Rennes, 01/04/2026

Safe and Globally Optimal Robot Control

Beyond control invariance and reinforcement learning

IDRA
INTERDEPARTMENTAL ROBOTICS LABS

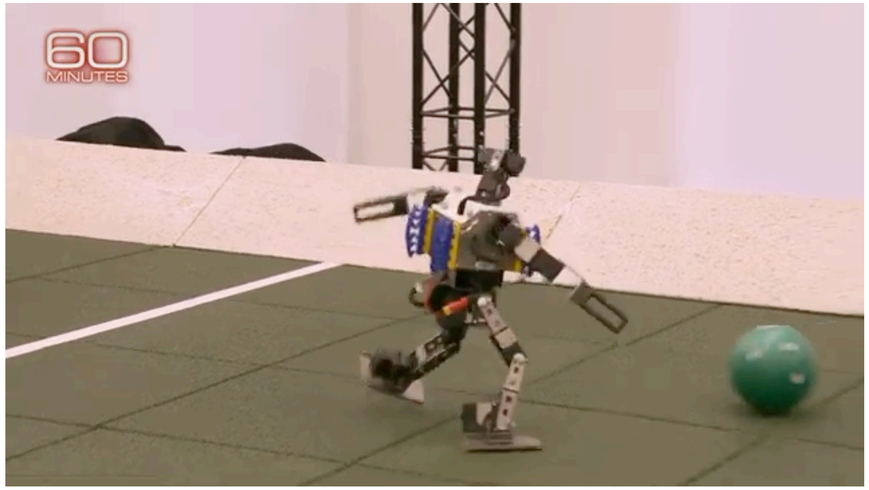
Andrea Del Prete

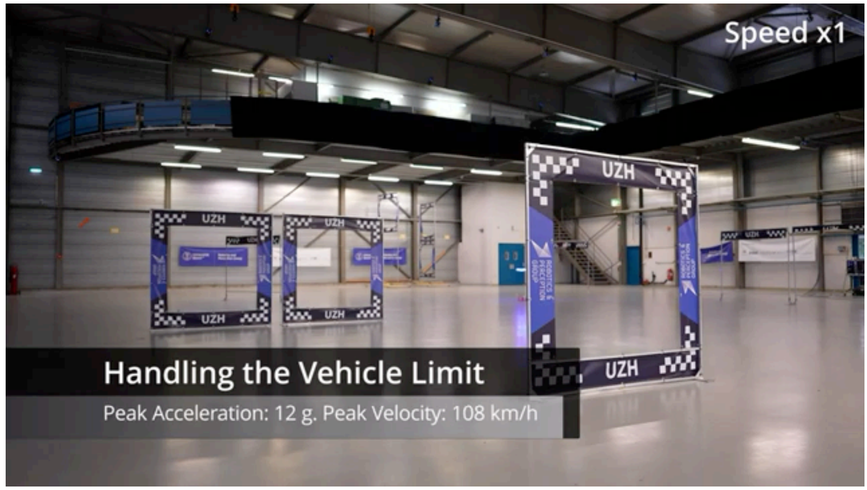
UNIVERSITY OF TRENTO

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?

Reinforcement Learning ~~VS~~ Trajectory Optimization WITH?

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

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

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

Reinforcement Learning

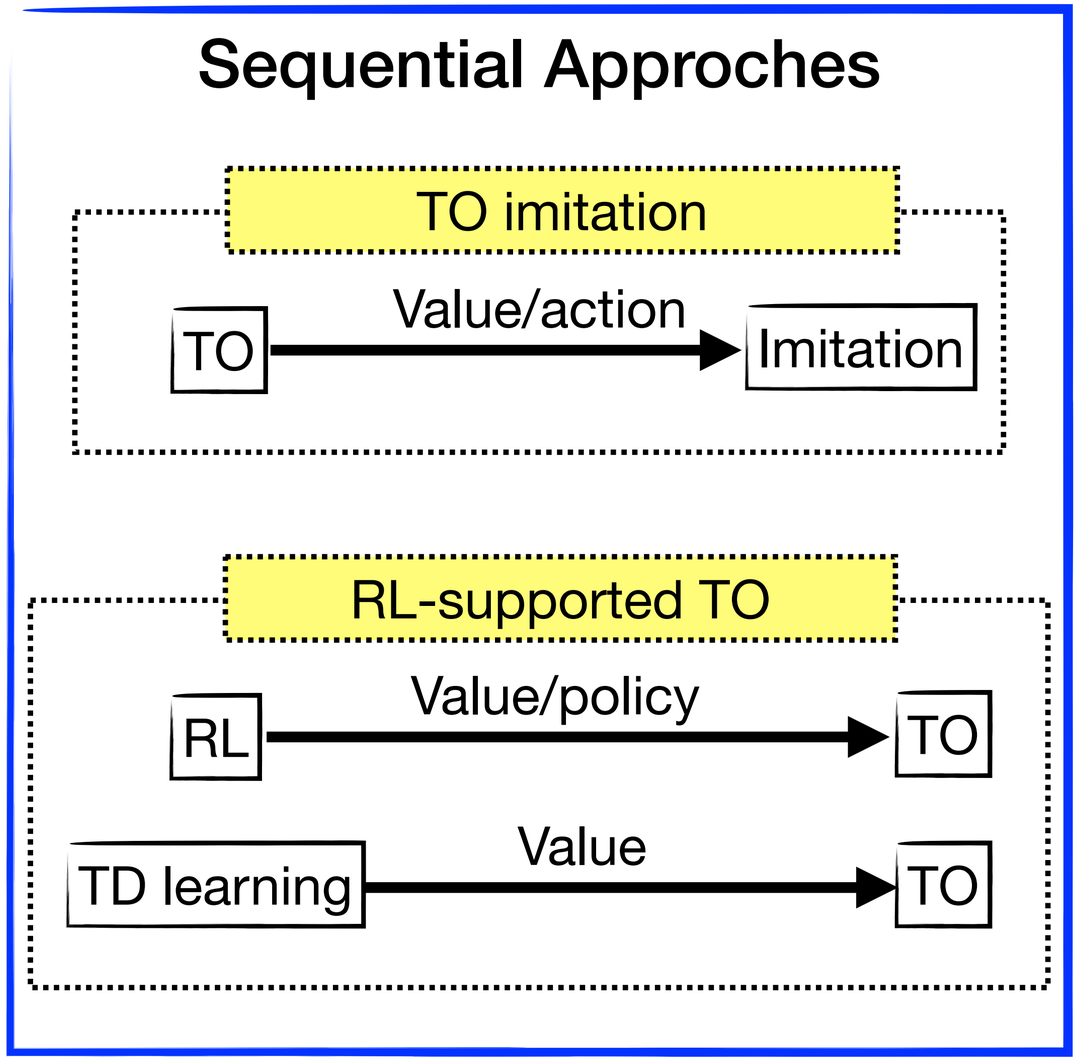
- \+ Less prone to poor local minima
- \+ Derivative free (easy to implement)
- \+ Fast online policy evaluation
- \+ Typically stochastic
- − Poor data efficiency (slow training)
- − Does not account for constraints

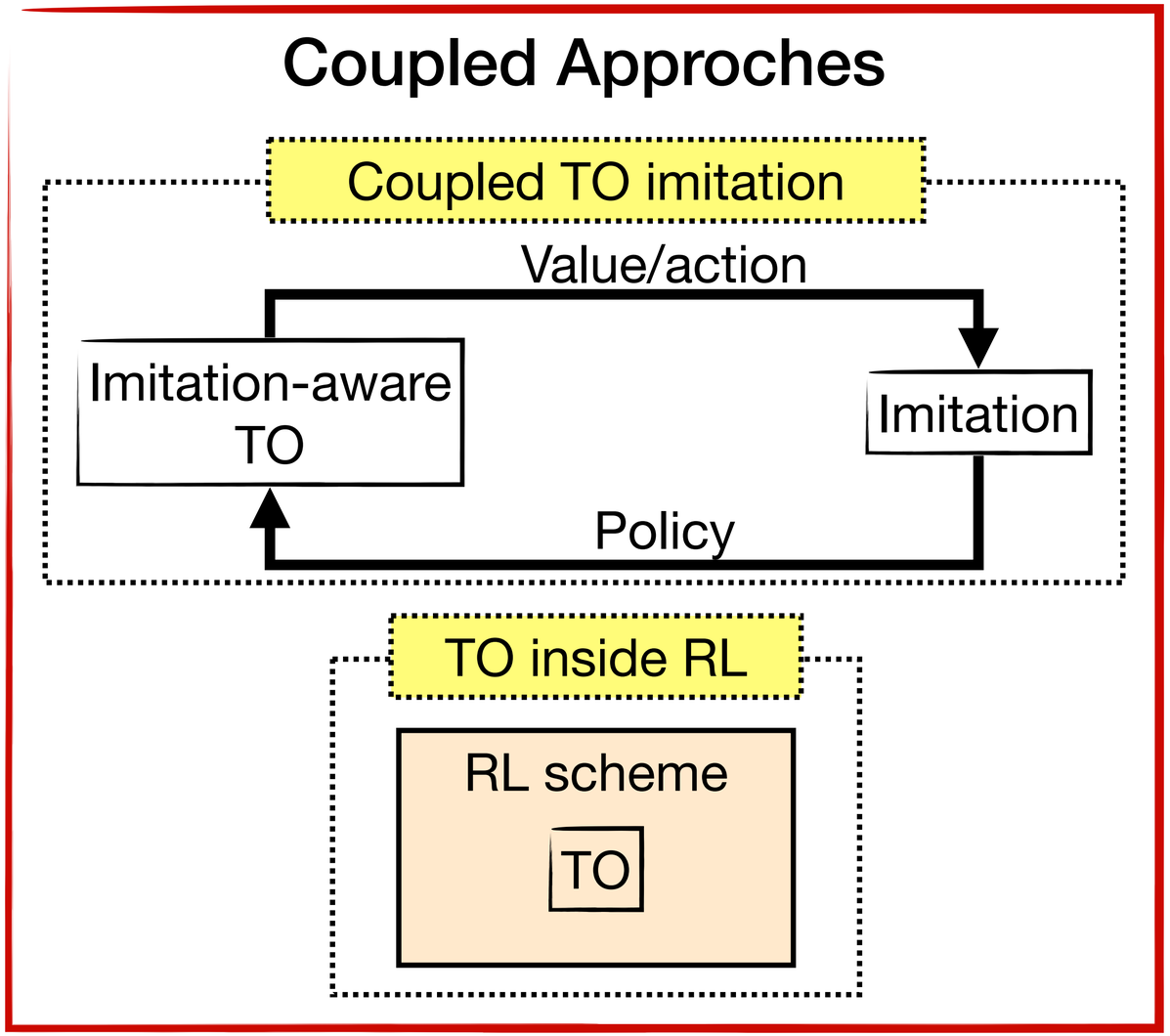
Trajectory Optimization

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

Architectures Combining RL and TO

A Taxonomy





* UNIVERSITY OF TRENTO
** UNIVERSITY OF NOTRE DAME
*** ENS | PSL

CACTO: Continuous Actor-Critic with Trajectory Optimization

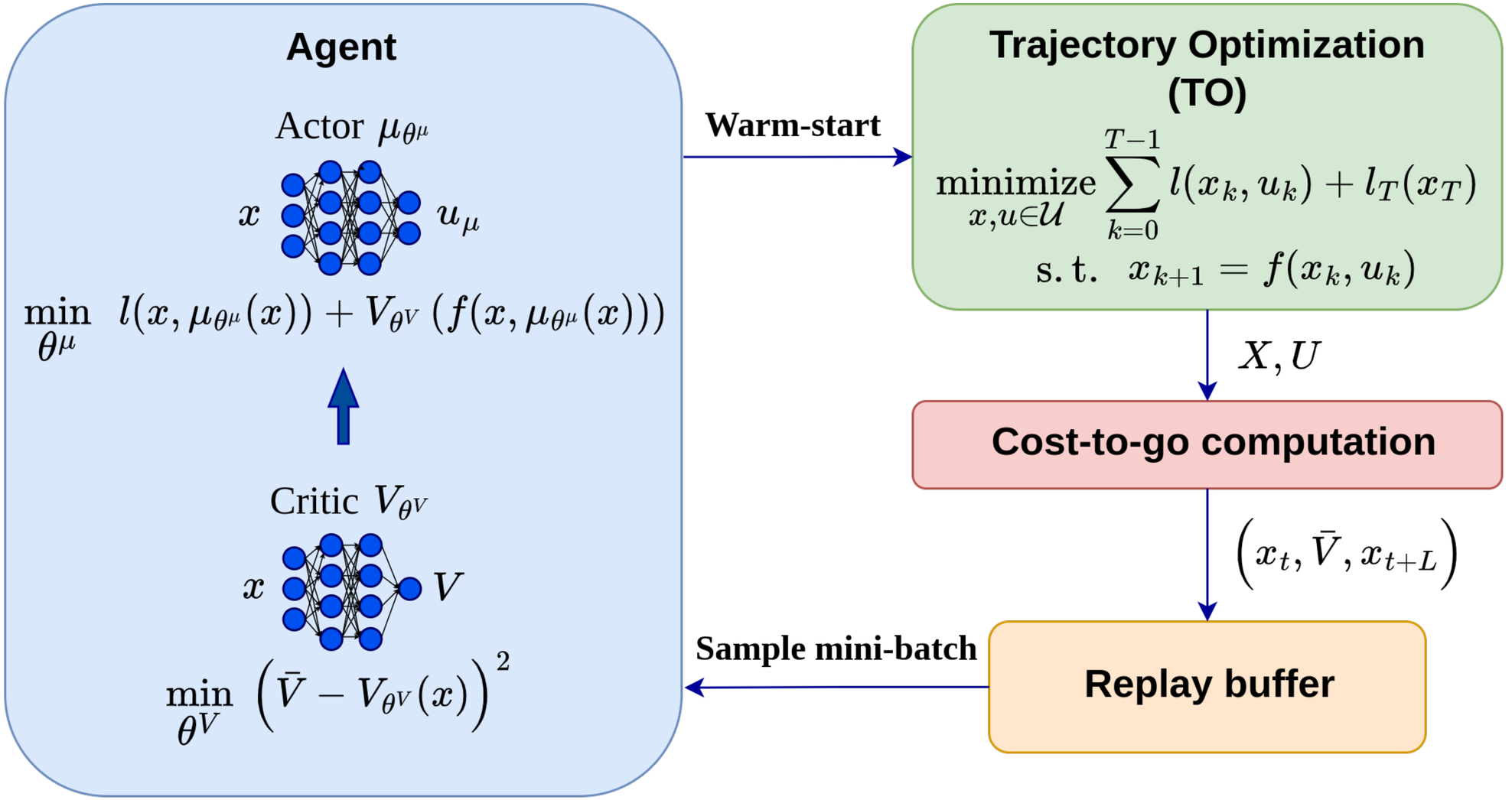
Gianluigi Grandesso*,
Elisa Alboni*,
Gastone Rosati Papini*,
Patrick Wensing,**
Justin Carpentier*,**
Andrea Del Prete*





[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
[3] (2026) CACTO-BIC: Scalable Actor-Critic Learning via Biased Sampling and GPU-Accelerated Trajectory Optimization. IEEE RA-L (under review)

