Safe and Efficient Reinforcement Learning

Combining learning and trajectory optimization

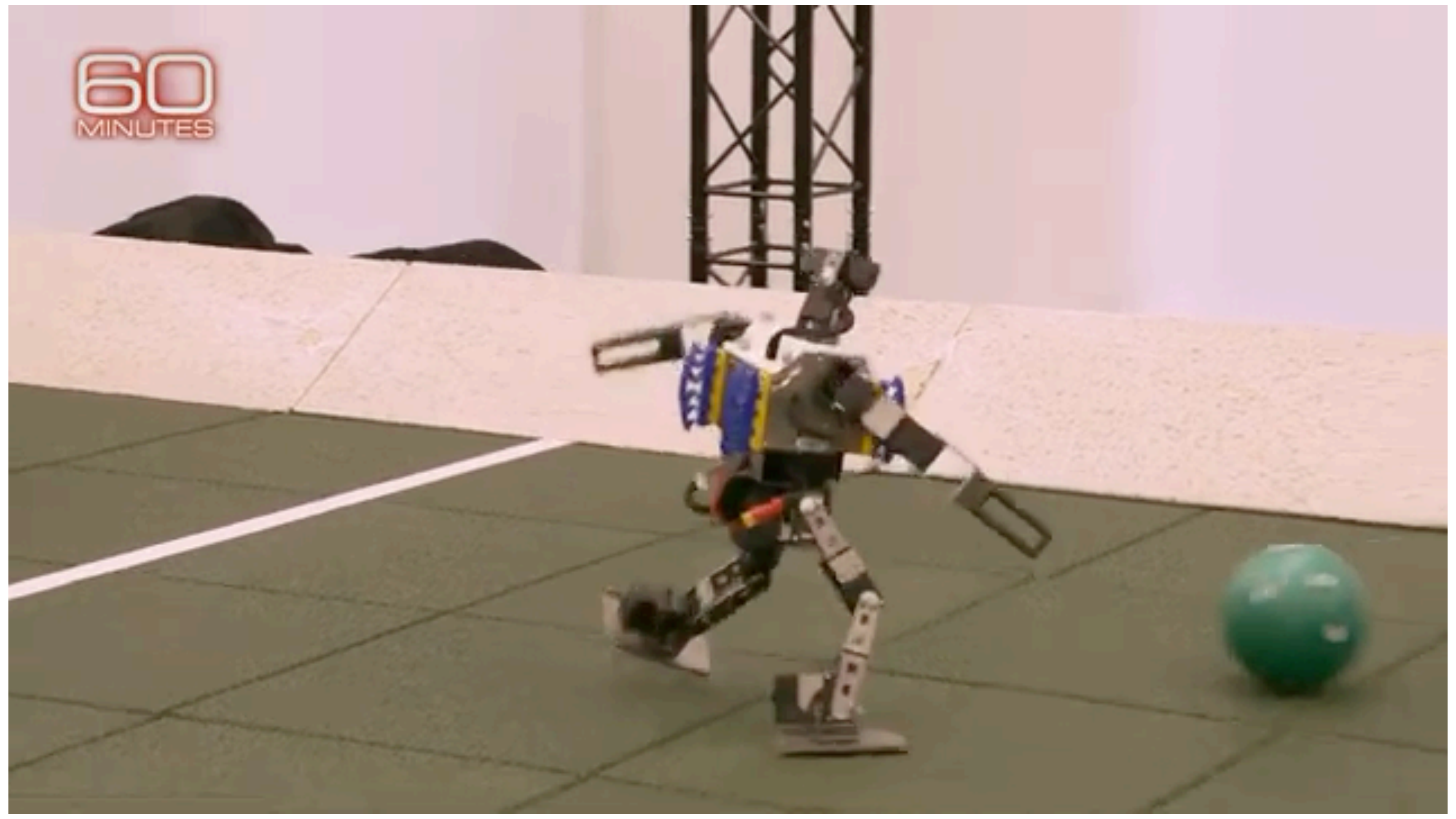
Andrea Del Prete

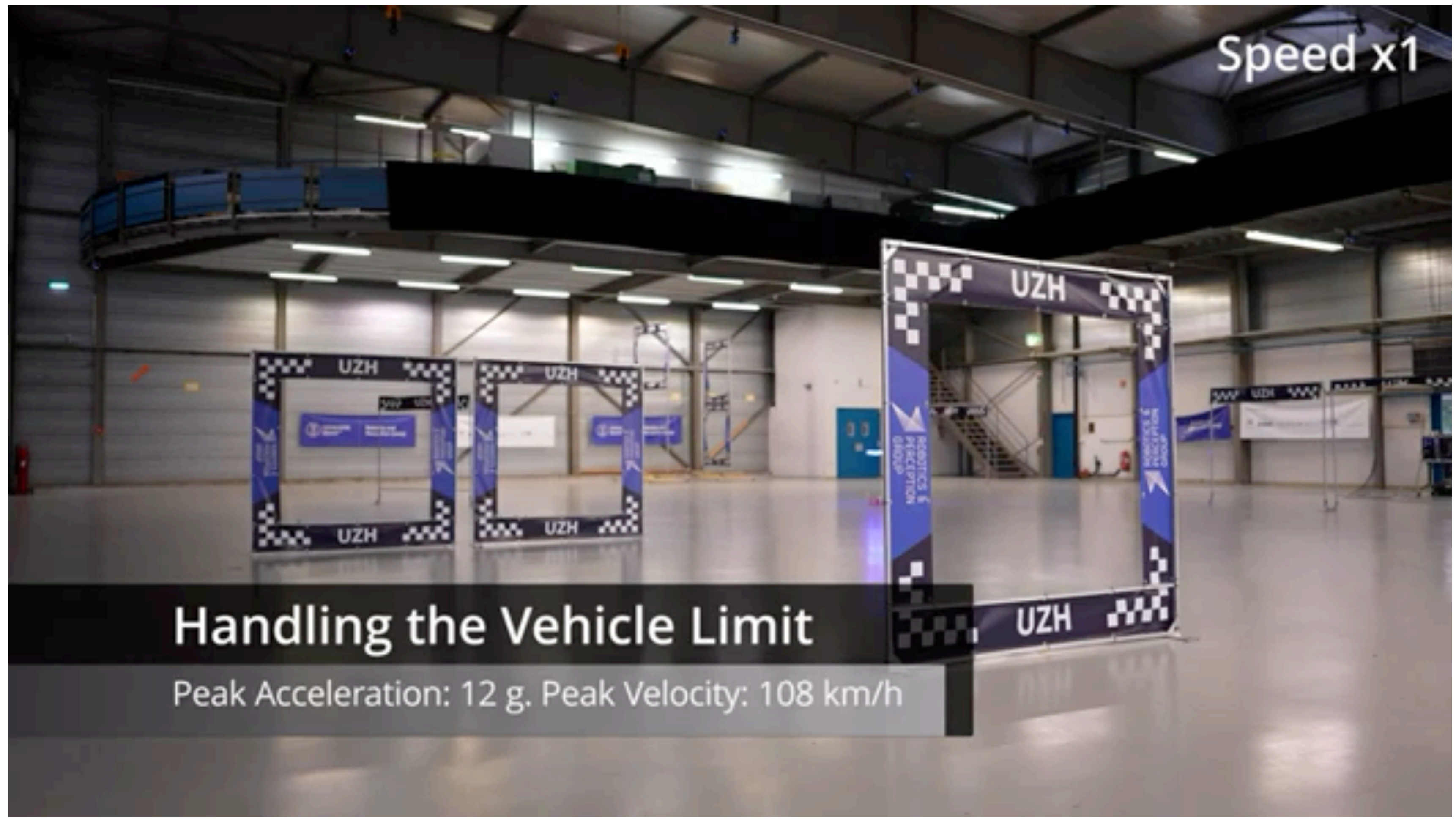
UNIVERSITY OF TRENTO

Is there anything Reinforcement Learning can't do?

Lee, Hwangbo, Wellhausen, Koltun, Hutter (2020). Learning quadrupedal locomotion over challenging terrain. Science Robotics

Haarnoja, T., Moran, B., Lever, G., Huang, S. H., Tirumala, D., Wulfmeier, M., … Heess, N. (2023). Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning





Song, Romero, Müller, Koltun, Scaramuzza, (2023). Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. Science Robotics

The issues with RL

My two cents

Poor efficiency

- Data efficiency
- Energy efficiency
- Time efficiency

Poor safety

- No explicit constraints
- No guarantees
- Safety-critical applications

Can we use ideas from Trajectory Optimization to make RL safe and efficient?

Safe and Efficient Reinforcement Learning

Combining learning and trajectory optimization

Andrea Del Prete

UNIVERSITY OF TRENTO

* 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

Reinforcement Learning ~~VS~~ Trajectory Optimization
WITH?

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

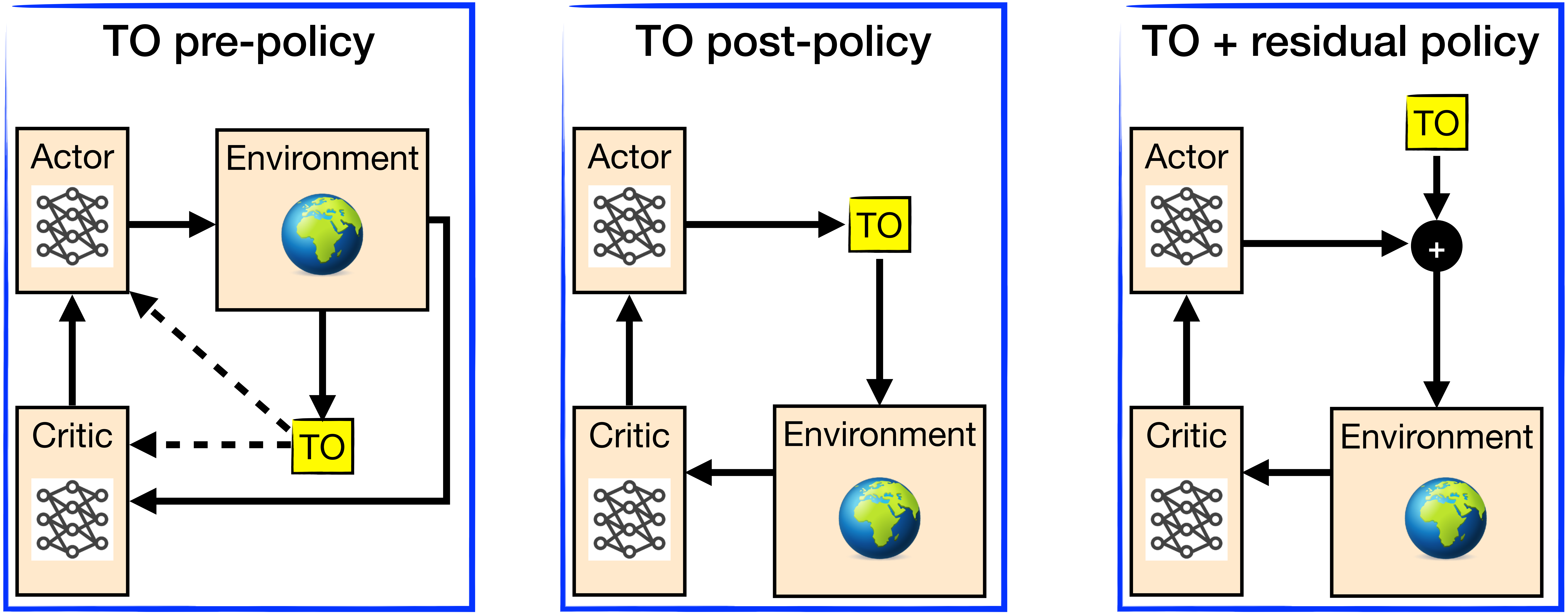
Reinforcement Learning

- \+ 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

Where should TO be introduced?

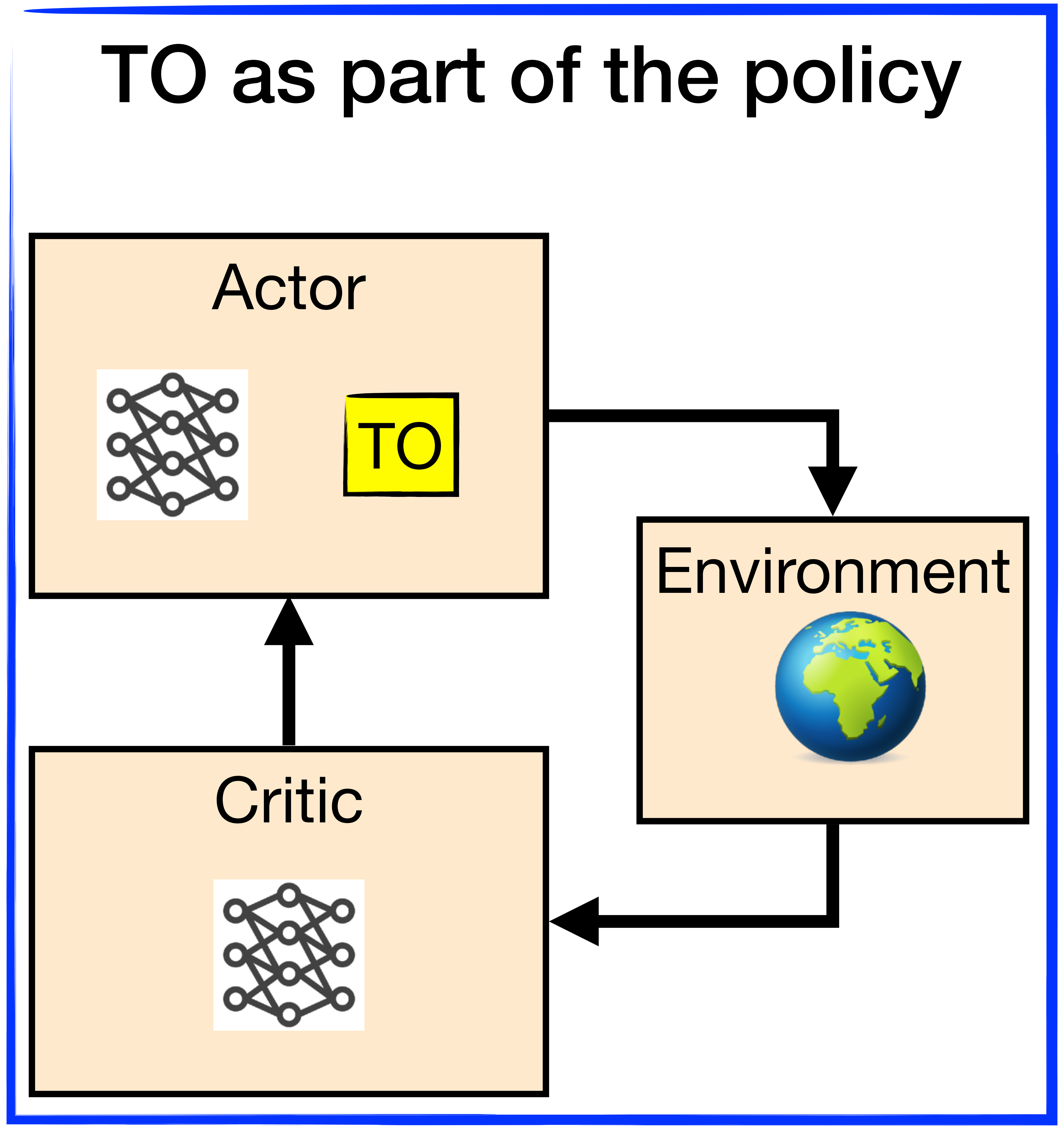


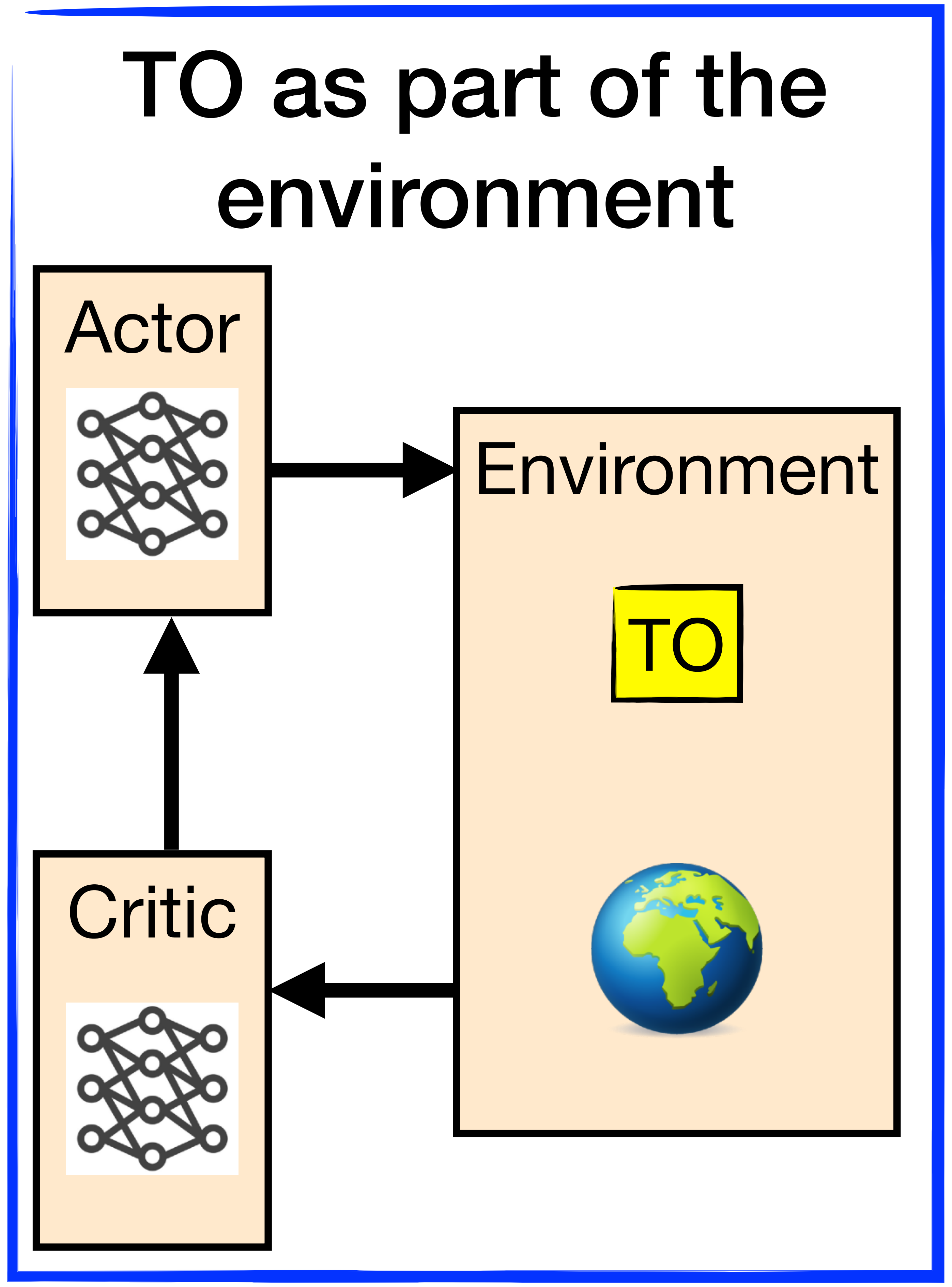
In which block should TO be considered?

