IMITATION LEARNING

Scott Niekum

Assistant Professor, Department of Computer Science

The University of Texas at Austin

TEXAS

The University of Texas at Austin

PeARL

Personal Autonomous Robotics Lab

Imitation learning

Part 1: Modes of input

Introduction

Sensing

Modes of input

Introduction: Why learn from demonstration?

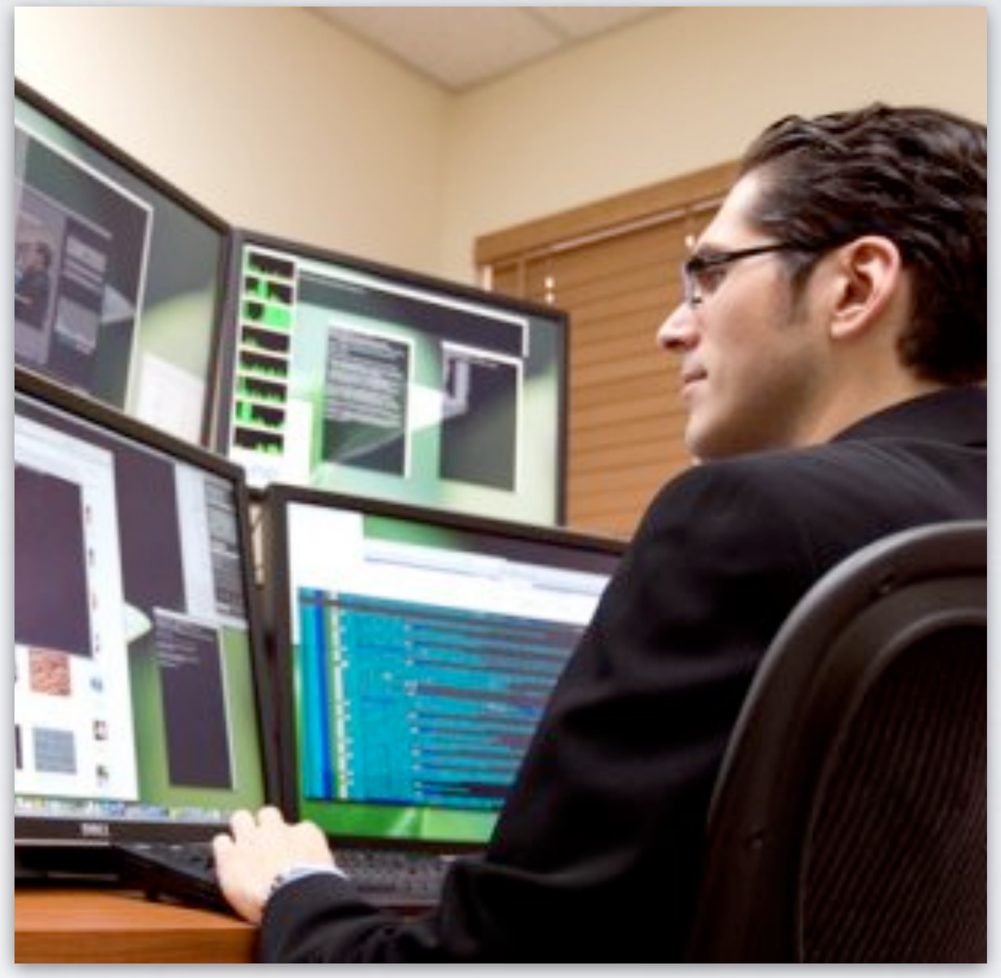
Introduction: Why learn from demonstration?

General purpose robot

Introduction: Why learn from demonstration?

General purpose robot → Specific task

Introduction: Why learn from demonstration?



General purpose robot → Specific task → Expert engineer

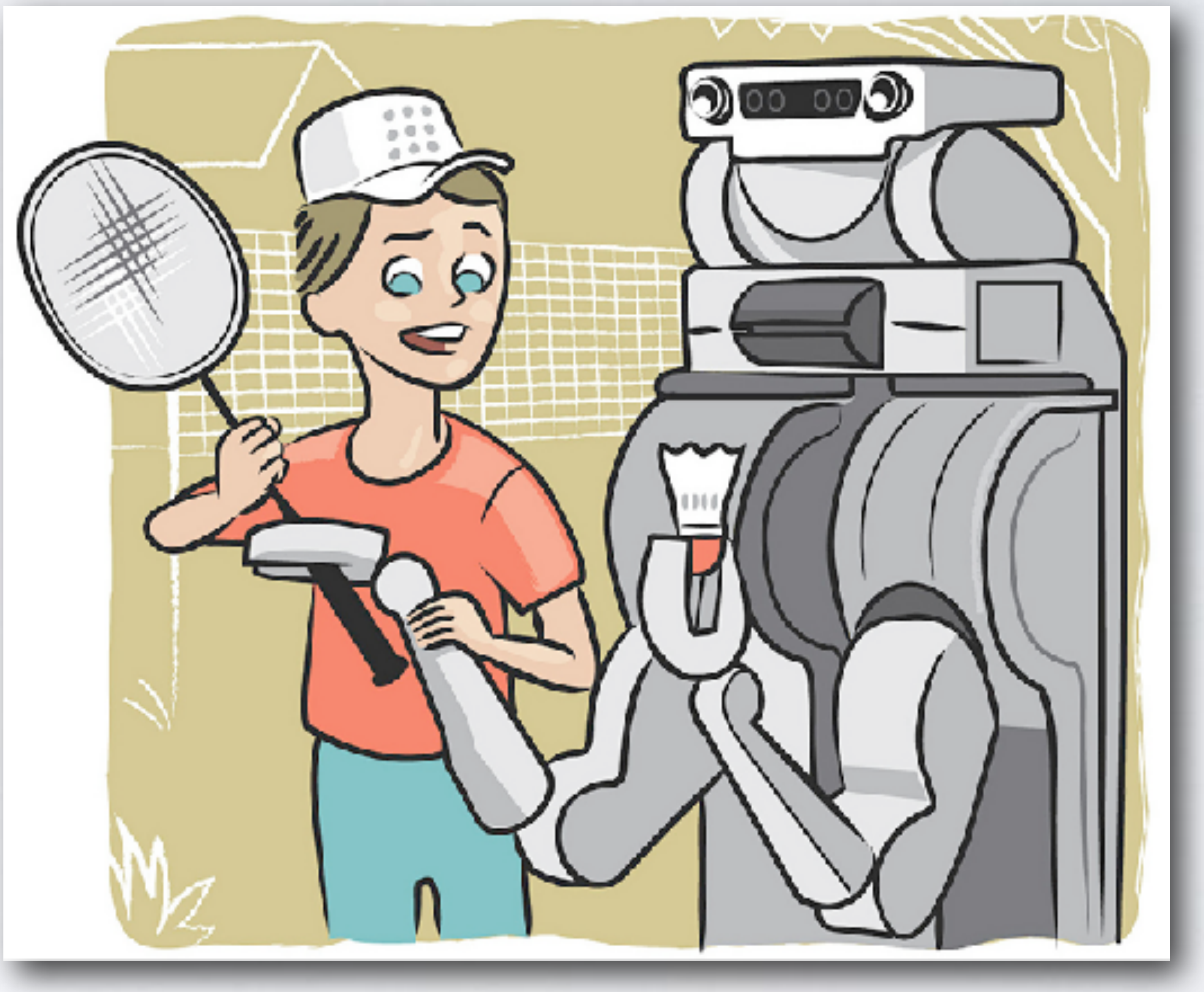
Introduction: Why learn from demonstration?

Programming robots is hard!

- Huge number of possible tasks
- Unique environmental demands
- Tasks difficult to describe formally
- Expert engineering impractical

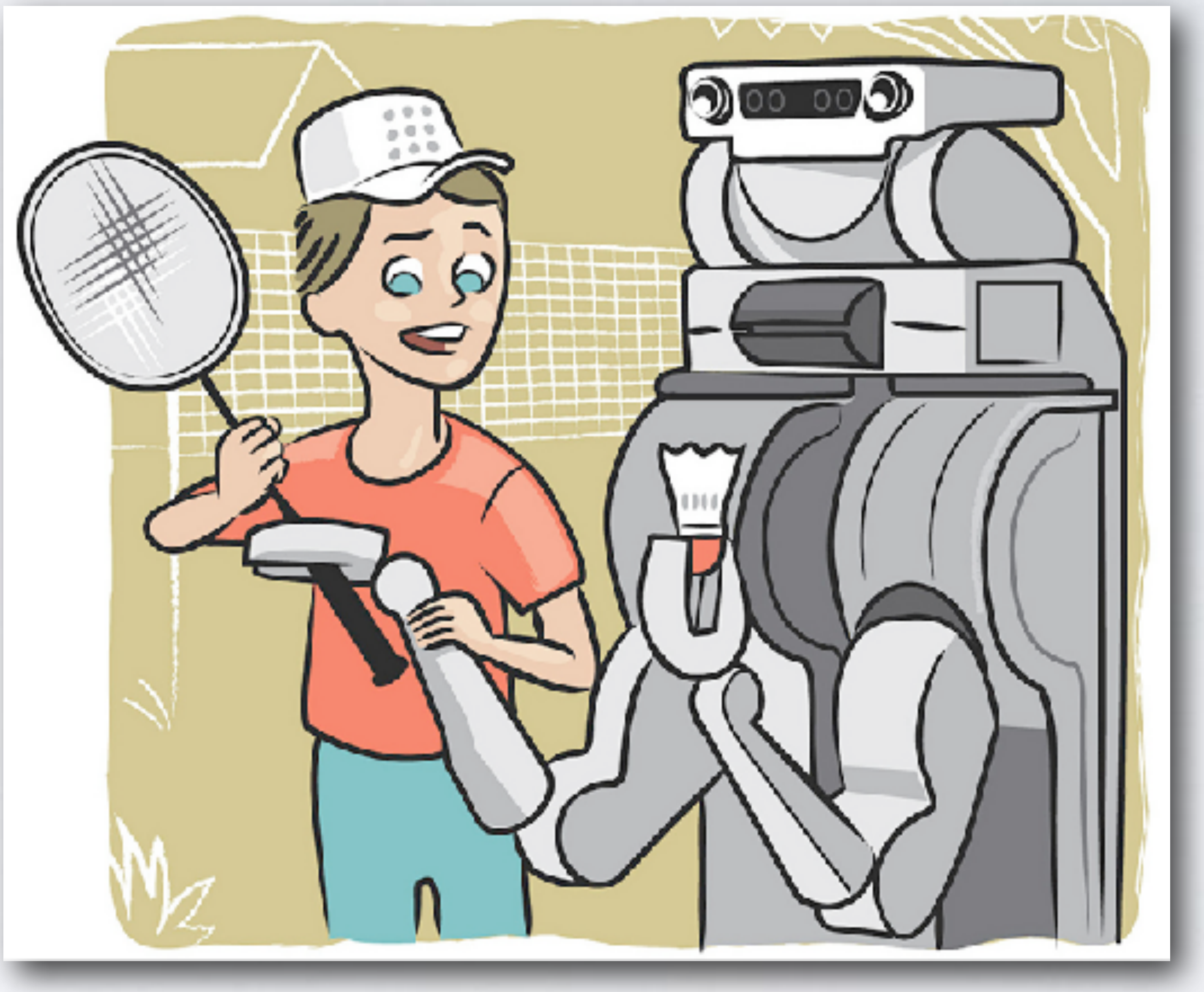
Introduction: Why learn from demonstration?

- Natural, expressive way to program
- No expert knowledge required
- Valuable human intuition
- Program new tasks as-needed



Introduction: Why learn from demonstration?

- Natural, expressive way to program
- No expert knowledge required
- Valuable human intuition
- Program new tasks as-needed



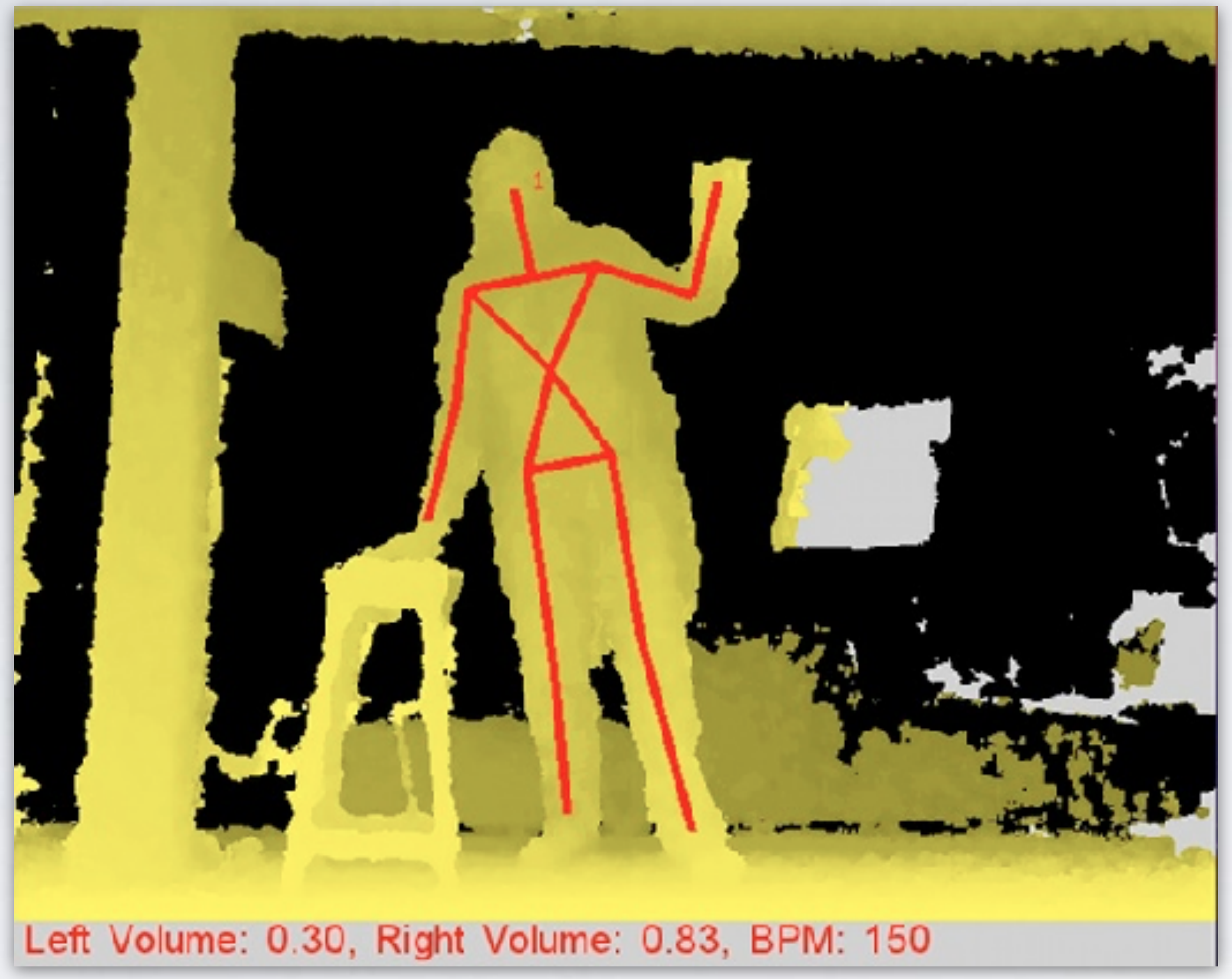
How can robots be shown how to perform tasks?

Introduction

Sensing

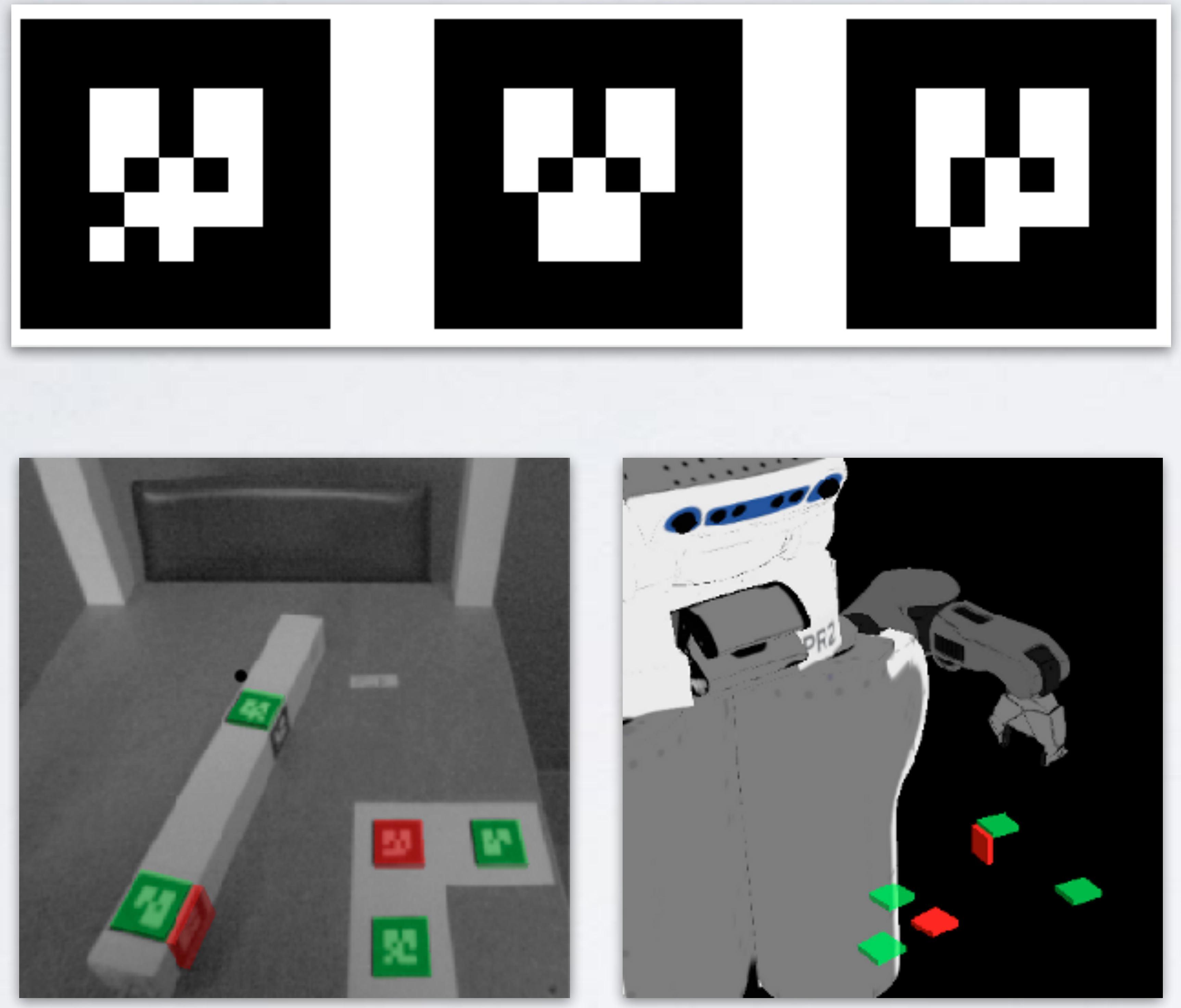
Modes of input

Sensing: RGB(D) cameras, depth sensors



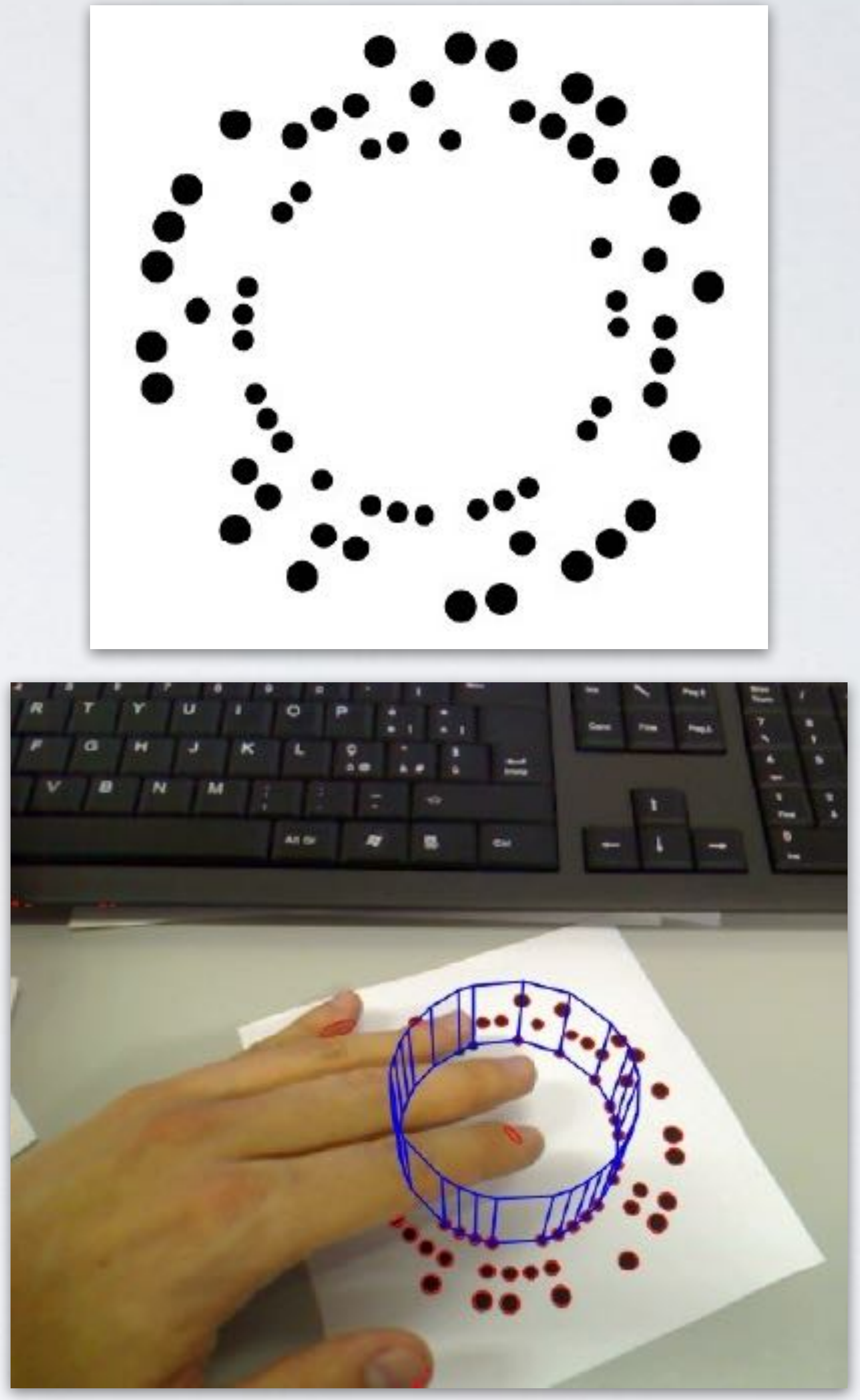
- Standard RGB cameras
- Stereo: Bumblebee
- RGB-D: Microsoft Kinect
- Time of flight: Swiss Ranger
- LIDAR: SICK

Sensing: Visual fiducials



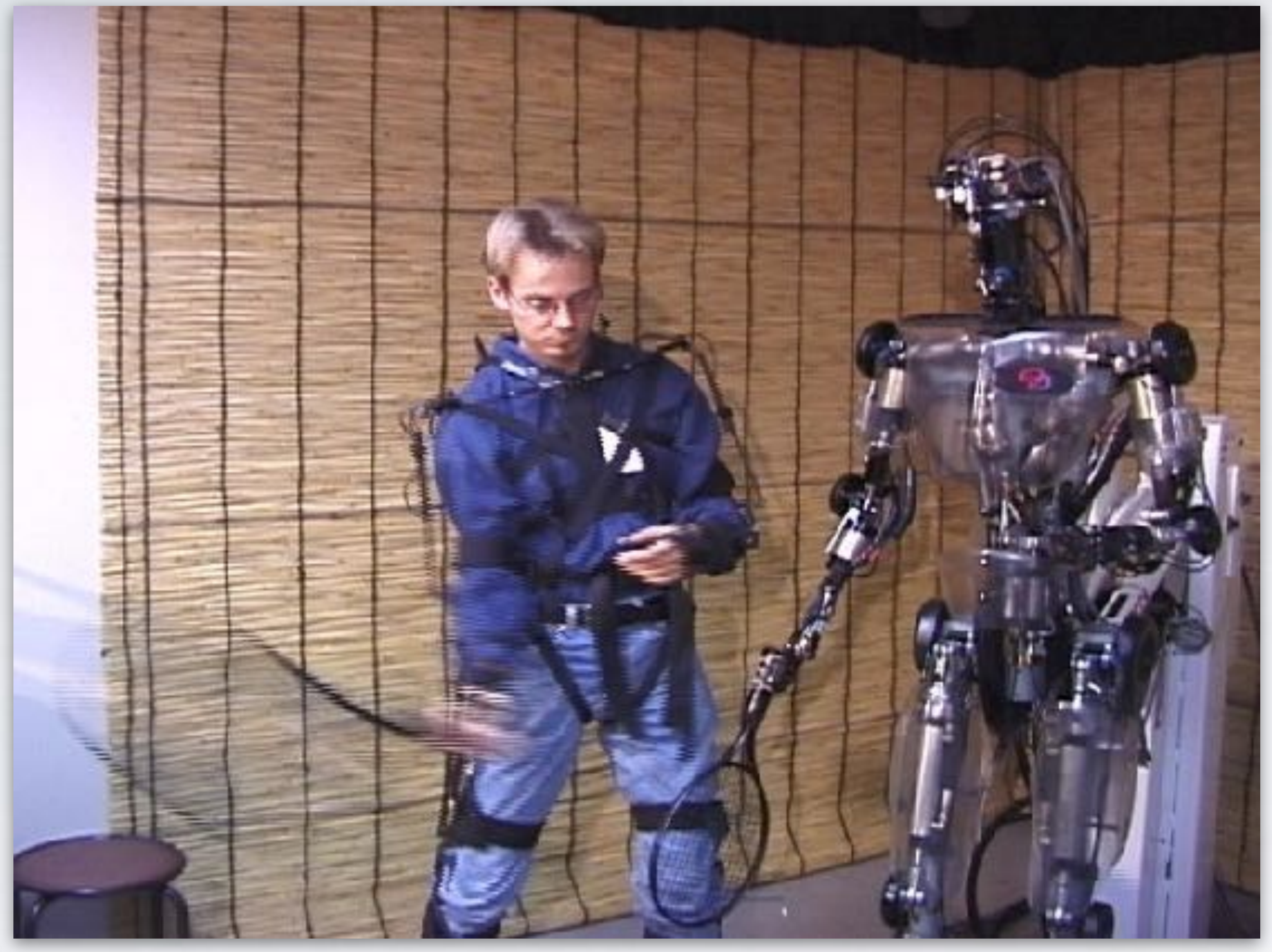
AR tags

http://wiki.ros.org/ar_track_alvar



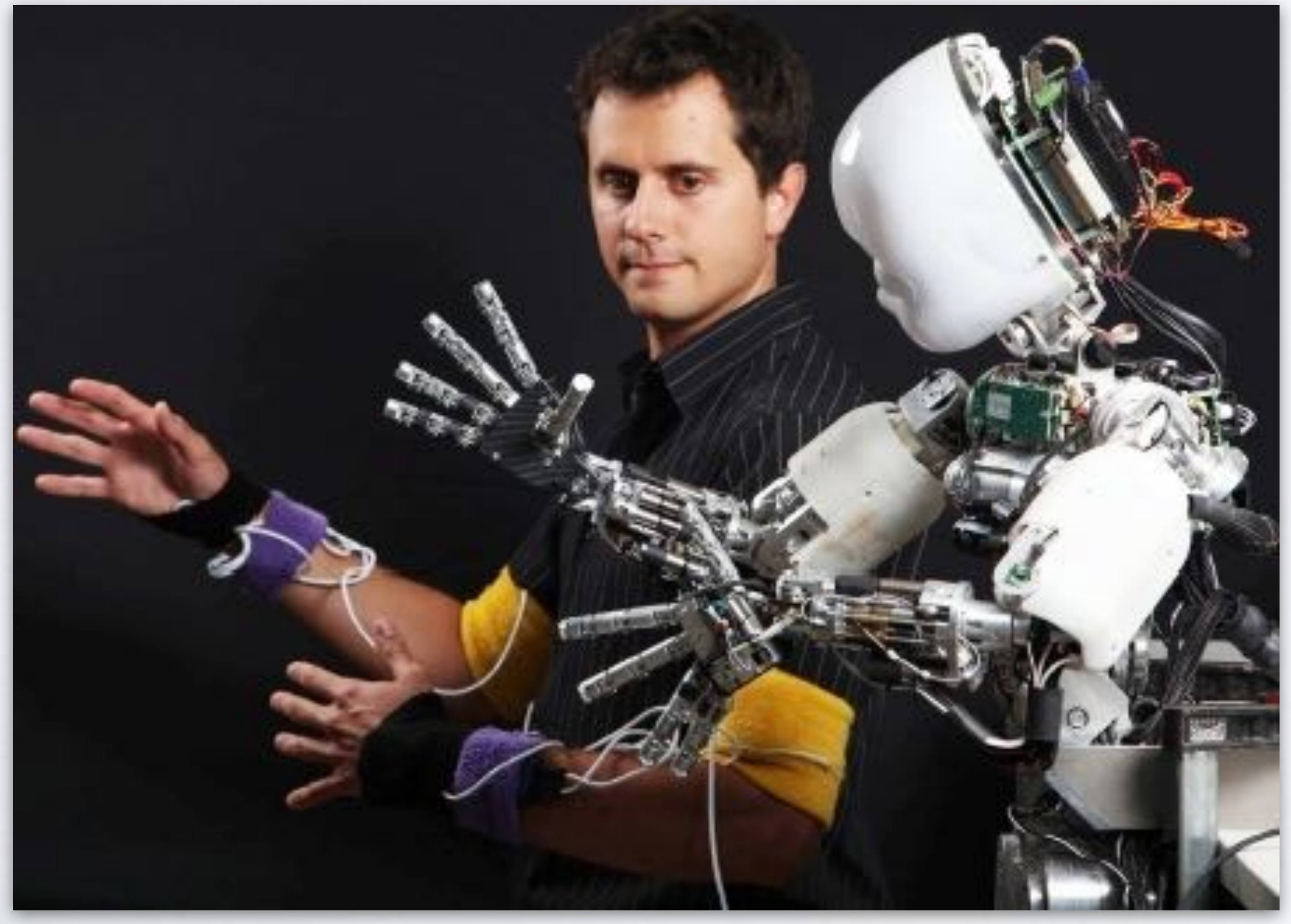
RUNE-129 tags

Sensing: Wearable sensors



SARCOS Sensuit:

