Trajectory Design for Space Systems

EN.530.626 (Fall 2025)

Lecture 18

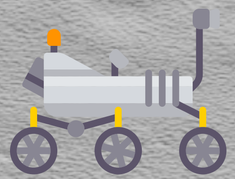
Instructor: Prof. Abhishek Cauligi
Course Assistant 1: Arnab Chatterjee
Course Assistant 2: Mark Gonzales

Motion planning

Optimal motion planning

- Last time: we saw that SBMP algorithms typically do not plan with the full-fidelity of the system constraints
  - Kinodynamic constraints, state and control constraints, etc.
- How can we plan for state and control trajectories that satisfy full system constraints?

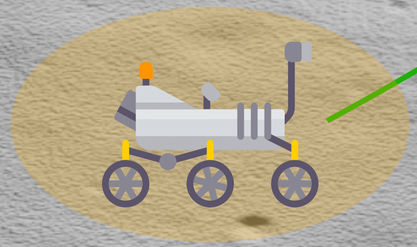
$$
\begin{aligned}
\underset{x_{0:N}, u_{0:N}, \delta_{0:N}}{\text{minimize}} \quad & \sum_{k=0}^{N} J(x_k, u_k, \delta_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k, \delta_k) \\
& g(x_k, u_k, \delta_k) \leq 0 \\
& x_0 = x_{\text{init}} \\
& x_N \in \mathcal{X}_{\text{goal}},
\end{aligned}
$$



Sense (Navigation)

Sense (Navigation)

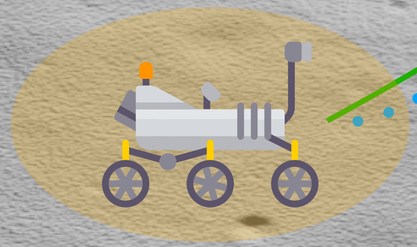
Plan (Guidance)



Sense (Navigation)

Plan (Guidance)

Act (Control)



Hierarchical planning approach

- Key idea: decompose optimization problem into hierarchical approach
- Regard the combinatorial decisions in the problem $\delta_{0:N}$ as the *complicating* variables
  - First: solve *global planning* consisting of combinatorial decisions
  - Second: solve *local planning* problem to compute continuous states and actions
  - Finally: use (unconstrained) feedback control to track trajectory

$$
\begin{aligned}
\underset{x_{0:N}, u_{0:N}, \delta_{0:N}}{\text{minimize}} \quad & \sum_{k=0}^{N} J(x_k, u_k, \delta_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k, \delta_k) \\
& g(x_k, u_k, \delta_k) \leq 0 \\
& x_0 = x_{\text{init}} \\
& x_N \in \mathcal{X}_{\text{goal}},
\end{aligned}
$$

Hierarchical planning approach

Global Planner

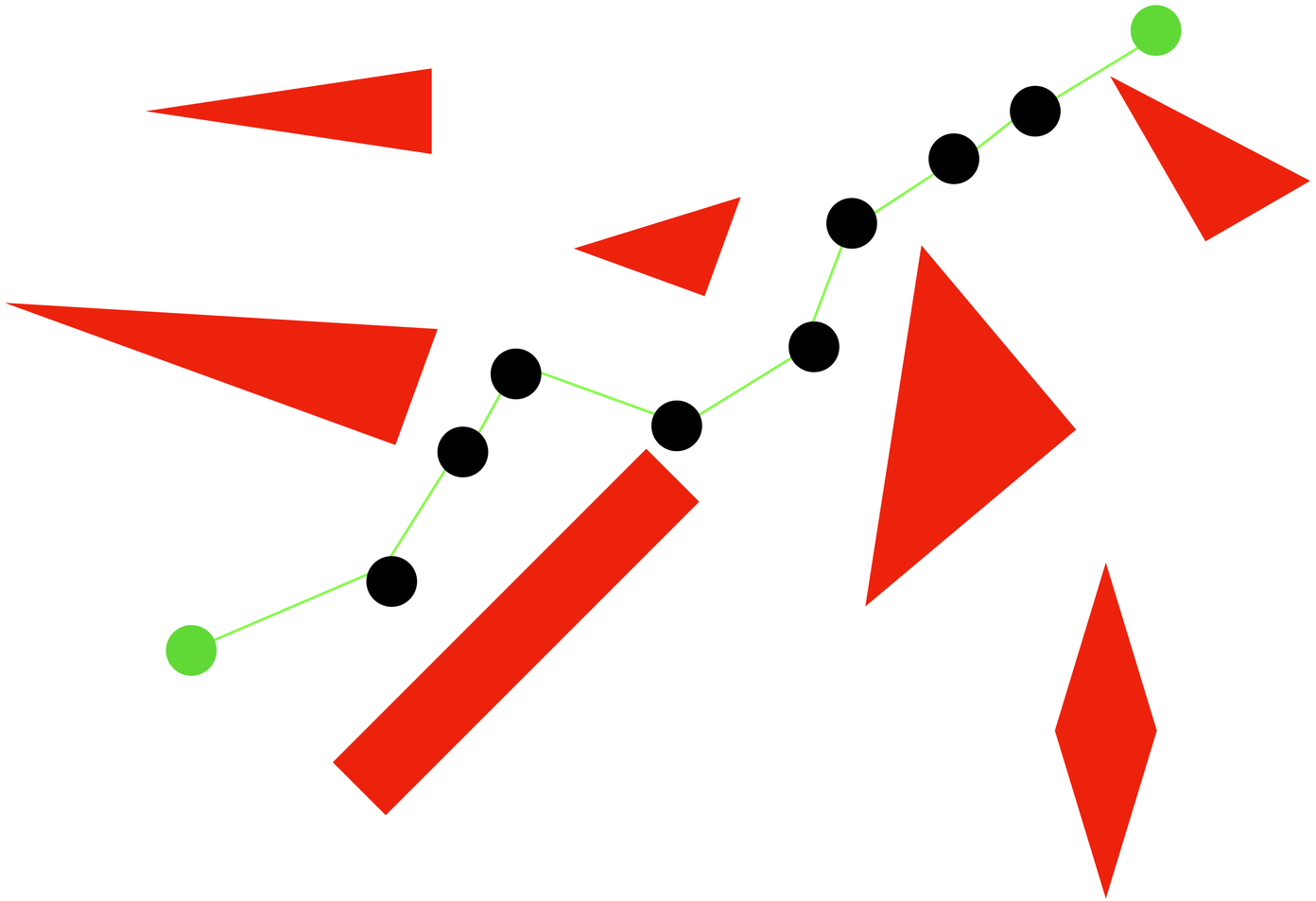
- The Global Planner typically reasons about the combinatorial decision-making aspect of the trajectory generation problem
- Succinctly, the motion planner determines the modes δ_k

$$
\begin{aligned}
\underset{x_{0:N}, u_{0:N}, \delta_{0:N}}{\text{minimize}} \quad & \sum_{k=0}^{N} J(x_k, u_k, \delta_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k, \delta_k) \\
& g(x_k, u_k, \delta_k) \leq 0 \\
& x_0 = x_{\text{init}} \\
& x_N \in \mathcal{X}_{\text{goal}},
\end{aligned}
\qquad \longrightarrow \qquad
\begin{aligned}
\underset{x_{0:N}, u_{0:N}}{\text{minimize}} \quad & \sum_{k=0}^{N} J(x_k, u_k, \delta_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k, \delta_k) \\
& g(x_k, u_k, \delta_k) \leq 0 \\
& x_0 = x_{\text{init}} \\
& x_N \in \mathcal{X}_{\text{goal}},
\end{aligned}
$$

Hierarchical planning approach

Global Planner

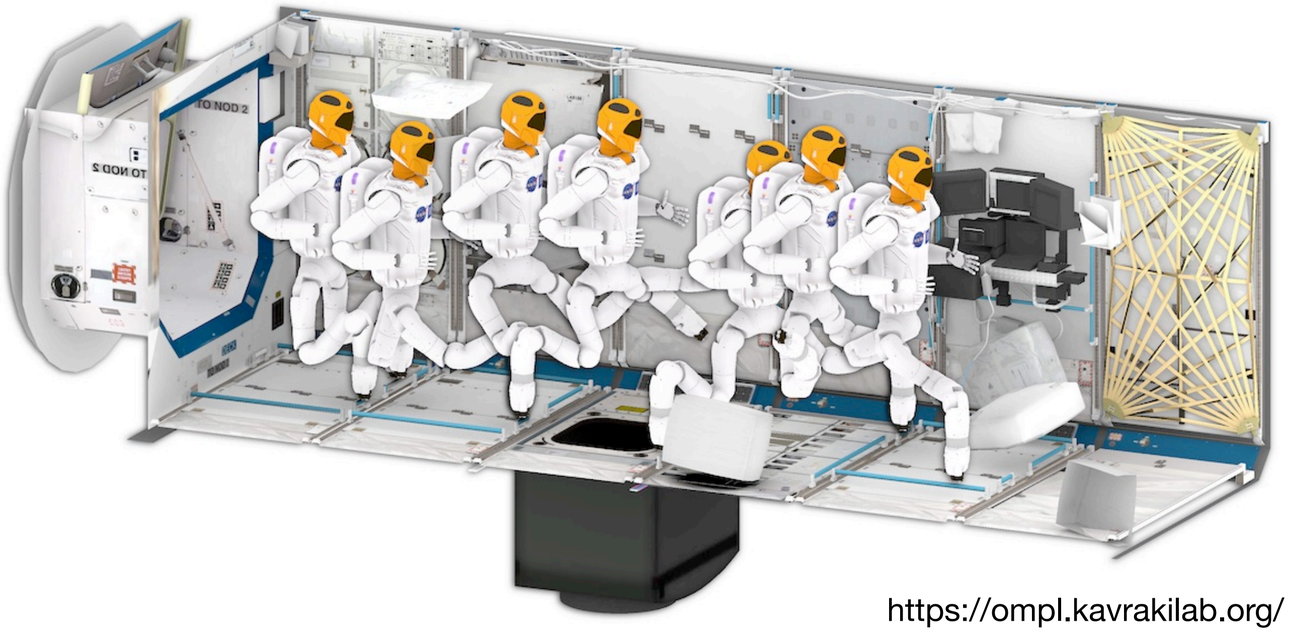
- Different approaches to solving the Global Planner problem:
  - Mixed-integer programming
  - Sampling-based planning
  - Derivative-free trajopt
  - Reinforcement learning



