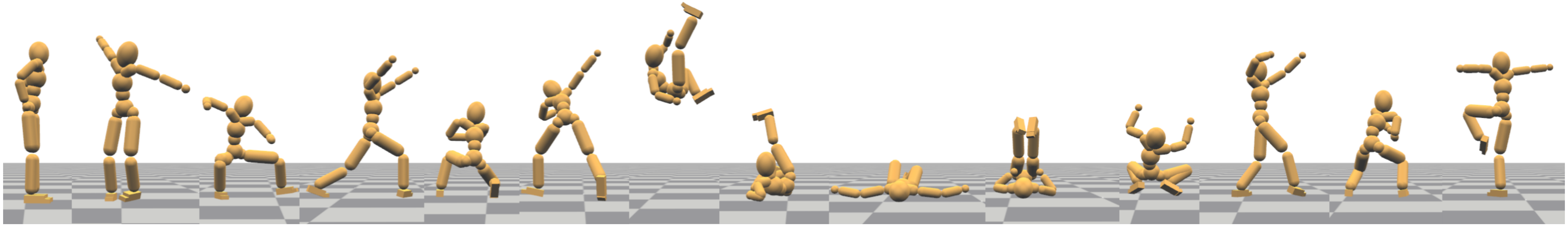
Learning Physics-based Tracking Control using Reinforcement Learning

Libin Liu (libin@deepmotion.com)

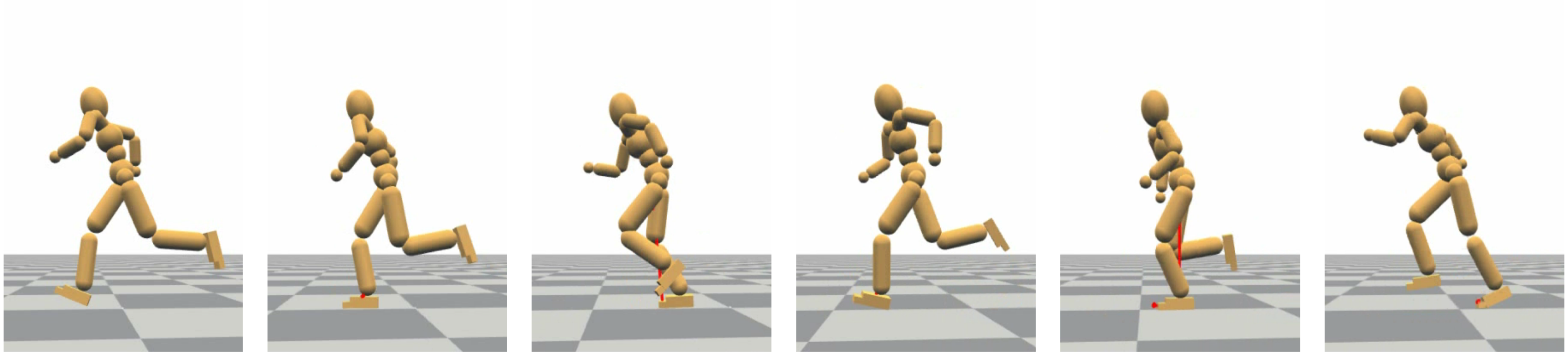
DeepMotion Inc.

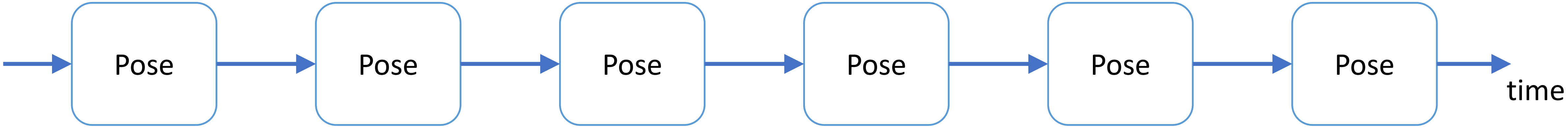


Outline

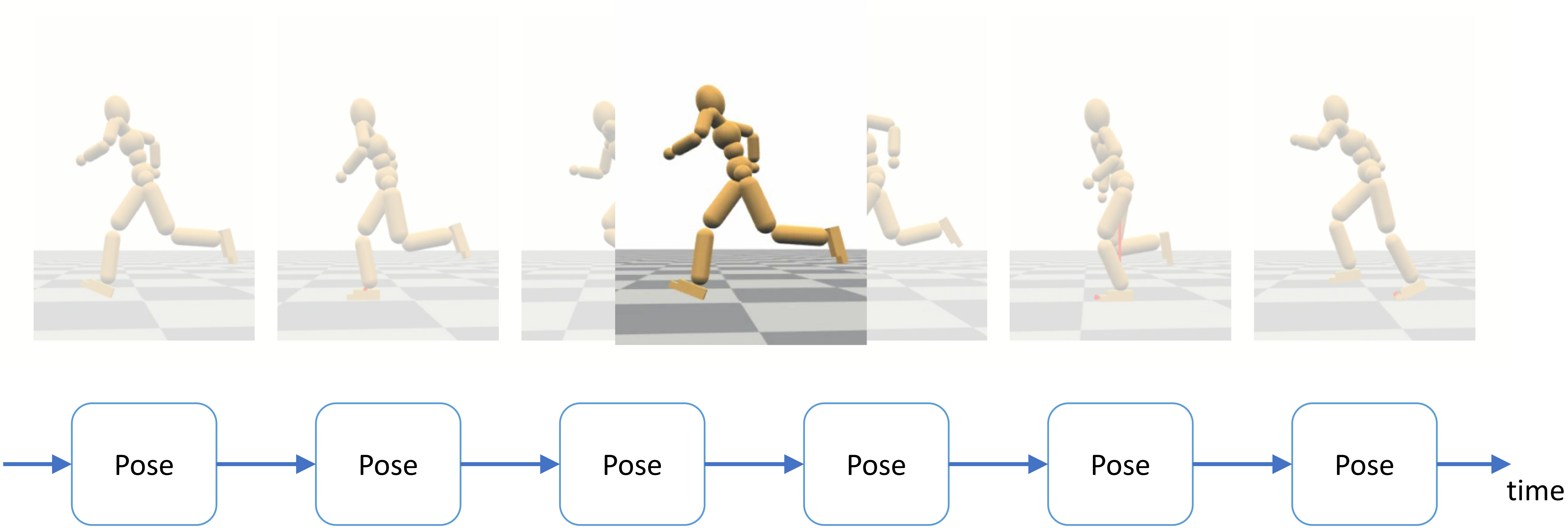
- Physics-based Character Animation
- Tracking control
  - Sampling-based motion control (SAMCON)
  - Linear feedback policy
- Reinforcement Learning
  - Reward-weight regression
  - Policy gradient & nonlinear policy
  - Scheduler