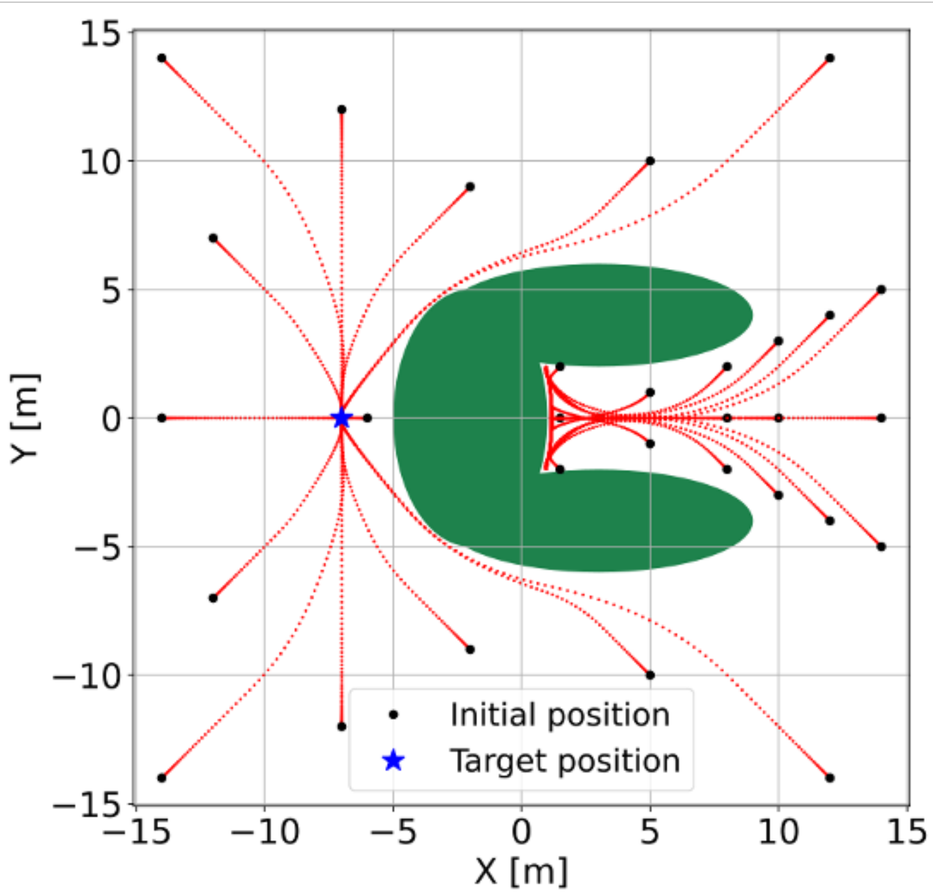
CACTO

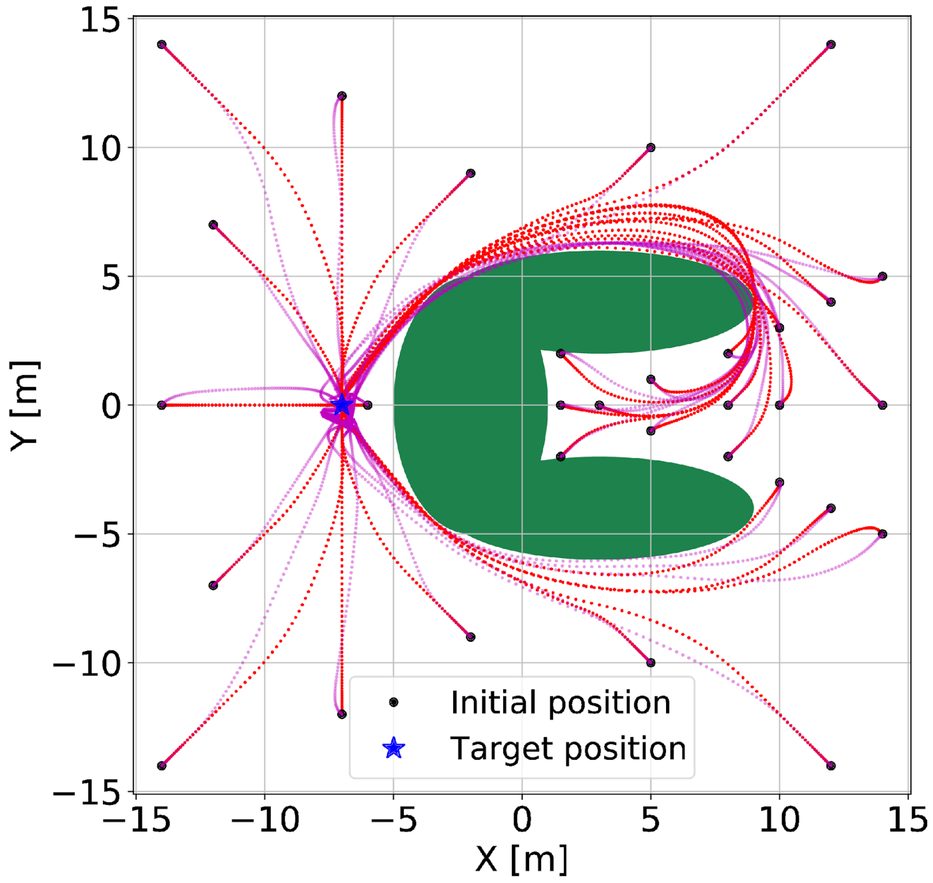


[1] Grandesso, Alboni, Rosati Papini, Wensing, Del Prete (2023). CACTO: Continuous Actor-Critic With Trajectory Optimization - Towards Global Optimality. IEEE Robotics and Automation Letters

Results

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

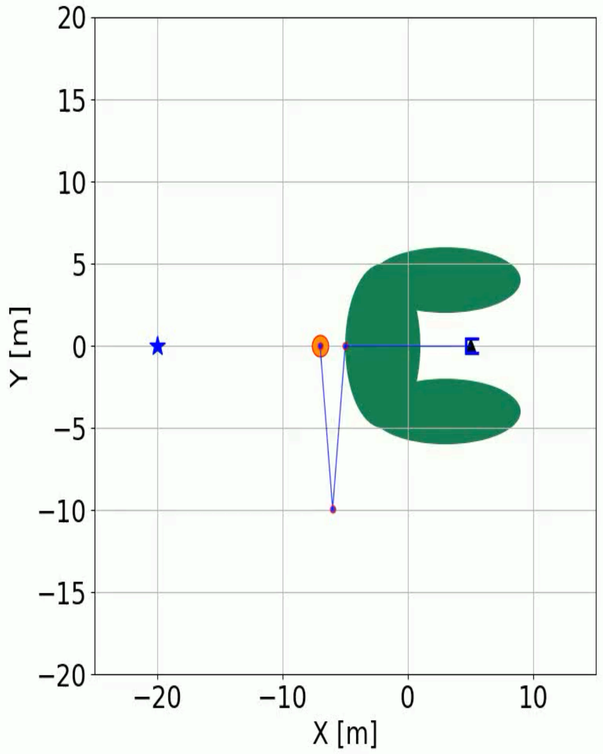




Systems: 2D single/double integrator, 6D car model, 3-joint manipulator

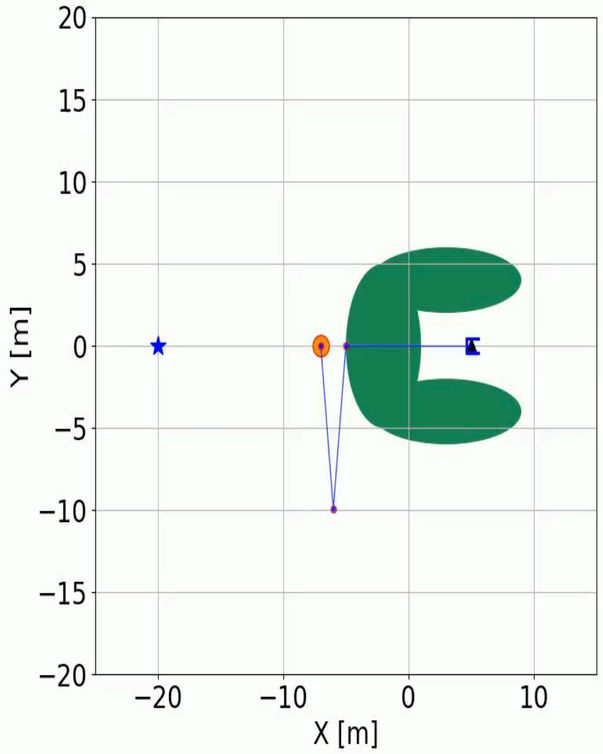
Results: 3-DoF Manipulator

Initial Conditions
warm-start



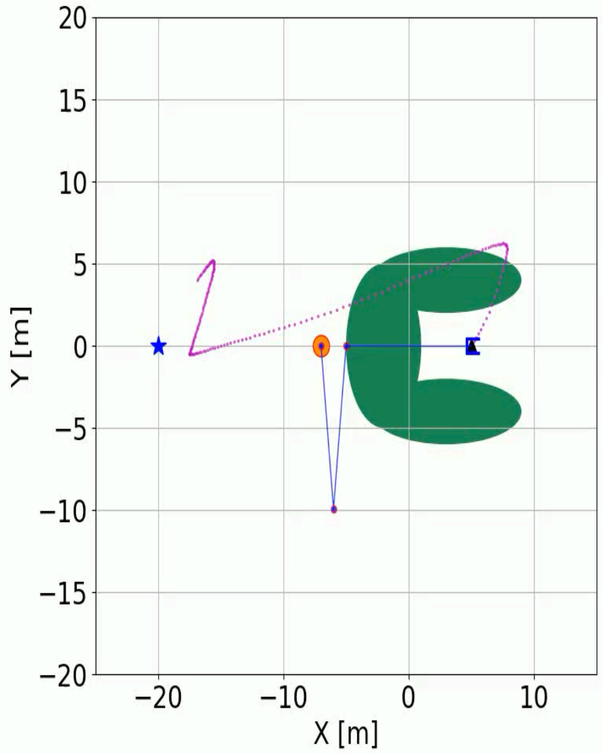
Cost = 70800

Random
warm-start



Cost = 88647

CACTO
warm-start

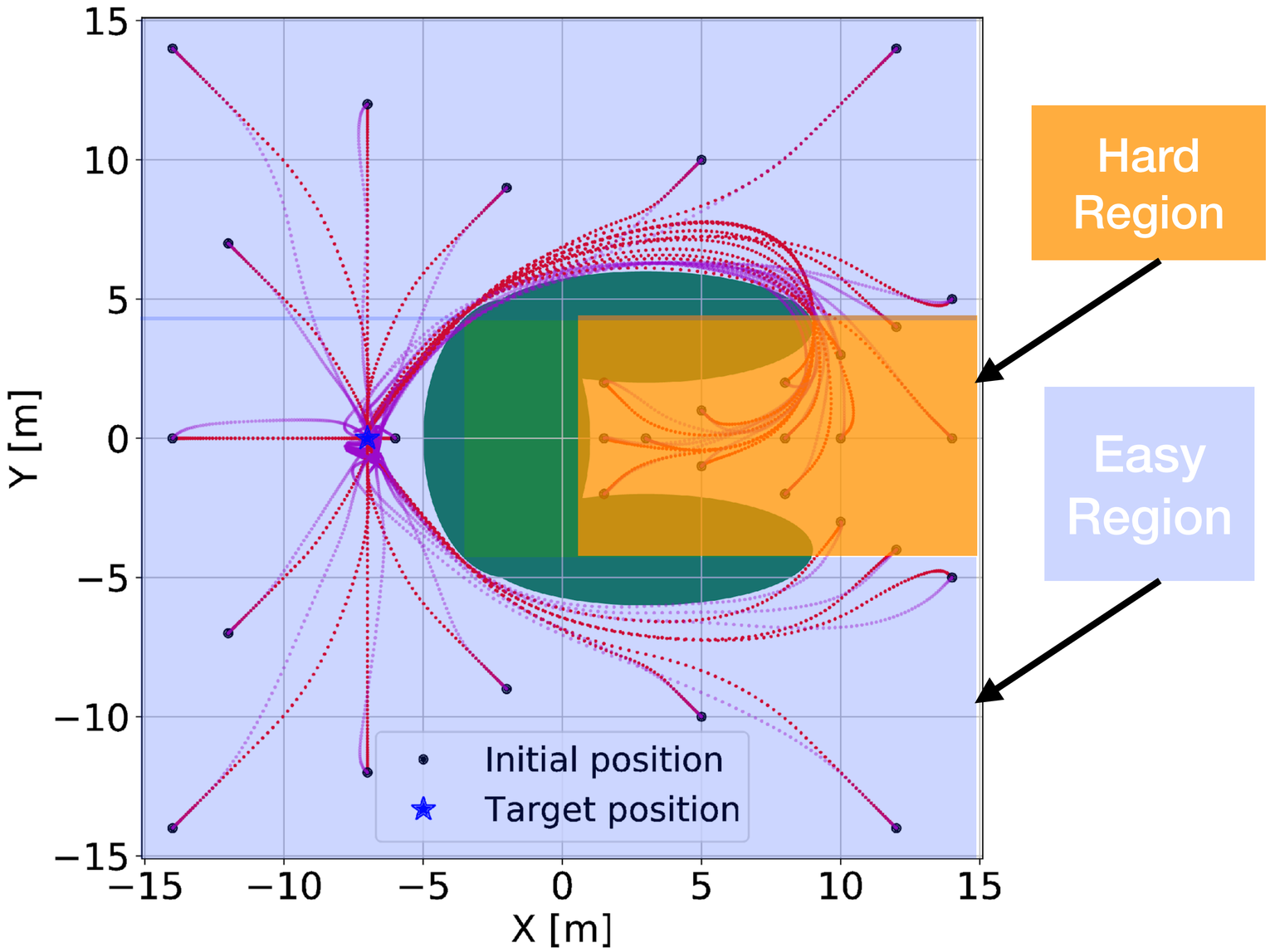


Cost = -145875

Can we improve sample efficiency?

CACTO - Biased Initial Conditions

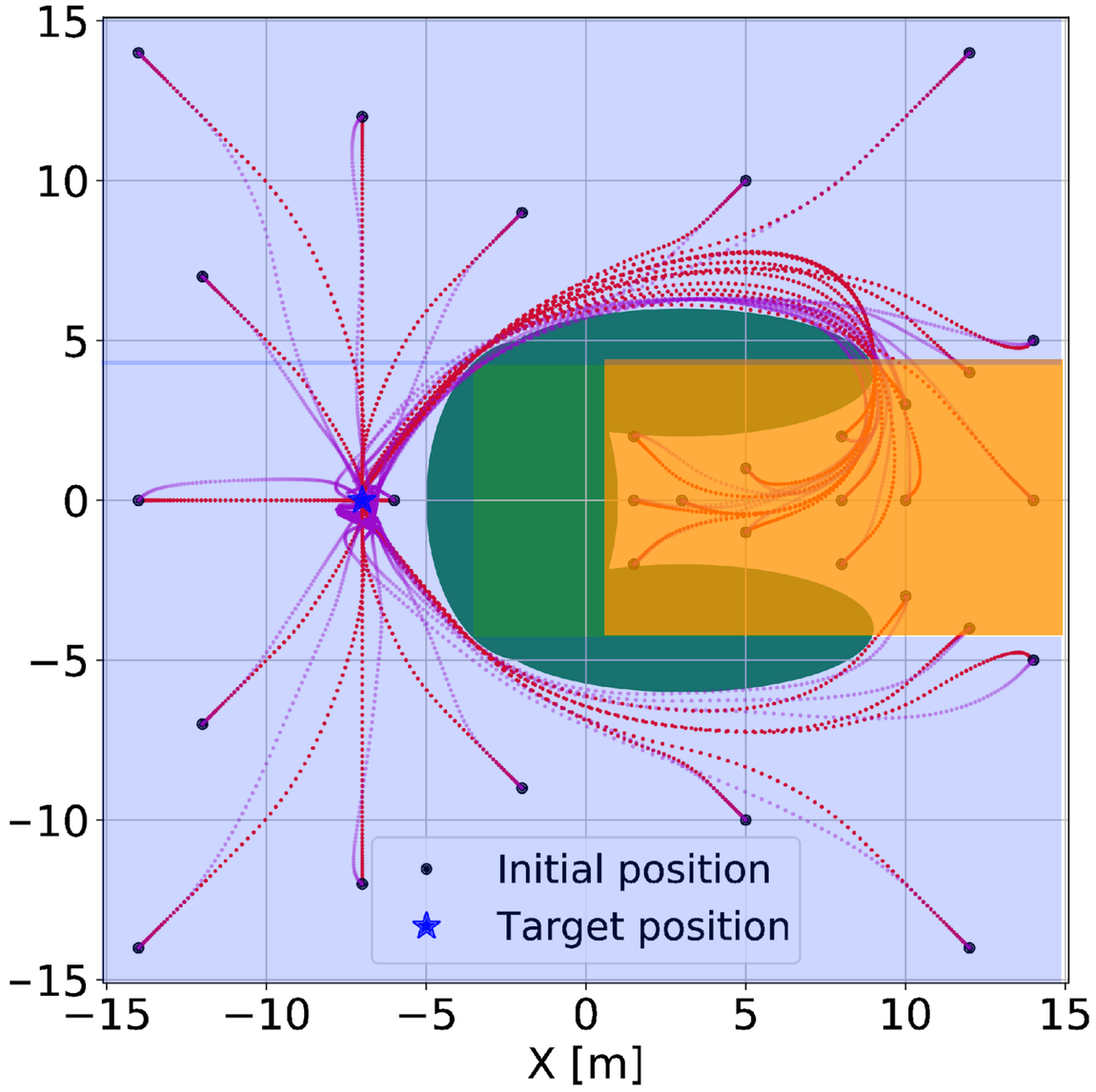
- TO is already globally optimal in some regions
- Policy improvement is only possible in some regions
- Can we identify these regions and bias sampling of initial states in these regions?



