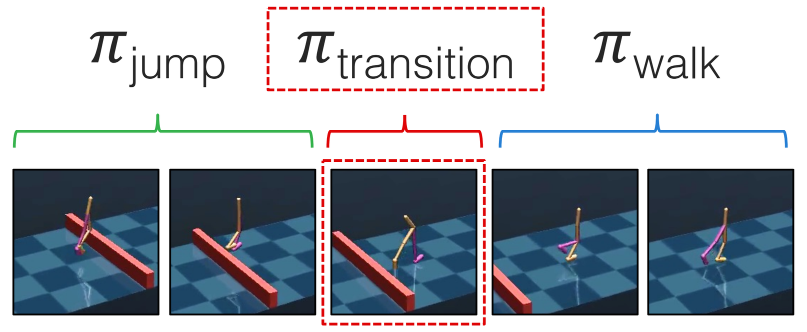


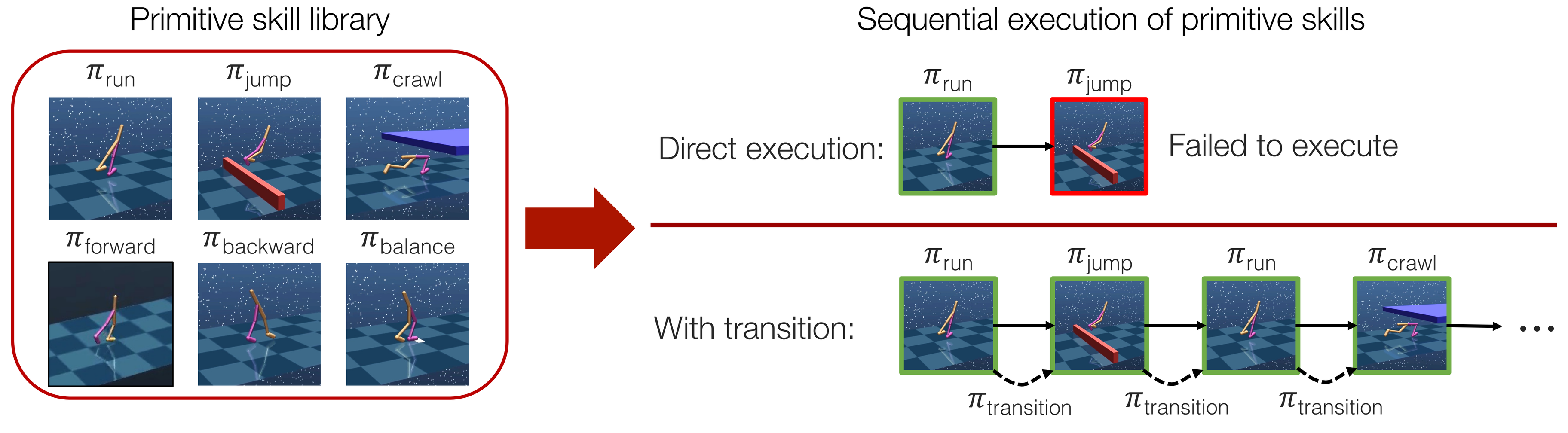
Composing Complex Skills by Learning Transition Policies

