TUM School of Engineering and Design
Technical University of Munich

Egocentric Decision-Making for Autonomous Systems

A Relative Viewpoint for Optimization, Planning and Control

Jon Arrizabalaga Aguirregomezcorta

Complete reprint of the dissertation approved by the TUM School of Engineering and Design of the Technical University of Munich for the award of the academic degree of

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

Chair:

Prof. Dr.-Ing. Philipp Reiss

Examiners of the dissertation:

1. Prof. Dr.-Ing. Markus Ryll
2. Prof. Dr. Zachary Manchester
3. Prof. Dr. Moritz Diehl

The dissertation was submitted to the Technical University of Munich on 22.04.2025 and accepted by the TUM School of Engineering and Design on 08.10.2025.

Acknowledgments

I began my PhD journey in late 2020. Now, five years later—in 2025—after a global pandemic, the shockwave of the "ChatGPT moment", the rise and fall of crypto, and three German presidencies, it's finally drawing to a close. Looking back, there's no doubt that much has changed, both globally and personally. Over these past years, I've had the privilege of traveling widely and connecting with brilliant, inspiring people. In many ways, each of them has left a mark on this thesis. It's only right to acknowledge their importance here.

First and foremost, I want to thank my advisor, *Prof. Markus Ryll*, at the Technical University of Munich (TUM), for your trust and mentorship—both technically and personally. The rare combination of confidence and freedom you offered me was crucial, allowing me to explore and shape my research independently. Without this, the work presented here simply wouldn't have materialized. I'm also deeply grateful to my colleagues and collaborators at TUM—*Tim, Lukas, Yuxia, Christoph, Nicole, David, Luka, Michael, Ahmad, Felix,* and *Riddhi*—for the many stimulating discussions, shared efforts, and good times along the way. In addition, I want to thank the Aerospace undergraduate students I taught in *Engineering Mechanics*; your enthusiasm and energy made those lectures a true highlight of my time at TUM. To all of you, **danke schön**!

Second, I would like to sincerely thank *Prof. Zbynek Šír* at Charles University (Prague, Czech Republic) for opening a window into the world of pure mathematics and offering a perspective far removed from my usual work in robotics. Your insights into differential geometry—and especially Pythagorean Hodograph curves—have left a lasting imprint on this dissertation. For that, **děkuju**!

Third, my thanks go to *Prof. Zachary Manchester* at Carnegie Mellon University (CMU) for hosting me at the Robotic Exploration Lab, for sharing your passion and expertise in numerical optimization, for your mentorship, and for making me feel like one of the team. This appreciation naturally extends to all the amazing members of the RexLab: *Swami, Fausto, JJ, Paulo, Mitch, John, Arun, Juan, Pedro, Will, Ashley, Aaron, Sofia, Frederik, Sam, Khai,* and *Ibrahima*. For everything, **thank you**!

Last but not least, I want to thank my family—*Aita, Ama, Ekain, Tati, Lur,* and *Vera*—back home in Donostia–San Sebastián. Your constant support, infinite patience, and willingness to listen to my ramblings on endless video calls have kept me grounded throughout this journey. I also want to thank my second family—my *koadrilla*—for making sure that, no matter how far I was, things back home stayed as awesome as ever. *Osel, Aizpuru, Santa, Begi, Beñat, Danel, Demi, Egoitz, Eneko, Iker, Iribar, Viejo, Jonmi, Juan, Lander, Laxa, Manex, Manso, Markel, Martin, Pablo, Reparaz, Bayon, Cebe,* and *Ochoa* —thank you for being exactly who you are. On that note, **milesker**!

Abstract

Robotic systems have made remarkable progress in recent years, with quadrupeds and quadrotors demonstrating increasingly sophisticated capabilities. However, they remain less reliable, adaptable, and efficient than their biological counterparts. Even state-of-the-art humanoid robots, while capable of impressive stunts, still struggle with routine tasks. This dissertation examines a key reason for this gap: the perspective from which actions are decided and executed. While biological organisms act from a first-person, egocentric viewpoint, most autonomous robotic systems rely on third-person abstractions. Motivated by this discrepancy, this work investigates how adopting an egocentric perspective in control and decision-making can help bridge the performance gap between robots and animals.

To this end, the dissertation develops a foundation for egocentricity in all modules of autonomous decision-making—ranging from motion description to environment representation, and their application to real-world systems. Central to this framework is a rethinking of the three core elements that define motion: the system's states, its reference frame, and its perception of the surroundings. This work proposes principled approaches for handling each of these components from an egocentric standpoint.

Specifically, the dissertation introduces three major contributions: (i) a spatial projection technique that allows system dynamics to be expressed relative to any local frame, without imposing constraints on that frame's motion; (ii) a detailed analysis of how to represent the motion of the local frame, leading to the identification of a particularly effective reference frame for robotic autonomy; and (iii) a differentiable, continuous, and smooth representation of safe space—whether defined by obstacle-free regions or reachable sets—that naturally fits within the egocentric paradigm.

Together, these components establish a unified framework that spans the full spectrum of egocentric techniques, encompassing traditional path-following approaches as well as more advanced methods such as contouring control, progress-based Model Predictive Control, and Reinforcement Learning.

The effectiveness and generality of the proposed framework are demonstrated across a diverse set of applications, including motion planning for autonomous mobile robots, the design of robust control strategies, and the generation of slosh-free maneuvers for robotic manipulators. In each case, the egocentric perspective enables the execution of complex motions that are challenging—or even infeasible—to achieve with conventional, non-egocentric approaches.

Contents

List of Authored Publications

[1] Jon Arrizabalaga and Markus Ryll. “Towards time-optimal tunnel-following for quadrotors”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 4044–4050.

[2] Jon Arrizabalaga and Markus Ryll. “Spatial motion planning with pythagorean hodograph curves”. In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE. 2022, pp. 2047–2053.

[3] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. “Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms”. In: *IEEE Robotics and Automation Letters* 8.4 (2023), pp. 2397–2404.

[4] Jon Arrizabalaga and Markus Ryll. “Sctomp: Spatially constrained time-optimal motion planning”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 4827–4834.

[5] Jon Arrizabalaga and Markus Ryll. “Pose-following with dual quaternions”. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE. 2023, pp. 5959–5966.

[6] Jon Arrizabalaga, Lukas Pries, Riddhiman Laha, Runkang Li, Sami Haddadin, and Markus Ryll. “Geometric Slosh-Free Tracking for Robotic Manipulators”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 1226–1232.

[7] Tim Salzmann, Jon Arrizabalaga, Joel Andersson, Marco Pavone, and Markus Ryll. “Learning for casadi: Data-driven models in numerical optimization”. In: *6th Annual Learning for Dynamics & Control Conference*. PMLR. 2024, pp. 541–553.

[8] Jon Arrizabalaga, Zachary Manchester, and Markus Ryll. “Differentiable collision-free parametric corridors”. In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024, pp. 1839–1846.

[9] Jon Arrizabalaga, Fausto Vega, Zbyněk ŠÍR, Zachary Manchester, and Markus Ryll. “PHODCOS: Pythagorean Hodograph-based Differentiable Coordinate System”. In: *IEEE Aerospace Conference (AERO)* (2025).

[10] Kevin Tracy, John Zhang, Jon Arrizabalaga, and Zachary Manchester. “The Trajectory Bundle Method: Unifying Sequential Convex Programming and Sampling-Based Trajectory Optimization”. In: *Under Review* (2025).

[11] Jon Arrizabalaga, Zbyněk Šír, Zachary Manchester, and Markus Ryll. “A Universal Formulation for Path-Parametric Planning and Control”. In: *Under review* (2025).

Part I

Preamble

We begin by presenting the rationale behind adopting an egocentric approach to decision-making in autonomous systems. In Chapter 1, we first examine the core motivation from a conceptual and philosophical perspective. After defining the required preliminaries in Chapter 2, we further establish a principled foundation for the egocentric paradigm in Chapter 3, which introduces a case study that illustrates its advantages in autonomous system design. This initial part lays the groundwork for the discussions and developments presented throughout the remainder of this dissertation.

1 Introduction

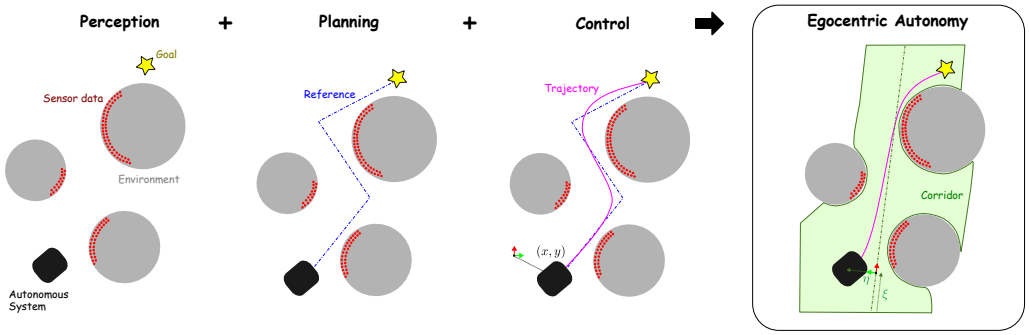


Figure 1.1 The software stack of autonomous systems has traditionally been divided into three modular stages: perception, planning, and control. In this dissertation, we challenge this convention by adopting an egocentric paradigm—one where decision-making is rooted in a first-person perspective. This shift enables algorithms with superior spatial awareness, unlocking geometrically complex behaviors that remain beyond the reach of traditional, non-egocentric approaches.

In the realm of robotics and autonomous systems, the triad of Perception, Planning, and Control has long been regarded as distinct and often compartmentalized stages in the decision-making process. Perception involves understanding and interpreting sensory data from the environment, planning encompasses the formulation of actions or strategies based on this perceived information, and control entails the execution of these planned actions to achieve desired outcomes. Fragmenting these components into separate stages inhibits real-time responsiveness, adaptability, and overall system performance, thereby underscoring the critical need for a more cohesive and integrated approach to address the complexities of robotic autonomy.

In contrast, these processes in humans are inherently intertwined and operate in parallel, allowing for rapid adaptation and flexible behavior in complex and uncertain environments. One approach to address this disparity between human and robots is through the development of end-to-end algorithms, which learn directly from sensor observations, bypassing the need for explicit segmentation of perception, planning, and control. Despite overcoming the limitations of classical approaches, such methods often suffer from being perceived as black-box systems, where the internal decision-making processes are opaque and difficult to interpret.

Recognizing the limitations of both traditional modular approaches and learned methods, the goal of this dissertation is to unify perception, planning, and control in a transparent and understandable manner, by combining the interpretability and explainability of modular decomposition with the flexibility and adaptability of learned models.

To make it happen, the presented content centers around the development of algorithms that empower autonomous systems to adapt to and fully exploit their environment. At the core of this approach is the concept of an *egocentric perspective* that shifts planning and

control to a relative first-person viewpoint, similar to how humans and animals perceive and interact with their surroundings. This egocentric perspective allows to embed the geometric properties of the environment into the underlying optimization problem, providing easy access to navigationally meaningful attributes, such as progress, curvature or the distance to the closest obstacle, thereby facilitating the formulation of behaviors that fully exploit the environment and are extremely hard to formulate with standard nonegocentric formulations.

1.1 Motivation

The animal kingdom offers some of the most remarkable examples of locomotion. For virtually any task and environment, nature has evolved the optimal morphology to solve it. Covering long distances across open fields? Horses. Descending near-vertical cliffs? Goats. Agile flight through dense forests? Hummingbirds. High-speed, aggressive aerial maneuvers? Peregrine falcons. Swimming against strong currents? Salmon. Nature even achieves seemingly impossible feats, such as running on water—exemplified by the basilisk lizard.

In contrast, non-organic counterparts still lag far behind. While quadrupedal robots have recently demonstrated the ability to traverse challenging terrain, their reliability and adaptability remain significantly inferior to that of horses or goats. Quadrotors have made impressive progress in agile flight, yet their performance is typically restricted to controlled environments and does not generalize to unstructured settings. Furthermore, their reliance on propellers—a highly inefficient morphology—renders them incapable of tasks that birds perform effortlessly, such as crossing an ocean and landing on a branch to rest. In these cases, the ability to combine agility and efficiency in a single system remains an open challenge.

