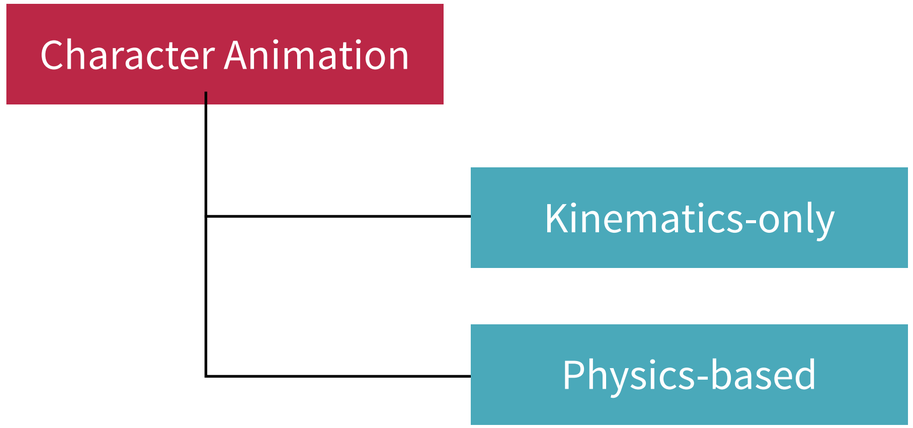
Lecture 18:

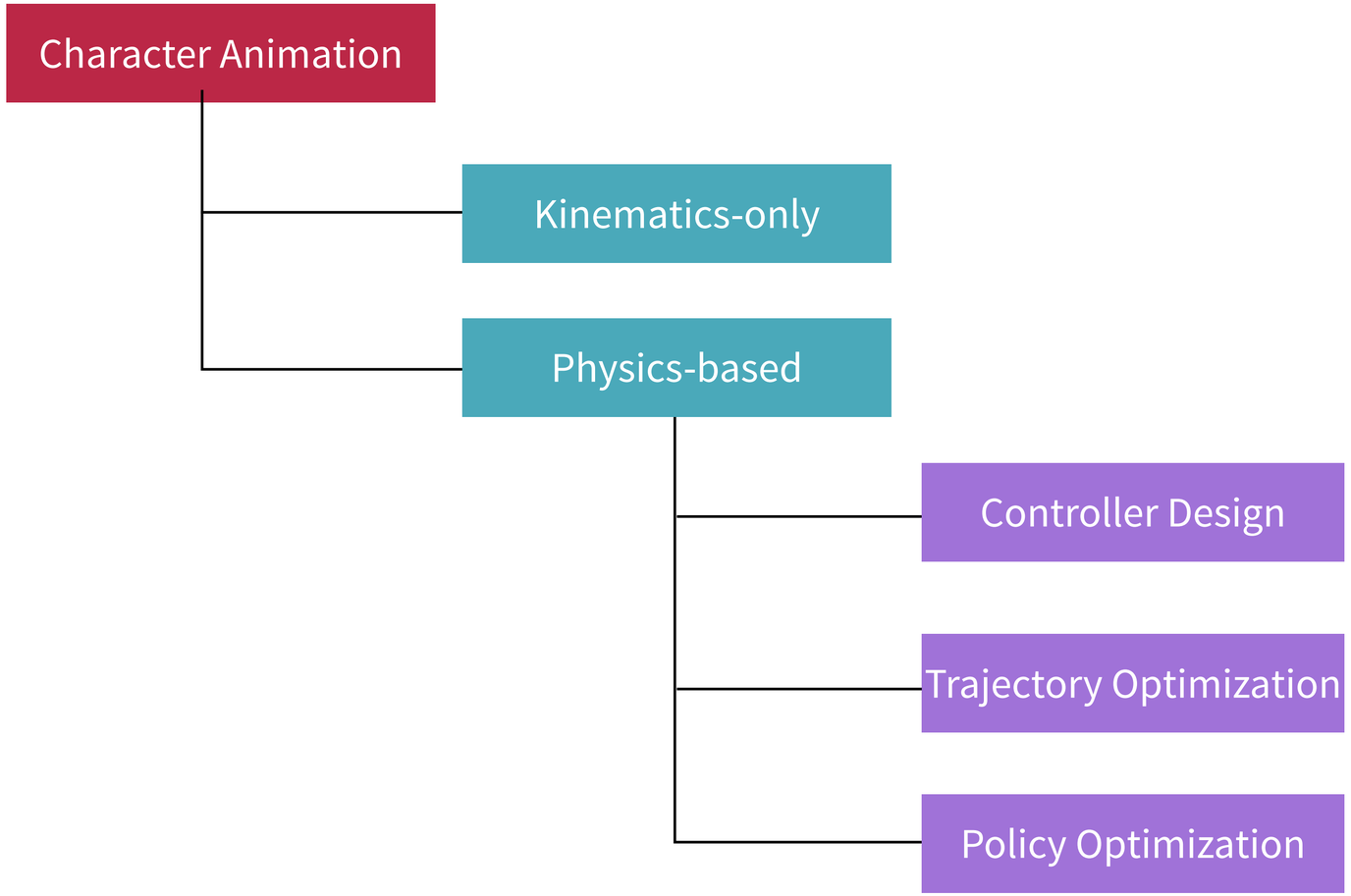
Physics-based Animation

FUNDAMENTALS OF COMPUTER GRAPHICS

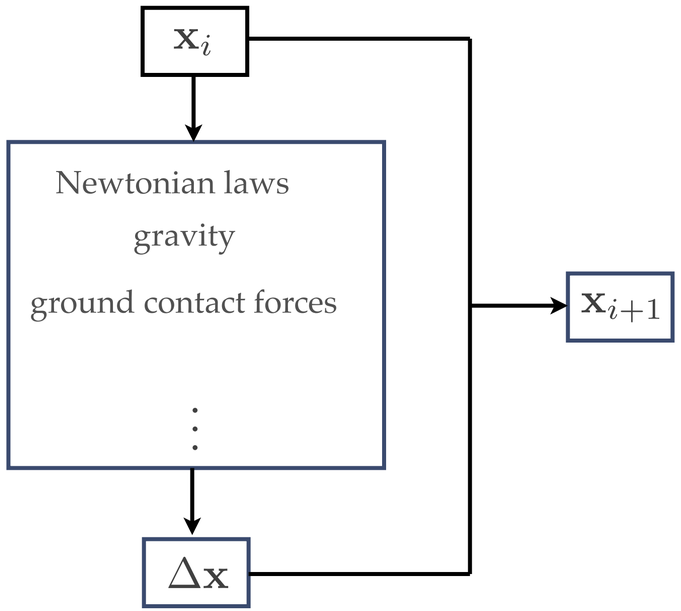
Animation & Simulation

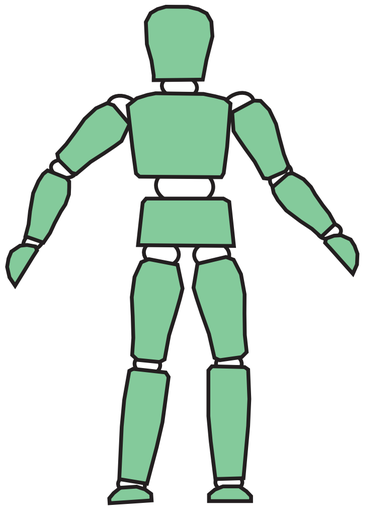
Stanford CS248B, Fall 2023

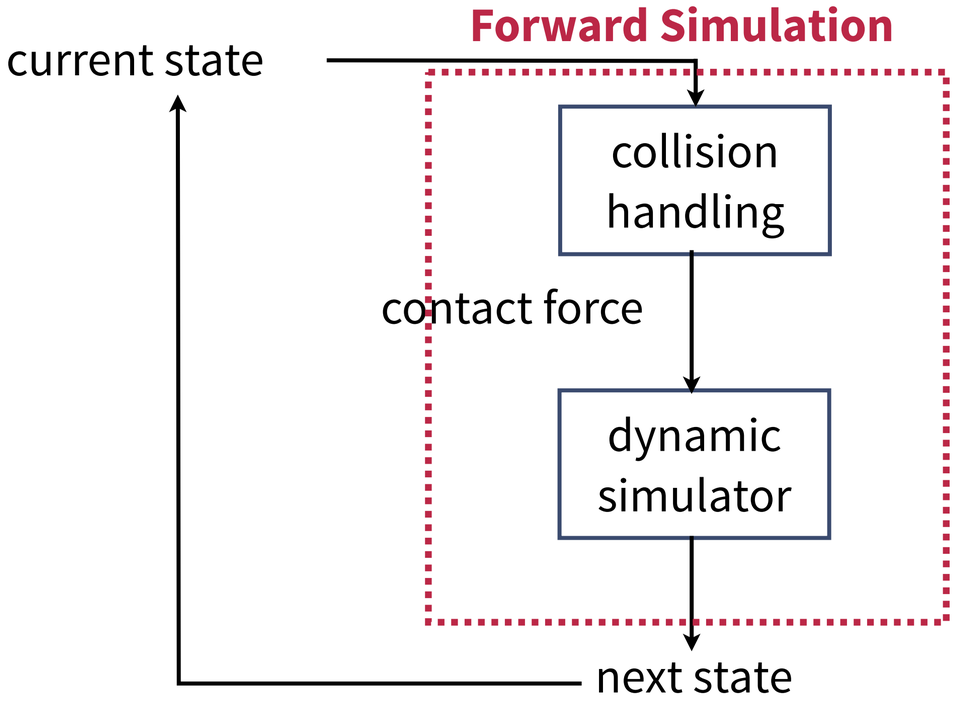

Character Animation
Kinematics-only
Physics-based


Character Animation
Kinematics-only
Physics-based
Controller Design
Trajectory Optimization
Policy Optimization

Simulation

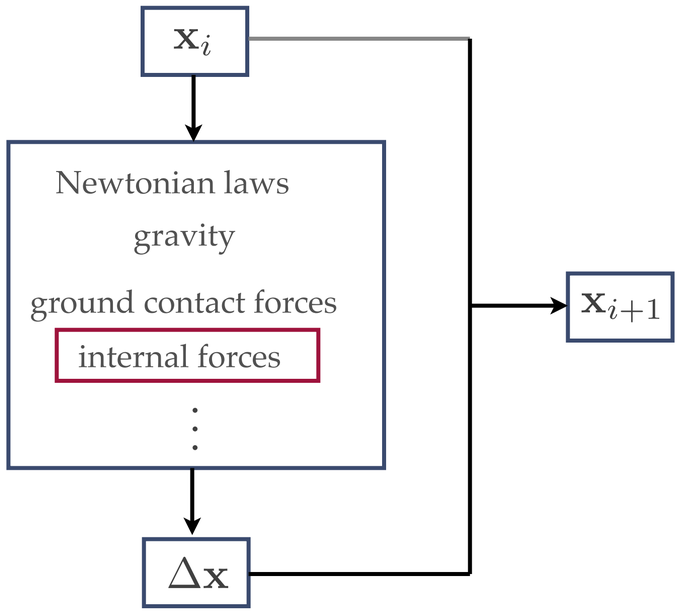

$\mathbf{x}_i$
Newtonian laws
gravity
ground contact forces
$\mathbf{x}_{i+1}$
$\Delta\mathbf{x}$

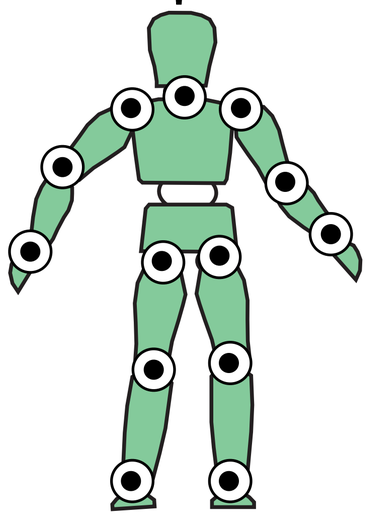



Forward Simulation
current state
collision handling
contact force
dynamic simulator
next state

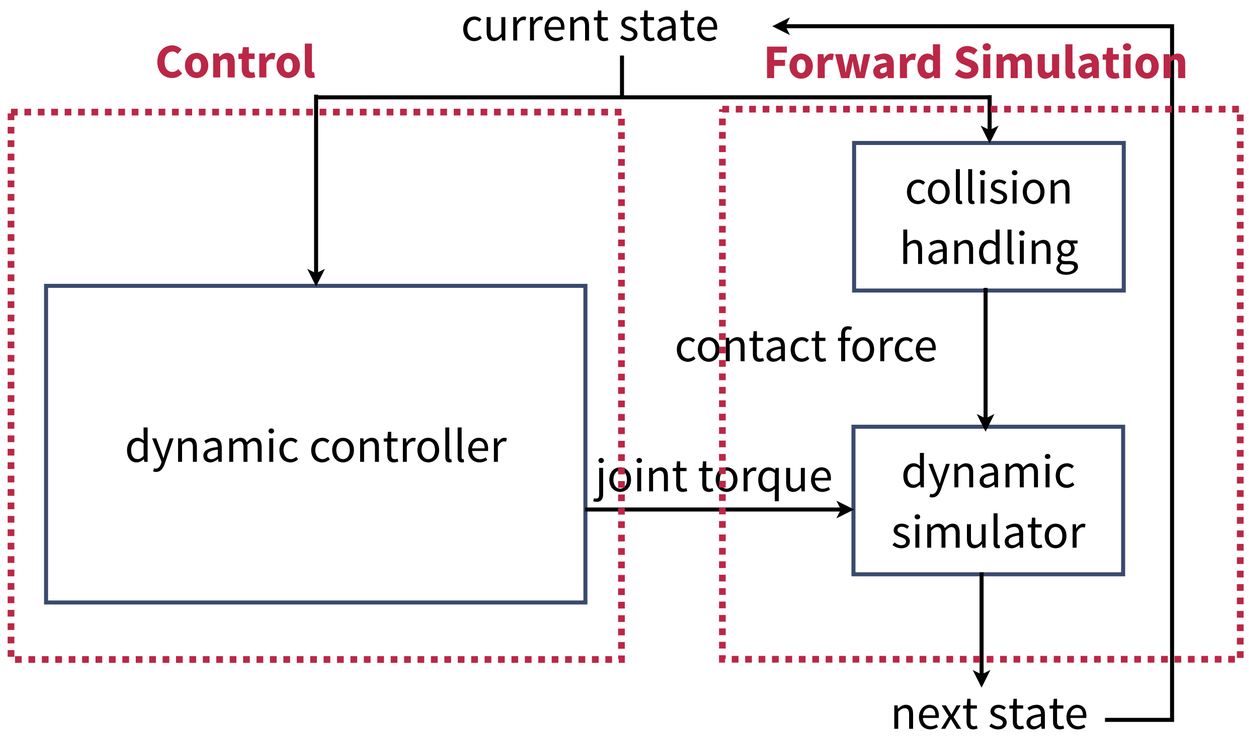
Simulation + Control

If # of effective actuators is smaller than #dofs, the system is **under-actuated**.

