Combining (Reinforcement) Learning and Trajectory Optimization

The best of both worlds

Andrea Del Prete

UNIVERSITY OF TRENTO

Reinforcement Learning ~~VS~~ Trajectory Optimization
WITH?

$$\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) \\
\text{subject to} \quad & x_{i+1} = f(x_i, u_i) \qquad i = 0 \ldots N-1 \\
& x_{i+1} \in \mathcal{X}, u_i \in \mathcal{U} \qquad i = 0 \ldots N-1
\end{aligned}$$

Reinforcement Learning

+ Less prone to poor local minima
+ Derivative free (easy to implement)
+ Fast online policy evaluation
+ Typically stochastic
− Poor data efficiency (slow training)
− Does not account for constraints

Trajectory Optimization

+ Data efficient (fast)
+ Exploits dynamics derivatives
+ Accounts for constraints
− Can get stuck in poor local minima
− Online computational burden
− Typically deterministic

RL and TO - What's the difference?

Model-free VS model-based?

	How is it optimized?		
What is optimized?		**Derivative Based**	**Derivative Free**
	Trajectory	Trajectory Optimization	Model Predictive Path Integral Control
	Policy	Model-based policy optimization	Reinforcement Learning

Model-based Policy Optimization

Model-based Policy Optimization

Discussion

$$\min_{\theta} \frac{1}{N} \sum_{i} J(X^i, U^i)$$

$$\text{s.t.} \quad u_t^i = \pi_\theta(x_t^i)$$

Policy

$$x_{t+i}^i = f(x_t^i, u_t^i)$$

$$\forall \; i \in [0, N-1], \; t \in [0, T-1]$$

Number of trajectories

Horizon length

- Optimize policy that gives best average performance over horizon T for a set of N initial conditions
- Exploit dynamics derivatives
- Efficient policy evaluation at deployment
- Can account for uncertainties via domain randomization
- Less efficient than TO due to coupling between time steps introduced by θ
- Local minima due to optimizing over a prediction horizon

High-Level Architectures for combining RL and TO

Overview

Combining RL and TO

Vast literature over last decade

Different assumptions

- Dynamics:
  - known or unknown?
  - deterministic or stochastic?
- Policy:
  - deterministic or stochastic?
  - state or sensor-feedback?

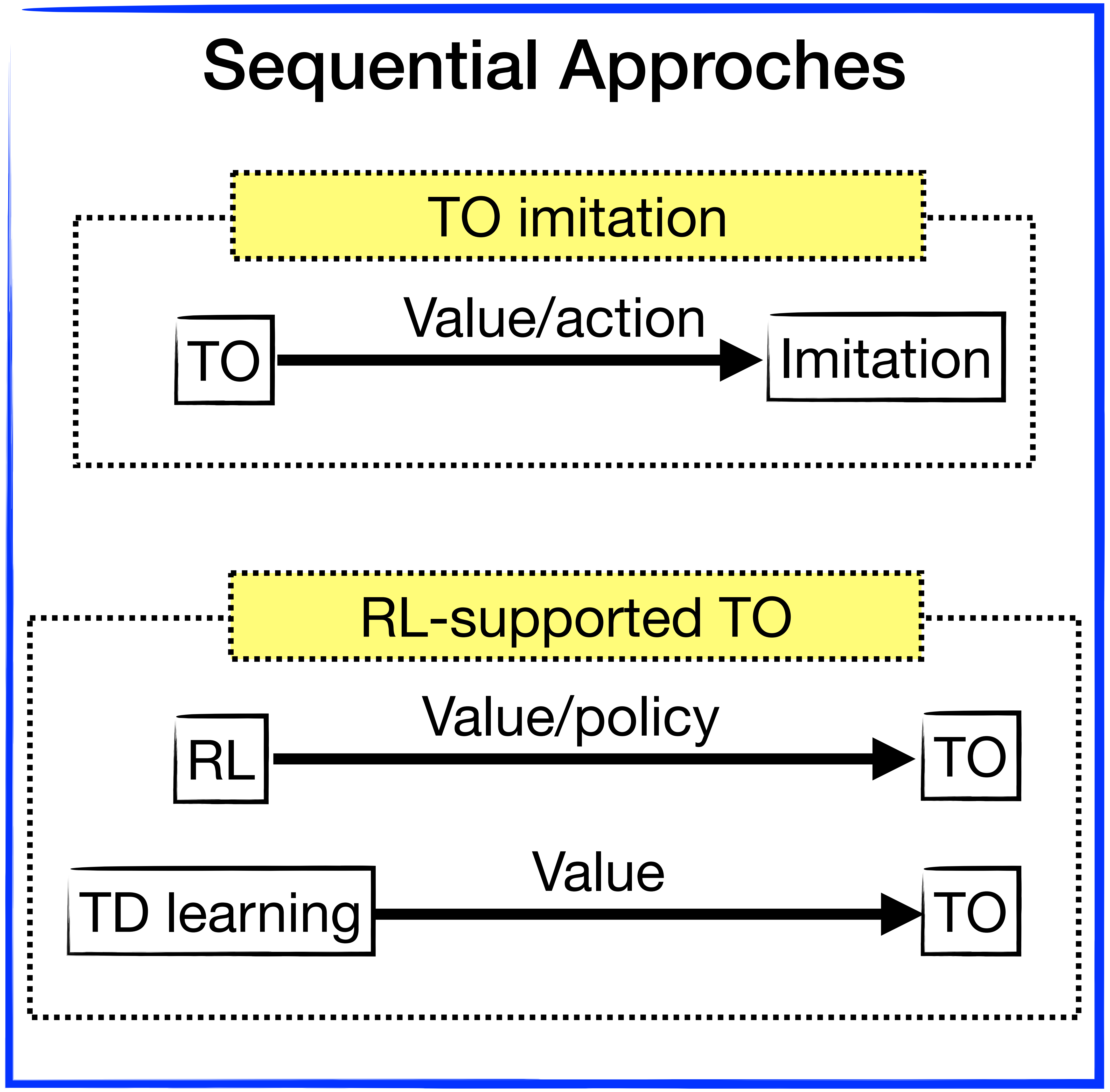
Different objectives

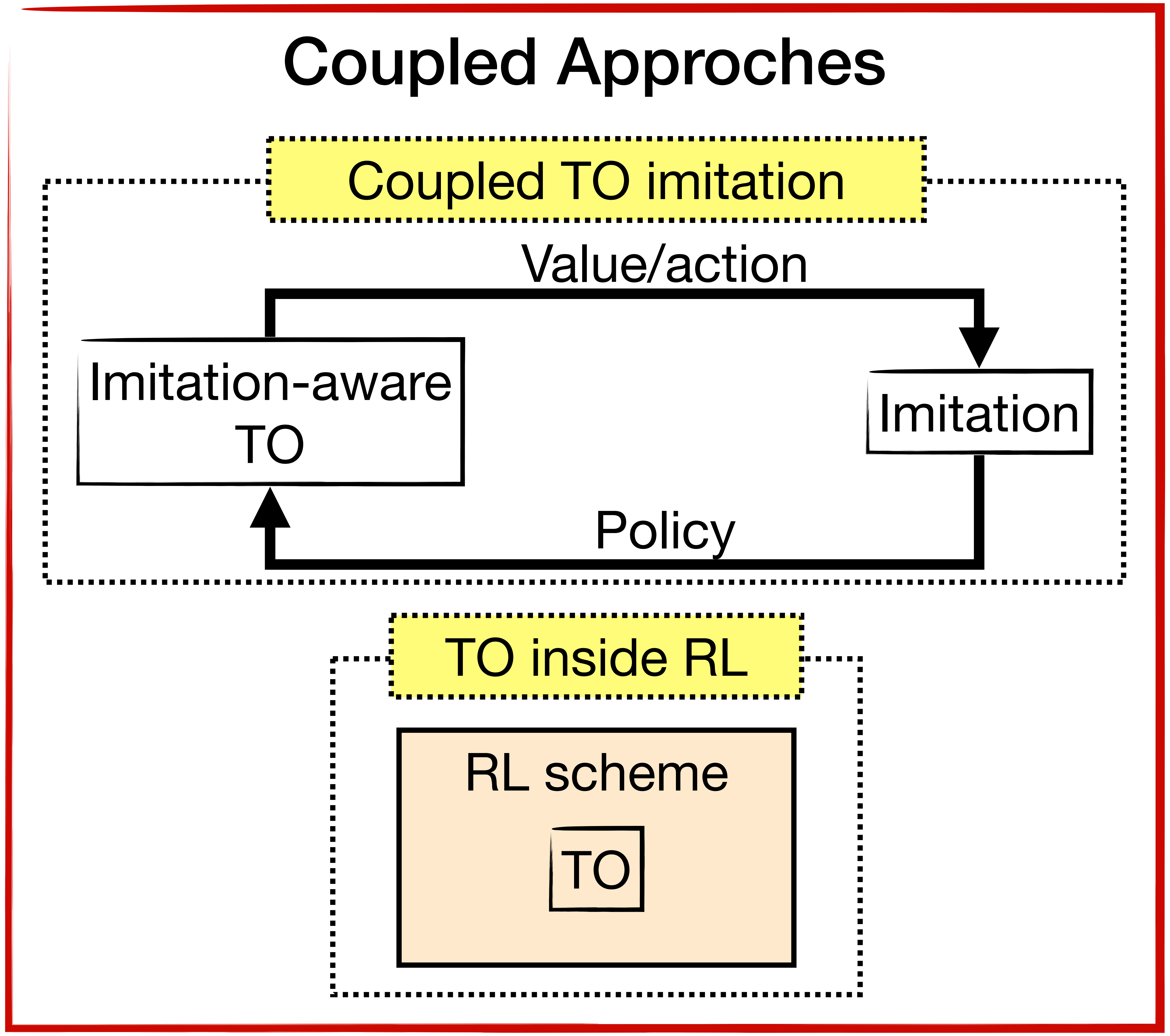
- Speed-up RL training
- Speed-up TO computation (for MPC)
- Improve RL policy via local refinement or enforcing constraints
- Help TO find global optimum or satisfy constraints
- Exploit sensor data

Reiter, Hoffmann, Reinhardt, Messerer, Baumgärtner, Sawant, Boedecker, Diehl, Gros (2025)
"Synthesis of MPC and RL: Survey and Classification"

Architectures Combining RL and TO

Overview





Sequential Approach: TO Imitation

Discussion

TO imitation

Action-based imitation

$$\min_{\theta} ||\pi_\theta(x_i) - a_i||^2$$

Value-based imitation

$$\min_{\theta} l(x_i, \pi_\theta(x_i)) + V_{i+1}(f(x_i, \pi_\theta(x_i)))$$

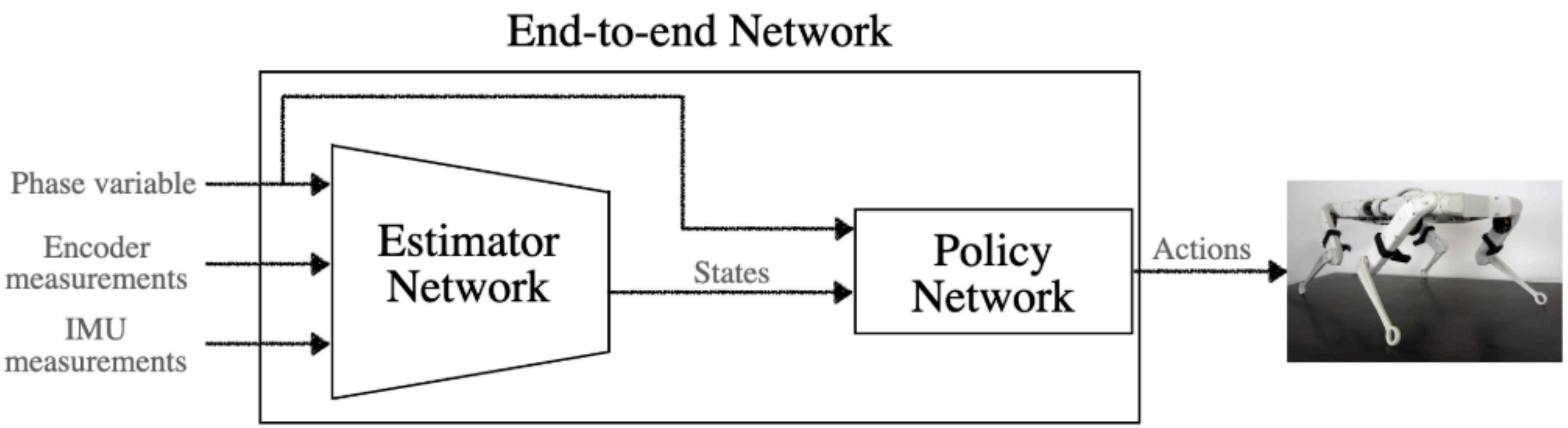
- Typically assume deterministic dynamics & policy (stochastic policy help dealing with multi-modality)
- Value-based imitation makes policy aware of consequences of errors
- **Objectives**:
  - get rid of computational burden of TO
  - exploit sensor data
- **Limitation**: cannot help TO find good solutions or satisfy constraints

TO Imitation

Examples

Learning Locomotion Skills from MPC in Sensor Space

Khadiv, Meduri, Zhu, Righetti, Schölkopf (L4DC 2023)



- Behavior cloning from MPC data
- Learn map from sensors to actions
- No need to address distribution mismatch
- Learning 2 maps (sensors to state + state to actions) outperformed direct map from sensors to actions

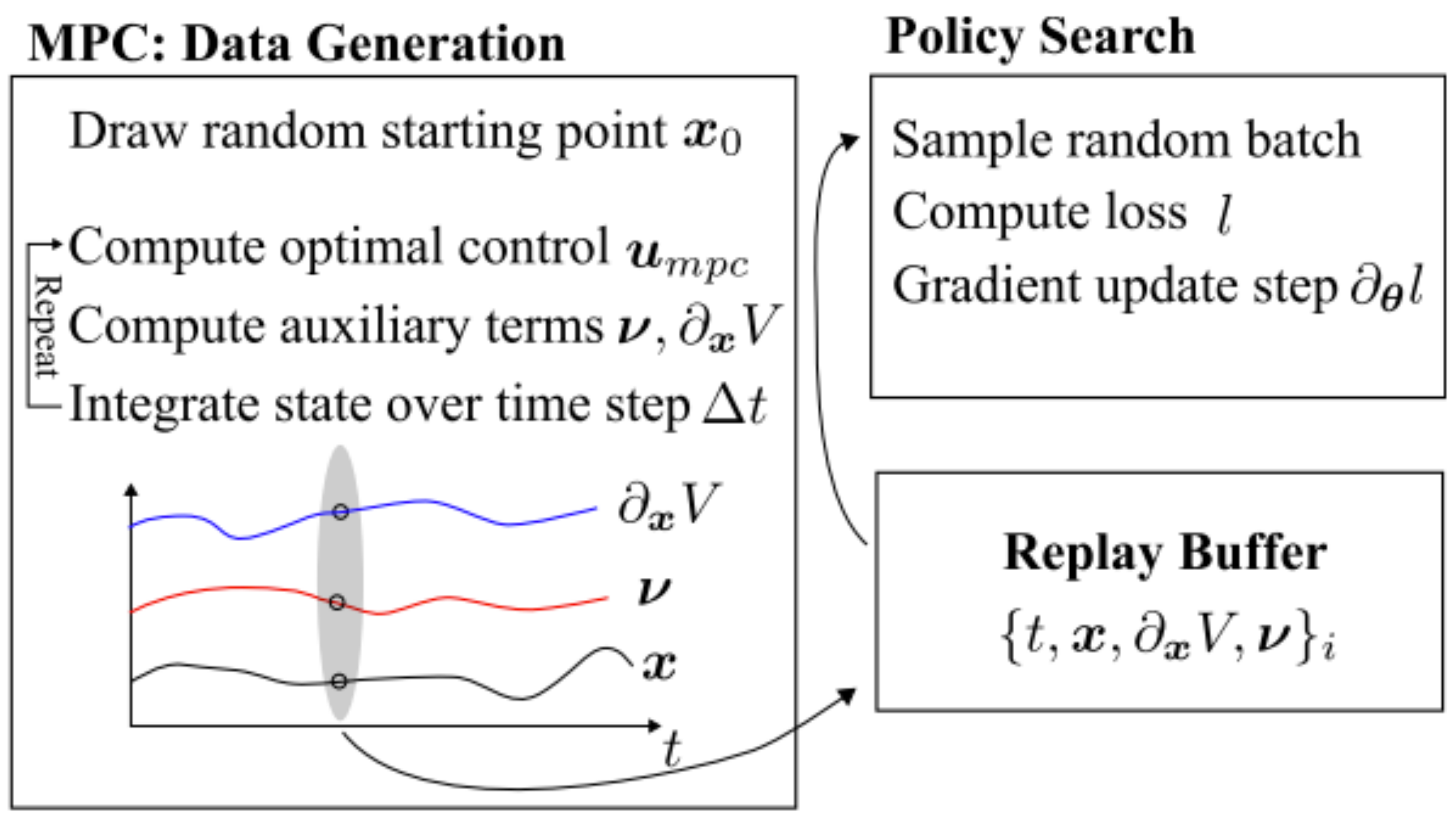
TO imitation

MPC-Net: A First Principles Guided Policy Search

Carius, Farshidian, Hutter (RAL 2020)

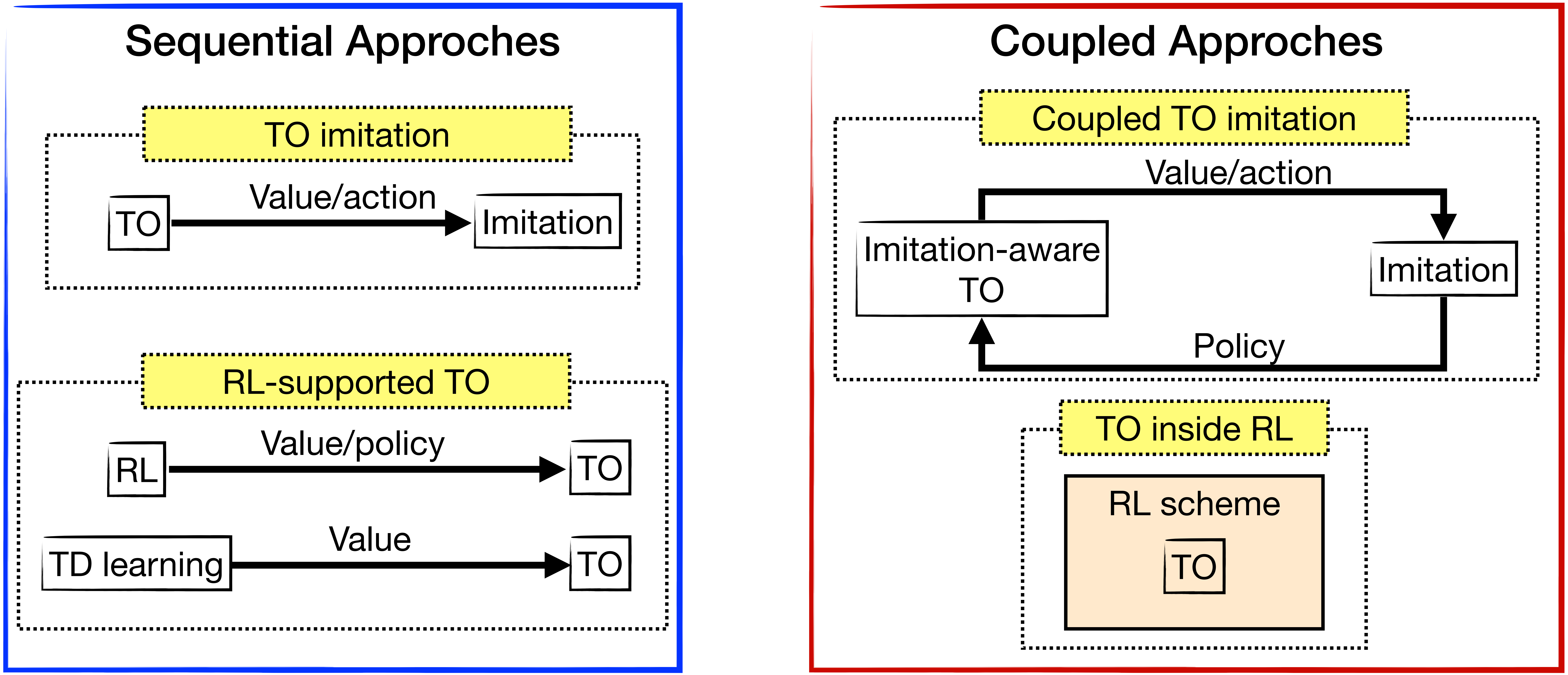
TO imitation

- Generate trajectories with MPC from random initial states and store $(t, x, \partial_x V)$ in buffer
- Define Hamiltonian as:
$$H(t, x, u) = l(t, x, u) + \partial_x V(t, x)^\top f(t, x, u)$$
- Sample from buffer and optimize neural policy $\pi_\theta(t, x)$ as:
$$\min_\theta H(t, x, \pi_\theta(t, x))$$
- Address distribution mismatch by collecting extra samples around optimal trajectories with behavior policy:
$$\pi(t, x) = (1 - \alpha)\pi_{MPC} + \alpha\pi_\theta(t, x)$$
- α goes from 0 to 1 throughout the algorithm iterations



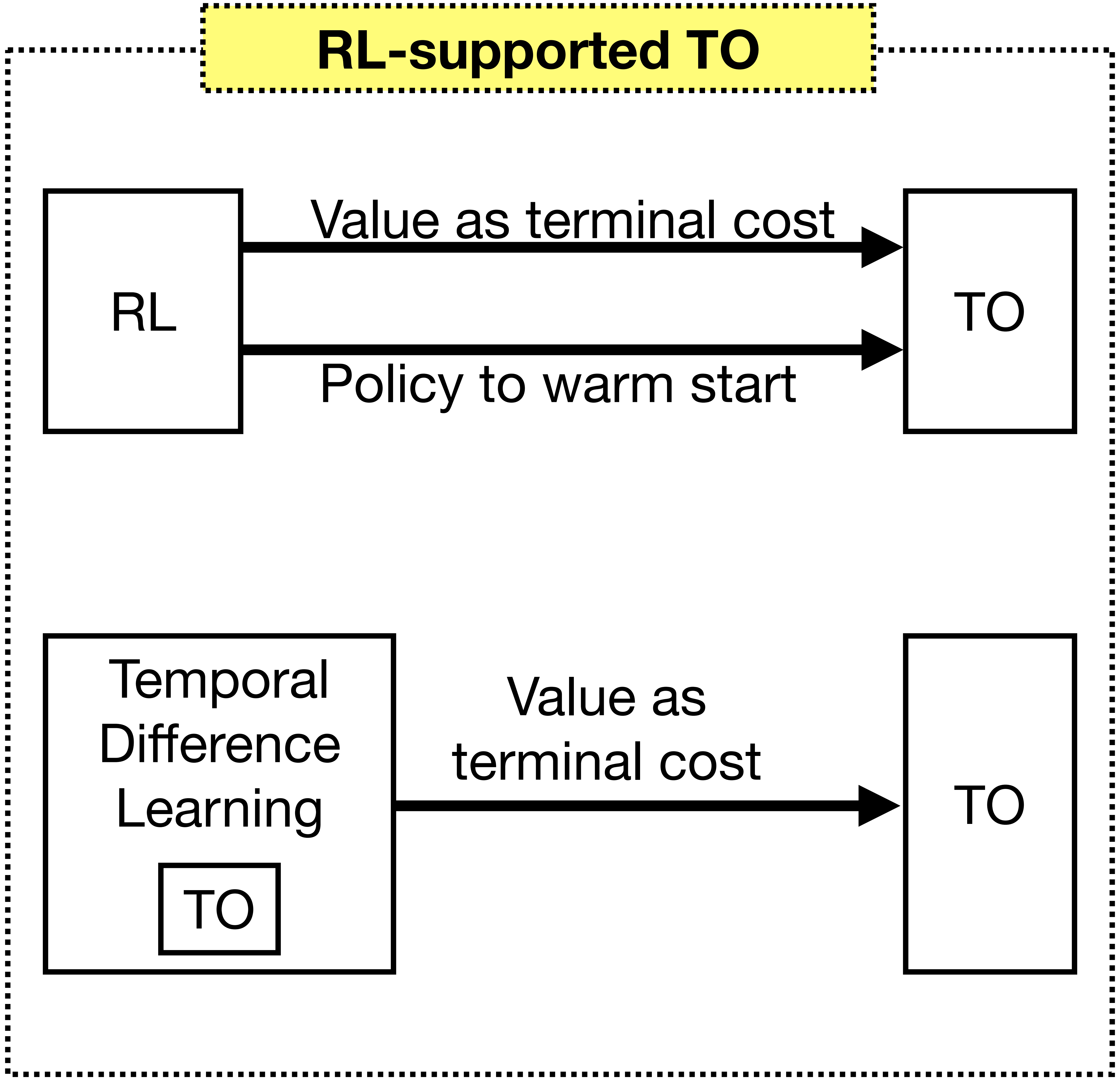
Architectures Combining RL and TO

Overview



Sequential Approach: RL-supported TO

Discussion



- Typically assumes known deterministic dynamics & policy
- **Objectives:**
  - Speed-up and guide TO through policy-based warm-start
  - Guide TO towards better solutions using Value function
- **Limitations:**
  - Does not speed up RL training
  - Value computed by TD is as good as TO