A similar gap exists in the realm of humanoid robots. While some can execute acrobatic maneuvers beyond the capabilities of the average human, they still struggle with routine tasks such as unloading a dishwasher or walking a child to school. Given these disparities between biological and artificial systems, a fundamental question arises: What accounts for this difference? What key principle has yet to be incorporated into autonomous systems?

While general intelligence undoubtedly plays a role, this dissertation focuses on the execution of movement—the process of translating high-level decisions into physical actions. When doing so, a critical yet often overlooked aspect of biological locomotion is the perspective from which decisions are made. Every living organism inherently operates from a first-person, egocentric viewpoint. From the moment we wake up, every step we take and every movement we make is relative to our own frame of reference.

Yet, most decision-making algorithms adopt a third-person perspective, akin to controlling a character in a video game. While this abstraction has certain advantages, it is a fundamental departure from how biological systems interact with their environment. If we aim to bridge the gap between artificial and natural locomotion, reconsidering this perspective is essential.

This dissertation systematically explores this idea, first investigating whether an egocentric approach to algorithmic design can enhance system performance. We demonstrate that integrating such a perspective improves agility, safety, and spatial awareness. Building on this insight, we develop a framework that facilitates the design of egocentric decision-making algorithms, addressing key challenges along the way: How can we enforce an egocentric perspective in autonomous systems? How do we ensure that it generalizes across

Figure 1.2 Nature offers countless examples of remarkable locomotion, whether on land, in the air, in water, or in hybrid environments. These movements—optimized for agility, speed, or efficiency—are shaped by evolution through morphology. Yet, what unites them all is a decision-making process rooted in a first-person, egocentric perspective. In this dissertation, we seek to translate this philosophy to artificial autonomous systems.

different environments and tasks? And once this foundation is established, how do we design decision-making processes that effectively leverage it?

By addressing these questions, this dissertation advances the development of egocentric decision-making in autonomous systems, moving closer to the adaptability, efficiency, and fluidity observed in nature.

1.2 Problem Statement and Research Roadmap

This dissertation focuses on the algorithmic design required to enable the autonomous motion of dynamical systems. Given a system, an environment, and a task, our objective is to develop algorithms that allow the system to execute the task autonomously—without human intervention—purely through computational decision-making. Consistent with the perspective outlined in the previous section, these algorithms are designed from an egocentric viewpoint, shifting the decision-making process from a fixed, external reference frame to an agent-centered perspective. This approach defines the *Egocentric Decision-Making problem*.

Any problem concerning motion design—whether in motion planning, trajectory optimization, or low-level control—can be formulated from an egocentric perspective. Fully adopting this viewpoint requires moving beyond the standard Euclidean parameterization and instead employing a local representation that describes motion in a subjective manner. Intuitively, operating a vehicle from a third-person perspective is significantly more difficult—if not impossible—compared to driving from within the cockpit. While a comprehensive explanation of this phenomenon is beyond the scope of this thesis, it is evident that a first-person perspective provides a clearer understanding of one's position relative to the environment. For example, when driving, having direct knowledge of one's progress along a road and distance from the centerline enables continuous assessment of surroundings and necessary maneuvers to safely reach a destination. Building on this intuition, this thesis extends the premise that *first-person navigation is inherently easier* into the algorithmic domain.

A natural system representation that facilitates egocentricity is one in which the vehicle's position is projected onto the centerline of the road, resulting in an alternative representation defined by *tangential* and *transversal* components. The tangential component quantifies progress along the path, while the transversal component captures the deviation from the centerline. This reformulation, referred to as the *spatial representation*, plays a central role in this dissertation. Importantly, this representation does not impose a constraint requiring strict adherence to the centerline; deviations within the bounds of the road remain permissible. By explicitly encoding geometrically meaningful properties, such as progress along a path and orthogonal distance to it, this approach not only simplifies the design of decision-making algorithms but also enhances their performance, enabling navigation capabilities that are unattainable through non-egocentric methods.

While the car example provides an intuitive introduction, this dissertation addresses the problem in its most general form, considering three-dimensional motion for arbitrary, non-hybrid, underactuated, and constrained dynamical systems. On the one hand, this generality is essential to ensuring that the findings presented herein contribute to a broader paradigm shift in the algorithmic design of autonomous systems by embedding egocentricity. On the other hand, this generality introduces fundamental research challenges that must be carefully addressed.

1.2.1 Research Objectives

Building upon this motivation and extending the current state of the field, this dissertation pursues three primary objectives:

G1 To understand the role of egocentricity in the design of decision-making algorithms that enable optimal operation.

G2 To identify underlying connections among existing egocentric methods, thereby broadening their applicability across a wider range of problems.

G3 To develop a toolkit that allows practitioners to seamlessly incorporate the egocentric perspective into the decision-making processes of autonomous systems.

Achieving these objectives enables autonomous systems to adopt an egocentric perspective, thereby enhancing the spatial awareness of existing decision-making algorithms within the autonomy software stack.

1.2.2 Research Questions

To meet these objectives, we aim to integrate egocentricity into all autonomy modules, including (1) system modeling, (2) environment representation, and (3) planning & control. This approach allows us to structure our research around the following key questions:

Q1 How can we *enable egocentricity* for any underactuated, constrained dynamical system?

Q2 What is the most suitable *collision-free spatial representation* for egocentric methods?

Q3 How can we *conduct planning and control* from a relative—*egocentric*—viewpoint?

Answering these questions will facilitate the formulation of egocentric decision-making methods, enhancing the spatial awareness of existing autonomy algorithms.

1.2.3 Research Contributions

This dissertation advances the field by introducing key contributions that formalize and extend the egocentric decision-making paradigm, providing both theoretical foundations and practical implementations:

C1 A *universal framework* that unifies state-of-the-art egocentric methods, simplifying the design of egocentric decision-making algorithms regardless of system morphology, environmental geometry, or task objectives.

C2 A *continuous and differentiable state-space representation* method for collision-free environments that aligns with the egocentric paradigm.

C3 *Applications of egocentric methods* across motion planning, trajectory optimization, and low-level control for a diverse range of robotic morphologies, including quadrotors, autonomous vehicles, and robotic manipulators.

The manner in which the scientific publications presented in this dissertation address the outlined contributions is summarized in Table 1.1. By engaging with these research challenges, the dissertation lays a foundation for advancing egocentric decision-making in autonomous systems, ultimately enhancing their adaptability, efficiency, and spatial awareness.

Table 1.1 Scientific publications included in this dissertation, organized by the contribution they support and the chapter in which they are discussed. Relevant supplementary materials—such as code repositories and presentation videos or talks—are accessible via clickable links (in the digital version of this document) or can be found as URL references in the Appendix V.

Publication	Ref.	Suppl. Material	Contribution	Chapter
Towards Time-Optimal Tunnel-Following for Quadrotors	[1]	Video	C1	3
Spatial Motion Planning with Pythagorean Hodograph Curves	[2]	Video, Talk	C1	4
Pythagorean Hodograph-based Differentiable Coordinate System	[3]	Code, Talk	C1	5
Differentiable Collision-Free Parametric Corridors	[4]	Code, Video, Talk	C2	6
Spatially Constrained Time-Optimal Motion Planning	[5]	Video, Talk	C3	7
Pose-Following with Dual Quaternions	[6]	Code, Talk	C3	7
Geometric Slosh-Free Tracking for Robotic Manipulators	[7]	Code, Video, Talk	C3	7
A Universal Framework for Path-Parametric Planning & Control	[8]	Website, Code	C1, C2, C3	8

1.3 Thesis Structure

Building upon the motivations and foundational concepts introduced earlier, the structure of this dissertation (also illustrated in Fig. 1.3) is organized into four parts, each comprising several chapters that progressively develop the central themes and contributions of this work:

Part 1 - Preamble lays the groundwork for the content that is going to be presented in the dissertation.

Chapter 2 introduces the foundational concepts essential to this dissertation. These evolve around the definition of the geometric reference, its parameterization, and the relative/moving or local frame used to provide an egocentric perspective.

Chapter 3 quantitatively assesses the advantages of an egocentric perspective in designing decision-making algorithms for autonomous vehicles. To demonstrate this, we employ an illustrative example highlighting the advantages of an egocentric approach.

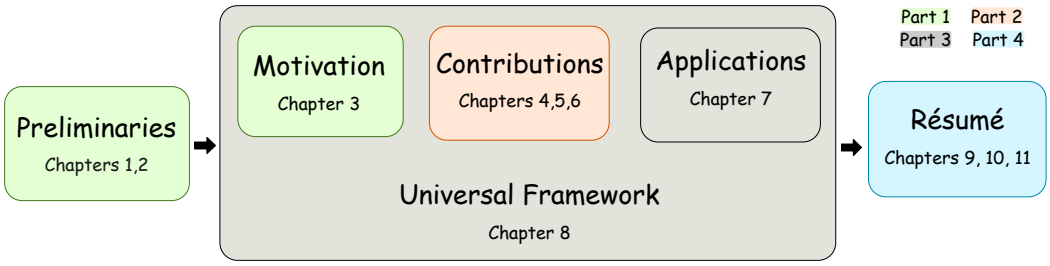


Figure 1.3 Structure of the dissertation. The presented content is divided into four parts, each containing multiple chapters.

Part 2 - Formulation of Egocentric Methods outlines the key elements that will enable the generalization of egocentric decision-making across a broad range of autonomy applications. These elements are grouped into three main areas, each explored in a separate chapter.

> **Chapter 4** evolves around enabling egocentricity by representing the motion as a spatial parameterization of the system dynamics that is fully independent of the specific details underlying the path parameterization or reference frame.
>
> **Chapter 5** focuses on how to compute the relative frame that will be used to enable egocentricity in such a way that it is compatible with gradient-based decision-making algorithms.
>
> **Chapter 6** evolves around the representation on the safe or obstacle-free space, such that it perfectly interplays with the other ingredients of egocentric-decision-making presented in the previous chapters.

Part 3 - Applications to Robot Autonomy reinforces the findings presented in Part II by offering a more applied perspective through real-world examples and by presenting the contributions through a unified lens. This is conducted in two separate chapters:

> **Chapter 7** showcases real-world implementations of the proposed contributions. It focuses on three distinct projects that vary in theoretical depth and span key aspects of autonomy, from low-level control to motion planning, as well as full-stack system and application-oriented design.
>
> **Chapter 8** unifies the theoretical developments from the previous part within a general framework that standardizes the design of egocentric decision-making methods for autonomous systems.

Part 4 – Résumé concludes the dissertation by summarizing the key contributions, highlighting the most pressing open research questions, and reflecting on how the work aligns with the dissertation's original mission. Each of these aspects is addressed in a dedicated chapter:

> **Chapter 9** provides a self-contained synthesis that brings together the main concepts and results presented throughout the dissertation.
>
> **Chapter 10** discusses promising directions for future research, outlining questions that naturally arise from this work and that could accelerate the adoption of egocentric approaches in decision-making.

Chapter 11 offers a reflective perspective on how the contributions of this dissertation support the overarching goal of empowering autonomous systems with capabilities that match or surpass those of biological agents.

2 Background

In this chapter, we introduce the foundational concepts essential to this dissertation. These evolve around the definition of the geometric reference, its parameterization, and the relative/moving or local frame used to provide an egocentric perspective.

2.1 Assigning a path parameter to the reference path

Existing autonomous navigation systems typically represent the reference path in one of two ways: either as a discrete set of waypoints [9] or as a continuous parametric function generated by a higher-level planner [10, 11].

In the waypoint-based approach, the reference path is defined by an ordered sequence of waypoints, denoted as $\boldsymbol{wp} = [\boldsymbol{wp}_1, \boldsymbol{wp}_2, \ldots, \boldsymbol{wp}_n] \in \mathbb{R}^{3\times n}$. These waypoints are then interpolated to form a smooth curve using methods such as those described in [12]. In contrast, the parametric function approach bypasses the need for interpolation, as the planner directly provides a smooth parametric representation, typically using B-splines or similar basis functions.