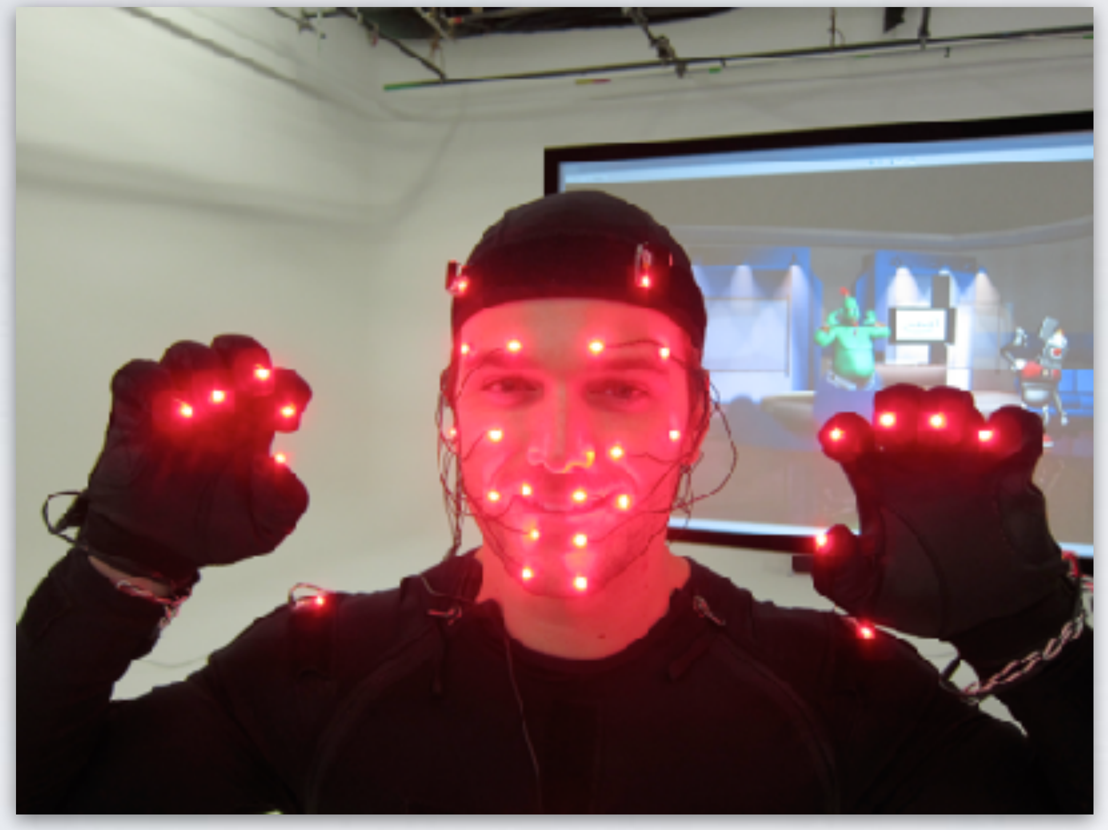
Record 35-DOF poses
at 100 Hz



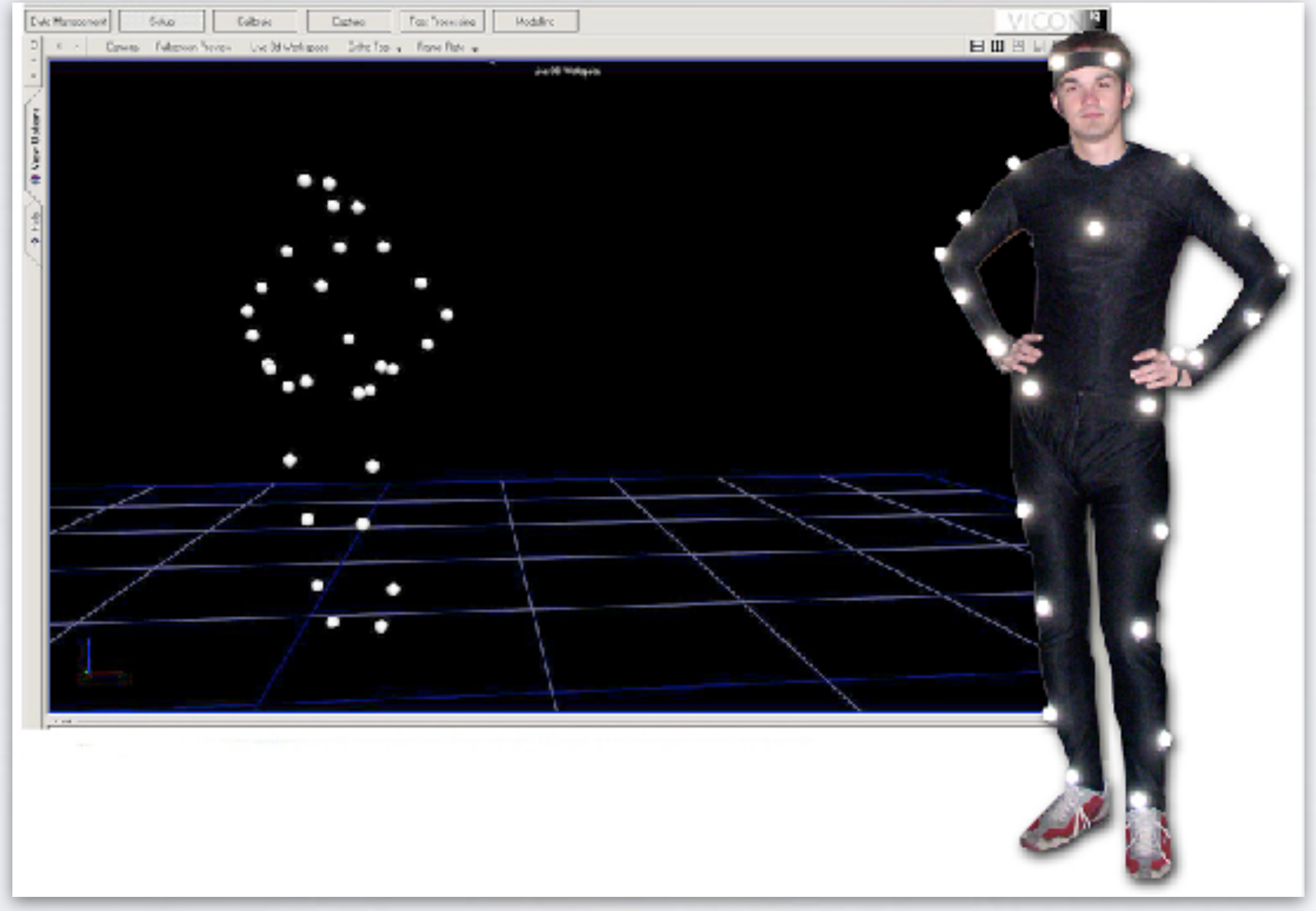
Other wearables:

- Accelerometers
- Pressure sensors
- First-person video

Sensing: Motion capture



Phasespace



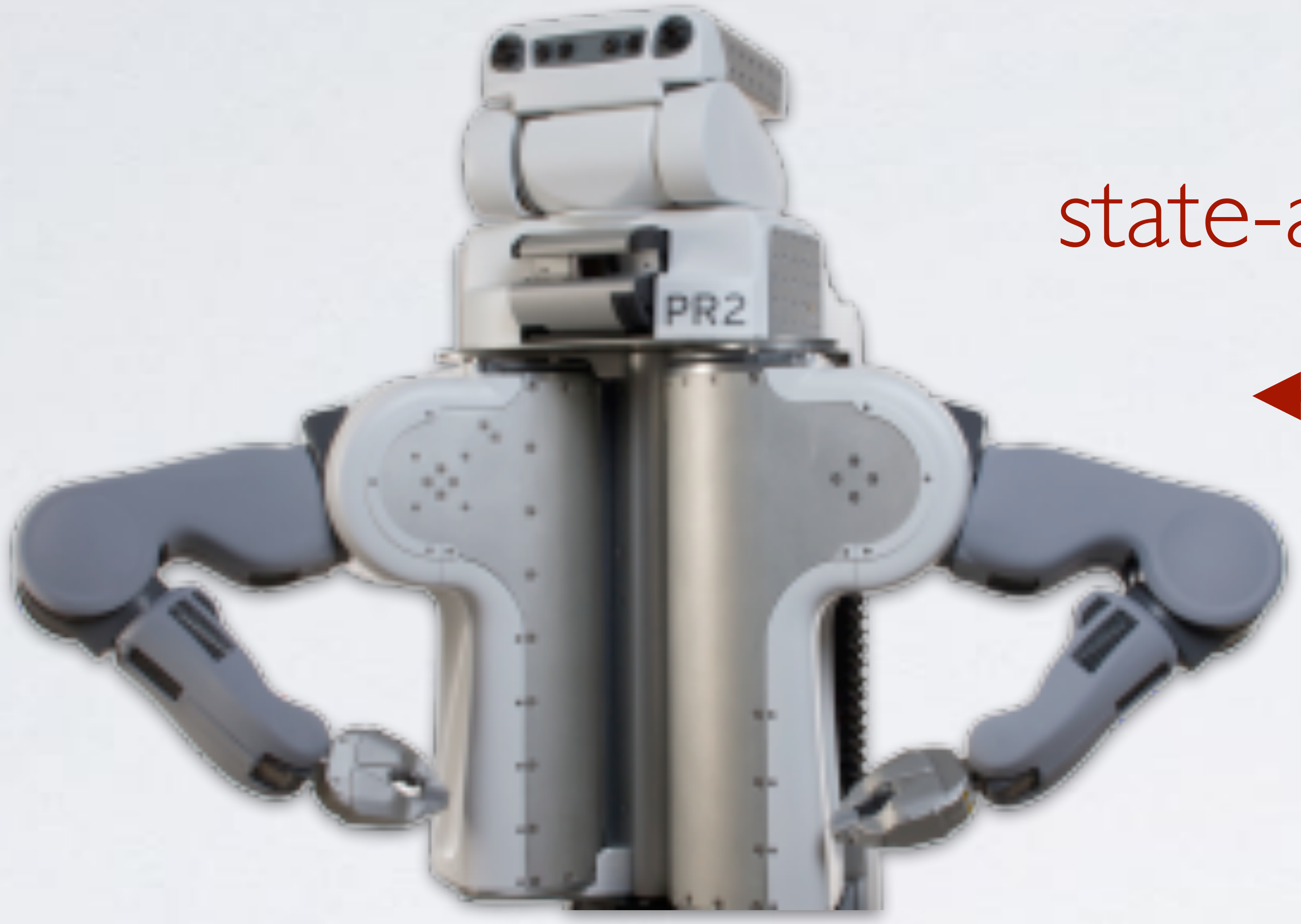
Vicon

Introduction

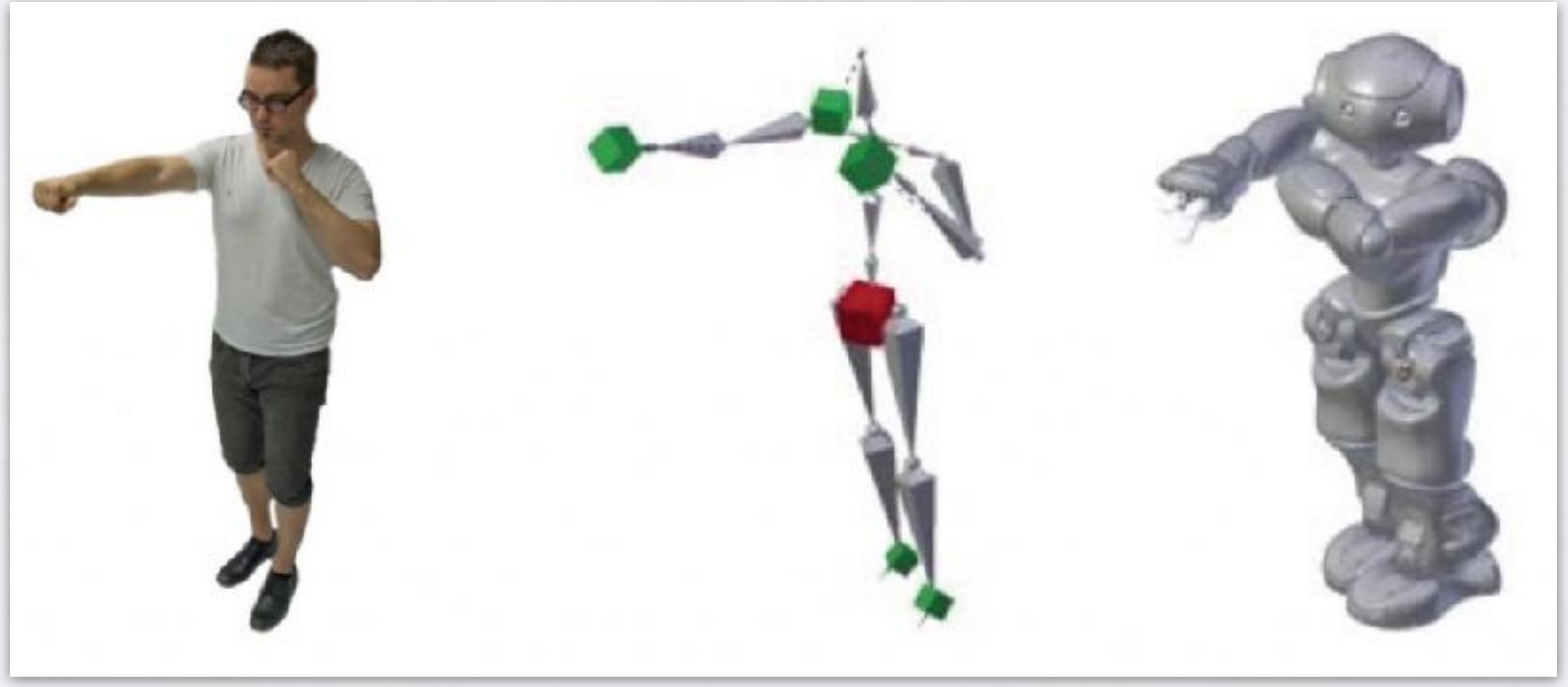
Sensing

Modes of input

The correspondence problem



state-action mapping?



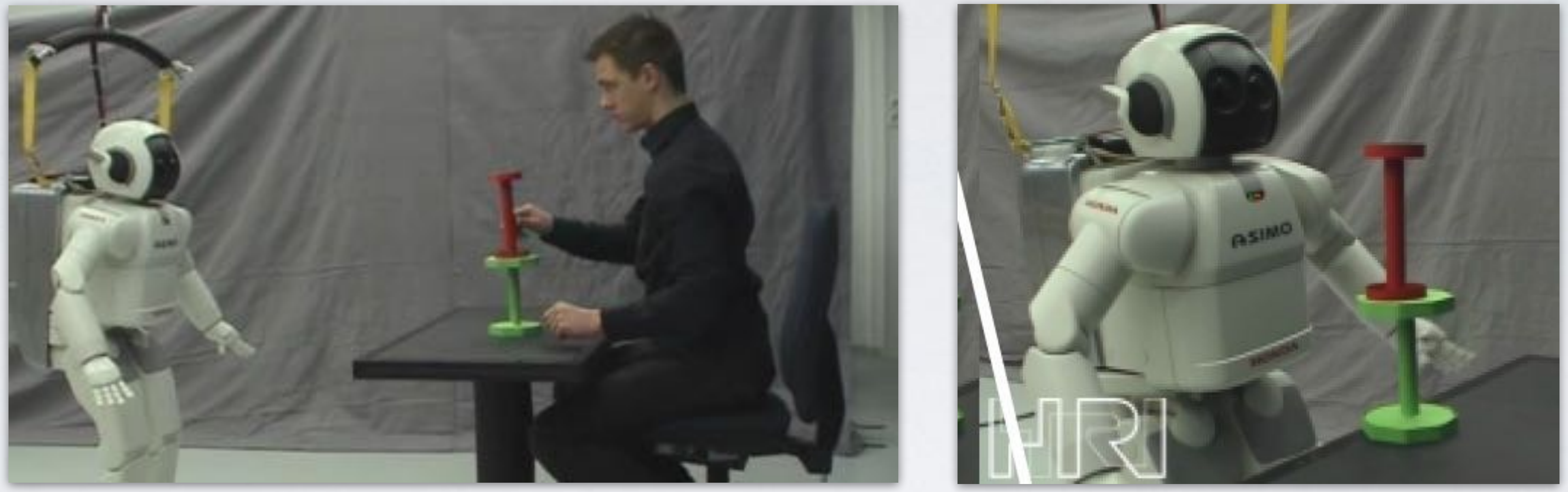
The correspondence problem

How to provide demonstrations?
Two primary modes of input:

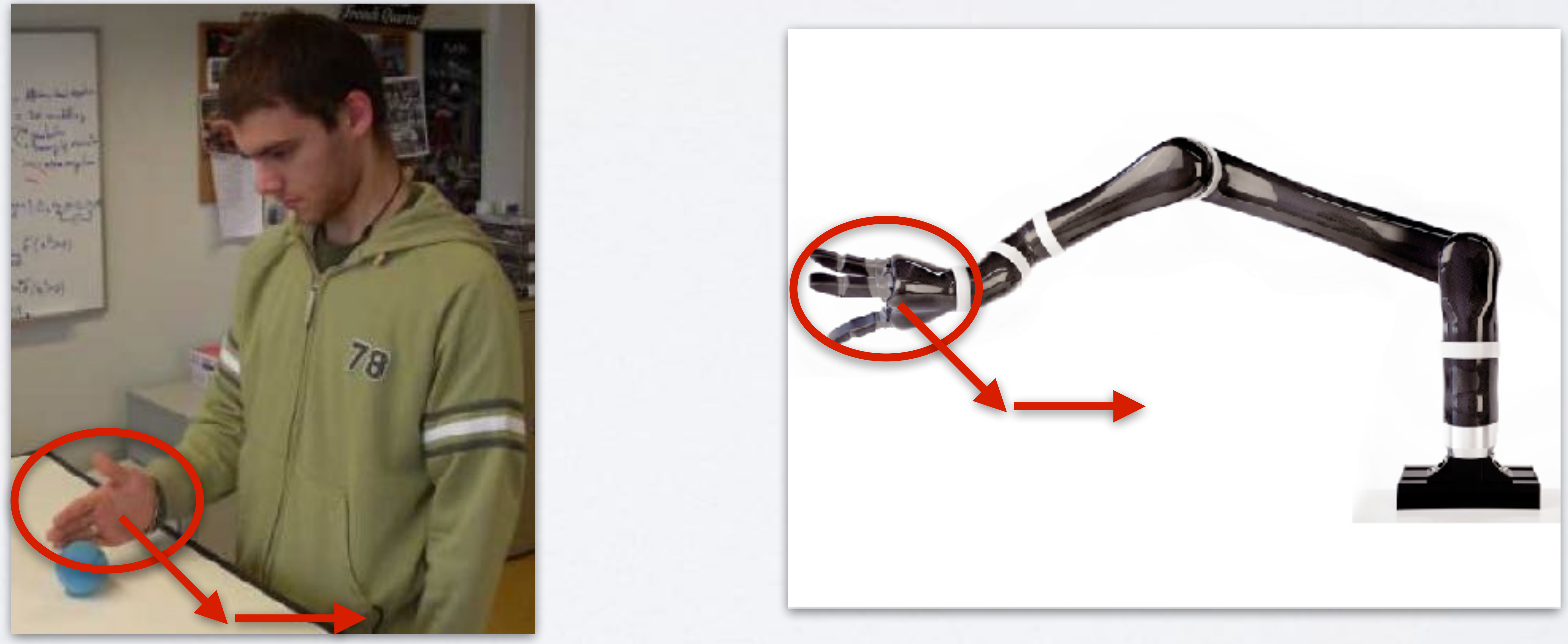
Learning by watching: Define / learn a correspondence