Actor or environment?

Need to differentiate TO!

Actions are the output of TO



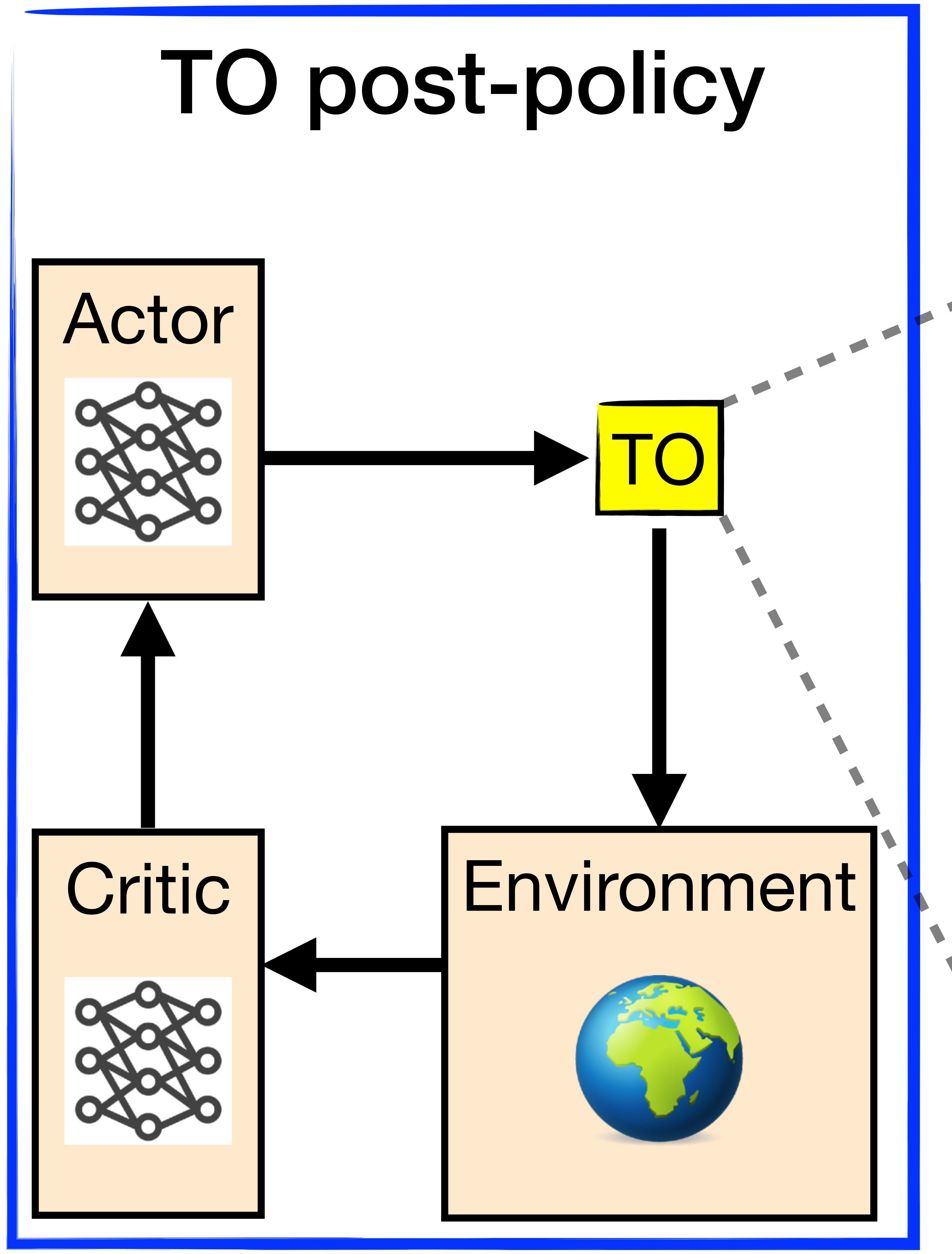


No need to differentiate TO!

Actions are the output of the actor policy

TO post-policy

What should the policy learn?



Running cost

Value function
(i.e. terminal cost)

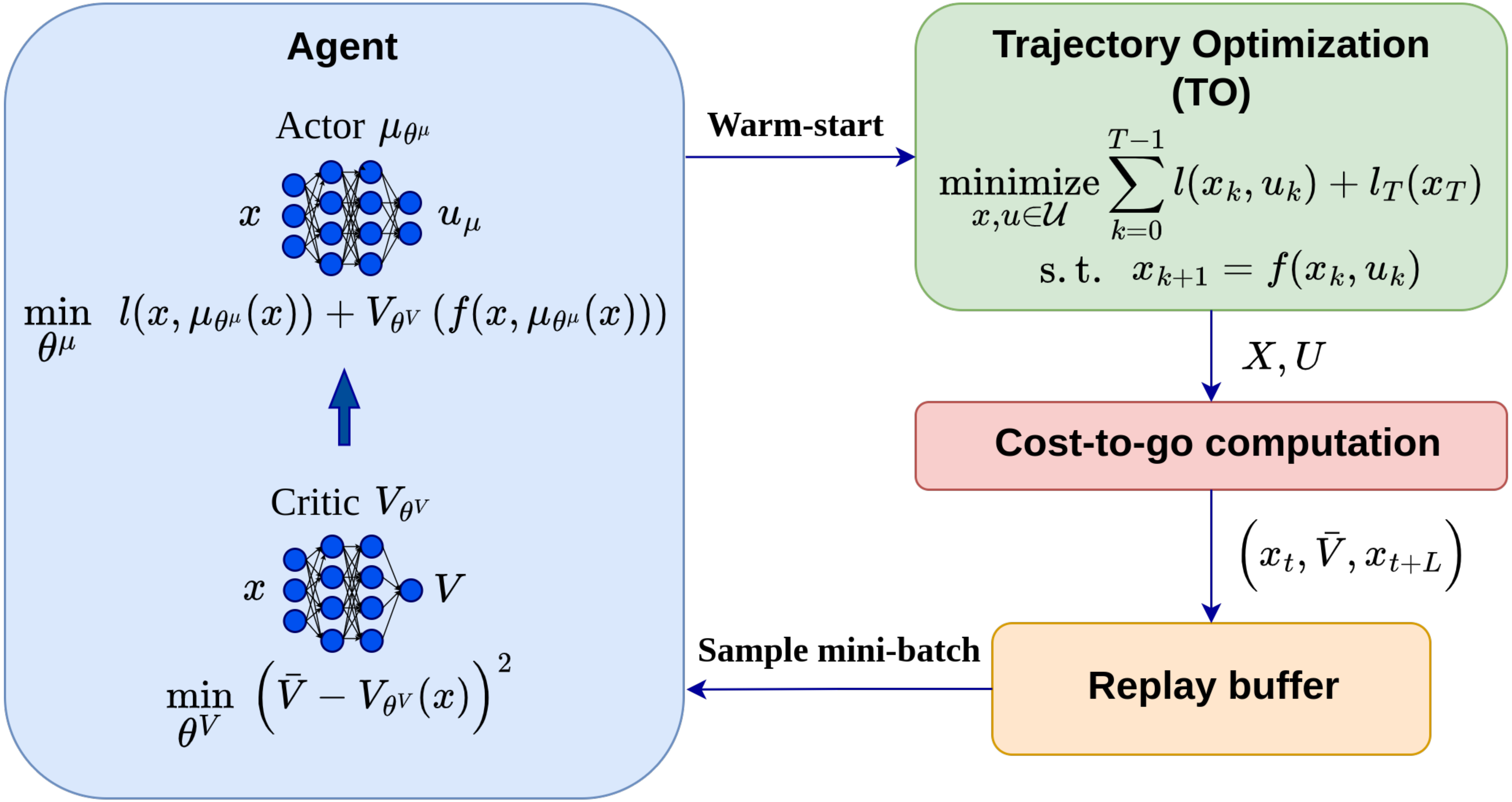
$$
\begin{aligned}
\underset{\{x_i\}_0^N,\{u_i\}_0^{N-1}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \boxed{\ell_i(x_i, u_i)} + \boxed{\ell_N(x_N)} \\
\text{subject to} \quad & x_{i+1} = \boxed{f(x_i, u_i)} \qquad i = 0 \ldots N-1 \\
& x_{i+1} \in \boxed{\mathcal{X}}, u_i \in \mathcal{U} \qquad i = 0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

Policy
(used to warm-start)

Running
constraints

Dynamics

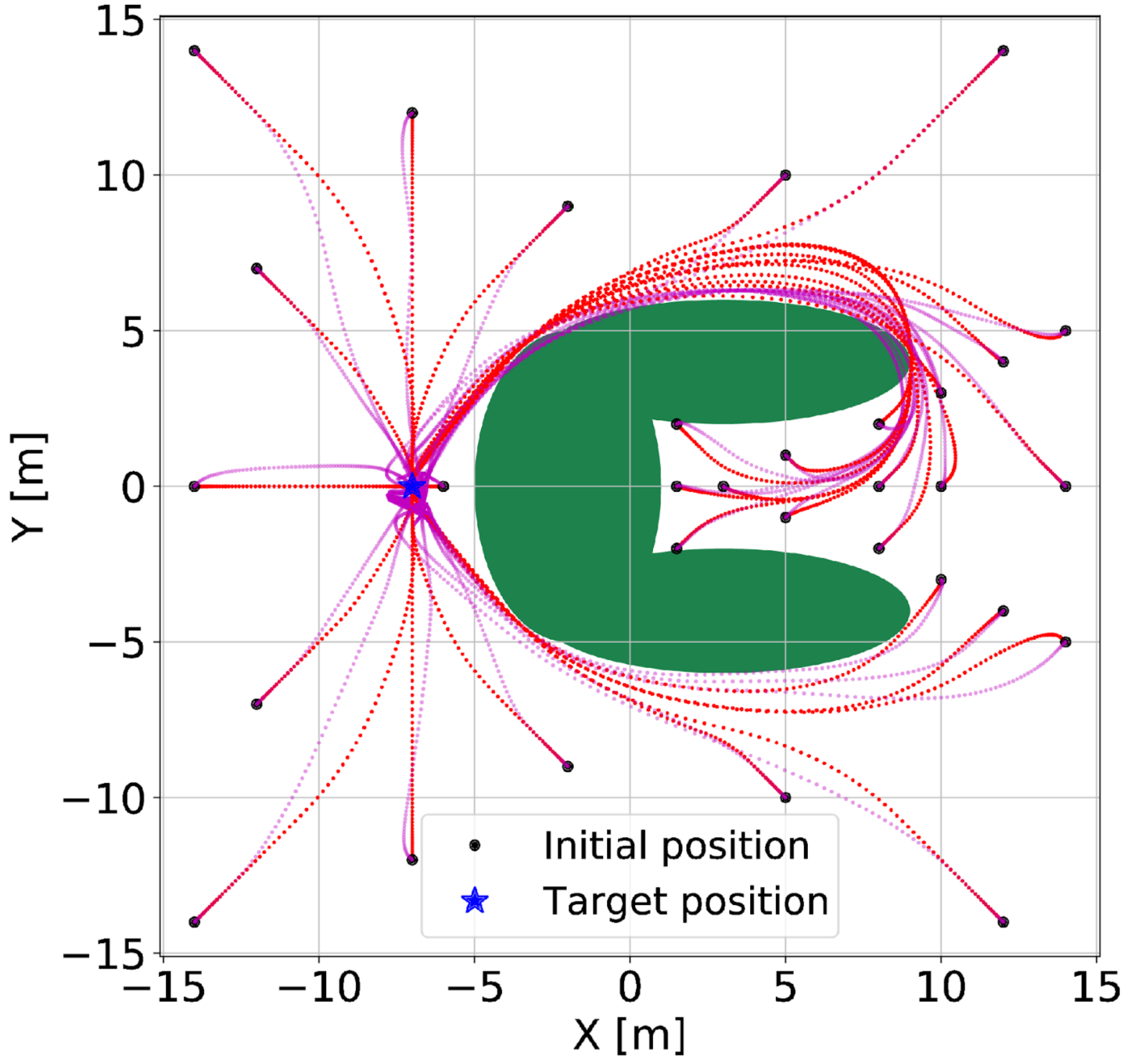
CACTO



[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

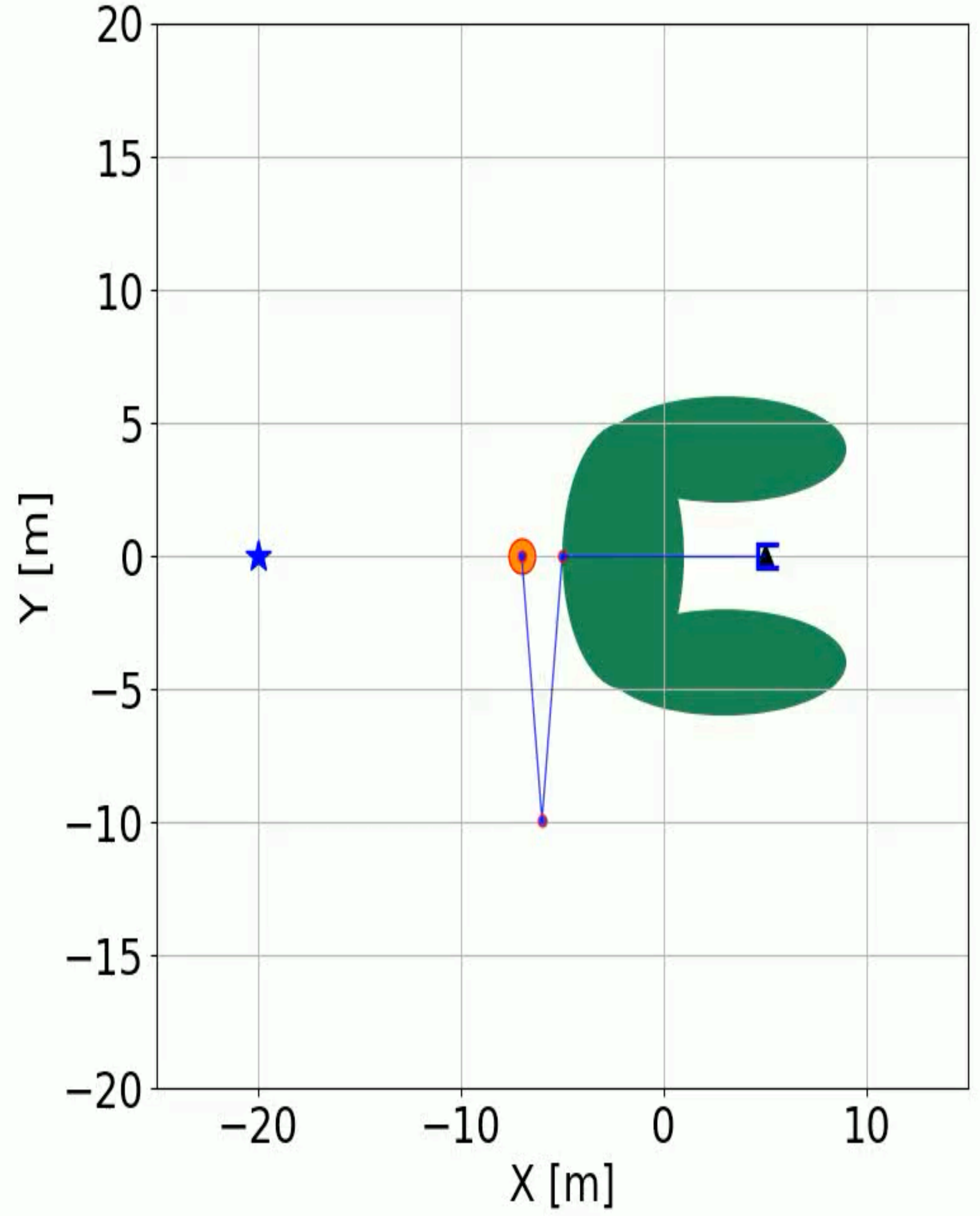
Task: find shortest path to target using low control effort and avoiding obstacles