Both approaches ultimately yield a smooth parametric representation of the reference path, which we denote as $\boldsymbol{\gamma} : \mathbb{R} \to \mathbb{R}^3$, where θ is the path parameter. The arc-length of the path, measured from some initial parameter value θ_0, is given by

$$l(\theta) = \int_{\theta_0}^{\theta} \|\boldsymbol{\gamma}'(\theta)\| \, d\theta. \tag{2.1}$$

This formulation reveals an important but often overlooked distinction: the arc-length $l(\theta)$ is generally different from the path parameter θ. To emphasize this distinction, we define the *parametric speed* as the magnitude of the path derivative:

$$\sigma(\theta) = \|\boldsymbol{\gamma}'(\theta)\|. \tag{2.2}$$

The parametric speed quantifies the relationship between changes in the path parameter and corresponding changes in arc-length. Notably, only when $\sigma(\theta) \equiv 1$ does the path parameter coincide with arc-length, in which case we say the curve is parameterized by arc-length[1].

2.2 Assigning a moving frame to the reference path

Egocentricity involves shifting decision-making to a local frame that moves alongside the dynamical system. Without loss of generality, let this frame be attached to a path Γ. We augment the position function $\boldsymbol{\gamma}(\theta)$ by associating it with a moving frame whose rotation

[1] For a detailed discussion of arc-length parameterization, see [12].

matrix is given by a parametric function $\mathrm{R} : \mathbb{R} \to \mathbb{R}^{3\times3}$, also parameterized by θ. Together, the position function $\boldsymbol{\gamma}(\theta)$ and the rotation function $\mathrm{R}(\theta)$ define the *geometric reference* as:

$$\Gamma = \left\{\theta \in [\theta_0, \theta_f] \subseteq \mathbb{R} \mapsto \boldsymbol{\gamma}(\theta) \in \mathbb{R}^3, \mathrm{R}(\theta) \in \mathbb{R}^{3\times3}\right\}. \tag{2.3}$$

We refer to the moving frame $\mathrm{R}(\theta) = [\boldsymbol{e}_1(\theta), \boldsymbol{e}_2(\theta), \boldsymbol{e}_3(\theta)]$ as the *path-frame* $(\cdot)^\Gamma$, which is assumed to be *adapted*, meaning that the first basis vector aligns with the path's tangent:

$$\boldsymbol{e}_1(\theta) = \frac{\boldsymbol{\gamma}'(\theta)}{\|\boldsymbol{\gamma}'(\theta)\|} = \frac{\boldsymbol{\gamma}'(\theta)}{\sigma(\theta)}. \tag{2.4}$$

The variation of this frame with respect to the path parameter is dictated by the angular velocity $\boldsymbol{\omega}(\theta) = [\omega_1(\theta), \omega_2(\theta), \omega_3(\theta)]$, which can also be expressed in the path-frame as $\boldsymbol{\omega}^\Gamma(\theta) = \left[\omega_1^\Gamma(\theta), \omega_2^\Gamma(\theta), \omega_3^\Gamma(\theta)\right]$:

$$\boldsymbol{\omega}(\theta) = \boldsymbol{\omega}^\Gamma(\theta)\,\mathrm{R}^\intercal(\theta) = \omega_1^\Gamma(\theta)\boldsymbol{e}_1(\theta) + \omega_2^\Gamma(\theta)\boldsymbol{e}_2(\theta) + \omega_3^\Gamma(\theta)\boldsymbol{e}_3(\theta). \tag{2.5}$$

The motion of the moving frame $\mathrm{R}(\theta)$ is governed by:

$$\mathrm{R}'(\theta) = \underbrace{\begin{bmatrix} 0 & -\omega_3(\theta) & \omega_2(\theta) \\ \omega_3(\theta) & 0 & -\omega_1(\theta) \\ -\omega_2(\theta) & \omega_1(\theta) & 0 \end{bmatrix}}_{\Omega(\theta)} \mathrm{R}(\theta) \equiv \mathrm{R}(\theta) \underbrace{\begin{bmatrix} 0 & -\omega_3^\Gamma(\theta) & \omega_2^\Gamma(\theta) \\ \omega_3^\Gamma(\theta) & 0 & -\omega_1^\Gamma(\theta) \\ -\omega_2^\Gamma(\theta) & \omega_1^\Gamma(\theta) & 0 \end{bmatrix}}_{\Omega^\Gamma(\theta)}, \tag{2.6}$$

where $\Omega(\theta)$ and $\Omega^\Gamma(\theta)$ are the skew-symmetric matrices corresponding to the angular velocity vectors in the world frame $\boldsymbol{\omega}(\theta)$ and the path-frame $\boldsymbol{\omega}^\Gamma(\theta)$, respectively[2]. From Eq. (2.6), the components of the angular velocity in the moving frame are given by:

$$\left[\omega_1^\Gamma(\theta), \omega_2^\Gamma(\theta), \omega_3^\Gamma(\theta)\right] = \left[\boldsymbol{e}_2'(\theta)\boldsymbol{e}_3(\theta), \boldsymbol{e}_3'(\theta)\boldsymbol{e}_1(\theta), \boldsymbol{e}_1'(\theta)\boldsymbol{e}_2(\theta)\right]. \tag{2.7}$$

As discussed in Chapter 5, the angular velocity of the moving frame is directly linked to the *curvature* (κ) and *torsion* (τ) of the path and strongly depends on the chosen frame. In the second part of this dissertation, we examine various strategies for defining the moving frame and ultimately propose an efficient, simple, and compact algorithm for computing it.

[2]For a detailed proof of the equivalence in Eq. (2.6), see Theorem 1 in [13].

3 Why Egocentric? A Motivating Case-Study

We have already discussed the compelling reasons for adopting an egocentric approach in algorithmic design, supported by intuitive and conceptual examples that illustrate the philosophy behind egocentric decision-making. In this chapter, we move beyond these conceptual foundations to quantitatively assess the advantages of an egocentric perspective in designing decision-making algorithms for autonomous vehicles.

To demonstrate this, we employ an illustrative example highlighting the advantages of an egocentric approach. The results yield two key insights: First, the improved navigation performance underscores the potential of egocentricity, serving as both a validation of its benefits and a core motivation for this dissertation. Second, it reveals limitations in existing egocentricity-enabling tools, generating critical research questions that guide the trajectory of this thesis.

3.1 The Egocentric Decision-Making Problem

We consider continuous time, nonlinear systems of the form

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x_0}\,, \tag{3.1a}$$

$$\boldsymbol{y}(t) = h(\boldsymbol{x}(t))\,, \tag{3.1b}$$

where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ define state and input constraints. Function $f : \mathbb{R}^{n_x} \times \mathbb{R}^m \mapsto \mathbb{R}^{n_x}$ refers to the equations of motion of an arbitrary dynamic system, while the map $h : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ defines the output of the system $\boldsymbol{y} \in \mathbb{R}^{n_y}$. Both f and h functions are assumed to be sufficiently continuously differentiable. The initial state is contained in a subset of the constrained state set, i.e., $\boldsymbol{x}_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$.

Egocentric Decision-Making refers to the problem where the system states $\boldsymbol{x}(t)$ benefit from a relative frame that moves alongside the dynamical system f in eq. (3.1). As introduced in the previous chapter, without loss of generality, this frame is assumed to be given by a parametric function $\mathrm{R}(\cdot)$ and attached to a geometric reference Γ, as defined in eq. (2.3).

Egocentric methods can be categorized based on how system (3.1) benefits from the relative frame R. Broadly, these benefits fall into two categories: either they offer a novel way to represent the system's dynamics, or they contribute to achieving a desired behavior. Another axis of classification stems from their methodological foundations, be it classical control, optimization, or reinforcement learning. Figure 3.1 illustrates this taxonomy, along with the core principles that underpin egocentric methods. Before delving deeper into these categories, the following subsection provides a historical perspective on how egocentric methods have evolved over the past few decades.

3.2 The Evolution of Egocentric Methods: From Path-Following to MPC and RL

In the domain of planning and control, egocentric approaches began to gain prominence in the 1980s, particularly within the context of robotic manipulators. Early contributions highlighted the advantages of incorporating the path parameter into control and planning strategies, enabling tasks such as rescaling infeasible trajectories [14] and computing time-optimal end-effector motions [15, 16]. These pioneering efforts were among the first to utilize the path parameter to reformulate time-based dynamics in spatial terms, as depicted in Fig. 3.1 and elaborated in the subsequent subsection.

Shortly thereafter, similar principles were extended to mobile robot navigation [17, 18, 19], where deviations from the path were decomposed into tangential and orthogonal components. This decomposition enabled the design of controllers with explicit influence over progression along the path. Such foundational ideas eventually matured into more advanced frameworks [20, 21], culminating in the emergence of path-following as a robust and superior alternative to path-tracking [22, 23, 24, 25].

A major turning point followed, marking the transition of egocentric methods from analytical formulations to optimization-driven strategies. Two distinct but complementary lines of work laid the foundation for this shift.

On one hand, the influential study in [26] showed that the time-optimal traversal of a path by a robotic manipulator—originally explored in [15, 16]—could be recast as a convex optimization problem. On the other hand, research in [27, 28] emphasized the benefits of embedding the tangential and orthogonal error components—first introduced in [17, 18, 19]—within a receding horizon control framework, commonly referred to as Model Predictive Control (MPC). This formulation, often known as Contouring Control or MPCC when embedded in an MPC scheme, introduced the notion of intentional path deviation, allowing a balance between tracking accuracy and traversal efficiency.

Fueled by advances in numerical solvers and growing computational capabilities, these developments led to a surge in practical applications of egocentric methods, particularly those relying on MPC. A large share of real-world systems employing egocentric strategies today trace their lineage to this family of approaches. Moreover, the rise of Reinforcement Learning (RL) extended the reach of egocentric formulations even further. Their natural alignment with geometric progression and path-centric metrics made them well-suited for shaping reward functions in learning-based frameworks.

In contrast to earlier rigid path-tracking schemes, many modern egocentric methods accommodate deviations by defining safe corridors, tunnels, or tubes—allowing any feasible trajectory within these bounds. This flexibility proved invaluable in constrained environments, signaling a shift away from strict path adherence.

As a result, egocentric methods have rapidly proliferated across diverse domains, including autonomous driving [29, 30, 31, 32, 33, 34, 35], agile aerial robotics [36, 37, 1, 38, 39, 40], robotic arms [41, 42], and even unconventional systems like cranes [43].

This retrospective reveals that path-parametric approaches have undergone substantial evolution, bridging theory and practice across a wide spectrum of applications. Nevertheless, despite this progress, egocentric methods are often presented as standalone techniques, with limited connections among them. This dissertation addresses that gap by uncovering the

missing links and offering a unified perspective, ultimately facilitating the integration of egocentric methods into the broader framework of autonomous decision-making.

3.3 Methods for Egocentric Decision-Making

At the beginning of this section, we established that egocentric decision-making approaches are characterized by the use of a relative reference frame that moves with the dynamical system. We also introduced three complementary perspectives for categorizing existing methods: (1) how they represent system dynamics, (2) their implementation and methodological foundations, and (3) how they achieve the desired performance. Together, these perspectives offer a multifaceted framework for understanding the structure and design principles of egocentric approaches. An overview of this classification is depicted in Fig. 3.1.

