Synthesis of Model Predictive Control and Reinforcement Learning

– Overview over Synthesis of MPC and RL –

Dirk Reinhardt, Jasper Hoffmann

Fall School on Model Predictive Control and Reinforcement Learning
Freiburg, 6-10 October 2025

universität freiburg

NTNU | Norwegian University of Science and Technology

Dirk's Spotlight

- 🇳🇴 Postdoc at Cybernetics Department, NTNU, Norway
- 💻 Working on combining MPC and RL for real-world systems
- ⚡ Interested in applications in energy systems
- 🎓 PhD in Cybernetics, worked on Optimal Control of Unmanned Aerial Vehicles (acados inside)

Two Communities

🎮 Reinforcement Learning

- Machine-Learning Community: *Use learning to mitigate complexities*
- **Philosophy:** Trial-and-error learning to build policies directly from experience (implicit)
- Offline Solution (explicit function)
- Simulation first
- Roots in Computer Science

🏭 Model Predictive Control

- Control Community: *Use feedback to mitigate uncertainties*
- **Philosophy:** Model-based optimization over engineering models for planning (explicit)
- Online Solution (implicit function)
- Theory first
- Roots in Process Control

But both want to solve sequential decision-making problems!

[Video: Miki et al. 2022] [Video: Romero et al. 2024]

Synthesis of MPC and RL

Motivation:

- MPC and RL are complementary approaches for solving MDPs
- Nearly orthogonal advantages: weaknesses of one are strengths of the other
- Growing research interest in combination methods

Goals of this presentation:

- Illuminate differences, similarities, and fundamentals enabling combinations
- Categorize existing work based on actor-critic RL framework

Synthesis of Model Predictive Control and Reinforcement Learning: Survey and Classification*

Rudolf Reiter[a,**], Jasper Hoffmann[b,**], Dirk Reinhardt[d], Florian Messerer[a], Katrin Baumgärtner[a], Shambhuraj Sawant[d], Joschka Bödecker[a], Moritz Diehl[b,c], Sebastien Gros[d]

[a]Department of Microsystems Engineering, University of Freiburg, Freiburg im Breisgau, 79110, Baden-Württemberg, Germany
[b]Department of Computer Science, University of Freiburg, Freiburg im Breisgau, 79110, Baden-Württemberg, Germany
[c]Department of Mathematics, University of Freiburg, Freiburg im Breisgau, 79104, Baden-Württemberg, Germany
[d]Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, 7034, Norway

Outline

Outline

Comparison of Practical Properties

Property	MPC	RL
state-space	model specific ✗	(quite) arbitrary ✓
model requirements	differentiability/ online simulation ✗	offline simulation ✓
uncertainty	guarantees with known uncertainty ✓	probabilistic guarantees ✓
stability	strong theory ✓	minor theory ✗
constraint handling	inherent ✓	hard to achieve ✗
online computation time	high ✓	low ✗
offline computation time	low ✗	high ✓
adaptability	inherent ✓	needs retraining ✗
generalization	inherent ✓	poor when out-of-distribution ✗

Comparison of practical properties between MPC and RL.
The evaluation is simplified and conceptual. Exemptions may exist.

Examples: MPC vs RL - Changes in Dynamics

MPC Problem

$$
\begin{aligned}
\min_{z} \quad & \sum_{k=0}^{N-1} l^{\mathrm{MPC}}(x_k, u_k) + \bar{V}^{\mathrm{MPC}}(x_N) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f^{\mathrm{MPC}}(x_k, u_k), \\
& 0 \leq h^{\mathrm{MPC}}(x_k, u_k), \\
& 0 \leq h_N^{\mathrm{MPC}}(x_N).
\end{aligned}
$$

Change in dynamics?

Modify $f^{\mathrm{MPC}} \rightarrow$ new optimal policy

RL Policy (DDPG)

$$
\begin{aligned}
& w \overset{Q}{\leftarrow} w + \frac{\alpha_w}{B} \sum_{i=1}^{B} \delta_i \nabla_w Q_w(S_i, A_i), \\
& \theta \overset{\mu}{\leftarrow} \theta + \frac{\alpha_\theta}{B} \sum_{i=1}^{B} \nabla_\theta \mu_\theta(S_i) \; \nabla_a Q_w(S_i, a)\big|_{a=\mu_\theta(S_i)}, \\
& \delta_i \coloneqq l(S_i, A_i) + \gamma Q_{\bar{w}}(S_i^+, A_i^+) - Q_w(S_i, A_i), \\
& \text{with } (S_i, A_i, S_i^+) \sim \mathcal{D}^{\mathrm{buffer}}, \; A_i^+ = \mu_\theta(S_i^+),
\end{aligned}
$$

Change in dynamics?

Collect new data $\mathcal{D}^{\mathrm{buffer}} \rightarrow$ retrain policy

Examples: MPC vs RL - Constraint Handling

MPC Problem

$$\begin{aligned}\min_{z} \quad & \sum_{k=0}^{N-1} l^{\mathrm{MPC}}(x_k, u_k) + \bar{V}^{\mathrm{MPC}}(x_N) \\ \text{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f^{\mathrm{MPC}}(x_k, u_k), \\ & 0 \leq h^{\mathrm{MPC}}(x_k, u_k), \\ & 0 \leq h_N^{\mathrm{MPC}}(x_N).\end{aligned}$$

Constraints?

Directly handled through h^{MPC} and h_N^{MPC}

RL Policy (DDPG)

$$w \overset{Q}{\leftarrow} w + \frac{\alpha_w}{B} \sum_{i=1}^{B} \delta_i \nabla_w Q_w(S_i, A_i),$$

$$\theta \overset{\mu}{\leftarrow} \theta + \frac{\alpha_\theta}{B} \sum_{i=1}^{B} \nabla_\theta \mu_\theta(S_i)\ \nabla_a Q_w(S_i, a)|_{a=\mu_\theta(S_i)},$$

$$\delta_i := l(S_i, A_i) + \gamma Q_{\bar{w}}(S_i^+, A_i^+) - Q_w(S_i, A_i),$$

$$\text{with } (S_i, A_i, S_i^+) \sim \mathcal{D}^{\mathrm{buffer}},\ A_i^+ = \mu_\theta(S_i^+),$$

Constraints?