RL-supported TO

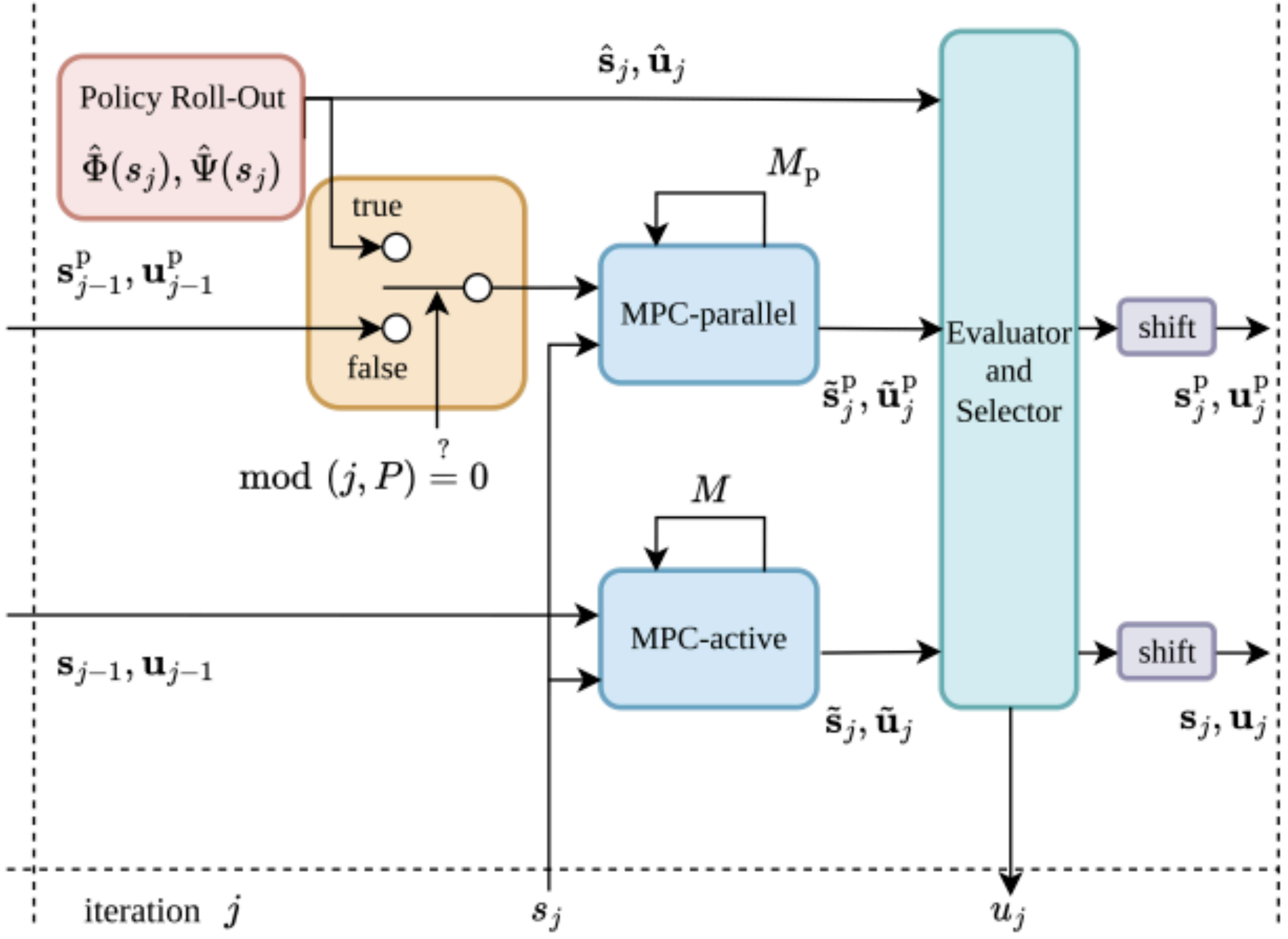
Examples

AC4MPC: Actor-Critic RL for NMPC

Reiter, Ghezzi, Baumgartner, Hoffmann, McAllister, Diehl (2024)

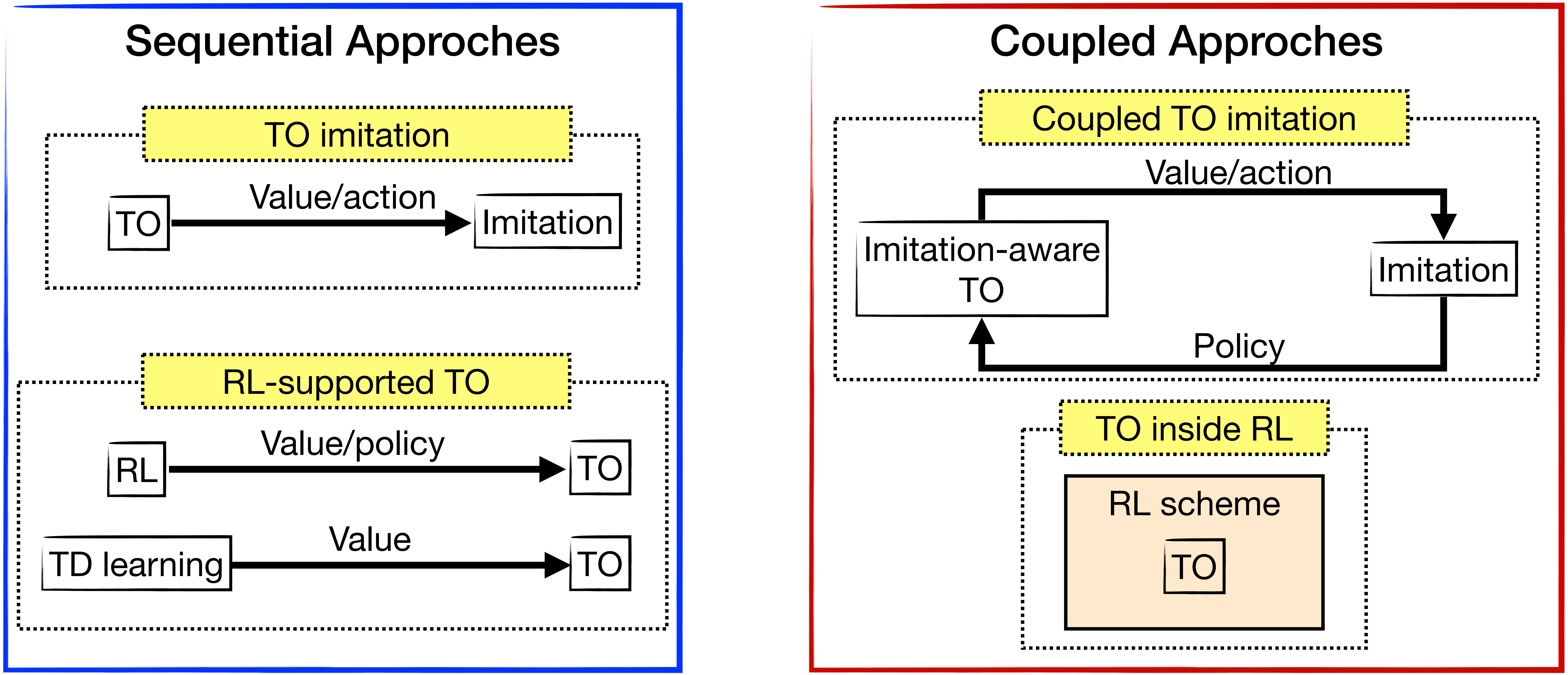
RL-supported TO

- First use RL (PPO, SAC) to compute policy and critic
- Then use RL critic as terminal cost in NMPC
- Solve MPC for two initial guesses: actor roll-out and shifted previous solution
- Use RL actor and critic to choose best solution based on approximate infinite horizon cost



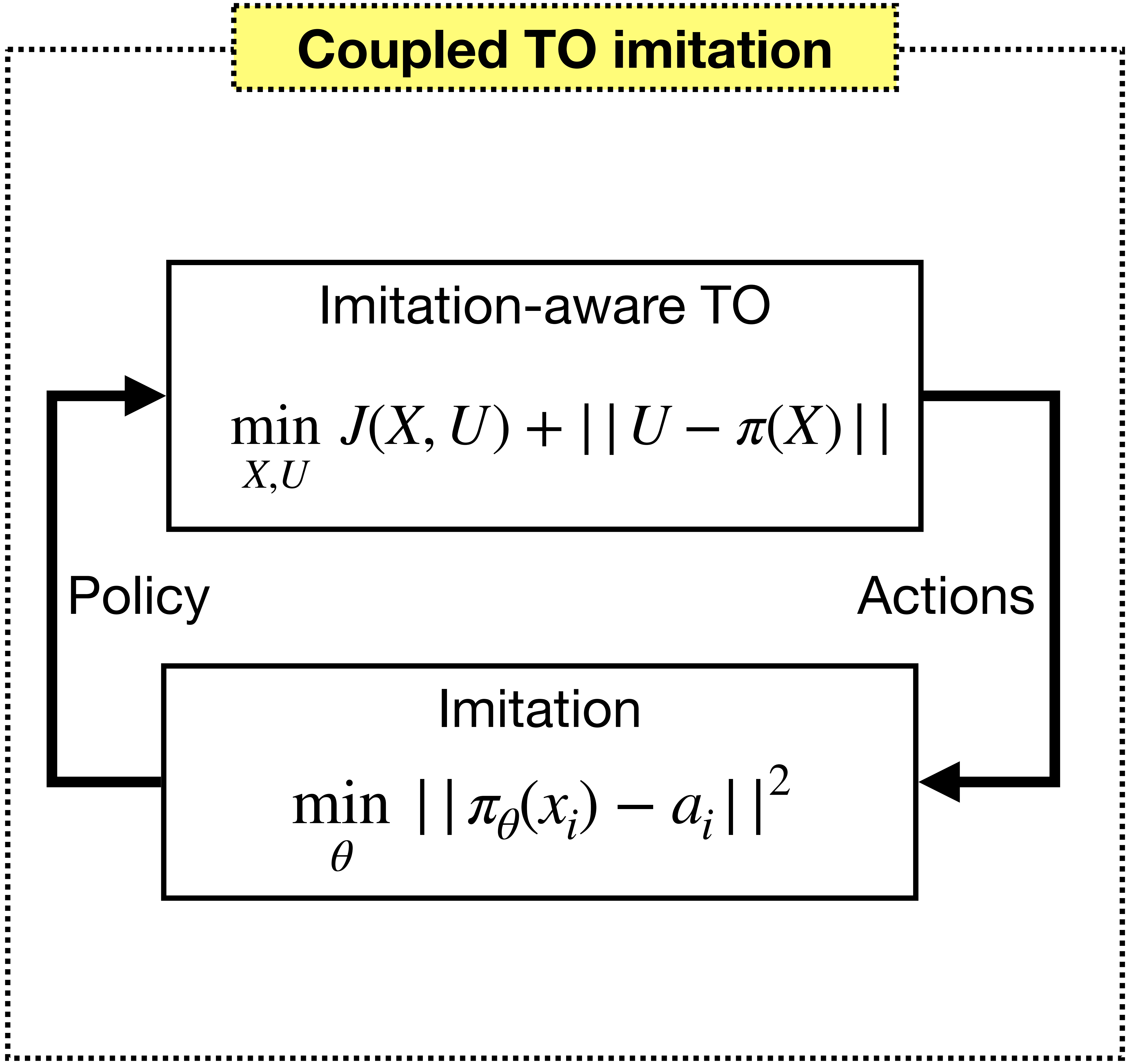
Architectures Combining RL and TO

Overview



Coupled TO Imitation

Discussion



- Improvement over vanilla TO imitation
- Account for policy errors in TO
- Help TO discover actions that are easy to learn
- Same objectives and limitations as vanilla TO imitation

Coupled TO Imitation

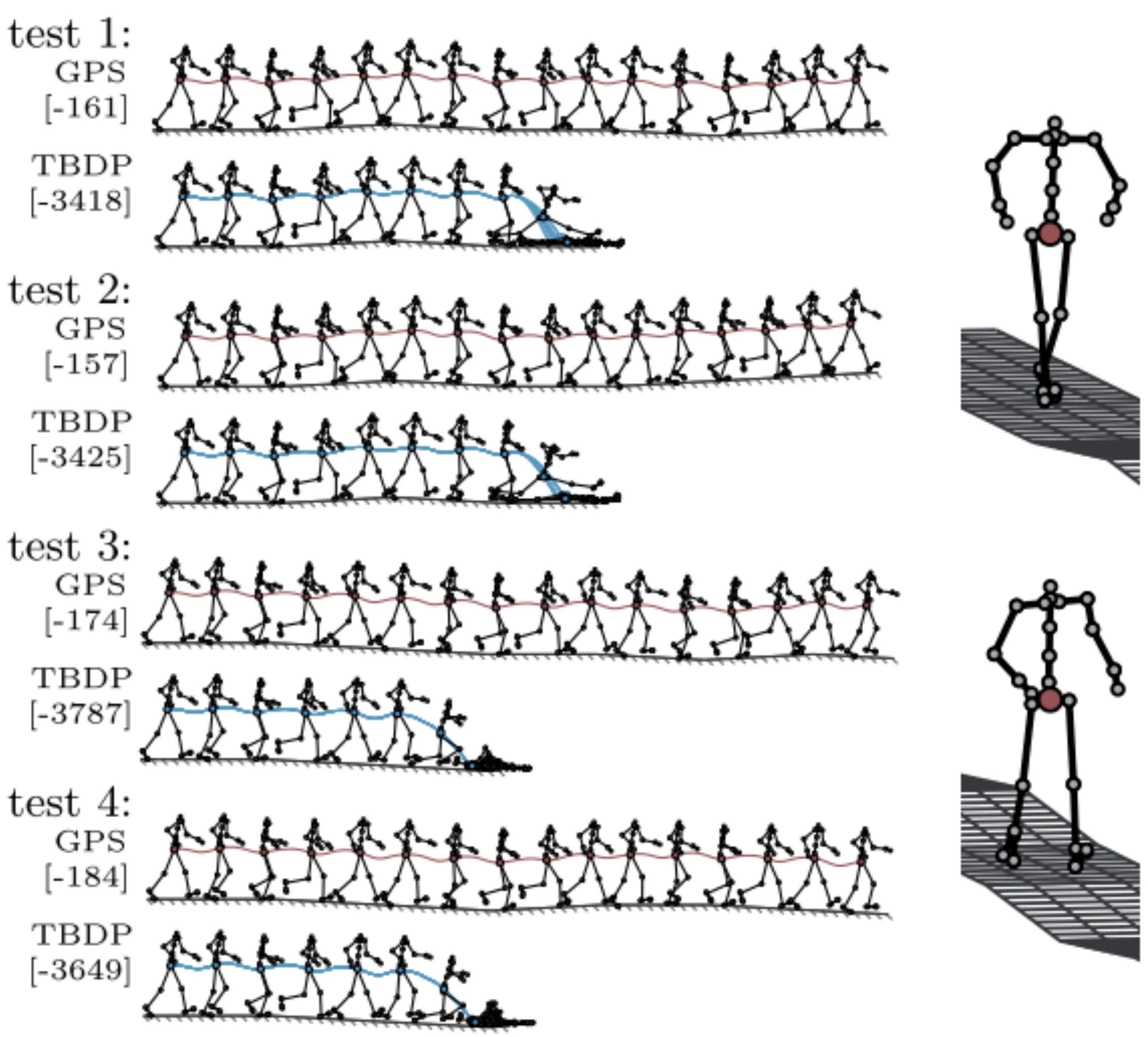
Examples

Guided Policy Search

Levine, Koltun (ICML 2013)

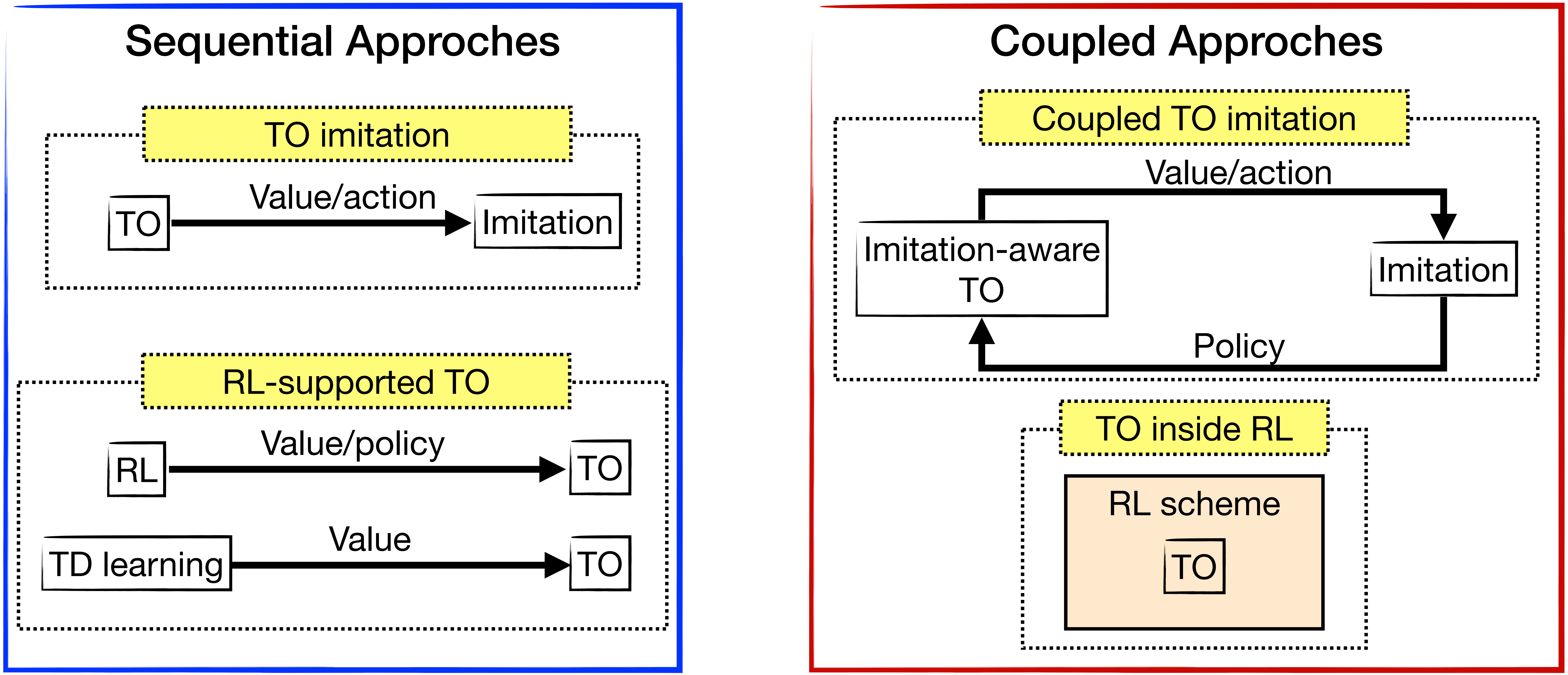
Coupled TO imitation

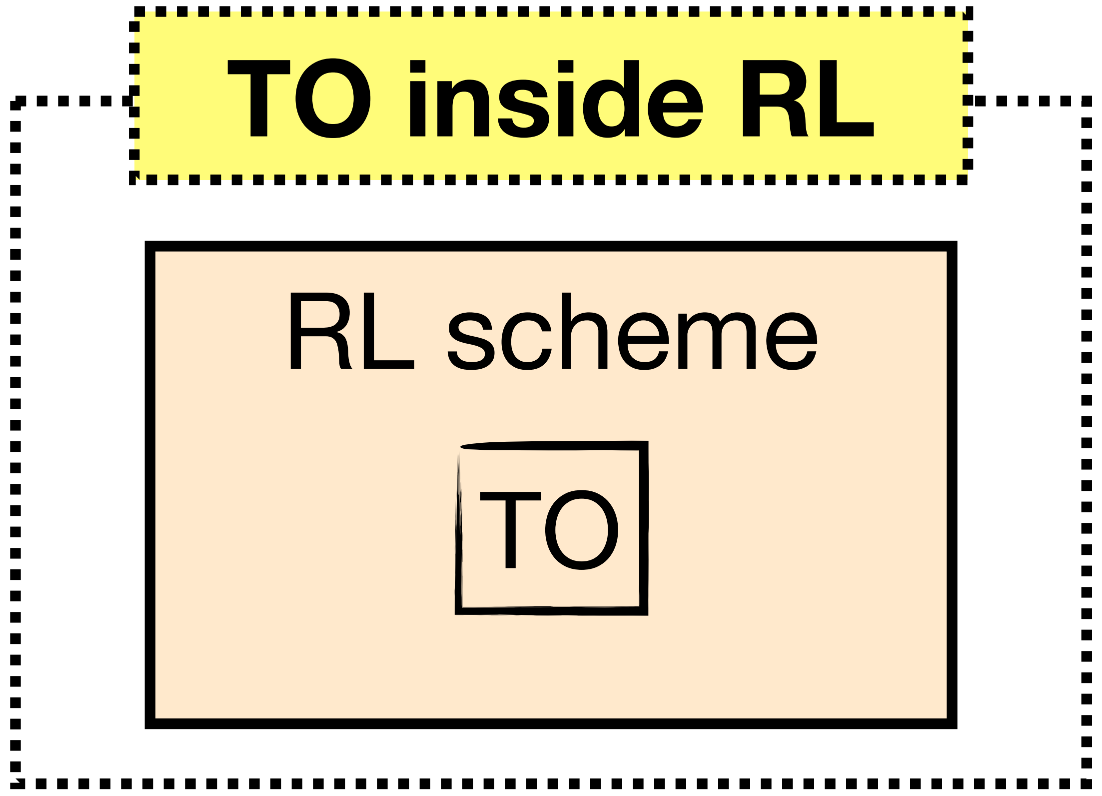
- Off-policy policy gradient method
- Optimize locally optimal trajectories with iLQR
- iLQR cost includes penalty to stay close to neural policy: log $\pi_\theta(u \mid x)$
- Penalty is necessary when initial samples cannot be reproduced by any policy, e.g. when they act differently in similar states
- Build stochastic policy from local iLQR controllers
- Train neural policy to imitate trajectories

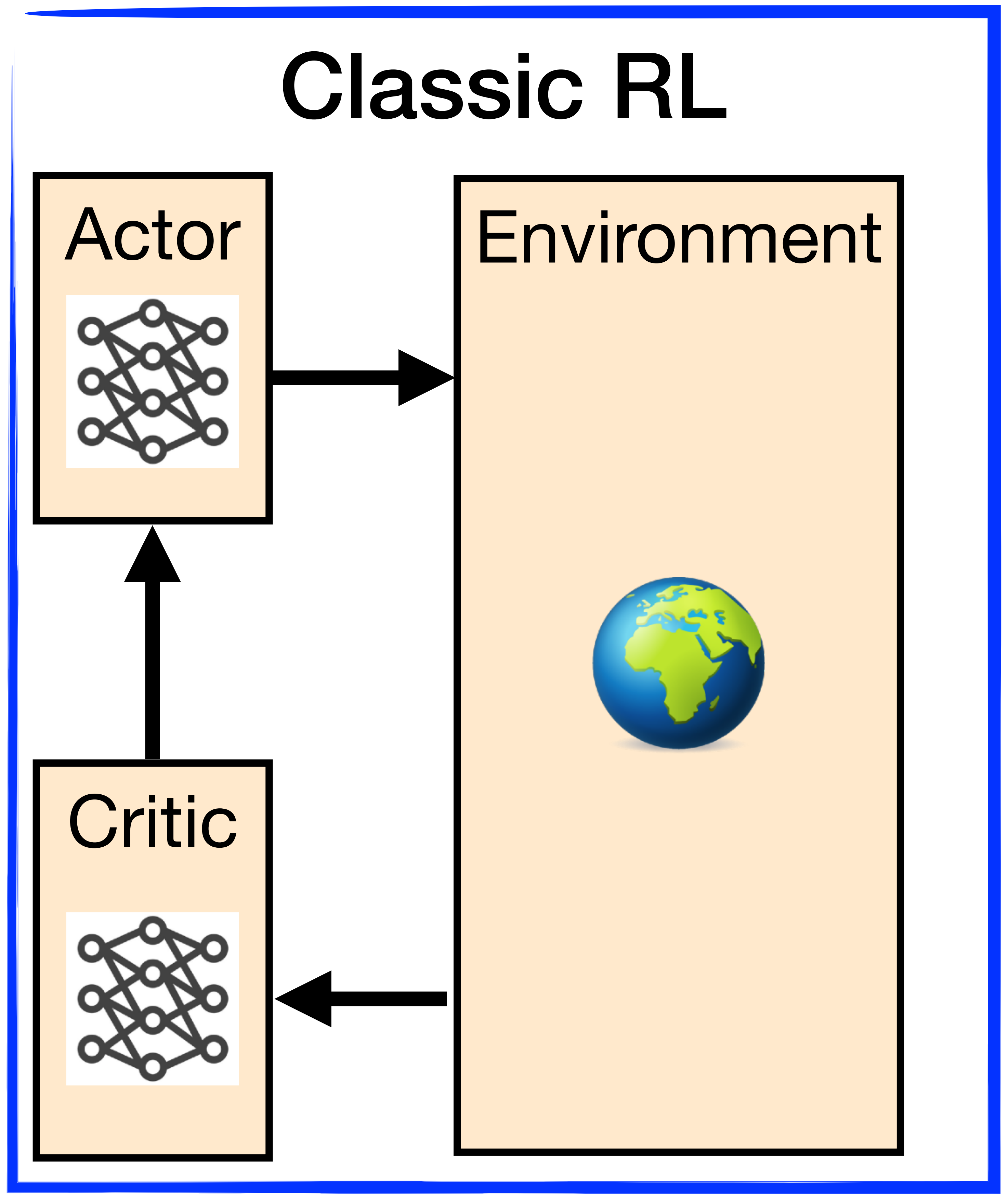


Architectures Combining RL and TO

Overview





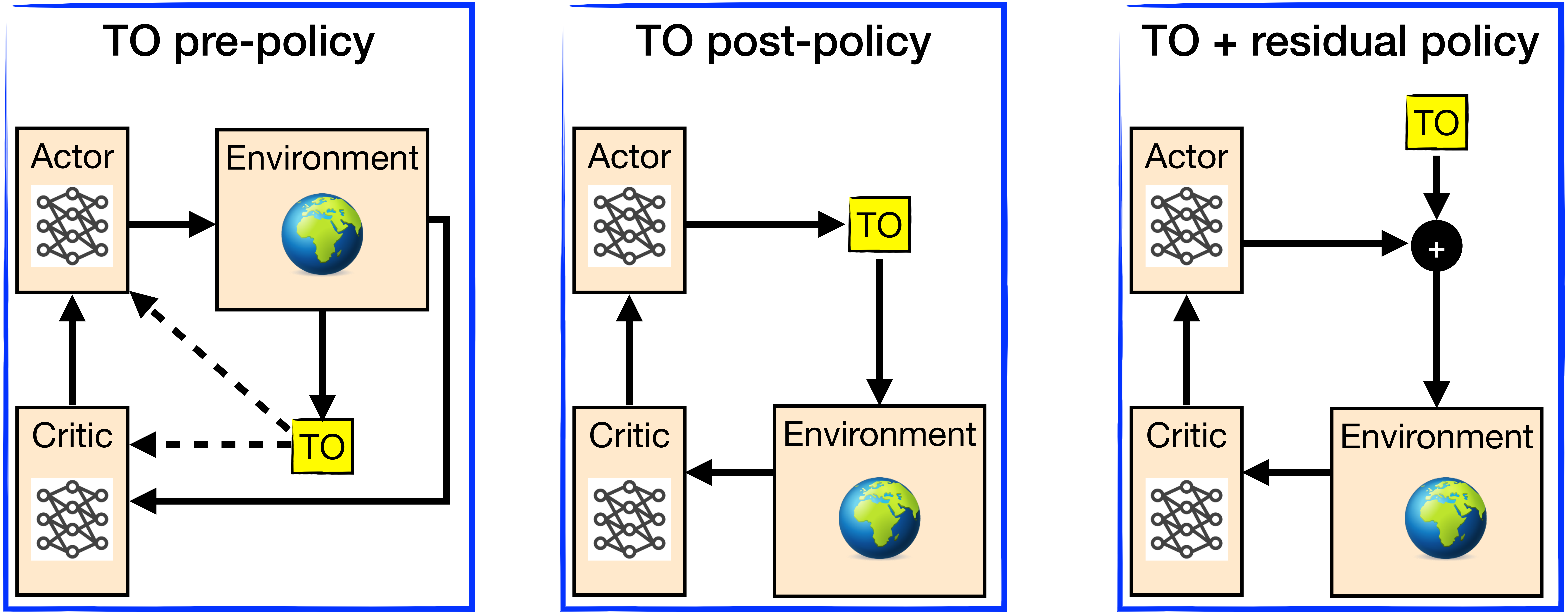


TO inside RL

Introduction

- Broad category
  - Where should TO be introduced?
  - What should be learned?
  - Should TO solve the same problem as RL?
- Try to overcome limitations of sequential and imitation-based approaches
- **Objectives**:
  - Speed-up RL training
  - Speed-up TO online computation
  - Guide TO towards high-quality solutions

Where should TO be introduced?