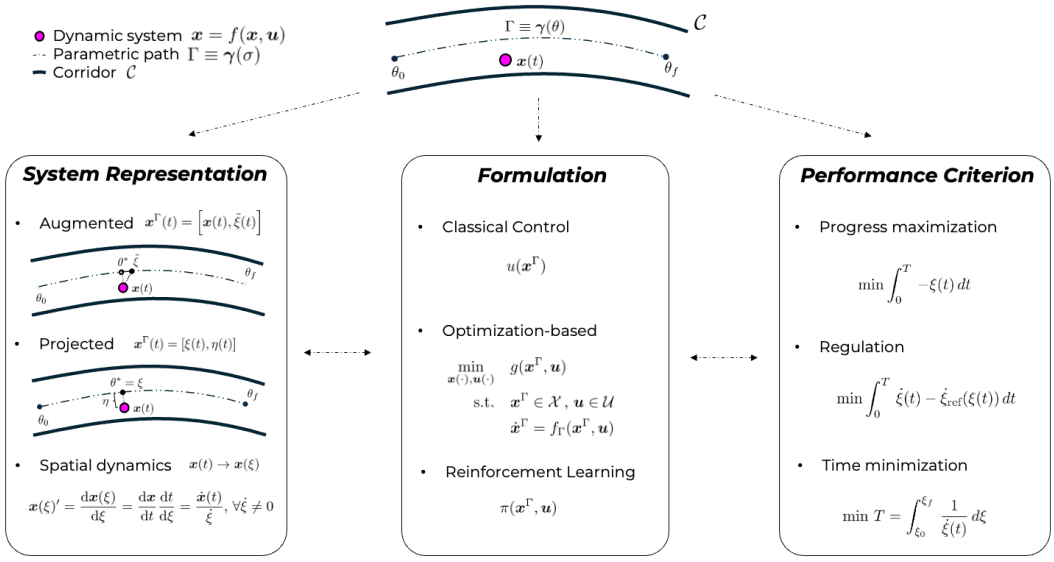


Figure 3.1 Egocentric methods utilize a moving reference frame that follows the dynamical system along a parametric path, typically confining system evolution within a bounded corridor. These methods can be classified by (1) how the relative frame represents system dynamics, (2) their implementation formulation, and (3) the selected performance criterion.

Importantly, any method based on egocentric principles—whether part of the historical evolution described in the previous subsection or belonging to future or alternative developments—can be characterized as a specific combination of choices within these three categories. In other words, viewing the literature through the prism of these dimensions reveals the underlying commonalities across different approaches. This perspective helps identify the missing conceptual links, shedding light on why the field often appears fragmented despite a shared foundation.

3.3.1 System dynamics description

The first way to distinguish existing egocentric methods is according to how they adopt egocentricity in the parameterization of the system's motion.

Projecting the system states

The most natural way to embrace egocentricity is to represent the system's motion in the relative frame. Instead of relying on the standard Euclidean parameterization, the system states are projected into the relative frame R. The resulting states from this projection are referred to as *spatial coordinates* and are given by

$$\boldsymbol{x}^{\Gamma}(t) = [\xi(t), \boldsymbol{\eta}(t)] \;\text{ with }\; \xi(t) = \theta^{*}\,, \tag{3.2}$$

where $\xi(t) \in \mathbb{R}$ and $\boldsymbol{\eta}(t) \in \mathbb{R}^2$ represent the tangential and transversal motion along the geometric reference Γ. In other words, $\xi(t)$ denotes the progress up to the closest point θ^* of the dynamical system with respect to Γ, while $\boldsymbol{\eta}(t)$ represents the distance to this point, lying in the normal plane formed by the second and third components of the relative frame R.

Explicit access to these states enables the formulation of decision-making approaches that facilitate intricate motions, which are difficult to achieve using a standard Euclidean representation. However, this representation requires determining the closest point on the geometric path Γ, which itself constitutes an optimization problem and may pose challenges when fast and accurate feedback is required.

Augmenting the system states

An alternative to projecting the entire state description onto the relative frame is to preserve the original state representation while augmenting the system states with a *virtual state* that tracks progress along the path. This approach is formulated as

$$\boldsymbol{x}^{\Gamma}(t) = [\boldsymbol{x}(t), \tilde{\xi}(t)]\,, \tag{3.3}$$

where $\tilde{\xi}(t)$ serves as an approximation of $\theta^* = \xi(t)$. Ideally, these quantities would be identical, but since the dynamics of the virtual state $\tilde{\xi}(t)$ are decoupled from the original system dynamics, a drift between them is inevitable. In practice, decision-making algorithms are designed to ensure that this discrepancy remains sufficiently small.

Spatially transforming the system dynamics

A compelling alternative to projection and augmentation is to perform a spatial transformation of the time-based system dynamics. This transformation is achieved by leveraging the chain rule as follows:

$$\boldsymbol{x}' \coloneqq \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\xi} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\xi} \tag{3.4}$$

Observing that $\frac{\mathrm{d}t}{\mathrm{d}\xi} = 1/\dot{\xi}$ and assuming $\dot{\xi} \neq 0$, equation (3.4) simplifies to

$$\boldsymbol{x}' = \frac{1}{\dot{\xi}} f(\boldsymbol{x}, \boldsymbol{u})\,, \quad \forall \dot{\xi} \neq 0\,.$$

The resulting spatially transformed equations of motion for the dynamical system (3.1) are

$$\boldsymbol{x}'(\xi) = \frac{f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))}{\dot{\xi}(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))}, \quad \boldsymbol{x}(\xi_0) = \boldsymbol{x_0}\,, \tag{3.5a}$$

$$\boldsymbol{y}(\xi) = h(\boldsymbol{x}(\xi))\,, \tag{3.5b}$$

where $\dot{\xi}(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))$ is yet to be defined. Comparing the *spatially transformed* system (3.5) with the original system (3.1), it becomes evident that the dynamics now evolve with respect to the *path parameter* ξ rather than *time* t.

A notable drawback of this representation is that the spatial transformation is not applicable when the system is stationary with respect to the geometric reference—i.e., when $\dot{\xi}(t) = 0$. Consequently, this approach requires the system to be in continuous motion.

3.3.2 Performance criterion

Another way to differentiate egocentric methods is based on the desired performance. Having privileged access to geometrically meaningful features, such as progress along the path and the orthogonal distance to it, enables the formulation of decision-making algorithms with objectives that are geometrically more intricate than those of non-egocentric approaches.

The manner in which these criteria are incorporated depends on the specific approach used to implement the decision-making algorithm—whether through an error function, a cost function, or a reward function. This aspect will be discussed in the following category. Here, we focus on how these criteria are formulated to generate different behaviors. Without loss of generality, assuming a finite horizon T, this can be expressed as

$$\min_{x} g(x)\,, \tag{3.6}$$

where $g(\cdot)$ represents the function defining the behavior or performance criterion that we aim to optimize.

Progress maximization

The most straightforward way to leverage the egocentric parameterization is to enforce advancement. In the egocentric domain, this is as simple as maximizing the path parameter, which, for any of the three representations discussed above, is directly accessible. This can be achieved by

$$g\left(x^{\Gamma}(t)\right) = -\int_0^T \xi(t)\, dt\,. \tag{3.7}$$

This formulation provides a clean and direct method for driving the system toward its goal. As a result, it is commonly employed in racing scenarios or as an approximation for minimum-time problems. However, due to its unbounded nature, it often introduces numerical challenges, and its application requires careful consideration.

Tracking of velocity profile

An alternative to purely maximizing progress is to track a predefined velocity profile along the path parameter, i.e.,

$$g\left(x^{\Gamma}(t)\right) = -\int_0^T \left(\dot{\xi}(t) - \dot{\xi}_{\mathrm{ref}}(t)\right)\, dt\,, \tag{3.8}$$

where $\dot{\xi}_{\mathrm{ref}}(t)$ is a reference velocity profile specified by the user. This formulation is particularly appealing for applications where the traversal speed at which a task is completed depends on domain knowledge.

For instance, in manufacturing or inspection tasks performed by a robotic manipulator, the end-effector may need to reduce its speed in certain critical sections of a part while accelerating through less significant areas. By encoding this knowledge into a velocity profile, this approach enables precise control over motion speed, ensuring that task-specific constraints and requirements are met effectively.

Time minimization

The original time minimization problem is an infinite horizon problem. However, by revisiting the fact that $\frac{\mathrm{d}t}{\mathrm{d}\xi} = 1/\dot{\xi}$ and leveraging the spatially transformed system dynamics in (3.5), the time-minimizing problem can be reformulated as

$$g\left(x^{\Gamma}(t)\right) = T = \int_0^T dt = \int_{\xi_0}^{\xi_f} \frac{1}{\dot{\xi}(\xi)} d\xi \,. \tag{3.9}$$

The ability to reformulate the problem as a finite horizon is highly advantageous. However, it requires an analytical representation of $\dot{\xi}(\cdot)$, which, in the most general case, is not straightforward. In fact, deriving this representation is one of the contributions of this dissertation.

3.3.3 Decision-Making Formulations

Another way to distinguish egocentric methods is by considering the formulation methods used for their implementation. In the literature, egocentric methods can be clustered into three main categories, which we will now analyze.

Classical Control

Egocentric formulations align naturally with classical control theory, particularly in geometric control and path-following frameworks. Two properties make egocentricity especially appealing in this context. First, it removes explicit temporal dependence from the control objectives by expressing system dynamics in a body-fixed (local) frame. Second, it decouples tangential (along-path) and orthogonal (cross-path) motions, enabling modular controller design for guidance and stabilization. These features underpin the development of *path-following* strategies, which are widely regarded as more robust and general than traditional time-parameterized trajectory tracking [22, 23, 24, 25].

In classical control, such problems are often cast in terms of minimizing a geometric or kinematic error function defined in the egocentric frame. This leads to feedback controllers—often designed using Lyapunov theory, input-output linearization, or backstepping techniques—that regulate these errors to zero. In many cases, the problem structure allows for a modular control architecture, where the stability and performance of each controller can be analyzed independently and then composed for the full system.

Optimization-based

The appealing attributes observed with classical control can be seamlessly adopted by optimization-based approaches, where the decision-making process is transcribed into an

optimization problem solved through numerical methods. The richer geometrical representations associated with the egocentric methods, combined with the capacity to compute optimal solutions, fully expose the true potential of egocentric methods for enabling autonomy behaviors that would otherwise be unattainable. In its most generic form, this requires solving the following optimization problem:

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \quad g(\boldsymbol{x}^{\Gamma}(\tau)) \tag{3.10a}$$

$$\text{s.t.} \quad \boldsymbol{x}^{\Gamma}(0) = \boldsymbol{x_0}\,, \tag{3.10b}$$

$$\dot{\boldsymbol{x}}^{\Gamma} = f_{\Gamma}(\boldsymbol{x}^{\Gamma}(\tau), \boldsymbol{u}(\tau)), \qquad \tau \in \left[\tau_0, \tau_f\right] \tag{3.10c}$$

$$\boldsymbol{x}^{\Gamma}(\tau) \in \mathcal{X}\,, \quad \boldsymbol{u}(\tau) \in \mathcal{U}\,, \quad \tau \in \left[\tau_0, \tau_f\right]\,, \tag{3.10d}$$

where f_τ refers to the system dynamics from the egocentric perspective, $g(\boldsymbol{x}^{\Gamma}(\tau))$ is the performance criterion introduced in the previous section, and the evolution variable τ is either time t or path-parameter ξ, depending on the specific formulation of the system dynamics.

As we will see across many instances of this dissertation, depending on how (3.10) is formulated, it can either be used as a low-level prediction-based controller online or to solve very complex motion planning problems offline.

Reinforcement Learning

Egocentric representations also offer substantial advantages in the context of Reinforcement Learning (RL) [44], particularly in formulating reward functions. By expressing the agent's goal relative to its own local frame, egocentric formulations provide geometrically meaningful signals—such as advancement along a path, distance to a goal, or alignment with a reference direction—that are inherently sparse or difficult to specify in global coordinates.

This is especially beneficial in high-dimensional, continuous control domains typical of robotics. Designing reward functions based on egocentric cues leads to more informative gradients and facilitates credit assignment, improving both sample efficiency and policy performance. Notably, egocentric representations have been crucial in enabling breakthroughs in domains like video game car racing [45] and autonomous drone racing [40], where the agent must learn to make fast, precise decisions based on high-dimensional sensory input.

3.4 A motivating case-study: Time-Optimal Tunnel-Following

If we were to attempt implementing the information provided in the previous section, multiple questions would arise. Ultimately, the intent of this dissertation is to clarify these doubts and build a foundation that strengthens our current understanding of egocentric methods. However, before doing so, we want to be certain that going down the egocentric rabbit hole is worth it. For this reason, we provide an illustrative case study that exposes all the attributes that make egocentric methods so worthwhile.