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

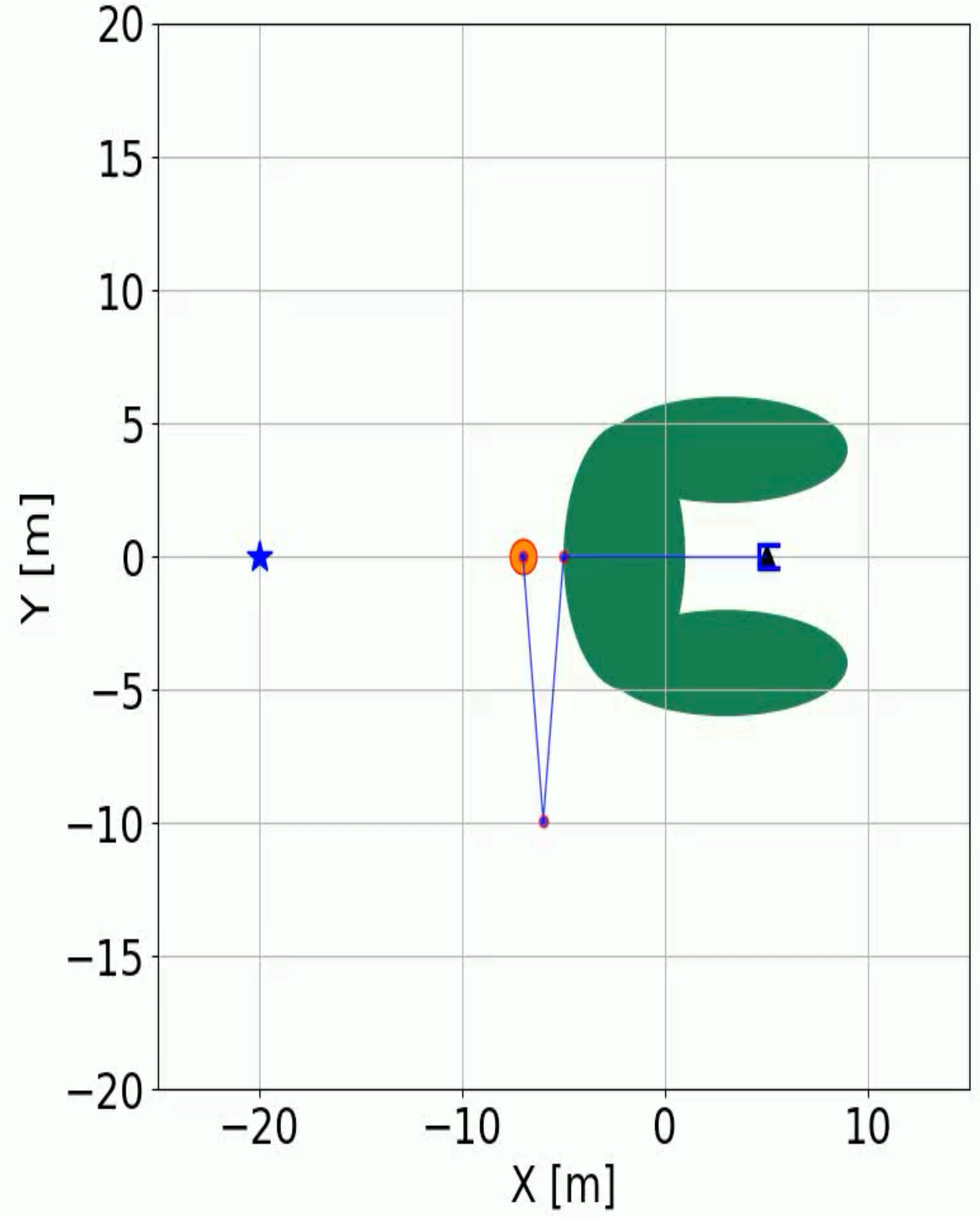
Results: 3-DoF Manipulator

Initial Conditions
warm-start



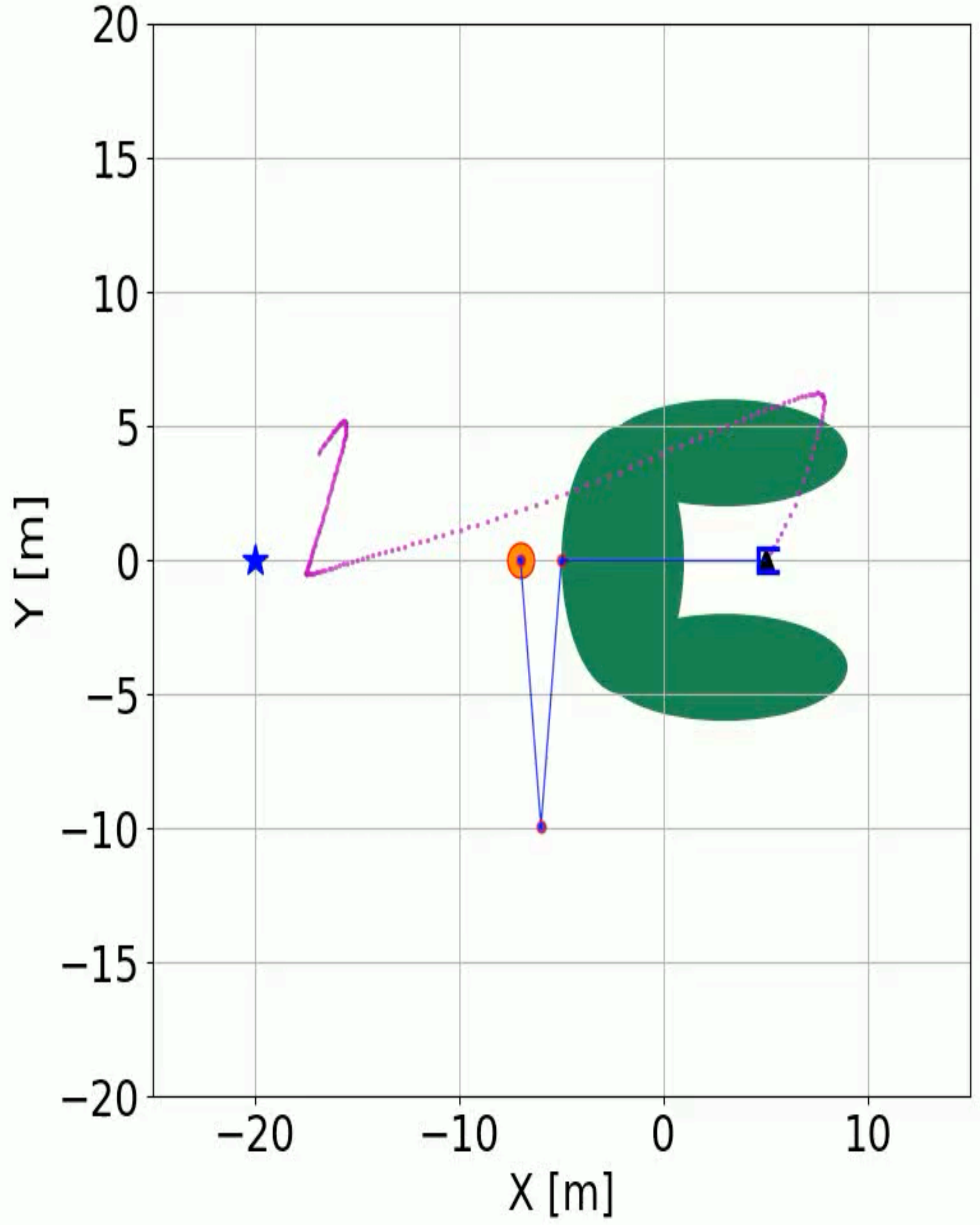
Cost = 70800

Random
warm-start



Cost = 88647

CACTO
warm-start



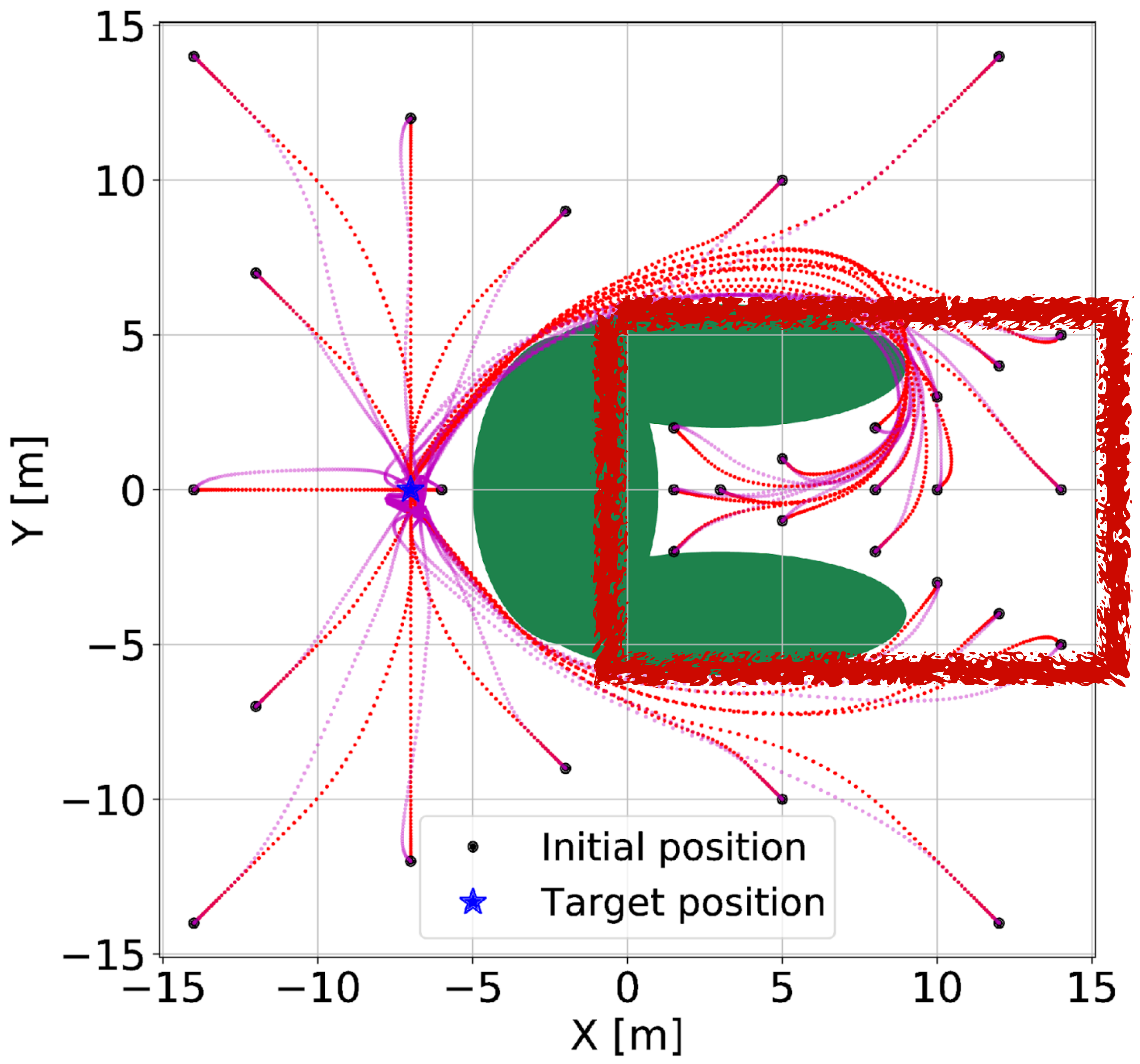
Cost = -145875

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

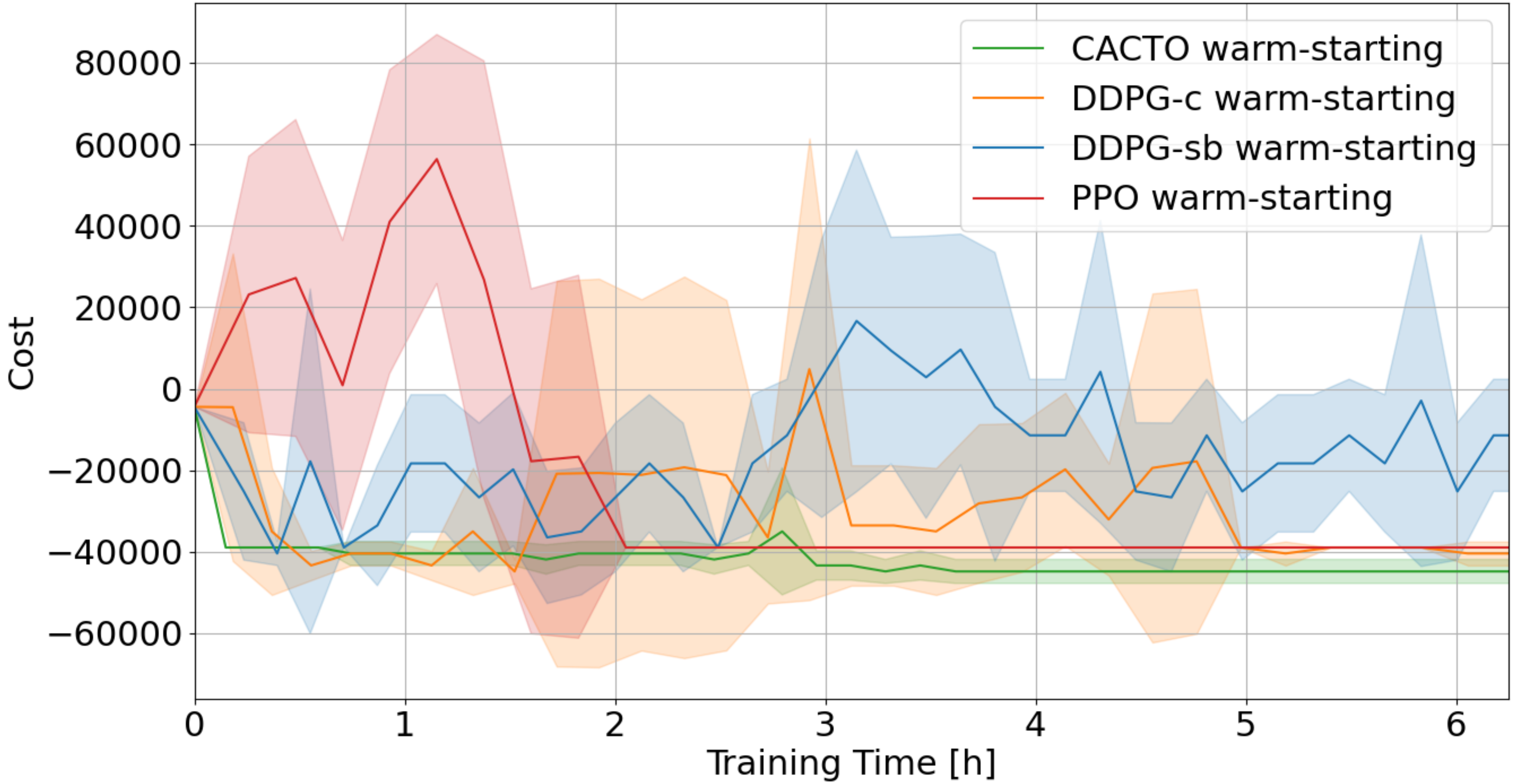
System	Hard Region	
	CACTO < (≤) Random	CACTO < (≤) ICS
2D Single Integrator	**99.1%** (99.1%)	**92%** (99.1%)
2D Double Integrator	**99.9%** (99.9%)	**92%** (99.1%)
Car	**100%** (100%)	**92.9%** (100%)
Manipulator	**87.5%** (87.5%)	**100%** (100%)



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



CACTO - Conclusions

- Novel RL scheme exploiting Trajectory Optimization
  - Proof of global convergence in discrete-space setting
  - Empirically superior to TO and RL alone

On-going/future work

- Fully-GPU implementation
- Handle uncertainties
- Handle sensor feedback
- Handle state constraints

Safe and Efficient Reinforcement Learning

Combining learning and trajectory optimization

Andrea Del Prete

UNIVERSITY OF TRENTO

Safety Definition

What is safety?