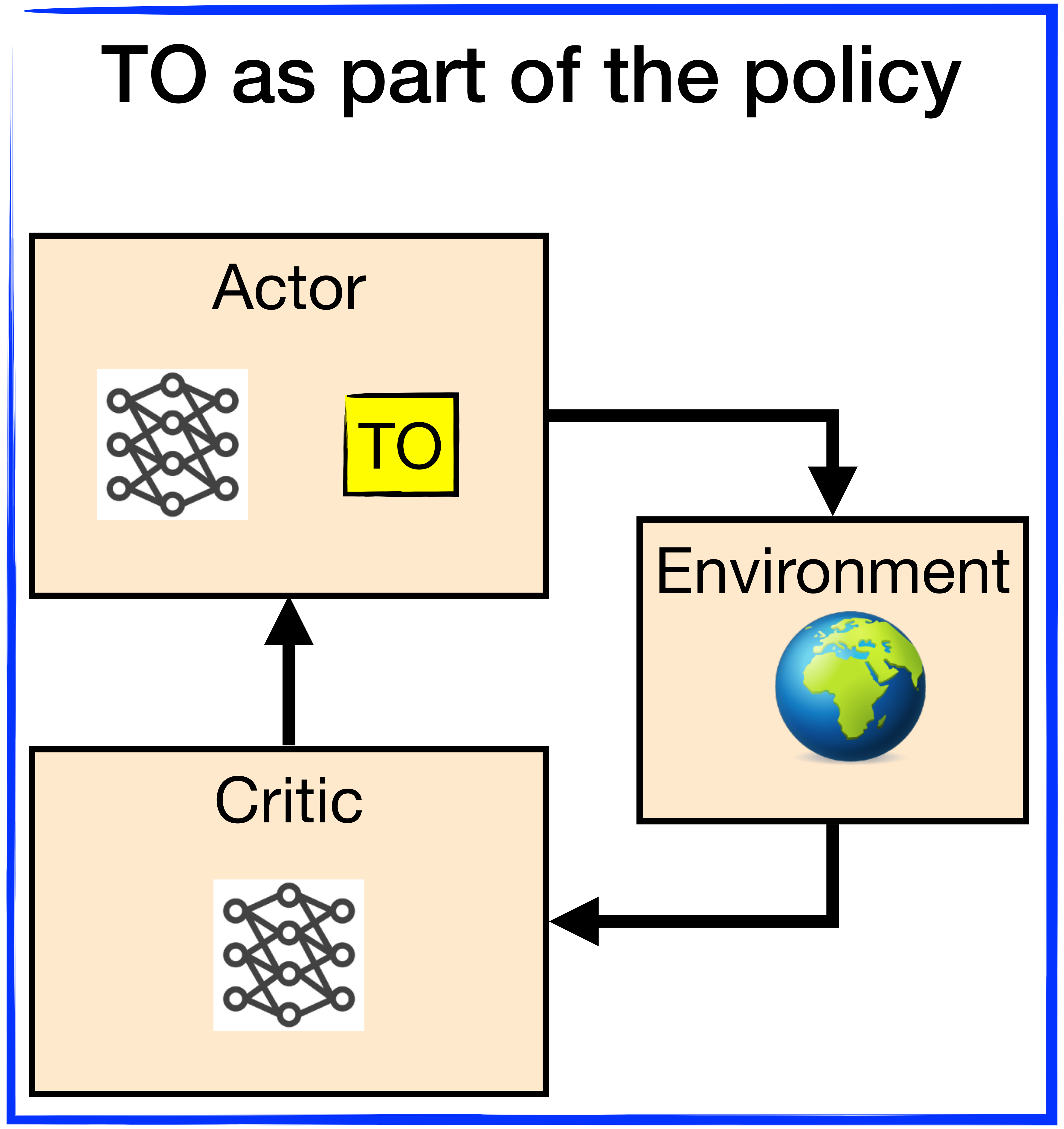


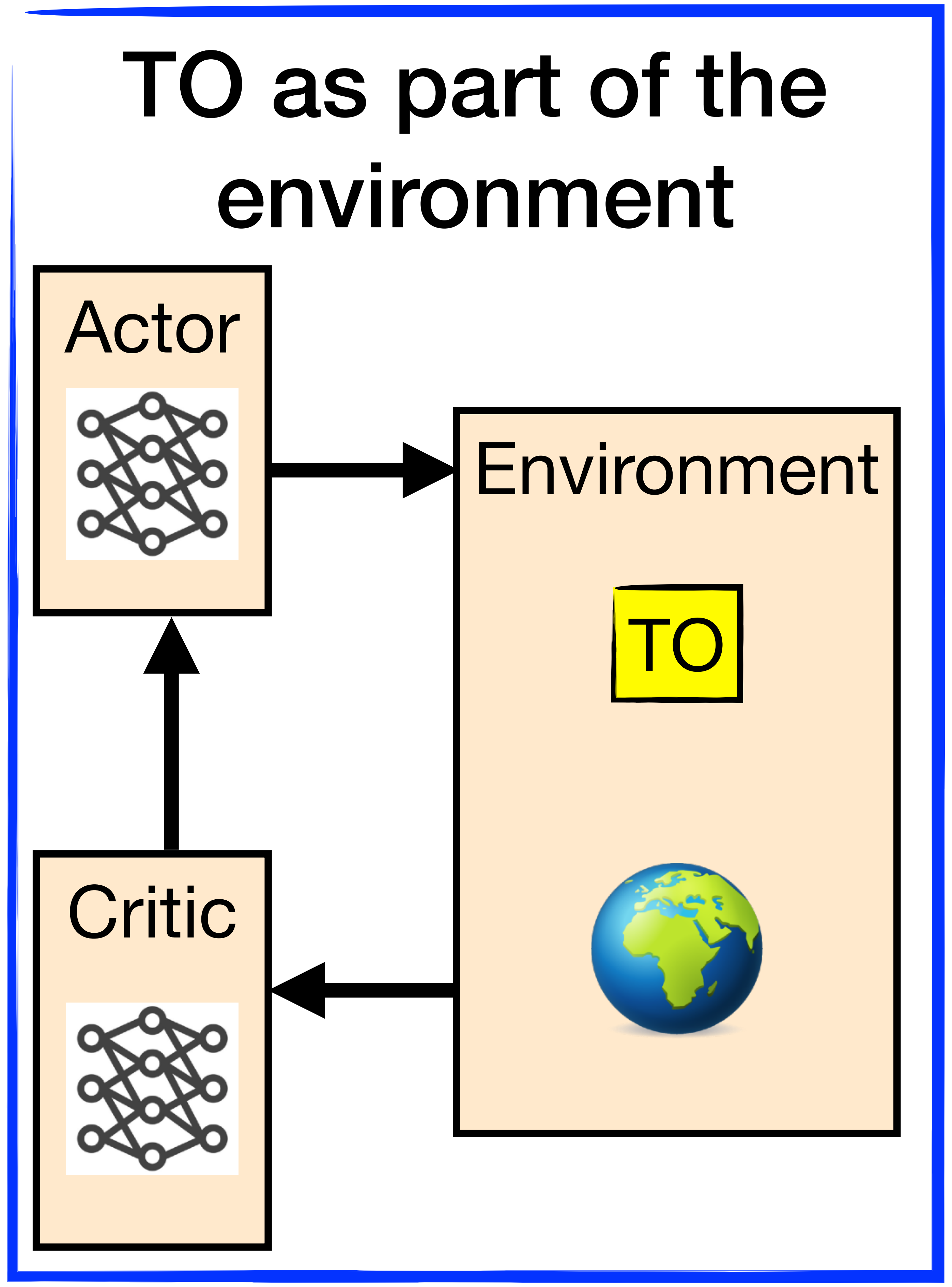
In which block should TO be considered?

Actor or environment?

Need to differentiate TO!

Actions are the output of TO



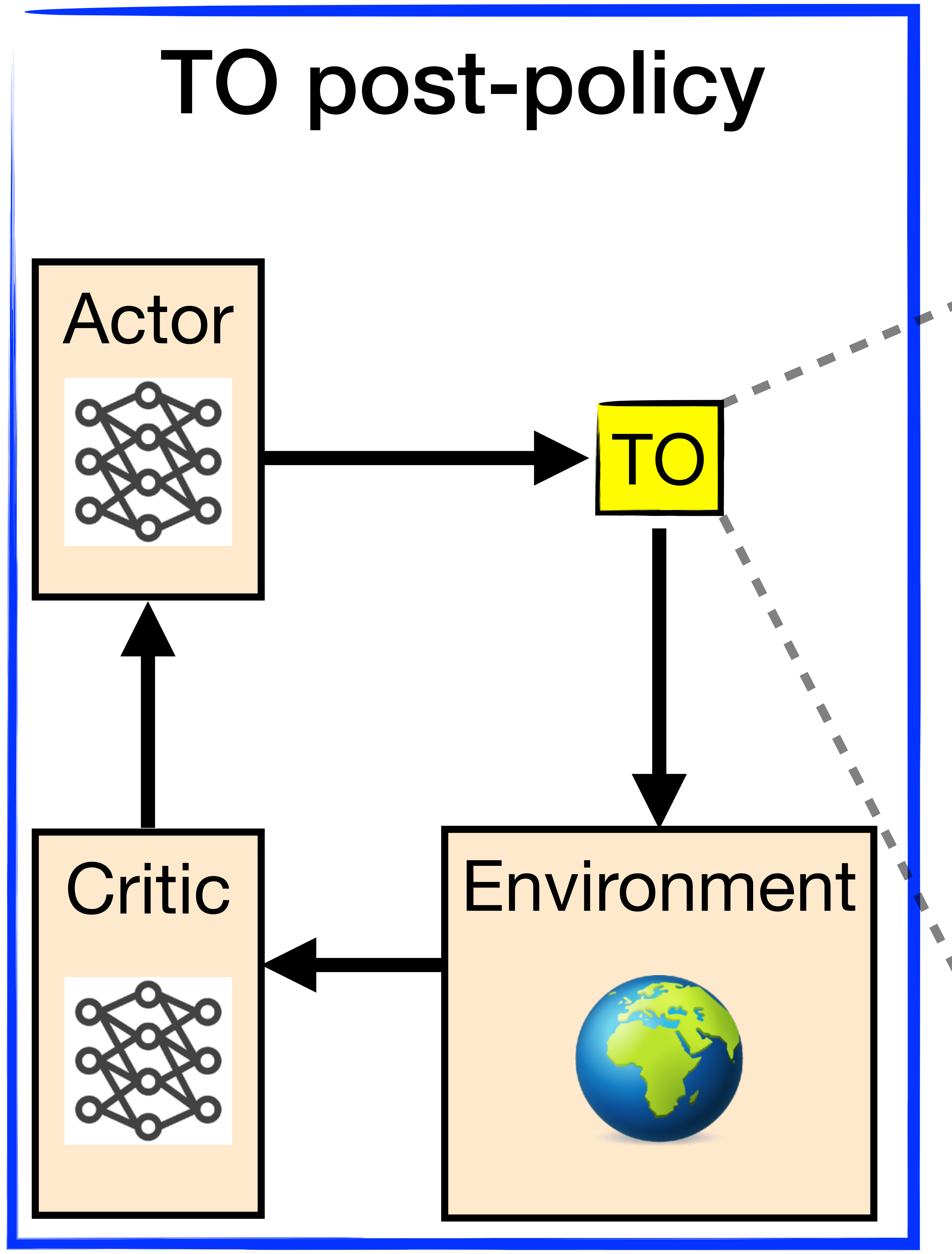


No need to differentiate TO!

Actions are the output of the actor policy

TO post-policy

What should the policy learn?



Running cost

Value function (i.e. terminal cost)

$$\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} \quad \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N)$$

$$\text{subject to} \quad x_{i+1} = f(x_i, u_i) \qquad i = 0 \ldots N-1$$

$$x_{i+1} \in \mathcal{X}, u_i \in \mathcal{U} \qquad i = 0 \ldots N-1$$

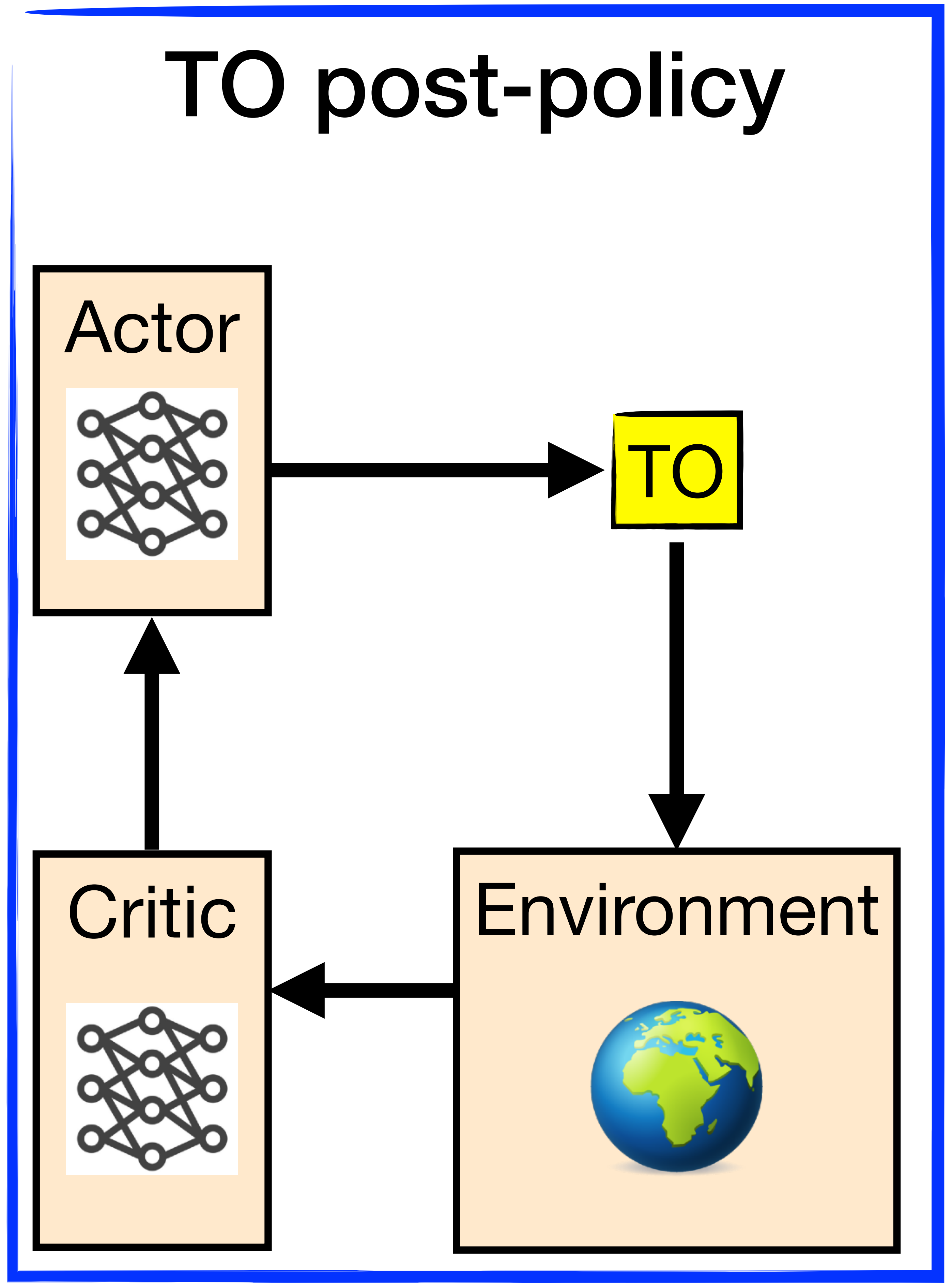
Policy (used to warm-start)

Running constraints

Dynamics

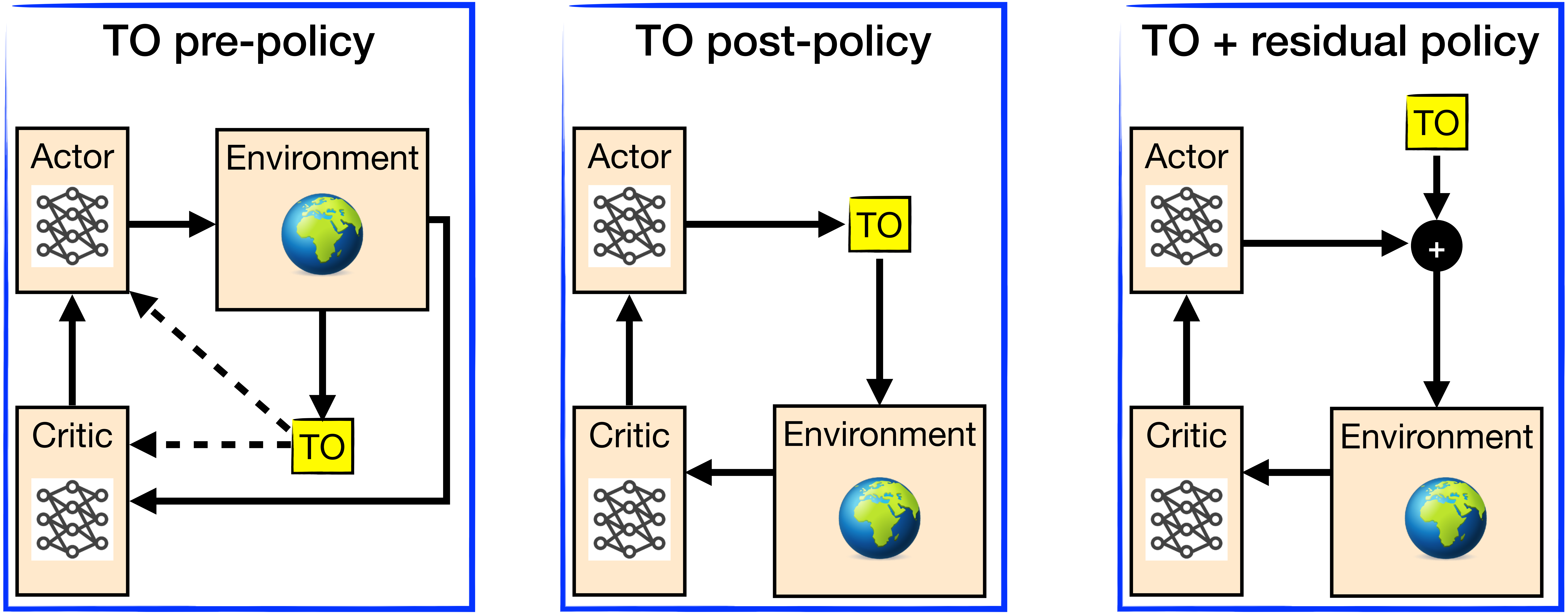
TO post-policy

What should the policy learn?



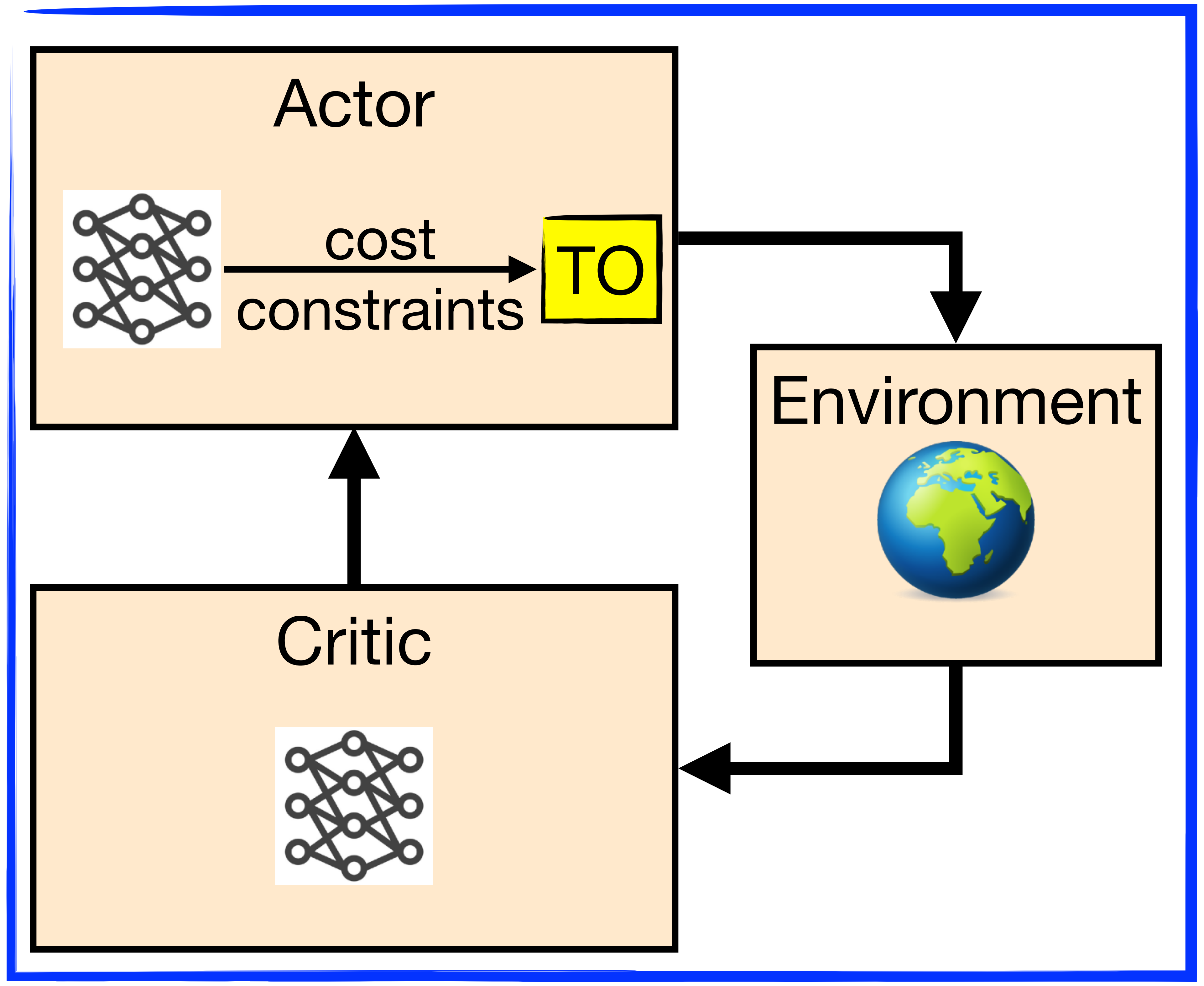
Do TO and RL solve the same problem?	
NO	**YES**
Running cost\constraints	Policy (used to warm-start)
Dynamics	Value function (used as terminal cost)

Where should TO be introduced?