Hierarchical planning approach

Global Planner

- The OMPL library provides many implementations of sampling-based motion planning algorithms

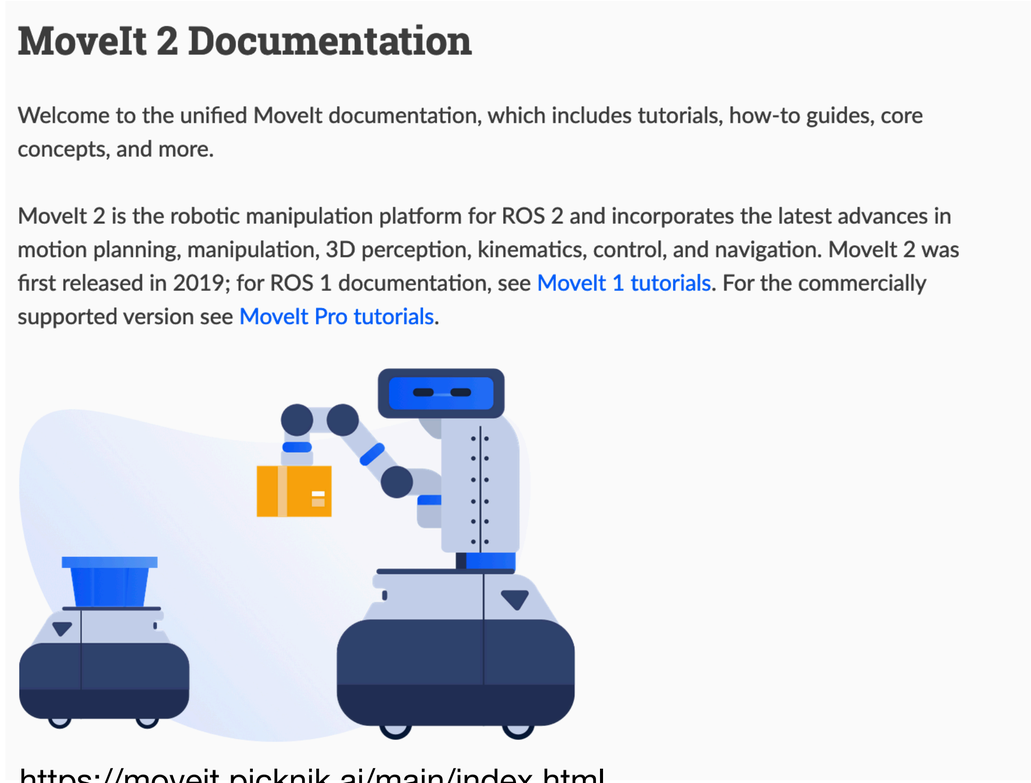


https://ompl.kavrakilab.org/

Hierarchical planning approach

Global Planner

- The MoveIt library provides a robotics interface to use OMPL with off-the-shelf collision checking and other utilities



https://moveit.picknik.ai/main/index.html

Hierarchical planning approach

Local Planner

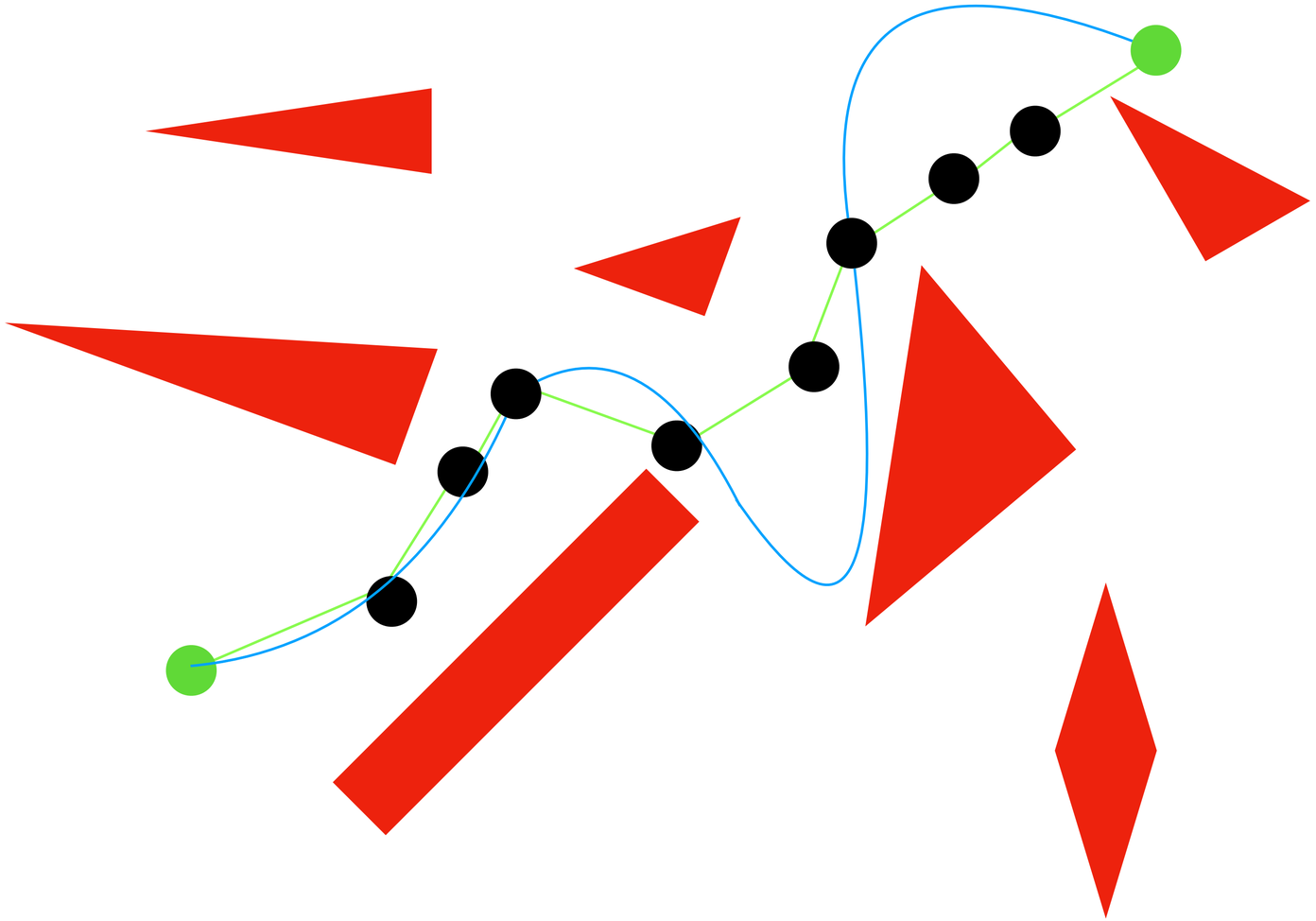
- The Local Planner solves for a dynamically feasible state and control trajectory using the Global Planner trajectory as the reference

$$
\begin{aligned}
\underset{x_{0:N}, u_{0:N}}{\text{minimize}} \quad & \sum_{k=0}^{N} J(x_k, u_k, \delta_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k, \delta_k) \\
& g(x_k, u_k, \delta_k) \leq 0 \\
& x_0 = x_{\text{init}} \\
& x_N \in \mathcal{X}_{\text{goal}},
\end{aligned}
\qquad \longrightarrow \qquad (x^*, u^*)
$$

Hierarchical planning approach

Local Planner

- Different approaches to solving the Local Planner problem:
  - Convex/non-convex trajectory optimization
  - Sampling-based MPC
  - Reinforcement learning



Hierarchical planning approach

Local Planner

- In the past decade, many embeddable convex solvers developed for solving convex problems for Local Planner applications

OSQP: an operator splitting solver for quadratic programs

Bartolomeo Stellato[1] · Goran Banjac[2] · Paul Goulart[3] · Alberto Bemporad[4] · Stephen Boyd[5]

Judo: A User-Friendly Open-Source Package for Sampling-Based Model Predictive Control

Albert H. Li*,†, Brandon Hung†, Aaron D. Ames*
Jiuguang Wang†, Simon Le Cleac'h†,‡, and Preston Culbertson†,‡

TinyMPC: Model-Predictive Control on Resource-Constrained Microcontrollers

Anoushka Alavilli*, Khai Nguyen*, Sam Schoedel*, Brian Plancher, Zachary Manchester

Clarabel: An interior-point solver for conic programs with quadratic objectives

Paul J. Goulart, Yuwen Chen

MPPI-Generic: A CUDA Library for Stochastic Trajectory Optimization

Bogdan Vlahov[1], Jason Gibson[1], Manan Gandhi[1], Evangelos A. Theodorou[1]

PROXQP: an Efficient and Versatile Quadratic Programming Solver for Real-Time Robotics Applications and Beyond

