Trajectory Design for Space Systems

EN.530.626 (Fall 2025)

Lecture 25

Instructor: Prof. Abhishek Cauligi
Course Assistant 1: Arnab Chatterjee
Course Assistant 2: Mark Gonzales

Model-based optimal control

- Until now, our class has focused on *model*-based trajectory optimization

$$\begin{aligned} \min_{x_{0:N}, u_{0:N}} \quad & g_N(x_N) + \sum_{k=0}^{N-1} g(x_k, u_k) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k; \theta), \quad k = 0, \ldots, N-1, \\ & h_i(x_k, u_k; \theta) \leq 0, \quad k = 0, \ldots, N. \end{aligned}$$

- This allows us to make strong assumptions on the structure and form of the system dynamics, constraints, and objective function

Model-based optimal control

Stochastic setting

- We recently extended it to settings with *uncertainty:*
    - e.g., Additive disturbances w_k

$$
\begin{aligned}
\min_{x_{0:N}, u_{0:N}} \quad & g_N(x_N) + \sum_{k=0}^{N-1} g(x_k, u_k) \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k, w_k; \theta), \quad k = 0, \ldots, N-1, \\
& h_i(x_k, u_k; \theta) \leq 0, \quad k = 0, \ldots, N, \\
& w_k \sim \mathscr{W} .
\end{aligned}
$$

- Other sources of uncertainty include:
    - Parameter uncertainty, dynamics mismatch, estimation/sensing noise

Reinforcement learning

A primer

- What is reinforcement learning and how does it connect with optimal control?
- Where does machine learning arise in the context of reinforcement learning?
- What are different “flavors” of reinforcement learning?

Reinforcement learning

Preliminaries

- Principle of Optimality by Richard Bellman [1957]:

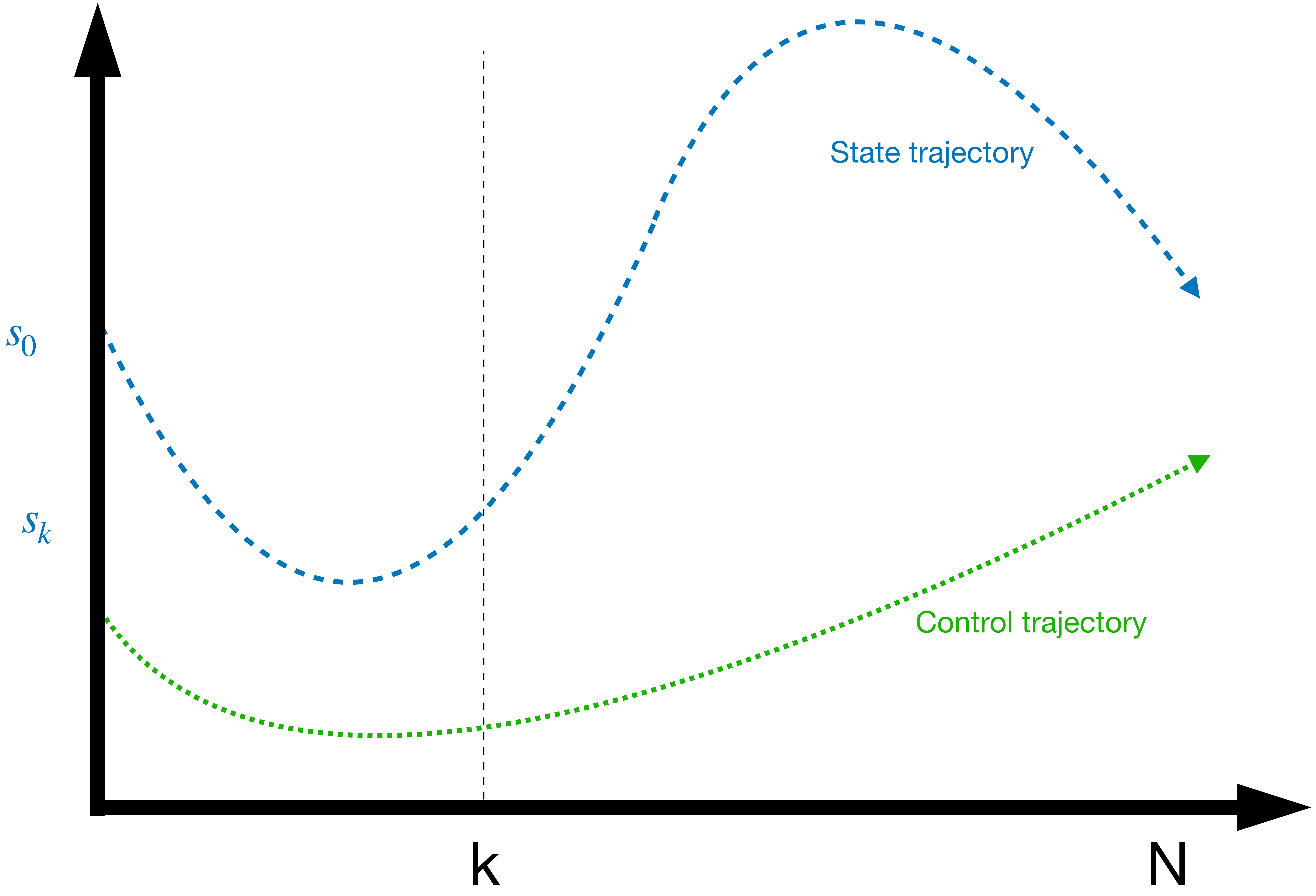
> “An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.”

- Informally: the tail of an optimal trajectory is optimal too

Reinforcement learning

Preliminaries

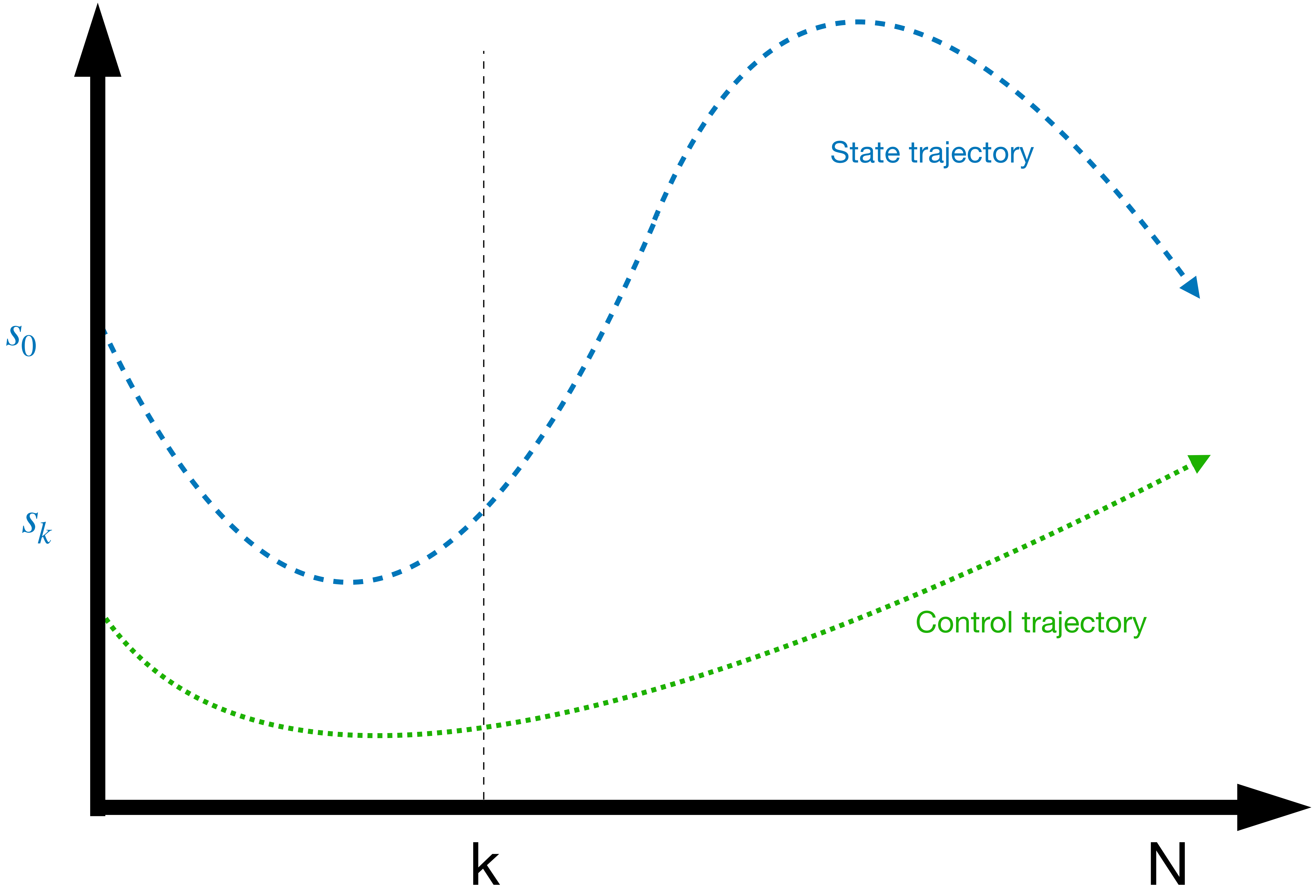
Informally: the tail of an optimal trajectory is optimal too



Reinforcement learning

Preliminaries

Sub-trajectory $[u_k, \dots, u_{N-1}]$ is optimal when starting from s_k at time k