Hard to handle formally → use penalties in cost l or restrict action space of policy μ_θ.

Examples: MPC vs RL - Inference Time

MPC Problem

$$
\begin{aligned}
\min_{z} \quad & \sum_{k=0}^{N-1} l^{\mathrm{MPC}}(x_k, u_k) + \bar{V}^{\mathrm{MPC}}(x_N) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f^{\mathrm{MPC}}(x_k, u_k), \\
& 0 \leq h^{\mathrm{MPC}}(x_k, u_k), \\
& 0 \leq h_N^{\mathrm{MPC}}(x_N).
\end{aligned}
$$

Inference time?

Often high due to online optimization $\rightarrow$ may be prohibitive for fast systems

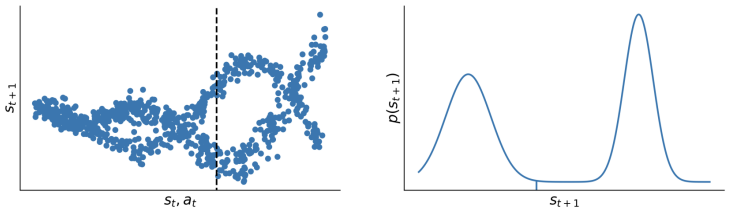
RL Policy (DDPG)

$$
\begin{aligned}
& w \overset{Q}{\leftarrow} w + \frac{\alpha_w}{B} \sum_{i=1}^{B} \delta_i \nabla_w Q_w(S_i, A_i), \\
& \theta \overset{\mu}{\leftarrow} \theta + \frac{\alpha_\theta}{B} \sum_{i=1}^{B} \nabla_\theta \mu_\theta(S_i) \; \nabla_a Q_w(S_i, a)|_{a=\mu_\theta(S_i)}, \\
& \delta_i \coloneqq l(S_i, A_i) + \gamma Q_{\bar{w}}(S_i^+, A_i^+) - Q_w(S_i, A_i), \\
& \text{with } (S_i, A_i, S_i^+) \sim \mathcal{D}^{\mathrm{buffer}}, \; A_i^+ = \mu_\theta(S_i^+),
\end{aligned}
$$

Inference time?

Low, only need forward pass of μ_θ $\rightarrow$ suitable for fast systems

Stochasticity



[Image: Moerland et al. 2022]

- **Definition**: Environment is inherently stochastic - deterministic future state prediction impossible even with perfect model knowledge
- **RL approaches**:
  - **Natural fit**: Stochasticity inherent in MDP framework
  - **Direct training**: Policies trained on real-world environment to account for actual uncertainty
  - **Challenge**: Rare events (large but infrequent deviations) make value estimation difficult

Model Mismatch

Challenge: Training environment often differs from deployment environment

RL approaches:

- Domain randomization
- Robust RL
- Meta RL - training on adaptive distribution of models

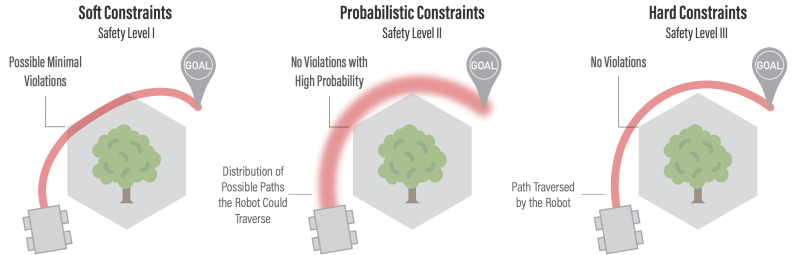
RL assumption: Real-world environment lies within training model distribution

MPC approaches:

- Stochastic MPC - explicitly accounts for model mismatch
- Robust MPC - handles correlated predictive errors across time
- Distributionally robust MPC - robustifies against distribution mismatches

MPC advantage: Inherent robustness - performs well even without explicit model mismatch consideration

Safety & Constraints



[Image: Brunke et al. 2023]

- **MPC**: Constraint satisfaction and stability are key strengths:
  - **Challenge**: Hard to handle probabilistic constraints with (unknown) uncertainty
  - **Common MPC approaches**: Robust MPC, tube-based MPC
- **RL**: Increasingly focused on safety guarantees using constrained MDPs:
  - **Challenge**: Safe RL policies often become too conservative or may still violate constraints.
  - **Common RL approaches**: Safe action sets, Lagrange formulation, trust-region optimization, control-barrier functions, MPC based safety filter

Outline

Parameterized OCP

Parameterized OCP with parameters ϕ:

OCP:

$$
\begin{aligned}
\min_{z} \quad & L_\phi(z) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f_\phi^{\text{MPC}}\big(x_k, u_k\big),\ 0 \le k < N, \\
& 0 \le h_\phi^{\text{MPC}}\big(x_k, u_k\big),\ 0 \le k < N, \\
& 0 \le \tilde{h}_\phi^{\text{MPC}}\big(x_N\big),
\end{aligned}
$$

Objective:

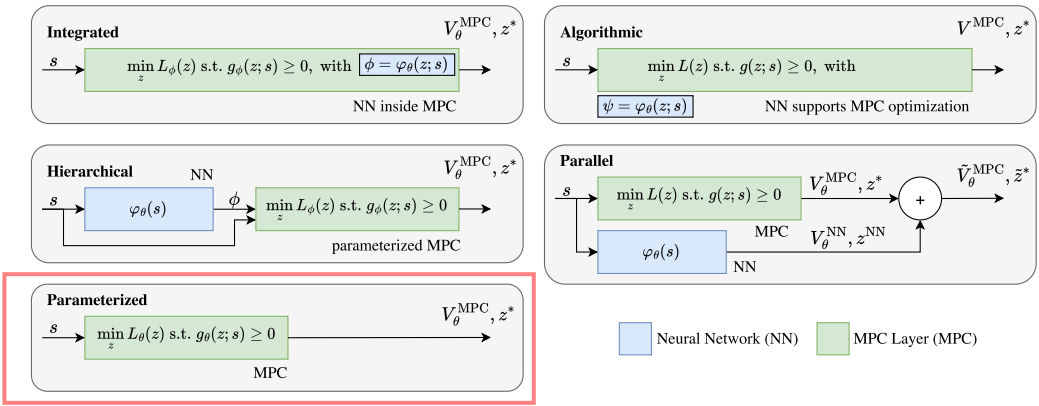
$$
L_\phi(z) := \bar{V}_\phi^{\text{MPC}}(x_N) + \sum_{k=0}^{N-1} l_\phi^{\text{MPC}}\big(x_k, u_k\big).
$$