Youngwoon Lee* Shao-Hua Sun* Sriram Somasundaram Edward S. Hu Joseph J. Lim

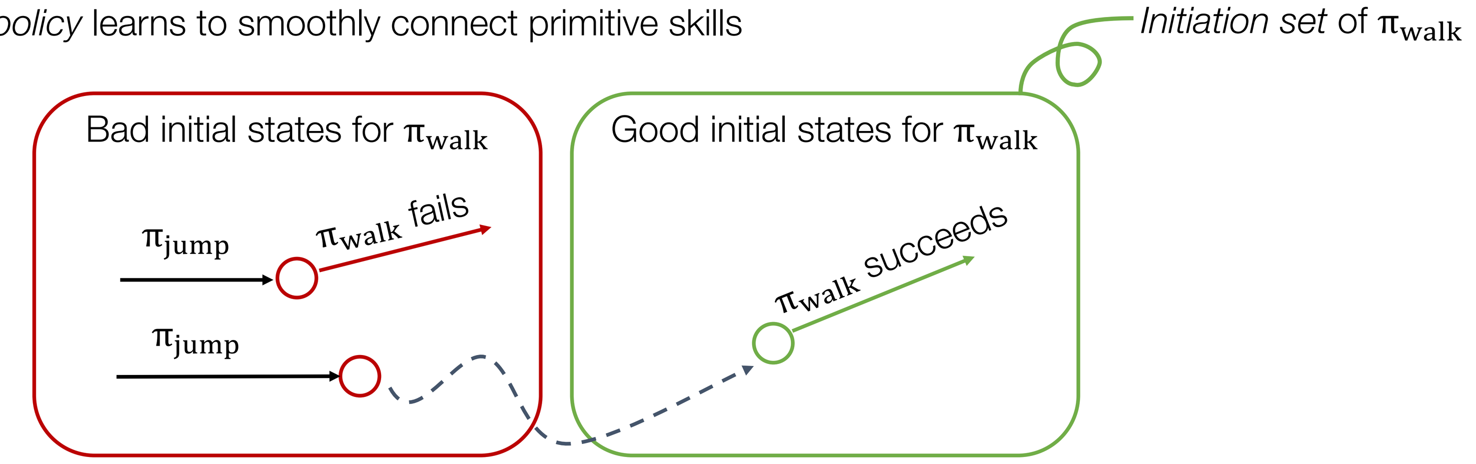
USC

Composing Complex Skills

- Primitive skills can be achieved by hard-coding, RL, and imitation learning
- We can compose a complex skill by sequentially executing primitive skills
- However, an ending state of a primitive skill may not be a good state to initiate the following skill



- A *transition policy* learns to smoothly connect primitive skills



Transition policy execution

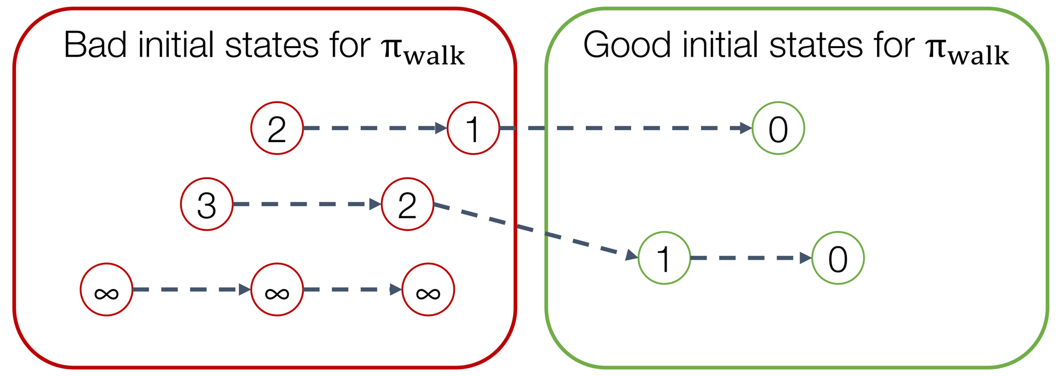
Proximity Reward

- A successful execution of a transition policy is defined by success of the following primitive skill
- The success/failure reward is too sparse to train a transition policy
- A *proximity predictor* learns to predict the proximity of a state to the initiation set

