ICRA 2026 Keynote Tutorial

Learning Agile Vision-based Quadrotor Flight: from Simulation to Real-world Adaptation

Closing the Sim-to-Real Gap with Online Planning

Part 2 — MPC meets Reinforcement Learning for Agile Quadrotor Flight

Rudolf Reiter

Robotics and Perception Group, University of Zurich

with Davide Scaramuzza & Ismail Geles · June 2, 2026 · Vienna

Outline

1. **Introduction — why keep online planning?**
   MPC as the industry standard for safety-critical control

2. **How to combine MPC with RL?**
   Taxonomy, architectures, and learning paradigms from our recent survey

3. **ACMPC — a pioneering success story**
   Differentiable MPC embedded in actor-critic RL for 21 m/s drone racing

MPC: an industry standard for safety-critical control

- Solves **approximate MDP at current state s** at every time step
- Explicitly handles **constraints** (state, actuator, safety)
- Explicit **parameterization** of a model
- **Adaptable** to new conditions without retraining
- **Generalizes** to model not training data distribution
- **Industry-proven**: process plants, energy, automotive, robotics

$$
\begin{aligned}
\min_{\substack{x_0,\dots,x_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f(x_k, u_k), \quad k = 0, \dots, N-1, \\
& x_k \in \mathcal{X}_k, \quad k = 0, \dots, N, \\
& u_k \in \mathcal{U}_k, \quad k = 0, \dots, N-1.
\end{aligned}
$$

Key paradigm for this talk

Keep online planning as an integral part of the algorithm.
Combine the proven performance of MPC with the learning of RL.

Mayne, Rawlings, Diehl (2017) — Model Predictive Control: Theory, Computation, and Design

MPC: an industry standard for safety-critical control

- Solves **approximate MDP** at **current state s** at every time step
- Explicitly handles **constraints** (state, actuator, safety)
- Explicit **parameterization** of a model
- **Adaptable** to new conditions without retraining
- **Generalizes** to model not training data distribution
- **Industry-proven**: process plants, energy, automotive, robotics

$$
\begin{aligned}
\min_{\substack{x_0,\dots,x_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f(x_k, u_k), \quad k = 0, \dots, N-1, \\
& x_k \in \mathcal{X}_k, \quad k = 0, \dots, N, \\
& u_k \in \mathcal{U}_k, \quad k = 0, \dots, N-1.
\end{aligned}
$$

Key paradigm for this talk

Keep online planning as an integral part of the algorithm.
Combine the proven performance of MPC with the learning of RL.

Mayne, Rawlings, Diehl (2017) — Model Predictive Control: Theory, Computation, and Design

MPC: an industry standard for safety-critical control

- Solves **approximate MDP at current state s** at every time step
- Explicitly handles **constraints** (state, actuator, safety)
- Explicit **parameterization** of a model
- **Adaptable** to new conditions without retraining
- **Generalizes** to model not training data distribution
- **Industry-proven**: process plants, energy, automotive, robotics

$$
\begin{aligned}
\min_{\substack{x_0,\dots,x_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f(x_k, u_k), \quad k = 0, \dots, N-1, \\
& x_k \in \mathcal{X}_k, \quad k = 0, \dots, N, \\
& u_k \in \mathcal{U}_k, \quad k = 0, \dots, N-1.
\end{aligned}
$$

Key paradigm for this talk

Keep online planning as an integral part of the algorithm.
Combine the proven performance of MPC with the learning of RL.

Mayne, Rawlings, Diehl (2017) — Model Predictive Control: Theory, Computation, and Design

MPC: an industry standard for safety-critical control

- Solves **approximate MDP at current state s** at every time step
- Explicitly handles **constraints** (state, actuator, safety)
- Explicit **parameterization** of a model
- **Adaptable** to new conditions without retraining
- **Generalizes** to model not training data distribution
- **Industry-proven**: process plants, energy, automotive, robotics

$$
\begin{aligned}
\min_{\substack{x_0,\dots,x_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f(x_k, u_k) \quad k = 0, \dots, N-1, \\
& x_k \in \mathcal{X}_k, \quad k = 0, \dots, N, \\
& u_k \in \mathcal{U}_k, \quad k = 0, \dots, N-1.
\end{aligned}
$$

Key paradigm for this talk

Keep online planning as an integral part of the algorithm.
Combine the proven performance of MPC with the learning of RL.

Mayne, Rawlings, Diehl (2017) — Model Predictive Control: Theory, Computation, and Design

MPC vs. RL — practical properties

Property	MPC	RL
State space	model specific ✗	(quite) arbitrary ✓
Model requirement	differentiable / smooth ✗	offline simulation ✓
Uncertainty	guarantees w/ known unc. ✓	probabilistic guarantees ✓
Stability	strong theory ✓	less theory ✗
Constraint handling	inherent ✓	harder to achieve ✗
Online computation	high ✗	low ✓
Offline computation	low ✓	high ✗
Adaptability	inherent ✓	needs retraining ✗
Generalization	inherent ✓	poor when OOD ✗

Reiter et al., Synthesis of MPC and RL: Survey and Classification — Annual Reviews in Control, 2026