To achieve this, we take a very pragmatic approach to solving a problem that requires most of the crucial ingredients necessary for the egocentric methods, yet we bypass most of the conceptual difficulties, rendering a solution that is problem-specific – it does not generalize – but still solves the problem and gives us a taste of what is achievable if we were to embrace

egocentricity. In other words, it serves as a motivating example to extend this solution to any arbitrary case study, something that we do in the upcoming chapters of this dissertation.

3.4.1 The case-study: Time-Optimal Tunnel Following for Quadrotors

To showcase the power of egocentricity, we pick a problem that is challenging enough to demonstrate the potential withheld by egocentric methods. Specifically, we focus on time-optimal navigation for quadrotors in dynamically varying environments. To understand why we choose this example, let's break down the description of this problem:

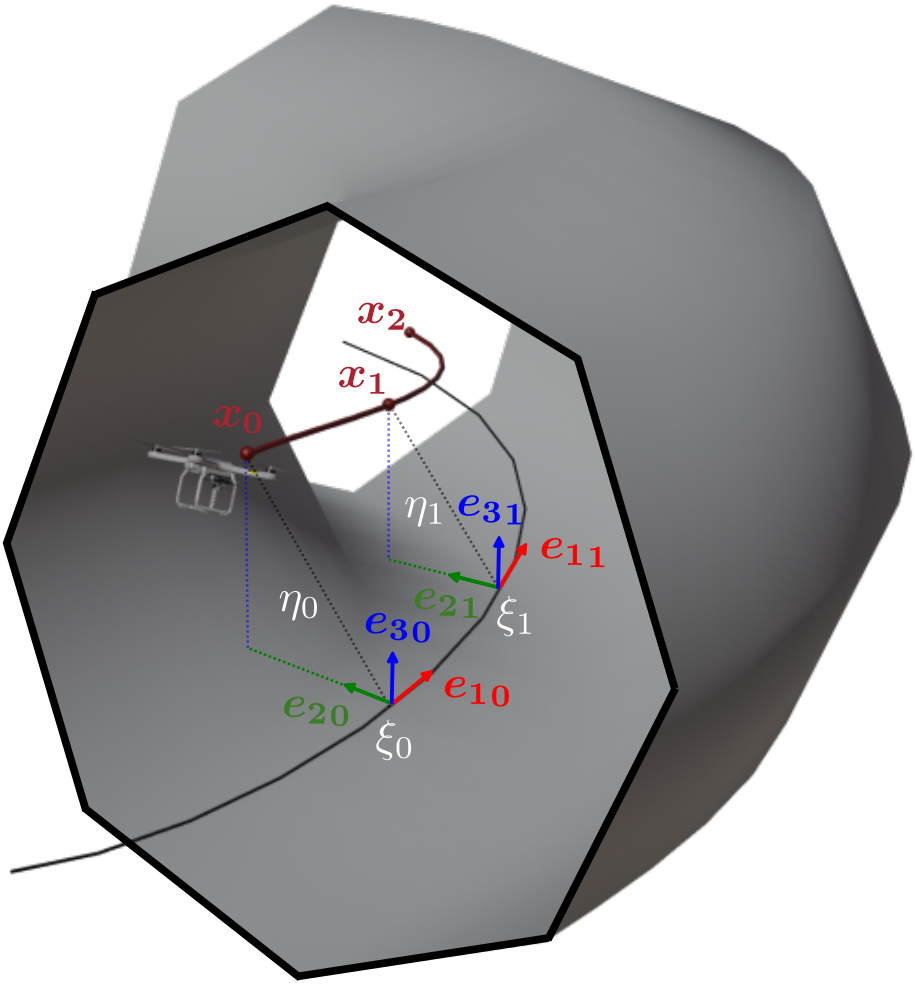


Figure 3.2 We showcase the capabilities of egocentric methods through an illustrative example of time-optimal navigation with quadrotors in dynamic environments. This example highlights the potential of egocentricity as a paradigm worth pursuing and serves as motivation for the remainder of this dissertation, which aims to refine egocentricity into a more principled and broadly applicable framework.

- **Time-Optimal navigation**: Minimum-time navigation is a critical problem with wide-ranging real-world applications—from enhancing the efficiency of automated manufacturing to improving the success rate of time-sensitive operations like search and rescue. Beyond its practical relevance, it is a well-studied and computationally

demanding challenge in algorithmic research. Most notably, it provides a valuable lens through which to examine the agility inherent in egocentric decision-making methods.

- **Quadrotors**: Their agility and capacity for highly aggressive maneuvers make quadrotors an ideal platform for research on autonomous navigation algorithms. Moreover, their operation in three-dimensional space highlights challenges unique to egocentric navigation—challenges that are often absent in two-dimensional systems like autonomous ground vehicles.
- **Dynamic Environment**: The real world is constantly evolving –either because the surrounding objects move or the perception updates–, and therefore, having the ability to adapt to these changes is a feature that cannot be overlooked.

To focus on the computation of motions, without having to worry about higher-level planning, we reduce our problem to *tunnel-following*, where a corridor within which navigation is allowed is supposed to be a priori defined. Doing so removes the need for a higher-level planner and allows us to focus on computing time-optimal motions for quadrotors within this tunnel. As mentioned before, given that we assume the environment is dynamic, this tunnel can change its shape and size while we are navigating within it. Putting everything together leads to the problem *Time-Optimal Tunnel-Following for Quadrotors* that we use as a motivation for the remainder of this thesis.

This combination of challenges yields a problem whose algorithmic development could have a high societal impact, but given its non-trivialities, it is a problem where egocentric methods excel.

Ingredients

With this example, we aim to explore the potential benefits of egocentric methods. To achieve this, we rely on several approximations that help us bypass most of the complexities that motivate this thesis. These approximations allow us to quickly gain insights into what can be accomplished by integrating egocentric principles into the design of autonomous software stacks. With this in mind, the key components of this case study can be outlined as follows:

- We approximate the computation of time-optimal motions using a **Model Predictive Controller** (MPC). In other words, rather than solving the entire problem simultaneously, we formulate a finite-horizon optimization problem, which is solved in a receding horizon fashion. This optimization problem computes the optimal motions that enable the quadrotor to navigate within a subset of the tunnel.
- We model the quadrotor system dynamics using a **spatial projection**, as described in 3.3.1. This allows us to replace Cartesian position, velocity, and acceleration with a spatial coordinate representation.
- To construct the spatial projections, we parameterize the center of the tunnel using its **arc length** and employ the **Frenet-Serret frame** to define the relative frame associated with it.
- We approximate time-optimality by **tracking a slightly infeasible constant velocity profile** $\dot{\xi} - \dot{\xi}_{\text{ref}}$, which is assumed to be known. This approach is motivated by

two key advantages: First, it encourages time-optimal behavior without dealing with the complexities of infinite horizon or spatially transformed problems, as discussed in 3.3.2. Second, it ensures numerical stability, a feature that unbounded approaches such as progress maximization lack, ultimately resulting in increased robustness and faster computation times.

Through these approximations, we gain a preliminary understanding of the potential offered by egocentric methods. To fully grasp the extent of the results, we refer to the following subsection.

Results

The results obtained, which are presented in the paper attached at the end of this document, can be categorized according to the following three key features:

- The egocentric MPC closely approximates the theoretical time-optimal trajectories calculated by offline planners. In contrast, the egocentric MPC relies solely on online computations, eliminating the need for offline calculations. These behaviors generalize across tunnels of varying shapes and sizes. We attribute this to the geometric features embedded in the egocentric representation, which incorporate the environment's geometric properties into the underlying optimization problem, thereby facilitating the computation of time-optimal motions.
- The egocentric MPC can effectively handle dynamic environments by constraining its motions to remain within the tunnel boundaries. If changes occur outside the optimization horizon, the controller's performance remains unaffected, maintaining both the solution's quality and robustness. If changes occur within the horizon, the controller can adapt to these changes, provided that a feasible solution exists.
- The same egocentric controller is capable of operating across a wide range of dynamic regimes, from very high flying speeds—over 40m/s—to stop-and-go maneuvers, all with the same tuning and hyperparameters, without requiring any adjustments. This characteristic makes it immediately deployable across a variety of problem formulations.

The combination of these properties results in a controller that approximates the performance of offline planners, can handle environmental changes, and is versatile enough to be applied across a wide range of operational modes. This ability to fully exploit the system's dynamic envelope while maximizing the available free space is what makes egocentric methods particularly appealing. Ultimately, this motivates us to explore how we can apply such methods to any arbitrary case study, without relying on the aforementioned approximations.

3.4.2 An inspiration for this thesis

The performance demonstrated in this example makes egocentric methods a promising avenue for further exploration. However, the simplifications and approximations we employed render this solution problem-specific, preventing its direct generalization. For instance, the spatial reformulation is tailored specifically to the quadrotor, and the parameterization of the tunnel relies on the centerline being parameterized by arc length and the Frenet-Serret frame. The key takeaway from this example is that egocentricity has the potential to un-

lock high-performance behaviors, potentially surpassing state-of-the-art methods. However, the implementation is highly problem-specific, raising the question: what would it take to generalize it to any problem? Attempting to answer this immediately introduces several questions, some of which are listed below:

- How can we compute $\dot{\xi}_{\text{ref}}$?
- How can we generalize the spatial projection of the quadrotor's states to apply to any dynamical system?
- How can we generalize the geometric reference beyond the arc-length parameterization and the Frenet-Serret relative frame?
- How can we generalize the tunnel description so that it is applicable to any problem, including unforeseen environments, rather than relying on predefined centerlines?

The answers to these questions are explored throughout the remainder of this dissertation. In essence, this thesis is dedicated to addressing these questions (and more), and after carefully tackling each one, it unifies existing egocentric methods into a universal framework. This framework generalizes the formulation of egocentric decision-making algorithms, making them applicable to any autonomous system.

3.5 Enclosed Publication

The case-study presented in the previous section is included as a scientific publication attached to this chapter, and it is cited in [1]. A video associated with this publication can be viewed at `https://youtu.be/Apc8MCu7Yvo`.

Towards Time-Optimal Tunnel-Following for Quadrotors

Jon Arrizabalaga and Markus Ryll

published in

IEEE International Conference on Robotics and Automation (ICRA)

2022

Contribution This paper presents a real-time control method based on Nonlinear Model Predictive Control (NMPC) for generating near-optimal trajectories for quadrotors in dynamic, constrained environments. It is the first approach to address time-optimality in time-varying spaces without the need for prior environmental knowledge or offline computation. Regarding my role in the project, I was responsible for all the stages that lead to this finding, from the mathematical derivations, to the software implementation and experiments that, ultimately, lead to the results presented in the paper.

Towards Time-Optimal Tunnel-Following for Quadrotors

Jon Arrizabalaga[1] and Markus Ryll[1]

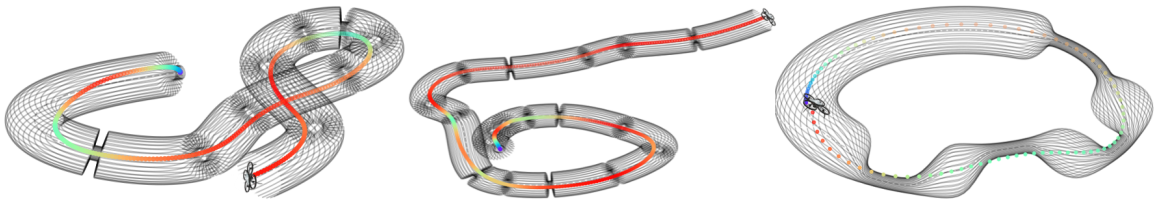


Fig. 1: A quadrotor navigates at near-time-optimal performance through three different dynamic tunnels: *AlphaPilot* (left), *AirSim* (center) and *TubeTwist* (right). The respective motions have been computed by our NMPC based real-time control method capable of handling time-variant spatial bounds.