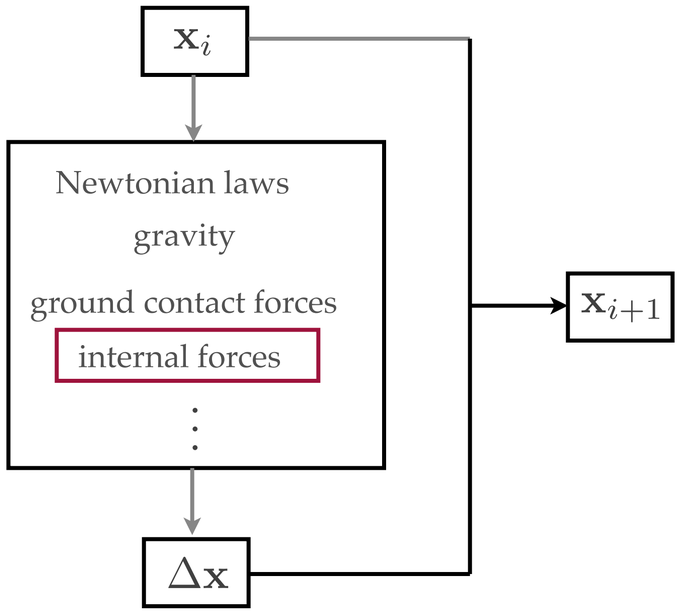


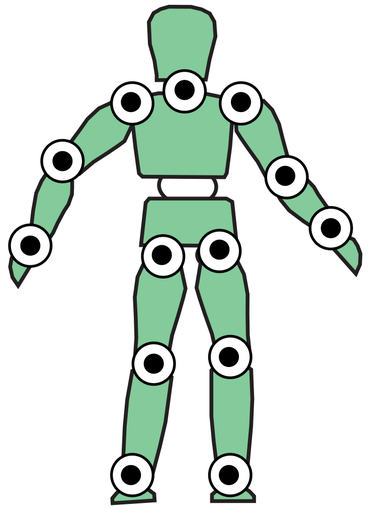


actuators

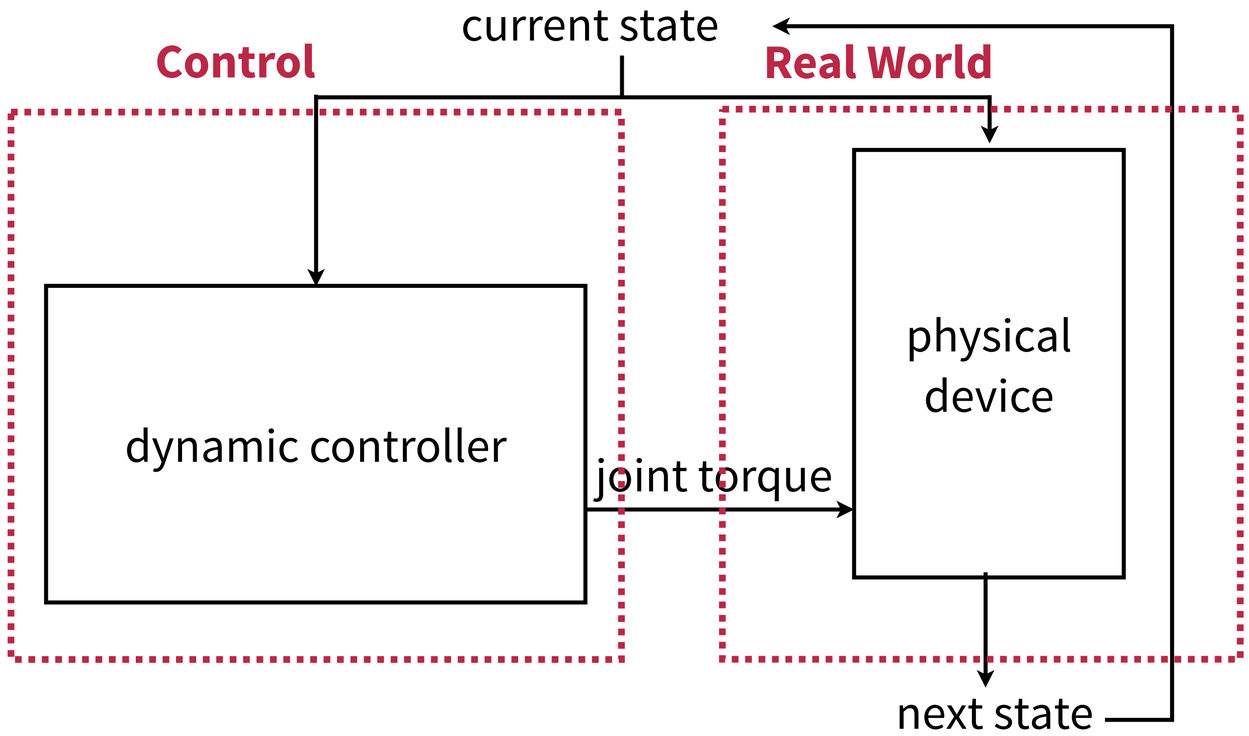


Simulation + Control





actuators

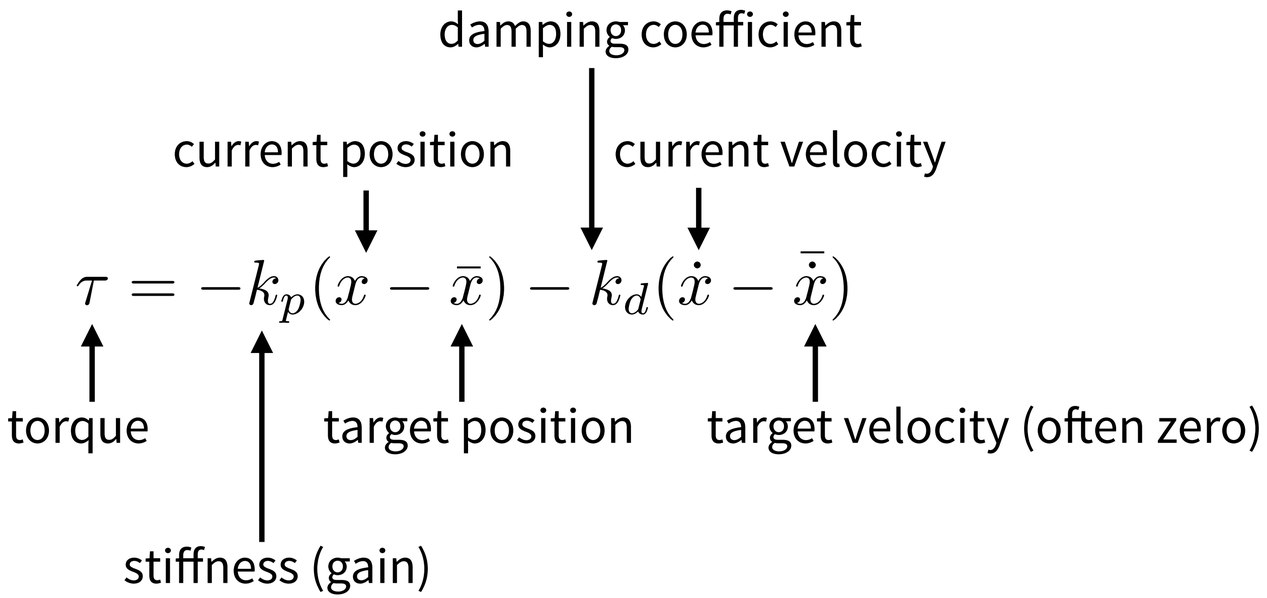


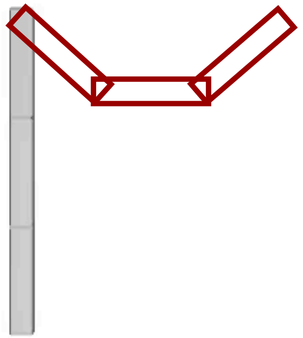
Feedforward vs Feedback

- Feedforward control applies a pre-defined signal without responding to the current state of the system.
  - The control signal must be pre-computed based on the mathematical model of the system and knowledge about the process disturbances.
  - The world must be predictably unchanging with time and there must not be any unmeasured disturbances.
- Feedback control takes account of the current state to adjust the control signal.
  - The feedback signal is observed and used it to compute control signal that improves the performance for the next state.
  - Can counter unexpected disturbance, model uncertainty and instability in the system.

Proportional-Derivative control

- Use proportional-derivative (PD) controllers to compute joint torques.
- Each joint is actuated by a PD servo to track a desired pose or trajectory





Stiffness and damping

high stiffness (gain)

$$\tau = -k_p(x - \bar{x}) - k_d\dot{x}$$

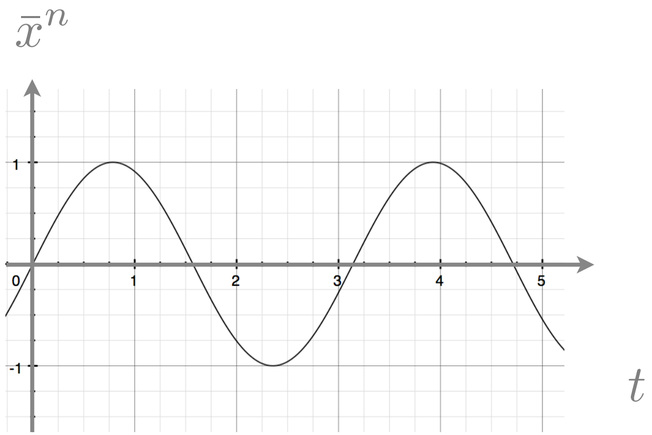
high damping

$$\tau = -k_p(x - \bar{x}) - k_d\dot{x}$$

Track a trajectory using PD

- Can track a sequence of poses by varying the target pose over time.
- Tuning coefficients is even harder when tracking a time sequence.

$$\tau^n = -k_p(x^n - \bar{x}^n) - k_d\dot{x}^n$$



Tracking mocap

Zordan and Hodgins 2002

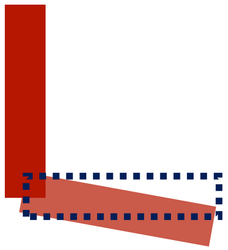
Problems with PD

- Gain and damping coefficients are difficult to determine for complex articulated systems.
- High-gain feedback controllers are less stable.
- Precision and stability are often in conflict.

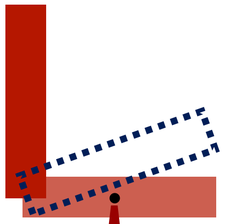
Gravity compensation

- **PD control never achieves the desired joint value under gravity.**
- **Adjust the target position to compensate the effect of gravity.**

target = desired pose ($\frac{\pi}{2}$)



gravity compensated target



$\mathbf{x}$: com of this link

$$-k_p(\theta - \bar{\theta}) = \left(\frac{\partial \mathbf{x}}{\partial \theta}\right)^T \cdot m\mathbf{g}$$

$$\bar{\theta} = \boxed{\frac{m}{k_p}\left(\frac{\partial \mathbf{x}}{\partial \theta}\right)^T \cdot \mathbf{g}} + \theta, \quad \text{where } \theta = \frac{\pi}{2}$$

gravity compensation term

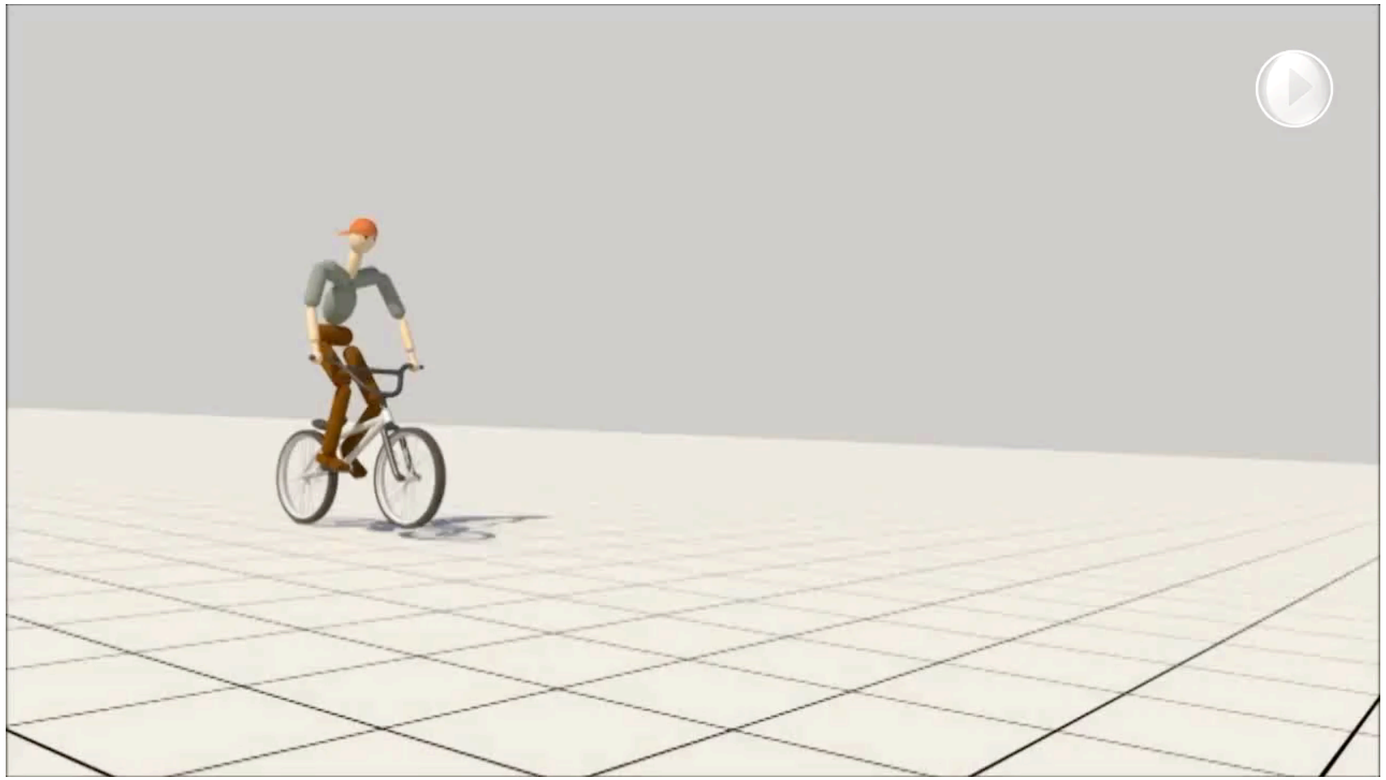
Low-impedance control

- Combine feedforward torques and low-gain feedback controllers.