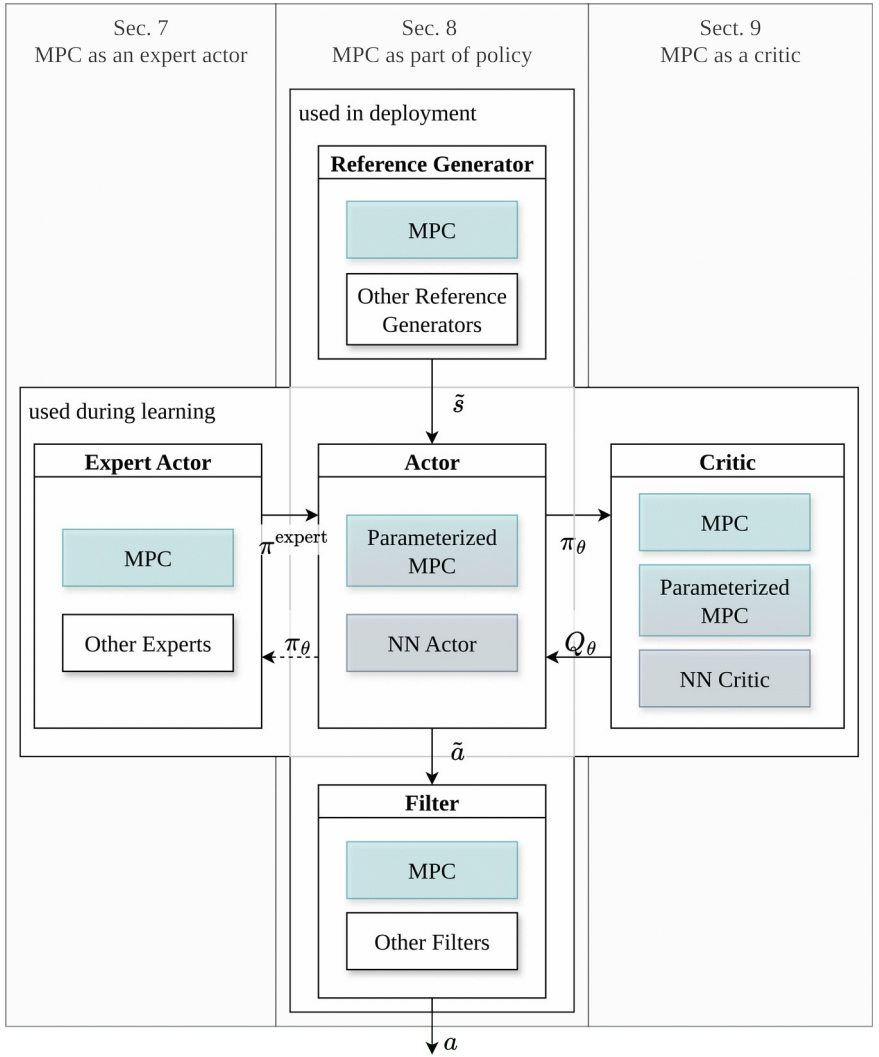
How to combine MPC and RL?

Their weaknesses are each other's strengths

- Both solve sequential decision-making problems (MDPs)
- Nearly orthogonal practical advantages
- Rapid growth in combination methods (200+ papers surveyed)
- We classify methods using the actor-critic RL framework
- Survey: "Synthesis of Model Predictive Control and Reinforcement Learning: Survey and Classification", Reiter et al., Annu. Rev. Control, 2026

Scan to read the survey

Reiter et al., Annu. Rev. Control, 2026



Reiter et al. (2026) — taxonomy of MPC roles in actor-critic RL.

Three main roles of MPC in the RL framework

1. MPC as expert actor

MPC generates demonstrations; an RL policy imitates the behaviour.

- ✓ Fast inference at deployment
- ✓ Leverages **domain knowledge**
- ✗ **Bounded** by MPC performance

Examples:

BC, DAgger, GPS — Karg & Lucia 2020, Schulz & Hoffmann 2024

2. MPC within the policy

MPC runs at deployment time; may or may not be used during training

- ✓ **Constraints** + **adaptability**
- ✓ Leverages **domain knowledge**
- ✓ **Sample efficient** training
- ✓ **Interpretability**
- ✗ **Slower** training

- ★ **This is where ACMPC lives**

Examples:

ACMPC, AC4MPC, TD-MPC2, safety filters, reference generators

3. MPC as the critic

MPC provides value estimates Q^MPC(s,a) used by RL.

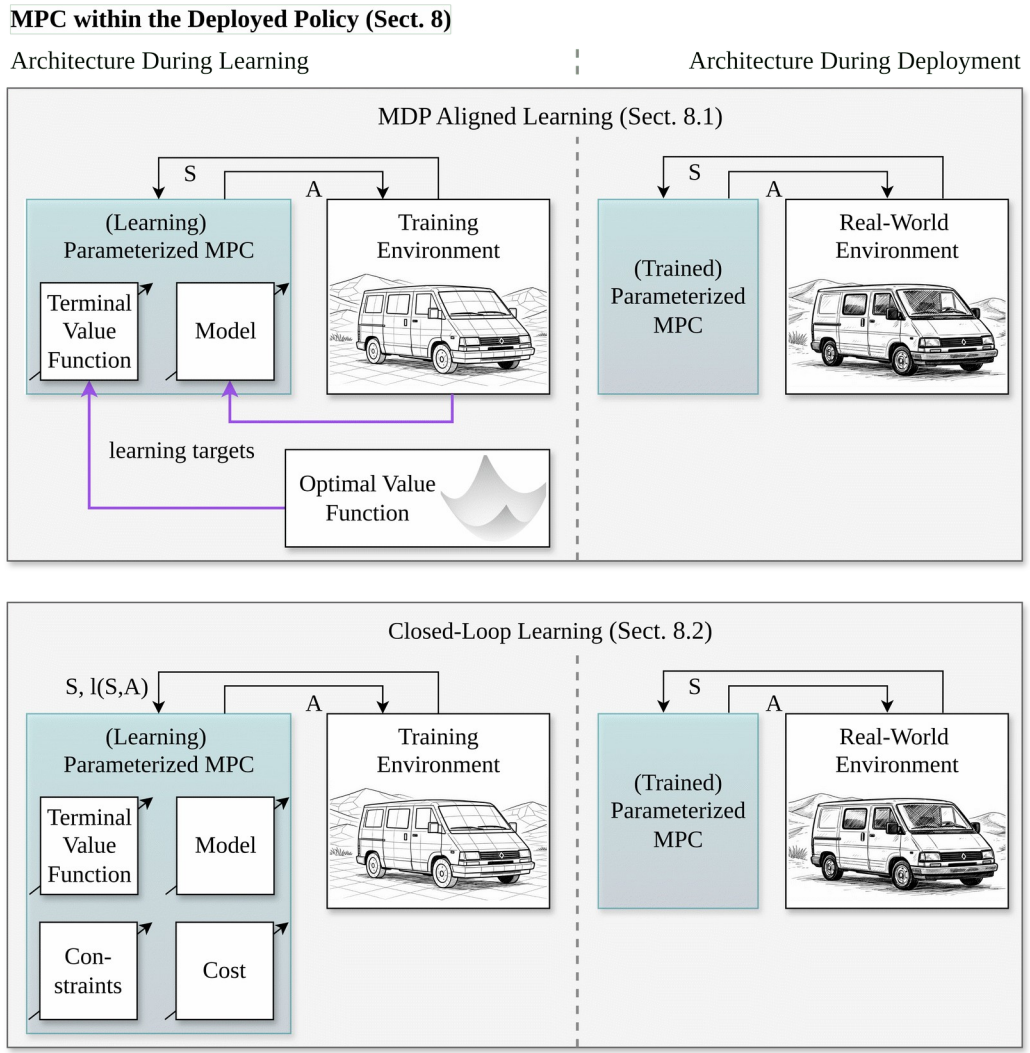
- ✓ Model-based value structure
- ✓ Leverages **domain knowledge**
- ✓ **Sample efficient** training
- ✓ Can guide policy improvement
- ✗ MPC needed during training