Character Animation

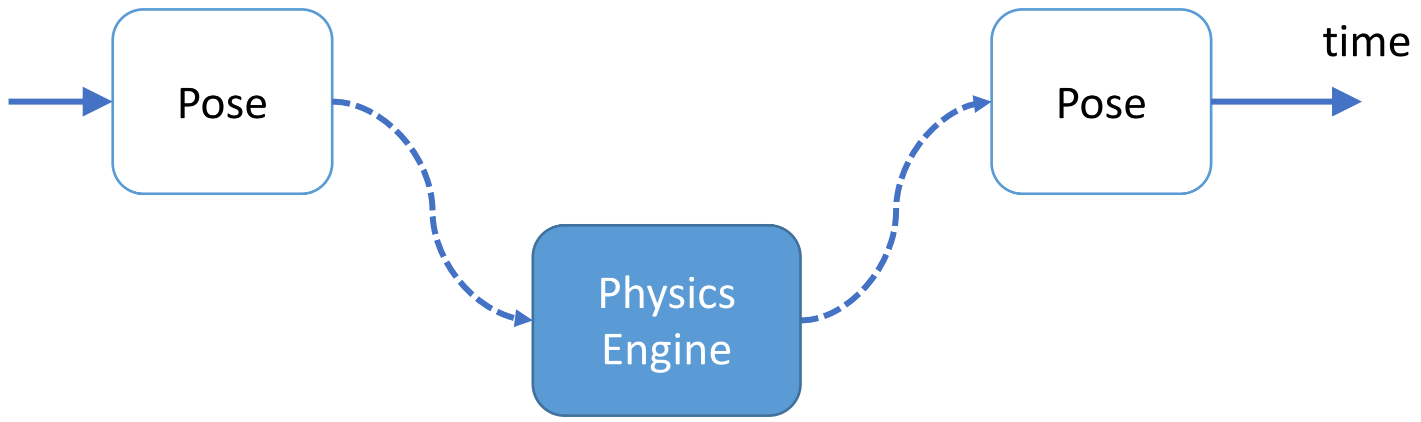




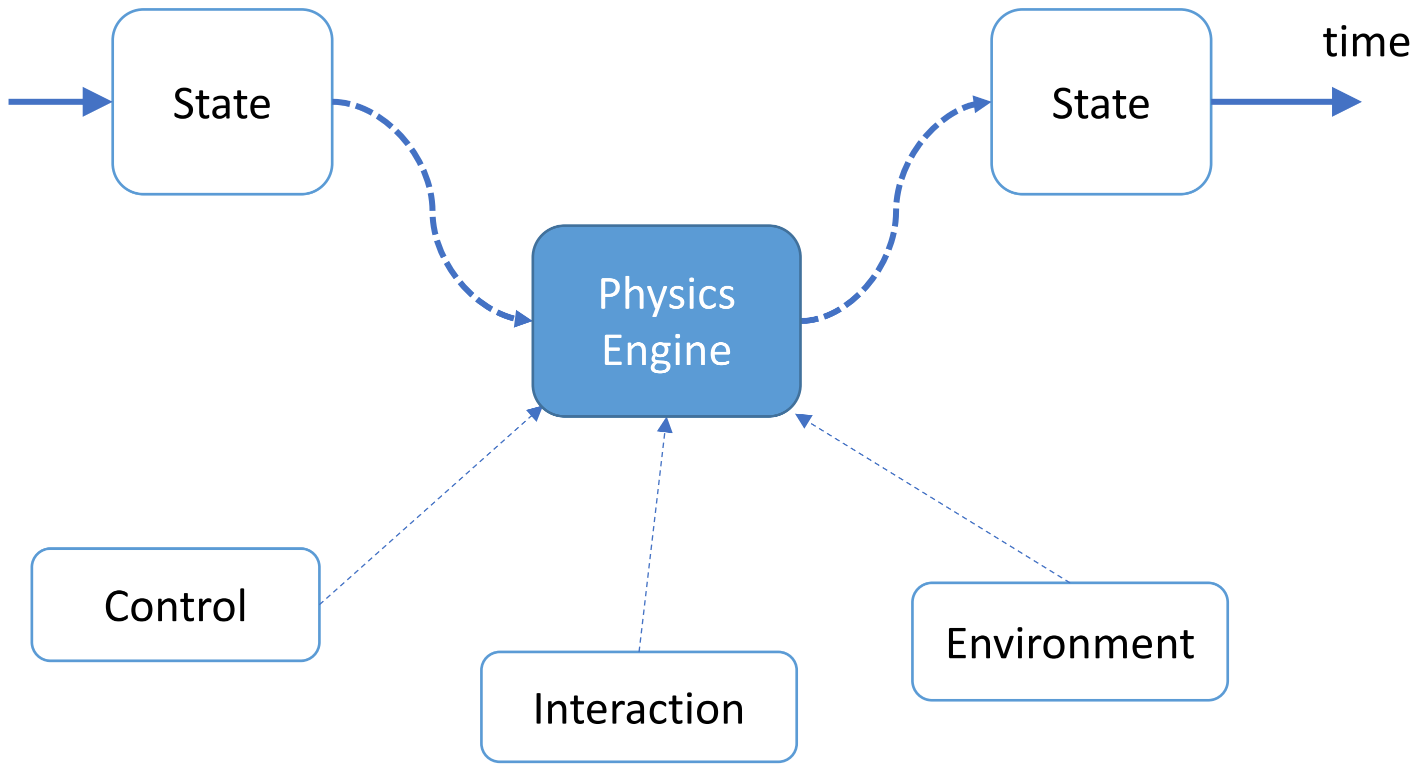
Character Animation



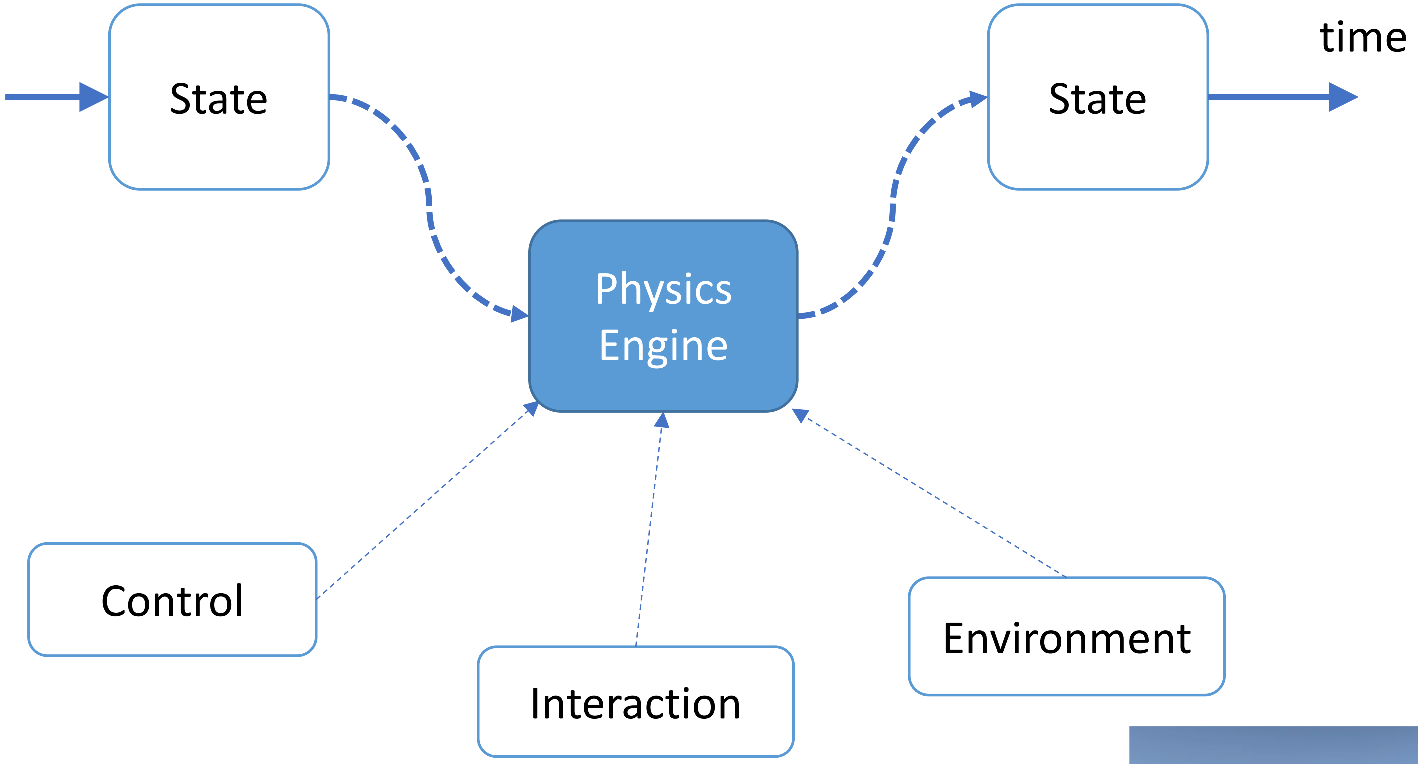
Physics-based Character Animation

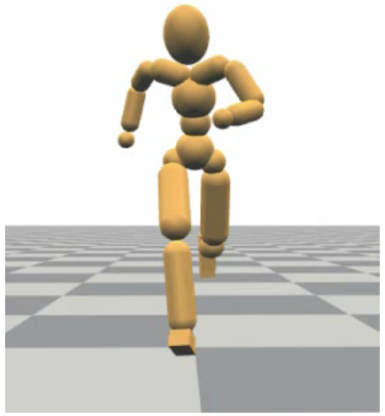


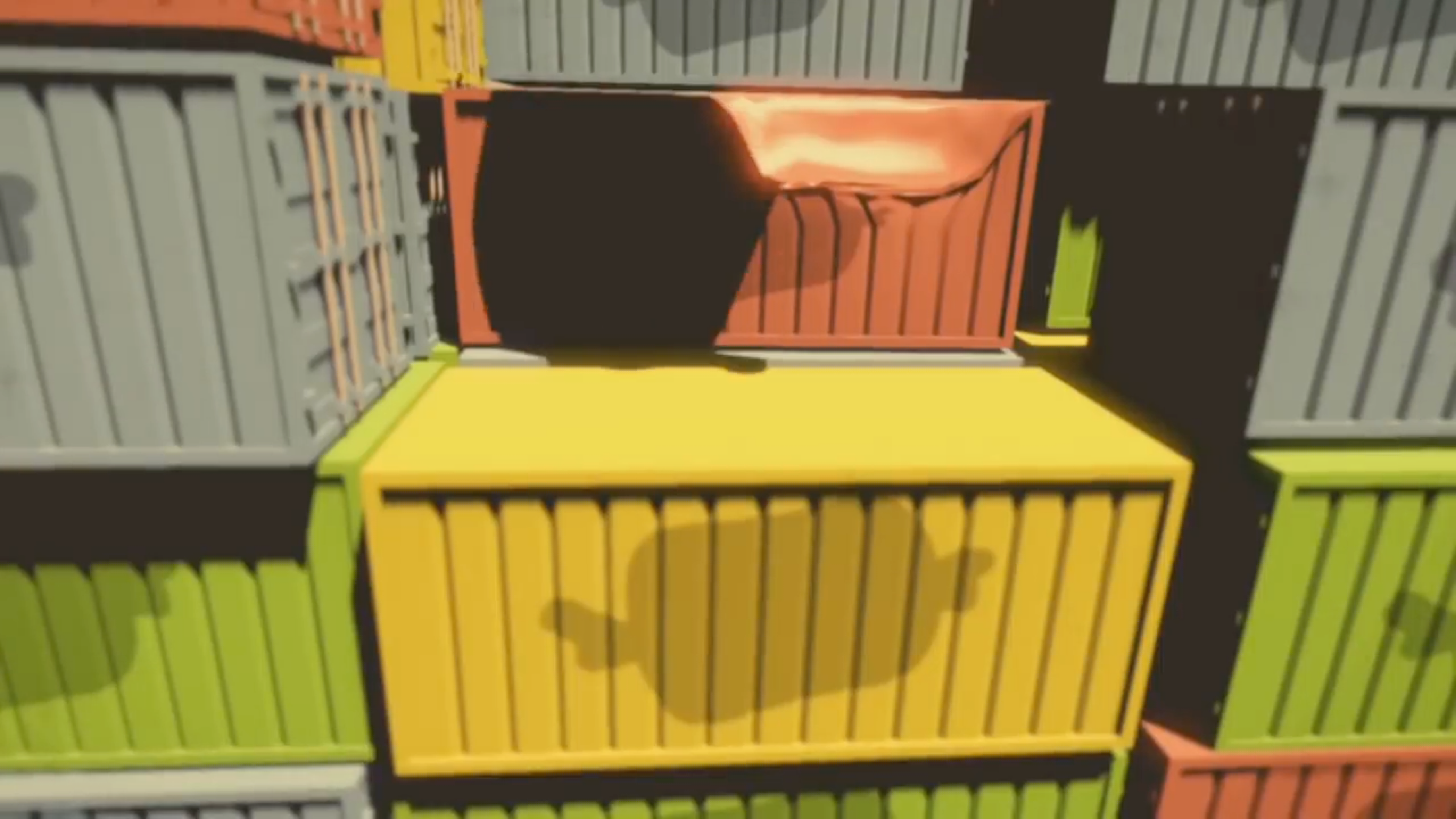
Physics-based Character Animation



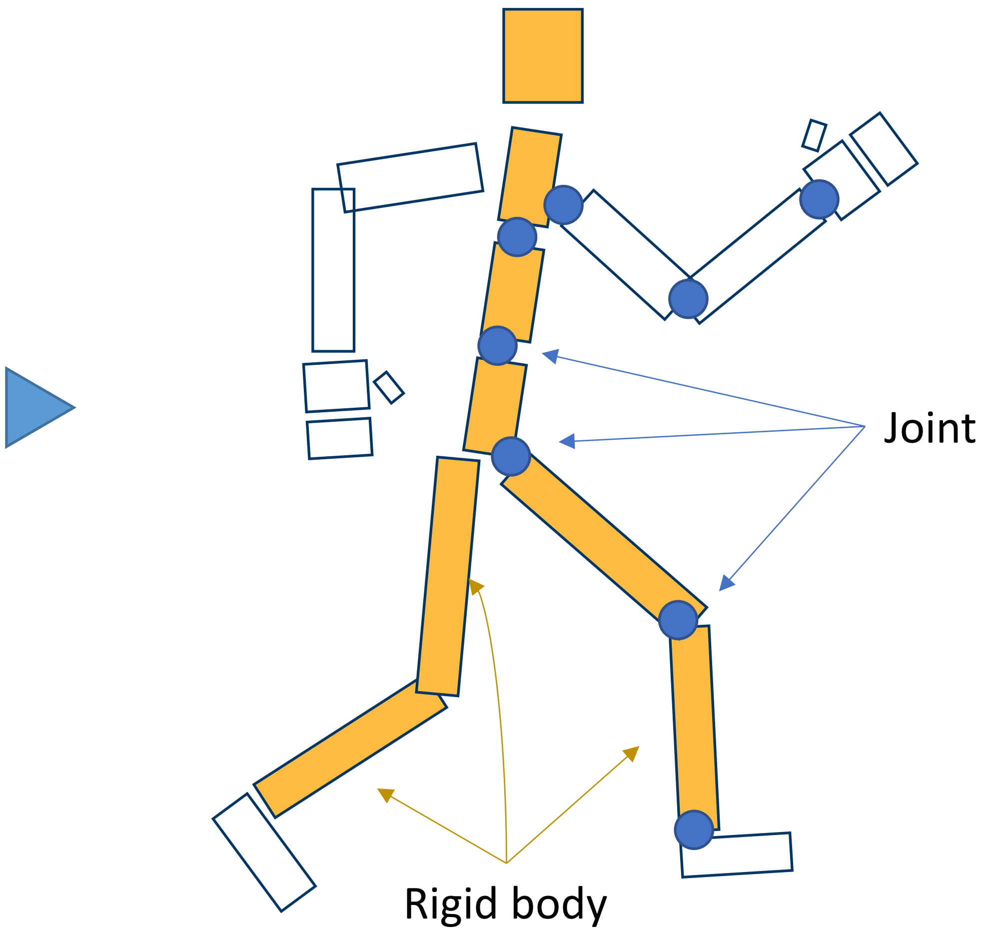
Physics-based Character Animation



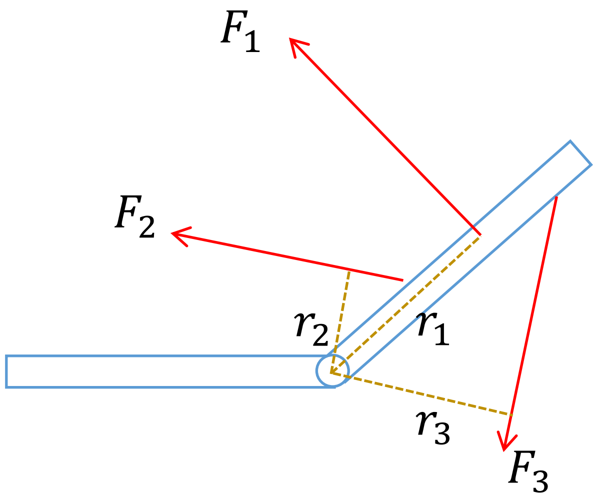


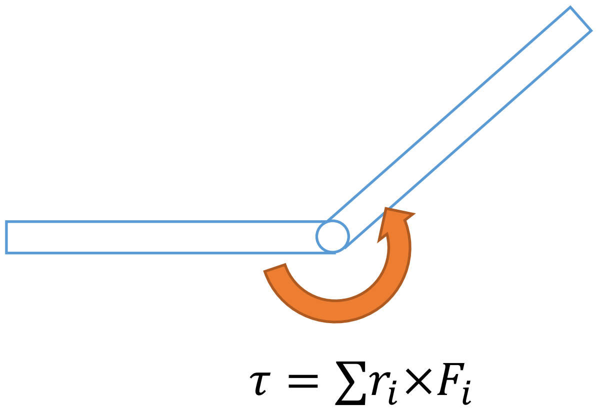


Skeleton Model

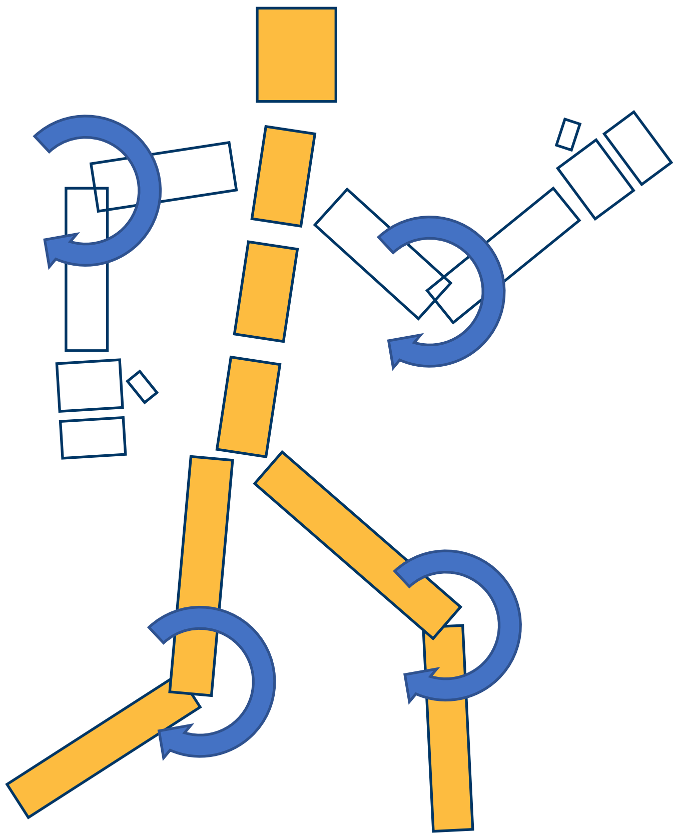


Force & Torque

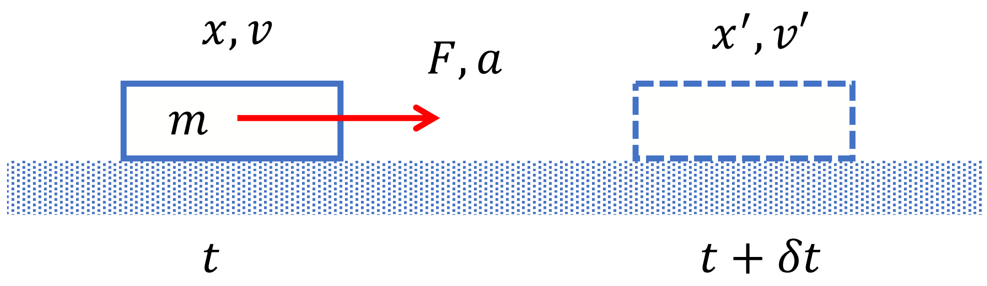




Joint Torques



Newton's Law



$$ma = F$$

$$v' \leftarrow v + a\delta t$$

$$x' \leftarrow x + v\delta t$$

↑ Numerical integration

Rigid Body Dynamics

$q, \dot{q}$

t

$$M(q)\ddot{q} + C(q, \dot{q}) = \tau + J^T \lambda$$

$$J\dot{q} \geq 0$$

$$\dot{q} \leftarrow \dot{q} + \ddot{q}\delta t$$

$$q \leftarrow q + \dot{q}\delta t$$

$q', \dot{q}'$

$t + \delta t$

* A good tutorial:
https://www.cc.gatech.edu/~karenliu/RTQL8_files/dynamics.pdf

Physics Engine

$q, \dot{q}$

t

Collision detection

Constraints

Equations of Motion

Solver

Numerical Integration

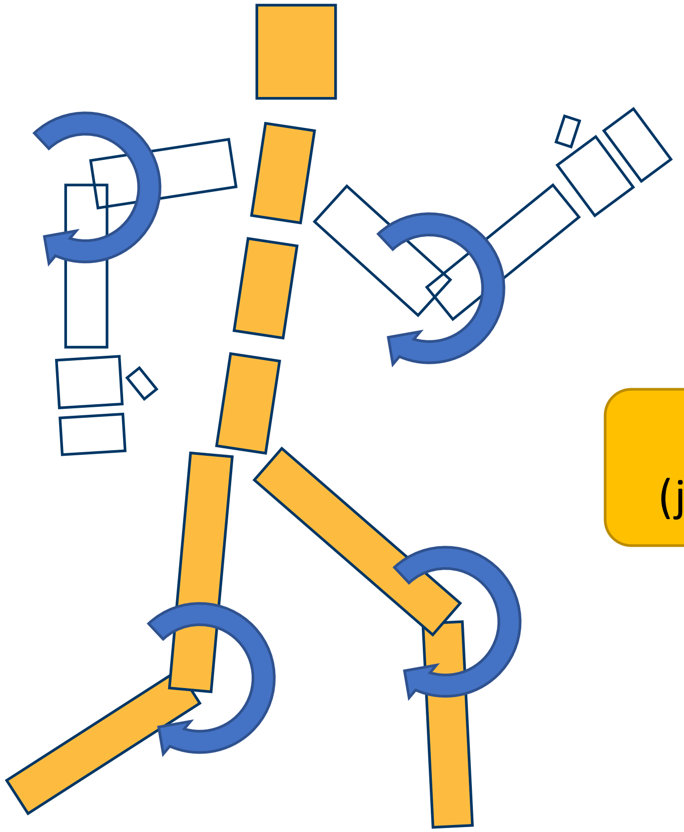
Physics Engine

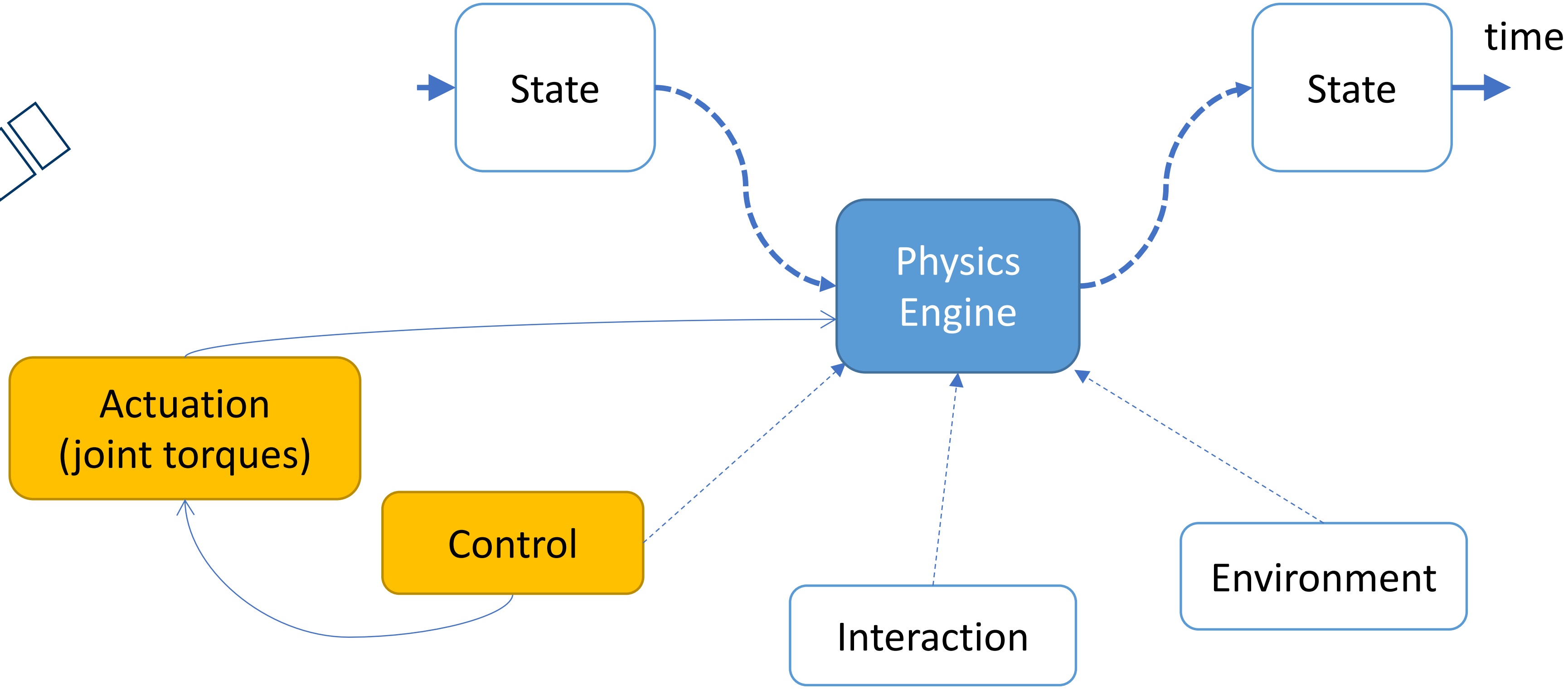
ODE, Bullet, PhysX, Dart, Mujoco, Havoc…

$q', \dot{q}'$

$t + \delta t$

Designing Controller

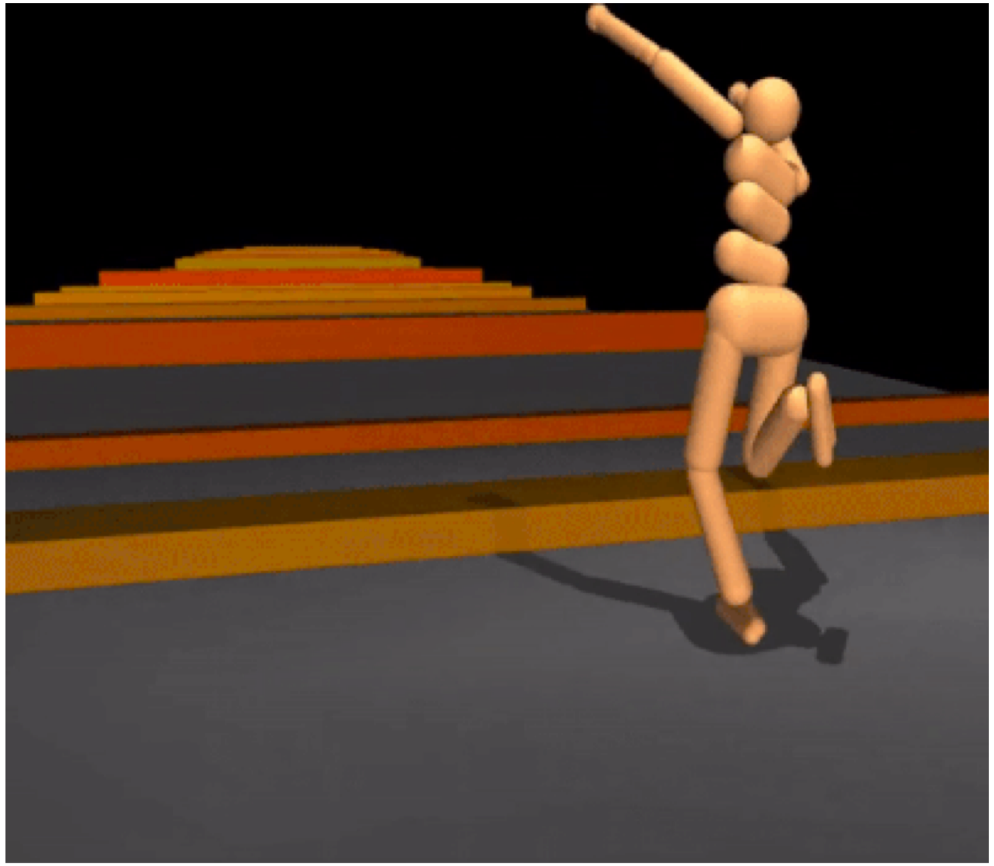


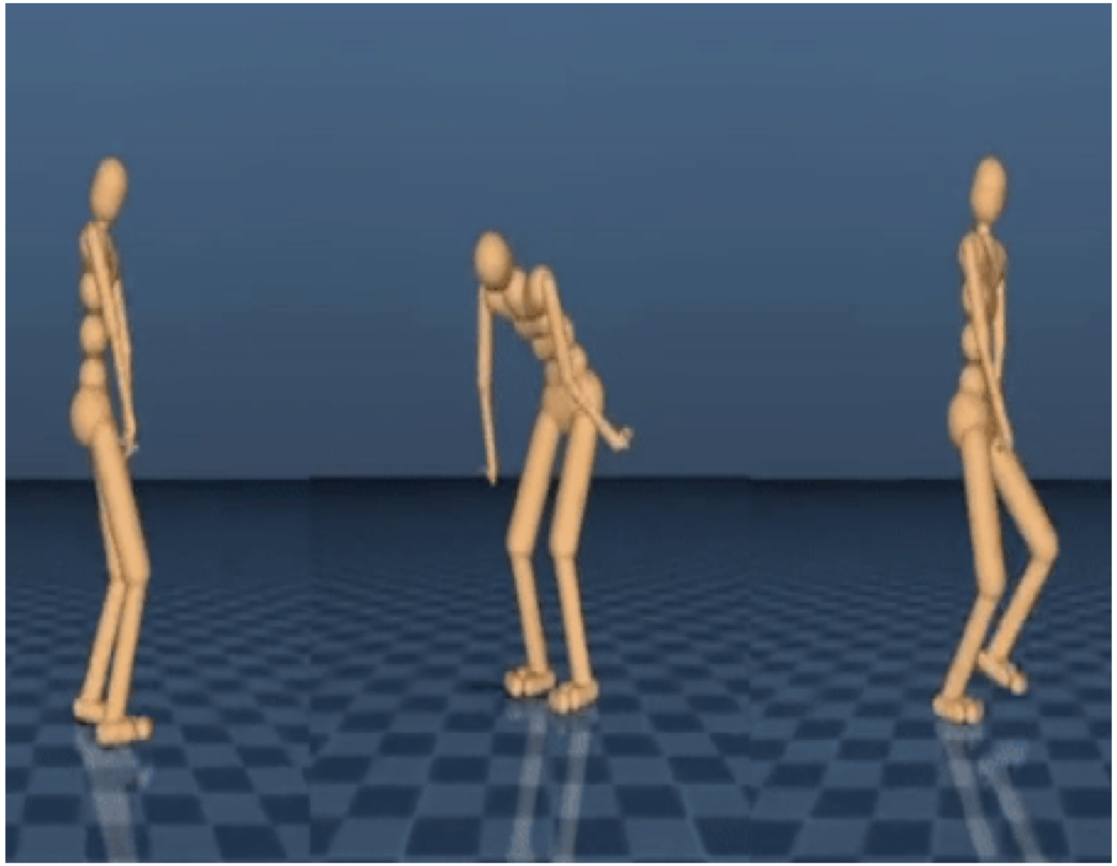


Manual Design

[**QWOP** - http://www.foddy.net/Athletics.html](http://www.foddy.net/Athletics.html)

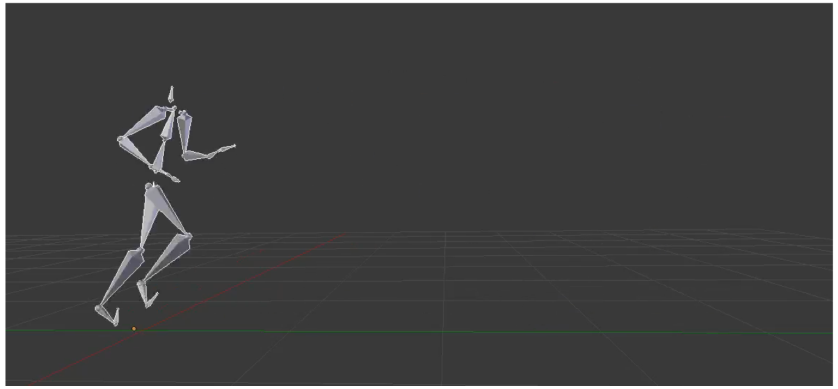
Learning Controller





https://deepmind.com/blog/producing-flexible-behaviours-simulated-environments/

Tracking Controller



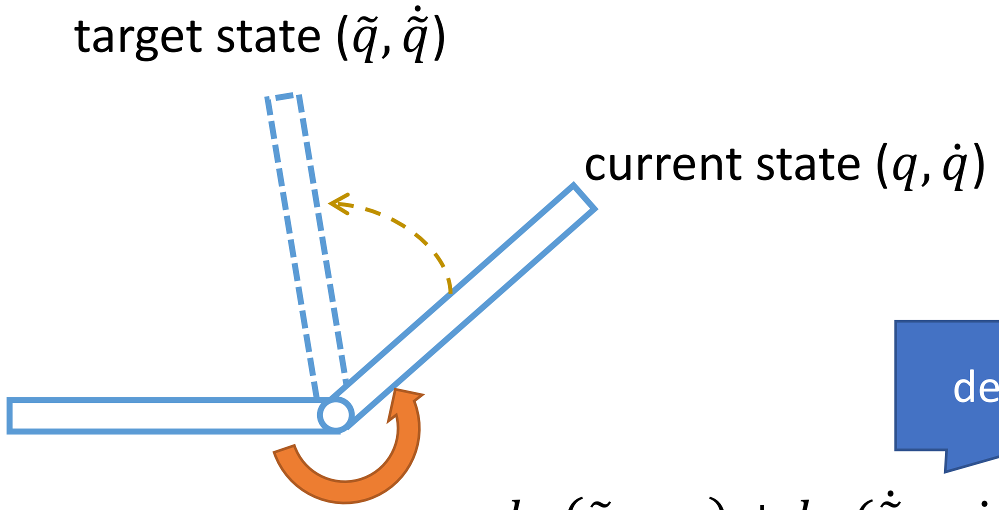
Reference Motion
(Keyframes)

Control Policy
(physics-based simulation)

Outline

- Physics-based Character Animation
- Tracking control
  - Sampling-based motion control (SAMCON)
  - Linear feedback policy
- Reinforcement Learning
  - Reward-weight regression
  - Policy gradient & nonlinear policy
  - Scheduler

Proportional-Derivative (PD) Control



$$\tau = k_p(\tilde{q} - q) + k_d(\dot{\tilde{q}} - \dot{q})$$

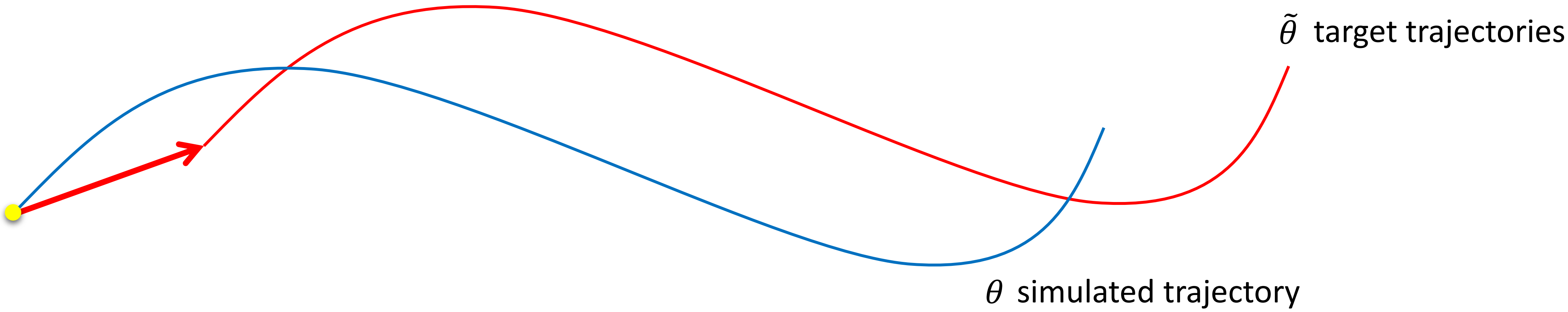
derivative

proportional

Tracking Control

PD servo

$$\tau = k_p(\tilde{\theta} - \theta) - k_d\dot{\theta}$$



Proportional-Derivative (PD) Control

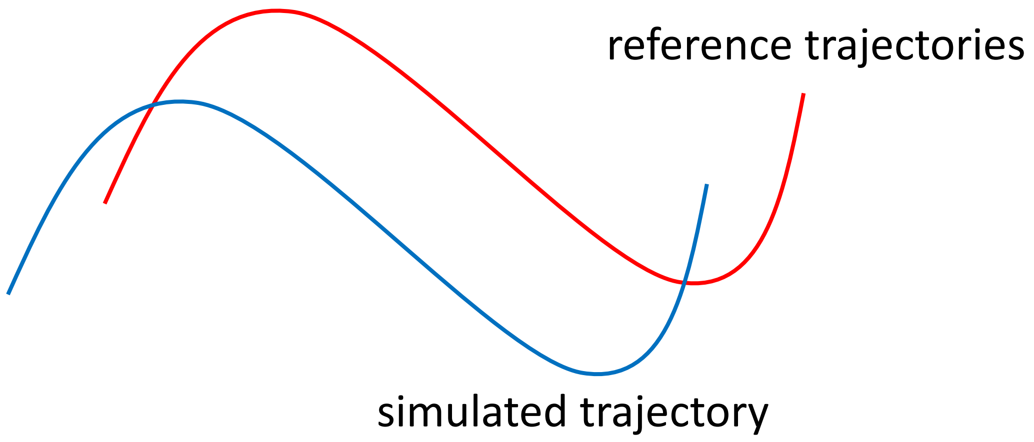
Designing target trajectory to reproduce reference

Usually better than raw torques

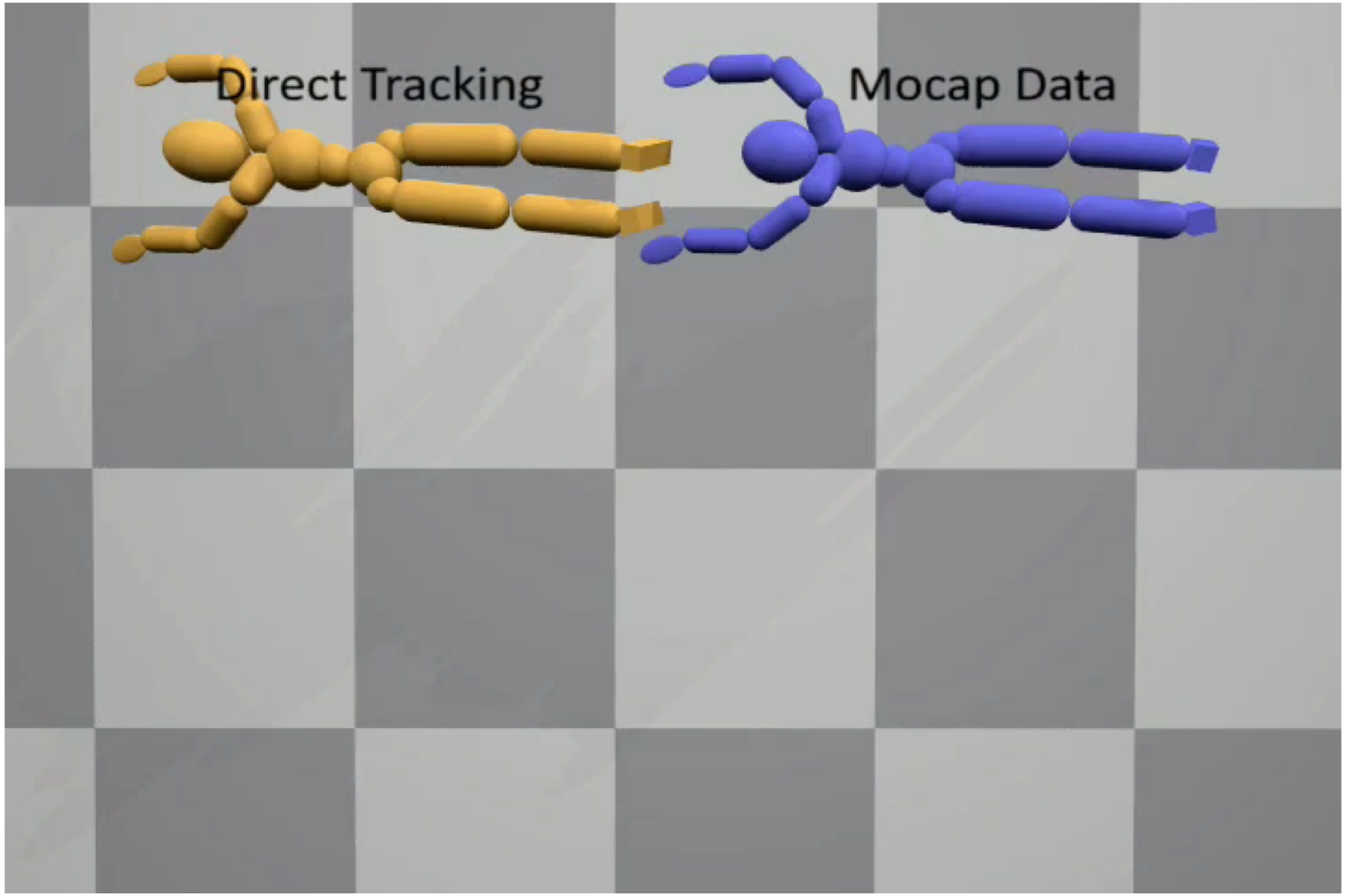
See also [Peng and van de Panne 2017 - Learning Locomotion Skills Using DeepRL: Does the Choice of Action Space Matter?]