Learning by doing: Avoid correspondence entirely

Learning by watching: Simplified mimicry

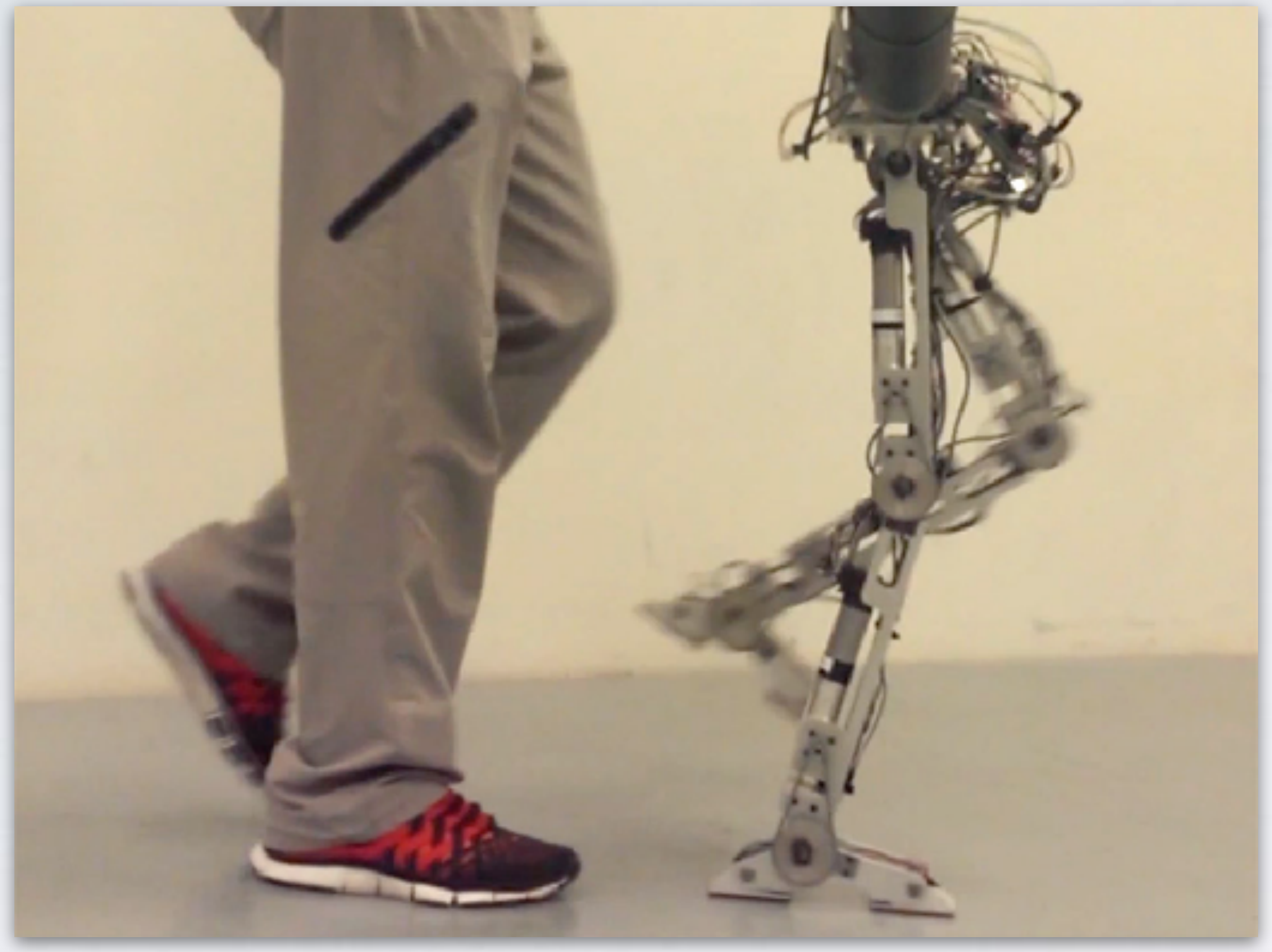


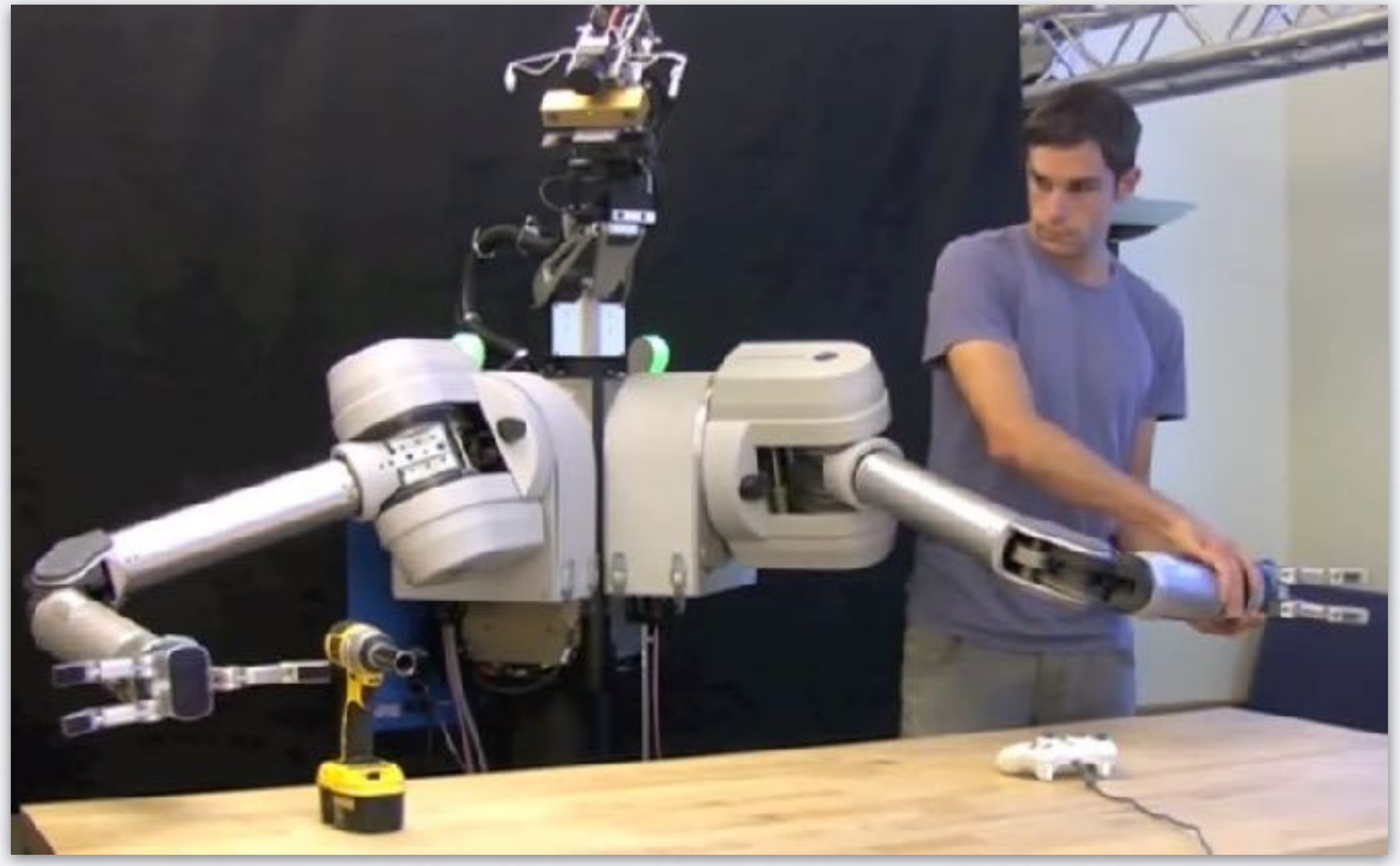
Object-based



End effector-based

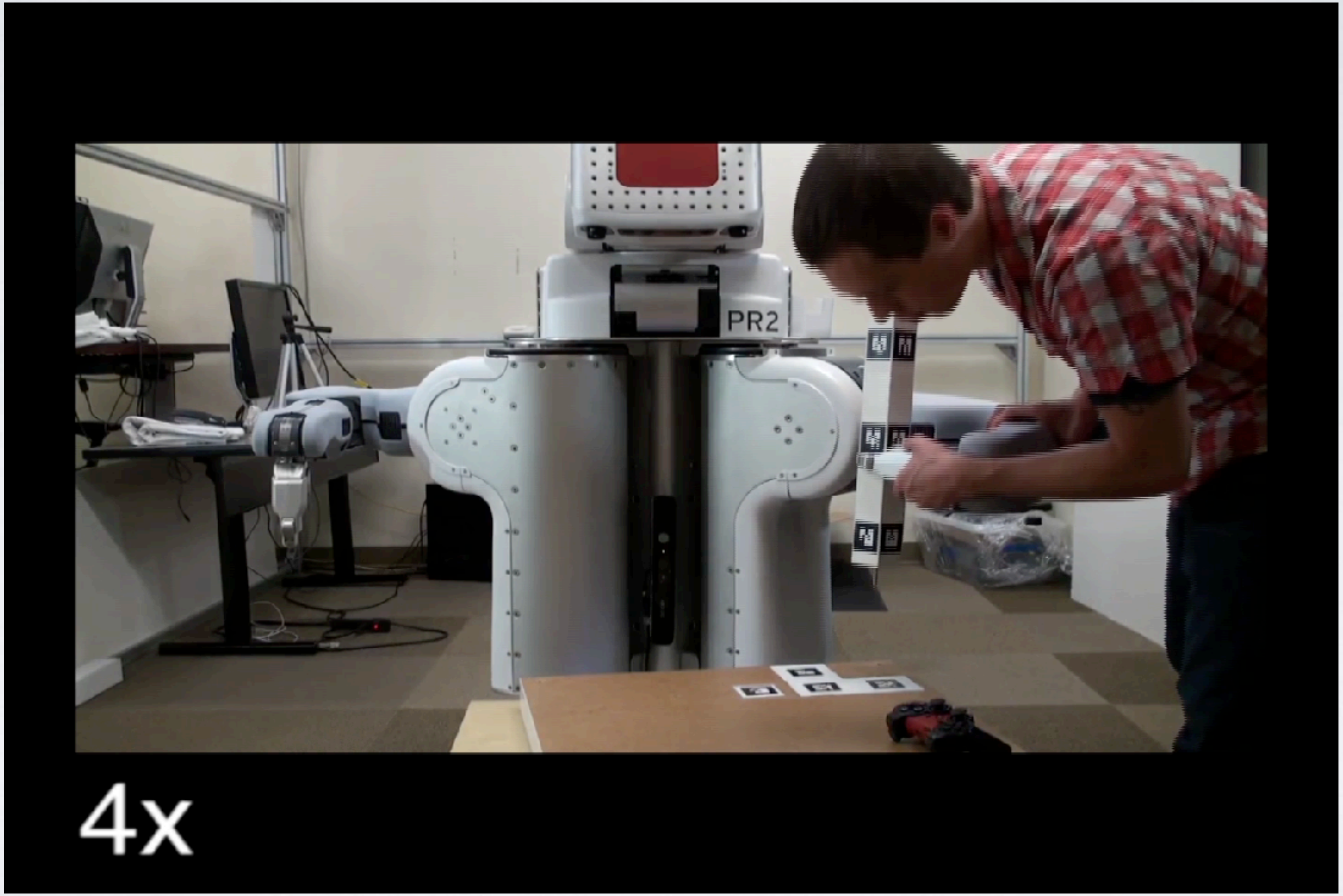
Learning by watching: Shadowing



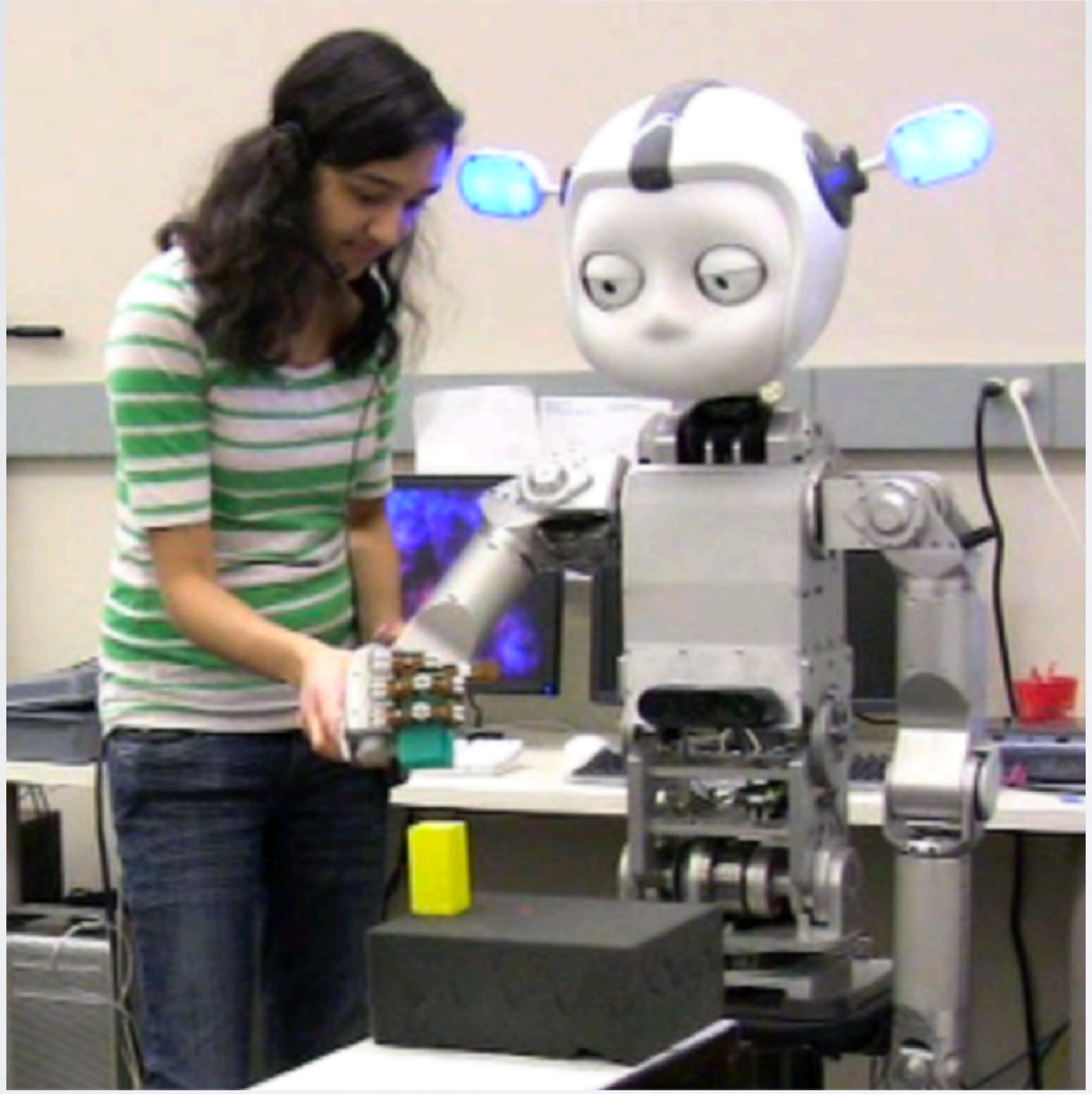


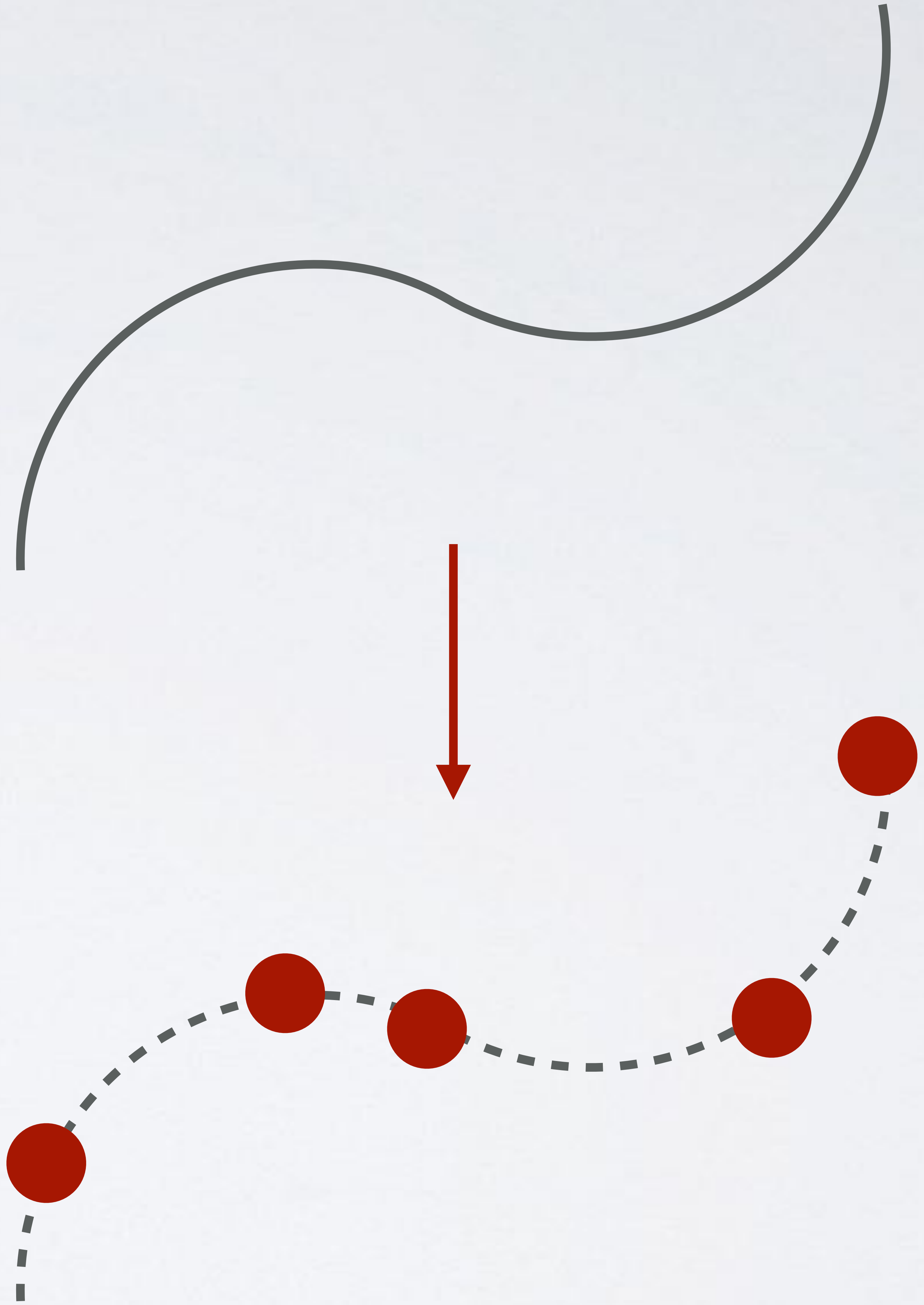
Learning by doing: Teleoperation

Learning by doing: Kinesthetic demonstration



Learning by doing: Keyframe demonstration

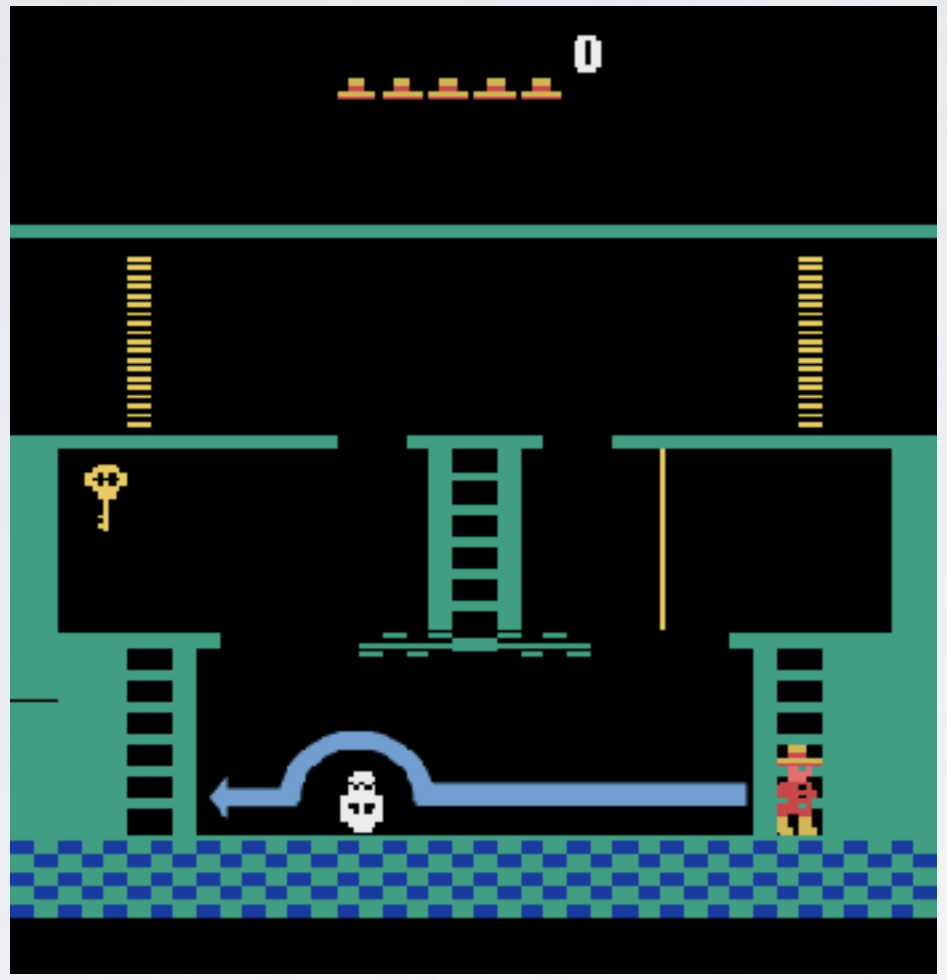




[Akgun et al. 2012]

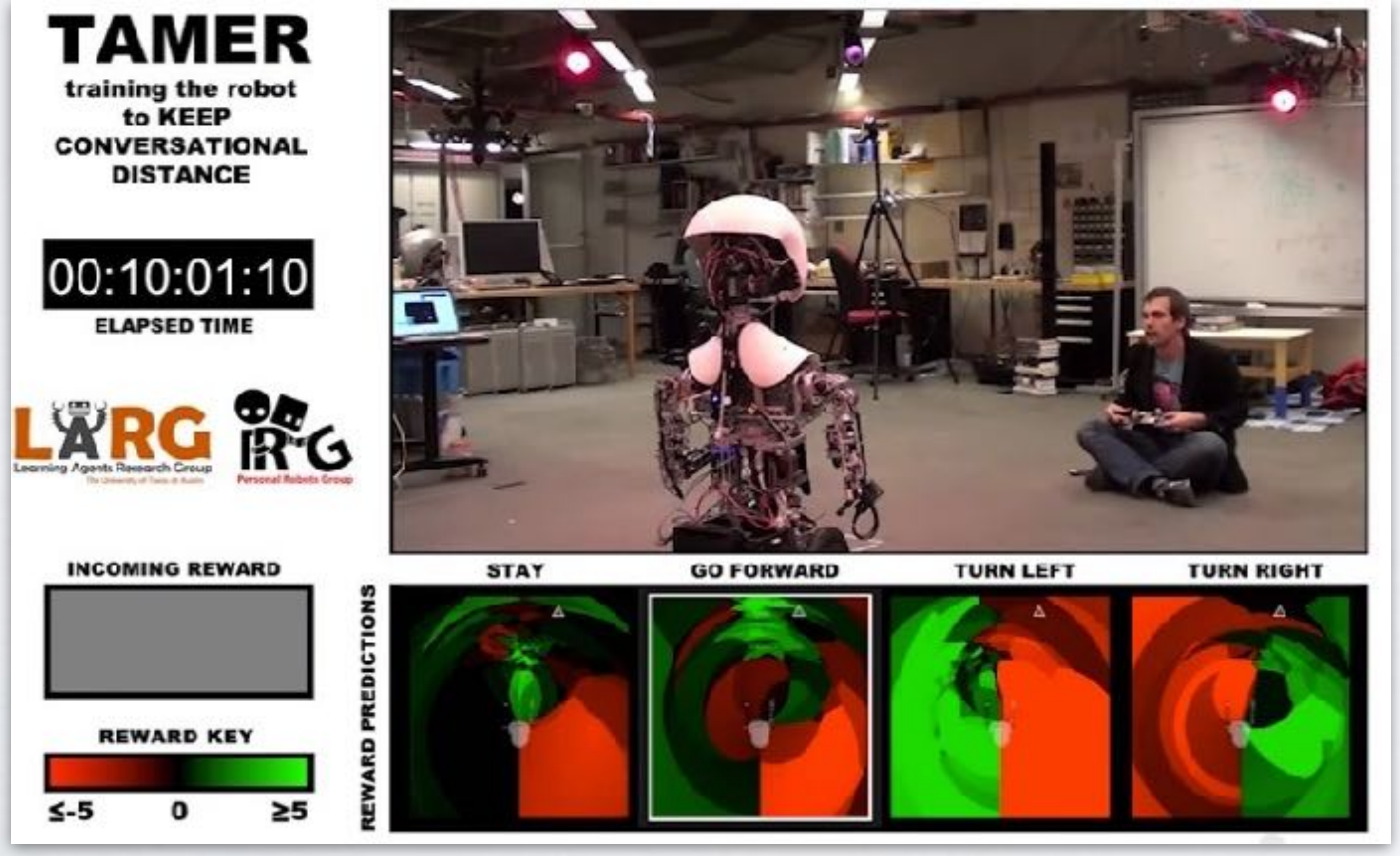
Supplementary information: Speech and critique

"Jump over the skull while going to the left"



Interpreting natural language commands

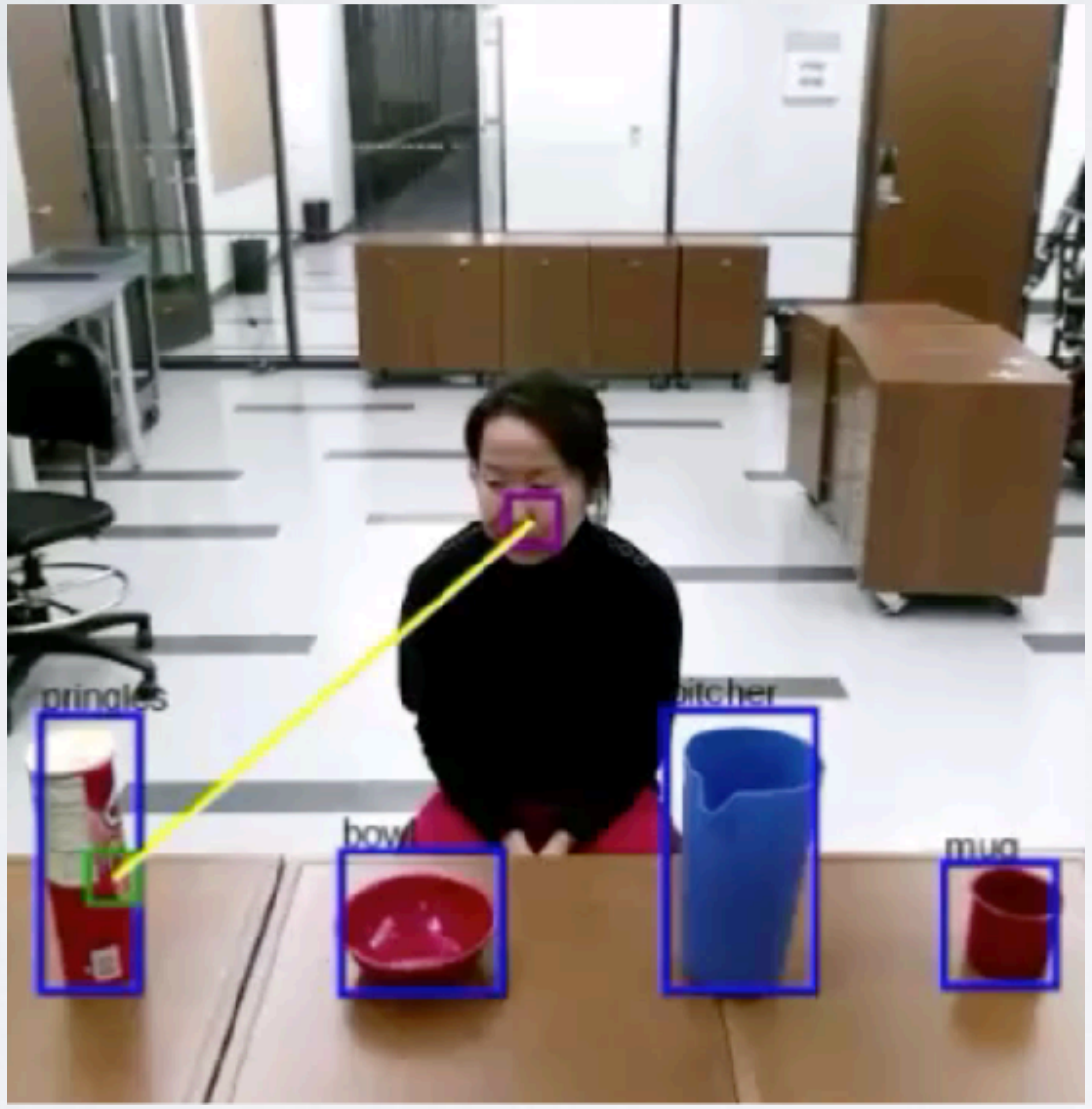
[Goyal et al. 2019]



Realtime user feedback given to RL system

[Knox et al. 2008]

Supplementary information: gaze



Human gaze to communicate intention of a demonstration

[Saran et al. 2019]

Imitation learning

Part 2: Algorithms

Behavioral cloning

Supervised learning problem:

Demos → Policy

i.e. from example (s,a) pairs, learn pi(s,a)

Behavioral cloning

Supervised learning problem:

Demos $\longrightarrow$ Policy

i.e. from example (s,a) pairs, learn pi(s,a)

What if we want to learn from experience via RL?

Inverse reinforcement learning:

Demos $\longrightarrow$ Inferred intent (reward function) $\longrightarrow$ Policy

Learning task objectives: Inverse reinforcement learning

Helicopter tricks

[Abbeel et al. 2007]

Littledog walking

[Kolter et al. 2007]

Learning task objectives: Inverse reinforcement learning

Reinforcement learning basics:

states actions transition dynamics

MDP: (S, A, T, γ, D, R)

discount rate start state distribution reward function

Policy: $\pi(s, a) \rightarrow [0, 1]$

Value function: $V^{\pi}(s_0) = \sum_{t=0}^{\infty} \gamma^t R(s_t)$

What if we have an **MDP/R**?

Learning task objectives: Inverse reinforcement learning

1. Collect user demonstration $(s_0, a_0), (s_1, a_1), \ldots, (s_n, a_n)$ and assume it is sampled from the expert's policy, π^E

2. Explain expert demos by finding R^* such that:

$$E[\textstyle\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi^E] \quad \geq \quad E[\textstyle\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi] \quad \forall \pi$$

$$E_{s_0 \sim D}[V^{\pi^E}(s_0)] \quad \geq \quad E_{s_0 \sim D}[V^{\pi}(s_0)] \quad \forall \pi$$

How can search be made tractable?

[Abbeel and Ng 2004]

Learning task objectives: Inverse reinforcement learning

Define R^* as a linear combination of features:

$$R^*(s) = w^T\phi(s), \text{ where } \phi : S \rightarrow \mathbb{R}^n$$

Then,

$$\begin{aligned} E[\textstyle\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] &= E[\textstyle\sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t)|\pi] \\ &= w^T E[\textstyle\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \\ &= w^T \mu(\pi) \end{aligned}$$

Thus, the expected value of a policy can be expressed as a weighted sum of the expected features $\mu(\pi)$

[Abbeel and Ng 2004]

Learning task objectives: Inverse reinforcement learning

Originally - Explain expert demos by finding R^* such that:

$$E[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi^E] \quad \geq \quad E[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] \quad \forall \pi$$

Use expected features:

$$E[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] = w^T \mu(\pi)$$

Restated - find w^* such that:

$$w^* \mu(\pi^E) \quad \geq \quad w^* \mu(\pi) \quad \forall \pi$$

[Abbeel and Ng 2004]

Learning task objectives: Inverse reinforcement learning

Goal: Find w^* such that: $w^* \mu(\pi^E) \geq w^* \mu(\pi) \;\; \forall \pi$

1. Initialize π_0 to any policy

Iterate for i = 1, 2, … :

2. Find w^* s.t. expert maximally outperforms all previously examined policies $\pi_{0 \ldots i-1}$:

$$\max_{\epsilon, w^* : \|w^*\|_2 \leq 1} \epsilon \quad \text{s.t.} \quad w^* \mu(\pi^E) \geq w^* \mu(\pi_j) + \epsilon$$

3. Use RL to calc. optimal policy π_i associated with w^*

4. Stop if $\epsilon \leq$ threshold

[Abbeel and Ng 2004]

Learning task objectives: Inverse reinforcement learning

Goal: Find w^* such that: $w^* \mu(\pi^E) \quad \geq \quad w^* \mu(\pi) \;\; \forall \pi$

1. Initialize π_0 to any policy

Iterate for i = 1, 2, … :

2. Find w^* s.t. expert maximally outperforms all previously examined policies $\pi_{0 \ldots i-1}$:

$$\max_{\epsilon, w^* : \|w^*\|_2 \leq 1} \epsilon \quad \text{s.t.} \quad w^* \mu(\pi^E) \geq w^* \mu(\pi_j) + \epsilon$$

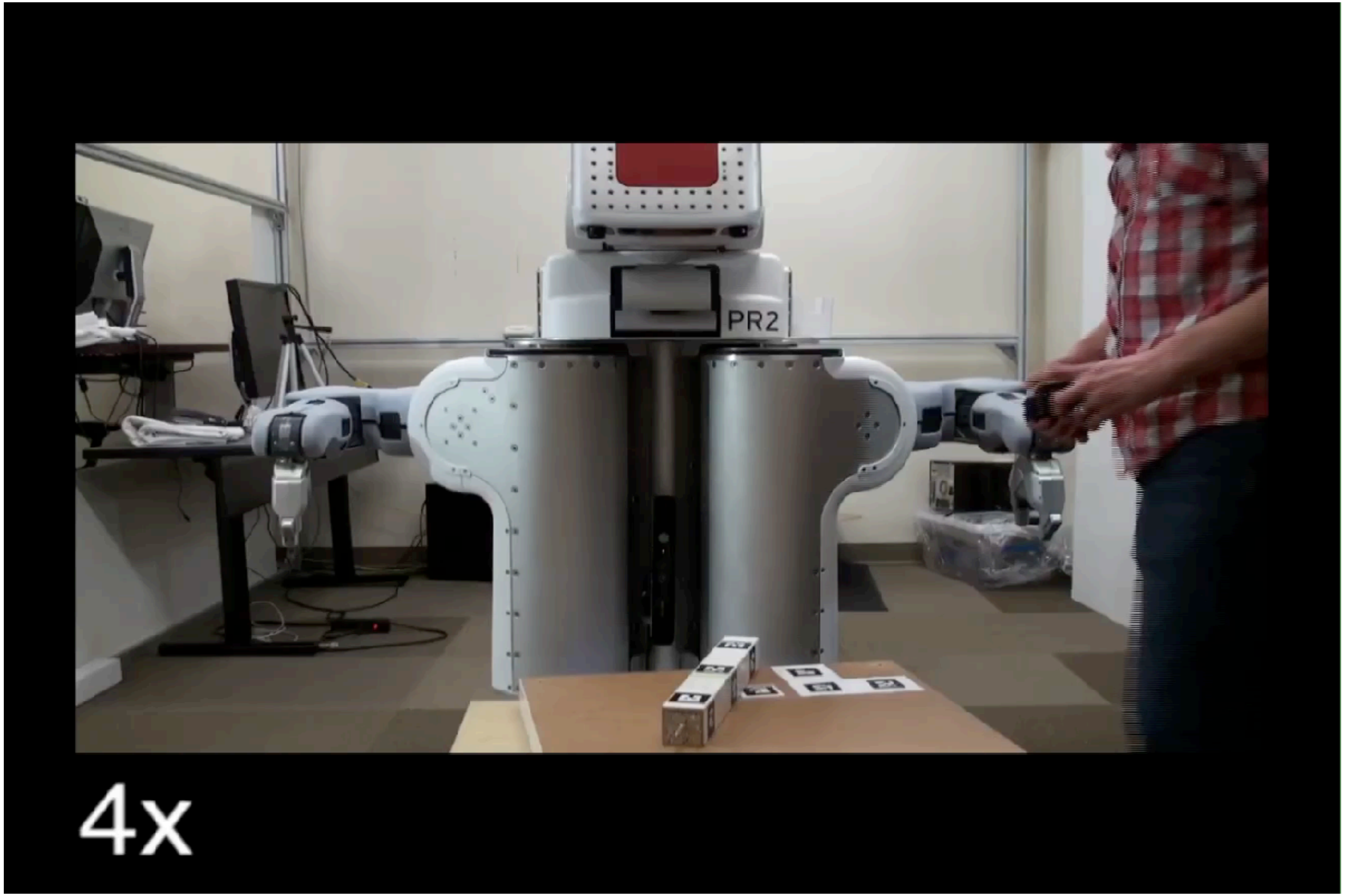
SVM solver

3. Use RL to calc. optimal policy π_i associated with w^*

4. Stop if $\epsilon \leq$ threshold

[Abbeel and Ng 2004]

Imitation learning



Resolving ambiguity: Bayesian Inverse Reinforcement Learning

[Ramachandran and Amir 2007]

- Use MCMC to sample from posterior:

$$P(R|D) \propto P(D|R)P(R)$$

- Assume demonstrations follow softmax policy with temperature c:

$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b\in A} e^{cQ^*(s,b,R)}}$$

Resolving ambiguity: Maximum Entropy IRL

[Ziebart et al. 2008]

Problem: Don't assume any more about what decisions you should make than what the data directly implies. In all other cases, be agnostic.

MaxEnt IRL finds the reward function that induces the highest entropy ("flattest") trajectory distribution that matches the features counts of the expert, under the following likelihood function:

$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}}$$

Note that all trajectories with the same return have the same probability.