TO post-policy: Learning the cost/constraints

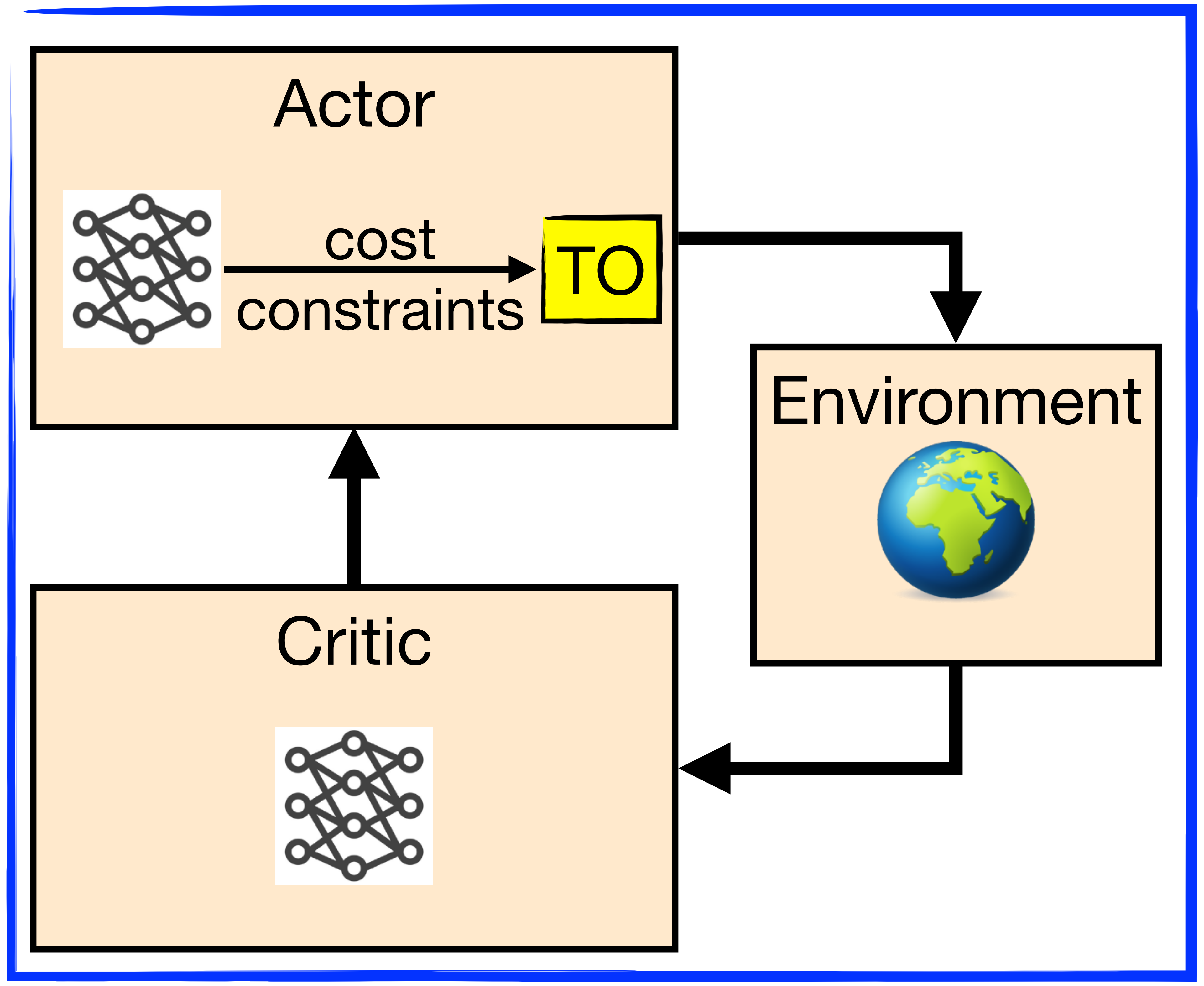
Overview



- Sensor-based neural network parametrizes cost/constraints, which can be:
  - physics-based (e.g. target to reach),
  - or not (e.g. general quadratic function)
- TO is either part of the actor or of the environment (i.e. differentiated or not)
- RL and TO solve different problems
- Could theoretically parametrize also dynamics, but not done in practice

TO post-policy: Learning the cost/constraints

Discussion



- **Objectives**:
  - Speed-up RL (and potentially TO)
  - Exploit sensor data in TO (neural cost can be sensor-based)
  - Better handle out of distribution behaviors
  - Exploit TO's guarantees in RL (safety, stability)
- **Limitations**:
  - Need to solve TO online to use policy
  - Hard to ensure TO is fast, safe and generalizes when out of distribution

TO post-policy: Learning the cost /constraints

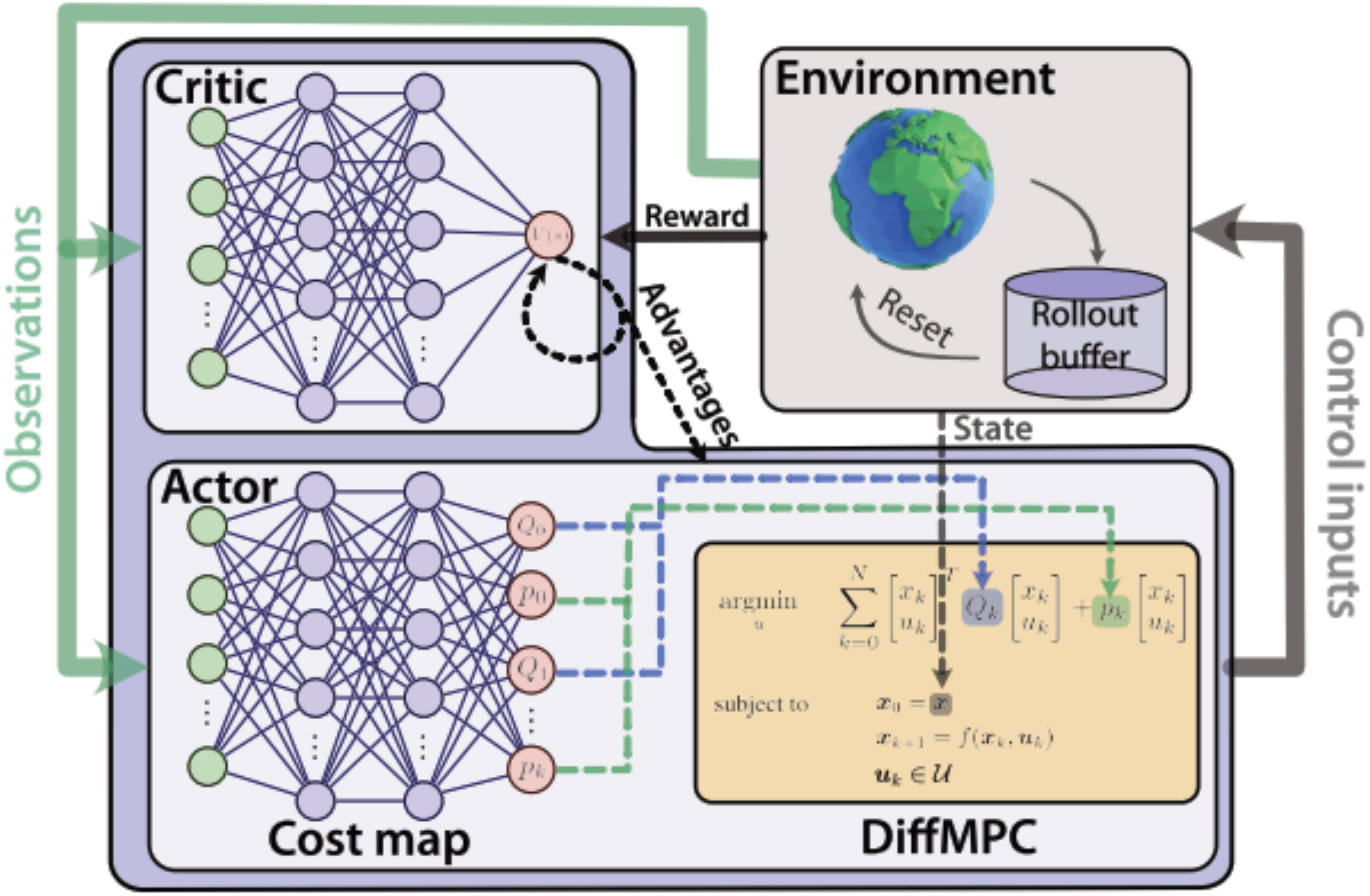
Examples

Actor-Critic Model Predictive Control

Romero, Song, Scaramuzza, ICRA 2024

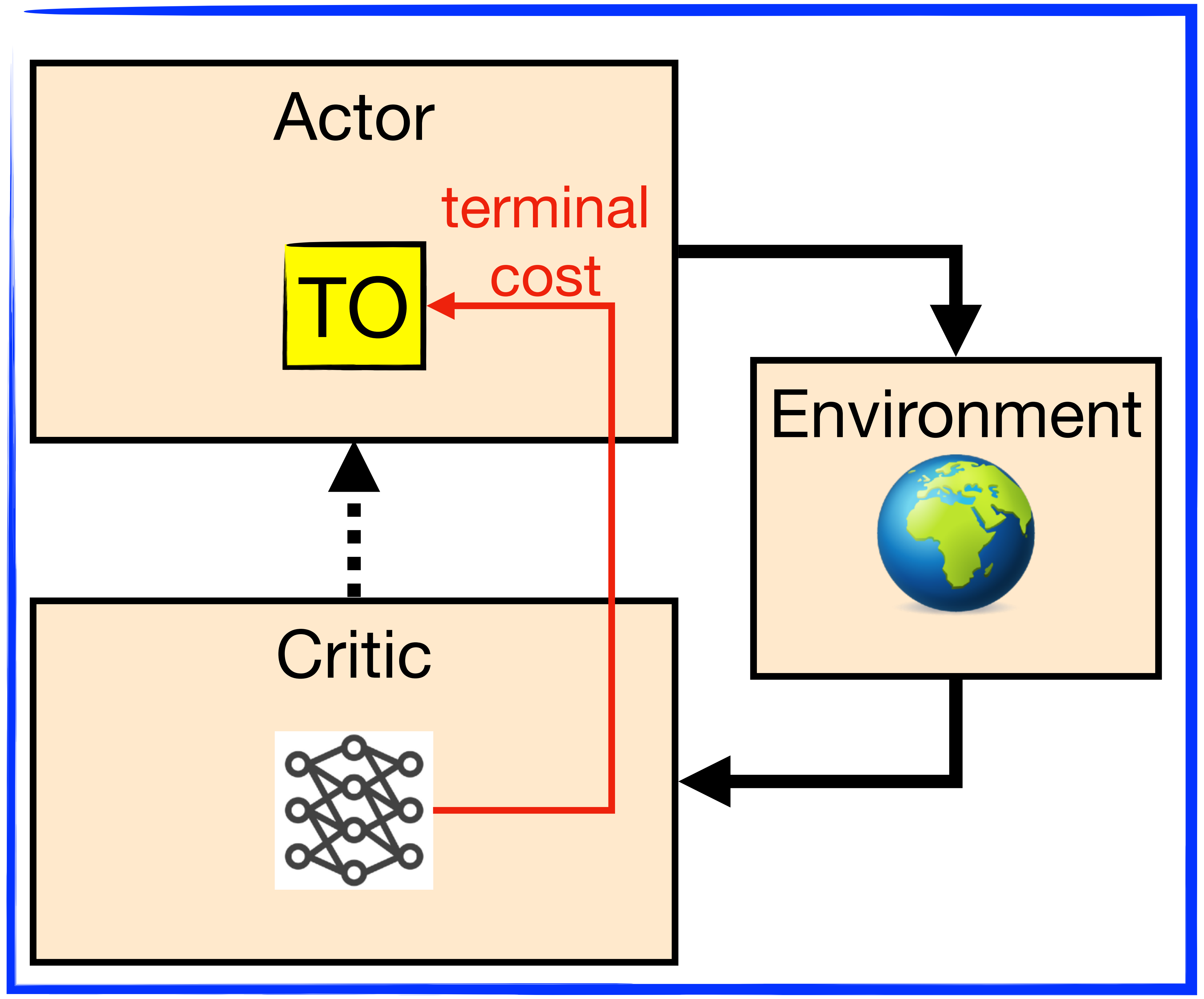
TO post-policy: learning the cost

- Differentiable MPC with input constraints as last layer of actor policy
- Quadratic MPC cost with parametric coefficients
- Exploration by sampling u around output of MPC with variance controlled by PPO
- Validation in simulation and real quadcopter platform



TO post-policy: Learning the terminal cost

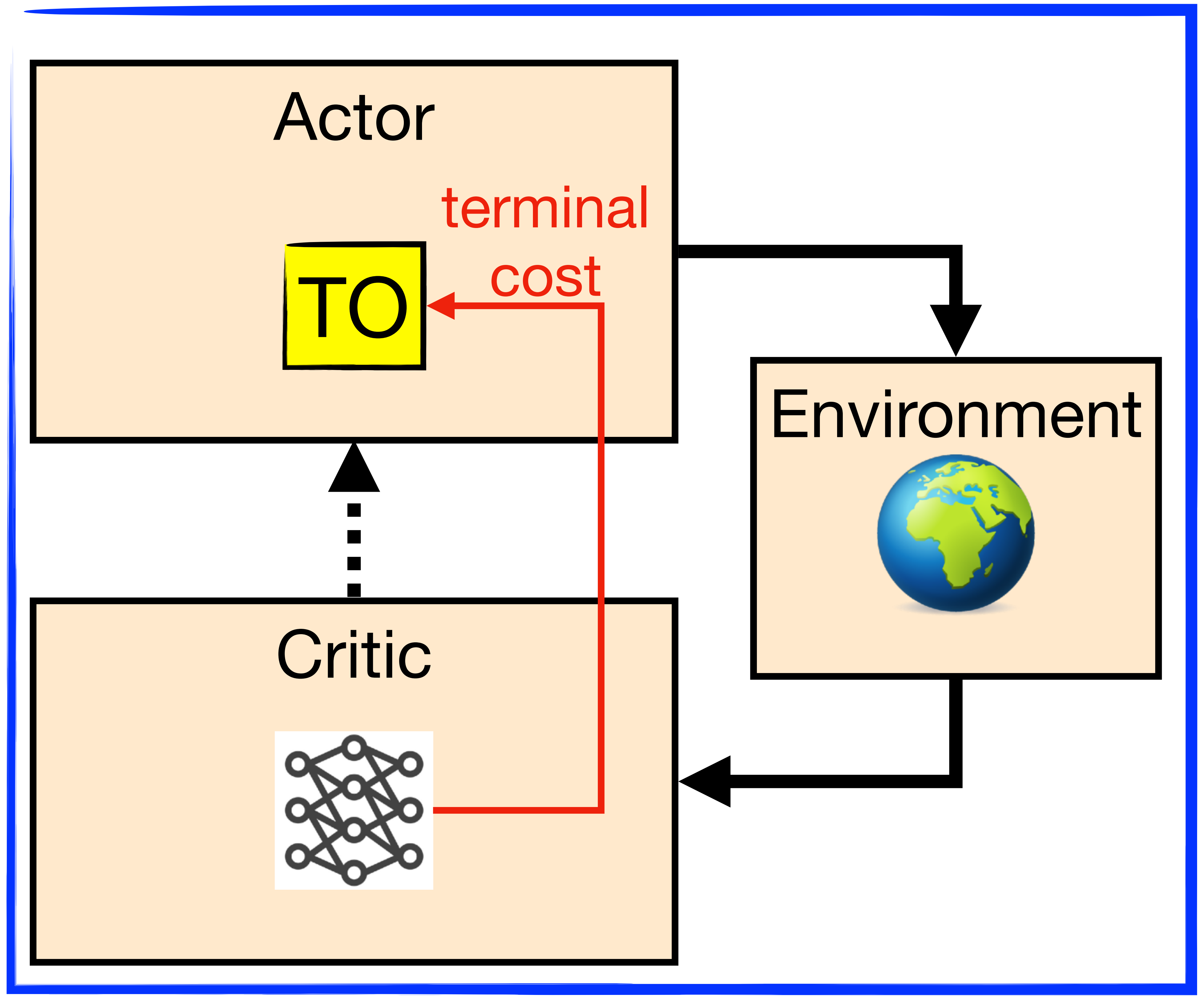
Overview



- Neural network parametrizes terminal cost
- Network trained with TD learning to match Value function
- No policy representation
- No policy improvement step
- Same problem solved by TO and RL

TO post-policy: Learning the terminal cost

Discussion



- **Objectives:**
  - Speed-up RL training and TO computation
  - Help TO find high-quality solutions
  - Satisfy constraints (but no recursive feasibility)
- **Limitations:**
  - TO could be slow or find bad solutions even with perfect Value function
  - Need to solve TO at deployment
  - TO does not exploit sensor data

TO post-policy: Learning the terminal cost

Examples

Plan online, learn offline (POLO)

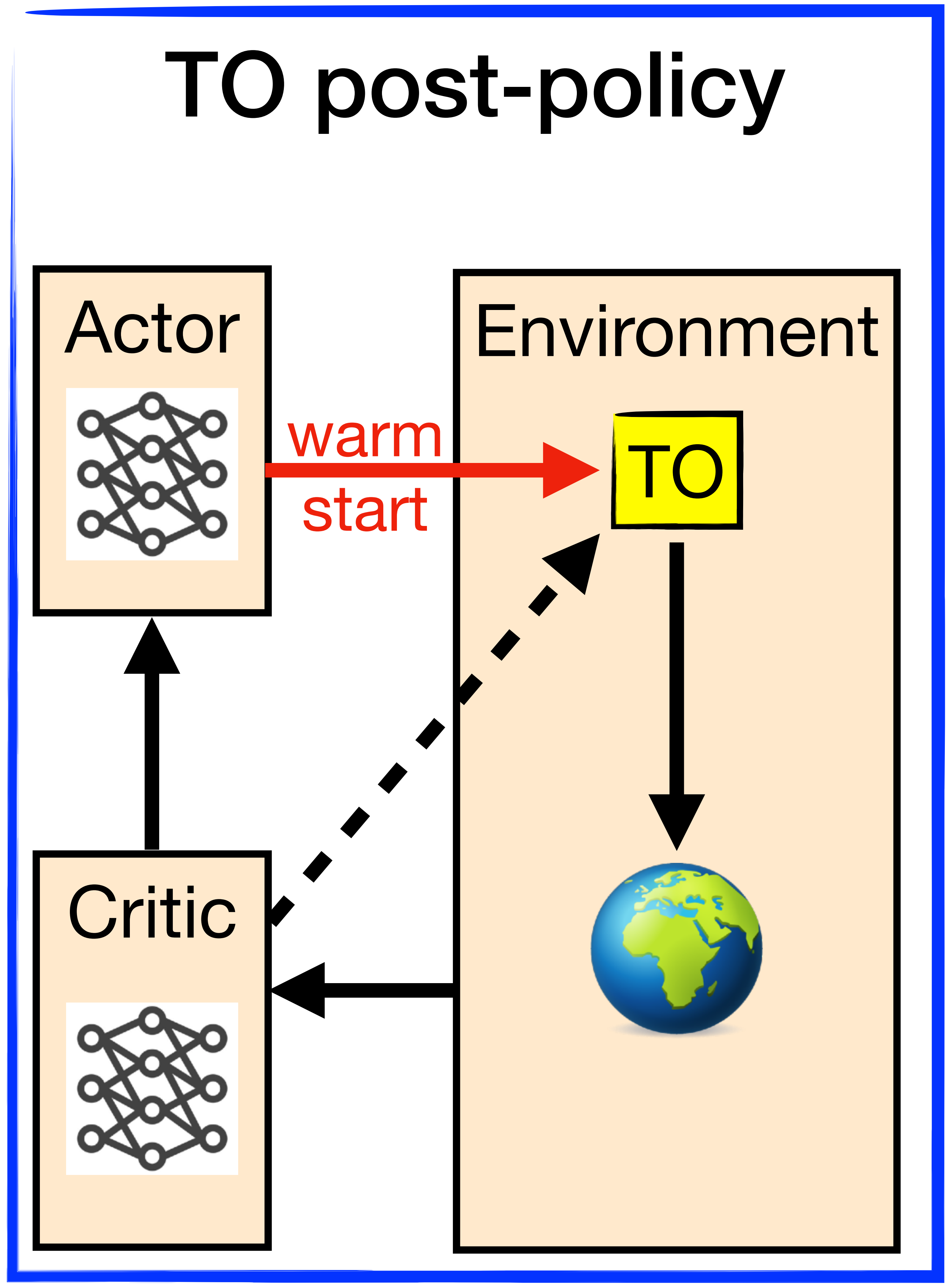
Lowrey, Rajeswaran, Kakade, Todorov, Mordatch (ICLR 2019)

TO post-policy: learning the terminal cost

- TO with learned Value function
- Use an ensemble of Value function approximators to capture uncertainty
- Use soft-max (computed as log-sum-exp) of Value functions to encourage exploration according to “optimism in the face of uncertainty”
- Use MPPI for TO
- Exploration strategy designed to avoid TO exploiting Value approximation errors

TO post-policy: Learning a warm-start policy

Overview



- Actor used to warm-start TO
- Optionally, critic used in TO as terminal cost
- **Objectives**:
  - Speed up RL training (and potentially TO)
  - Satisfy constraints
  - Help TO find high-quality solutions
  - Get policy for fast deployment
- **Limitations**:
  - TO/policy do not exploit sensor data

TO post-policy: learning a warm-start policy

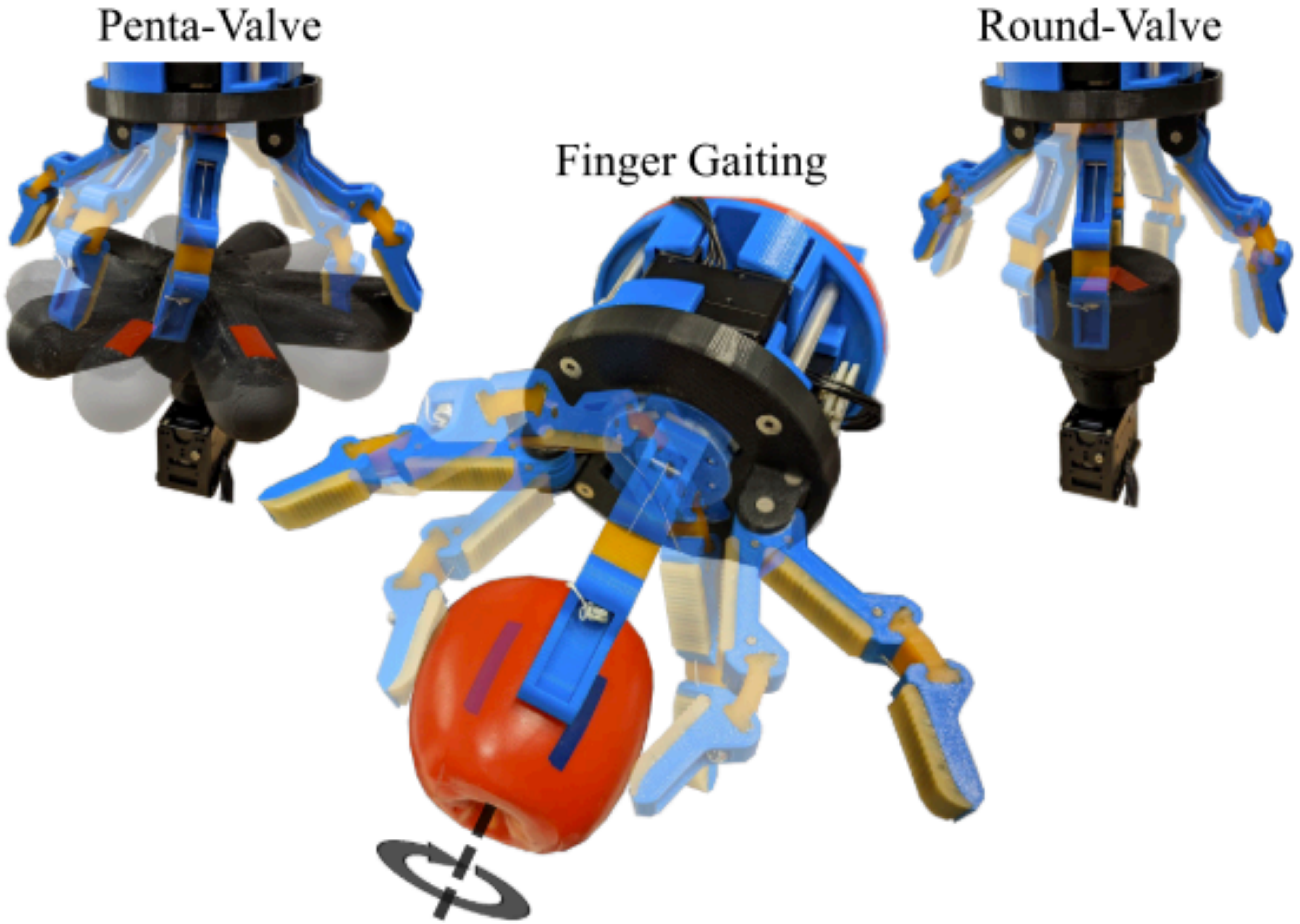
Examples

Model Predictive Actor-Critic (MoPAC)

Morgan, Nandha, Chalvatzaki, D'Eramo, Dollar, Peters (ICRA 2021)

- Ensemble model learning from environment transitions
- i-MPPI warm-started by neural policy and using neural Value as terminal cost
- SAC: soft policy evaluation and soft policy improvement (maximum entropy) over mixture of models and environments
- SAC encourages exploration, while MPPI encourages exploitation
- Similar to CACTO (next slide), but does not exploit model derivatives