Antoine Bambade*,[1,2], Fabian Schramm[1], Sarah El-Kazdadi[1],
Stéphane Caron[1], Adrien Taylor[1] and Justin Carpentier[1]

Case study

Amazon robotics

Case study

Amazon robotics

- For mobile robots, sensory information updates with latency
- The Global Planner finds a path between the initial state to goal state
- The Local Planner generates a smooth trajectory along this GP path
- The controller corrects for small deviations away from the LP trajectory

The science behind Astro's graceful, responsive motion

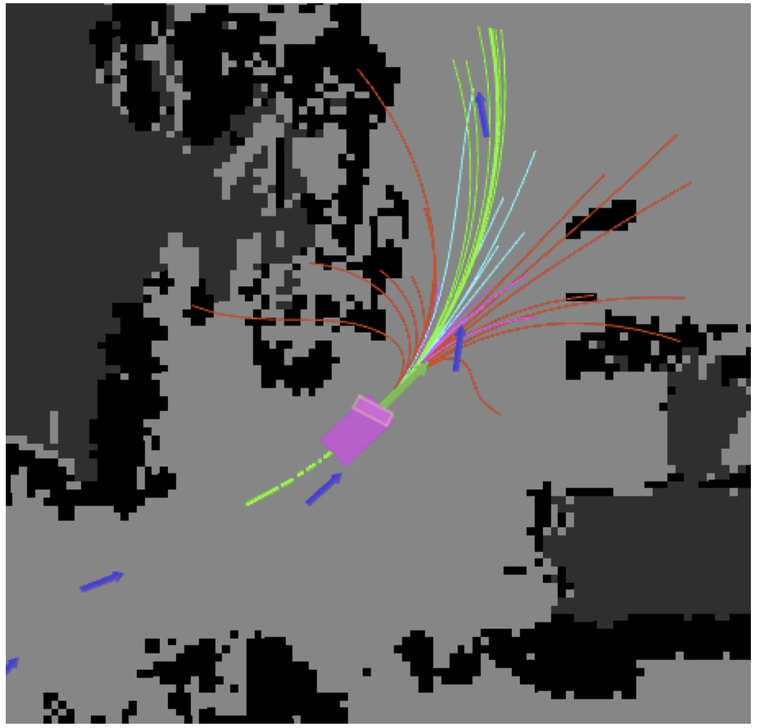
Predictive planning, uncertainty modeling, uniquely constrained trajectory optimization, and multiscale planning help customers trust Astro.

By Jong Jin Park

March 9, 2023

Share

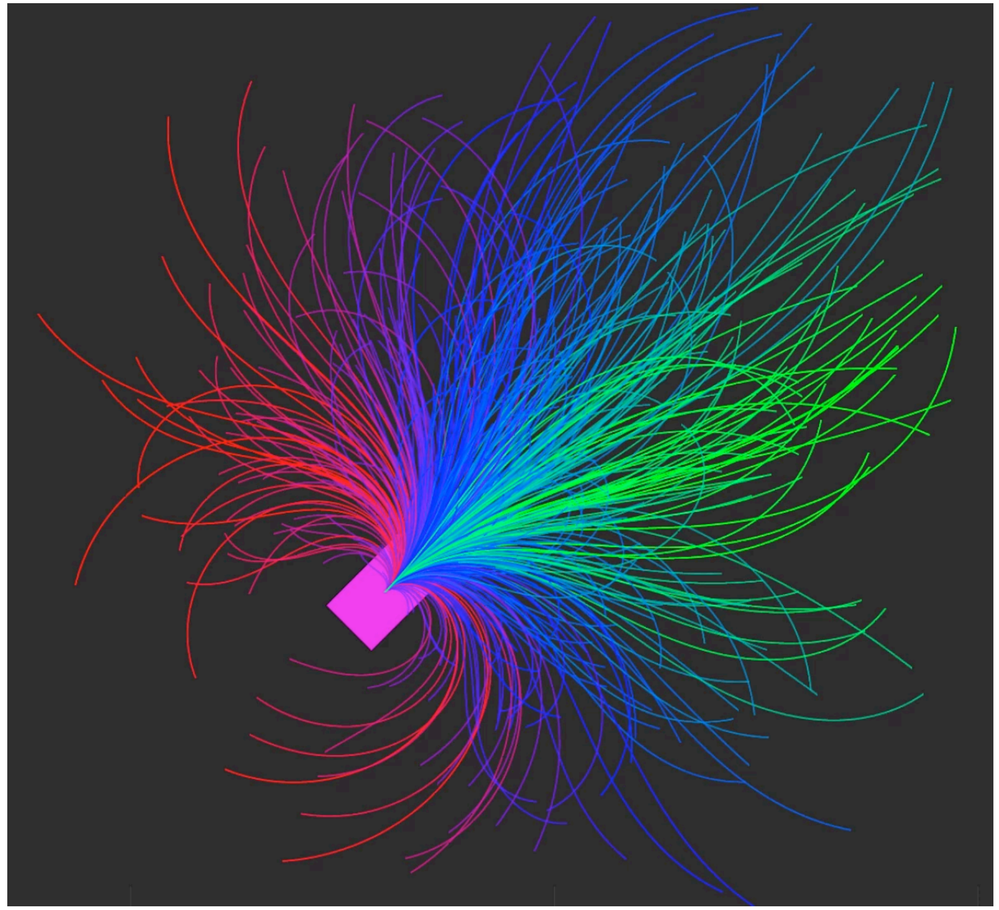
With Astro, we are building something that was a distant dream just a few years ago: a mobile robot that can move with grace and confidence, can interact with human users, and is available at a consumer-friendly price.



https://www.amazon.science/blog/the-science-behind-astros-graceful-responsive-motion

Case study
Amazon robotics

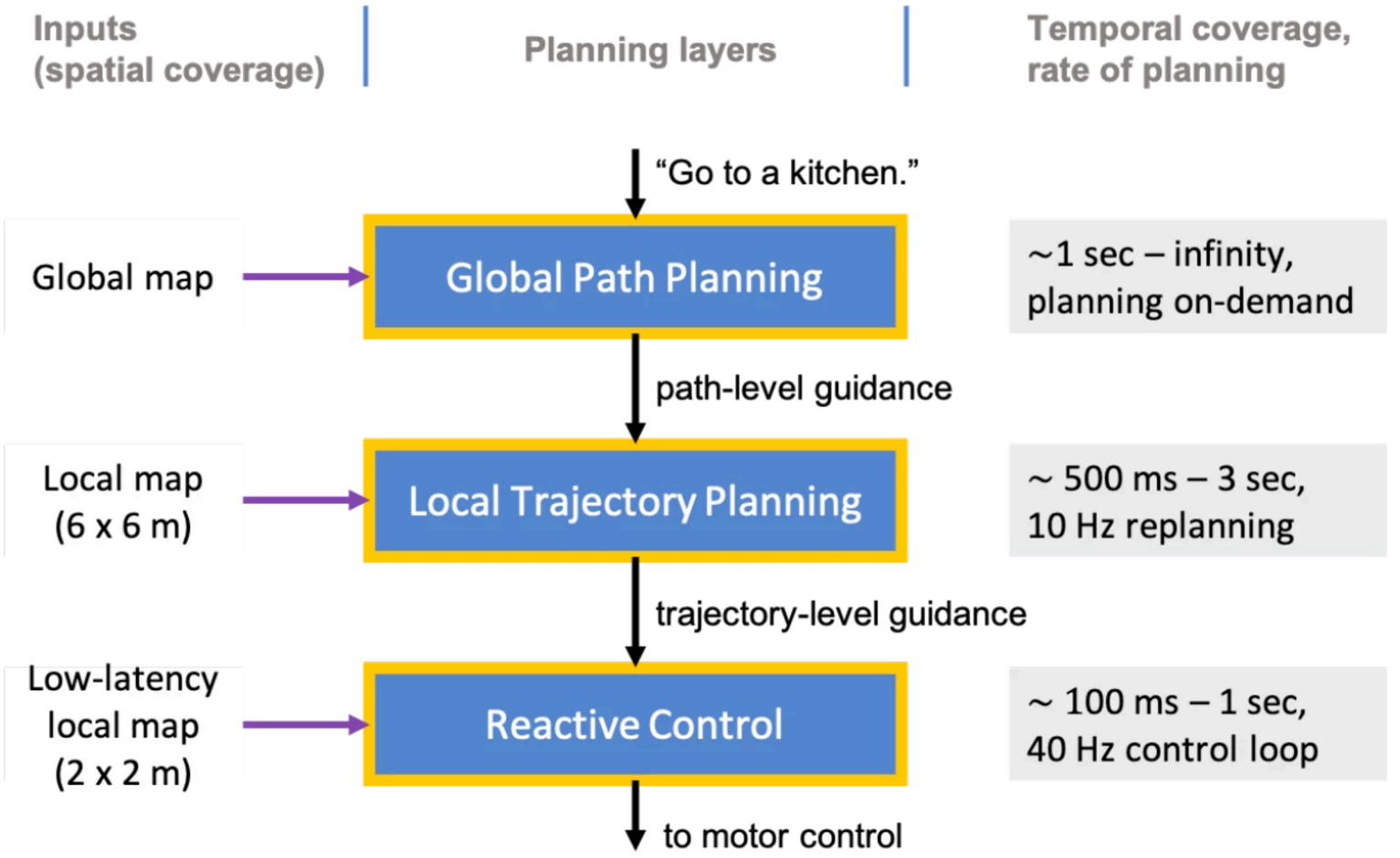
- Hierarchical planner with different layers of planning resolution, horizon, and fidelity
- Note that the GP and LP are not solved at the same frequency



https://www.amazon.science/blog/the-science-behind-astros-graceful-responsive-motion