Problems with standard inverse reinforcement learning

Policy learning in inner loop

- some methods learn optimal policy / value function for candidate reward functions
- others alternate policy updates and reward updates

Cannot outperform demonstrator

- matches feature counts or maximizes p(demo | reward fxn)
- Assumes demonstrator is (near) optimal

Assumption:

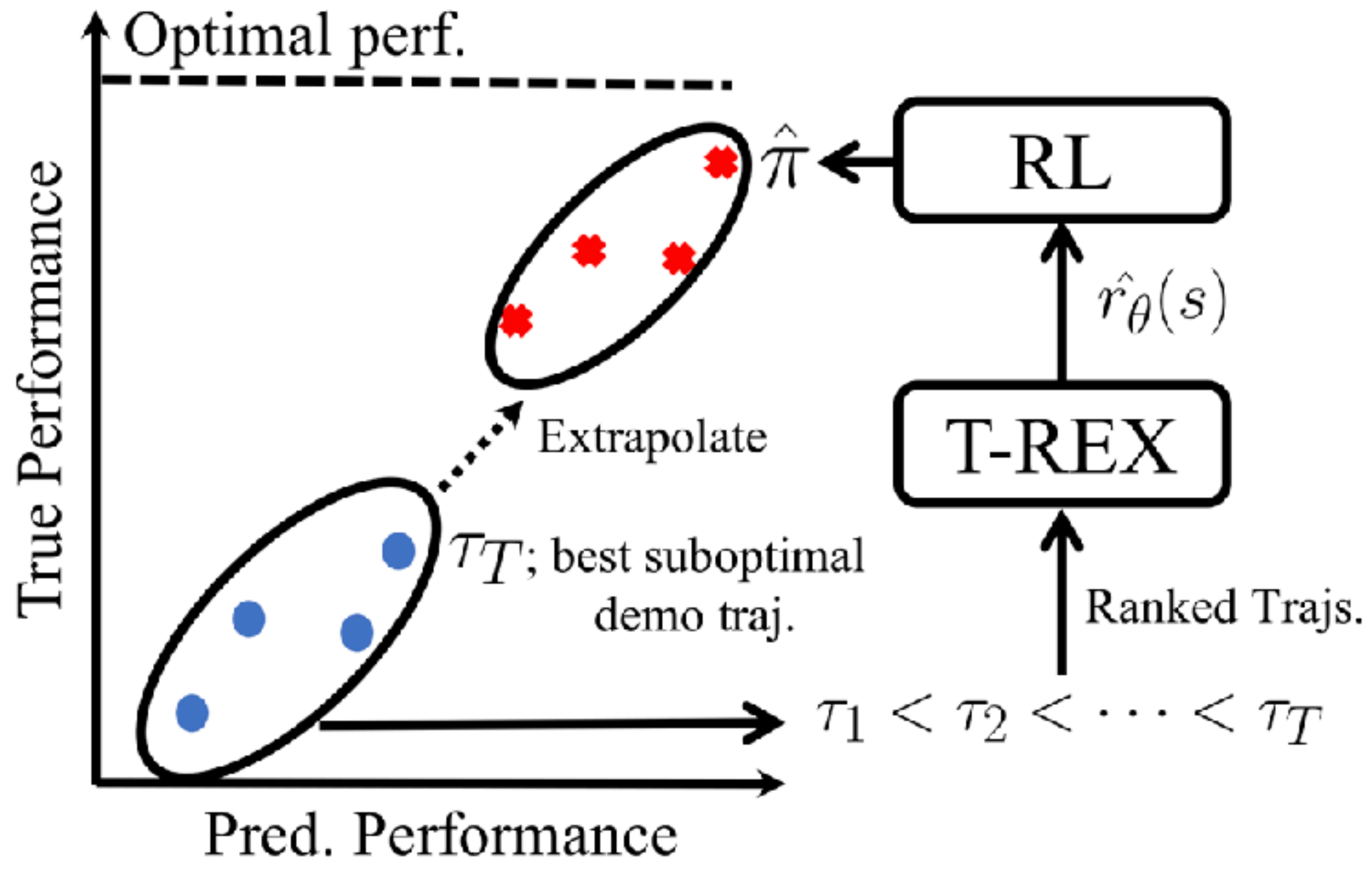
IRL should assume that the expert is near-optimal

↓

Ranked, suboptimal demonstrations provide significant computational and performance benefits

D.S. Brown, W. Goo, and S. Niekum.
[Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations].
International Conference on Machine Learning (ICML), June 2019.

T-REX: Trajectory-ranked Reward Extrapolation

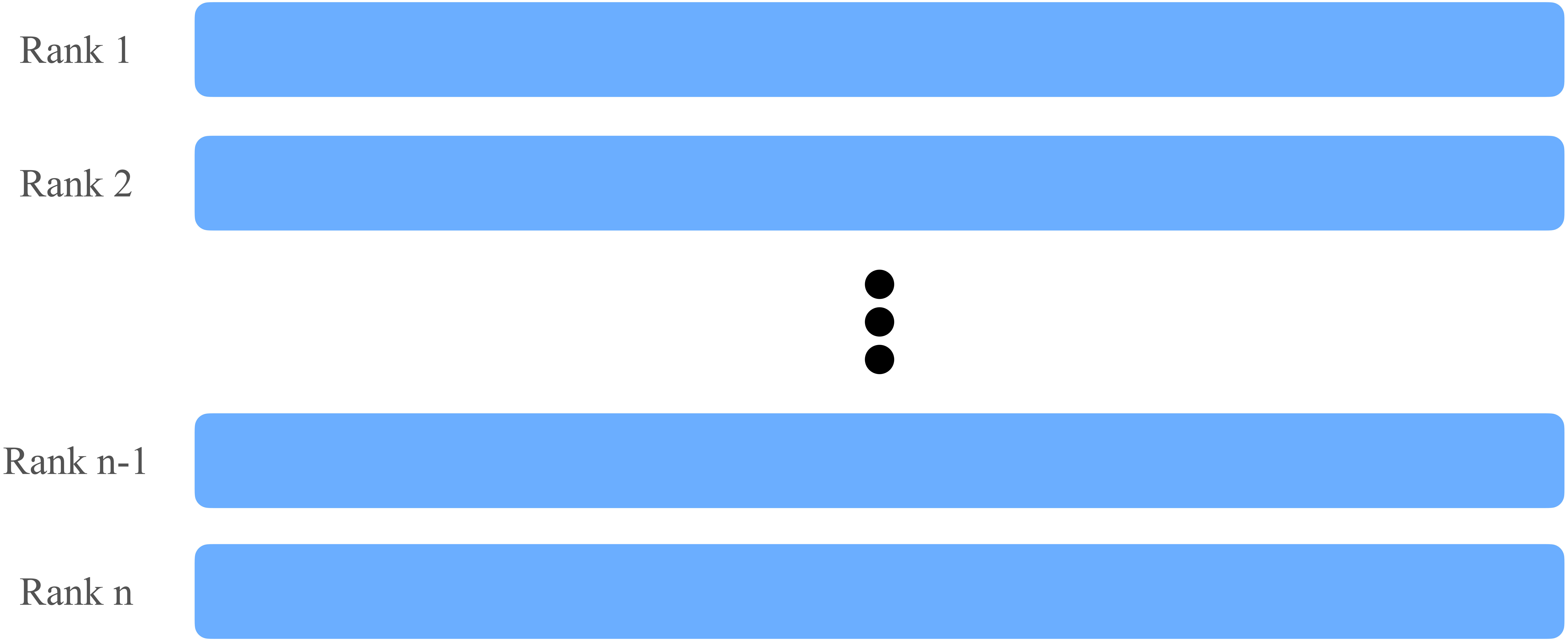


$$\mathcal{L}(\theta) = \mathbf{E}_{\tau_i,\tau_j \sim \Pi}\left[\xi\left(\mathrm{P}\left(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)\right), \tau_i \prec \tau_j\right)\right]$$

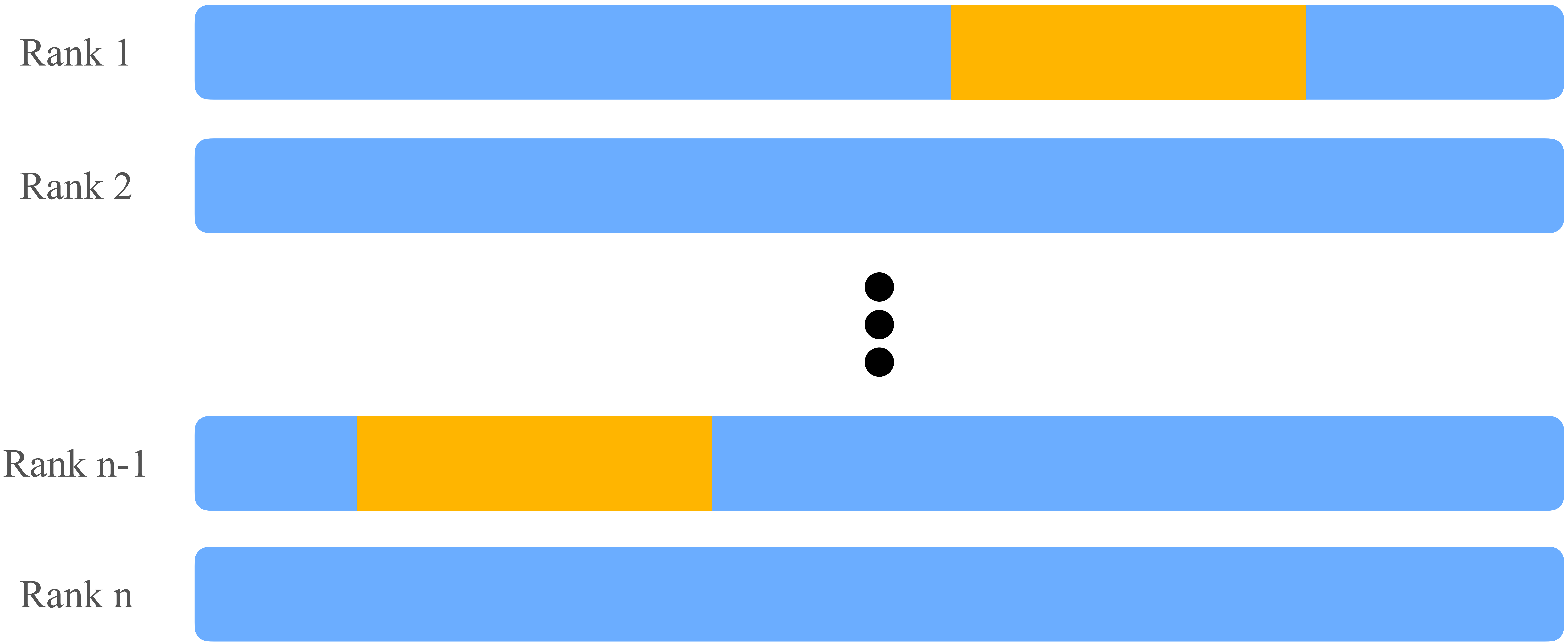
$$\mathrm{P}\left(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)\right) \approx \frac{\exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s) + \exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}$$

- Fully supervised — no policy learning
- No action labels required
- Extrapolation potential
- Works on high-dim (e.g. Atari) with ~10 demos