How to find suboptimal regions?

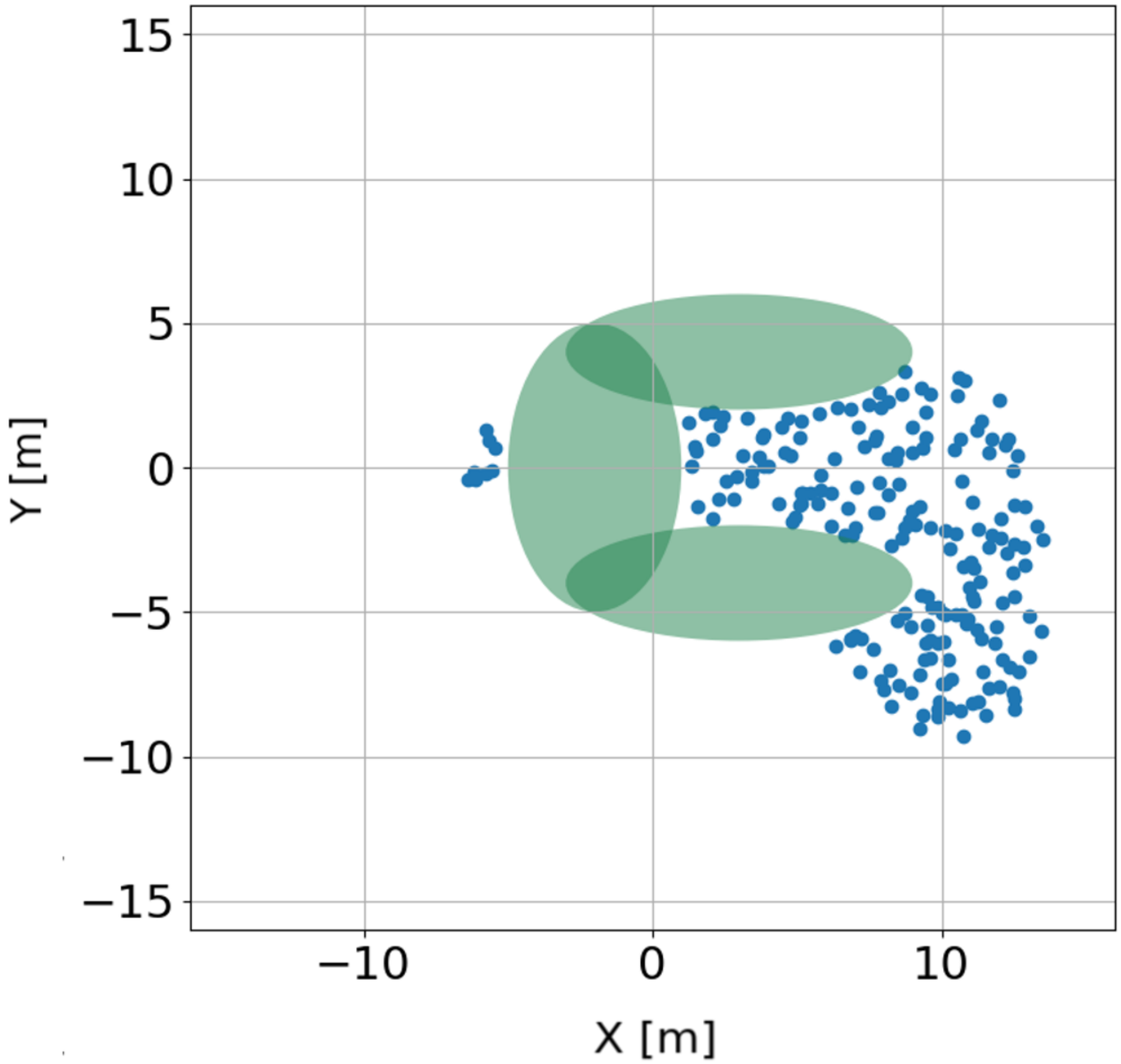
- Find regions where TO's solution switches from global to local optimality
- Value function is discontinuous
- Critic network makes larger errors

- Idea: introduce new network V^{std} to predict critic's uncertainty
- $$\min_{\theta} \ln(V^{\text{std}}(x|\theta)^2) + \frac{(\bar{V} - V(x|\theta^V))^2}{V^{\text{std}}(x|\theta)^2}$$

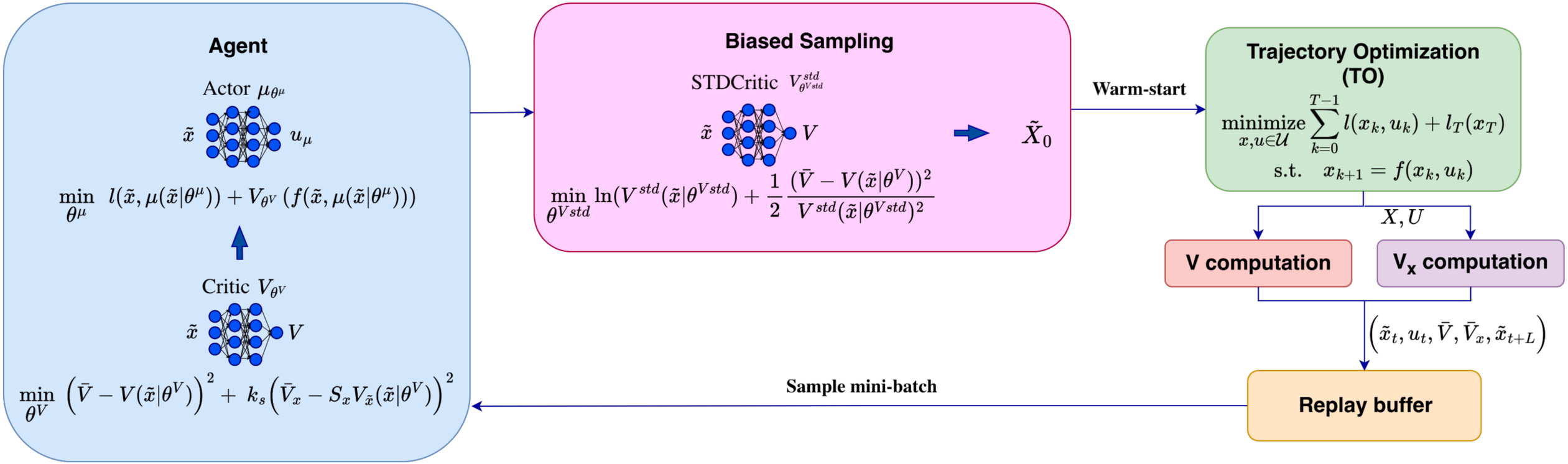


How to find suboptimal regions?

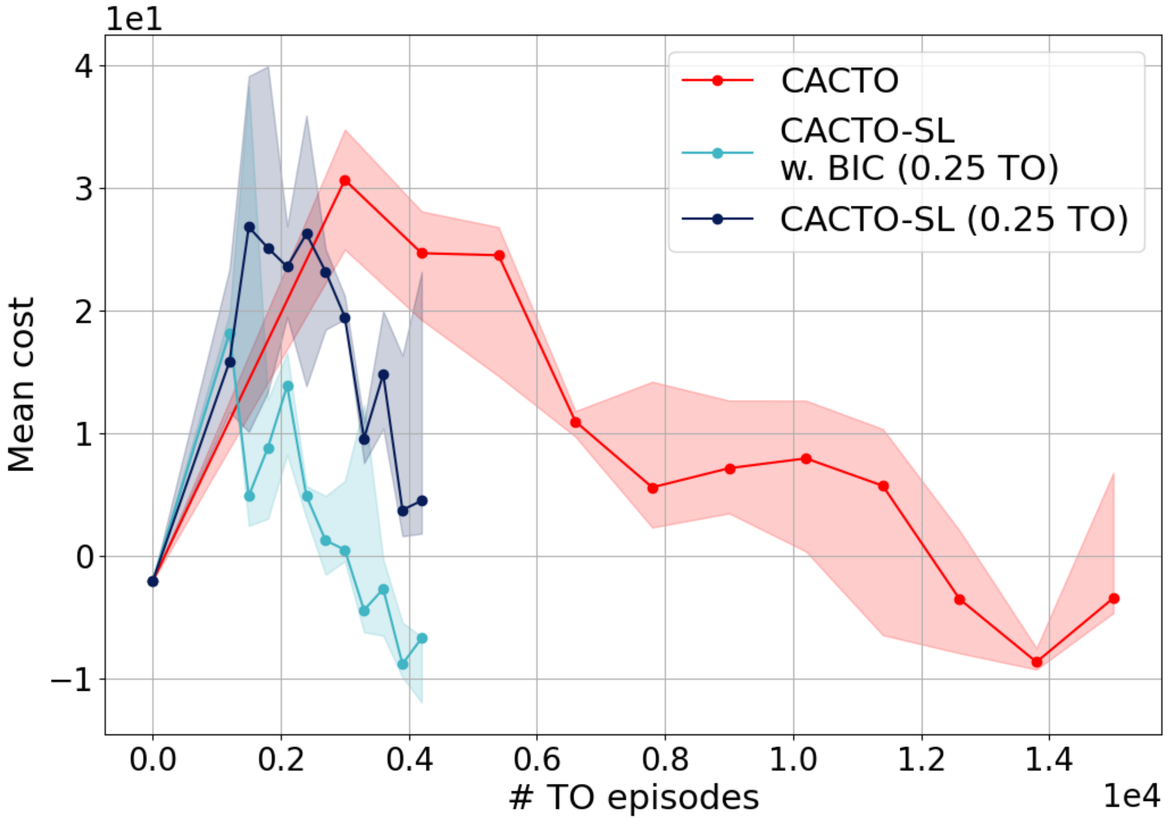
- Sample 10N states
- Choose N states with highest V^{std}



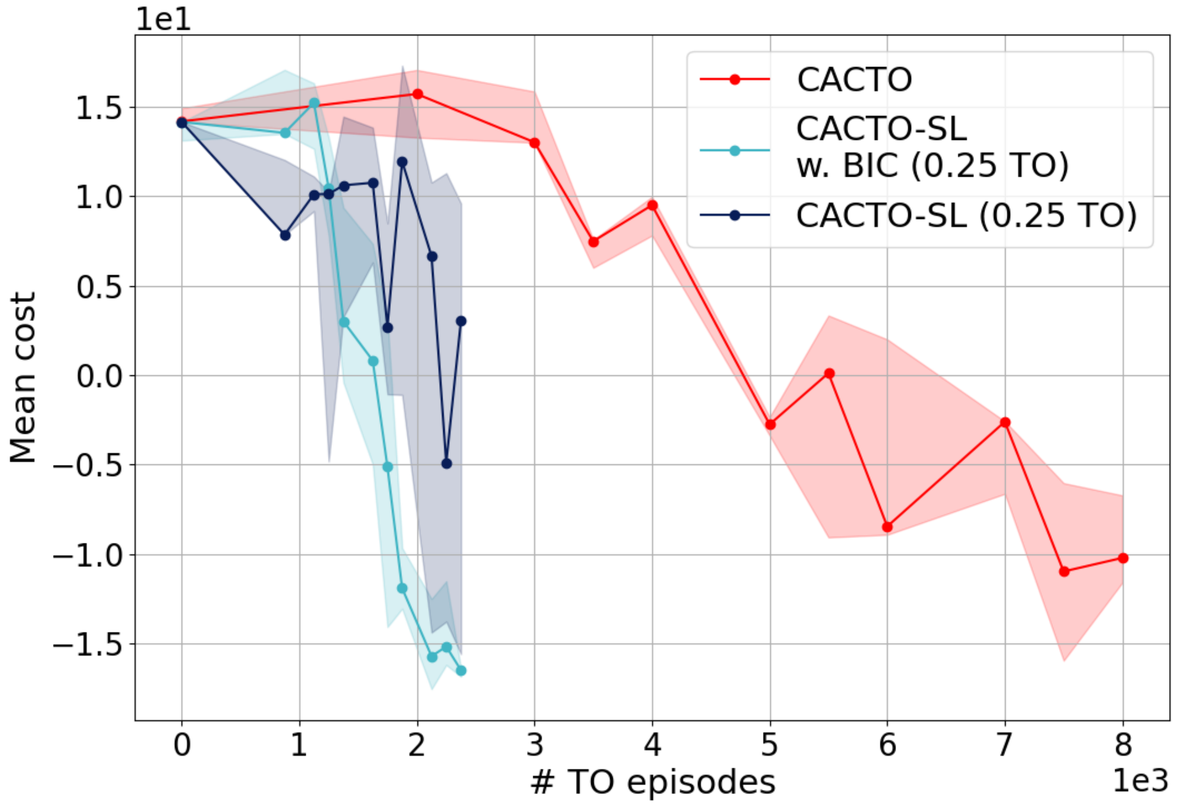
CACTO-BIC


Agent
Actor μ_{θ^μ}
$\tilde{x}$
u_μ
$\min_{\theta^\mu} \; l(\tilde{x}, \mu(\tilde{x}|\theta^\mu)) + V_{\theta^V}(f(\tilde{x}, \mu(\tilde{x}|\theta^\mu)))$
Critic V_{θ^V}
$\tilde{x}$
V
$\min_{\theta^V} \left(\bar{V} - V(\tilde{x}|\theta^V)\right)^2 + k_s\left(\bar{V}_x - S_x V_{\tilde{x}}(\tilde{x}|\theta^V)\right)^2$
Biased Sampling
STDCritic $V^{std}_{\theta^{Vstd}}$
$\tilde{x}$
V
$\min_{\theta^{Vstd}} \ln(V^{std}(\tilde{x}|\theta^{Vstd}) + \frac{1}{2}\frac{(\bar{V} - V(\tilde{x}|\theta^V))^2}{V^{std}(\tilde{x}|\theta^{Vstd})^2}$
$\tilde{X}_0$
Warm-start
Trajectory Optimization (TO)
$\underset{x,u\in\mathcal{U}}{\text{minimize}} \sum_{k=0}^{T-1} l(x_k, u_k) + l_T(x_T)$
s.t. $x_{k+1} = f(x_k, u_k)$
X, U
V computation
V_x computation
$\left(\tilde{x}_t, u_t, \bar{V}, \bar{V}_x, \tilde{x}_{t+L}\right)$
Replay buffer
Sample mini-batch