- Joint angle, velocity, torque limits
- Collision avoidance
  - Self-collision
  - Static obstacles (e.g., table, wall)
  - Dynamic obstacles (e.g., humans, other robots)
- Collision management:
  - Contact shall not result in pain or injury

$$x \in \mathscr{X}, u \in \mathscr{U}$$

State of the art

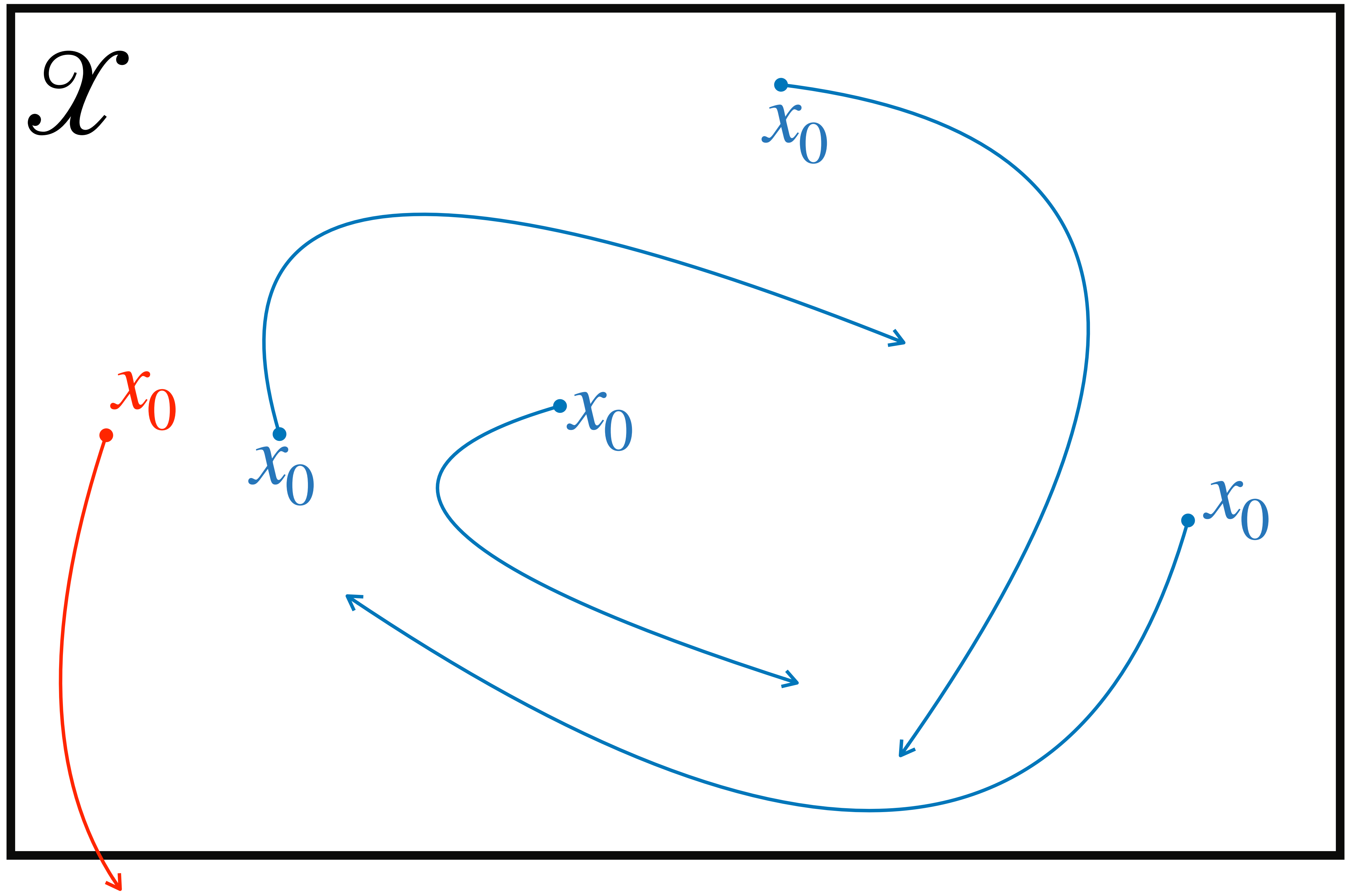
Constrained Dynamical System

Constrained discrete-time dynamical system:

$$x_{i+1} = f(x_i, u_i) \qquad x \in \mathcal{X}, \quad u \in \mathcal{U}$$

x is the state (positions, velocities)

u is the control input



Model Predictive Control

Trajectory Optimization inside the control loop

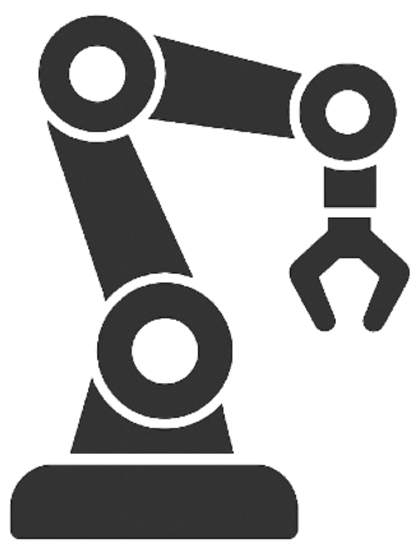
$x = x_{init}$

$$
\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) \\
\text{subject to} \quad & x_0 = x_{init} \\
& x_{i+1} = f(x_i, u_i) \qquad i = 0 \dots N-1 \\
& x_{i+1} \in \mathcal{X}, u_i \in \mathcal{U} \qquad i = 0 \dots N-1
\end{aligned}
$$

u_0

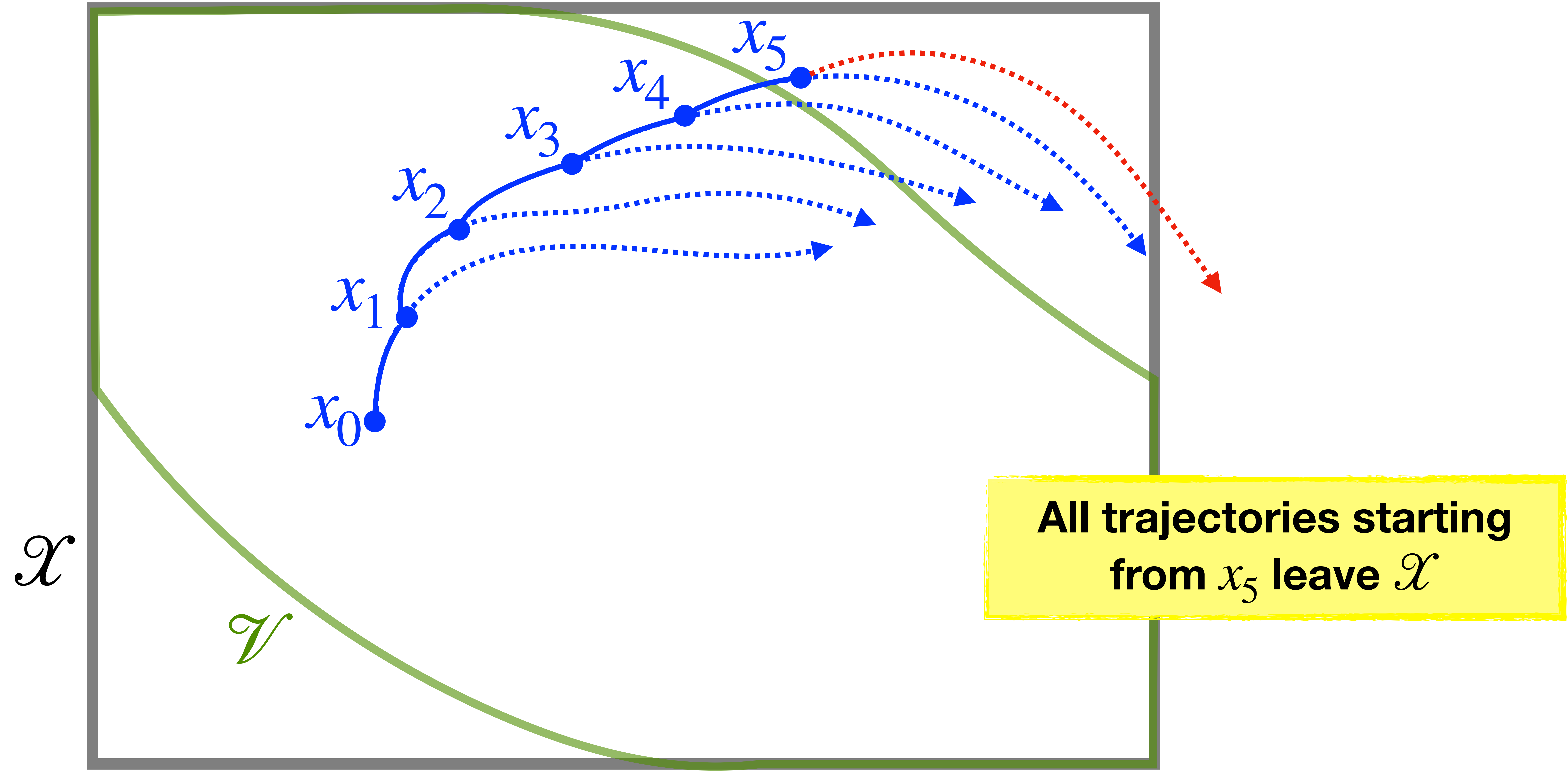
Robot

$f(x, u)$



Model Predictive Control

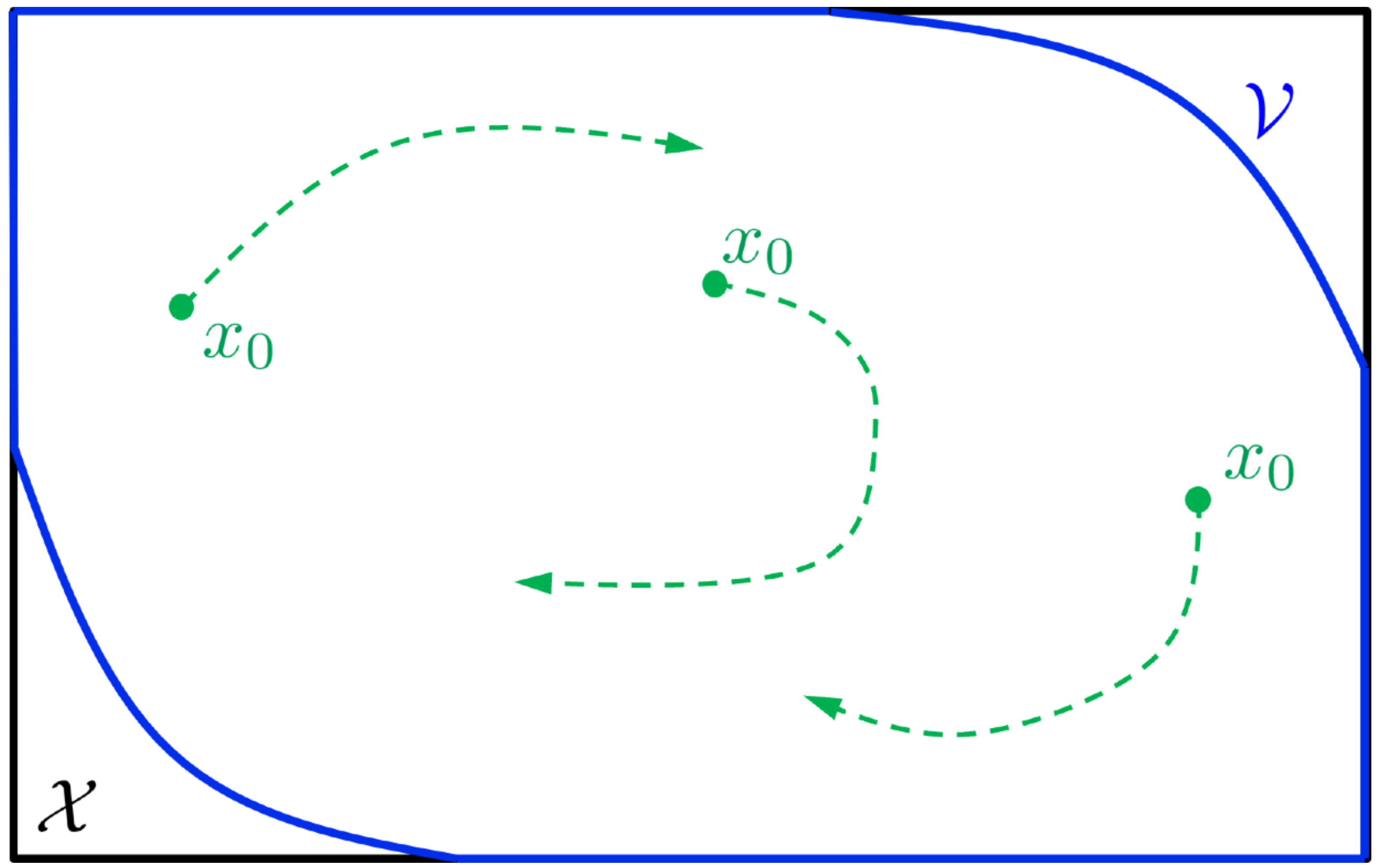
Can it ensure safety? No



Safety Guarantees

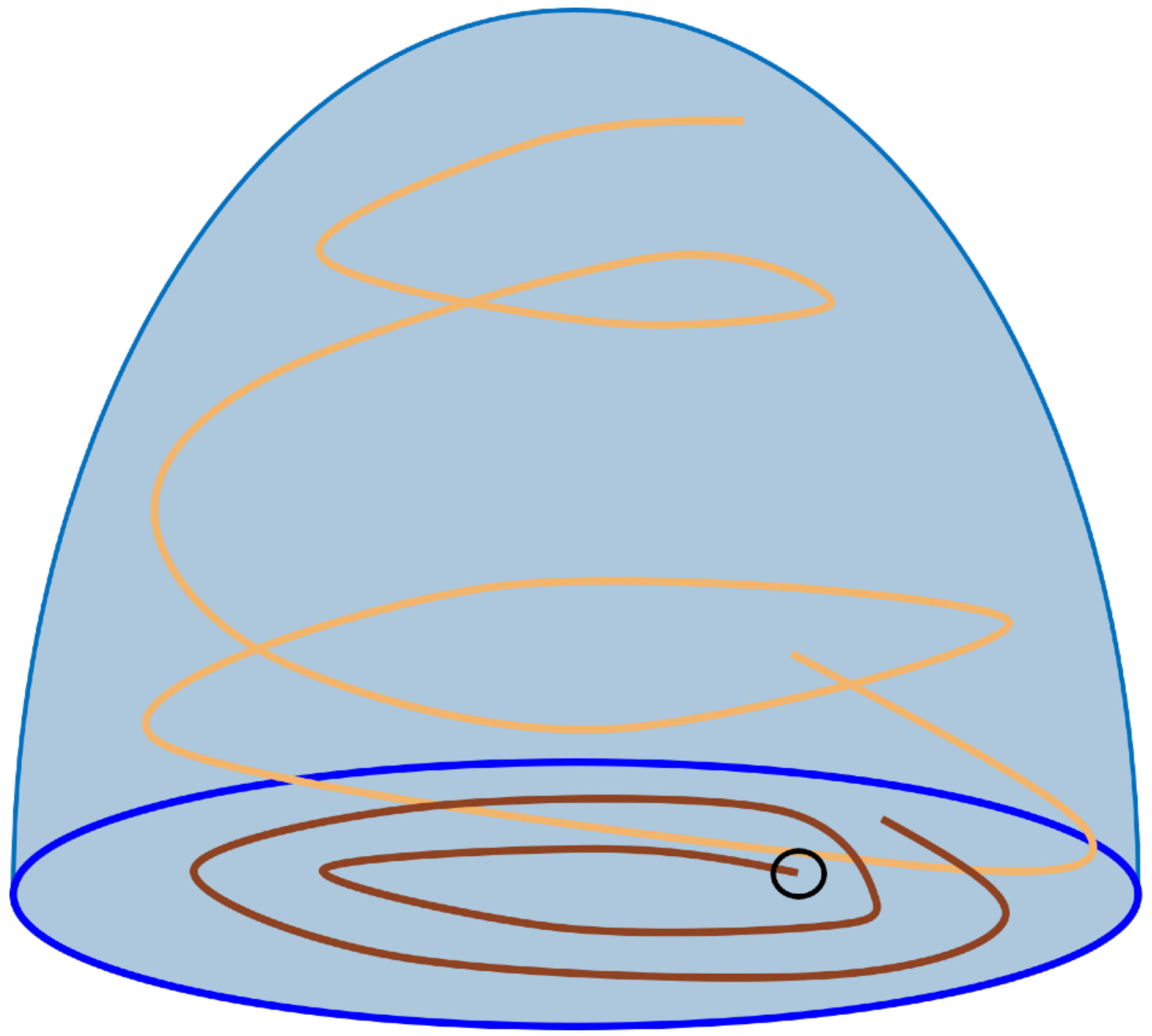
State of the art

Control-Invariant Sets (CIS)