Reinforcement learning

Preliminaries

- This tail optimality can also be expressed via the recursive definition of the cost-to-go $V_k(z)$

$$V_k(z) = \min_w g_k(z, w) + V_{k+1}(f(z, w))$$

Linear quadratic regulator

- Recall from our lecture on LQR that there were two key assumptions used to solve the problem:
  - Search only over the control sequence $\{u_0, \ldots, u_{N-1}\}$ rather than the full state and action space
  - Assume the cost-to-go is a quadratic function (“ansatz”)

$$\min_{x_{0:N}, u_{0:N}} x_N Q_g x_N + \sum_{k=0}^{N-1} x_k Q_k x_k + u_k R_k u_k$$

$$\text{s.t.} \quad x_{k+1} = A x_k + B u_k, \quad k = 0, \ldots, N-1,$$

$$x_0 = x_{\text{init}} \cdot$$

Linear quadratic regulator

- Recall from our lecture on LQR that there were two key assumptions used to solve the problem:
  - Search only over the control sequence $\{u_0, \ldots, u_{N-1}\}$ rather than the full state and action space
  - Assume the cost-to-go is a quadratic function (“ansatz”)
- We then recovered both a value function and a control policy

$$V_k(z) = \frac{1}{2} z^T P_k z$$

$$u_k = -K_k z \xrightarrow{\qquad} \pi_k(z) = -K_k z$$

Linear quadratic regulator

- Utility of the value function and control policy

$$V_k(z) = \frac{1}{2} z^T P_k z \qquad\qquad \pi_k(z) = -K_k z$$

  - The value function explicitly characterizes cost-to-go at time *k* starting from some state *z*
  - The policy yields the optimal control without having to explicitly solve from the control each iteration
- In a *closed-loop* setting, LQR allows us to regulate to some goal x_{goal} in a receding horizon fashion
- Note: $V_k(z)$ and $\pi_k(z)$ are inextricably linked
  - $V_k(z)$ is the cost-to-go while following policy $\pi_k(z)$

Reinforcement learning

Beyond LQR

- LQR only works when
  - The dynamics are deterministic and linear
  - There are no state or control constraints
- How can we compute global policies when:
  - Dynamics are highly nonlinear, discontinuous, or intractable to model?
  - System experiencing stochastic transitions or uncertain parameters?
  - Conventional trajectory optimization solvers are too brittle?

Reinforcement learning

Markov decision process

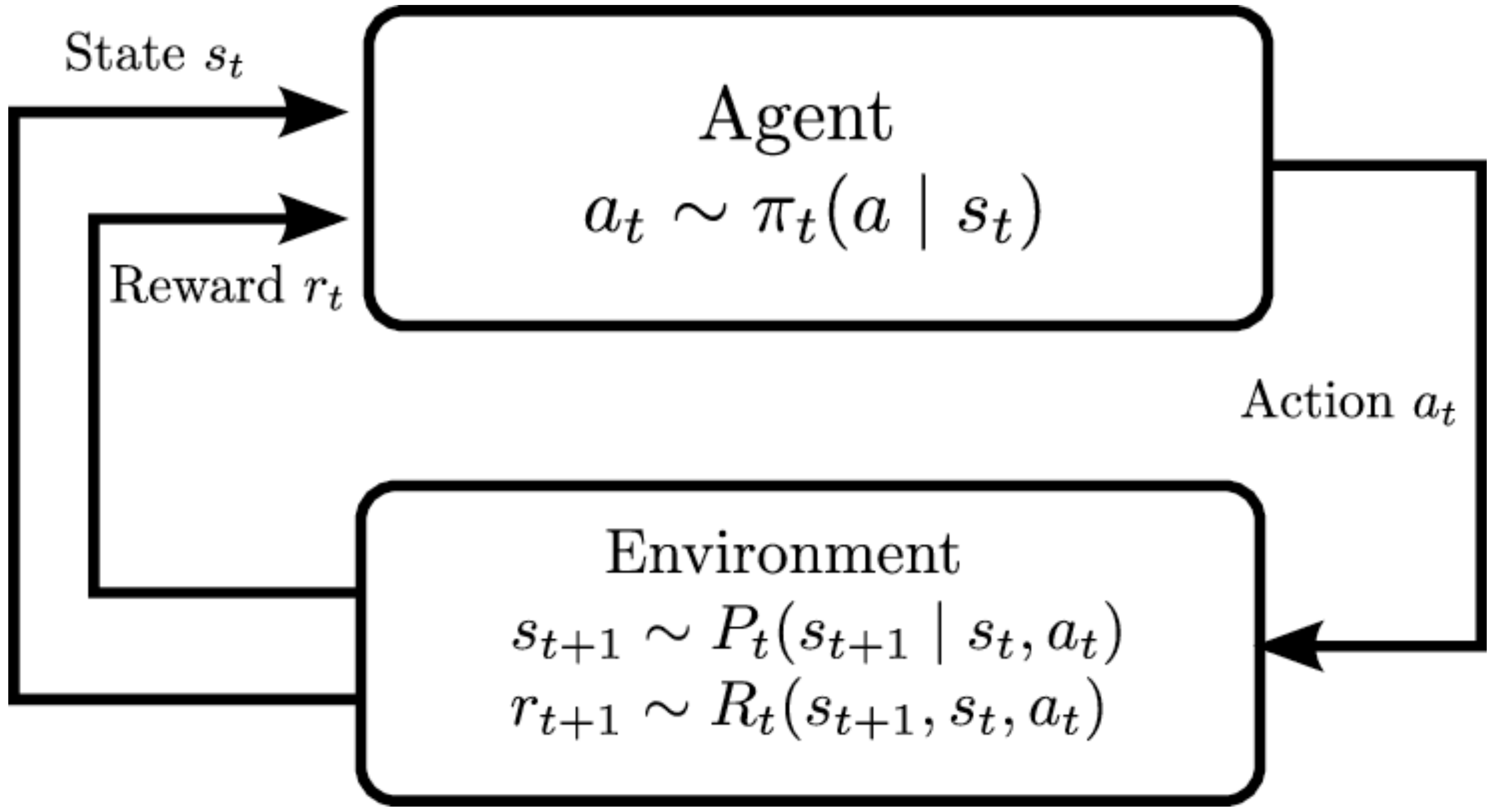
- Model the system as a Markov decision process (MDP)
- Generic modeling formulation for decision making problems (beyond dynamical systems)

$\mathcal{S}$	State space
$\mathcal{U}$	Action/control space
$x_{k+1} \sim T(x_{k+1} \mid x_k, u_k)$	Transition distribution
$g_k(x_k, u_k)$	Cost/reward function
$\gamma \in [0,1]$	Discount factor

Reinforcement learning

Beyond LQR

- Model the system as a Markov decision process (MDP)
- Simulate experiences and learn a value function and/or policy via data
- Approximate the optimal solution via exhaustive simulations


State s_t
Agent
$a_t \sim \pi_t(a \mid s_t)$
Reward r_t
Action a_t
Environment
$s_{t+1} \sim P_t(s_{t+1} \mid s_t, a_t)$
$r_{t+1} \sim R_t(s_{t+1}, s_t, a_t)$

Sutton & Barto [1998]

Reinforcement learning

Function approximation

- Where does deep learning enter the picture?
- For LQR, the policy was linear and the cost-to-go was quadratic

$$V_k(z) = \frac{1}{2} z^T P_k z \qquad \pi_k(z) = -K_k z$$

- In general, we cannot make such strong assumptions for an arbitrary system
- Instead, a common approach is to employ *function approximation* and learn some parameters ϕ:

$$V_k(z) \approx V_{\phi,k}(z) \qquad \pi_k(z) \sim \pi_{\phi,k}(z)$$

Reinforcement learning

Function approximation

- Instead, a common approach is to employ *function approximation* and learn some parameters ϕ:

$$V_k(z) \approx V_{\phi,k}(z) \qquad\qquad \pi_k(z) \sim \pi_{\phi,k}(z)$$

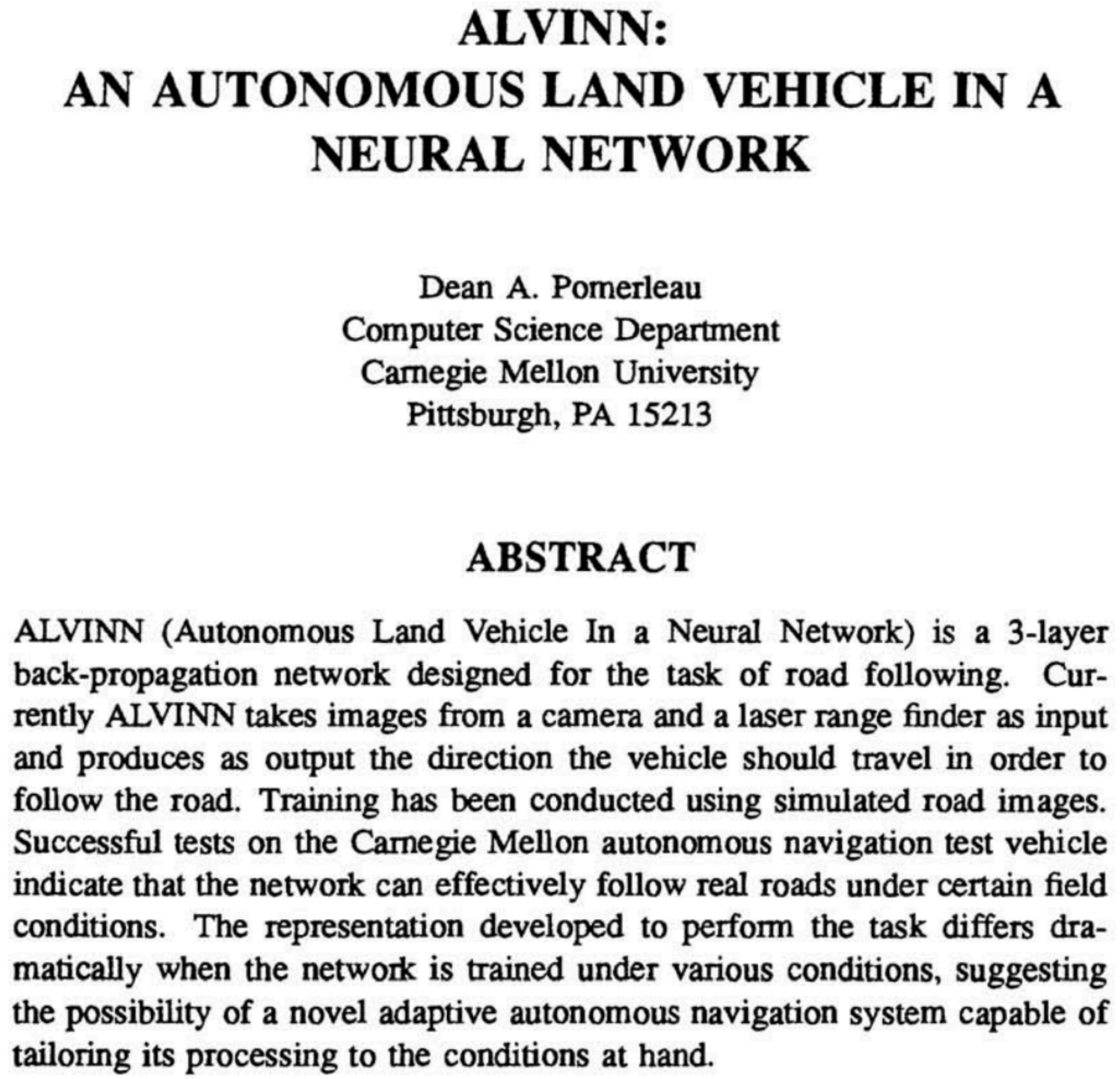
- Neural networks: feedforward, RNN, LSTM, transformer, etc.

