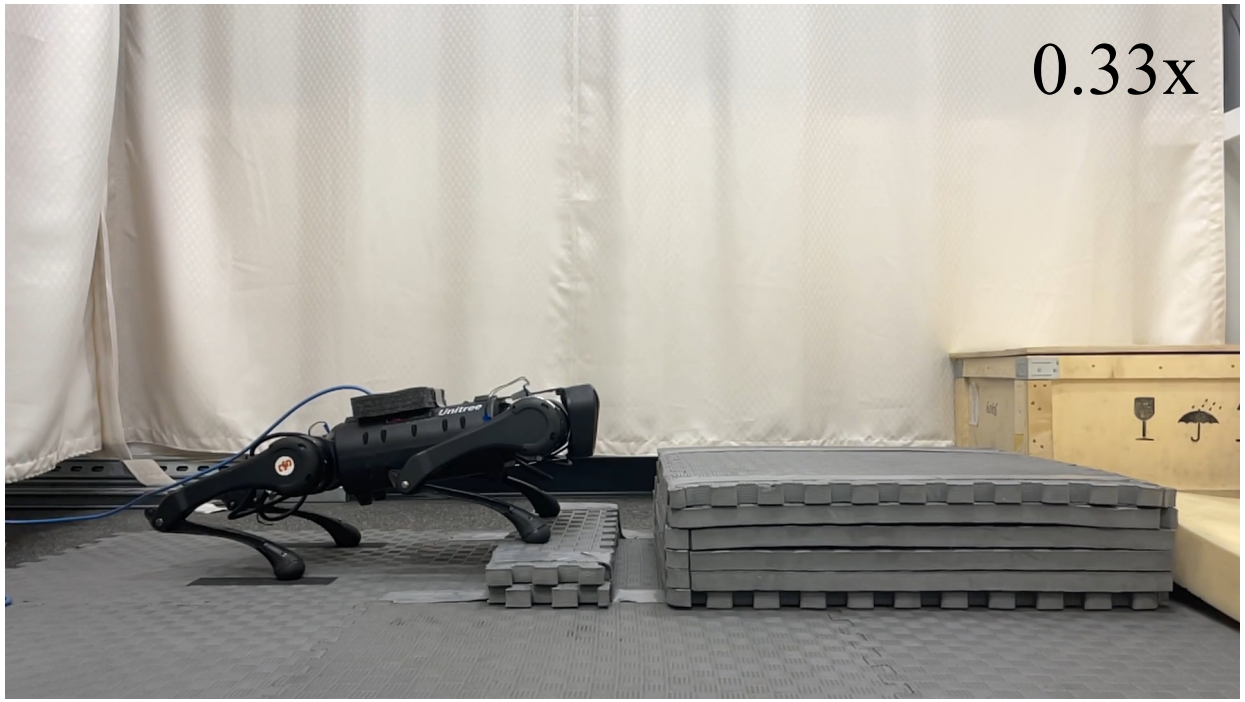


G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

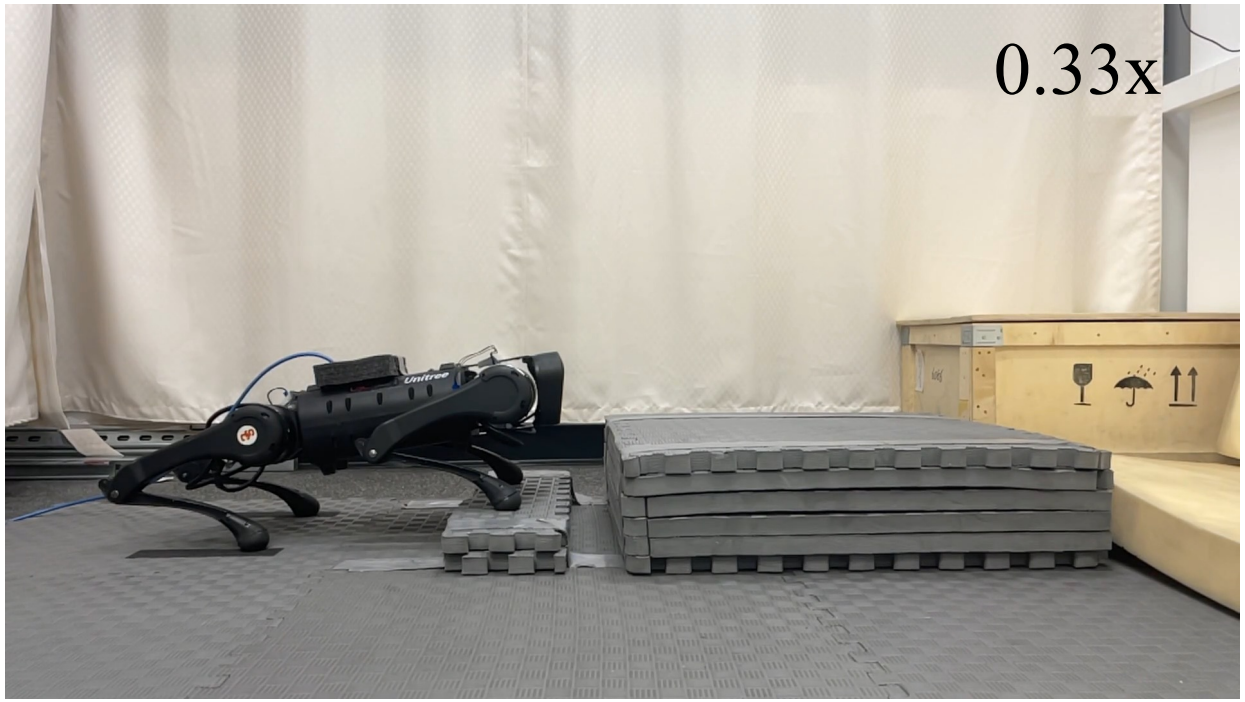
Example Hardware Result

- Jumping height and distance: **(0.2, 0.6) m**
- Ground uncertainty: **0.06 m** block under front feet

Baseline - Pure Trajectory Optimization



Our method – Trajectory Optimization + DRL



G. Bellegarda, C. Nguyen, Q. Nguyen. "Robust Quadruped Jumping via Deep Reinforcement Learning," Robotics and Autonomous Systems 2024.

Outline

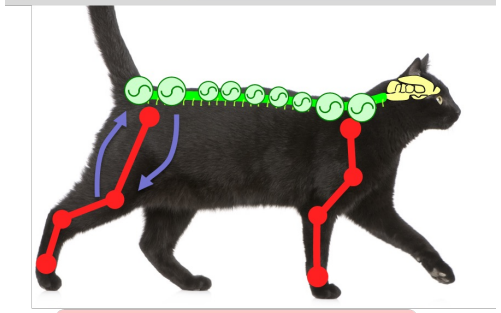
I. Motivation and Goals
II. Model-Predictive Control vs. Reinforcement Learning (same?)
III. Model-based methods
    I. Background on RoboSimian and designing skating motions
    II. Trajectory optimization allowing wheel slip
IV. Learning-based methods
    I. Action space in reinforcement learning [RoboSimian, A1]
    II. Augmenting motion planning with deep reinforcement learning [A1]
V. Bio-inspired learning
VI. Conclusion

From Lecture 4

Descending modulation

Spinal cord

Reflexes | Central pattern generators



Musculoskeletal system

100%

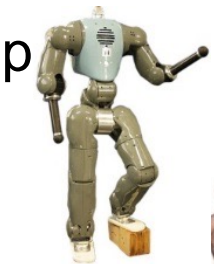
Respective Role in motor control

lamprey

salamander

“Comp ıimal sp

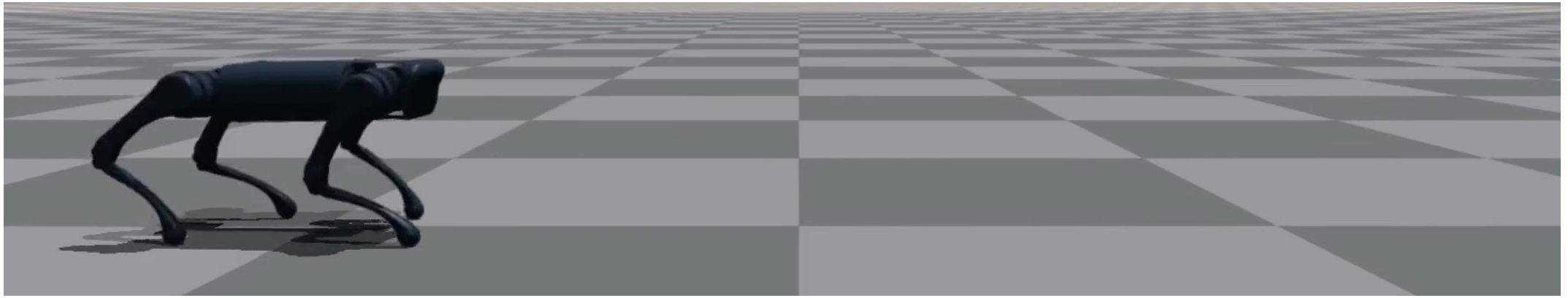
cat



human

Traditional Central Pattern Generators for Quadruped Locomotion

Trot



Amplitude:

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

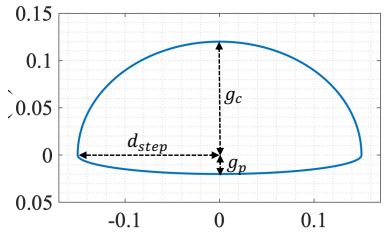
Phase:

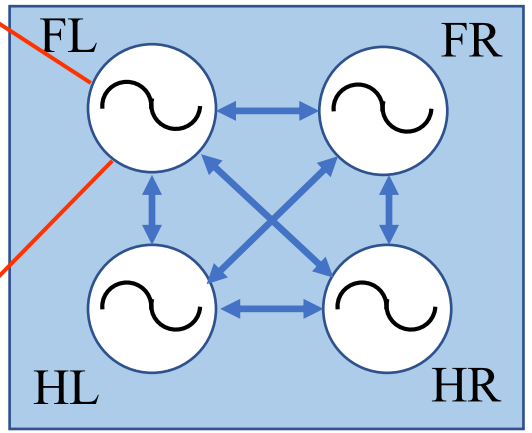
$$\dot{\theta}_i = \omega_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}), \quad \omega_i = \begin{cases} \omega_{\text{swing}} & \text{if } 0 \le \theta_i < \pi \\ \omega_{\text{stance}} & \text{if } \pi \le \theta_i < 2\pi \end{cases}$$

Output:

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

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

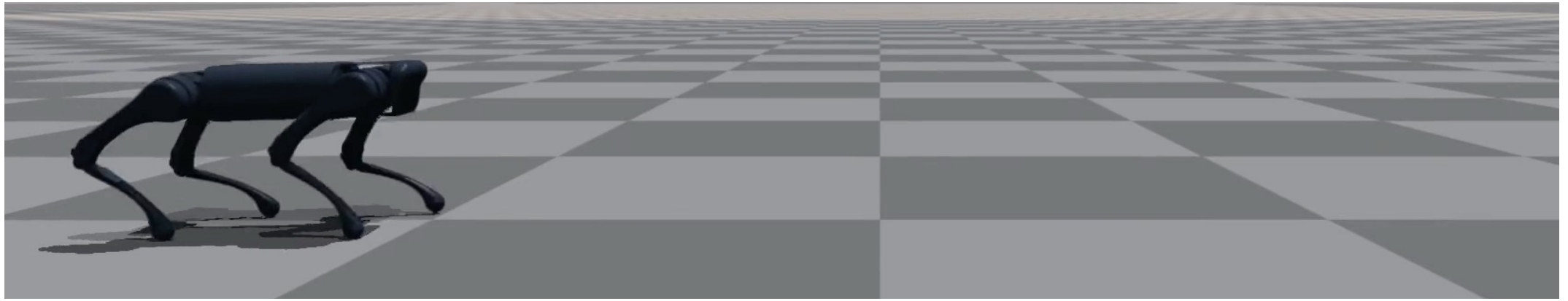




A. Ijspeert, A. Crespi, D. Ryczko, J.M. Cabelguen. *From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model.* Science 2007

Traditional Central Pattern Generators for Quadruped Locomotion

Pace



Amplitude:

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

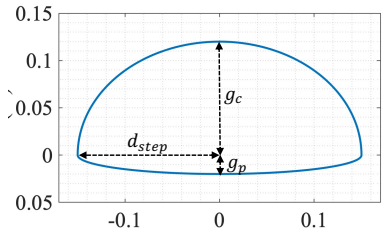
Phase:

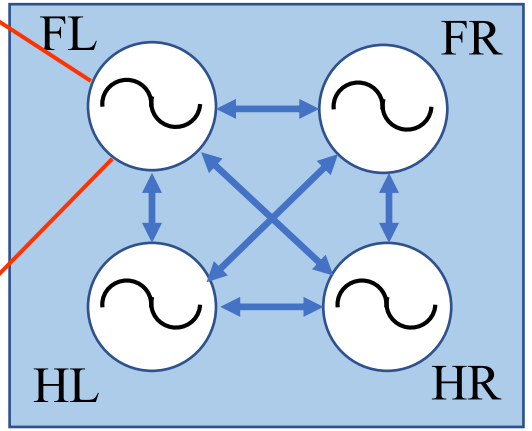
$$\dot{\theta}_i = \omega_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}), \quad \omega_i = \begin{cases} \omega_{\text{swing}} & \text{if } 0 \le \theta_i < \pi \\ \omega_{\text{stance}} & \text{if } \pi \le \theta_i < 2\pi \end{cases}$$

Output:

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

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

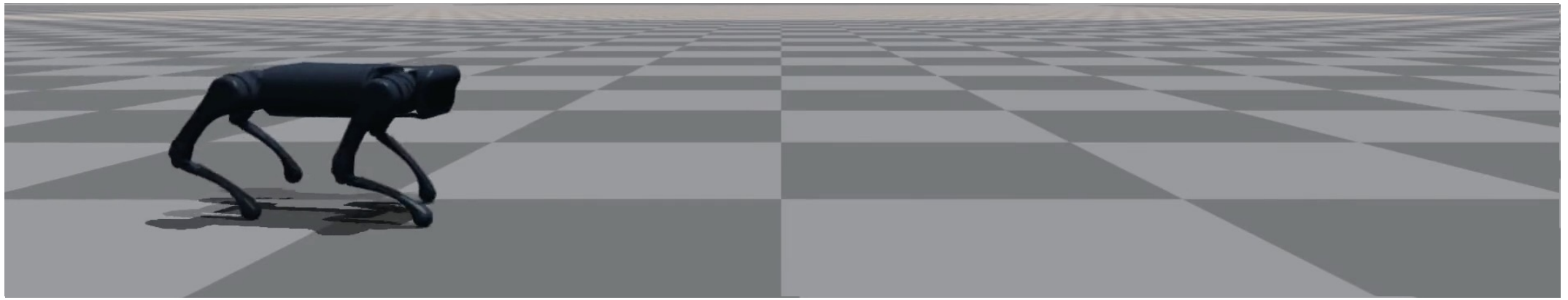




A. Ijspeert, A. Crespi, D. Ryczko, J.M. Cabelguen. *From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model.* Science 2007

Traditional Central Pattern Generators for Quadruped Locomotion

Walk



Amplitude:

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

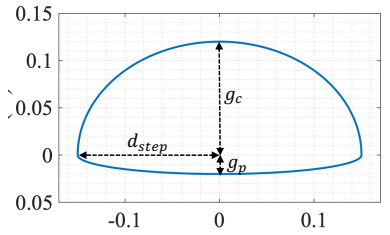
Phase:

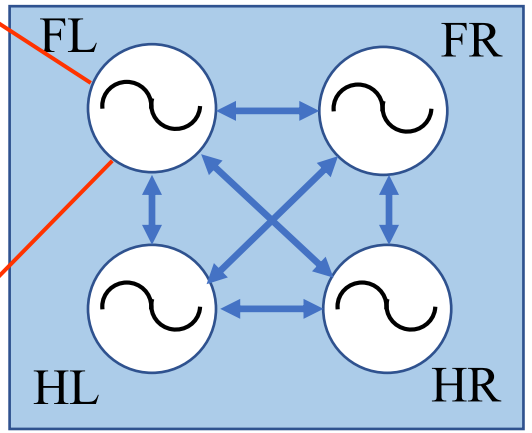
$$\dot{\theta}_i = \omega_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}), \quad \omega_i = \begin{cases} \omega_{\text{swing}} & \text{if } 0 \le \theta_i < \pi \\ \omega_{\text{stance}} & \text{if } \pi \le \theta_i < 2\pi \end{cases}$$

Output:

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

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





A. Ijspeert, A. Crespi, D. Ryczko, J.M. Cabelguen. *From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model.* Science 2007

Traditional Central Pattern Generators for Quadruped Locomotion

- Require careful parameter tuning (optimization-based or hand-tuned)
- Result in a single fixed gait (must re-tune for each desired gait and speed)
- Difficult to specify omni-directional motion (i.e. add control like VMC to turn)

Can we improve the general capabilities of CPGs with deep reinforcement learning?

Amplitude:

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

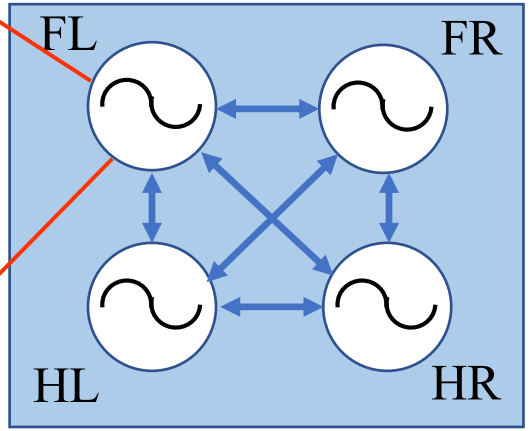
Phase:

$$\dot{\theta}_i = \omega_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}), \quad \omega_i = \begin{cases} \omega_{\text{swing}} & \text{if } 0 \le \theta_i < \pi \\ \omega_{\text{stance}} & \text{if } \pi \le \theta_i < 2\pi \end{cases}$$

Output:

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

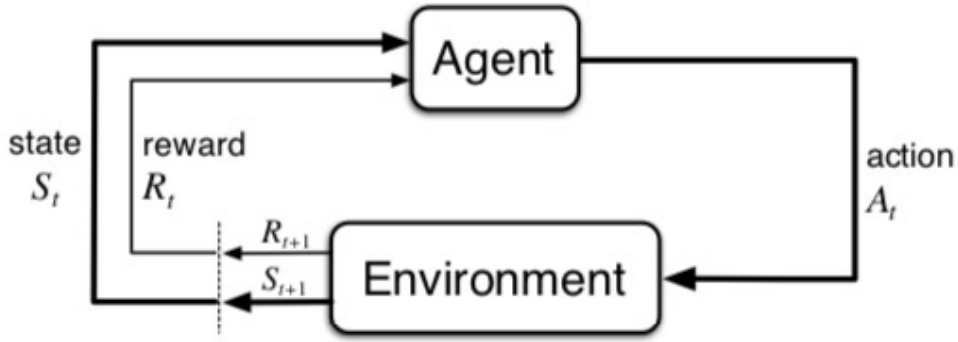
$$z_{\text{foot},i} = \begin{cases} -h + g_c \sin(\theta_i) & \text{if } \sin(\theta_i) > 0 \\ -h + g_p \sin(\theta_i) & \text{otherwise} \end{cases}$$

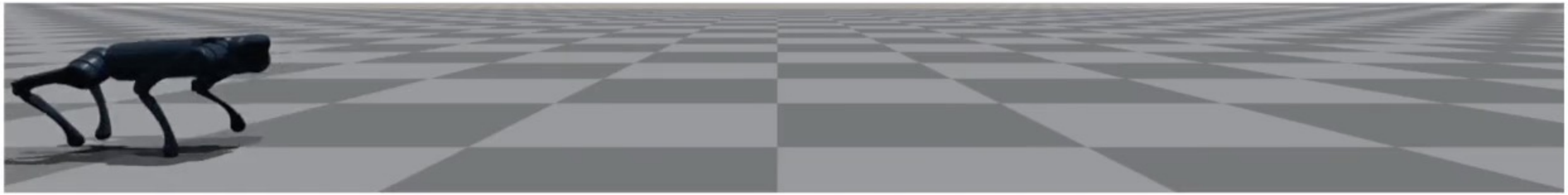


M. Ajallooeian, S. Pouya, A. Sproewitz, A. Ijspeert. *Central Pattern Generators Augmented with Virtual Model Control for Quadruped Rough Terrain Locomotion.* ICRA 2013

Training Control Policies with Deep Reinforcement Learning

- Training in joint space with task-minimalistic reward functions gives rise to unnatural gaits
  - Reward function:
    - + Track base velocity command (v_x, v_y, ω_z)
    - - Penalize other base velocities $(v_z, \omega_x, \omega_y)$
    - - Penalize energy
  - Add terms for:
    - + Track base height h
    - + Reward foot air time

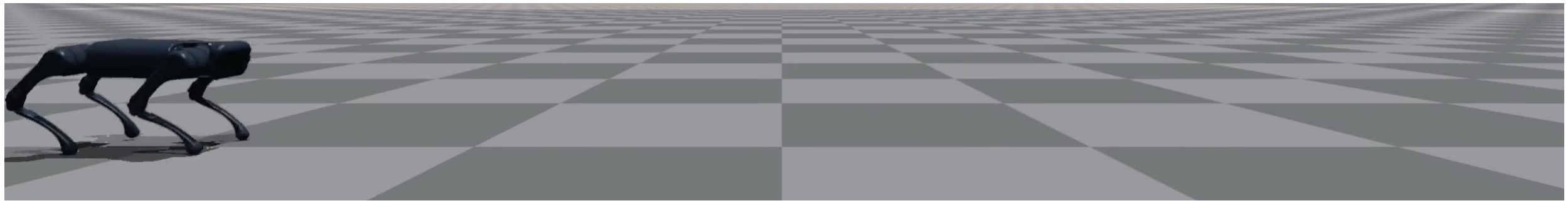




Training Control Policies with Deep Reinforcement Learning

- Training in joint space with task-minimalistic reward functions gives rise to unnatural gaits
- With extensive reward shaping, more natural behaviors can emerge
  - Reward function:

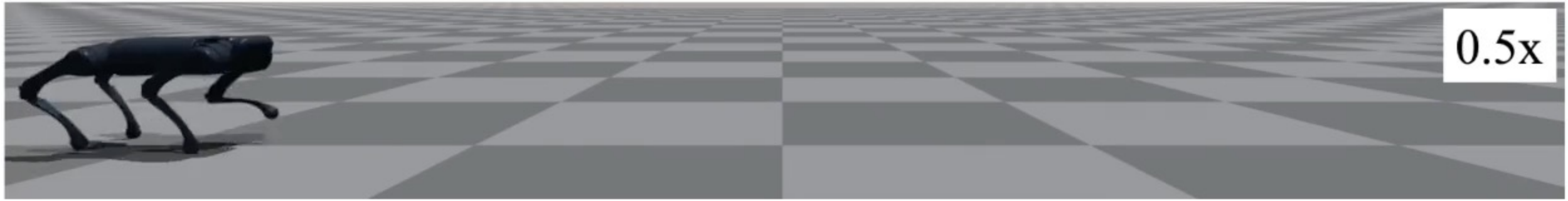
	definition	weight
Linear velocity tracking	$\phi(\mathbf{v}^*_{b,xy} - \mathbf{v}_{b,xy})$	$1dt$
Angular velocity tracking	$\phi(\boldsymbol{\omega}^*_{b,z} - \boldsymbol{\omega}_{b,z})$	$0.5dt$
Linear velocity penalty	$-\mathbf{v}^2_{b,z}$	$4dt$
Angular velocity penalty	$-\|\|\boldsymbol{\omega}_{b,xy}\|\|^2$	$0.05dt$
Joint motion	$-\|\|\ddot{\mathbf{q}}_j\|\|^2 - \|\|\dot{\mathbf{q}}_j\|\|^2$	$0.001dt$
Joint torques	$-\|\|\boldsymbol{\tau}_j\|\|^2$	$0.00002dt$
Action rate	$-\|\|\mathbf{q}^*_j\|\|^2$	$0.25dt$
Collisions	$-n_{collision}$	$0.001dt$
Feet air time	$\sum_{f=0}^{4}(\mathbf{t}_{air,f} - 0.5)$	$2dt$



N. Rudin, D. Hoeller, P. Reist, M. Hutter. *Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning.* CORL 2021

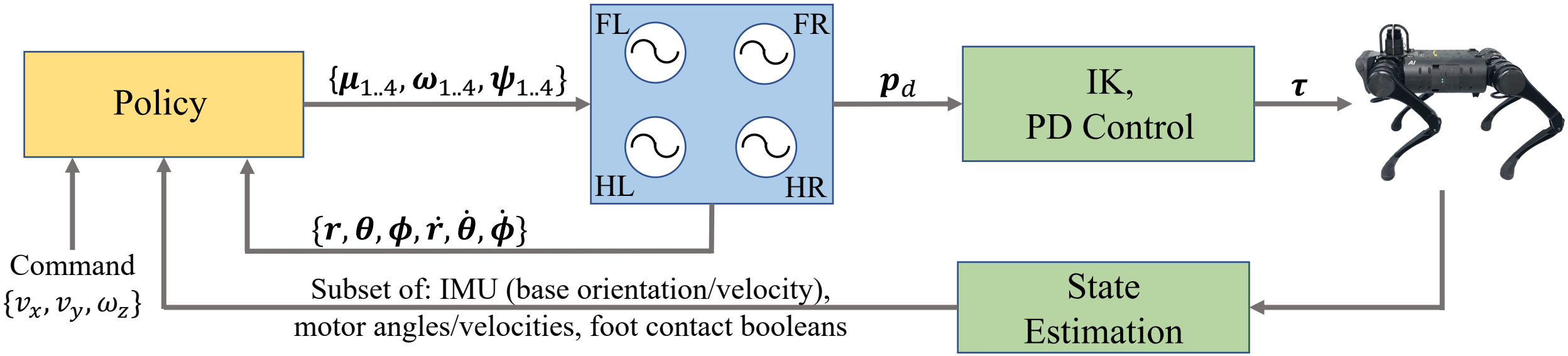
Training Control Policies with Deep Reinforcement Learning

- Training in joint space with task-minimalistic reward functions gives rise to unnatural gaits
- With extensive reward shaping, more natural behaviors can emerge
- However:
  - What has been learned? (interpretability?)
  - How should we weight and shape reward functions for this (or other) tasks?
  - Can we specify a desired swing foot ground clearance?
  - Can we draw any connections to biological systems?



N. Rudin, D. Hoeller, P. Reist, M. Hutter. *Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning.* CORL 2021

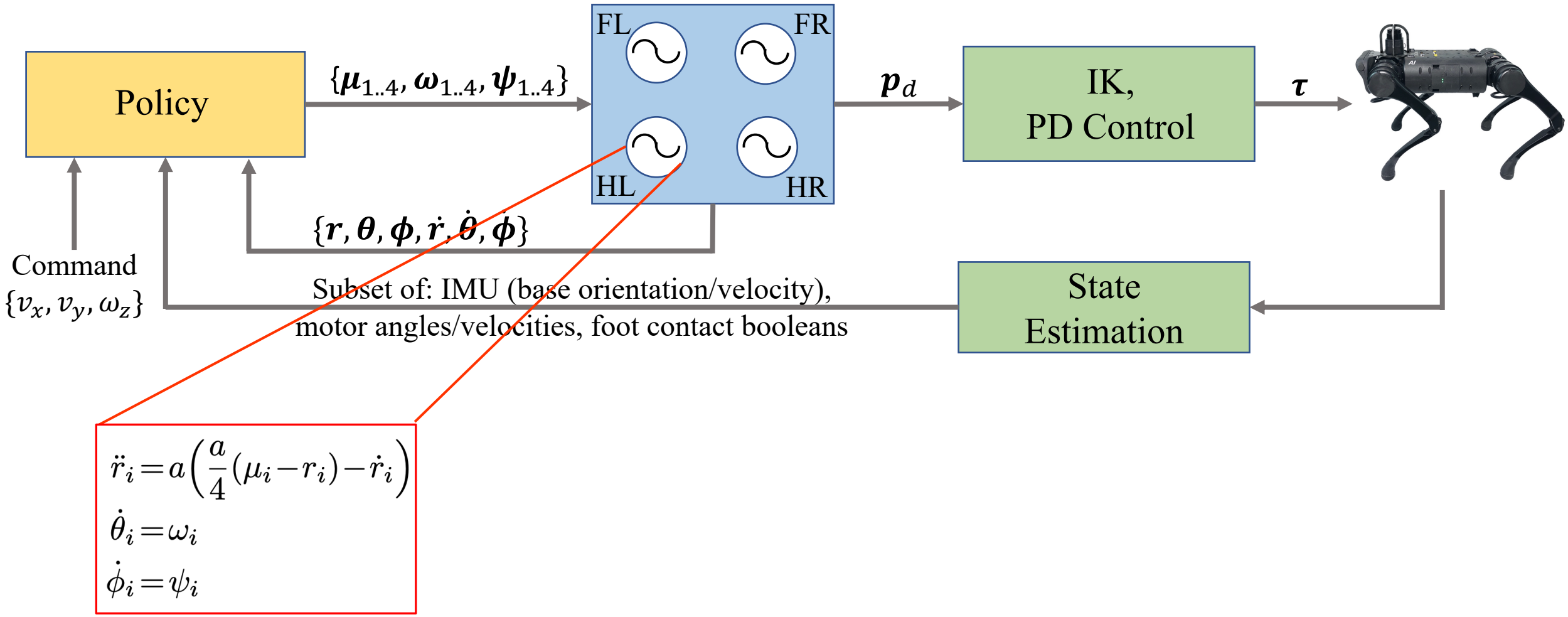
CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion



In this work, we propose to learn to modulate the neural oscillator intrinsic amplitude and frequency of each oscillator that together forms a Central Pattern Generator with deep reinforcement learning.

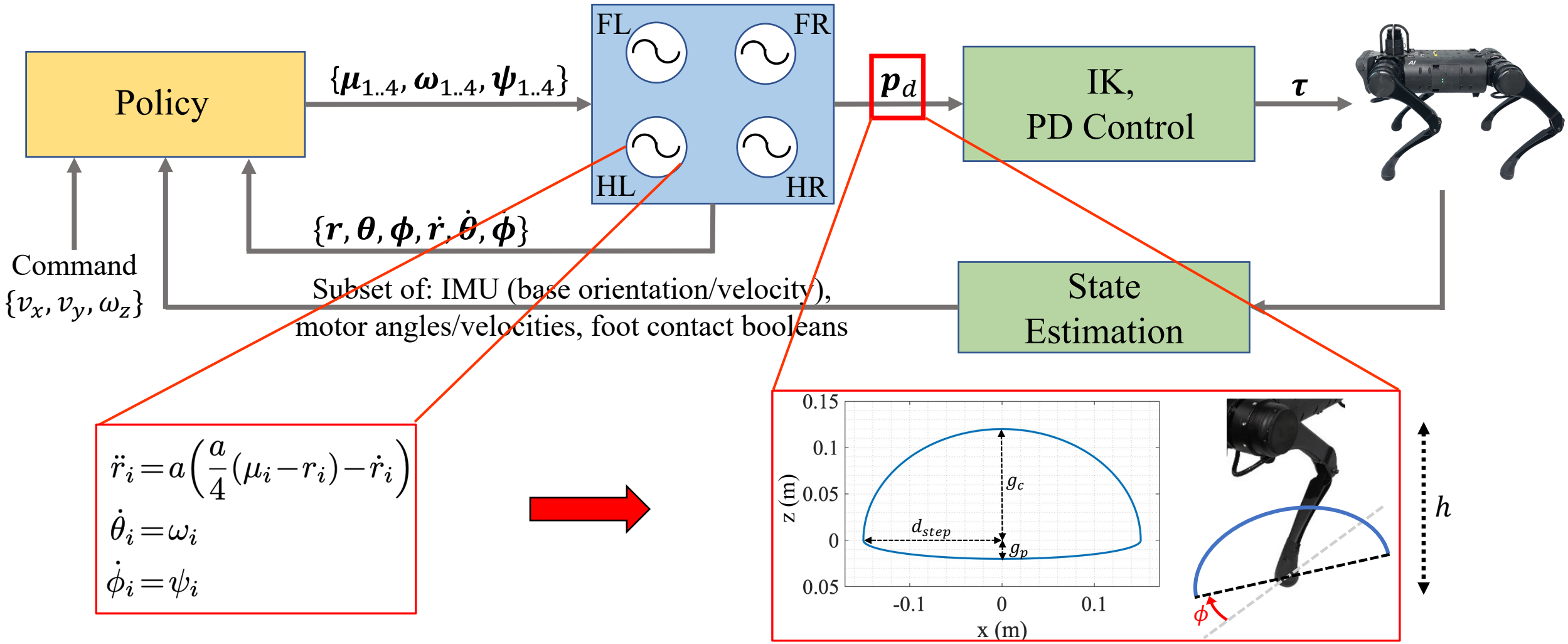
G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion



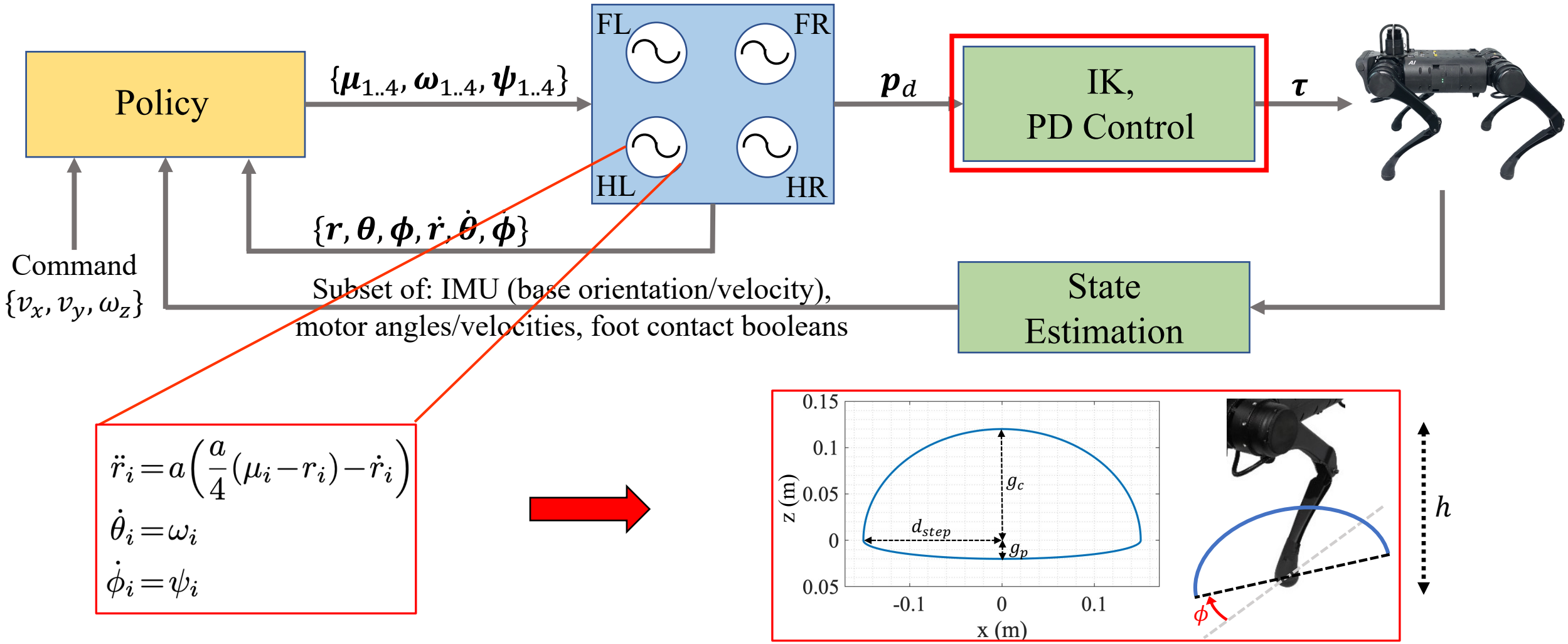
G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion



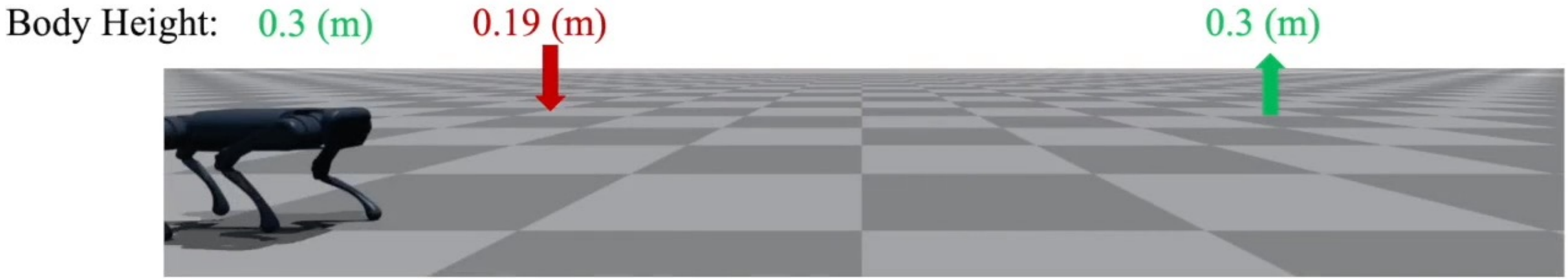
G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

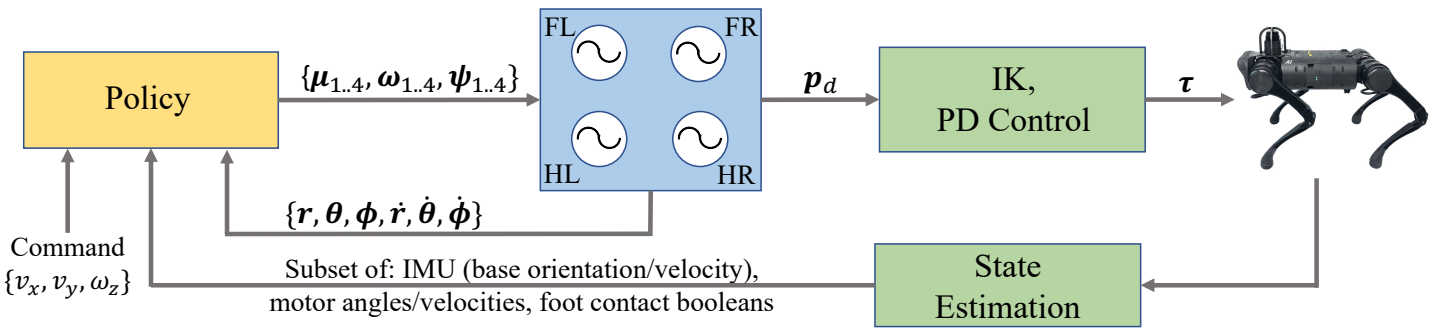
CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion

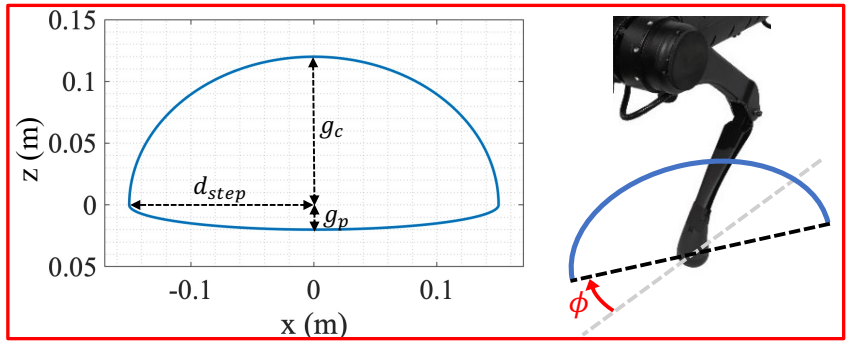


G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

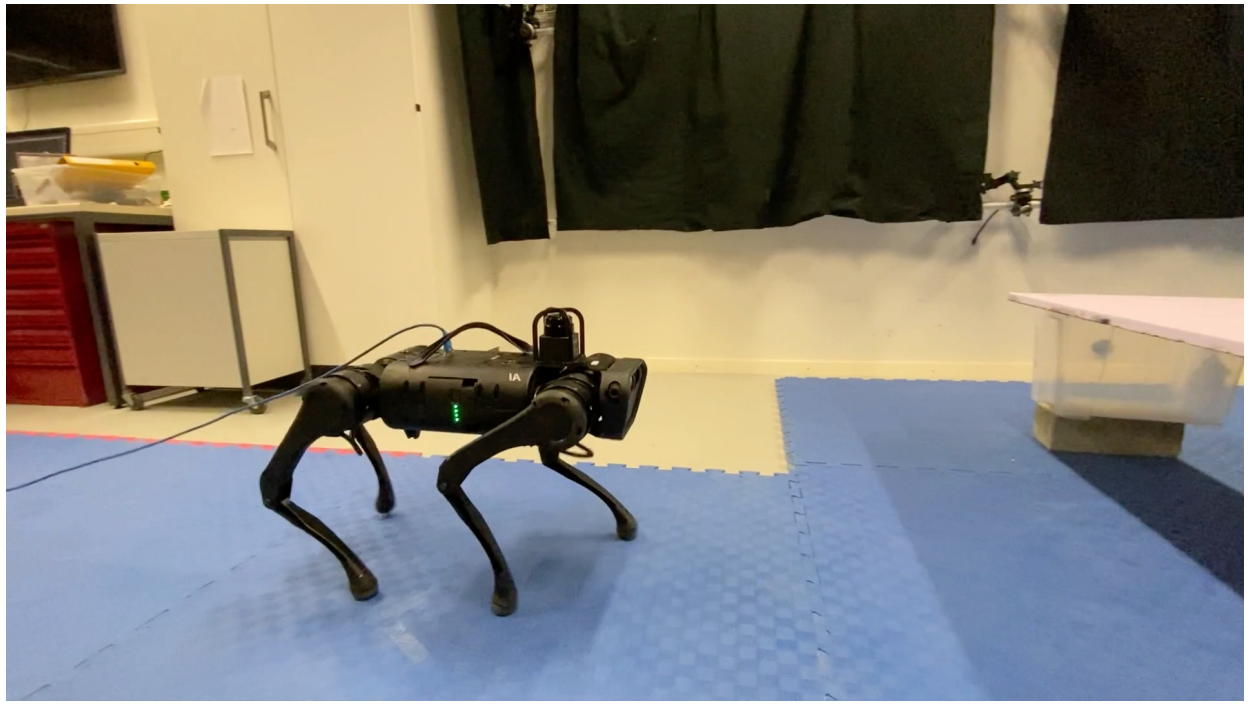
Body Height and Swing Foot Height can be adjusted on the fly