***Abstract*— Minimum-time navigation within constrained and dynamic environments is of special relevance in robotics. Seeking time-optimality, while guaranteeing the integrity of time-varying spatial bounds, is an appealing trade-off for agile vehicles, such as quadrotors. State-of-the-art approaches, either assume bounds to be static and generate time-optimal trajectories offline, or compromise time-optimality for constraint satisfaction. Leveraging nonlinear model predictive control and a path parametric reformulation of the quadrotor model, we present a real-time control that approximates time-optimal behavior and remains within dynamic corridors. The efficacy of the approach is evaluated by simulated results, showing itself capable of performing extremely aggressive maneuvers as well as stop-and-go and backward motions.**

Video: `https://youtu.be/Apc8MCu7Yvo`

I. INTRODUCTION

Autonomous robotic systems are deployed in dynamic environments, where conditions and obstacles are time-variant. Successfully navigating under these circumstances, implies remaining within a safe and time-variant corridor [1]. Time-optimality within dynamic environments poses a particular challenge, since the minimum-time trajectory changes along with the environment.

Because of their inherit agility [2], quadrotors are frequently used in space-constrained and time-critical operations, such as delivery, surveillance, inspection, and search-and-rescue [3], [4]. These are distinguished by the necessity of being time-optimal and subjected to strong environmental changes. Pushing the physical limits of quadrotors in unknown variant environments brings up a twofold challenging trade-off: 1) quadrotors are underactuated systems, i.e., longitudinal and angular dynamics are coupled, 2) the motion aggressiveness is bounded by the dynamicity of the environment. In other words, time-optimal performance in dynamic environments consists on *finding the optimal trade-off between longitudinal and angular accelerations (1) in accordance with variant spatial bounds (2)*.

Previous works have decoupled these two features, either by assuming the environment to be invariant and tracking an offline-generated minimum-time trajectory [5], [6], limiting time-optimality to waypoint following and neglecting spatial bounds [7], or compromising time-optimality due to actuation limit relaxations [8]–[11] and collision-free constraints [12].

Given its recent applications on fast dynamical systems and its support for nonlinear system models [13], [14], Nonlinear Model Predictive Control (NMPC) is a well-suited framework for motion planning of quadrotors. NMPC can find the (locally) optimal control action considering a nonlinear cost function while incorporating constraints on nonlinear functions of states and inputs [15]. Therefore, NMPC can account for time-varying spatial bounds, while enforcing constraints in the true physical limits, i.e. thrust commands, and thus, ensuring near-time-optimality of the computed solution.

Multiple research challenges arise, including how to construct a lightweight NMPC capable of running in real-time and how its performance compares to the previously mentioned offline approaches.

Contributions

We present an NMPC-based real-time control that computes near-time-optimal quadrotor trajectories within the free space associated with a dynamic environment – denoted as *tunnel*. To the best of the authors' knowledge, this is the first real-time approach tackling time-optimality within constrained and time-variant spaces in quadrotors. In contrast to state-of-the-art methods, our solution is immediately deployable, without requiring prior environmental knowledge or offline computation, and is capable of ranging from extremely aggressive maneuvers to stopping and even reversing.

Our method is comprised of two main ingredients: 1) we perform a path-parametric reformulation of the quadrotor system dynamics, allowing to embed path properties and variant spatial constraints effectively in the optimization and 2) by solely bounding thrust commands, the NMPC is empowered to reach the true physical limits of the system.

According to empirical results, the suggested approach 1) approximates time-optimal behavior, while adapting its speed depending on the shape and size of the tunnel 2) is agnostic

[1]Autonomous Aerial Systems Lab, Department of Aerospace and Geodesy, Technical University of Munich, Germany. E-mail: `jon.arrizabalaga@tum.de` and `markus.ryll@tum.de`

Published in IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, USA, May 2022.

to tunnel changes beyond the prediction horizon, 3) can stop or navigate backwards when abrupt tunnel modifications occur within the horizon and 4) shows convergence to the tunnel if a given modification is excessively aggressive.

The remainder of this paper is structured as follows: Section II provides more details on quadrotor motion planning and spatial reformulation of dynamical systems. Section III presents the path-parametric reformulation of the quadrotor dynamics. Section IV exemplifies the complete NMPC formulation, including the tunnel's parametrization. Experimental setup and results are shown in Section V before Section VI presents the conclusions.

II. RELATED WORK

A. Time-optimal and collision-free planning

Path planning algorithms for quadrotors can be classified into two variants based on the state-space of the planning. Exploiting the property of differential flatness, all quadrotor states can be mapped into four differentially flat outputs. By approximating them to piecewise constant polynomials and imposing continuity constraints in the respective derivatives, smooth trajectories can be generated. Since this concept was presented in [8], further improvements to solve the underlying quadratic program have allowed for extremely fast computations [9], [16]. This has played a key role in standard decoupled approaches like [12], where a collision-free path provided by a global planner is further refined by a local planner running at a higher sampling rate.

The computational advantages brought by the flatness property and its respective polynomials come at the expense of 1) highly non-linear mappings from the flat outputs to thrust commands and 2) non-time-optimality inherited from the minimization of polynomials. The first suggests that applying differential flatness prevents setting the thrust constrains required for reaching physical limits. The second implies that minimizing polynomials is in direct opposition to maximizing acceleration.

Therefore, methods that account for the entire state-space are better suited to converge to minimum-time trajectories. Focusing on time-optimality when navigating through a set of locations, [5] formulated an optimization problem that parametrizes progress with dynamic waypoints through Complementary Progress Constraints (CPC). Computed paths prove to be theoretically optimal according to the physical limits of the quadrotor and capable of outperforming expert pilots. Tackling the same problem, [17] presents a Deep Reinforcement Learning based algorithm that calculates near-time-optimal solutions while dealing with uncertainty in waypoint poses. Despite these methods allow for a close approximation of the true time-optimal path, they are limited to offline computations, and thus, are intractable for real-time applications. This burden is alleviated in [7], where a Contouring Control MPC approach inspired by [18] and [19] allows for near-time-optimal performance in real-time. However, the proposed approach is limited to waypoint following and does not account for spatial bounds. Moreover, the underlying gradient in the Contouring Control necessitates additional constraints, compromising the solution's versatility and applicability for dynamic scenarios, such as when reversal motions are required.

In contrast, our approach runs in real-time, requires no prior knowledge of the environment, and is capable of performing aggressive maneuvers as well as stopping and navigating backwards. These features are attributed to a coordinate transformation that enable a more precise embedding of the environment's geometric properties into the NMPC's underlying optimization problem.

B. Path-parametric reformulation

Benefits of transforming time-dependent dynamics into spatial-dependent were first presented for robot systems in [20], [21]. Extensions of this reformulation to planar vehicles proved its ability to trade-off between reference tracking and obstacle avoidance [22], [23]. Leveraging this reparameterization and exploiting advances in embedded optimization solvers, [24] developed a real-time NMPC for miniature racing cars that could approximate time-optimal performance. In [25], further modifications in the spatial representation of the plant model allowed including online obstacle avoidance, while still being near-time-optimal. Focusing on robot manipulators, [26] extended the path-parametric reformulation to all three dimensions and presented an NMPC capable of trading-off between tracking accuracy and speed.

In the context of quadrotors, by combining path-parametrization with differential flatness, [10] presented an efficient optimization problem for time-optimal trajectory planning of two-dimensional models in cluttered environments. In [27] the spatial reformulation was implemented to all three dimensions, allowing to decouple the tangential and transverse dynamics of the quadrotor. Similarly, [6] benefited from a three-dimensional reparameterization for offline computation of minimum-time and collision-free trajectories.

Previous work suggests that, in contrast to planar vehicles, the realm of quadrotors has not fully leveraged the capabilities inherited in path-parametrization. We deem to close this gap by combining time-optimality techniques from the aforementioned two-dimensional solutions with a complete spatial conversion of the quadrotor dynamics.

III. PATH-PARAMETRIC MODEL

A. Quadrotor Model

The quadrotor is modeled as a six degree-of-freedom rigid body. Its states are described in the world-frame with $\boldsymbol{x} = [\boldsymbol{p}_{\mathrm{W}}, \boldsymbol{v}_{\mathrm{W}}, \boldsymbol{q}_{\mathrm{W}}, \boldsymbol{\omega}_{\mathrm{W}}]$, where $\{\boldsymbol{p}_{\mathrm{W}}, \boldsymbol{v}_{\mathrm{W}}, \boldsymbol{\omega}_{\mathrm{W}}\} \in \mathbb{R}^3$ refer to the position, velocity and body rates, while the attitude is represented by a unit quaternion $\boldsymbol{q}_{\mathrm{W}} \in \mathbb{R}^4$. The collective thrust and torques in body-frame are the inputs with $\{\boldsymbol{f}_{\mathrm{B}}, \boldsymbol{\tau}_{\mathrm{B}}\} \in \mathbb{R}^3$. Since frame subscripts remain constant, they will be dropped. The dynamics of the system are as common

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \quad (1a) \qquad \dot{\boldsymbol{v}} = \boldsymbol{g} + \frac{1}{m}\mathrm{R}(\boldsymbol{q})\boldsymbol{f} \quad (1b)$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\Lambda(\boldsymbol{\omega})\boldsymbol{q} \quad (1c) \qquad \dot{\boldsymbol{\omega}} = \mathrm{J}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathrm{J}\,\boldsymbol{\omega}\right) \quad (1d)$$

with m, J and $\boldsymbol{g}$ being the quadrotor's mass, moment of inertia matrix and gravity, while Λ and R refer to the skew-symmetric matrix of the body rates and rotation matrix of the quaternion. Body forces $\{\boldsymbol{f}, \boldsymbol{\tau}\}$ can be further mapped to single rotor thrust commands $\boldsymbol{\eta} = [\eta_1, \eta_2, \eta_3, \eta_4]$ according to

$$\boldsymbol{f} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} \eta_i \end{bmatrix} \quad \text{and} \quad \boldsymbol{\tau} = \begin{bmatrix} l/\sqrt{2}(\eta_1 + \eta_2 - \eta_3 - \eta_4) \\ l/\sqrt{2}(-\eta_1 + \eta_2 + \eta_3 - \eta_4) \\ c_\tau(\eta_1 - \eta_2 + \eta_3 - \eta_4) \end{bmatrix}$$

where l is the arm-length and c_τ the force-torque mapping constant of the quadrotor.

B. Path-Parametric Reformulation

To embed the path characteristics into the system dynamics, we reformulate the equations of motions for position (1a) and velocity (1b). This is done in two steps: first we describe the path using Frenet-Serret's theorem, and then perform a spatial reformulation. When doing so, we will combine ideas from [6] and [26].

As a starting point, we introduce a *progress variable* s exemplifying the arc-length of a curve and a C^∞ continuous *progress-function* $\boldsymbol{\gamma}(s)$ that maps s to its respective three-dimensional position in the Euclidean space. Thus, the path can be defined as $\Gamma = \{\boldsymbol{\gamma}(s) \in \mathbb{R}^3 \mid s \in [0, L]\}$.

Discretizing Γ into M points and assigning each one an orthonormal basis in the Frenet-Serret coordinate system allows for describing the pose as a function of the path's curvature κ, torsion σ and progress-function $\boldsymbol{\gamma}$ (see Fig. 2). These are related by Frenet-Serret's theorem [28]:

$$\mathfrak{t} = \boldsymbol{\gamma}', \quad \mathfrak{n} = \boldsymbol{\gamma}''/||\kappa||_2\,, \quad \mathfrak{b} = \mathfrak{t} \times \mathfrak{n}, \qquad (2a)$$

$$\mathfrak{t}' = \kappa\,\mathfrak{n}, \quad \mathfrak{n}' = -\kappa\,\mathfrak{t} + \sigma\,\mathfrak{b}\,, \quad \mathfrak{b}' = -\sigma\,\mathfrak{n} \qquad (2b)$$

where $\mathfrak{t}$, $\mathfrak{n}$ and $\mathfrak{b}$ are the tangent, normal and binormal components of the Frenet-Serret frame $\{\mathrm{FS}\}$ and $(\cdot)'$ represents the derivative with respect to the progress variable s. Notice that all terms in (2) are continuous and uniquely dependent on s, and thus, we can approximate them by third order B-splines.