- Non-parametric (no ϕ to learn): Gaussian processes, KNN, etc.

Reinforcement learning

Function approximation

- Aside: neural networks have a long history of use in optimal control



ALVINN: AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK

Dean A. Pomerleau
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

ALVINN (Autonomous Land Vehicle In a Neural Network) is a 3-layer back-propagation network designed for the task of road following. Currently ALVINN takes images from a camera and a laser range finder as input and produces as output the direction the vehicle should travel in order to follow the road. Training has been conducted using simulated road images. Successful tests on the Carnegie Mellon autonomous navigation test vehicle indicate that the network can effectively follow real roads under certain field conditions. The representation developed to perform the task differs dramatically when the network is trained under various conditions, suggesting the possibility of a novel adaptive autonomous navigation system capable of tailoring its processing to the conditions at hand.

D. Pomerleau, “ALVINN: An Autonomous Land Vehicle in a Neural Network”, in *Conf. On Neural Information Processing Systems*, 1989.

Reinforcement learning

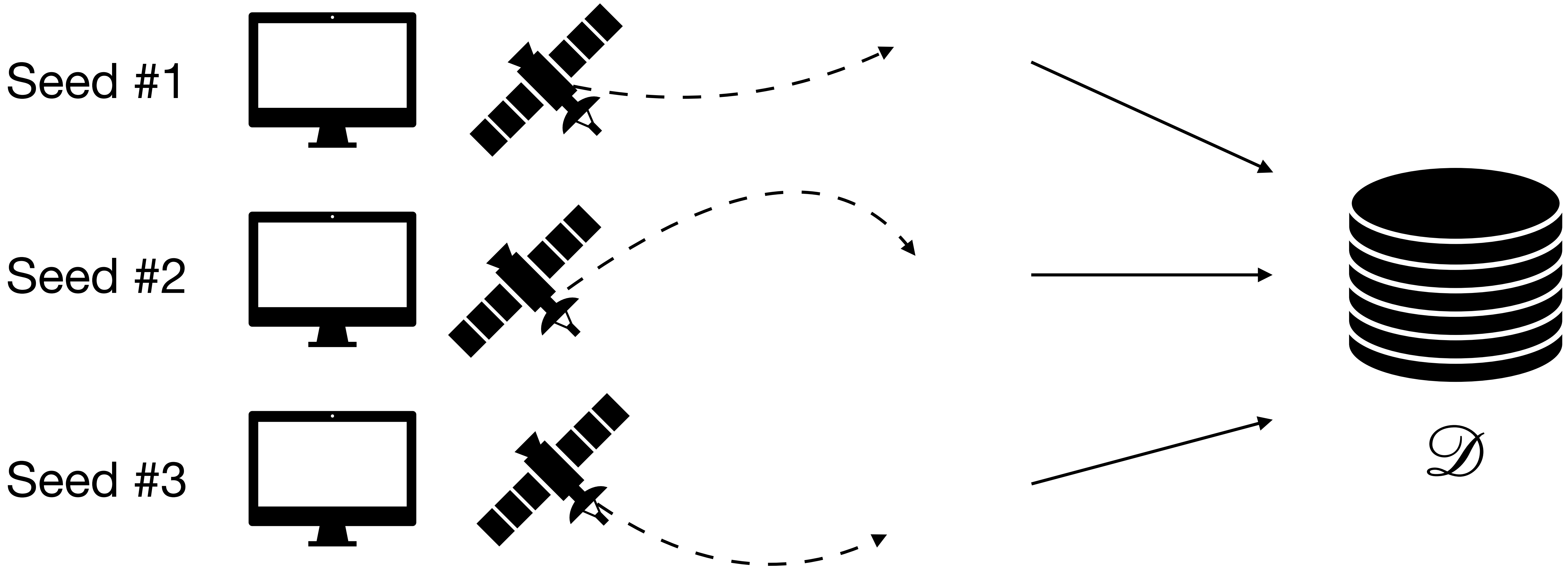
Function approximation

- Aside: neural networks have a long history of use in optimal control
- What changed in the past decade?
    - Innovations in hardware (GPUs), machine learning libraries (`cuda`/`PyTorch`/`TensorFlow`), and simulators

Reinforcement learning

Function approximation

- For *deep* reinforcement learning, the value and policy functions are updated using stochastic gradient descent (typically first-order update equations)



$\pi_{\phi,k}(z), V_{\phi,k}(z)$

$$\phi^{(k+1)} = \phi^{(k)} - \alpha \nabla_{\phi} \mathscr{L}(\mathscr{D})$$

Reinforcement learning

Engineering considerations

- Variety of RL algorithms that vary in terms of the underlying MDP modeling assumptions
- The exploration vs. exploitation problem must be considered to trade off gathering new data to find right loss landscape
- RL is notoriously brittle - need to use the right function class, conduct architecture search, tune the optimizer hyperparameters

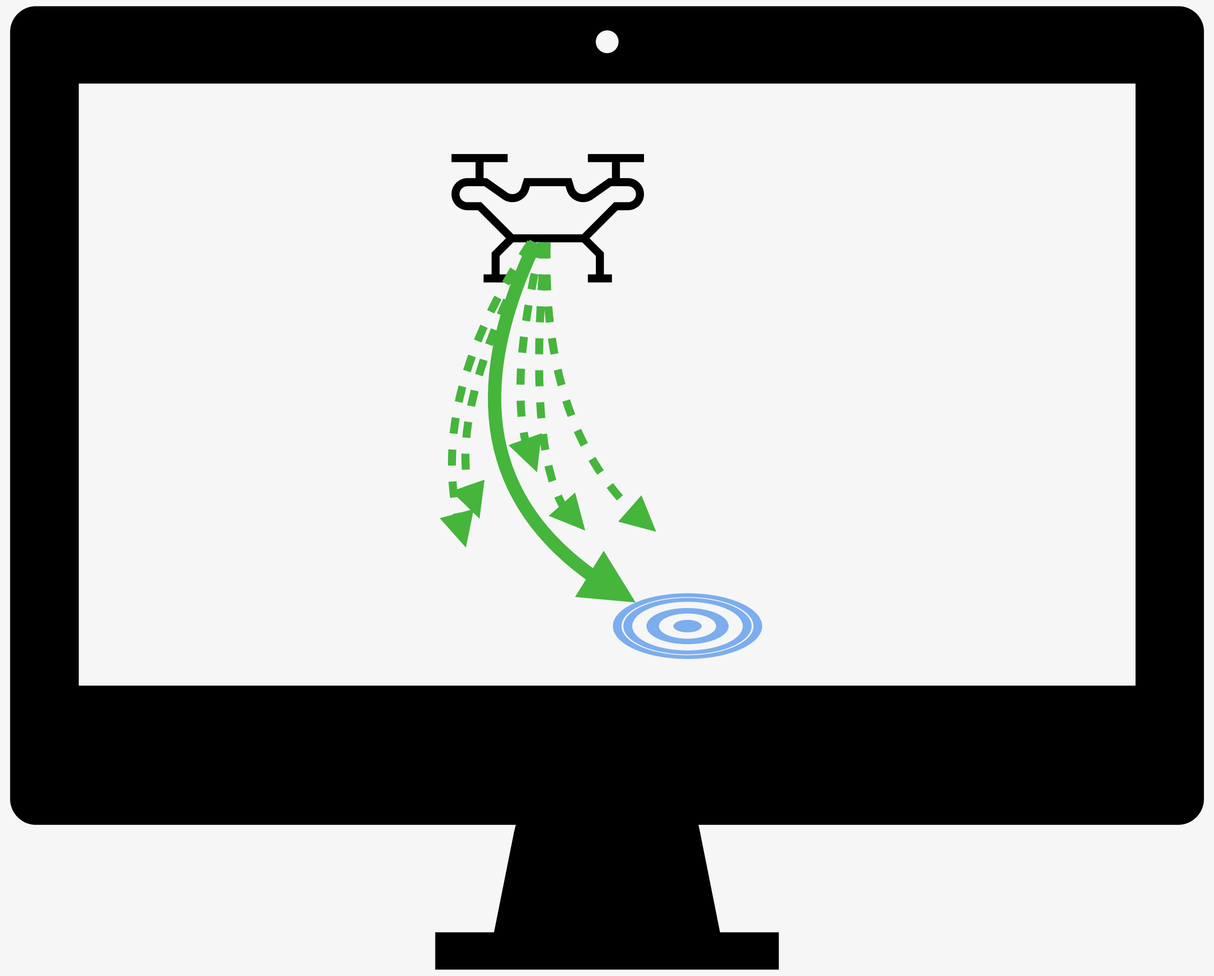
Research directions

- Recently, there is a surge of interest in connecting techniques developed in the domain of reinforcement learning back into optimal control formulations