$$P(s) = \delta^{step(s)}$$

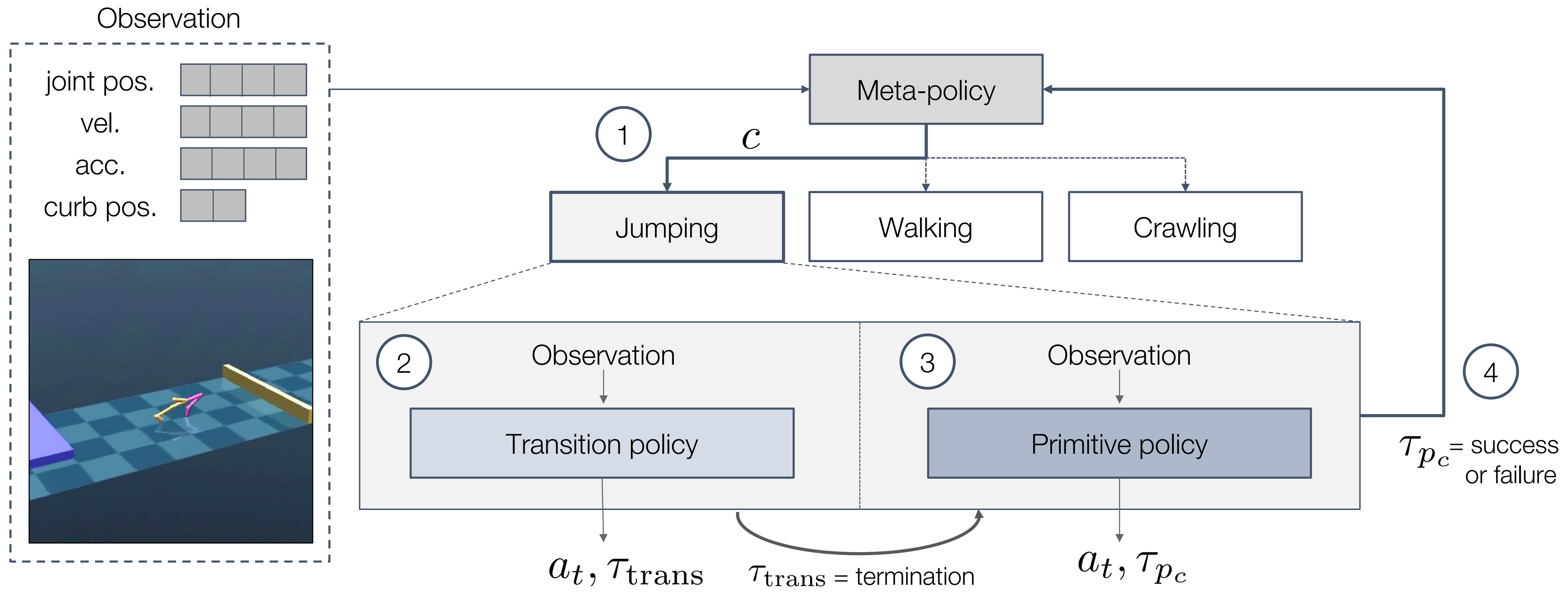
- We use the increase of predicted proximity to the initiation set as a dense reward