Error → correction

Delay

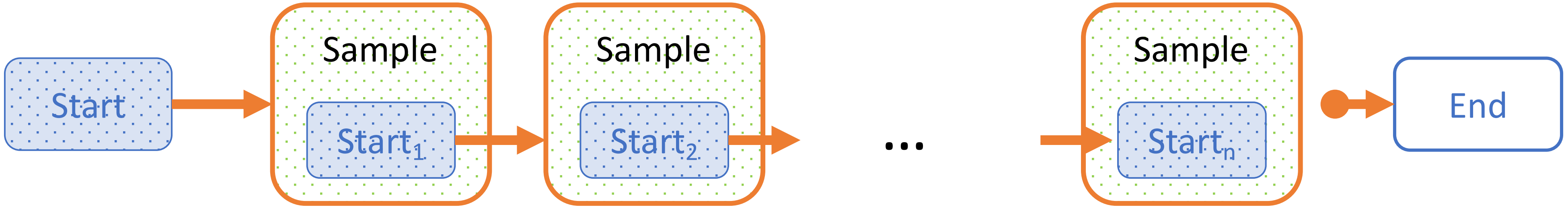


Direct Tracking Reference Motion

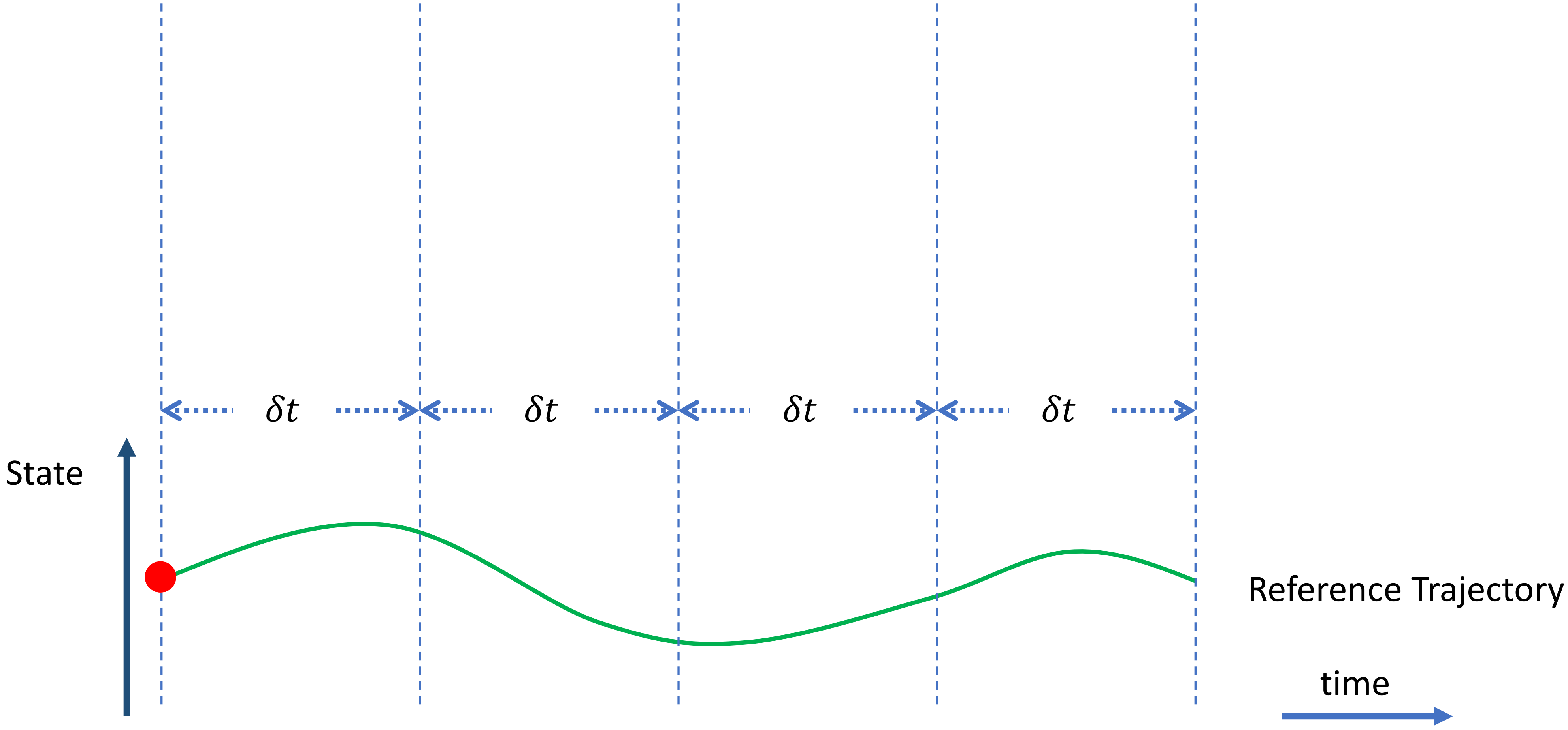


SAMCON

- **SA**mpling-based **M**otion **CON**trol [Liu et al. 2010, 2015]
  - Motion Clip → Open-loop control trajectory