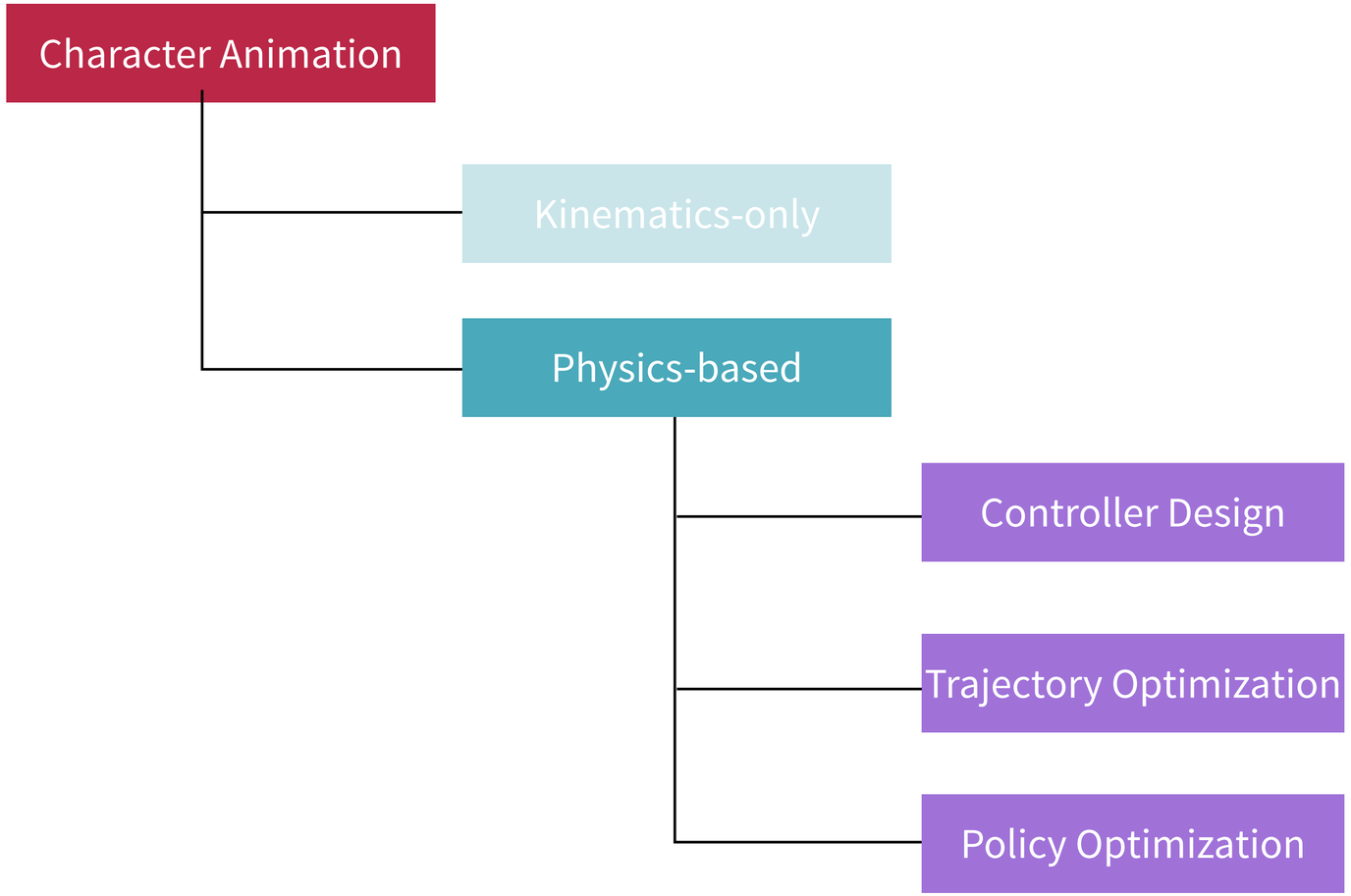
$$\boldsymbol{\tau} = \boldsymbol{\tau}(\mathbf{x}_{target}) - k_p(\mathbf{x} - \mathbf{x}_{target})$$

  - Feedforward controller executes motor control in an open-loop manner.
  - Feedback control is necessary to deal with small deviations from desired trajectory.
- Explains well-trained motion exhibits low joint stiffness.

Quiz

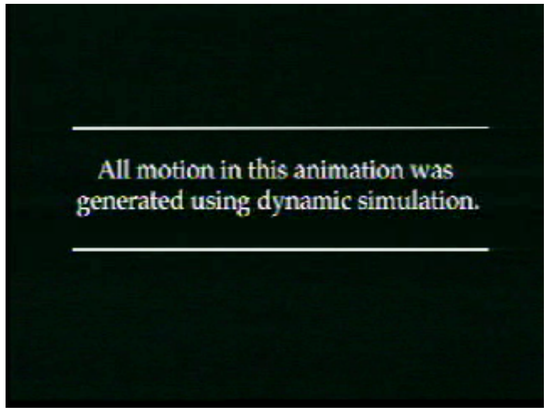


Which phase requires more feedback control?


Character Animation
Kinematics-only
Physics-based
Controller Design
Trajectory Optimization
Policy Optimization

Construct a full-body controller

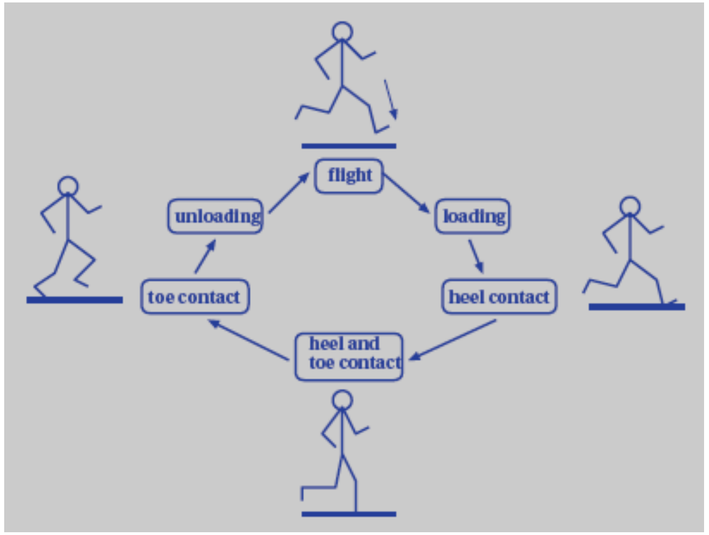
- Build on top of basic control mechanisms to control more complex behaviors.
- Organize basic controllers in a hierarchical structure.
- Require hand-tuning and trial-and-error effort for each specific human model and task.



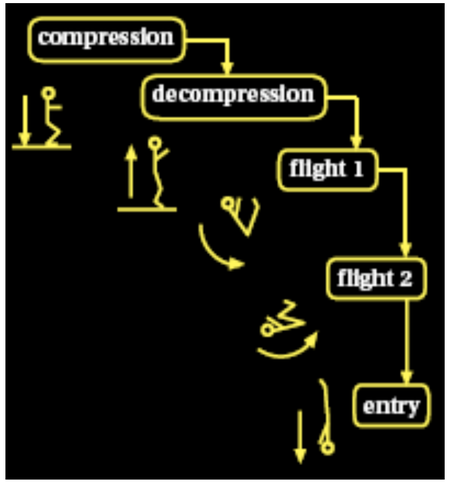
Hodgins et al 1995

Manually designed controller

State machine



Transition sequence



PD pose tracking

Low level control:
proportional–derivative servos

$$\tau = k_p(\theta_d - \theta) - k_v\dot{\theta}$$

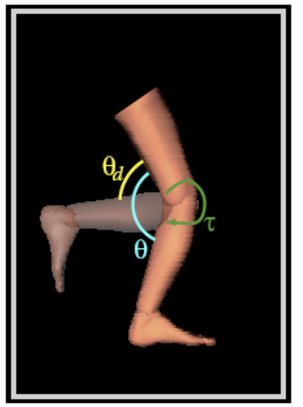
τ = Torque

$(\theta_d - \theta)$ = Error between desired and actual joint angle

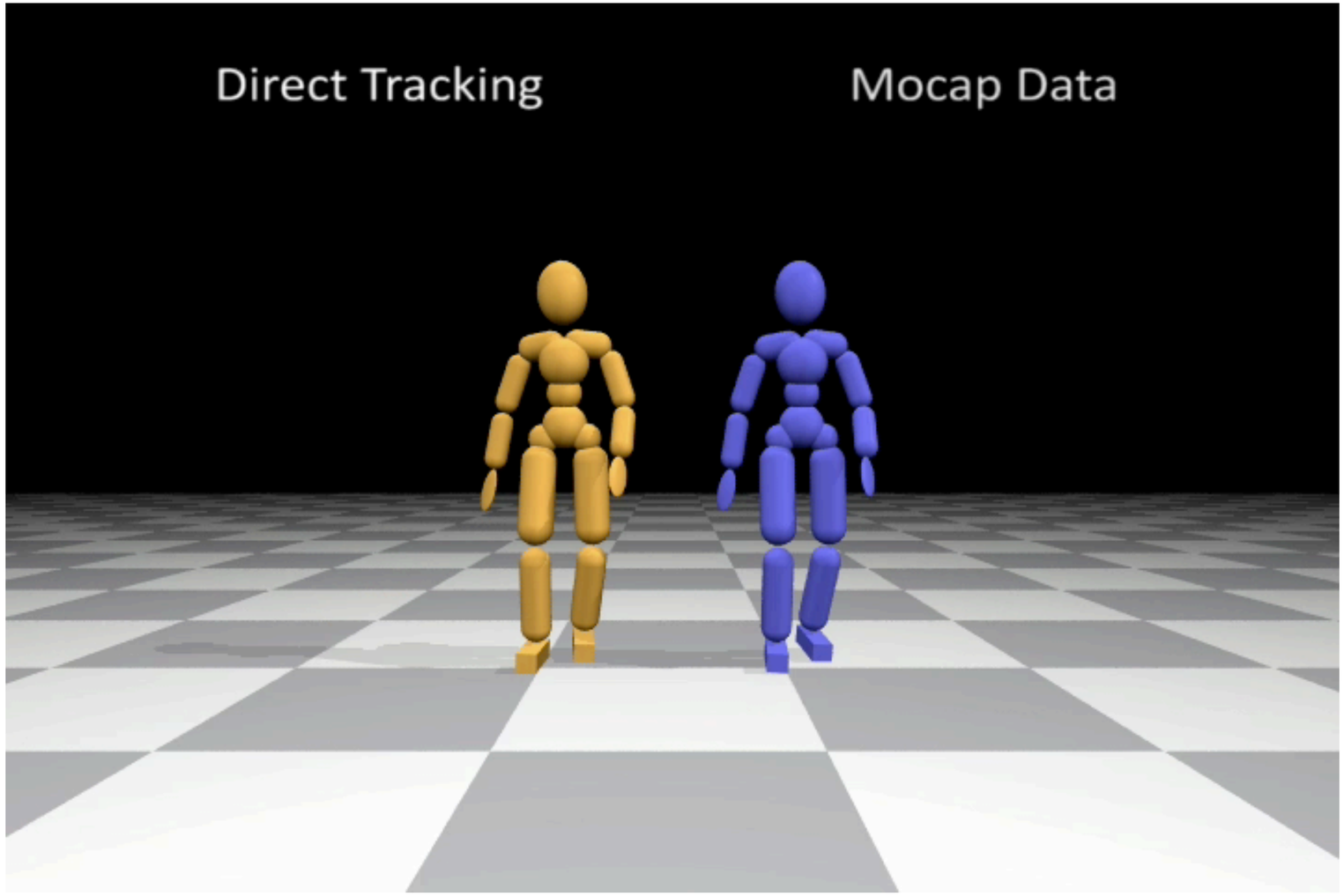
$\dot{\theta}$ = Angular velocity

k_p = Position control gain

k_v = Velocity control gain



Tracking actuated dofs



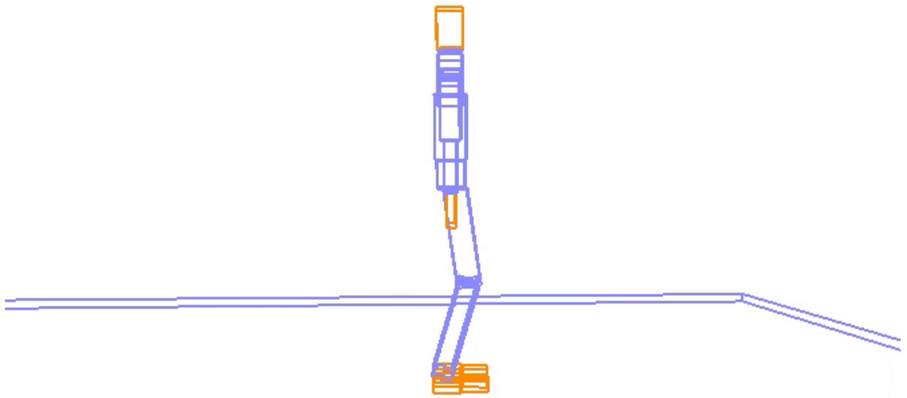
Static balance

- For static balance, projection of the mass point must be within the support polygon.
- Use hip strategy or ankle strategy to maintain balance.

projected center of mass

$$\tau = -k_p(\mathbf{c} - \bar{\mathbf{c}}) - k_d\dot{\mathbf{c}}$$

torque applied at ankle or hip



Balanced locomotion

- Simple finite state machine
- Torso and swing hip control
- Balance feedback

SIMBICON: Simple Biped Locomotion Control

KangKang Yin Kevin Loken Michiel van de Panne*

University of British Columbia

Figure 1: *Real-time physics-based character simulation with our framework. (a) A single controller for a planar biped responds to unanticipated changes in terrain. (b) A walk controller reconstructed from motion capture data responds to a 350N, 0.2s diagonal push to the torso.*

Abstract

Physics-based simulation and control of biped locomotion is difficult because bipeds are unstable, underactuated, high-dimensional dynamical systems. We develop a simple control strategy that can be used to generate a large variety of gaits and styles in real-time, including walking in all directions (forwards, backwards, sideways, turning), running, skipping, and hopping. Controllers can be authored using a small number of parameters, or their construction can be informed by motion capture data. The controllers are applied to 2D and 3D physically-simulated character models. Their robustness is demonstrated with respect to pushes in all directions, unexpected steps and slopes, and unexpected variations in kinematic and dynamic parameters. Direct transitions between controllers are demonstrated as well as parameterized control of changes in direction and speed. Feedback-error learning is applied to learn predictive torque models, which allows for the low-gain control that typifies many natural motions as well as producing smoother simulated motion.