Short notation:

$$
\min_{z} L_\phi(z) \quad \text{s.t.} \quad g_\phi(z; s) \ge 0.
$$

Parameters ϕ can affect: objective L_ϕ, dynamics f_ϕ^{MPC}, and constraints h_ϕ^{MPC}.
Note: g_ϕ encodes both dynamics and constraints.

Inference Architectures

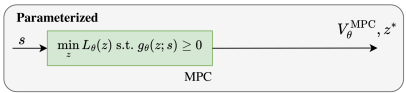


Parameterized Architecture

Architecture: Parameters θ are constant and independent of decision variables z or state s

Key characteristics:

- Parameters learned offline, fixed during deployment
- Optimization-friendly MPC formulation
- No evaluation of highly nonlinear functions



Examples:

- Simplified models [Cai et al. 2023]
- Cost function tuning

Advantages:

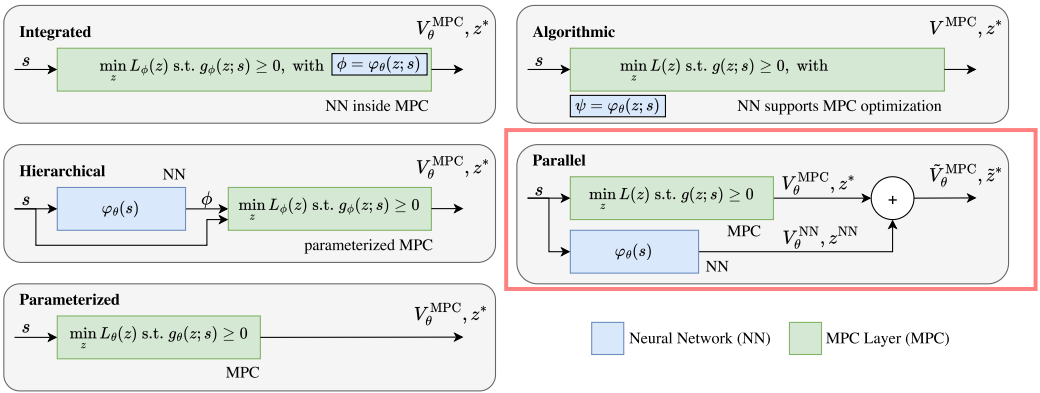
- ✓ Preserves MPC problem structure
- ✓ No NN evaluation during inference
- ✓ Suitable for standard MPC solvers
- ✓ Lower computational complexity
- ✗ Less flexibility than state-dependent architectures

A Learning-Based Model Predictive Control Strategy for Home Energy Management Systems

WENQI CAI[1], SHAMBHURAJ SAWANT[1], DIRK REINHARDT[1], SOROUSH RASTEGARPOUR[2], AND SÉBASTIEN GROS[1]

- Build a simplified MPC scheme and parameterize model, cost constraints
- Learn parameters via RL
- Maintains MPC structure, lower inference time compared to full model

Inference Architectures



Parallel Architecture

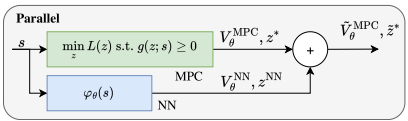
Architecture: MPC problem usually not parameterized; NN evaluated in parallel to correct MPC output

Key characteristics:

- MPC problem typically not parameterized
- NN evaluated in parallel to MPC optimization
- NN output corrects optimal solution or value function
- MPC and NN operate independently

Trade-offs:

- No differentiation through optimization ✓
- Potentially unsafe actions due to perturbation ✗



Examples:

- Value function approximation
  [Bhardwaj et al. 2020]

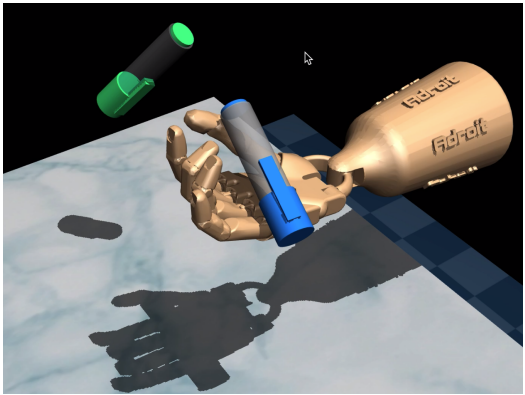
Parallel: Example

BLENDING MPC & VALUE FUNCTION APPROXIMATION FOR EFFICIENT REINFORCEMENT LEARNING

Mohak Bhardwaj[1] **Sanjiban Choudhury**[2] **Byron Boots**[1]

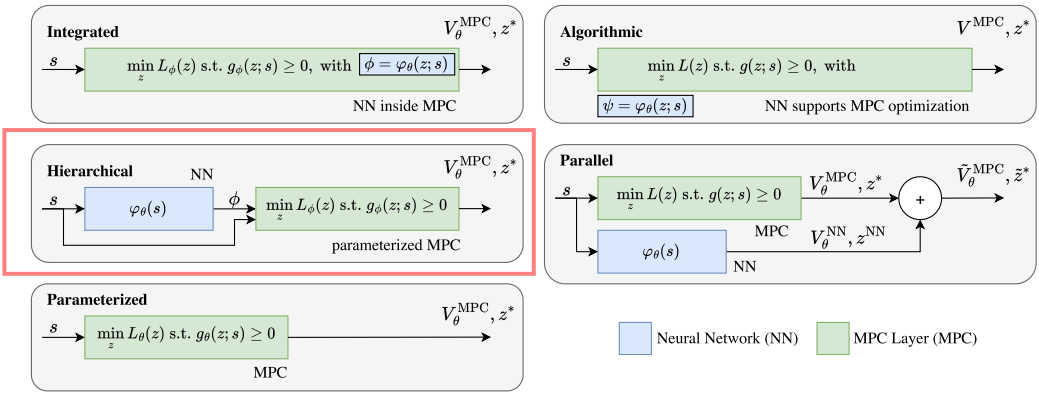
[1] University of Washington [2] Aurora Innovation Inc.