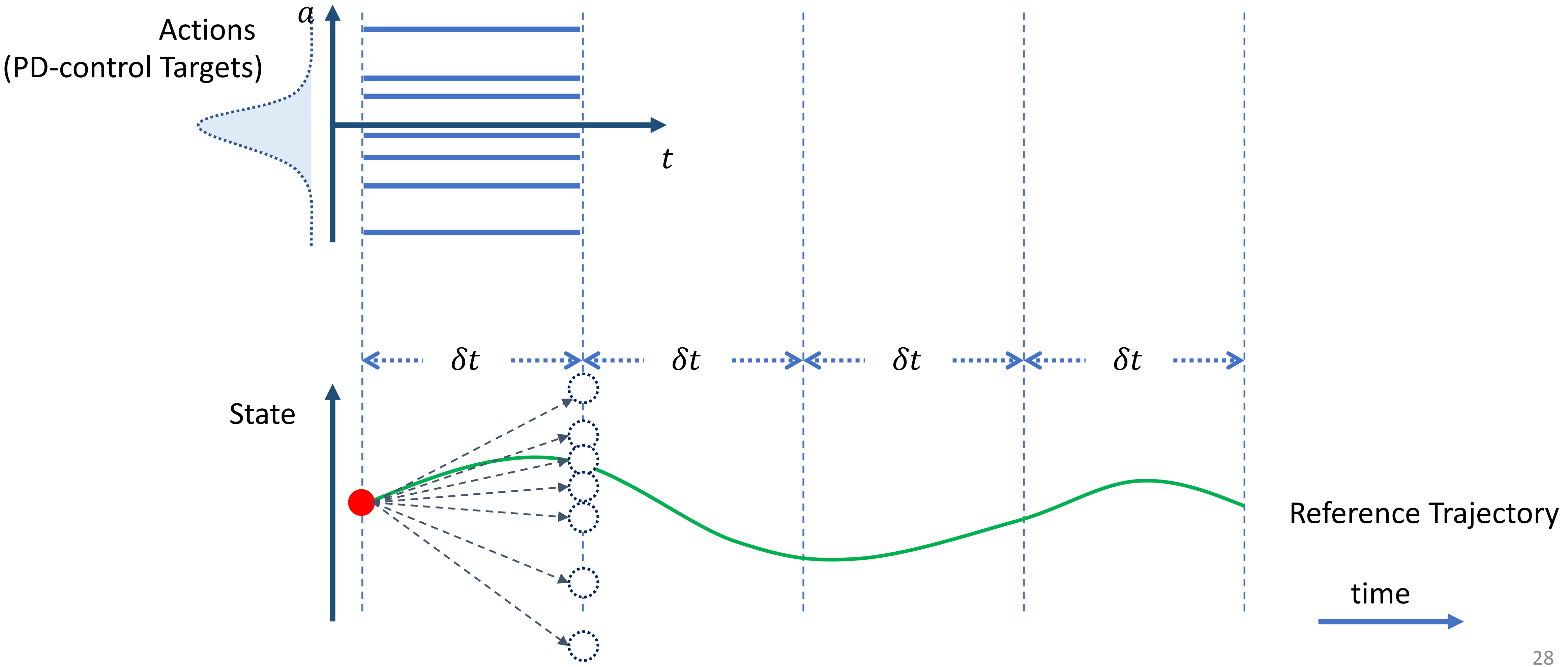


SAMCON

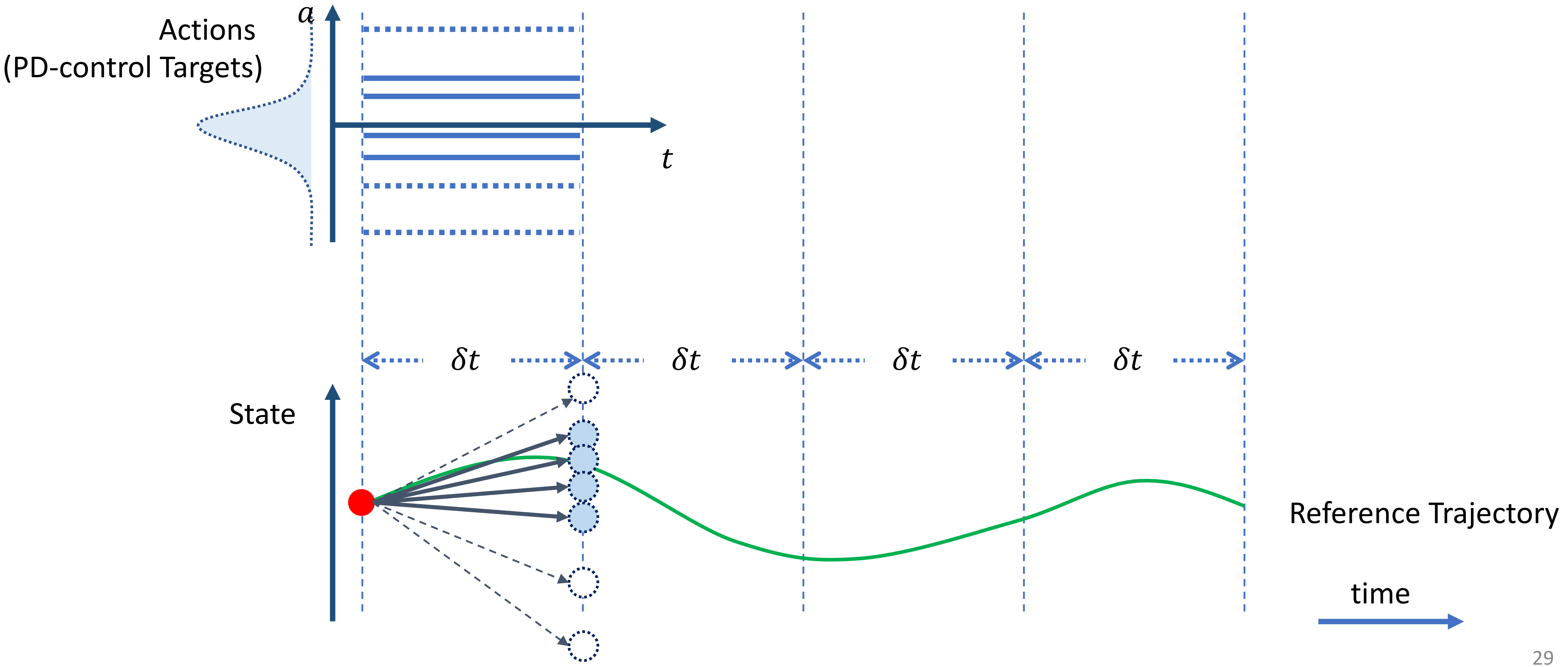


δt
δt
δt
δt
State
Reference Trajectory
time

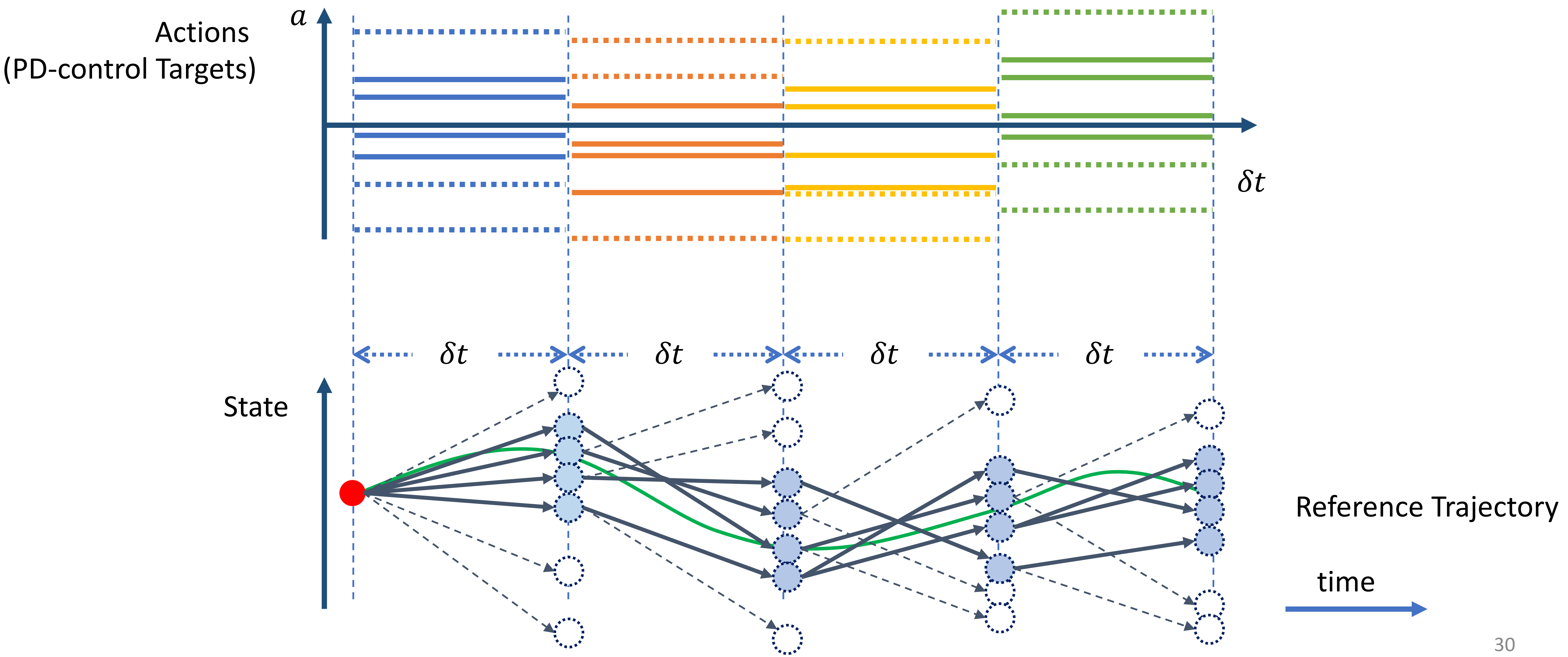
Sampling & Simulation



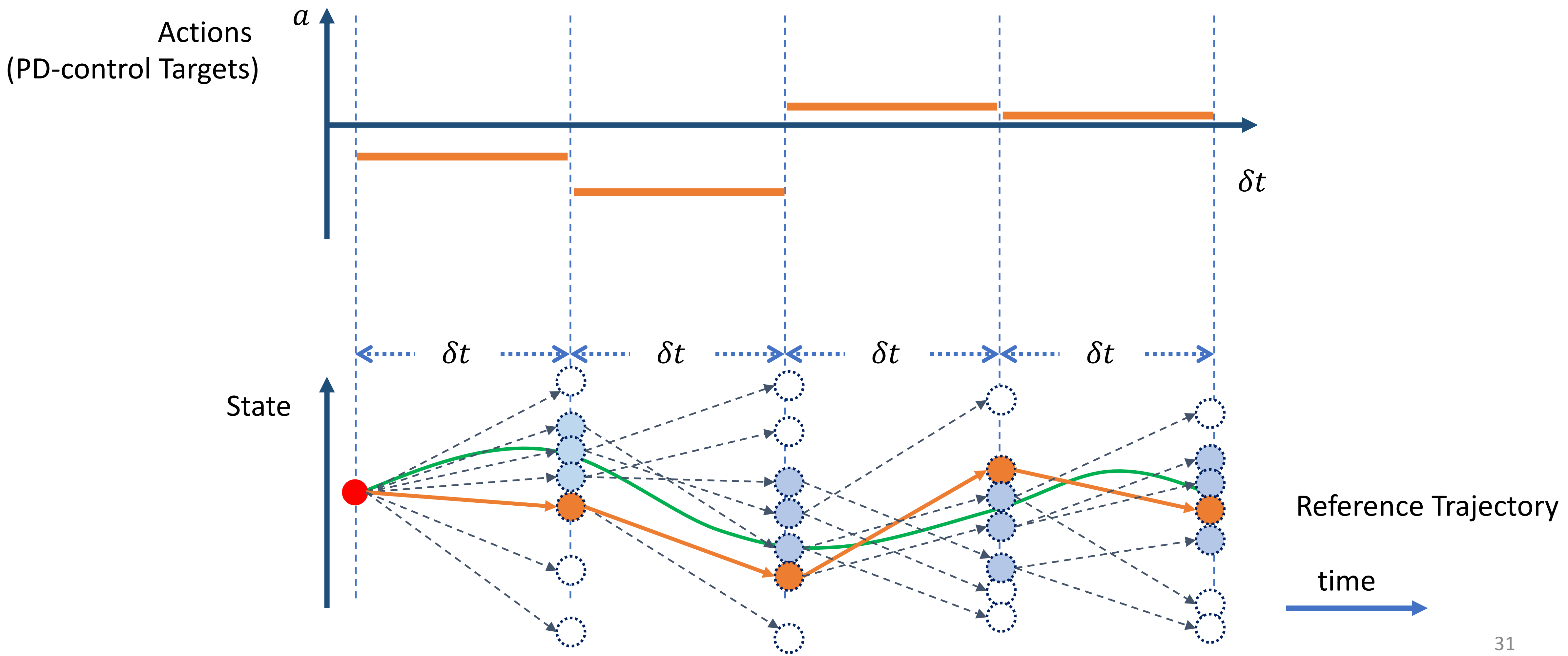
Sample Selection

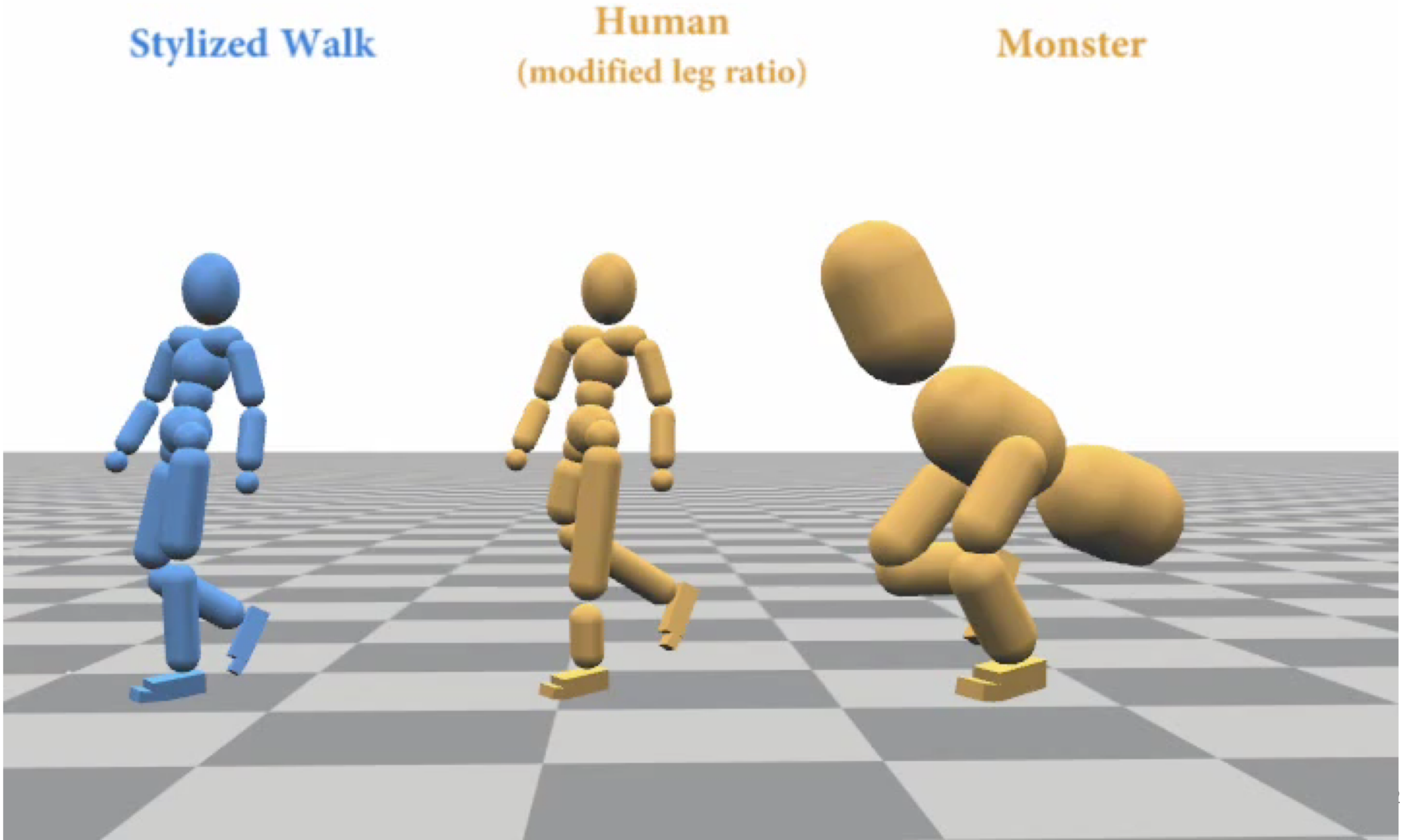


SAMCON Iterations

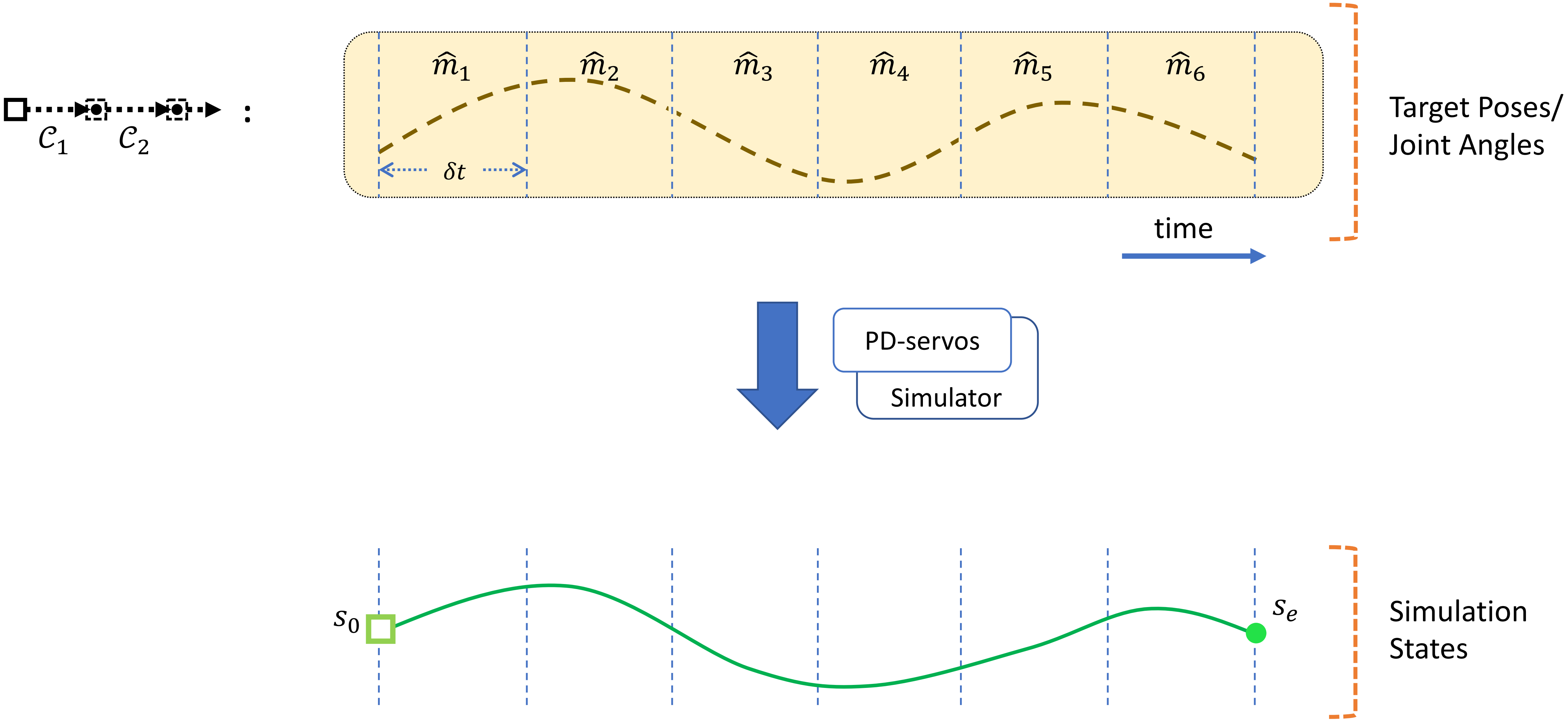


Constructed Open-loop Control Trajectory

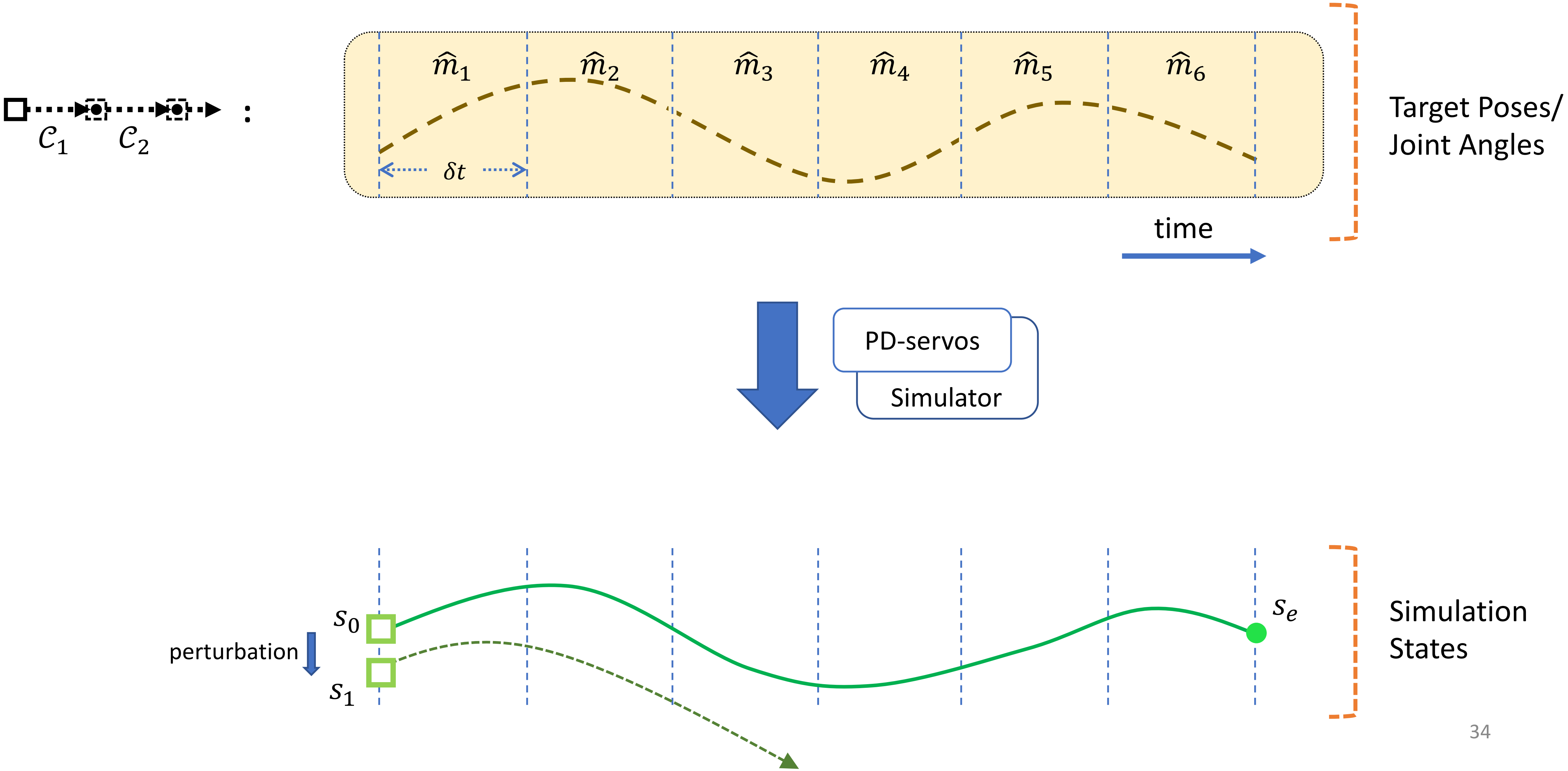



Stylized Walk
Human
(modified leg ratio)
Monster

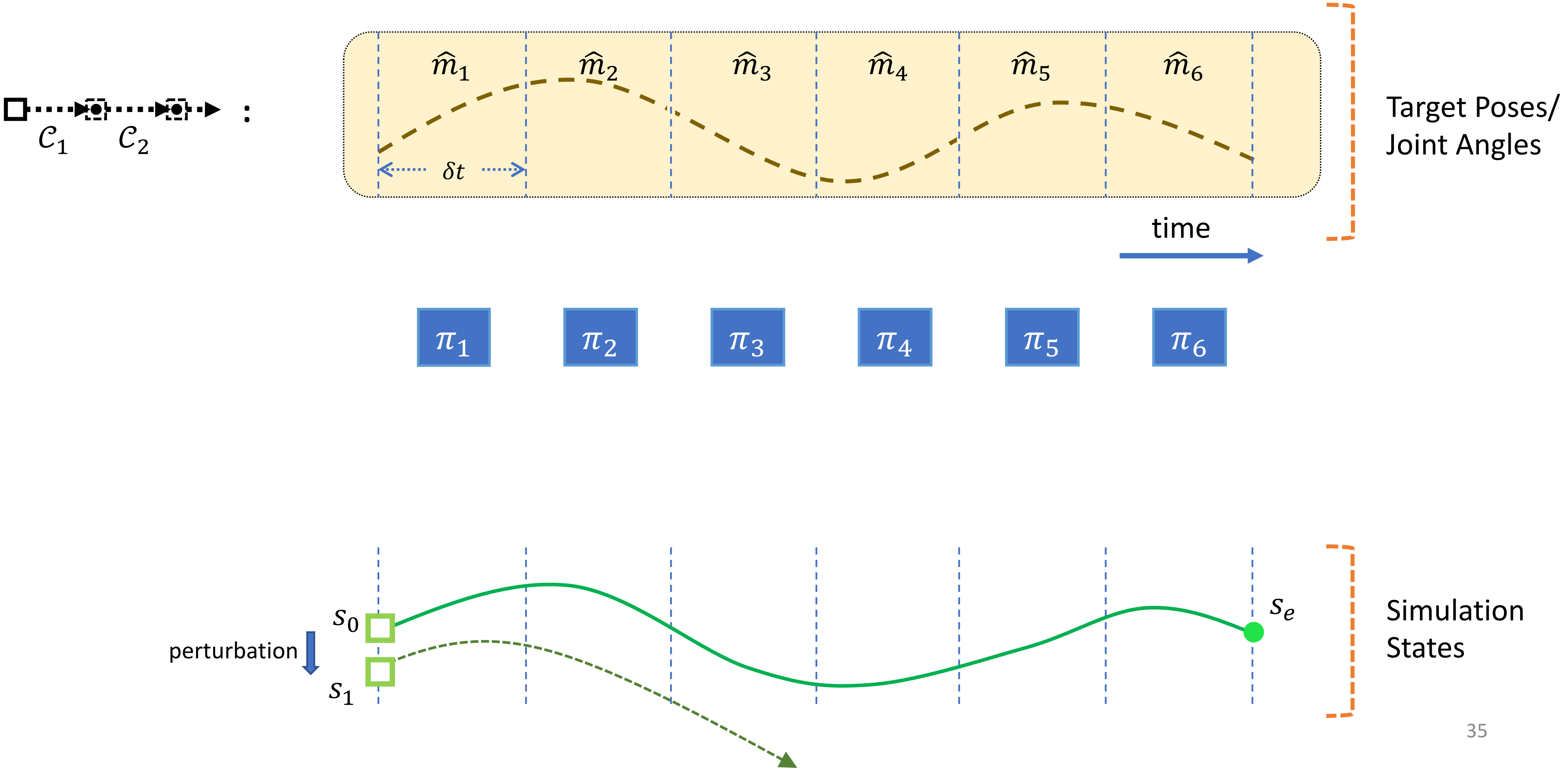
Open-loop Control Trajectory



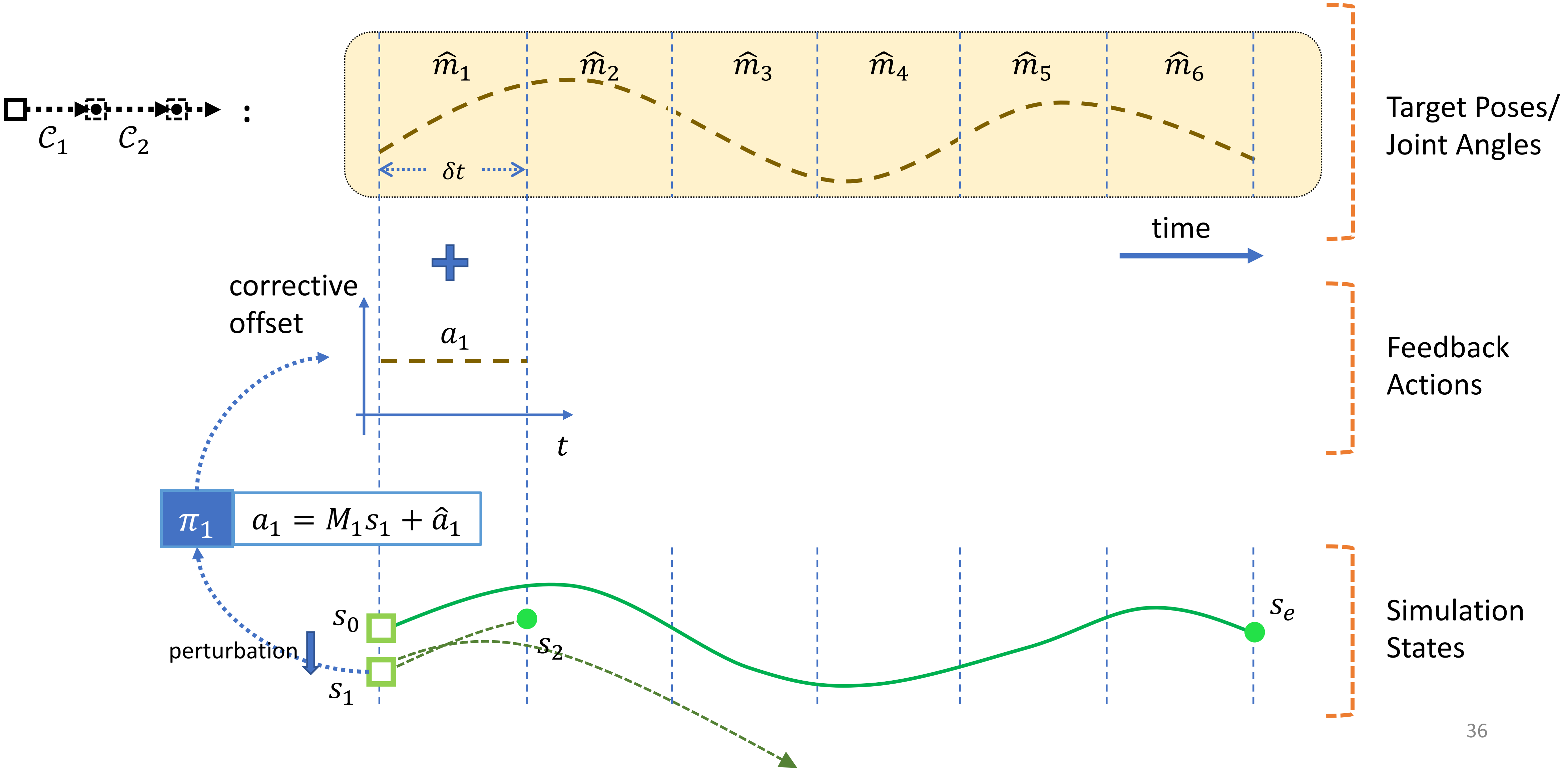
Open-loop Control Trajectory



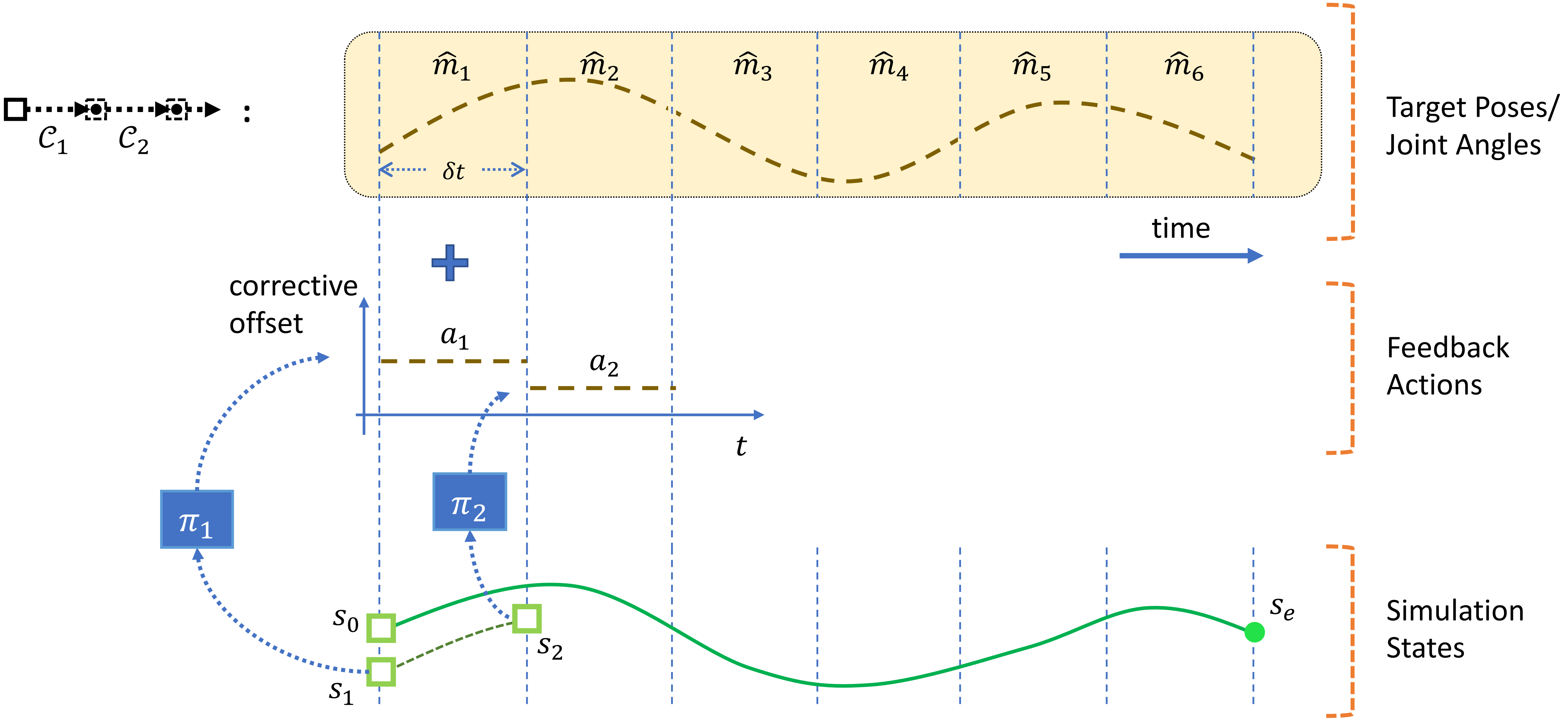
Open-loop Control Trajectory



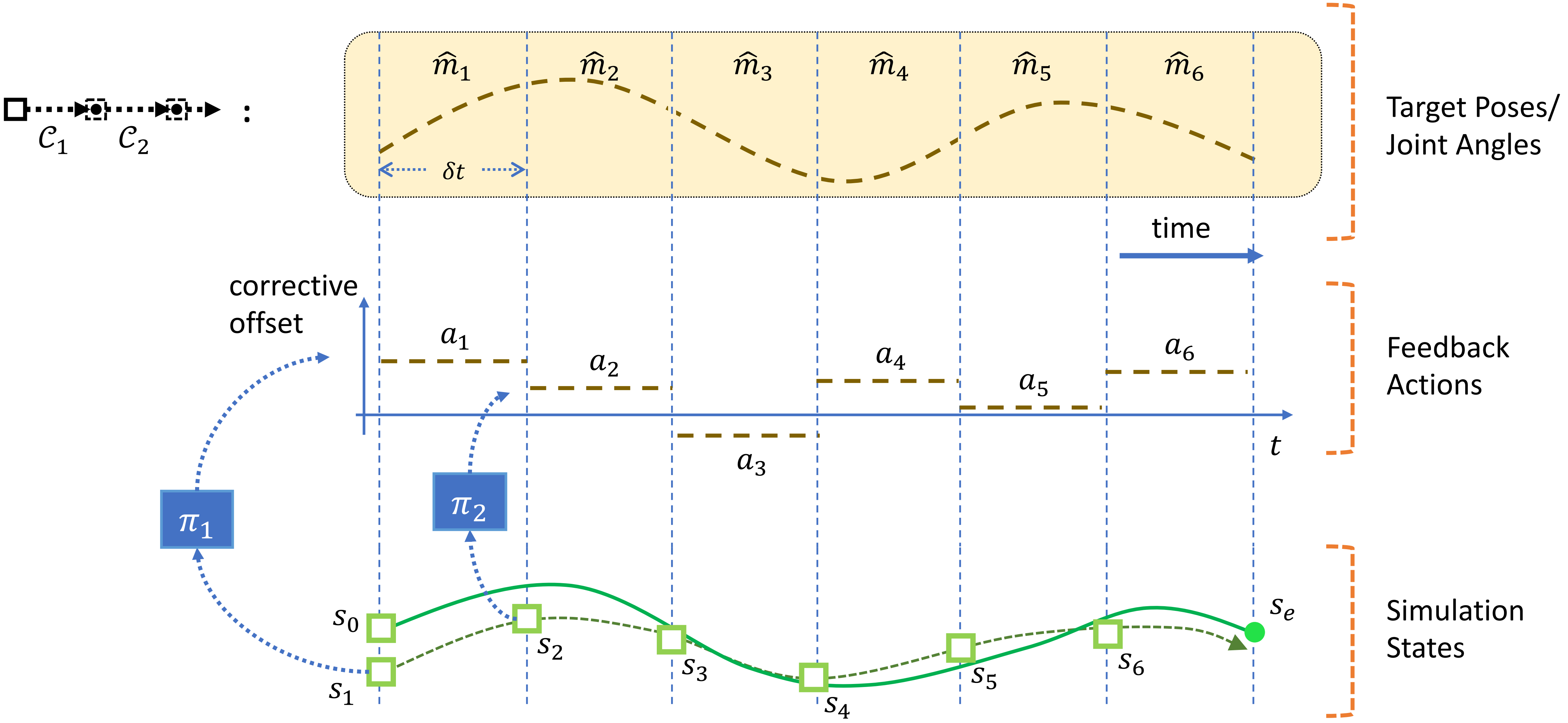
Feedback Policy



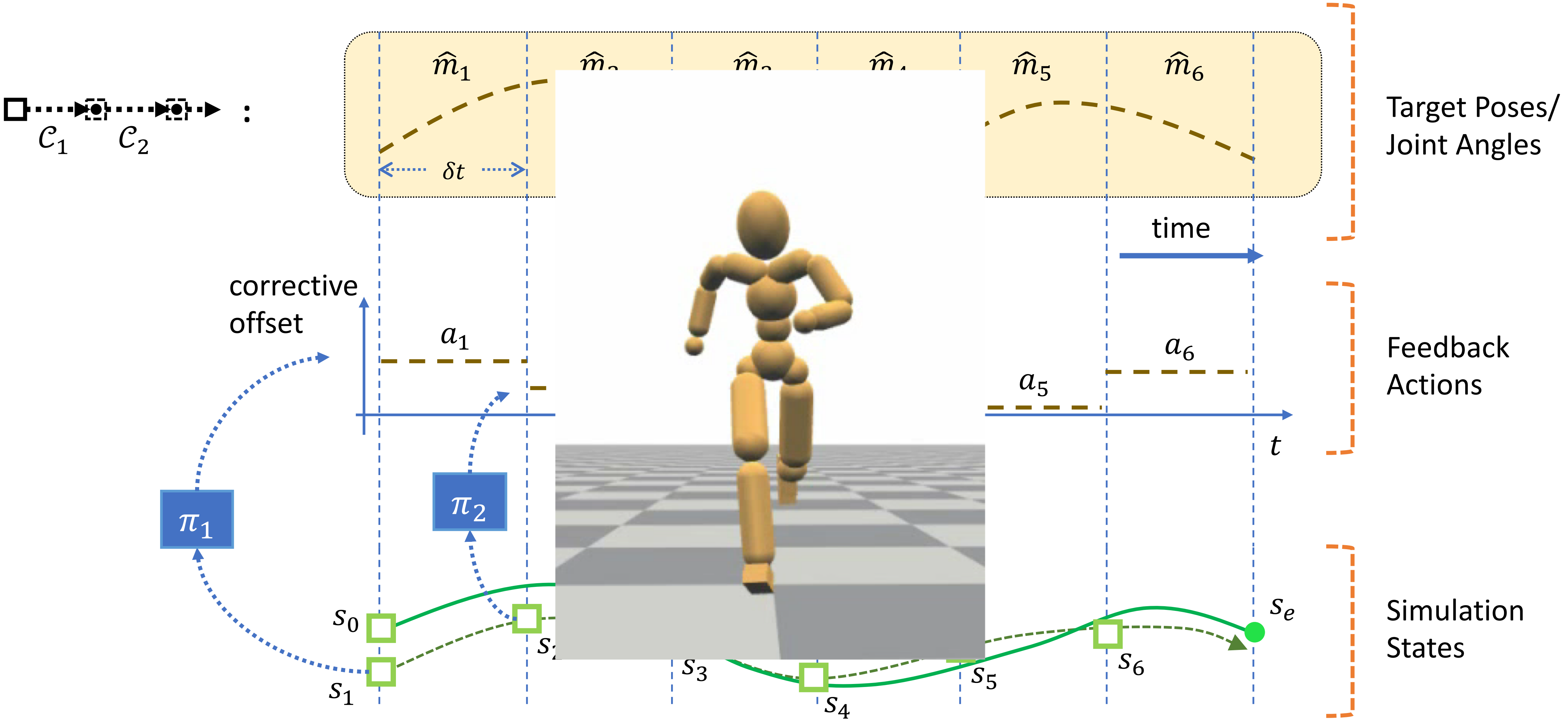
Feedback Policy



Feedback Policy



Feedback Policy

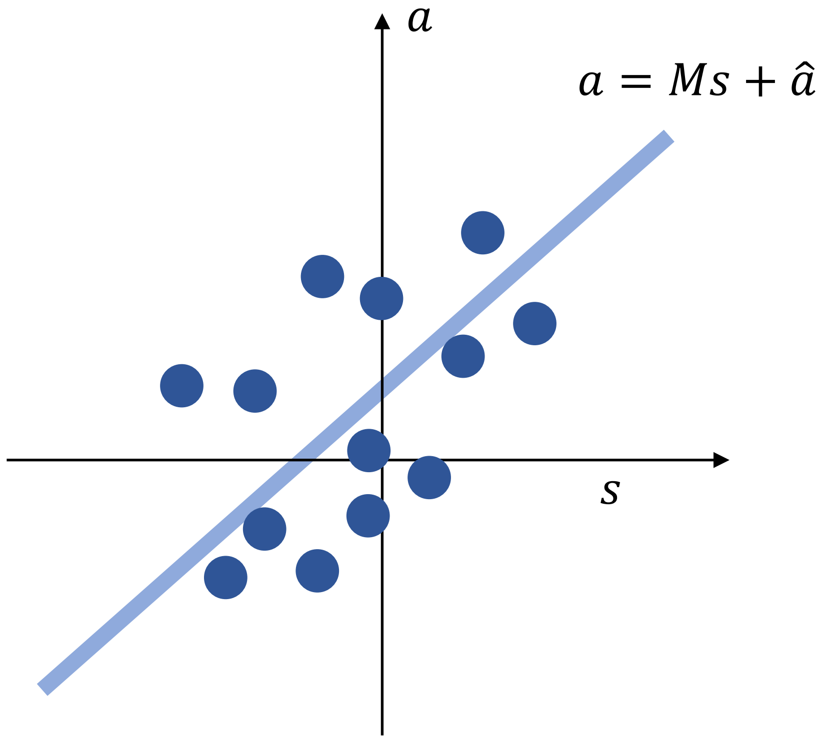


Linear Regression

Given samples $\{(s_i, a_i)\}$

Find linear approximator $a = Ms + \hat{a}$

$$\min_{M,\hat{a}} \sum_i \|a_i - (Ms_i + \hat{a})\|_2$$



Linear Regression

$$\min_{M,\hat{a}} \sum_i \|a_i - (Ms_i + \hat{a})\|_2$$

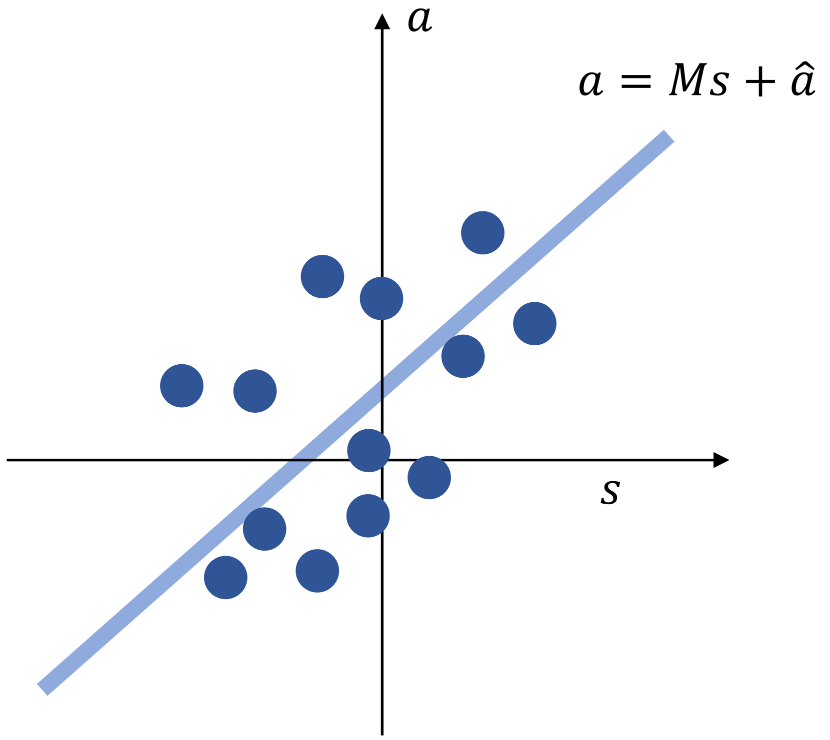
gives