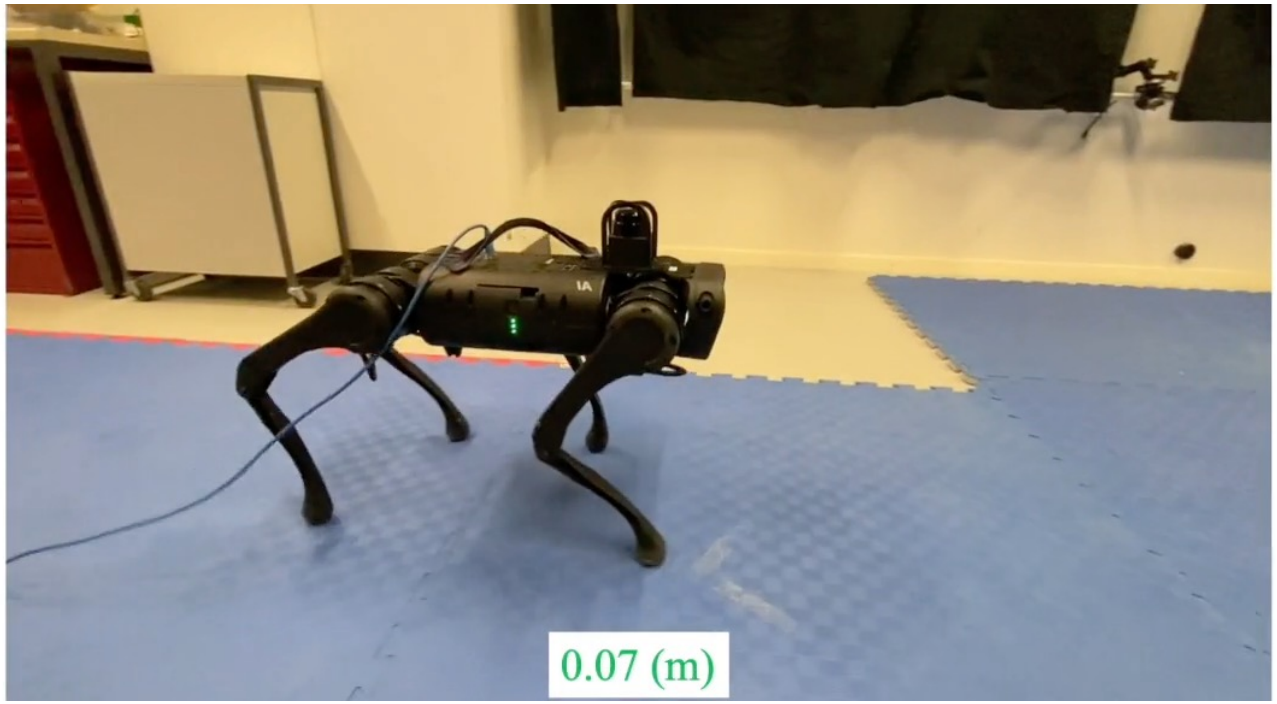


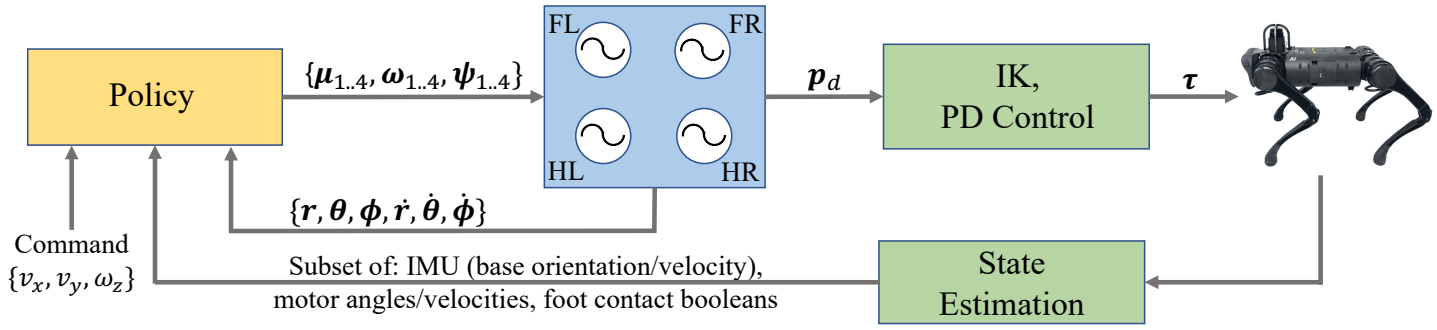


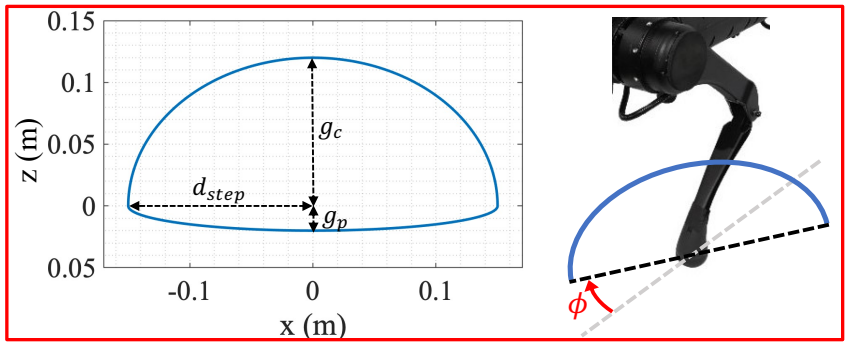


Body Height and Swing Foot Height can be adjusted on the fly



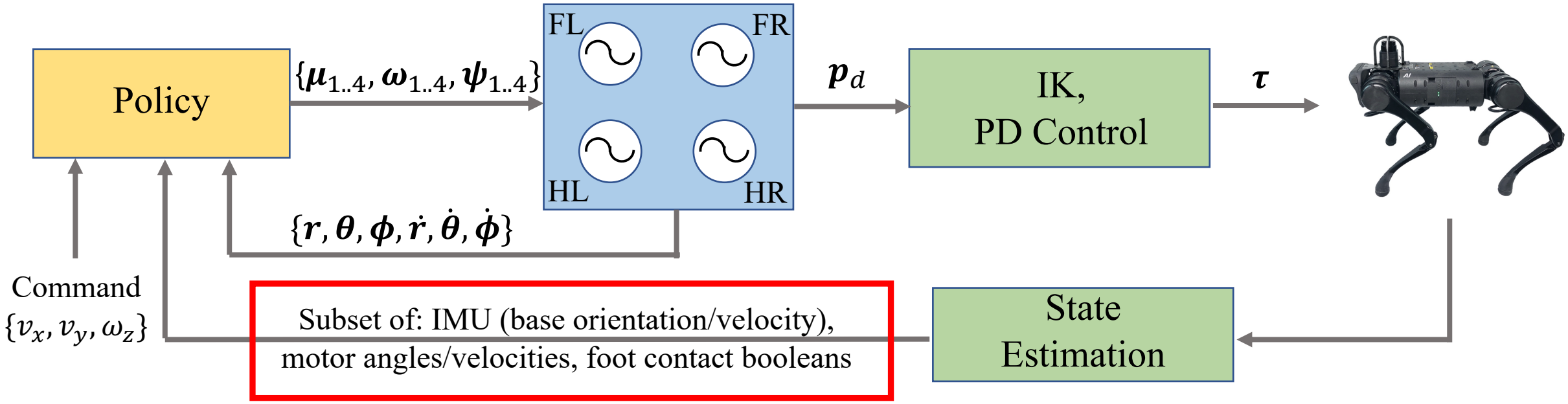






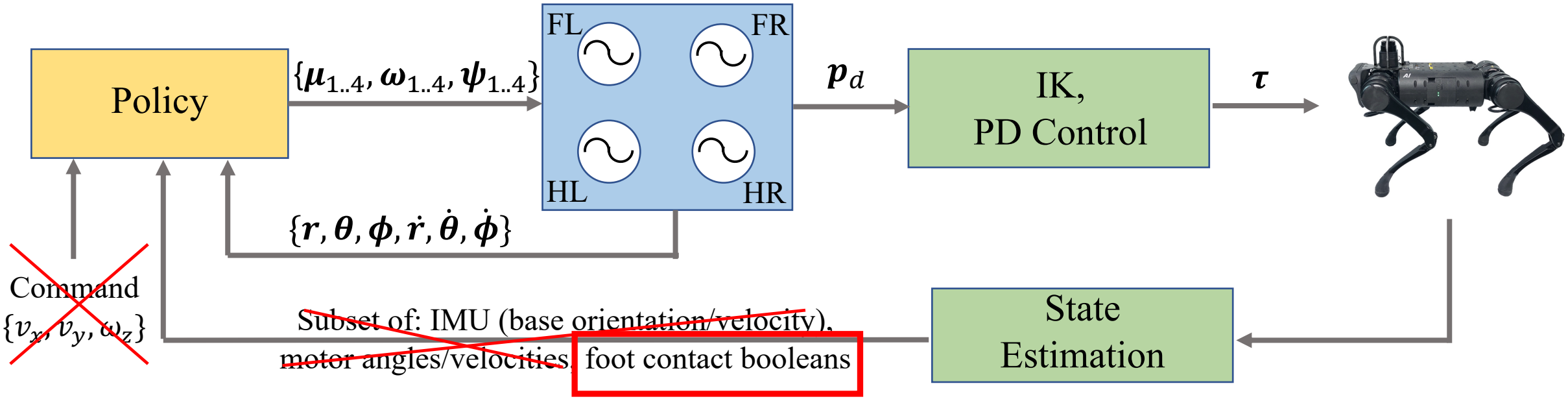
G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

What sensory information should be in the observation space?



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

What sensory information should be in the observation space?



- Similar to the "Tegotae" feedback idea:

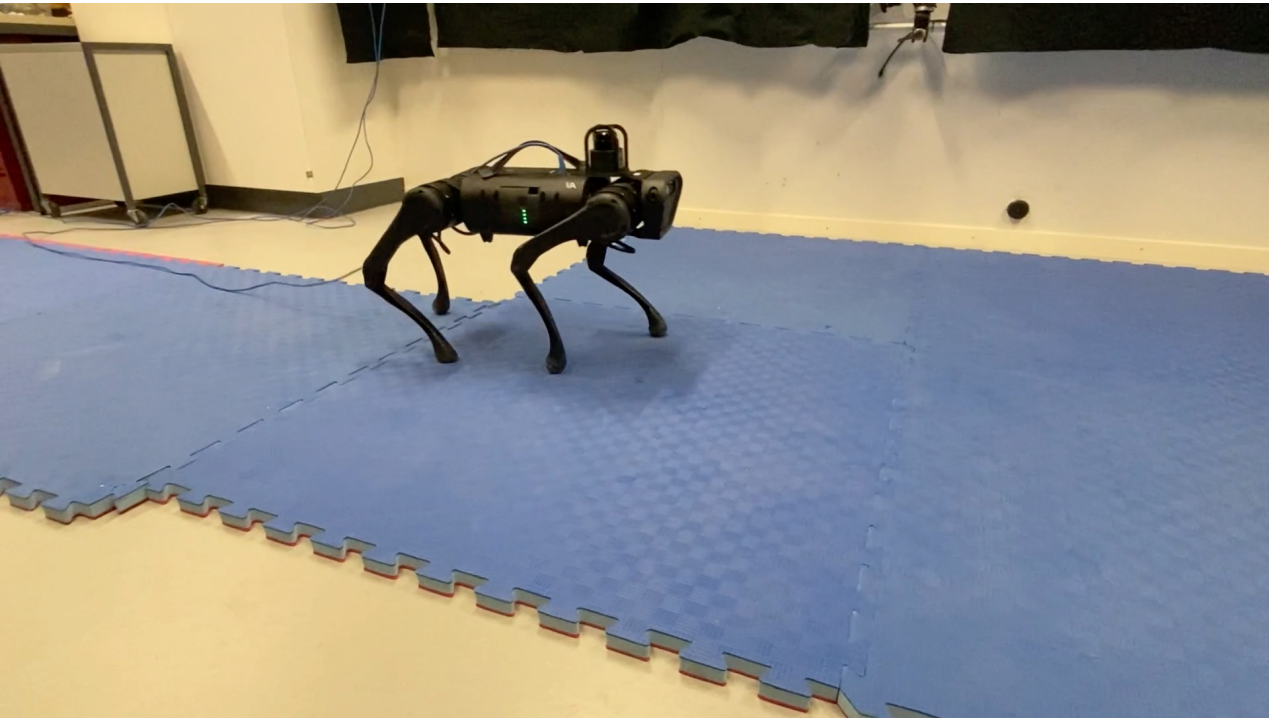
$$\dot{\theta}_i = \omega_i + \sum r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}) - \sigma N_i \cos\theta_i$$