- Use MPC for local value estimates
- NN provides corrections
- Blend both estimates via weighting factor λ
- Improves data efficiency and adds structure



[Image: Bhardwaj et al. 2020]

Inference Architectures



Hierarchical Architecture

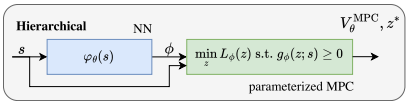
Architecture: NN provides input to MPC, parameter ϕ depends on state via $\phi = \varphi_\theta(s)$

Key characteristics:

- NN evaluated *before* solving optimization problem
- Function $\varphi_\theta(s)$ evaluated independently of MPC
- Nonlinearity does not affect optimization structure
- Parameter changes may affect convergence

Tradeoffs:

- ✓ Optimization not harder to solve
- ✓ No NN inside optimization
- ✓ Standard MPC solvers work
- ✗ MPC theory may not hold



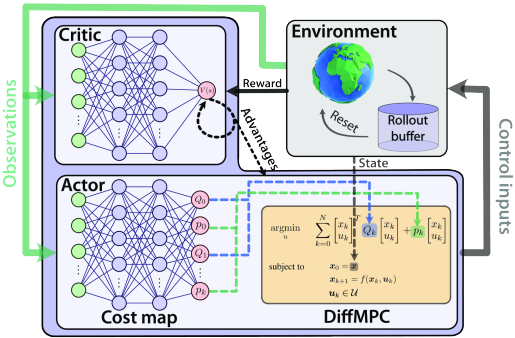
Examples:

- Reference trajectory [Brito et al. 2021] [Reiter et al. 2023]
- Cost modification [Romero et al. 2024]
- Constraint modification [Jacquet et al. 2024]

Hierarchical: Example

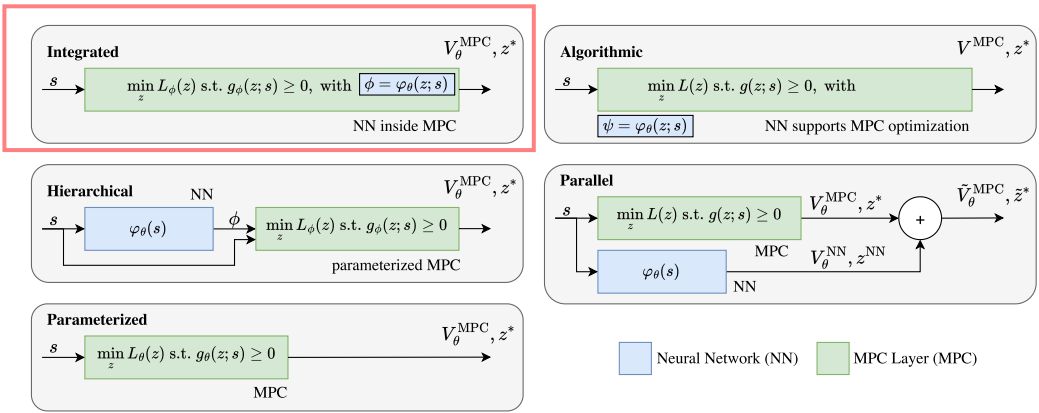
Actor-Critic Model Predictive Control

Angel Romero, Yunlong Song, Davide Scaramuzza



[Image: Romero et al. 2024]

Inference Architectures



Integrated Architecture

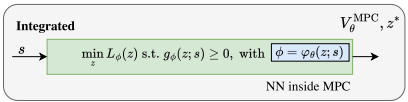
Architecture: Parameters depend on both state s and optimization variables z via $\phi = \varphi_\theta(s, z)$

Key characteristics:

- NN is part of the optimization problem itself
- Cannot be evaluated separately from optimization algorithm
- NN encoders/decoders infer states, rewards, transition models
- Outputs value function V_θ^{MPC} or decision variables $z^\star$

Tradeoffs:

- ✗ Highly nonlinear, possibly nonconvex
- ✗ Computationally expensive
- ✗ Requires $\nabla_z \varphi_\theta(z; s)$



Examples:

- Latent space MPC [Lowrey et al. 2019]
- NN transition models [Salzmann et al. 2023][Adhau et al. 2024]
- Cost function [Seel et al. 2022]

Integrated: Example

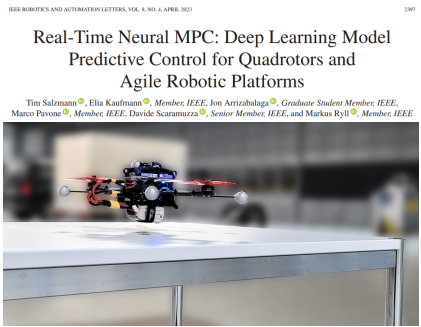
Idea:

- Learn NN dynamics model from data
- Use model for MPC planning

Anyone wants to try?

