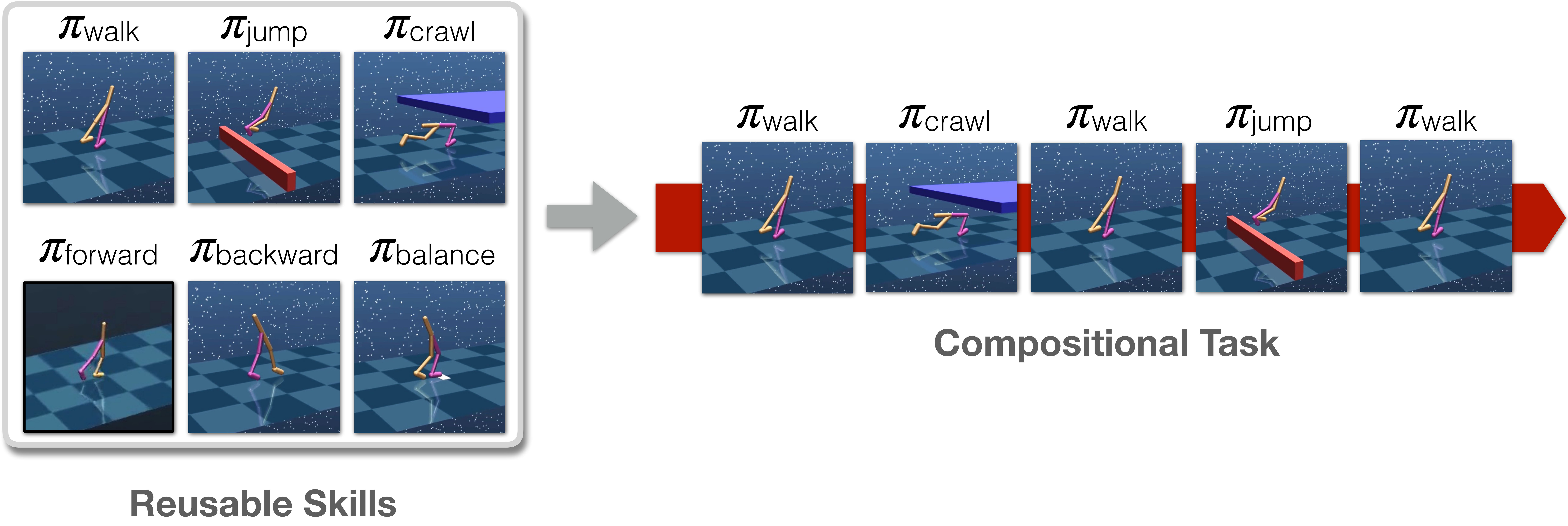
Composing Complex Skills by Learning Transition Policies

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

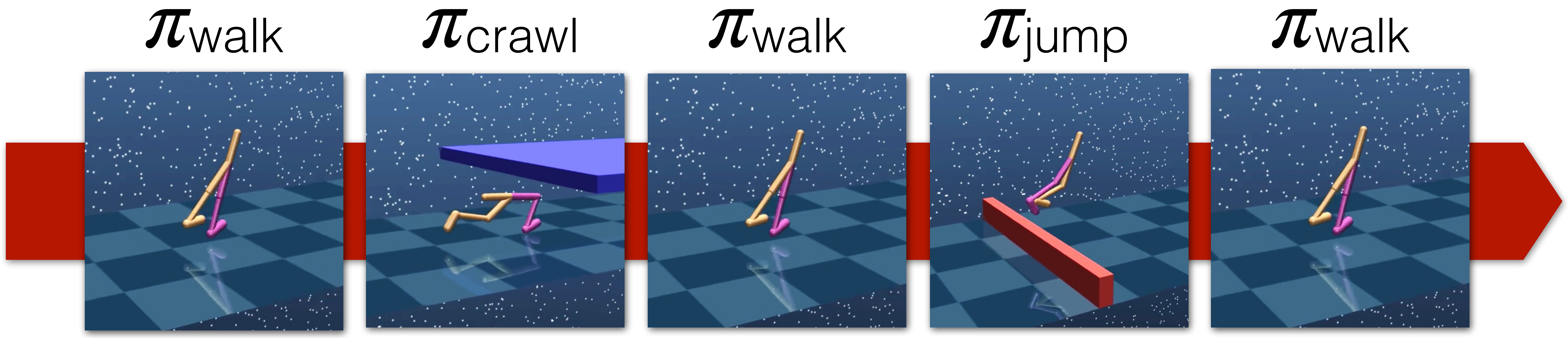
Presented in ICLR 2019

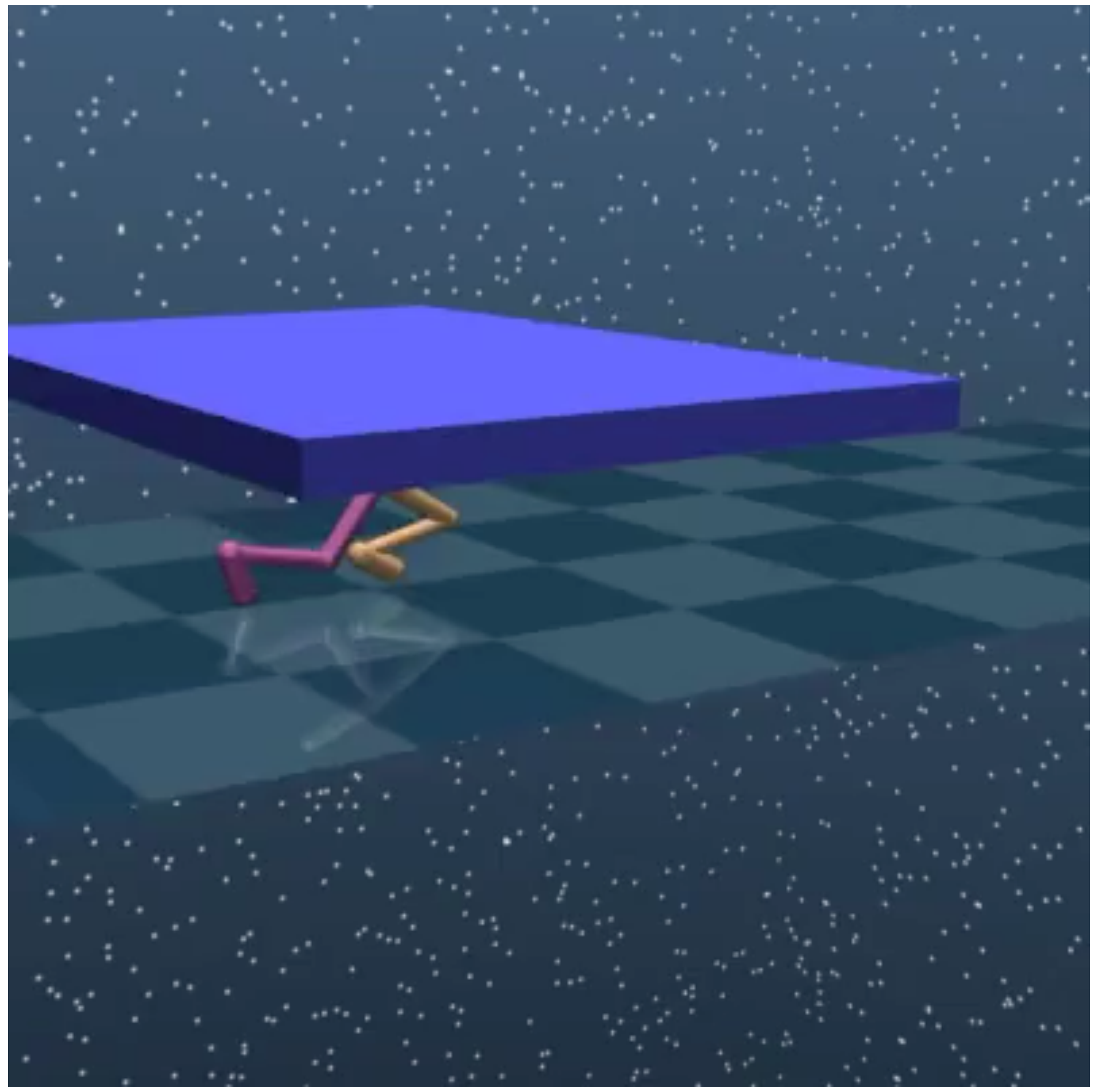


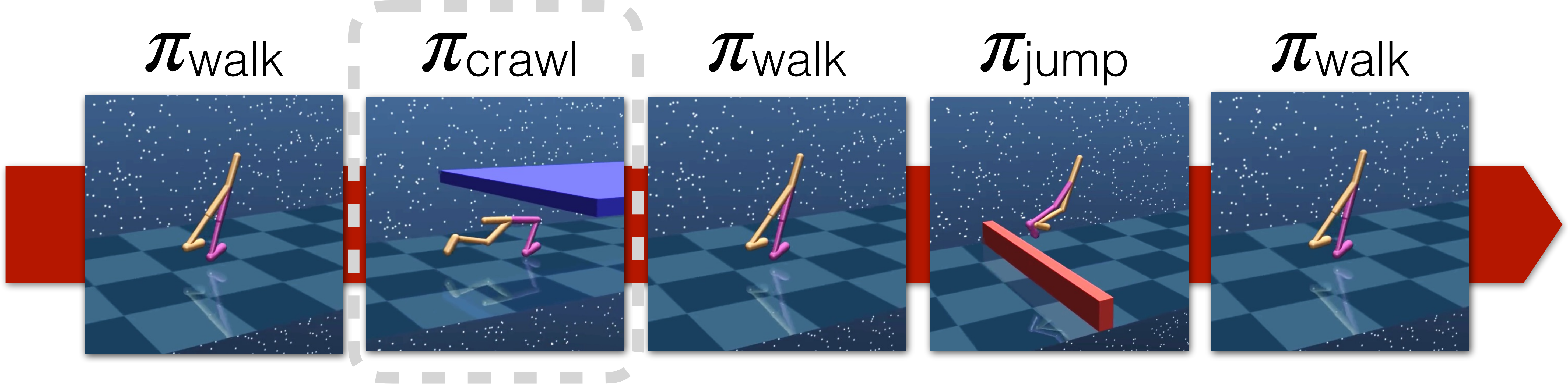


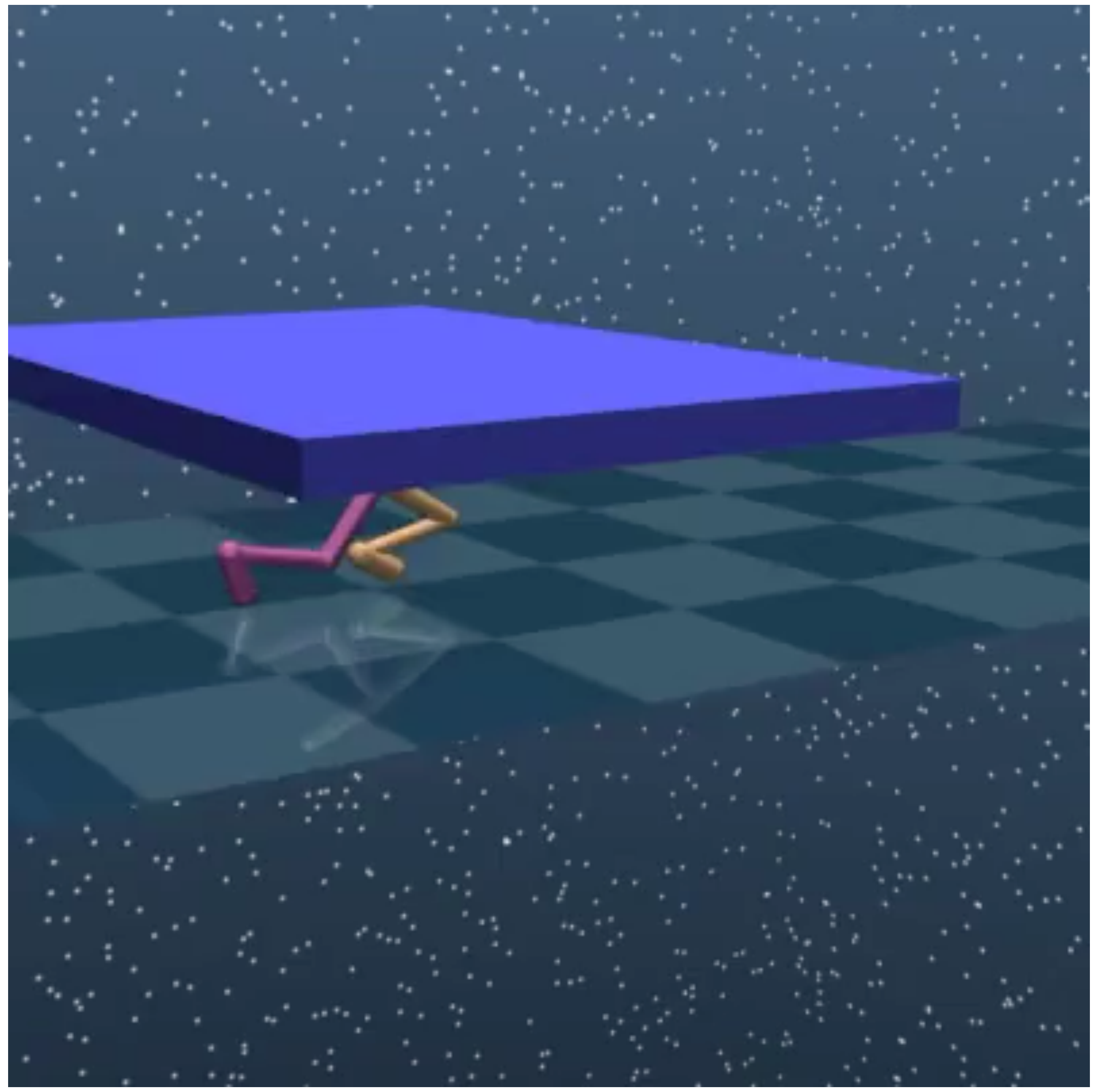


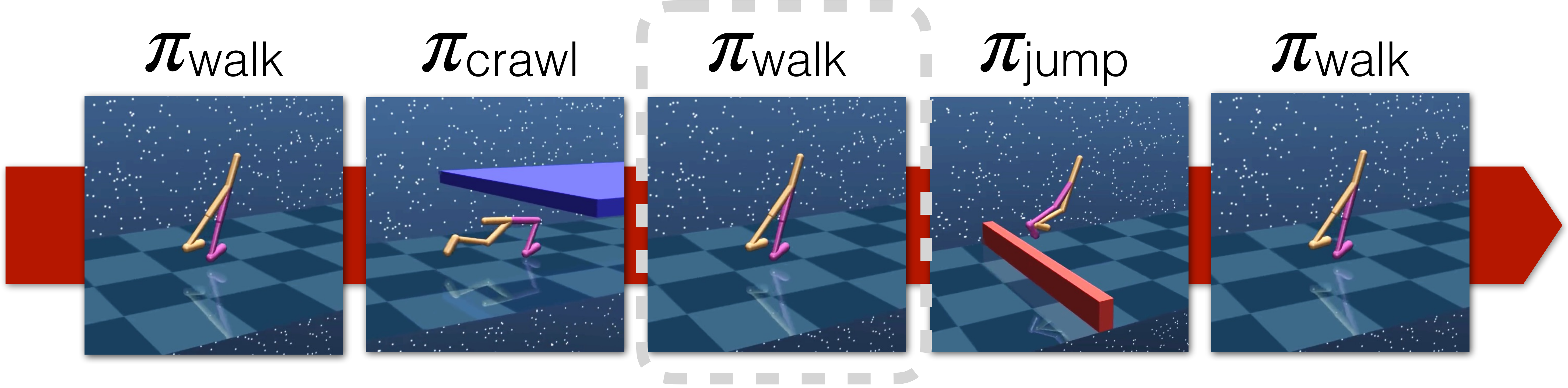
π_{walk}
π_{jump}
π_{crawl}
$\pi_{forward}$
$\pi_{backward}$
$\pi_{balance}$
Reusable Skills
π_{walk}
π_{crawl}
π_{walk}
π_{jump}
π_{walk}
Compositional Task


π_{walk}
π_{crawl}
π_{walk}
π_{jump}
π_{walk}

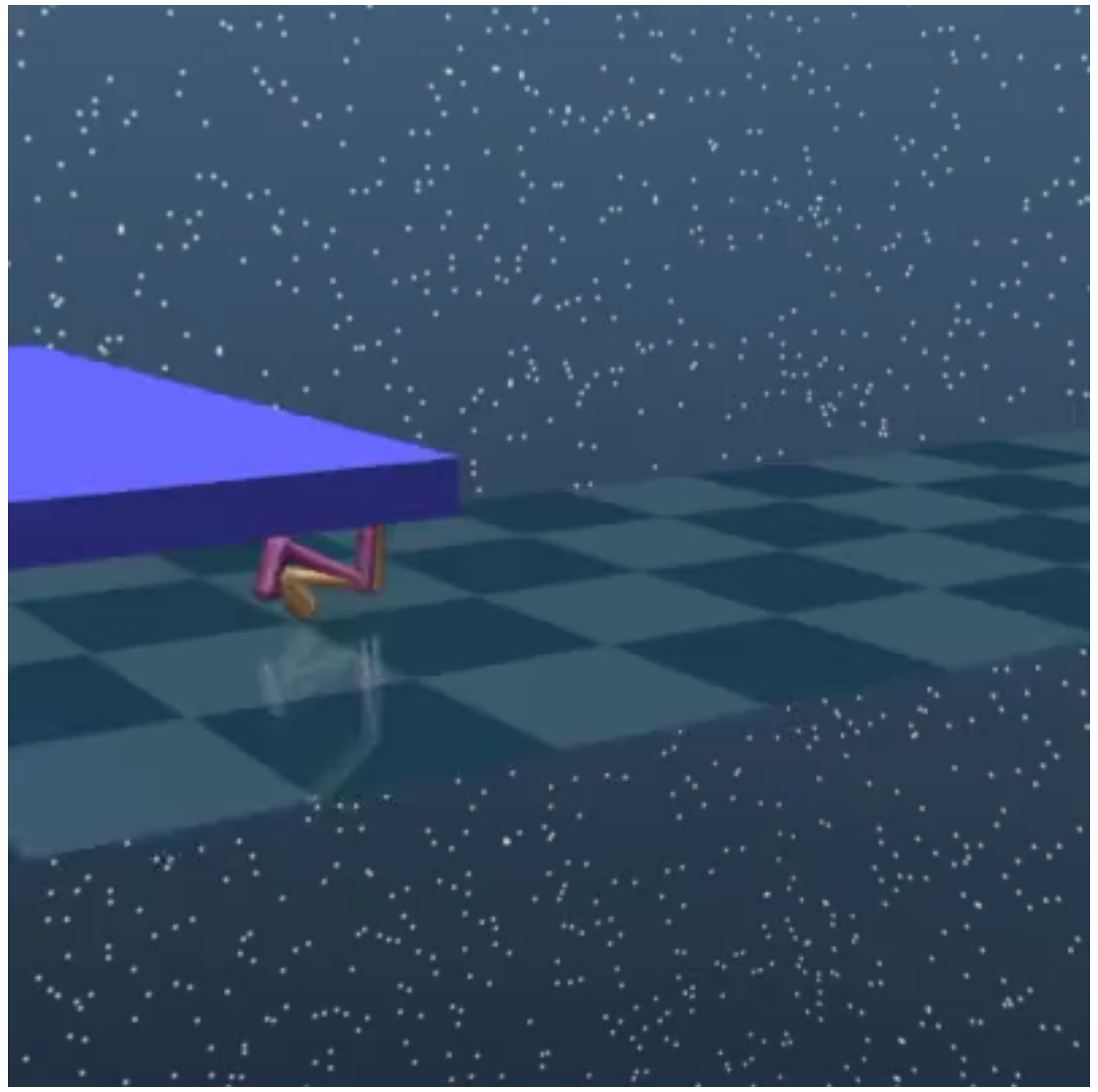


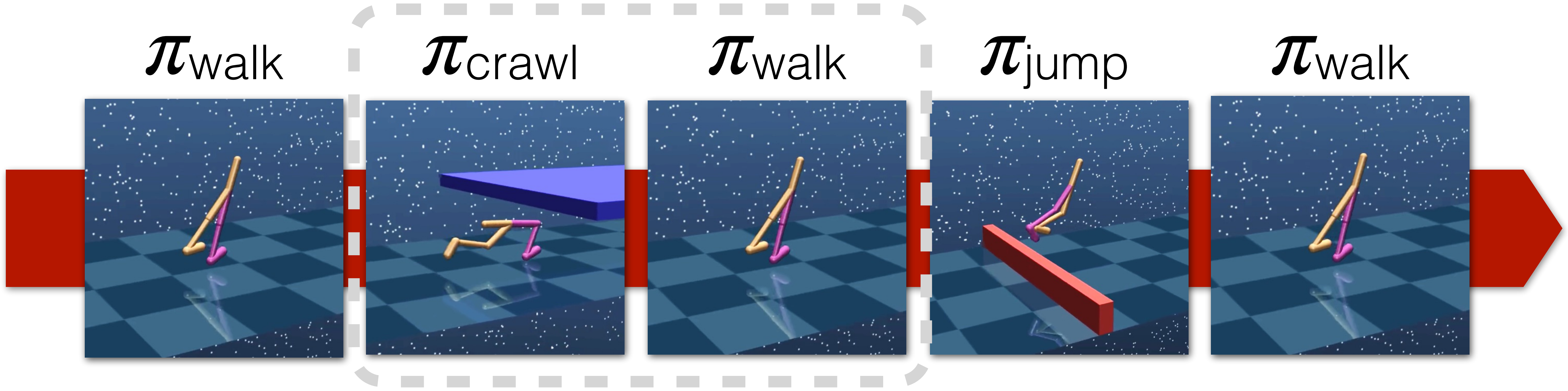

π_{walk}
π_{crawl}
π_{walk}
π_{jump}
π_{walk}



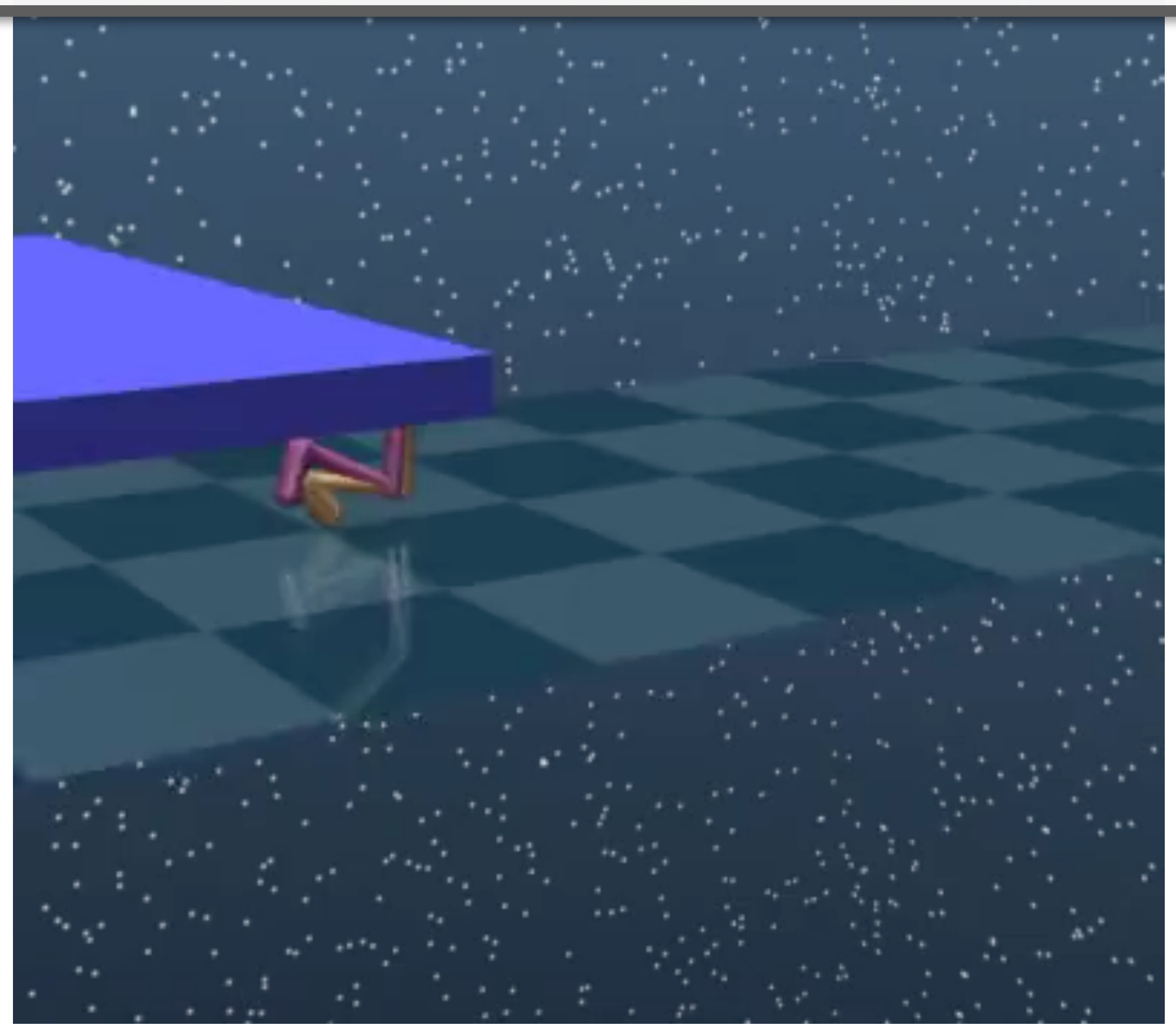


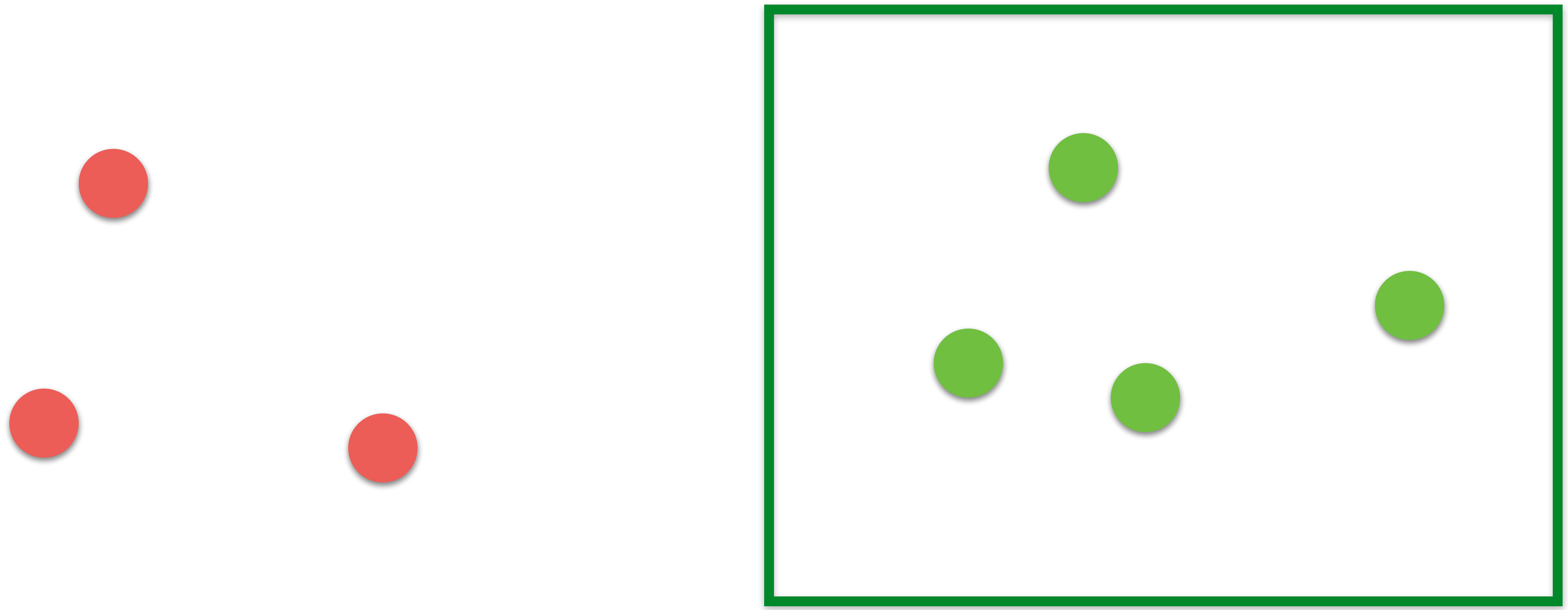
π_{walk}
π_{crawl}
π_{walk}
π_{jump}
π_{walk}





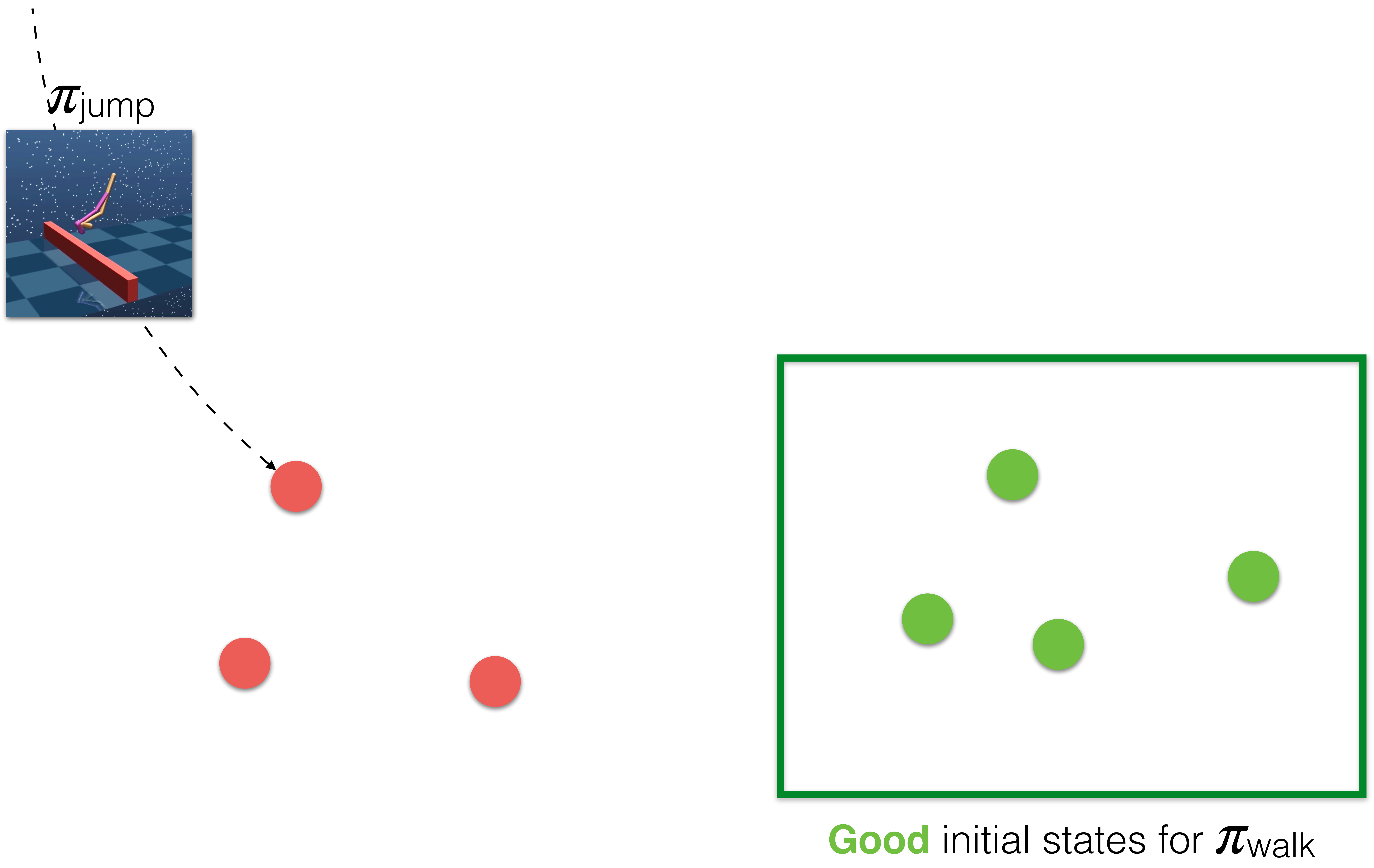
Fail since these skills never learned to connect