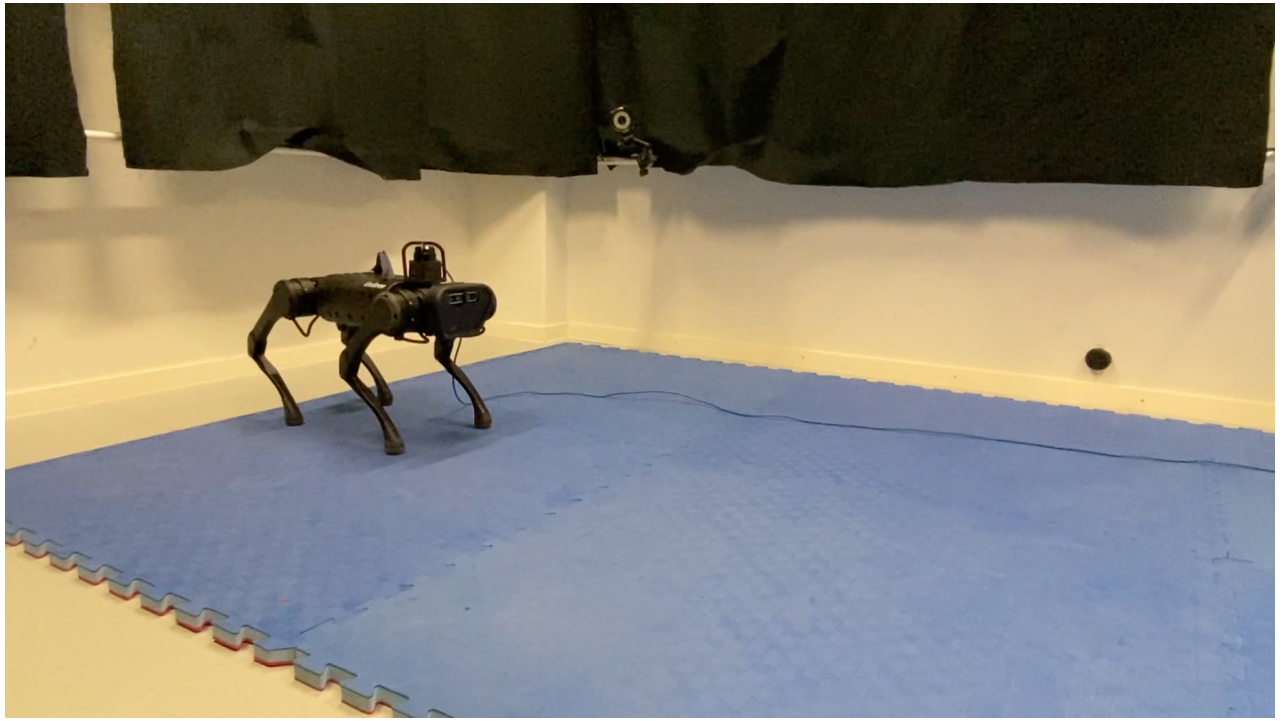
D. Owaki et al. *A Minimal Model Describing Hexapedal Interlimb Coordination: The Tegotae-Based Approach*

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

Observation space: {Foot contact booleans, CPG states}

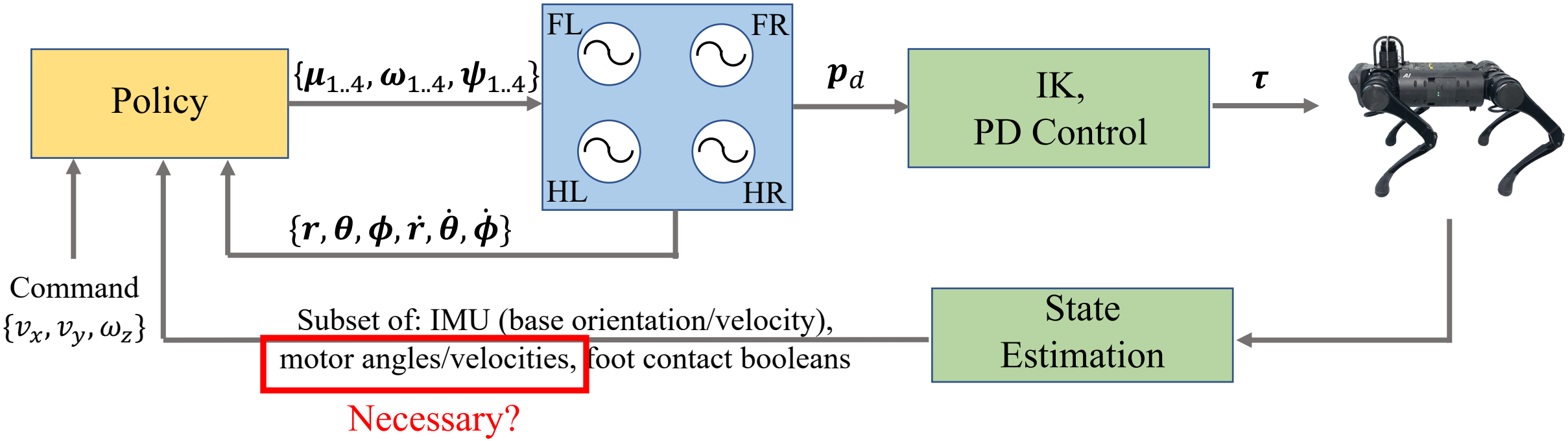
- No domain randomization or noise during training





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

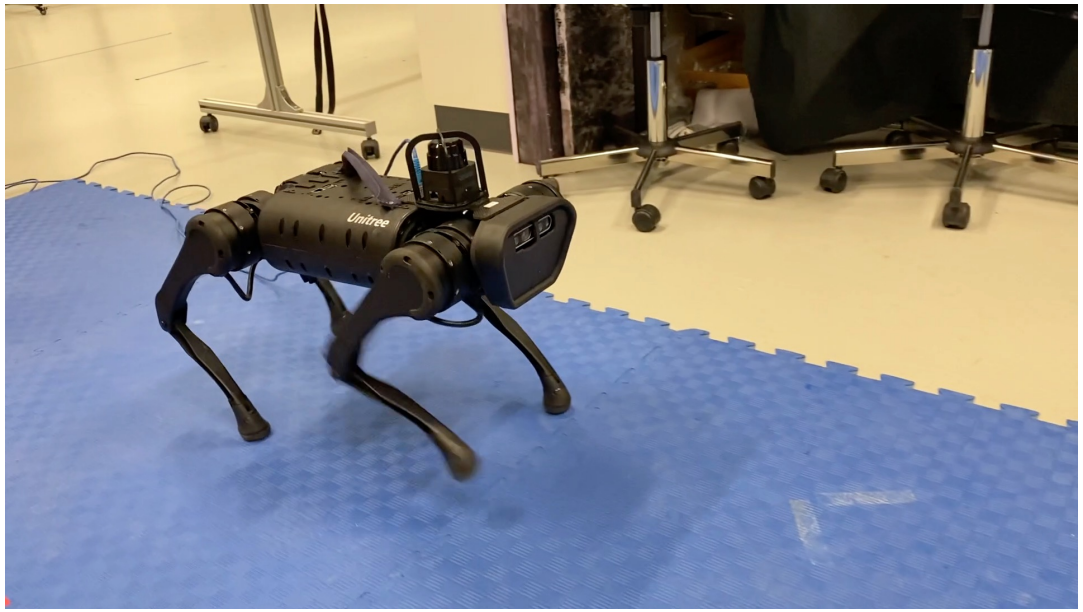
What sensory information should be in the observation space?

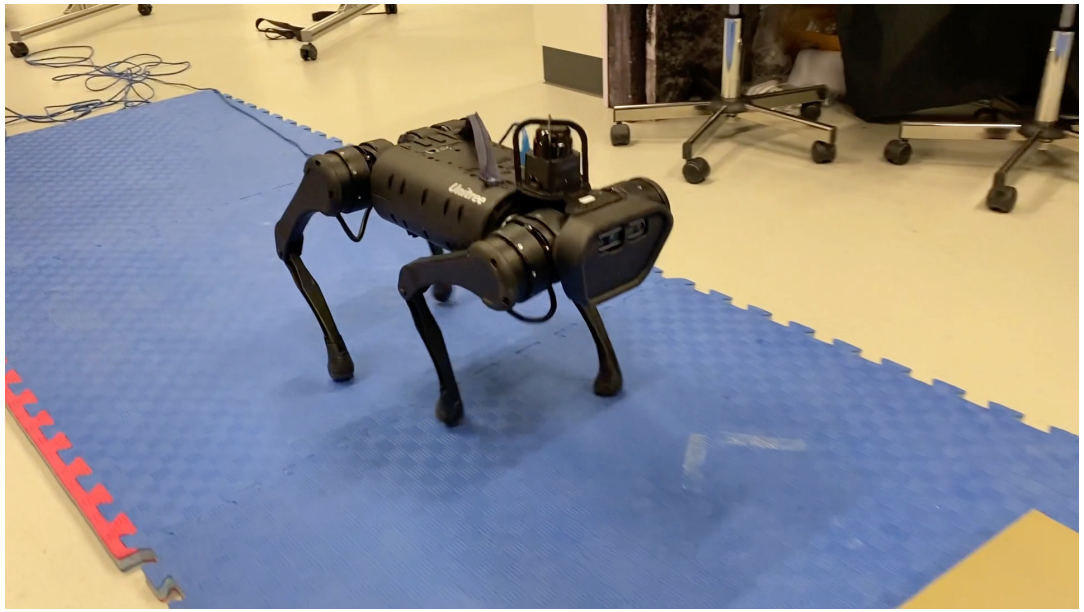


G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

Full Obs. Space, $v^*_{b,x} = 0.3$ [m/s]

Medium Obs. Space, $v^*_{b,x} = 0.8$ [m/s]
(No joint states in obs. space)