Case study
Amazon robotics

- Hierarchical planner with different layers of planning resolution, horizon, and fidelity
- Note that the GP and LP are not solved at the same frequency

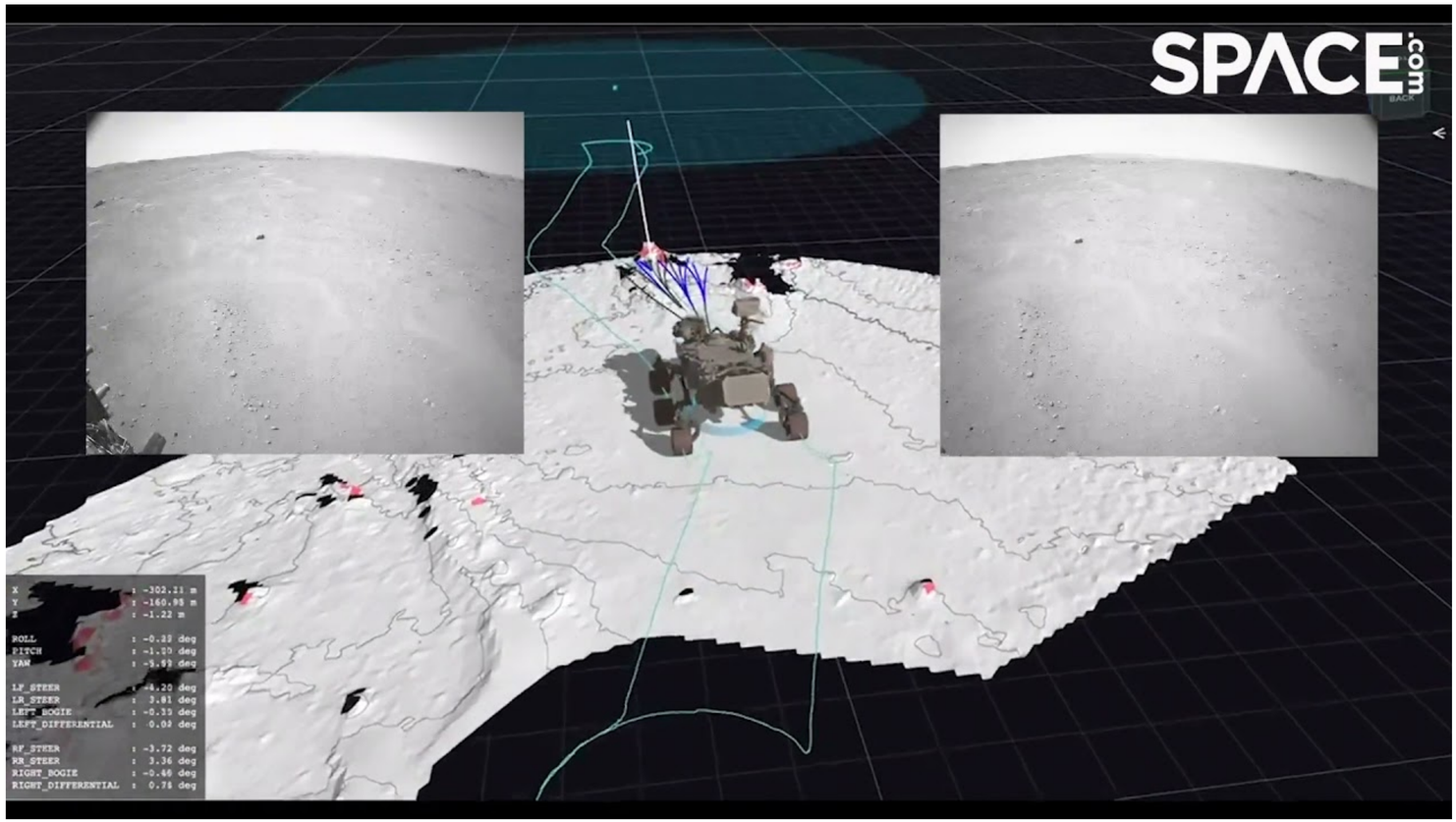


https://www.amazon.science/blog/the-science-behind-astros-graceful-responsive-motion

Case study

Mars rover planning

- Mars rovers travel ~4cm/s
- Global planner evaluates arcs and chooses *best* collision free ones
- Path is tracked using a feedback controller

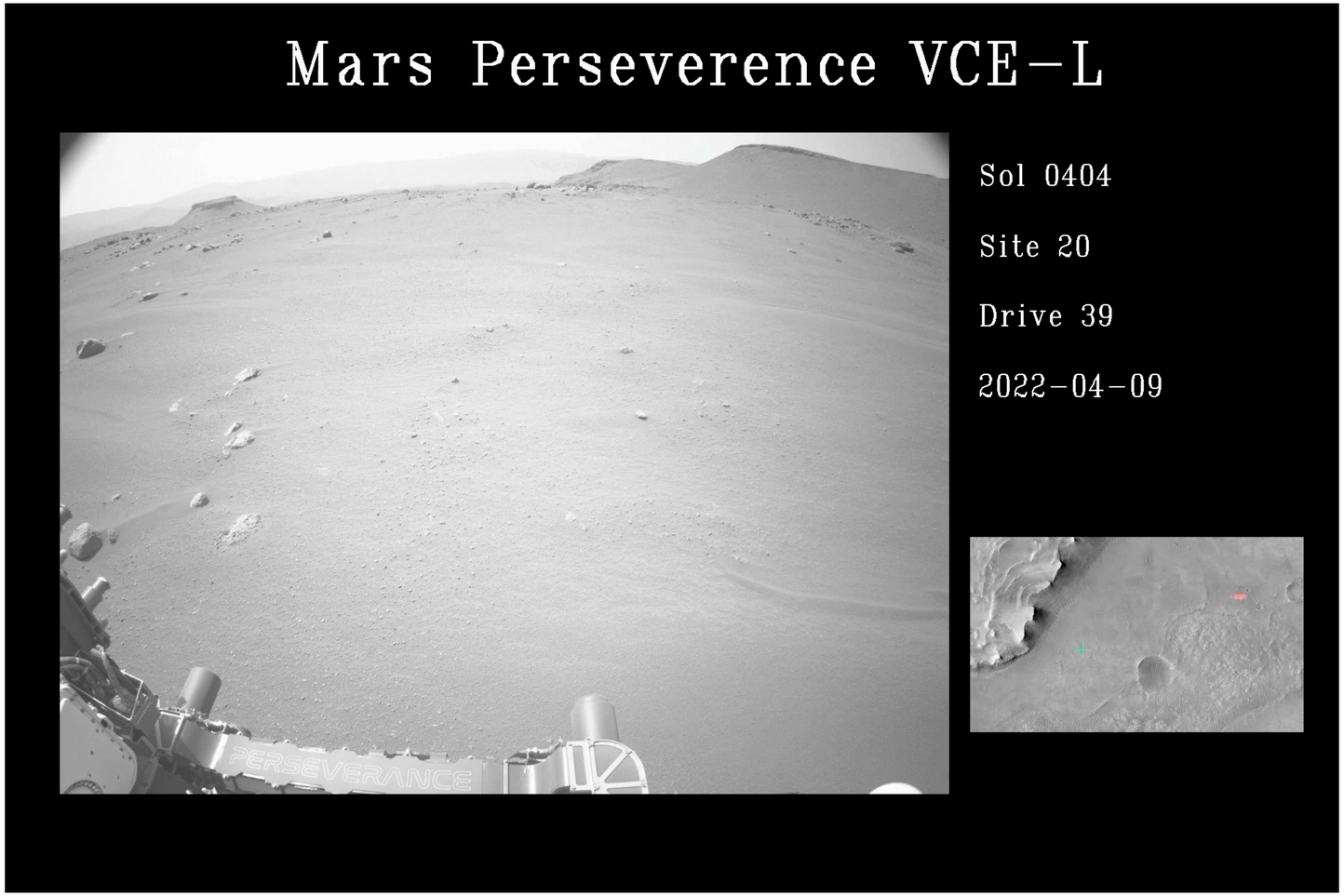


O. Toupet, T. Del Sesto, et al., “A ROS-based Simulator for Testing the Enhanced Autonomous Navigation of the Mars 2020 Rover,” in *IEEE Aerospace Conference*, 2020.