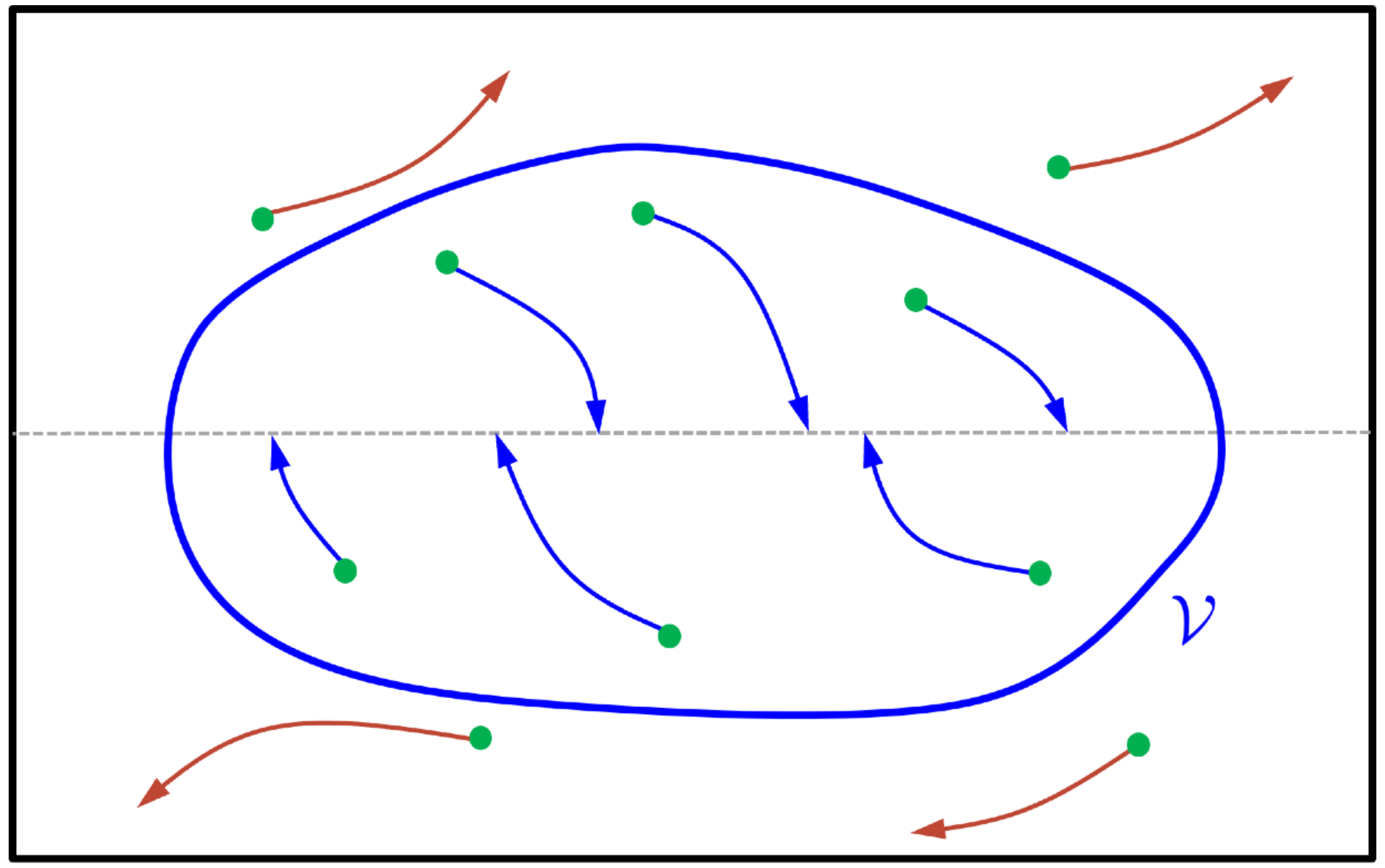
Model Predictive Control

Control Barrier Functions (CBF)



Quadratic Program

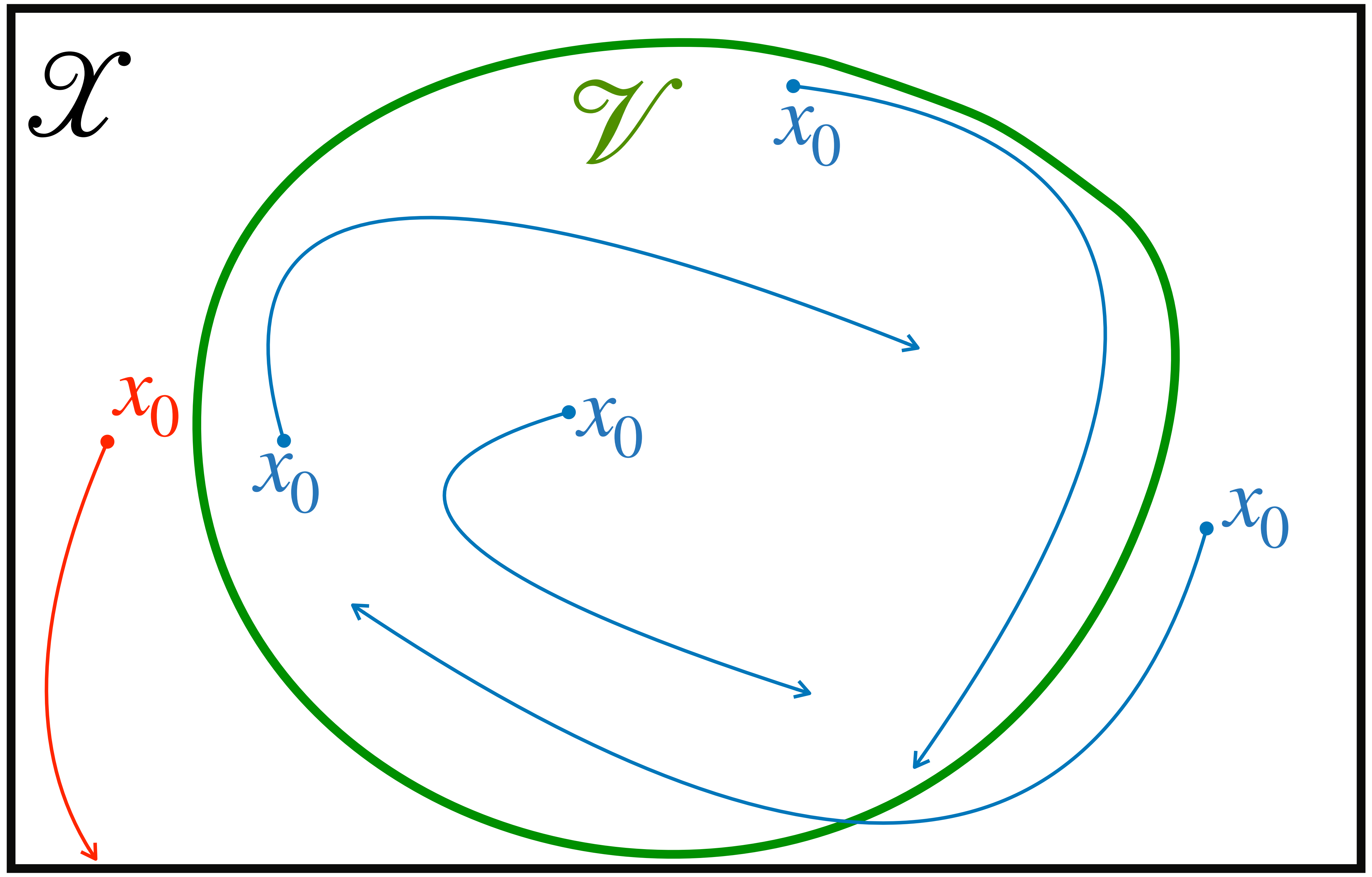
Back-up Policies (BUP)



Reinforcement Learning

Safety via Control Invariant Sets

$\mathscr{V} \subseteq \mathscr{X}$ is a control invariant set ⟷ Once x is in $\mathscr{V}$, it can remain in $\mathscr{V}$



Recursive Feasibility

Model Predictive Control (MPC)

Using a CIS $\mathscr{V}$ as terminal set ensures recursive feasibility in MPC

$$
\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) \\
\text{subject to} \quad & x_0 = x_{init} \\
& x_{i+1} = f(x_i, u_i) && i = 0 \dots N-1 \\
& x_i \in \mathcal{X}, u_i \in \mathcal{U} && i = 0 \dots N-1 \\
& \boxed{x_N \in \hat{\mathcal{V}}}
\end{aligned}
$$

What if the terminal set is an approximation of a CIS $\hat{\mathscr{V}} \approx \mathscr{V}$?

→

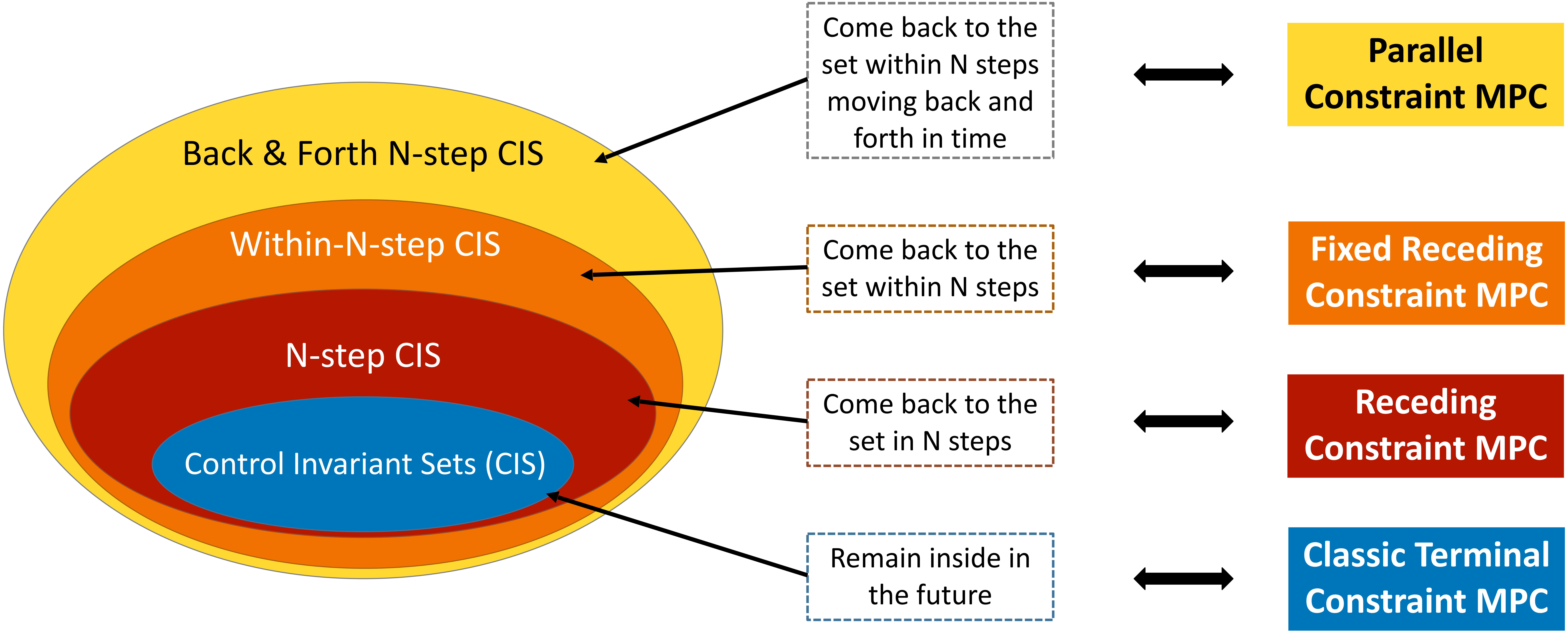
MPC problem can become unfeasible using $\hat{\mathscr{V}}$ instead of $\mathscr{V}$!

Beyond Control Invariant Sets

- CIS are unknown for nonlinear systems
- Numerical approximation techniques exist, however:
  - They are computationally demanding (curse of dimensionality)
  - A numerical approximation of a CIS is **not** a CIS
    - $\rightarrow$ all safety guarantees are lost!

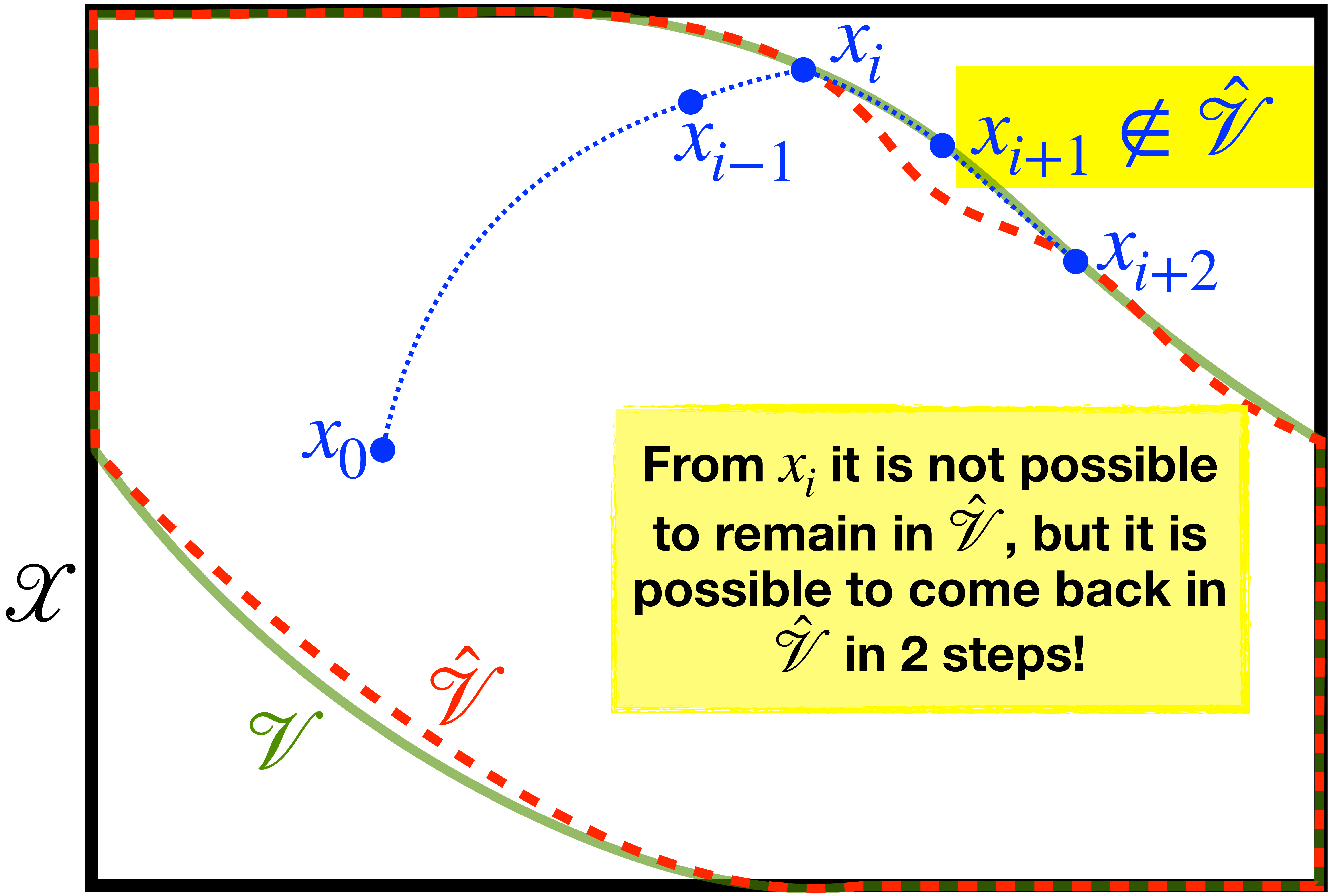
Do we really need Control Invariant Sets to ensure safety?