1 Introduction

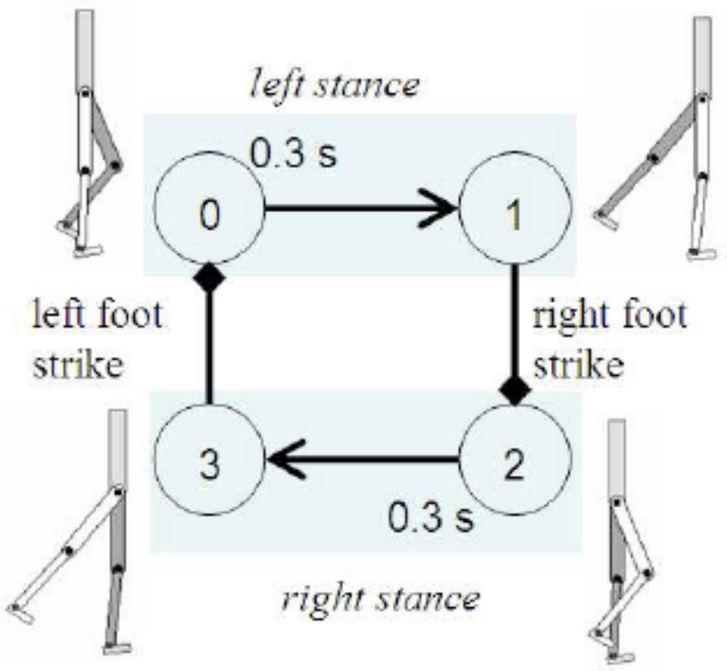
Locomotion is at the heart of many motions, both real and animated. Animated motion is most often created directly by animators using keyframing, or by capturing and then processing human motion. However, these approaches fail to scale to the very large set of possible motions that might arise in a realistic environment. For example, there are an infinite number of ways in which two developing controllers to drive forward dynamics simulations. Controller-based approaches have the advantage that they can synthesize motion at interactive rates and produce motion by using feedback strategies that continually adapt to the real-world as necessary.

The control of walking and other biped locomotion gaits has been of long-standing interest to the robotics and computer graphics communities. It is a challenging problem for many reasons. Walking bipeds are unstable and underactuated and their control involves high-dimensional states and high-dimensional actions. Locomotion involves joint limit constraints, torque-limit constraints, contact constraints, and contact impacts. Locomotion may have a number of contradictory goals, including robustness and energy usage. Lastly, while data-driven approaches have been very successful at generating kinematic models of locomotion, it is unclear whether such strategies can be successfully adopted to learn control strategies for dynamic simulations.

There exists a vast literature related to the control of bipedal walking, much of it in the robotics, control, and biomechanics communities. Common approaches to locomotion control include: (a) the use of passive walking as a starting point for the design of active walkers; (b) the use of "zero moment point" control; (c) using a fixed control architecture and applying parameter search to find the parameter settings that yield successful walking gaits; and (d) developing feedback laws based upon insights into balance and locomotion. Our proposed framework buikds on the last of these approaches.

Simple state machine

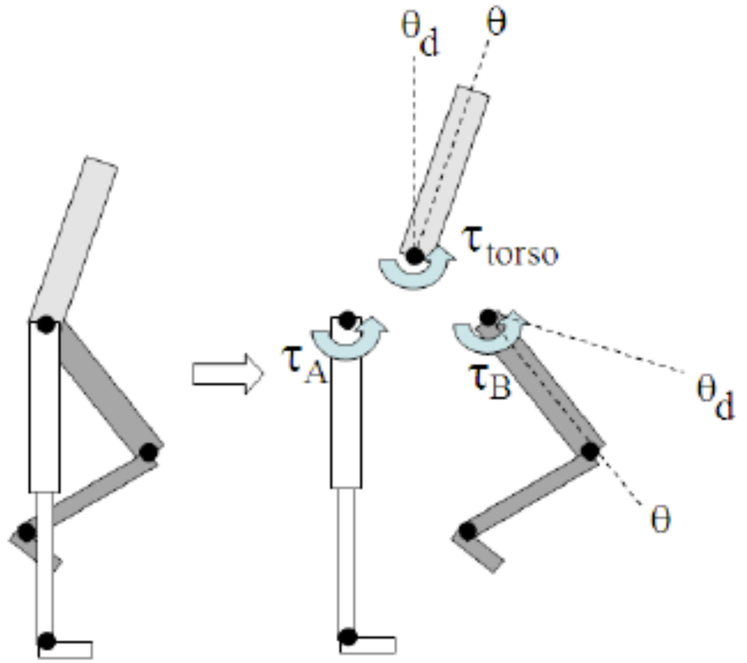
- Each state consists of a pose representing target angles.
- All independent joints are controlled by PD servos.
- Transitions between states occur after fixed duration of time or foot contact events.



Torso and swing-hip control

- Both torso and swing-hip have target angles expressed in world coordinates.
- The stance-hip torque is determined by the torque of torso and swing hip:

$$\tau_A = -\tau_{torso} - \tau_B$$

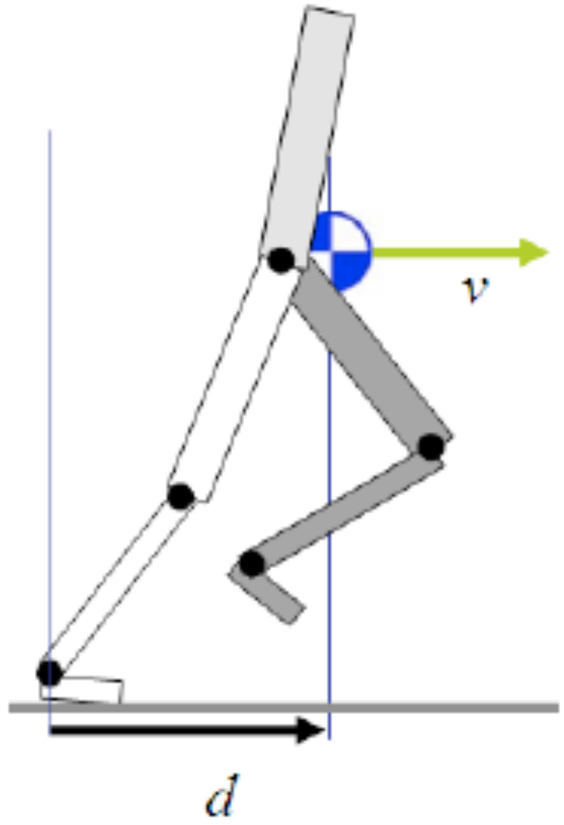


Balance feedback

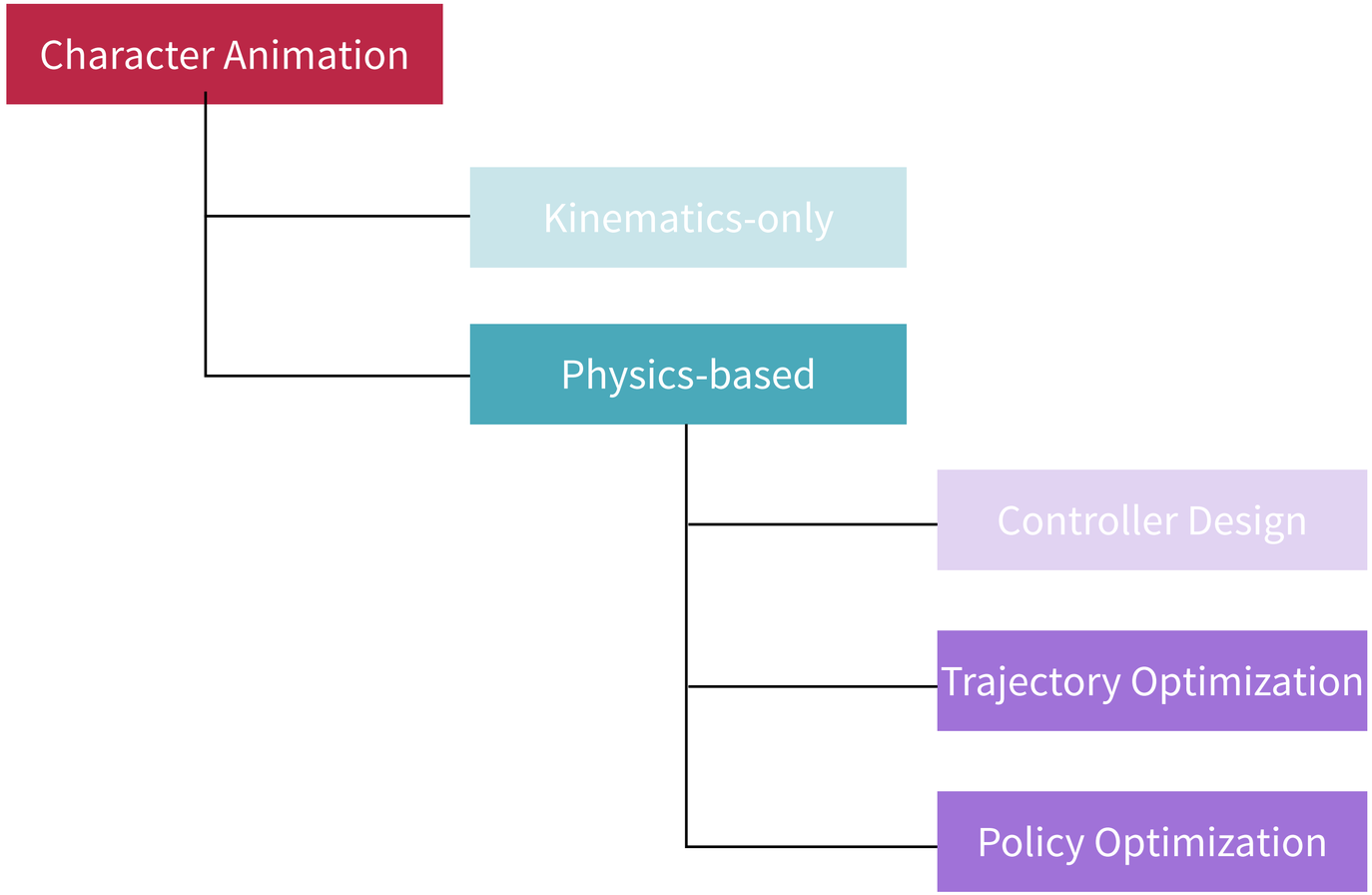
- Swing-hip target angle is continuously modified based on the center of mass

$$\theta_d = \theta_{d0} + c_d d + c_v v$$

- This allows the character to change the future point of support

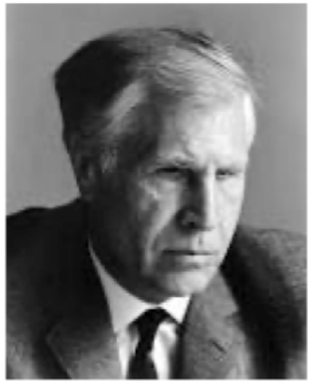


SIMBICON

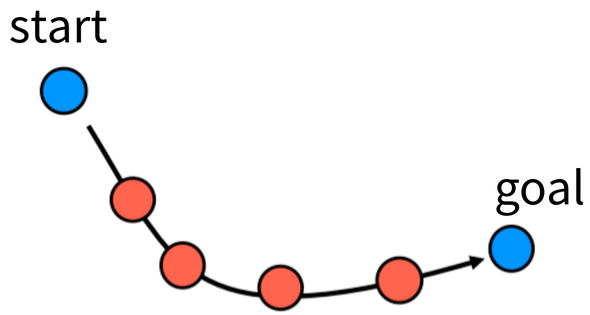

Character Animation
Kinematics-only
Physics-based
Controller Design
Trajectory Optimization
Policy Optimization

Classic optimal control theory

Pontryagin's Maximum Principle (PMP)



Lev Pontryagin
Moschow State University



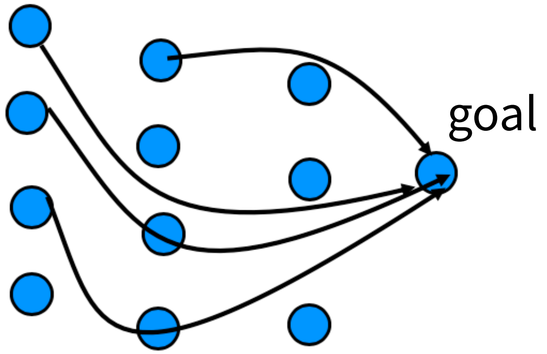
Open-loop control: given an initial condition $\mathbf{x}_0$, find sequence of actions $\mathbf{u}_0, \cdots \mathbf{u}_T$ to reach the goal.