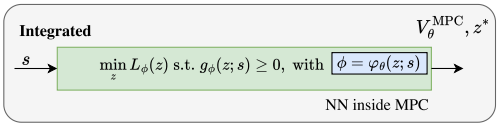
[l4casadi]

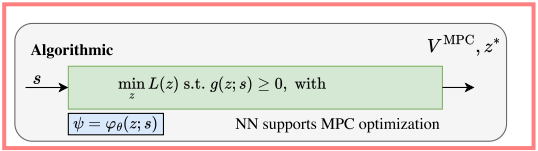
[l4acados]

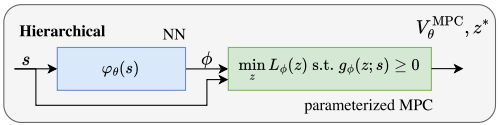


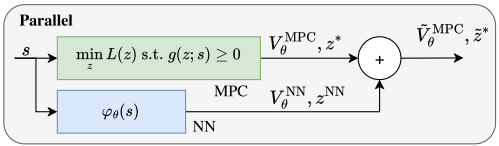
[Image: Salzmann et al. 2023]

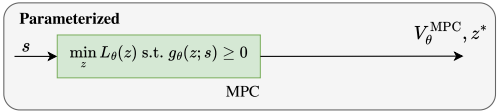
Inference Architectures











Algorithmic Architecture

Architecture: RL policy guides the MPC optimization algorithm without modifying the optimization problem itself

Key characteristics:

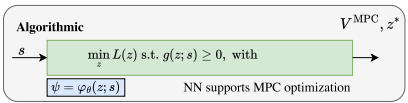
- RL policy alters hyperparameters ψ of the solver
- Optimization problem remains unchanged
- Examples of hyperparameters:
  - Initial trajectory guess
  - Trajectory samples (MPPI)
  - Active-set predictions
  - MPC recompute flag
  - Horizon length

Tradeoffs:

✓ Improve online computation time

✓ Can help escape local minima

✗ Increased complexity



Examples:

- TD-MPC2 [Hansen et al. 2024]
- Warm-starting [Grandesso et al. 2023]

Algorithmic: Example

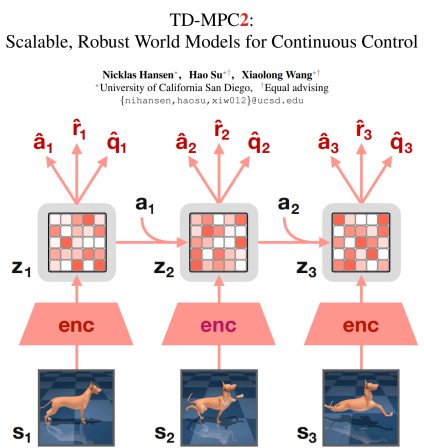
RL Components:

- Joint-embedding prediction
- Reward prediction
- TD learning (Q-function)
- Long-term credit assignment

MPC Components:

- Local trajectory optimization
- Planning in latent space
- Policy prior guidance

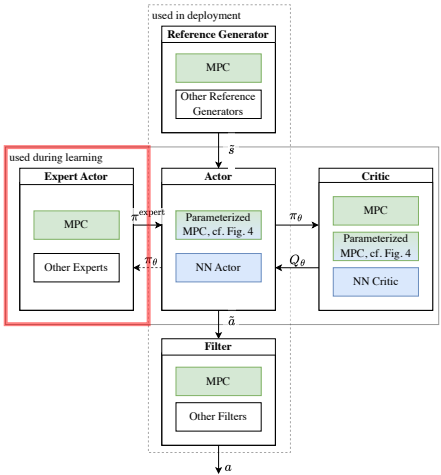
Key Innovation:
Bootstraps beyond planning horizon using learned Q-function



[Image: Hansen et al. 2023]

Outline

Taxonomy


used in deployment
Reference Generator
MPC
Other Reference Generators
$\tilde{s}$
used during learning
Expert Actor
MPC
Other Experts
π^{expert}
π_θ
Actor
Parameterized MPC, cf. Fig. 4
NN Actor
π_θ
Q_θ
Critic
MPC
Parameterized MPC, cf. Fig. 4
NN Critic
$\tilde{a}$
Filter
MPC
Other Filters
a

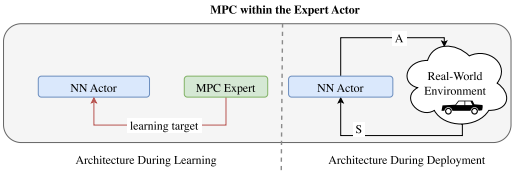
MPC as an Expert Actor

MPC used to generate training data for RL policy

Purpose: Use MPC to provide expert trajectories for imitation learning

Characteristics:

- MPC generates state-action pairs
- RL policy learns to mimic MPC behavior
- Can use behavioral cloning or DAgger
- See also [Karg and Lucia 2020]



Examples:

- Autonomous driving [Hoffmann et al. 2024]

Benefits:

- ✓ Faster inference than MPC
- ✓ Leverages domain knowledge
- ✓ Sample-efficient learning

Challenge:

- ✗ Limited to MPC performance

MPC as an Expert Actor: Example

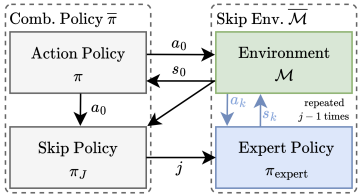
Learning When to Trust the Expert for Guided Exploration in RL

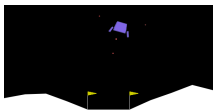
Felix Schulz *
Neurorobotics Lab
University of Freiburg
Germany
felix.schulz@student.hpi.de

Jasper Hoffmann *
Neurorobotics Lab
University of Freiburg
Germany
hoffmaja@informatik.uni-freiburg.de

Yuan Zhang
Neurorobotics Lab
University of Freiburg
Germany
yzhang@informatik.uni-freiburg.de

Joschka Bödecker
Neurorobotics Lab
University of Freiburg
Germany
jboedeck@informatik.uni-freiburg.de