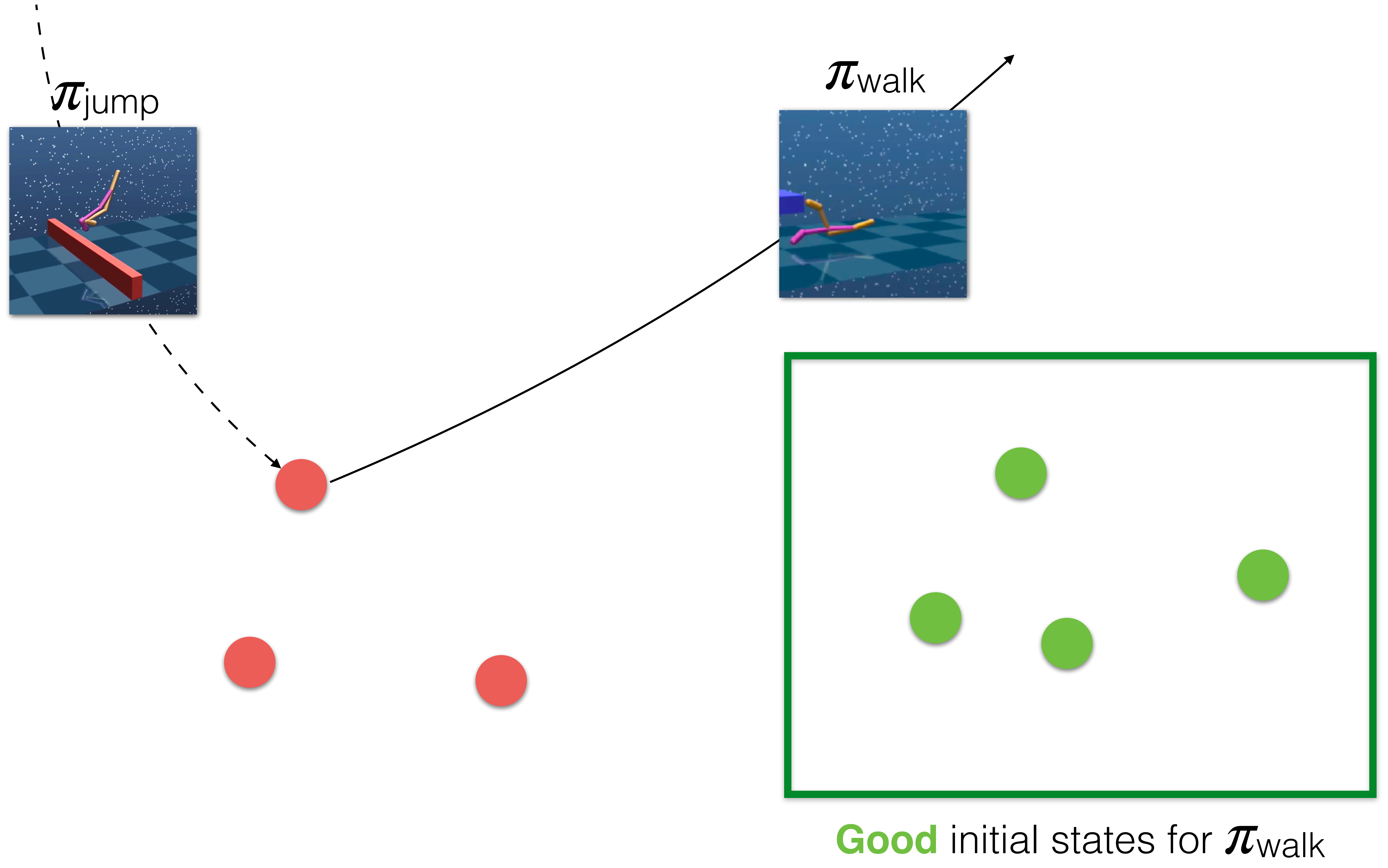




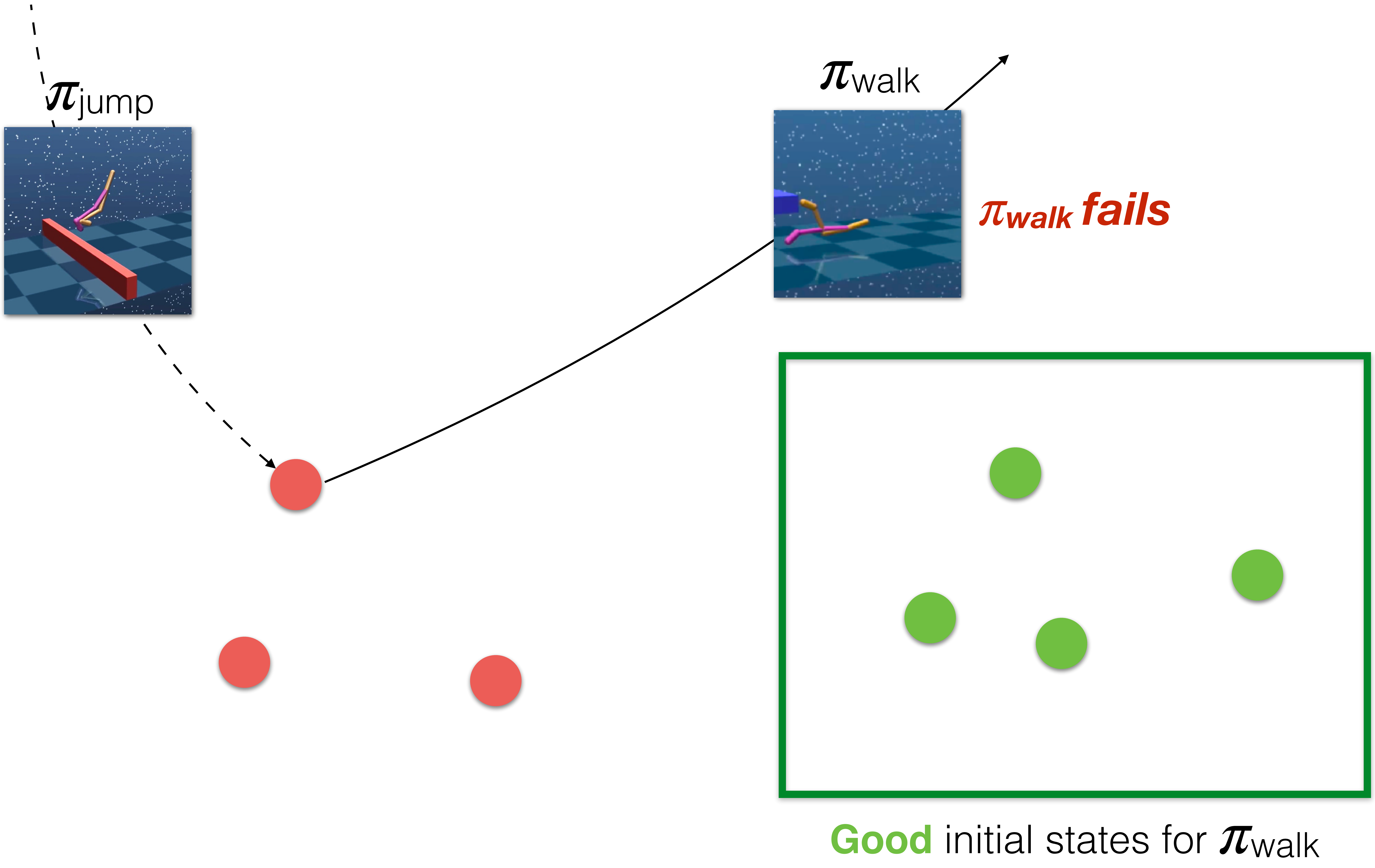
Good initial states for π_{walk}


π_{jump}
Good initial states for π_{walk}

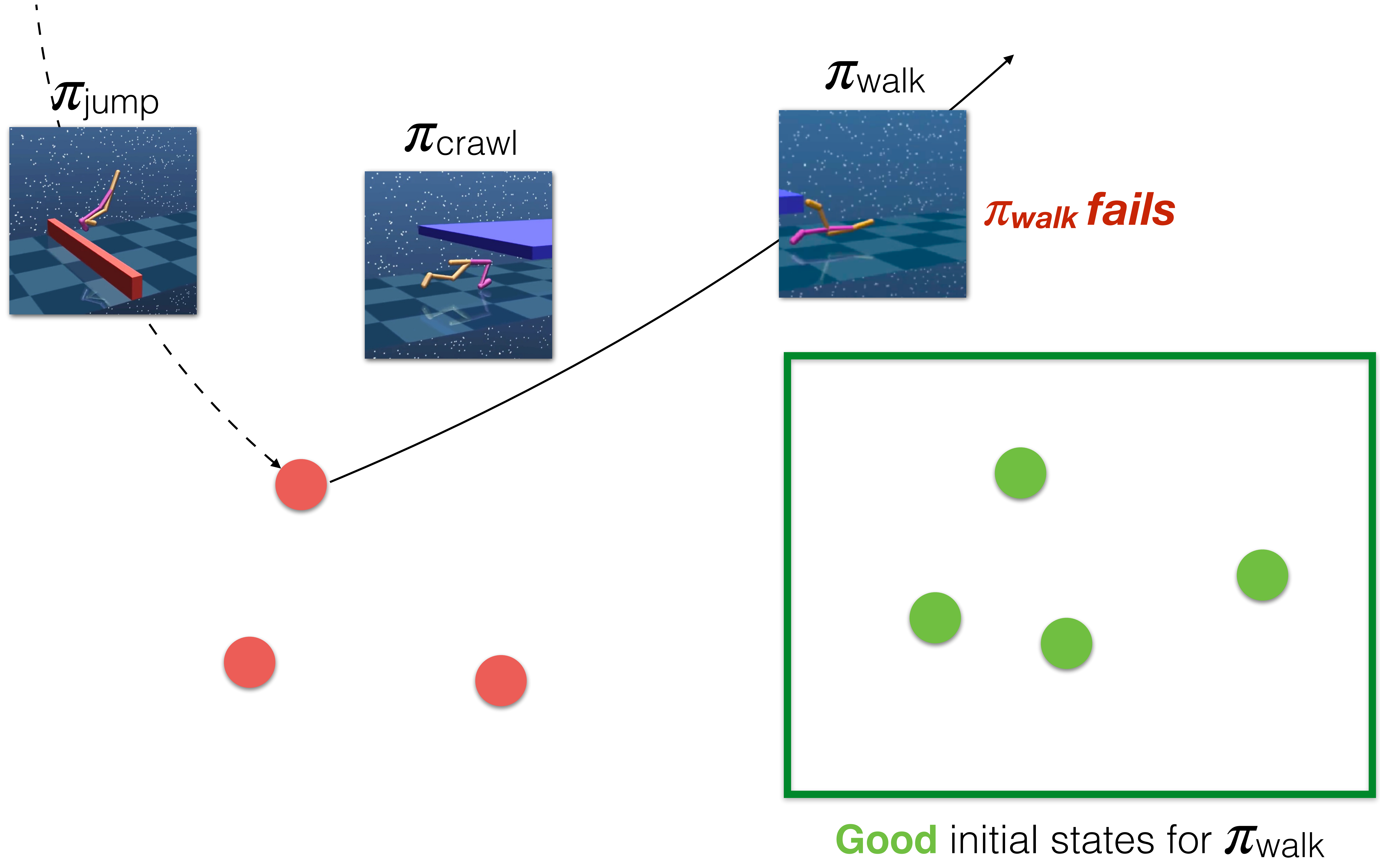

π_{jump}
Good initial states for π_{walk}



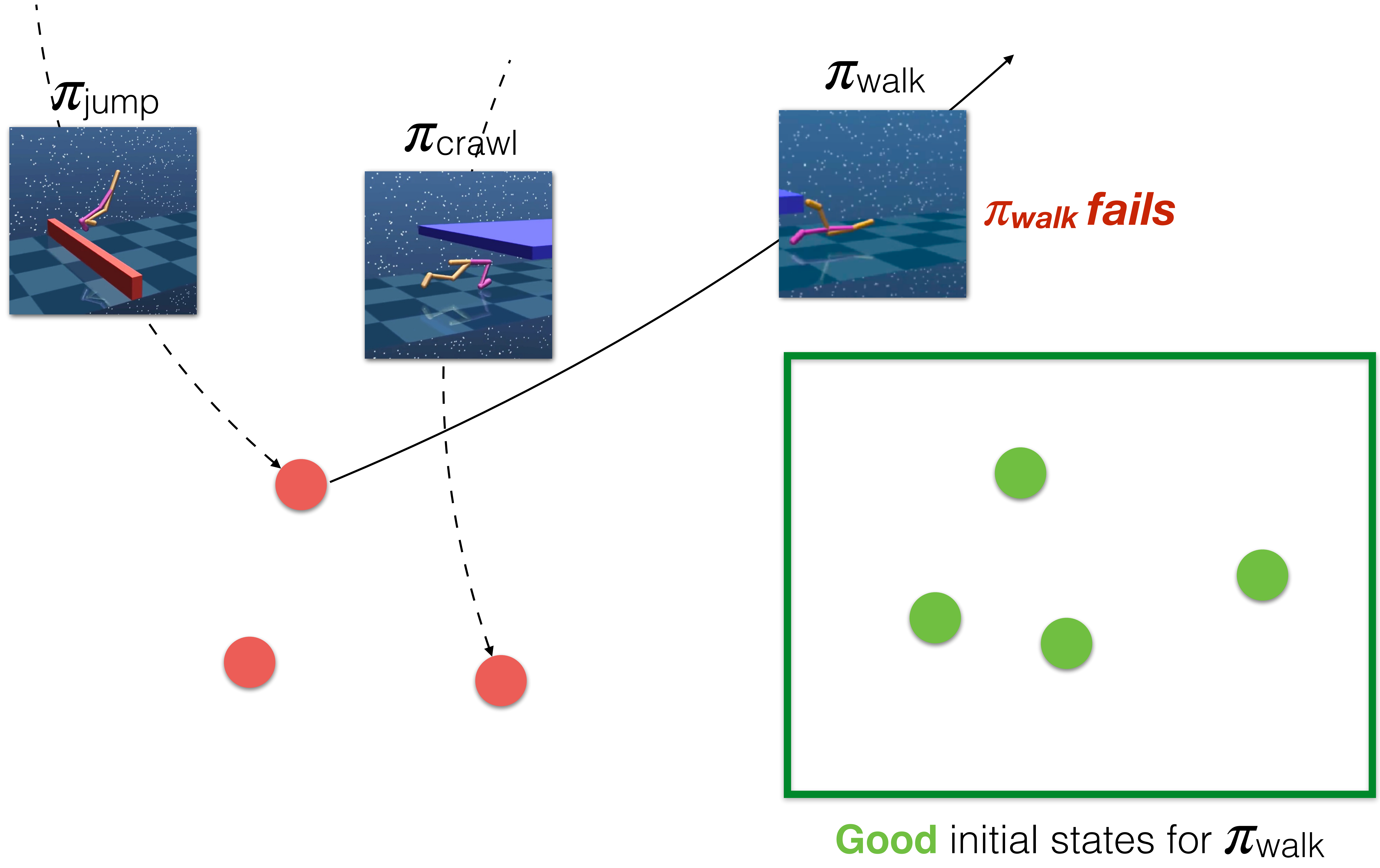
π_{jump}
π_{walk}
Good initial states for π_{walk}



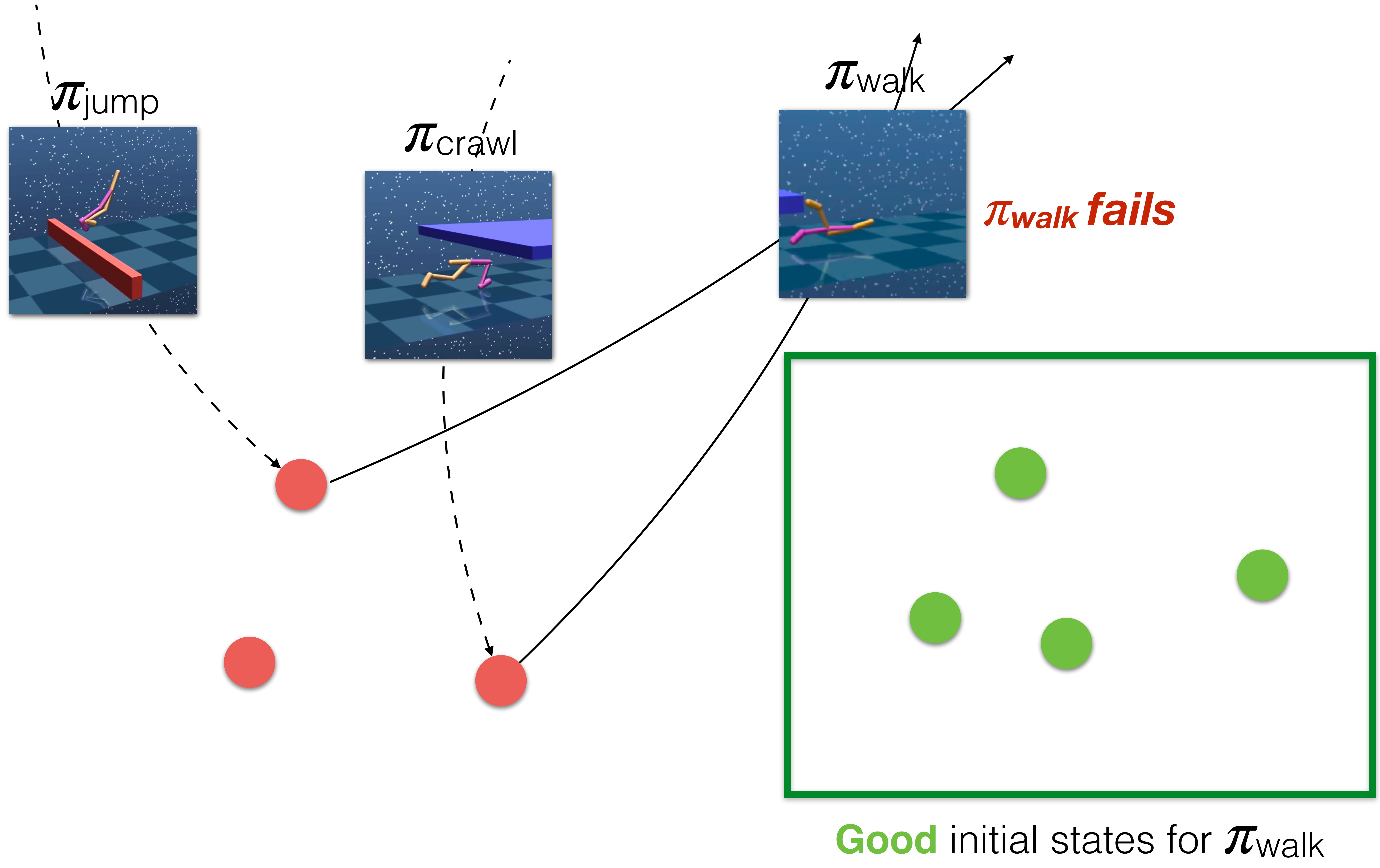
π_{jump}
π_{walk}
π_{walk} fails
Good initial states for π_{walk}



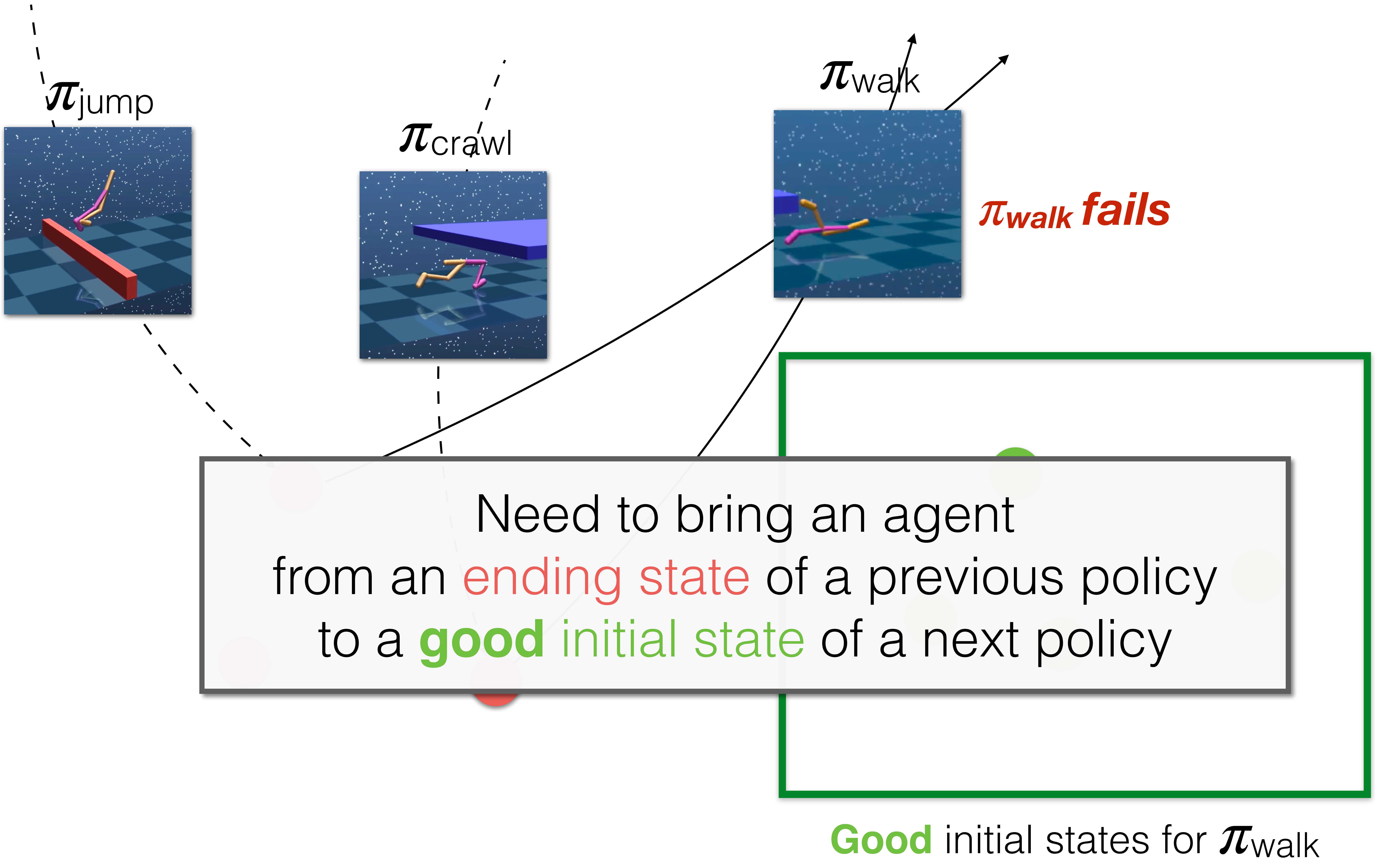
πjump
πcrawl
πwalk
πwalk fails
Good initial states for πwalk



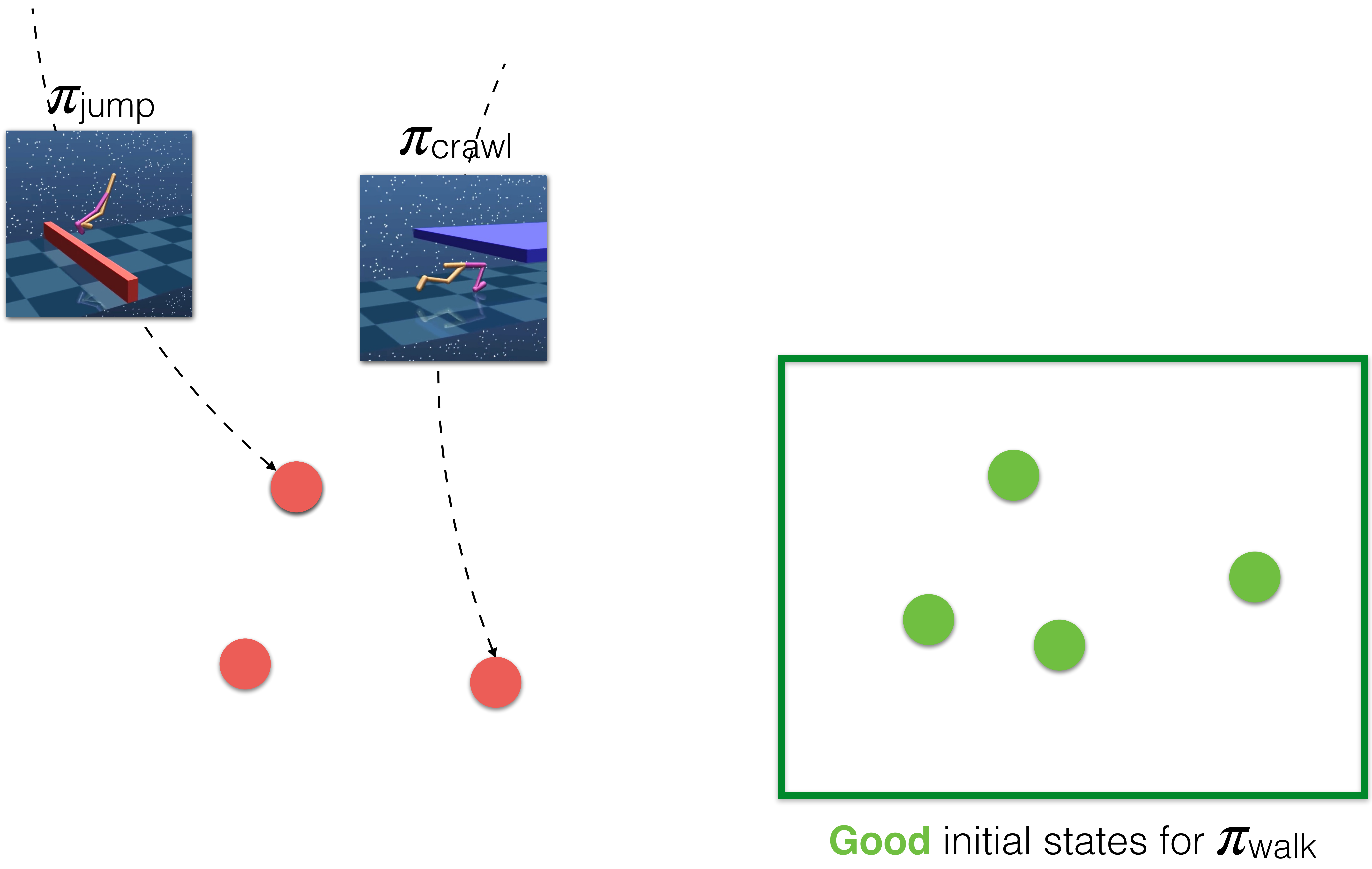
π_{jump}
π_{crawl}
π_{walk}
π_{walk} **fails**
Good initial states for π_{walk}

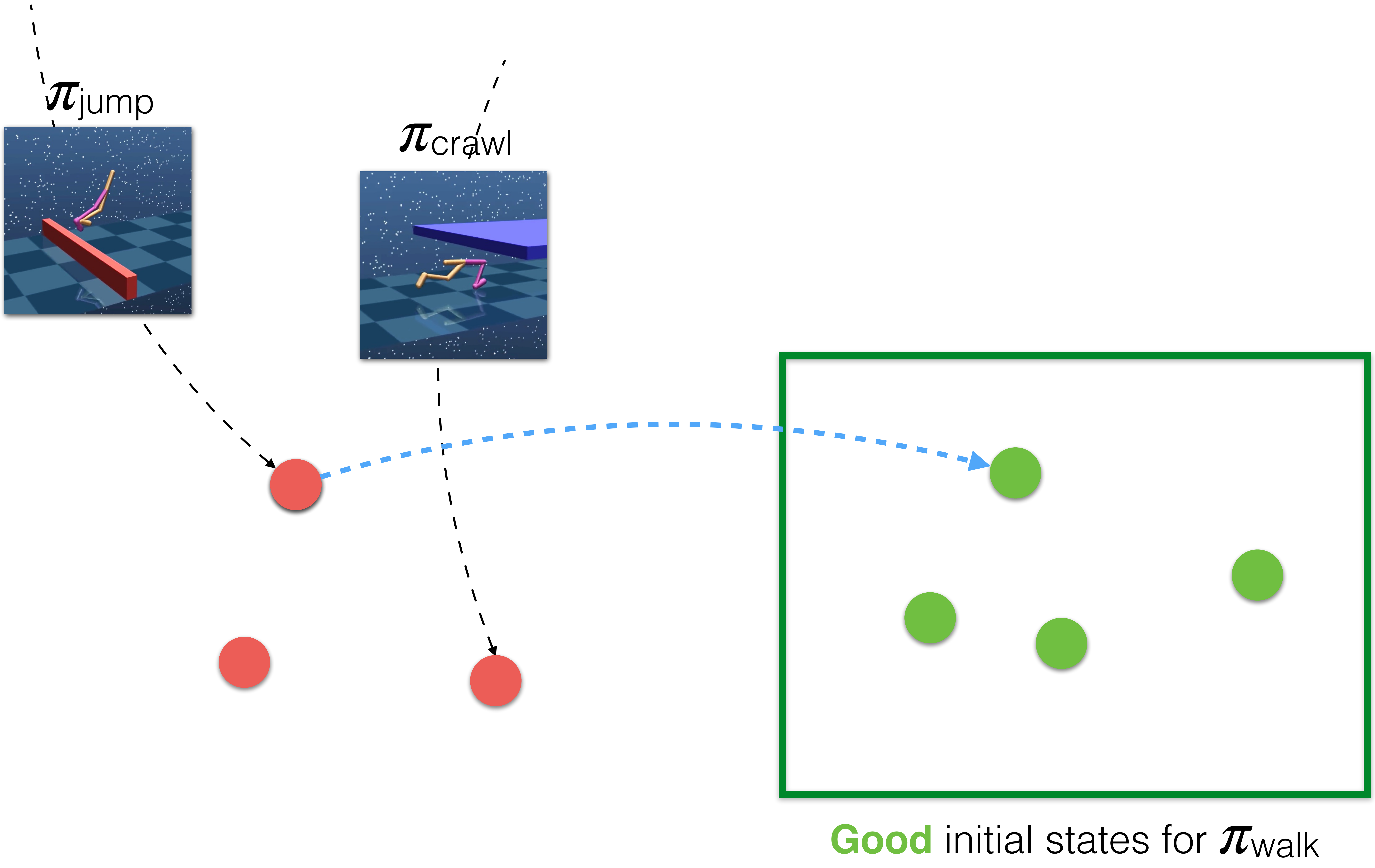


π jump
π crawl
π walk
π walk fails
Good initial states for π walk

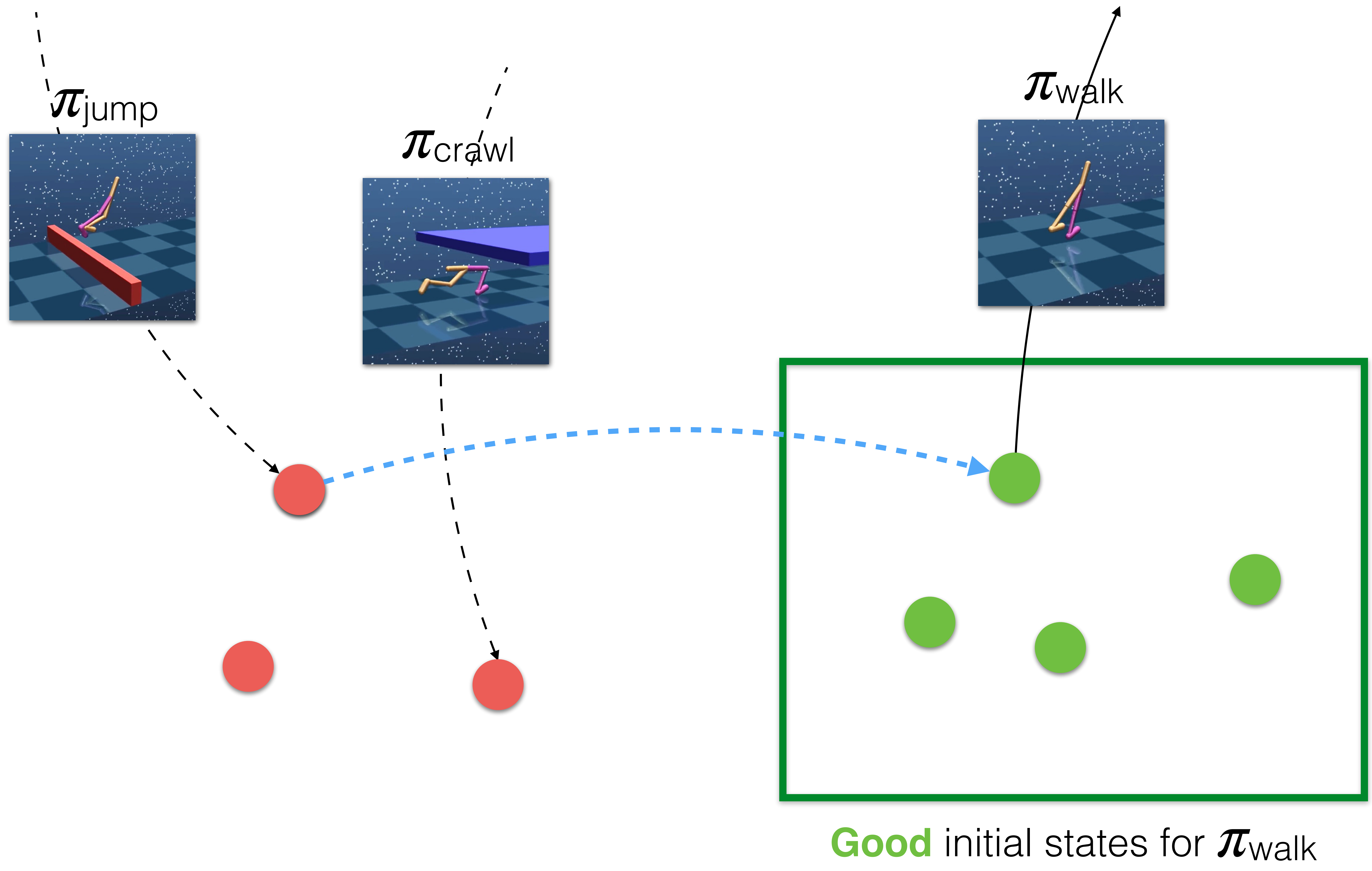


Need to bring an agent
from an ending state of a previous policy
to a **good** initial state of a next policy

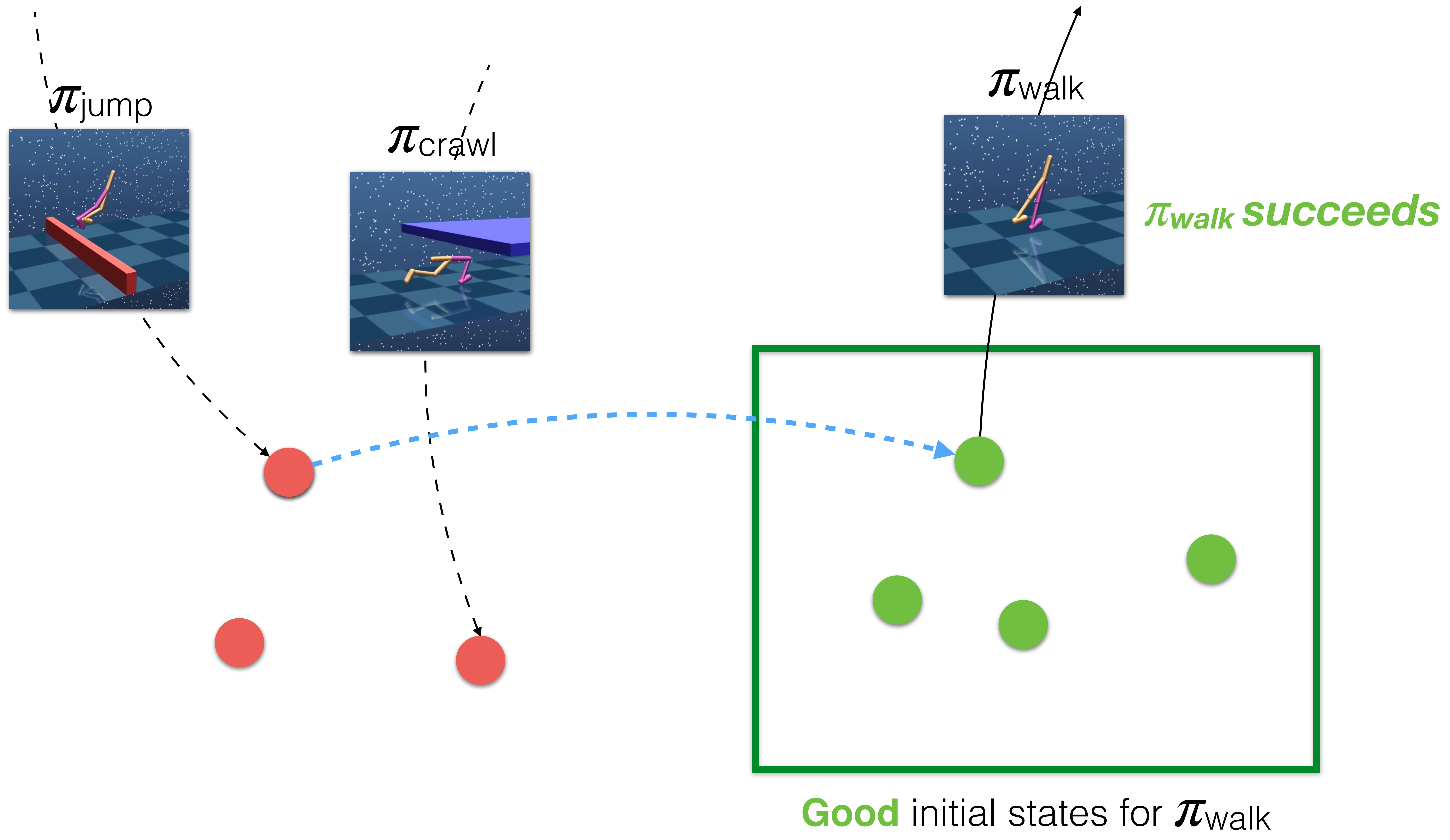

π_{jump}
π_{crawl}
Good initial states for π_{walk}



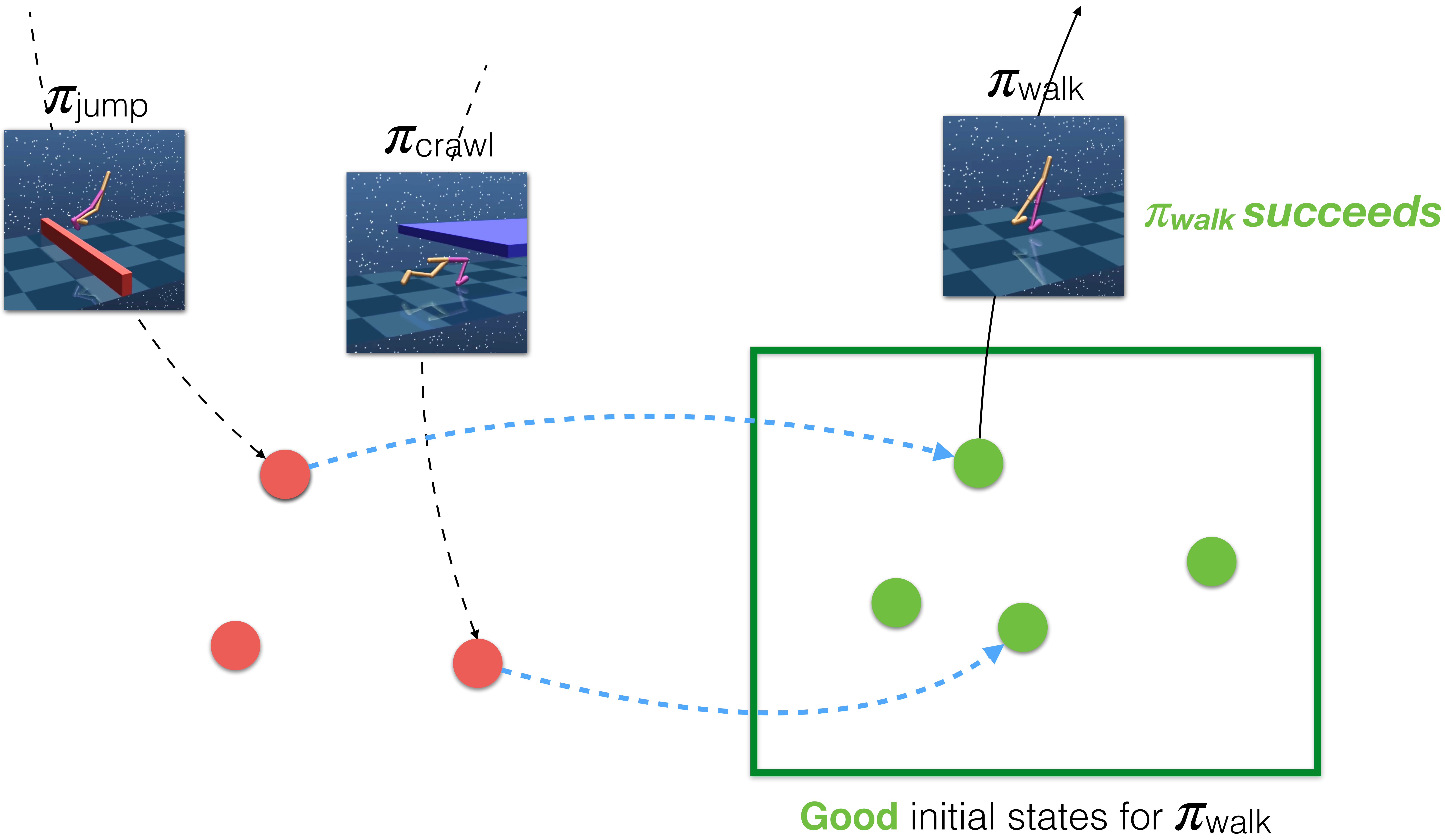
π_{jump}
π_{crawl}
Good initial states for π_{walk}