$$M = [(S^T S)^{-1}(S^T A)]^T$$

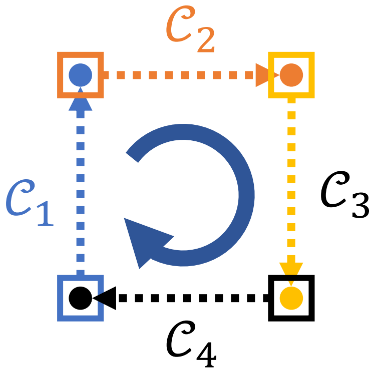
$$\hat{a} = \bar{a} - M\bar{s}$$

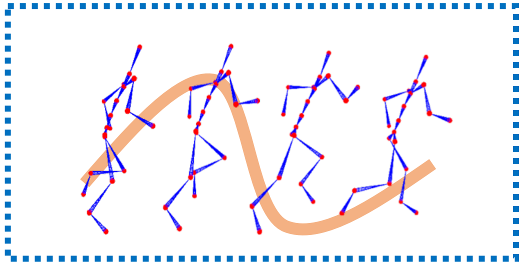
where

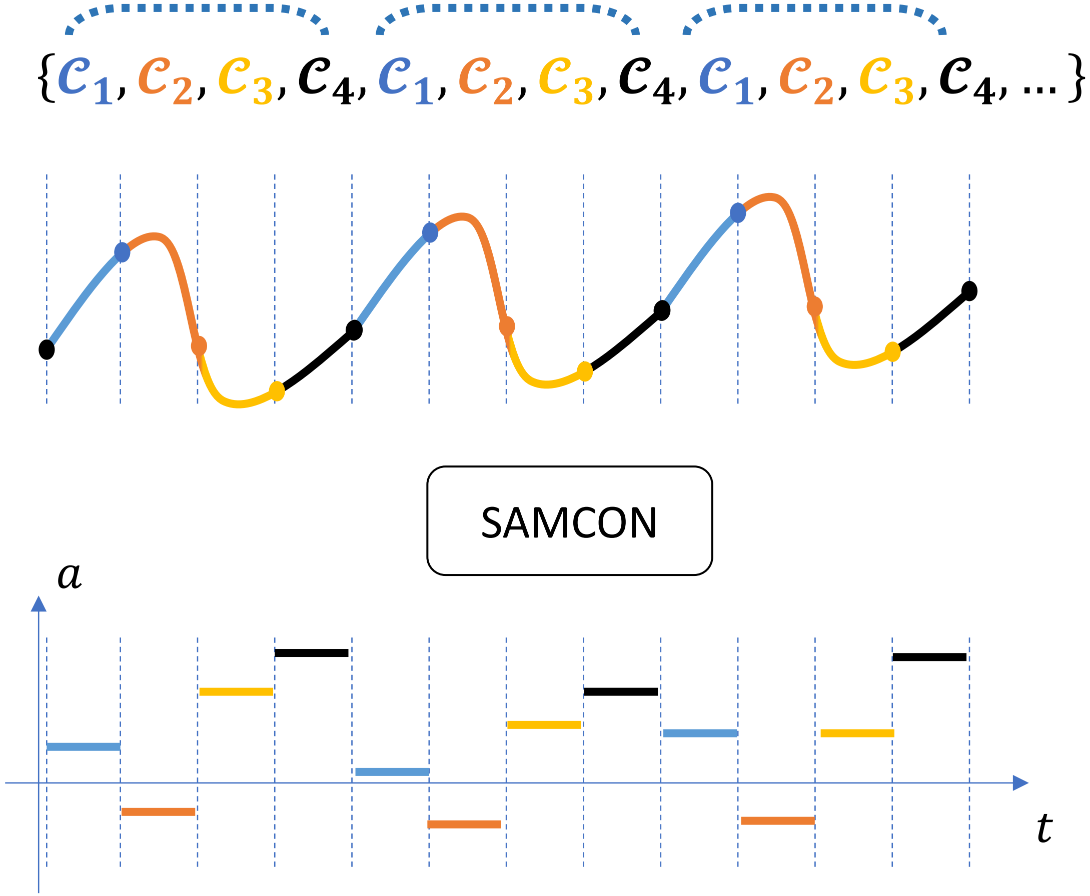
$$S = \begin{bmatrix} \vdots & \vdots & \vdots \\ s_1 - \bar{s} & \cdots & s_n - \bar{s} \\ \vdots & \vdots & \vdots \end{bmatrix}^T \quad A = \begin{bmatrix} \vdots & \vdots & \vdots \\ a_1 - \bar{a} & \cdots & a_n - \bar{a} \\ \vdots & \vdots & \vdots \end{bmatrix}^T$$



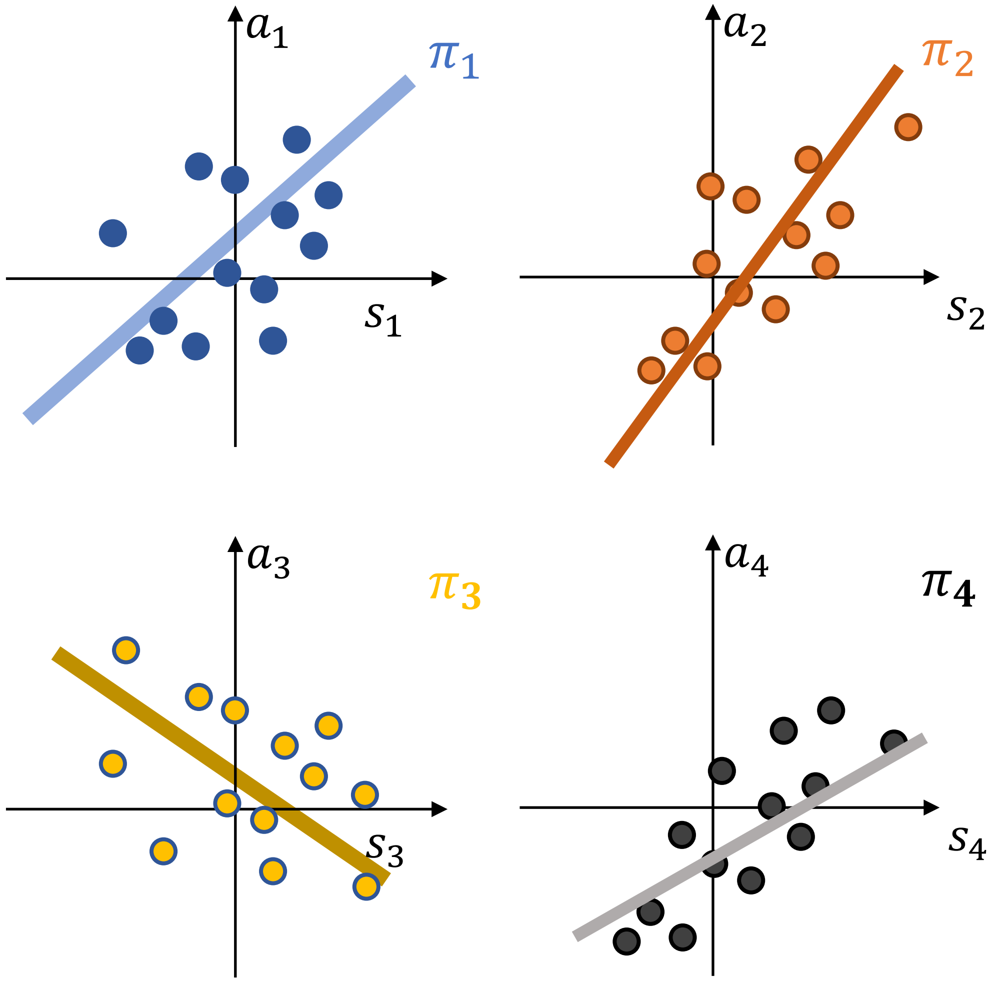
Collect Samples for Policy Regression

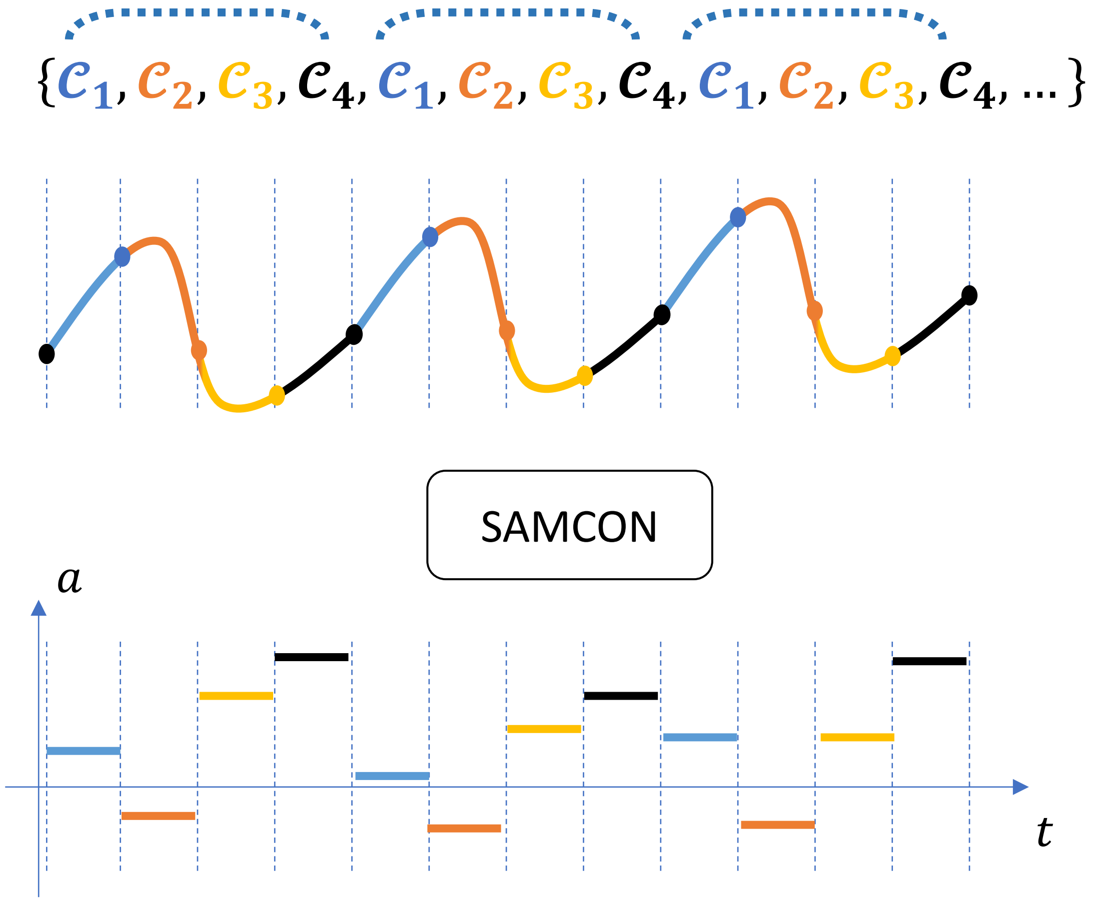




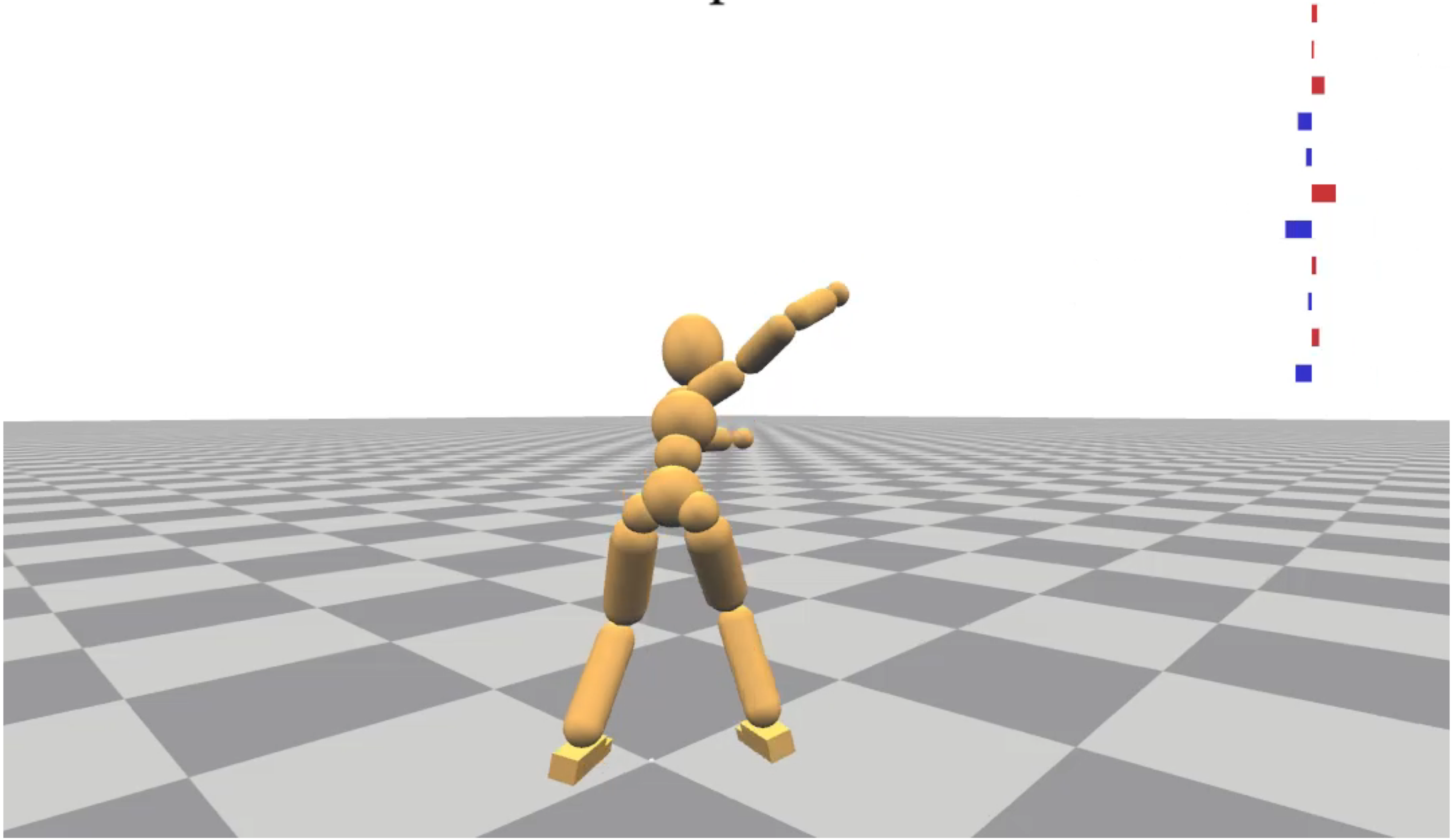


Stepwise Linear Policy Regression

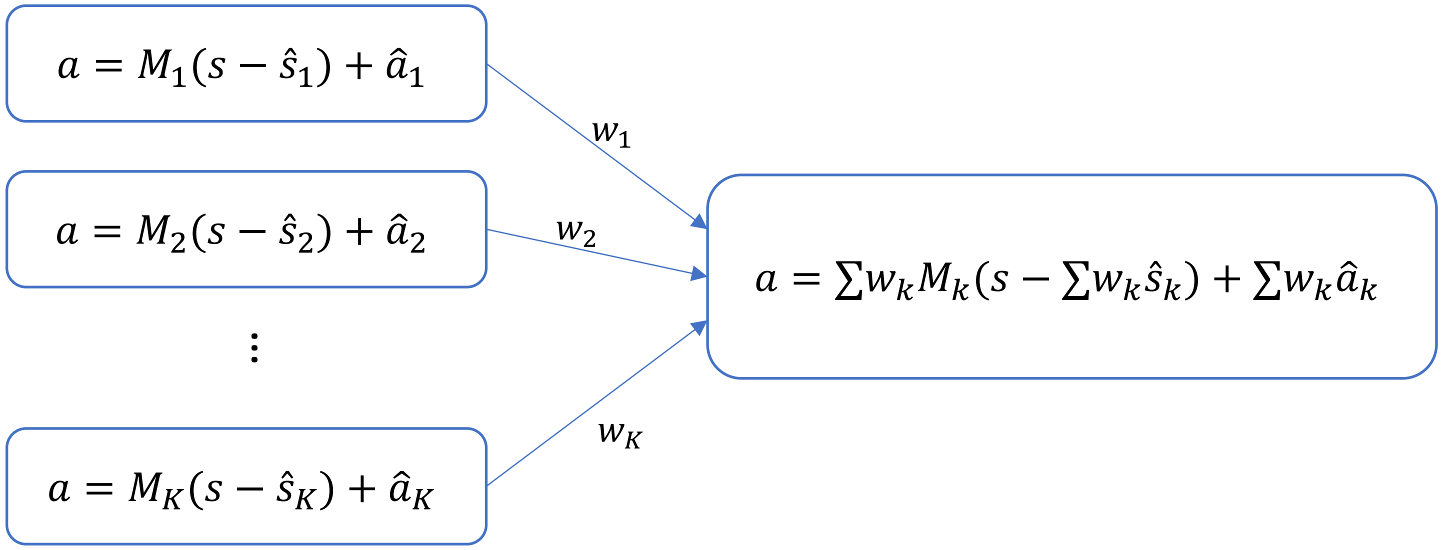




Spin Kick

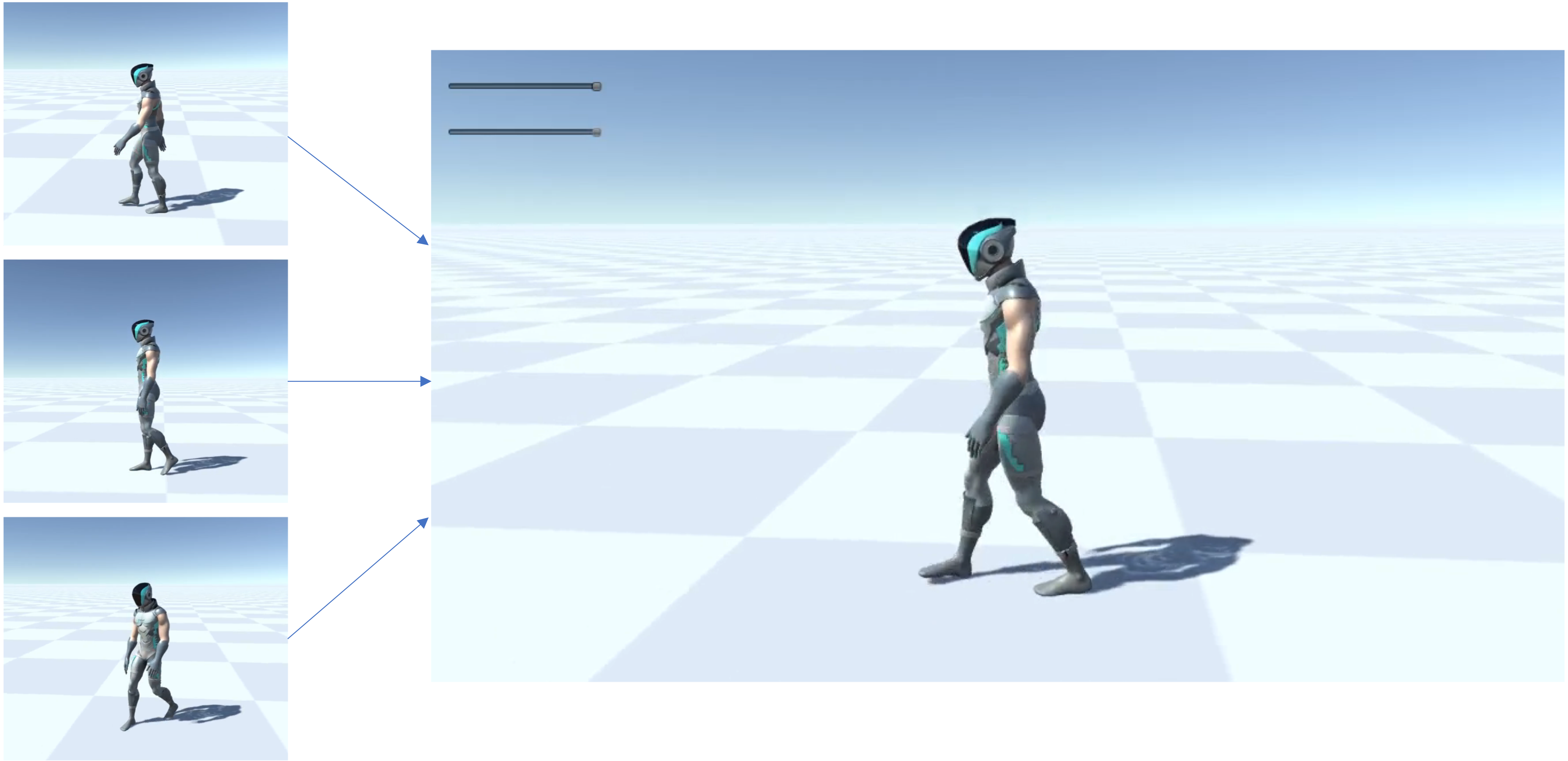


Blending Between Linear Policies



Blending Between Linear Policies

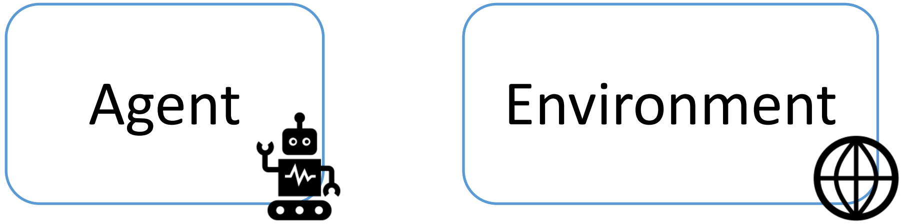
Blending Between Linear Policies



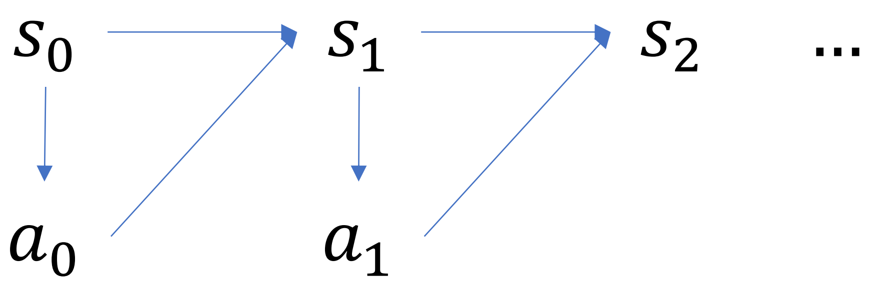
Outline

- Physics-based Character Animation
- Tracking control
  - Sampling-based motion control (SAMCON)
  - Linear feedback policy
- Reinforcement Learning
  - Reward-weight regression
  - Policy gradient & nonlinear policy
  - Scheduler