TO post-policy: learning warm-start policy

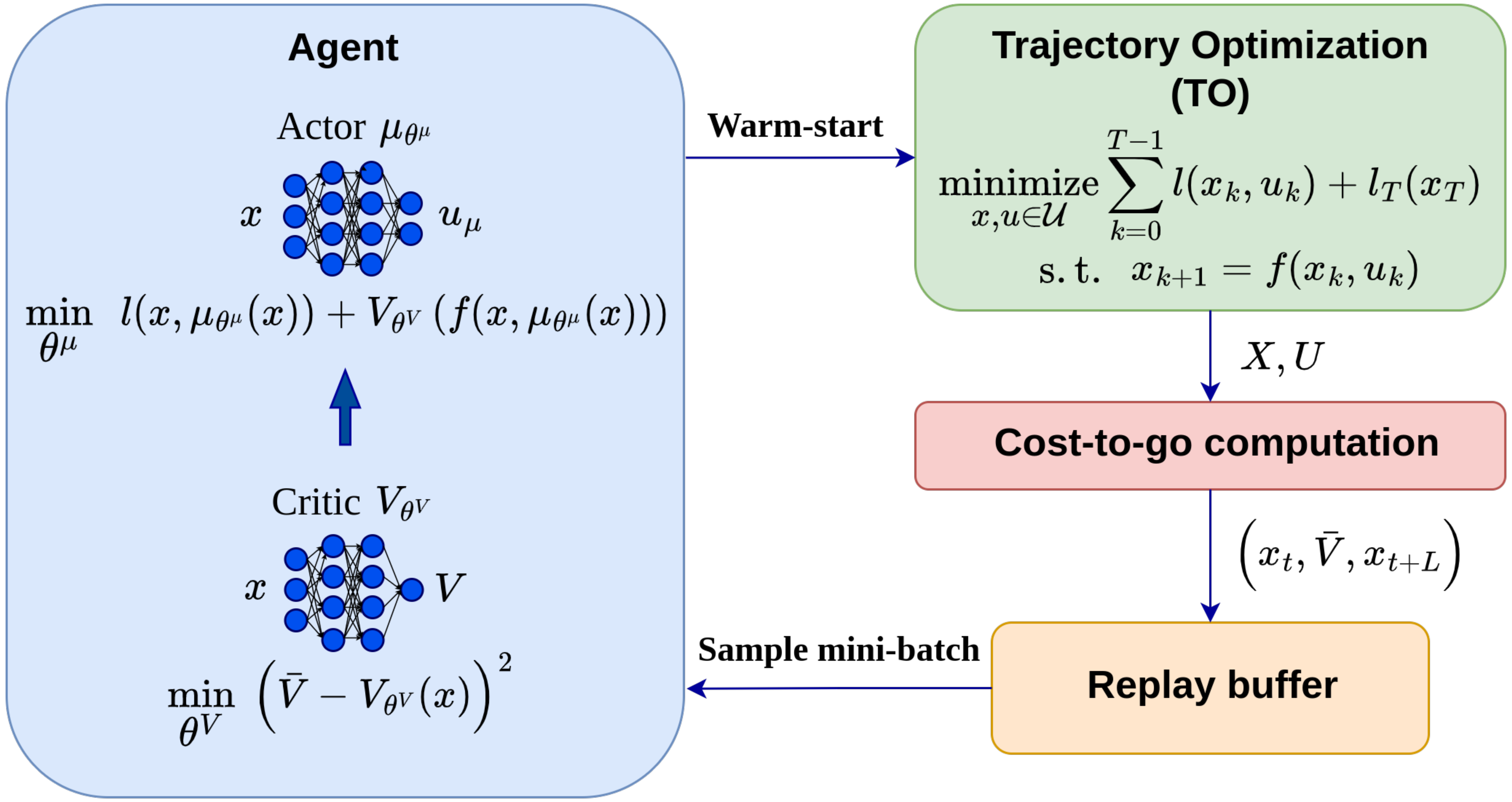


Continuous Actor-Critic with TO (CACTO)

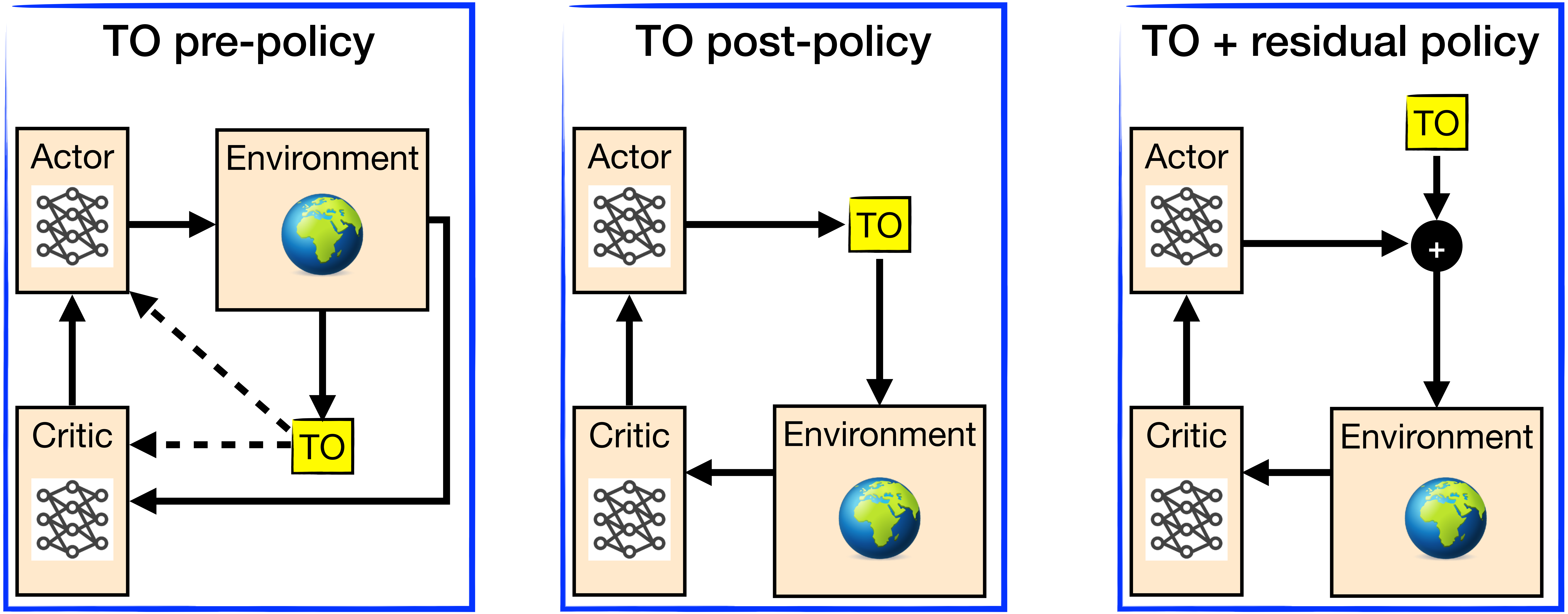
Grandesso, Alboni, Rosati Papini, Wensing, Del Prete (RAL 2023)

TO post-policy: learning warm-start policy

- Derivative-based TO optimizes warm-start computed with policy roll-out
- Value network trained with TD(N)
- Deterministic policy improved minimizing Q function
- Dynamics assumed to be known and differentiable
- Exploration ensured by uniform sampling of initial TO states (no need to explore action space)

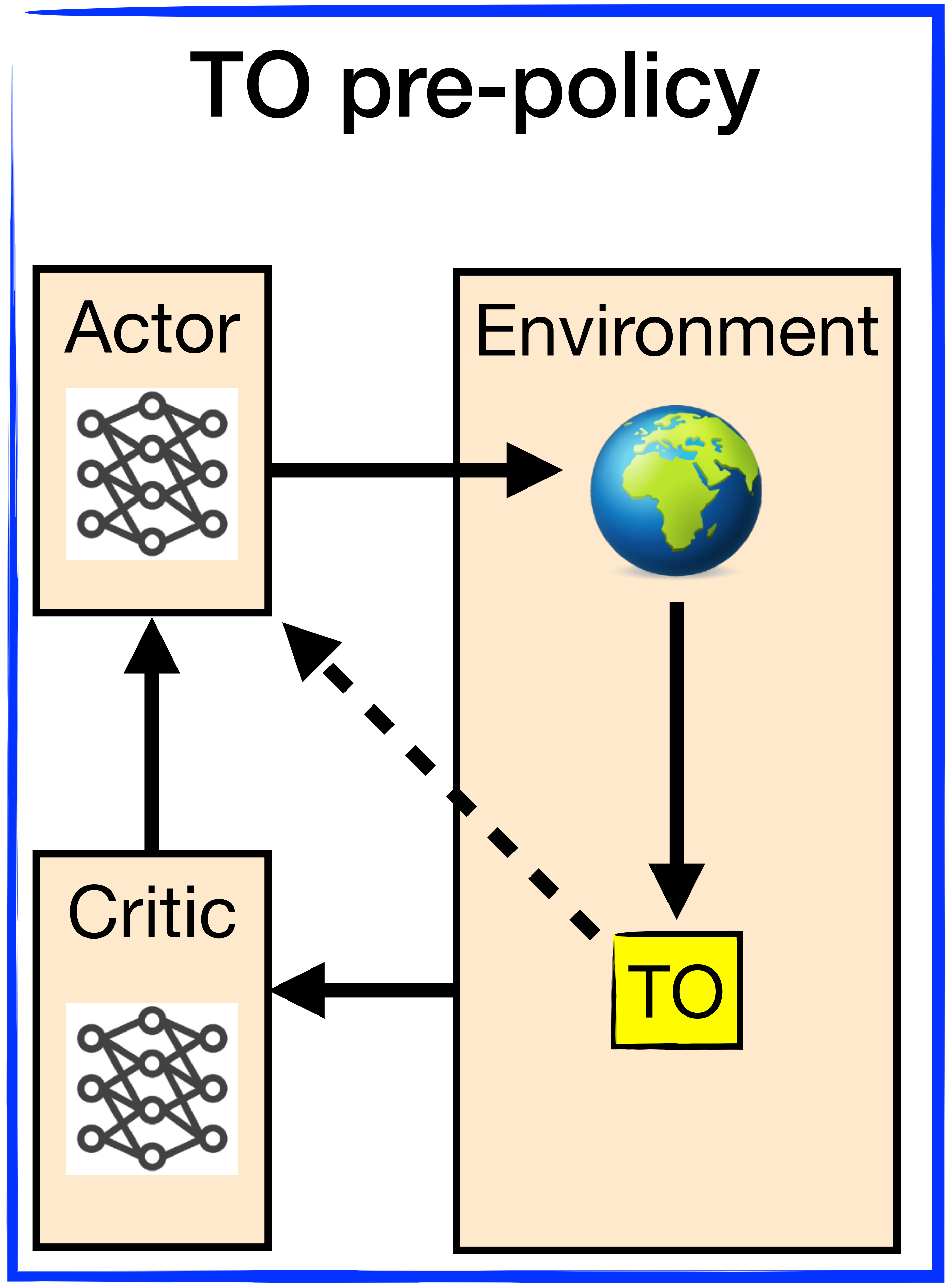


Where should TO be introduced?



TO pre-policy

Overview



- TO computes some quantities (e.g. reference motion to track) that is inputted to the RL policy
- TO could rely on simplified model for speed
- **Objectives**:
  - Speed up RL training
  - Satisfy constraints
- **Limitations**:
  - TO must be solved online to use policy
  - Rely on TO's ability to find good solutions
  - Policy could violate constraints even if TO does not

Pre-policy TO

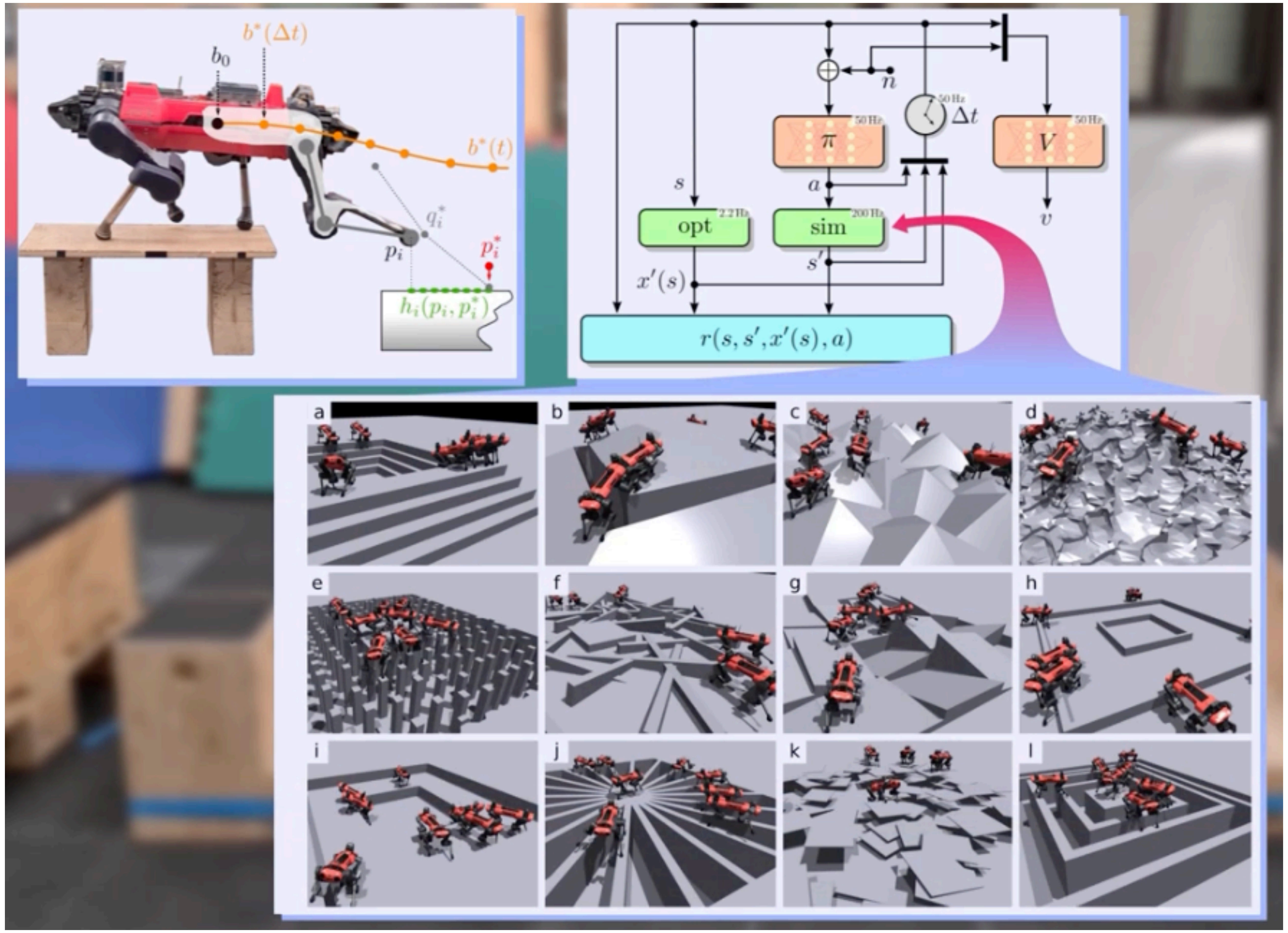
Examples

DTC: Deep Tracking Control

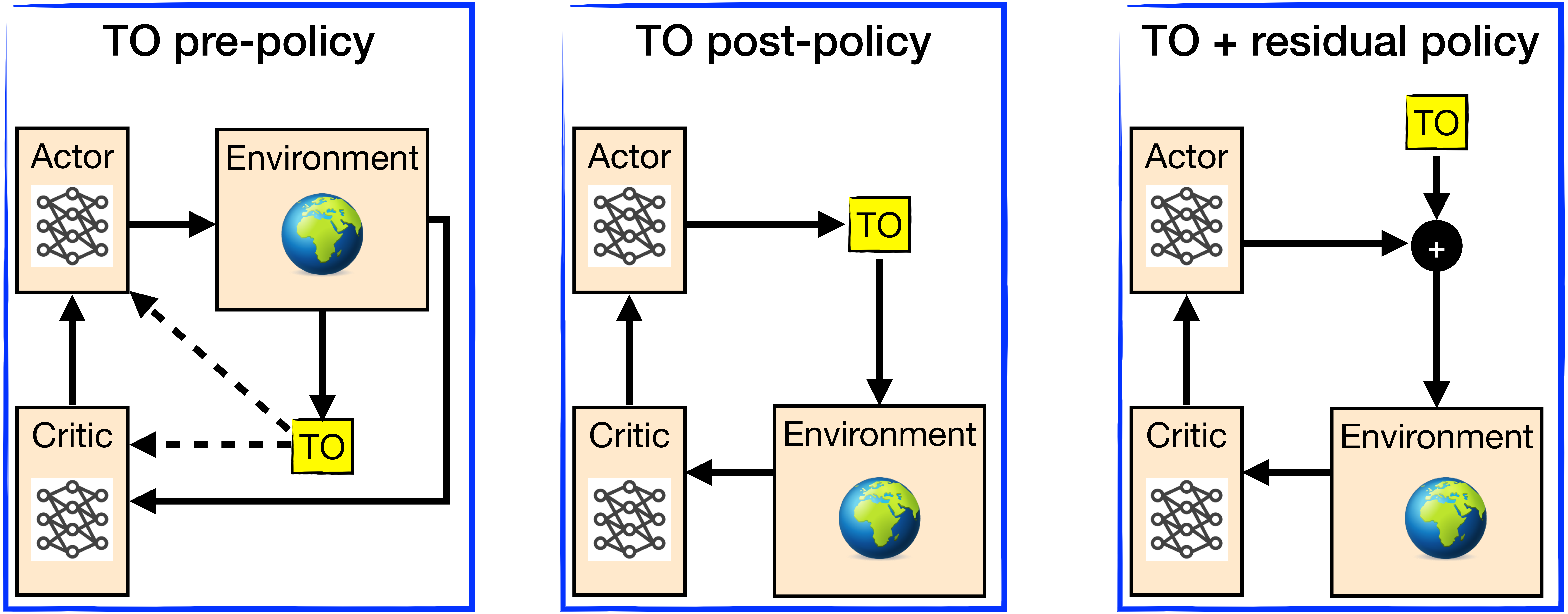
Jenelten, He, Farshidian, Hutter (Science Robotics 2024)

TO pre-policy: learning to track reference

- High-level MPC plans footholds at low rate
- Low-level RL policy follows the footholds at high rate
- MPC used during training
- Reward desired foothold positions at planned touch-down
- MPC on CPU, RL (PPO) on GPU

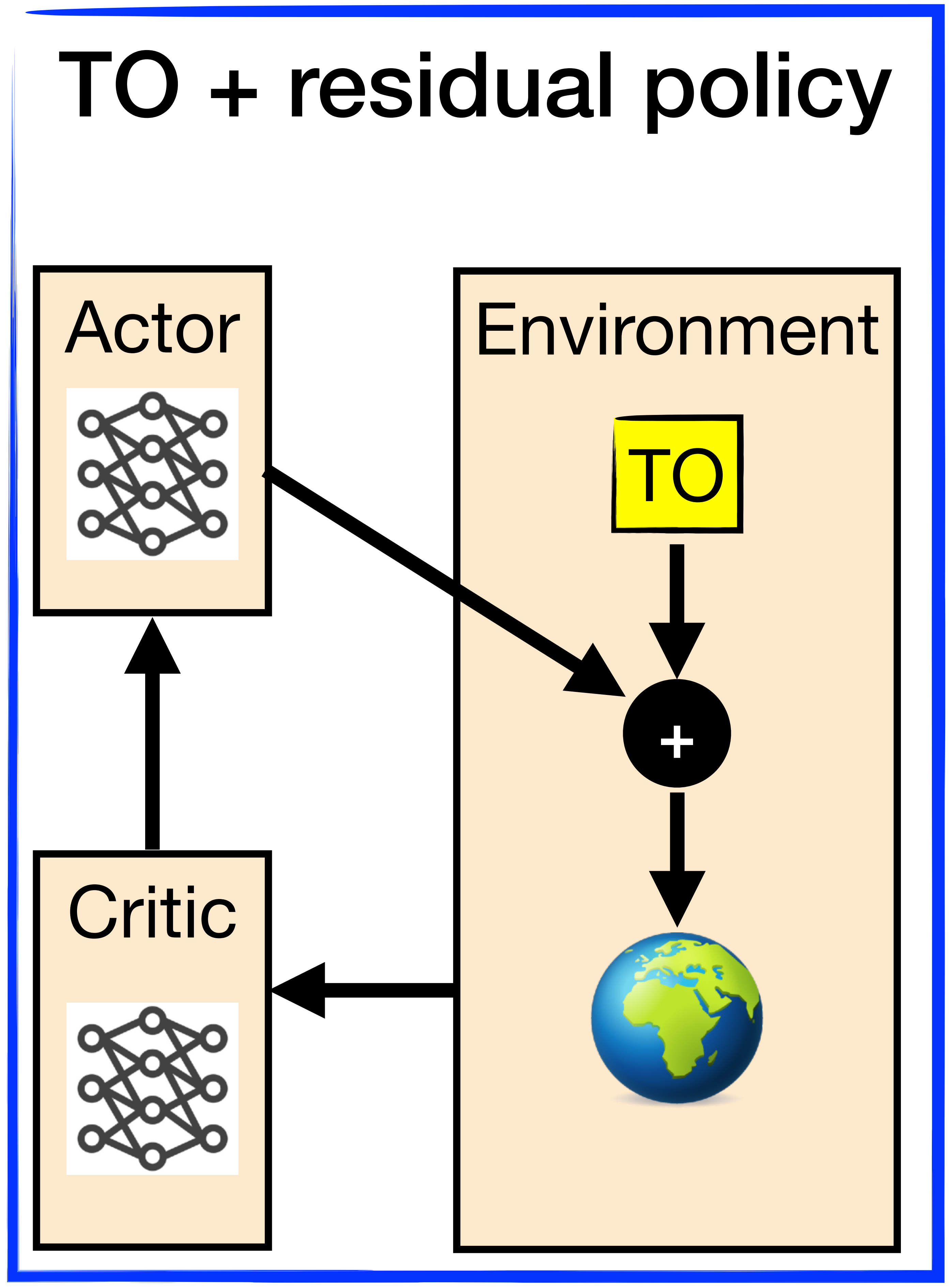


Where should TO be introduced?



TO + Residual policy

Discussion



- TO computes nominal control inputs
- RL policy computes additional control inputs to improve closed-loop performance
- **Objective**:
  - Speed up RL training
- **Limitations**:
  - TO must be solved online to use policy
  - Rely on TO's ability to find (roughly) good solutions
  - Policy could violate constraints even if TO does not

TO + Residual policy

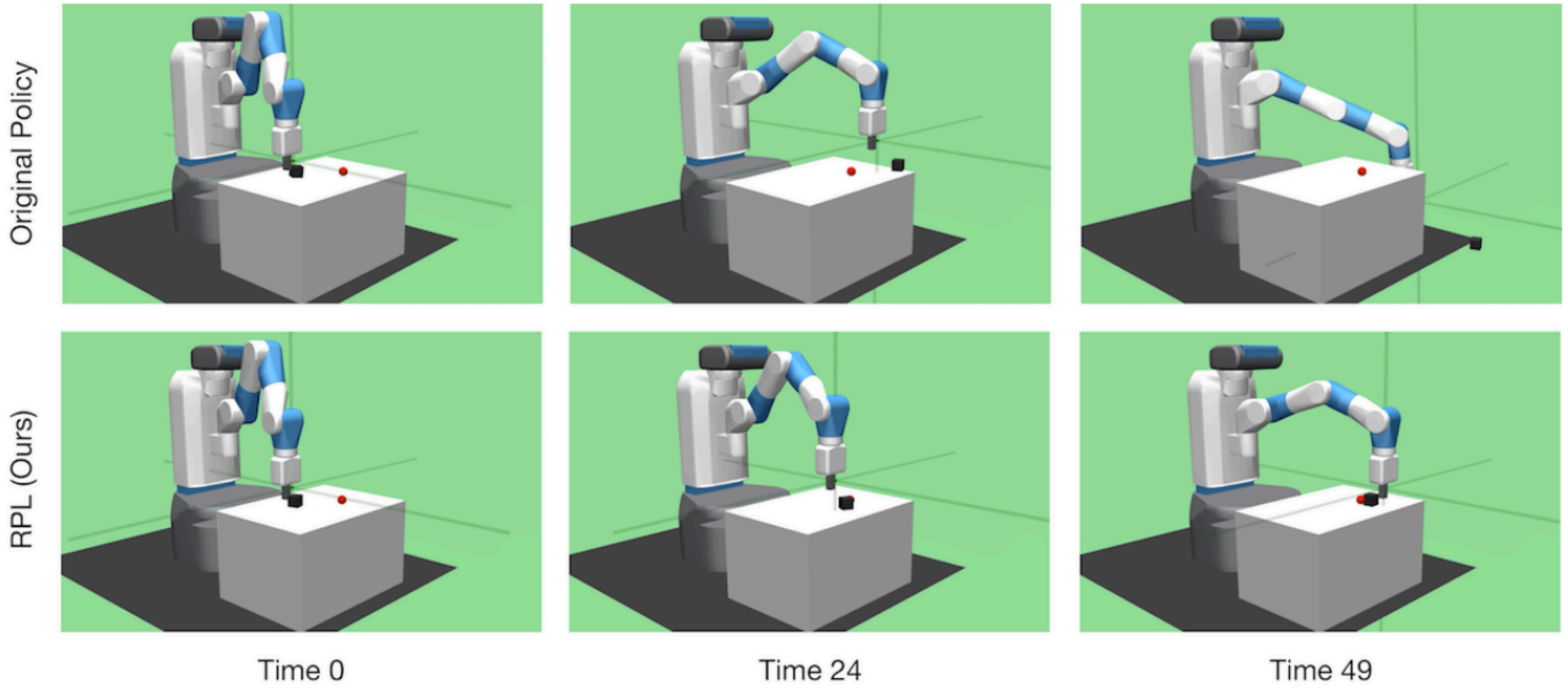
Examples

Residual Policy Learning

Silver, Allen, Tenenbaum, Kaelbling (2018)

TO + residual policy

- Assume good but imperfect controllers are available
- RL from scratch may be data-inefficient or intractable
- Initialize residual policy to output zero
- Train critic alone for “burn in” period while leaving policy fixed



Summary

Architectures Combining RL and TO

Take-home messages

Sequential Approches

TO imitation

If you are satisfied with TO's solutions but want to speed it up

RL-supported TO

If you are satisfied with RL's training time but want to refine policy

Coupled Approches

Coupled TO imitation

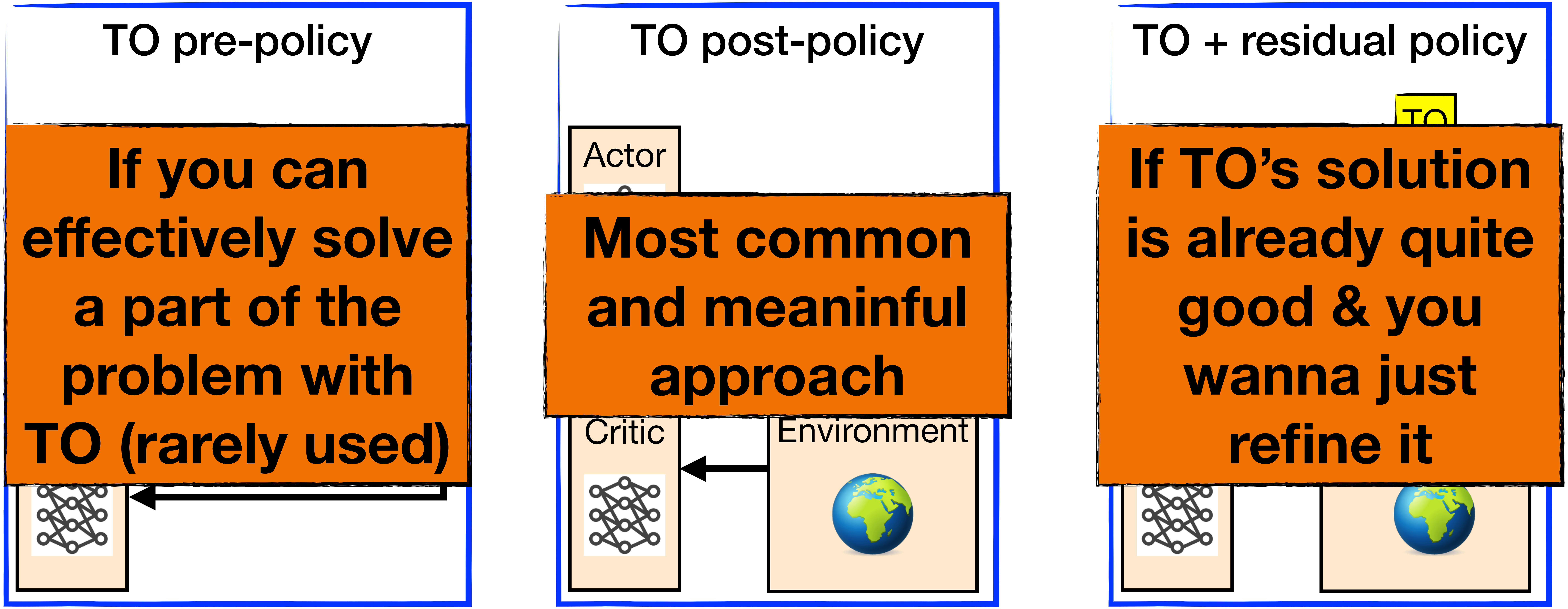
Policy

TO inside RL

If TO's solutions are not good enough & you wanna speed up RL

Where should TO be introduced?