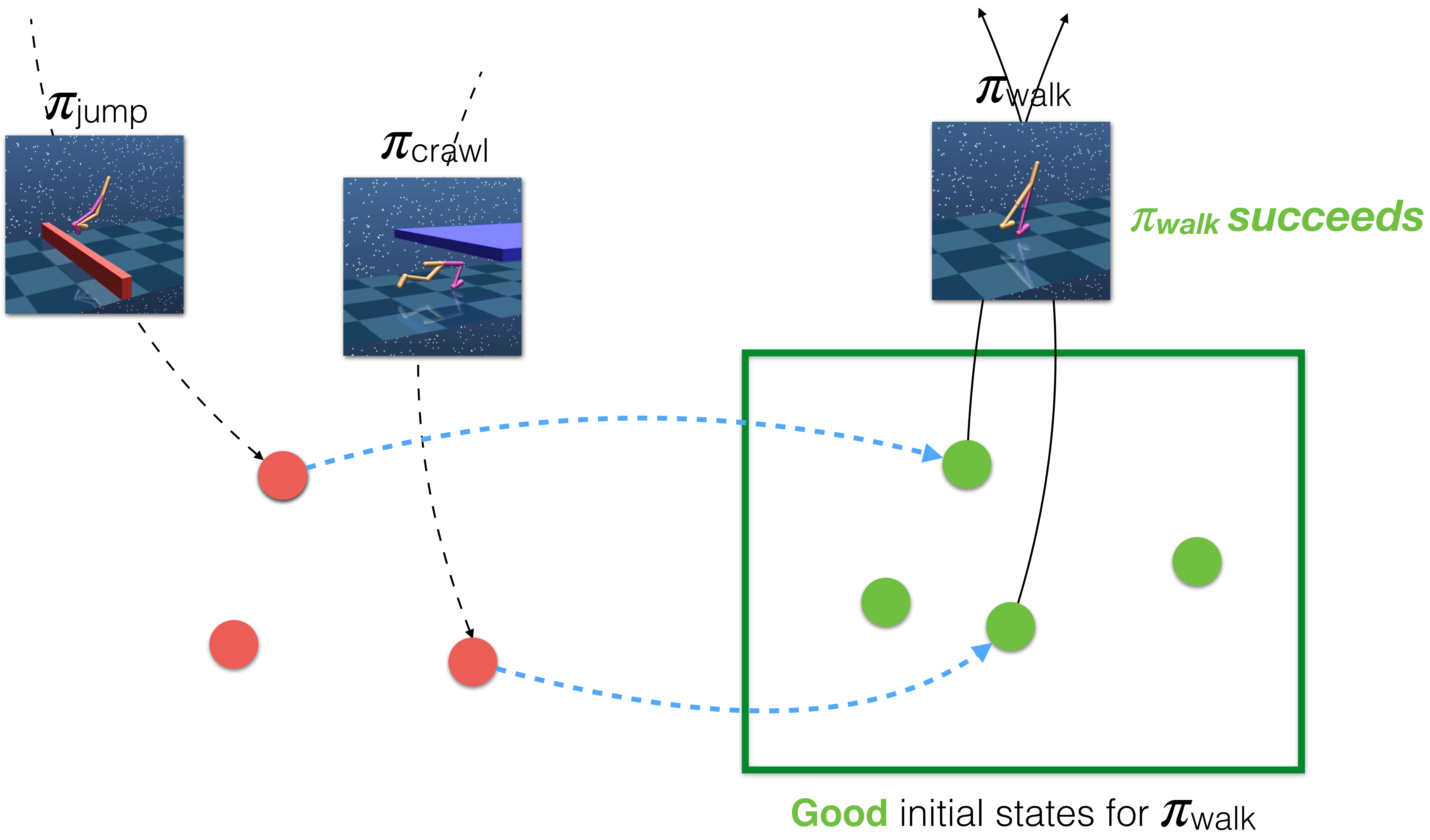
π_{jump}
π_{crawl}
π_{walk}
Good initial states for π_{walk}



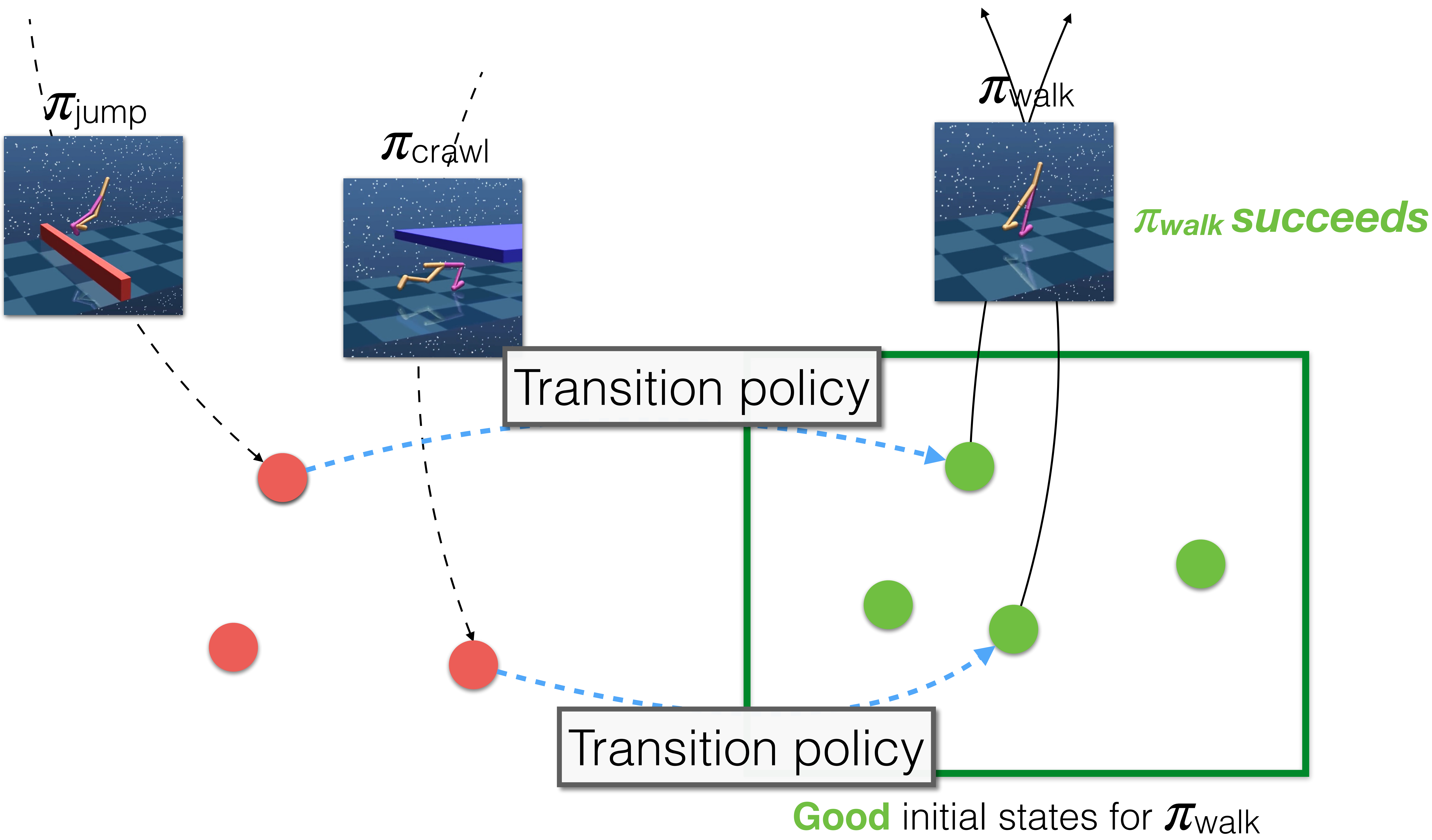
π_{jump}
π_{crawl}
π_{walk}
π_{walk} succeeds
Good initial states for π_{walk}



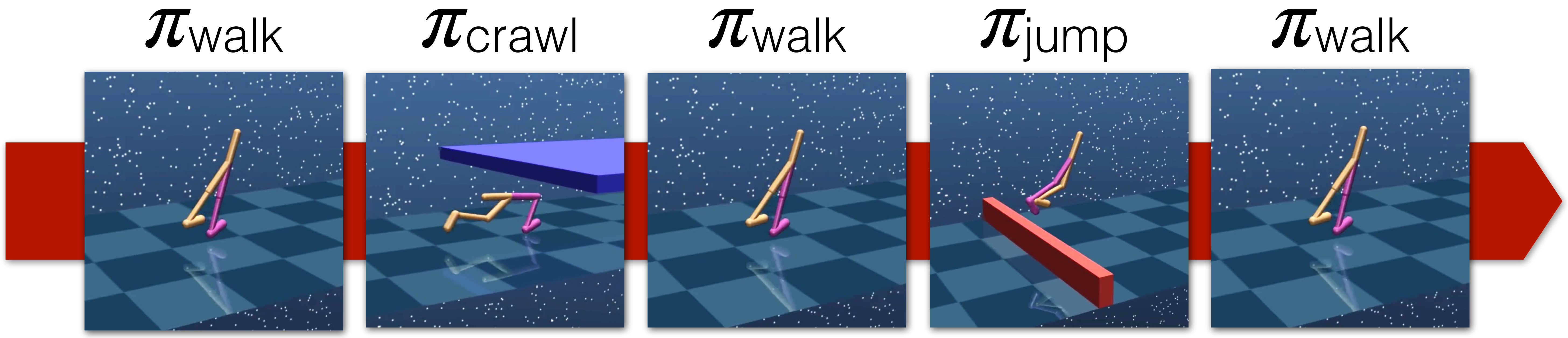
π_{jump}
π_{crawl}
π_{walk}
π_{walk} succeeds
Good initial states for π_{walk}



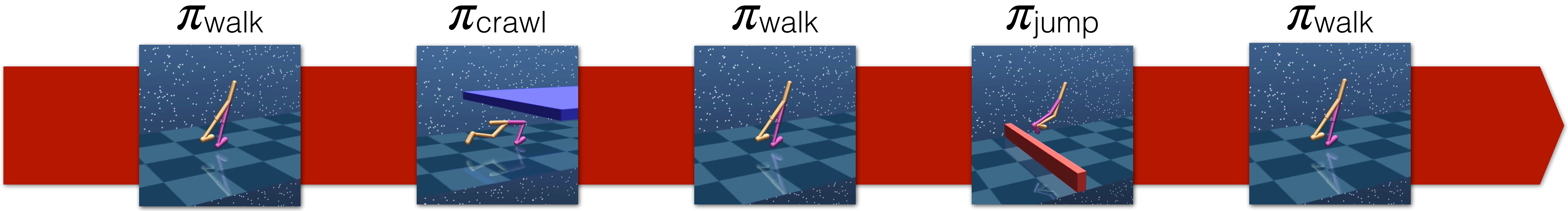
π_{jump}
π_{crawl}
π_{walk}
π_{walk} succeeds
Good initial states for π_{walk}

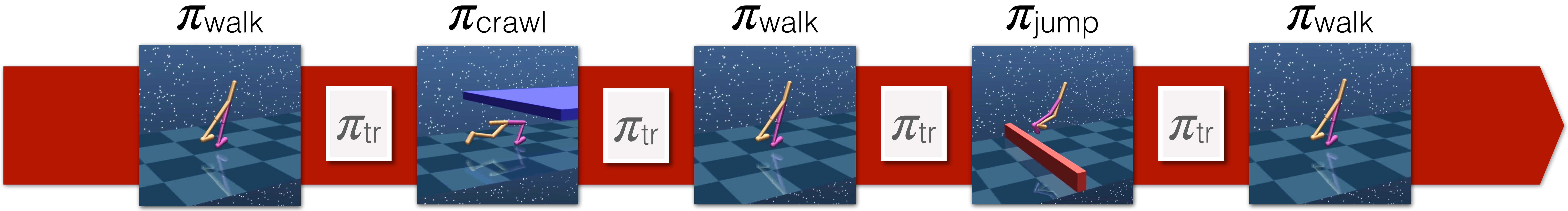


π_{jump}
π_{crawl}
π_{walk}
π_{walk} succeeds
Transition policy
Transition policy
Good initial states for π_{walk}



π_{walk}
π_{crawl}
π_{walk}
π_{jump}
π_{walk}

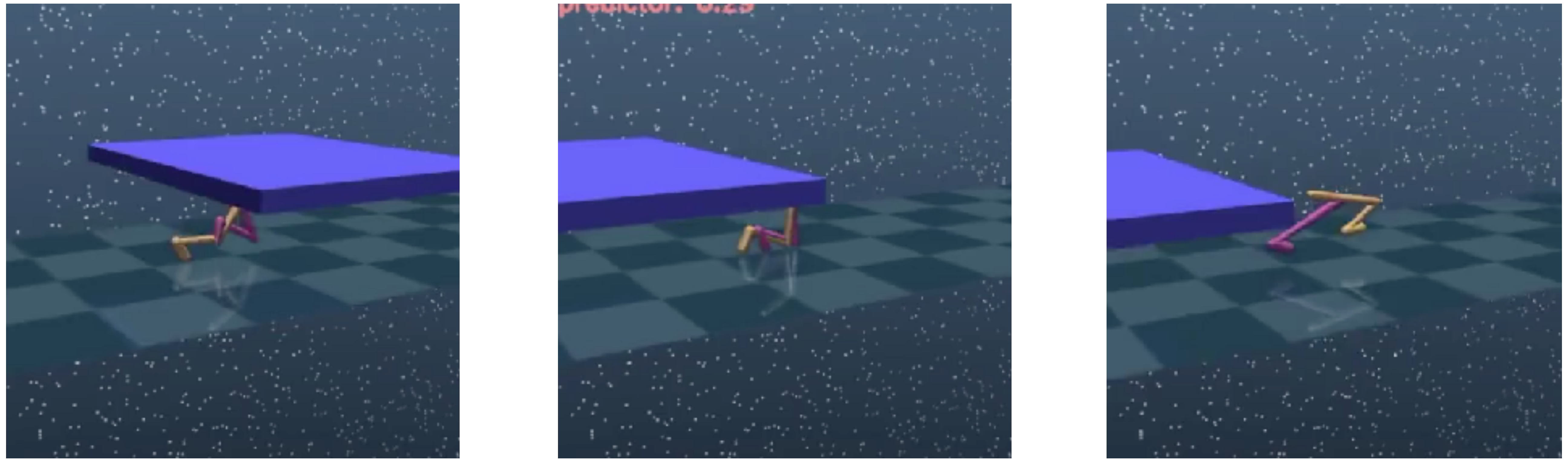

π_{walk}
π_{crawl}
π_{walk}
π_{jump}
π_{walk}



π_{walk}
π_{tr}
π_{crawl}
π_{tr}
π_{walk}
π_{tr}
π_{jump}
π_{tr}
π_{walk}

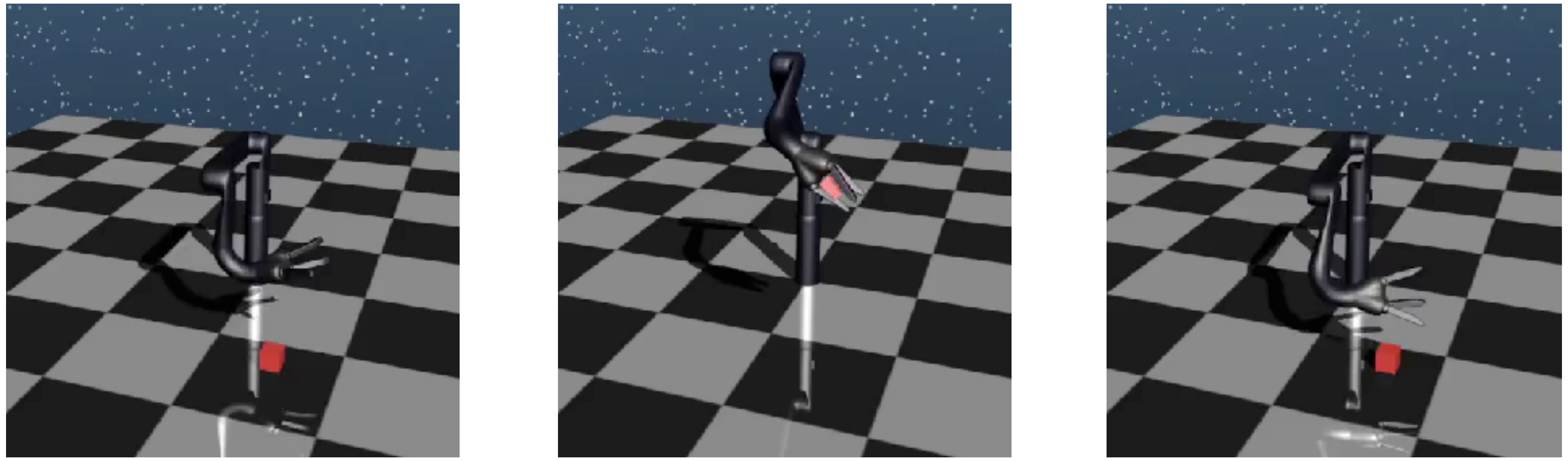
Obstacle course

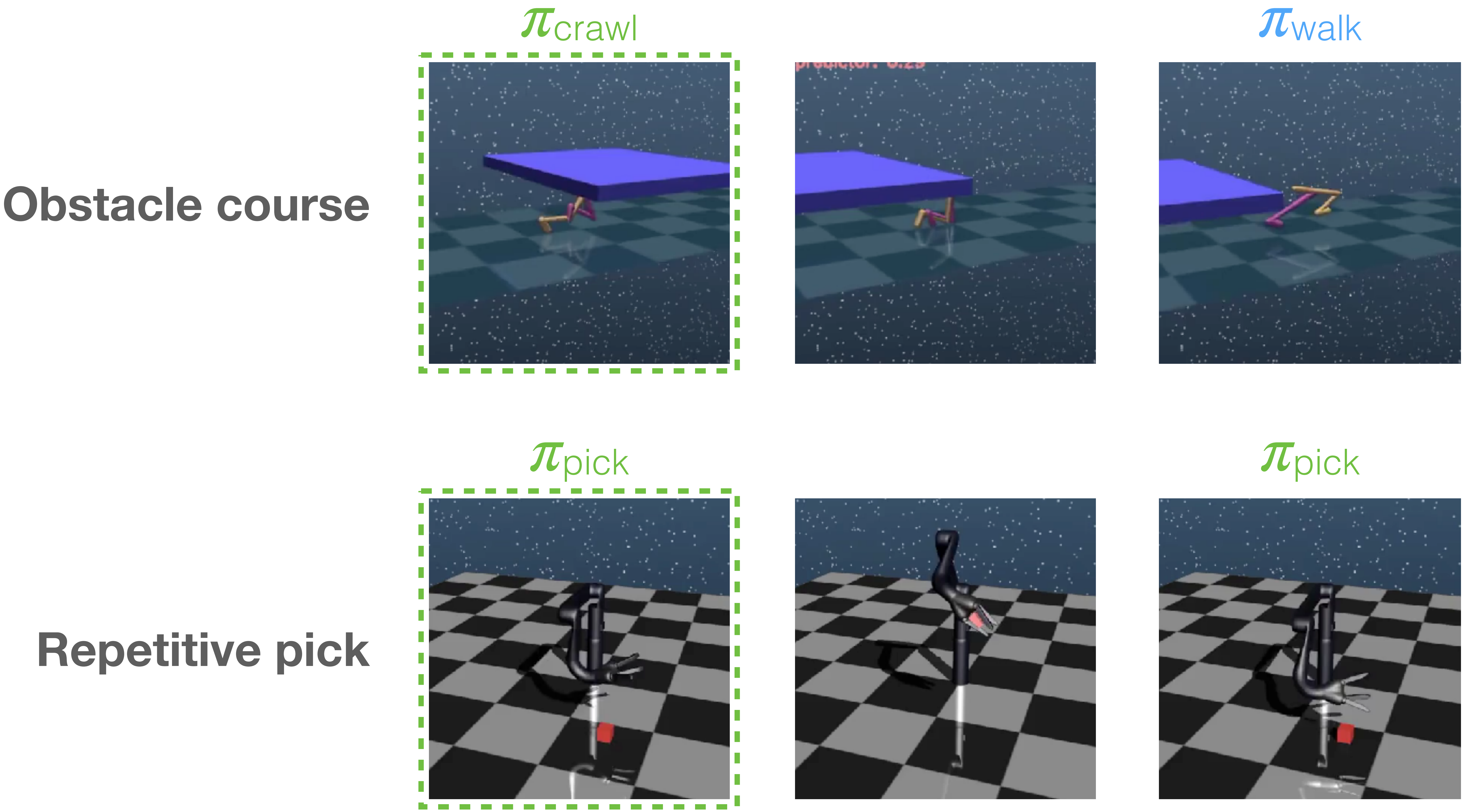
π_{crawl} π_{walk}



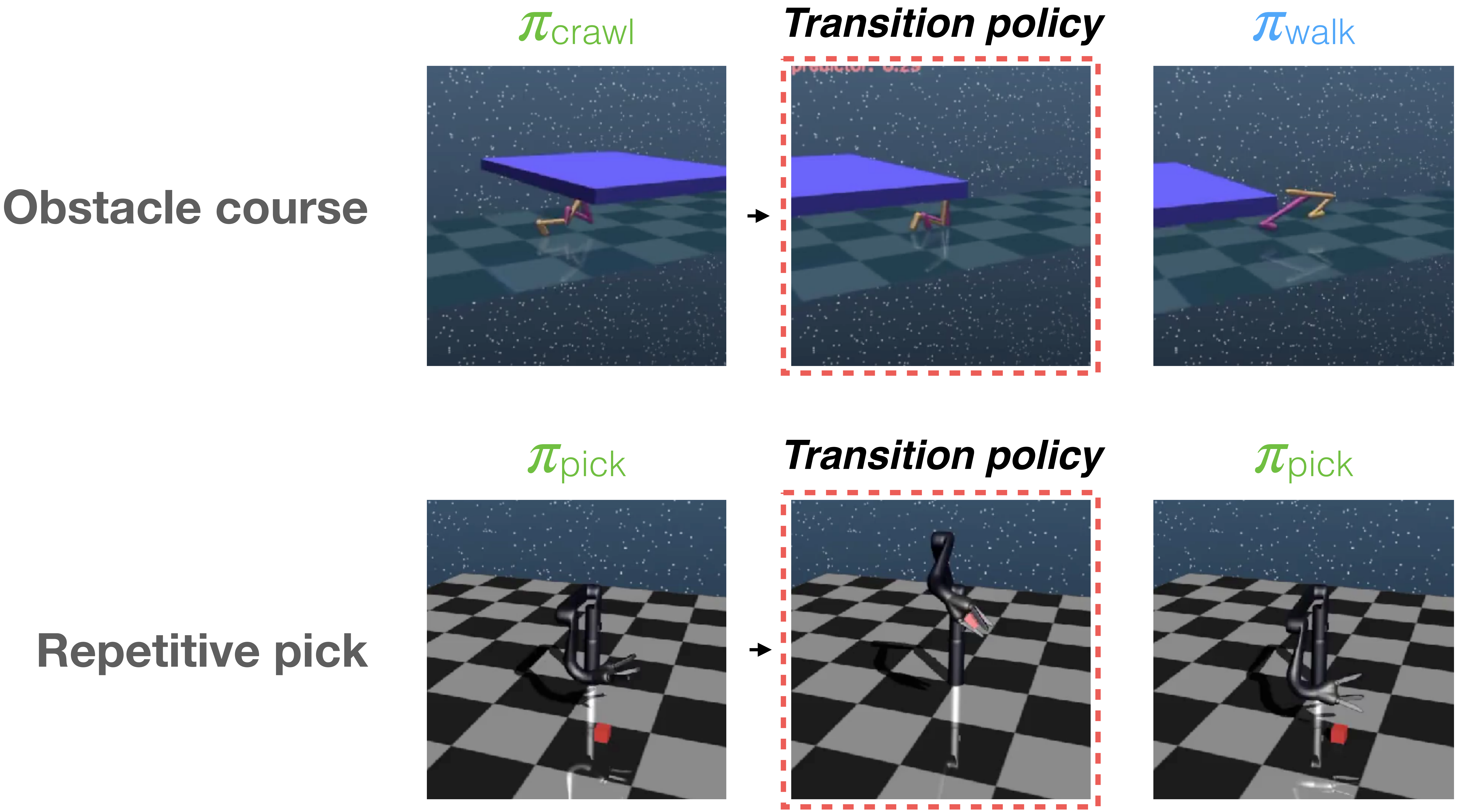
Repetitive pick

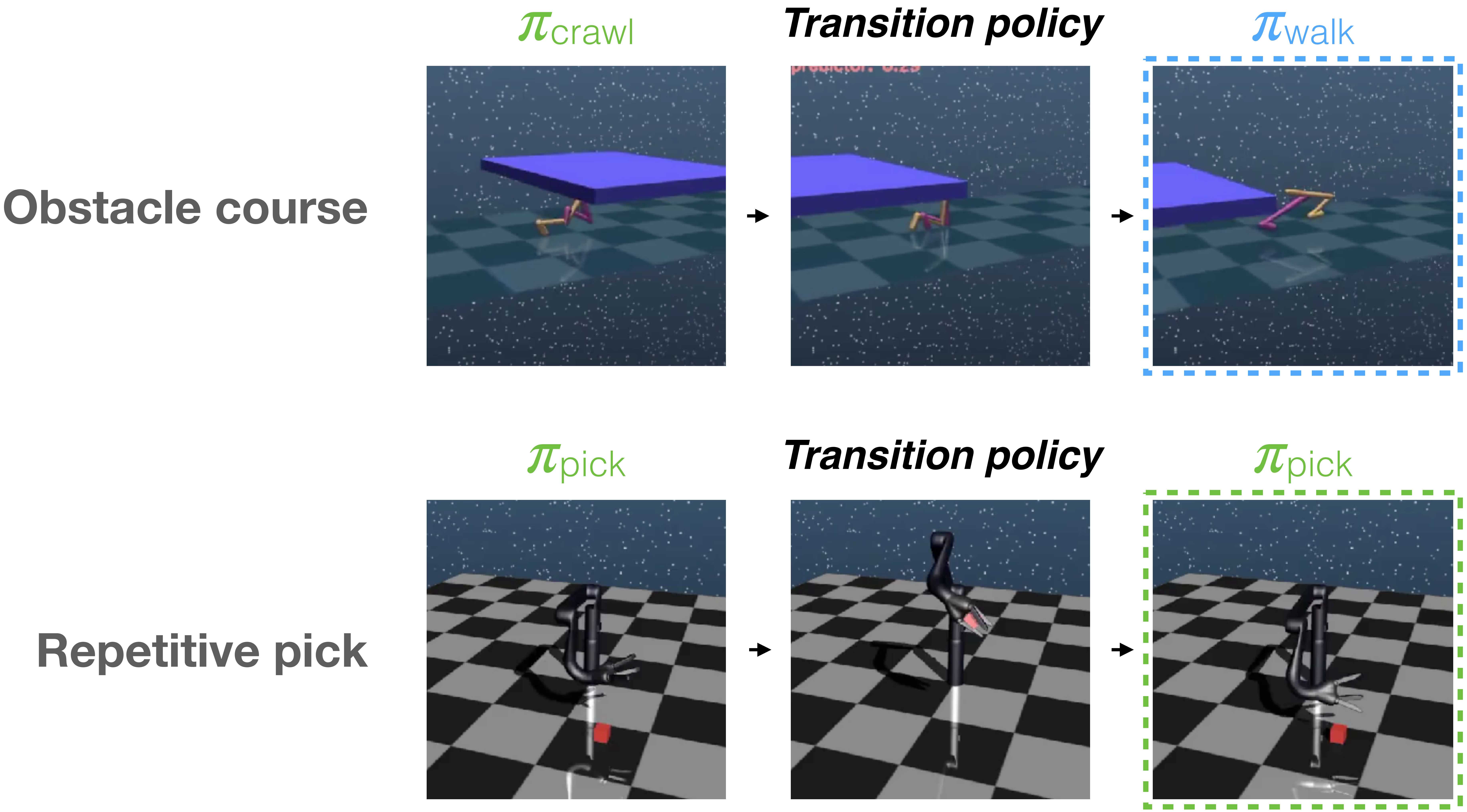
π_{pick} π_{pick}



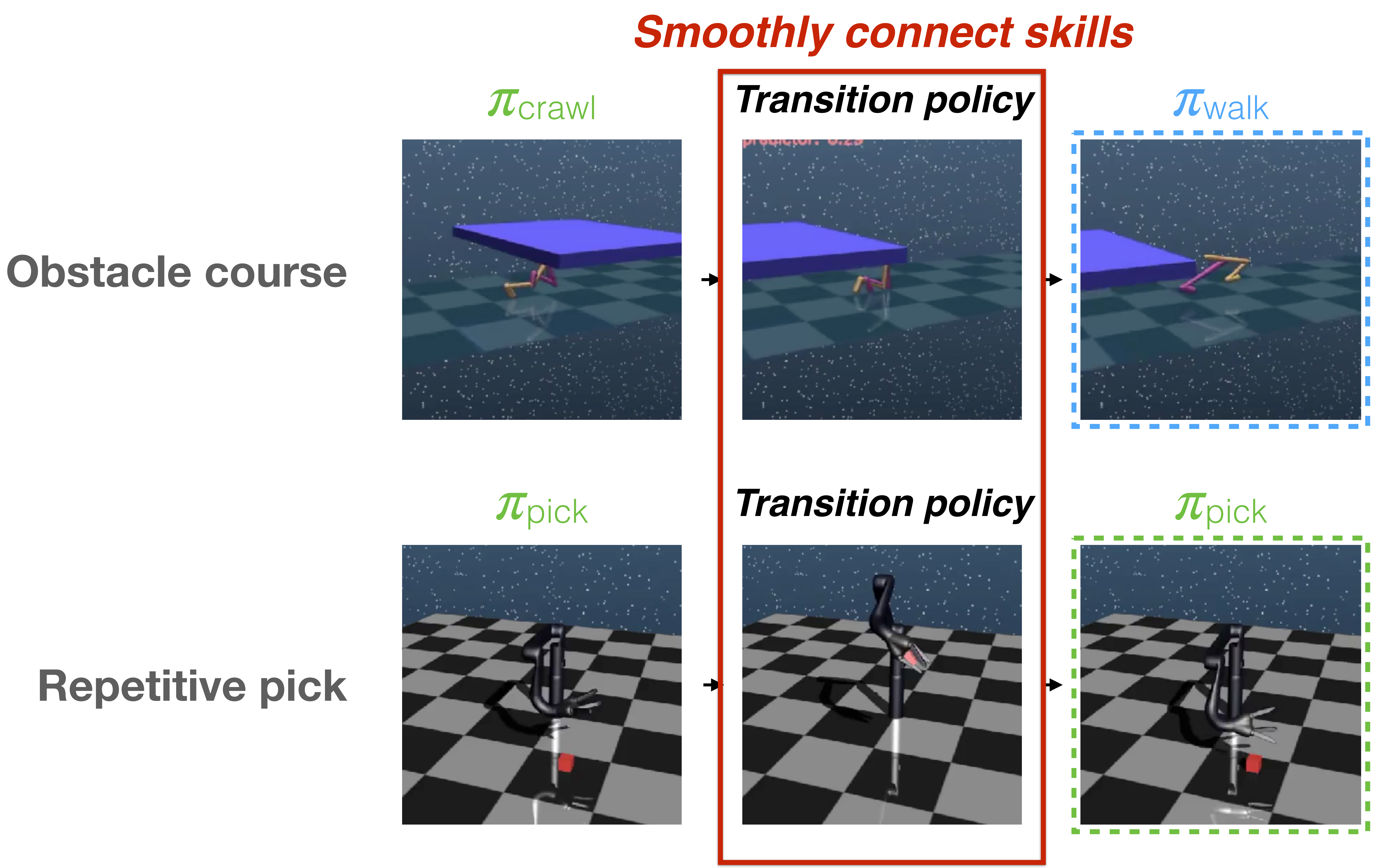


π_{crawl}
π_{walk}
Obstacle course
π_{pick}
π_{pick}
Repetitive pick


Obstacle course
Repetitive pick
πcrawl
Transition policy
πwalk
πpick
Transition policy
πpick

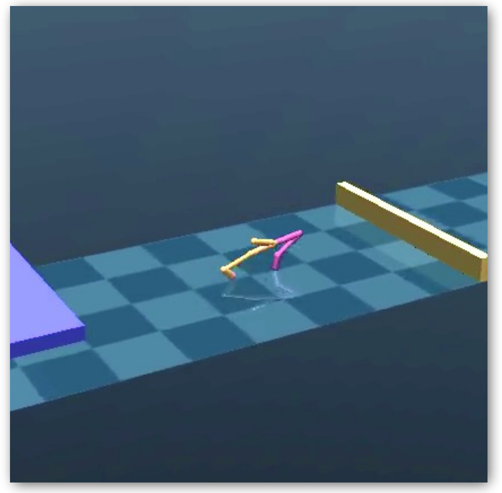


Obstacle course
π_{crawl}
Transition policy
π_{walk}
Repetitive pick
π_{pick}
Transition policy
π_{pick}



Smoothly connect skills
Obstacle course
Repetitive pick
π_{crawl}
Transition policy
π_{walk}
π_{pick}
Transition policy
π_{pick}