Solve **Boundary Value Problem** for $\mathbf{u}(t)$

Easier problem because it only solves $\mathbf{u}(t)$ for one $\mathbf{x}_0$ and executes it blindly.

But perturbation can push the system off track!

Dynamic Programming and
Bellman Principle of Optimality



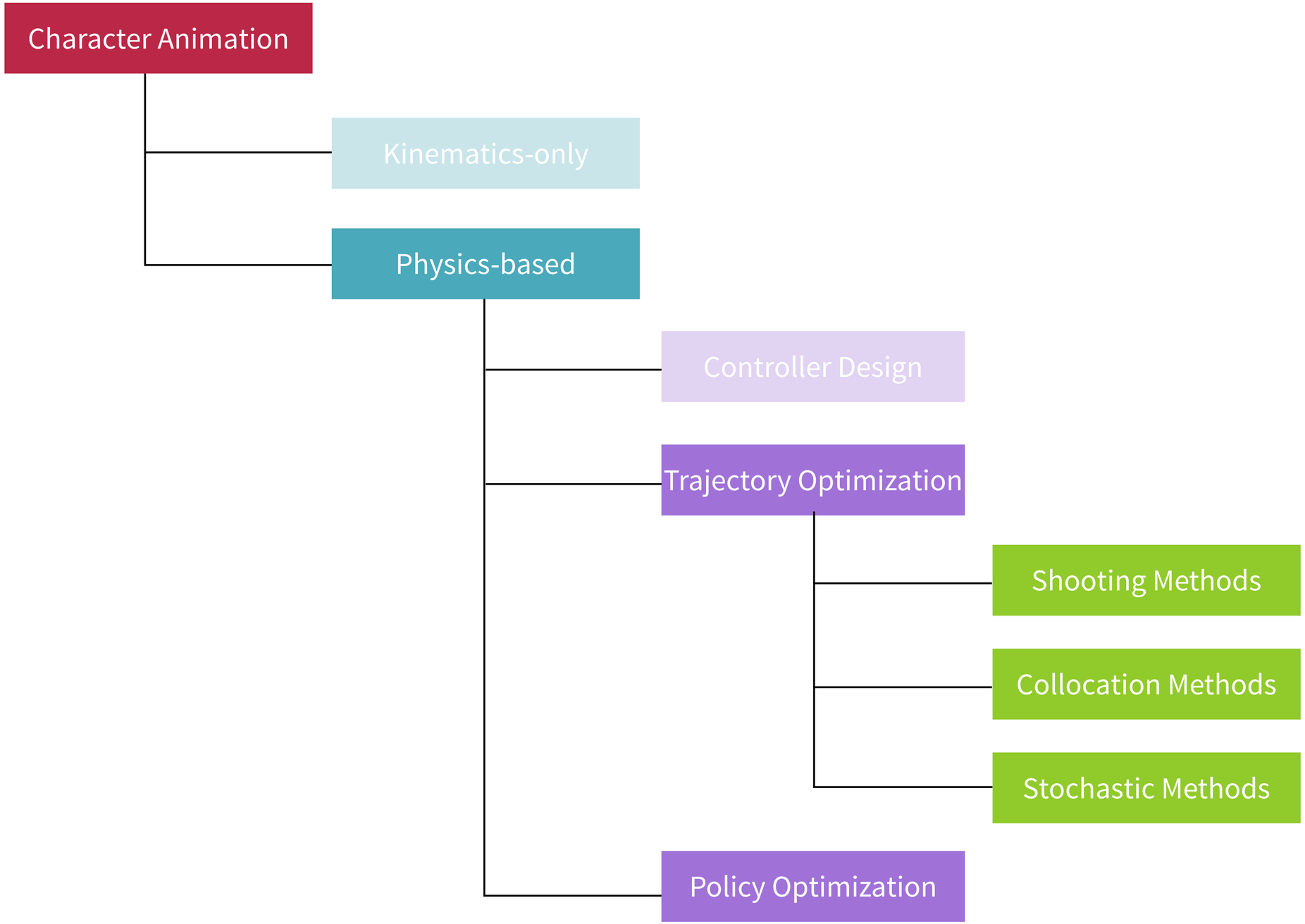
Richard Bellman
USC

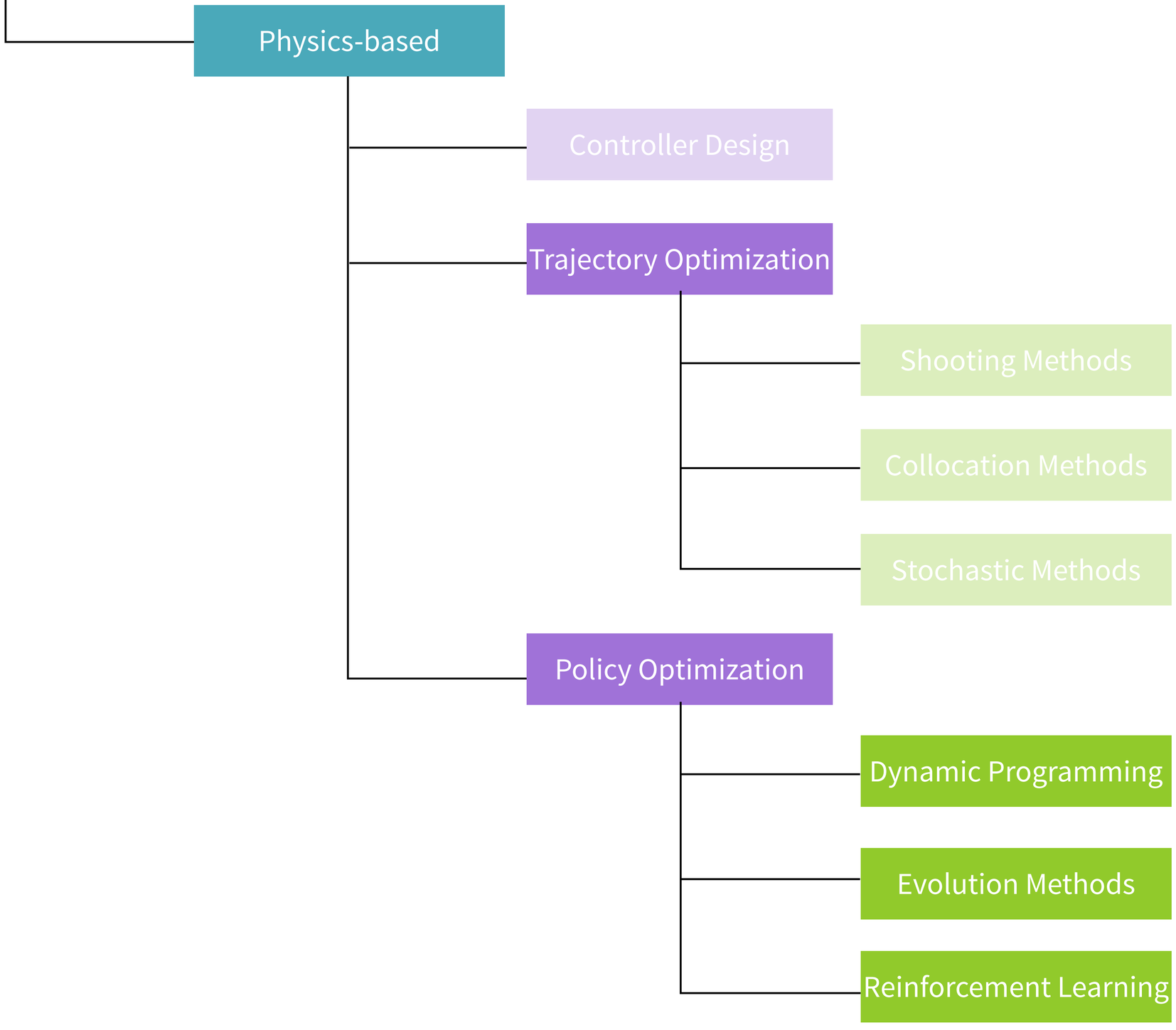
Closed-loop control: for ANY state $\mathbf{x}$ and time t, find the corresponding action $\mathbf{u}$.

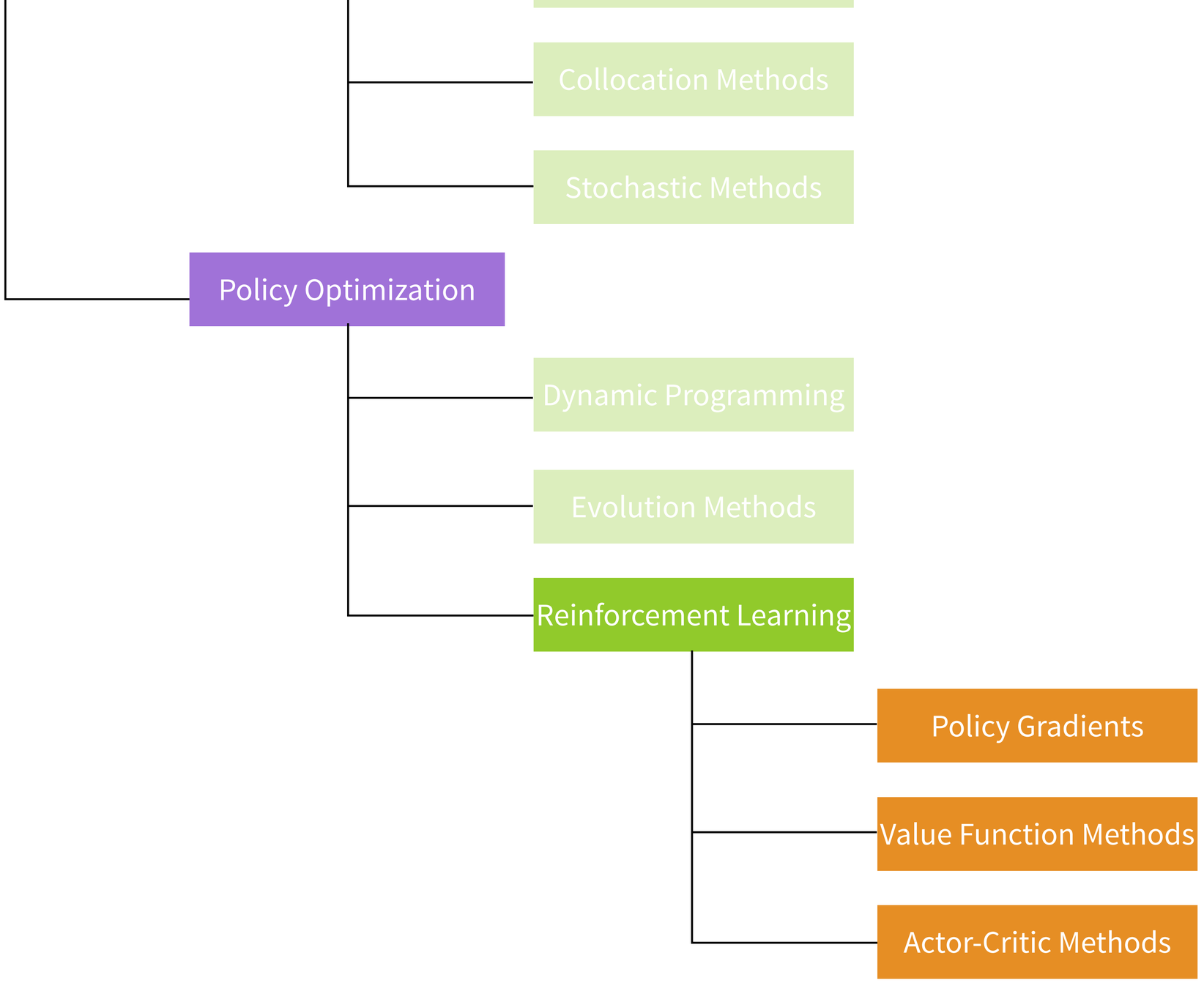
Solve **Partial Differential Equation** for $\mathbf{u}(\mathbf{x}, t)$

Harder problem because it's like solving PMP for all possible initial conditions $\mathbf{x}_0$.

But it is more robust to perturbations!

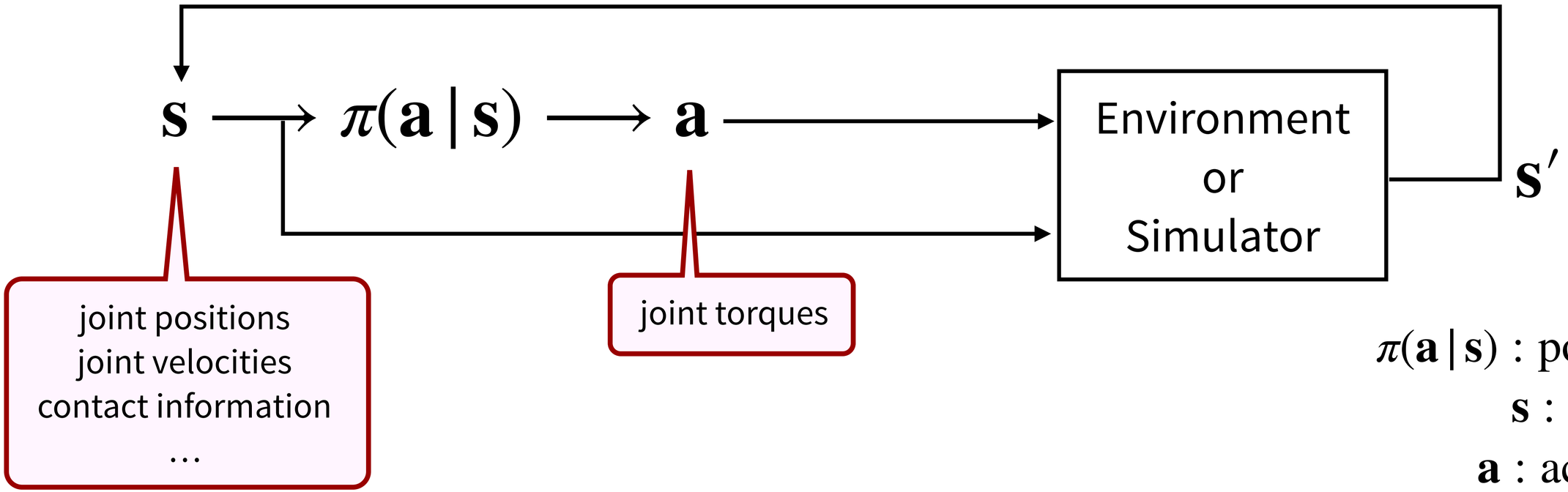

Character Animation
Kinematics-only
Physics-based
Controller Design
Trajectory Optimization
Shooting Methods
Collocation Methods
Stochastic Methods
Policy Optimization


Physics-based
Controller Design
Trajectory Optimization
Shooting Methods
Collocation Methods
Stochastic Methods
Policy Optimization
Dynamic Programming
Evolution Methods
Reinforcement Learning


Collocation Methods
Stochastic Methods
Policy Optimization
Dynamic Programming
Evolution Methods
Reinforcement Learning
Policy Gradients
Value Function Methods
Actor-Critic Methods

Definitions

How can an agent take actions in an environment so as to maximize long-term reward?