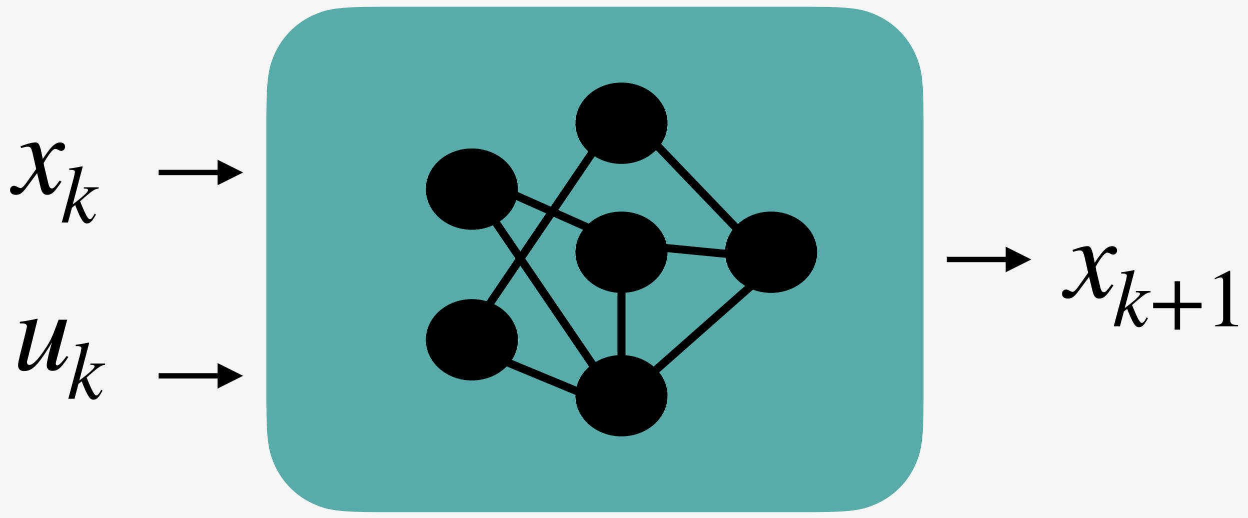
Model-based reinforcement learning

An MPC approach

Offline

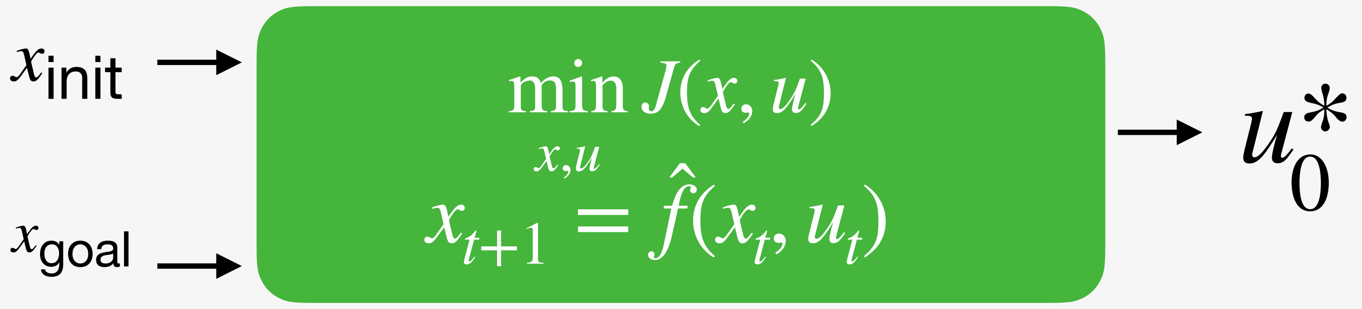


Simulate experiences



Estimate $\hat{f}(x_k, u_k)$

Online



$$\min_{x,u} J(x, u)$$
$$x_{t+1} = \hat{f}(x_t, u_t)$$

Solve optimal control problem

Value function learning

- Value function learning approaches seek to simulate a closed-loop MPC controller and use function approximation to *learn* the cost-to-go

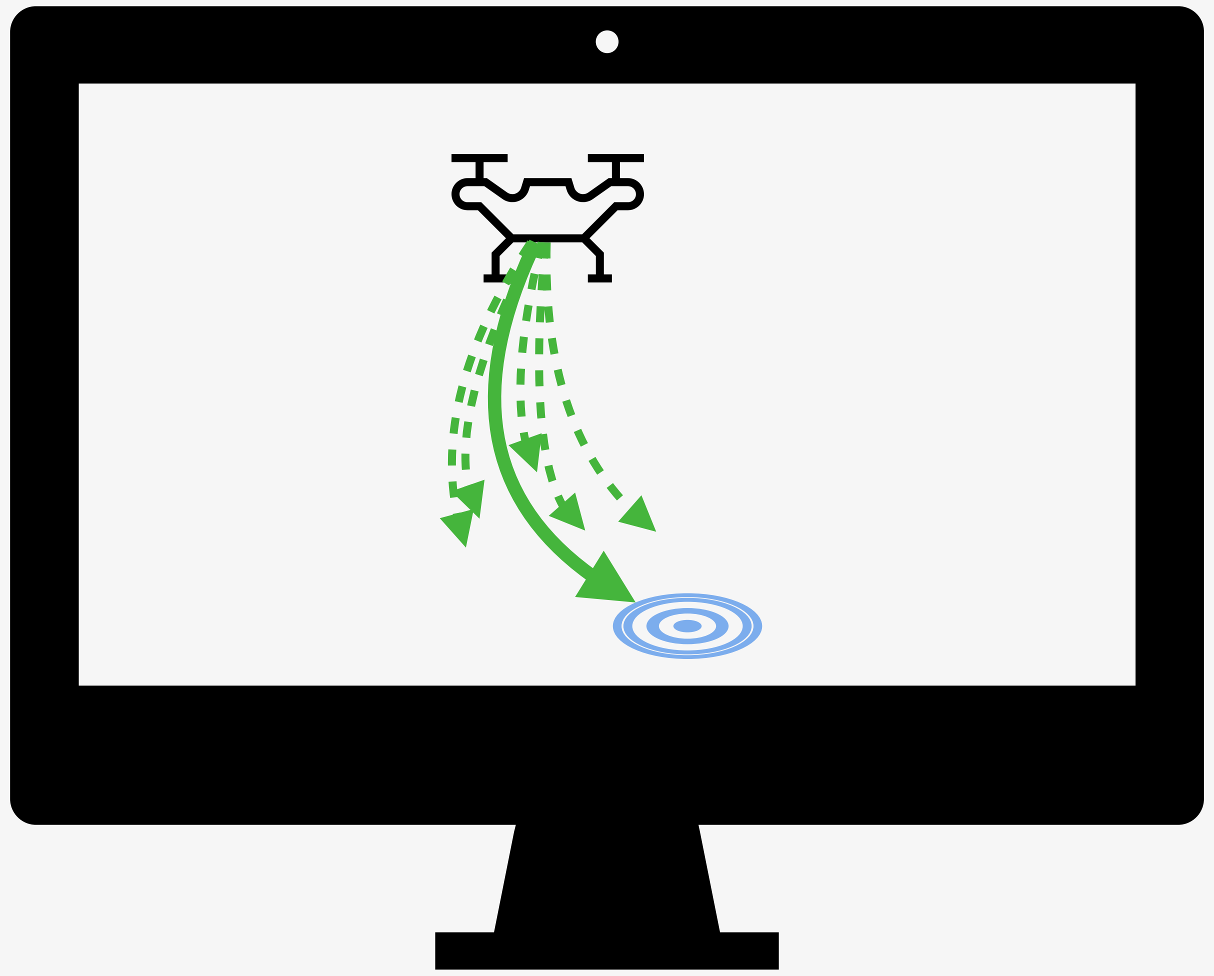
- Key idea: learn long-horizon behavior offline (horizon N) and approximate value function for use with shorter horizon N_h (where $N_h < < N$)

$$J(x_{0:N}, u_{0:N}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \xrightarrow{N_h < < N} J(x_{0:N_h}, u_{0:N_h}) = V_\phi(x_{N_h}) + \sum_{k=0}^{N_h-1} g_k(x_k, u_k)$$

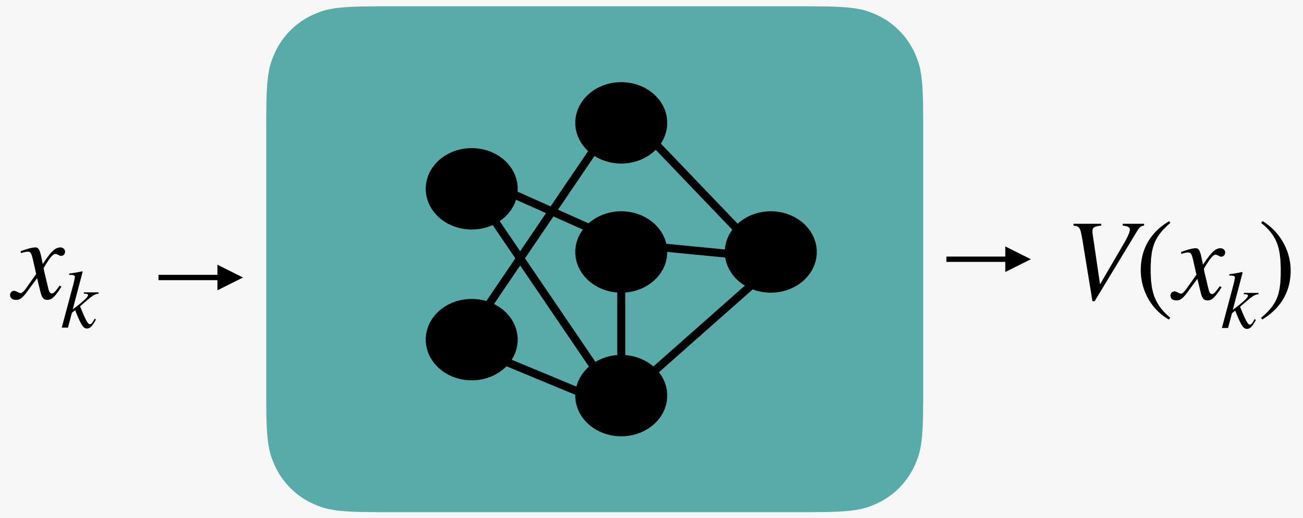
$$V_\phi(z) \approx g_N(z) + \sum_{\tau=N_h}^{N-1} g_k(z_\tau, u_\tau)$$