Model

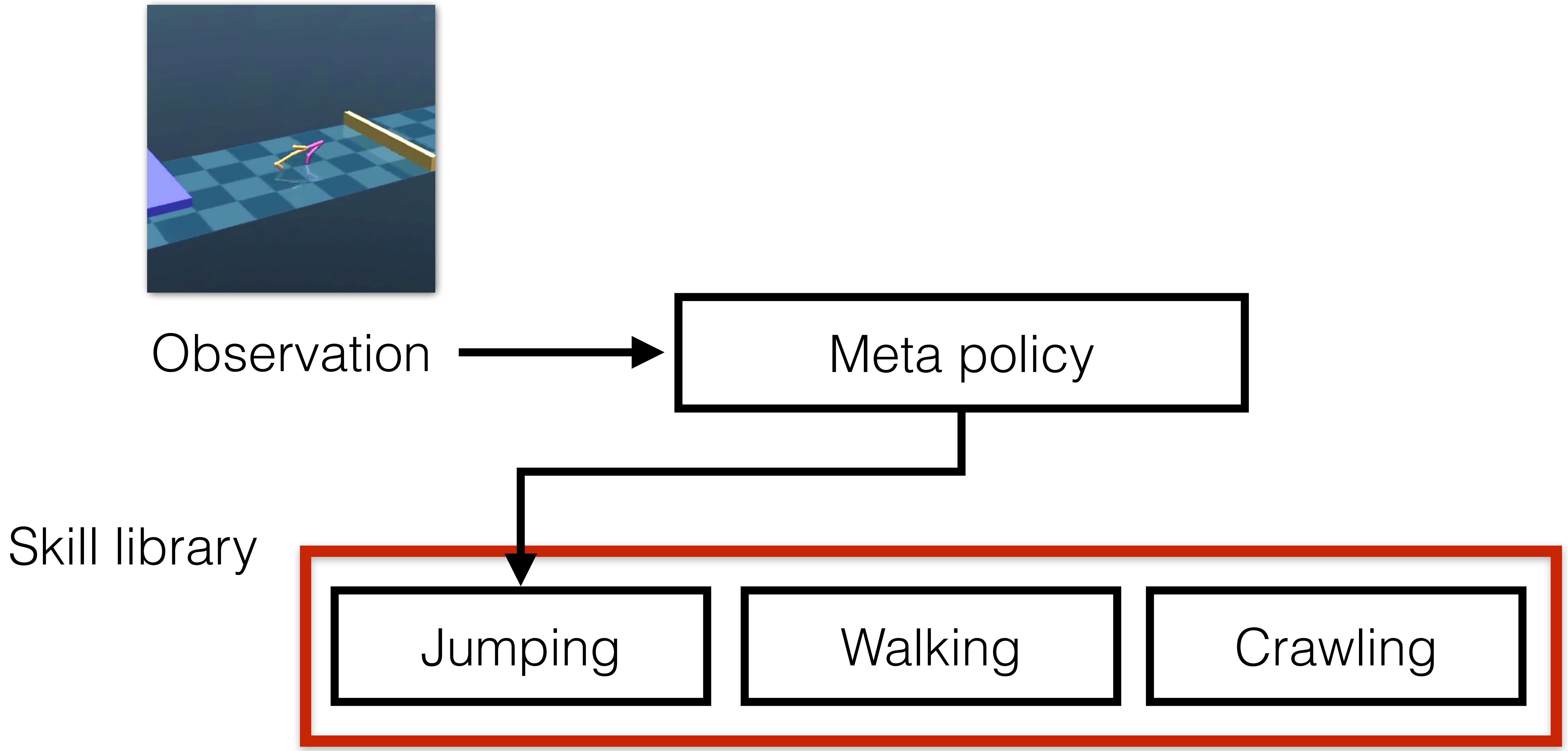


Observation → Meta policy

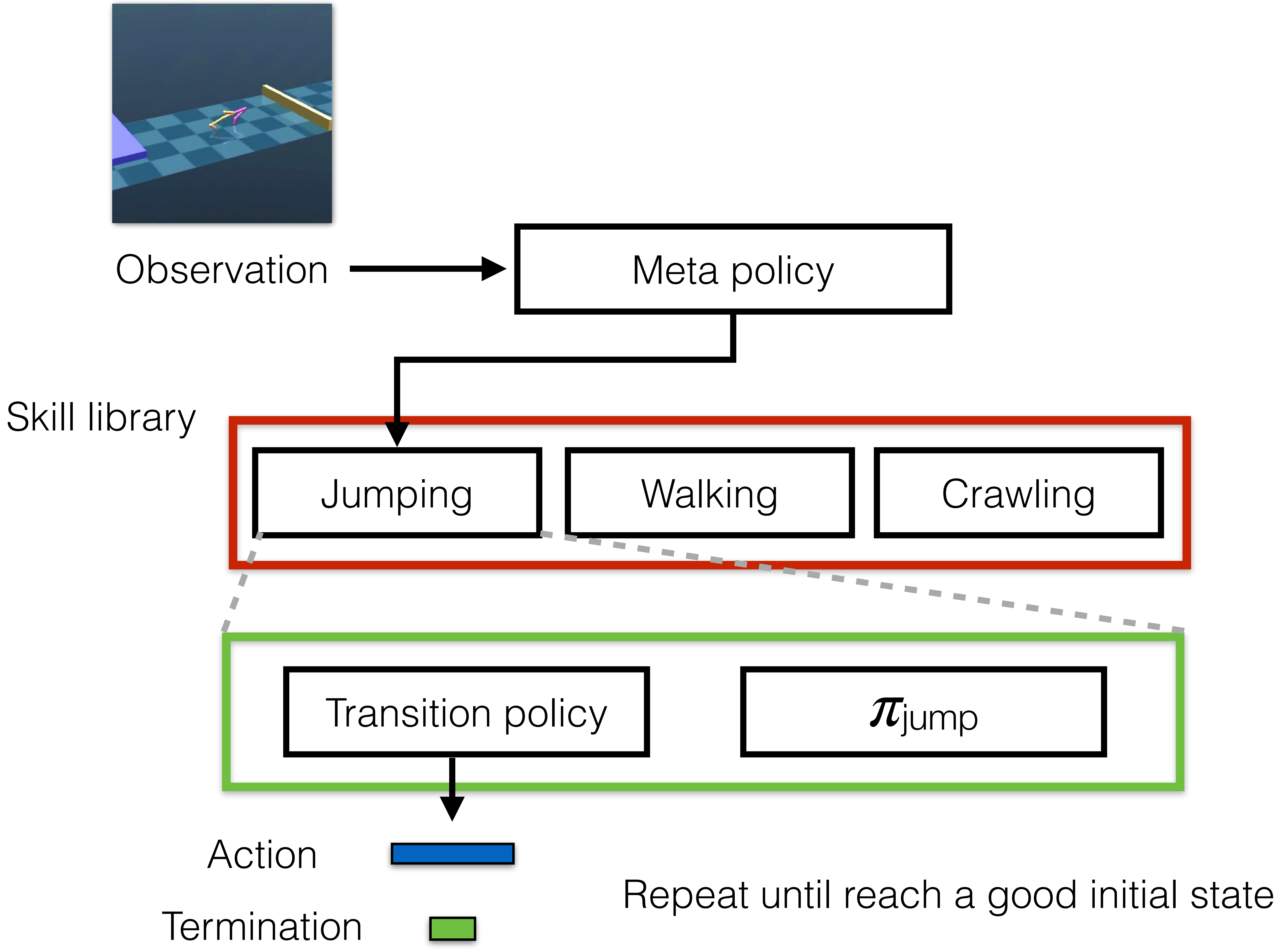
Skill library

Jumping | Walking | Crawling

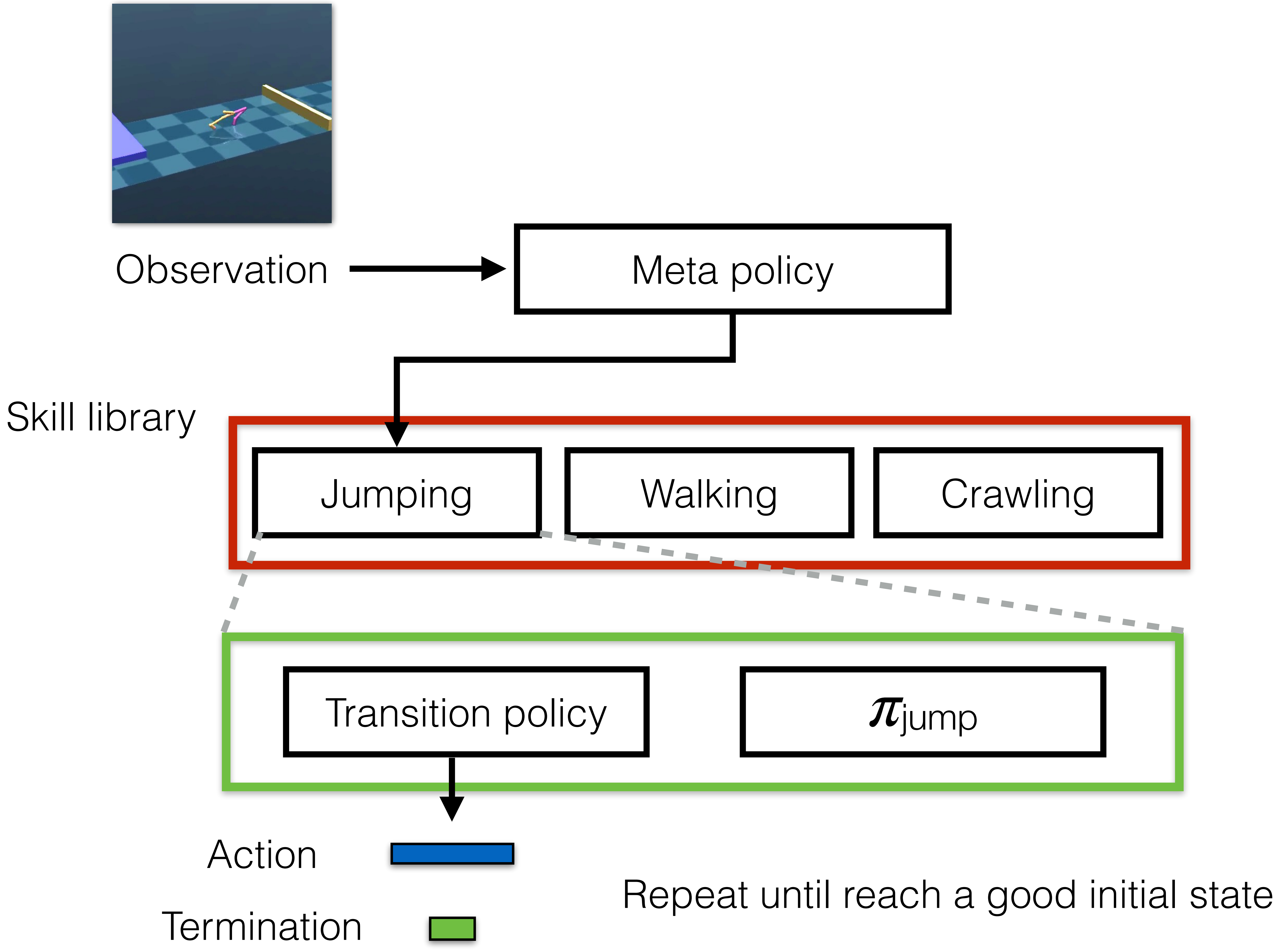
Model


Observation
Meta policy
Skill library
Jumping
Walking
Crawling

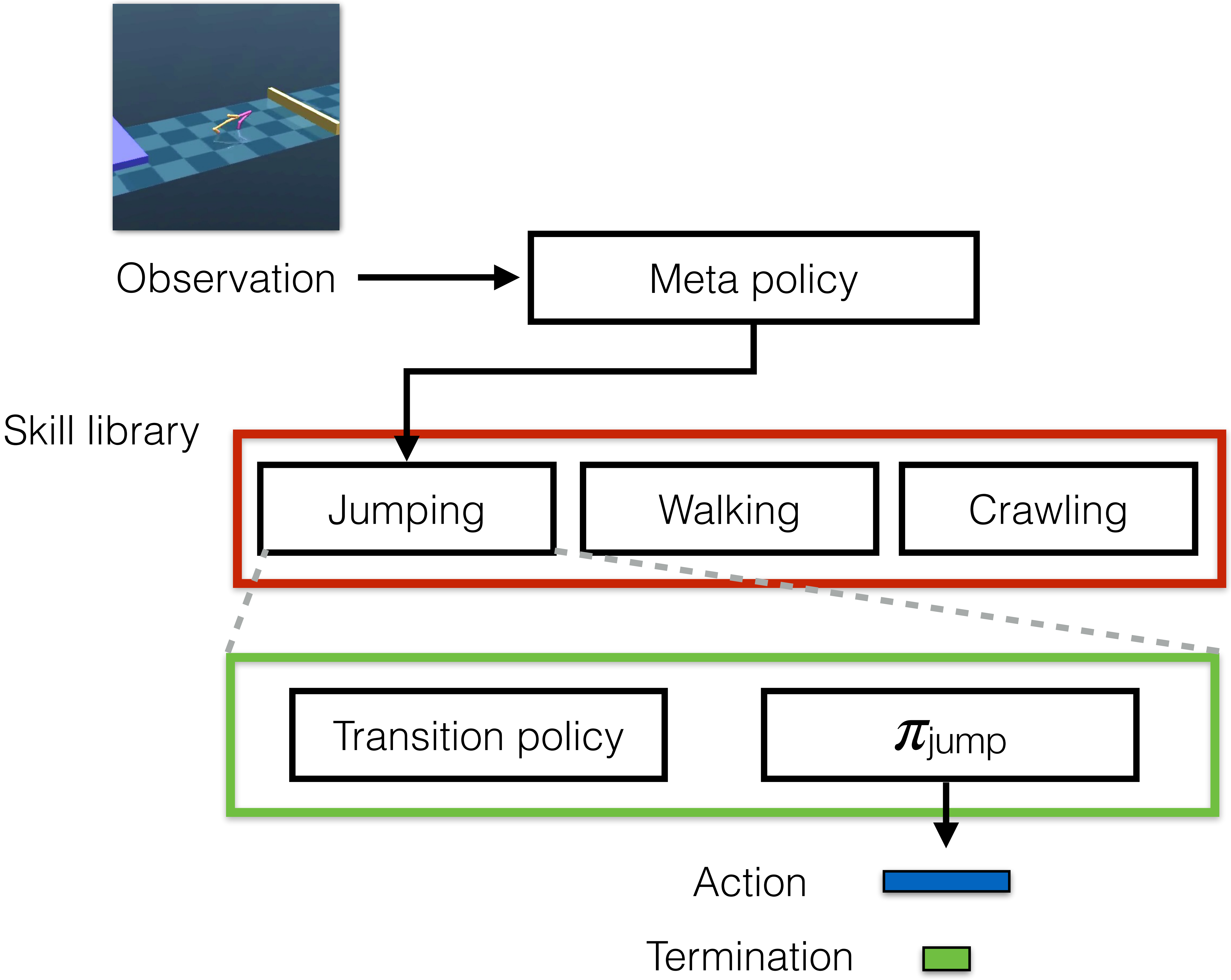
Model


Observation
Meta policy
Skill library
Jumping
Walking
Crawling
Transition policy
π_{jump}
Action
Termination
Repeat until reach a good initial state

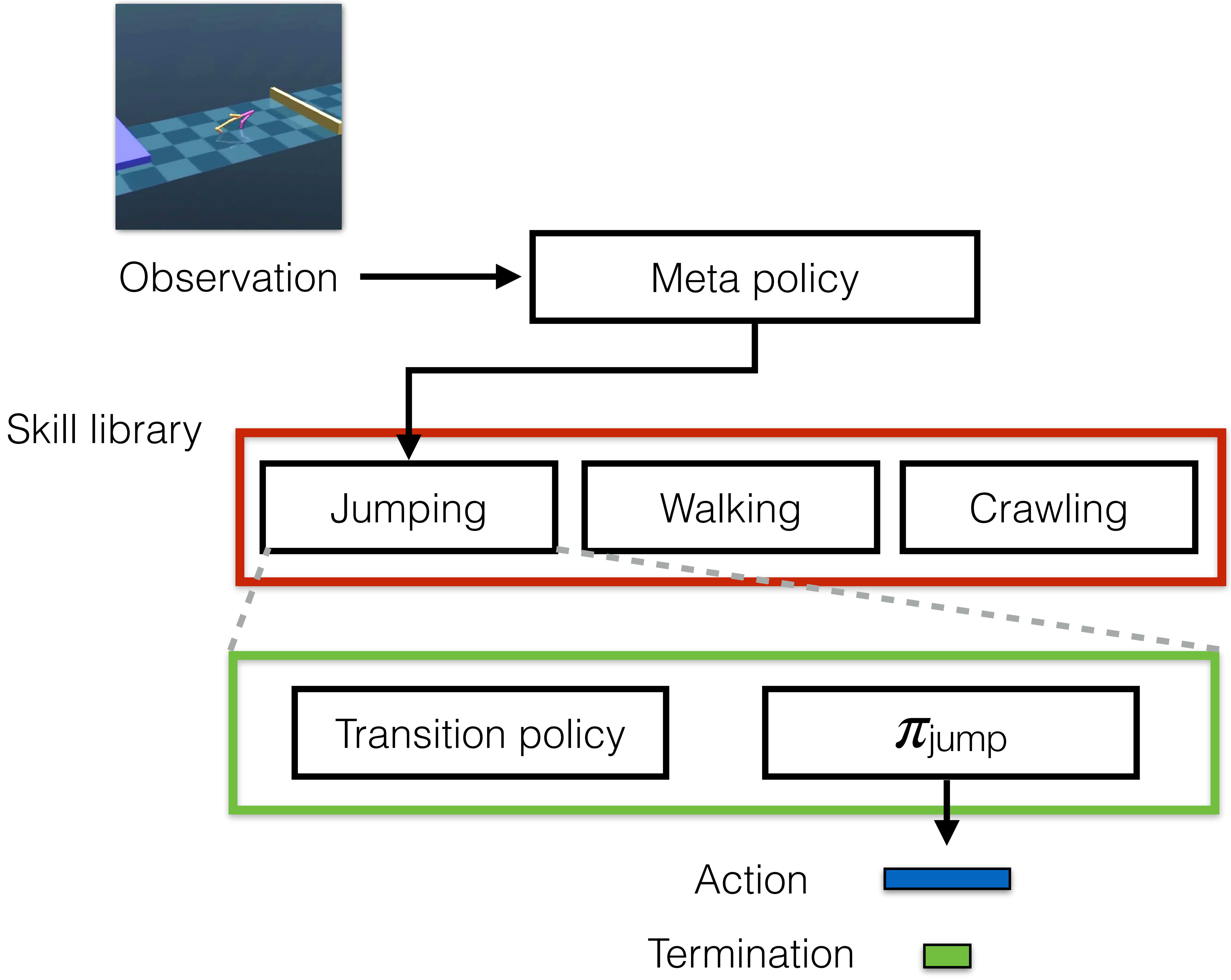
Model


Observation
Meta policy
Skill library
Jumping
Walking
Crawling
Transition policy
π_{jump}
Action
Termination
Repeat until reach a good initial state

Model

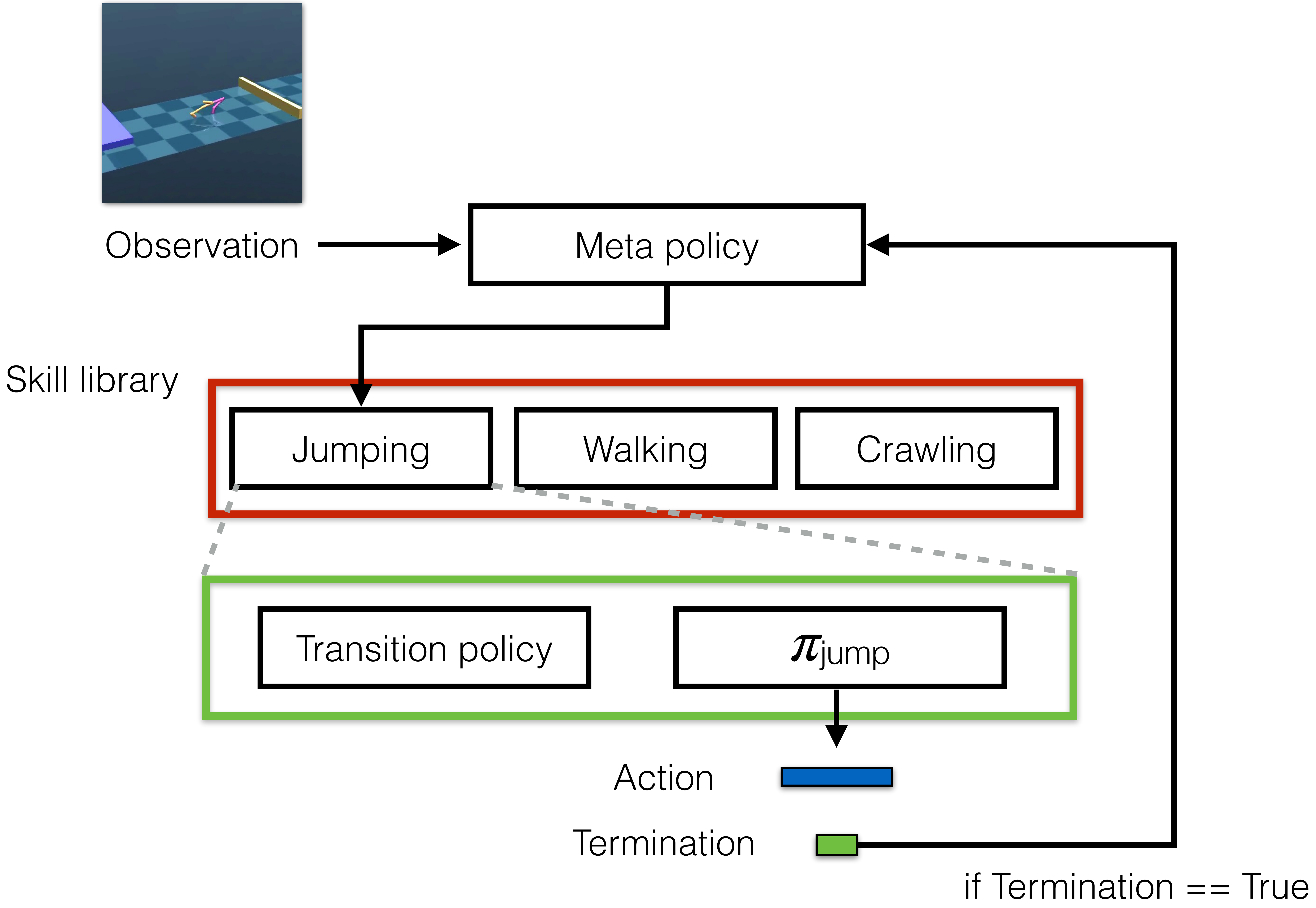

Observation
Meta policy
Skill library
Jumping
Walking
Crawling
Transition policy
π_{jump}
Action
Termination

Model

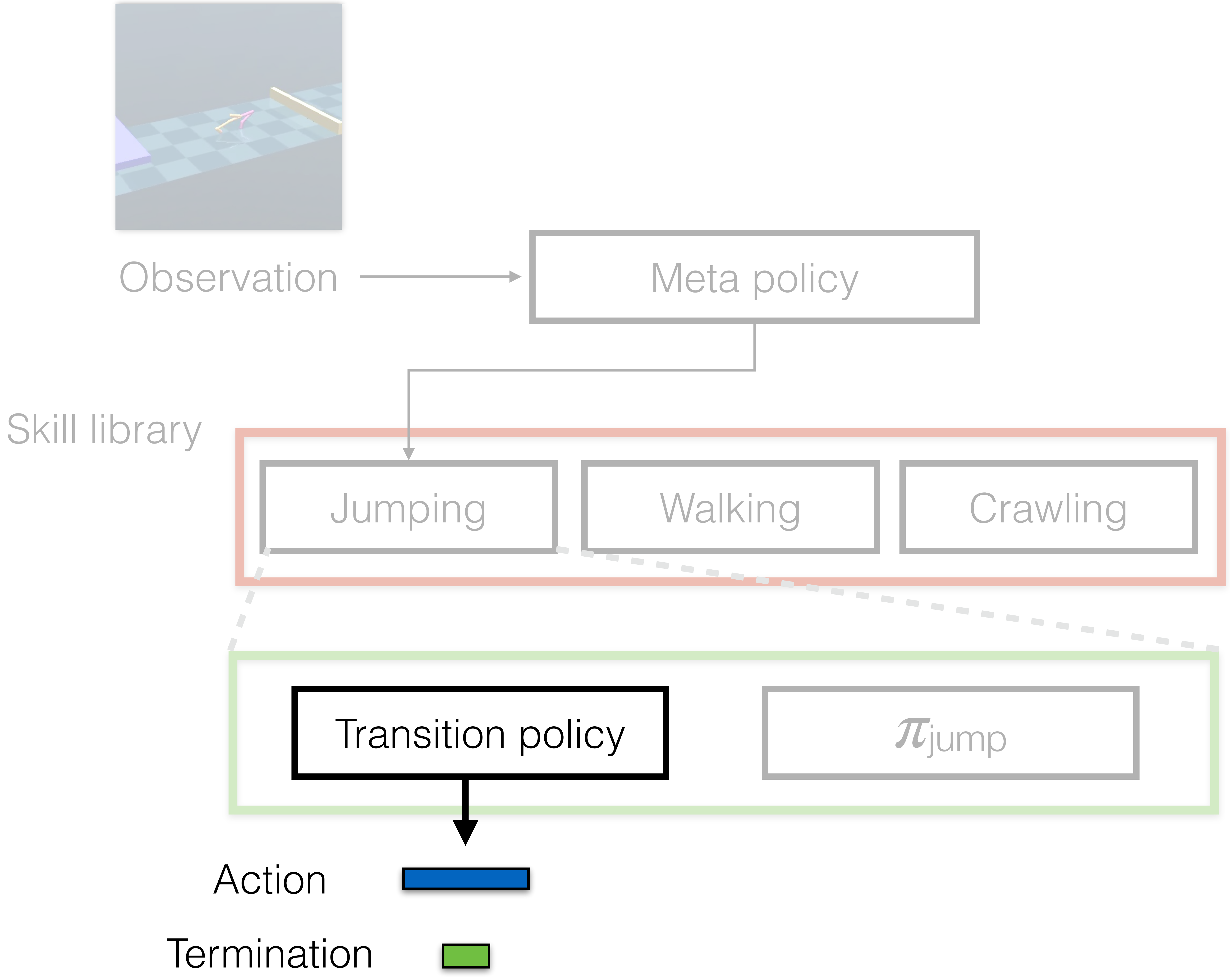


Observation
Meta policy
Skill library
Jumping
Walking
Crawling
Transition policy
π_{jump}
Action
Termination

Model

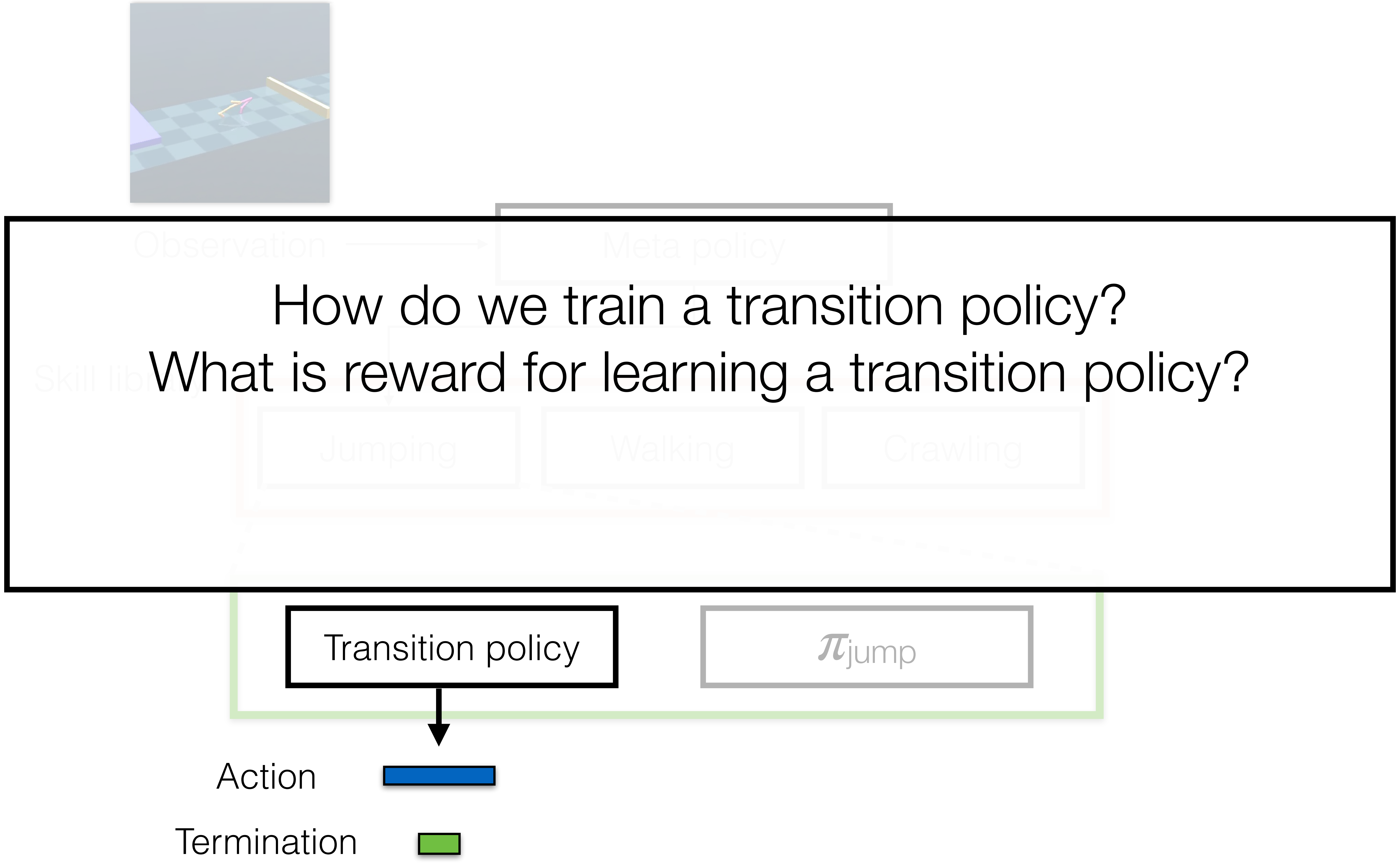

Observation
Meta policy
Skill library
Jumping
Walking
Crawling
Transition policy
π_{jump}
Action
Termination
if Termination == True

Model



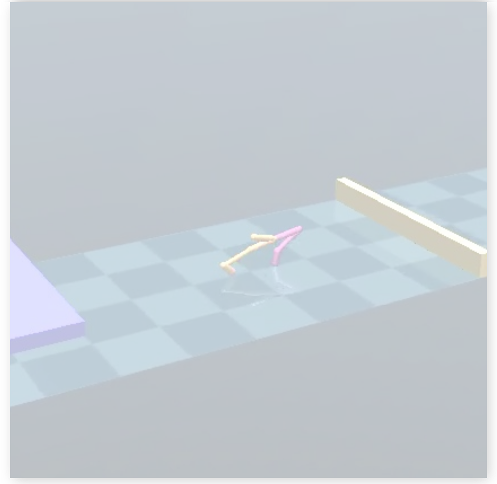
Observation
Meta policy
Skill library
Jumping
Walking
Crawling
Transition policy
π_{jump}
Action
Termination

Model



How do we train a transition policy?
What is reward for learning a transition policy?

Model



How do we train a transition policy?
What is reward for learning a transition policy?