Data augmentation

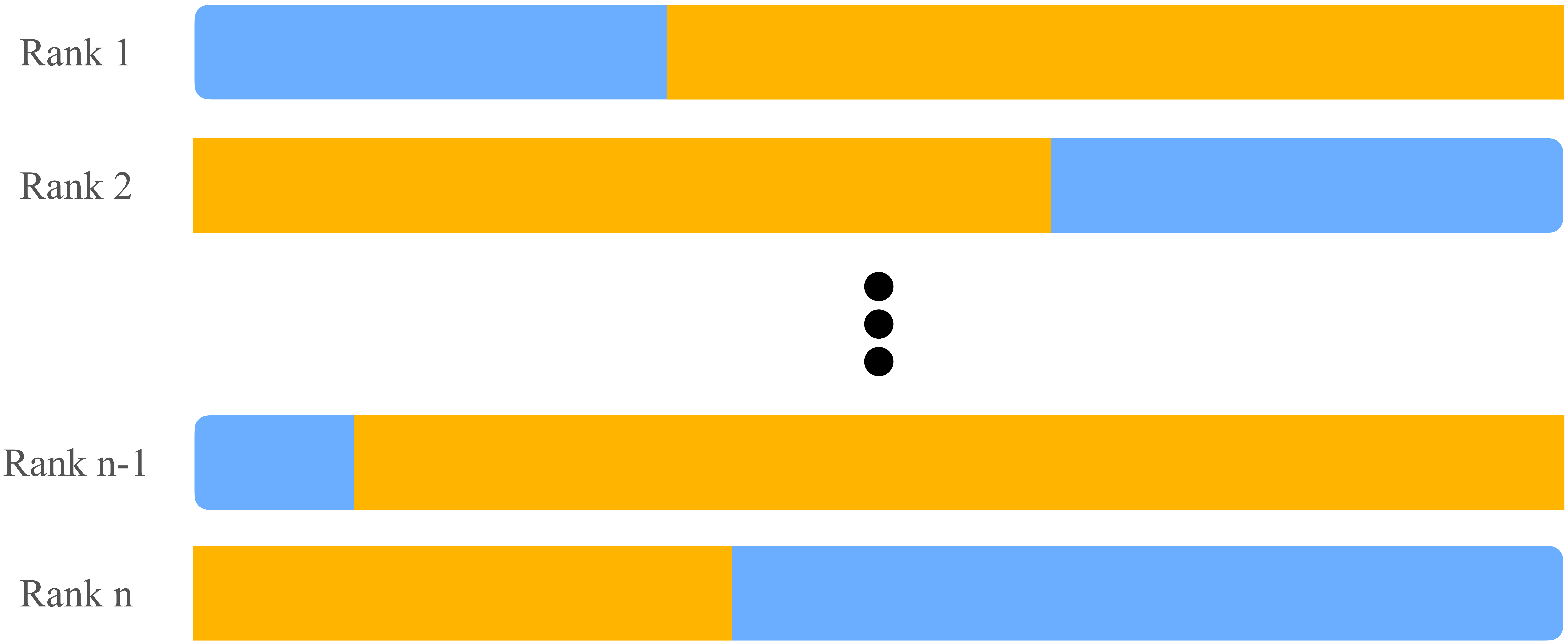


Data augmentation



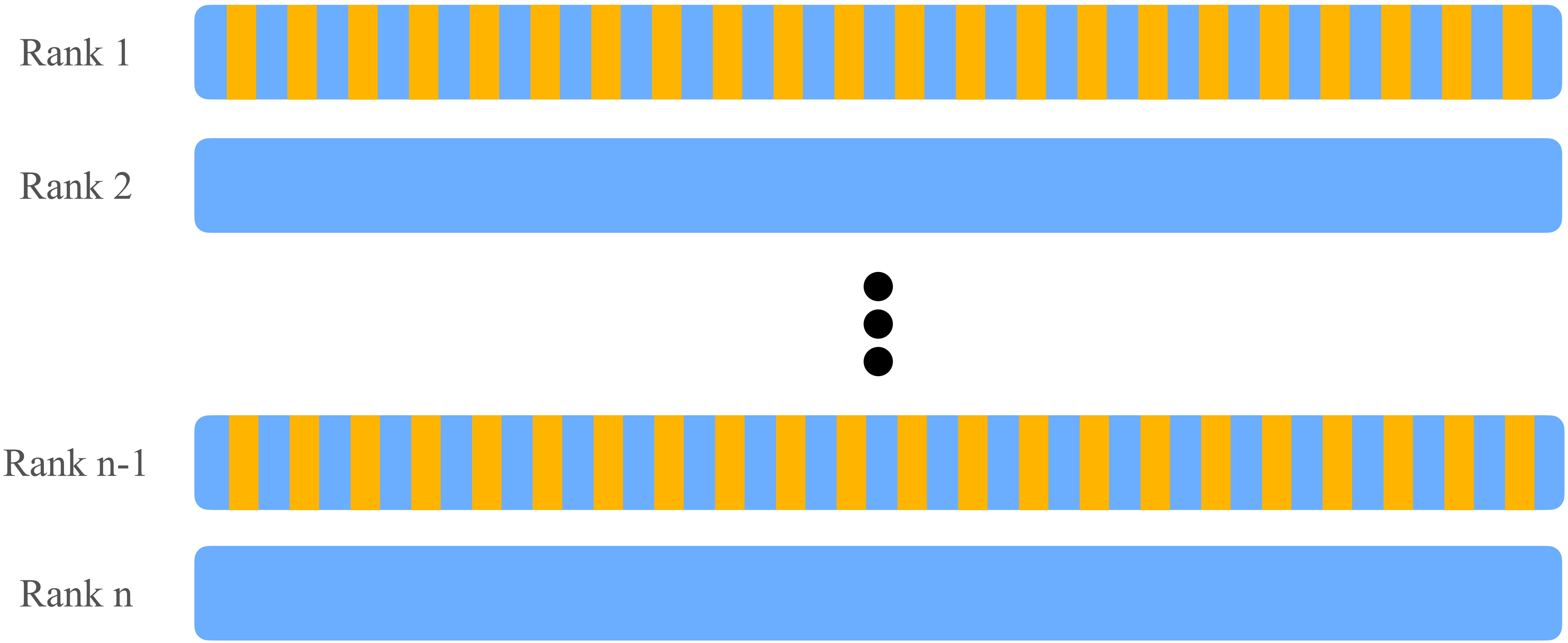
Subsampling

Data augmentation



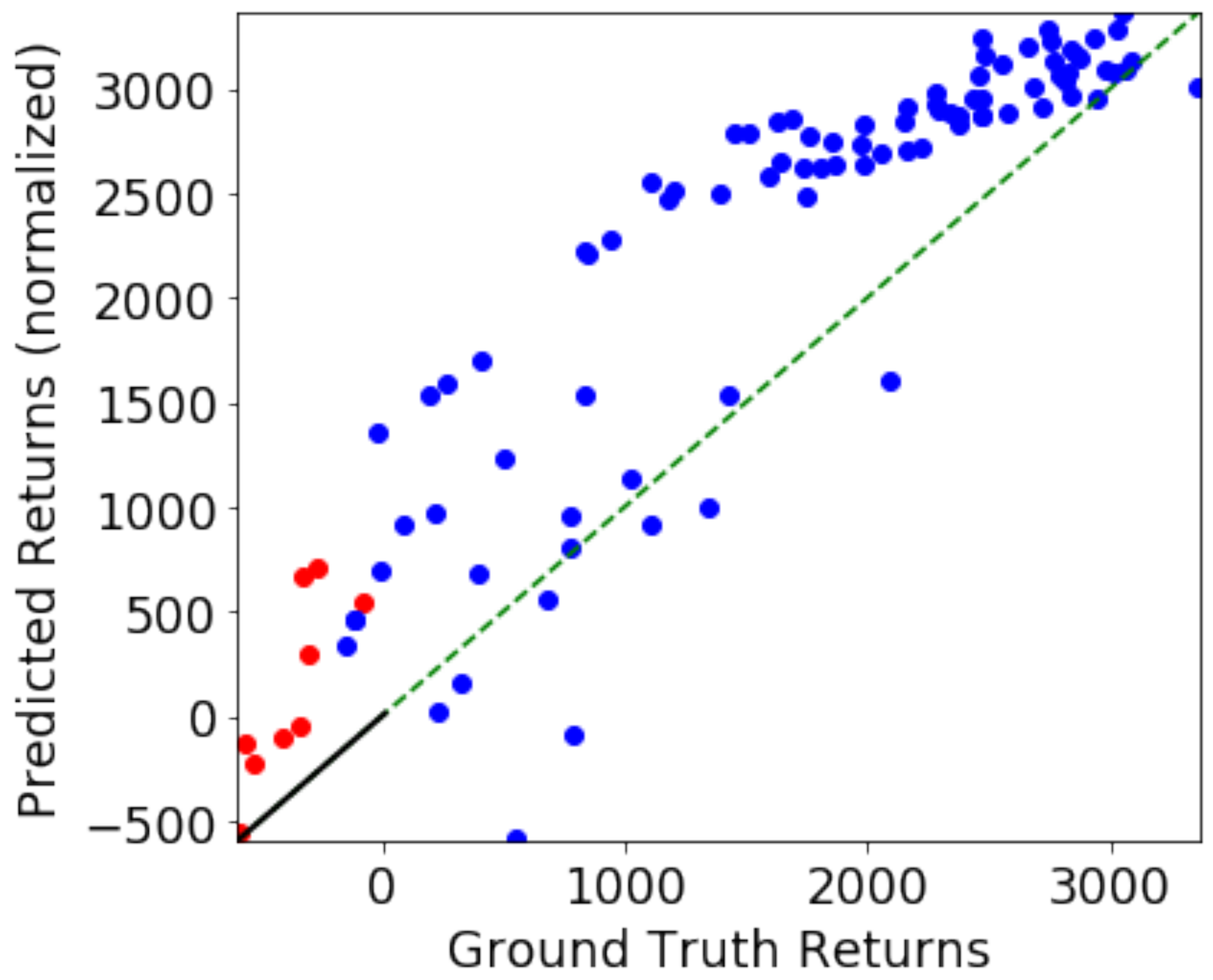
Supersampling

Data augmentation

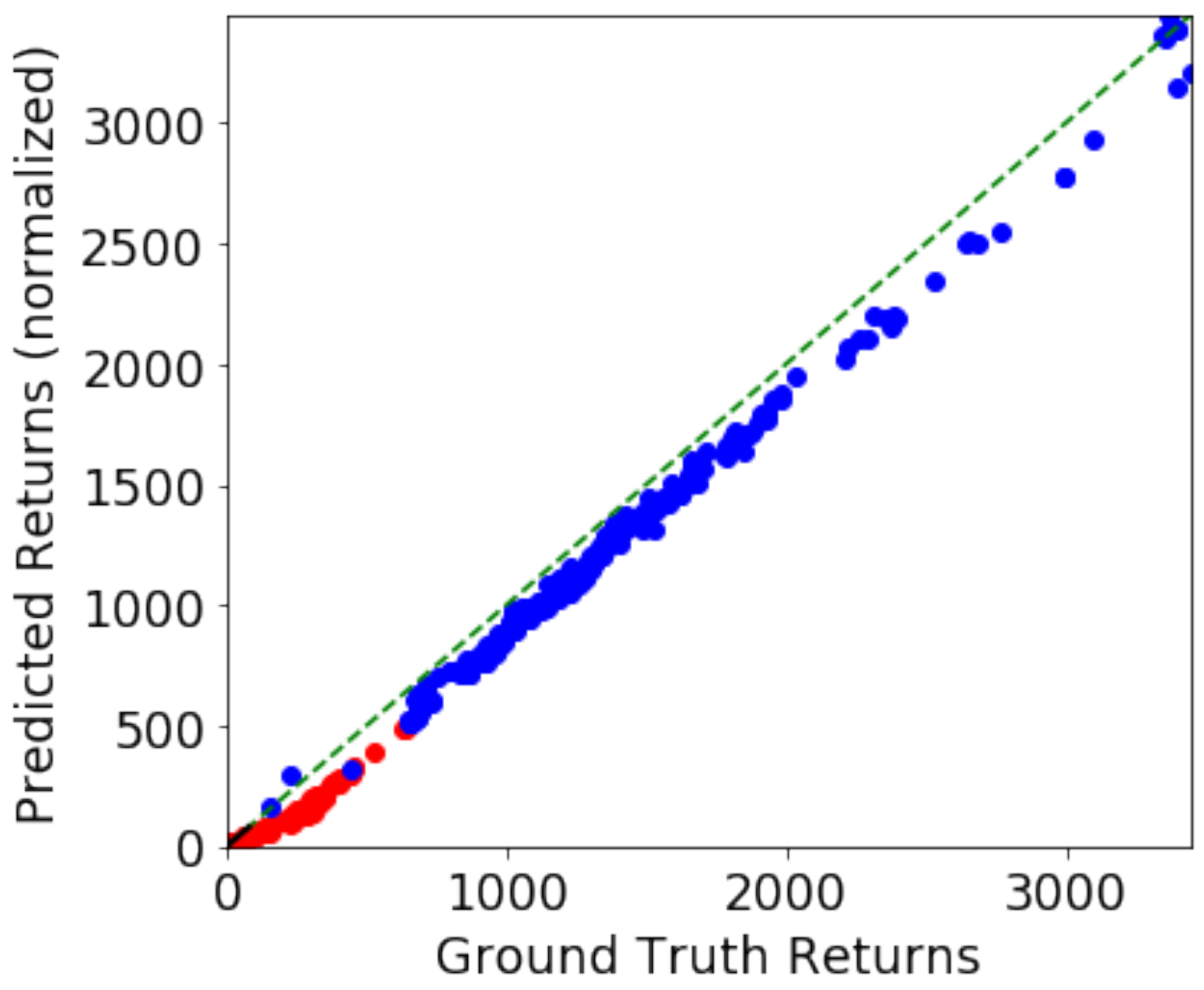


Frame skipping

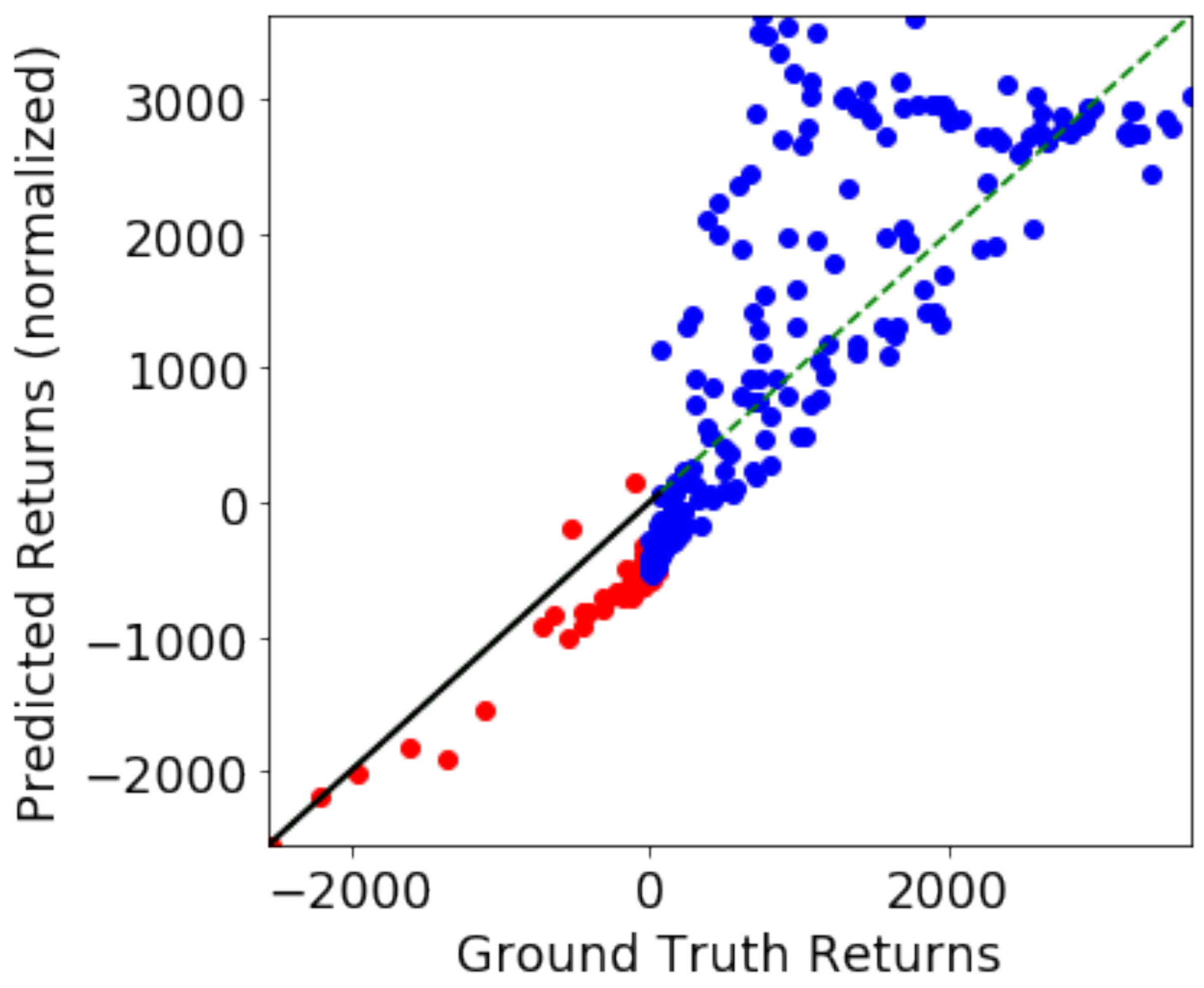
T-REX reward prediction



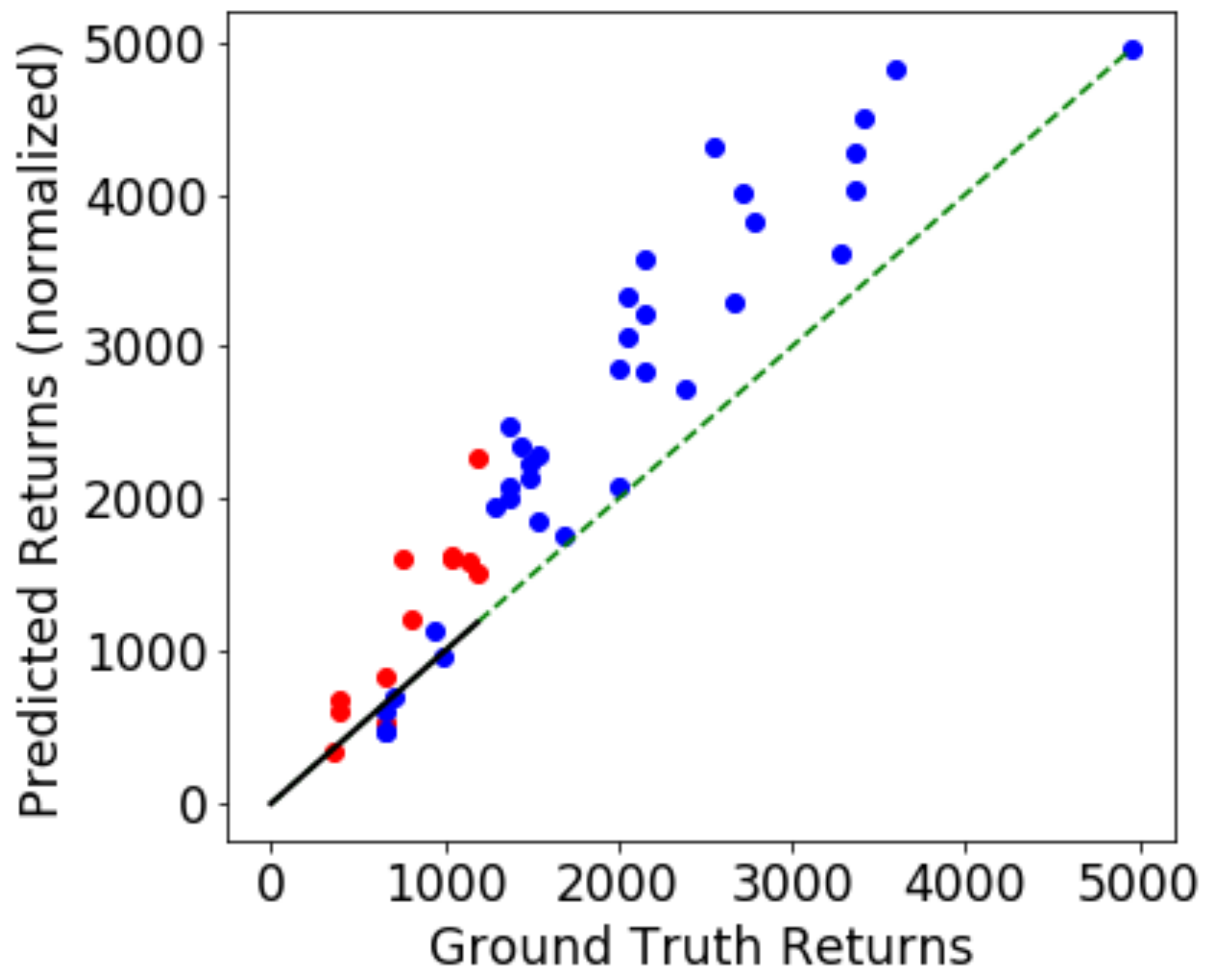
HalfCheetah



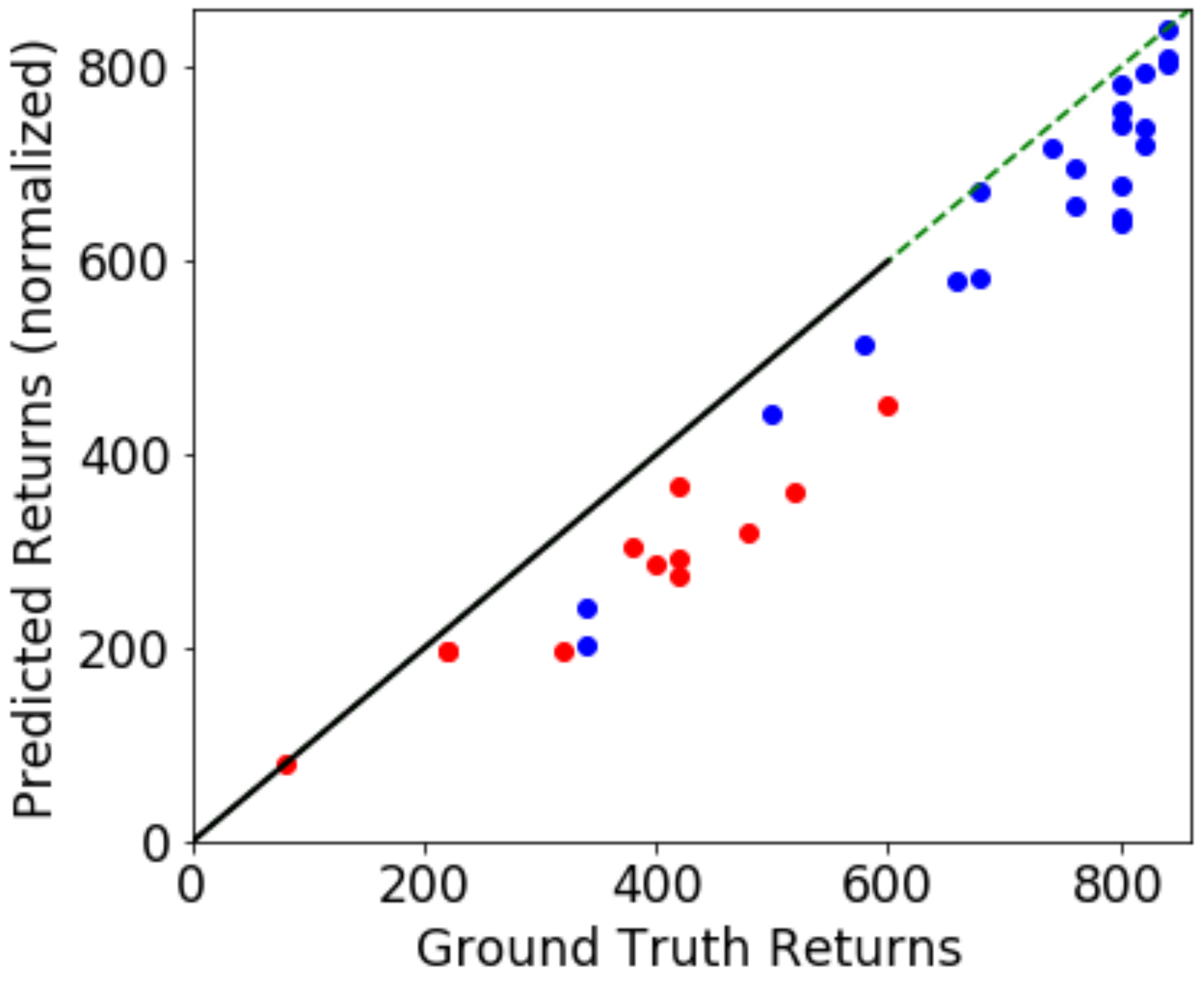
Hopper



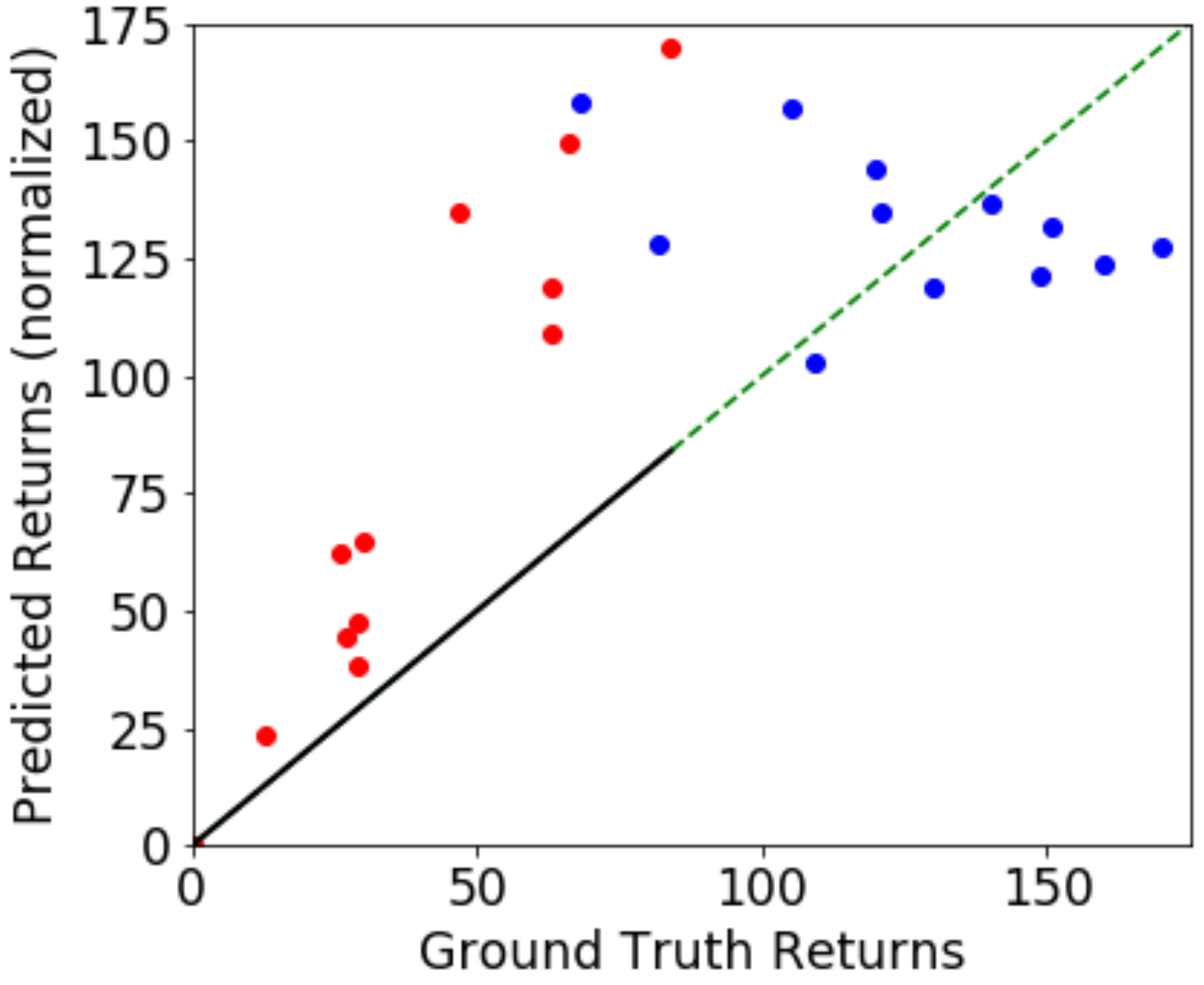
Ant



Beam Rider

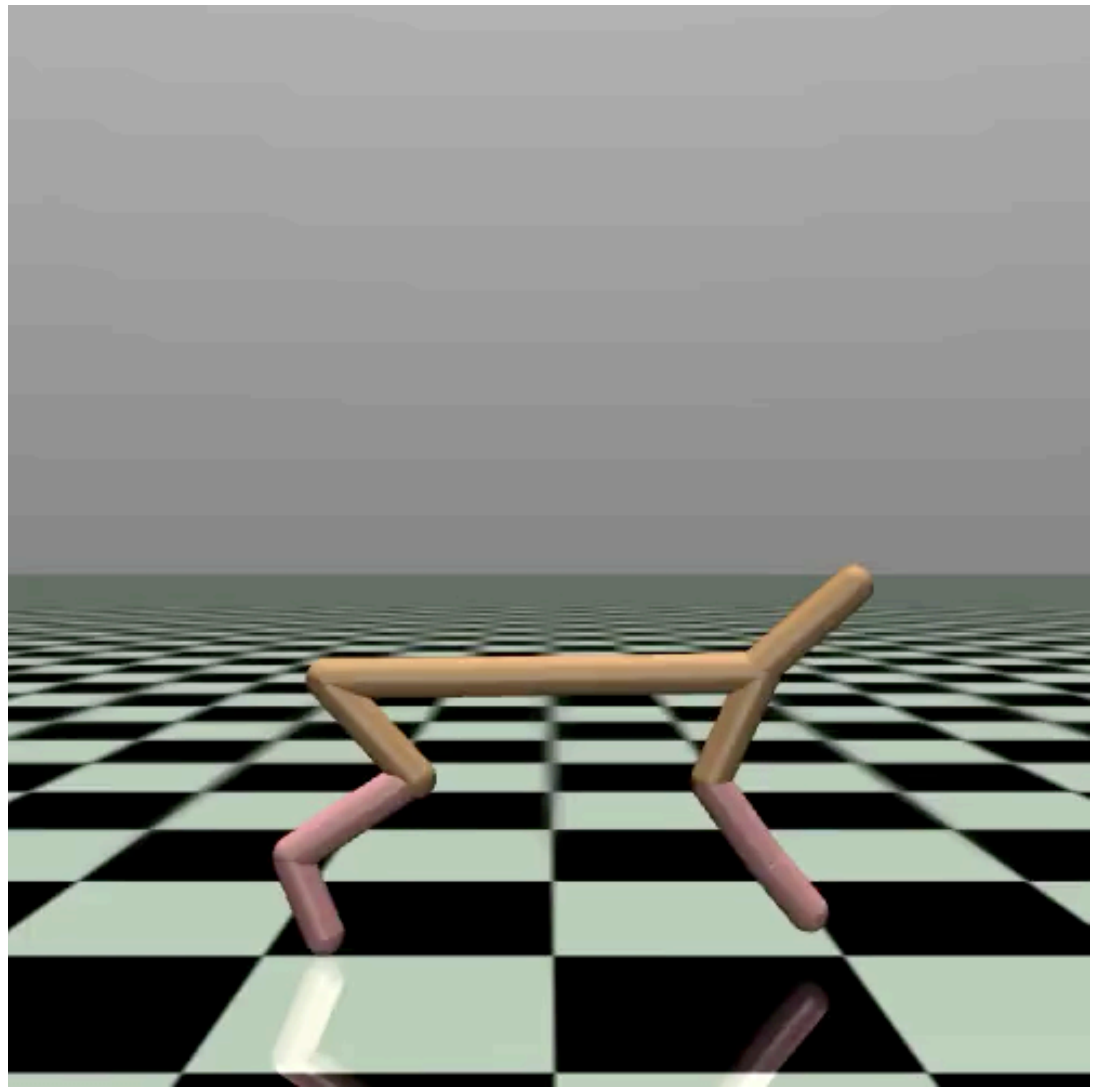


Seaquest

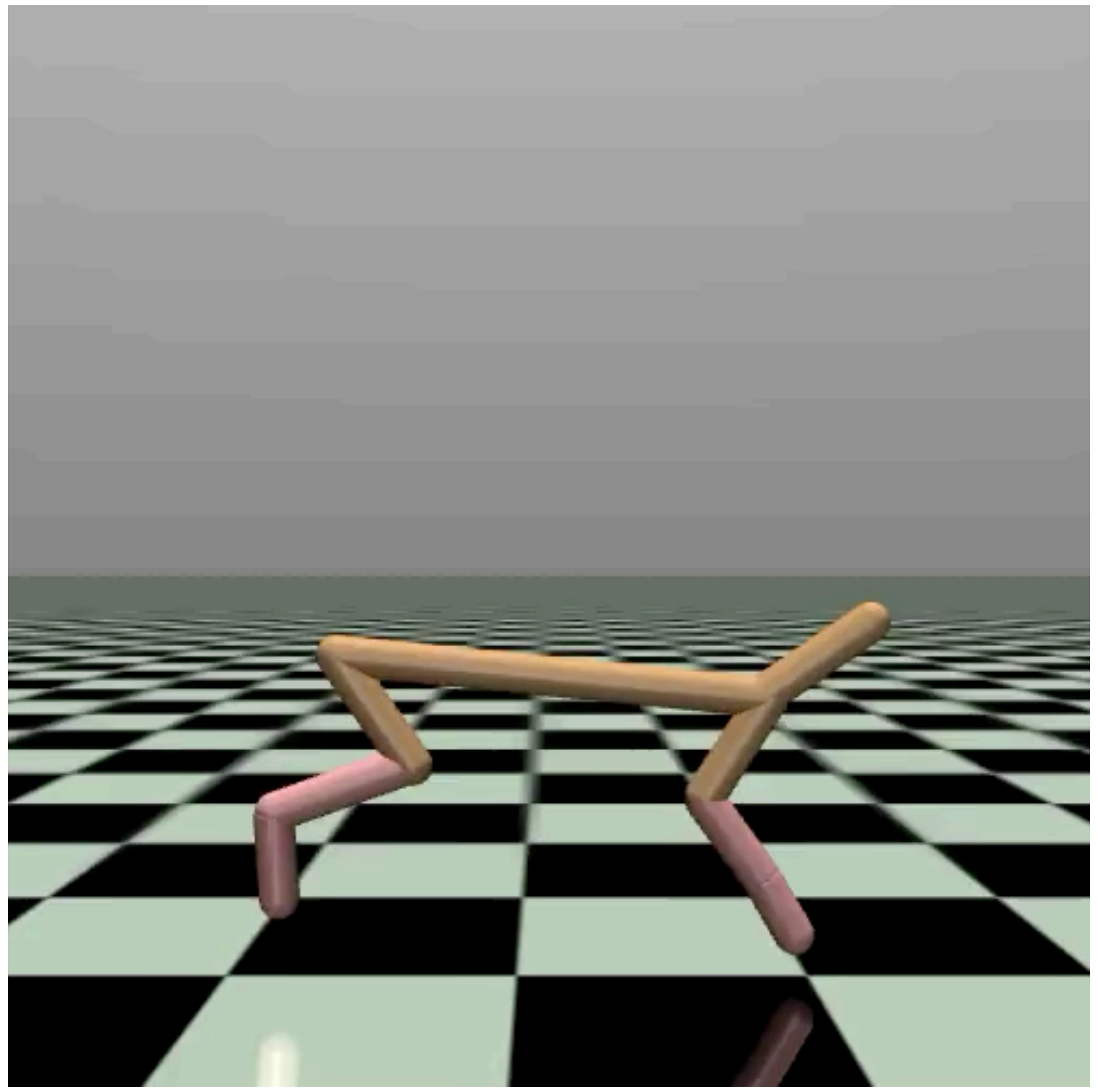


Enduro

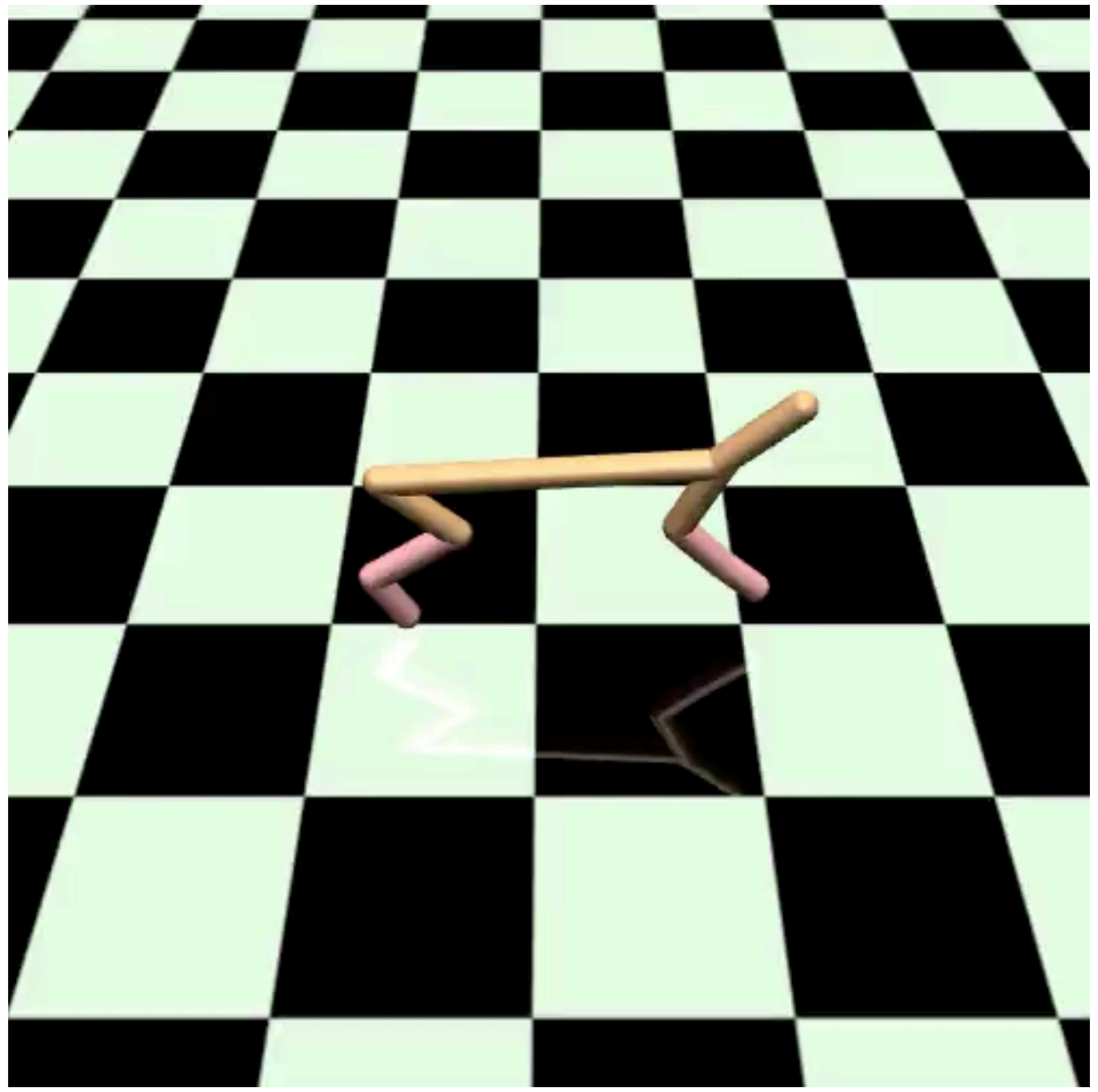
Ranked demonstrations: HalfCheetah



12.52

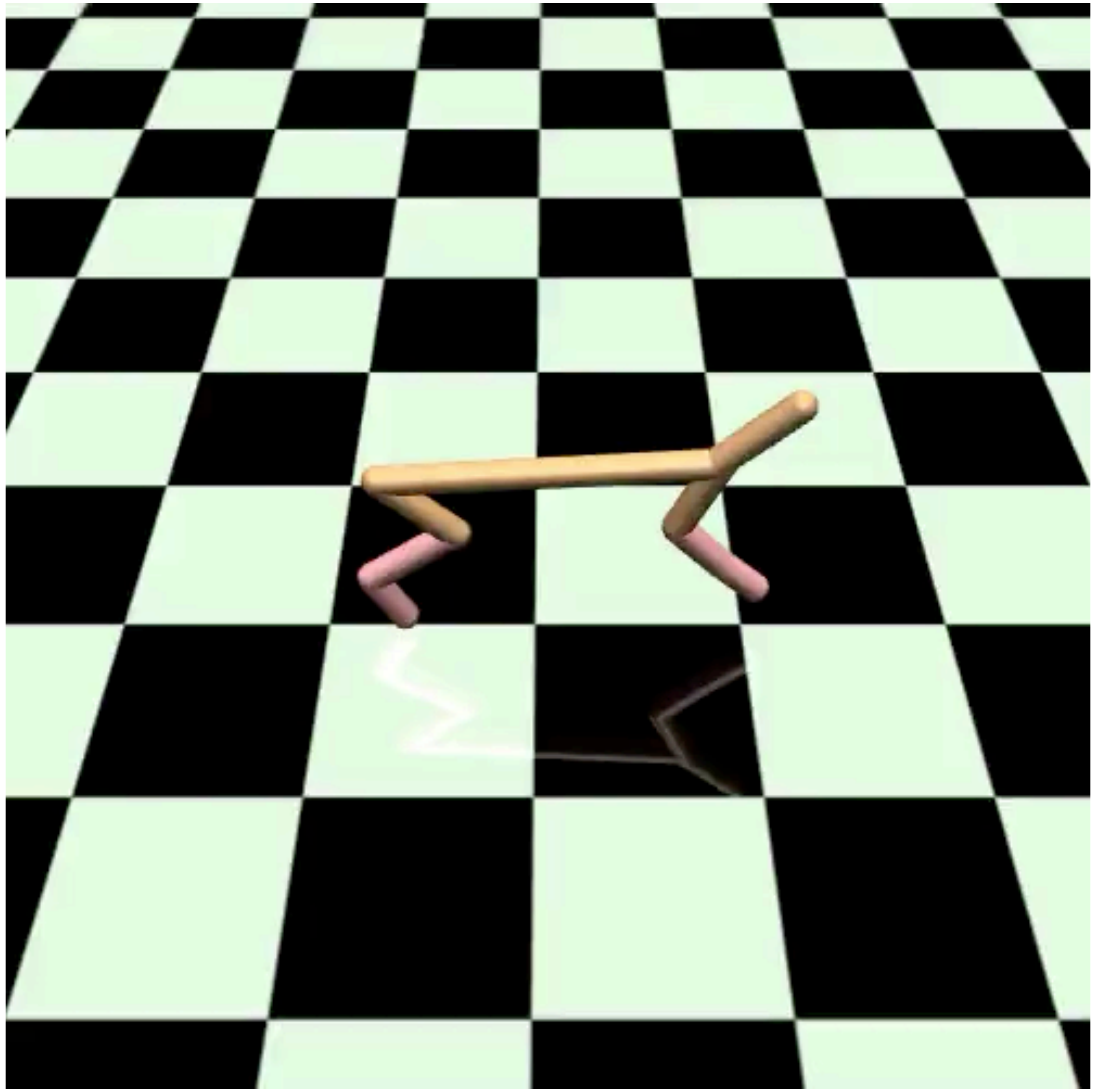


44.98

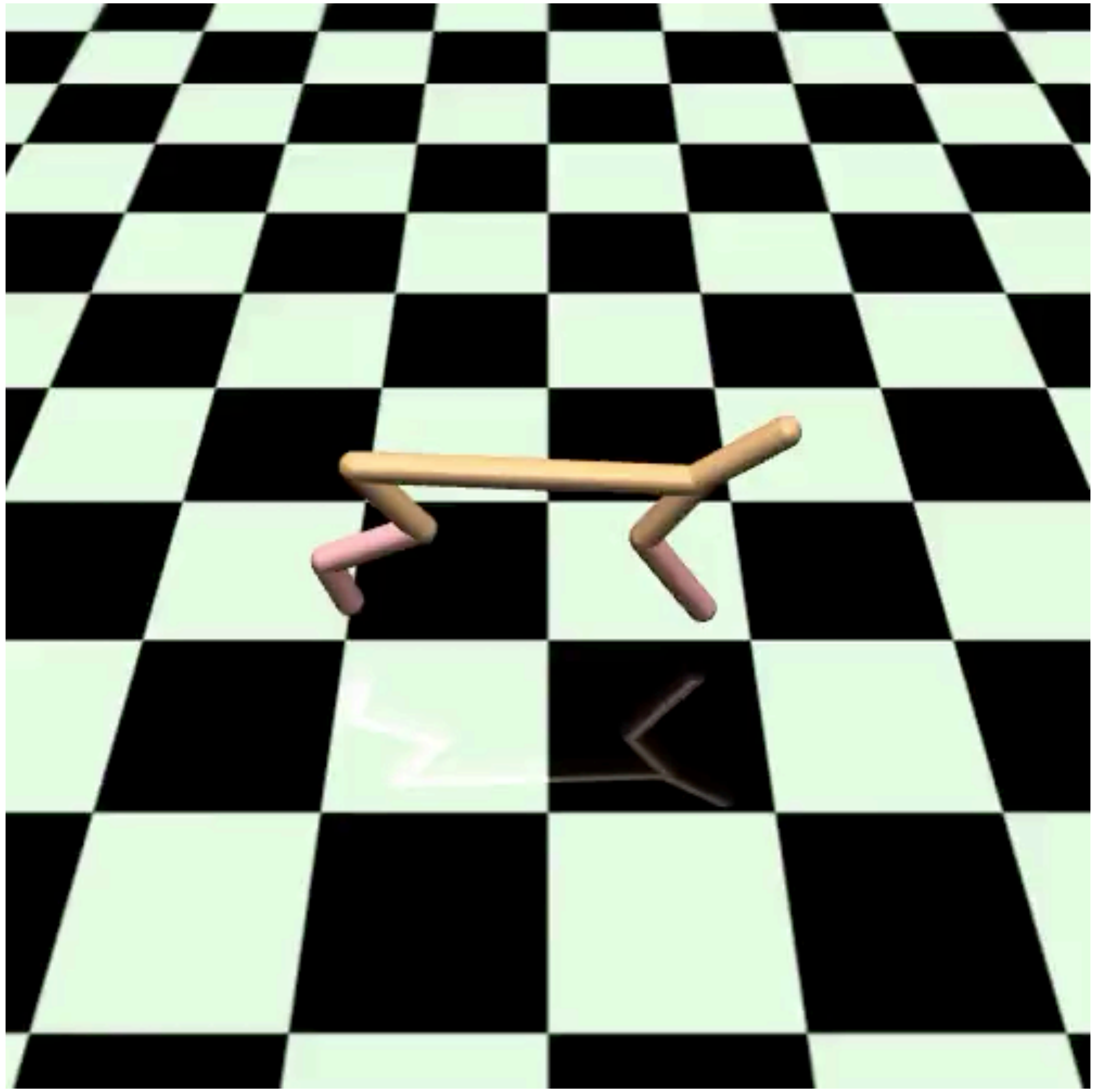


88.97

Results: HalfCheetah

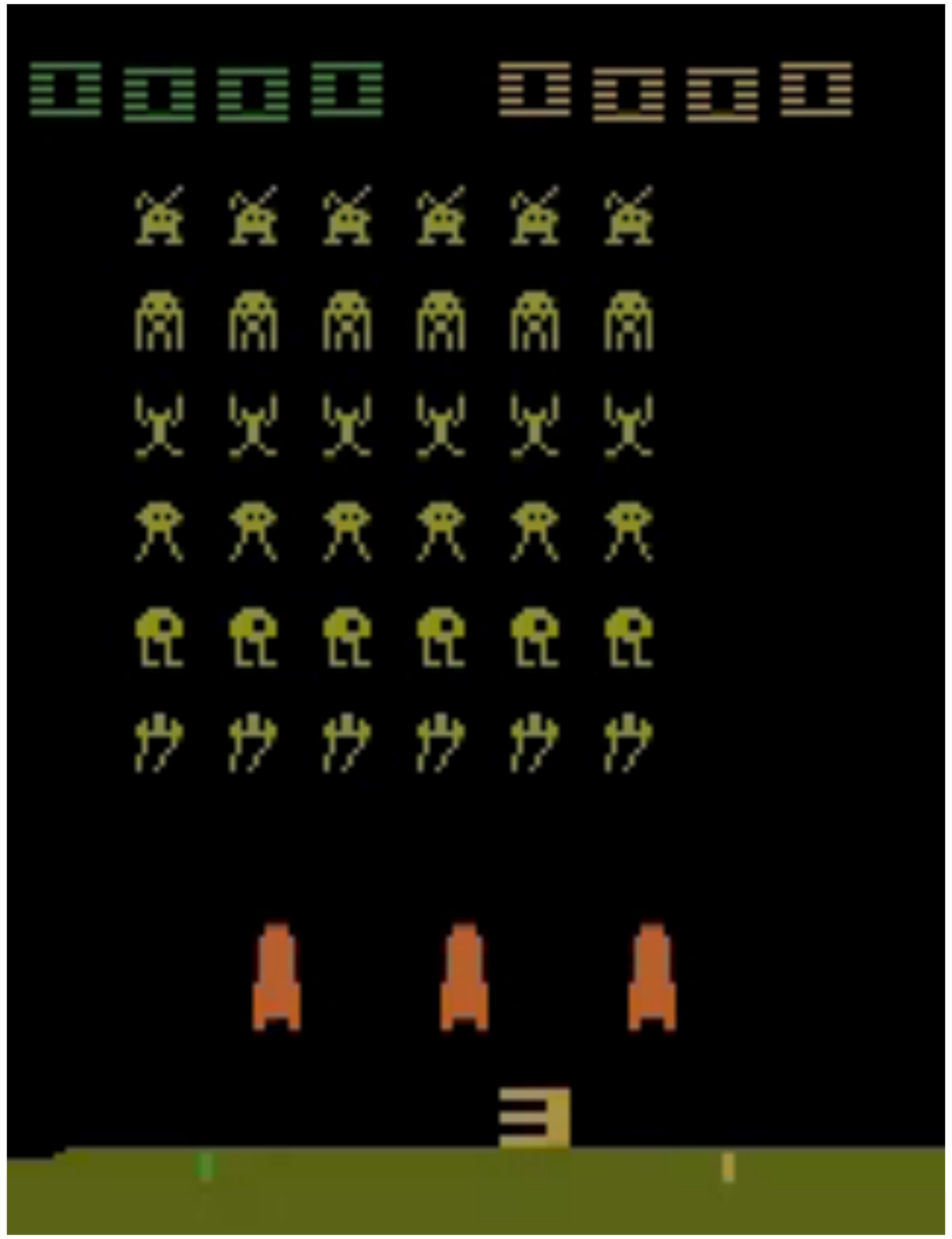


Best demo (88.97)