Value function learning

Offline



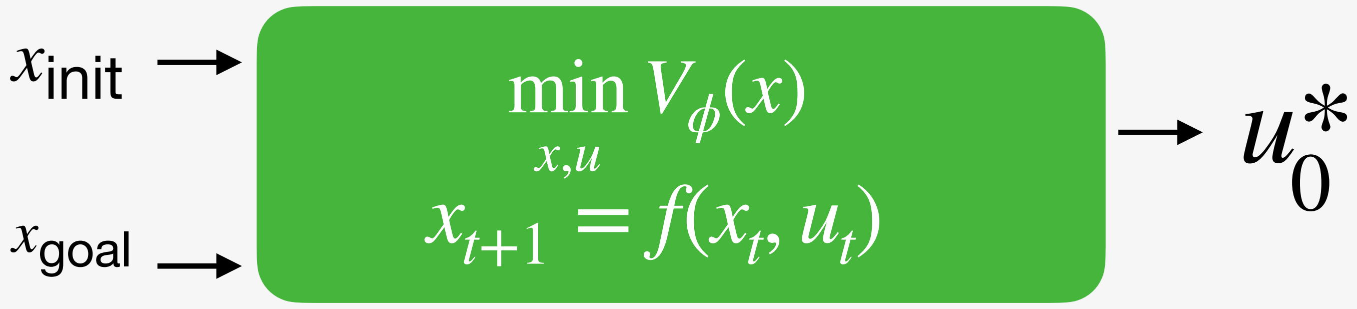
Simulate experiences



$x_k \rightarrow$ $\rightarrow V(x_k)$

Estimate $\hat{V}(x)$

Online



$x_{\text{init}} \rightarrow$

$x_{\text{goal}} \rightarrow$

$$\min_{x,u} V_\phi(x)$$
$$x_{t+1} = f(x_t, u_t)$$

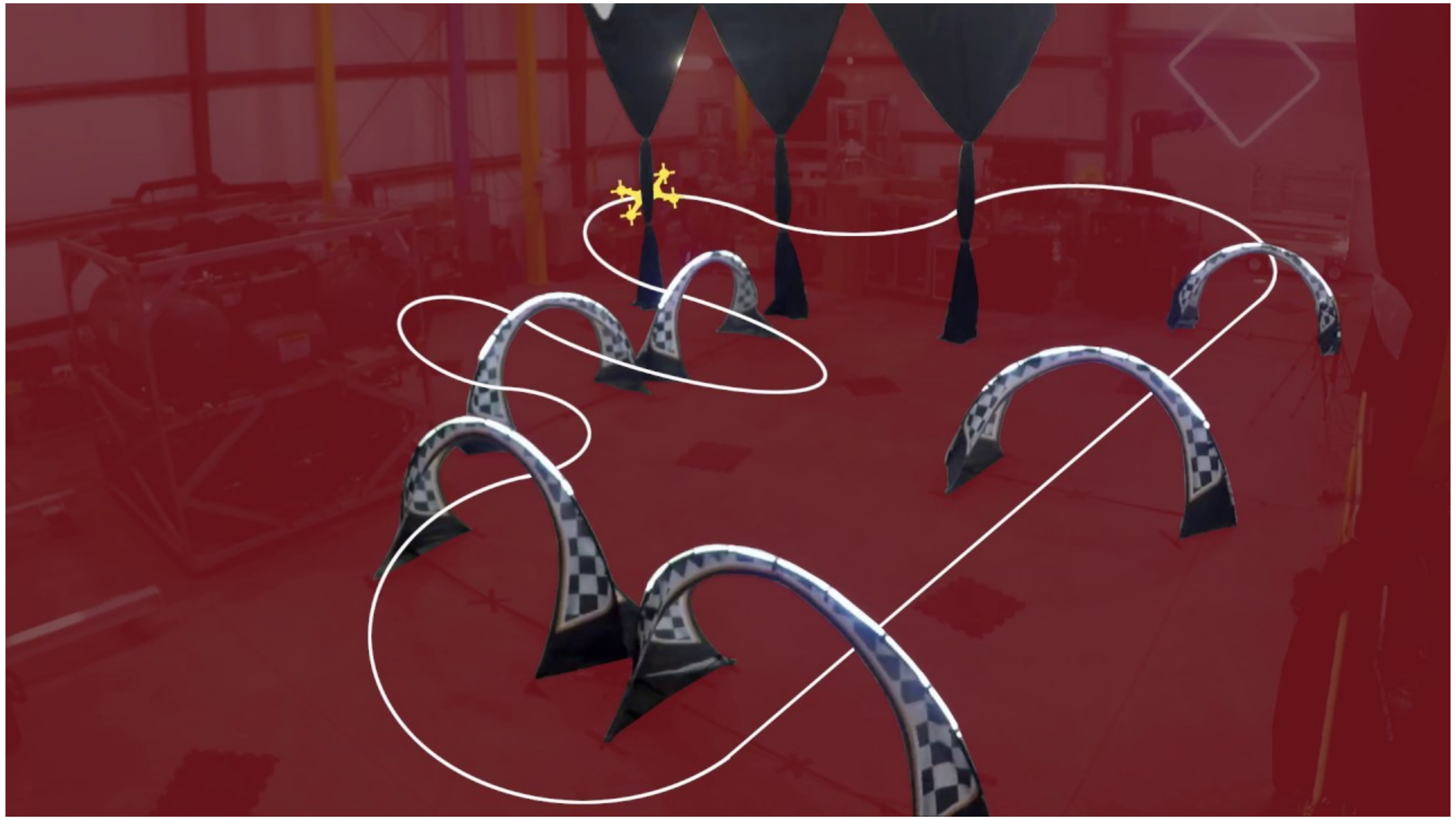
$\rightarrow u_0^*$

Solve optimal control problem

Case study: drone racing

Differential flatness-based trajectory optimization

"Compared to Loo, the drones flew more cautiously but consistently. Their algorithms are still a work in progress. For example, the drones sometimes moved so fast that motion blur caused them to lose track of their surroundings…For the official laps, Loo averaged 11.1 seconds, compared to the autonomous drones, which averaged 13.9 seconds."



https://www.nasa.gov/aeronautics/drone-race-human-versus-artificial-intelligence/

Case study: drone racing

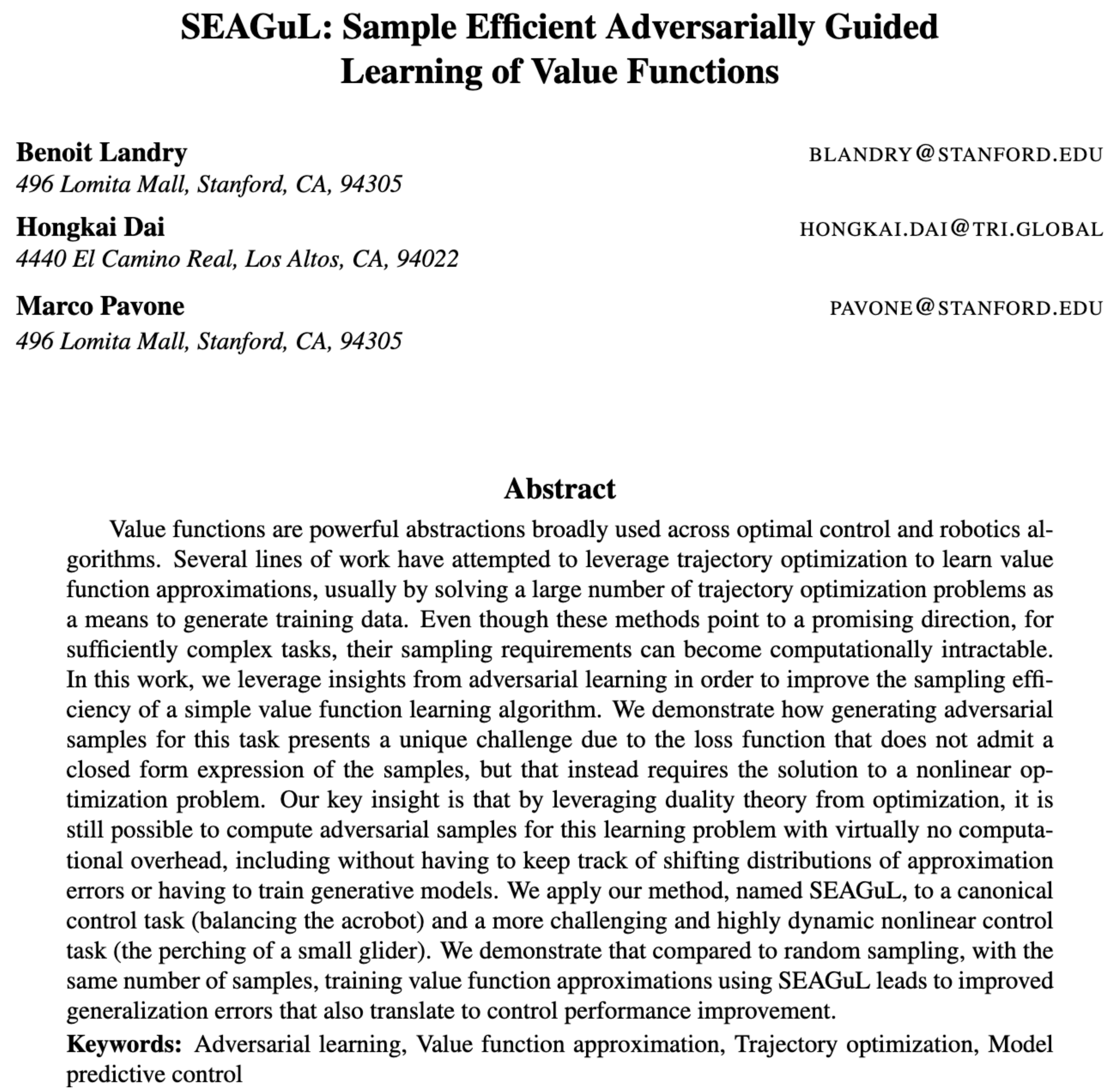
End-to-end reinforcement learning

“Swift won several races against each of the human pilots and achieved the fastest race time recorded during the events. Our work marks the first time, to our knowledge, that an autonomous mobile robot achieved world-champion-level performance in a real-world competitive sport.”