Case study

Mars rover planning

- Mars rovers travel ~4cm/s
- Global planner evaluates arcs and chooses *best* collision free ones
- Path is tracked using a feedback controller

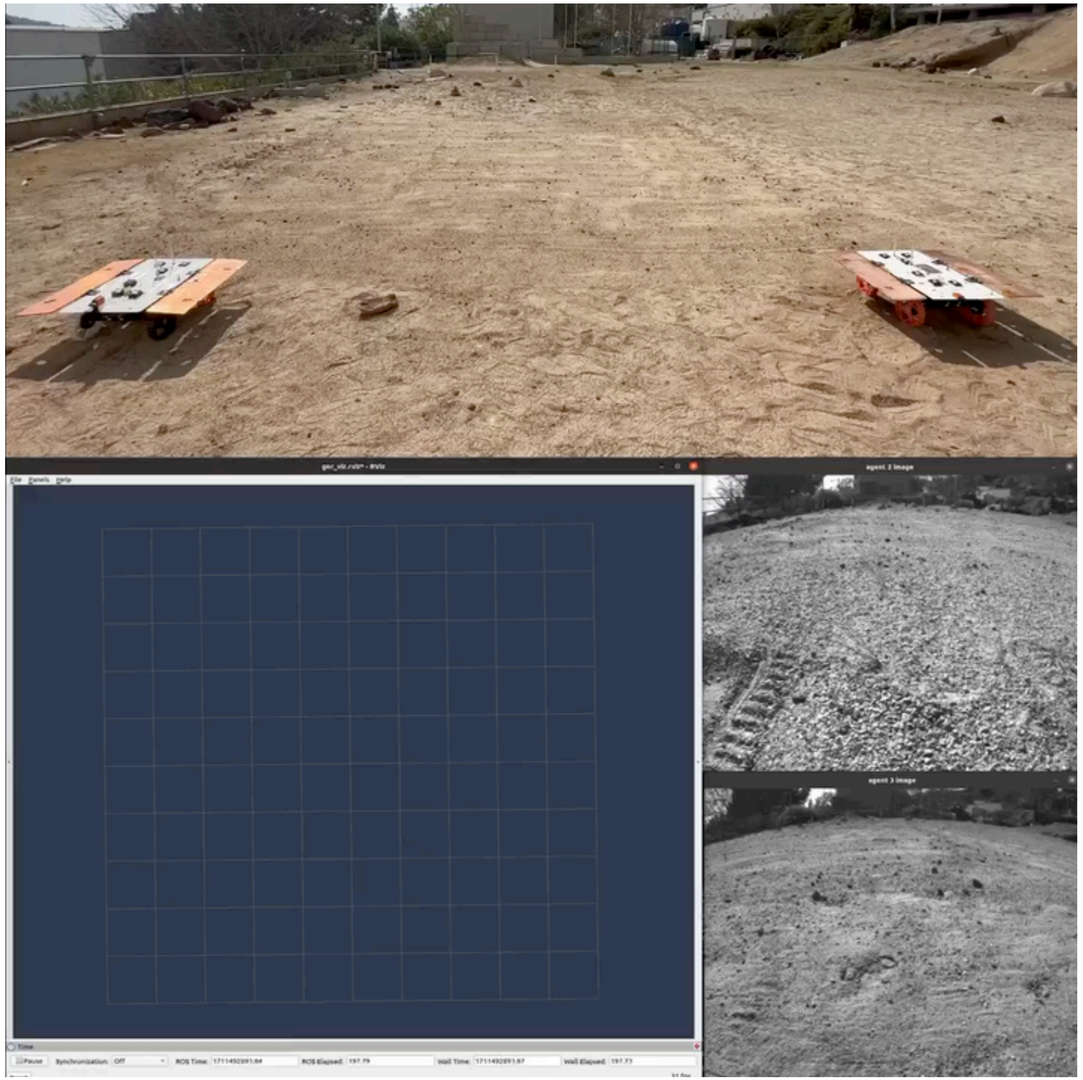


O. Toupet, T. Del Sesto, et al., “A ROS-based Simulator for Testing the Enhanced Autonomous Navigation of the Mars 2020 Rover,” in *IEEE Aerospace Conference*, 2020.

Case study

Lunar rover planning

- The proposed CADRE Lunar rover mission uses a three-stage planning approach
- The first layer uses SBMP to plan multi-agent paths for three Lunar rovers
- The Global and Local Planners for each agent use on-board trajectory optimization to plan at the single-agent level
- A low-level PD controller tracks wheel velocity commands to follow Local Planner trajectory

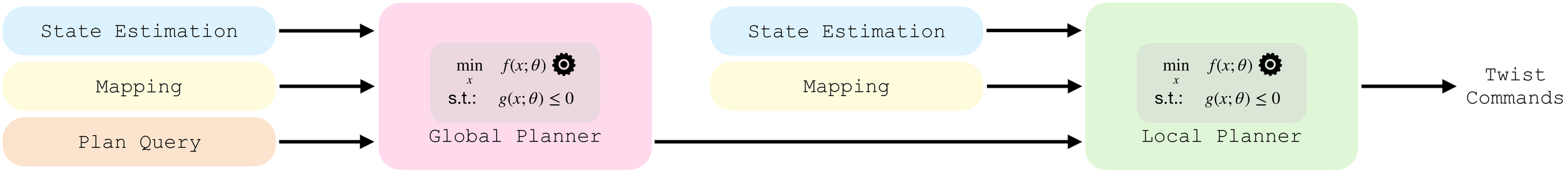


S. Nayak, G. Lim, F. Rossi, M. Otte, and J.-P. de la Croix, "Multi-Robot Exploration for the CADRE Mission," *Autonomous Robots*, 2025.

A. Cauligi*, K. Albee*, et al., "CRESCENT: Collision-Free Highly-Constrained Trajectory Optimization for Driving on the Moon," under review, 2025.

Case study

Lunar rover planning



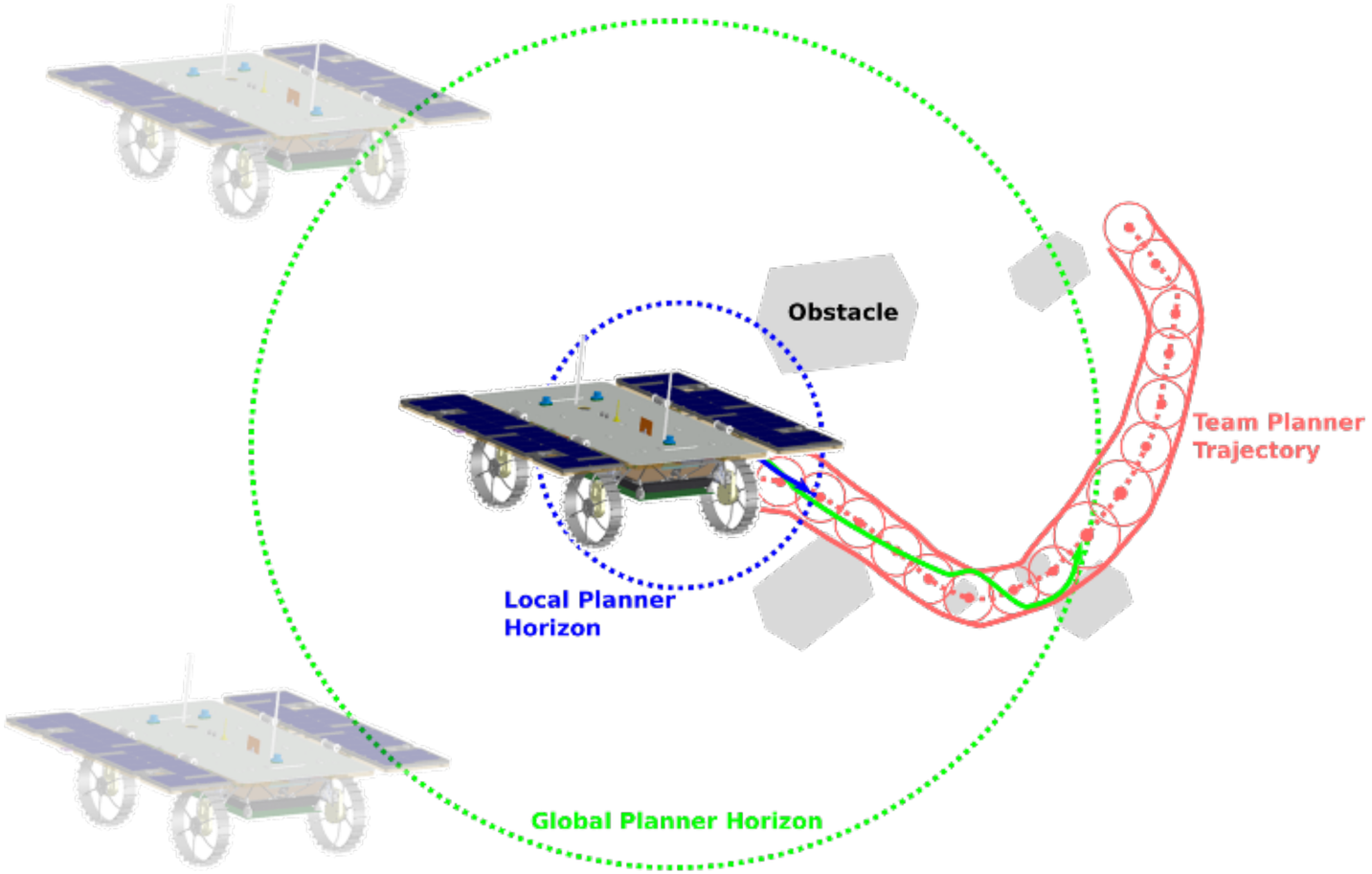
A. Cauligi*, K. Albee*, et al., “CRESCENT: Collision-Free Highly-Constrained Trajectory Optimization for Driving on the Moon,” under review, 2025.