Examples:

Bhardwaj et al. 2020, Anand et al. 2023, Ghezzi et al. 2023

Reiter et al. (2026) — based on the actor-critic decomposition common to most modern RL algorithms.

MPC within the deployed policy - Paradigms



MDP-Aligned Learning

- Learn MPC components to match the true MDP
- ✓ Interpretable, generalizes well
- ✗ Sub-optimal closed-loop performance

$$\text{Model:} \quad f_\theta^{\text{MPC}} \approx P, \qquad \text{Terminal cost:} \quad \bar{V}_\theta^{\text{MPC}} \approx V^\star$$

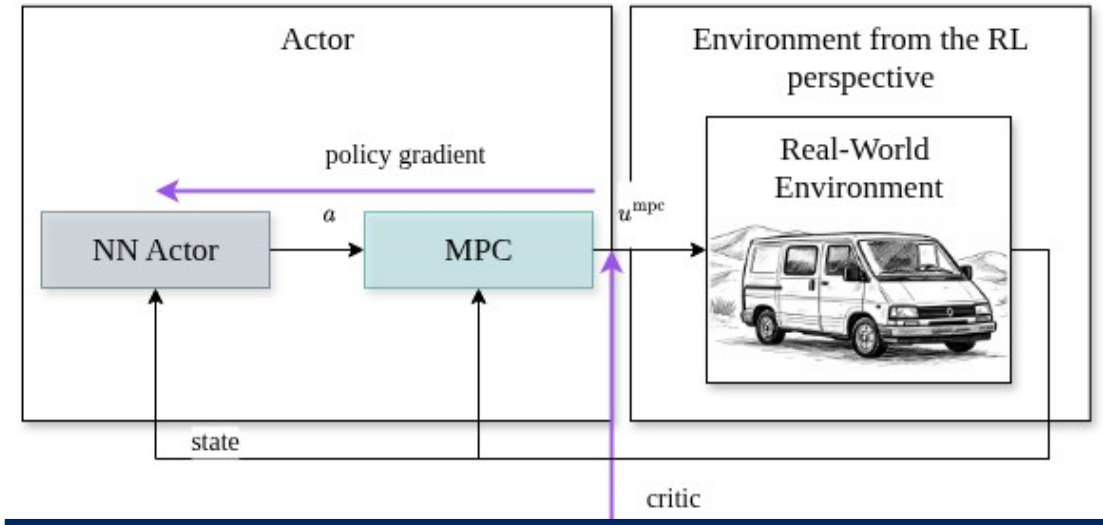
Closed-Loop Optimal Learning

- Tune MPC end-to-end for the actual closed-loop objective
- ✓ Directly optimizes task performance
- ✗ Less interpretable, weaker generalization
- ★ **This is where ACMPC lives**

$$\mu_\theta^{\text{MPC}} \approx \mu^\star, \qquad \min_\phi \; J\left(\mu_\phi^{\text{MPC}}\right)$$

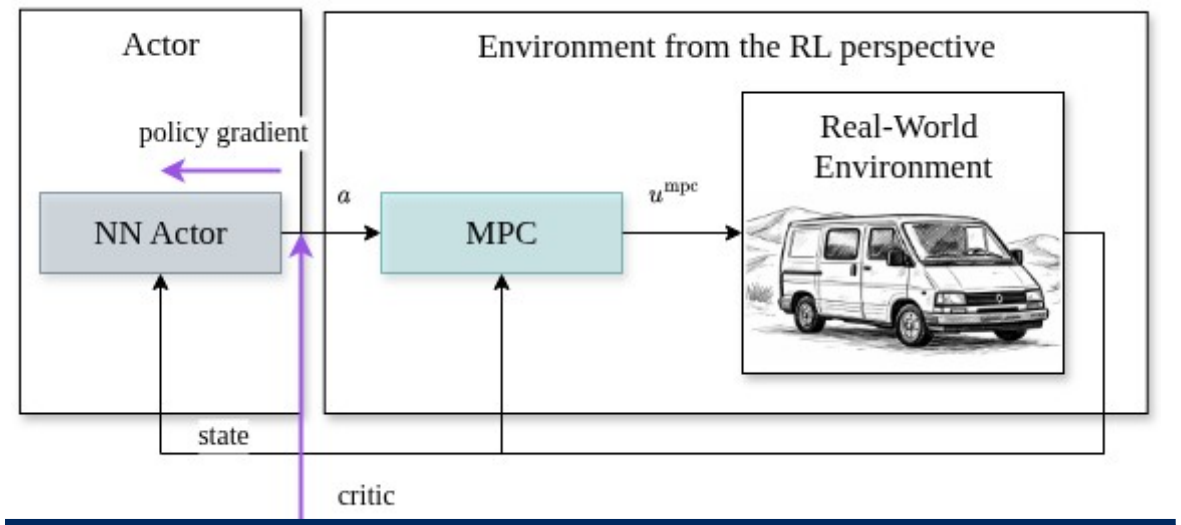
Reiter et al. (2026) — Sections 8.1 & 8.2 of the survey, Fig. 5 (top).