Markov Decision Process



Markov Decision Process

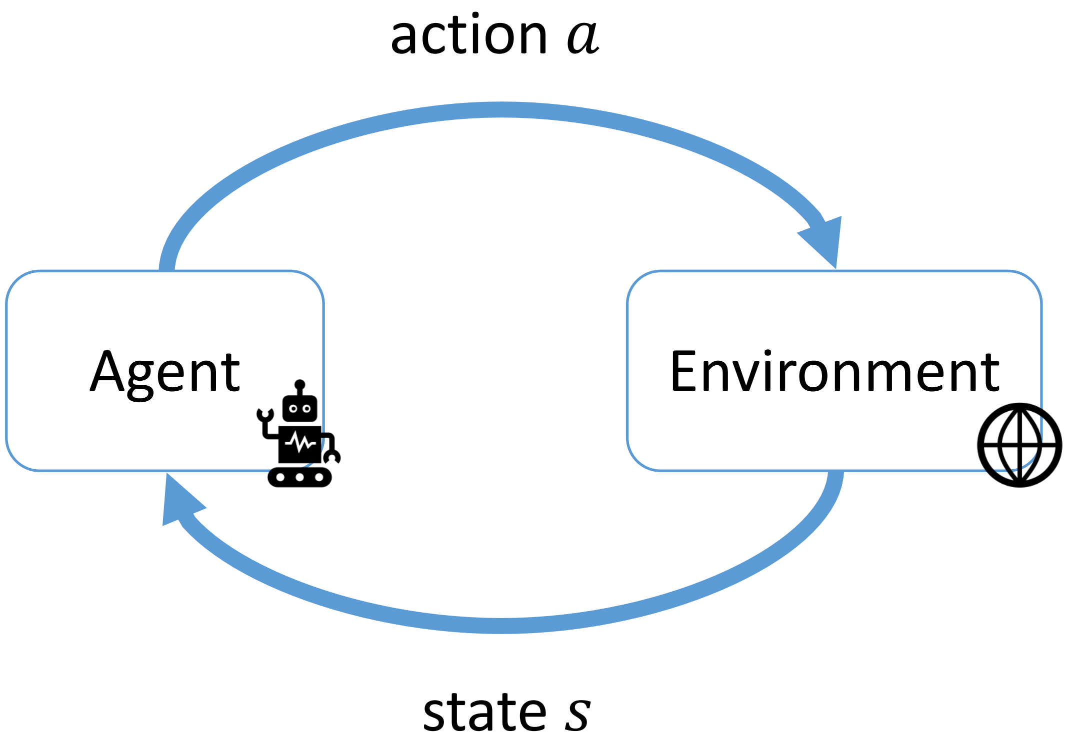


Policy $a_t \sim \pi(\cdot\,|s_t, \theta)$

Transition probability

$$s_{t+1} \sim p(\cdot\,|s_t, a_t)$$

- Unknown
- Independent of $s_{T \le t-1}, a_{T \le t-1}$
- Markov property



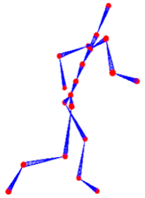
Markov Decision Process

Trajectory

$$p(\tau|\theta) = p(s_0)\prod_i \pi(a_i|s_i,\theta)p(s_{i+1}|s_i,a_i)$$

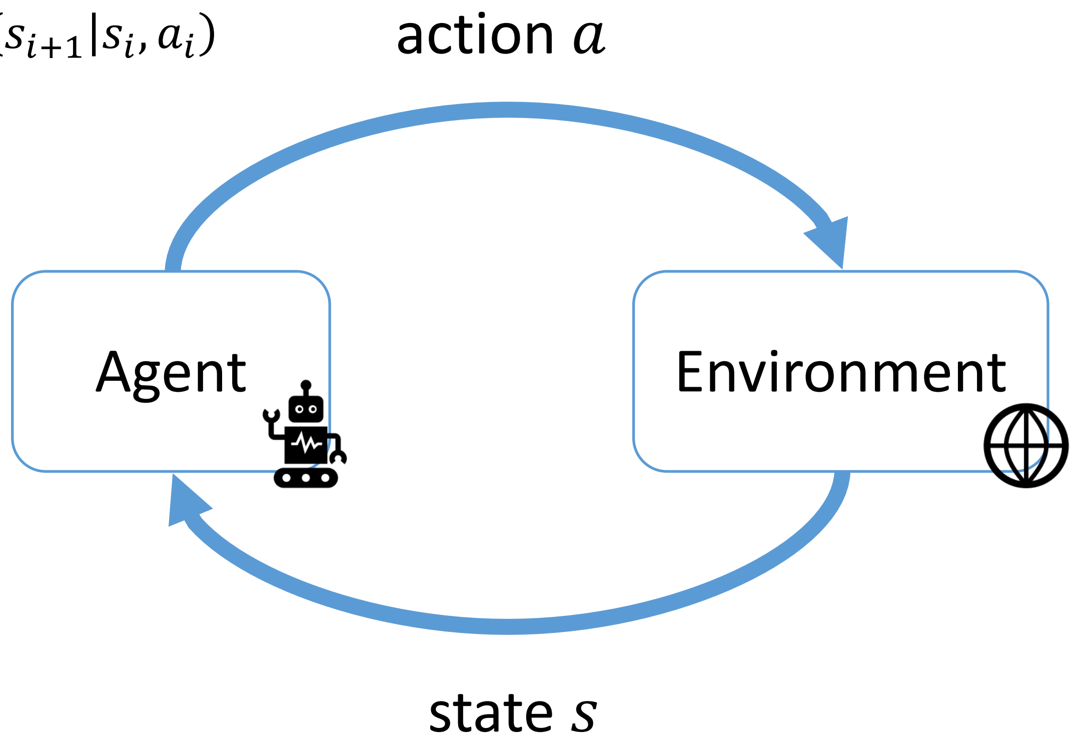
$$\tau = s_0 \quad a_0 \quad s_1 \quad a_1 \quad s_2 \quad \ldots$$

Reward

$r(s_t, a_t) = \|$  $-$  $\| + \cdots$

Return

$$R(\tau) = \Sigma_i \gamma^i r(s_i, a_i)$$



.

Reinforcement Learning

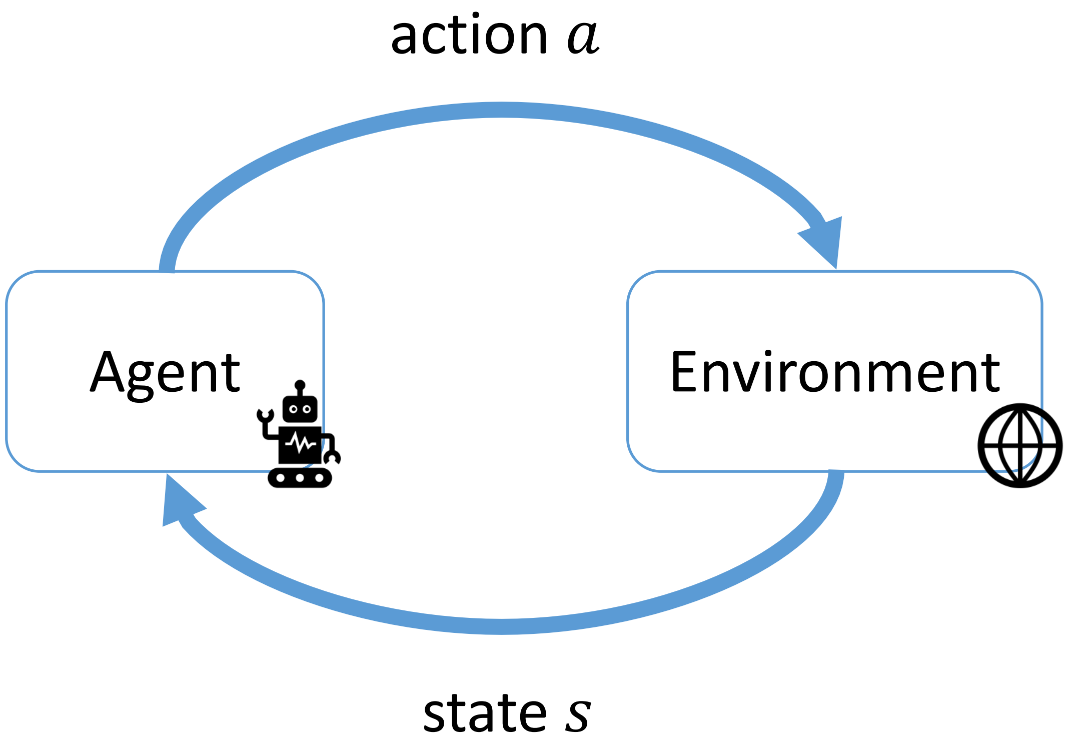
Find policy $\pi|\theta$ that maximize objective

$$J(\theta) = \int_\tau p(\tau|\theta)R(\tau)\,\mathrm{d}\tau$$

where

$$p(\tau|\theta) = p(s_0)\prod_i \pi(a_i|s_i,\theta)p(s_{i+1}|s_i,a_i)$$

unknown



Reward-Weighted Regression

[Jan Peters and Stefan Schaal. 2007]. *Reinforcement learning by reward-weighted regression for operational space control*

To find the optimal policy $\pi(a|s,\theta)$ that maximize

$$J(\theta) = \int_\tau p(\tau|\theta) R(\tau)\, \mathrm{d}\tau$$

$$\frac{p(s_0)\Pi_i \pi(a_i|s_i,\theta) p(s_{i+1}|s_i,a_i)}{p(s_0)\Pi_i \pi(a_i|s_i,\theta') p(s_{i+1}|s_i,a_i)}$$

consider the lower bound (assume $J(\theta')$ is known)

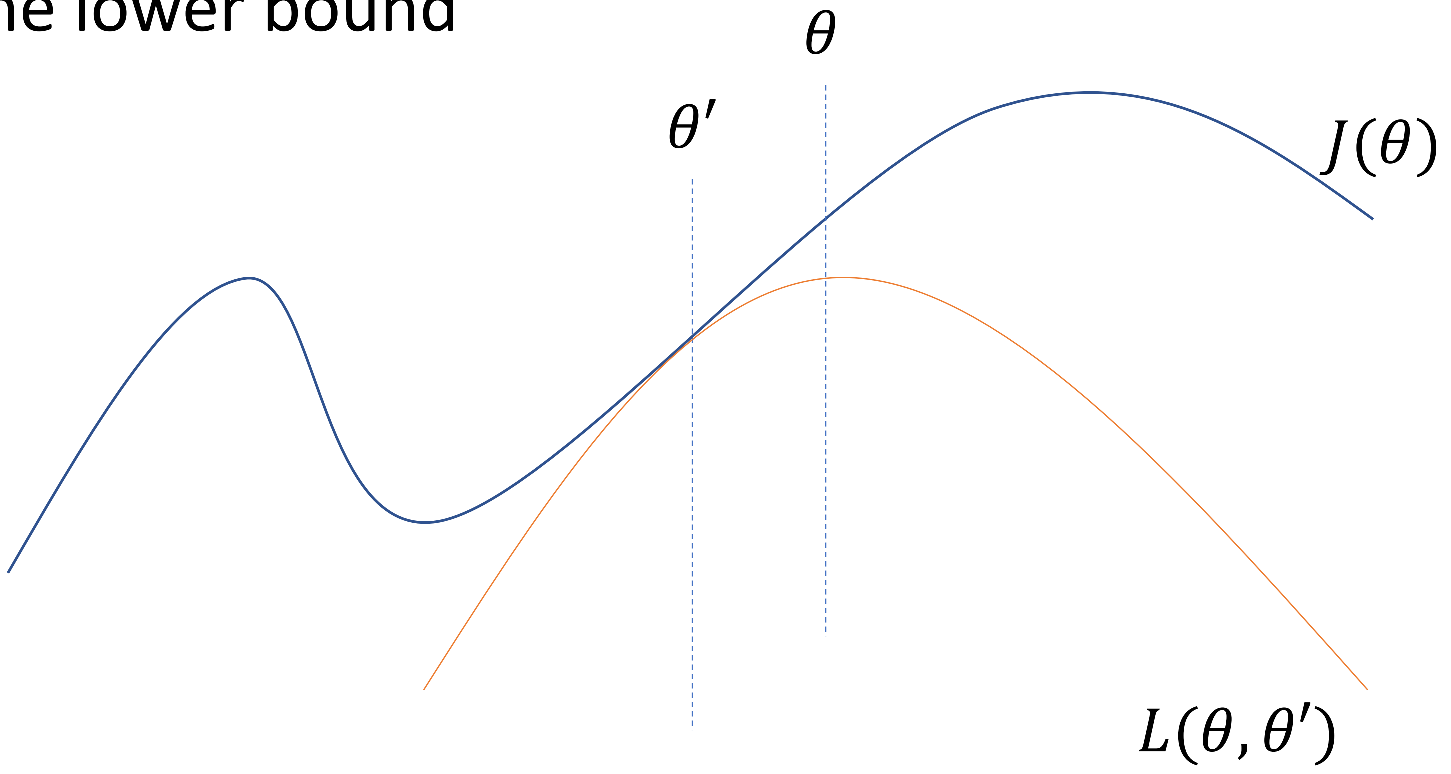
$$\log \frac{J(\theta)}{J(\theta')} \geq \frac{1}{J(\theta')} \int_\tau p(\tau|\theta') R(\tau) \log \frac{p(\tau|\theta)}{p(\tau|\theta')}\, \mathrm{d}\tau$$

$$\propto \int_\tau p(\tau|\theta') R(\tau) \sum_{i=0}^{n-1} \log \pi(a_i|s_i,\theta)\, \mathrm{d}\tau + C(\theta')$$

$$\approx \sum_{\tau\sim\theta'} R(\tau) \sum_{i=0}^{n-1} \log \pi(a_i|s_i,\theta) + C(\theta') \quad = L(\theta,\theta') + C(\theta')$$

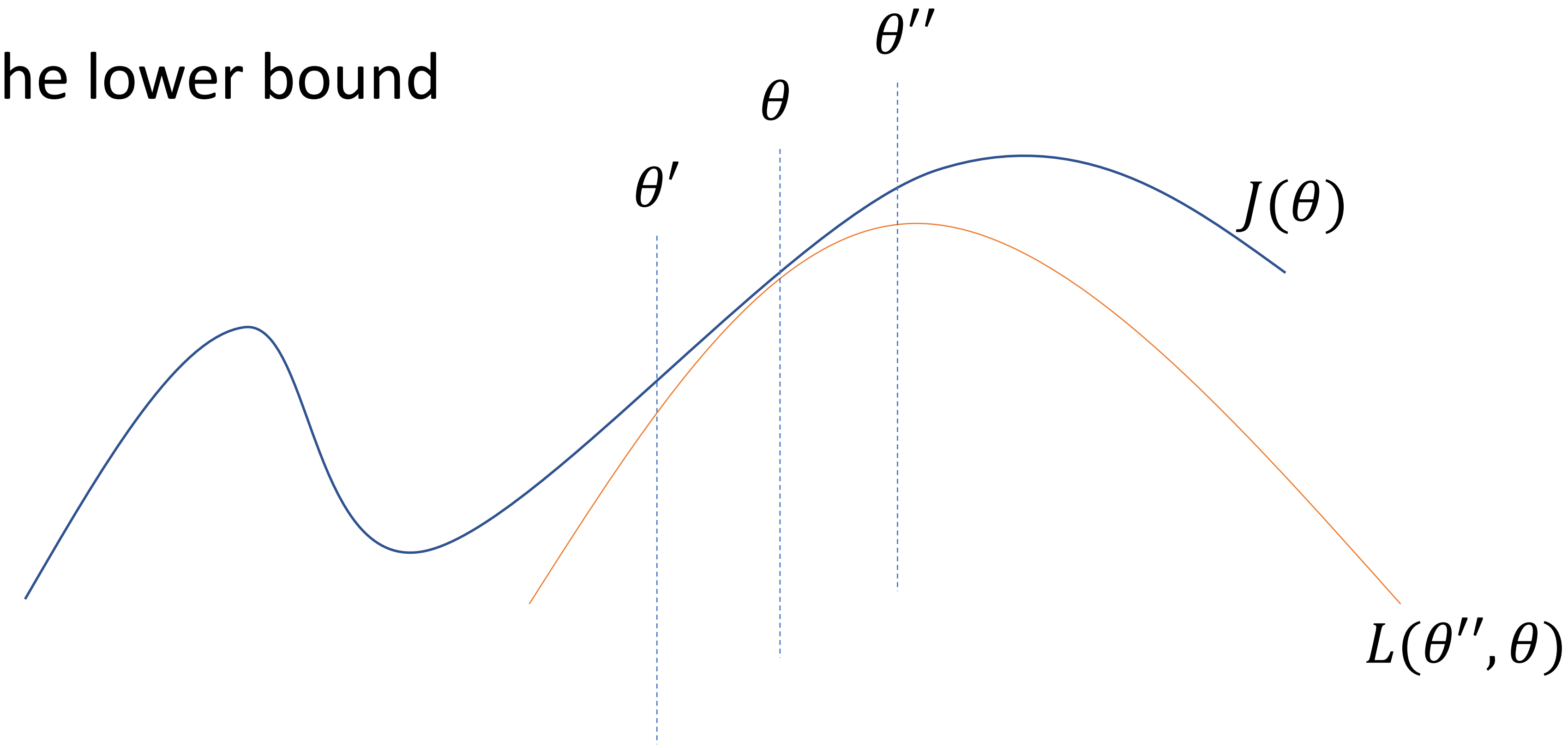
Reward-Weighted Regression

Maximize the lower bound



Reward-Weighted Regression

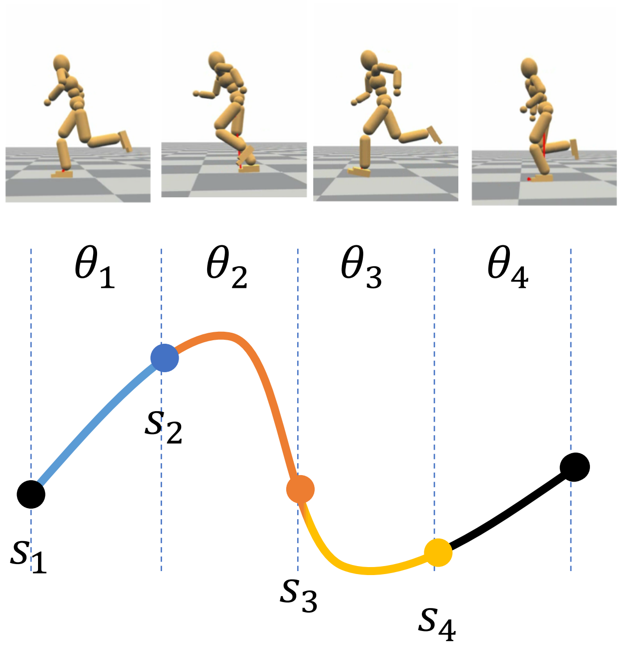
Maximize the lower bound



Stepwise Linear Policy

Control Policy

$$\pi(a_i|s_i,\theta) = \pi(a_i|s_i,\theta_i)$$
$$= \mathcal{N}(M_i s_i + b_i, \Sigma_i)$$



Return function

$$R(\tau) = \begin{cases} 1 & \text{if } \tau \text{ is close to the reference} \\ 0 & \text{otherwise} \end{cases}$$

Stepwise Linear Policy

Optimal lower bound

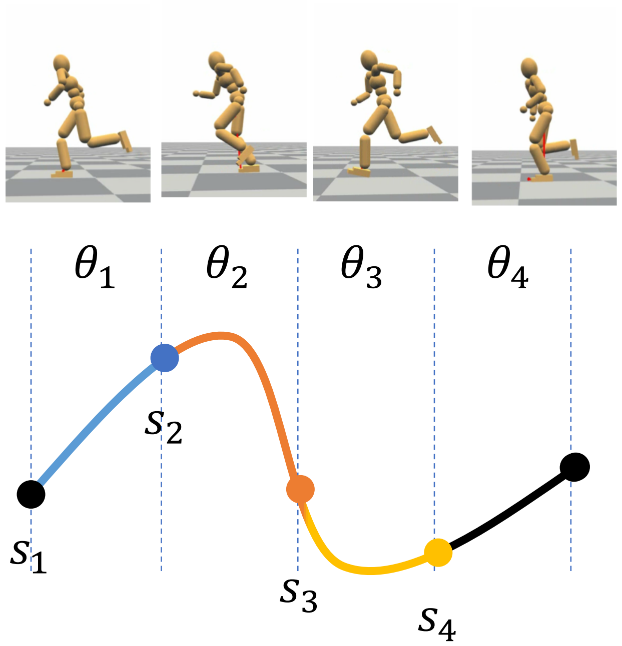
$$L(\theta,\theta') = \sum R(\tau) \sum_i \log \pi(a_i|s_i,\theta)$$

$$= -\frac{1}{2}\sum R(\tau) \sum_i \|a_i - (M_i s_i + b_i)\|_{\Sigma_i^{-1}} + \det \Sigma_i$$

$\Rightarrow$ Linear regression

$$M_i = \left(S_i^T S_i\right)^{-1}\left(S_i^T A_i\right)$$

$$b_i = \bar{a}_i - M_i \bar{s}_i$$

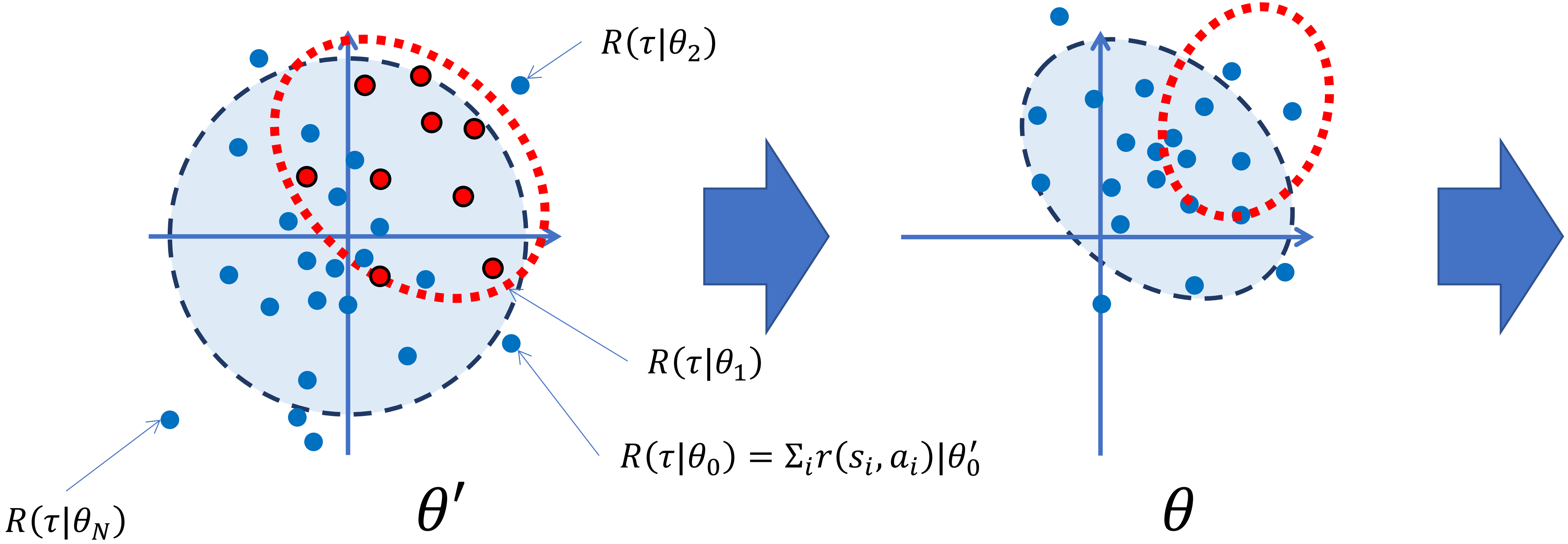


$$\pi(a_i|s_i,\theta) = \mathcal{N}(M_i s_i + b_i, \Sigma_i)$$
$$= \frac{1}{\sqrt{(2\pi)^k \det \Sigma_i}} \exp\left[-\frac{1}{2}\|a_i - (M_i s_i + b_i)\|_{\Sigma_i^{-1}}\right]$$

Gradient-free Policy Search

CMA-ES [Hansen 2006]

$$\max_\theta J(\theta) = \max_\theta \int_\tau p(\tau|\theta) R(\tau)\, \mathrm{d}\tau$$

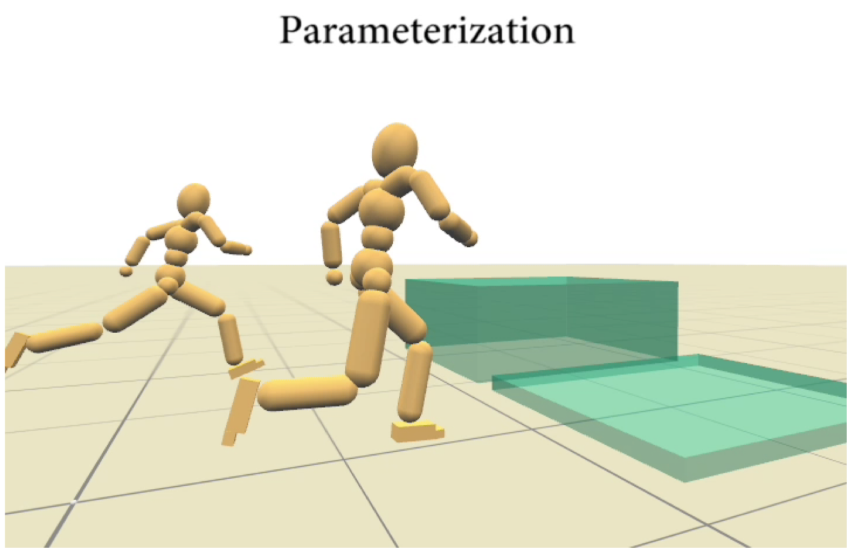


Gradient-free Policy Search

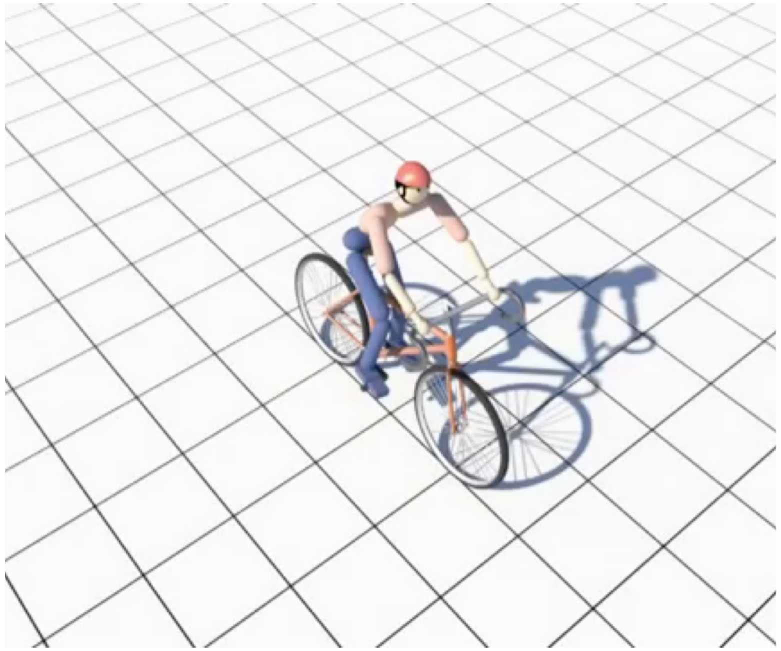
$$\max_{\theta} J(\theta) = \max_{\theta} \int_{\tau} p(\tau|\theta) R(\tau) \, \mathrm{d}\tau$$

CMA-ES, NEAT, etc.