- Success of the following skill

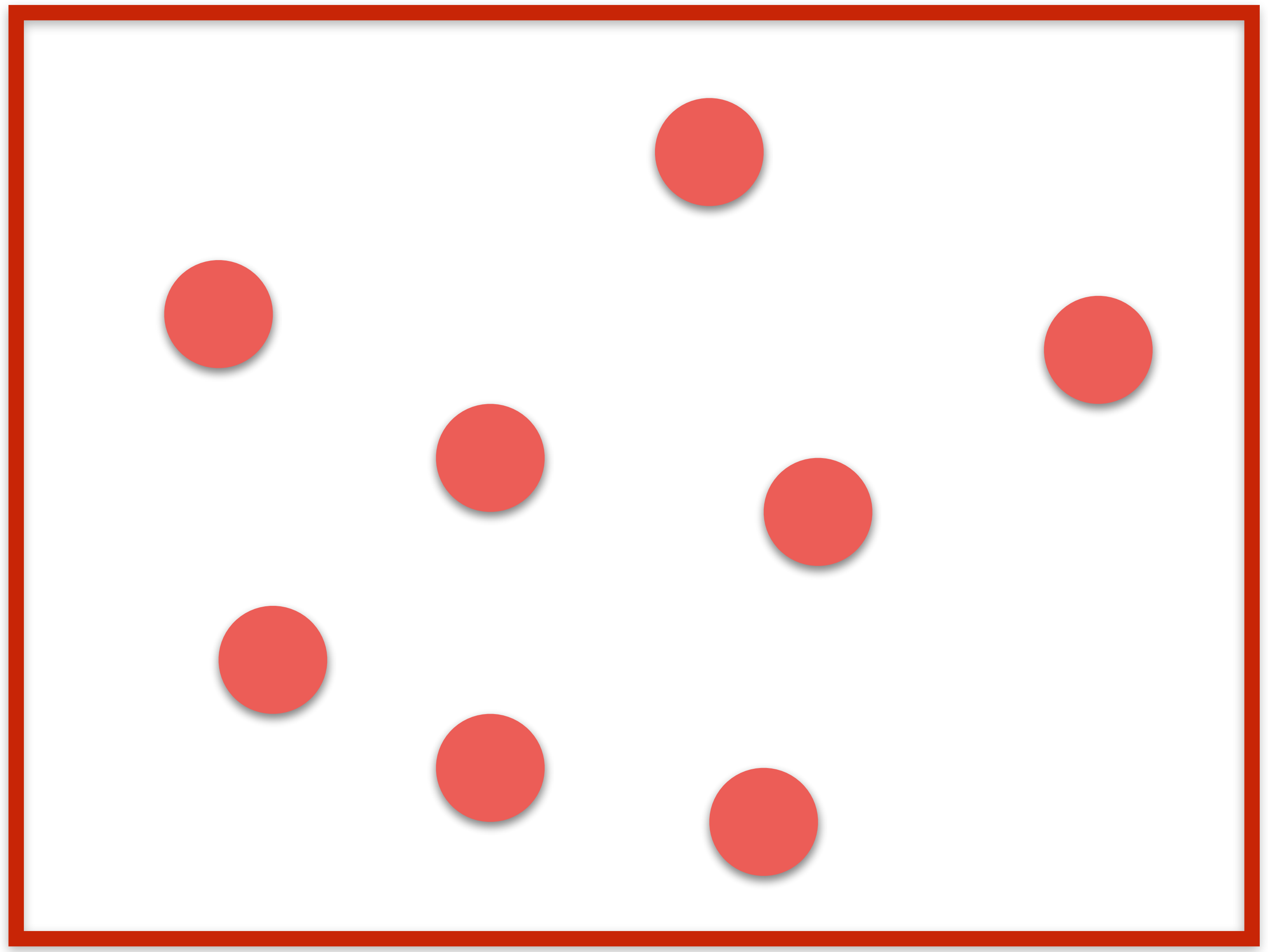
Transition policy

π_{jump}

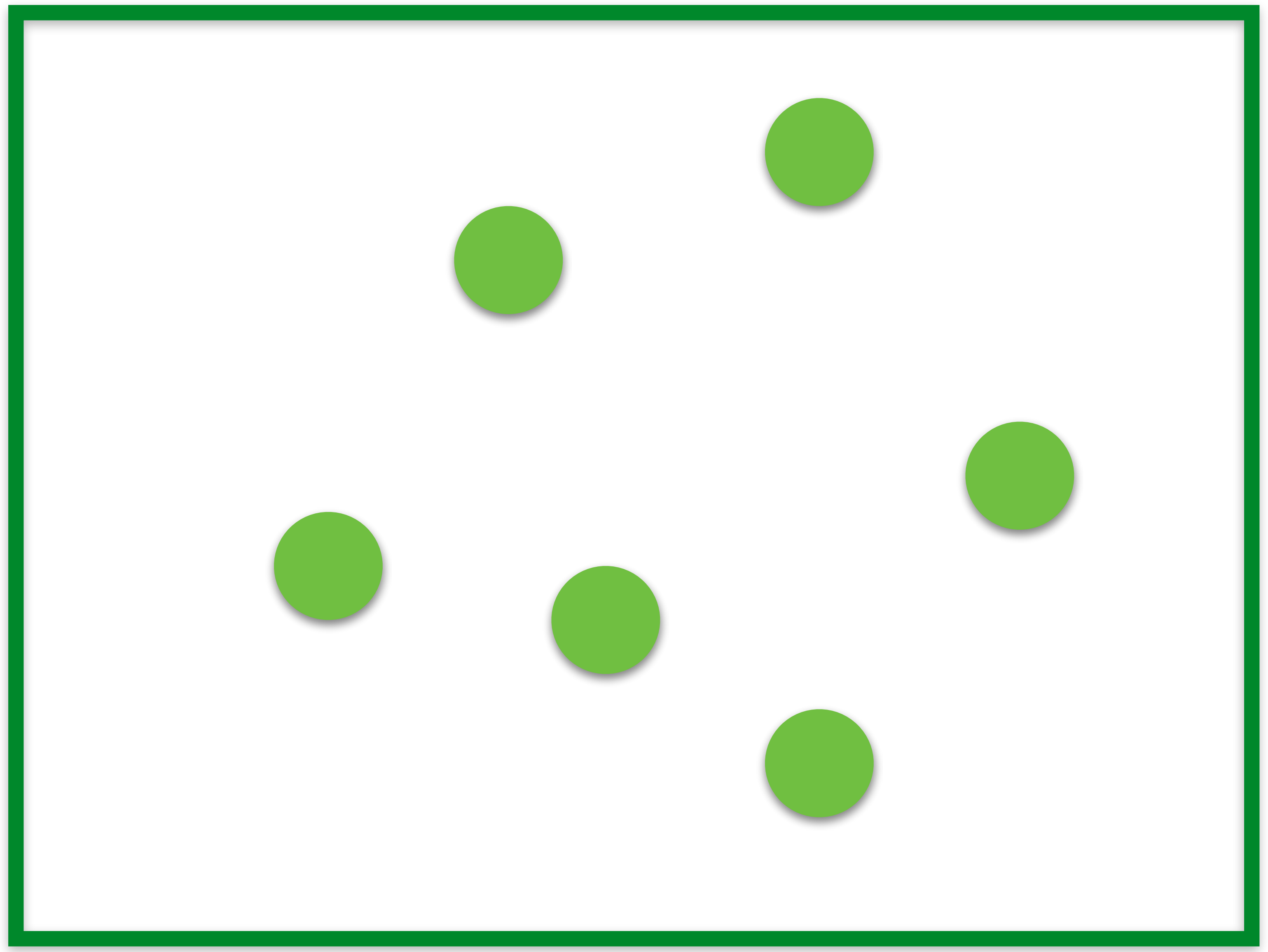
Action

Termination

Learning Transition Policy

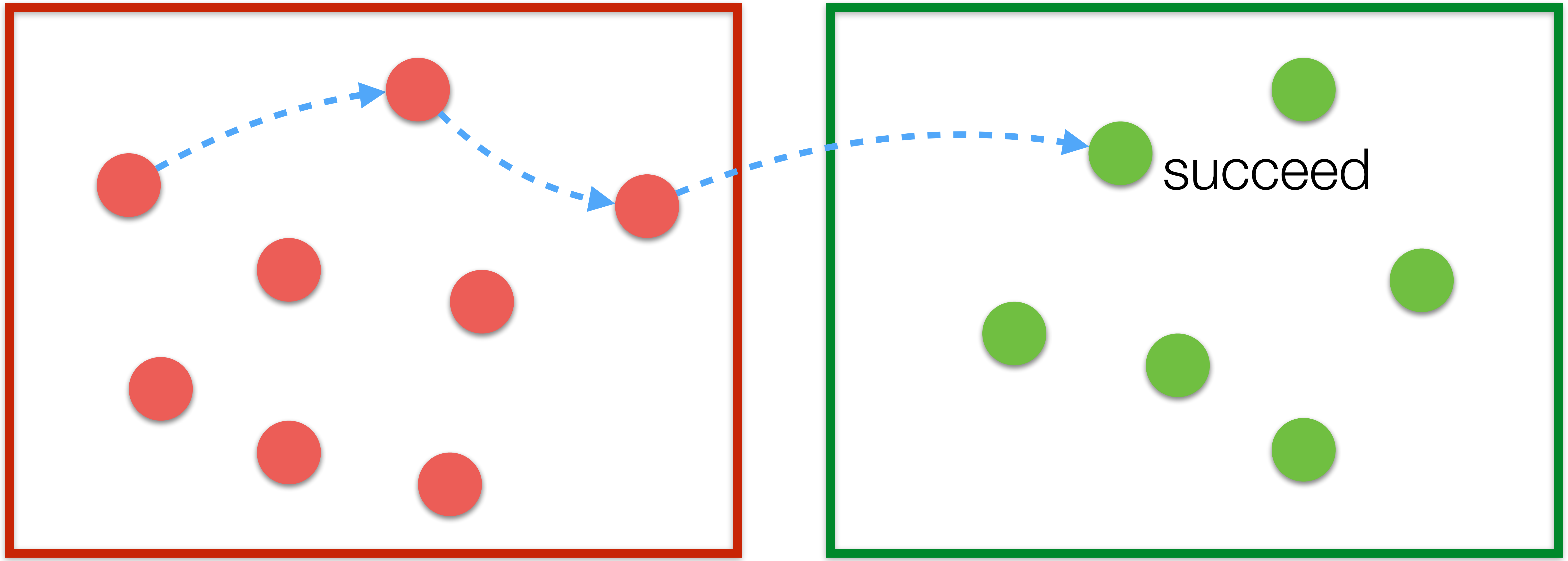


Bad initial states for π_{walk}



Good initial states for π_{walk}

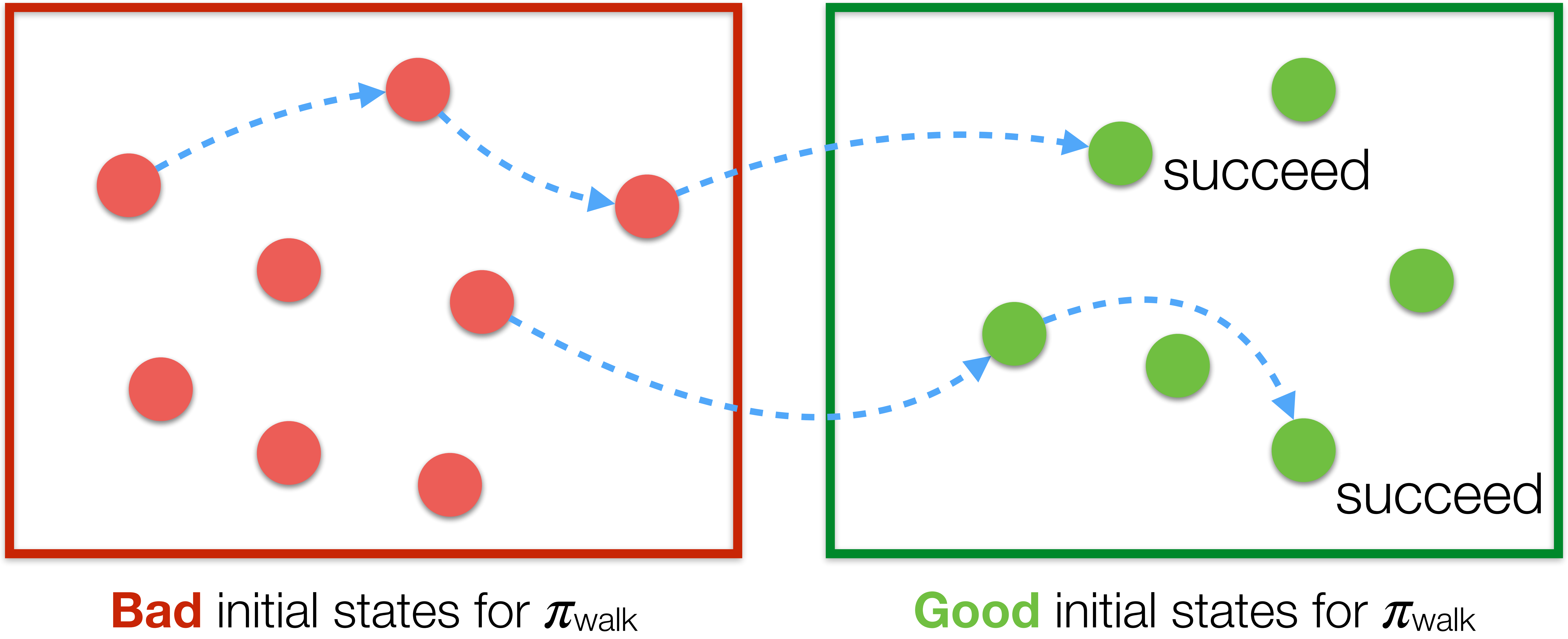
Learning Transition Policy



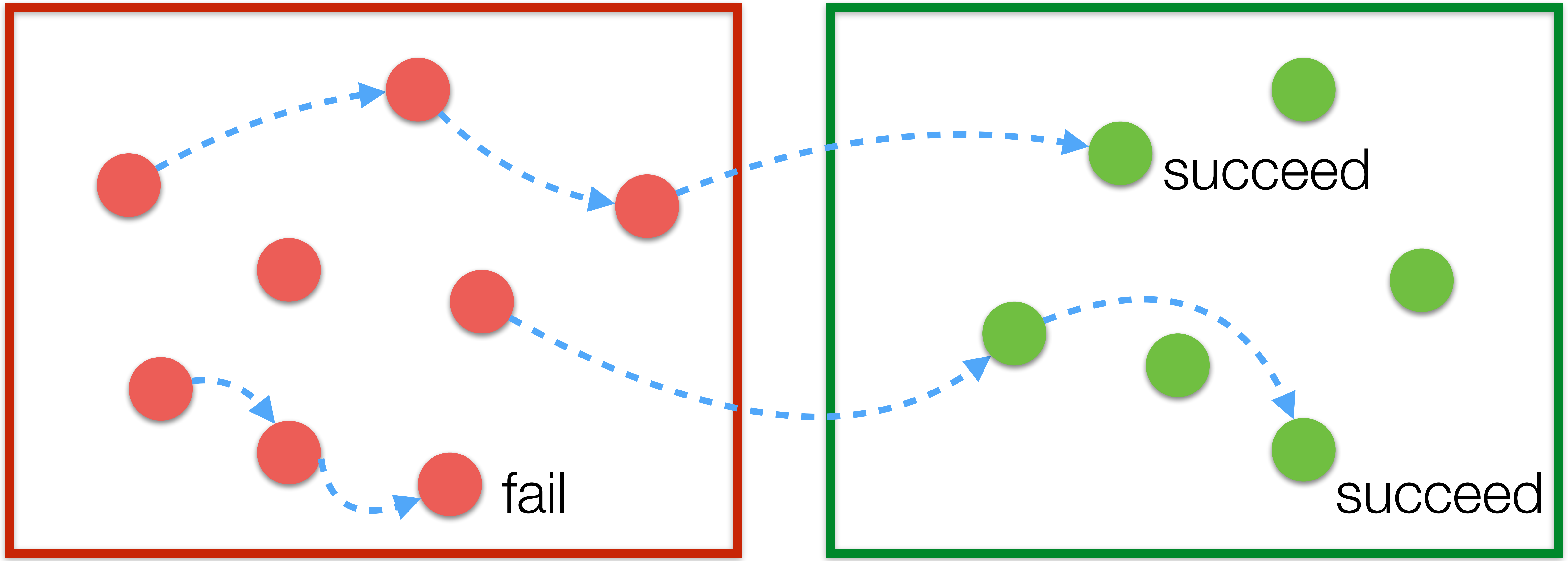
Bad initial states for π_{walk}

Good initial states for π_{walk}

Learning Transition Policy



Learning Transition Policy

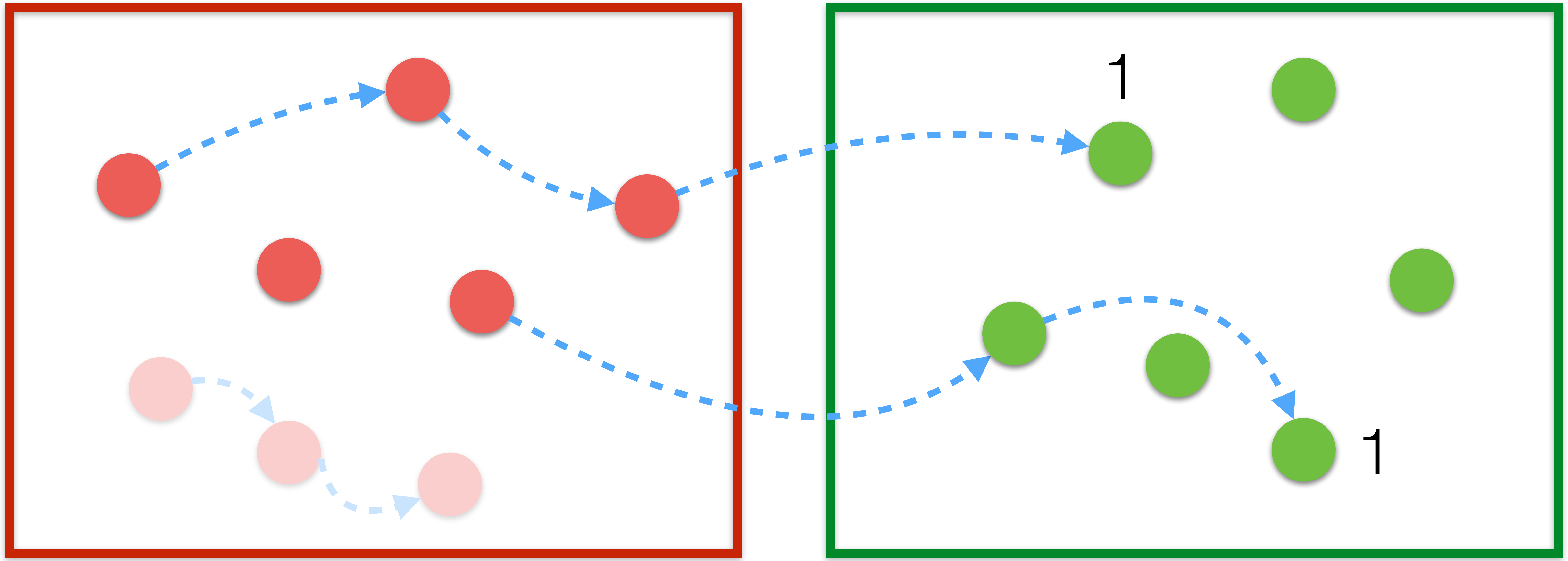


Bad initial states for π_{walk}

Good initial states for π_{walk}

Learning Transition Policy

Successful execution of the following skill: +1



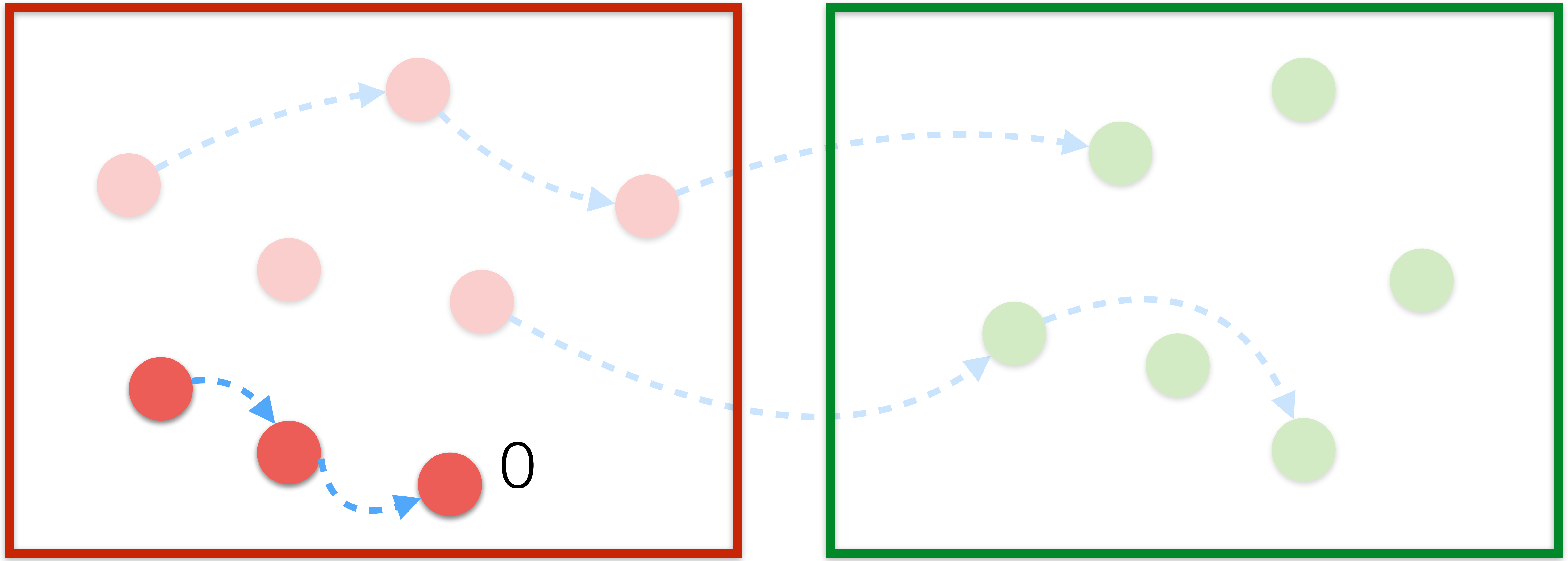
Bad initial states for π_{walk}

Good initial states for π_{walk}

Learning Transition Policy

Successful execution of the following skill: +1

Failing execution of the following skill: 0

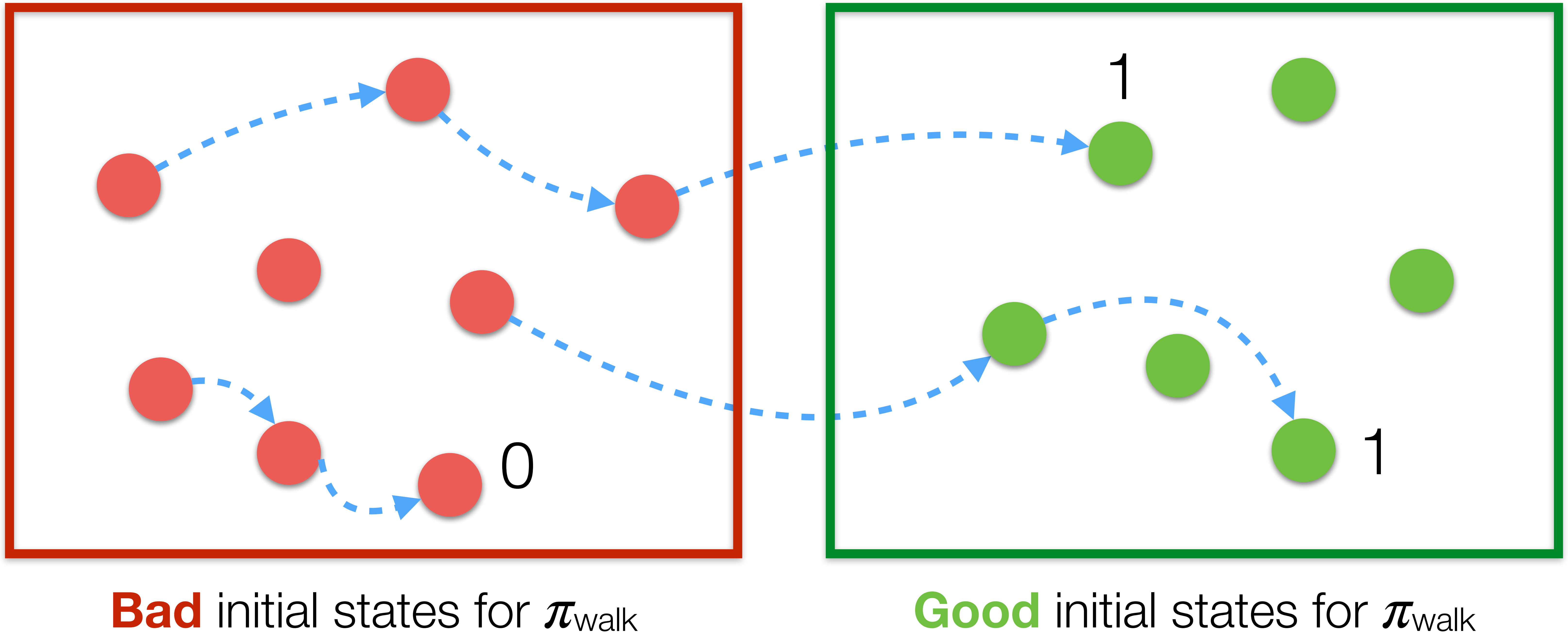


Bad initial states for π_{walk} **Good** initial states for π_{walk}

Learning Transition Policy

Successful execution of the following skill: +1

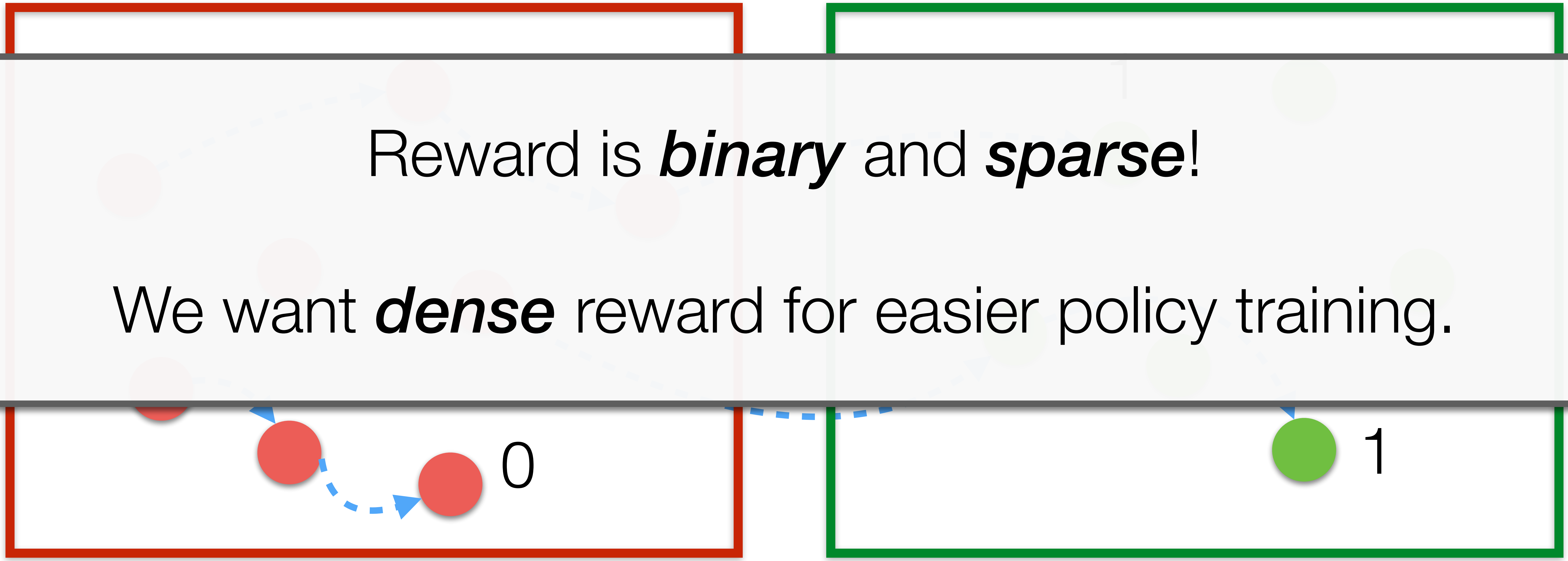
Failing execution of the following skill: 0



Learning Transition Policy

Successful execution of the following skill: +1

Failing execution of the following skill: 0



Reward is ***binary*** and ***sparse***!

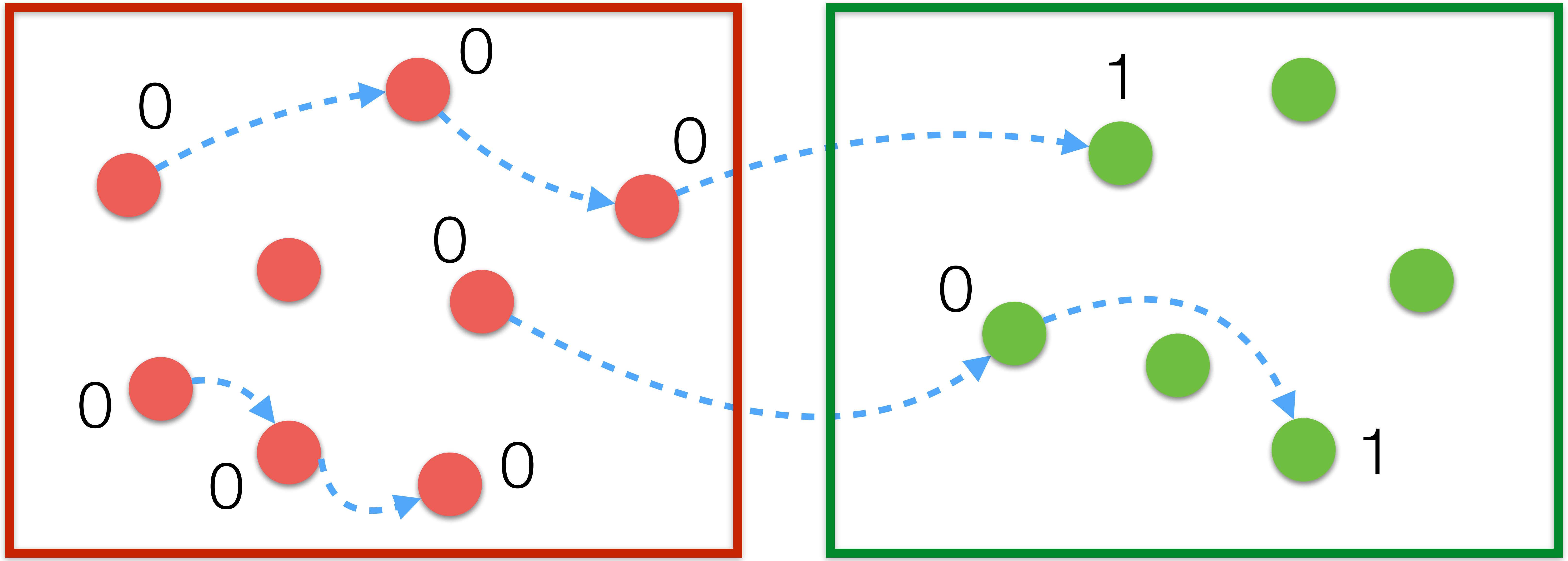
We want ***dense*** reward for easier policy training.

Bad initial states for π_{walk}

Good initial states for π_{walk}

Proximity Reward

Instead of binary reward

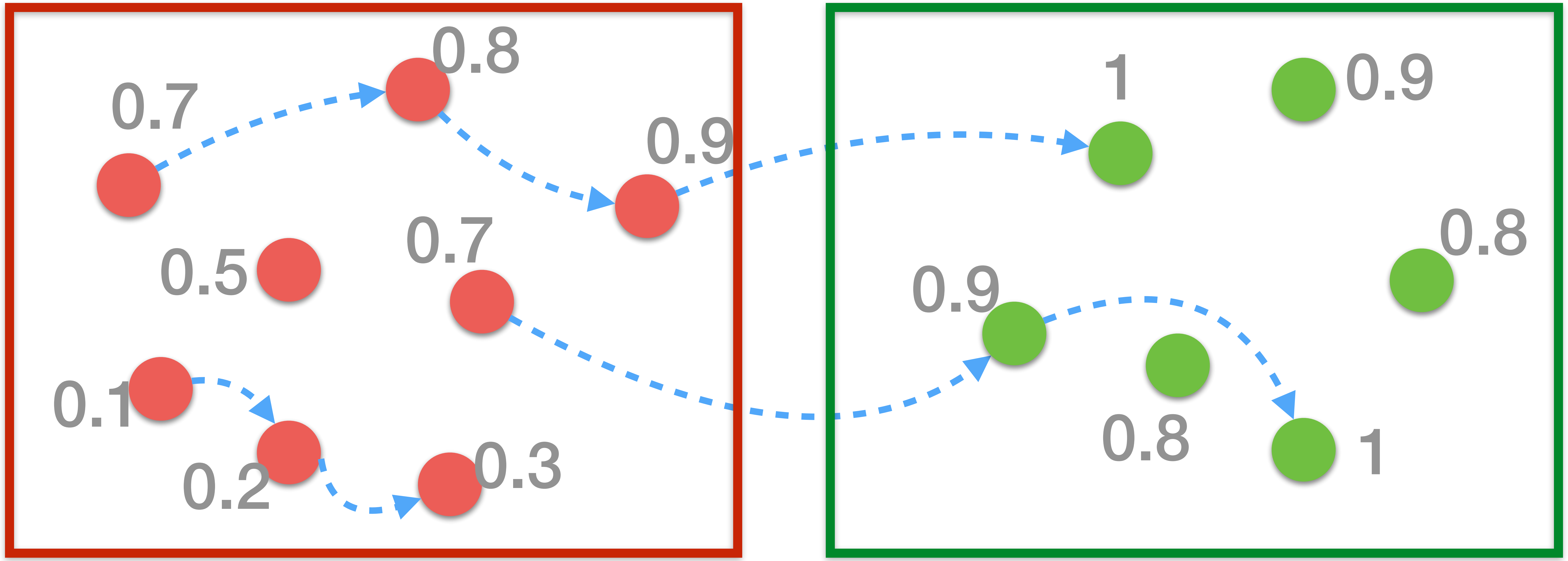


Bad initial states for π_{walk}

Good initial states for π_{walk}

Proximity Reward

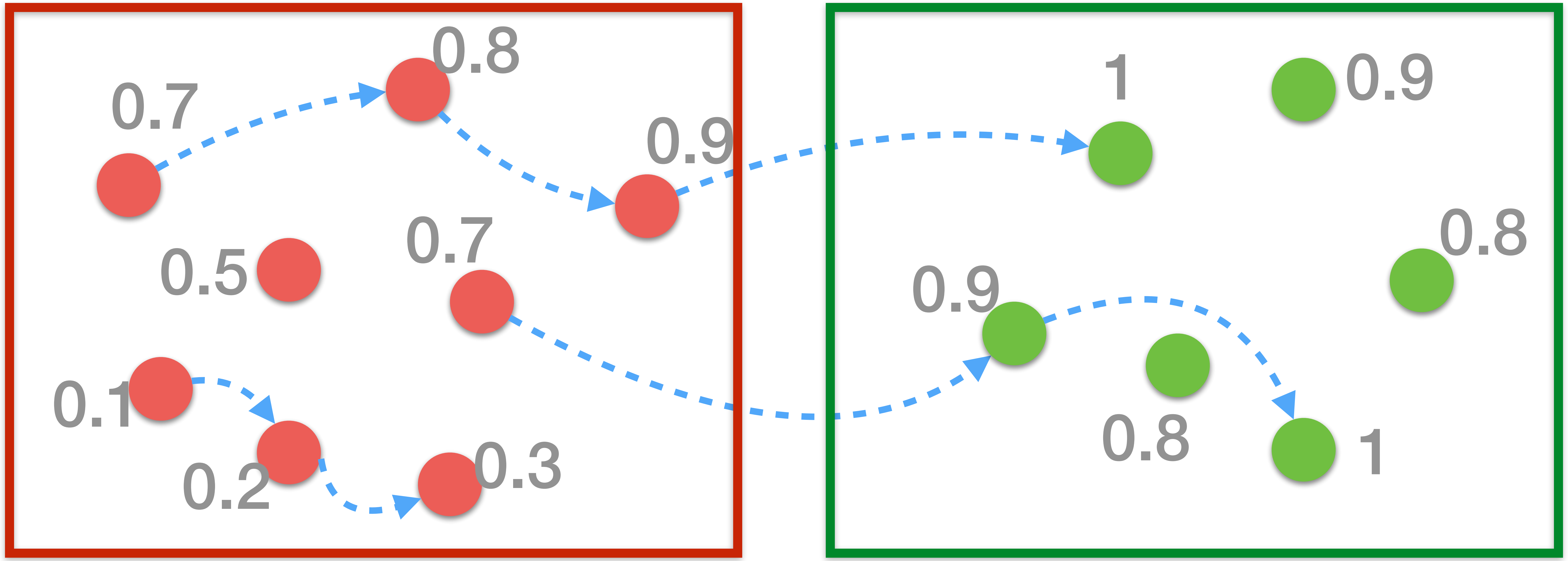
Instead of binary reward, use ***"proximity prediction"***,
which estimates how close to good initial states



Bad initial states for π_{walk} **Good** initial states for π_{walk}

Proximity Reward

Instead of binary reward, use ***"proximity prediction"***,
which estimates how close to good initial states

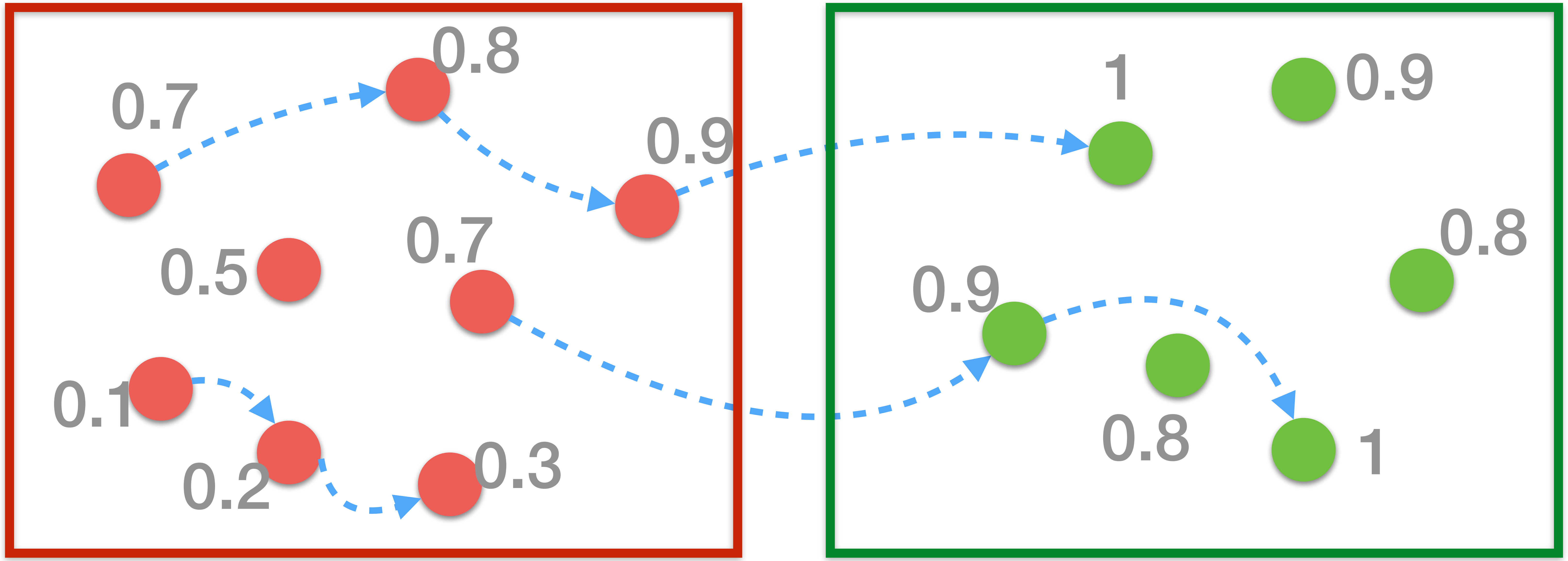


Bad initial states for π_{walk} **Good** initial states for π_{walk}

We define ***proximity*** as: $P(s) = \delta^{step}$

Proximity Reward

Instead of binary reward, use ***"proximity prediction"***,
which estimates how close to good initial states

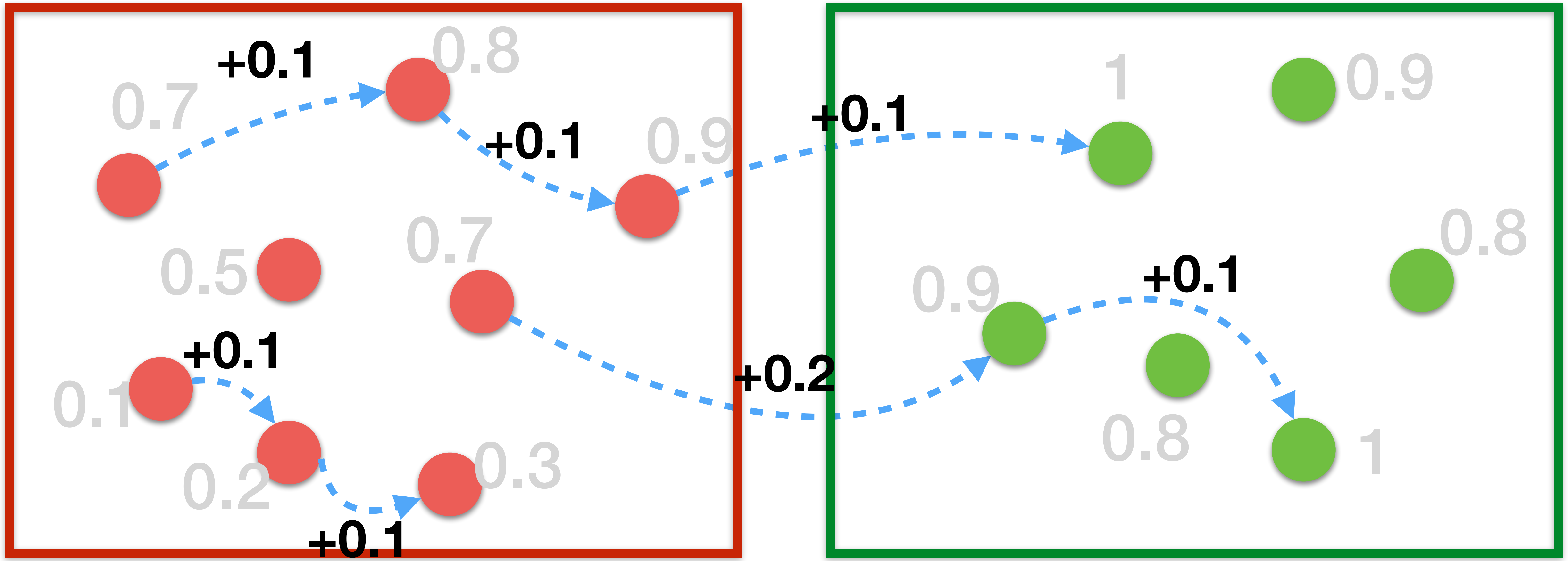


Bad initial states for π_{walk} **Good** initial states for π_{walk}

We define ***proximity*** as: $P(s) = \delta^{step}$

Proximity Reward

Instead of binary reward, use ***"proximity prediction"***,
which estimates how close to good initial states