Closed-loop learning – RL embedding



A. MPC as part of actor

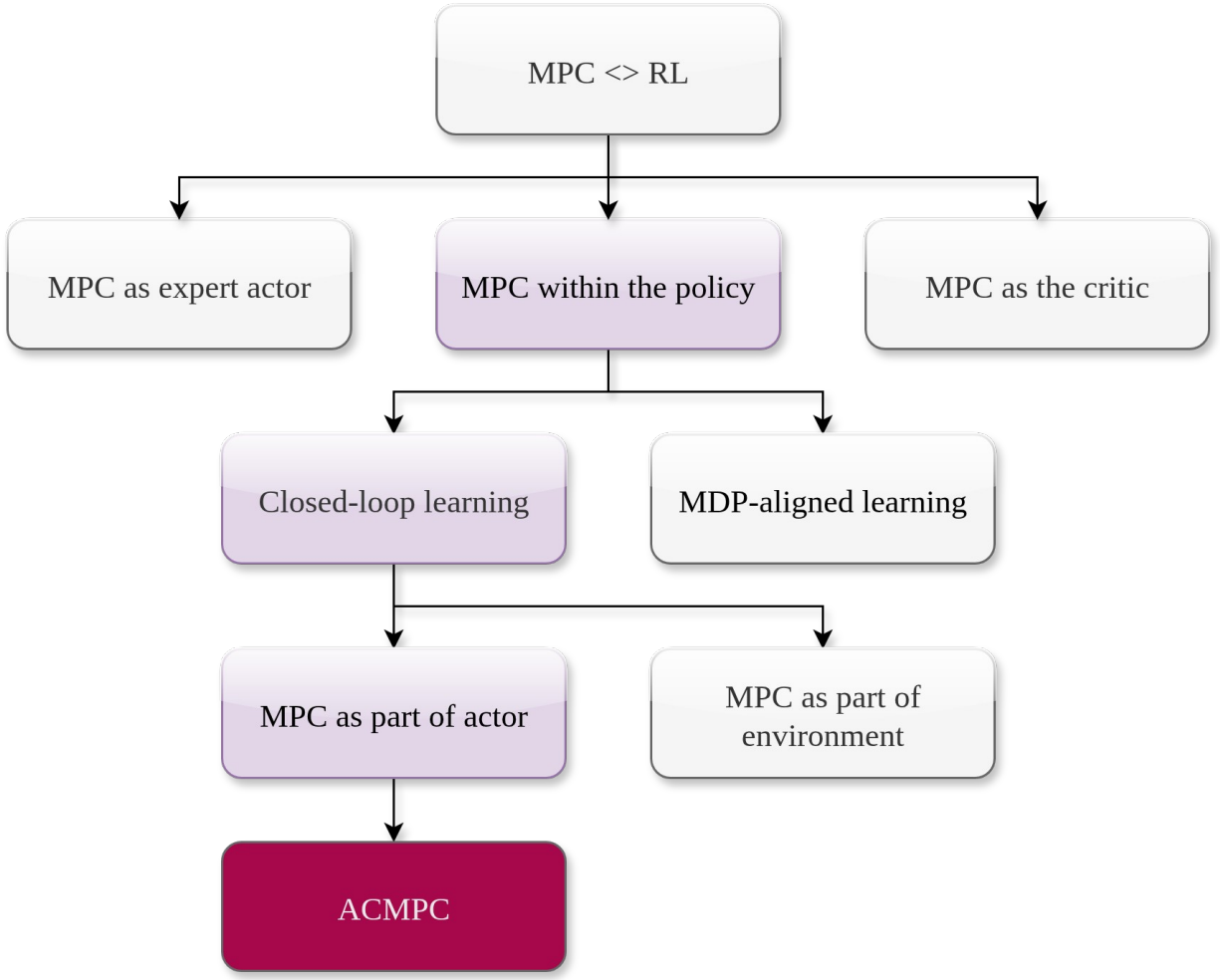
- MPC is an optimization layer inside the actor
- Backprop gradients through the solver via the KKT system
- ✓ End-to-end gradient
- ✗ Requires solver to expose sensitivities
- ★ **This is where ACMPC lives**



B. MPC as part of the environment

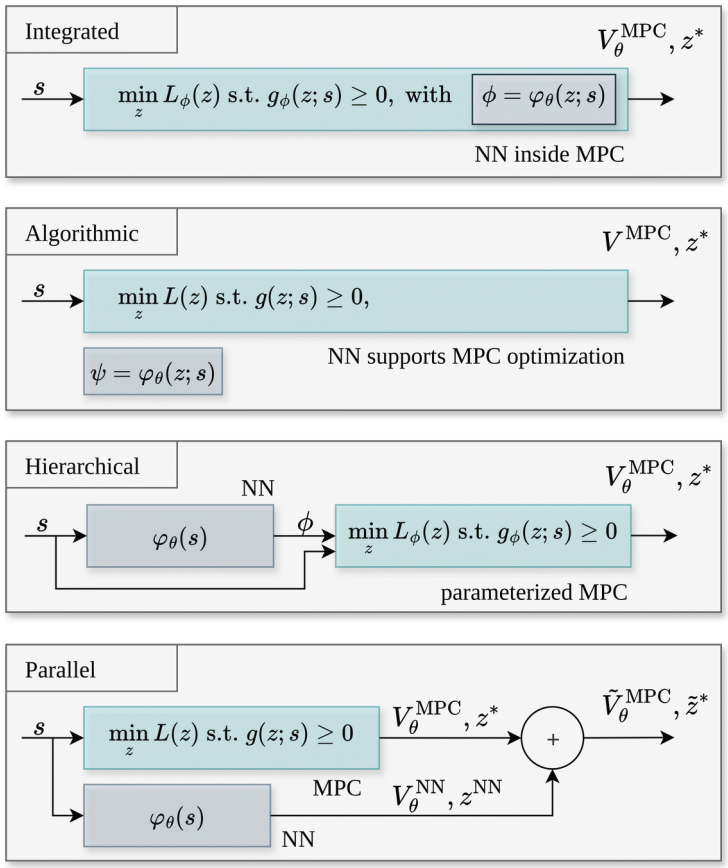
- MPC parameters are treated as the RL action
- MPC + plant viewed as a single black-box environment
- No solver gradients: sensitivities come from sampling
- Action-space dimension grow with parameters
- ✓ Any standard RL algorithm applies
- ✗ Loses the first-order gradient signal through the optimizer