E. Kaufmann, L. Bauersfeld, et al., “Champion-level drone racing using deep reinforcement learning”, in *Nature*, vol. 620, pp. 982 —987, 2023.

Research directions

Value function learning



SEAGuL: Sample Efficient Adversarially Guided Learning of Value Functions

Benoit Landry BLANDRY@STANFORD.EDU
496 Lomita Mall, Stanford, CA, 94305

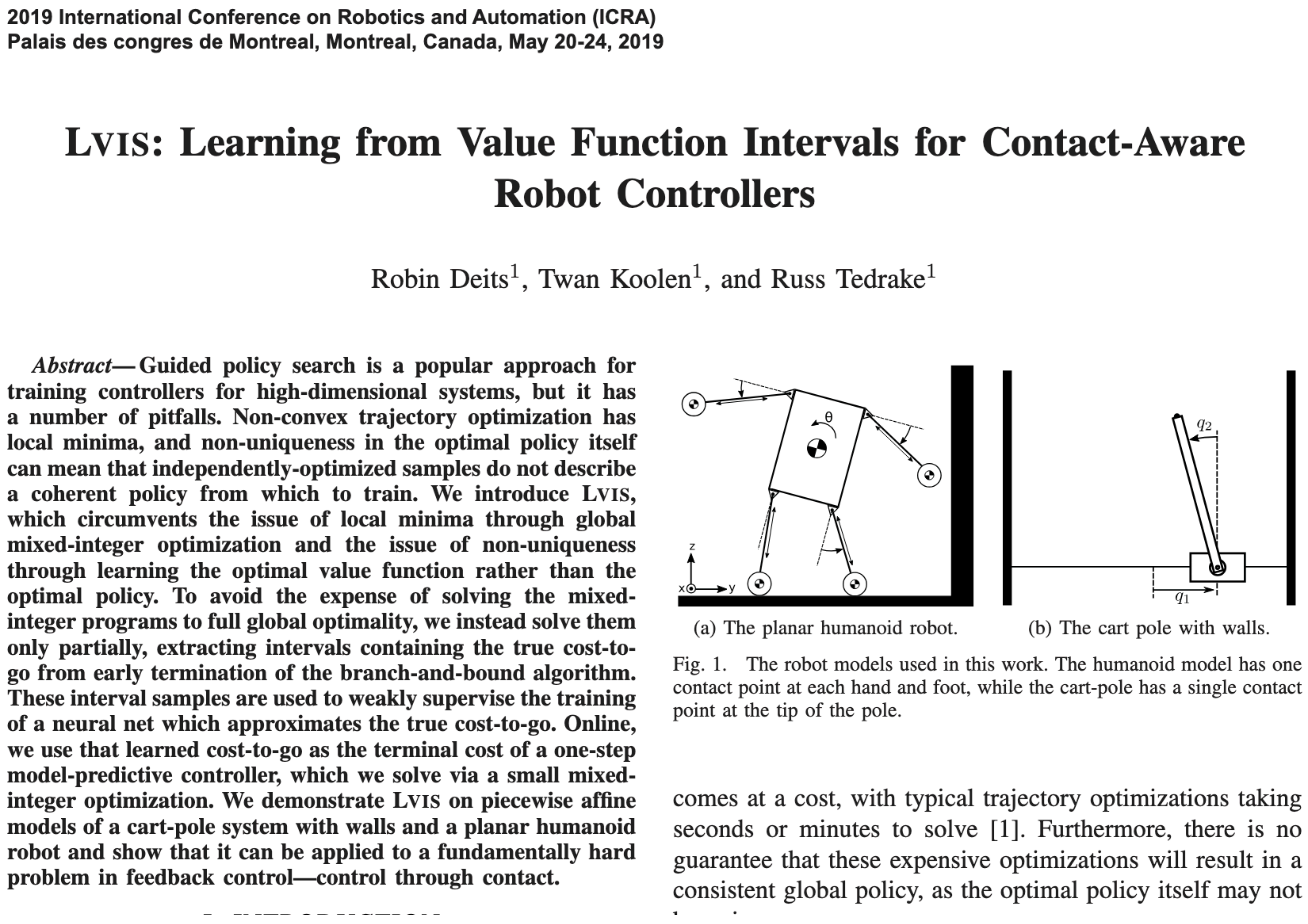
Hongkai Dai HONGKAI.DAI@TRI.GLOBAL
4440 El Camino Real, Los Altos, CA, 94022

Marco Pavone PAVONE@STANFORD.EDU
496 Lomita Mall, Stanford, CA, 94305

Abstract

Value functions are powerful abstractions broadly used across optimal control and robotics algorithms. Several lines of work have attempted to leverage trajectory optimization to learn value function approximations, usually by solving a large number of trajectory optimization problems as a means to generate training data. Even though these methods point to a promising direction, for sufficiently complex tasks, their sampling requirements can become computationally intractable. In this work, we leverage insights from adversarial learning in order to improve the sampling efficiency of a simple value function learning algorithm. We demonstrate how generating adversarial samples for this task presents a unique challenge due to the loss function that does not admit a closed form expression of the samples, but that instead requires the solution to a nonlinear optimization problem. Our key insight is that by leveraging duality theory from optimization, it is still possible to compute adversarial samples for this learning problem with virtually no computational overhead, including without having to keep track of shifting distributions of approximation errors or having to train generative models. We apply our method, named SEAGuL, to a canonical control task (balancing the acrobot) and a more challenging and highly dynamic nonlinear control task (the perching of a small glider). We demonstrate that compared to random sampling, with the same number of samples, training value function approximations using SEAGuL leads to improved generalization errors that also translate to control performance improvement.

Keywords: Adversarial learning, Value function approximation, Trajectory optimization, Model predictive control



2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

LVIS: Learning from Value Function Intervals for Contact-Aware Robot Controllers

Robin Deits[1], Twan Koolen[1], and Russ Tedrake[1]

Abstract— **Guided policy search is a popular approach for training controllers for high-dimensional systems, but it has a number of pitfalls. Non-convex trajectory optimization has local minima, and non-uniqueness in the optimal policy itself can mean that independently-optimized samples do not describe a coherent policy from which to train. We introduce LVIS, which circumvents the issue of local minima through global mixed-integer optimization and the issue of non-uniqueness through learning the optimal value function rather than the optimal policy. To avoid the expense of solving the mixed-integer programs to full global optimality, we instead solve them only partially, extracting intervals containing the true cost-to-go from early termination of the branch-and-bound algorithm. These interval samples are used to weakly supervise the training of a neural net which approximates the true cost-to-go. Online, we use that learned cost-to-go as the terminal cost of a one-step model-predictive controller, which we solve via a small mixed-integer optimization. We demonstrate LVIS on piecewise affine models of a cart-pole system with walls and a planar humanoid robot and show that it can be applied to a fundamentally hard problem in feedback control—control through contact.**

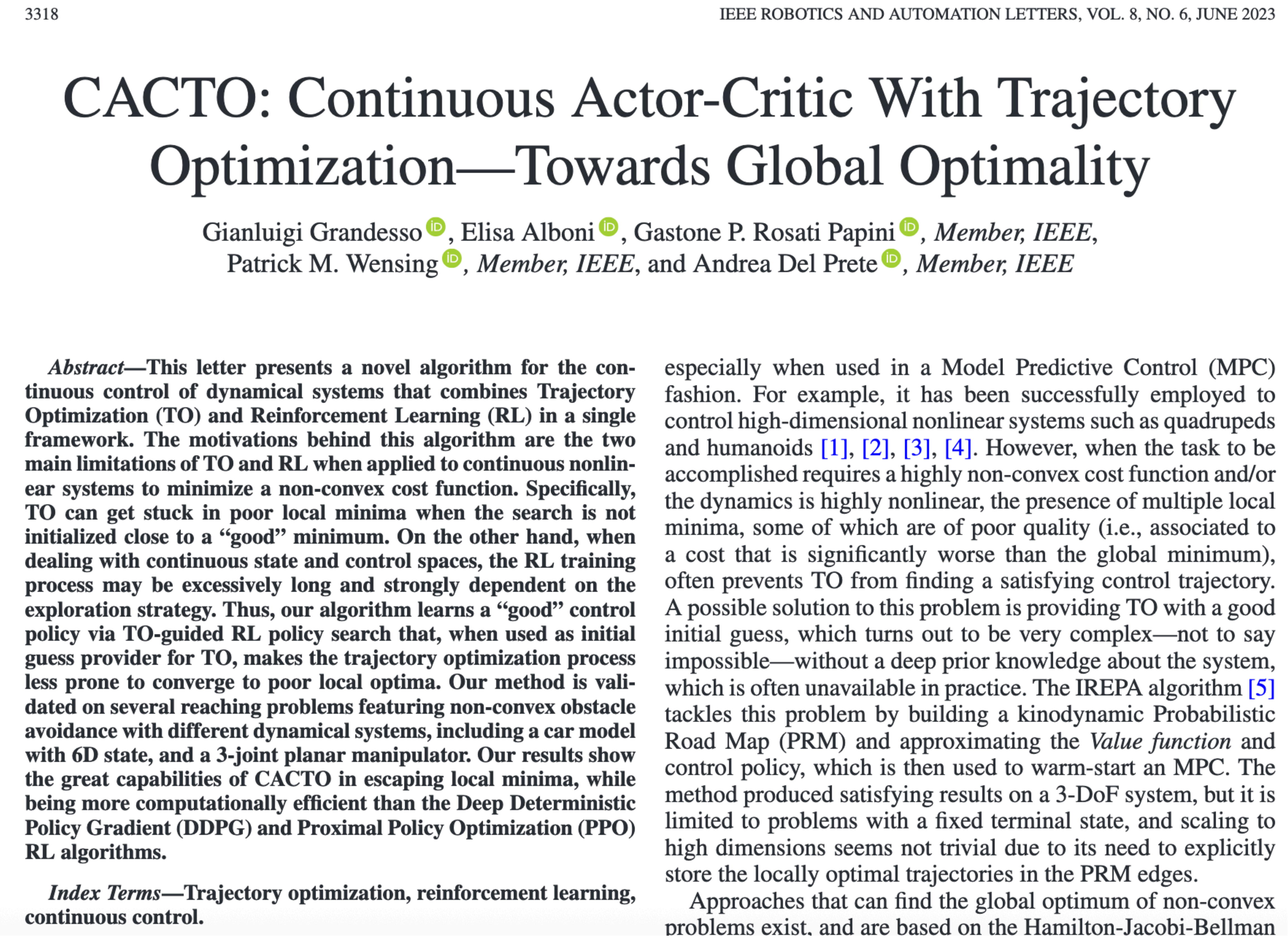
(a) The planar humanoid robot. (b) The cart pole with walls.