Case study

Lunar rover planning

Key differences between the planners

- Treatment of unknown vs. known space
- Planning horizon
- Awareness of science constraints

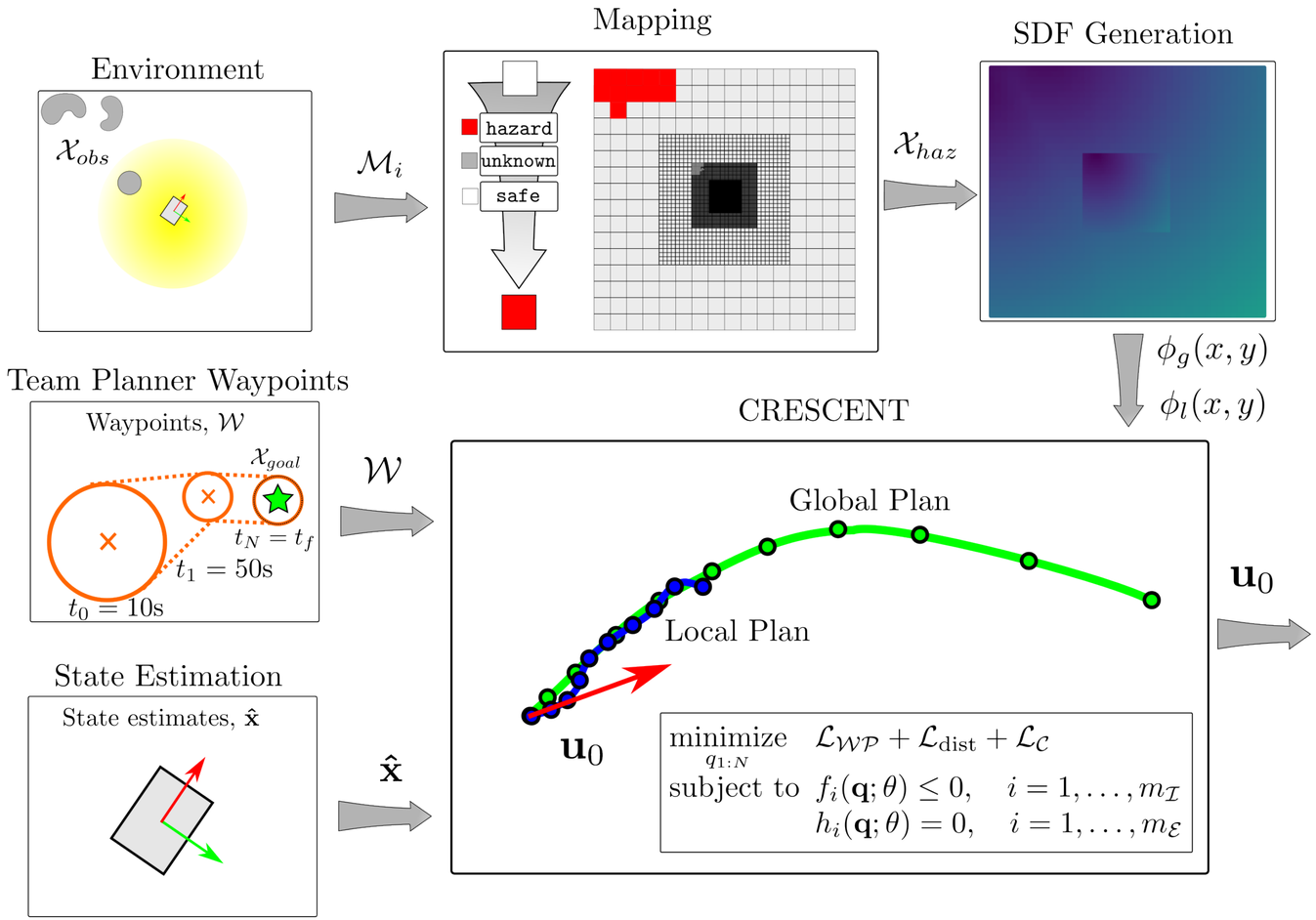


A. Cauligi*, K. Albee*, et al., “CRESCENT: Collision-Free Highly-Constrained Trajectory Optimization for Driving on the Moon,” under review, 2025.

Case study
Lunar rover planning

Key differences between the planners

- Treatment of unknown vs. known space
- Planning horizon
- Awareness of science constraints



A. Cauligi*, K. Albee*, et al., “CRESCENT: Collision-Free Highly-Constrained Trajectory Optimization for Driving on the Moon,” under review, 2025.

Hierarchical planning approach

Downsides

- Decomposing the original problem into two sequential ones might yield infeasible solutions where a feasible one exists for the original one
- Ensuring safety constraint satisfaction between the hand-off of both is challenging due to model mismatch between the planning layers

Hierarchical planning approach

Research directions

IEEE CSS IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 66, NO. 12, DECEMBER 2021 5861

FaSTrack: A Modular Framework for Real-Time Motion Planning and Guaranteed Safe Tracking

Mo Chen, Sylvia L. Herbert, Haimin Hu, *Graduate Student Member, IEEE*, Ye Pu, Jaime Fernández Fisac, *Member, IEEE*, Somil Bansal, *Member, IEEE*, SooJean Han, and Claire J. Tomlin, *Fellow, IEEE*

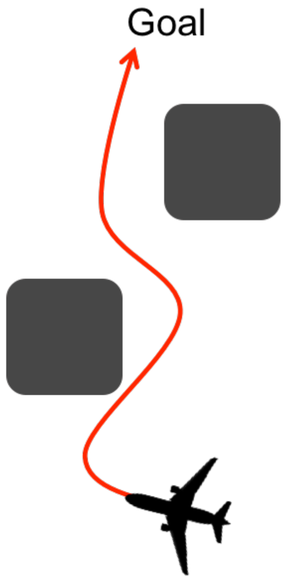
***Abstract*—Real-time, guaranteed safe trajectory planning is vital for navigation in unknown environments. However, real-time navigation algorithms typically sacrifice robustness for computation speed. Alternatively, provably safe trajectory planning tends to be too computationally intensive for real-time replanning. We propose FaSTrack, Fast and Safe Tracking, a framework that achieves both real-time replanning and guaranteed safety. In this framework, real-time computation is achieved by allowing any trajectory planner to use a simplified *planning model* of the system. The plan is tracked by the system, represented by a more realistic, higher dimensional *tracking model*. We precompute the tracking error bound (TEB) due to mismatch between the two models and due to external disturbances. We also obtain the corresponding tracking controller used to stay within the TEB. The precomputation does not require prior knowledge of the environment. We demonstrate FaSTrack using Hamilton–Jacobi reachability for precomputation and three different real-time trajectory planners with three different tracking-planning model pairs.**

***Index Terms*—Intelligent systems, nonlinear control systems, optimal control, real-time-systems, safety.**

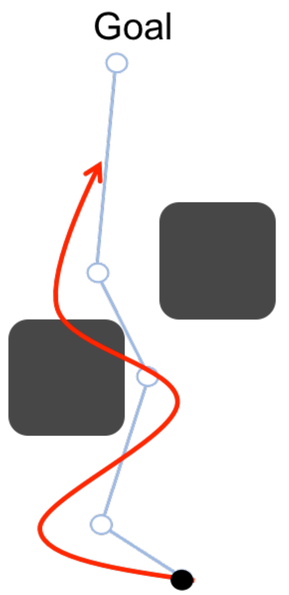
I. Introduction

ACHIEVING safe navigation in real time is difficult for many common dynamical systems due to the computational complexity of generating and formally verifying the safety of dynamically feasible trajectories. To achieve real-time planning, many algorithms use highly simplified model dynamics or kinematics to create a nominal trajectory that is then tracked by the system using a feedback controller, such as a linear quadratic regulator (LQR). These nominal trajectories may not be dynamically feasible for the true autonomous system, resulting in a tracking error between the planned path and the executed trajectory. This concept is illustrated in Fig. 1. Additionally, external disturbances (e.g., wind) can be difficult to account for

Slow and Accurate Planning (Hamilton-Jacobi Reachability):



Fast and Inaccurate Planning (Simple Path or Trajectory Planners):



https://bair.berkeley.edu/blog/2017/12/05/fastrack/

Hierarchical planning approach

Research directions

Robust Tracking with Model Mismatch for Fast and Safe Planning: an SOS Optimization Approach

Sumeet Singh[1], Mo Chen[1], Sylvia L. Herbert[2], Claire J. Tomlin[2], and Marco Pavone[1⋆]

[1] Dept. of Aeronautics and Astronautics, Stanford University
{ssingh19, mochen72, pavone}@stanford.edu
[2] Dept. of Electrical Engineering and Computer Science, University of California, Berkeley
{sylvia.herbert, tomlin}@berkeley.edu

Abstract. In the pursuit of real-time motion planning, a commonly adopted practice is to compute trajectories by running a planning algorithm on a simplified, low-dimensional dynamical model, and then employ a feedback tracking controller that tracks such a trajectory by accounting for the full, high-dimensional system dynamics. While this strategy of *planning with model mismatch* generally yields fast computation times, there are no guarantees of dynamic feasibility, which hampers application to safety-critical systems. Building upon recent work that addressed this problem through the lens of Hamilton-Jacobi (HJ) reachability, we devise an algorithmic framework whereby one computes, offline, for a pair of "planner" (i.e., low-dimensional) and "tracking" (i.e., high-dimensional) models, a feedback tracking controller and associated tracking bound. This bound is then used as a safety margin when generating motion plans via the low-dimensional model. Specifically, we harness the computational tool of sum-of-squares (SOS) programming to design a bilinear optimization algorithm for the computation of the feedback tracking controller and associated tracking bound. The algorithm is demonstrated via numerical experiments, with an emphasis on investigating the trade-off between the increased computational scalability afforded by SOS and its intrinsic conservativeness. Collectively, our results enable scaling the appealing strategy of planning with model mismatch to systems that are beyond the reach of HJ analysis, while maintaining safety guarantees.