Where ACMPC finally lives



Reiter et al. (2026) — based on the actor-critic decomposition common to most modern RL algorithms.

Inference architectures: NN / MPC



Integrated

NN inside the optimization → most flexible, hardest to solve

Salzmann et al. 2023; Hansen et al. 2024 (TD-MPC2 model)

Algorithmic

NN guides the solver (warm-starts, samples)

TD-MPC2 — Hansen et al. 2024; Grandesso et al. 2023

Hierarchical

NN outputs MPC parameters; MPC stays standard,

★ **This is where ACMPC lives** — *Romero et al. 2024/2025; Reiter et al. 2023*

Parallel

NN correction alongside MPC → residual learning

Bhardwaj et al. 2020

Reiter et al. (2026), Annu. Rev. Control — Fig. 3.

Practical design choices

Newton-type vs. sampling-based MPC?

Newton/SQP (acados) — fast, gradients available · MPPI — derivative-free, GPU-parallel

Where to add exploration noise?

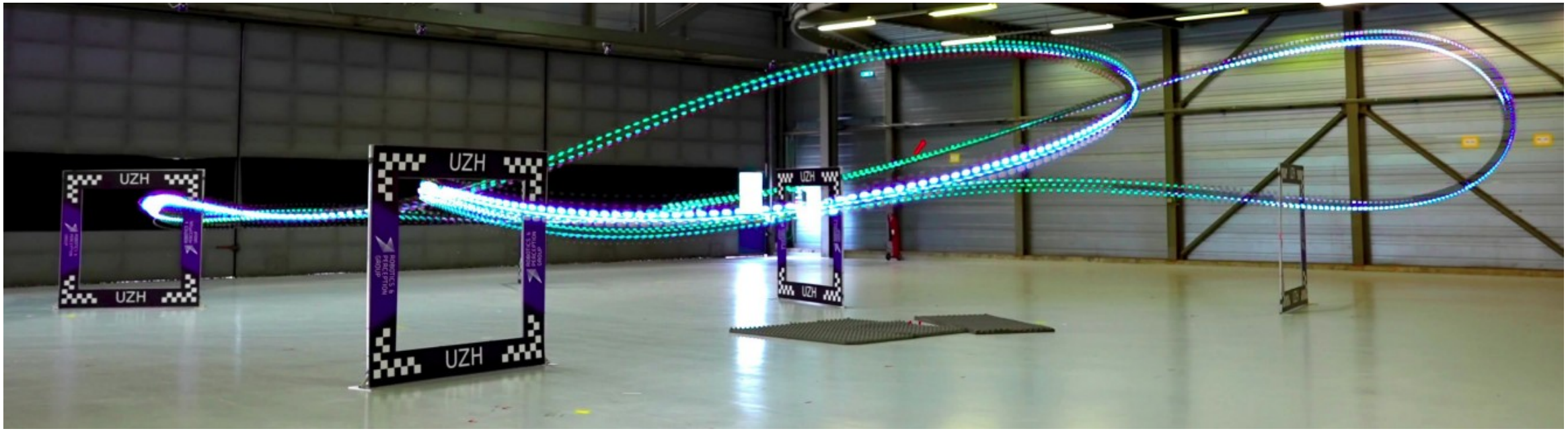
On MPC parameters (structured, but slower) · After MPC on actions (simple, weaker safety)

Which RL algorithm?

On-policy PPO (stable, sample-hungry — ACMPC default) · Off-policy SAC (sample-efficient)

Recurring design knobs that appear across most MPC+RL methods.