CACTO-BIC - Results



3-DoF Manipulator

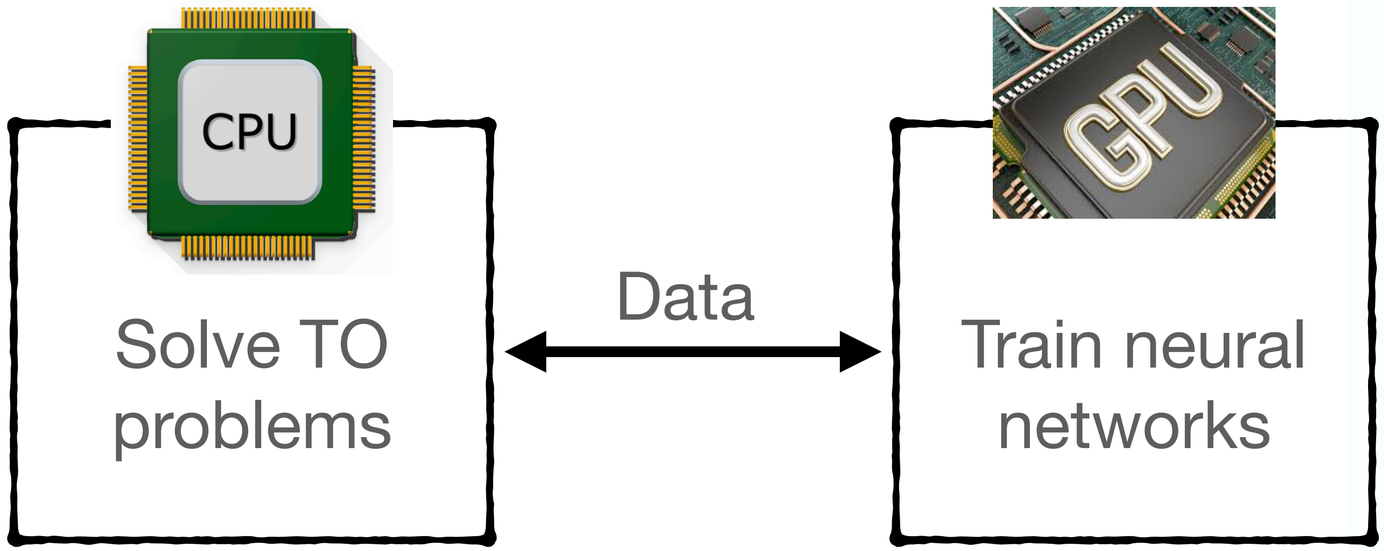


6D Dubins Car

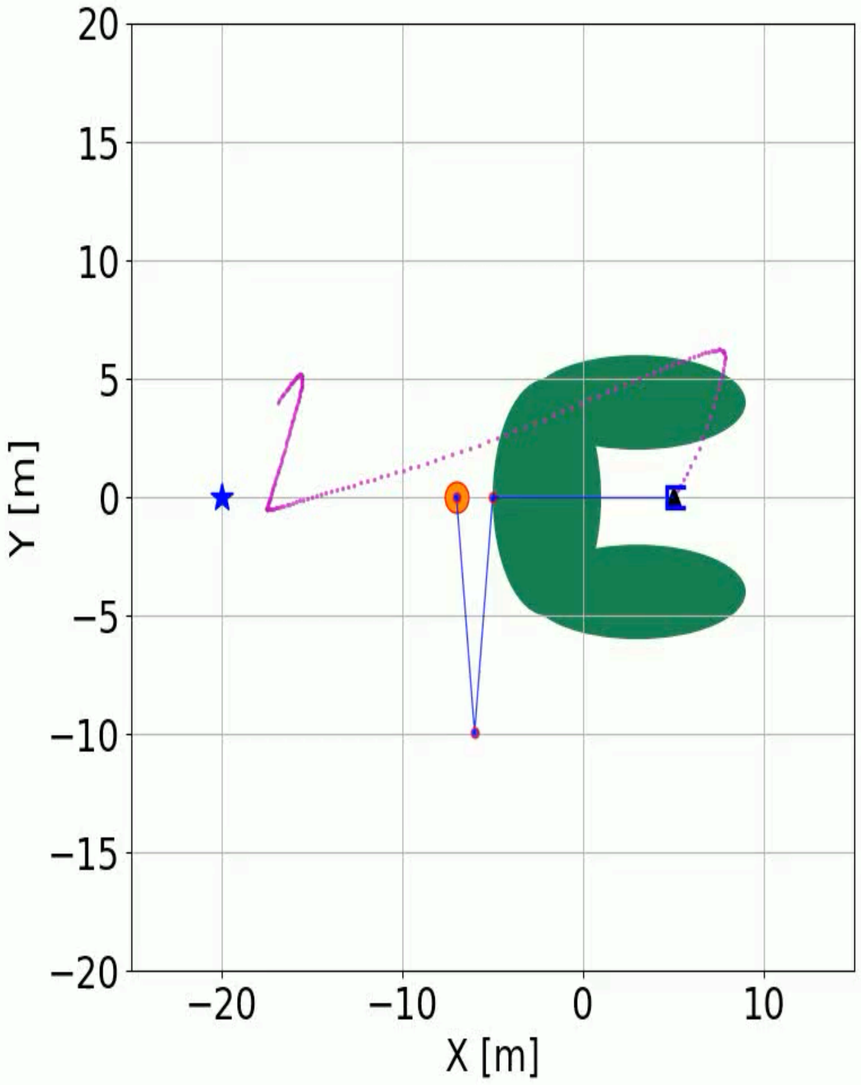
Same network's updates, 25% TO problems

How to scale it up?

- So far limited to low-dimensional problems
  - Good sample efficiency
  - Bad computation time (5 hours for 3-DoF arm)



<10% COMPUT. TIME



CACTO on GPU

CasADi
Dynamics and *Cost* definition

→

JAXADI
Conversion in JAX

→

TRAJAX - iLQR
Batch execution on GPU
With
- cost: if(t_i <= T_N, *Cost*, 0)
- max iteration = *MaxIter*

- New JAX-based implementation
- Solve TO problems in batches
  - 300-500 problems
  - iLQR algorithm
- 3 issues arise

1

- All problems have same horizon:
  - Set running cost to zero after randomly sample time T

2

- All problems use same number of iterations
  - Set max-iter = 99-th percentile of iteration counts (200-700)

3

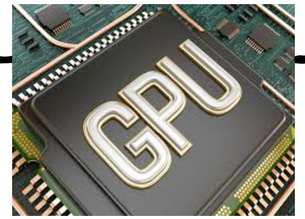
- Classic trial-and-error Hessian regularization schemes are inefficient:
  - 1-shot SVD-based regularization

CACTO on GPU - Computation times

~1 TFLOPS ~90 TFLOPS

	CPU (1 core) [min.]	CPU (10 cores) [min.]	GPU RTX6000 [min.]
Double integrator	23	22	<1
Dubins car	224	214	6
Manipulator	335	315	7

31-55X SPEED UP

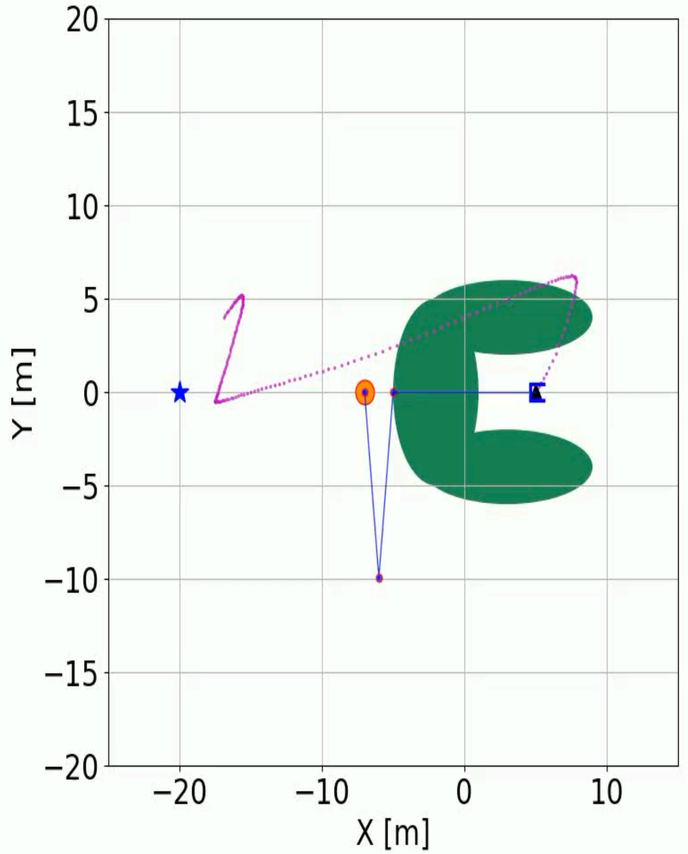


Solve TO problems

Train neural networks

40-80% COMPUT. TIME

20-60% COMPUT. TIME



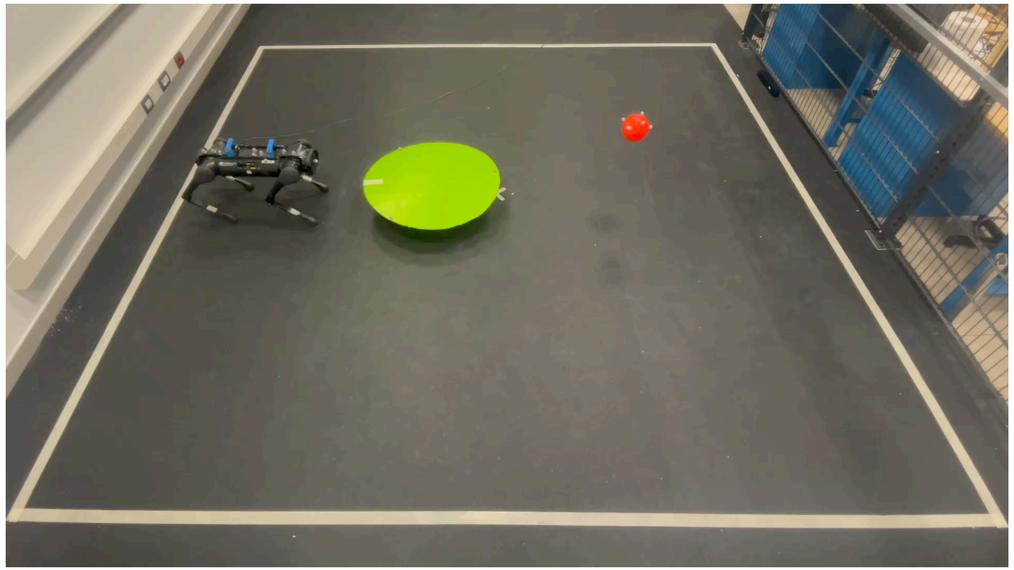
CACTO on high-dimensional problem

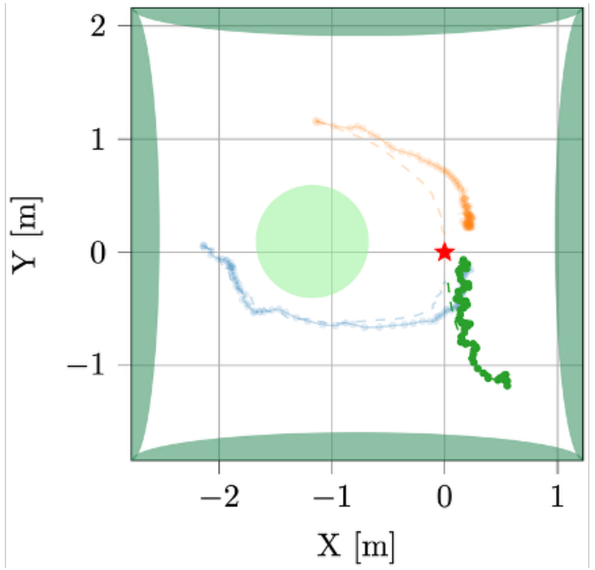
- Simplified 2D quadruped model (~LIPM)
- Variable footstep duration
- Dynamic obstacle avoidance
- 16 minutes of training
- Low-level policy mapping to full model

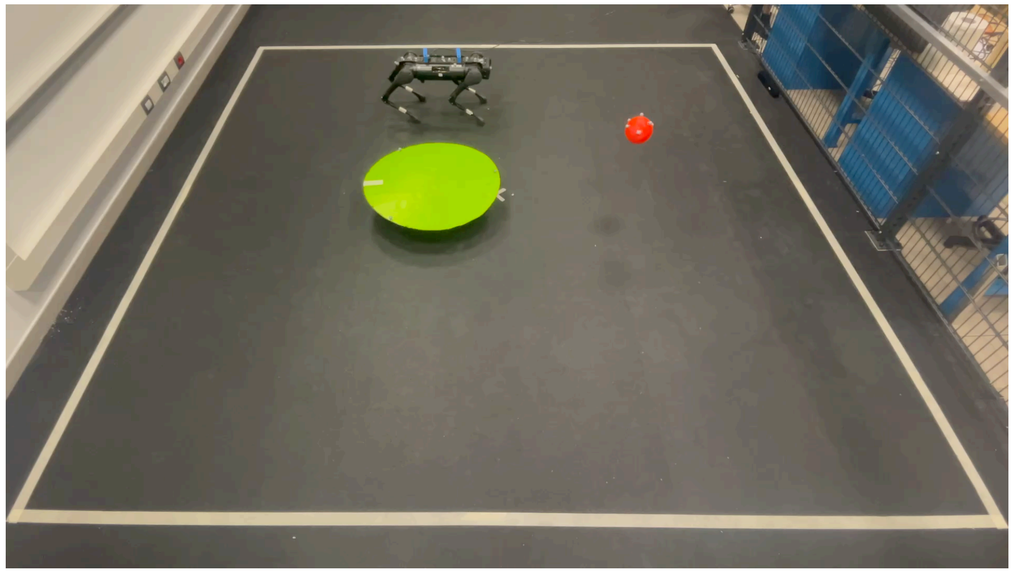
- **15-dimensional state space:**
  - 2D CoM pos. + 2D CoM vel.
  - 2D pos. of support feet
  - 6D obstacle positions
  - contact phase
- **6-dimensional control space:**
  - 2D pos. of next support feet
  - 1D center of pressure
  - 1D footstep duration