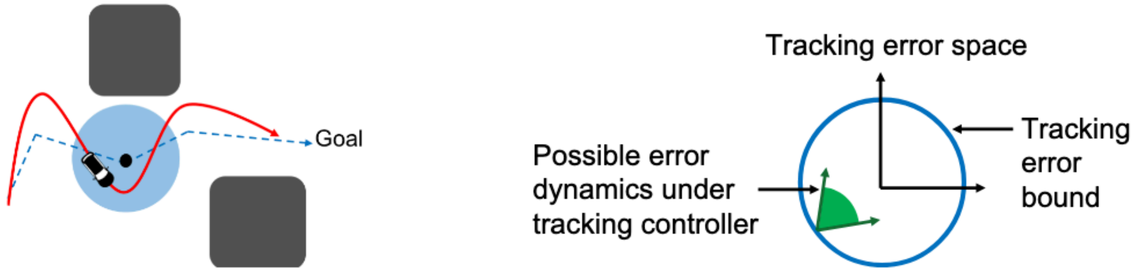


Fig. 1: *Left: Planning with model mismatch.* The tracking system (car, red trajectory) follows a motion plan computed via a low-fidelity dynamical model (blue trajectory). Tracking with model mismatch while maintaining safety requires keeping the maximum tracking error bound (TEB) below a certifiable value. *Right: TEB certification.* The TEB is characterized by the property that on its boundary, all possible error dynamics resulting from the nominal trajectory point inwards under some tracking controller, so that the TEB is never exceeded.

Tracking system	Tracking model	Planning system	Planning model	Transformation ϕ	Relative system dynamics	Suggested bounded state mapping $c(r)$
4D car (x, y) – position θ – heading v – speed u_ω – turn rate control u_a – accel. control	$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{v}\end{bmatrix} = \begin{bmatrix}v\cos\theta\\ v\sin\theta\\ u_\omega\\ u_a\end{bmatrix}$	**2D single integrator** $(\hat{x}, \hat{y})$ – position $(\hat{u}_{\hat{x}}, \hat{u}_{\hat{y}})$ – speed control	$\begin{bmatrix}\dot{\hat{x}}\\ \dot{\hat{y}}\end{bmatrix} = \begin{bmatrix}\hat{u}_{\hat{x}}\\ \hat{u}_{\hat{y}}\end{bmatrix}$	$\begin{bmatrix}R(\theta) & \mathbf{0}_{2\times2}\\ \mathbf{0}_{1\times2} & [0\ 1]\end{bmatrix}$	$\begin{bmatrix}\dot{x}_r\\ \dot{y}_r\\ \dot{v}\end{bmatrix} = \begin{bmatrix}v + u_\omega y_r - \tilde{u}_{\hat{x}}\\ -u_\omega x_r - \tilde{u}_{\hat{y}}\\ u_a\end{bmatrix}$, $\begin{bmatrix}\tilde{u}_{\hat{x}}\\ \tilde{u}_{\hat{y}}\end{bmatrix} = R(\theta)\begin{bmatrix}\hat{u}_{\hat{x}}\\ \hat{u}_{\hat{y}}\end{bmatrix}$	$\mathbf{I}_3$
5D car (x, y) – position θ – heading v – speed ω – turn rate u_a – accel. control u_α – ang. accel. control	$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}v\cos\theta\\ v\sin\theta\\ \omega\\ u_a\\ u_\alpha\end{bmatrix}$	**3D Dubins car** $(\hat{x}, \hat{y})$ – position $\hat{\theta}$ – heading $\hat{v}$ – constant speed $\hat{u}_\omega$ – turn rate control	$\begin{bmatrix}\dot{\hat{x}}\\ \dot{\hat{y}}\\ \dot{\hat{\theta}}\end{bmatrix} = \begin{bmatrix}\hat{v}\cos\hat{\theta}\\ \hat{v}\sin\hat{\theta}\\ \hat{u}_{\hat{\omega}}\end{bmatrix}$	$\begin{bmatrix}R(\hat{\theta}) & \mathbf{0}_{2\times3}\\ \mathbf{0}_{3\times2} & \mathbf{I}_3\end{bmatrix}$	$\begin{bmatrix}\dot{x}_r\\ \dot{y}_r\\ \dot{\theta}_r\\ \dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}-\hat{v} + v\cos\theta_r + \hat{u}_{\hat{\omega}} y_r\\ v\sin\theta_r - \hat{u}_{\hat{\omega}} x_r\\ \omega - \hat{u}_{\hat{\omega}}\\ u_a\\ u_\alpha\end{bmatrix}$	$\begin{bmatrix}x_r\\ y_r\\ \theta_r\\ v\cos\theta_r - \hat{v}\\ v\sin\theta_r\\ \omega\end{bmatrix}$
6D planar quadrotor (x, z) – position (v_x, v_z) – velocity θ – pitch ω – pitch rate u_T – thrust control u_τ – ang. accel. control	$\begin{bmatrix}\dot{x}\\ \dot{z}\\ \dot{v}_x\\ \dot{v}_z\\ \dot{\theta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}v_x\\ v_z\\ -u_T\sin\theta\\ u_T\cos\theta - g\\ \omega\\ u_\tau\end{bmatrix}$	**4D double integrator** $(\hat{x}, \hat{z})$ – position $(\hat{v}_x, \hat{v}_z)$ – velocity $(\hat{u}_{\hat{x}}, \hat{u}_{\hat{y}})$ – accel. control	$\begin{bmatrix}\dot{\hat{x}}\\ \dot{\hat{z}}\\ \dot{\hat{v}}_x\\ \dot{\hat{v}}_z\end{bmatrix} = \begin{bmatrix}\hat{v}_x\\ \hat{v}_z\\ \hat{u}_{\hat{x}}\\ \hat{u}_{\hat{z}}\end{bmatrix}$	$\mathbf{I}_6$	$\begin{bmatrix}\dot{x}_r\\ \dot{z}_r\\ \dot{v}_{x,r}\\ \dot{v}_{z,r}\\ \dot{\theta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}v_{x,r}\\ v_{z,r}\\ -u_T\sin\theta - \hat{u}_{\hat{x}}\\ u_T\cos\theta - g - \hat{u}_{\hat{y}}\\ \omega\\ u_\tau\end{bmatrix}$	$\mathbf{I}_6$

Receding horizon planning

- Most problems are not static
- As the system evolves over time, new information about the environment reveals itself or disturbances cause the system to deviate
- This leads to receding horizon planning

Receding horizon planning

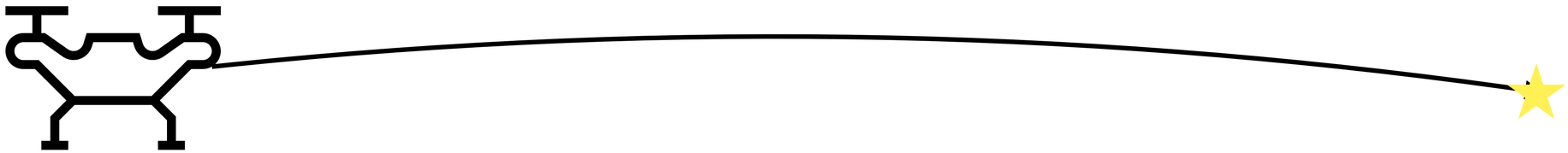
Model predictive control

- Model predictive control solves an optimal control problem with a receding horizon

    1. Solve optimal control problem $\mathscr{P}(\theta)$ to compute $\{u_0^*, u_1^*, \ldots, u_N^*\}$
    2. Apply the first control value u_0^*
    3. Propagate the system forward and update parameters θ