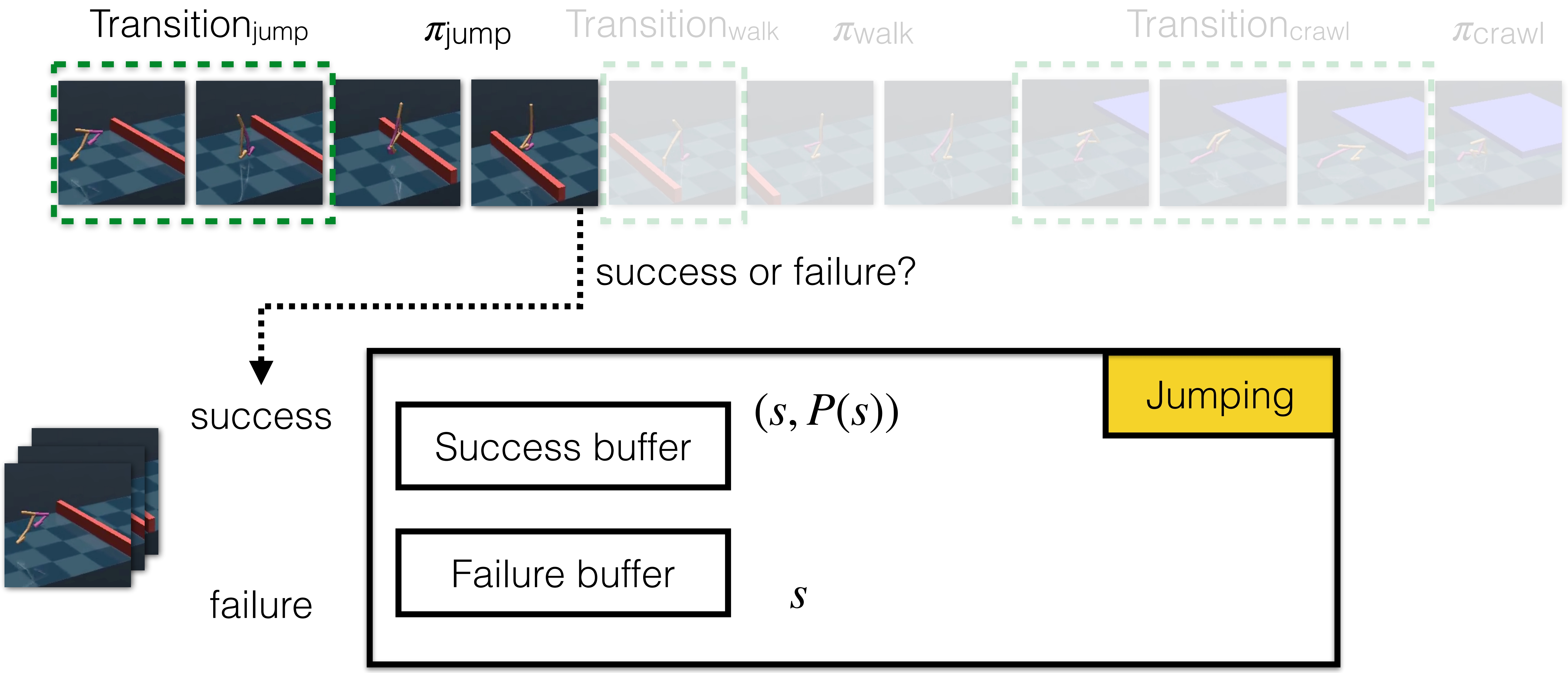


Bad initial states for π_{walk} **Good** initial states for π_{walk}

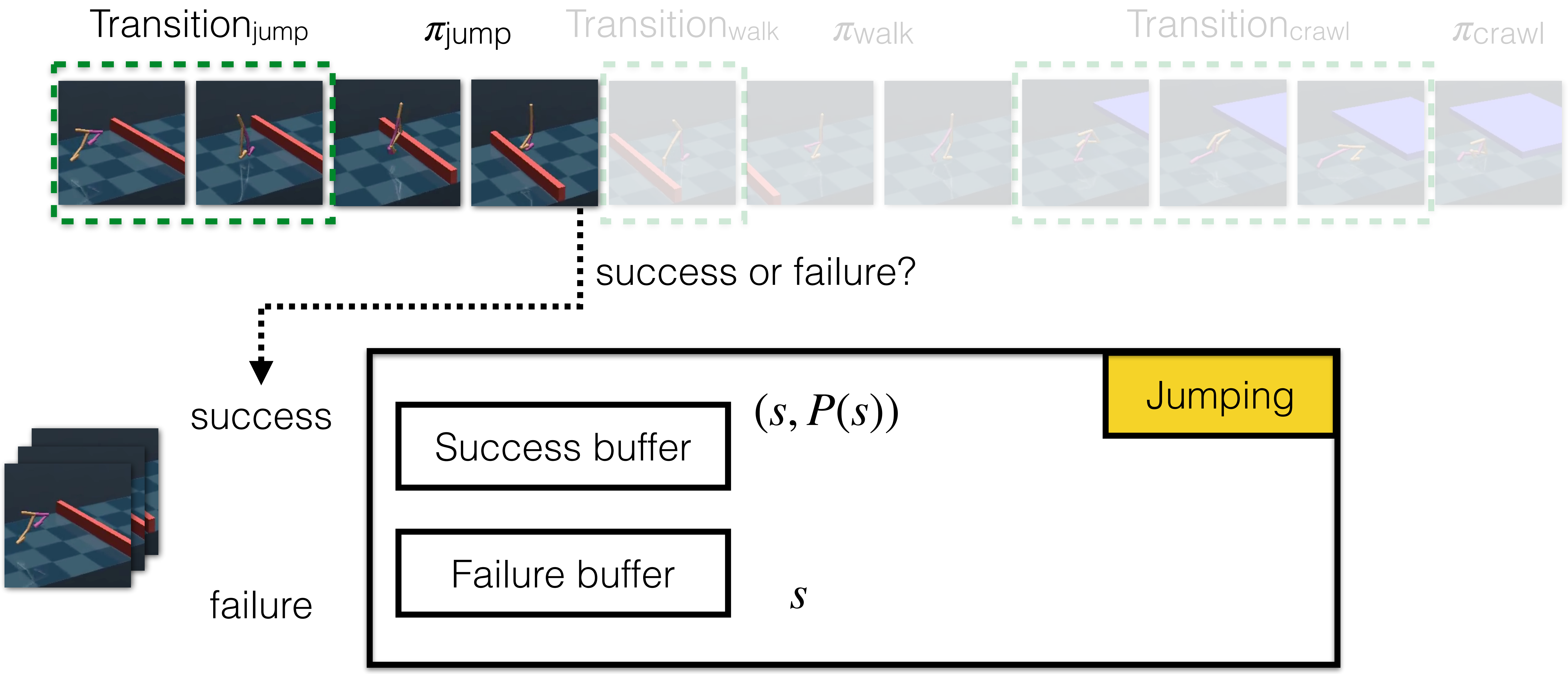
We define ***proximity*** as: $P(s) = \delta^{step}$

and provide ***proximity reward*** every step: $P(s_{t+1}) - P(s_t)$

Training Proximity Predictor

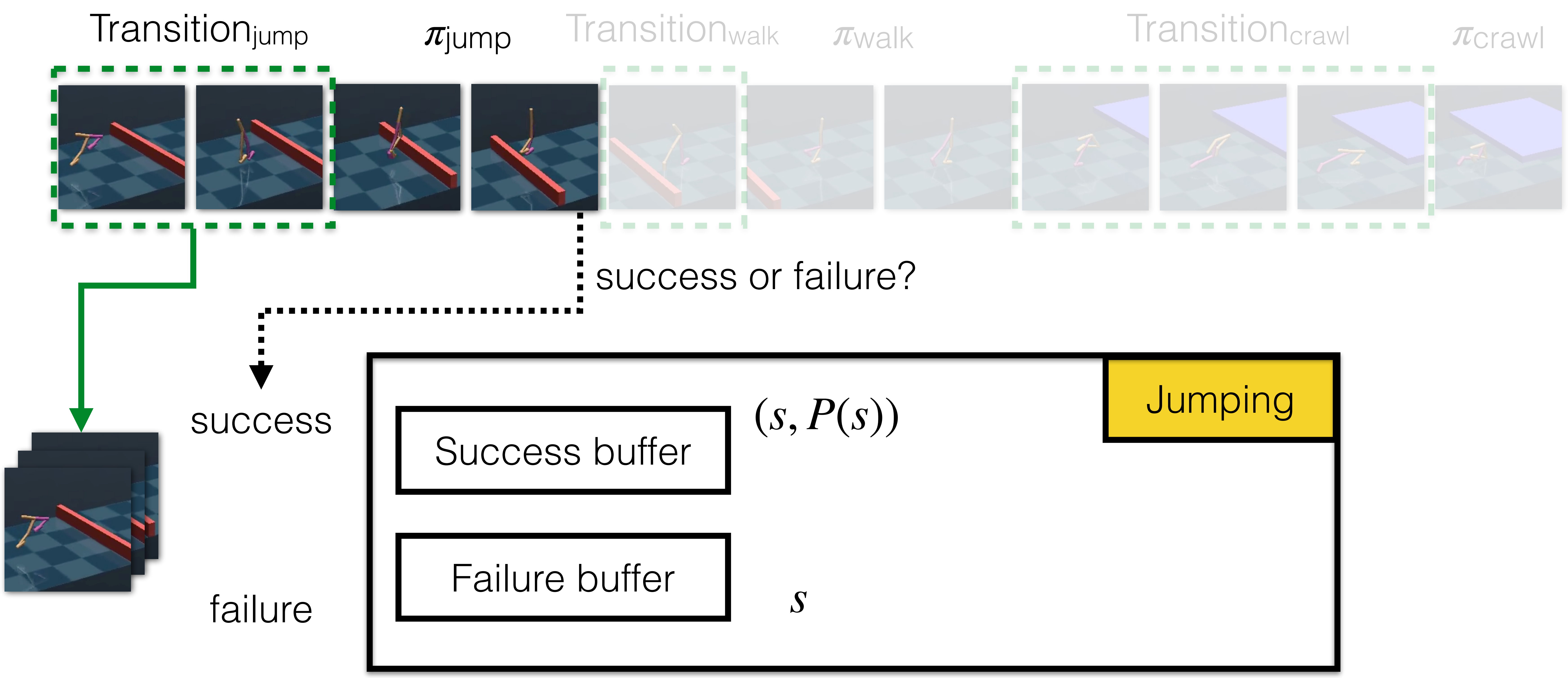


Training Proximity Predictor



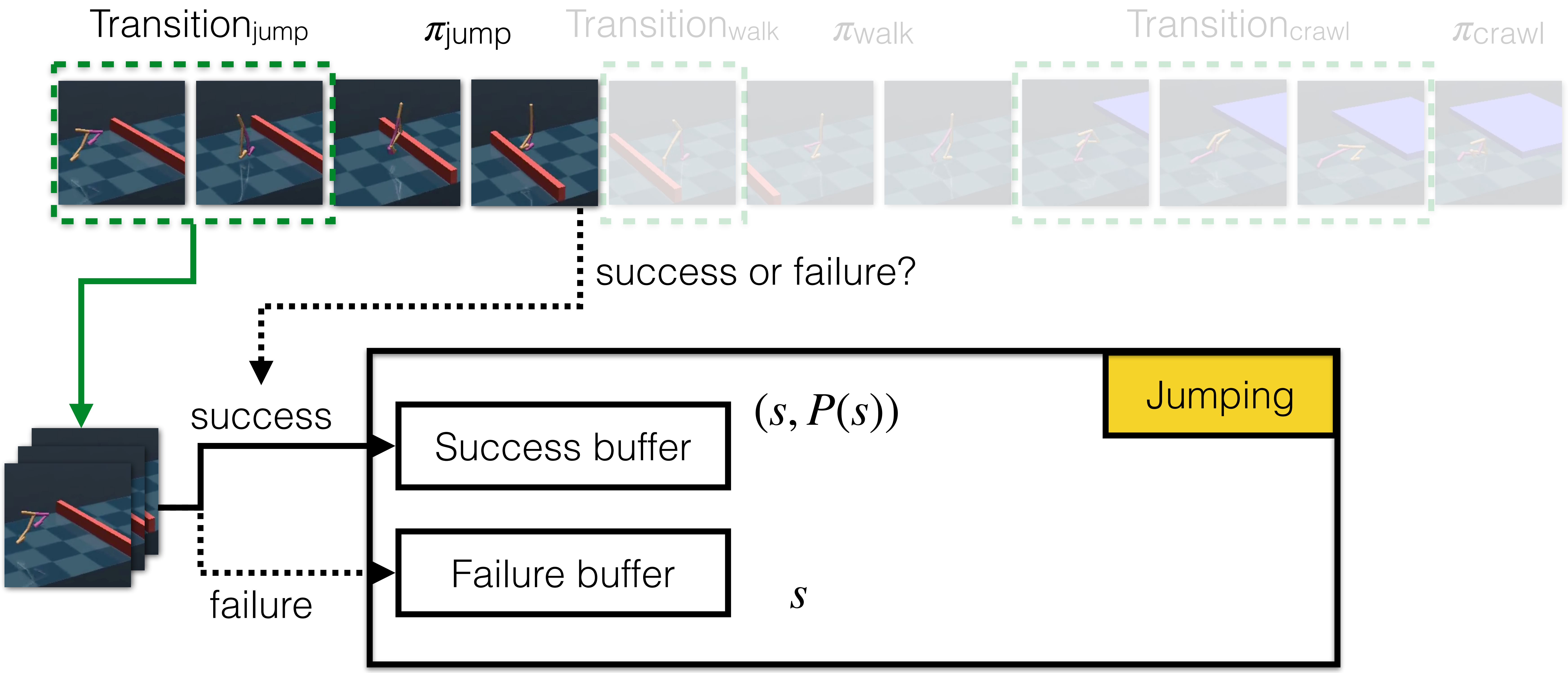
Collect training data for proximity predictors

Training Proximity Predictor



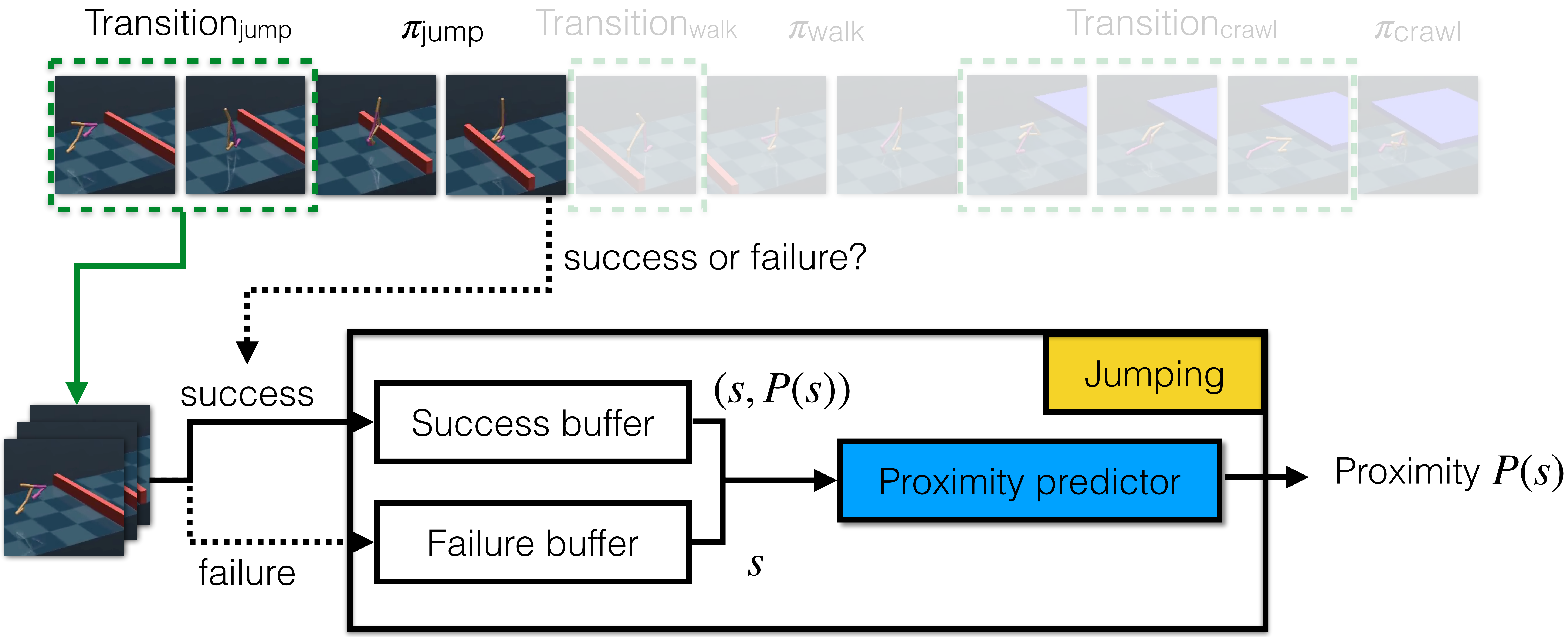
Collect training data for proximity predictors

Training Proximity Predictor



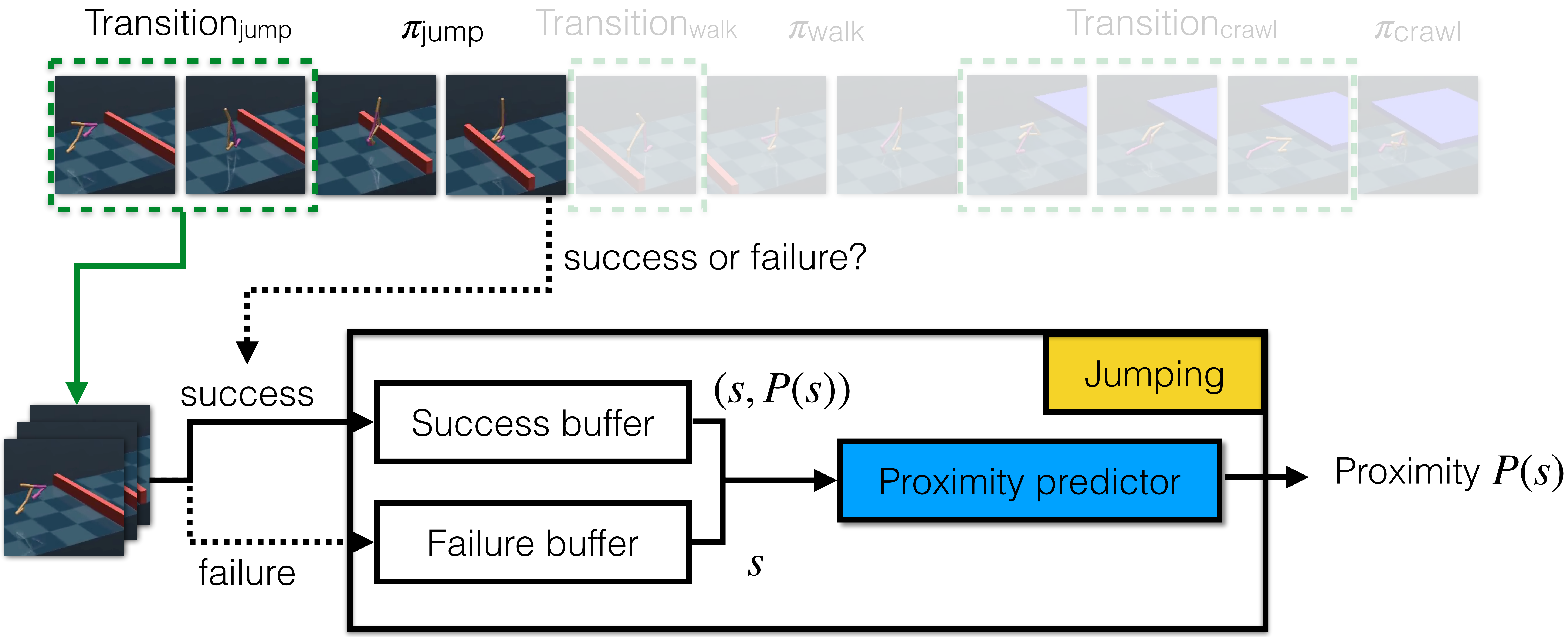
Collect training data for proximity predictors

Training Proximity Predictor



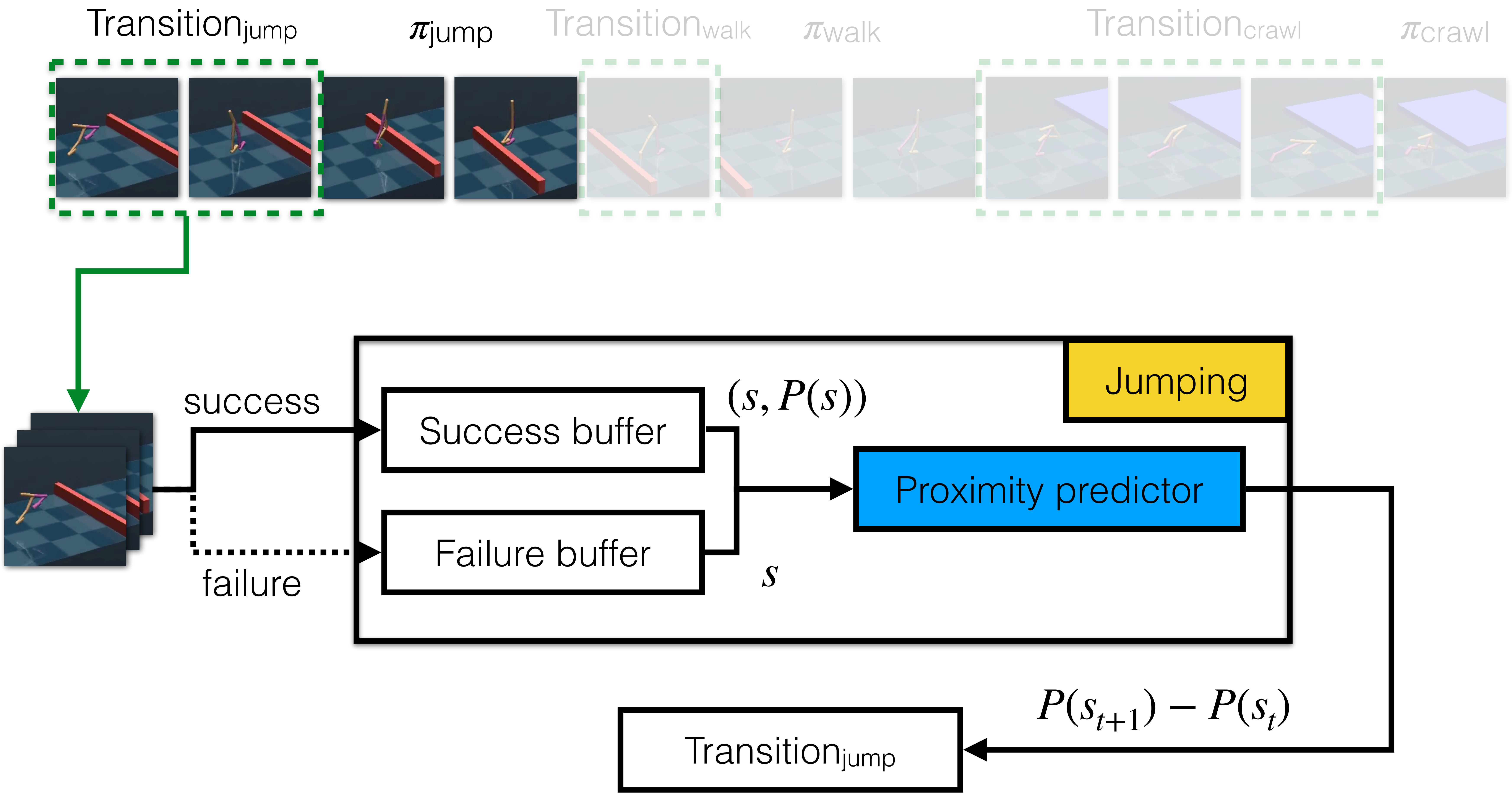
Train proximity predictors

Training Proximity Predictor



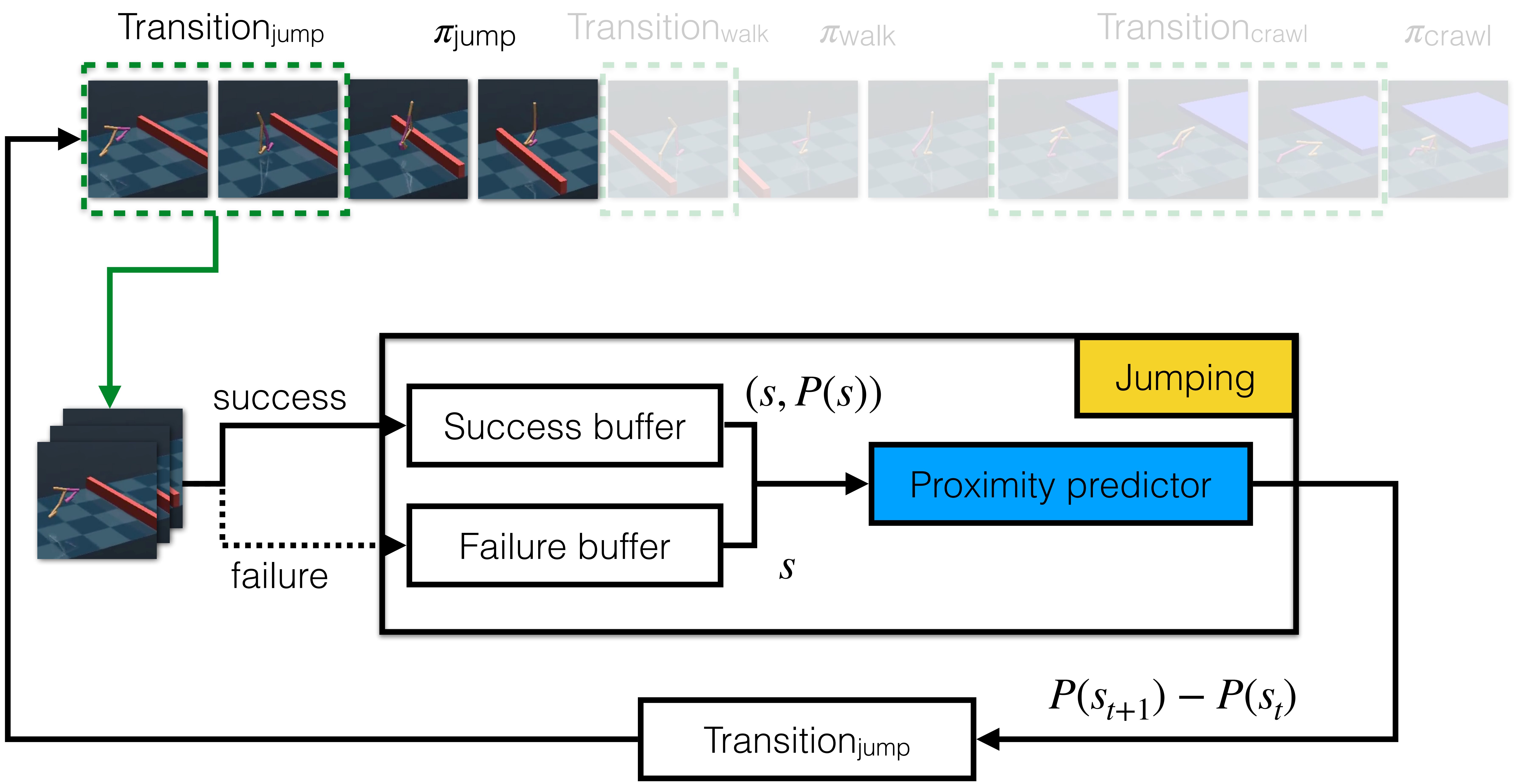
Train proximity predictors

Training Proximity Predictor



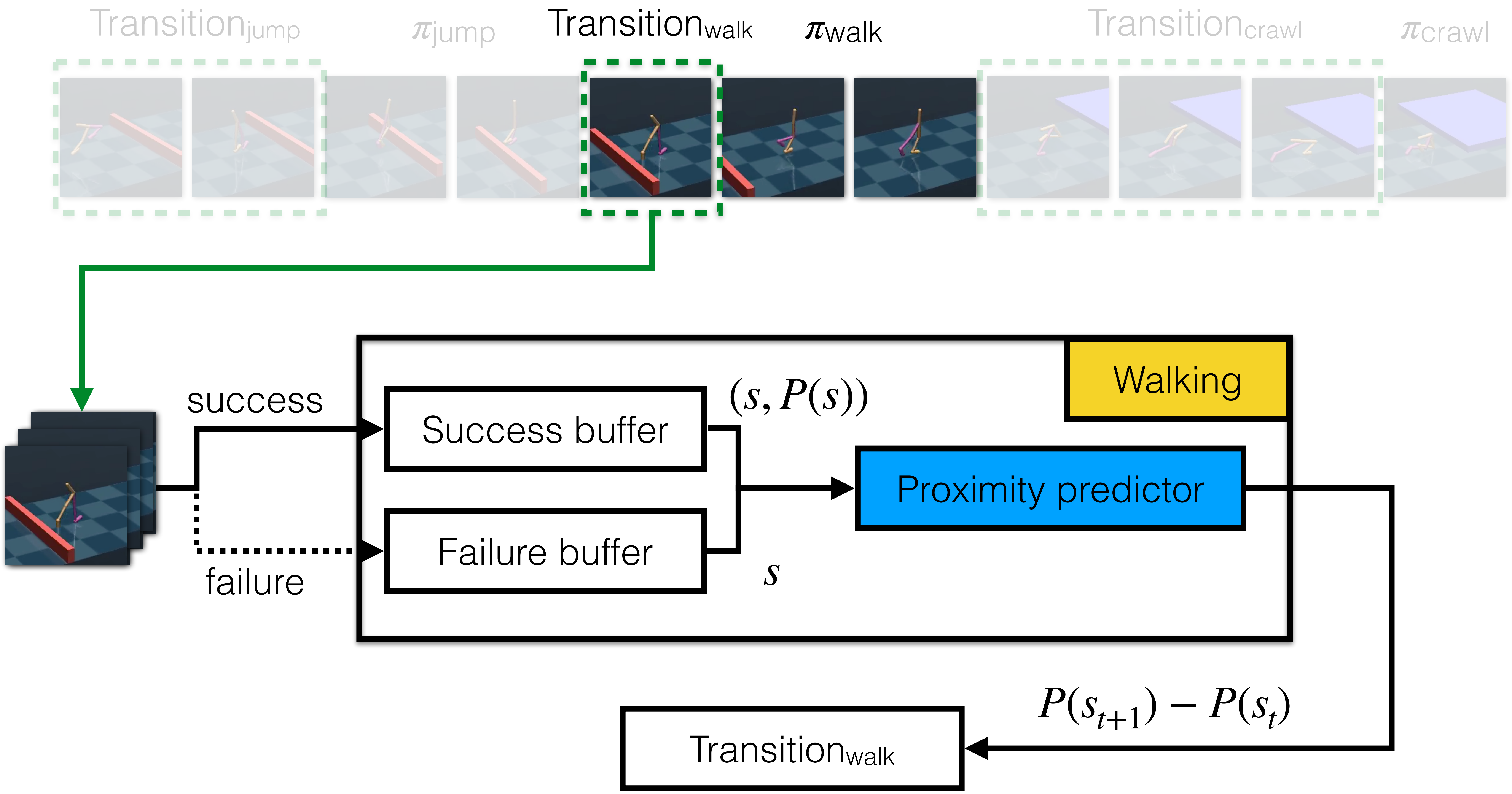
Provide more accurate proximity reward

Training Proximity Predictor



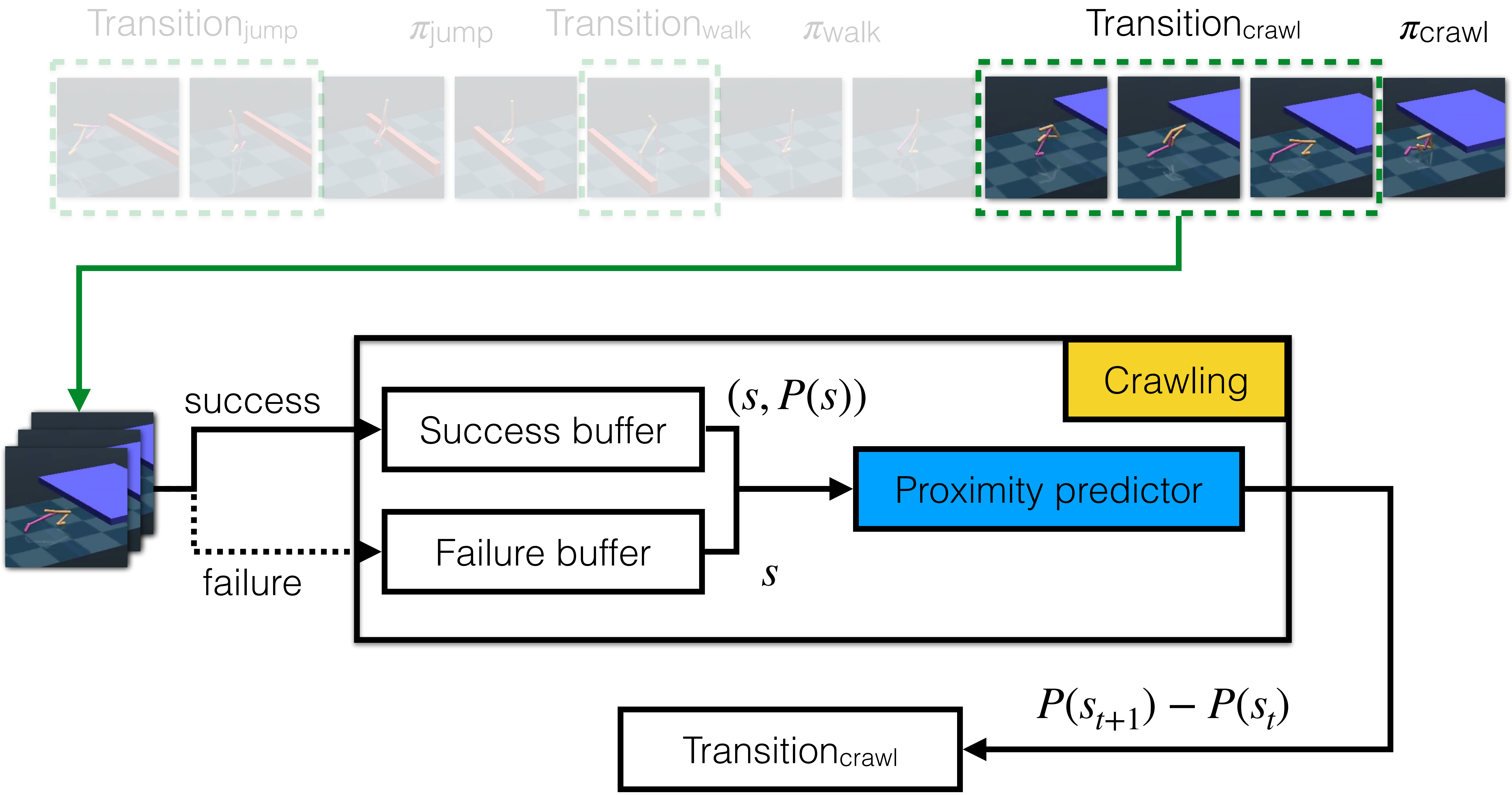
Getter better data with improved policy

Training Proximity Predictor



Train all transition policies simultaneously

Training Proximity Predictor



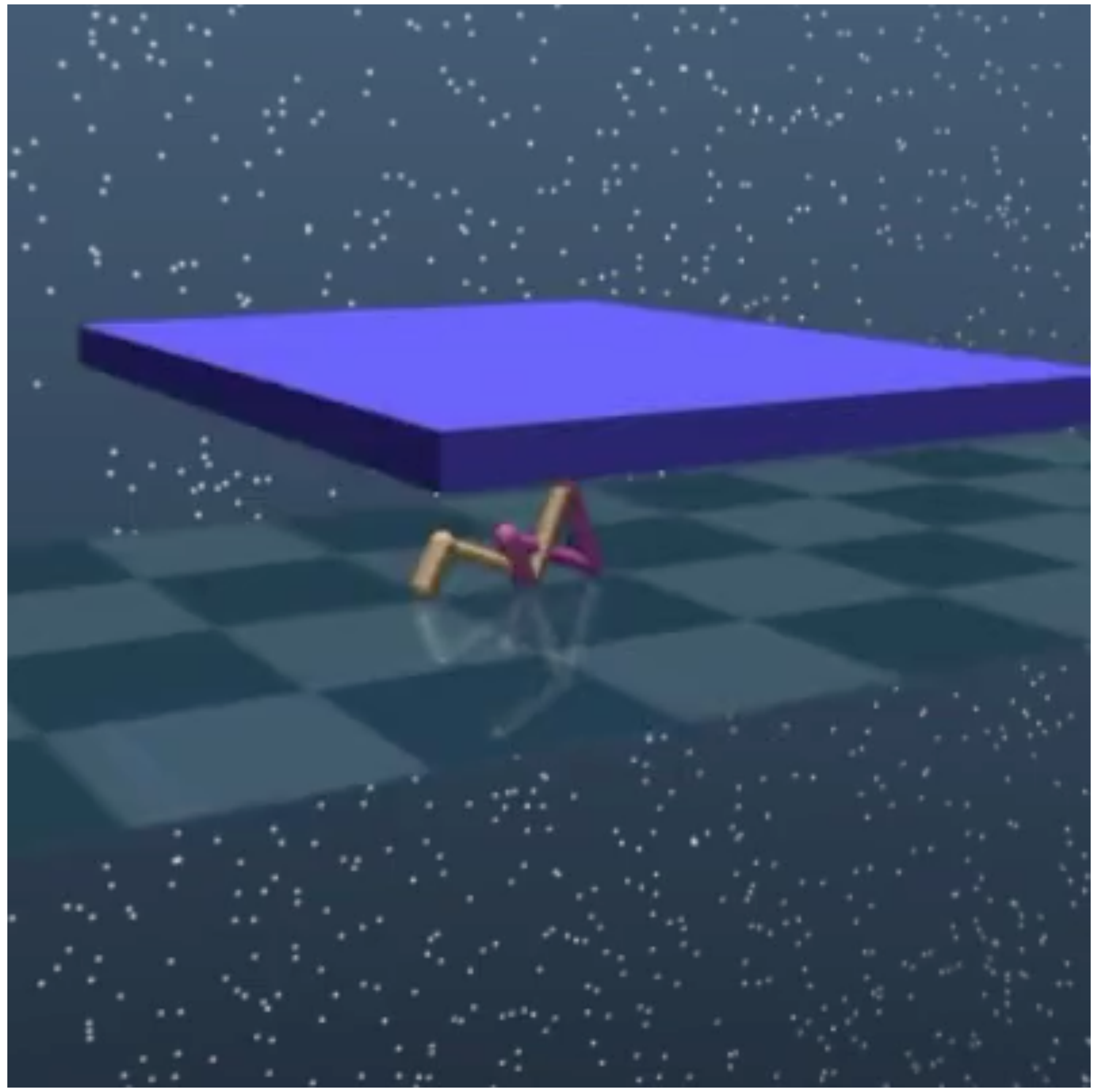
Train all transition policies simultaneously