$$R_{\text{proximity}}(s_t, s_{t+1}) = P(s_{t+1}) - P(s_t)$$



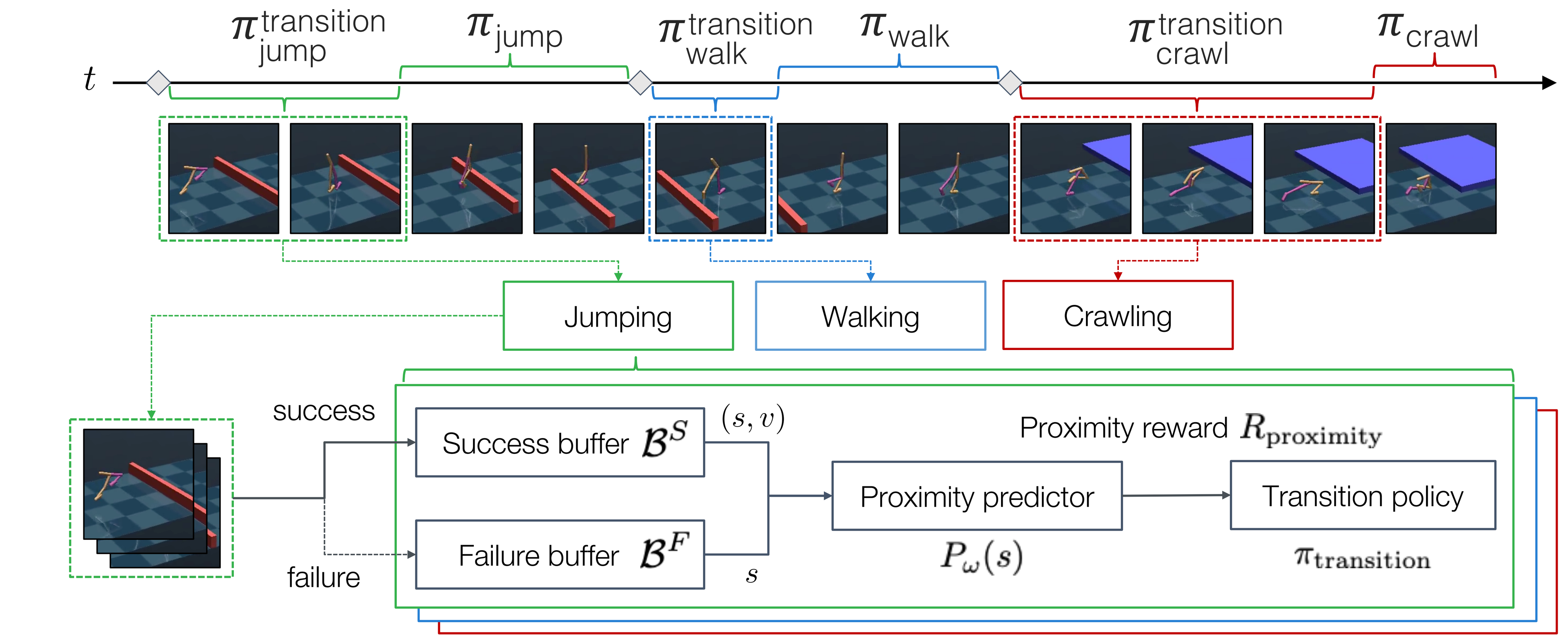
The numbers inside the circles (states) represent transition steps to the initiation set

Modular Framework with Transition Policies



(1) The meta-policy chooses a primitive policy of index c
(2) The corresponding transition policy helps initiate the chosen primitive policy
(3) The primitive policy executes the skill
(4) A success or failure signal for the primitive skill is produced

Training Transition Policies

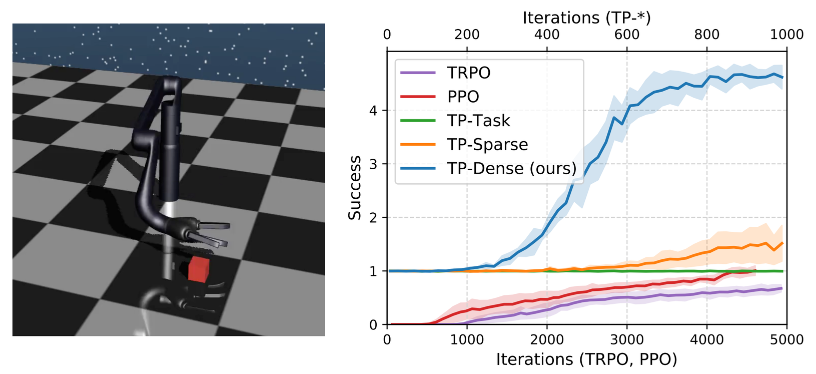


Jointly train proximity predictors and transition policies by optimizing the following objectives:

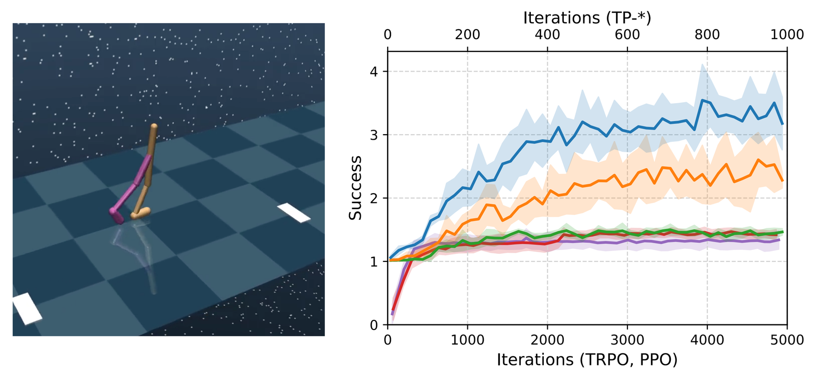
- Train proximity predictors: $L_P(\omega, \mathcal{B}^S, \mathcal{B}^F) = \frac{1}{2}\mathbb{E}_{(s,v)\sim\mathcal{B}^S}[(P_\omega(s) - v)^2] + \frac{1}{2}\mathbb{E}_{s\sim\mathcal{B}^F}[P_\omega(s)^2]$
- Train transition policies with proximity reward: $R_{\text{proximity}}(\phi) = \mathbb{E}_{(s_0,s_1,\ldots,s_T)\sim\pi_\phi}\left[\gamma^T P_\omega(s_T) + \sum_{t=0}^{T-1}\gamma^t(P_\omega(s_{t+1}) - P_\omega(s_t))\right]$

Results

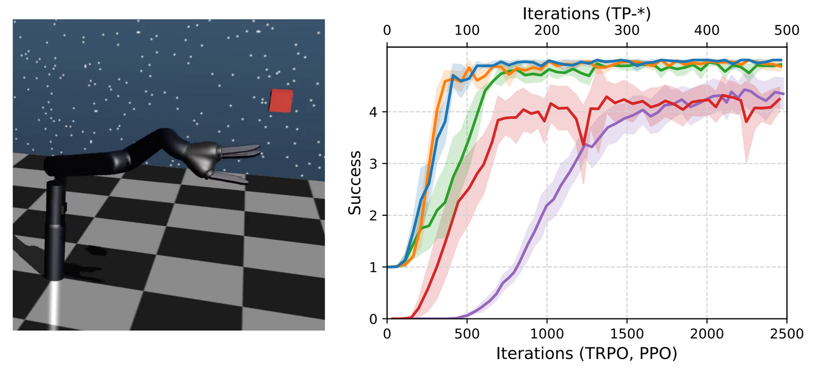
Baselines:
(1) TRPO, PPO: train a policy with a hand-engineered reward for 5x longer time — Dense reward
(2) TP-Task: train transition policies with a subtask completion reward
(3) TP-Sparse: train transition policies with a sparse proximity reward — Sparse reward
(4) TP-Dense (ours): train transition policies with a dense proximity reward