Beyond Control Invariant Sets



N-Step Control Invariant Set

- $\hat{\mathscr{V}}$ is an N-Step CIS iff:
  - For every $x_0 \in \hat{\mathscr{V}}$ it is possible to have $x_N \in \hat{\mathscr{V}}$
- Weaker condition than classic control invariance
- Possible to guarantee safety with novel MPC schemes



Beyond Control Invariant Sets

- Hypotheses:
  - new sets are easier to compute
  - new controllers work better even with approximate CIS

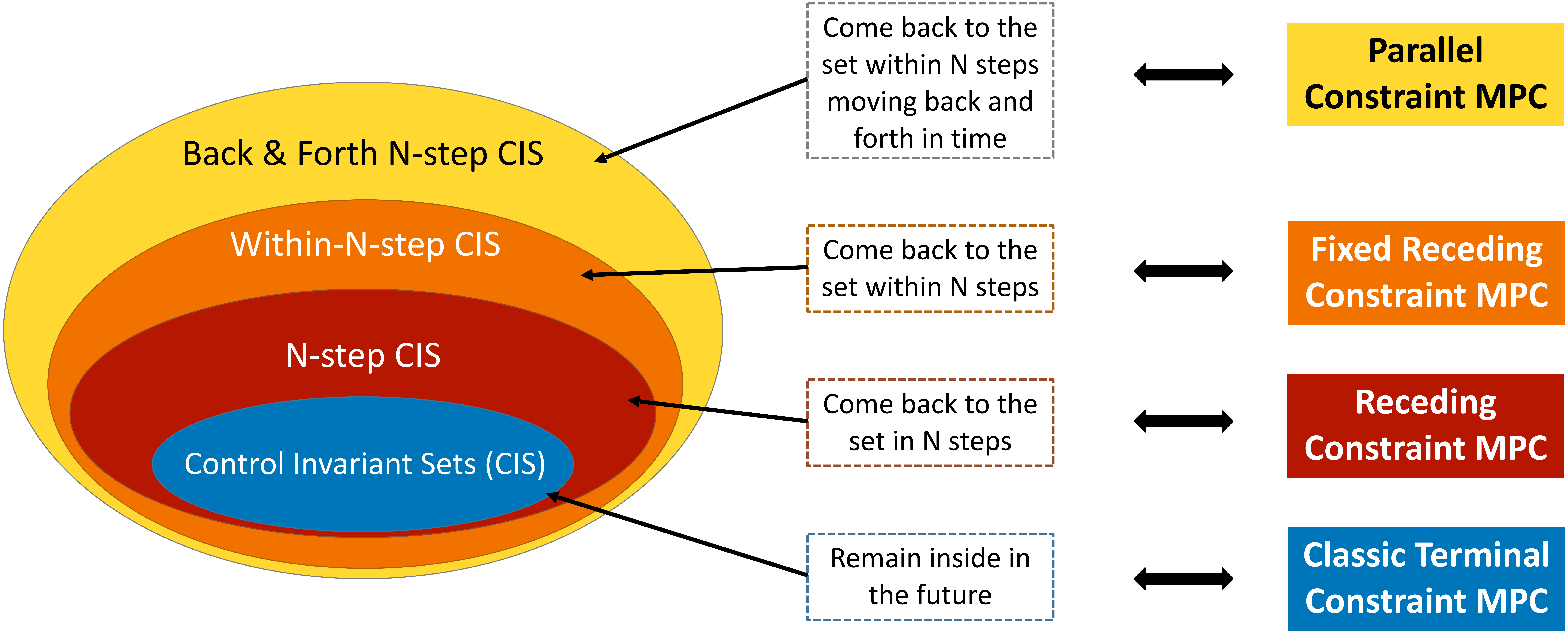
Receding-Constraint Model Predictive Control

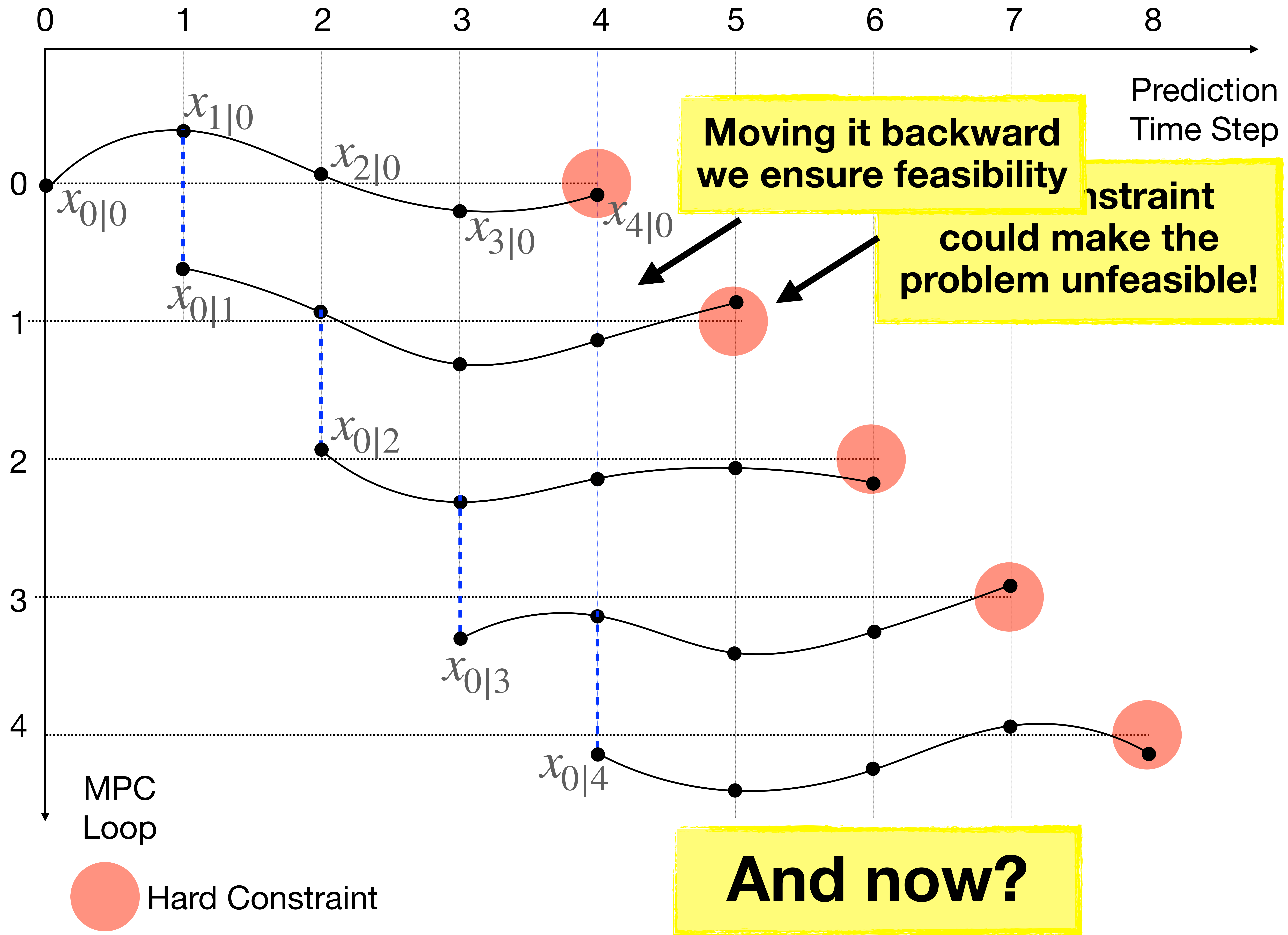
Gianni Lunardi
Asia La Rocca
Matteo Saveriano
Andrea Del Prete

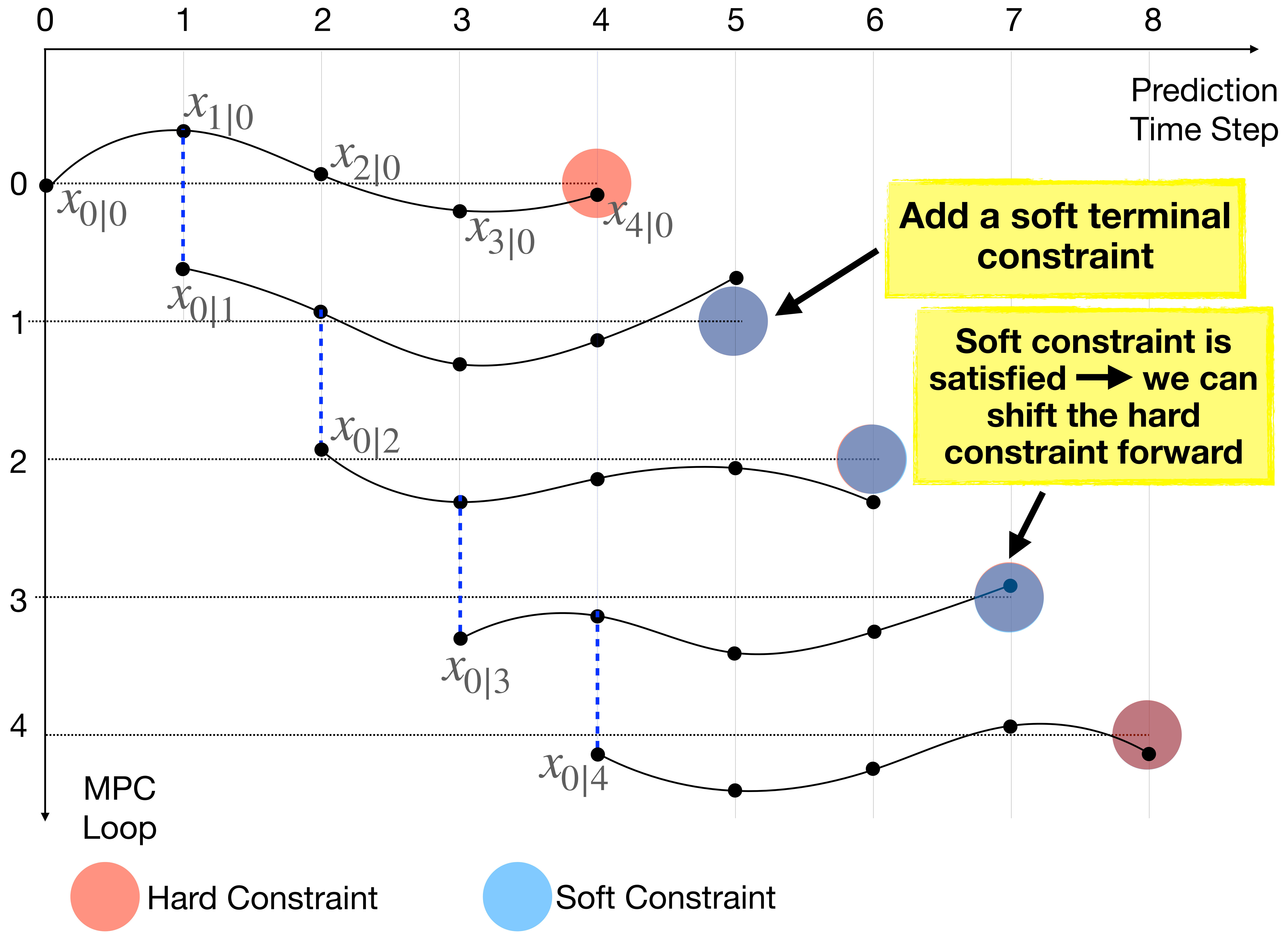
UNIVERSITY OF TRENTO

Lunardi, La Rocca, Saveriano, Del Prete (2024). Receding-Constraint Model Predictive Control using a Learned Approximate Control-Invariant Set. IEEE ICRA.

Beyond Control Invariant Sets




0
1
2
3
4
5
6
7
8
Prediction Time Step
$x_{1|0}$
$x_{2|0}$
$x_{3|0}$
$x_{4|0}$
$x_{0|0}$
$x_{0|1}$
$x_{0|2}$
$x_{0|3}$
$x_{0|4}$
Moving it backward we ensure feasibility
nstraint could make the problem unfeasible!
MPC Loop
Hard Constraint
And now?


0
1
2
3
4
5
6
7
8
Prediction Time Step
MPC Loop
$x_{0|0}$
$x_{1|0}$
$x_{2|0}$
$x_{3|0}$
$x_{4|0}$
$x_{0|1}$
$x_{0|2}$
$x_{0|3}$
$x_{0|4}$
Add a soft terminal constraint
Soft constraint is satisfied ⟶ we can shift the hard constraint forward
Hard Constraint
Soft Constraint

What if the problem gets unfeasible?

Safe Abort Procedure

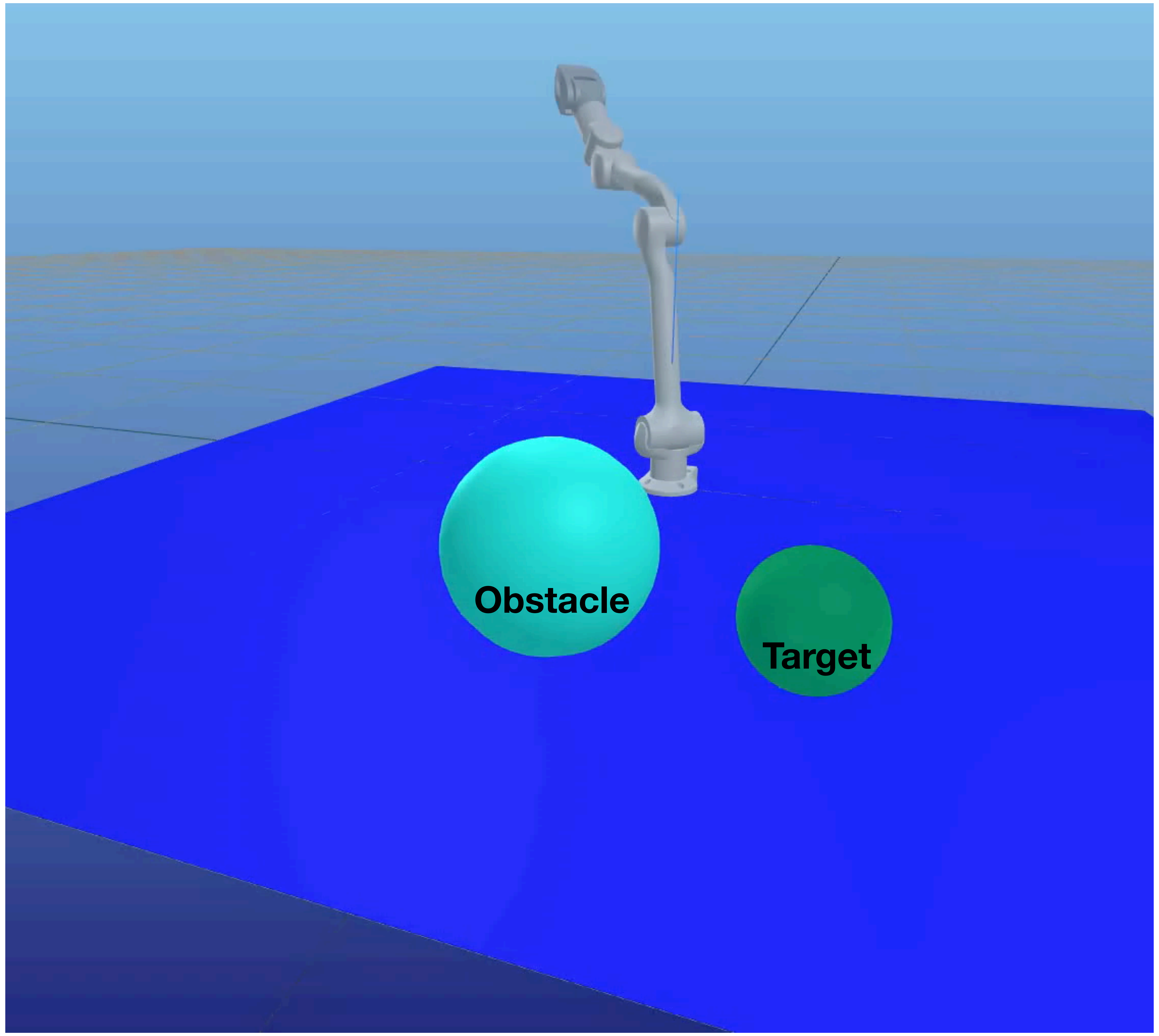
Assume $\hat{\mathscr{V}} \subseteq \mathscr{V}$ → Even if $\hat{\mathscr{V}}$ is not a CIS, any state in $\hat{\mathscr{V}}$ is "safe"

- **Safe Abort:**
  - If MPC problem becomes unfeasible
  - Find (and follow) trajectory that:
    - starts from last predicted state in $\hat{\mathscr{V}}$
    - reaches an equilibrium state
- Such a trajectory is guaranteed to exist

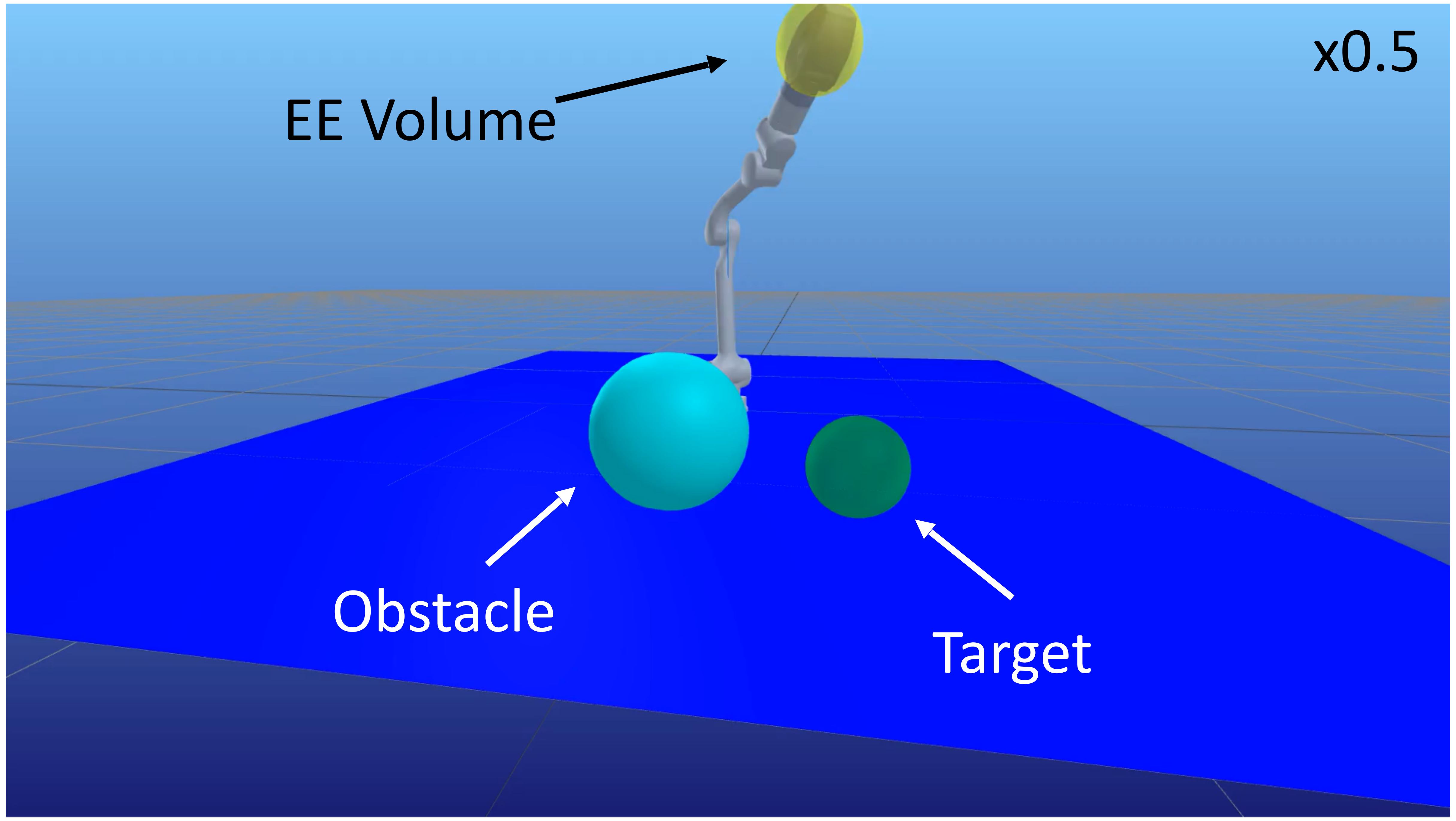
$$
\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) \\
\text{subject to} \quad & x_0 = x_{init} \\
& x_{i+1} = f(x_i, u_i) \qquad i = 0 \dots N-1 \\
& x_i \in \mathcal{X}, u_i \in \mathcal{U} \qquad i = 0 \dots N-1 \\
& \boxed{x_N = x_{N-1}}
\end{aligned}
$$