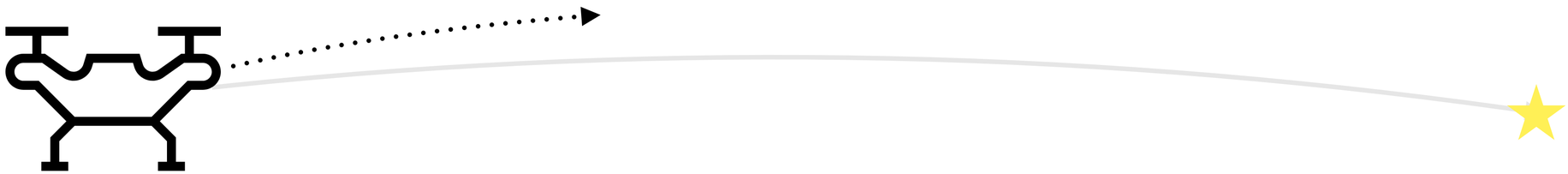
Receding horizon planning

Model predictive control



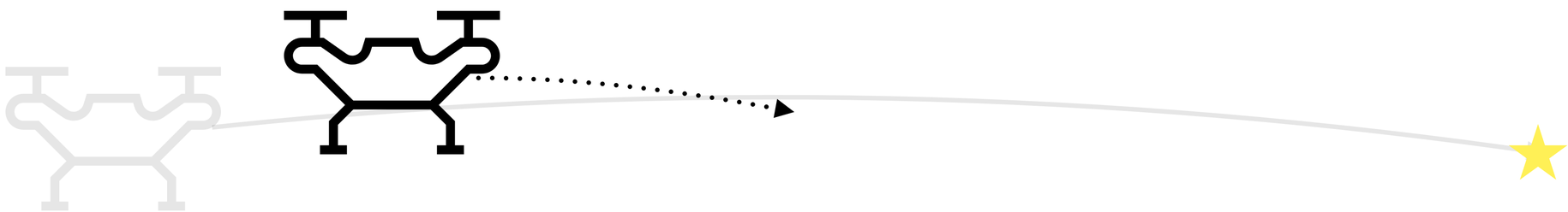
Receding horizon planning

Model predictive control



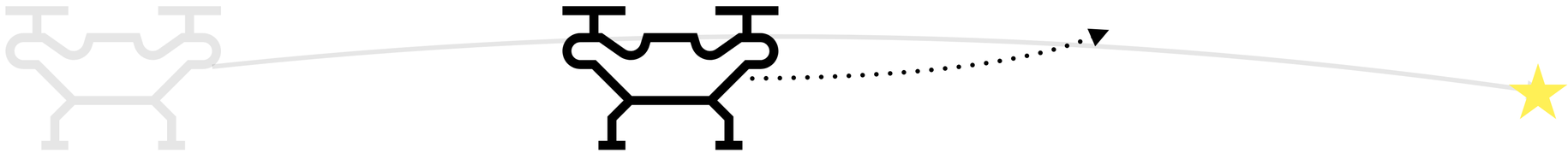
Receding horizon planning

Model predictive control



Receding horizon planning

Model predictive control



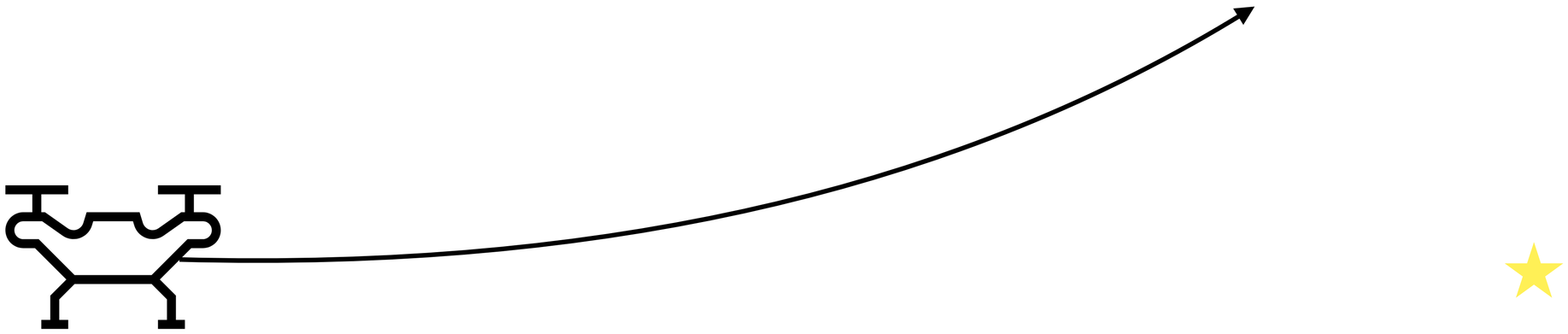
Underlying idea: solve a sequence of short horizon optimal control problems

Receding horizon planning

Model predictive control

- Myopic planning
  - If *N* is too short, then can lead to instability

$$
\begin{aligned}
\underset{x_{0:N}, u_{0:N}, \delta_{0:N}}{\text{minimize}} \quad & J_N(x_N) + \sum_{k=0}^{N-1} J(x_k, u_k, \delta_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k, \delta_k) \\
& g(x_k, u_k, \delta_k) \leq 0 \\
& x_0 = x_{\text{init}} \\
& x_N \in \mathcal{X}_{\text{goal}},
\end{aligned}
$$

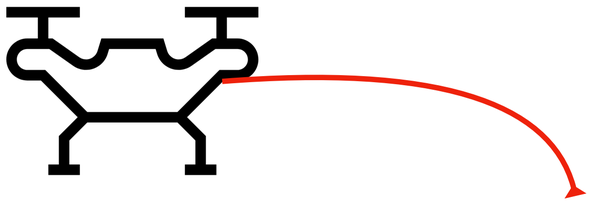


Receding horizon planning

Model predictive control

- Myopic planning
  - If *N* is too short, then can lead to instability
- Maintaining feasibility
  - If $\mathcal{X}_{\text{goal}}$ is too restrictive, then infeasibility becomes an issue

$$
\begin{aligned}
\underset{x_{0:N}, u_{0:N}, \delta_{0:N}}{\text{minimize}} \quad & J_N(x_N) + \sum_{k=0}^{N-1} J(x_k, u_k, \delta_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k, \delta_k) \\
& g(x_k, u_k, \delta_k) \leq 0 \\
& x_0 = x_{\text{init}} \\
& x_N \in \mathcal{X}_{\text{goal}},
\end{aligned}
$$



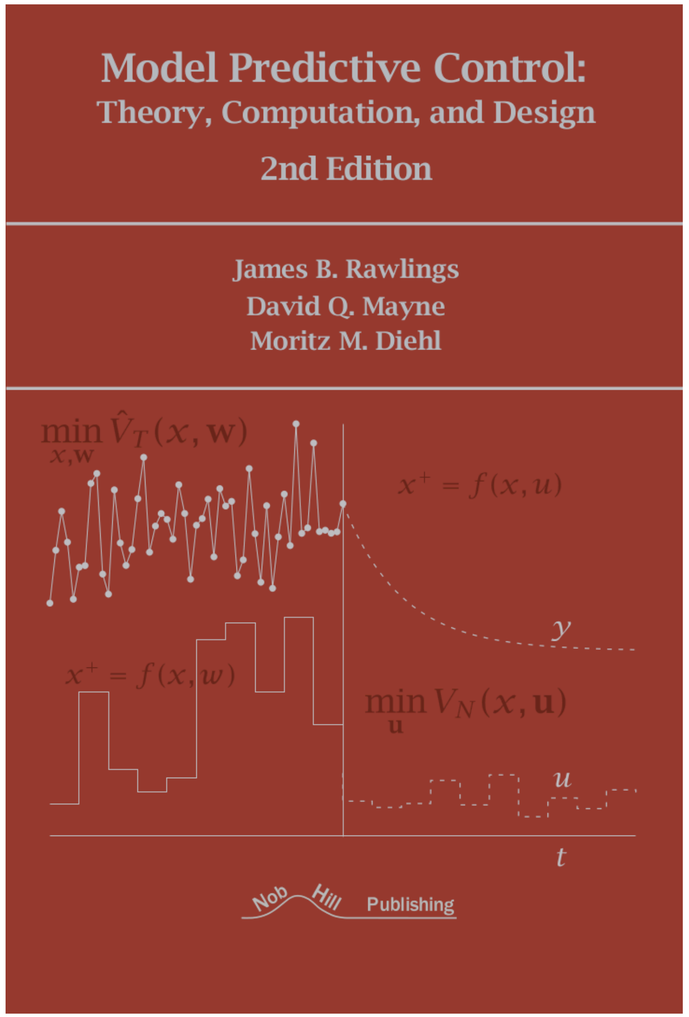
Receding horizon planning

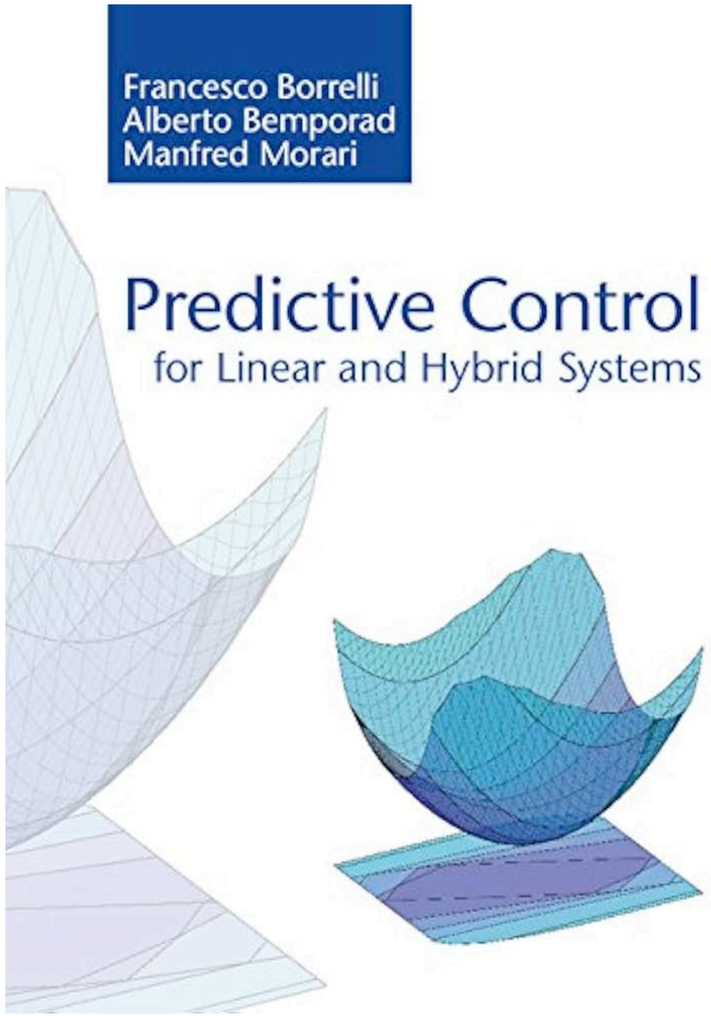
Practical considerations

- *Persistent feasibility*: it is important to ensure that the optimal control problem solved between iterations remains feasible
- *Stability*: the control input applied by solving successive optimal control problems should drive the system to the desired equilibrium point

Receding horizon planning

Practical considerations





Receding horizon planning

Control policy

- So far, we've studied the problem of open-loop trajectory optimization: how to compute a state-action trajectory to guide a system from x_{init} to x_{goal}
- Control policy: a (possibly stochastic) mapping $\mu : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ that takes the current state x_k and computes a control u_k to guide the system to x_{goal}
  - PD: $u_{\text{PD}} = k_p(x_{\text{ref}} - x_k) + k_d(\dot{x}_{\text{ref}} - \dot{x}_k)$
  - MPC: $u_{\text{MPC}} = \arg\min_{u_0} \mathscr{P}(x_k)$

References

- F. Borrelli, A. Bemporad, and M. Morari, “Predictive control for linear and hybrid systems,” Cambridge University Press, 2017.
- J. B. Rawlings, D. Q. Mayne, and M. Diehl, “Model Predictive Control: Theory, Computation, and Design,” 2017.