Fig. 1. The robot models used in this work. The humanoid model has one contact point at each hand and foot, while the cart-pole has a single contact point at the tip of the pole.

comes at a cost, with typical trajectory optimizations taking seconds or minutes to solve [1]. Furthermore, there is no guarantee that these expensive optimizations will result in a consistent global policy, as the optimal policy itself may not

Research directions

Actor-critic MPC



3318 IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 8, NO. 6, JUNE 2023

CACTO: Continuous Actor-Critic With Trajectory Optimization—Towards Global Optimality

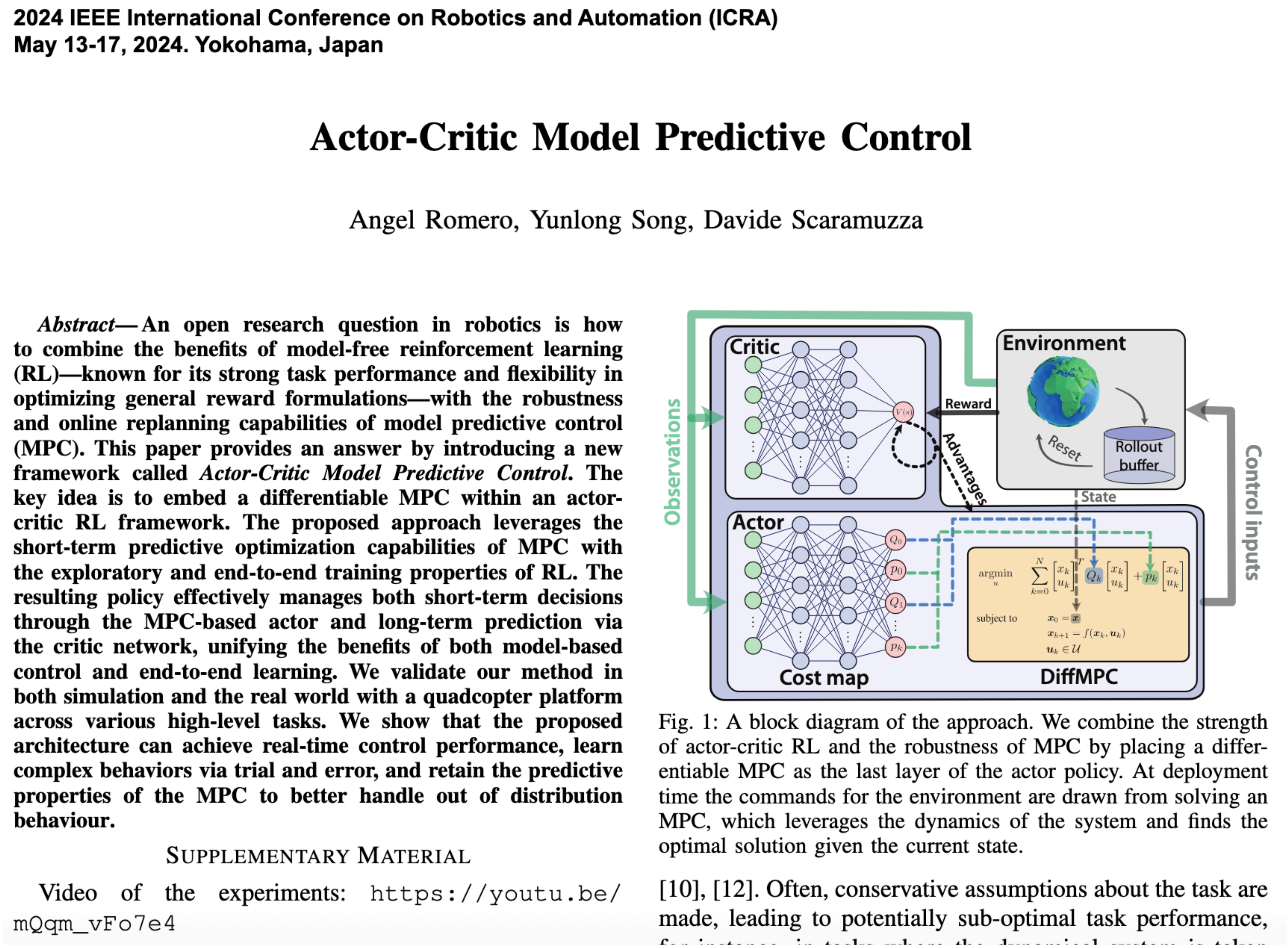
Gianluigi Grandesso, Elisa Alboni, Gastone P. Rosati Papini, *Member, IEEE*, Patrick M. Wensing, *Member, IEEE*, and Andrea Del Prete, *Member, IEEE*

***Abstract*—This letter presents a novel algorithm for the continuous control of dynamical systems that combines Trajectory Optimization (TO) and Reinforcement Learning (RL) in a single framework. The motivations behind this algorithm are the two main limitations of TO and RL when applied to continuous nonlinear systems to minimize a non-convex cost function. Specifically, TO can get stuck in poor local minima when the search is not initialized close to a "good" minimum. On the other hand, when dealing with continuous state and control spaces, the RL training process may be excessively long and strongly dependent on the exploration strategy. Thus, our algorithm learns a "good" control policy via TO-guided RL policy search that, when used as initial guess provider for TO, makes the trajectory optimization process less prone to converge to poor local optima. Our method is validated on several reaching problems featuring non-convex obstacle avoidance with different dynamical systems, including a car model with 6D state, and a 3-joint planar manipulator. Our results show the great capabilities of CACTO in escaping local minima, while being more computationally efficient than the Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO) RL algorithms.**

***Index Terms*—Trajectory optimization, reinforcement learning, continuous control.**

especially when used in a Model Predictive Control (MPC) fashion. For example, it has been successfully employed to control high-dimensional nonlinear systems such as quadrupeds and humanoids [1], [2], [3], [4]. However, when the task to be accomplished requires a highly non-convex cost function and/or the dynamics is highly nonlinear, the presence of multiple local minima, some of which are of poor quality (i.e., associated to a cost that is significantly worse than the global minimum), often prevents TO from finding a satisfying control trajectory. A possible solution to this problem is providing TO with a good initial guess, which turns out to be very complex—not to say impossible—without a deep prior knowledge about the system, which is often unavailable in practice. The IREPA algorithm [5] tackles this problem by building a kinodynamic Probabilistic Road Map (PRM) and approximating the *Value function* and control policy, which is then used to warm-start an MPC. The method produced satisfying results on a 3-DoF system, but it is limited to problems with a fixed terminal state, and scaling to high dimensions seems not trivial due to its need to explicitly store the locally optimal trajectories in the PRM edges.