Simulation Results

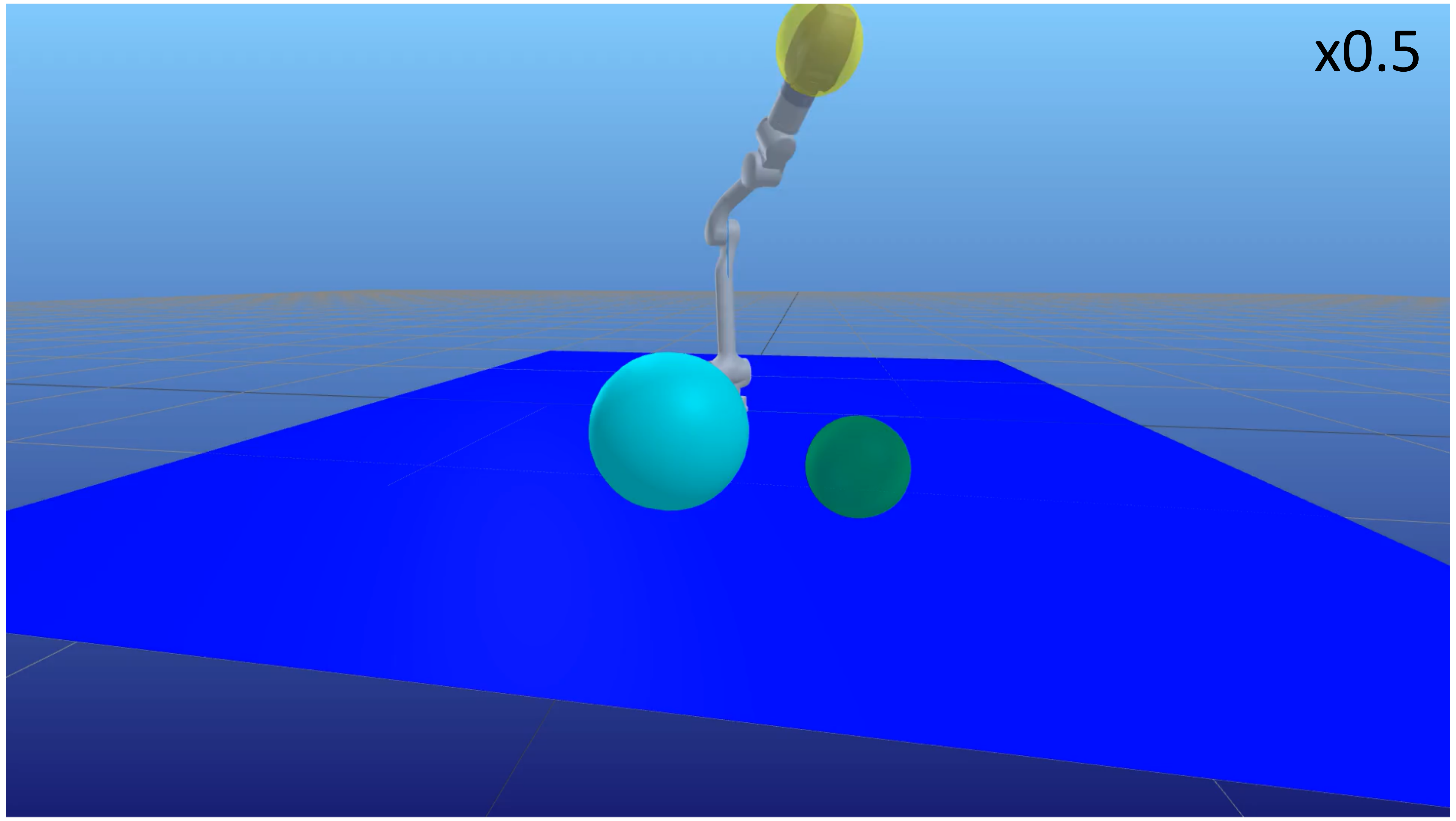
- Comparing several MPC formulations
- 4 DoF Z1 robot manipulator
- Acados software library
- Safe set $\hat{\mathcal{V}}$ represented with neural network
- 500 simulations from random initial configurations
- Max horizon N=45 to ensure computation time < dt (5 ms)
- https://github.com/idra-lab/safe-mpc