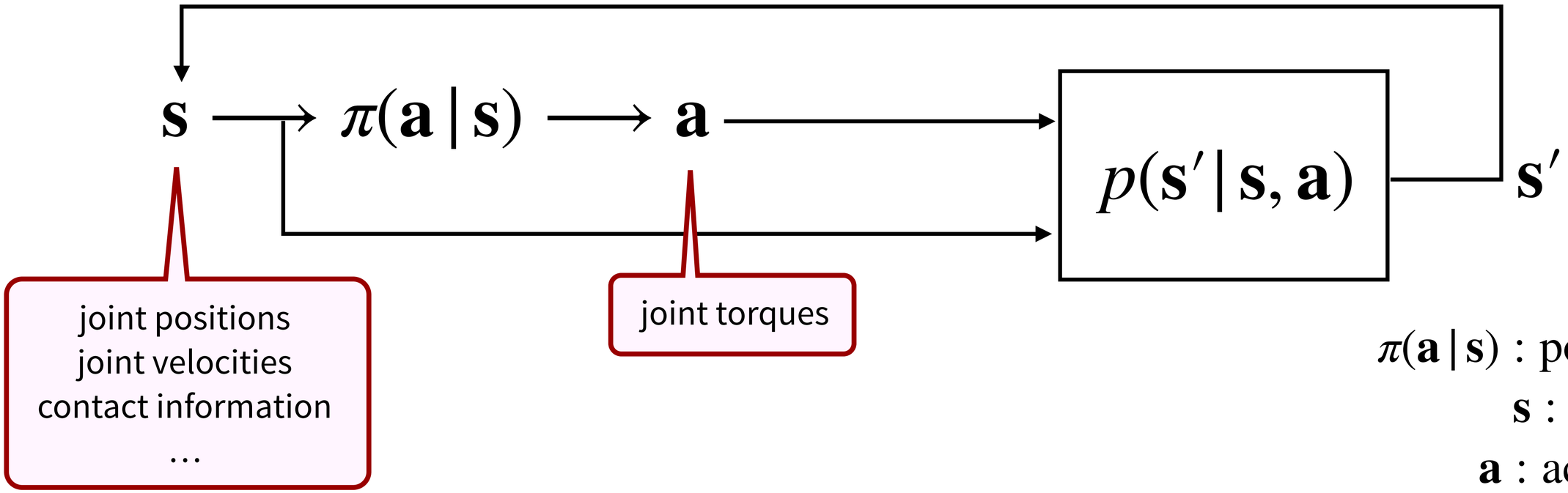
$\pi(\mathbf{a}\,|\,\mathbf{s})$: policy

$\mathbf{s}$: state

$\mathbf{a}$: action

Definitions

How can an agent take actions in an environment so as to maximize long-term reward?



$\pi(\mathbf{a}\,|\,\mathbf{s})$: policy

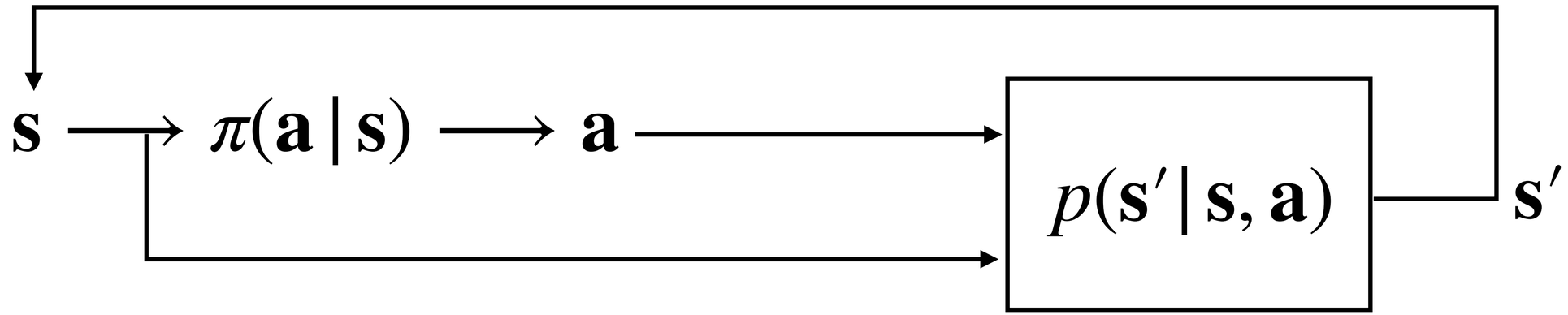
$\mathbf{s}$: state

$\mathbf{a}$: action

$p(\mathbf{s}'\,|\,\mathbf{s},\mathbf{a})$: dynamic transition

Definitions

How can an agent take actions in an environment so as to maximize long-term reward?



$$\sum_{t=0} r(\mathbf{s}_t, \mathbf{a}_t)$$

move in target speed
keep balance
use little energy
…

$\pi(\mathbf{a}\,|\,\mathbf{s})$: policy
$\mathbf{s}$: state
$\mathbf{a}$: action
$p(\mathbf{s}'\,|\,\mathbf{s},\mathbf{a})$: dynamic transition
$r(\mathbf{s},\mathbf{a})$: reward function

Markov decision process (MDP)

A discrete time stochastic control process that provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

Problem: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p, r\}$

state space

action space

Solution: $\pi(\mathbf{a}\,|\,\mathbf{s})$

$\pi(\mathbf{a}\,|\,\mathbf{s})$: policy

$\mathbf{s}$: state

$\mathbf{a}$: action

$p(\mathbf{s}'\,|\,\mathbf{s}, \mathbf{a})$: dynamic transition

$r(\mathbf{s}, \mathbf{a})$: reward function

Partially observable MDP

A POMDP models an agent decision process in which it is assumed that the system dynamics are determined by an MDP, but the agent cannot directly observe the underlying state

Problem: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, p, p_e, r\}$

observation space

emission probability

Solution: $\pi(\mathbf{a} \mid \mathbf{o})$

$\pi(\mathbf{a} \mid \mathbf{s})$: policy

$\mathbf{s}$: state

$\mathbf{a}$: action

$p_e(\mathbf{o} \mid \mathbf{s})$: emission probability

$p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$: dynamic transition

$r(\mathbf{s}, \mathbf{a})$: reward function

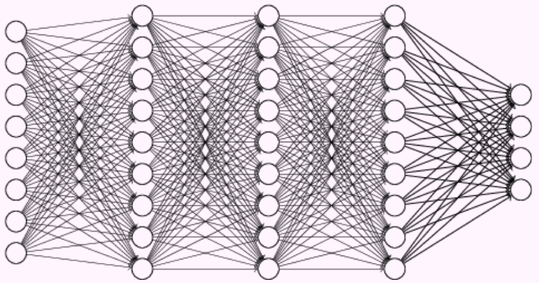
How to solve an MDP?

- **We need to find that "policy".**
  - **How to represent a p**

$$\pi_\theta(\mathbf{a} \mid \mathbf{s})$$

polynomials
radial basis functions
Gaussian processes
…

or



  - **How to define the "g**

expected long-term reward: $E_{\tau \sim (p, \pi_\theta)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$

a trajectory (rollout): $\tau = \{\mathbf{s}_1, \mathbf{a}_1, \cdots \mathbf{s}_T, \mathbf{a}_T\}$

Quiz

- **What is the probability of a rollout under a given dynamic transition and a given policy, assuming the initial state probability is $p(\mathbf{s}_1)$?**

$\pi(\mathbf{a}|\mathbf{s})$: policy

$p(\mathbf{s}'|\mathbf{s},\mathbf{a})$: dynamic transition

$p_\theta(\boldsymbol{\tau}) = ?$

Quiz

- **What is the probability of a single state-action pair under a given dynamic transition and a given policy, assuming the initial state probability is p(s1)?**

$\pi(\mathbf{a}\,|\,\mathbf{s})$: policy

$p(\mathbf{s}'\,|\,\mathbf{s},\mathbf{a})$: dynamic transition

$p_\theta(\mathbf{s}_t,\mathbf{a}_t) = ?$

$$p_\theta(\mathbf{s}_t,\mathbf{a}_t) = \int_{\mathbf{s}_1}\int_{\mathbf{a}_1}\cdots\int_{\mathbf{a}_{t-1}} p(\mathbf{s}_1)\pi(\mathbf{a}_1\,|\,\mathbf{s}_1)p(\mathbf{s}_2\,|\,\mathbf{s}_1,\mathbf{a}_1)\cdots p(\mathbf{s}_t\,|\,\mathbf{s}_{t-1},\mathbf{a}_{t-1})\pi(\mathbf{a}_t\,|\,\mathbf{s}_t)d\mathbf{a}_{t-1}\cdots d\mathbf{s}_1$$

Maximal expected return

Recall expected long-term reward: $E_{\boldsymbol{\tau}\sim(p,\pi_\theta)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$

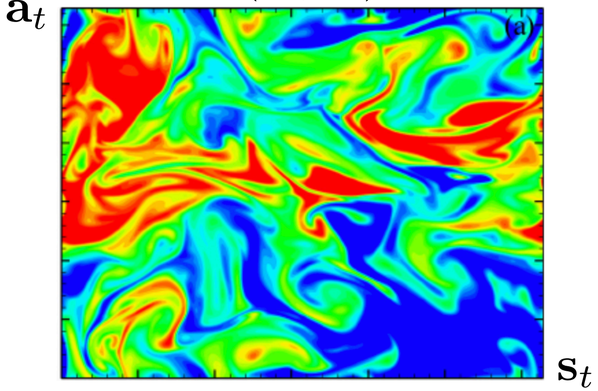
$$\boldsymbol{\theta}^* = \arg\max_\theta E_{\boldsymbol{\tau}\sim(p,\pi_\theta)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$\boldsymbol{\theta}^* = \arg\max_\theta \sum_t E_{(\mathbf{s}_t,\mathbf{a}_t)\sim p_\theta(\mathbf{s}_t,\mathbf{a}_t)}\left[r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

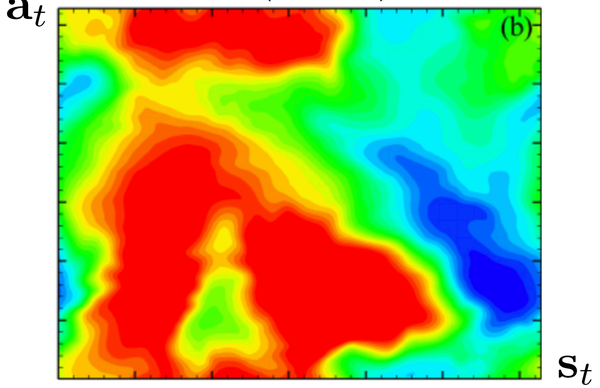
marginal distribution of one time step in the rollout

sum of reward for each state-action pair, weighted by the probability of the state-action pair