Scalability



[Liu et al. 2013]



[Tan et al. 2014 - Learning Bicycle Stunts]

Policy Gradient

To find the optimal policy $\pi(a|s,\theta)$ that maximize

$$J(\theta) = \int_\tau p(\tau|\theta)R(\tau)\,\mathrm{d}\tau$$

Consider the gradient

$$\nabla J(\theta) = \nabla \int_\tau p(\tau|\theta)R(\tau) = \int_\tau p(\tau)\nabla \log p(\tau|\theta)\, R(\tau) = \int_\tau p(\tau)R(\tau) \sum_{\mathrm{i}} \nabla \log \pi(a_i|s_i,\theta)$$

$$\nabla \log p(s_0) + \sum_{\mathrm{i}} \nabla \log \pi(a_i|s_i,\theta) + \nabla \log p(s_{i+1}|s_i,a_i)$$

$$\nabla J(\theta) = \mathbb{E}[\Psi(s,a)\nabla \log \pi(a|s,\theta)]$$

Policy Gradient

$$\nabla J(\theta) = \mathbb{E}[\Psi(s,a)\nabla \log \pi(a|s,\theta)]$$

where Ψ_t may be one of the following:

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^{\pi}(s_t, a_t)$: state-action value function.
5. $A^{\pi}(s_t, a_t)$: advantage function.
6. $r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$: TD residual.

[Schulman et al. - High-Dimensional Continuous Control Using Generalized Advantage Estimation]

See also:
[Sutton et al. - Policy Gradient Methods for Reinforcement Learning with Function Approximation]
[Peters et al. - Reinforcement learning of motor skills with policy gradients]

Policy Gradient

$$\nabla J(\theta) = \mathbb{E}[\Psi(s,a)\nabla \log \pi(a|s,\theta)]$$

Update rule:

$$\theta \leftarrow \theta + \alpha \nabla J(\theta)$$

- Generate trajectories/rollouts while sampling from current policy
- Update $\Psi(s,a)$ [Critic]
- Compute $\nabla J(\theta) = \frac{1}{N}\sum_i \Psi(s_i,a_i)\nabla \log \pi(a_i|s_i,\theta)$
- Update θ [Actor]
- Repeat

Training Non-linear Policy with Policy Gradient

DDPG, TRPO, PPO, …

Pros:

- Significantly more robust

Cons:

- Hard to tune training parameters
- Hard to estimate training cost
- Blending between networks?

Training Non-linear Policy with Policy Gradient

DDPG, TRPO, PPO, …

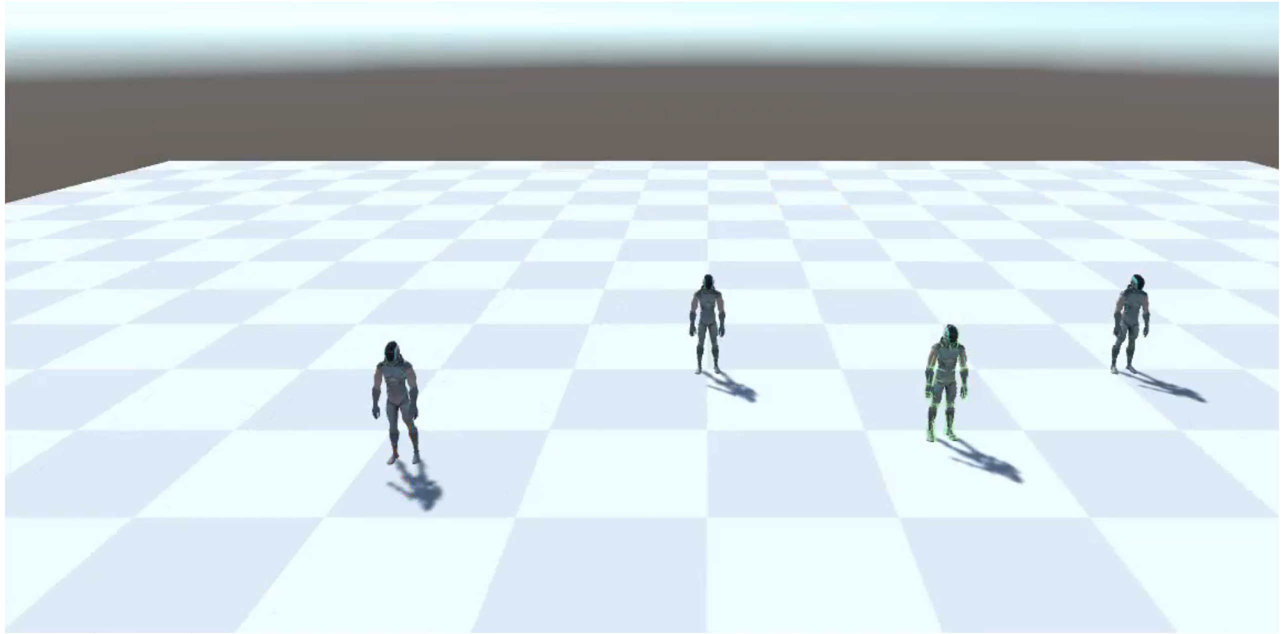
Pros:

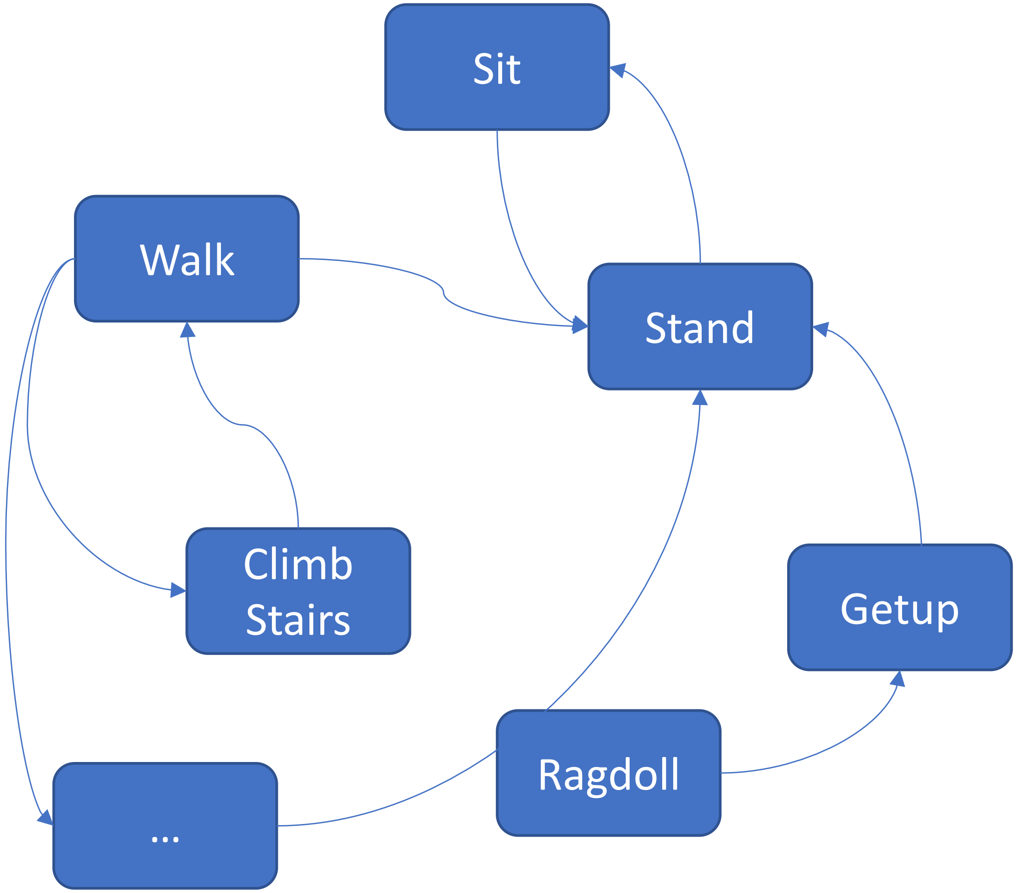
- Significantly more robust

Cons:

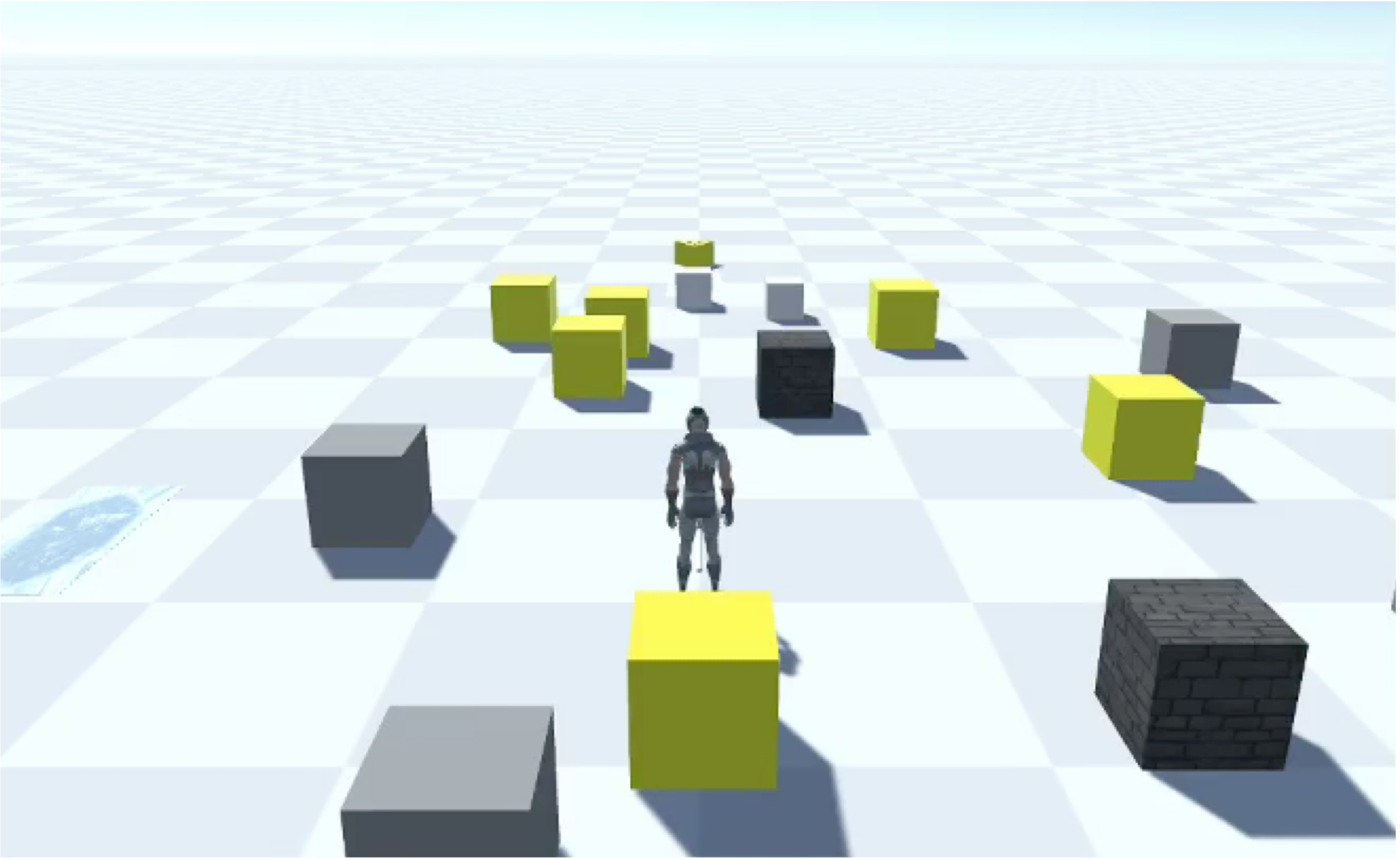
- Hard to tune training parameters
- Hard to estimate training cost
- Blending between networks?

Application – Control Graph





Application – Control Graph



Application – Basketball

Hand Control

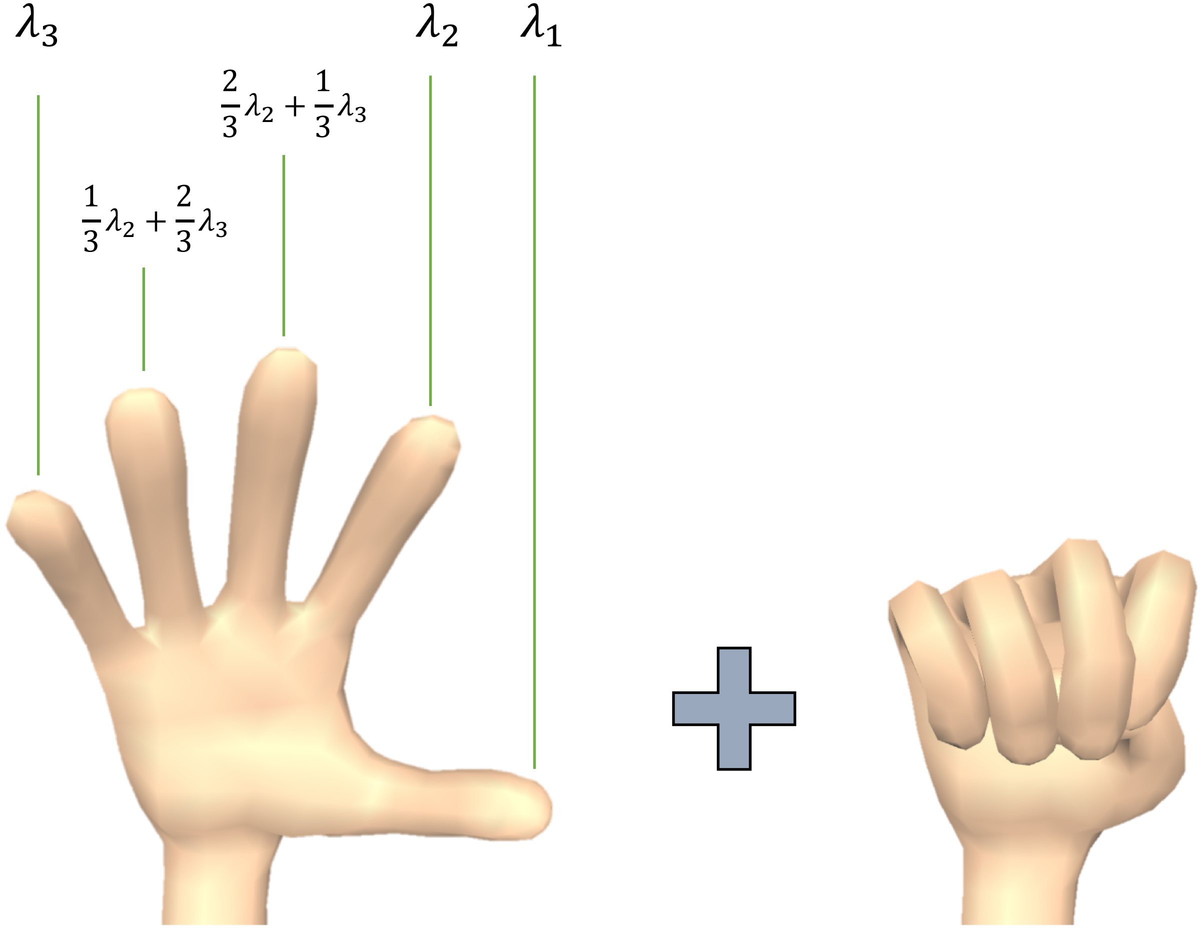
Target Pose for PD-Control

Interpolate between two example poses

$$\theta = (1-\lambda)\theta^{\text{flat}} + \lambda\theta^{\text{fist}}$$

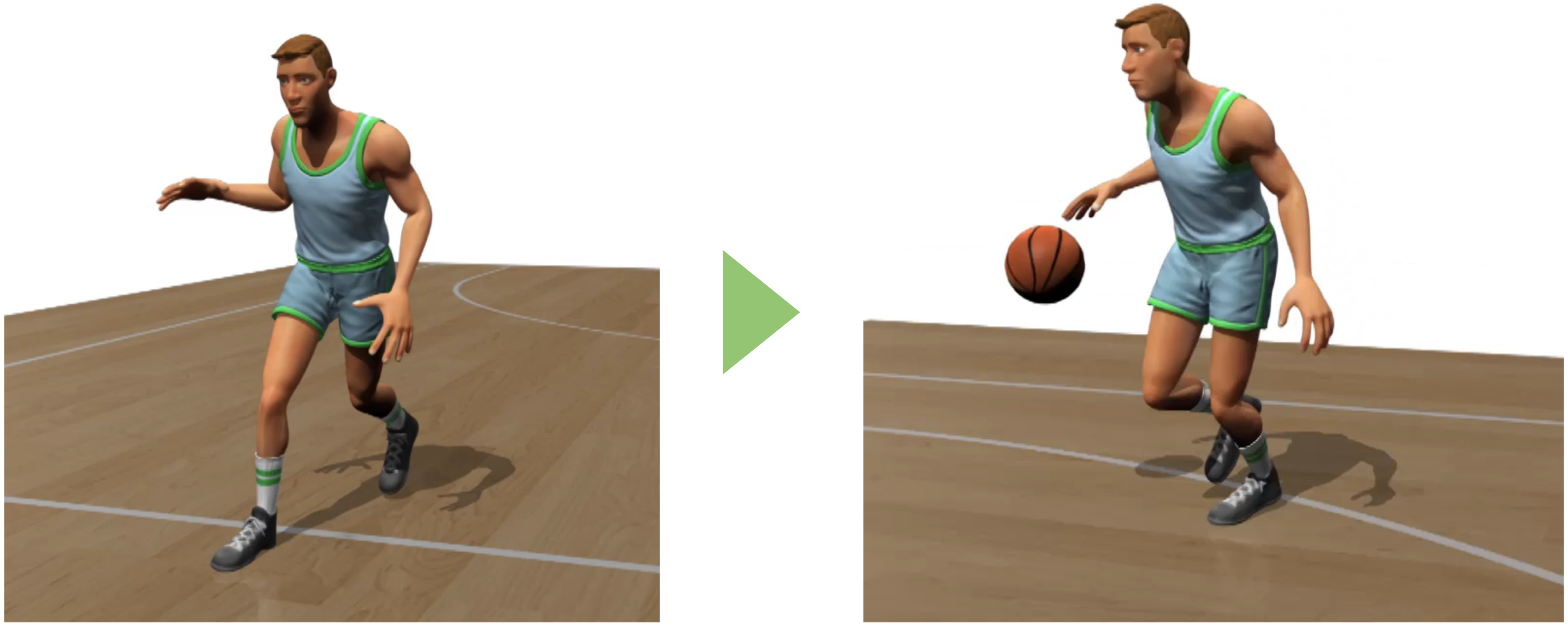
Three interpolation factors for fingers

{ λ_1: Thumb, λ_2: Index, λ_3: Pinky }



Trajectory Optimization

Recover ball movement



Trajectory Optimization

Optimization Problem

Minimize distance between ball and fingertips

User-specified frame range

Optimization Variables

$$\left(\boldsymbol{q}_{\text{shoulder}},\ q_{\text{elbow}},\ \boldsymbol{q}_{\text{wrist}},\ \alpha_{\text{fingers}} = [\lambda_1, \lambda_2, \lambda_3]\right)_K^{\text{left\textbackslash right}}$$

CMA-ES

Linear Policy Works Sometimes

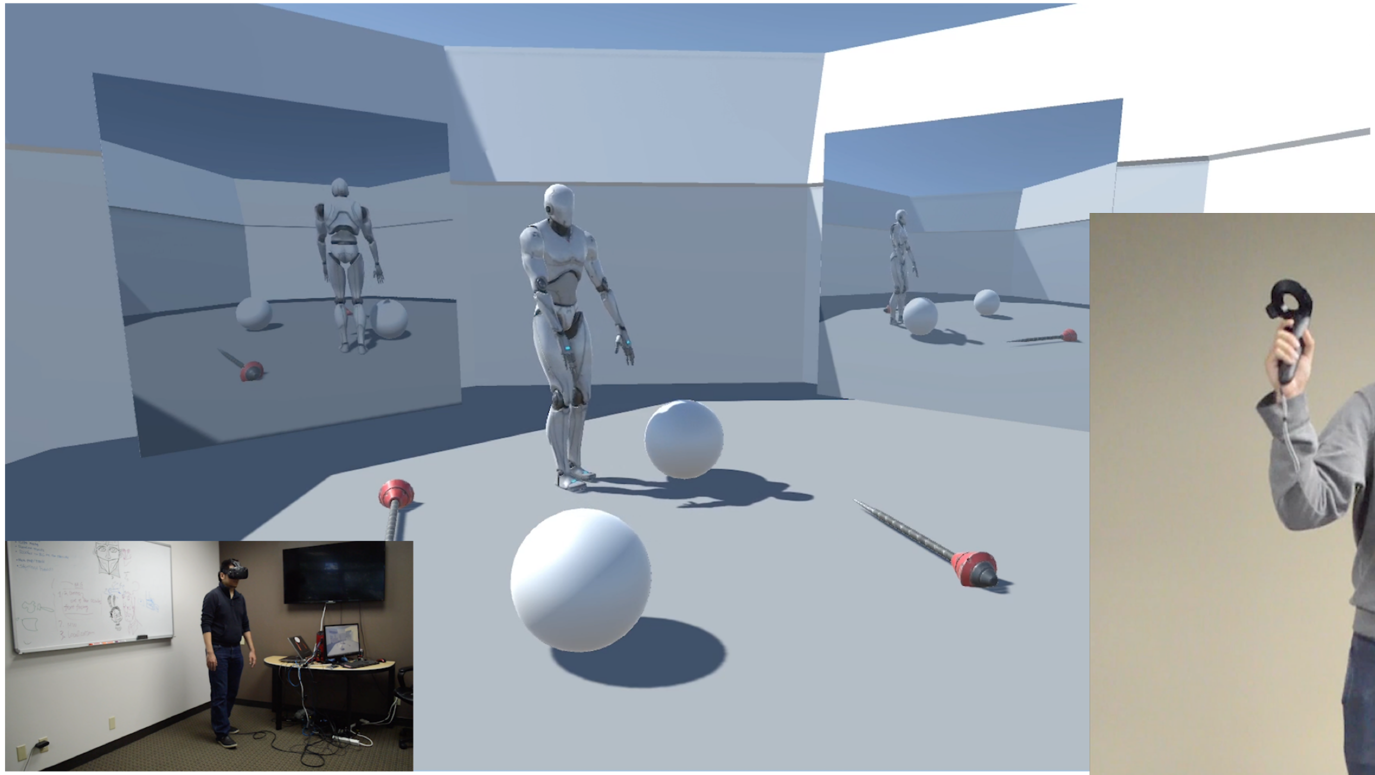
Trajectory Optimization → Linear Regression