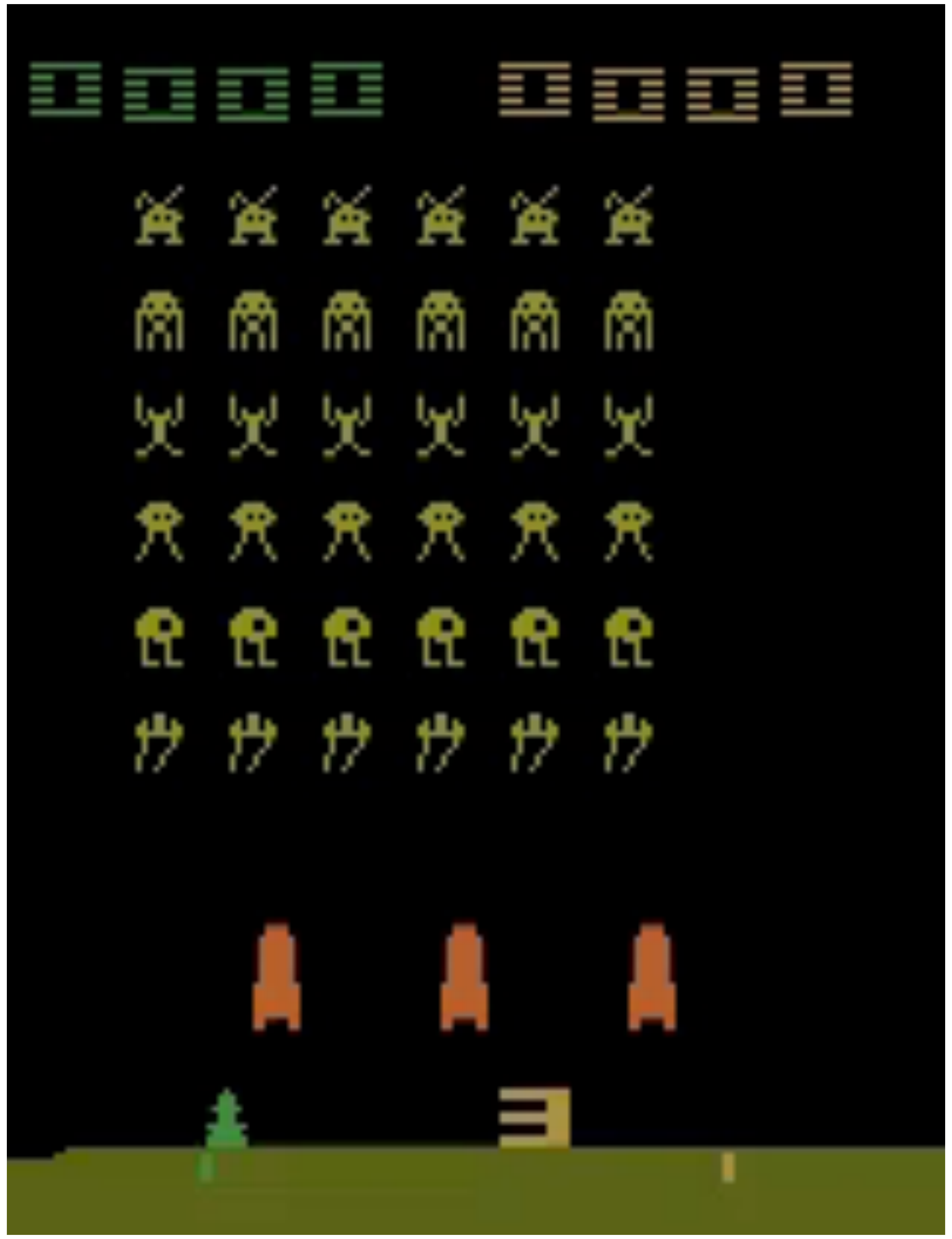


T-REX (143.40)

Results: Atari

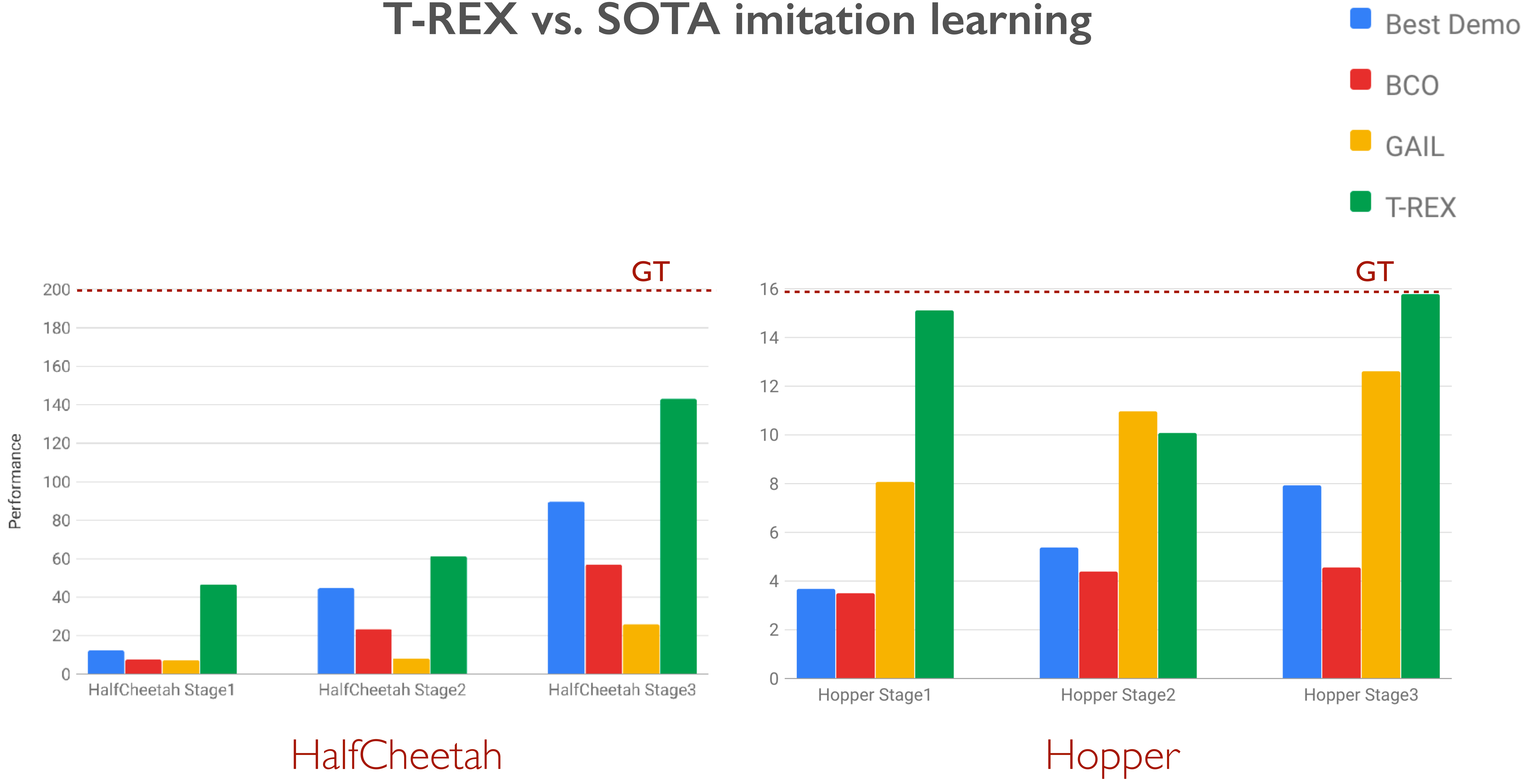


Best demo (600)

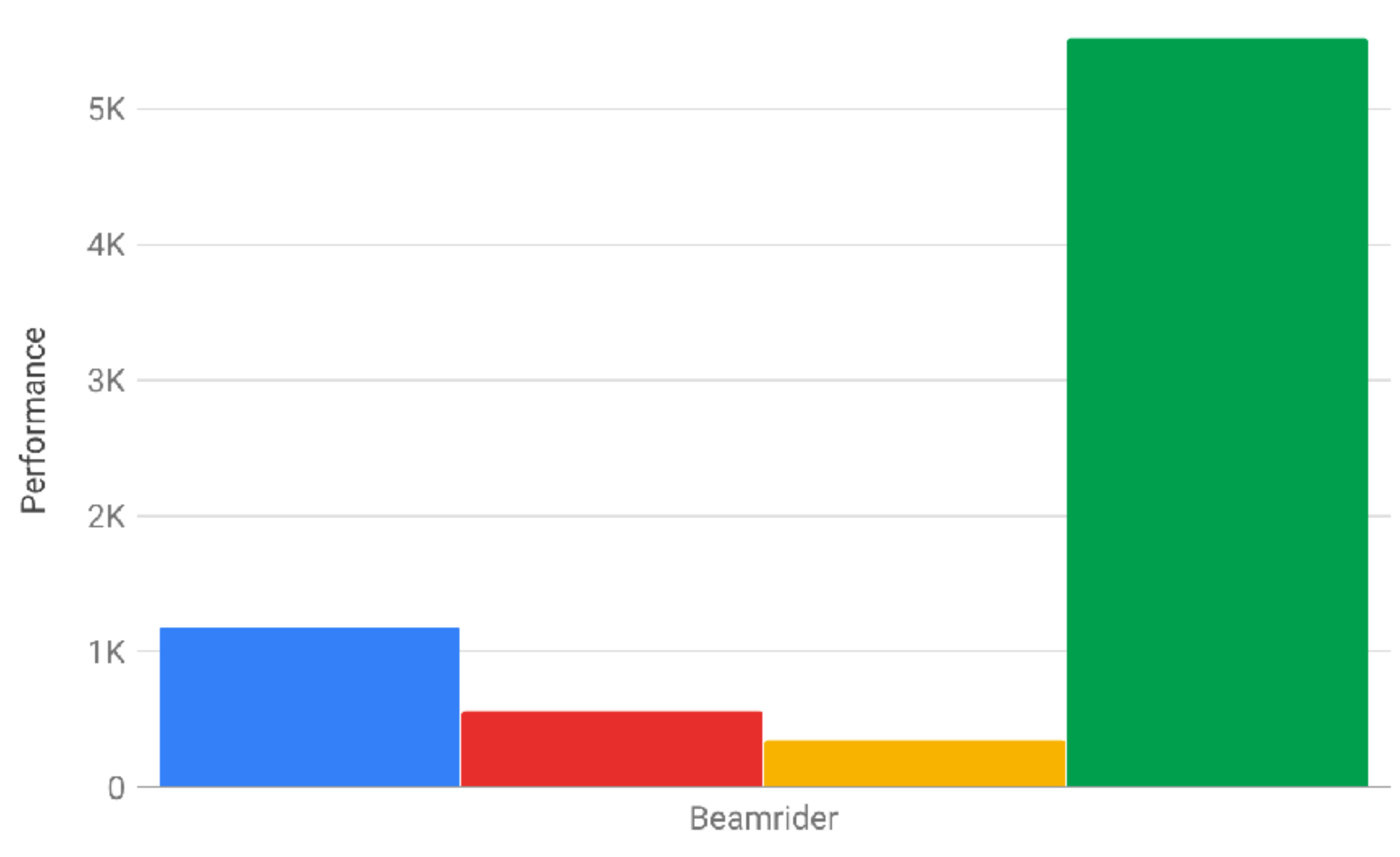


T-REX (1495)