$r(\mathbf{s}_t, \mathbf{a}_t)$



$p_\theta(\mathbf{s}_t, \mathbf{a}_t)$



$p_\theta(\mathbf{s}_t, \mathbf{a}_t)$ depends on both

$p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$: dynamic transition

$\pi(\mathbf{a}|\mathbf{s})$: policy

RL algorithms

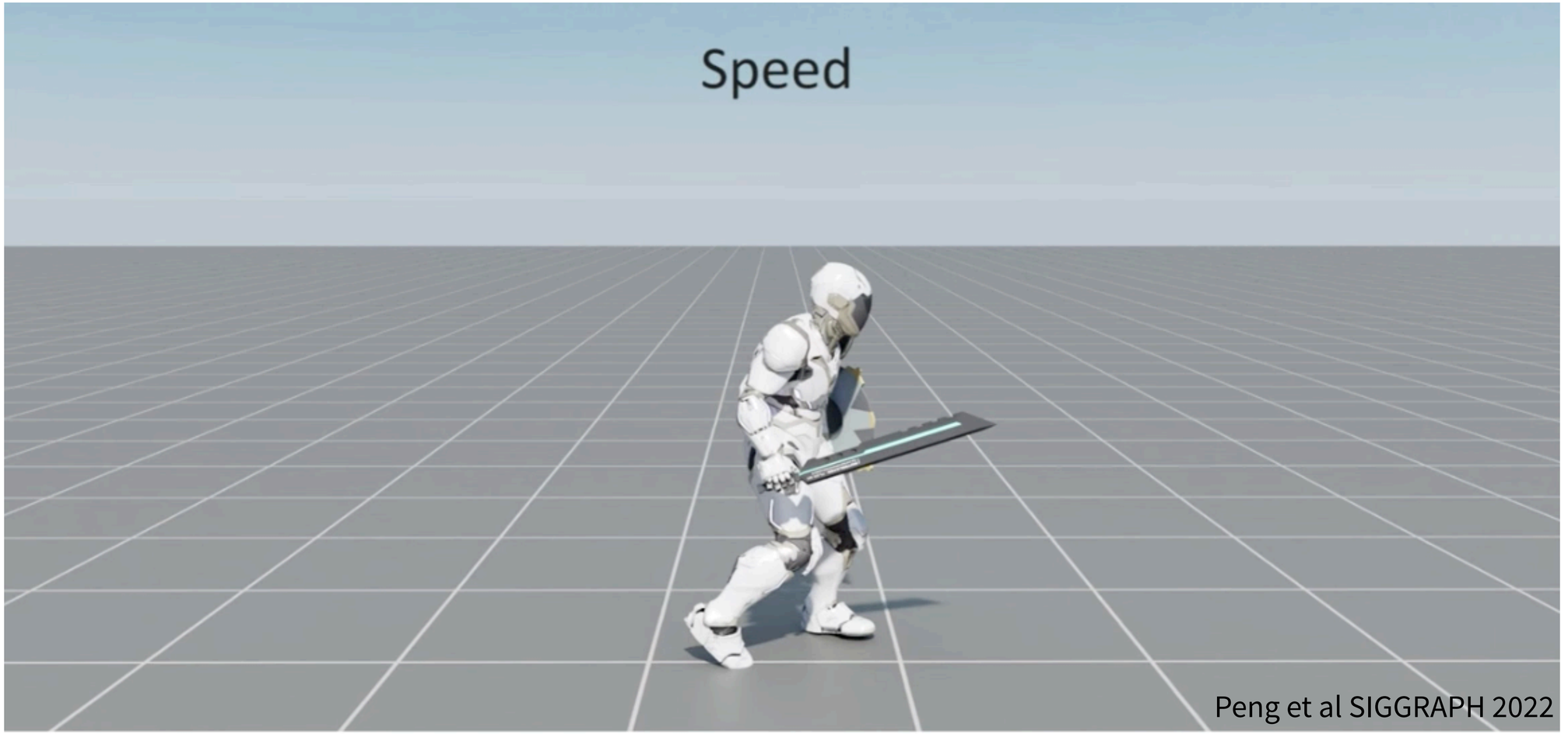
- Policy Gradients approach
  - Learn a policy by optimizing the long-term reward through policy gradients
  - REINFORCE, TRPO, PPO…
- Value function approach
  - Learn a value function or Q function without explicitly learning a policy
  - Value iteration, Q-learning, DQN…
- Actor-critic approach
  - Learn a value function alongside with a policy.
  - DDPG, AC3, TRPO-GAE (arguably), SAC…

RL algorithms

- Value function approach
  - Learn a value function or Q function without explicitly learning a policy
  - Value iteration, Q-learning, DQN…
- Policy gradients approach
  - Learn a policy by optimizing the long-term reward through policy gradients
  - REINFORCE, TRPO, PPO…
- Actor-critic approach
  - Learn a value function alongside with a policy.
  - DDPG, AC3, TRPO-GAE (arguably), SAC…

Properties of RL algorithms

- On-policy or Off-policy?
  - On-policy requires samples generated from the current policy being optimized.
  - Off-policy allows to use samples generated from other policies, including those generated by previous learning iterations.
- Stochastic or deterministic?
  - Policy and transition function can be stochastic or deterministic.
  - Policy gradients methods typically require policy to be stochastic.
- Continuous or discrete?
  - State and action spaces can be continuous or discrete.

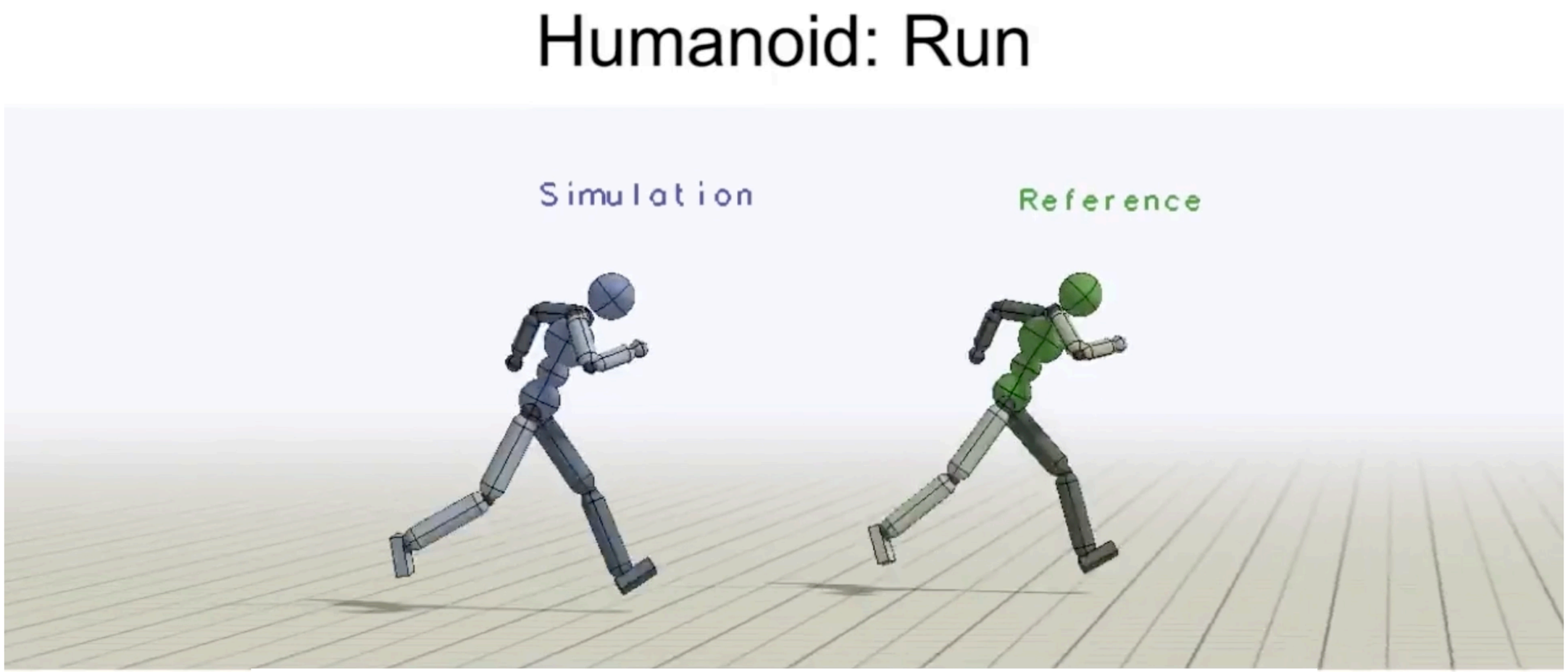


First, we have a target speed task,

DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills

XUE BIN PENG, PIETER ABBEEL, SERGEY LEVINE, MICHIEL VAN DE PANNE

Problem Statement



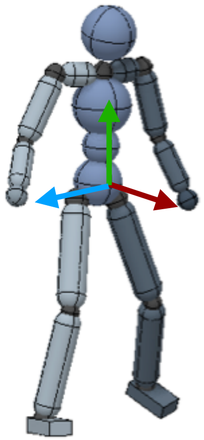
Learn a policy that controls a character's movement to track a given reference motion in physics simulation

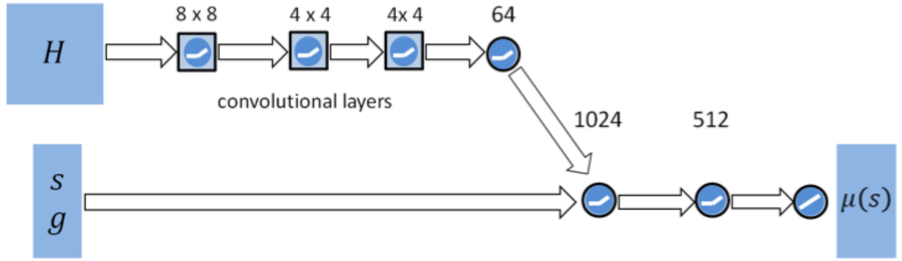
Approach

- Formulate a MDP and learn the policy using PPO
- Design a reward function for imitation and task completion
- Introduce two new training tips

MDP Formulation

- State ($\mathbf{s}$):
  - Cartesian link position, orientation (quaternion) in character's frame
  - Linear and angular velocities of each link
  - Phase variable from 0 to 1
  - (Optional) Surrounding height field
- Goal ($\mathbf{g}$): desired walking direction, hitting target, etc
- Action ($\mathbf{a}$): Target joint angles for PDs
- Policy representation: $\pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{g}) = N(\mu_\theta(\mathbf{s}, \mathbf{g}), \Sigma)$





Reward Function

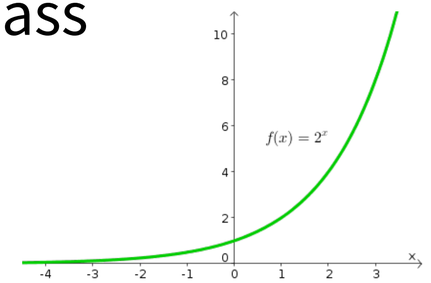
- Imitation reward: match the the trajectories generated by the policy to the reference trajectories in following aspects:
  - Pose reward: match joint orientation
  - Velocity reward: match joint velocity
  - End-effector reward: match hands and feet positions
  - Center-of-mass reward: match center-of-mass
- Task reward: encourage task-specific goals

How to compute difference between two quaternions?

1. Convert to rotation matrices $R(q_1)$ and $R(q_2)$ and compute the difference: $\Delta R = R(q_1)R(q_2)^{-1}$
2. Convert ΔR to axis angle and return the angle as the difference

$$r_t^p = \exp\left[-2\left(\sum_j \|\hat{q}_t^j \ominus q_t^j\|^2\right)\right]$$

$$r_t^v = \exp\left[-0.1\left(\sum_j \|\hat{\dot{q}}_t^j - \dot{q}_t^j\|^2\right)\right]$$



Training Tips

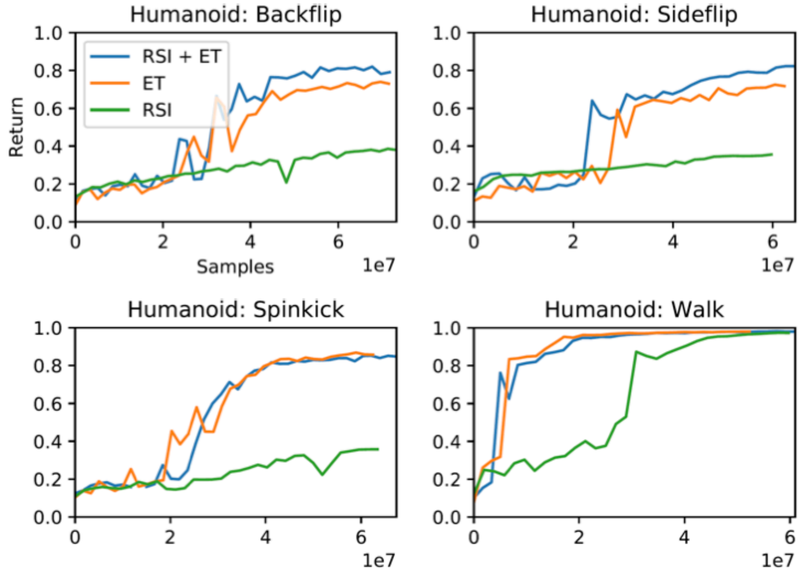
- Use PPO to train a policy and a value function
- Initial state distribution: Randomly sample an initial state from the reference trajectory
- Early termination: Terminate the rollout when the character’s torso or head makes contact with the ground.

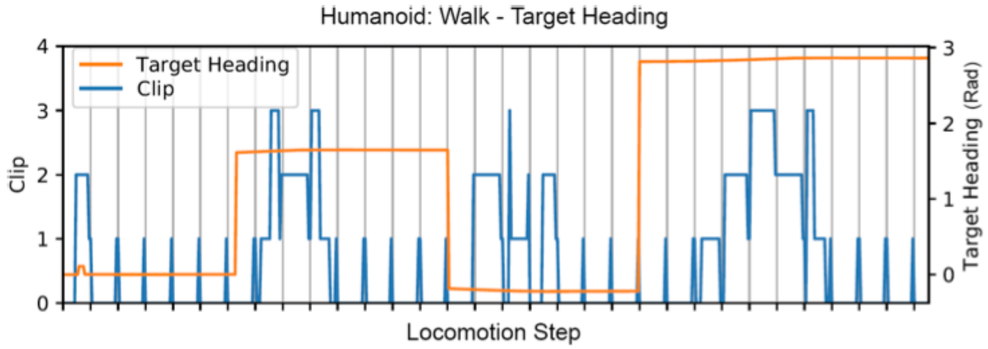
Evaluation

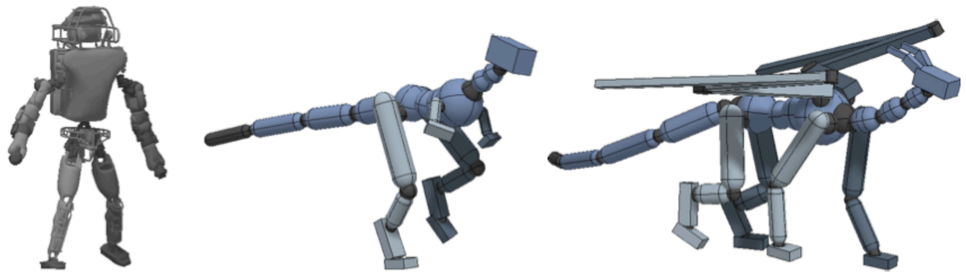
- Tasks
- Multi-skill integration
- Retargeting
- Ablations
- Robustness

Environment	$r^I + r^G$	r^I	r^G
Humanoid: Strike - Spinkick	99%	19%	55%
Humanoid: Baseball Pitch - Throw	75%	5%	93%