Obstacle Course

Crawl

Transition

Walk

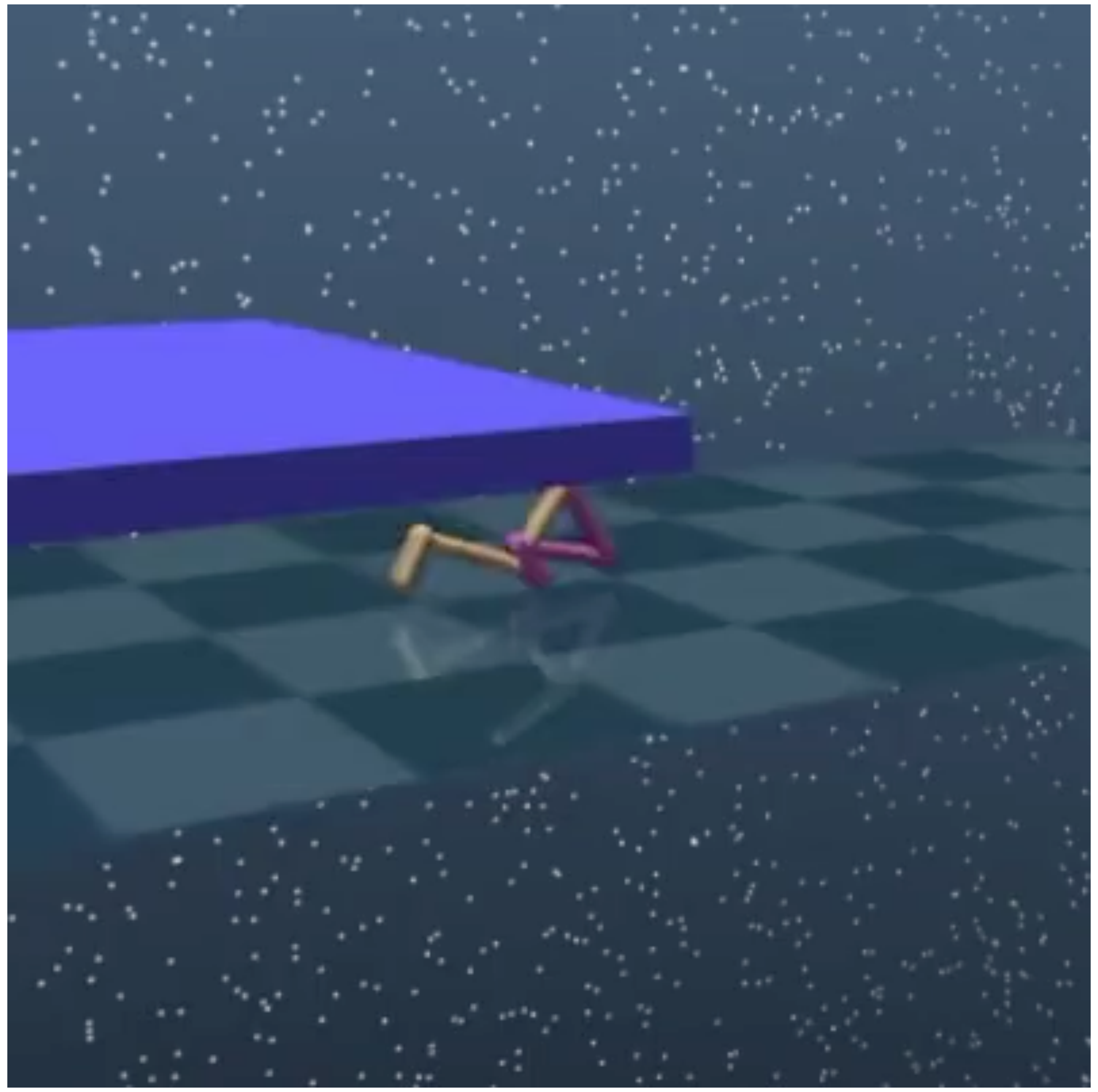


Obstacle Course

Crawl

Transition

Walk

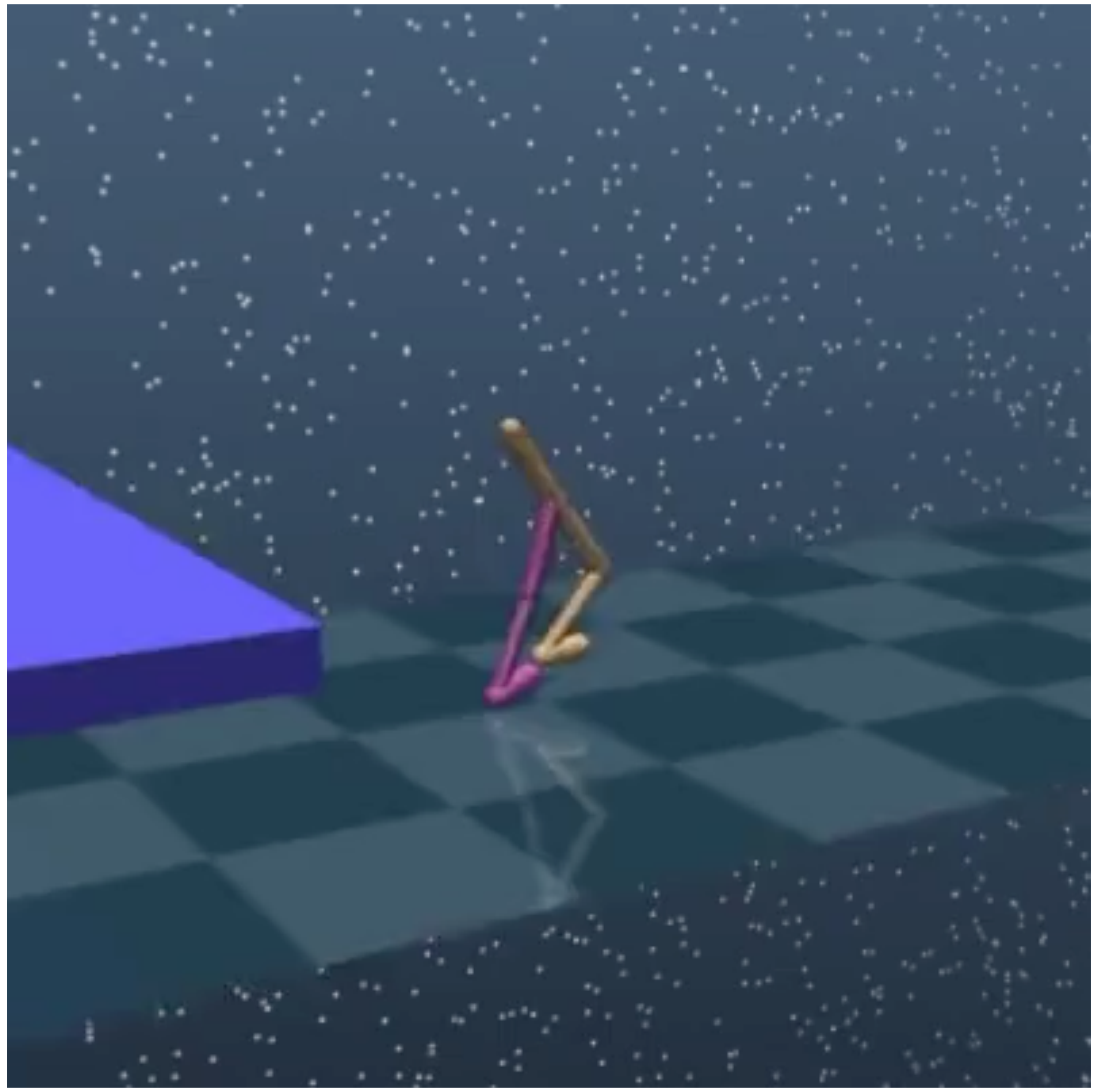


Obstacle Course

Crawl

Transition

Walk

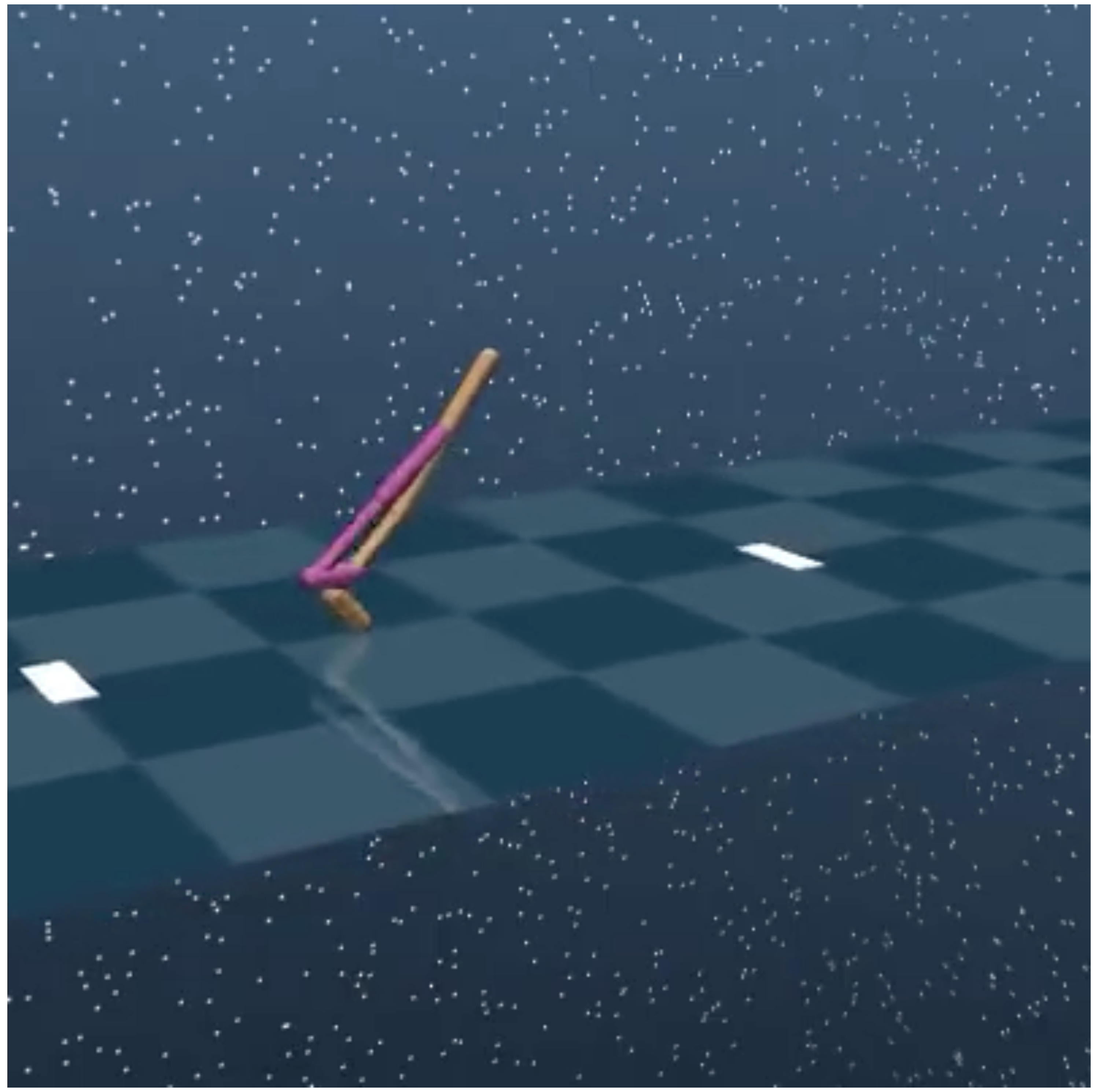


Walk Forward & Backward

Walk Forward

Transition

Walk Backward

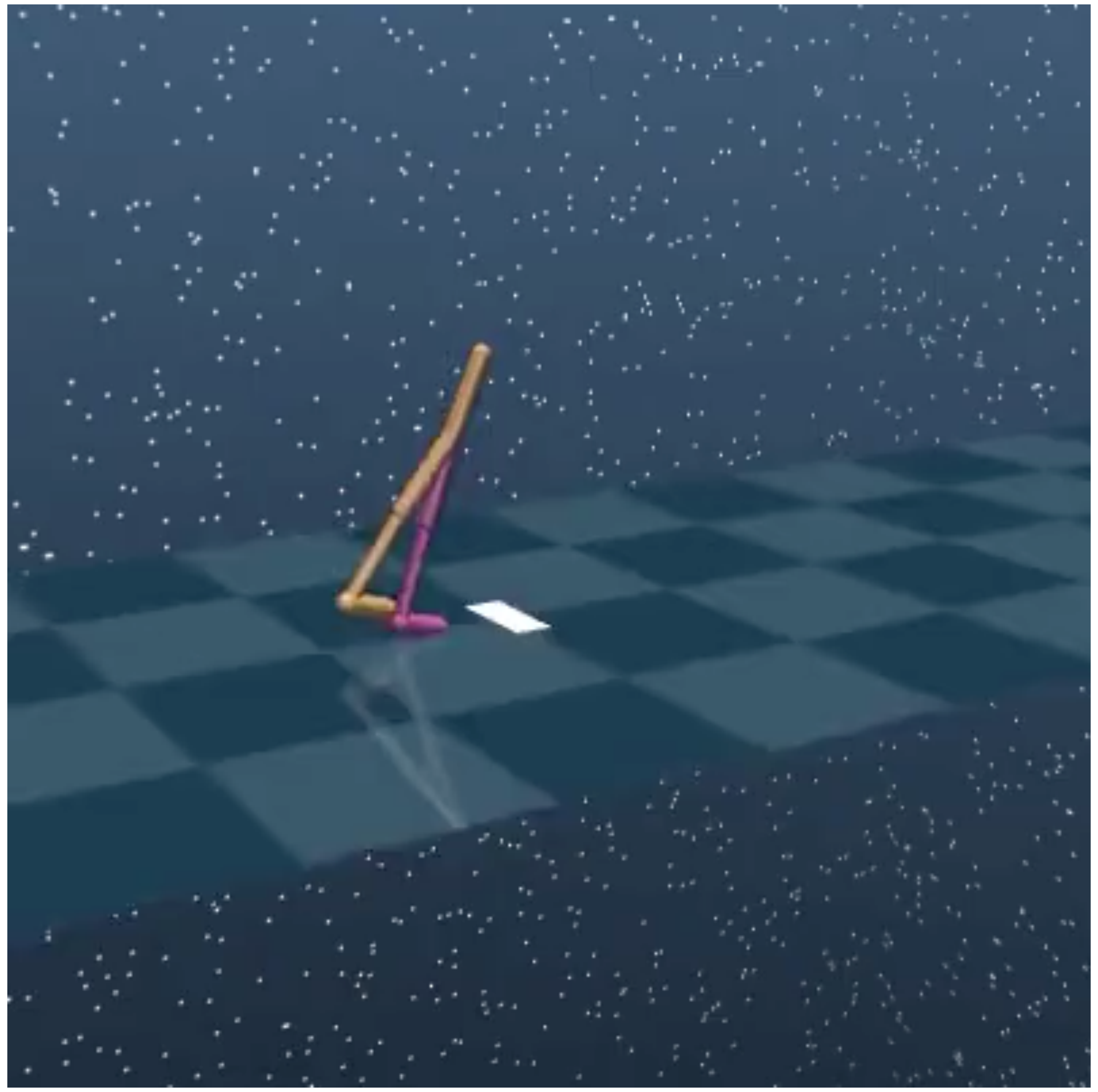


Walk Forward & Backward

Walk Forward

Transition

Walk Backward

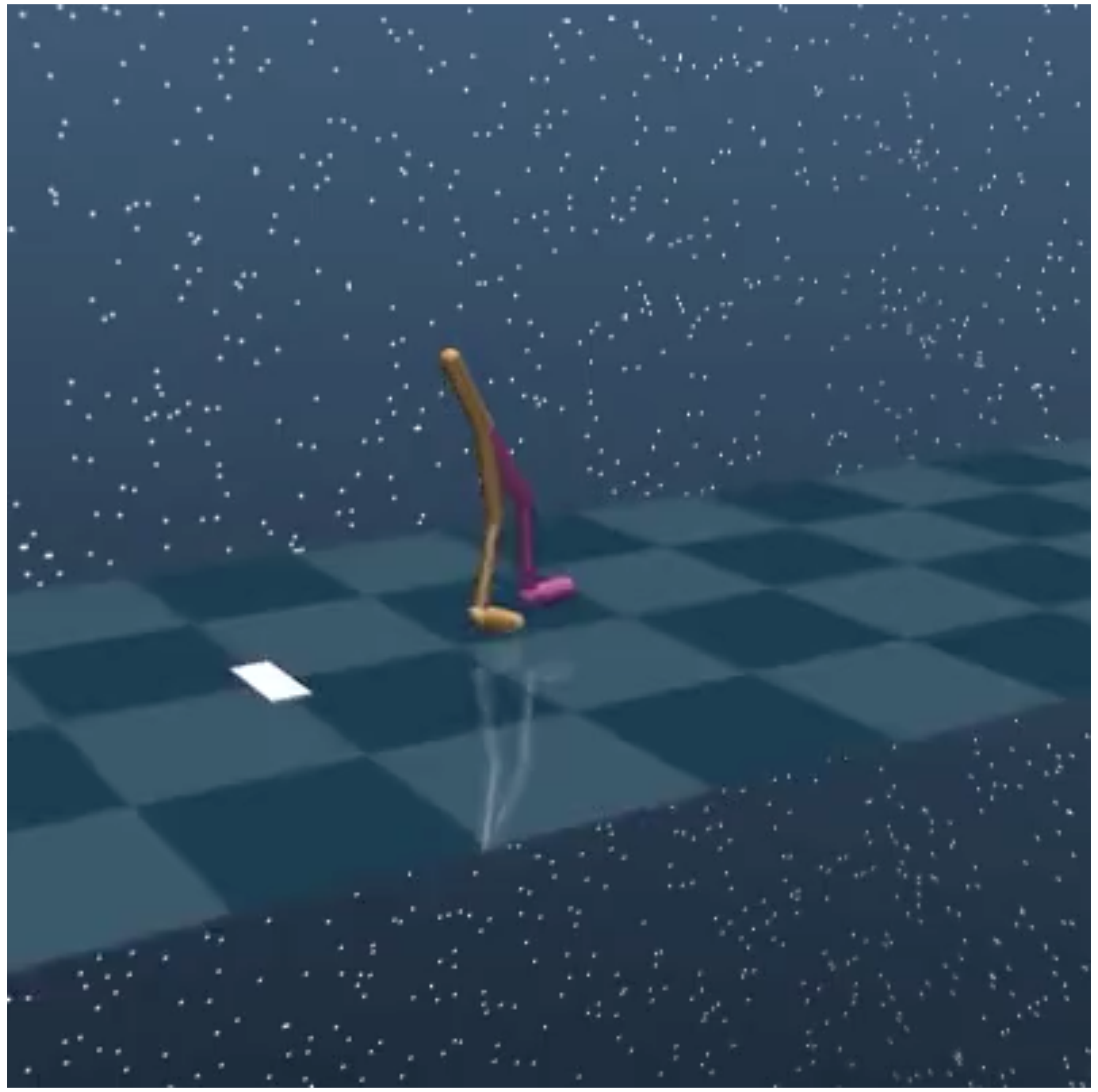


Walk Forward & Backward

Walk Forward

Transition

Walk Backward

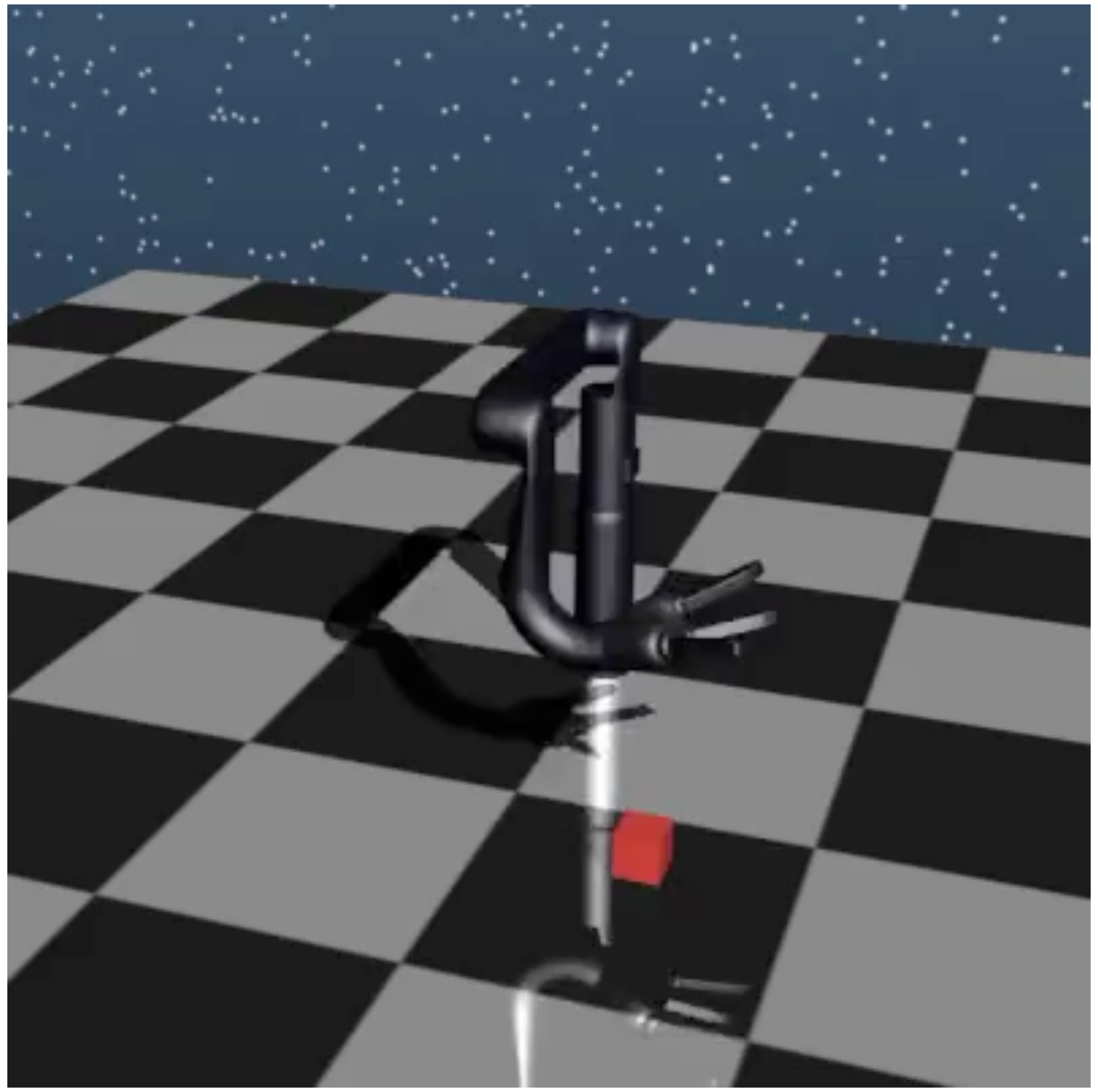


Repetitive Pick

Pick

Transition

Pick

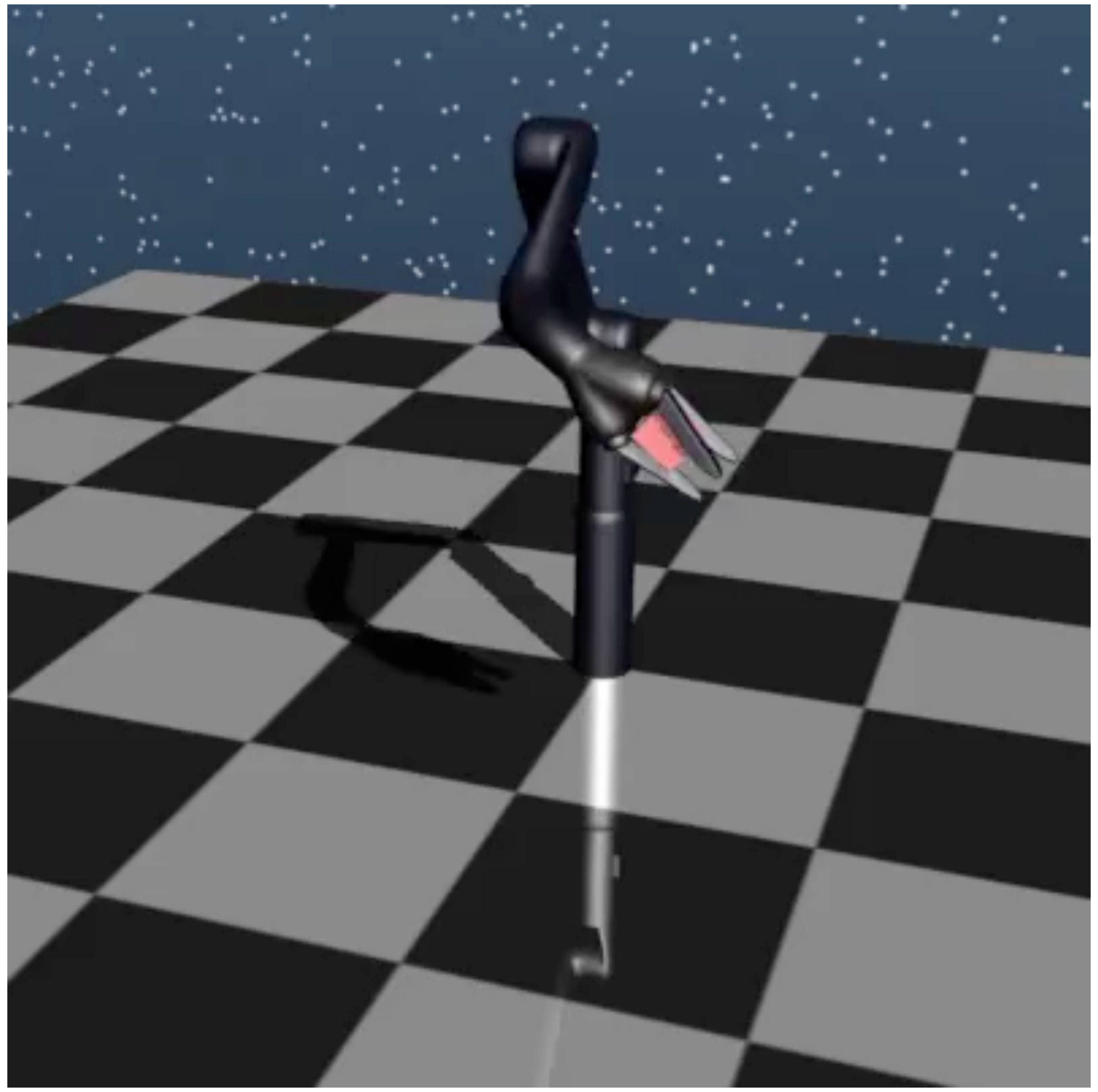


Repetitive Pick

Pick

Transition

Pick

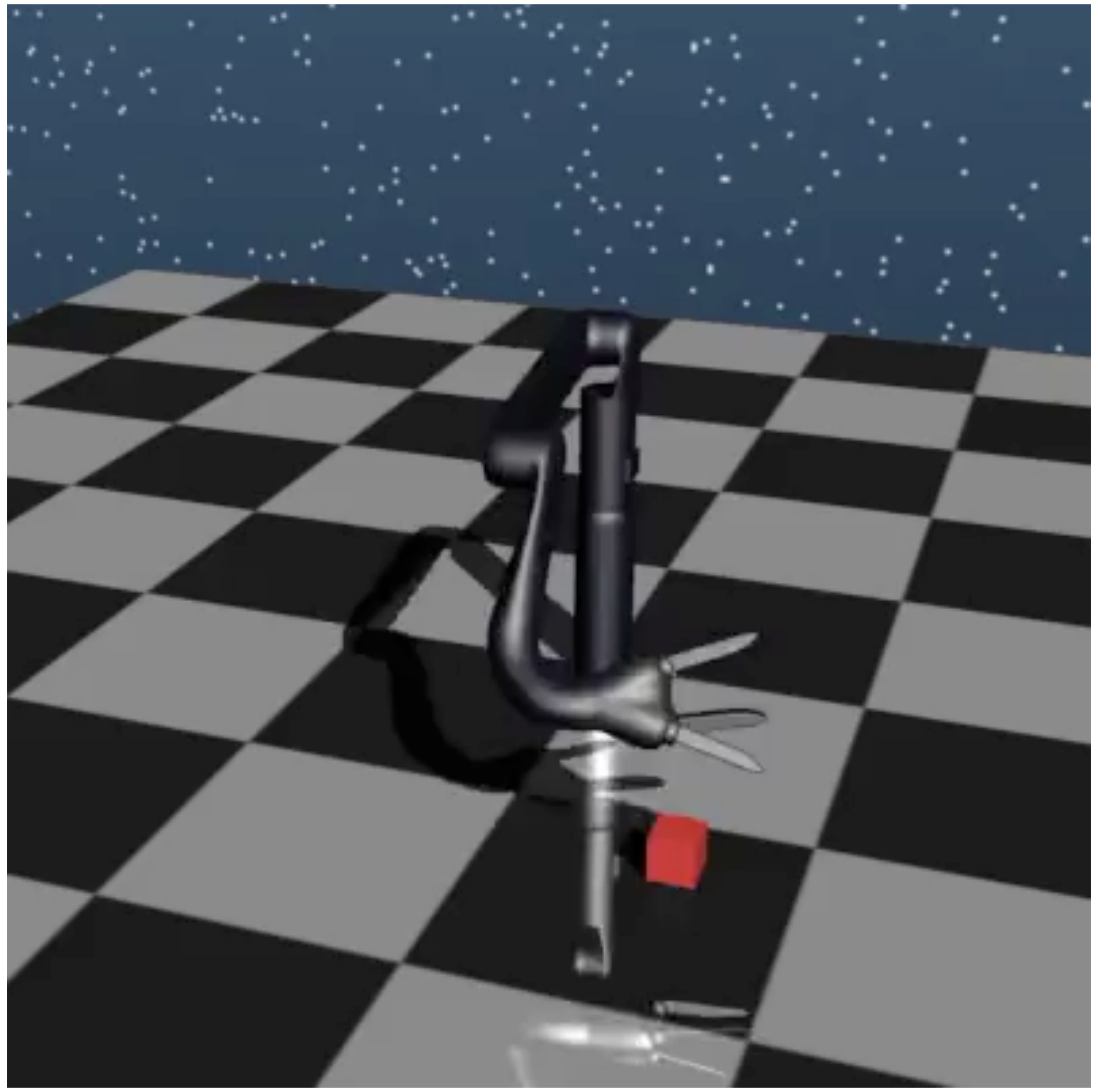


Repetitive Pick

Pick

Transition

Pick

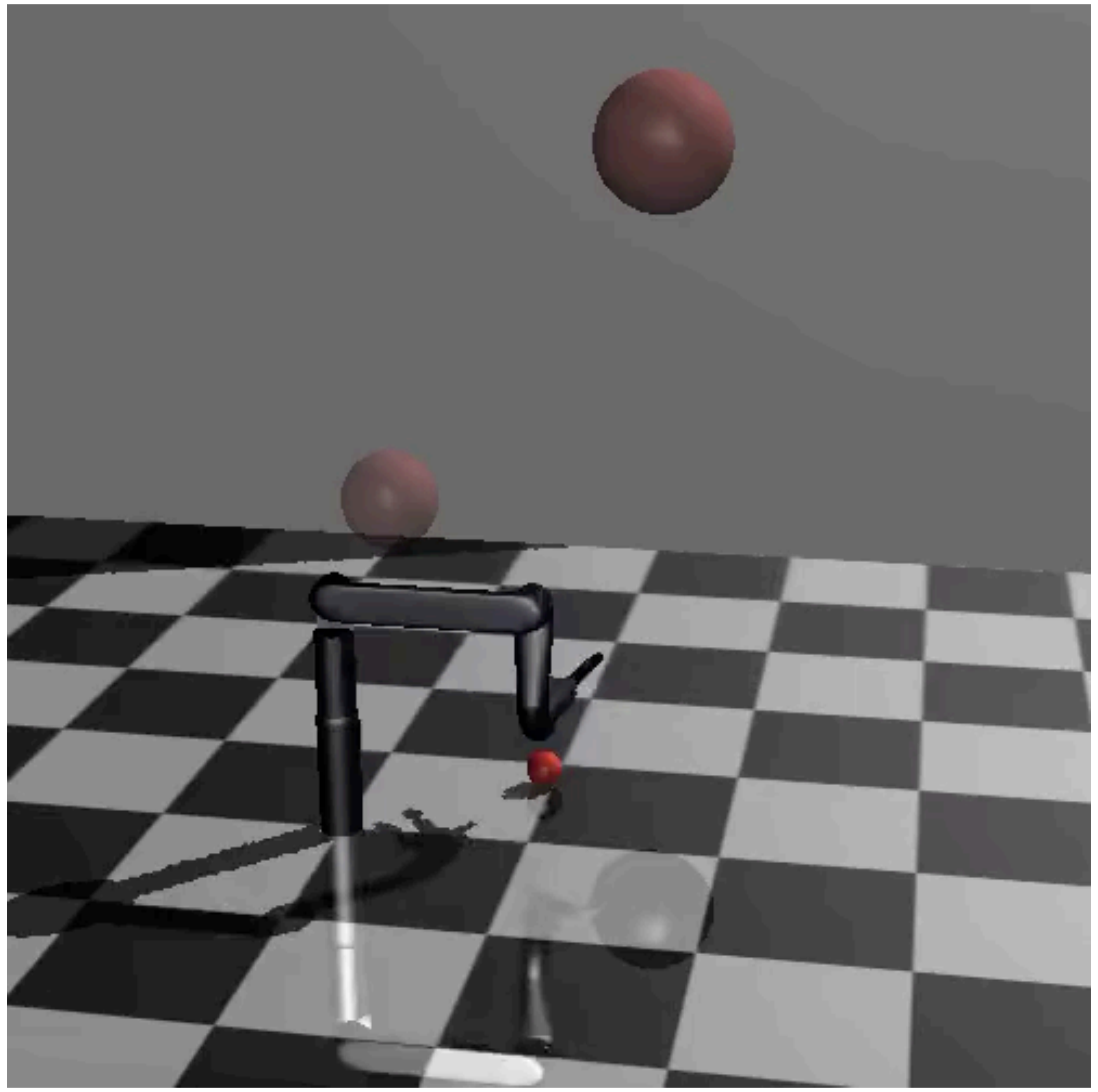


Toss & Hit

Toss

Transition

Hit

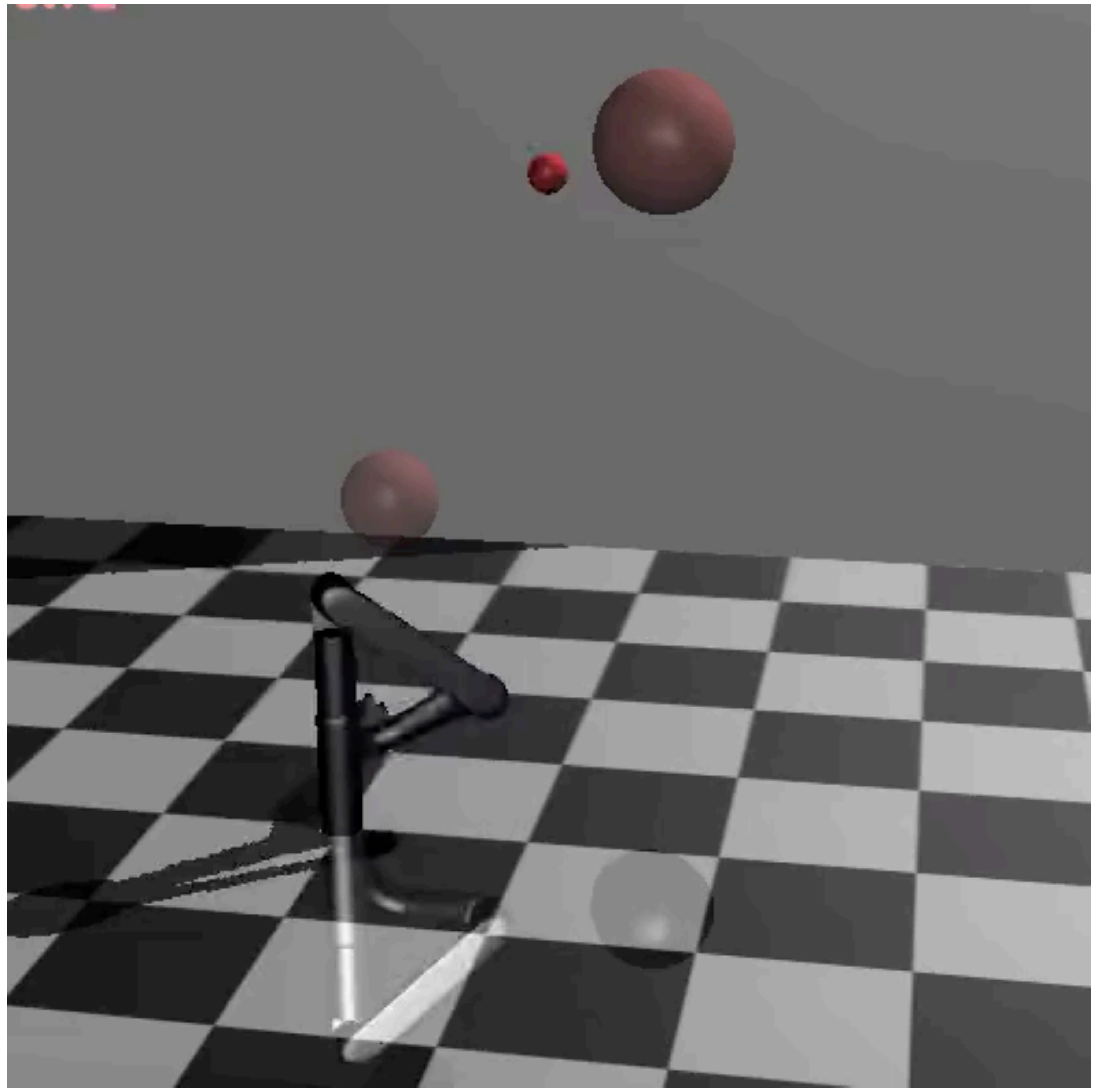


Toss & Hit

Toss

Transition

Hit

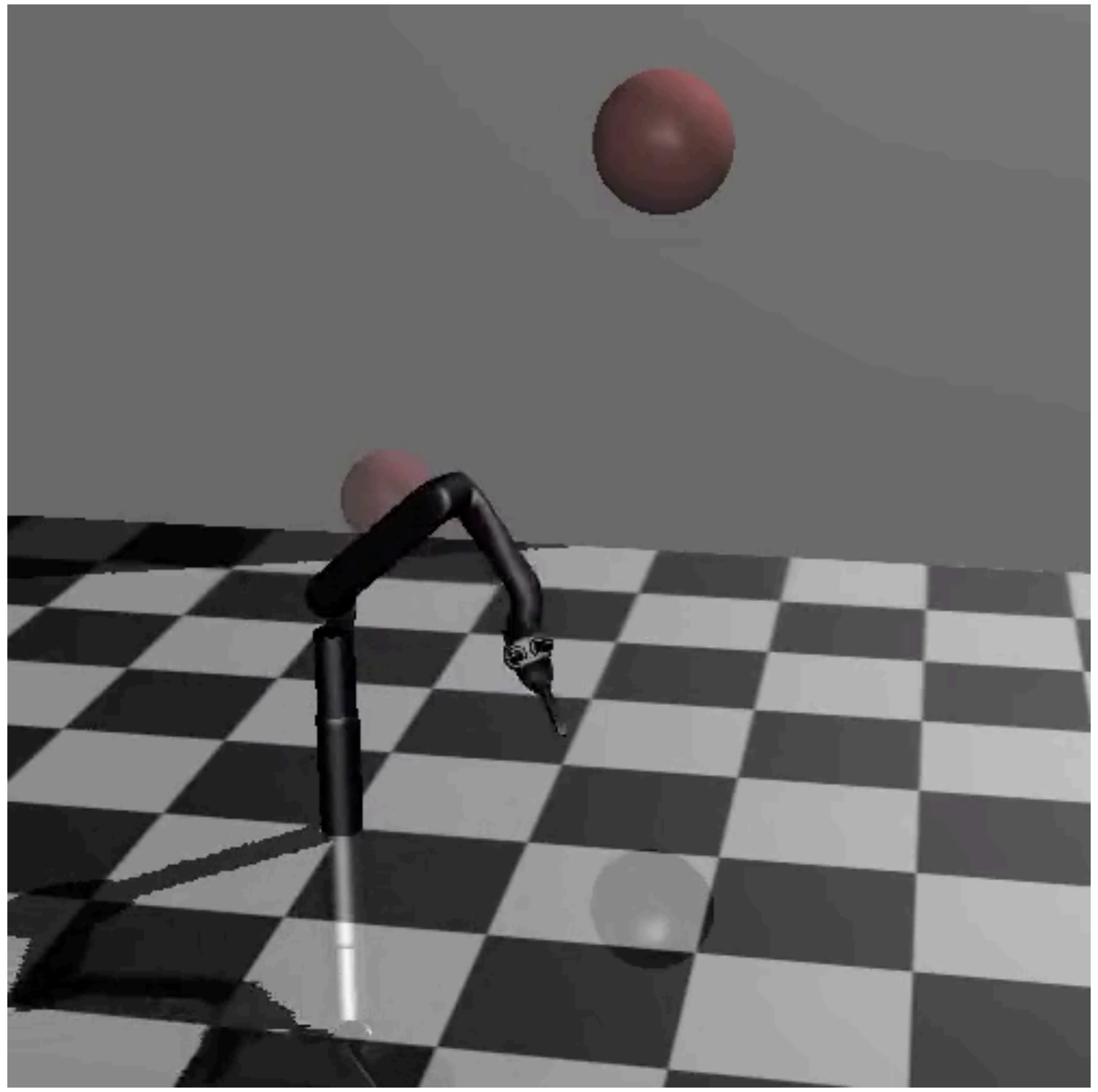


Toss & Hit

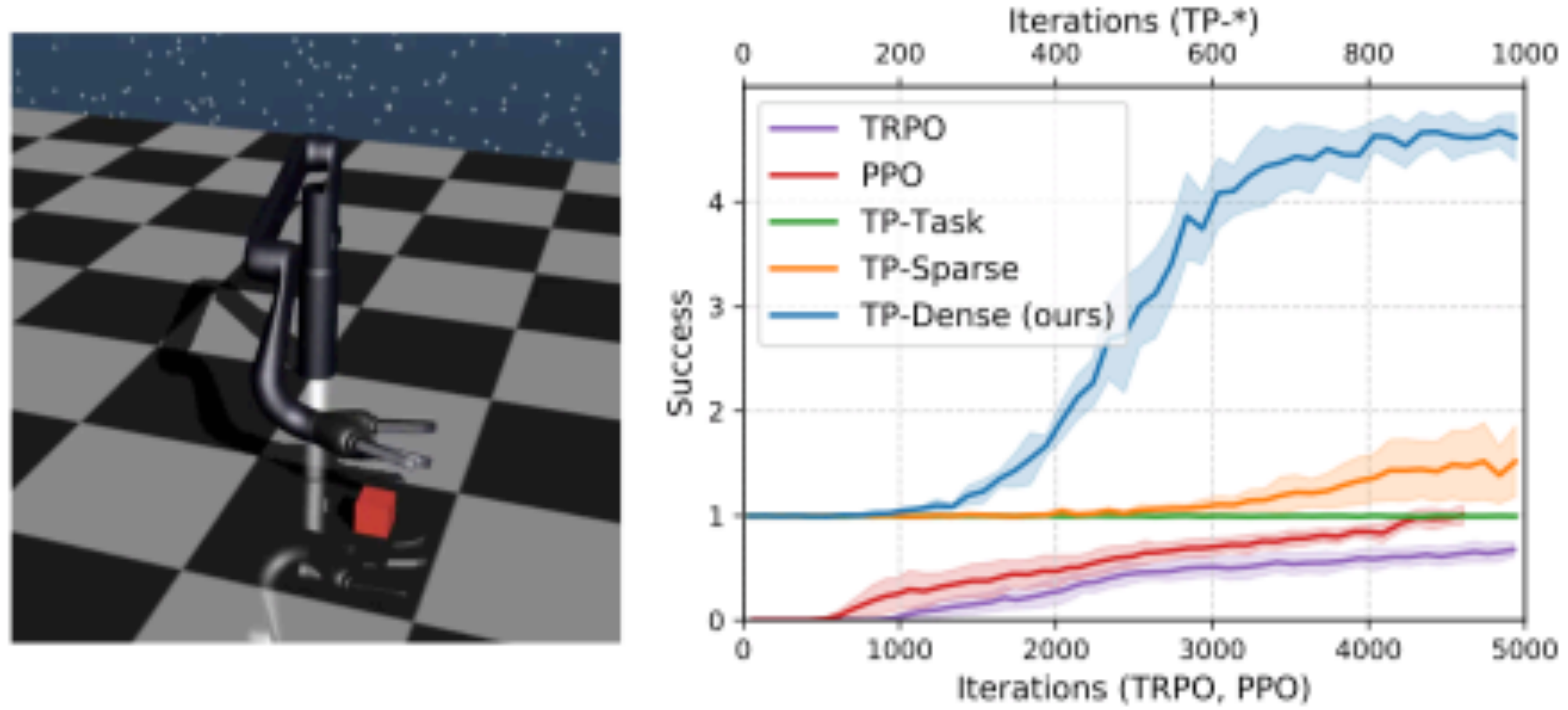
Toss

Transition

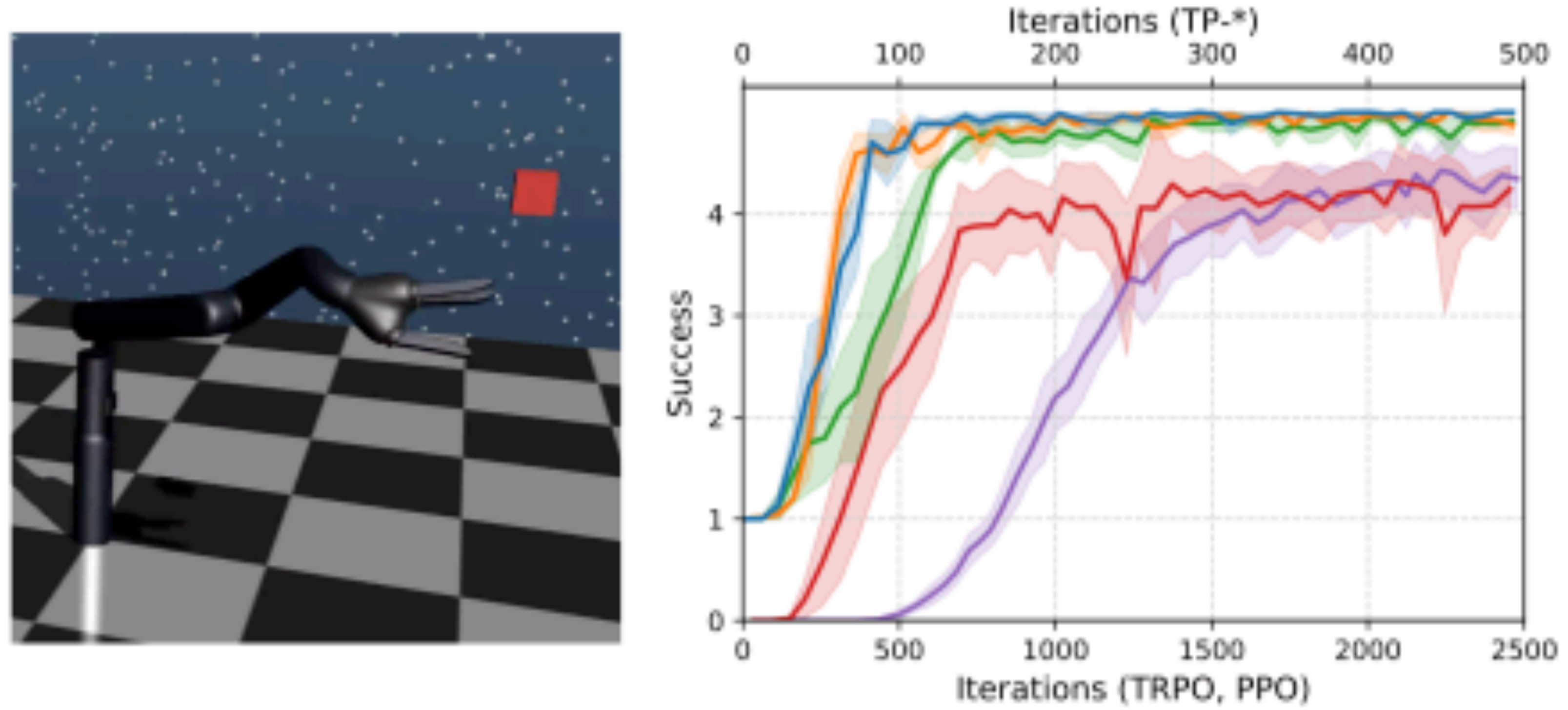
Hit



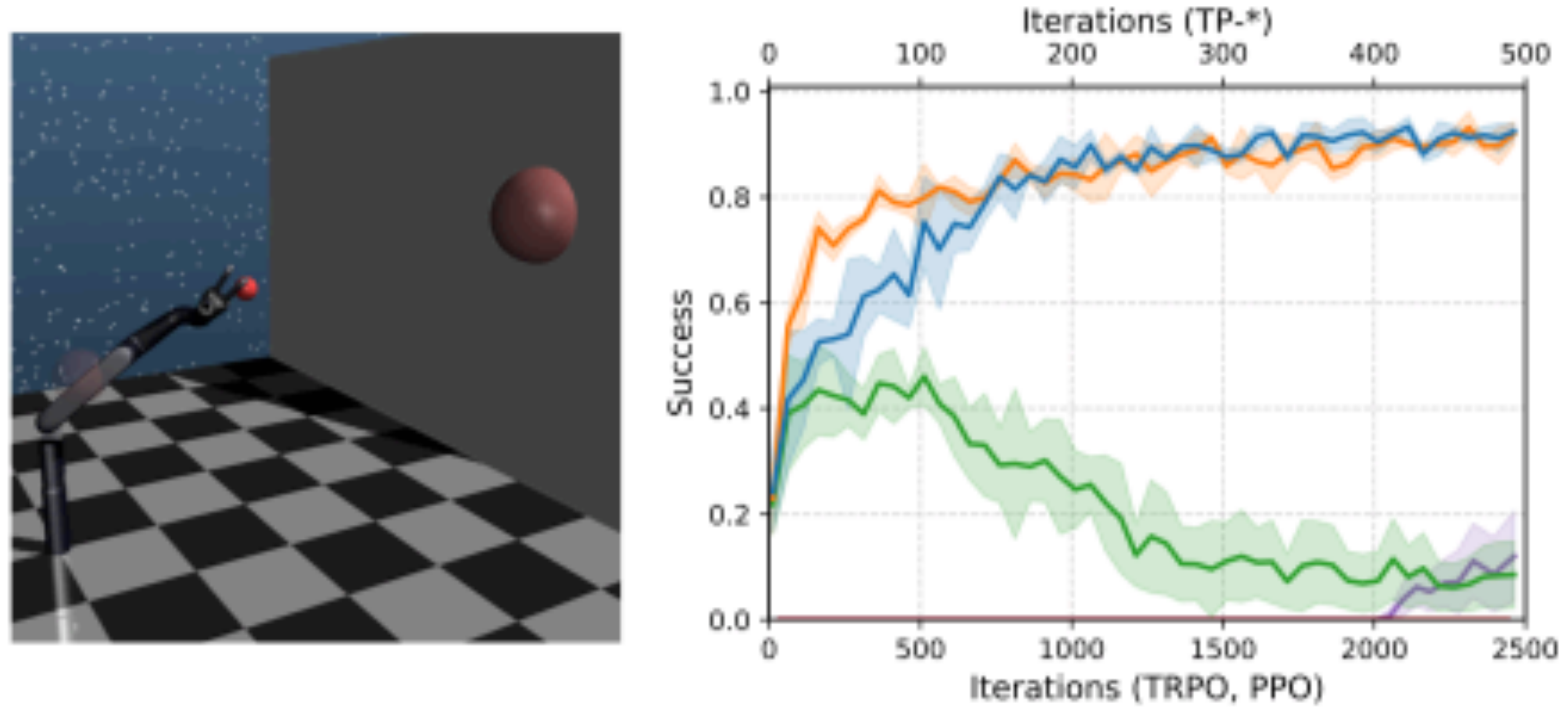
Quantitative Results



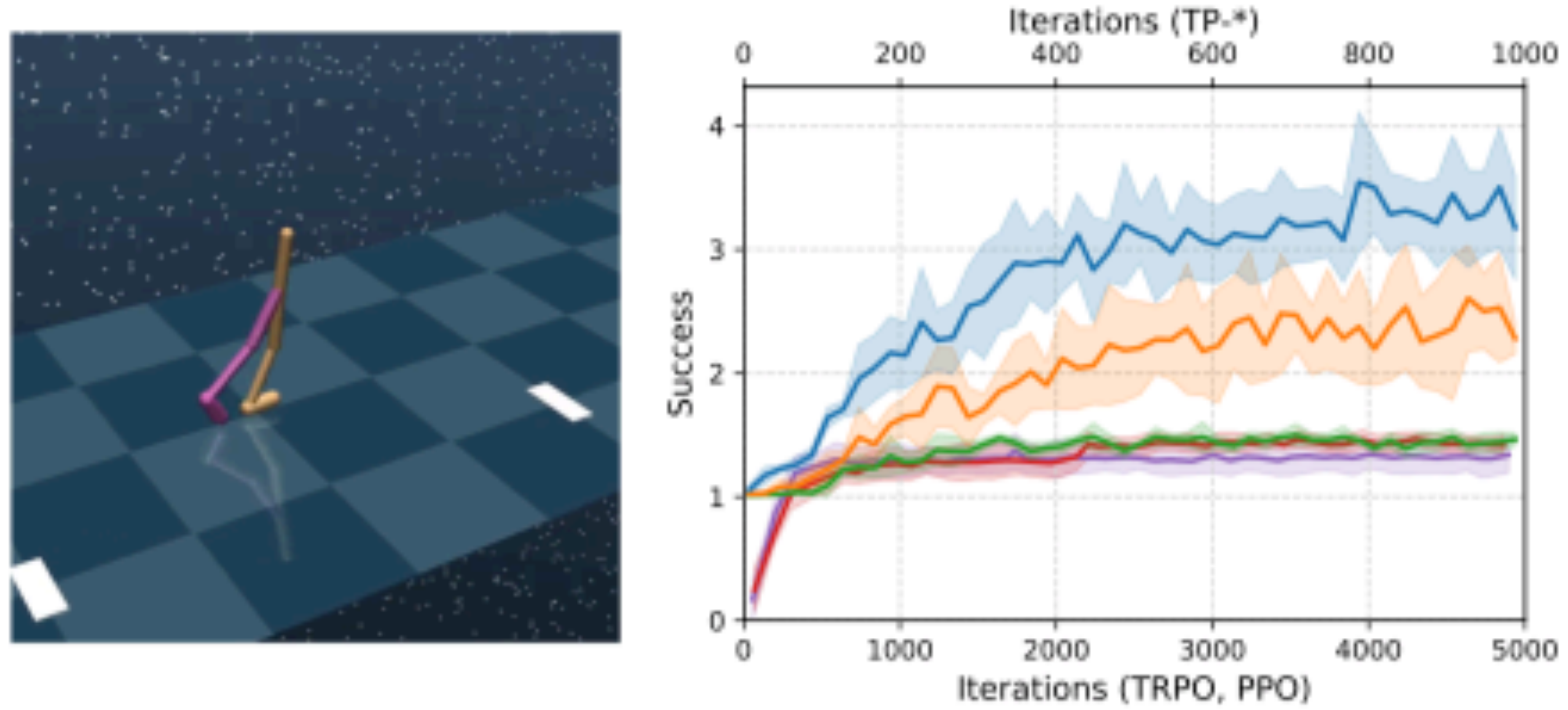
(a) Repetitive picking up



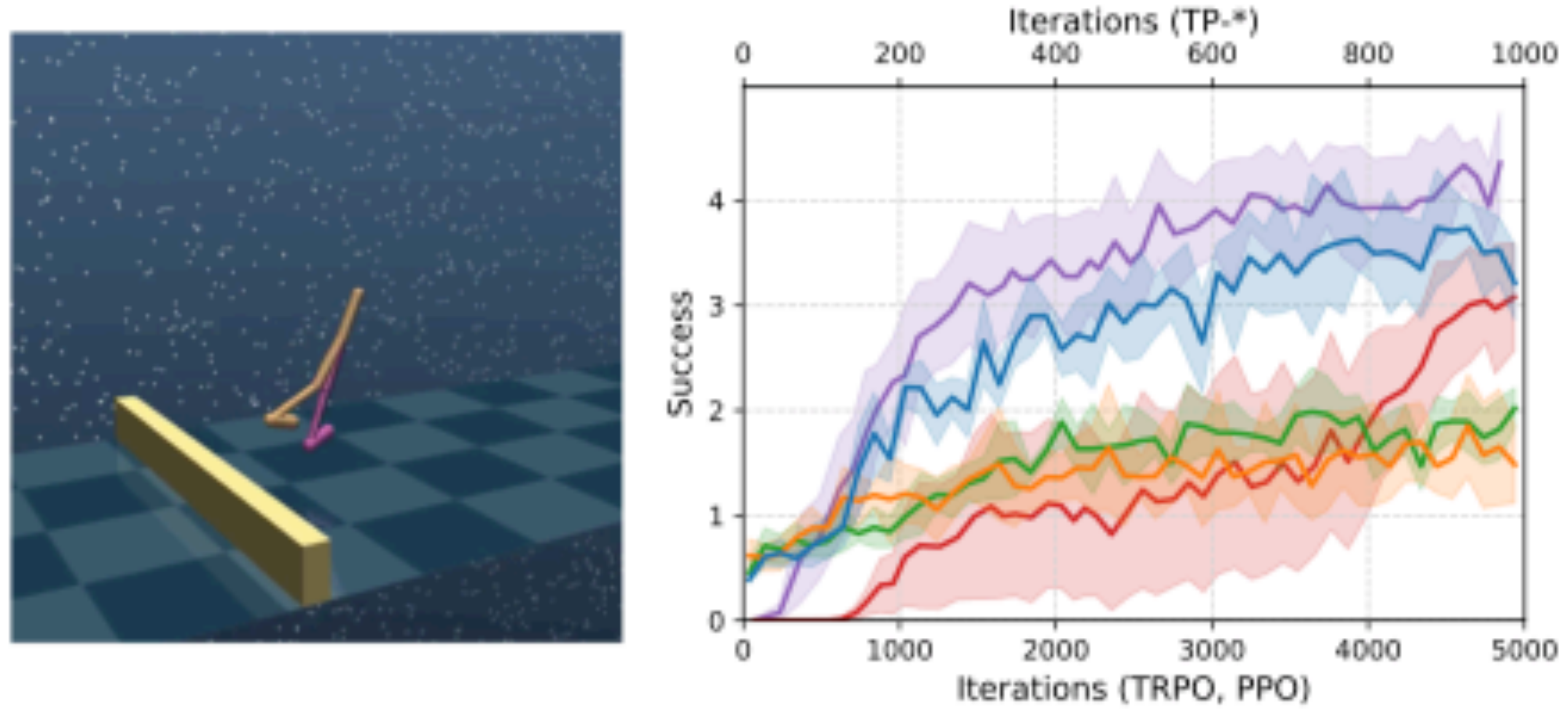
(b) Repetitive catching



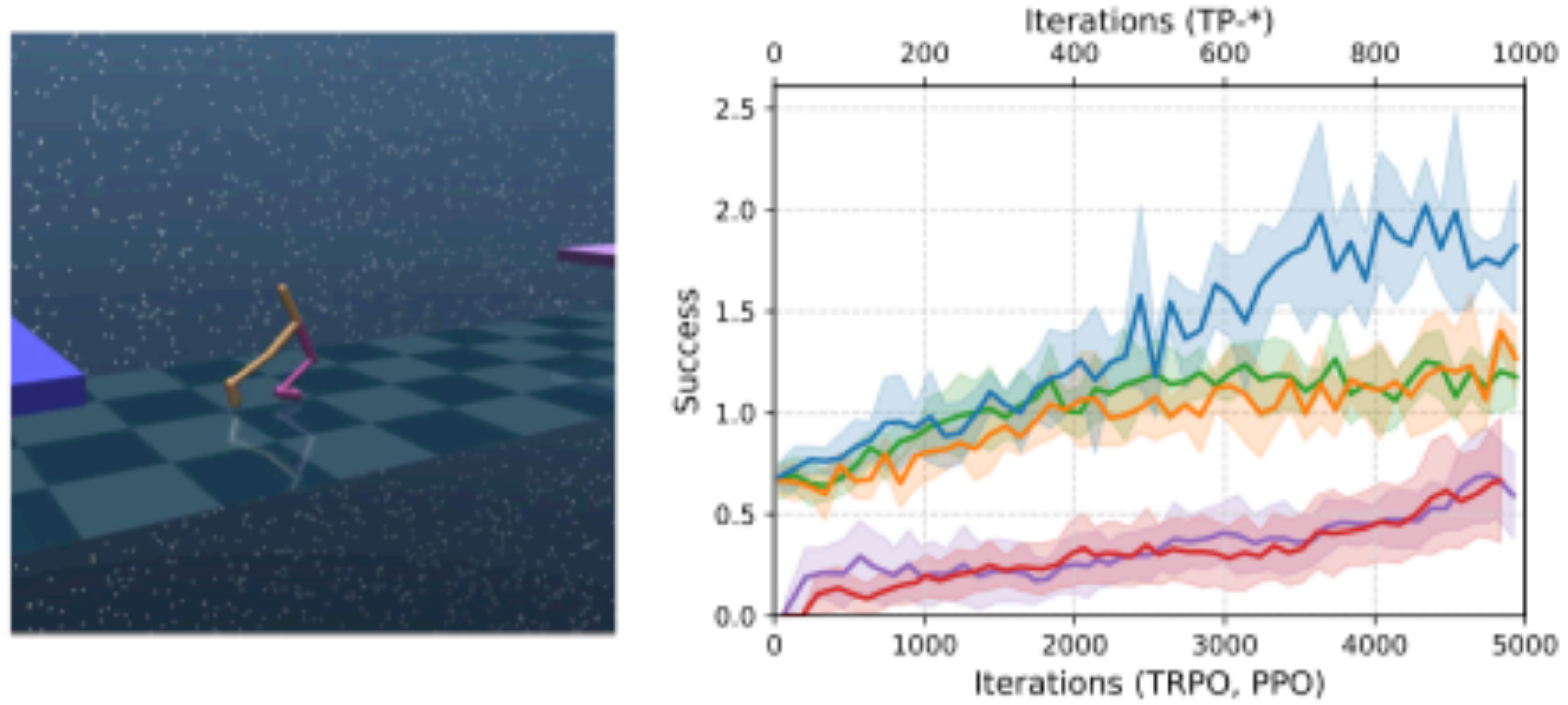