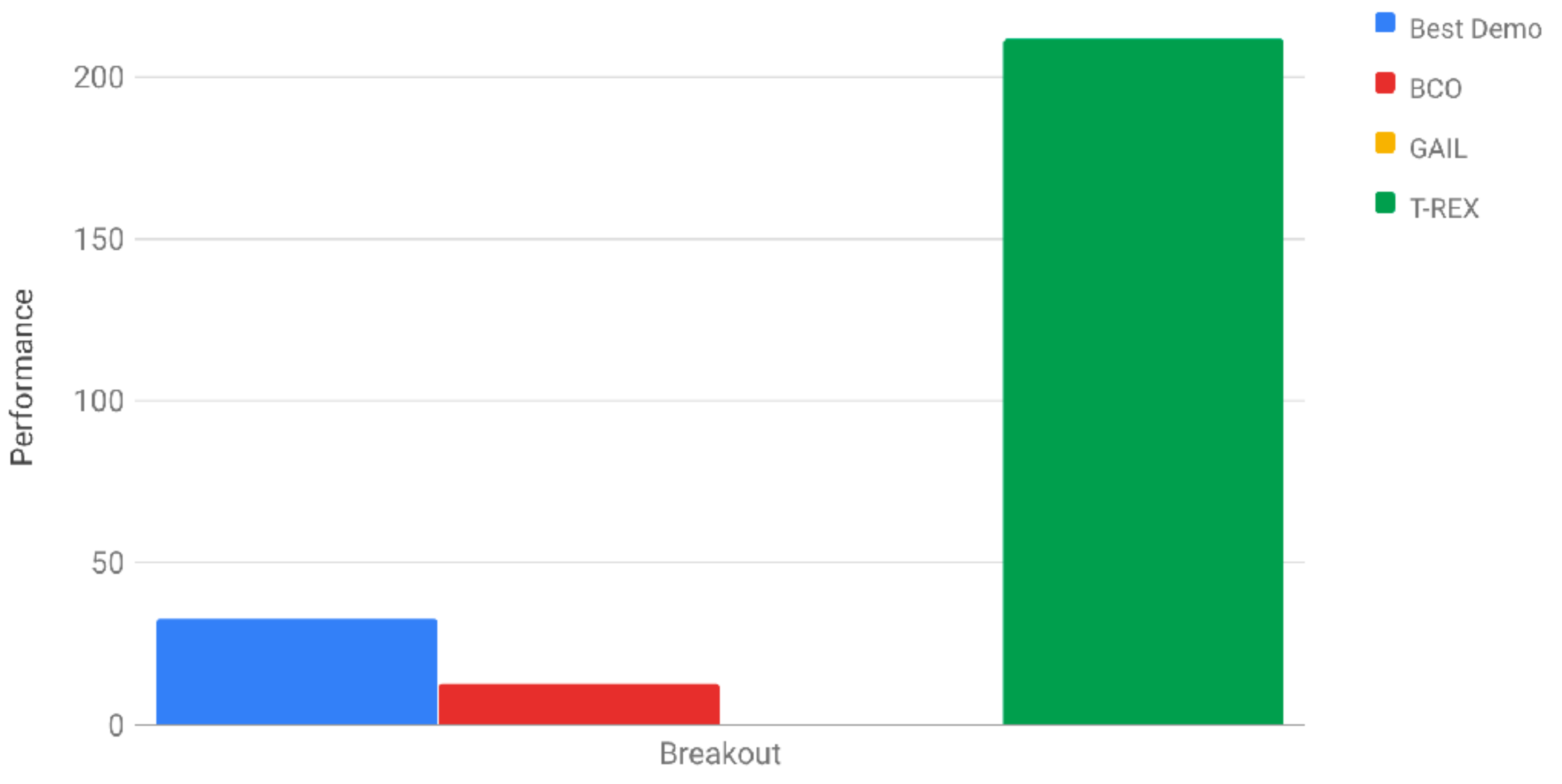
T-REX vs. SOTA imitation learning



T-REX vs. SOTA imitation learning



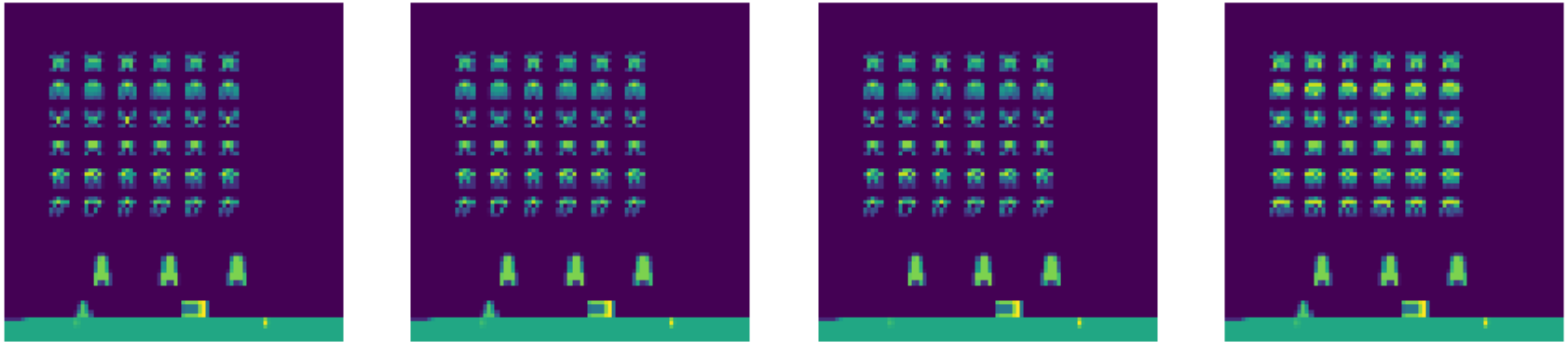
Beamrider



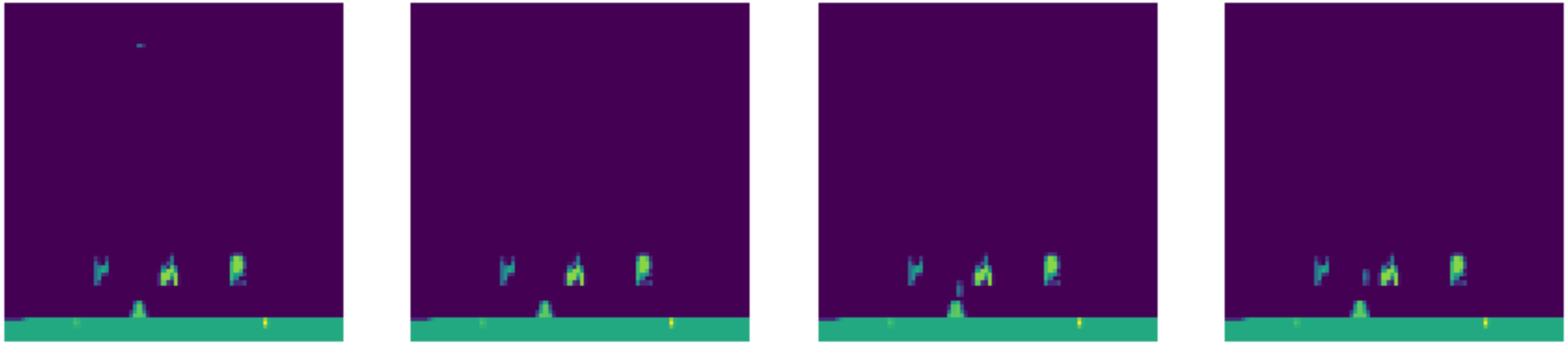
Breakout

Frame stacks: best vs. worst reward

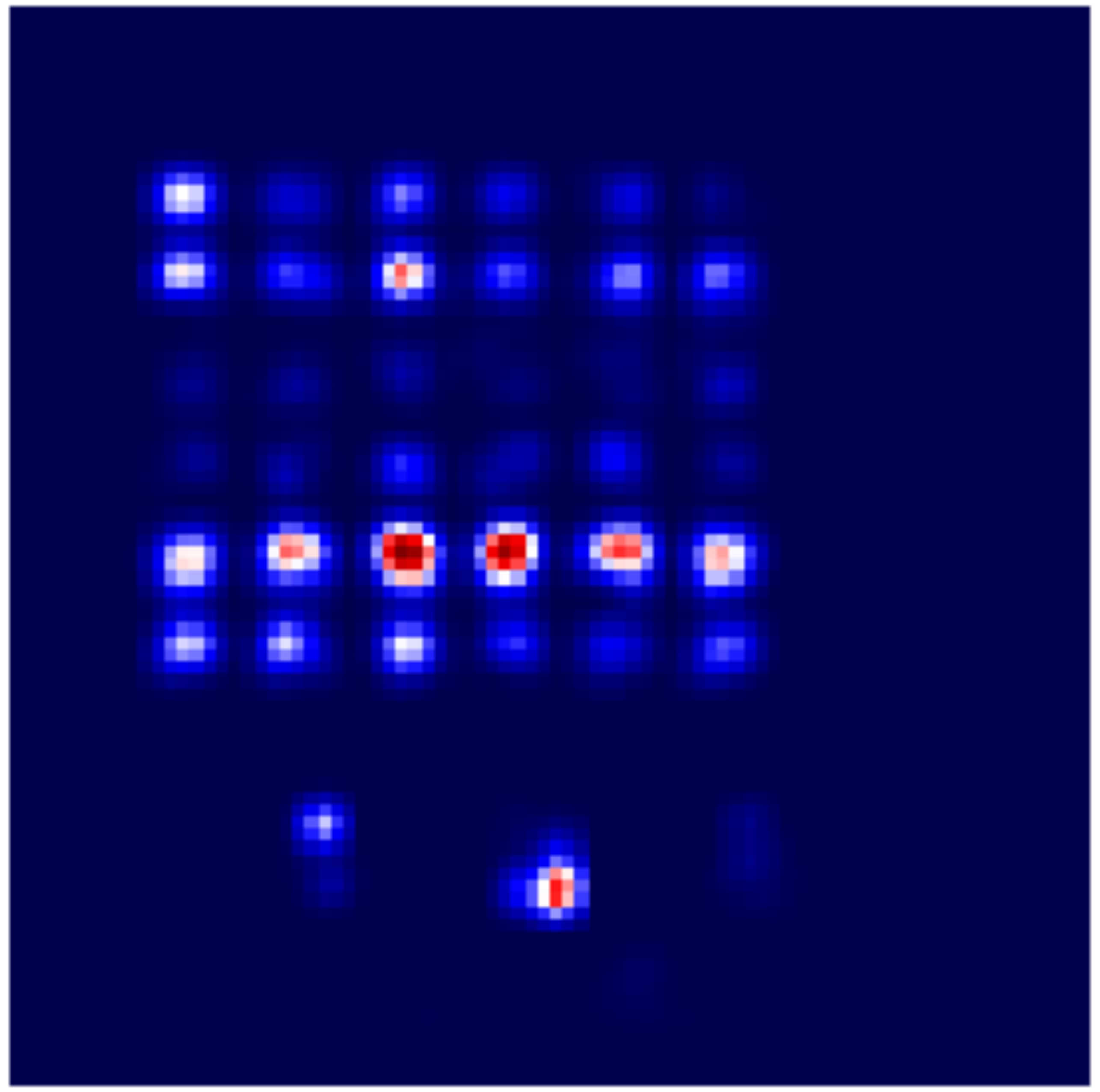
Worst



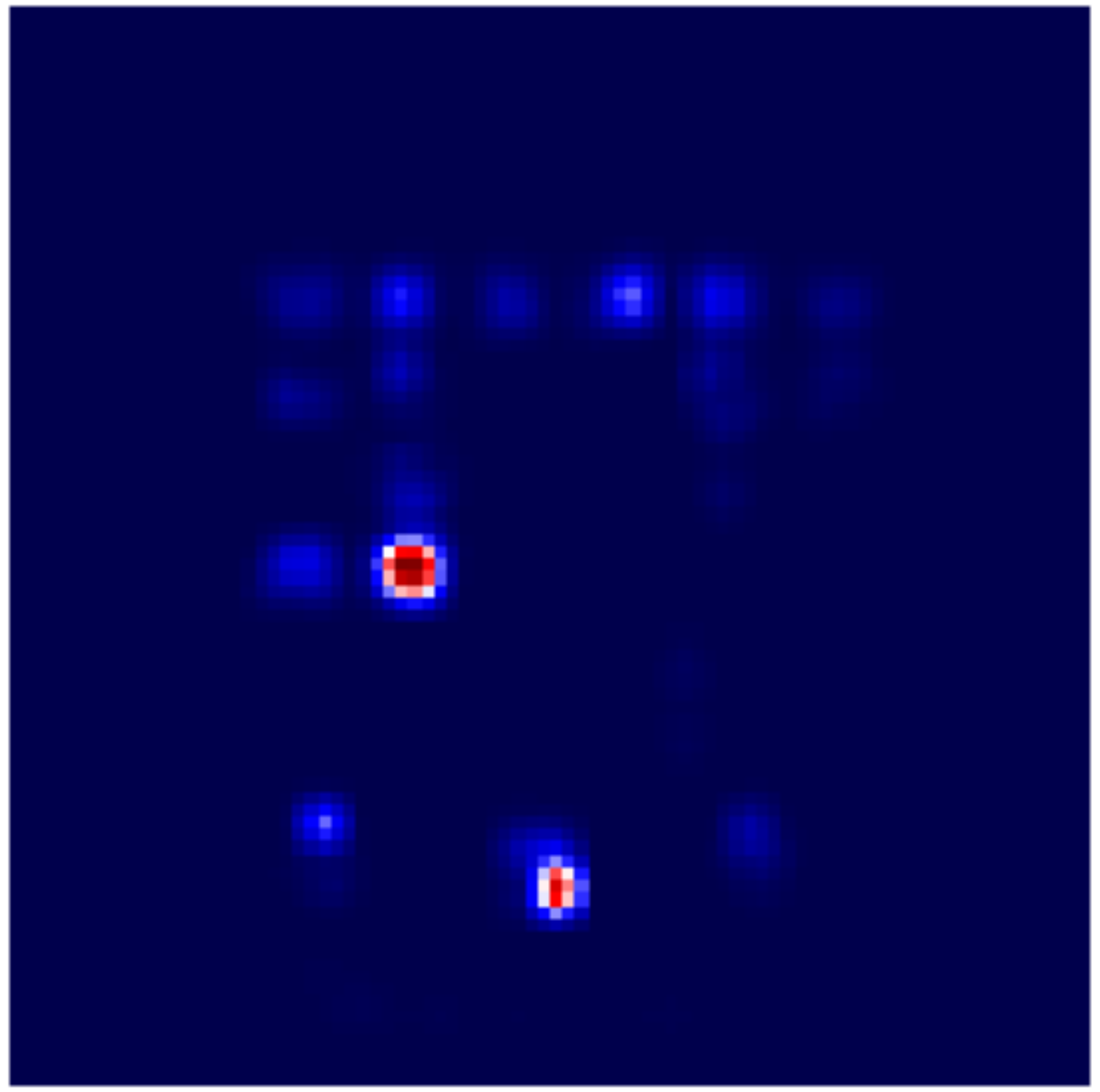
Best



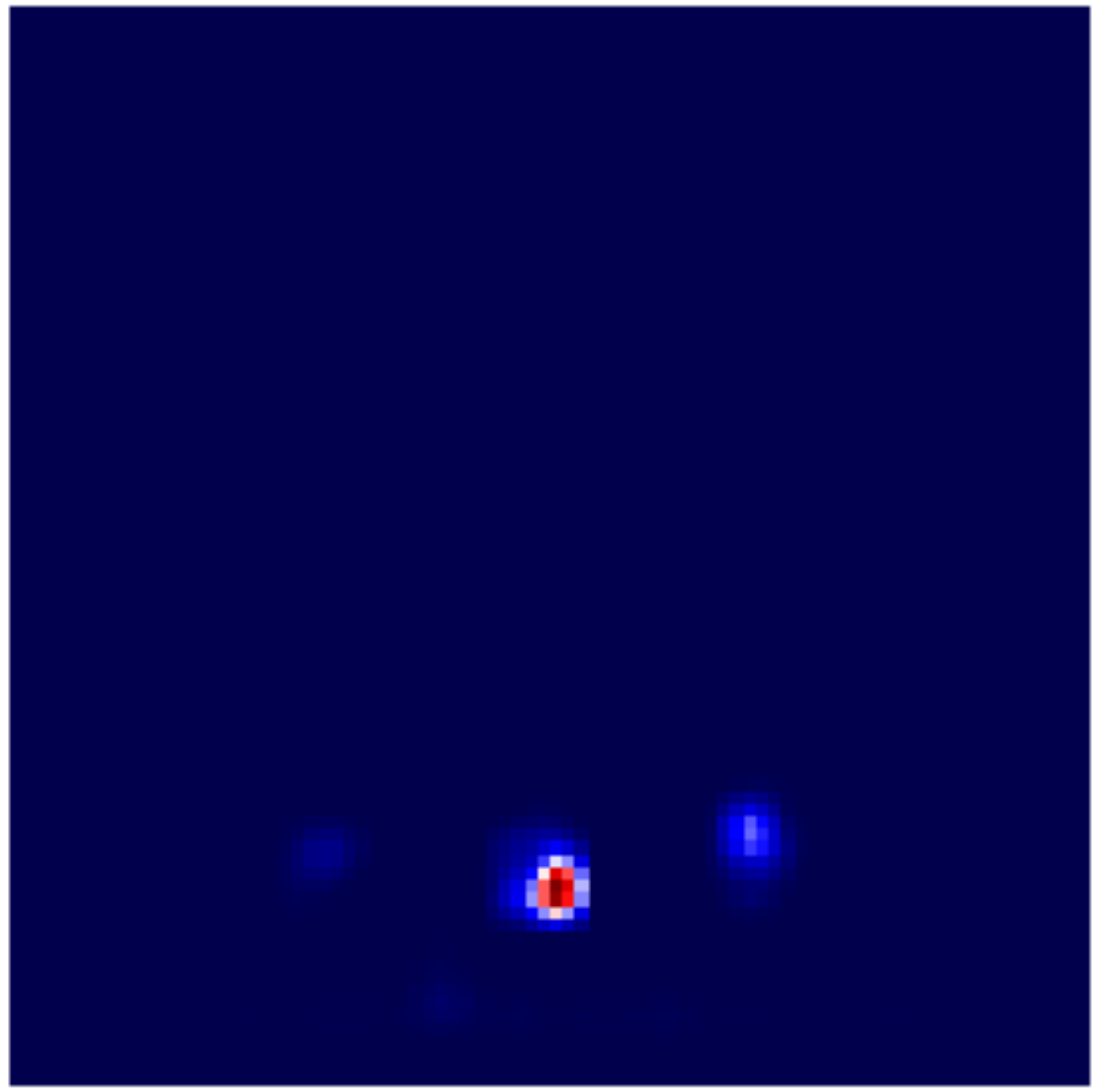
Reward heat maps



Min frame



Medium frame

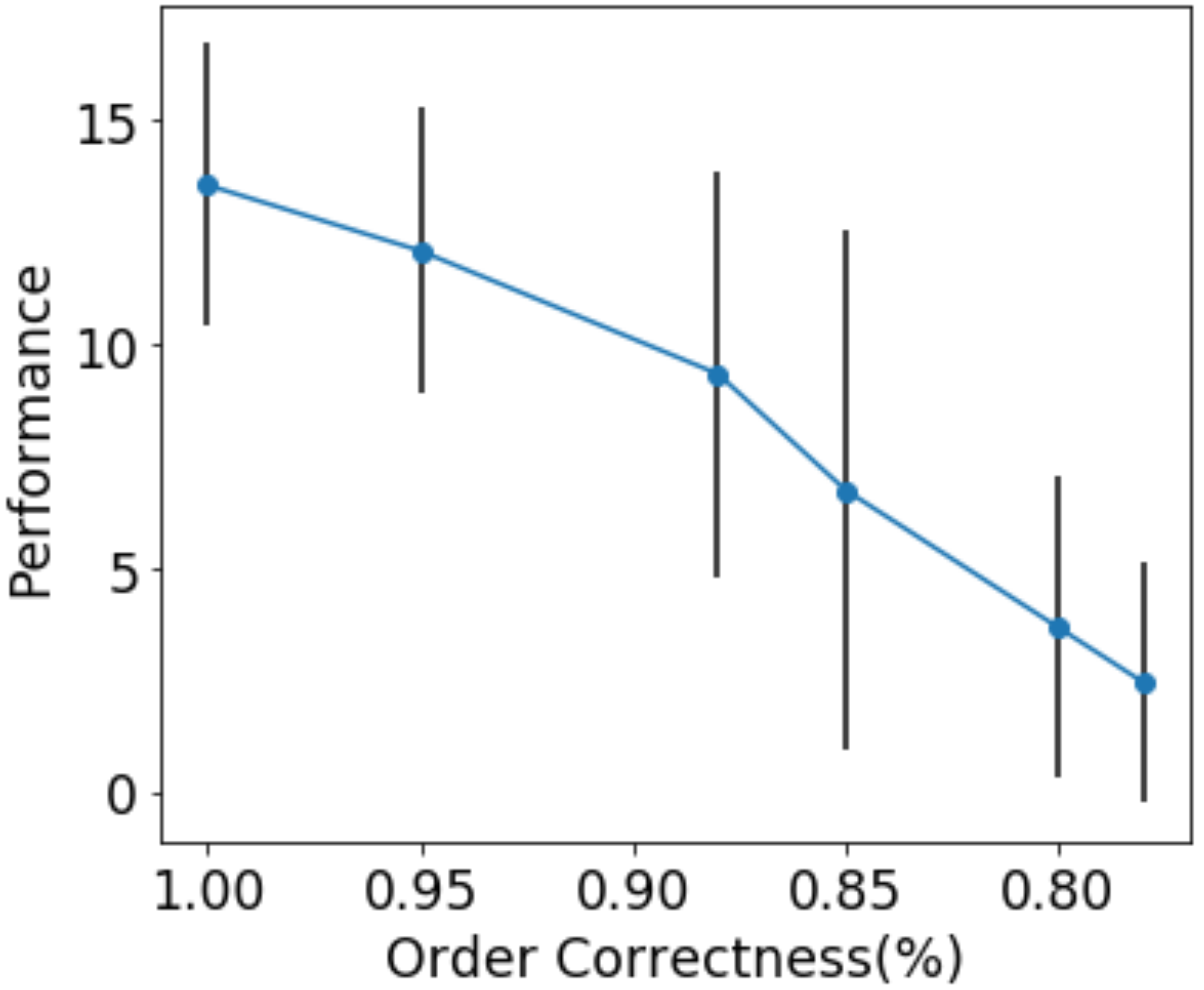


Max frame

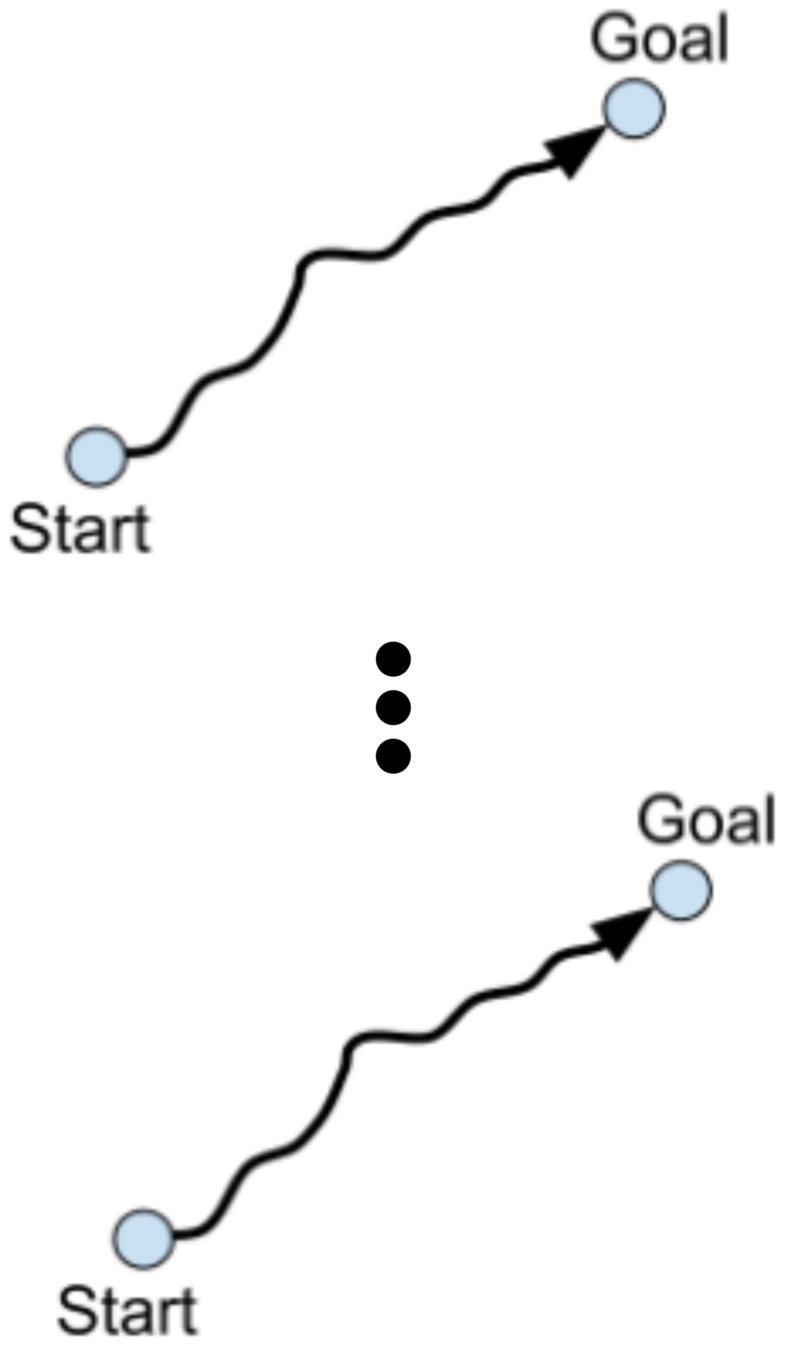
How hard is it to get rankings?

- Collect human trajectory rankings
- Have access to a performance metric, but infer more dense reward
- Watch a human (or agent) learn and noisily improve
- Add progressively more noise to near-optimal demonstrations

Robustness to pairwise ranking noise



D-REX: Auto-generated rankings

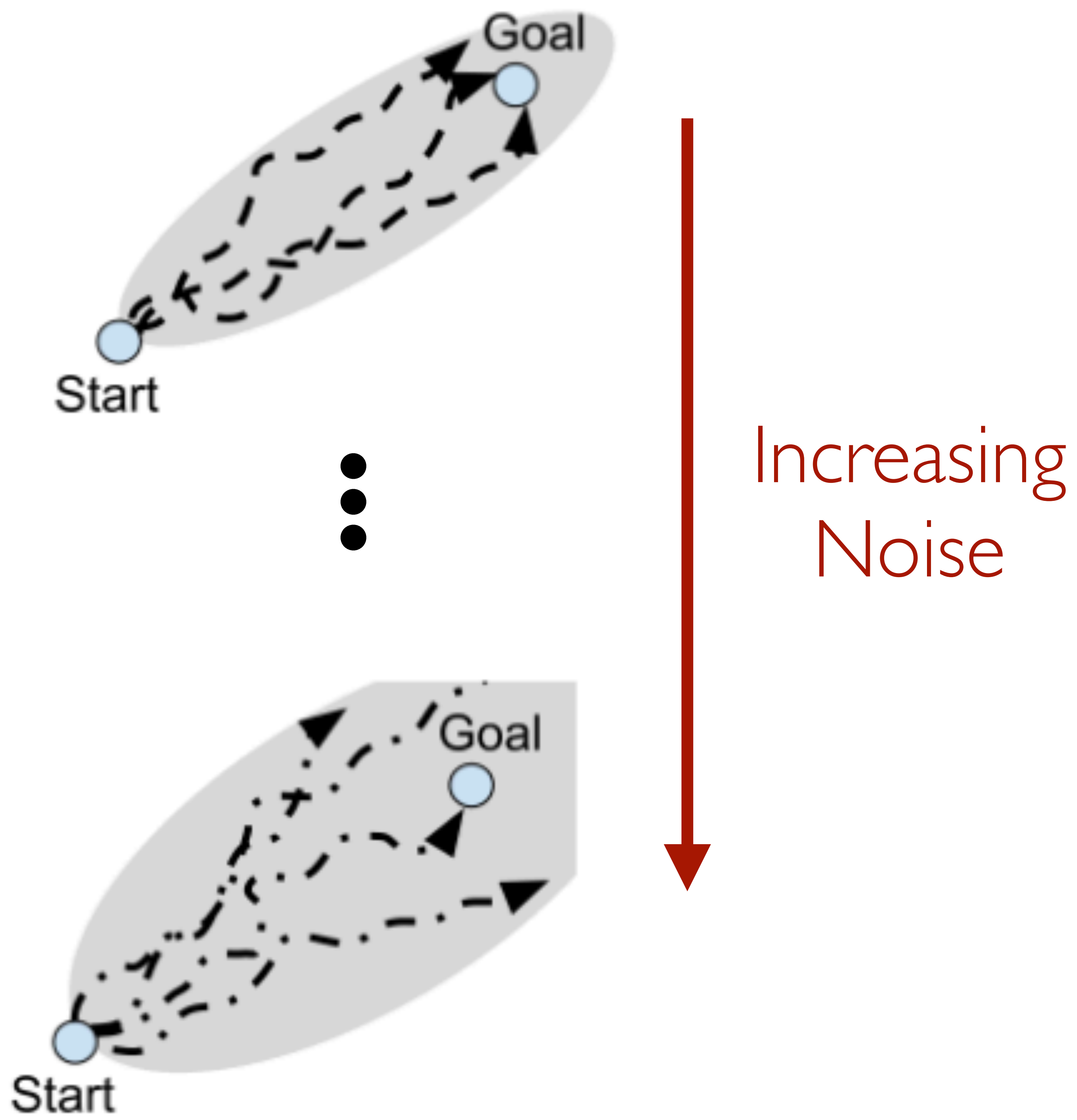


Unranked Demonstrations

Behavioral Cloning

$$\pi_{BC}$$

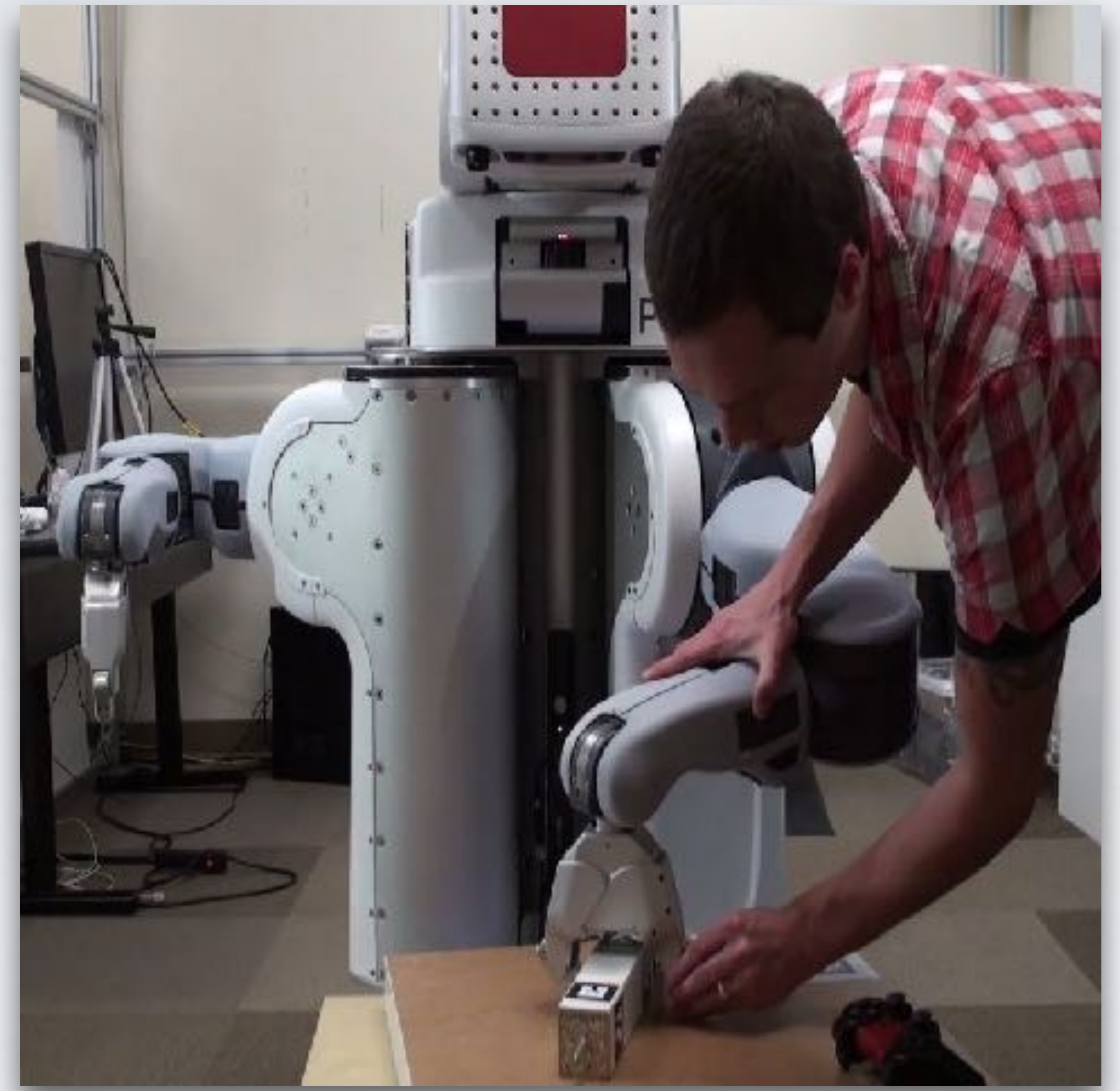
Noise-modified Rollouts



“Ranked” Trajectories

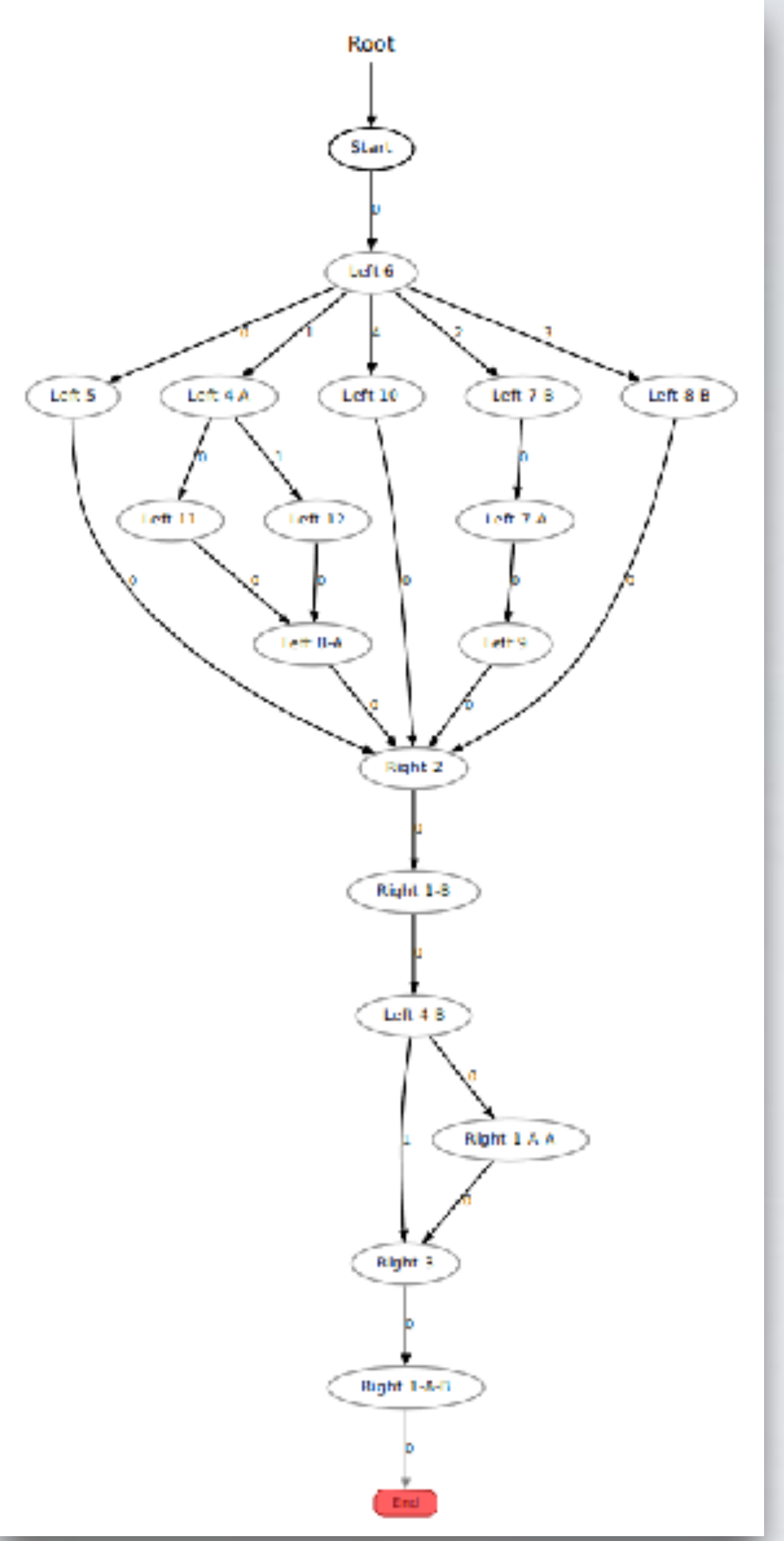
D. Brown, W. Goo, and S. Niekum.
Ranking-Based Reward Extrapolation without Rankings
Conference on Robot Learning (CoRL), October 2019.

Learning a task plan: Finite state automata



Unsegmented demonstrations of multi-step tasks

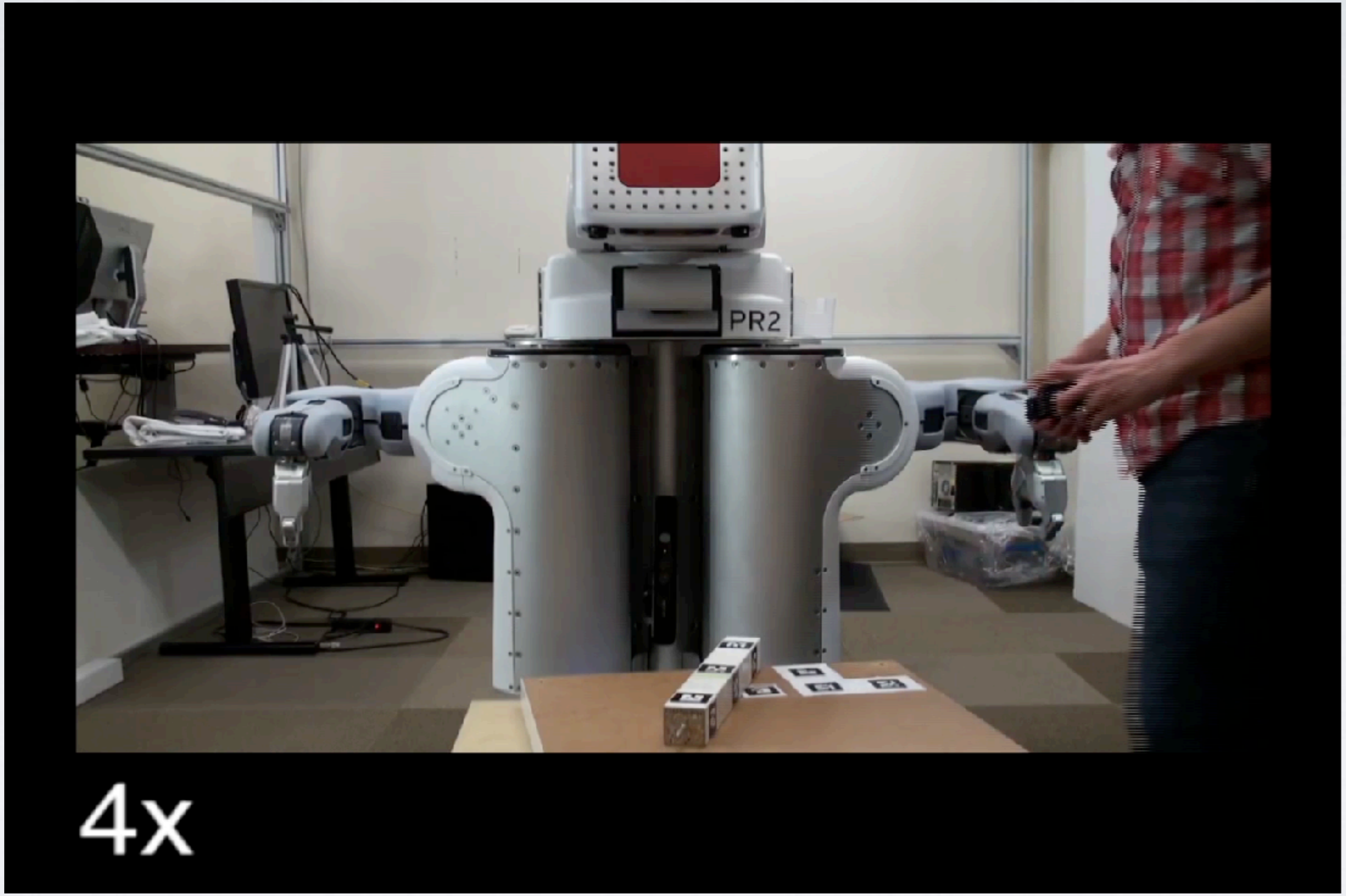
?



Finite-state task representation

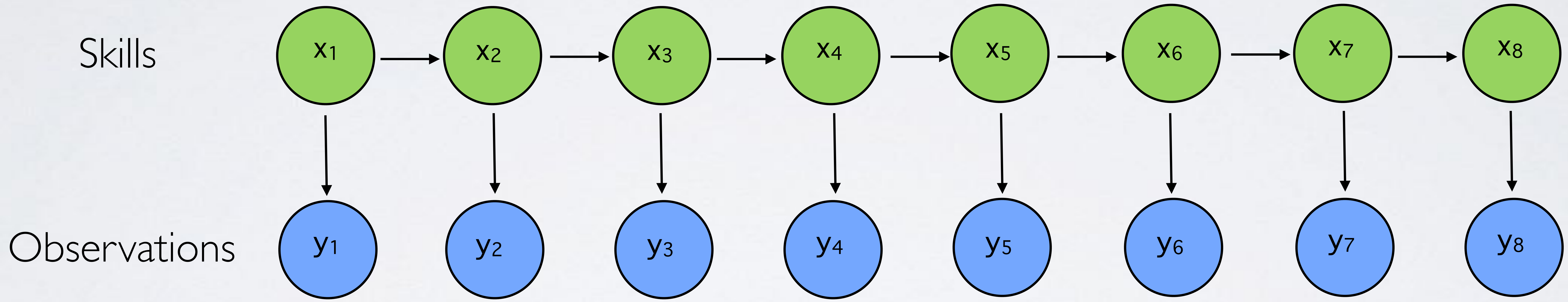
[Niekum et al. 2013]

Learning a task plan: Finite state automata



[Niekum et al. 2013]

Learning a task plan: Finite state automata

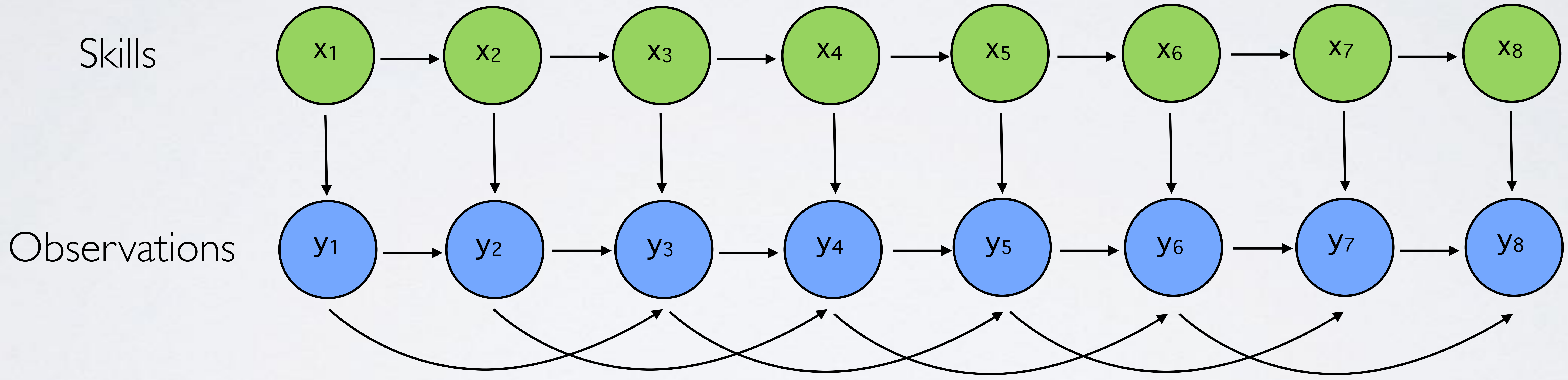


Standard Hidden Markov Model

[Niekum et al. 2013]

Learning a task plan: Finite state automata

$$\mathbf{y}_t^{(i)} = \sum_{j=1}^{r} A_{j,z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})$$

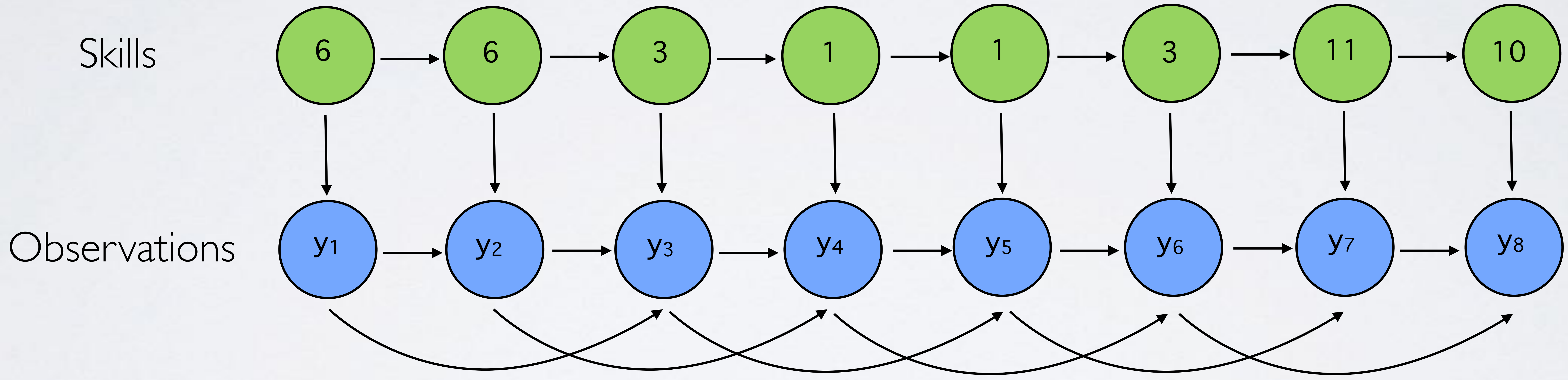


Autoregressive Hidden Markov Model

[Niekum et al. 2013]