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

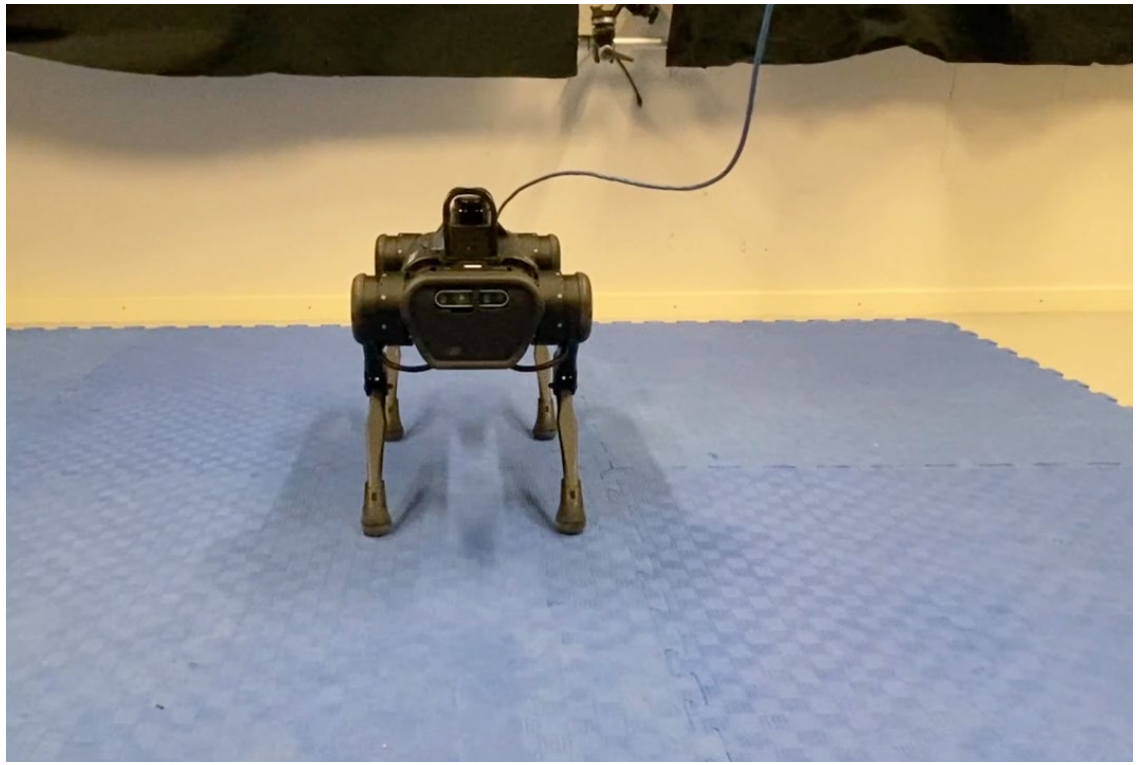
Dynamically Added Mass (5+5+2.5+1.25=13.75 kg)

Medium Observation Space, $v^*_{b,x} = 0.8$ [m/s]

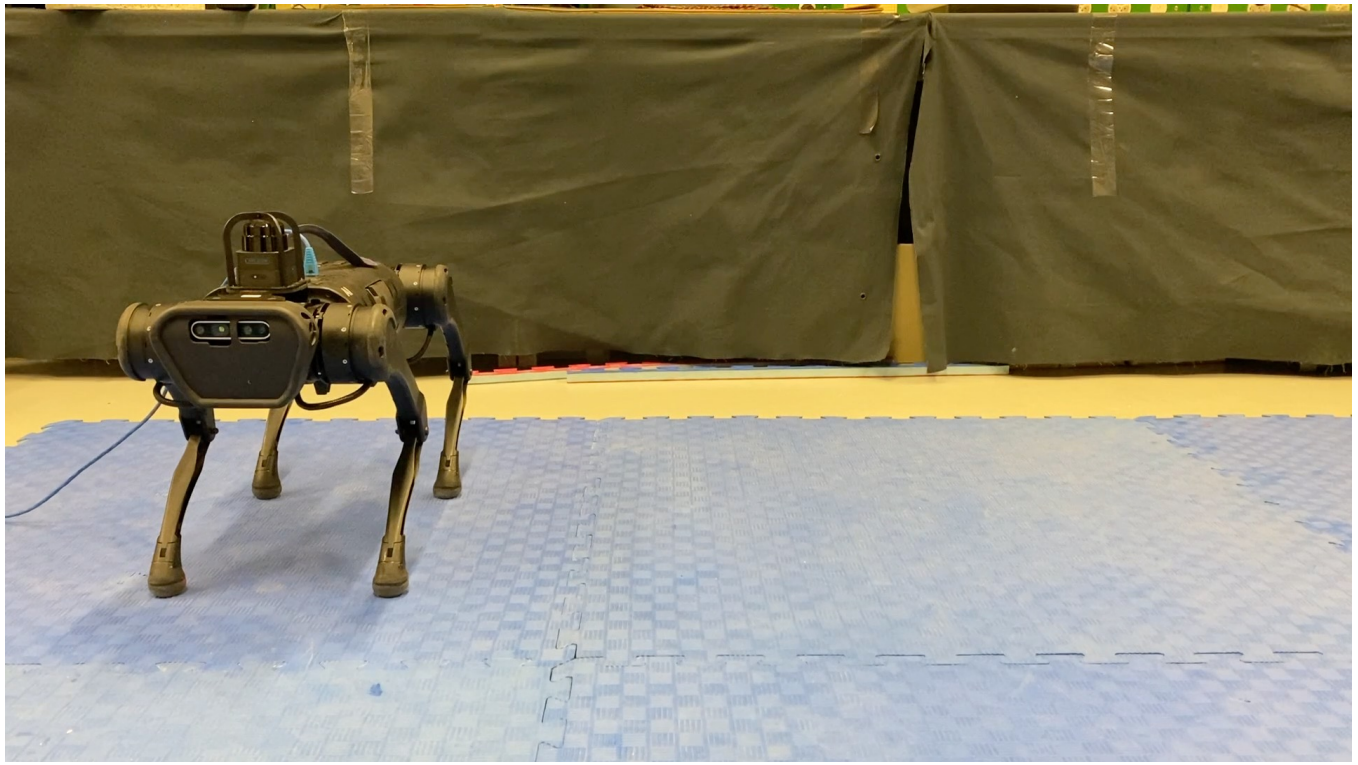
G. Bellegarda, A. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," RA-L 2022

Omnidirectional Locomotion

Full Obs. Space, $\omega^*_{b,z} = 0.5$ [rad/s]



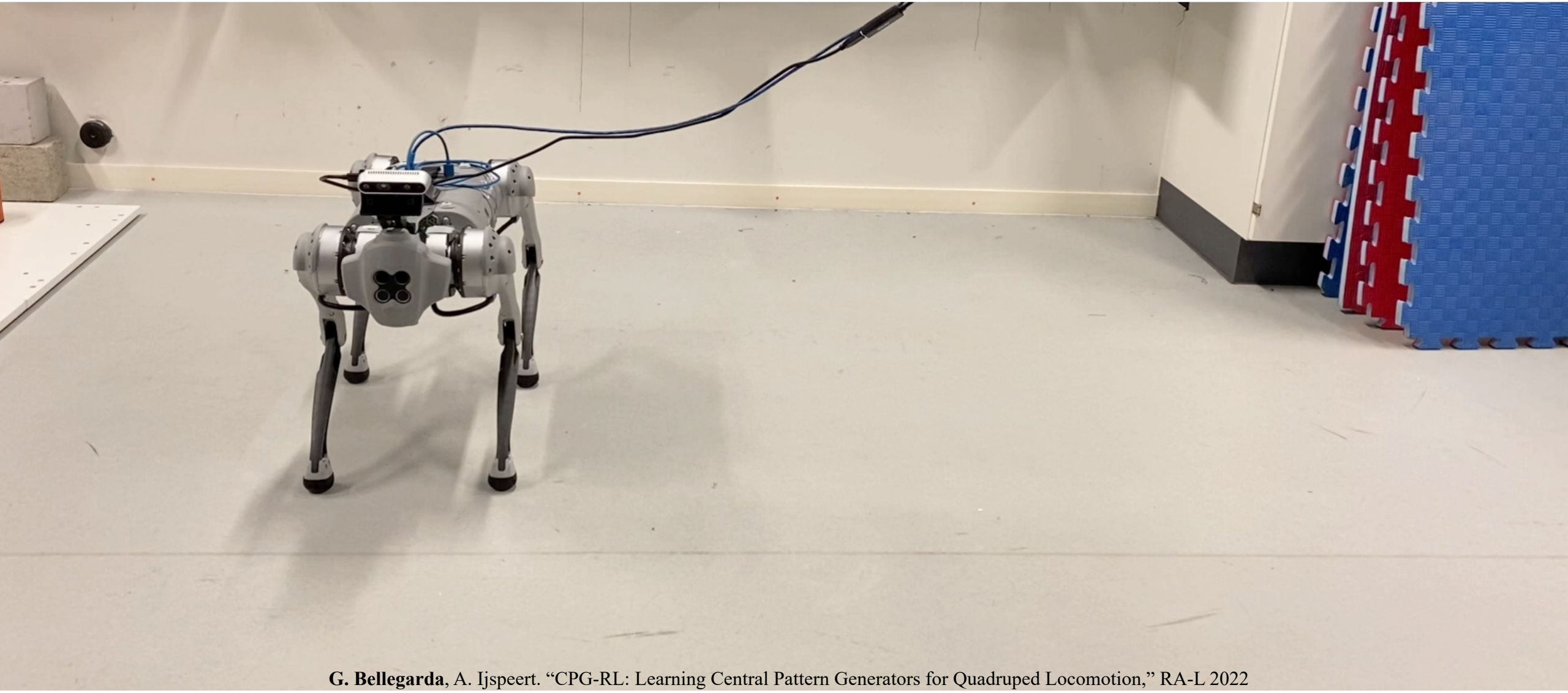
Medium Obs. Space, $v^*_{b,y} = 0.5$ [m/s]