Actor-Critic Model Predictive Control



From theory to real-world agile drone flight

Angel Romero: roangel.github.io

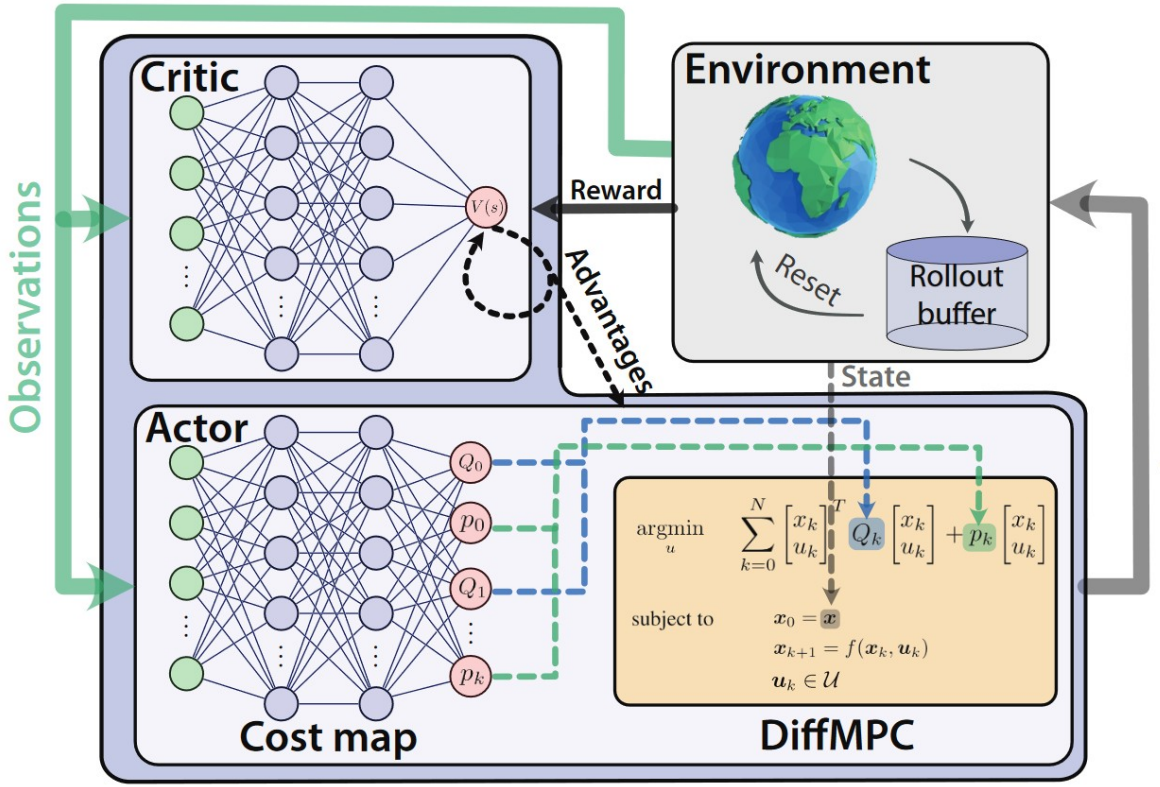
ACMPC: architecture

Core idea

Make the actor's last layer a differentiable MPC.

- **Neural cost map**: observations → (Q_k, R_k, p_k)
- MPC solves an **NLP** using quadrotor dynamics
- **PPO** trains end-to-end
- Policy gradients flow through the MPC solver
- Hierarchical architecture

$$o_t \xrightarrow{\text{Neural cost map}} (Q_k, R_k, p_k)_{k=0}^{N}$$



Milestone in MPC and RL for robotics

First differentiable MPC + RL variant validated in real-world high-performance flight at 21 m/s.

ACMPC — Romero et al.

ICRA 2024 + IEEE T-RO 2025

Romero, Aljalbout, Song, Scaramuzza — IEEE Transactions on Robotics, 2025. Source code: github.com/uzh-rpg/acmpc_public

ACMPC: implementation details

1. Standard tracking MPC

$$J_{\mathrm{MPC}}(x) = \sum_{k=0}^{N-1} \|\Delta x_k\|_Q^2 + \|\Delta u_k\|_R^2 + \|\Delta x_N\|_P^2$$

2. Expand → general quadratic program (QP)

$$J_{QP}(x) = \sum_{k=0}^{N} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top Q_k \begin{bmatrix} x_k \\ u_k \end{bmatrix} + p_k^\top \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

3. Restrict Q to diagonal (empirical best performance)

$$Q = \mathrm{diag}(q), \quad q = \begin{bmatrix} q_p \\ q_q \\ q_v \\ q_\omega \\ r_f \end{bmatrix} \in \mathbb{R}^{n_x+n_u}_{\geq 0}$$

Implementation choices

- QP solver: differentiable iLQR (https://locuslab.github.io/mpc.pytorch/)
- Cost parameters per stage: 34 (Q, R, p)
- Horizon N = 2 → 100
- Cost-matrix structures: **diagonal**, Cholesky, full
- Exploration noise added on actions, after MPC
- RL: PPO (on-policy actor-critic)

Model-Predictive Value Expansion

- Solving the MPC **generates a predicted trajectory** over its horizon
- **Only the first action** is applied to the drone, the remaining predictions are normally discarded.
- **Reuse** them as a multi-step **value target for the critic**.

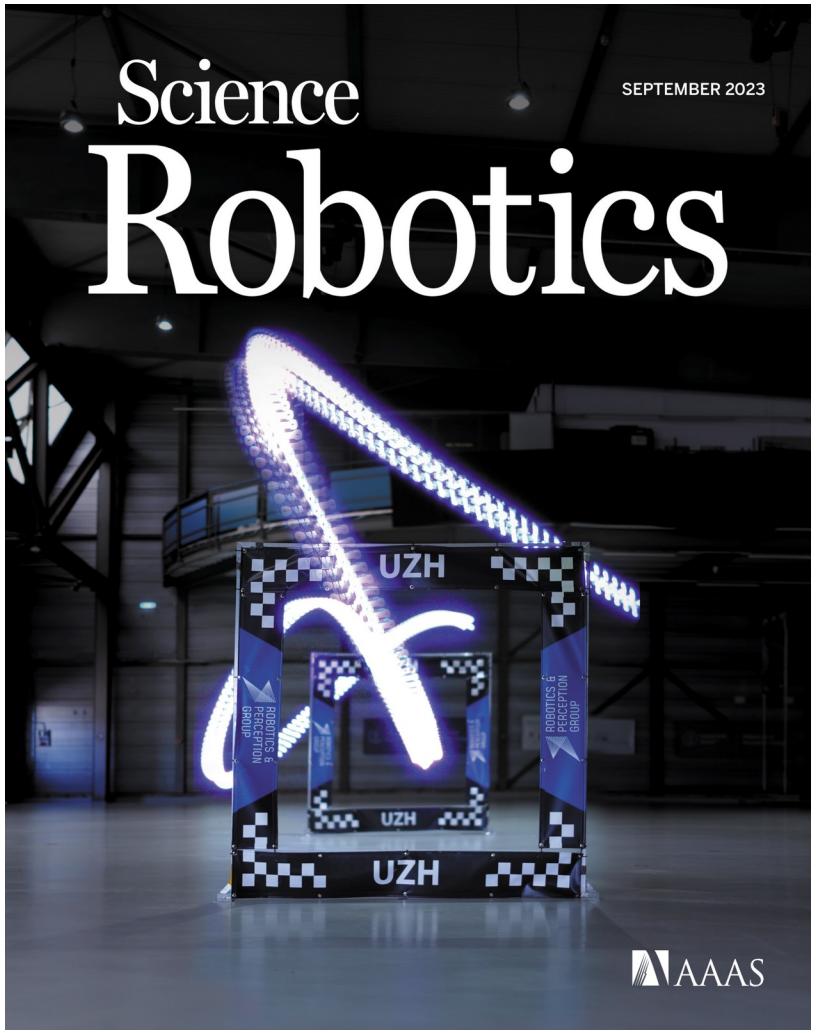
H-step value target:

$$\hat{V}_H(s_t) = \sum_{k=0}^{H-1} \gamma^k \, r(\hat{x}_k, \hat{u}_k) + \gamma^H \, V_\phi(\hat{x}_H)$$

Results: Compareable to best PPO baseline

Lap times — Nominal / Realistic / Real World

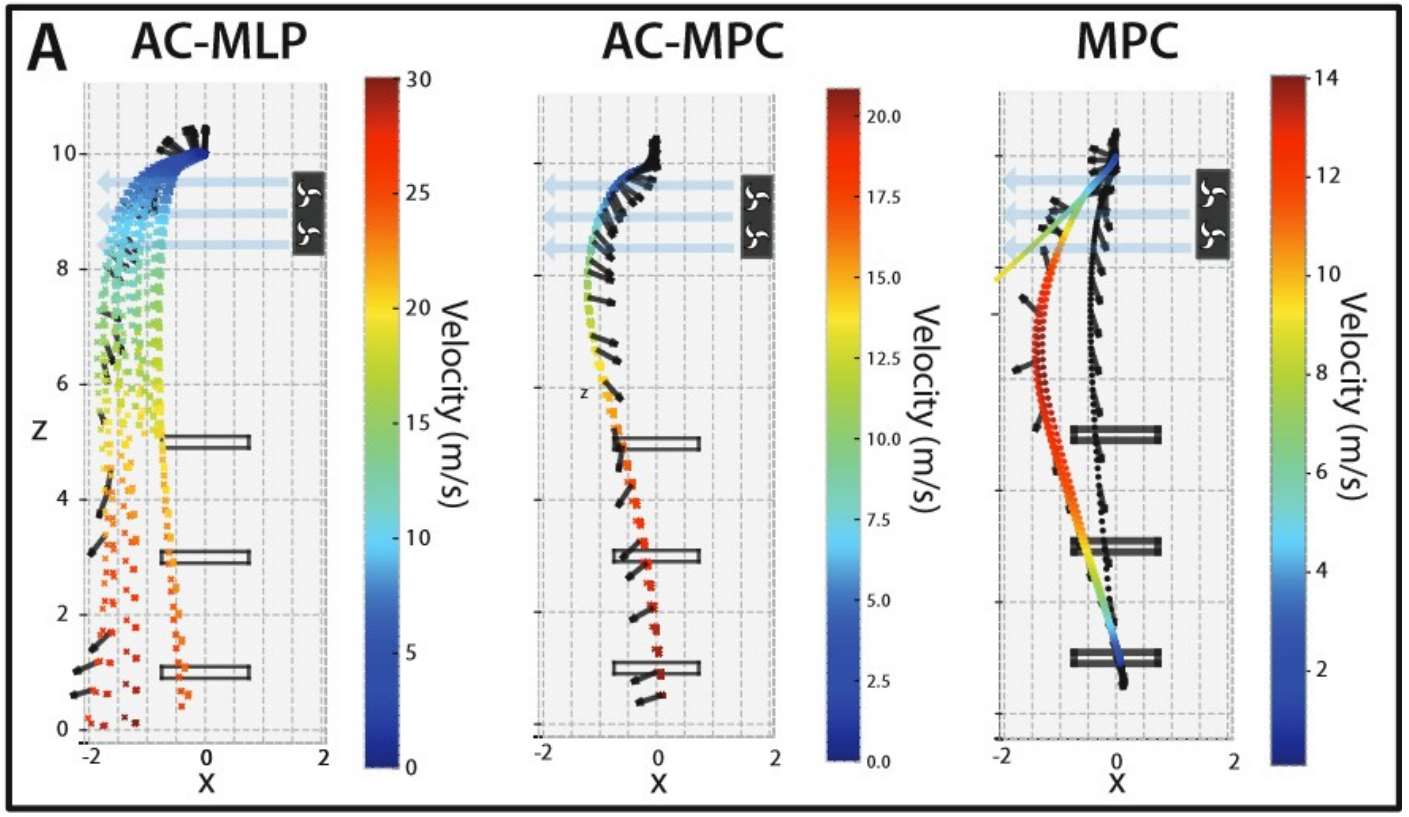
	AC-MLP		AC-MPC (N=2)	
Drone Model	Lap Time [s]	Success Rate [%]	Lap Time [s]	Success Rate [%]
Nominal	5.09 ± 0.008	100.0	5.13 ± 0.008	100.0
Realistic	5.179 ± 0.01	100.0	5.24 ± 0.01	100.0
Real World	5.39 ± 0.08	85.7	5.4 ± 0.082	87.5



- Peak speed: 21 m/s (76 km/h)
- On par with the best human racing pilots
- Sim-to-real transfer without fine-tuning
- Same hardware as the Science Robotics Paper (Y. Song, et al. 2023)

Romero et al., IEEE T-RO 2025 · Same drone-racing benchmark as Kaufmann et al., Nature 2023.
Y. Song, et al., Science Robotics, p. adg1462, 2023.