Hardware experiments

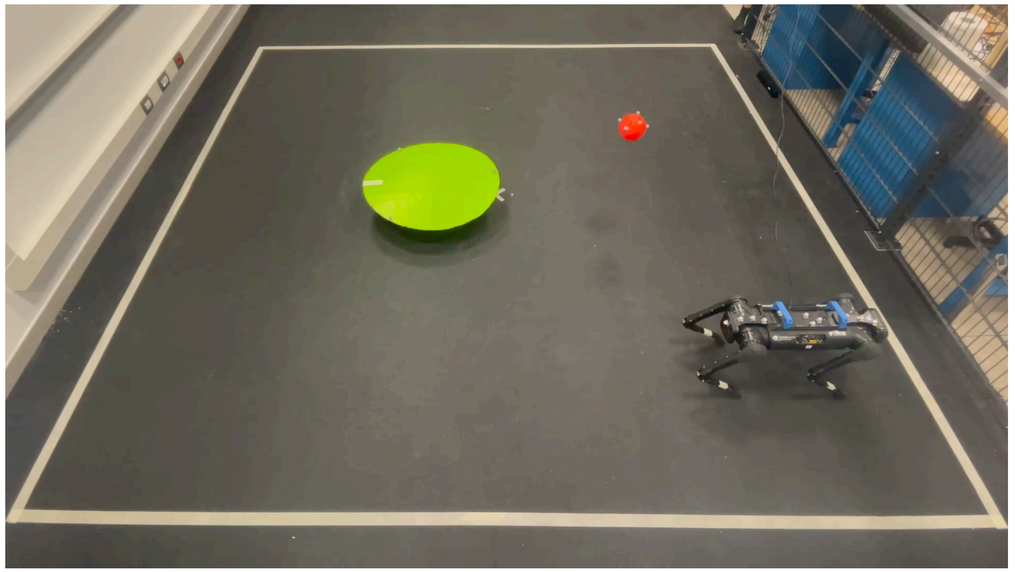
Static target and obstacle





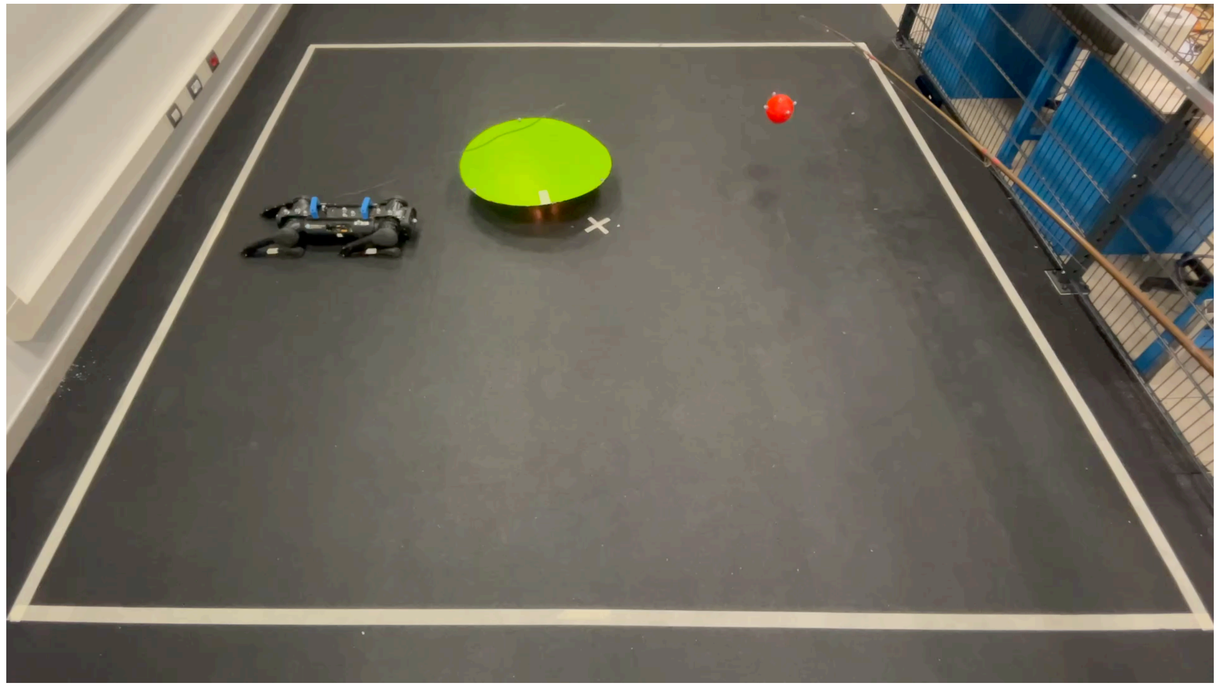


Speed X2

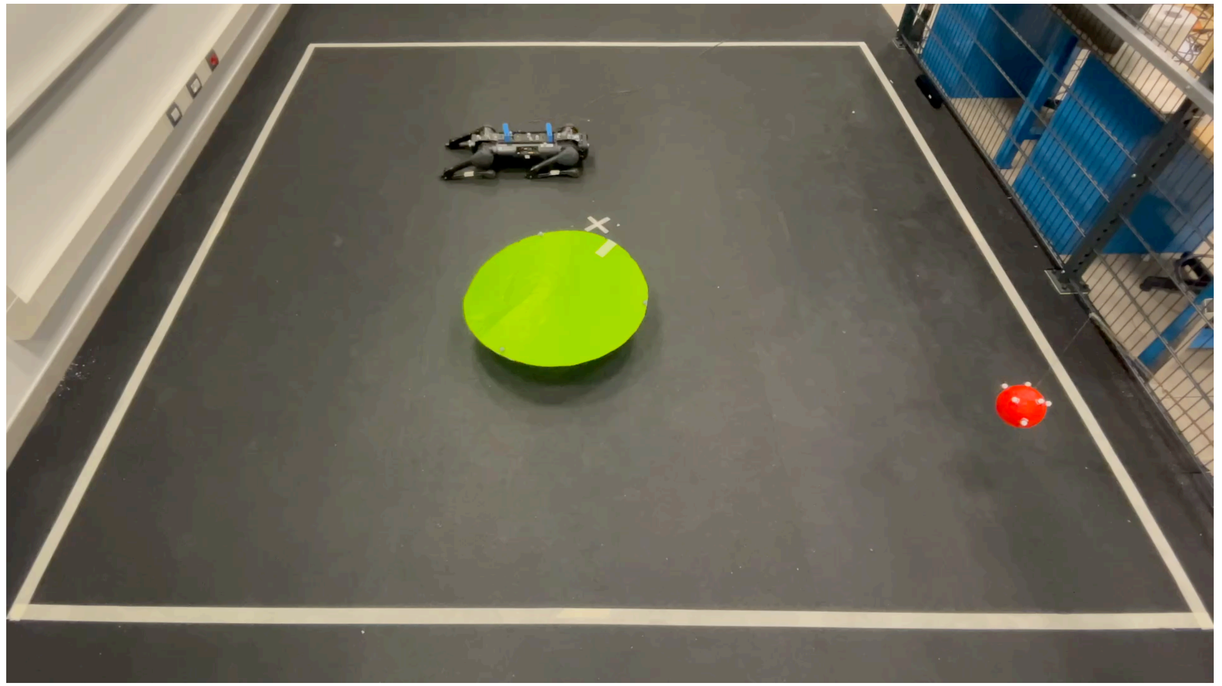


Hardware experiments

Dynamic target and obstacle

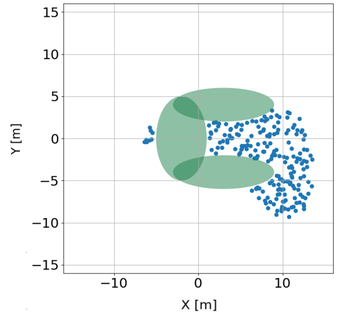


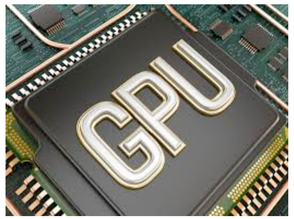
Speed X2



Speed X2

CACTO - Conclusions





- Novel RL scheme exploiting Trajectory Optimization
  - improved sample efficiency thanks to biased initial conditions
  - 30-50x speed up thanks to GPU implementation

On-going/future work

- Handle discontinuous dynamics => Sampling-based optimization
- Handle state constraints => Augmented Lagrangian
- Handle uncertainties => Domain randomization
- Handle sensor feedback => Sensor-based policy and Value

UNIVERSITY OF TRENTO

Beyond Terminal Constraints: Reliable Robot Control with Learned Safe Sets

Gianni Lunardi
Elias Fontanari
Asia La Rocca
Matteo Saveriano
Andrea Del Prete



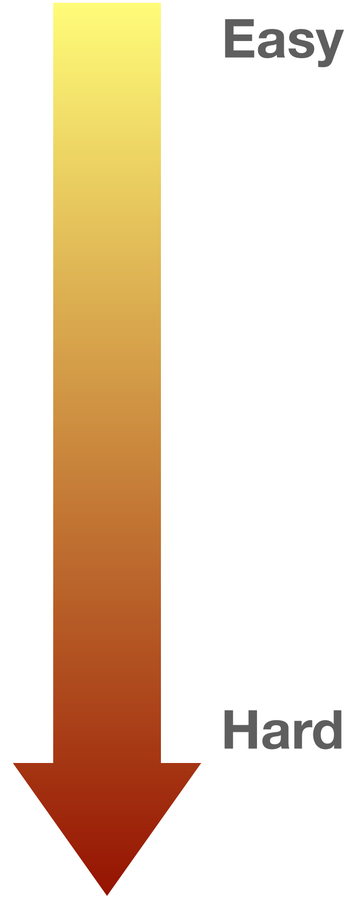

[1] Receding-Constraint MPC using a Learned Approximate Control-Invariant Set. IEEE ICRA 2024
[2] Parallel-Constraint MPC: Exploiting Parallel Computation for Improving Safety. IEEE ICRA 2025
[3] Beyond Terminal Constraints: Reliable Robot Control with Learned Safe Sets. IEEE T-Ro 2026 (under review)

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

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



Model Predictive Control

Trajectory Optimization inside the control loop

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

$$\text{subject to} \quad x_0 = x_{init}$$

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

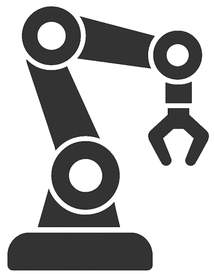
$$x_{i+1} \in \mathcal{X}, u_i \in \mathcal{U} \qquad i = 0 \ldots N-1$$

$x = x_{init}$

u_0

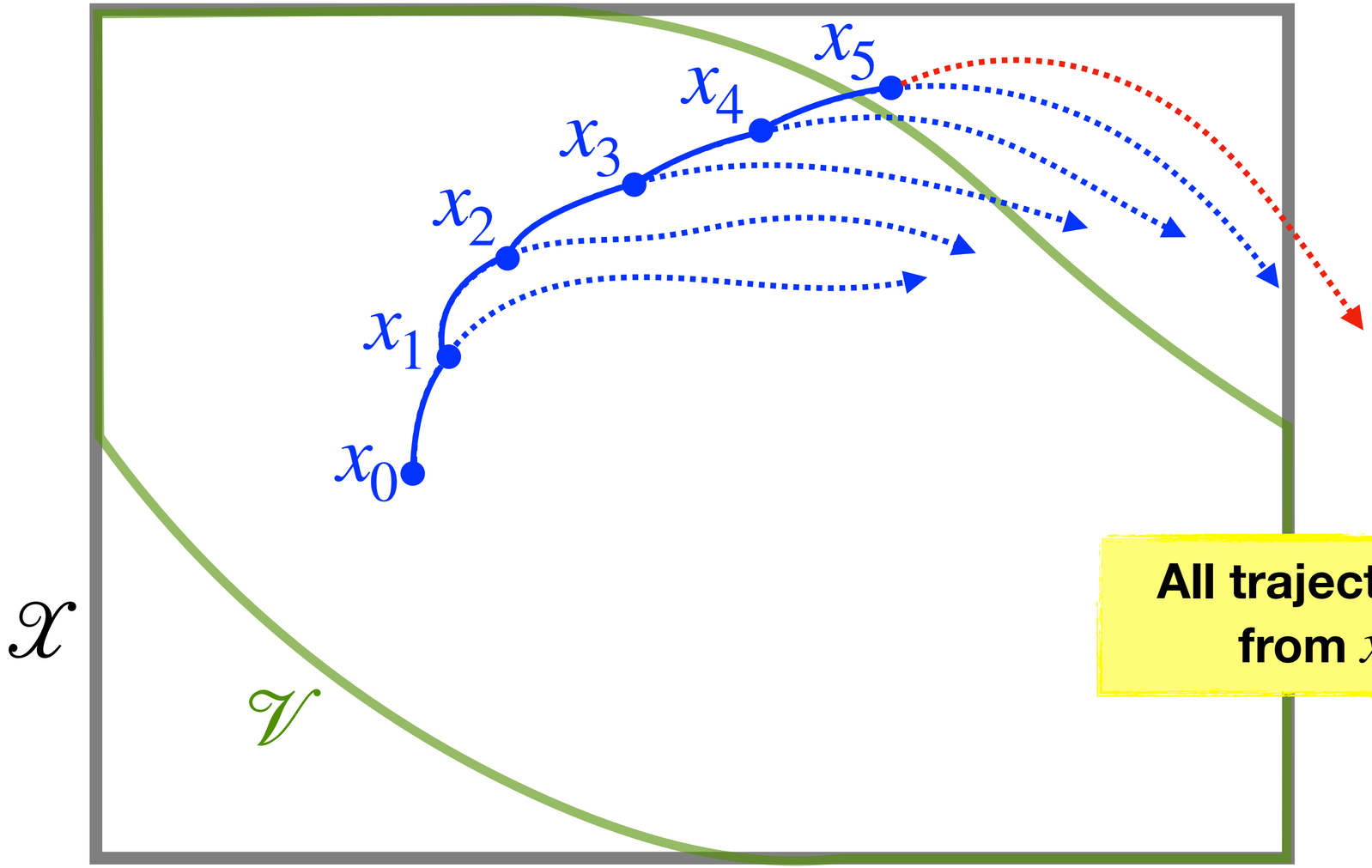
Robot

$f(x, u)$



Model Predictive Control

Can it ensure safety? No

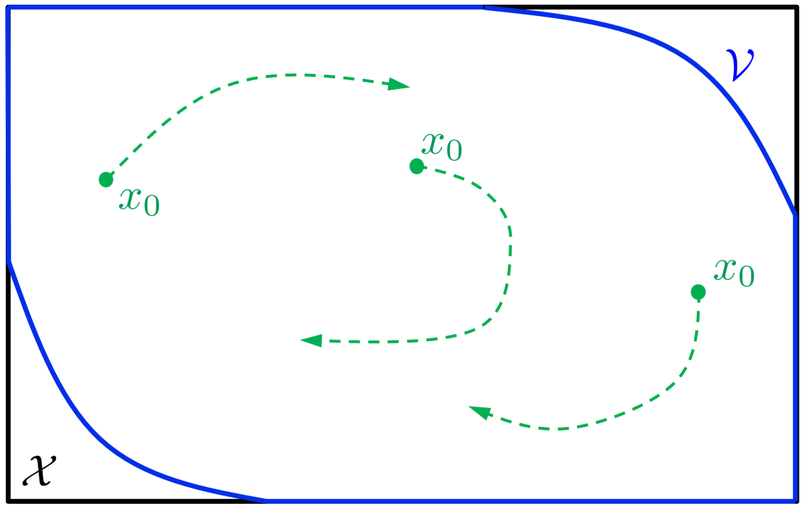


All trajectories starting from x_5 leave $\mathcal{X}$

Safety Guarantees

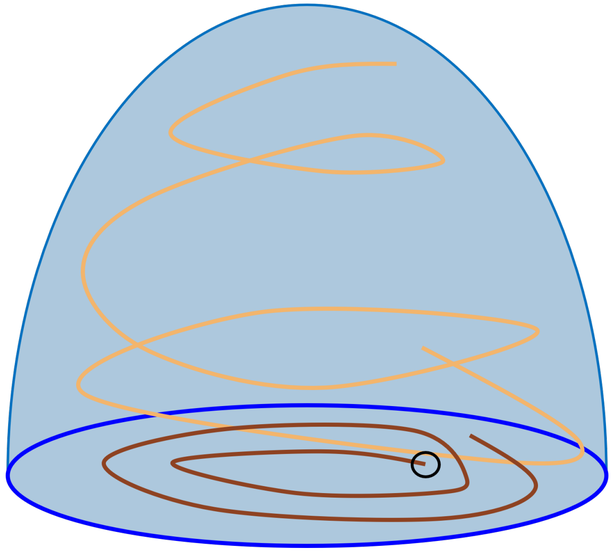
State of the art

Control-Invariant Sets (CIS)



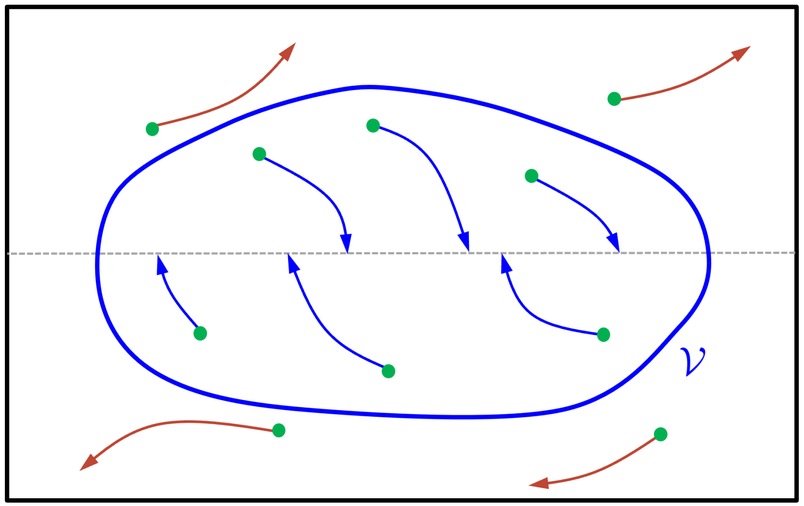
Model Predictive Control

Control Barrier Functions (CBF)



Quadratic Program

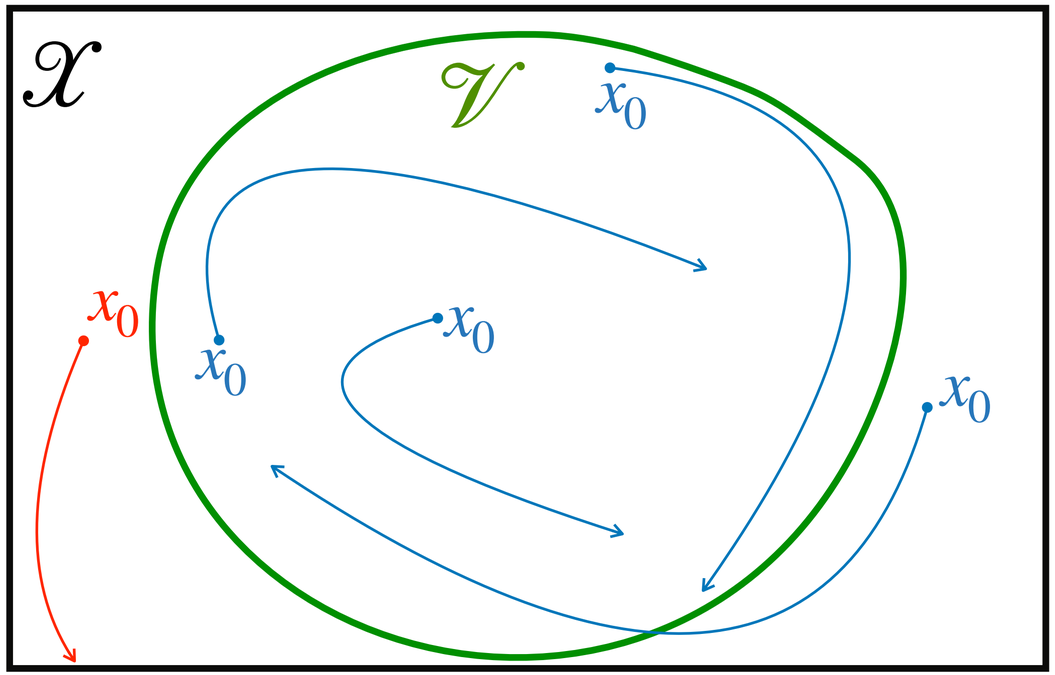
Back-up Policies (BUP)