Putting together all three components, the orientation along the path is defined by the Frenet-Serret rotation matrix $\mathrm{R_{FS}}(s) = [\mathfrak{t}(s), \mathfrak{n}(s), \mathfrak{b}(s)]$. Assuming that the quadrotor is located in position $\boldsymbol{p}(t)$, the progress of the closest point on the path Γ is defined as

$$s^*(t) = \arg\min_s \frac{1}{2}||\boldsymbol{d}_\mathrm{W}(t)||, \quad \boldsymbol{d}_\mathrm{W}(t) = \boldsymbol{p}(t) - \boldsymbol{\gamma}(s)\,. \quad (3)$$

From now onward the dependency of $s^*(t)$ on time will be dropped. Translating the distance vector to the Frenet-Serret frame associated to the closest point defines $\boldsymbol{d}_\mathrm{FS}(t) = \mathrm{R_{FS}}(s^*)^\intercal \boldsymbol{d}_\mathrm{W}(t)$, whose first row, i.e. the tangent component, is zero. Since the remaining two elements are the perpendicular projections of the distance from path Γ, they will be

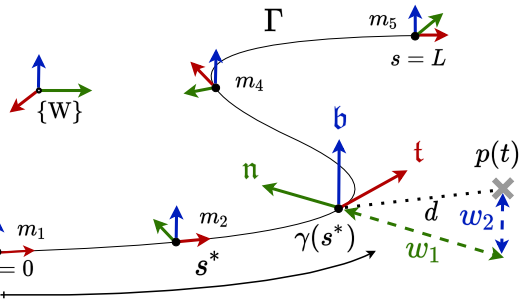


Fig. 2: Path representation according to progress variable s and Frenet-Serret frames. The distance d between the quadrotor $p(t)$, depicted as a gray cross, and the closest point on the path $\gamma(s^*)$ is projected onto the transverse coordinates w_1 and w_2. For readability, the illustration assumes the torsion along the entire path to be zero.

named *transverse coordinates* $\boldsymbol{w} = [w_1, w_2]$. Consequently, the distance is equivalent to $\boldsymbol{d}_\mathrm{FS}(t) = [0, w_1(t), w_2(t),]$ with

$$w_1(t) = \mathfrak{n}(s^*)\boldsymbol{d}_\mathrm{W}(t),\ w_2(t) = \mathfrak{b}(s^*)\boldsymbol{d}_\mathrm{W}(t)\,. \quad (4)$$

Either from applying the first optimality condition to (3) as in [26] or from rewriting equation (1a) with the transverse coordinates and derivating with respect to time as in [6], the velocity of the progress respective to the closest point on path Γ can be expressed as

$$\dot{s}^*(t) = \frac{\mathfrak{t}(s^*)^\intercal \boldsymbol{v}(t)}{1 - \kappa(s^*)w_1(t)} \quad (5a)$$

and combining it with (4) leads to

$$\dot{w}_1(t) = \mathfrak{n}(s^*)\boldsymbol{v}(t) + \sigma(s^*)\dot{s}^*(t)w_2(t)\,, \quad (5b)$$

$$\dot{w}_2(t) = \mathfrak{b}(s^*)\boldsymbol{v}(t) - \sigma(s^*)\dot{s}^*(t)w_1(t)\,. \quad (5c)$$

For a detailed derivation of (5b) and (5c), please refer to [6]. The singularity in (5a) can be circumvented by requiring $1 - \kappa(s^*)w1(t) > 0$, or equivalently $r(s^*) > w1(t)$, i.e. the normal component of the distance between the quadrotor and the path needs to be smaller than the radius of the curve.

The equations of motion for the tangential and transverse coordinates in (5) can already replace the position dynamics in (1a). However, to fully embed the path properties within the system dynamics, the integration chain of the path coordinates should be extended up to the degree where the input forces appear. In the case of a quadrotor, the collective thrust arises in the longitudinal acceleration (1b). Therefore, we are interested in differentiating the path coordinates up to the second order. This is done by taking the time derivatives of the equations in (5):

$$\ddot{s}^*(t) = \frac{\mathfrak{t}\dot{\boldsymbol{v}} + \dot{\mathfrak{t}}\boldsymbol{v}}{1 - \kappa w_1} + \mathfrak{t}\boldsymbol{v}\left[\frac{w_1\dot{\kappa} + \dot{w}_1\kappa}{(1 - \kappa w_1)^2}\right] \quad (6a)$$

$$\ddot{w}_1(t) = \mathfrak{n}\dot{\boldsymbol{v}} + \dot{\mathfrak{n}}\boldsymbol{v} + \dot{\sigma}\dot{s}^* w_2 + \sigma\ddot{s}^* w_2 + \sigma\dot{s}^*\dot{w}_2 \quad (6b)$$

$$\ddot{w}_2(t) = \mathfrak{b}\dot{\boldsymbol{v}} + \dot{\mathfrak{b}}\boldsymbol{v} - \dot{\sigma}\dot{s}^* w_1 - \sigma\ddot{s}^* w_1 - \sigma\dot{s}^*\dot{w}_1 \quad (6c)$$

where dependencies have been omitted for clarity. The time derivatives of curvature, torsion and Frenet-Serret coordinates can be calculated according to the chain rule $\dot{(\cdot)} = \frac{\partial(\cdot)}{\partial s}\frac{\partial s}{\partial t} = (\cdot)'\dot{s}^*$, while the longitunal acceleration $\dot{\boldsymbol{v}}$ is known from (1b). The velocity is the only variable that remains for being translated to spatial coordinates. To this end we

reformulate the position as $\boldsymbol{p}(t) = \boldsymbol{\gamma}(s^*) + \mathrm{R_{FS}}(s^*)\boldsymbol{d}_{\mathrm{FS}}(t)$ and differentiate with respect to time:

$$\boldsymbol{v}(t) = \dot{\boldsymbol{\gamma}} + \dot{\mathrm{R}}_{\mathrm{FS}}\, \boldsymbol{d}_{\mathrm{FS}} + \mathrm{R}_{\mathrm{FS}}\, \dot{\boldsymbol{d}}_{\mathrm{FS}} \tag{7}$$

whose terms can be obtained by applying the aforementioned chain rule. Finally, by combining (1b) with (6) and (7), the euclidean position and velocity states can be replaced by the progress variable, transverse coordinates and its respective derivatives, i.e. $\boldsymbol{x} = [s^*, \boldsymbol{w}, \dot{s}^*, \dot{\boldsymbol{w}}, \boldsymbol{q}, \boldsymbol{\omega}]$. From here on to simplify the notation s^* will be referred to as s.

IV. PROGRESS-MAXIMIZATION NMPC

The optimal control problem (OCP) addressing minimum-time navigation within dynamic corridors is formulated as:

$$\min_{\boldsymbol{x}(.),\boldsymbol{u}(.),T} \; T \tag{8a}$$
$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_T \tag{8b}$$
$$T \geq 0 \tag{8c}$$
$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad t \in [0,\, T] \tag{8d}$$
$$\underline{\boldsymbol{u}} \leq \boldsymbol{u}(t) \leq \overline{\boldsymbol{u}}, \quad t \in [0,\, T] \tag{8e}$$
$$\phi\left(\boldsymbol{w}(t), \xi(t)\right) \leq 0 \quad t \in [0,\, T] \tag{8f}$$

where $\boldsymbol{x} = [s, \boldsymbol{w}, \dot{s}, \dot{\boldsymbol{w}}, \boldsymbol{q}, \boldsymbol{\omega}]$ and $\boldsymbol{u} = [\boldsymbol{\eta}]$ refer to the quadrotor's states and controls, while f is the first-principles-based model derived in Section III. Other than bounding thrust commands by $\underline{\boldsymbol{u}}$ and $\overline{\boldsymbol{u}}$, we limit the navigation space to a time-varying set $\xi(t)$ according to function ϕ.

To approach online time-optimal navigation, we approximate the OCP (8) by a Nonlinear Program (NLP) according to the multiple-shooting approach [29] in which the optimization horizon T is split into N sections with constant decision variables. Similarly to [24] and [25], by tracking a slightly infeasible reference s_{ref} and its respective velocity $\dot{s}_{\mathrm{ref}}$, we incite time-optimal behavior. For a given progress velocity, the respective arc-length can be defined as $s_{\mathrm{ref},k} = s_0 + \frac{\dot{s}_{\mathrm{ref,k}}}{T}\frac{k}{N}$ for $k = 0, \cdots, N$, and thus $\boldsymbol{x}_{\mathbf{ref}} = [s_{\mathrm{ref}}, \boldsymbol{c}_{\boldsymbol{w}}(s_{\mathrm{ref}}), \dot{s}_{\mathrm{ref}}, \mathbf{0}]$, where $\boldsymbol{c}_{\boldsymbol{w}}$ refers to the transverse coordinates of the tunnel's center line.

To leverage the computational advantages of the Gauss-Newton Hessian approximation, the cost function is implemented in a quadratic least-squares fashion. When doing so, the tractable NLP that is solved at every NMPC iteration and approximates OCP (8) is given by:

$$\min_{\substack{\boldsymbol{x}_0,\cdots,\boldsymbol{x}_N,\\ \boldsymbol{u}_0,\cdots,\boldsymbol{u}_N}} \sum_{k=0}^{N-1} ||\boldsymbol{x}_k - \boldsymbol{x}_{\mathbf{ref},\,k}||^2_Q + ||\boldsymbol{u}_k||^2_R + ||\boldsymbol{x}_N - \boldsymbol{x}_{\mathbf{ref},\,N}||^2_{Q_N} \tag{9a}$$
$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_c \tag{9b}$$
$$\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \boldsymbol{u}_k, \Delta t), \quad k = 0, \cdots, N-1 \tag{9c}$$
$$\underline{\eta} \leq \eta_{i,\,k} \leq \overline{\eta}, \quad \substack{k=0,\cdots,N-1,\\ i=1,\cdots,4} \tag{9d}$$
$$\phi\left(\boldsymbol{w}_k, \xi(s_k)\right) \leq 0, \quad k = 0, \cdots, N-1 \tag{9e}$$

where $\boldsymbol{x}_c$ is the quadrotor's current state, F is a numerical integrator for the spatial quadrotor dynamics and Q, R, Q_N are weighting matrices. Control commands are softened by appending rotor-thrusts to the state vector $\boldsymbol{x} = [s, \boldsymbol{w}, \dot{s}, \dot{\boldsymbol{w}}, \boldsymbol{q}, \boldsymbol{\omega}, \boldsymbol{\eta}]$ and assigning their respective derivatives to the system's inputs $\boldsymbol{u} = [\dot{\boldsymbol{\eta}}]$. To guarantee feasibility, constraint (9e) is relaxed according to slack variables that are linearly and quadratically penalized in the cost function.

As stated in (8f) and (9e), to ensure that the quadrotor stays within an arbitrary geometry, we define a function ϕ. The loss of curvature inherited from the linearization of non-linear constraints might compromise convergence in the optimization problem [30]. Therefore, when formulating spatial bounds in ϕ, linear representations are preferred over non-linear ones. Without loss of generality, we approximate the tunnel to a polyhedron of n_p sides, whose inner space can be represented as the following convex set:

$$\Omega = \{s, w_1, w_2 \mid s \in [0, L] \wedge\ \boldsymbol{a}(s)w_1 + \boldsymbol{b}(s)w_2 < \boldsymbol{c}(s)\}\,,$$

where $\{\boldsymbol{a}(s), \boldsymbol{b}(s), \boldsymbol{c}(s)\} \in \mathbb{R}^{n_p}$ describe the polyhedron at a given progress $s \in [0, L]$. Putting the inequality in the form of (9e), results in

$$\phi := \xi(s) \begin{bmatrix} \boldsymbol{w} \\ -1 \end{bmatrix} \leq \mathbf{0}\,,$$

with $\xi(s) := [\boldsymbol{a}(s), \boldsymbol{b}(s), \boldsymbol{c}(s)] \in \mathbb{R}^{n_p \times 3}$, i.e., each row in the inequality refers to a plane of the polyhedron. In a similar manner to [25], we propose to embed the tunnel's bounds into ξ by n_p independent cubic polynomials, each with $k_\tau N - 1$ sections:

$${}^{i}\xi(s) = \sum_{n=0}^{k_\tau N-1} \sum_{p=0}^{3} {}^{i}_{n}\alpha_{\,p}\, s^p, \qquad i = 1, 2, 3 \tag{11}$$

where k_τ is the *tunnel fidelity factor* and i refers to the column of ξ. Coefficients ${}^{i}_{n}\alpha_{0,1,2,3}$ are computed by imposing continuity on first derivatives between two subsequent polynomials, n and $n+1$, and using the start and final radius of the tunnel's section n. This yields a C^2 continuous constraint. To account for time-variance, all $12 n_p\,(k_\tau N - 1)$ coefficients are updated at every NMPC iteration, as denoted in (9e).