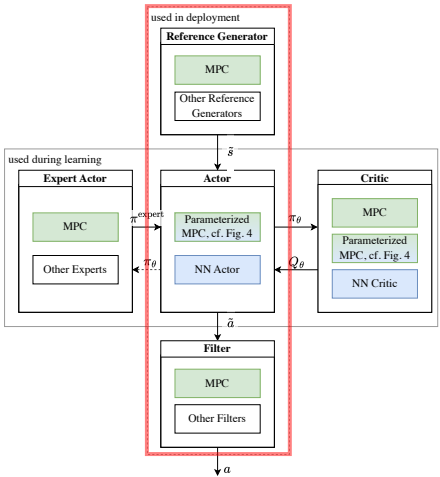




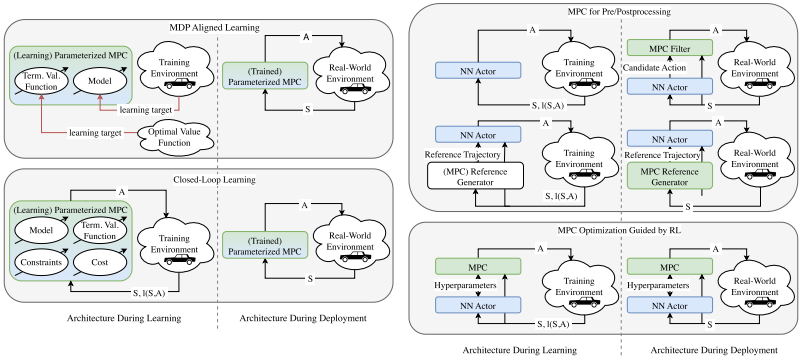
[Image: Schulz and Hoffmann et al. 2024]

- Use an expert actor for exploration during training
- A learned high-level skip policy π_J decides at each for how many steps it should either follow a learned action policy π or the MPC expert π_{expert}
- Reduce usage of expert during training

Taxonomy

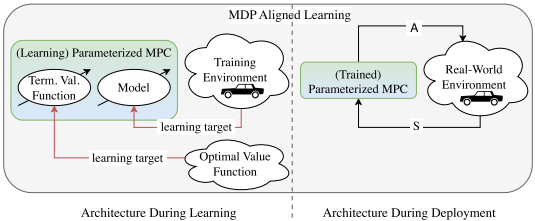

used in deployment
Reference Generator
MPC
Other Reference Generators
$\tilde{s}$
used during learning
Expert Actor
MPC
Other Experts
π^{expert}
π_θ
Actor
Parameterized MPC, cf. Fig. 4
NN Actor
π_θ
Q_θ
Critic
MPC
Parameterized MPC, cf. Fig. 4
NN Critic
$\tilde{a}$
Filter
MPC
Other Filters
a

MPC within the Deployed policy



In the following, we will focus on Aligned and Closed-Loop Learning!

Aligned Learning



Paradigm: Learn individual components of the MPC controller to better approximate the true environment. This aligns the controller's internal understanding with reality.

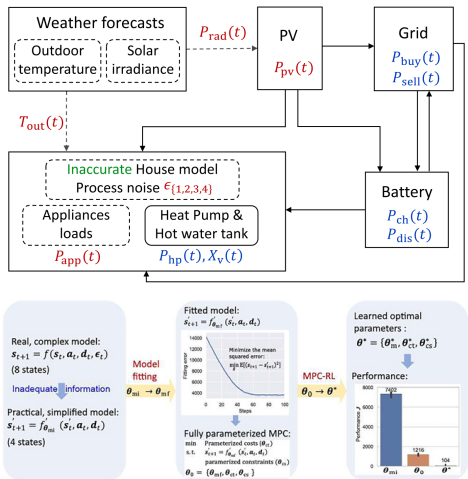
Components aligned with MDP:

- Approximate the system's dynamics by learning: $f_\theta^{\mathrm{MPC}} \approx P$
- Approximate the infinite-horizon cost with RL: $\bar{V}_\theta^{\mathrm{MPC}} \approx V^\star$

A Learning-Based Model Predictive Control Strategy for Home Energy Management Systems

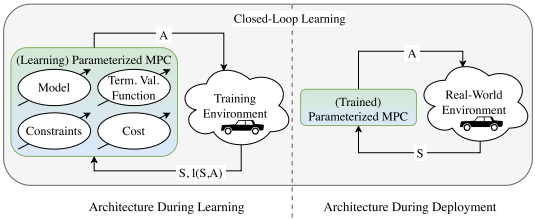
WENQI CAI[1], SHAMBHURAJ SAWANT[1], DIRK REINHARDT[1], SOROUSH RASTEGARPOUR[2], AND SÉBASTIEN GROS[1]

- Build a simplified MPC scheme and parameterize model, cost constraints
- Learn parameters via RL
- Maintains MPC structure, lower inference time compared to full model



[Image: Cai et al. 2023]

Closed-Loop Learning



Paradigm:

🏆 The focus is on what **works best**, not what is most **accurate**. The internal model or costs may become misaligned.

How is it implemented?

📈 Learn MPC parameters ϕ to directly optimize the final closed-loop performance $J(\phi)$.

🔗 **Differentiable MPC:** Treat the MPC as a layer in a policy network. Gradients are backpropagated *through* the optimization process to update parameters.

📦 **MPC as part of the Environment:** An outer-loop RL agent learns to choose the best MPC parameters ϕ as its "action", treating the MPC-controlled system as a black box.

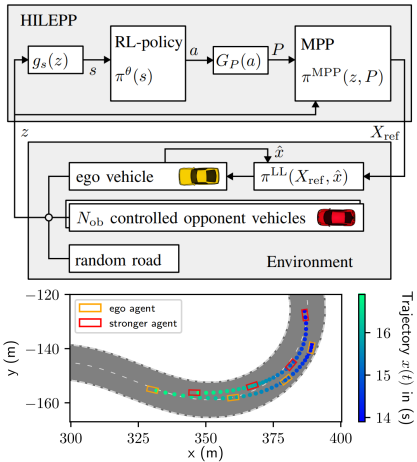
Closed-loop Optimal Learning: Example

2023 European Control Conference (ECC)
June 13-16, 2023. Bucharest, Romania

A Hierarchical Approach for Strategic Motion Planning in Autonomous Racing

Rudolf Reiter[1], Jasper Hoffmann[2], Joschka Boedecker[2] and Moritz Diehl[1,3]

- Car-racing environment
- RL learns to set reference trajectories
- MPC tracks learned reference trajectories



[Image: Reiter et al. 2023]

Comparison: Aligned Learning and Closed-Loop Learning

Aligned Learning

⛳ Approximate the true Markov Decision Process

Model Learning: Supervised learning on transition data, e.g.,

$$\min_{\theta} \mathcal{L}(S^+ - f_\theta^{\mathrm{MPC}}(S, A))$$

Terminal Value Learning: RL-style updates towards

$$V_\theta^{\mathrm{MPC}} \approx V^\star$$

Tradeoffs:

- ✓ Interpretable model, cost, constraints
- ✓ Division of design into manageable subtasks
- ✓ Safe, generalizes better
- ✗ Sub-optimal closed-loop performance
- ✗ May require complex models
- ✗ Restrictive model requirements (differentiability, smoothness)

Closed-Loop Learning

⛳ Approximate the optimal policy

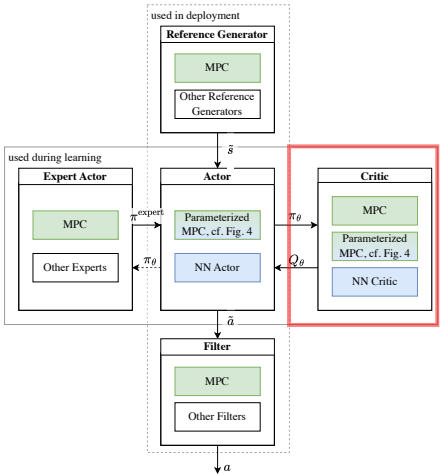
Policy Learning: Directly optimize closed-loop performance, towards

$$\mu_\theta^{\mathrm{MPC}} \approx \mu^\star$$

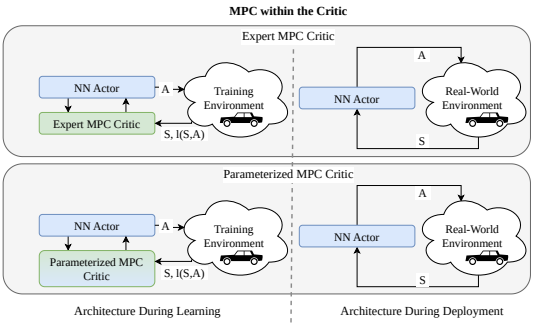
Tradeoffs:

- ✓ Direct optimization of actual objective
- ✓ Can correct for model errors through learning
- ✓ Flexibility in model choice
- ✗ Limited generalization to new scenarios
- ✗ Not representing true system
- ✗ Reduced adaptability when requirements change

Taxonomy


used in deployment
Reference Generator
MPC
Other Reference Generators
$\tilde{s}$
used during learning
Expert Actor
MPC
Other Experts
π^{expert}
π_θ
Actor
Parameterized MPC, cf. Fig. 4
NN Actor
π_θ
Q_θ
Critic
MPC
Parameterized MPC, cf. Fig. 4
NN Critic
$\tilde{a}$
Filter
MPC
Other Filters
a

MPC as the Critic



$$
\begin{aligned}
Q^{\mathrm{MPC}}(s,a) = \min_{z} \quad & \sum_{k=0}^{N-1} l^{\mathrm{MPC}}(x_k, u_k) + \bar{V}^{\mathrm{MPC}}(x_N) \\
\text{s.t.} \quad & x_0 = s, \\
& u_0 = a, \\
& x_{k+1} = f^{\mathrm{MPC}}(x_k, u_k),\ 0 \le k < N, \\
& 0 \le h^{\mathrm{MPC}}(x_k, u_k),\ 1 \le k < N, \\
& 0 \le h_N^{\mathrm{MPC}}(x_N).
\end{aligned}
$$

- We can use MPC to derive a Q-function Q^{MPC}, which can be used as a critic!
- We constrain the initial control u_0 to be a.

Example: Learnable Critic

Instead of using a fixed MPC critic, we can further learn to improve the critic:

- In [Bhardwaj et al. 2020], the author combine sampling-based MPC with a learned value function using a parallel architecture solving multiple robot manipulation tasks. A simplified version of the critic is given by:

$$Q(s,a) = (1-\lambda)\ \underbrace{Q_w(s,a)}_{\text{learned critic}} + \lambda\ \underbrace{Q^{\text{MPC}}(s,a)}_{\text{model-based}},$$

  where Q_w is a learned critic and Q^{MPC} is the MPC-based Q-function.
- The parameter λ balances the two Q-functions and is decayed over time.

Example: Learnable Critic from parameterized MPC

- [Anand et al. 2023] proposed a simple actor-critic scheme, where a single MPC scheme is used for both actor and critic.
- **Idea:** Close-to-optimal MPC parameters ϕ allow approximation of the optimal Q-function $Q^\star$.

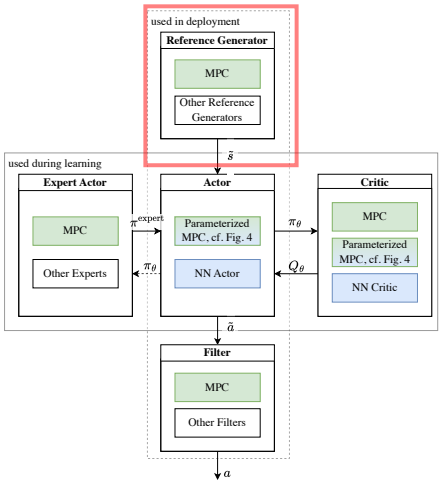
Local Q-function approximation:

$$Q_w(s,a) = Q_\phi^{\mathrm{MPC}}(s,a) + \nabla_\phi Q_\phi^{\mathrm{MPC}}(s,a)^\top w$$

MPC Q-Function

$$\begin{aligned} Q_\phi^{\mathrm{MPC}}(s,a) = \min_z \quad & \sum_{k=0}^{N-1} l_\phi^{\mathrm{MPC}}(x_k,u_k) + \bar{V}_\phi^{\mathrm{MPC}}(x_N) \\ \text{s.t.} \quad & x_0 = s, \\ & u_0 = a, \\ & x_{k+1} = f_\phi^{\mathrm{MPC}}(x_k,u_k), \ , \\ & 0 \le h_\phi^{\mathrm{MPC}}(x_k,u_k), , \\ & 0 \le h_{N_\phi}^{\mathrm{MPC}}(x_N). \end{aligned}$$