Reinforcement Learning

Safety via Control Invariant Sets

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



Recursive Feasibility

Model Predictive Control (MPC)

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

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

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

→

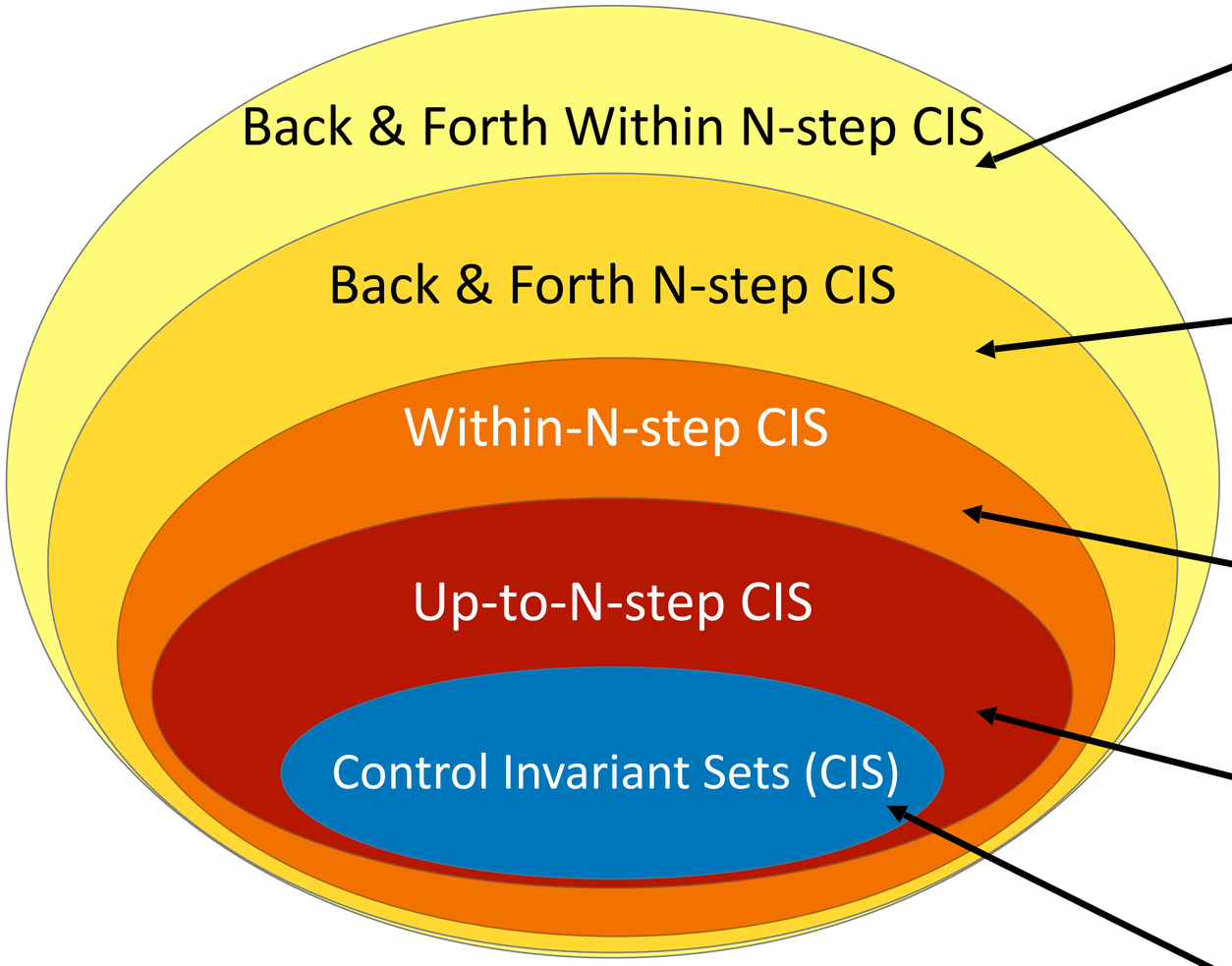
MPC problem can become unfeasible using $\hat{\mathscr{V}}$ instead of $\mathscr{V}$!

Beyond Control Invariant Sets (CIS)

- 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
    - → all safety guarantees are lost!

Do we really need Control Invariant Sets to ensure safety?

Beyond Control Invariant Sets



Come back to the set within N steps moving back and forth in time ⟷ **Parallel Constraint MPC**

Come back to the set in exactly N steps moving back and forth in time ⟷ **Terminal Constraint MPC w/ Safe Abort**

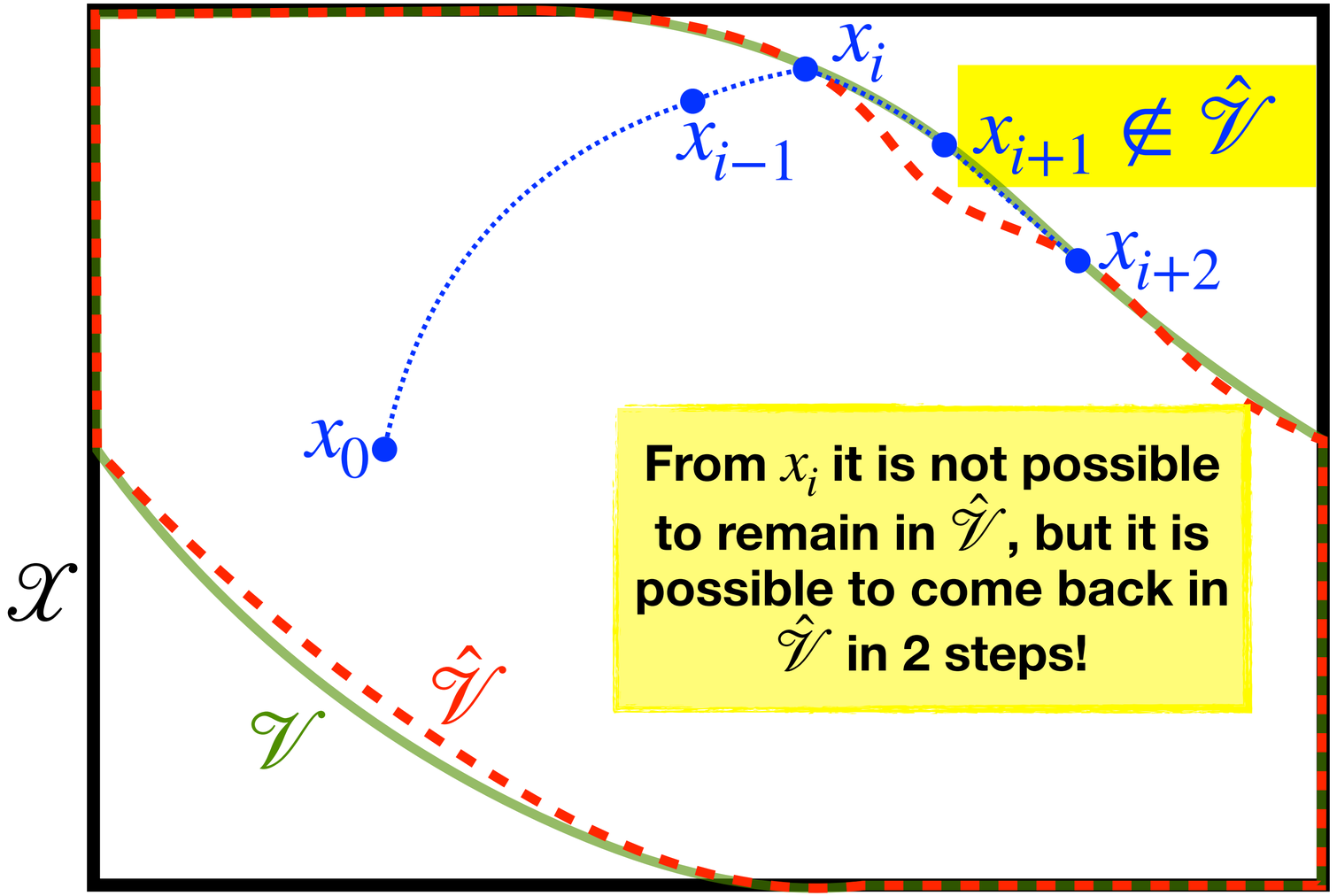
Come back to the set within N steps ⟷ **Fixed Receding Constraint MPC**

Come back to the set in exactly K steps with K up to N ⟷ **Receding Constraint MPC**

Remain inside in the future ⟷ **Classic Terminal Constraint MPC**

N-Step Control Invariant Set

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



Beyond Control Invariant Sets

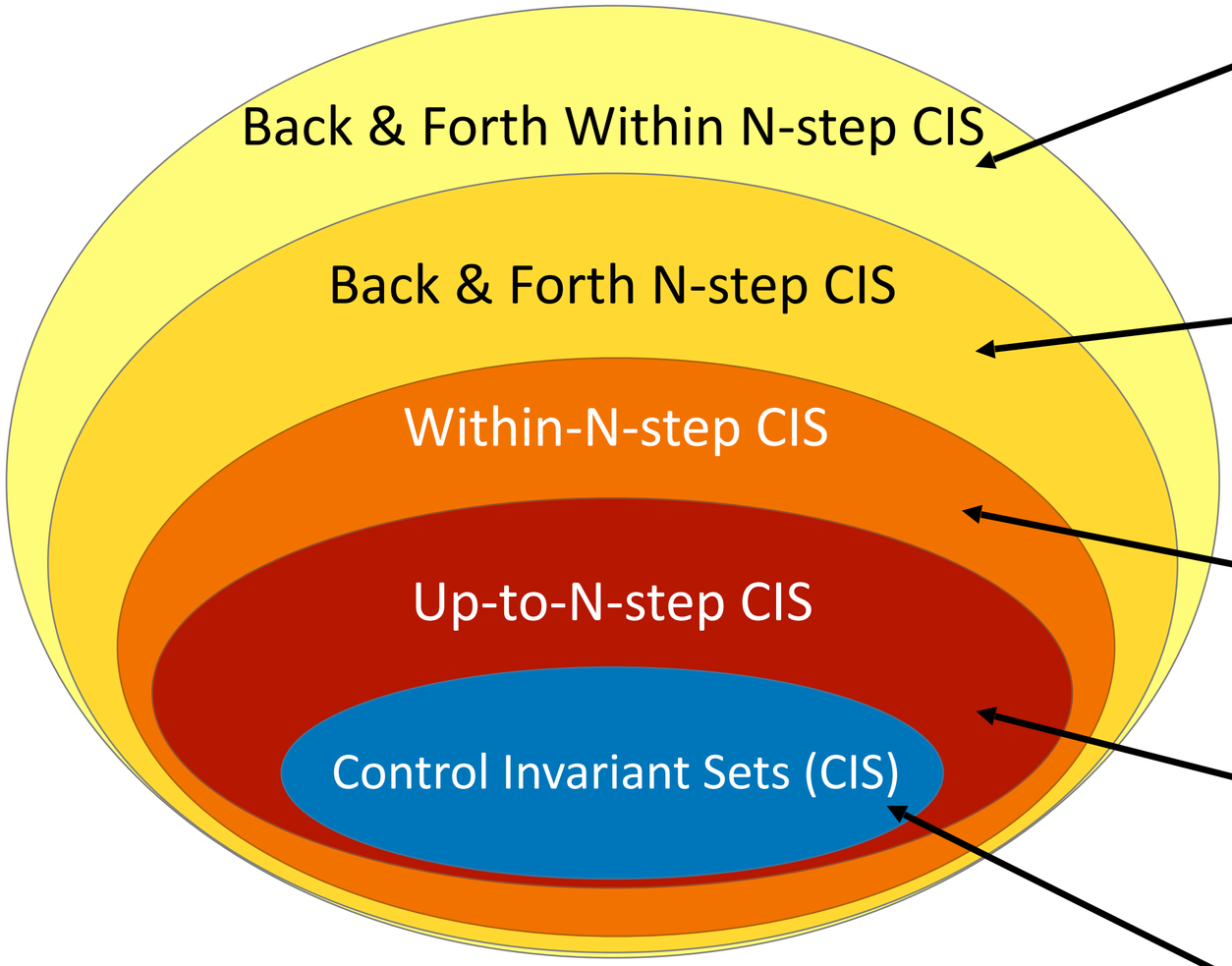
Two hypotheses:

New controllers work better even with approximate CIS

✅

New sets are easier to compute

Beyond Control Invariant Sets



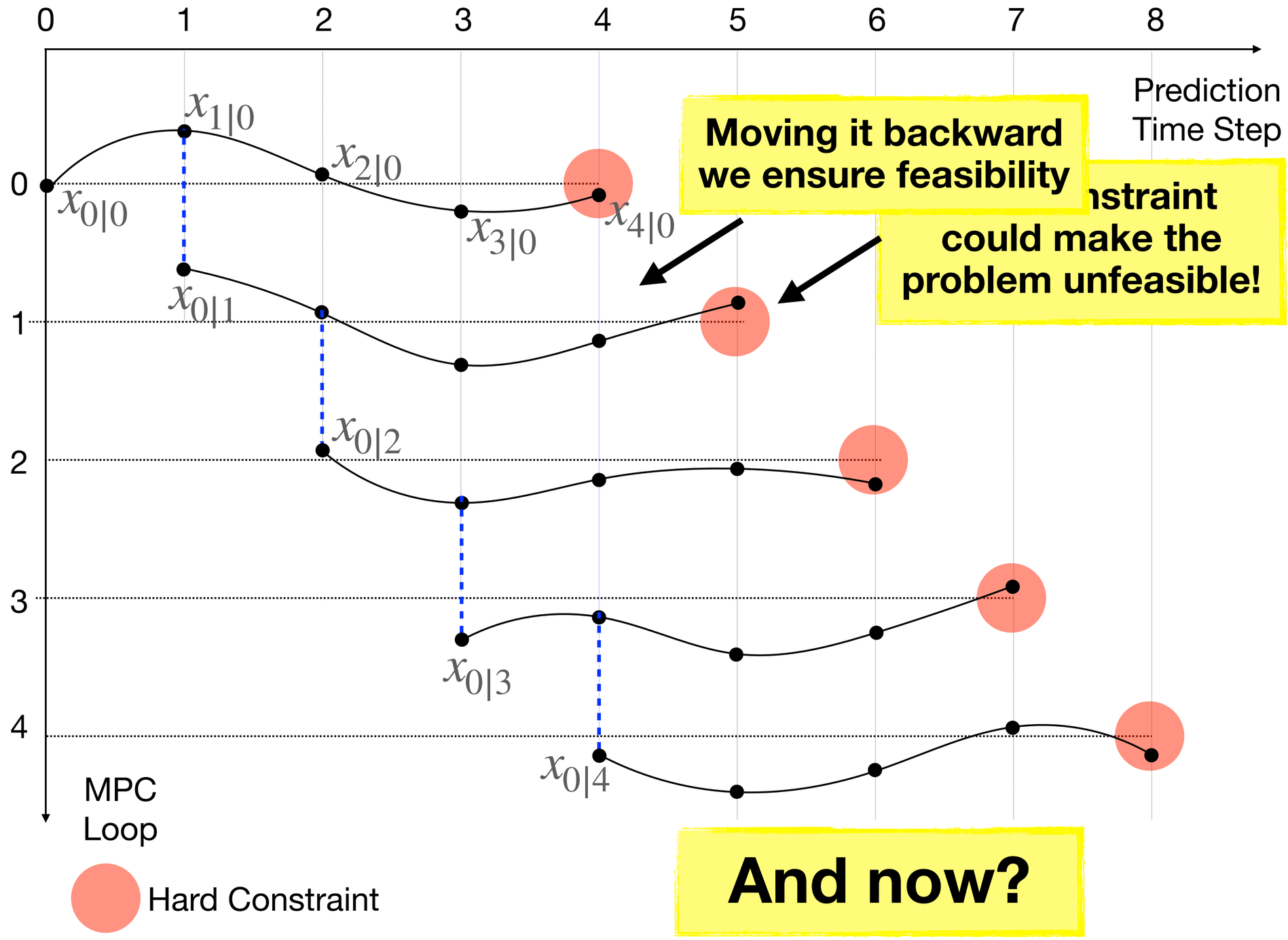
Come back to the set within N steps moving back and forth in time ⟷ **Parallel Constraint MPC**