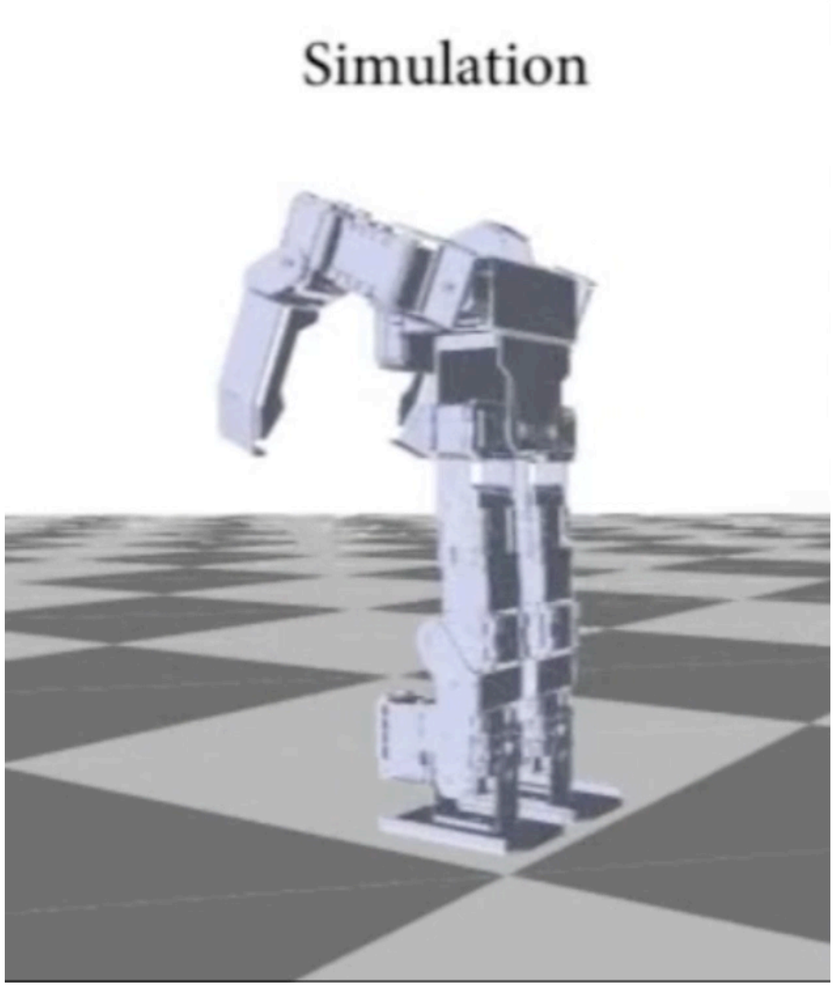
Skill	Forward (N)	Sideway (N)
Backflip	440	100
Cartwheel	200	470
Run	720	300
Spinkick	690	600
Walk	240	300



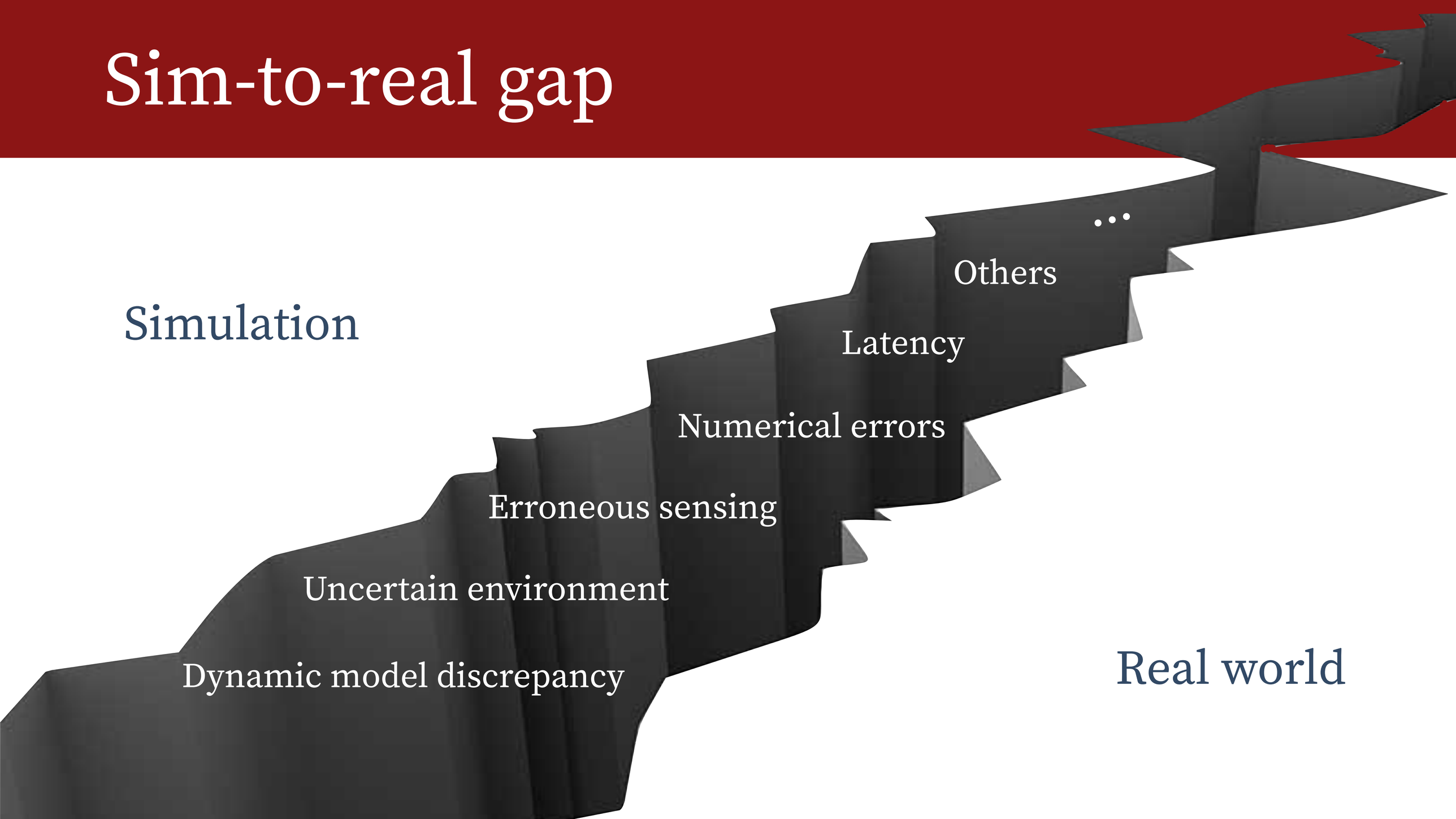




Teach robots to move like humans



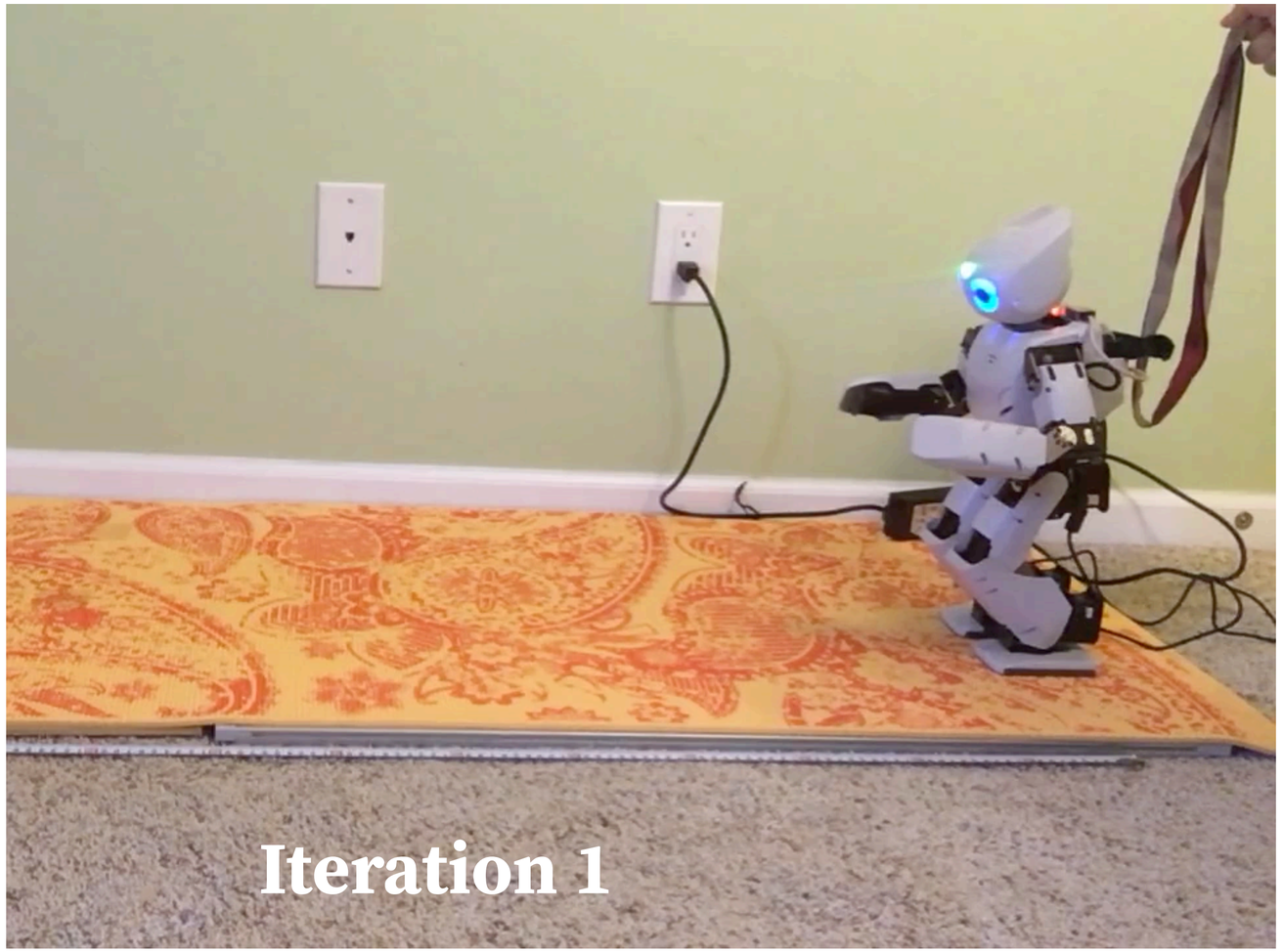
Sim-to-real gap



Transfer the policy from simulation to real world

- Careful system identification
- Domain randomization
- Adaptive policy

Biped walking

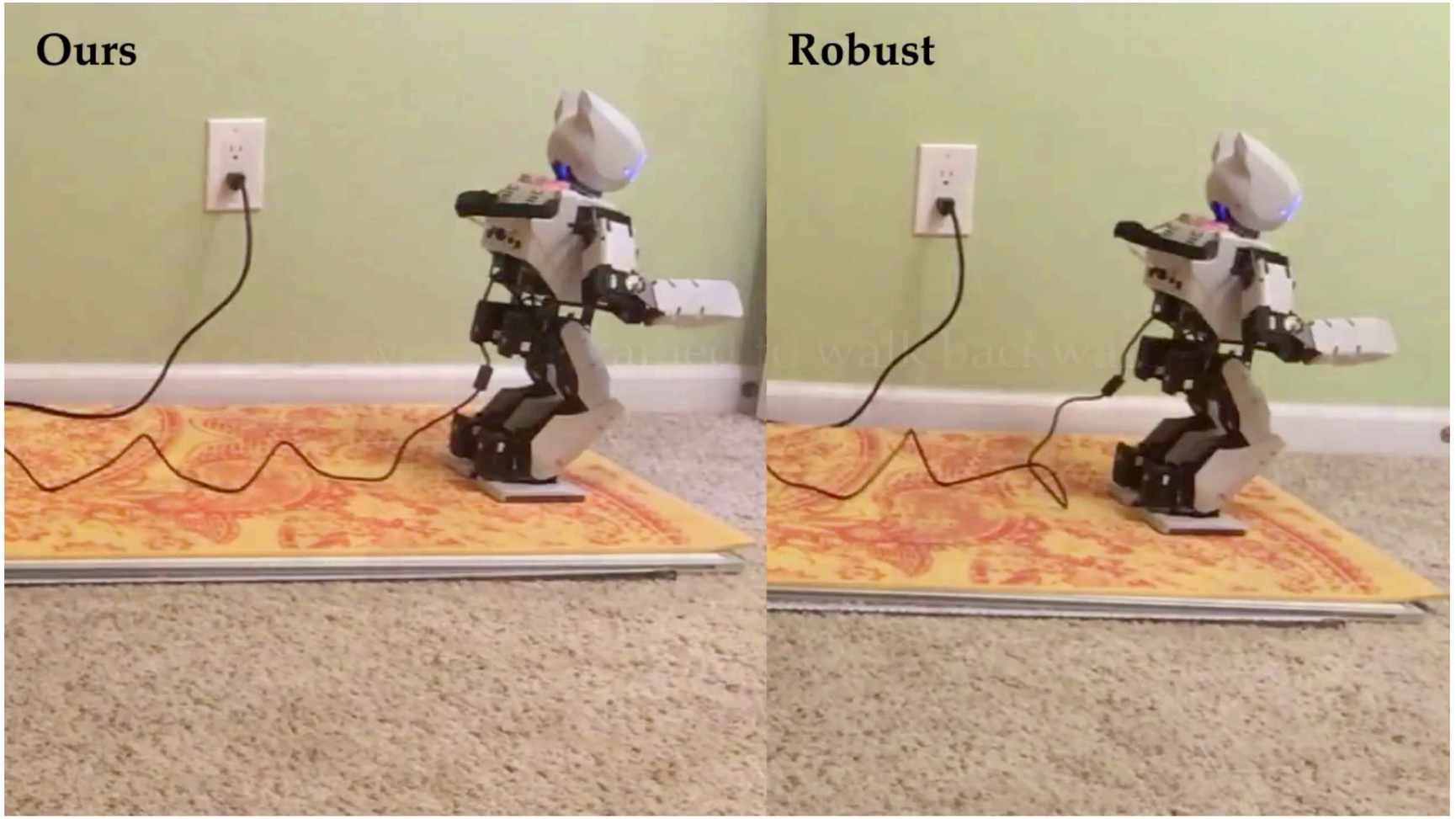


Yu et al, IROS 2019

Biped walking

Yu et al, IROS 2019

Walking backward



Yu et al, IROS 2019