Take-home messages

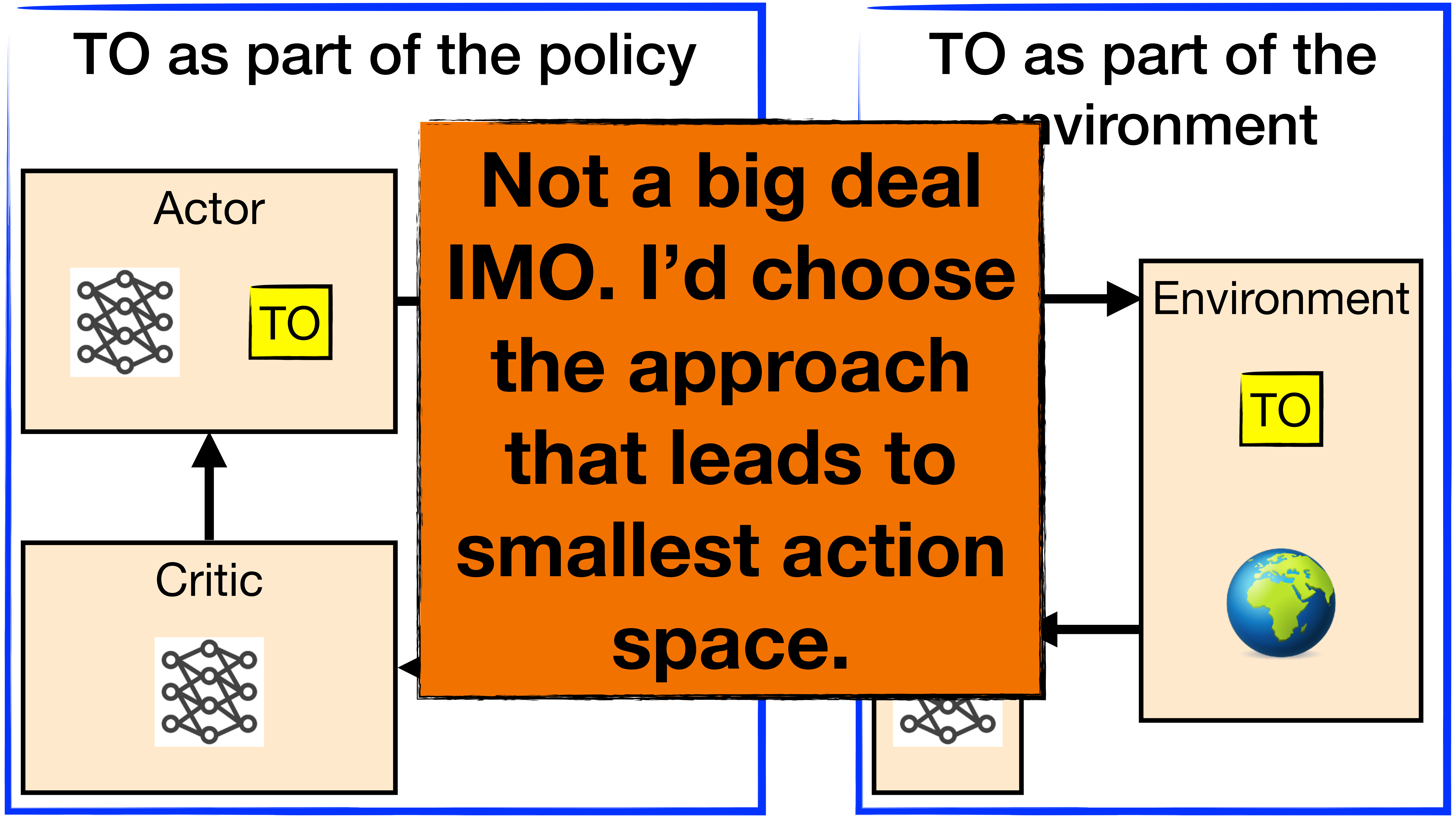


In which block should TO be considered?

Take-home messages

Need to differentiate TO!

Actions are the output of TO



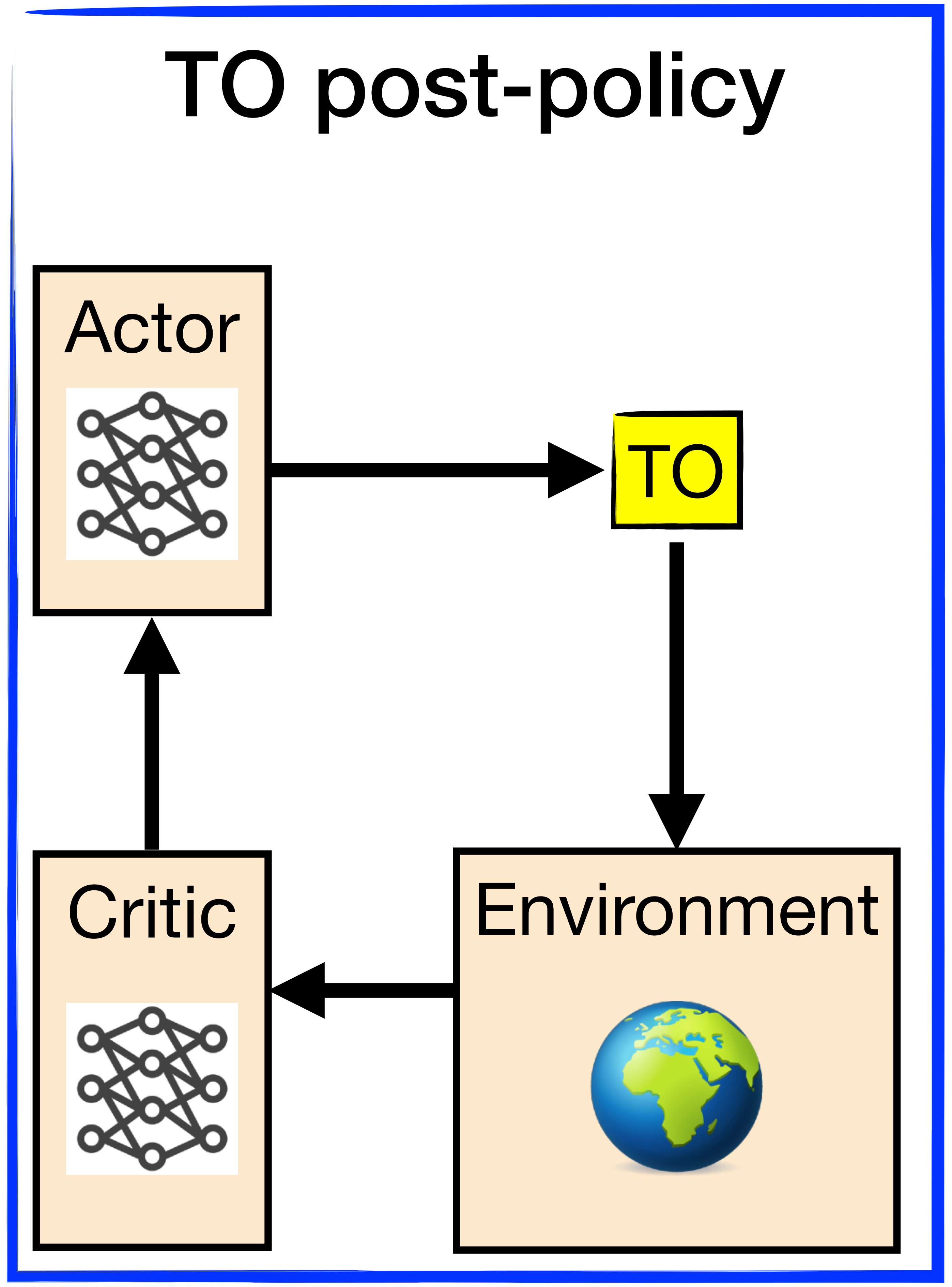
Not a big deal IMO. I'd choose the approach that leads to smallest action space.

No need to differentiate TO!

Actions are the output of the actor policy

What should the policy learn?

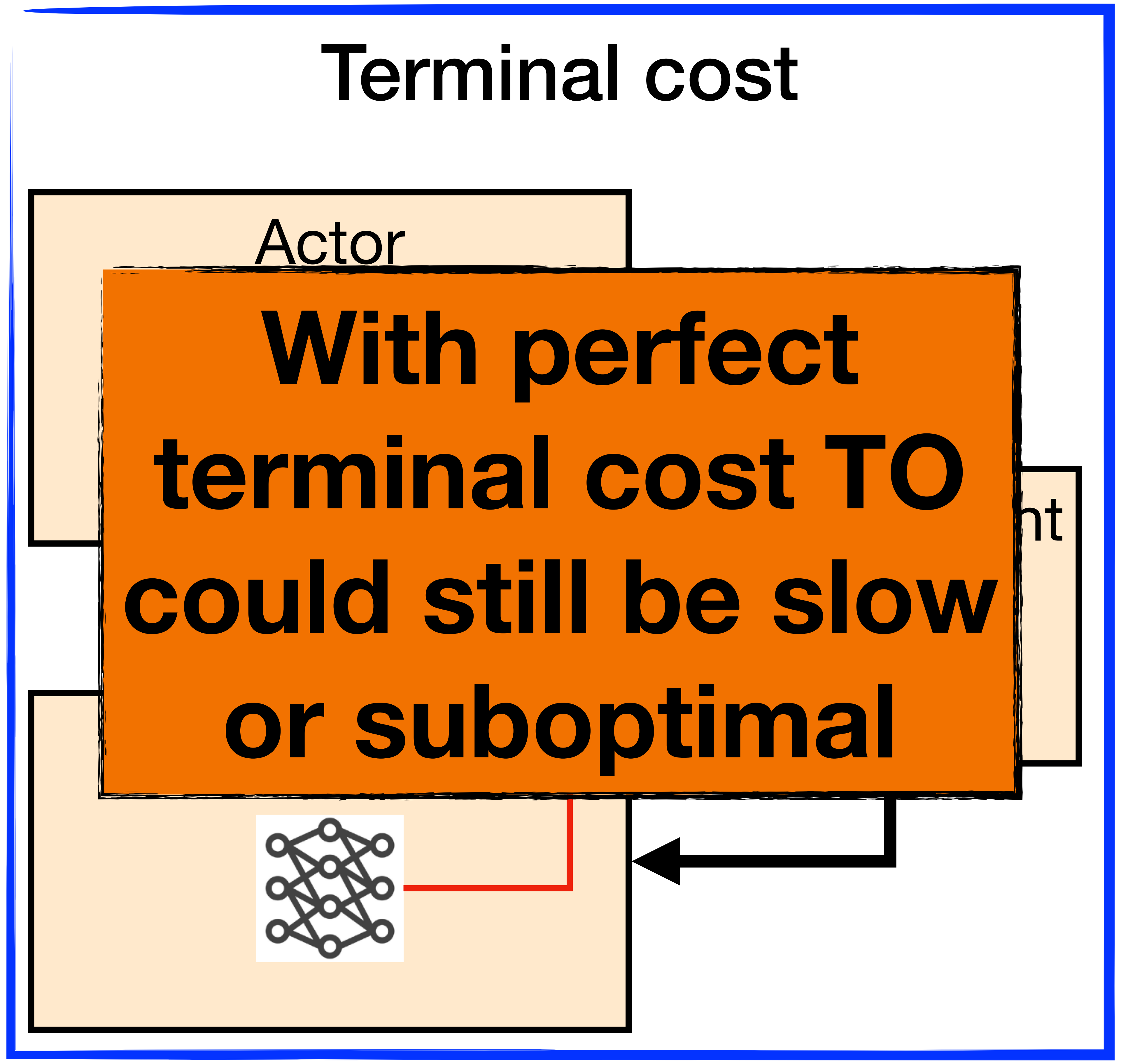
Take-home messages



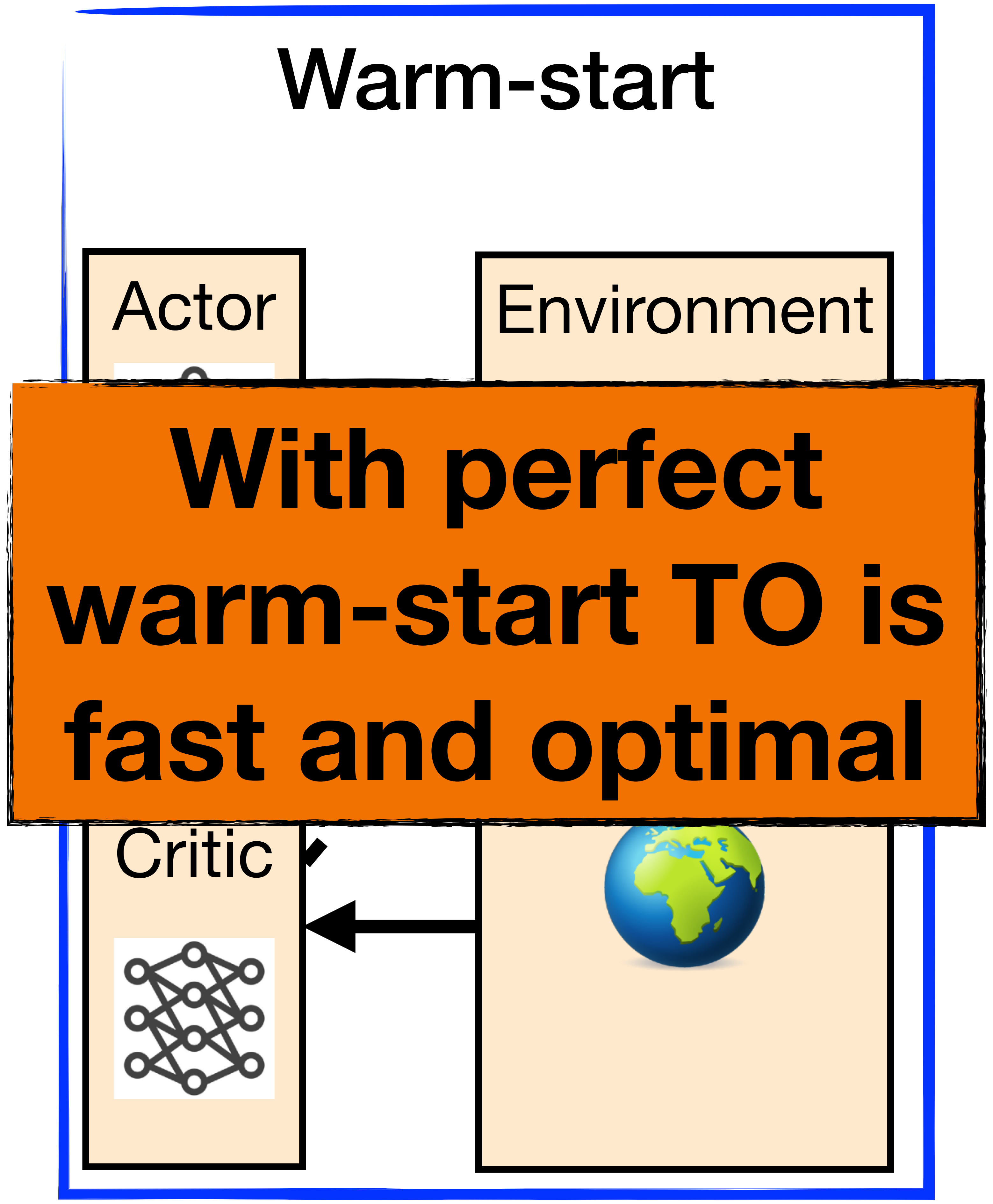
Do TO and RL solve the same problem?	
NO	**YES**
Easy to achieve sensor feedback. Need to solve TO at deployment.	**Best strategy if it can be used. Hard to achieve sensor feedback**

Learning the terminal cost VS warm-start

Take-home messages



One can learn both!



Conclusions

Combining RL and TO

- **Benefits:**
  - Speed up RL and TO
  - Guide TO towards global optimality
  - Exploit sensor feedback
- **Key ideas:**
  - Use dynamics derivatives to guide RL
  - Use Value function as terminal cost to shorten TO's horizon
  - Use policy to warm-start & guide TO

Current challenges:

- Getting the right architecture is fundamental
- Dynamics derivatives are ill-defined in contact-rich tasks
- TO struggles with stochasticity
- Solving TO on GPU is still hard

* UNIVERSITY OF TRENTO

** UNIVERSITY OF NOTRE DAME

*** ENS | PSL

CACTO: Continuous Actor-Critic with Trajectory Optimization

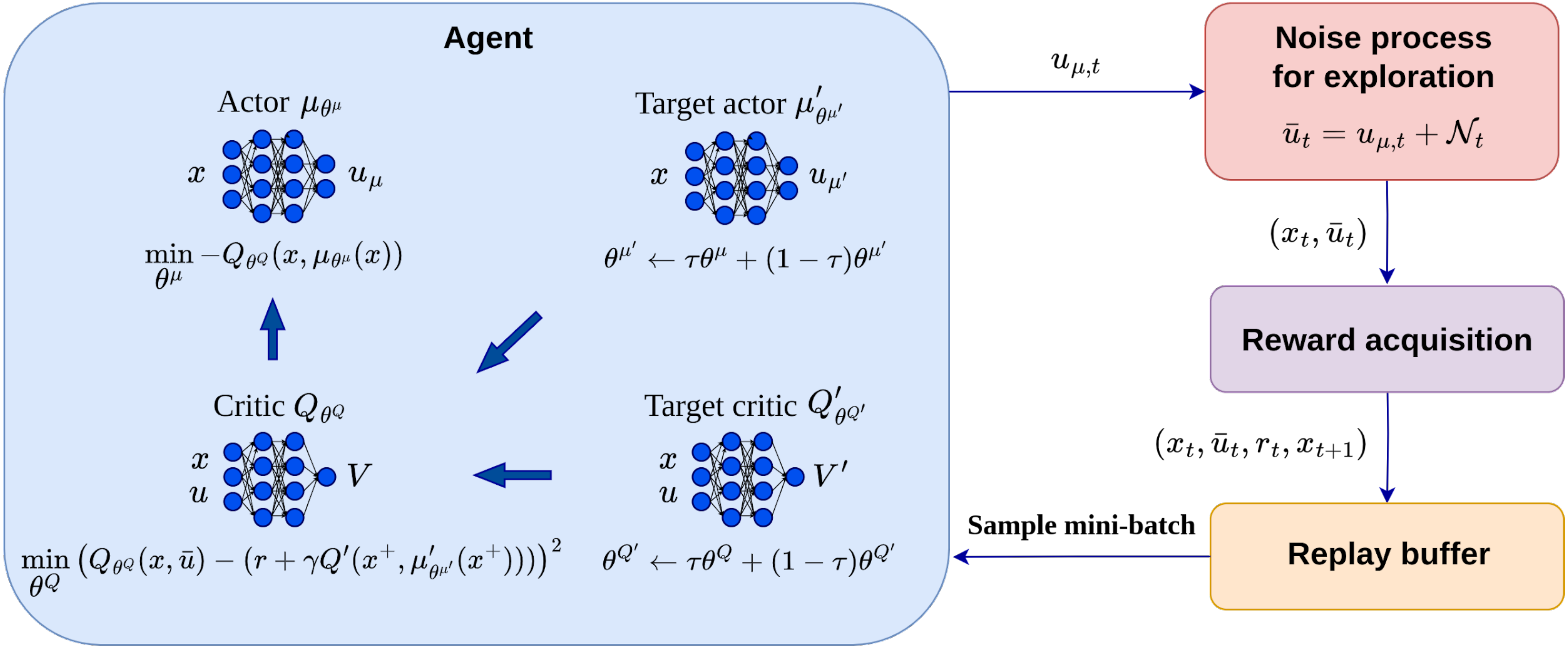
Gianluigi Grandesso*,
Elisa Alboni*,
Gastone Rosati Papini*,
Patrick Wensing,**
Justin Carpentier*,**
Andrea Del Prete*





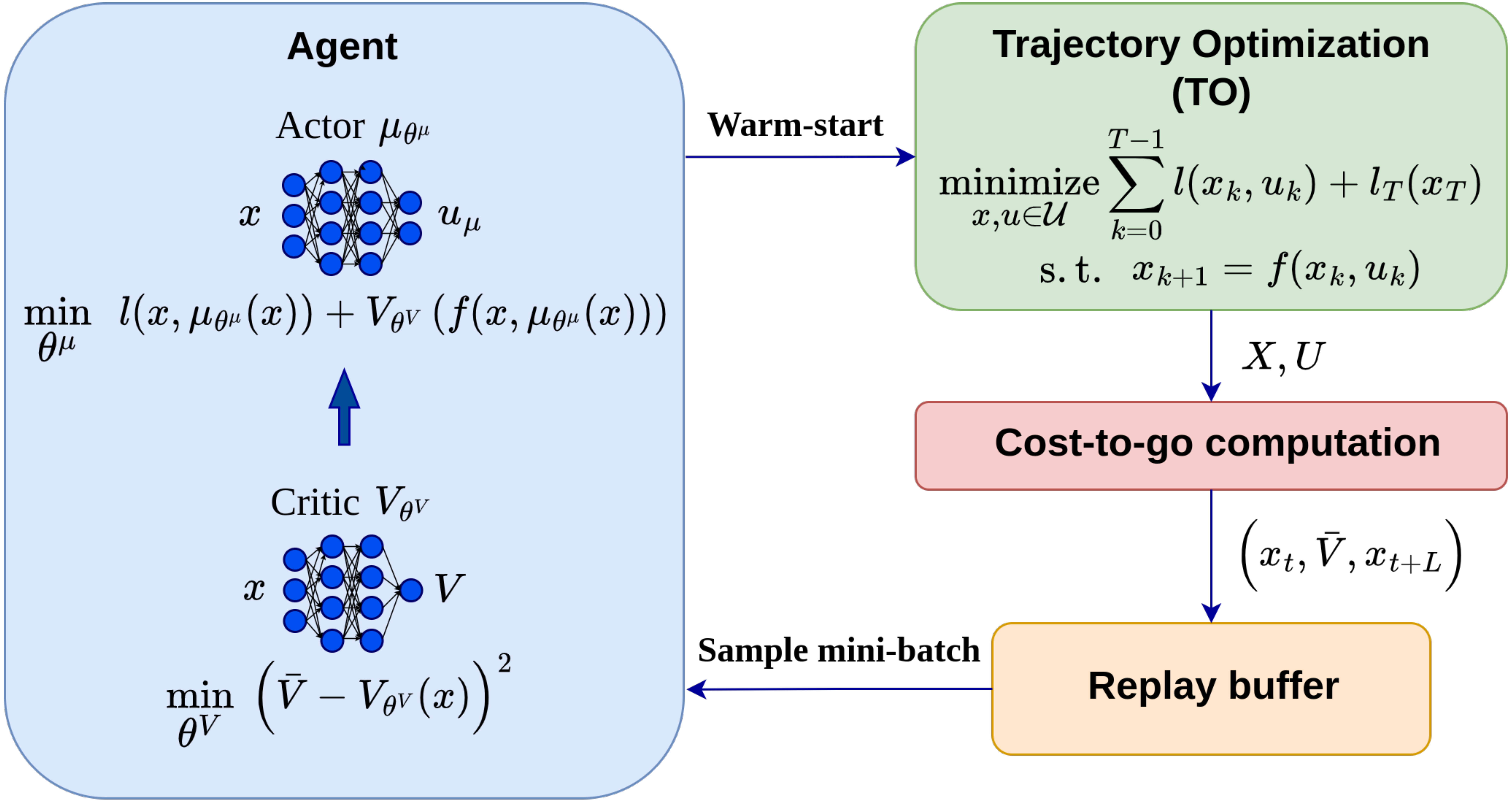
[1] (2023) CACTO: Continuous Actor-Critic With Trajectory Optimization - Towards Global Optimality. IEEE RA-L
[2] (2024) CACTO-SL: Using Sobolev Learning to improve Continuous Actor-Critic with Trajectory Optimization. In L4DC