Approaches that can find the global optimum of non-convex problems exist, and are based on the Hamilton-Jacobi-Bellman



2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Actor-Critic Model Predictive Control

Angel Romero, Yunlong Song, Davide Scaramuzza

***Abstract*— An open research question in robotics is how to combine the benefits of model-free reinforcement learning (RL)—known for its strong task performance and flexibility in optimizing general reward formulations—with the robustness and online replanning capabilities of model predictive control (MPC). This paper provides an answer by introducing a new framework called *Actor-Critic Model Predictive Control*. The key idea is to embed a differentiable MPC within an actor-critic RL framework. The proposed approach leverages the short-term predictive optimization capabilities of MPC with the exploratory and end-to-end training properties of RL. The resulting policy effectively manages both short-term decisions through the MPC-based actor and long-term prediction via the critic network, unifying the benefits of both model-based control and end-to-end learning. We validate our method in both simulation and the real world with a quadcopter platform across various high-level tasks. We show that the proposed architecture can achieve real-time control performance, learn complex behaviors via trial and error, and retain the predictive properties of the MPC to better handle out of distribution behaviour.**

SUPPLEMENTARY MATERIAL

Video of the experiments: `https://youtu.be/mQqm_vFo7e4`

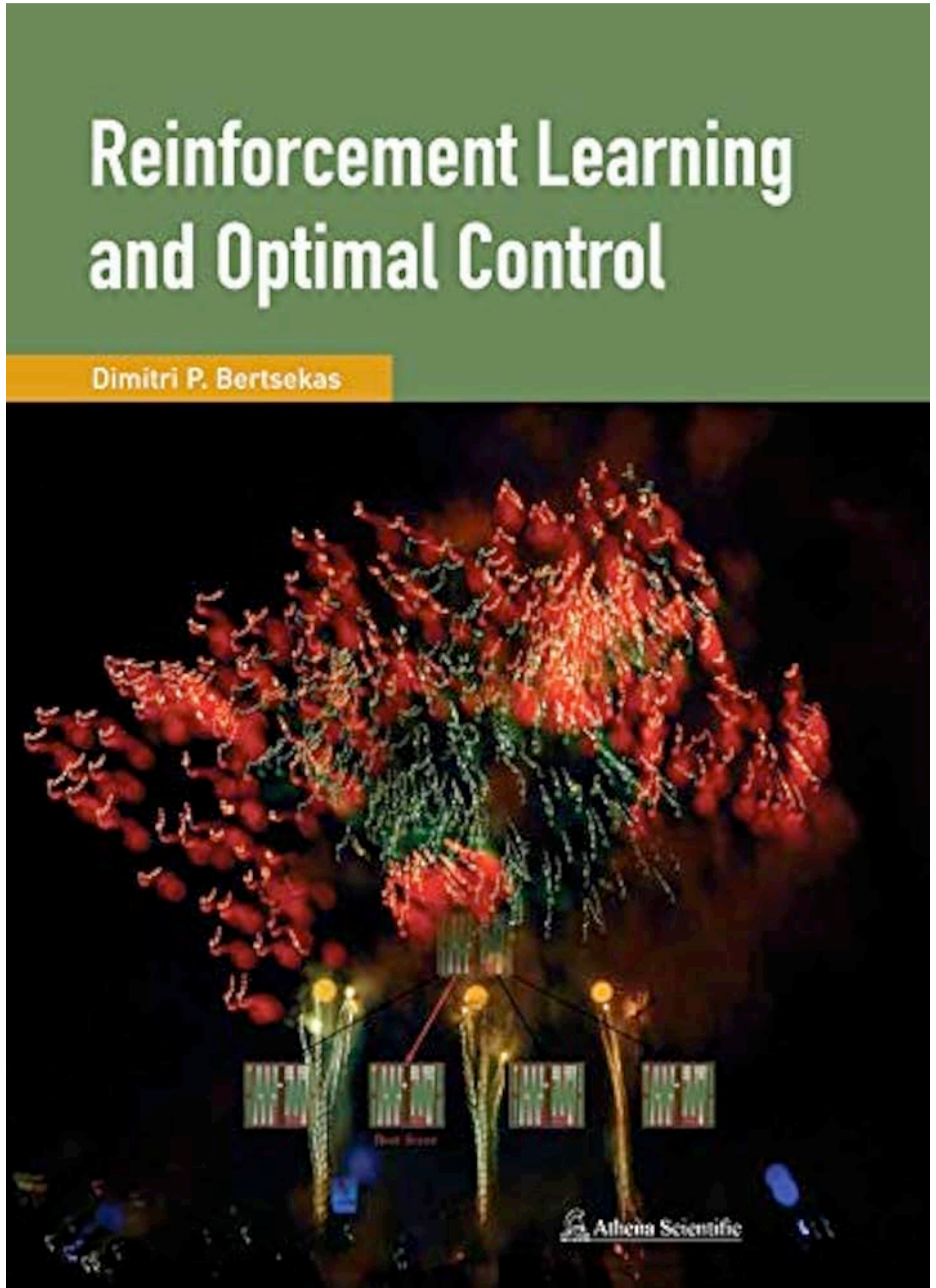
Fig. 1: A block diagram of the approach. We combine the strength of actor-critic RL and the robustness of MPC by placing a differentiable MPC as the last layer of the actor policy. At deployment time the commands for the environment are drawn from solving an MPC, which leverages the dynamics of the system and finds the optimal solution given the current state.

[10], [12]. Often, conservative assumptions about the task are made, leading to potentially sub-optimal task performance,

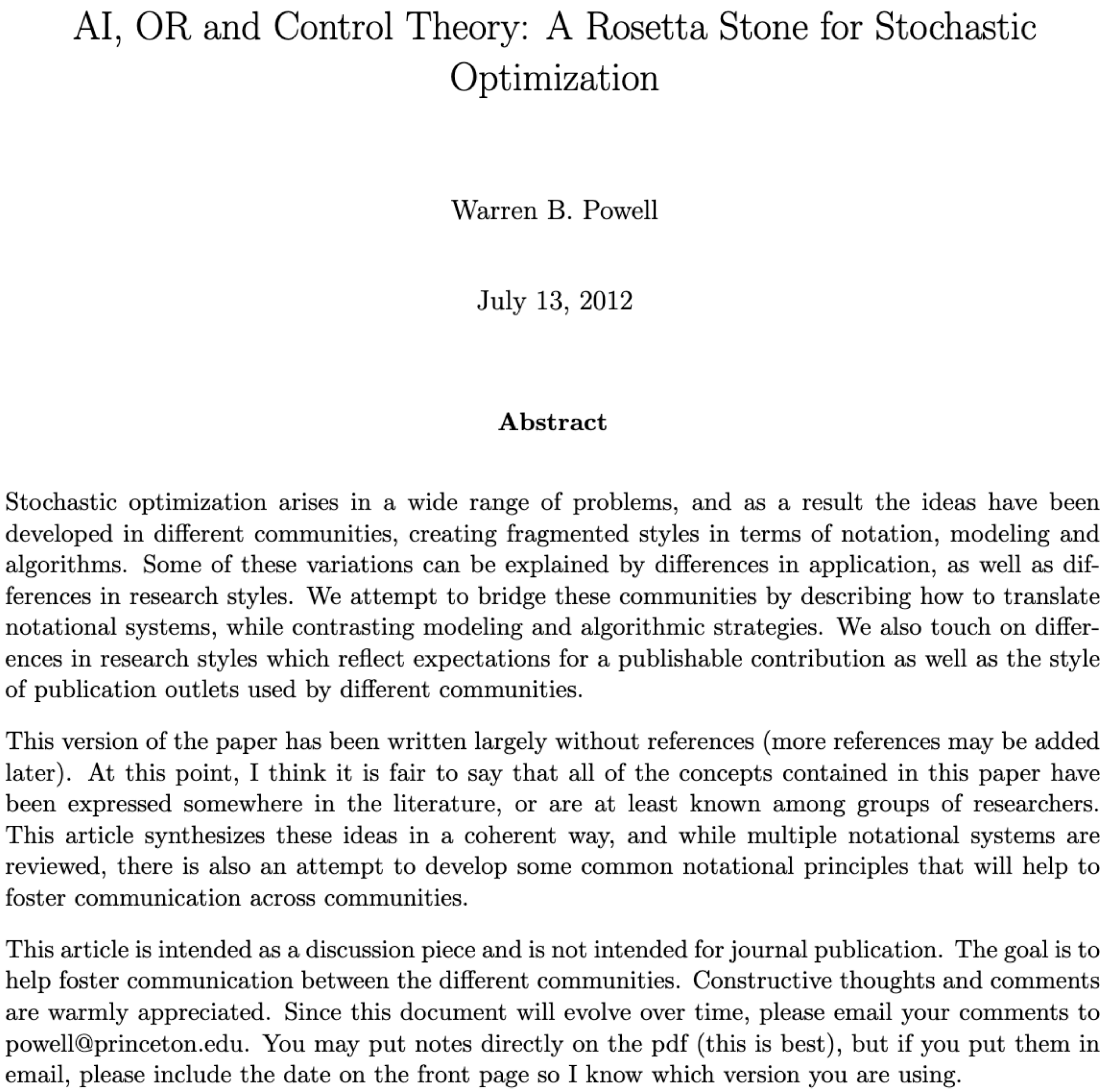
Bridging optimal control and RL

Optimal Control	Reinforcement Learning
• Cost function	• Reward function
• Dynamics model	• Transition function
• Controller	• Policy
• Cost-to-go (*minimization*)	• Value (*maximization*)

Bridging optimal control and RL



Bridging optimal control and RL



AI, OR and Control Theory: A Rosetta Stone for Stochastic Optimization

Warren B. Powell

July 13, 2012

Abstract

Stochastic optimization arises in a wide range of problems, and as a result the ideas have been developed in different communities, creating fragmented styles in terms of notation, modeling and algorithms. Some of these variations can be explained by differences in application, as well as differences in research styles. We attempt to bridge these communities by describing how to translate notational systems, while contrasting modeling and algorithmic strategies. We also touch on differences in research styles which reflect expectations for a publishable contribution as well as the style of publication outlets used by different communities.

This version of the paper has been written largely without references (more references may be added later). At this point, I think it is fair to say that all of the concepts contained in this paper have been expressed somewhere in the literature, or are at least known among groups of researchers. This article synthesizes these ideas in a coherent way, and while multiple notational systems are reviewed, there is also an attempt to develop some common notational principles that will help to foster communication across communities.

This article is intended as a discussion piece and is not intended for journal publication. The goal is to help foster communication between the different communities. Constructive thoughts and comments are warmly appreciated. Since this document will evolve over time, please email your comments to powell@princeton.edu. You may put notes directly on the pdf (this is best), but if you put them in email, please include the date on the front page so I know which version you are using.

https://castlelab.princeton.edu/Papers/AIOR_July2012.pdf

Takeaways

- Optimal control and RL are two sides of the same coin
- Both are modeling frameworks for solving sequential decision making problems under uncertainty
- RL classically has been considered “black box”, however approaches that interleave MPC and RL techniques are changing this perception
- Neither RL nor optimal control is “off-the-shelf” — performance of the controller/policy is sensitive to parameters and tuning

References

- R. Bellman, “Dynamic Programming,” *Princeton University Press*, 1957.
- R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” *MIT Press*, 1998.