Results: Robustness Against External Disturbance



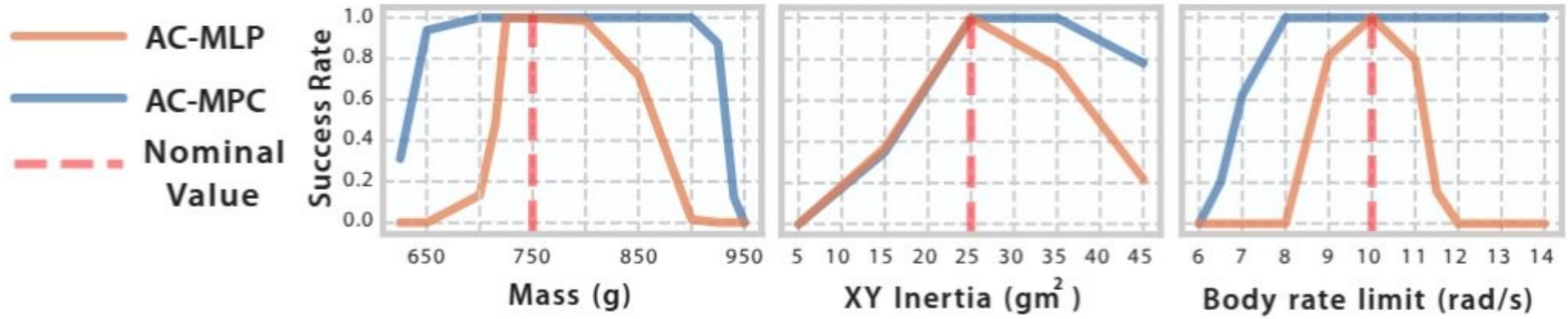
AC-MLP (PPO) vs ACMPC vs. MPC

- Vertical flight with wind distrubance
- Target: pass the gates
- ACMLP: gets out of training distribution
- ACMPC: robust to disturbance
- MPC: no domain randomization

Results

- ACMPC outperforms baselines
- N=2: ≈ 11 h training (MLP: 21 min)
- N=2: ≈ 13.5 ms inference (MLP: 0.5 ms)
- Diagonal Q best > Cholesky > full
- Cost: 34 parameters / stage

Results: Robustness Against Model Adaption



ACMPC keeps flying outside its training distribution, without re-training.

- ✓ Success @ ±30 % mass changes
- ✓ Success @ ±50 % body-rate limits
- ✓ Success @ shifts in XY inertia
- ✓ Beats adaptive L1-MPC in some regimes

- Why? The MPC keeps replanning every step using the physics model
- AC-MLP collapses outside its training distribution.
- This is an important practical argument for keeping a model in the online planning loop.

Final assesment of ACMPC

What ACMPC achieves ✓

- Better **out-of-distribution** behavior than pure RL
- Better **robustness** to model changes
- Improved **sample efficiency** (MPC as inductive bias)
- **Superhuman** performance: 21 m/s real-world drone racing
- **Sim-to-real transfer** without fine-tuning

Limitations ✗

- **No state constraints** — only input constraints via QP
- Exploration noise added after MPC → weak safety argument
- **Long training time**: N=2 takes ≈ 11 h (MLP: 21 min)
- **Higher inference time**: 13.5 ms (MLP: 0.5 ms)
- Requires a differentiable MPC solver

Many of these limitations are being addressed → next slide

- State constraints → leap-c (acados + PyTorch) with full NLP
- Training time → acados is up to 1200× faster than mpc.pytorch [Frey et al., 2025]

Tackling the computational bottleneck

Timing comparison

Table 1: Timings in [ms] for solving $n_{\text{batch}} = 128$ bounded LQR problems with $N = 20$, $n_x = 8$, $n_u = 4$, $n_\theta = 248$. In parentheses are multiples of the `acados` runtime.

	Nominal solution			Solution + adjoint sens.		
$u_{\max}$	`acados`	`mpc.pytorch`	`cvxpygen`	`acados`	`mpc.pytorch`	`cvxpygen`
10^4	8.5	78 (×9.2)	262 (×31)	34.5	125 (×3.6)	658 (×19)
1.0	17.6	21024 (×1200)	6402 (×360)	42.0	21899 (×520)	6845 (×160)

acados is up to 1200× faster than mpc.pytorch — and works for nonlinear MPC.

leap-c — Learning for Predictive Control

- Open-source: github.com/leap-c/leap-c
- acados ↔ PyTorch as a differentiable layer (forward + adjoint)
- Full nonlinear MPC — including state constraints
- Multithreaded batched solves for RL training
- Currently SAC supported → PPO in progress

github.com/leap-c

Frey, et al. — Differentiable Nonlinear MPC, arXiv:2505.01353, 2025.

What’s next?

Sim2Real: Model for online planning

Part 2 — Rudolf

- MPC re-optimizes every step
- Best OOD robustness
- Highest compute cost

Examples

- *ACMPC, Romero, T-RO 2025*
- *MPCC++, Romero, T-RO 2022*
- *AC4MPC, Reiter, TCST 2026*

Sim2Real: Model for offline planning

Part 3 — Ismail

- Differentiable simulation
- Learned world models (Dreamer)
- Semi-online residual models

Examples

- *Dreamer V3, Hafner, Nature 2025*
- *DiffSim quadrotor, Heeg, ICRA 2024*
- *BPTT for control, Wiedemann, ICRA 2023*

Sim2Real: No model (simulator only)

Part 3 — Ismail

- End-to-end RL from pixels
- PPO + domain randomization
- Fastest inference

Examples

- *Champ-level RL, Kaufmann, Nature 2023*
- *RL from pixels, Geles, IROS 2024*
- *PPO + DR, Song, RAL 2023*

Questions?

Rudolf Reiter · Robotics and Perception Group · University of Zurich

Survey: Reiter et al., Annu. Rev. Control 2026 · *ACMPC: Romero et al., IEEE T-RO 2025*

Survey

ACMPC