Deep Deterministic Policy Gradient (DDPG)



Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D. (2015). Continuous control with deep reinforcement learning. In Foundations and Trends in Machine Learning

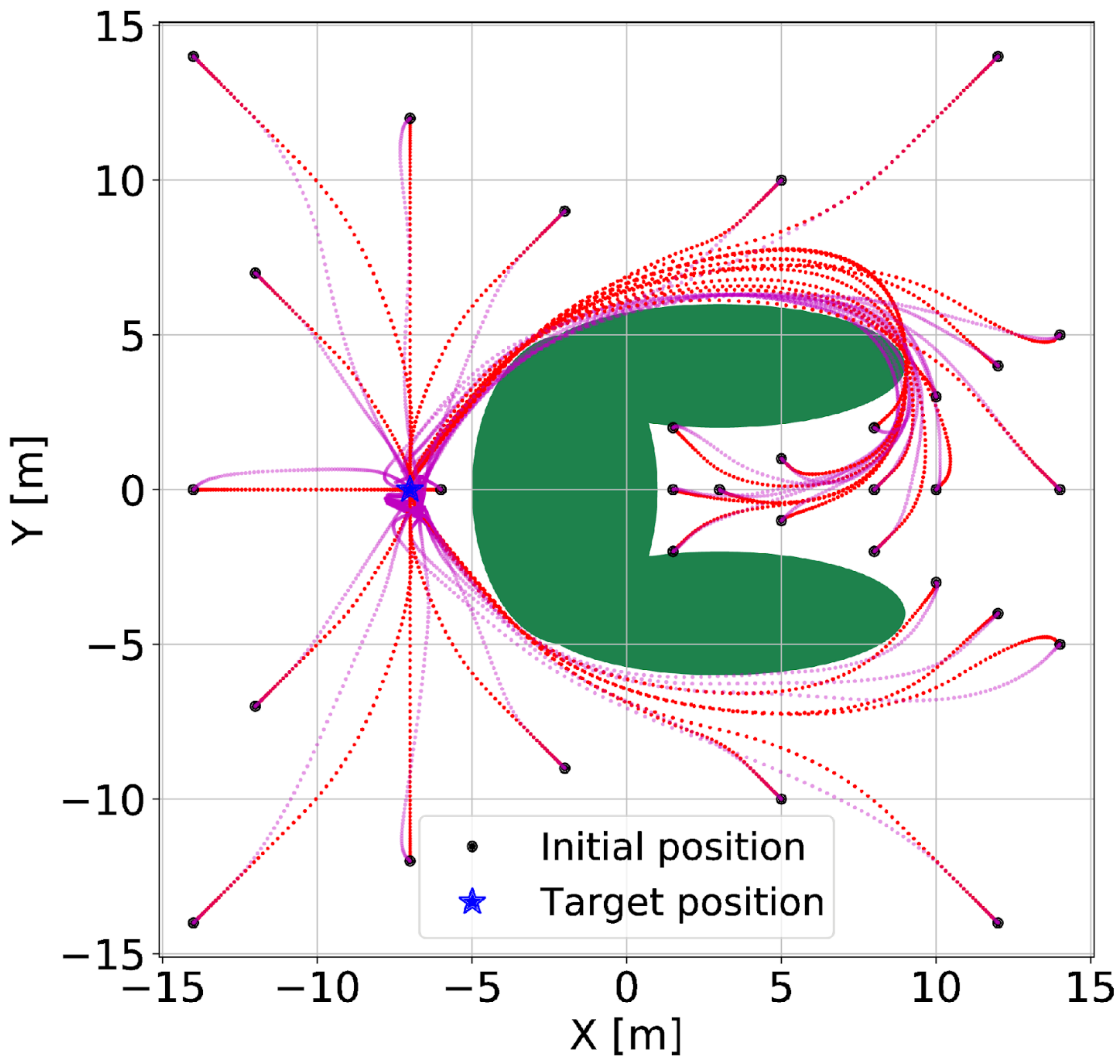
CACTO



[1] Grandesso, Alboni, Rosati Papini, Wensing, Del Prete (2023). CACTO: Continuous Actor-Critic With Trajectory Optimization - Towards Global Optimality. IEEE Robotics and Automation Letters

Results

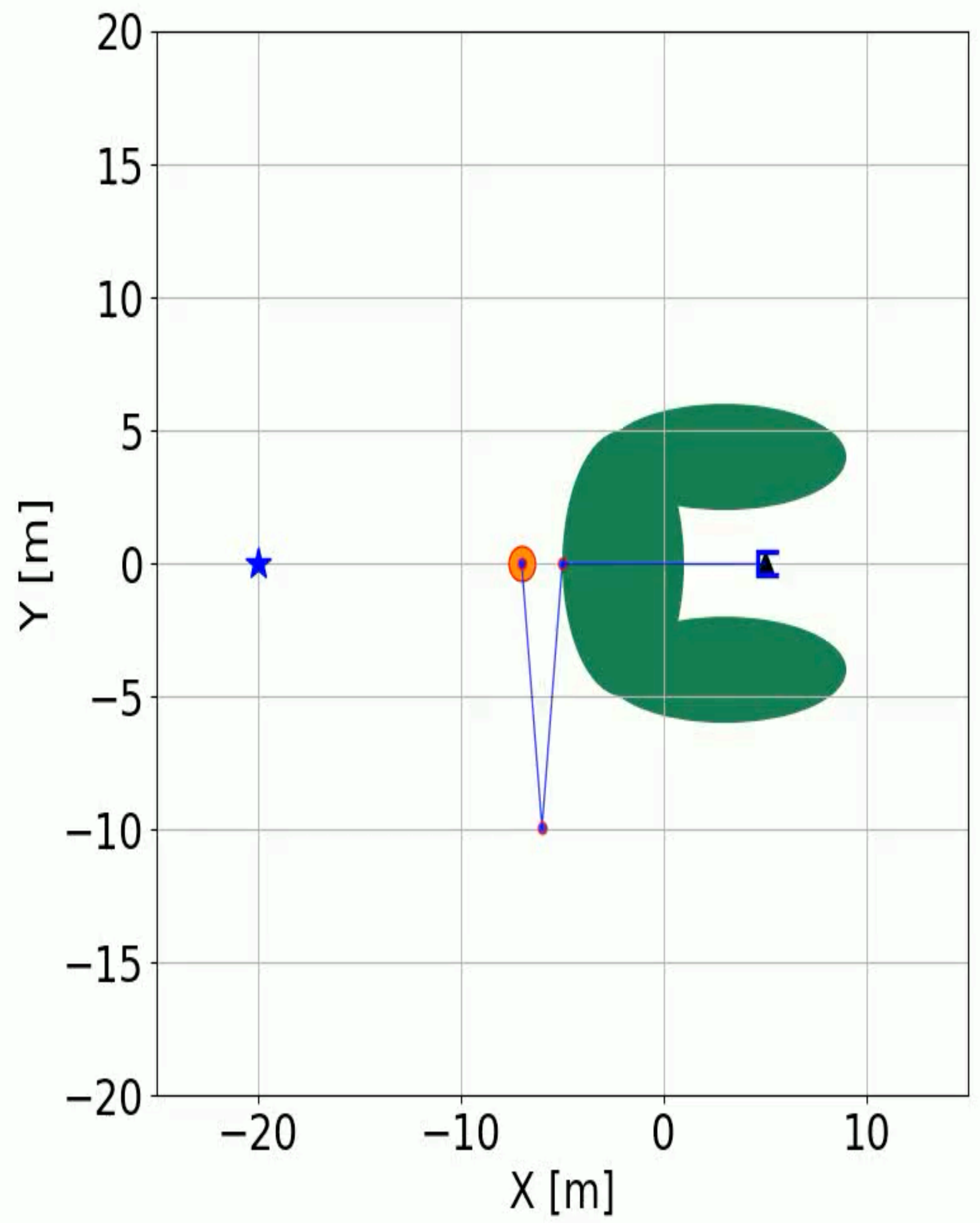
Task: find shortest path to target using low control effort and avoiding obstacles



Systems: 2D single/double integrator, 6D car model, 3-joint manipulator

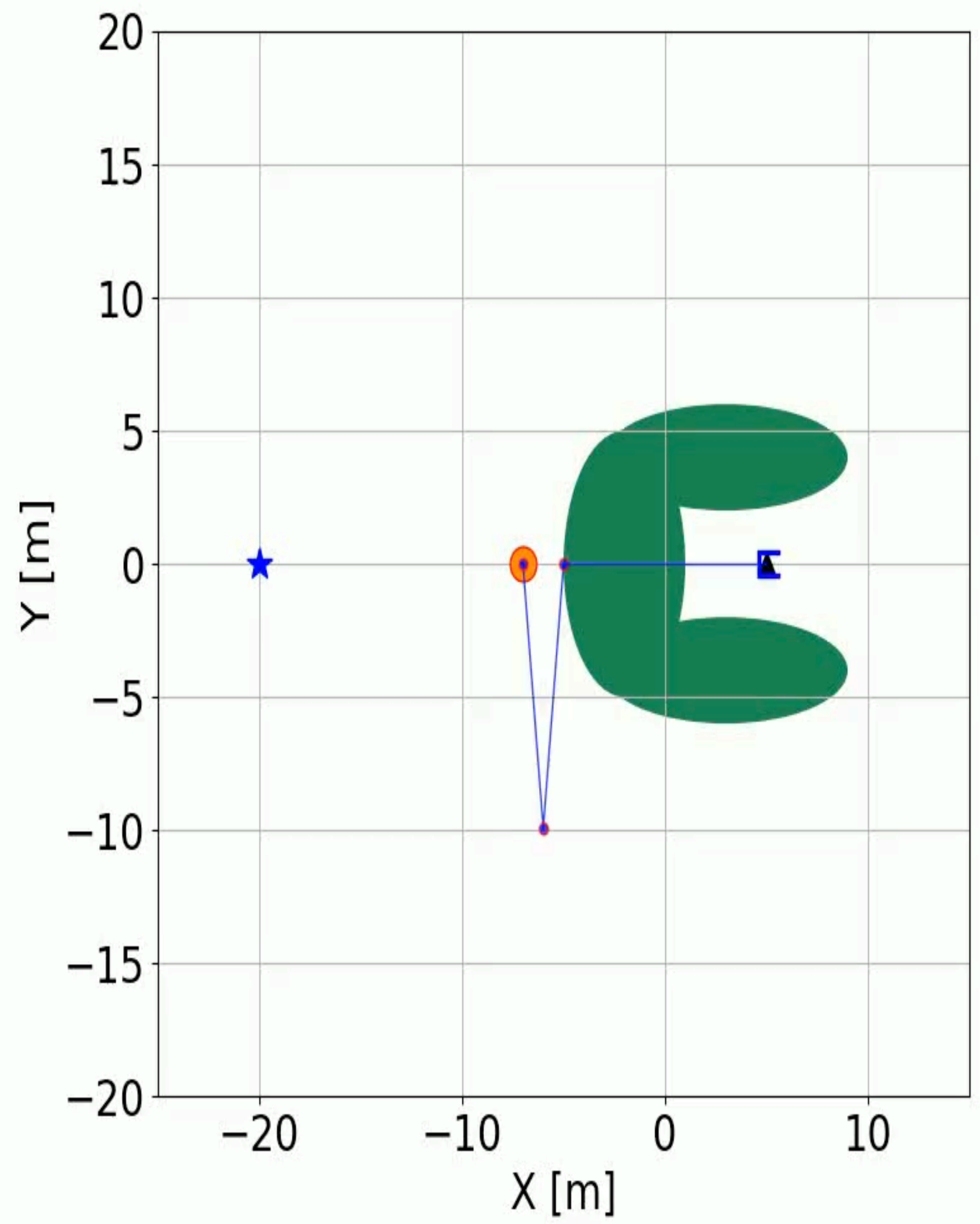
Results: 3-DoF Manipulator

Initial Conditions
warm-start



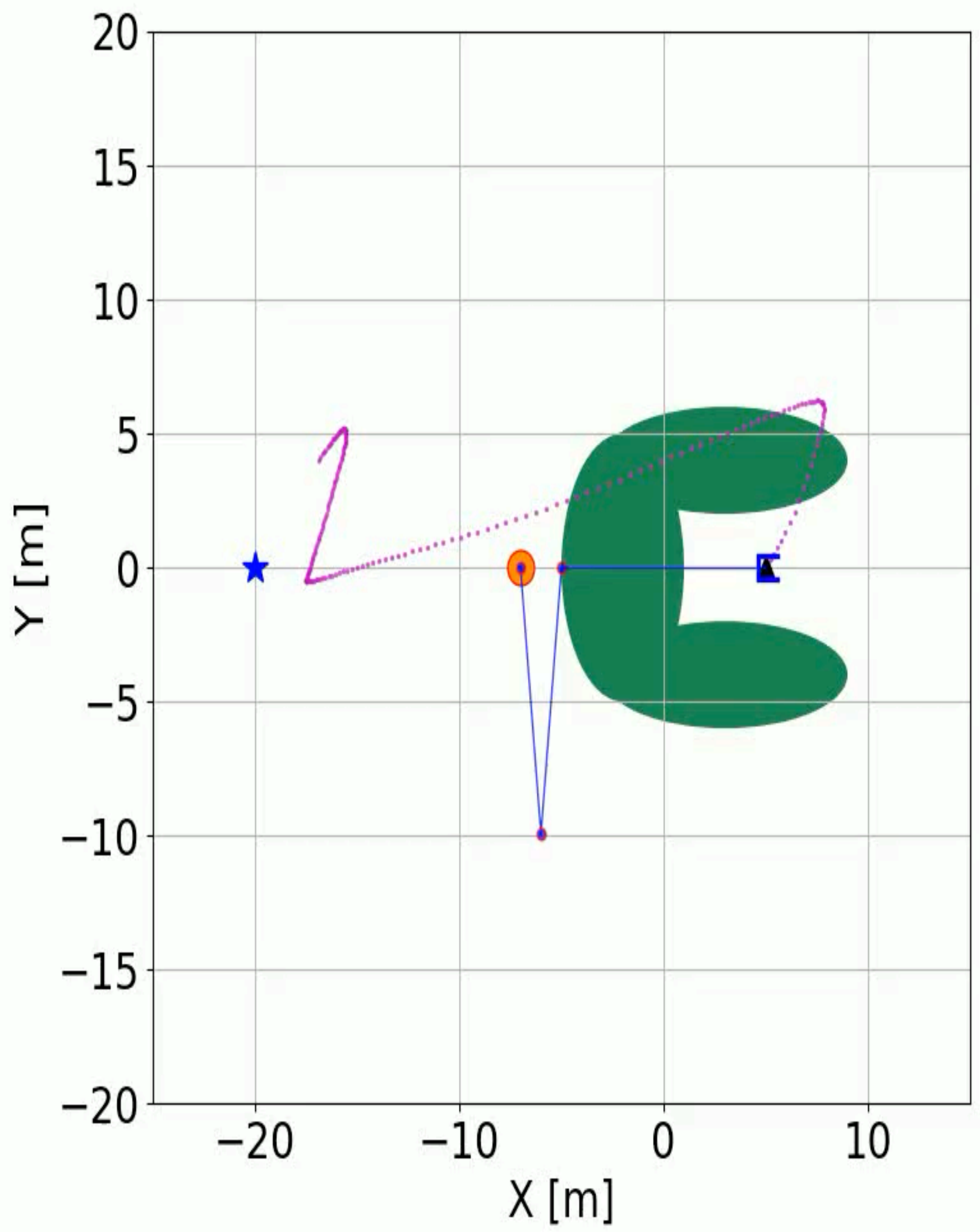
Cost = 70800

Random
warm-start



Cost = 88647

CACTO
warm-start



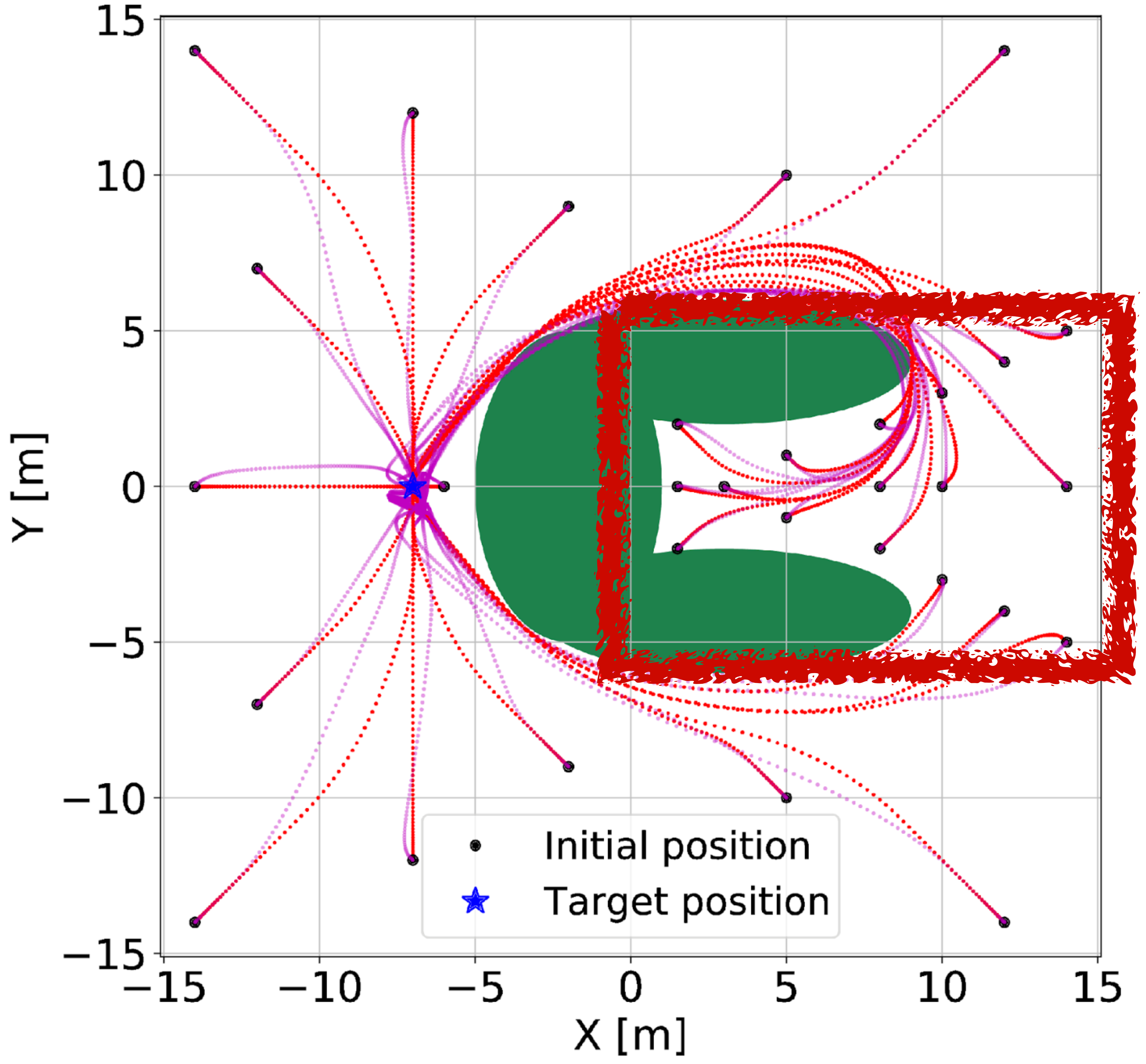
Cost = -145875

Comparison: CACTO vs TO

% of times TO finds better solution if warm-started with CACTO rather than:

- Random values
- Initial conditions (ICS) for states, zero for other variables

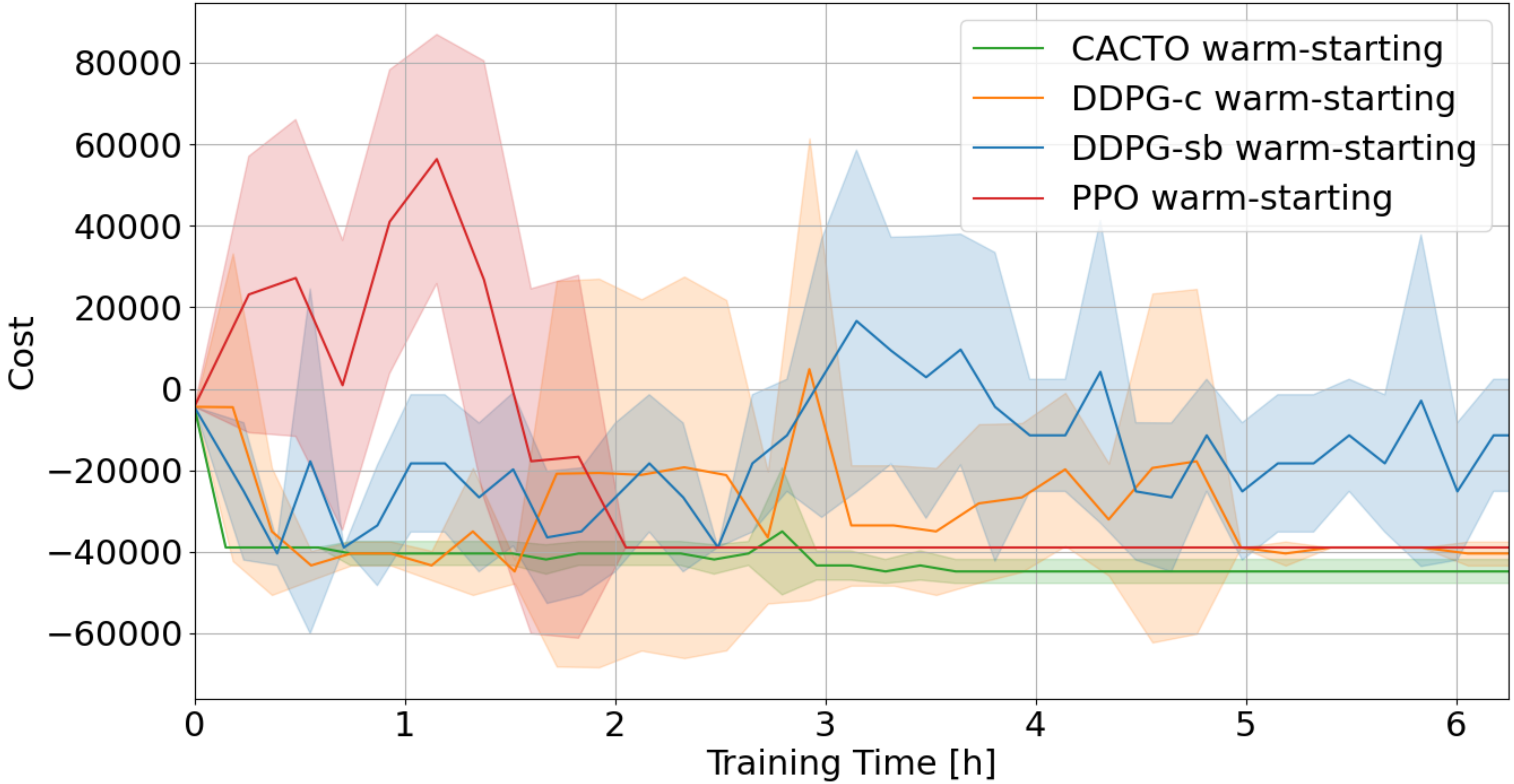
System	Hard Region	
	CACTO < (≤) Random	CACTO < (≤) ICS
2D Single Integrator	**99.1%** (99.1%)	**92%** (99.1%)
2D Double Integrator	**99.9%** (99.9%)	**92%** (99.1%)
Car	**100%** (100%)	**92.9%** (100%)
Manipulator	**87.5%** (87.5%)	**100%** (100%)



2D Double Integrator - CACTO warm-start

Comparison: CACTO, DDPG, PPO

Mean cost + std. dev. (across 5 runs) found by TO warm-started with different policies