Setpoint Task

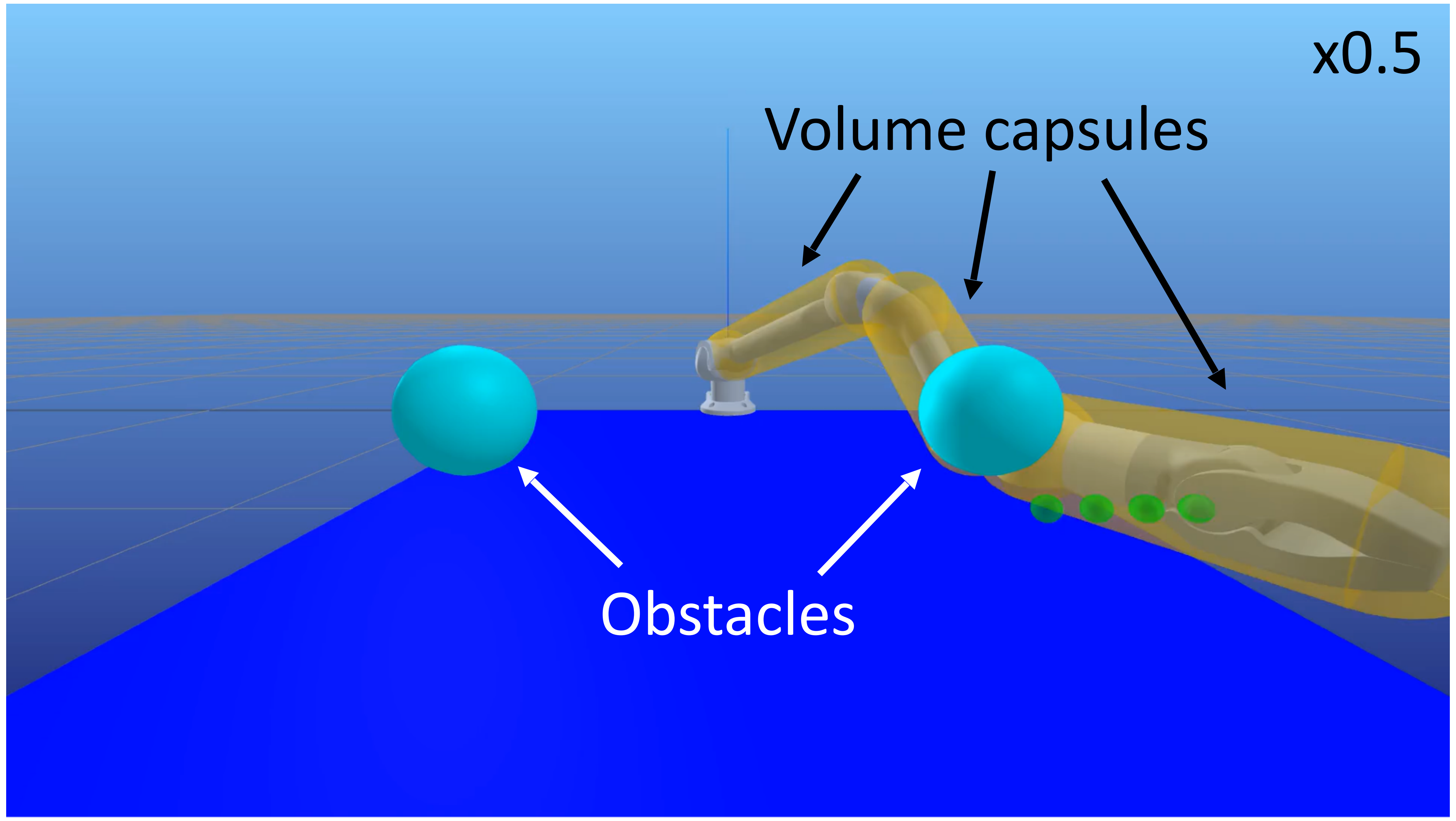


Naive MPC

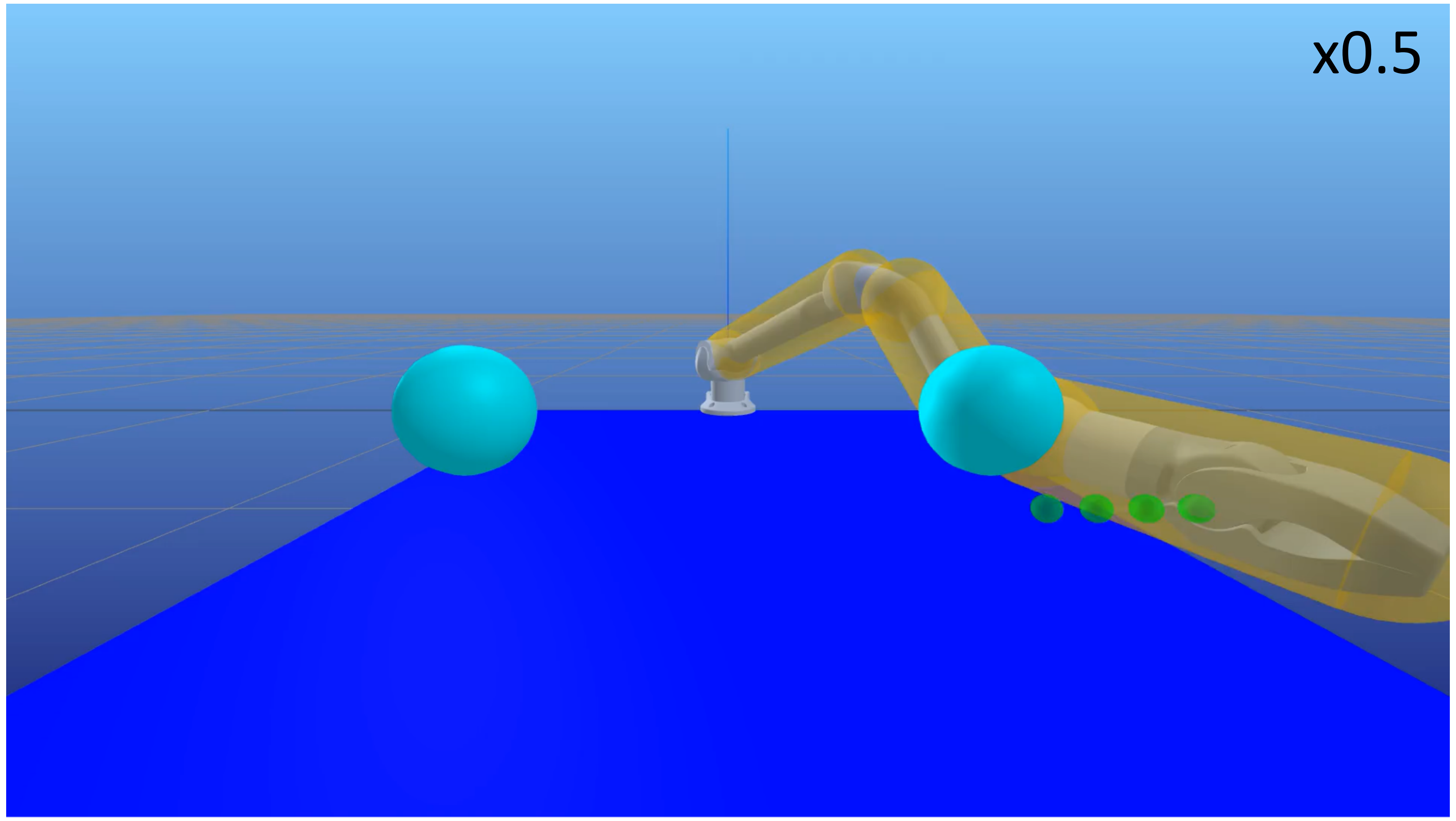


Receding MPC

Trajectory Tracking



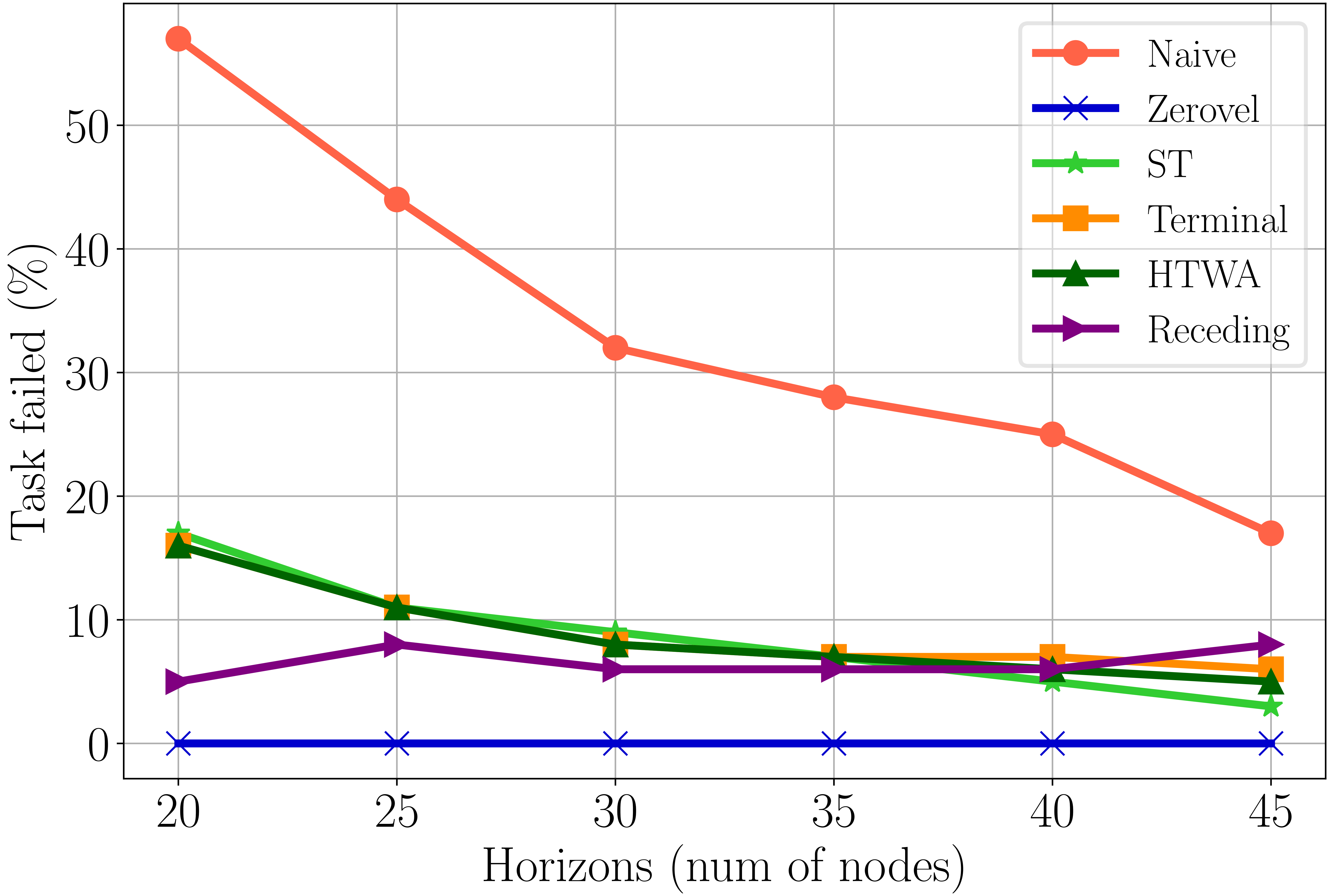
Naive MPC



Receding MPC

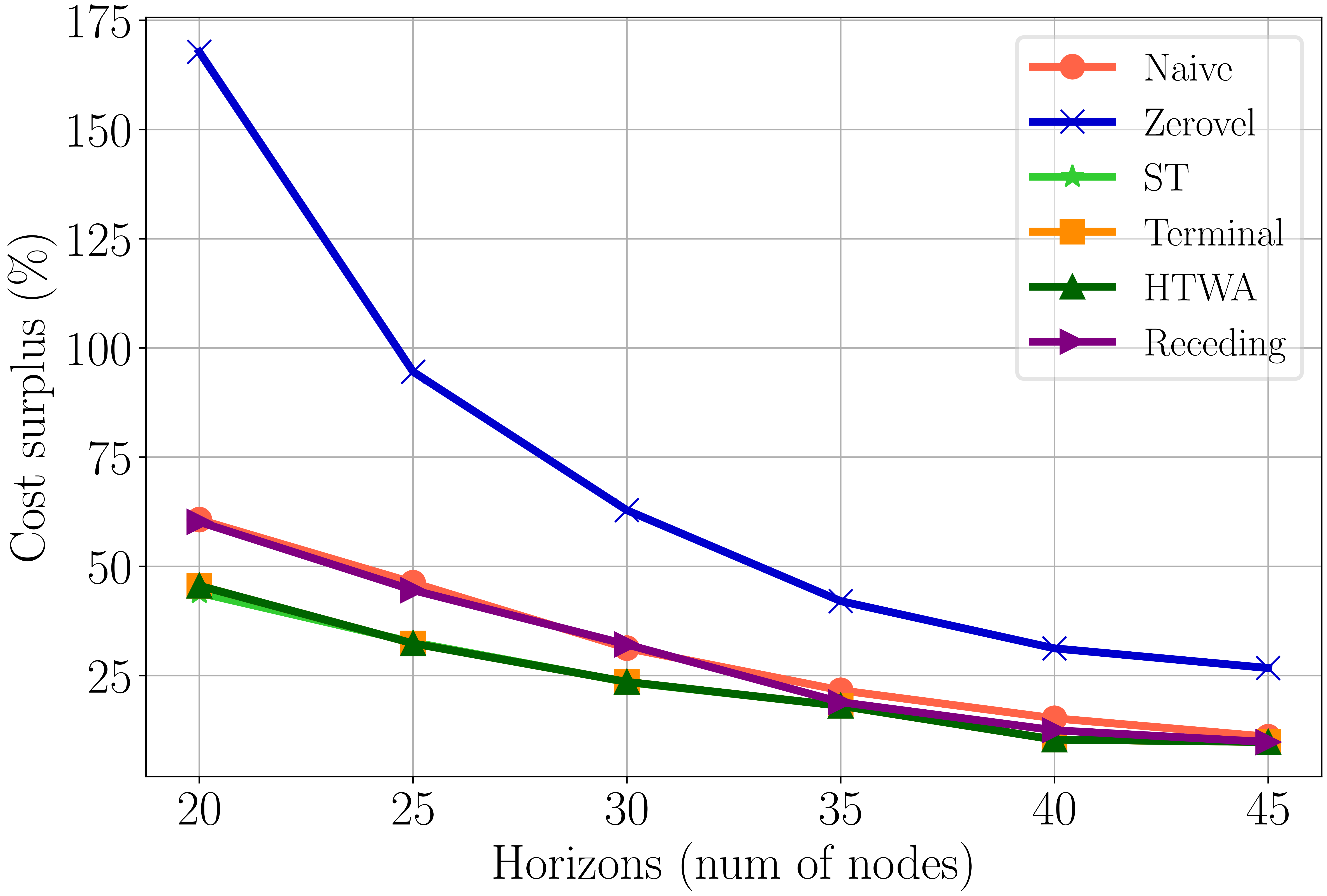
Simulation Results - Receding

- Naive: standard MPC formulation
- Zerovel: terminal constraint imposing zero velocity
- ST: soft terminal constraint $\hat{\mathscr{V}}$
- Terminal: hard terminal constraint $\hat{\mathscr{V}}$
- HTWA: hard terminal constraint $\hat{\mathscr{V}}$ with safe abort strategy



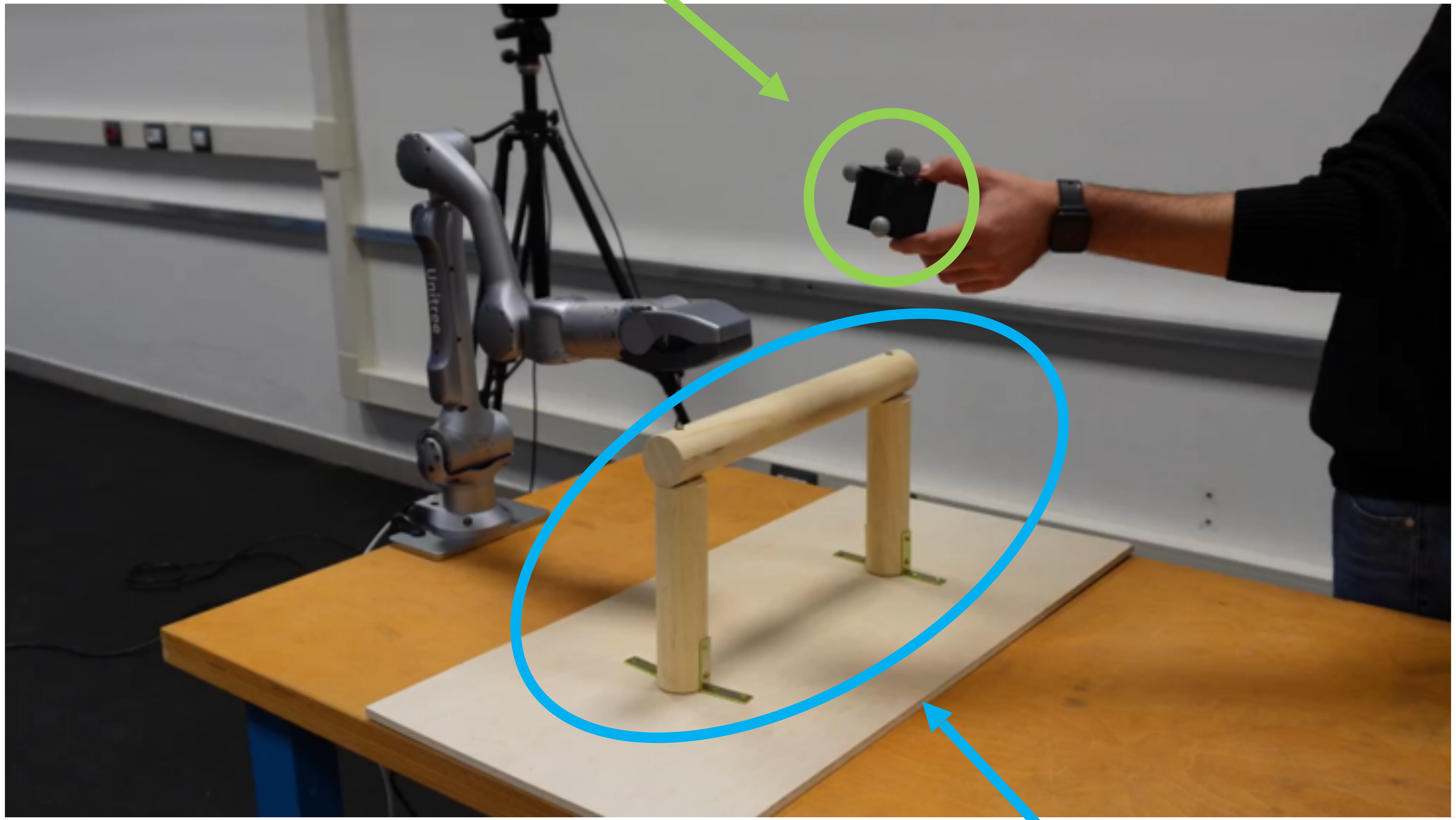
Simulation Results - Receding

- Naive: standard MPC formulation
- Zerovel: terminal constraint imposing zero velocity
- ST: soft terminal constraint $\hat{\mathscr{V}}$
- Terminal: hard terminal constraint $\hat{\mathscr{V}}$
- HTWA: hard terminal constraint $\hat{\mathscr{V}}$ with safe abort strategy

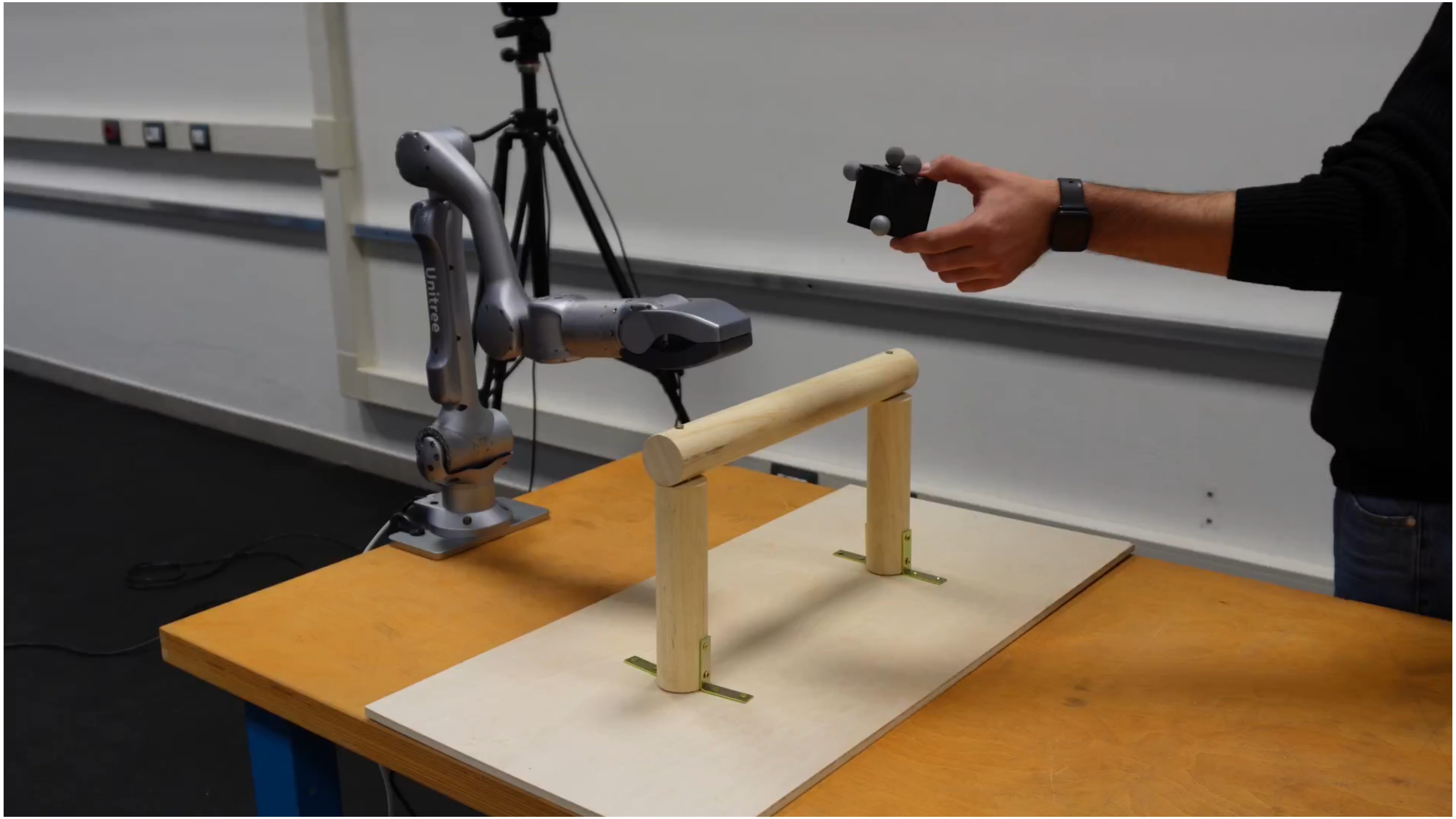


Experimental Setup

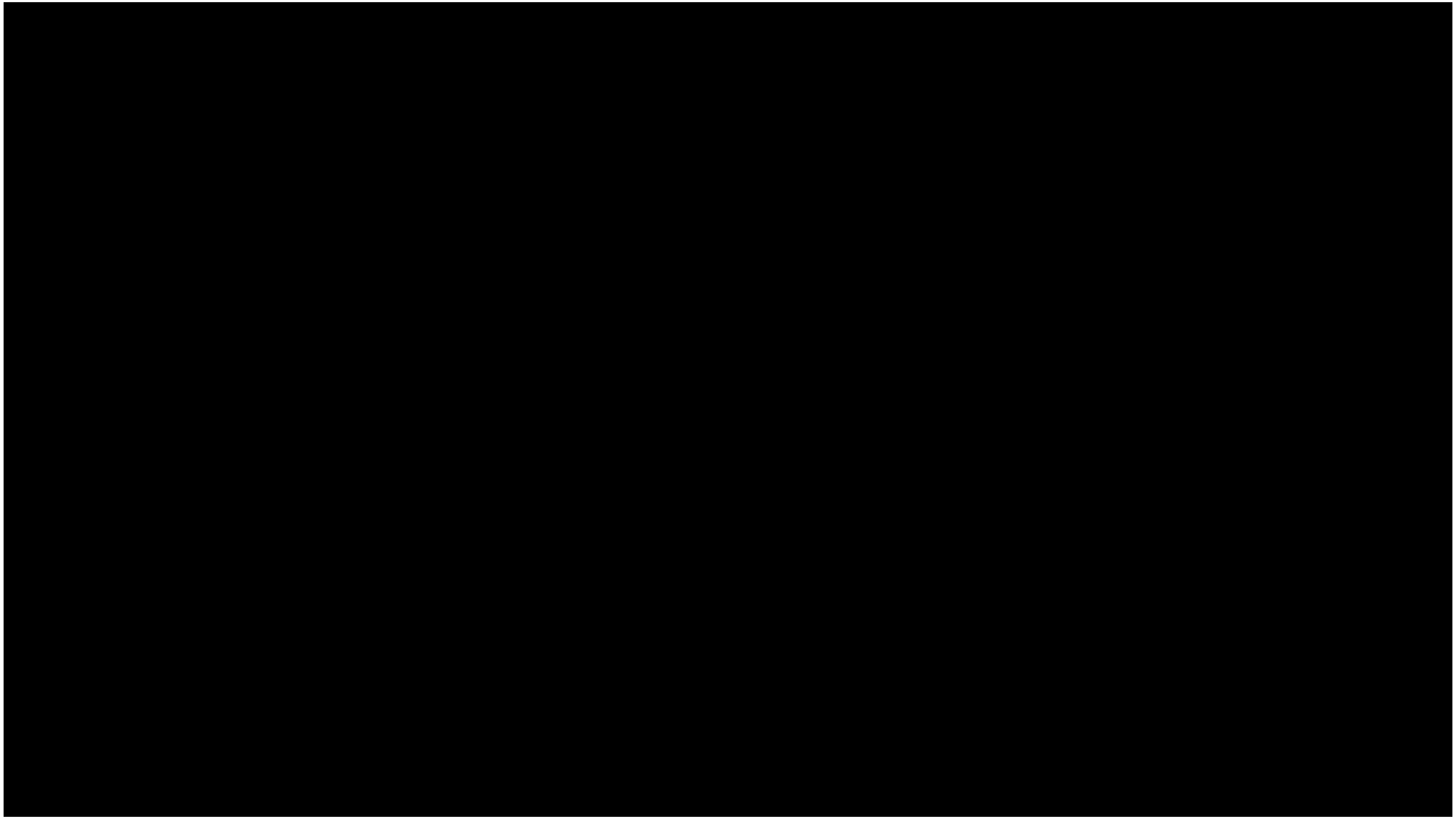
- Receding-Constraint MPC
- 6 degrees of freedom
- Z1 robot manipulator
- Tracking of moving target
- Target perceived with motion capture



Tracking Experiment - Side View



Tracking Experiment - Top View



Safe MPC - Conclusions

- Novel MPC formulations ensuring
  - Recursive feasibility under weaker conditions (N-Step CIS)
  - Safety under even weaker conditions (inner approx. of CIS)
  - Empirically superior when using approximate CIS

On-going/future work

- Computation/certification of N-Step CIS
- Handle dynamics uncertainties/obstacles
- Application as safety filter for RL policies

Take-Home Message

Globally Optimal and Safe Robot Control

- Using ideas from TO we can make RL efficient and safe
  - Use dynamics derivatives to guide RL exploration (CACTO)
  - Use novel safe sets to make control (RL) safe

Current challenges

- algorithms to compute $\hat{\mathscr{V}}$ do not scale and cannot certify set properties (e.g. N-Step Control Invariance)
- dynamics derivatives are ill-defined in contact-rich tasks

Safe and Efficient Reinforcement Learning

Combining learning and trajectory optimization









Andrea Del Prete

UNIVERSITY OF TRENTO