(c) Serve



(d) Patrol



(e) Hurdle



(f) Obstacle course

Quantitative Results

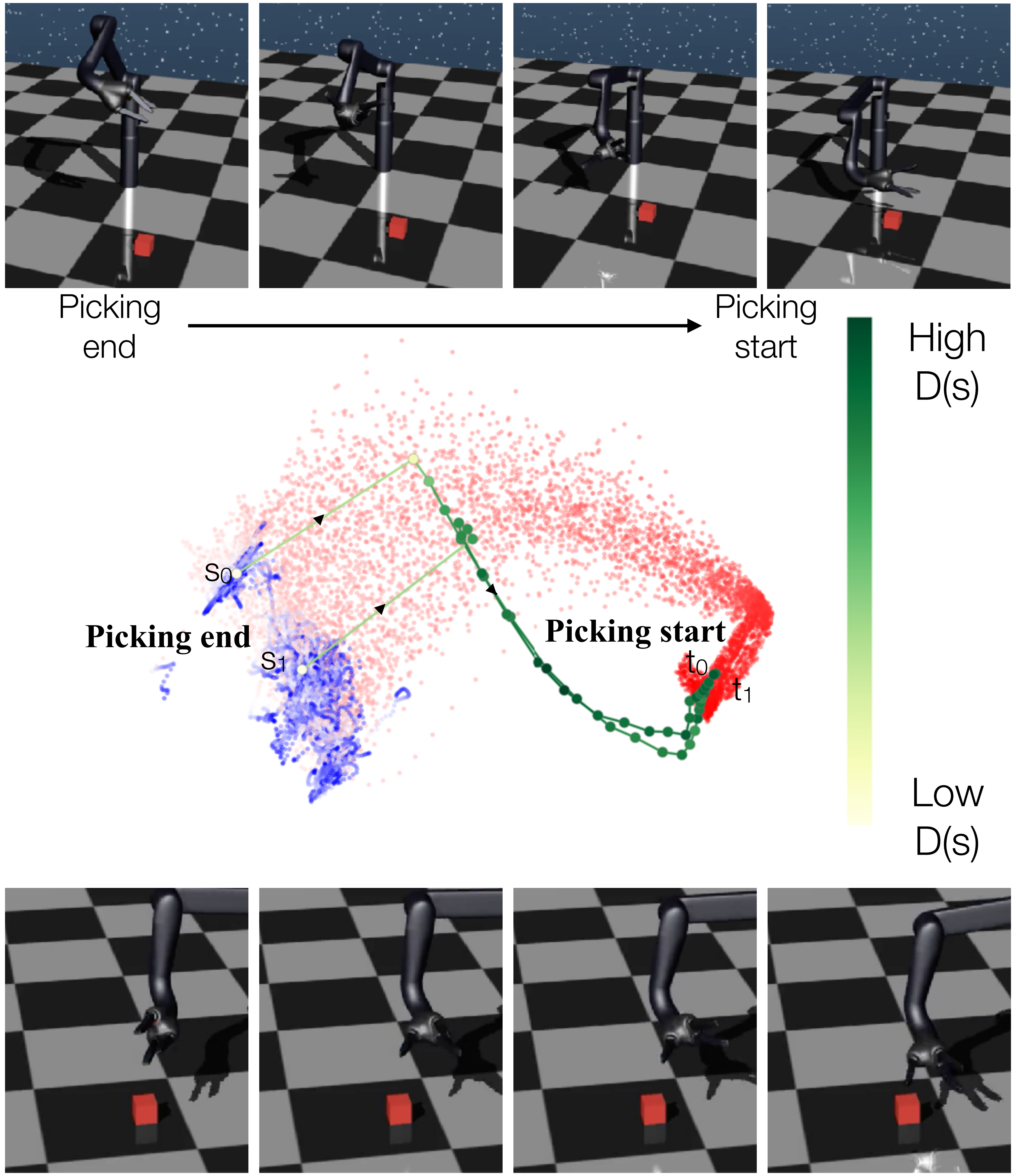
Manipulation

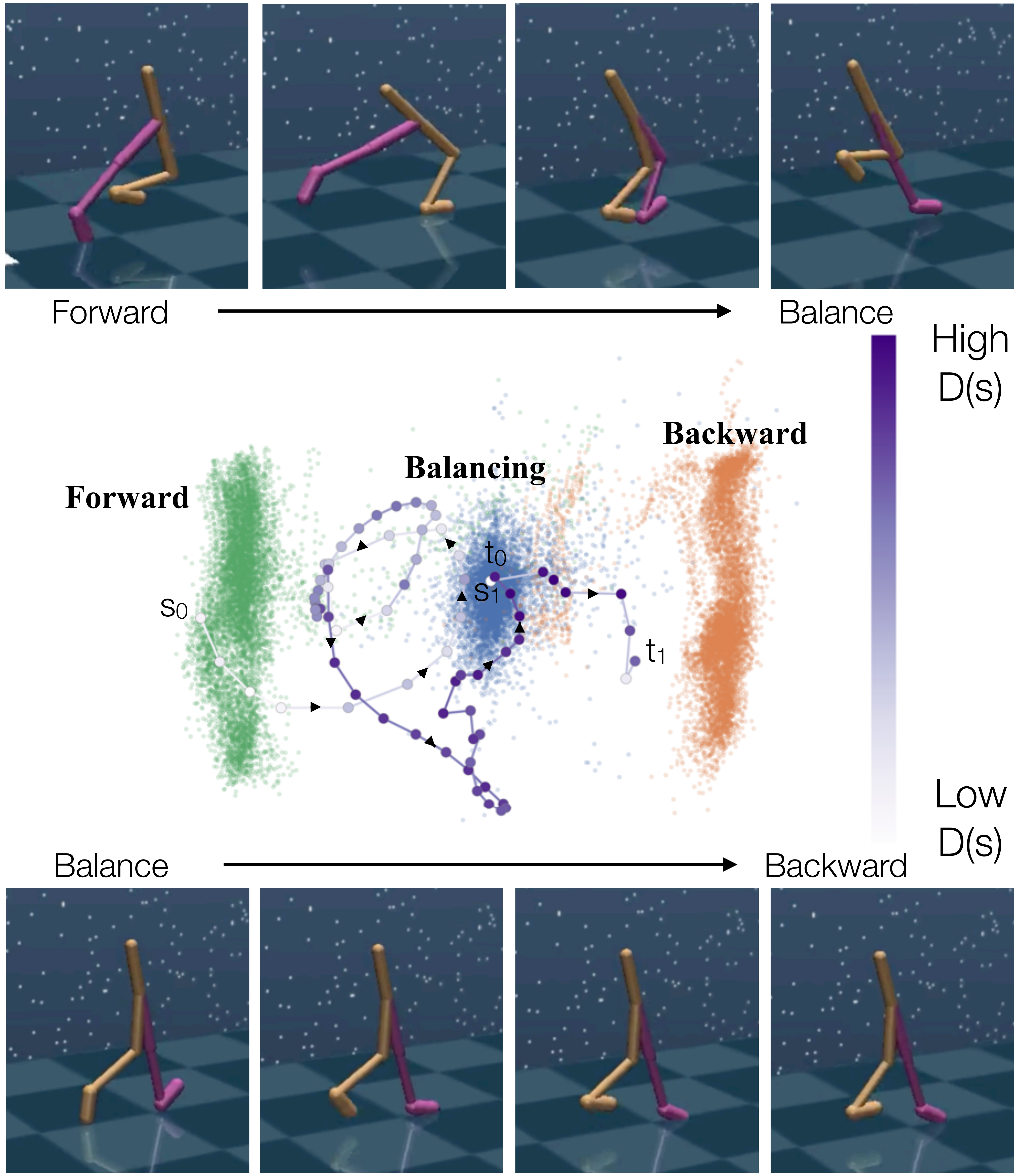
	Reward	Repetitive picking up	Repetitive catching	Serve
TRPO	dense	0.69 ± 0.46	4.54 ± 1.21	0.32 ± 0.47
PPO	dense	0.95 ± 0.53	4.26 ± 1.63	0.00 ± 0.00
Without TP	sparse	0.99 ± 0.08	1.00 ± 0.00	0.11 ± 0.32
TP-Task	sparse	0.99 ± 0.08	4.87 ± 0.58	0.05 ± 0.21
TP-Sparse	sparse	1.52 ± 1.12	4.88 ± 0.59	$\mathbf{0.92 \pm 0.27}$
TP-Dense (ours)	sparse	$\mathbf{4.84 \pm 0.63}$	$\mathbf{4.97 \pm 0.33}$	$\mathbf{0.92 \pm 0.27}$

Locomotion

	Reward	Patrol	Hurdle	Obstacle course
TRPO	dense	1.37 ± 0.52	$\mathbf{4.13 \pm 1.54}$	0.98 ± 1.09
PPO	dense	1.53 ± 0.53	2.87 ± 1.92	0.85 ± 1.07
Without TP	sparse	1.02 ± 0.14	0.49 ± 0.75	0.72 ± 0.72
TP-Task	sparse	1.69 ± 0.63	1.73 ± 1.28	1.08 ± 0.78
TP-Sparse	sparse	2.51 ± 1.26	1.47 ± 1.53	1.32 ± 0.99
TP-Dense (Ours)	sparse	$\mathbf{3.33 \pm 1.38}$	$\mathbf{3.14 \pm 1.69}$*	$\mathbf{1.90 \pm 1.45}$

Transition Trajectories





Summary

We propose to **reuse skills** to compose **complex, long-horizon tasks**.

Naive execution of skills fail since the skills never learned to connect.

Transition policies learn to smoothly connect skills.

Proximity predictors provide dense reward for efficient training of transition policies.