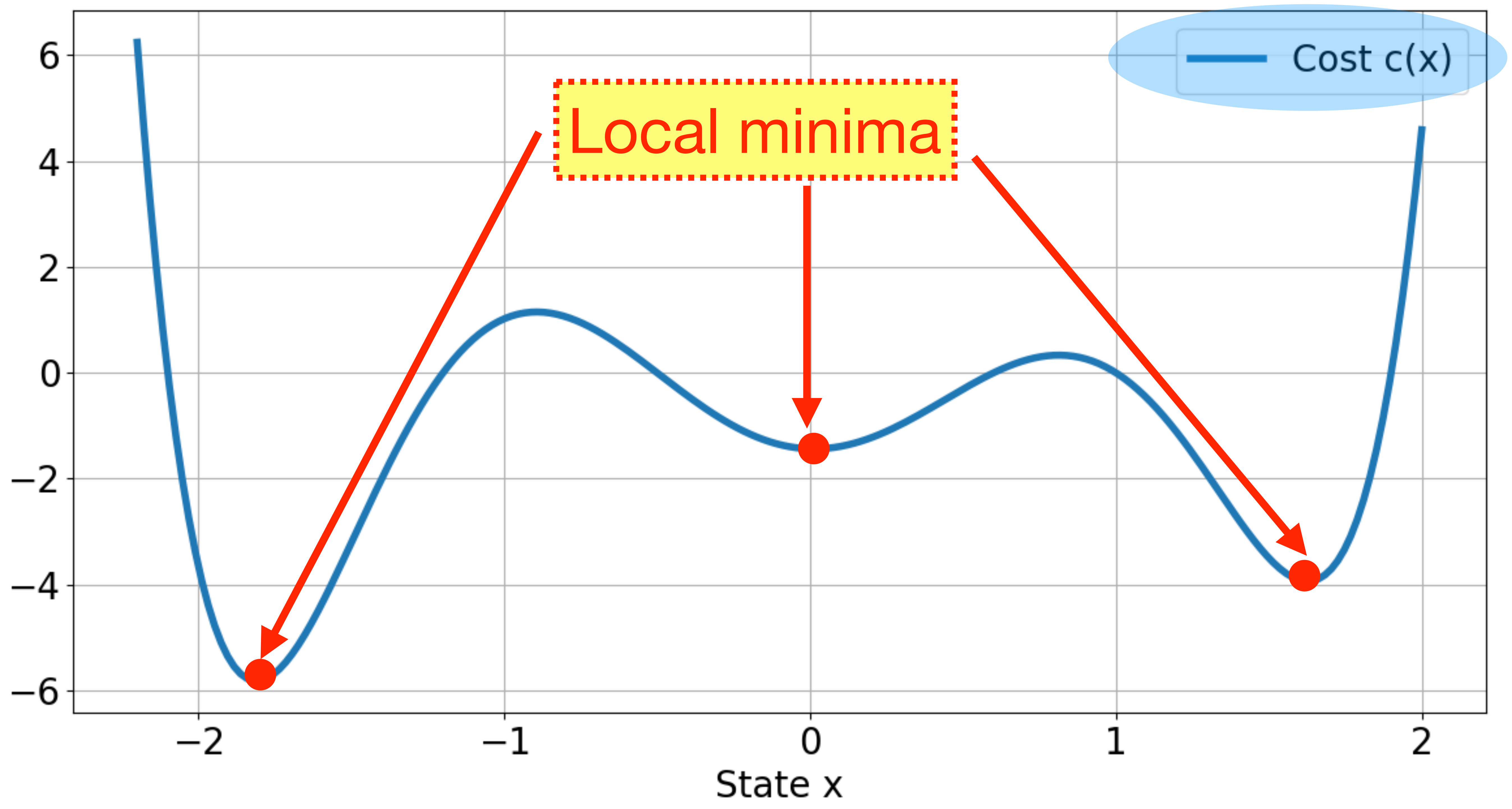


1D Example

$$\underset{X,U}{\text{minimize}} \quad \sum_{k=0}^{T-1} \left[c(x_k) + w_u ||u_k||^2 \right] + c(x_T)$$

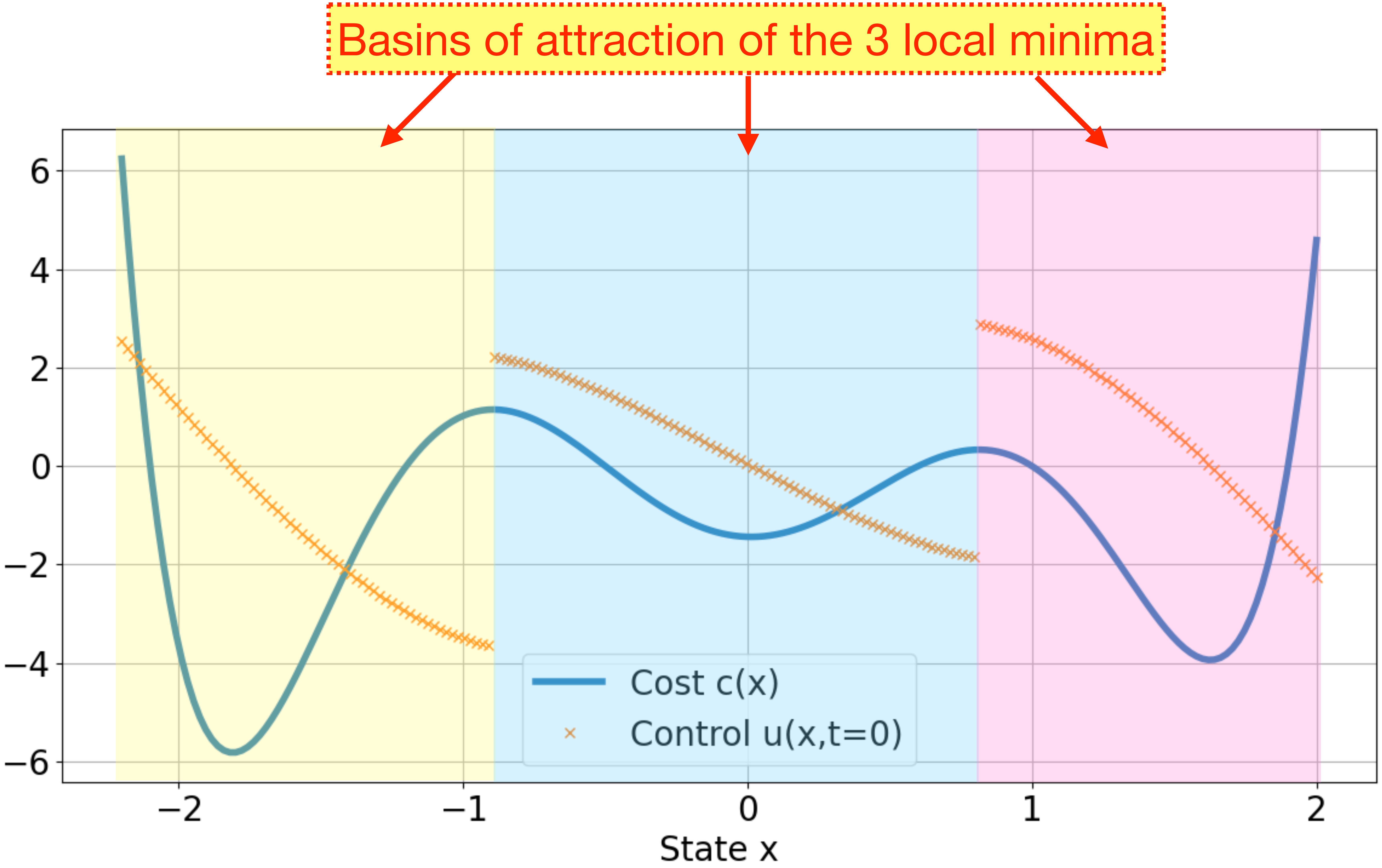
$$\text{subject to} \quad x_{k+1} = x_k + \Delta t \, u_k \quad \forall \, k = 0, \ldots, T-1$$

$$x_0 = x_{init}$$



Trajectory Optimization

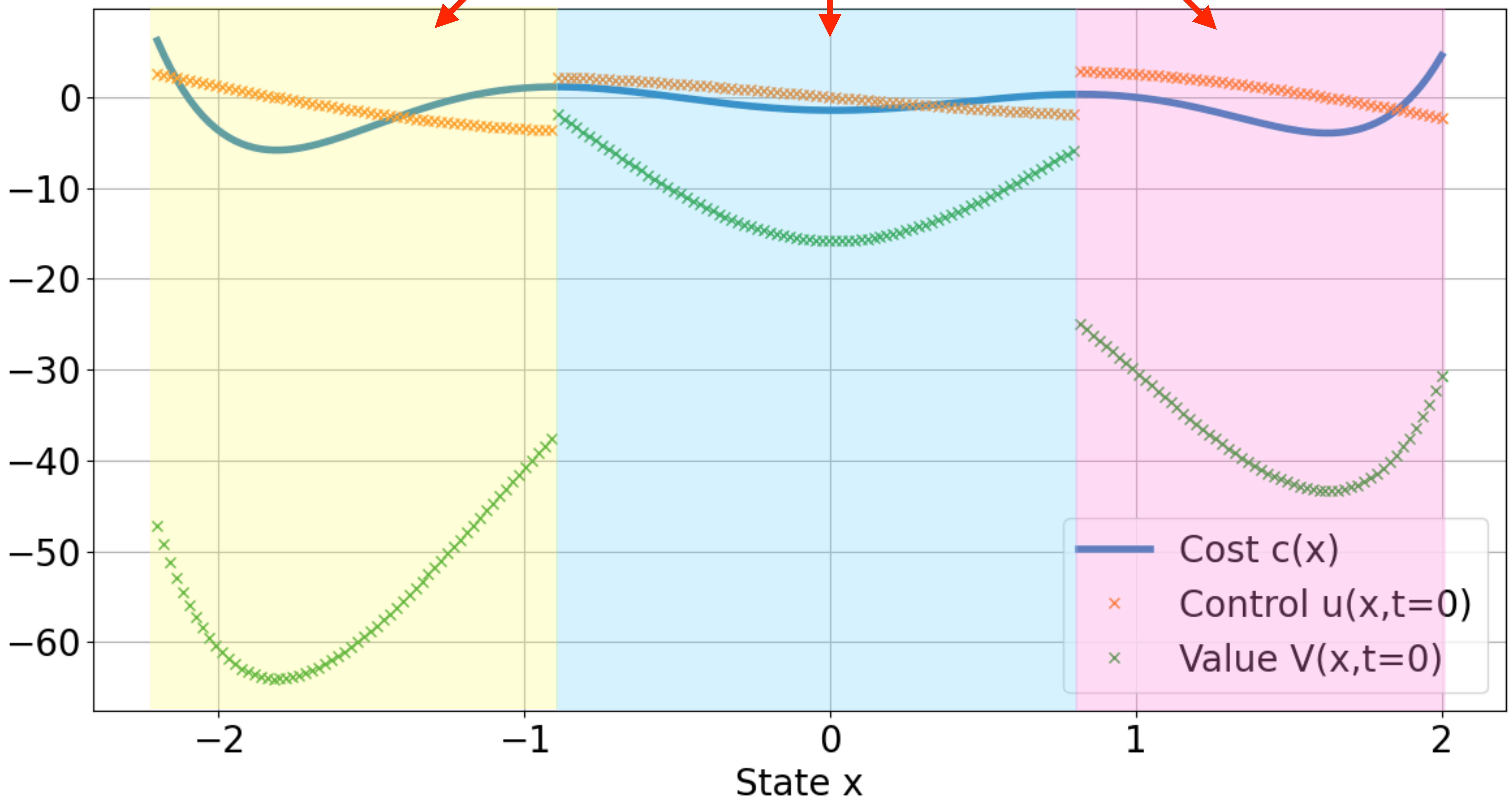
With naive initial guess



Trajectory Optimization

With naive initial guess

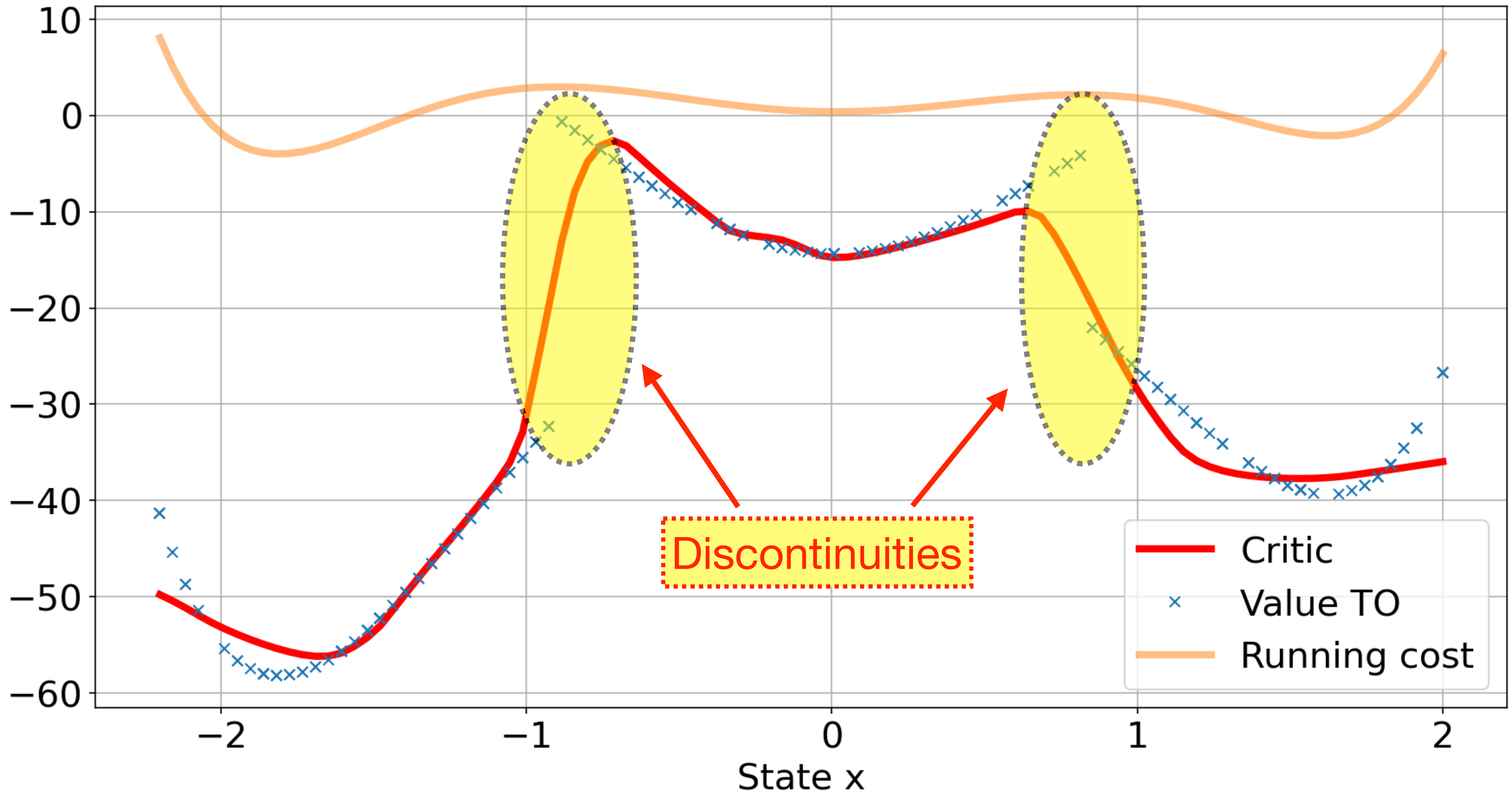
Basins of attraction of the 3 local minima



First Iteration

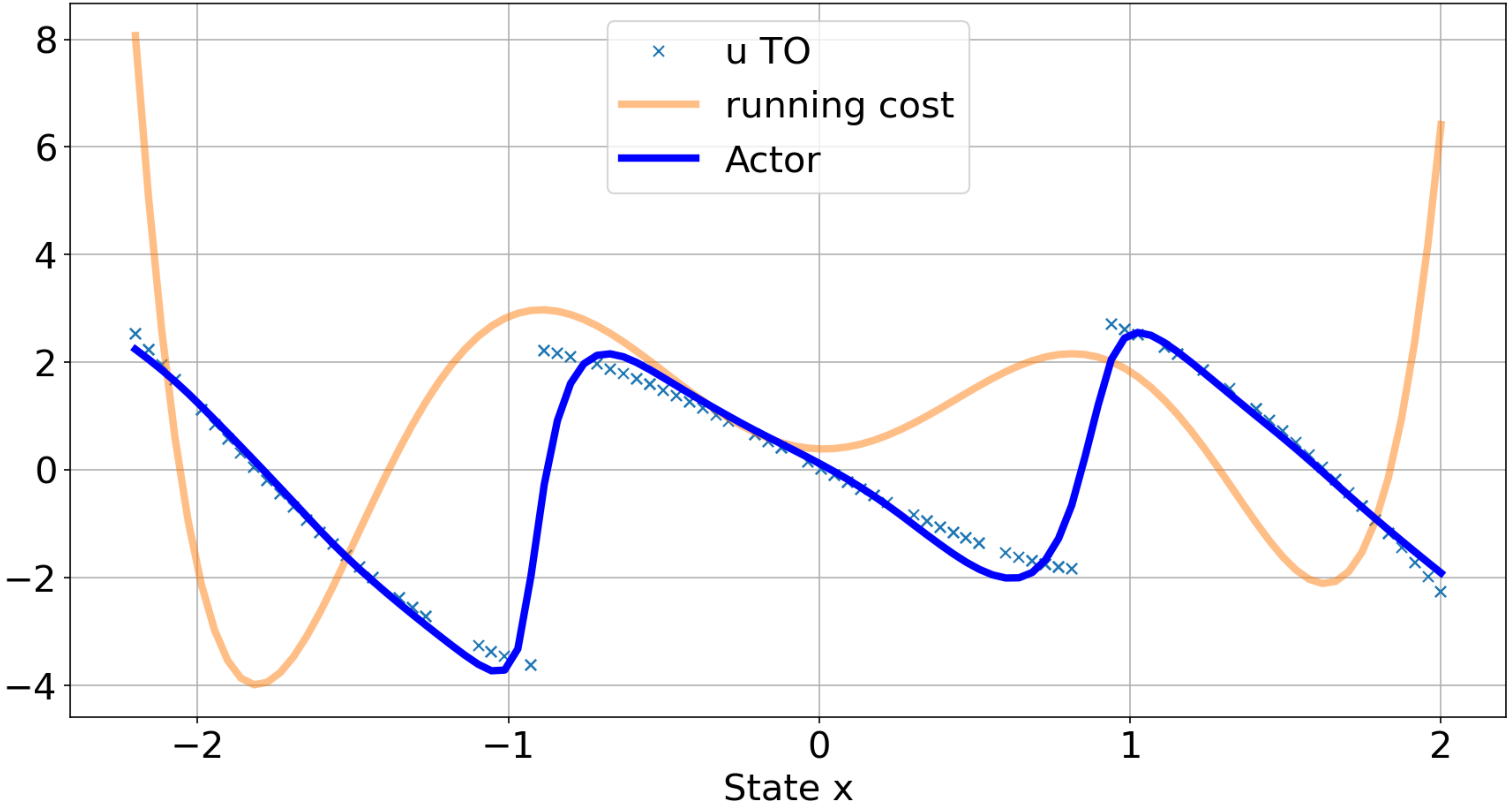
Learning the critic

The Value function is discontinuous so the network approximates it.



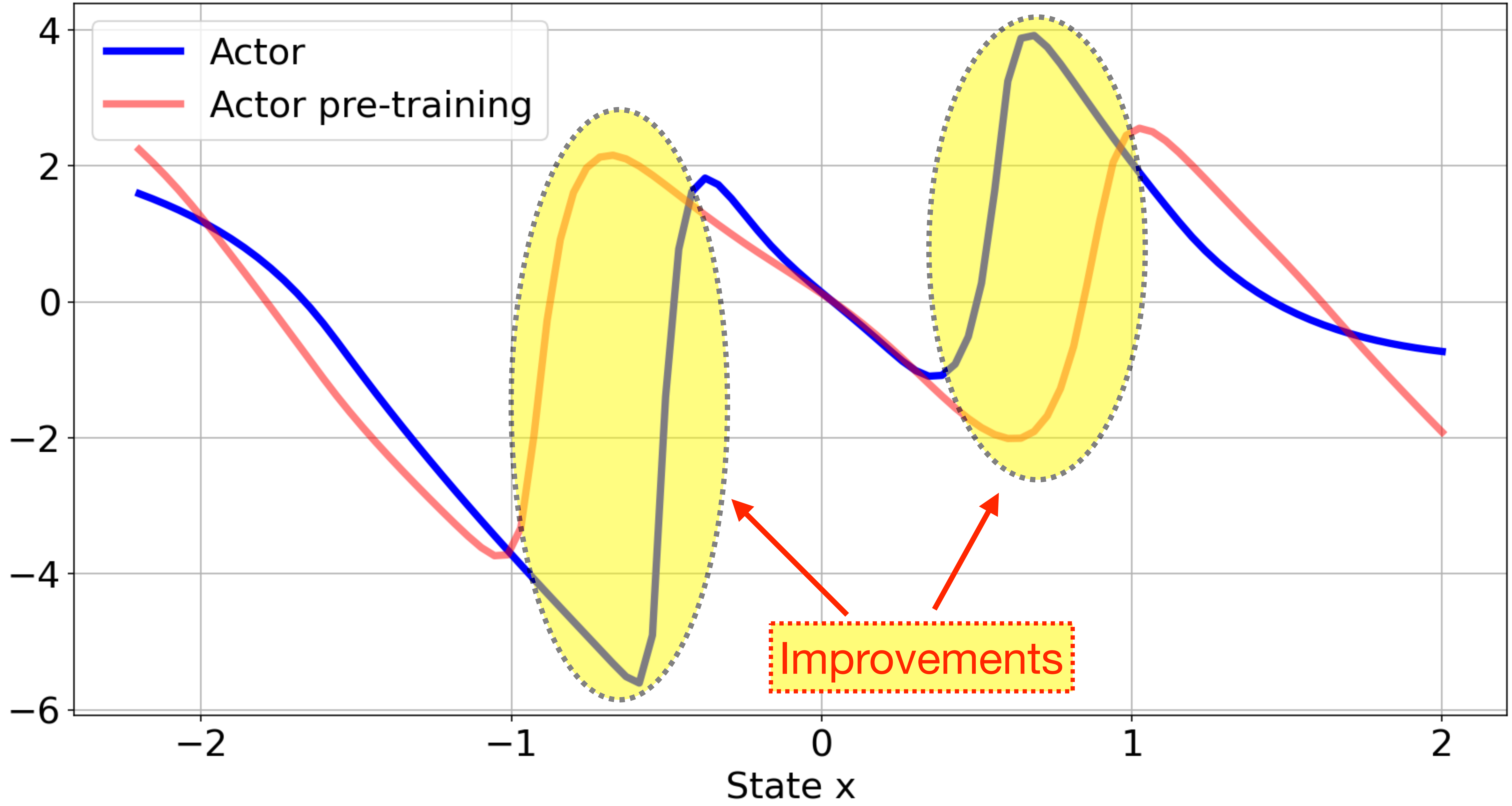
Supervised Learning of the actor

At the first iteration we pre-train the actor to imitate the control inputs of TO.



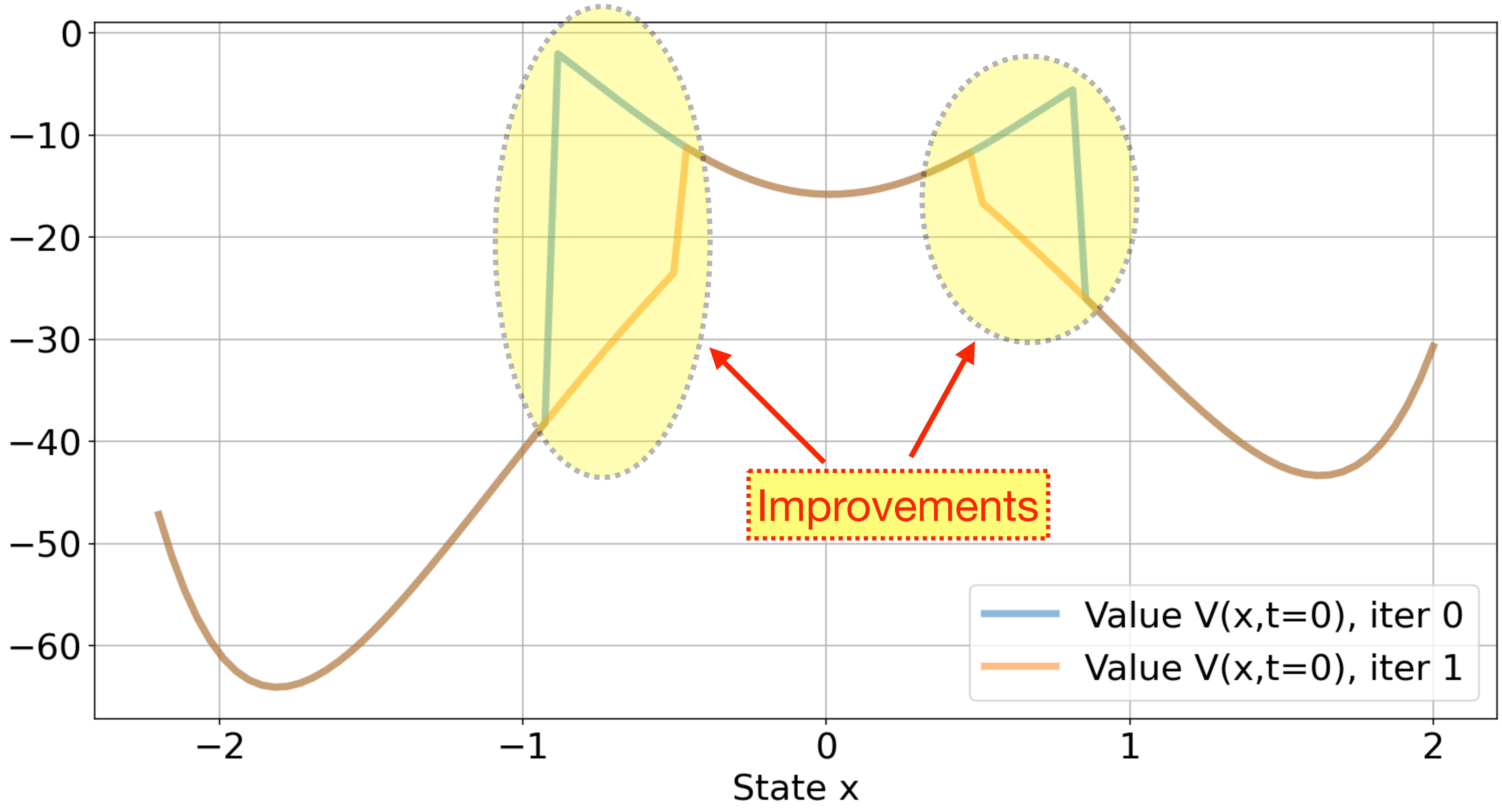
Learning the actor minimizing Q

We improve the actor by minimizing the Q function



Using the actor to warm-start TO

TO improves thanks to the initial guess of the actor



CACTO - Conclusions

- TO guides the RL exploration making it sample efficient
- RL policy guides TO towards globally optimal solutions
- Global convergence proof for discrete-space version of CACTO

Recent extensions

- Improve data efficiency leveraging derivative of Value function [2]
- Bias initial episode state to improve data efficiency (unpublished)
- Parallelize on GPUs (unpublished)

Future work

- Handle non-differentiable dynamics

[2] Alboni, Grandesso, Rosati Papini, Carpentier, Del Prete (2024). CACTO-SL: Using Sobolev Learning to improve Continuous Actor-Critic with Trajectory Optimization. In Learning for Dynamics and Control Conference (L4DC)