Taxonomy


used in deployment
Reference Generator
MPC
Other Reference Generators
$\tilde{s}$
used during learning
Expert Actor
MPC
Other Experts
π^{expert}
π_θ
Actor
Parameterized MPC, cf. Fig. 4
NN Actor
π_θ
Q_θ
Critic
MPC
Parameterized MPC, cf. Fig. 4
NN Critic
$\tilde{a}$
Filter
MPC
Other Filters
a

MPC for Preprocessing

MPC used in deployed policy, but not during training

Preprocessing: Reference Generator

Purpose: Provide reference trajectory for RL policy

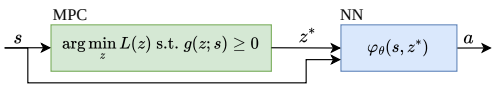
Characteristics:

- MPC outputs planned trajectory
- RL policy uses reference as input

Benefit:

- ✓ Interpretable reference trajectory
- ✓ Bandwidth and learning at lower level

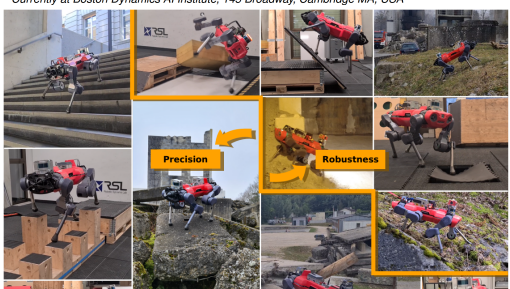
MPC as reference generator



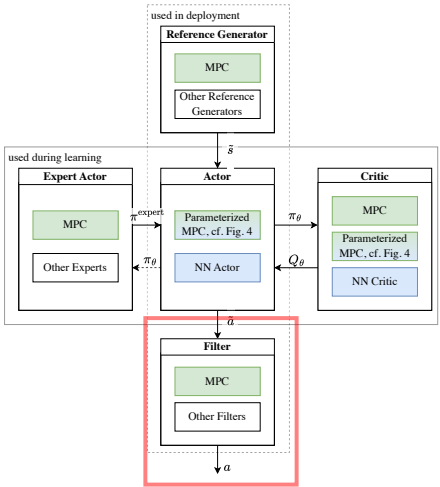
DTC: Deep Tracking Control

FABIAN JENELTEN,[1*] JUNZHE HE,[1] FARBOD FARSHIDIAN,[2] AND MARCO HUTTER[1]

[1] *Robotic Systems Lab, ETH Zurich, 8092 Zurich, Switzerland.*
[2] *Currently at Boston Dynamics AI Institute, 145 Broadway, Cambridge MA, USA*



Taxonomy


used in deployment
Reference Generator
MPC
Other Reference Generators
$\tilde{s}$
used during learning
Expert Actor
MPC
Other Experts
π^{expert}
π_θ
Actor
Parameterized MPC, cf. Fig. 4
NN Actor
π_θ
Q_θ
Critic
MPC
Parameterized MPC, cf. Fig. 4
NN Critic
a
Filter
MPC
Other Filters
a

MPC for Postprocessing

MPC used in deployed policy, but not during training

Postprocessing: Safety Filter

Purpose: Ensure safety w.r.t. model and constraints

Characteristics:

- MPC filters RL policy output
- Projects actions to safe set

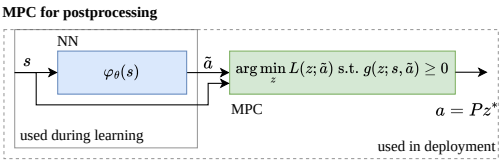
Benefit:

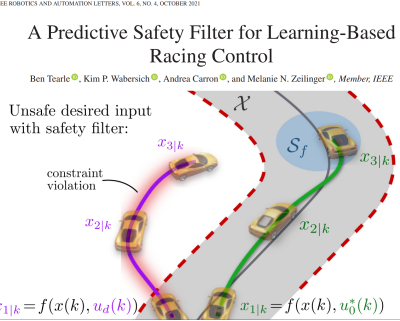
✓ Provides safety guarantees w.r.t. model and constraints

Issue:

✗ Policy unaware of filter during training

✗ May lead to suboptimal actions





Outline

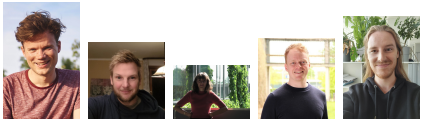
Next Steps and Current Challenges

- 🚀 **Computational efficiency:** Scaling to high-dimensional offline RL with massive datasets remains challenging
- 🛡️ **Risk-aware learning:** Moving beyond robust/stochastic MPC toward sophisticated risk metrics (e.g., mission completion probabilities)
- 🧩 **Expanding frameworks:** Generalization from MPC to broader planning (combinatorial optimization, MILP, stochastic multi-stage programming)
- 👥 **Multi-agent systems:** Coordination, communication, and distributed decision-making largely unresolved
- 💼 **Application domains:** Need for more practical demonstrations across diverse sectors (energy, healthcare, logistics)

Learning for Predictive Control (leap-c)

Key features:

- ⚙️ Hierarchical NN-MPC architecture implementation
- 🐍 Seamless integration of acados into PyTorch as differentiable layer
- 📈 Supports differentiation of control/state trajectories and Q-function w.r.t. parameters
- 🚀 Efficient multithreaded implementation for batched inputs
- 🧰 Modular codebase
- 🔁 Focus on closed-loop learning
- 🚧 Missing a differentiable convex programming layer
- 🔗 `https://github.com/leap-c/leap-c`



Summary

Having heard this lecture, you should have a better understanding of...

- ⚖️ Identify strengths and weaknesses of MPC and RL
- 🤝 Recognize potential benefits of combining both approaches
- 🏗️ Distinguish between parameterized MPC architectures:
  - Integrated, hierarchical, parallel, parameterized
- 🛠️ Design or use MPC-RL algorithms based on:
  - 👨‍🚀 MPC as expert actor
  - ⚙️ MPC within deployed policy
  - 🔍 MPC as critic
  - 🧭 MPC as reference generator
  - 🛡️ MPC as a safety filter

Thank you for your attention!