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

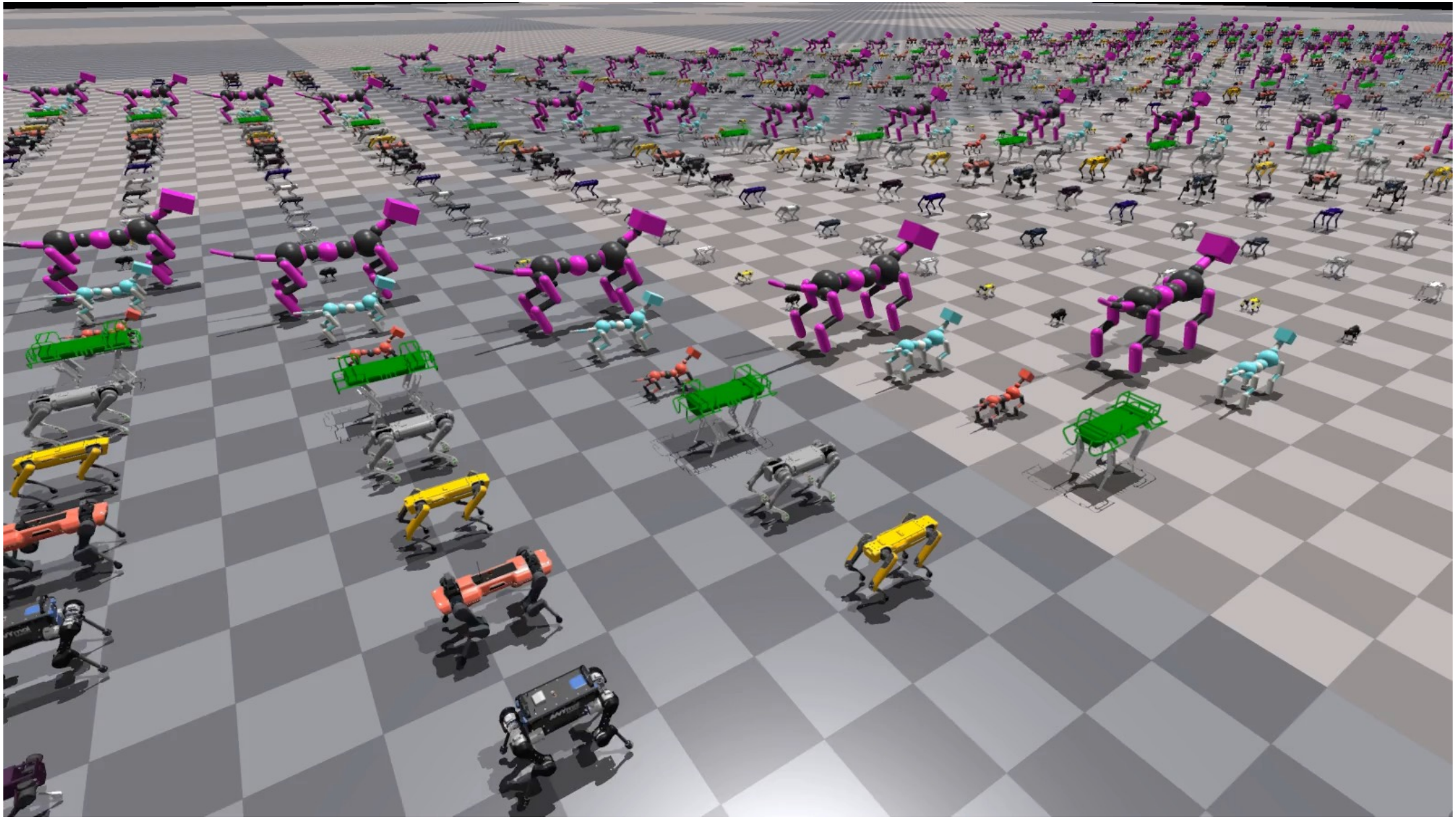
Omnidirectional Locomotion

- Command: $v^*_{b,y} = 0.4$ [m/s]



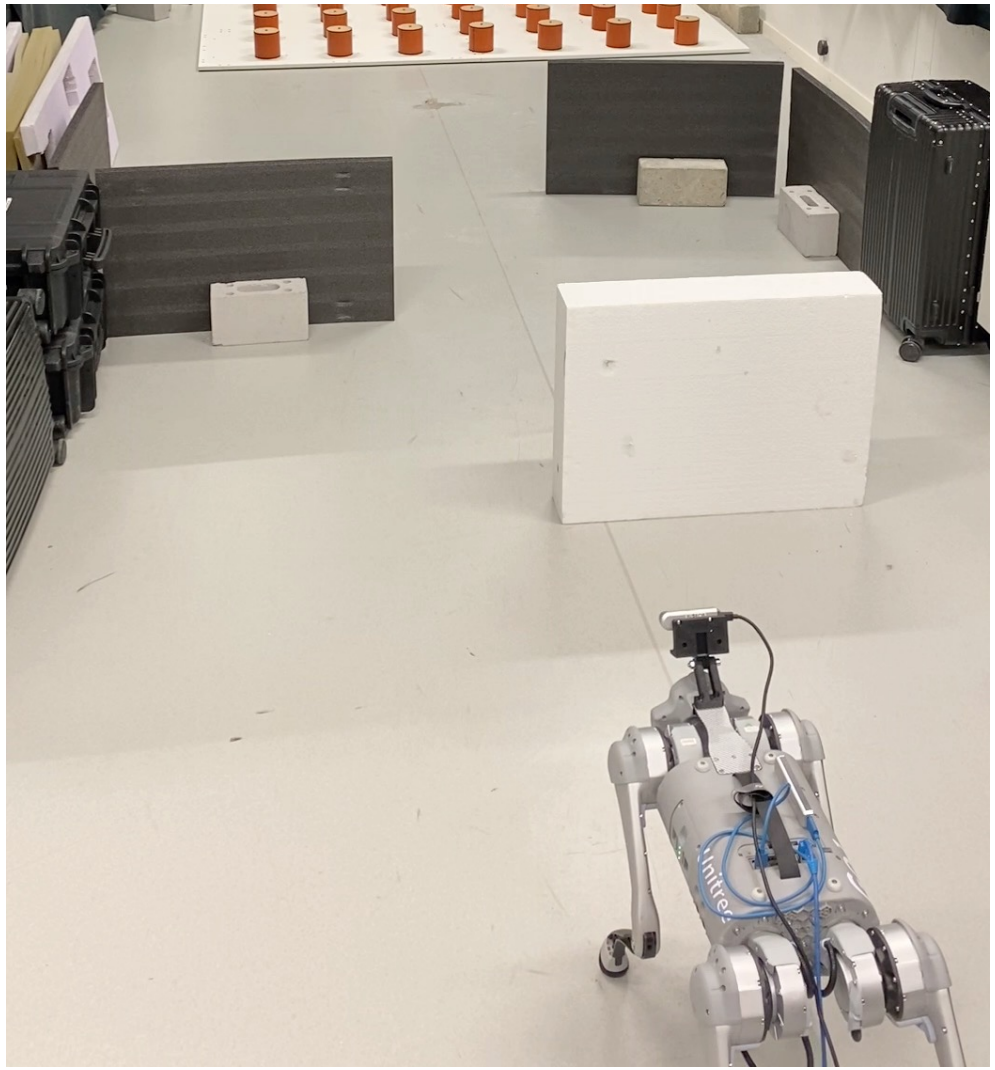
G. Bellegarda, A. Ijspeert. “CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion,” RA-L 2022

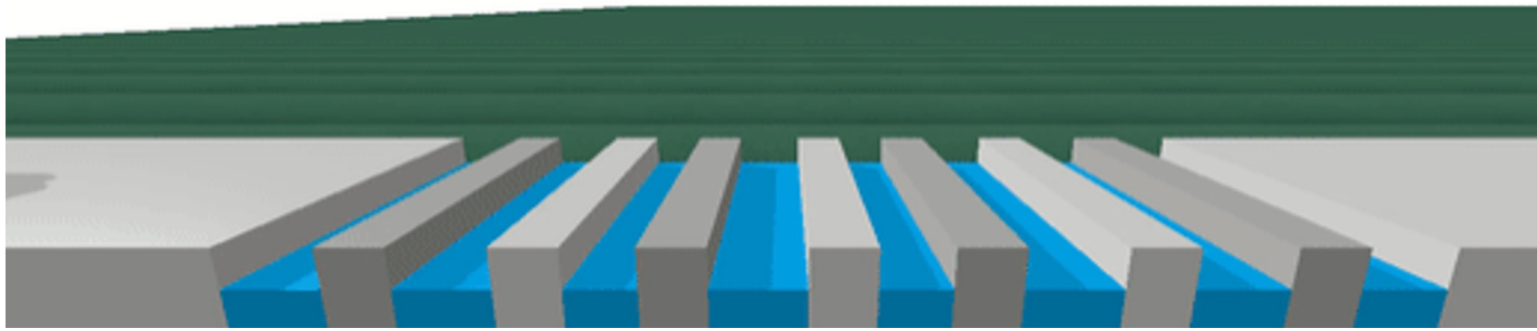
ManyQuadrupeds: A single locomotion policy capable of controlling diverse robots



M. Shafiee, **G. Bellegarda**, A. Ijspeert. “ManyQuadrupeds: Learning a Single Locomotion Policy for Diverse Quadruped Robots,” ICRA 2024

Visual CPG-RL





G. Bellegarda, M. Shafiee, A. Ijspeert. “Visual CPG-RL: Learning Central Pattern Generators for Visually-Guided Quadruped Locomotion,” ICRA 2024
M. Shafiee, **G. Bellegarda**, A. Ijspeert. “Viability Leads to the Emergence of Gait Transitions in Learning Anticipatory Quadrupedal Locomotion Skills,” Nature Communications 2024
M. Shafiee, **G. Bellegarda**, A. Ijspeert. “Learning Anticipatory Quadrupedal Locomotion Based on Interactions of a Central Pattern Generator and Supraspinal Drive,” ICRA 2023

CPG-RL Summary

- CPG-RL: integrated abstract oscillators into the deep reinforcement learning framework to learn quadruped locomotion
- Online modulation of body height and swing foot height
- Robust sim-to-real transfer (115% nominal mass load, uneven terrain)
- Minimal proprioceptive sensing (only contact Booleans)
- Training without any dynamics randomization or noise
- Can learn a single policy for Many Quadrupeds!
- Vision in-the-loop for navigation and gap crossing → emergence of gait transitions

Outline

I. Motivation and Goals
II. Model-Predictive Control vs. Reinforcement Learning (same?)
III. Model-based methods
    I. Background on RoboSimian and designing skating motions
    II. Trajectory optimization allowing wheel slip
IV. Learning-based methods
    I. Action space in reinforcement learning [RoboSimian, A1]
    II. Augmenting motion planning with deep reinforcement learning [A1]
V. Bio-inspired learning
VI. Conclusion

Conclusion

- Presented several methods for agile robot locomotion:
  - Designed skating trajectories [RoboSimian]
    - 3- vs. 4-limb skating on flat vs. “rough” terrain
  - Trajectory optimization methods [RoboSimian, car]
    - Novel wheel model to exploit or avoid slipping, gaits
  - Deep learning (incorporating ideas from control)
    - Simple methods like forward/inverse kinematics can have a huge benefit for learning [RS, A1]
    - Augmented jumping TO with DRL for improved robustness [A1]
  - Bio-inspired learning
    - Oscillators in the loop greatly simplify and robustify the sim-to-real transfer [A1,Go1]
- Explored trade-offs relating to our goals of robustness, agility and efficiency

Possible Exam Questions

- How are model-based methods and learning-based methods similar? How are they different?
- When might you consider using a model-based controller over a learning-based controller, and vice-versa?
- Assume you need to design a locomotion controller with trajectory optimization. What would your cost function include? What kind of constraints do you need?
- Assume you need to train a locomotion controller with deep reinforcement learning. How will you structure the Markov Decision Process (i.e. observation space, action space, reward function)?
- What makes reinforcement learning challenging? What might you consider when applying reinforcement learning to a new problem?