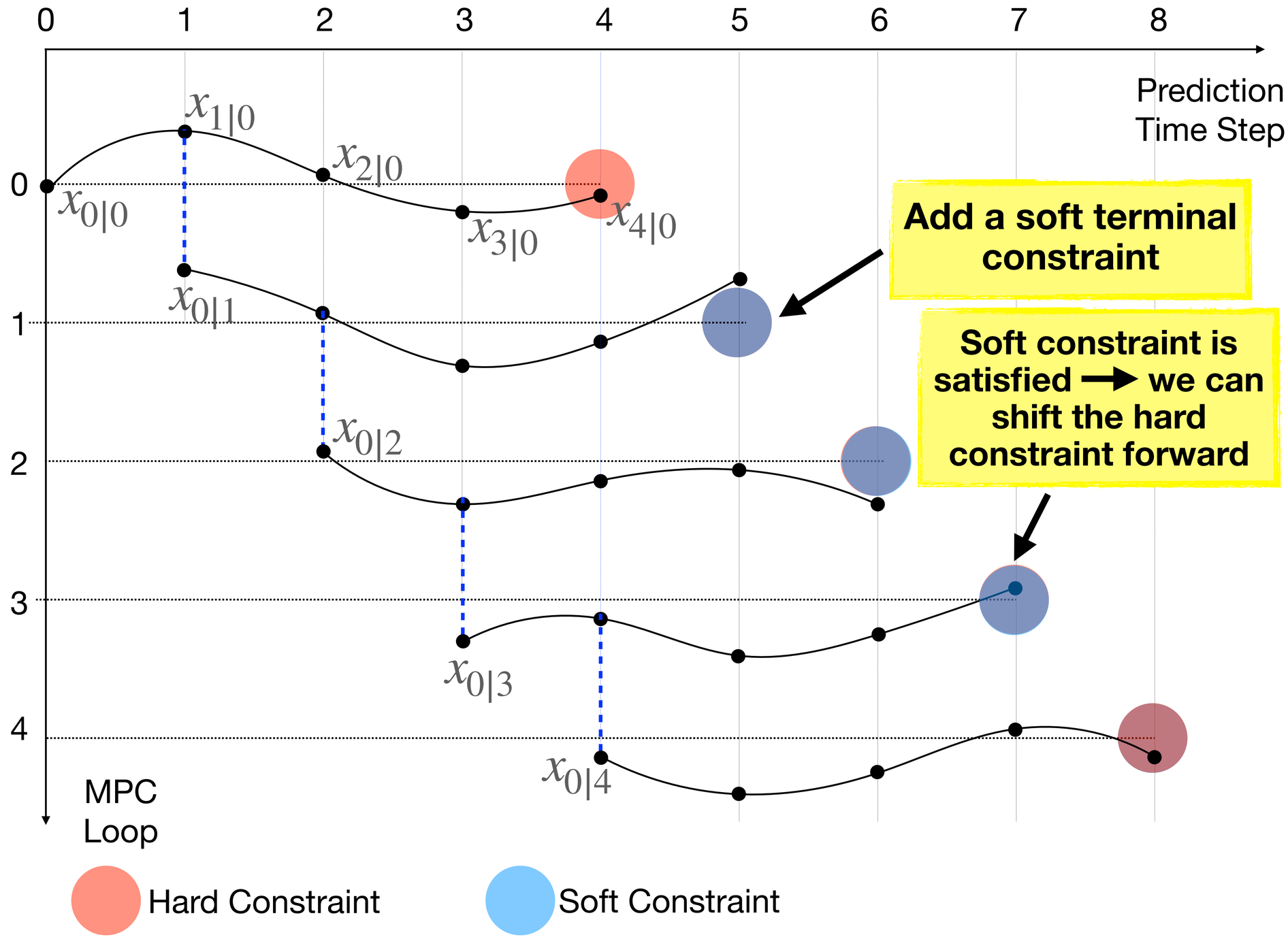
Come back to the set in exactly N steps moving back and forth in time ⟷ **Terminal Constraint MPC w/ Safe Abort**

Come back to the set within N steps ⟷ **Fixed Receding Constraint MPC**

Come back to the set in exactly K steps with K up to N ⟷ **Receding Constraint MPC**

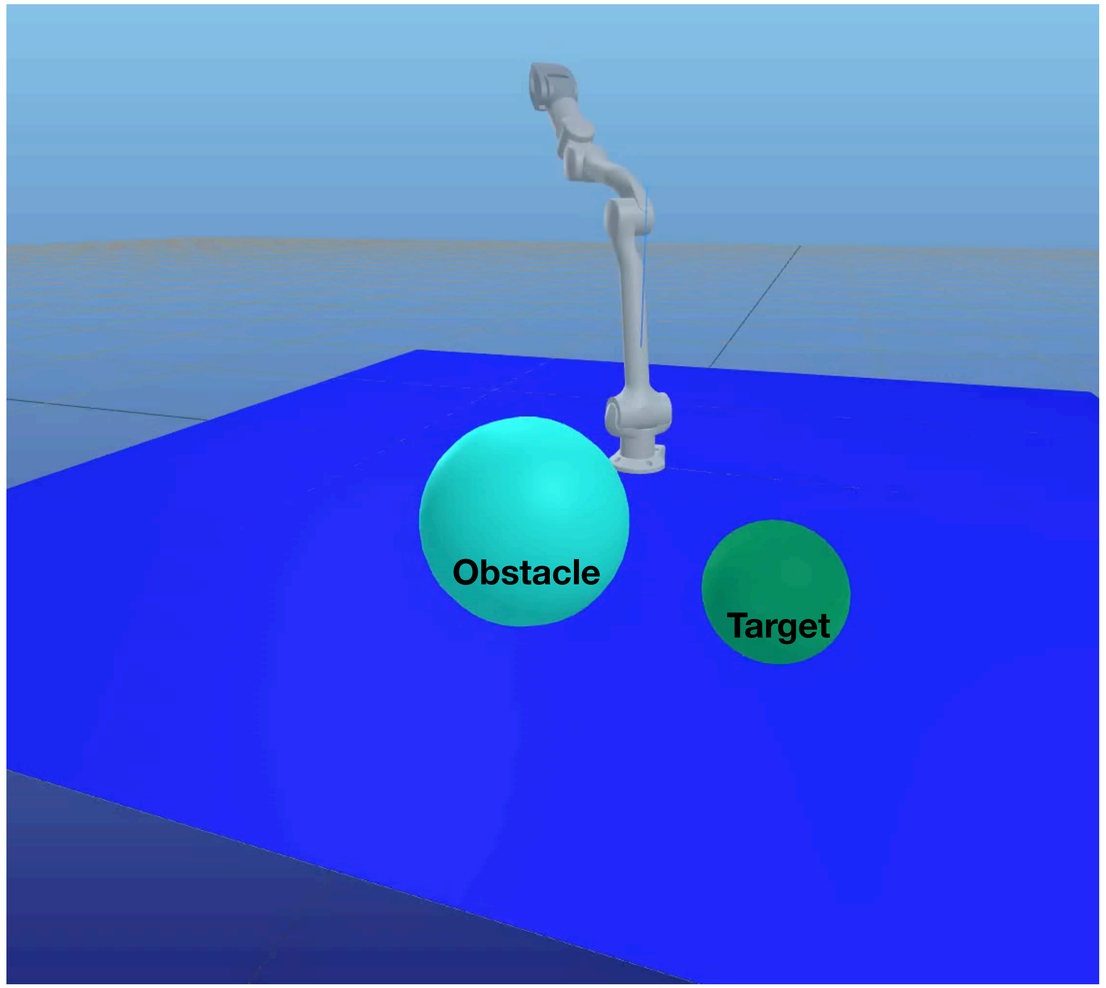
Remain inside in the future ⟷ **Classic Terminal Constraint MPC**


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

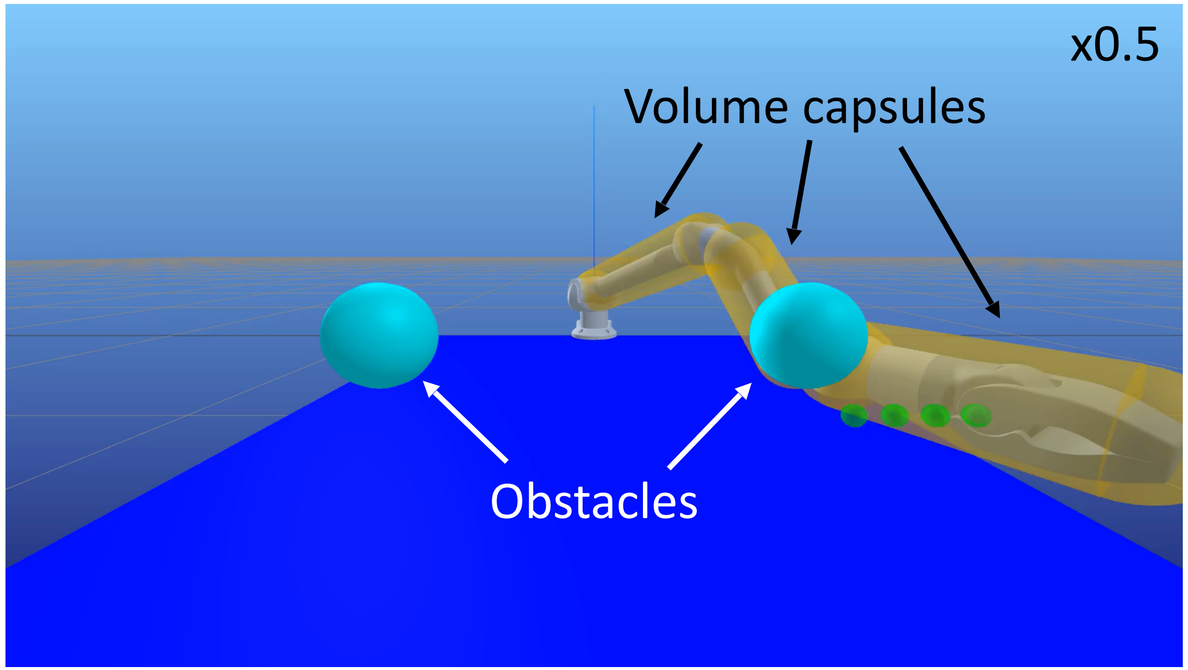

0
1
2
3
4
5
6
7
8
Prediction
Time Step
$x_{1|0}$
$x_{2|0}$
$x_{3|0}$
$x_{4|0}$
$x_{0|0}$
$x_{0|1}$
$x_{0|2}$
$x_{0|3}$
$x_{0|4}$
0
1
2
3
4
MPC
Loop
Add a soft terminal constraint
Soft constraint is satisfied → we can shift the hard constraint forward
Hard Constraint
Soft Constraint

Simulation Results

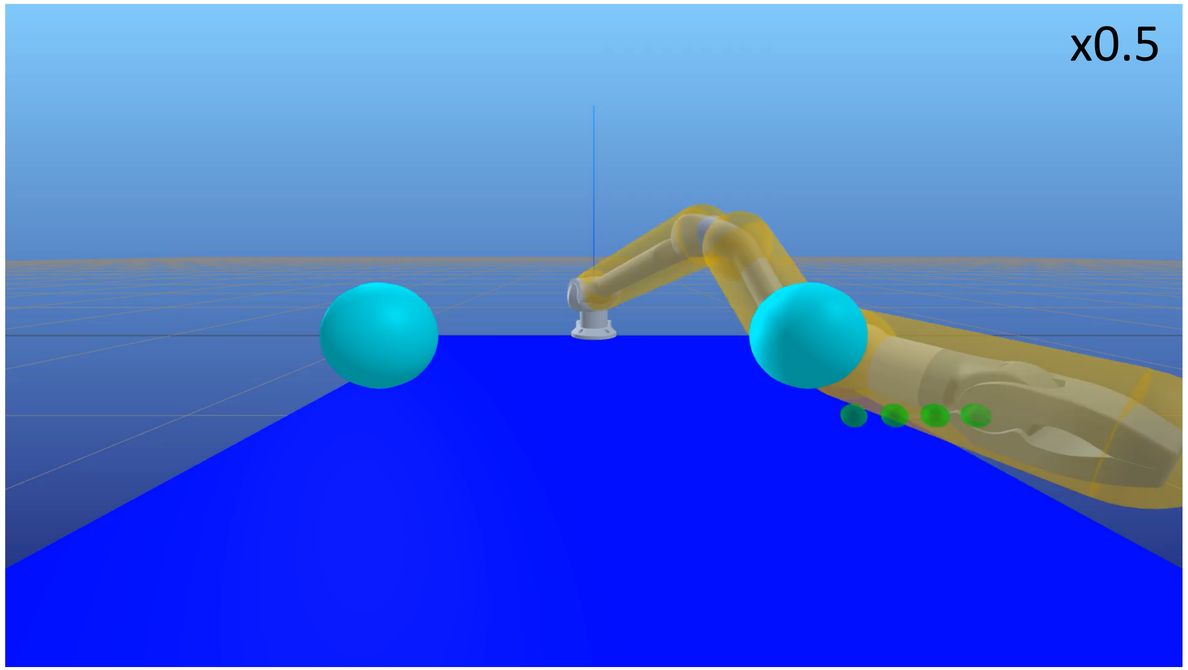
- Comparing several MPC formulations
- 4 DoF Z1 robot manipulator
- Acados software library
- Safe set $\hat{\mathscr{V}}$ represented with neural network
- 500 simulations from random initial configurations
- Max horizon N=45 to ensure computation time < dt (5 ms)
- https://github.com/idra-lab/safe-mpc