Learning a task plan: Finite state automata

$$\mathbf{y}_t^{(i)} = \sum_{j=1}^{r} A_{j,z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})$$

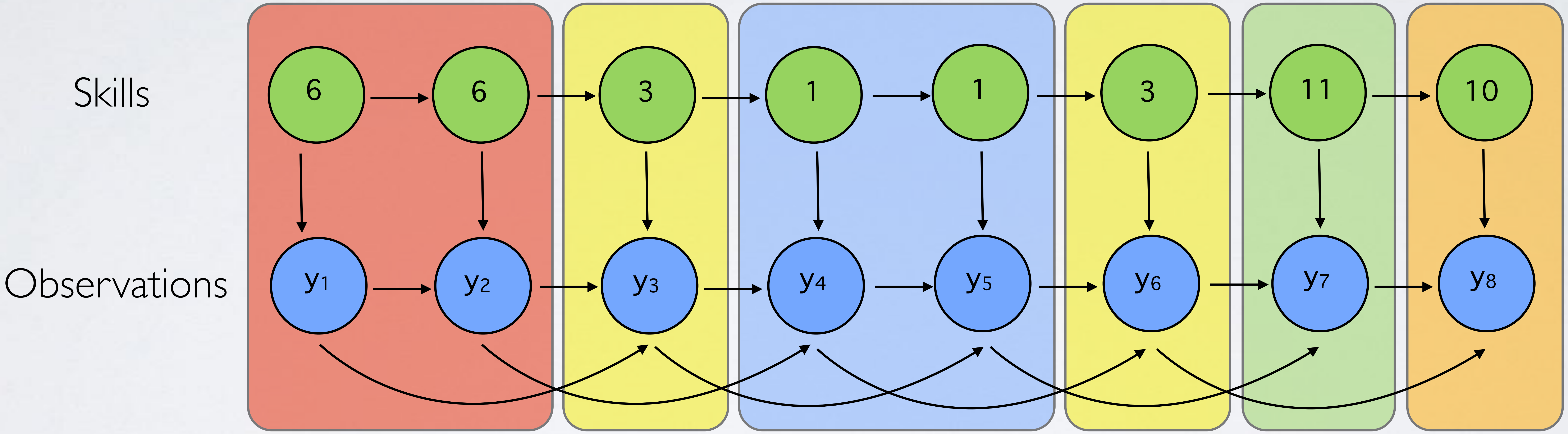


Autoregressive Hidden Markov Model

[Niekum et al. 2013]

Learning a task plan: Finite state automata

$$\mathbf{y}_t^{(i)} = \sum_{j=1}^{r} A_{j,z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})$$



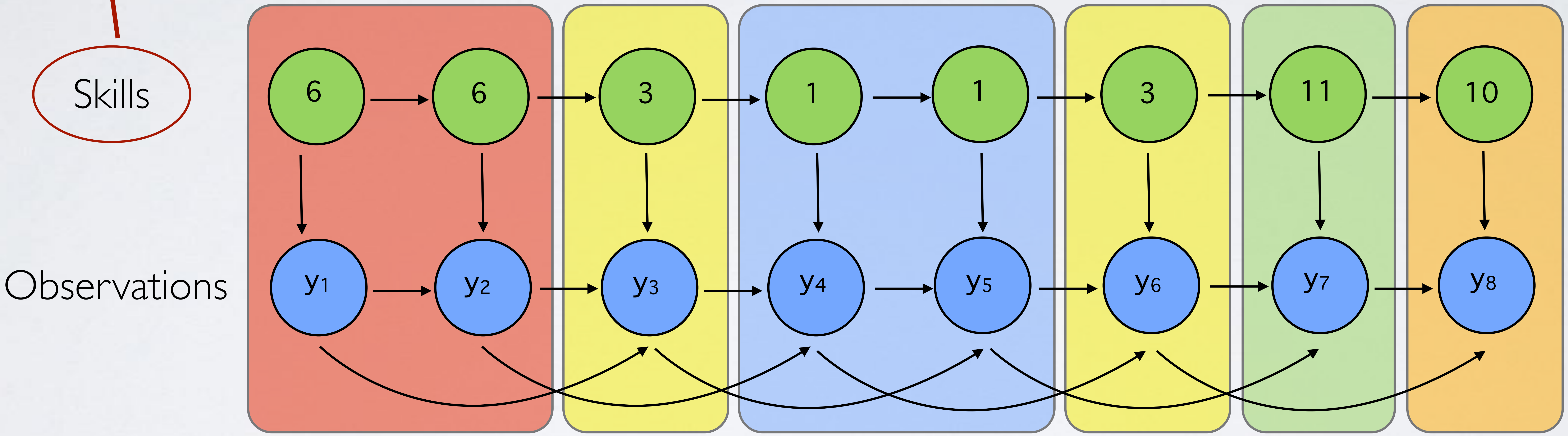
Autoregressive Hidden Markov Model

[Niekum et al. 2013]

Learning a task plan: Finite state automata

$$\mathbf{y}_t^{(i)} = \sum_{j=1}^{r} A_{j,z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})$$

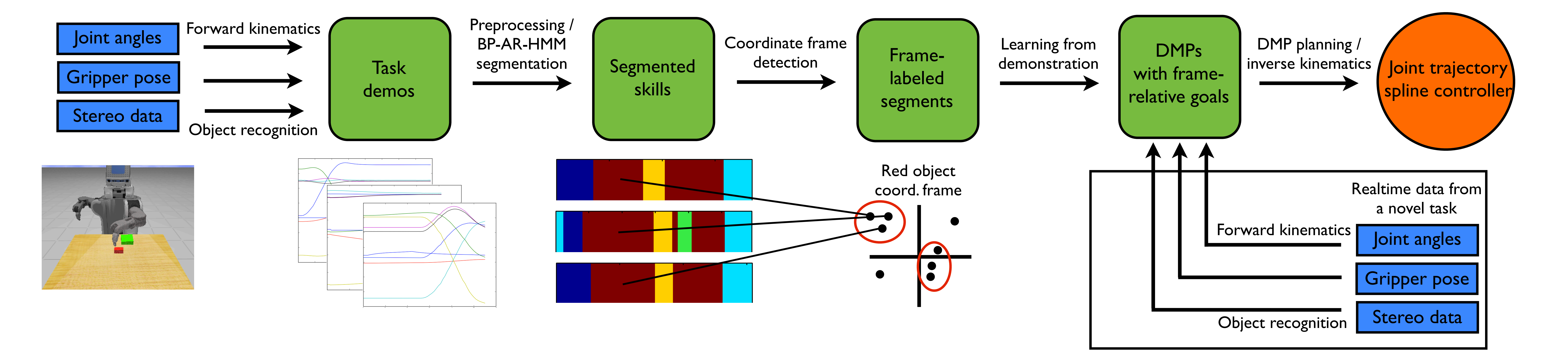
unknown number!



Beta Process Autoregressive Hidden Markov Model

[Niekum et al. 2013]

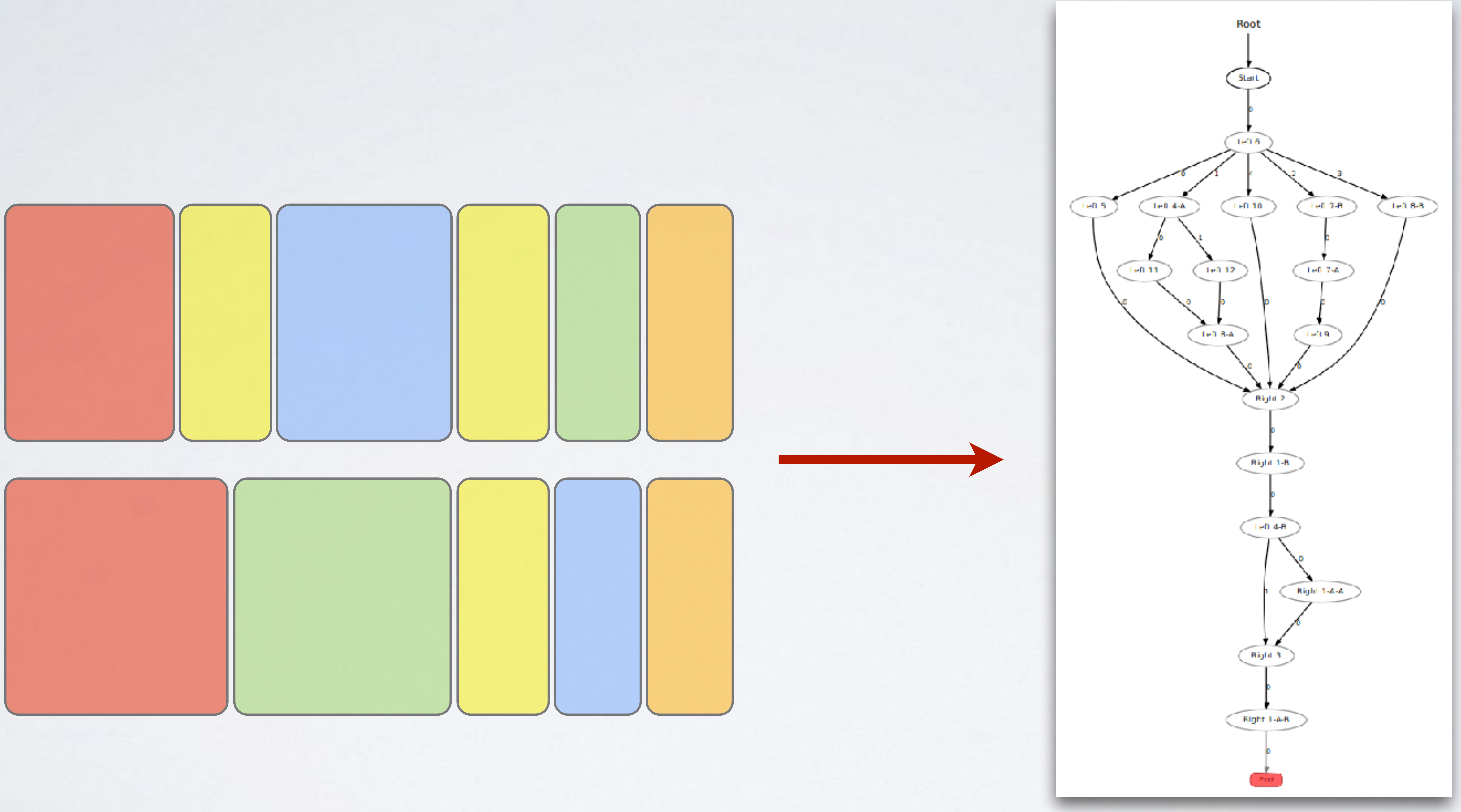
Learning a task plan: Finite state automata



Learning multi-step tasks from unstructured demonstrations

[Niekum et al. 2013]

Learning a task plan: Finite state automata

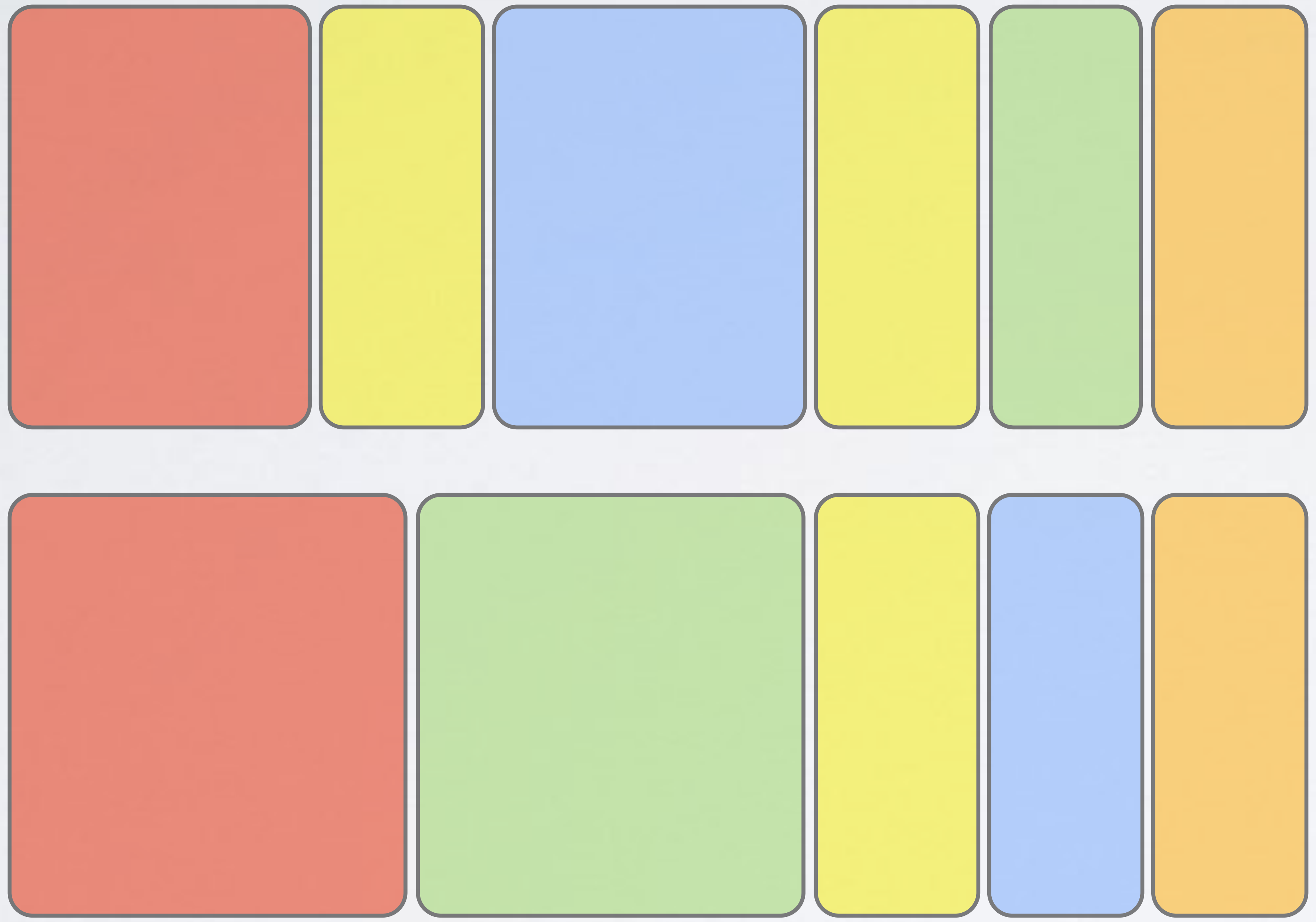


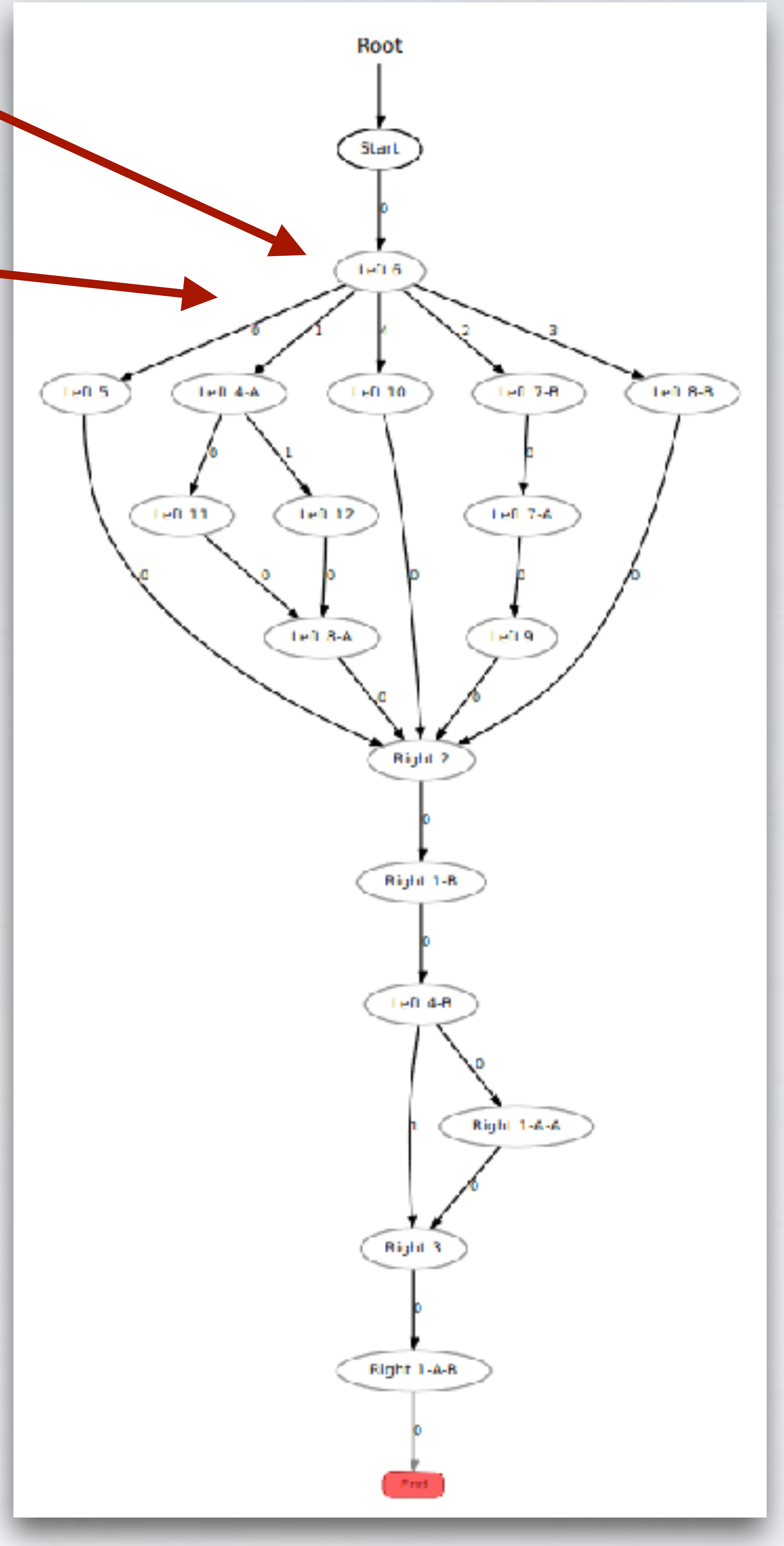
[Niekum et al. 2013]

Learning a task plan: Finite state automata

Controller built from motion category examples

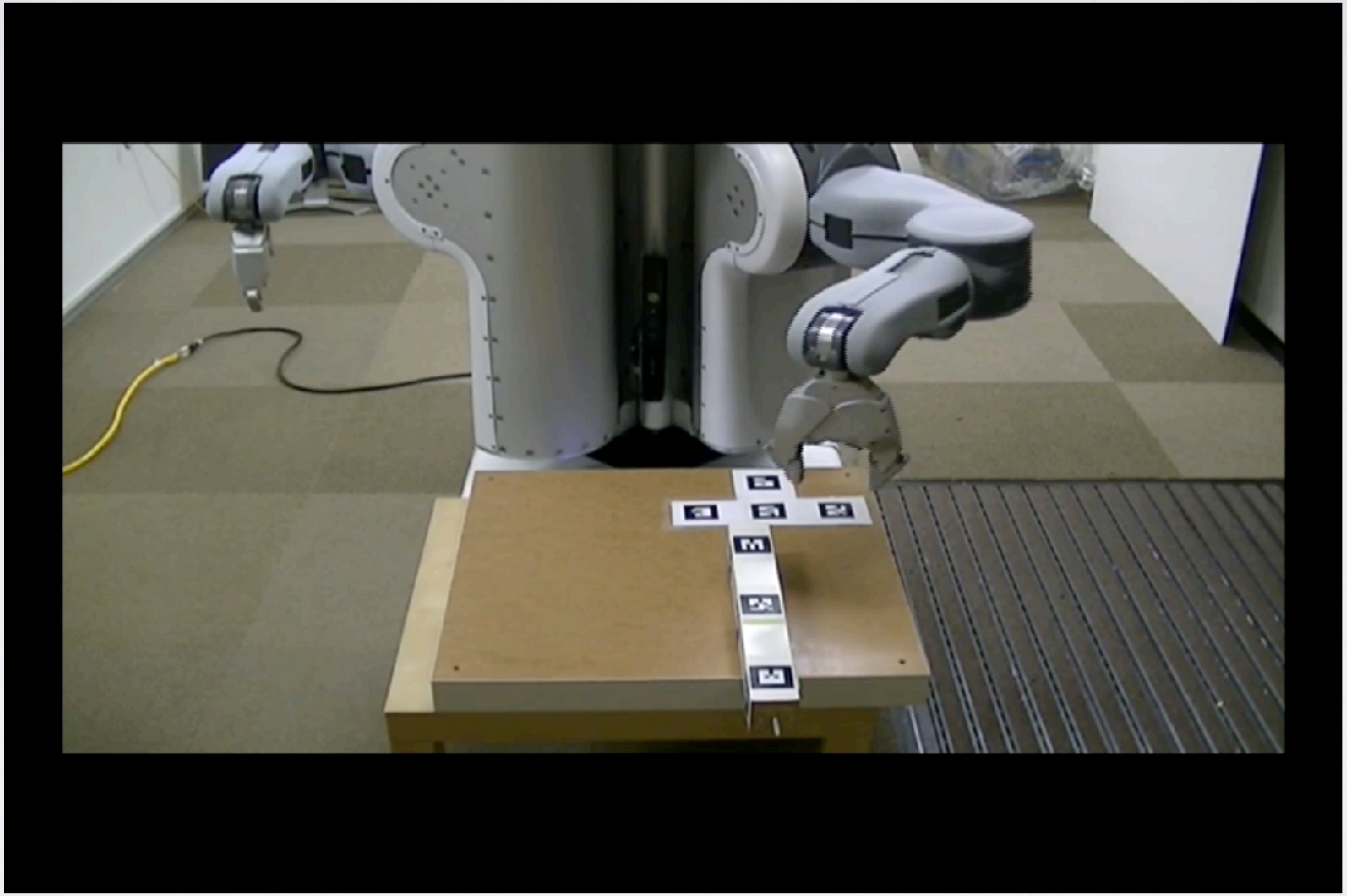
Classifier built from robot percepts





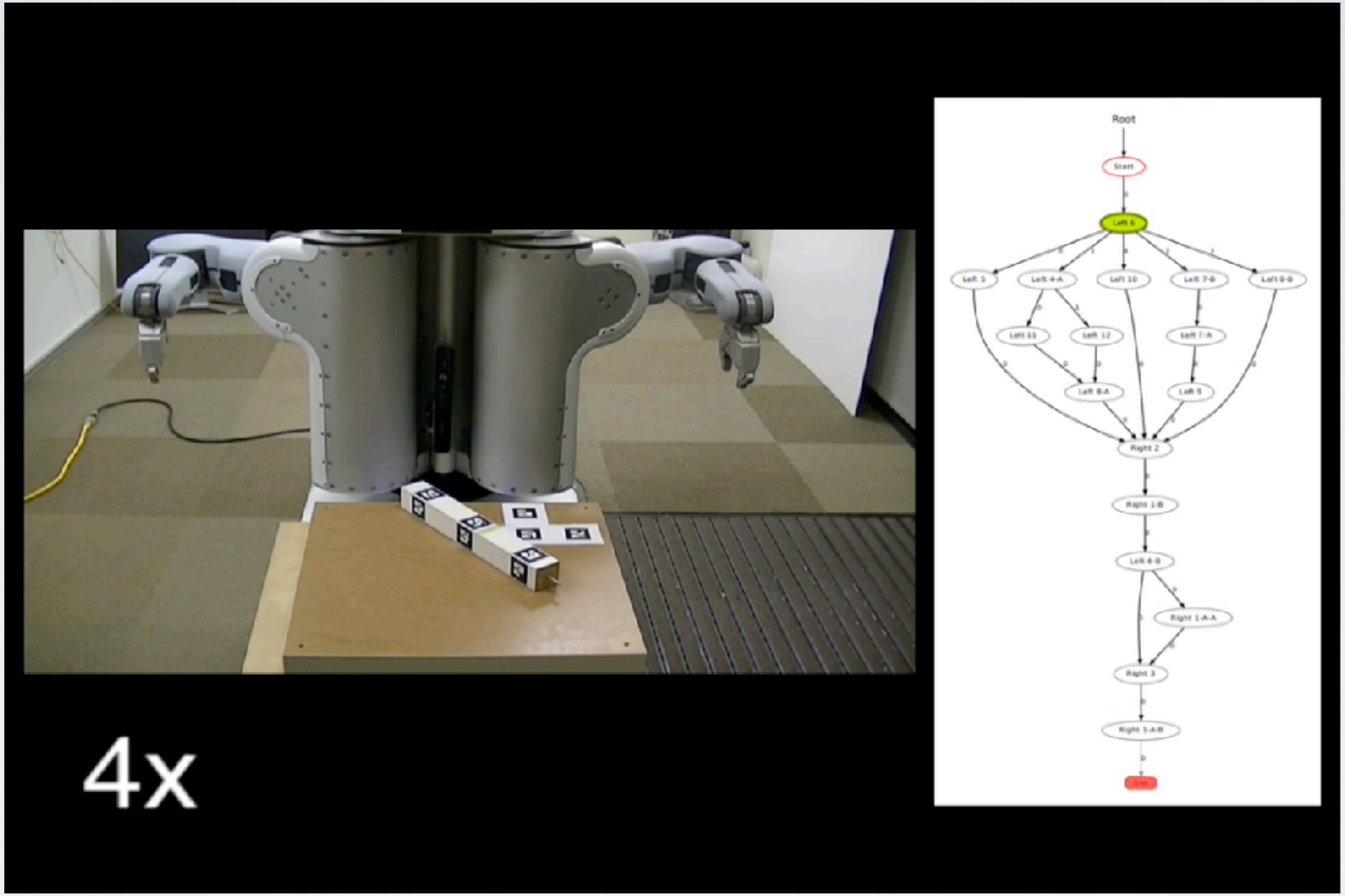
[Niekum et al. 2013]

Interactive corrections



[Niekum et al. 2013]

Replay with corrections: missed grasp

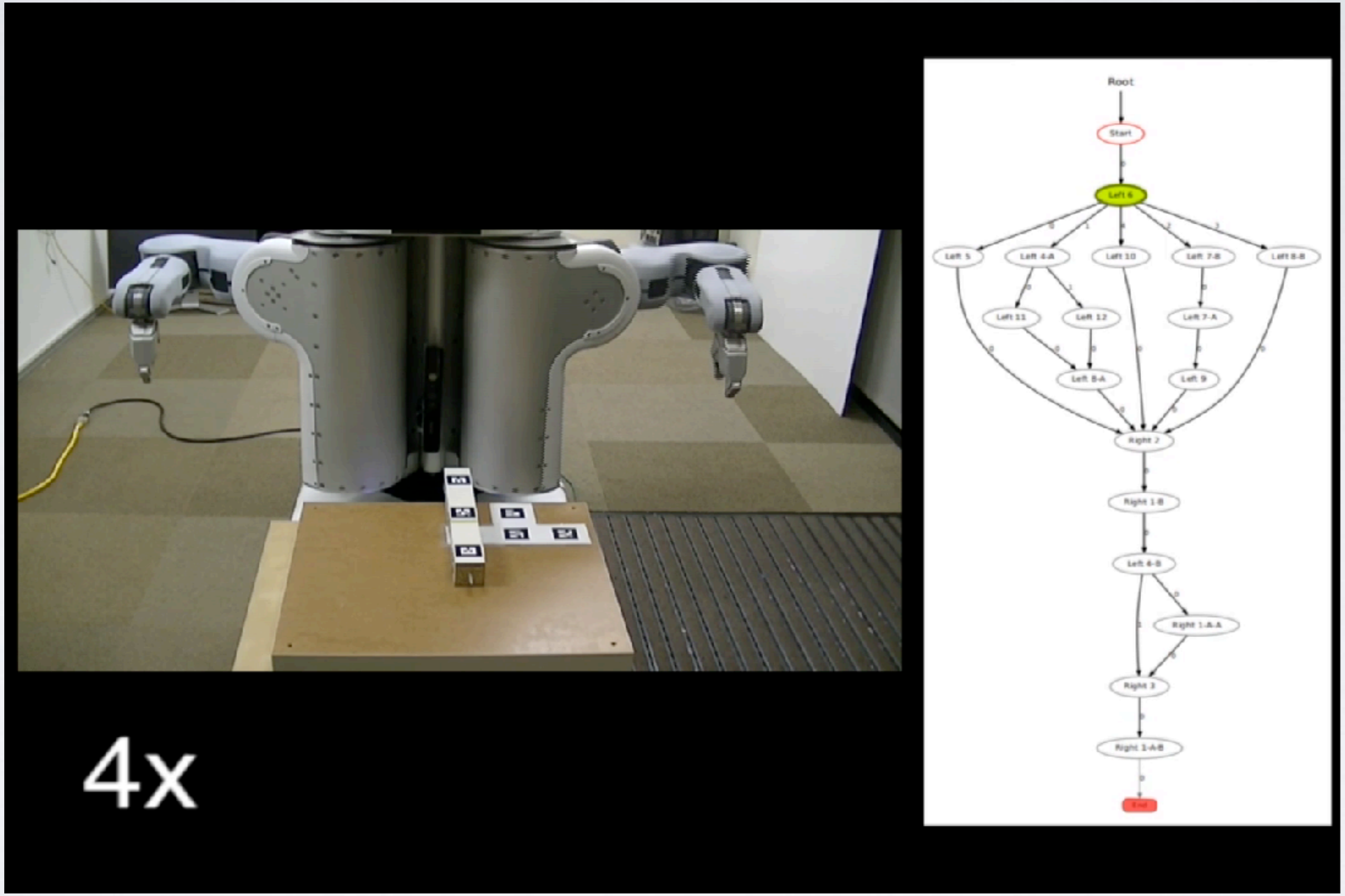


[Niekum et al. 2013]

Replay with corrections: too far away

[Niekum et al. 2013]

Replay with corrections: full run



[Niekum et al. 2013]

The Personal Autonomous Robotics Lab