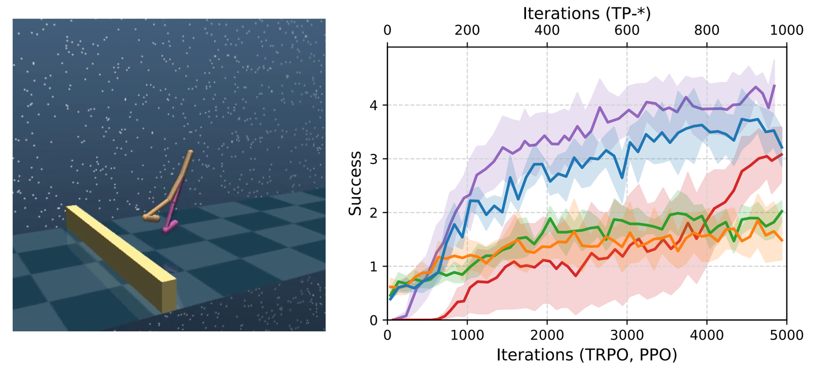
Repetitive picking up



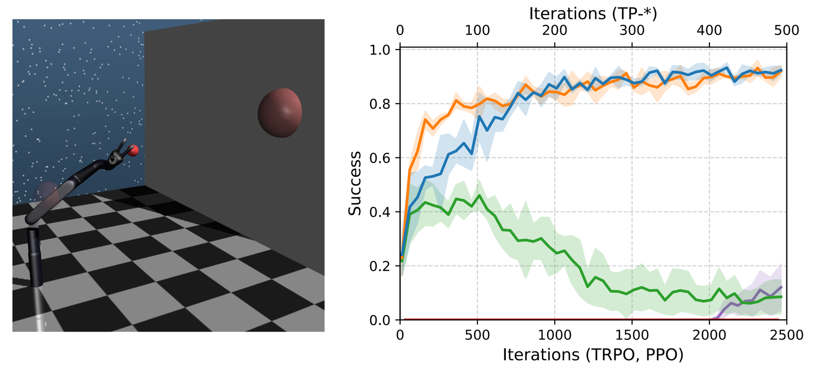
Patrol (Forward-Balance-Backward)



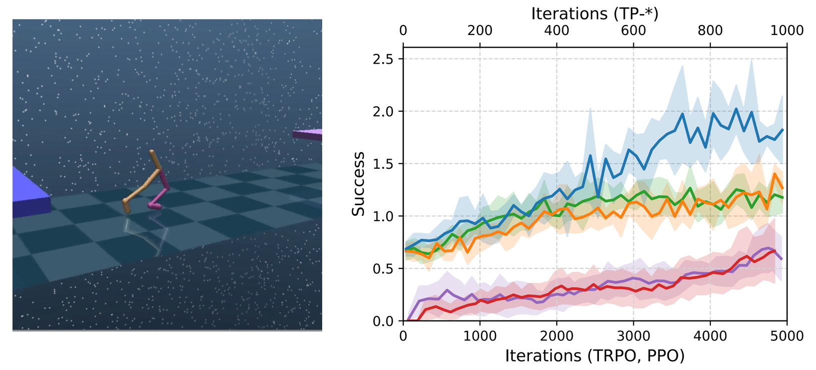
Repetitive catching



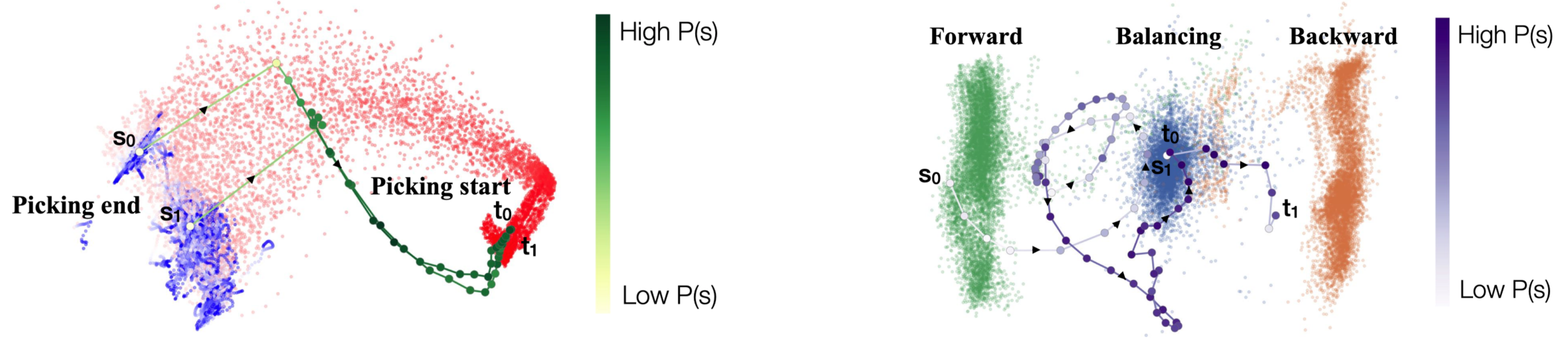
Hurdle (Forward-Jump)



Serve (Toss-Hit)



Obstacle course (Forward-Jump-Crawl)



Transition trajectories of "Repetitive picking up" and "Patrol"

Code is available

- Code and videos are available at https://youngwoon.github.io/transition
- Corresponding author: Youngwoon Lee (lee504@usc.edu)