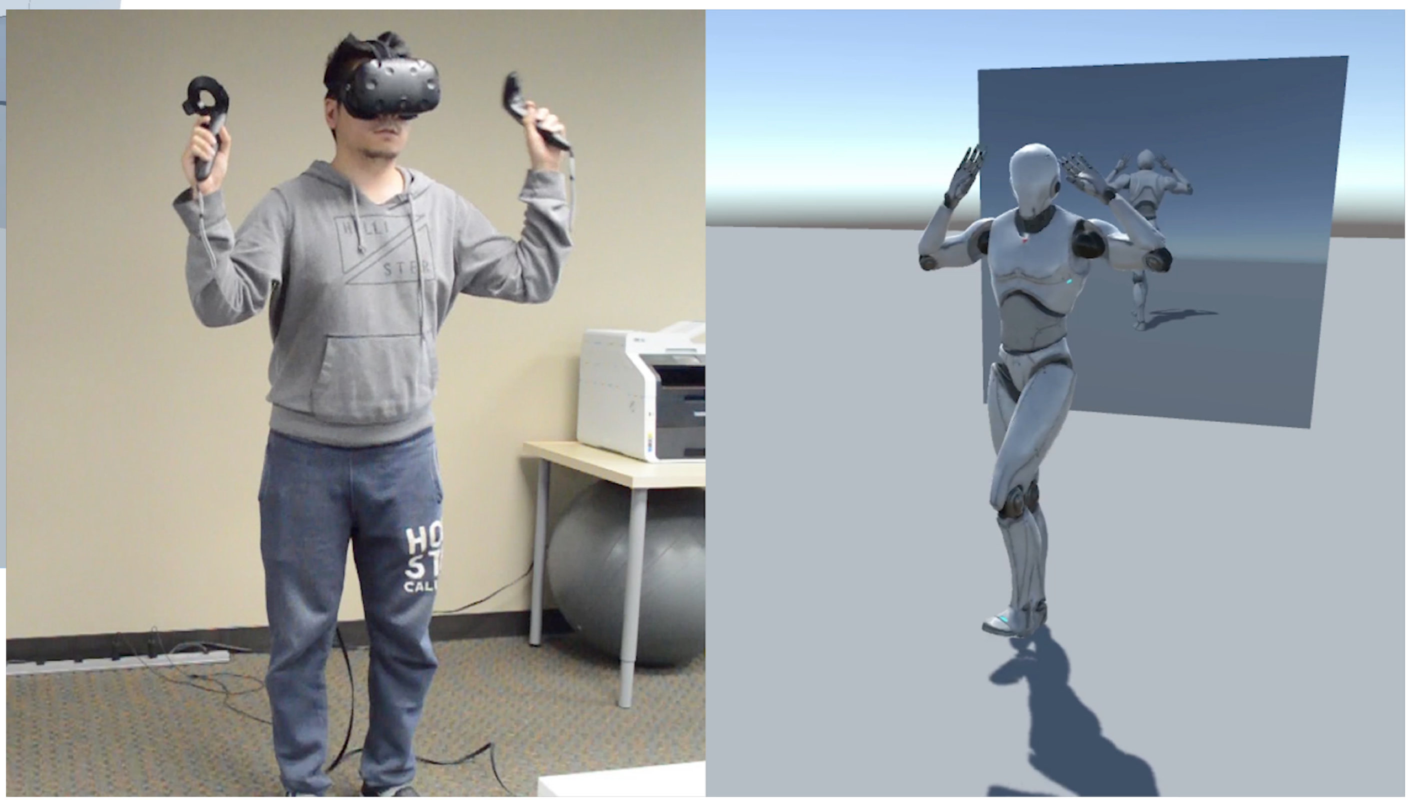
Deep Reinforcement Learning

Training

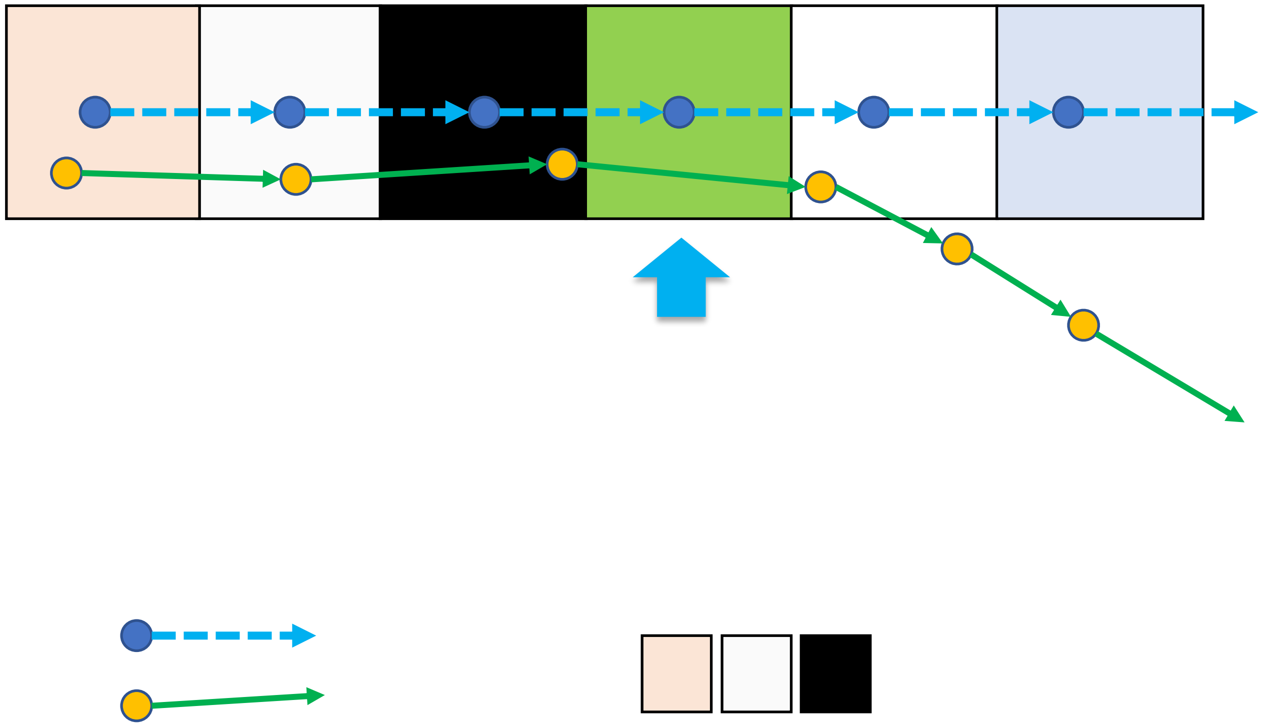
- Rollouts start from the state from trajectory optimization
- Stop immediately when the ball is out of reach
- Warm-start training:
  - DDPG: Linear policy is used for the first 20% of the replay buffer
  - Regularize KL-divergence to linear policy?
  - Policy distillation?

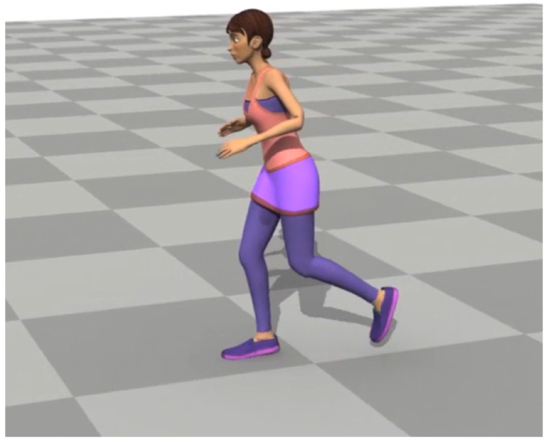
Application?
– Reconstruct Full-body Motion from a few Sensors



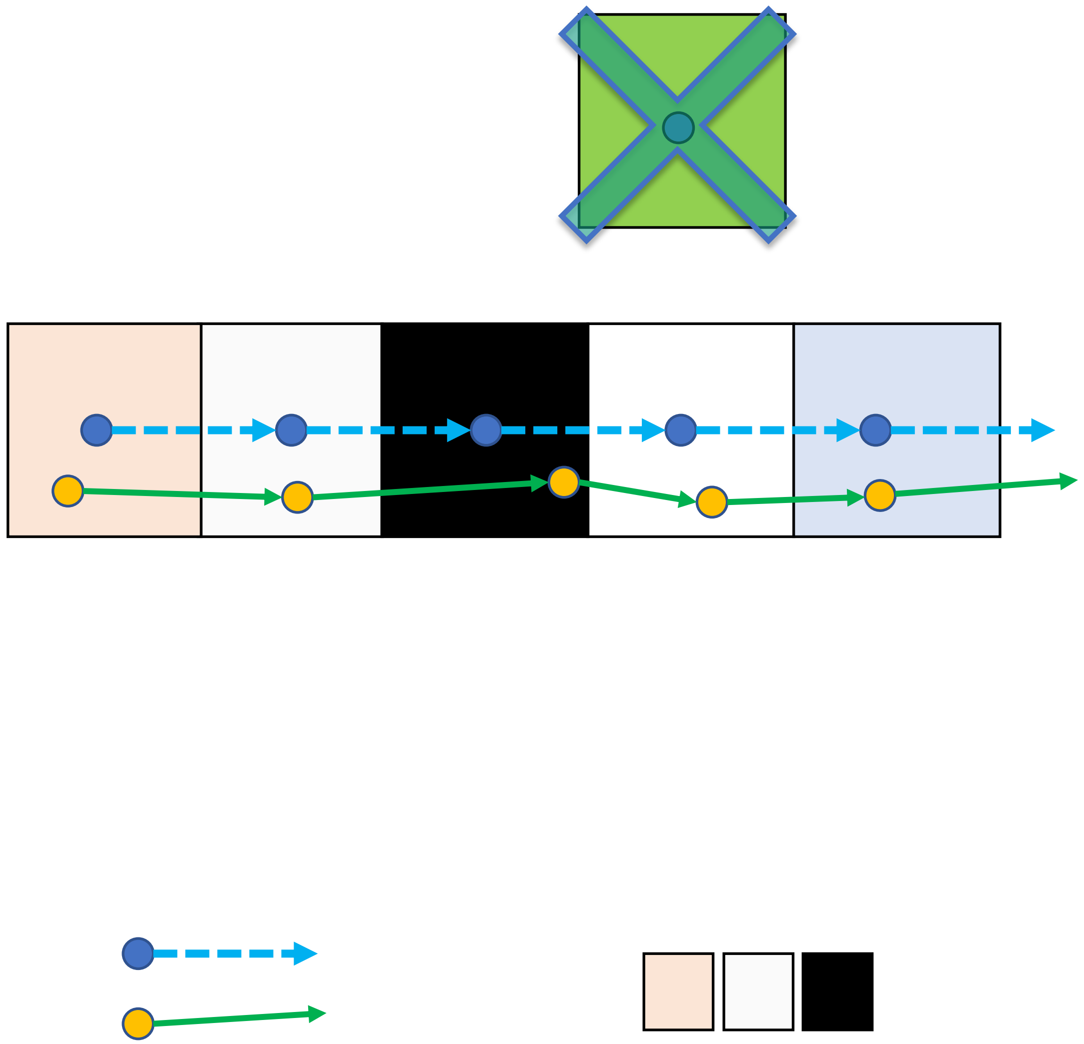


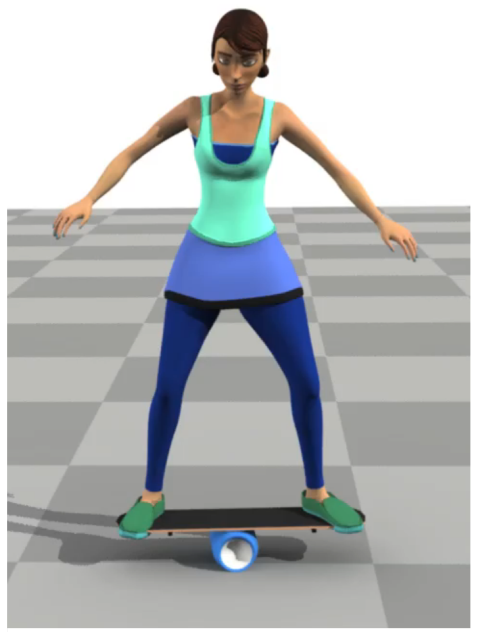
Problem of Fixed Time-Indexed Tracking

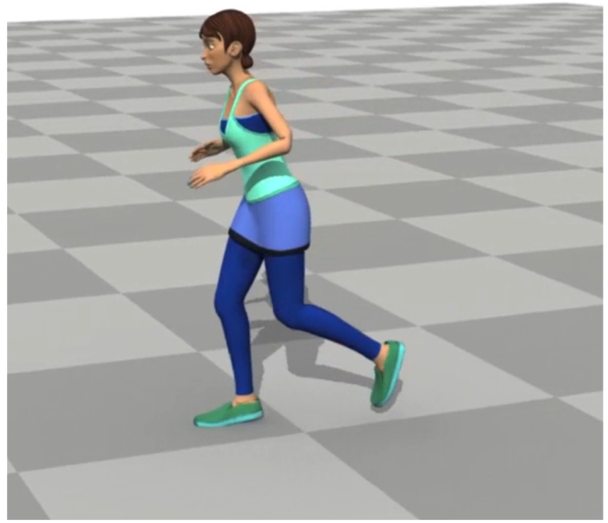




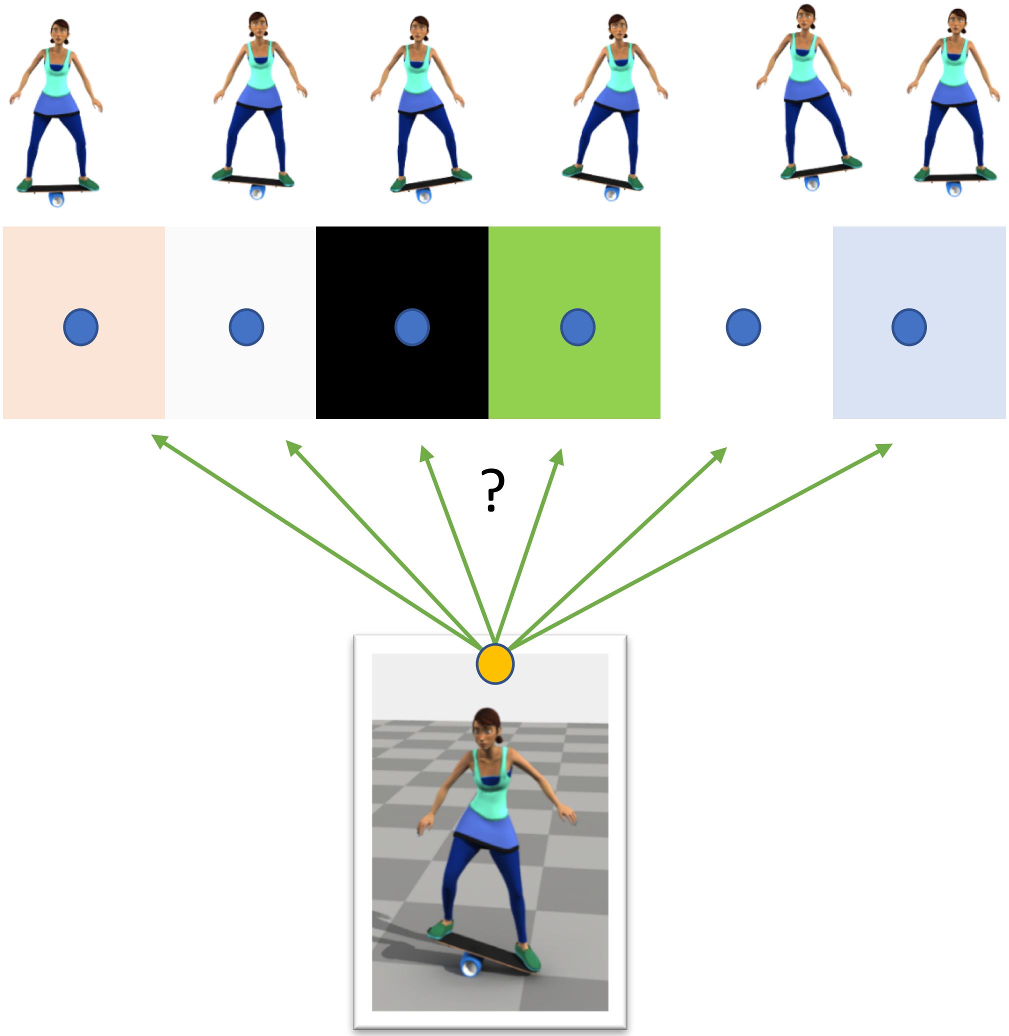
Scheduling







Scheduler



Q-Function

Return:

$$R(\tau) = r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 r(s_2, a_2) + \cdots$$

The expected return

$$\mathbb{E}[R(\tau)] = \int_{s_0} p(s_0) \int_{a_0} \pi(a_0|s_0) r(s0, a0) + \gamma \int_{s_1} p(s_1|s_0, a_0) \int_{a_1} \pi(a_1|s_1) r(s_1, a_1) + \cdots$$

$$Q(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots |s_t, a_t, \pi]$$

Q-Function

The optimal policy π^*

$$Q^*(s_t, a_t) = \max_{\pi} \mathbb{E}[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots | s_t, a_t, \pi]$$

The optimal action $a = \pi^*(s)$ can be found by solving

$$a = \arg\max_{a} Q^*(s, a)$$

Only tractable with discrete actions

Q-Learning

Bellman equation

$$Q^*(s,a) = \mathbb{E}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a')\right]$$

Q-learning:

- Generate rollouts (s,a,s') according to $Q(s,a)$
- Update $Q(s,a) \leftarrow r(s,a) + \gamma \max_{a'} Q(s',a'|\theta_0)$

Deep Q-Learning

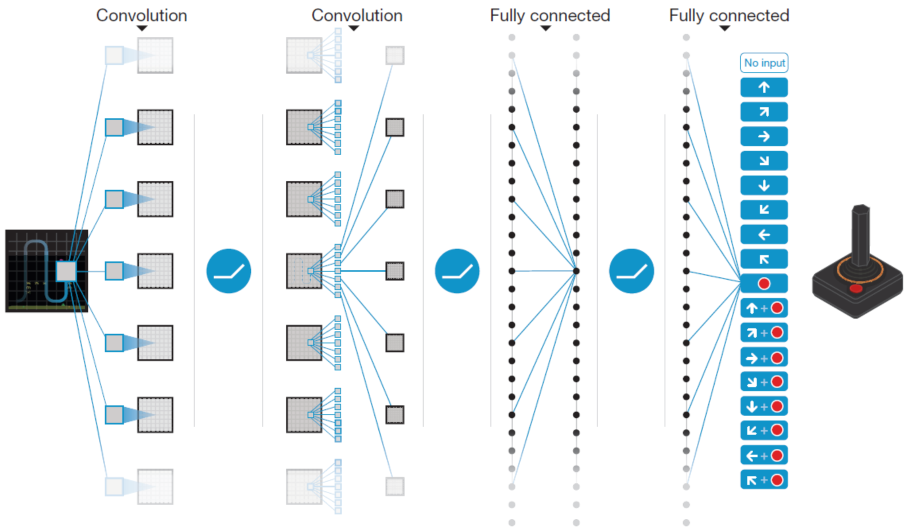
Learn to perform good actions

Raw image input

Deep convolutional network

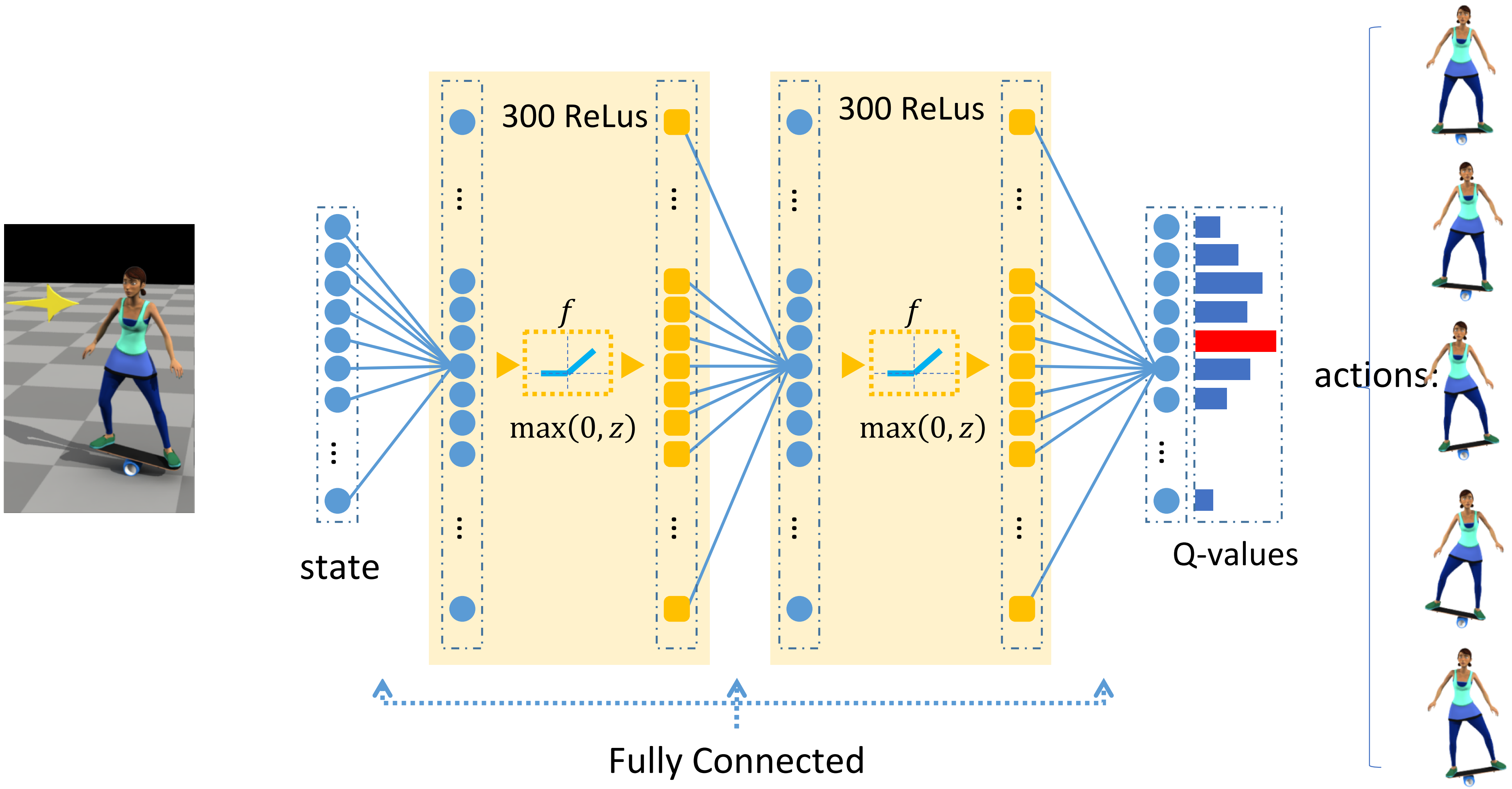
Objective function:

$$L(\theta) = ||r(s,a) + \gamma \max_{a'} Q(s', a'|\theta_0) - Q(s,a|\theta)||_2$$



[Mnih et al. 2015, Human-level control through deep reinforcement learning]

A Q-Network For Scheduling



Conclusion

Reference motion → Tracking control

- Good motion quality
- Linear policy works for a large range of motion
- Non-linear policy is preferred for better robustness

Graph of tracking controllers

Arm/upper body motion + balance control

Scheduling tracking control fragments

- May be necessary for some motion
- Good robustness and response to interaction
- Bad quality when jumping between fragments too frequently

Correct Response?

Unstructured Input