Trajectory Tracking



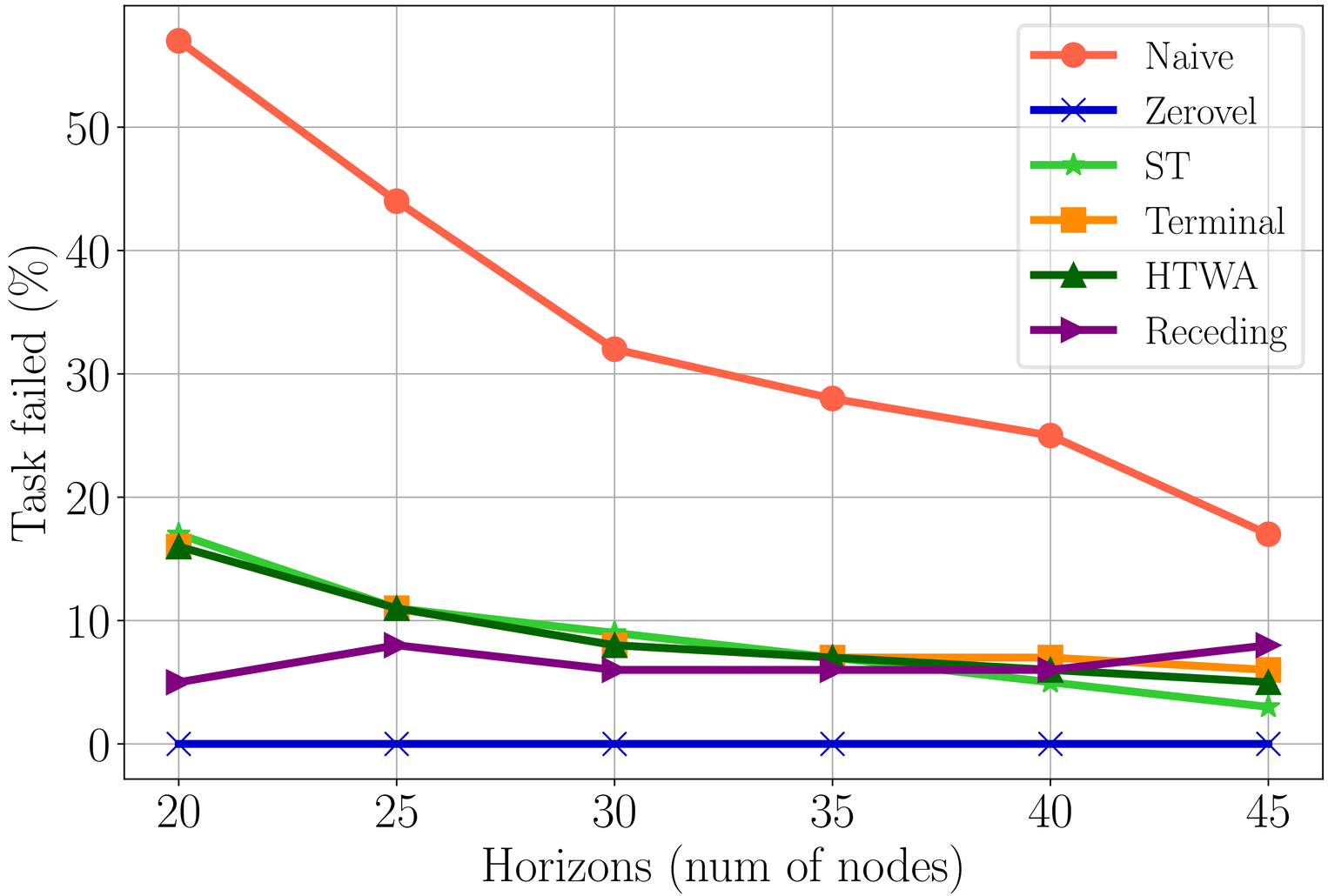
Naive MPC



Receding MPC

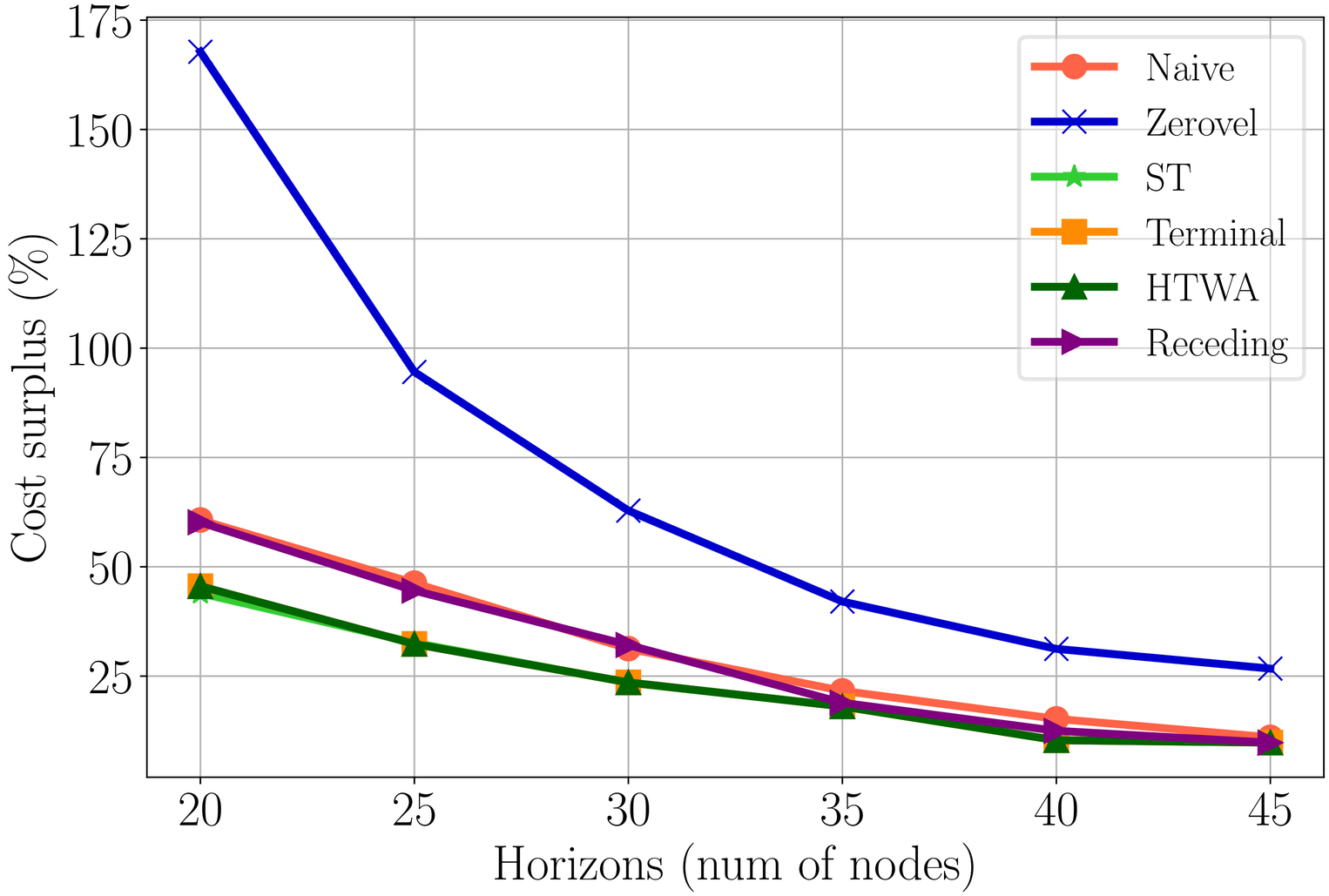
Simulation Results - Receding

- 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



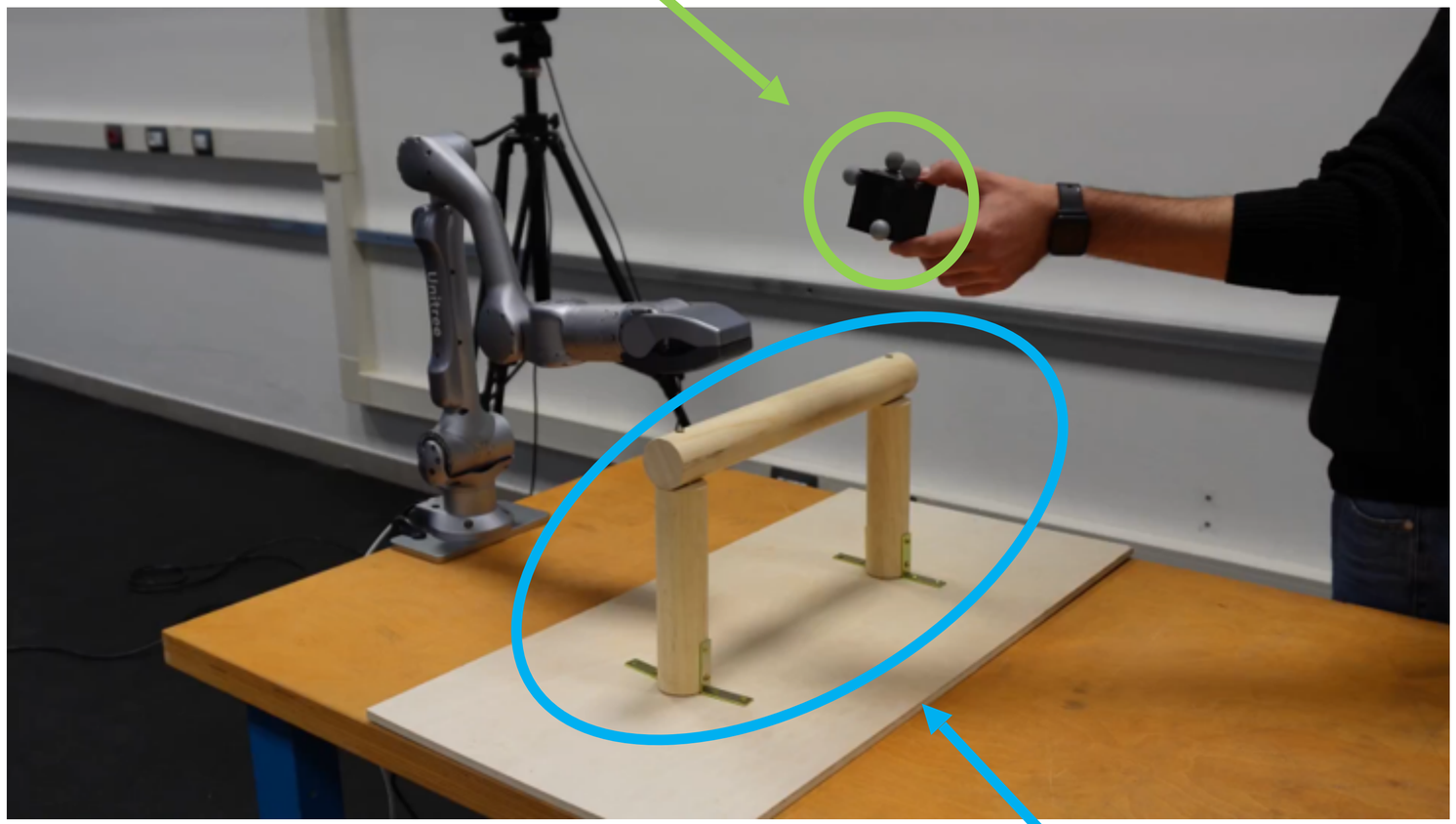
Simulation Results - Receding

- 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

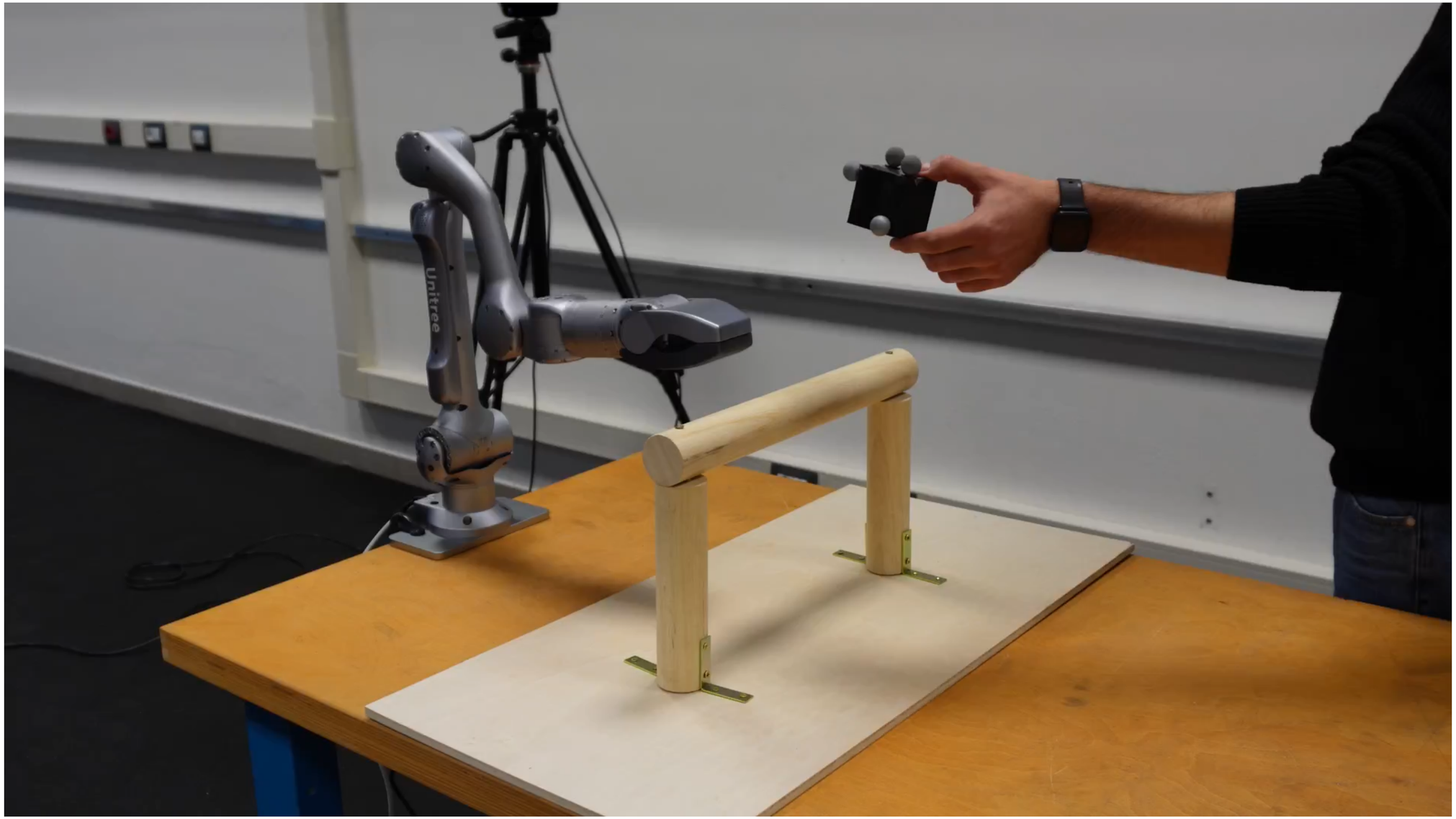


Experimental Setup

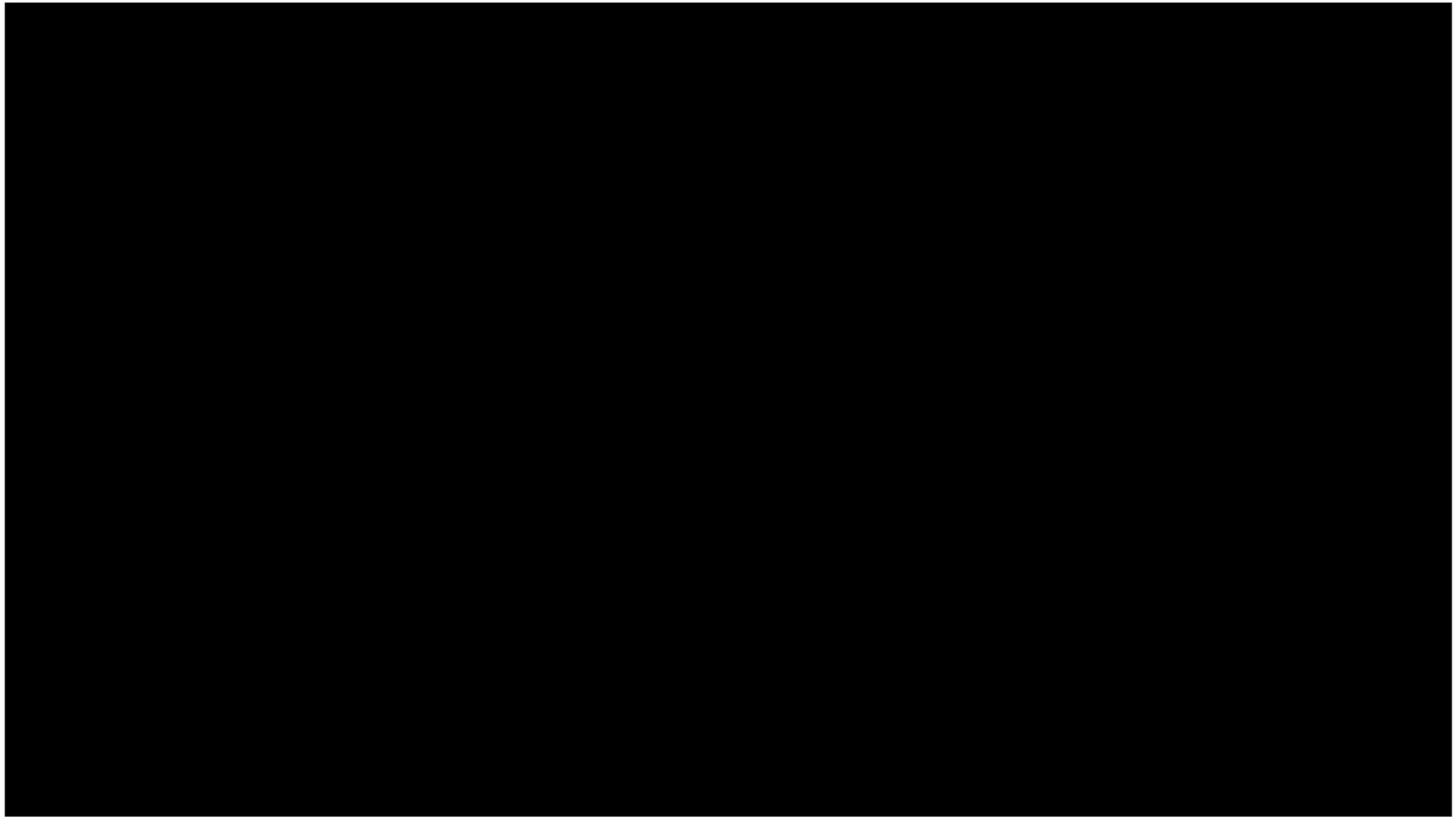
- Receding-Constraint MPC
- 6 degrees of freedom
- Z1 robot manipulator
- Tracking of moving target
- Target perceived with motion capture



Tracking Experiment - Side View



Tracking Experiment - Top View



Safe MPC - Conclusions

- Novel MPC formulations ensuring
  - Recursive feasibility under weaker conditions (multi-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

Safe and Globally Optimal Robot Control

Beyond control invariance and reinforcement learning









IDRA
INTERDEPARTMENTAL ROBOTICS LABS

Andrea Del Prete

UNIVERSITAS ATHESINA STUDIORUM
UNIVERSITY OF TRENTO