V. EXPERIMENTS

To evaluate our approach, we split the experimental analysis into three parts. First, we compare it to offline computed time-optimal trajectories in three different benchmarking tracks. Second, we test the method's ability to deal with tunnel changes outside the prediction horizon, and third, we increase the challenge with fully dynamic tunnels by also allowing modifications within the horizon.

A. Experimental Setup

We solve the NLP (9) using the optimal control framework ACADOS [31] by a sequential quadratic programming (SQP) method. To account for real-time applicability, we employ the real-time iteration variant (SQP-RTI) [32], [33], where solving for suboptimal controls enables fast enough computations. The underlying quadratic programs are solved by exploiting their multi-stage structure with HPIPM [34]. To integrate the dynamics, an explicit 4th-order Runge-Kutta method is used.

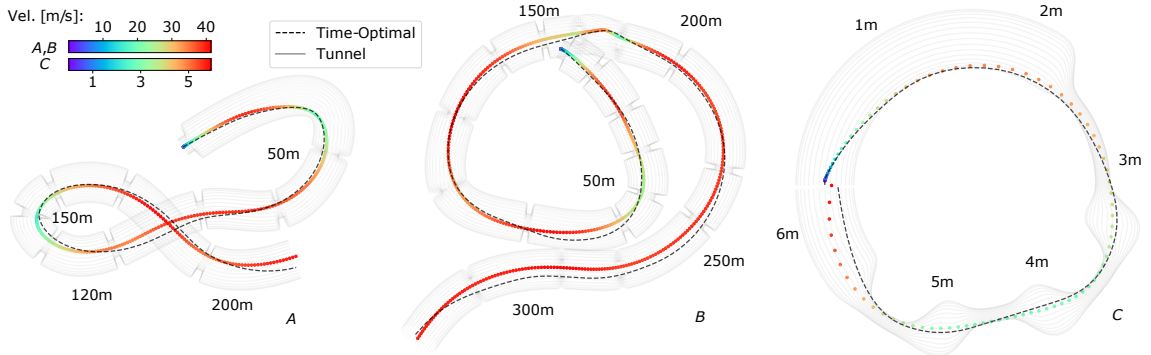


Fig. 3: Top view comparison of trajectories obtained from our path-parametric NMPC-based real-time control against the time-optimal trajectory in three different tracks: A) AlphaPilot, B) AirSim, C) TubeTwist. The progress stamps might help to visualize the scale of each track. For a full three dimensional perspective see Fig. 1.

We choose a prediction horizon of $1\,\mathrm{s}$ with 50 shooting nodes, resulting in a sampling time of $20\,\mathrm{ms}$ on a Intel Core i7-10850H notebook running Ubuntu 18.04. For all evaluations the weighting matrices are kept constant:

$$Q = \text{blkdiag}(10, 10^{-10}I_2, I_3, 10^{-10}I_7, 10^{-3}I_4), \quad R = I_4$$
$$Q_N = \text{blkdiag}(10^{-10}I_3, 1, 10I_2, 10^{-10}I_7, 10^{-3}I_4)$$

where I_n refers to an $n \times n$ identity matrix. States respective to the path coordinates and their derivatives along with rotor-thrusts are highly penalized, while the remaining diagonal values are kept small for numerical stability.

The B-Splines corresponding to the curvature, torsion, progress-function and Frenet-Serret frame components are computed by splitting the path into $30\,\mathrm{cm}$ intervals. We parametrize the tunnel as a four-sided polyhedron $n_p = 4$ with a fidelity factor of $k_\tau = 10$. Notice that the tunnels depicted in Figs. 1 and 3 have been computed with $n_p = 32$ and are intended for visualization purposes.

Regarding the quadrotor model, we use the same racing configuration as in [5], with thrust-to-weight ratio 6.4, mass $0.76\,\mathrm{kg}$, diagonal matrix inertia $[3, 3, 5]\ \mathrm{g\,m^2}$, arm-length $0.17\,\mathrm{m}$ and rotor constant $c_\tau = 0.01$.

B. Baseline Invariant Tunnels

We first compare the performance of our solution to the time-optimal trajectory when navigating along three different tracks: two racetracks, from the *AlphaPilot* [35] and *AirSim* [36] challenges, and a custom-made track named *TubeTwist*. Their differences on shape and scale allow for testing the versatility of the presented method. AlphaPilot and AirSim require navigating at very high speeds within wide-open spaces, while TubeTwist features flights within highly constrained, short and curled tunnels. In other words, the NMPC needs to trade-off between a local perspective required for dealing with short-term changes in narrow tunnels, such as TubeTwist, and a global perspective for approximating time-optimal behavior in long and wide tunnels, such as AlphaPilot or AirSim.

For TubeTwist we choose a reference progress velocity $\dot{s}_{\text{ref}}$ of $10\,\mathrm{m/s}$. Its tunnel has a nominal radius of $20\,\mathrm{cm}$ and contains two chicanes where it narrows down to $5\,\mathrm{cm}$ and deviates up to $10\,\mathrm{cm}$ from the center line. Regarding AlphaPilot and AirSim, we assign $\dot{s}_{\text{ref}}$ to be $40\,\mathrm{m/s}$ and a $10\,\mathrm{m}$ wide tunnel has been fitted to the vertices of their respective $3\,\mathrm{m}$ side-length gates. The solutions of our method when navigating along the three tunnels are presented in Figs. 1 and 3, while the respective lap times are summarized in Table I.

TABLE I: Lap times for baseline invariant tunnels shown in Figs. 1 and 3.

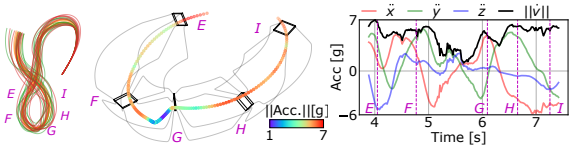
Method	AlphaPilot [s]	Airsim [s]	TubeTwist [s]
Time-Optimal	6.38	9.06	1.23
SQP	6.48	9.44	1.6
SQP-RTI	6.58	9.58	1.86

The *time-optimal* values have been computed offline by solving OCP 8 in a multiple-shooting fashion with an interior-point-based solver [37]. The corresponding solutions set a theoretical lower bound that can only be matched by a full exploitation of the actuation in a bang-bang manner. Reductions with respect to the times presented in [17] are attributed to absence of model-plant mismatch, such as drag, and the assumption that the quadrotor is a point mass capable of passing through the gates by tangentially touching their borders.

Lap times under SQP feature the converged optimal solution of NLP (9). Since the penalization of thrust commands is embedded into the NLP's formulation, bang-bang controls will by definition be suboptimal, and thus its lap times will always exceed the time-optimal ones. Seeking real-time applicability, SQP-RTI relaxes the convergence condition, leading to an average increase of $0.45\,\mathrm{s}$ across all tracks with respect to the time-optimal solution. However, in contrast to the optimal approaches, an averaged computation time of $21\,\mathrm{ms}$ allows for its real-time implementation.

C. Navigating Dynamic Tunnels

To account for more realistic scenarios with changing environments and conditions, we extend the preceding subsection by navigating along tunnels that are variant outside the prediction horizon. To this end, we introduce uncertainty in the position and size of the nominal AlphaPilot gates. The resultant tunnels have been grouped into three difficulty levels, based on the percentage of gates in the extreme configuration, i.e. minimum size and maximum displacement from the center line.



(a) (b) (c)

Fig. 4: *(a)* Top view of trajectories on different difficulty levels of dynamic AlphaPilot track, each evaluated for ten samples. From easy to difficult: orange, green, red. *(b)* 3D perspective of exemplary aggressive maneuver in hairpin of maximum difficulty track. *(c)* Single components and norm of acceleration while performing the maneuver.

We set the minimum and maximum gate side-lengths to $1.5\,\mathrm{m}$ and $3\,\mathrm{m}$, while allowing for up to $4\,\mathrm{m}$ displacements from the center line. At an increasing order of difficulty, the levels are defined as 25%, 75% and 100% and each of them is evaluated for ten different samples. As a performance metric, the corresponding time-optimal solutions have been computed. The trajectories of our method, when navigating through all randomized tracks, are depicted in Fig. 4a.

The averaged lap times listed in Table II show that the gap between SQP-RTI and time-optimal solution remains consistent regardless of the difficulty level. This proves that our method is agnostic to tunnel changes beyond the prediction horizon, making it suitable for navigation in time-varying tunnels that adapt to dynamic environments.

The nature of the maximum difficulty tracks, where all gates are of the smallest size and some of them are misaligned by $8\,\mathrm{m}$, combined with the fact that the reference progress velocity has been maintained constant with respect to the original experiment at $\dot{s}_{\mathrm{ref}} = 40\mathrm{m/s}$, results in highly challenging – close to infeasible – tracks. Figs. 4b and 4c show an exemplary maneuver that depicts this phenomenon, in which the longitudinal and lateral accelerations vary by $13\,\mathrm{g}$ within $3\,\mathrm{s}$. Given that SQP-RTI finds an approximate solution of NLP (9), navigating so near to the limit raises the risk of entering an infeasible state, where the only way out is to exit the tunnel by overusing the slack variables of constraint (9e). As summarized in the third column of Table II, this only happens for a single track at the maximum difficulty level and can be overcome by slightly lowering the reference progress velocity $\dot{s}_{\mathrm{ref}}$.

TABLE II: Averaged lap times and number of excursions for ten samples at three different difficulty levels with $\dot{s}_{\mathrm{ref}} = 40\mathrm{m/s}$. Values inside the parenthesis refer to the difference with respect to the time-optimal solution.

Diff. Level [%]	Avg. lap time [s]	# Excursions [/10]
25	6.78 (+0.21)	0
75	7.39 (+0.38)	0
100	7.5 (+0.38)	1

D. Handling Abrupt Tunnel Changes

Agile navigation within highly dynamic environments benefits from the ability to modify the spatial bounds within the prediction horizon. Therefore, in this subsection we study the robustness of our approach when abruptly altering the tunnel at different locations along the horizon.

To this end, we design an experiment in which the quadrotor navigates along a $2\,\mathrm{m}$ wide and $20\,\mathrm{m}$ long straight tunnel that suddenly shrinks to $1\,\mathrm{m}$ in width and vertically displaces by $2.5\,\mathrm{m}$, leaving a $50\,\mathrm{cm}$ transition gap. The abrupt change

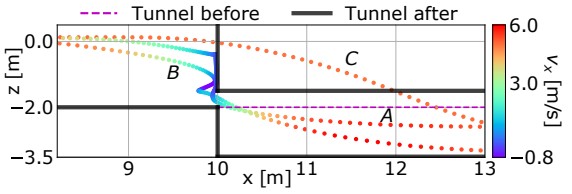


Fig. 5: Trajectories when navigating at $5\,\mathrm{m/s}$ along a $2\,\mathrm{m}$ wide and $20\,\mathrm{m}$ long straight tunnel that is abruptly modified at three different locations of the prediction horizon: A,B,C for $5\,\mathrm{m}$, $3\,\mathrm{m}$ and $1\,\mathrm{m}$. The upper tunnel bound of the first section ($x < 10\,\mathrm{m}$) is not visible. The dashed line represents the original second section, while the final tunnel shows in black.

is conducted in three different locations evenly distributed along the prediction horizon while navigating at $5\,\mathrm{m/s}$, at a distance of $1\,\mathrm{m}$, $3\,\mathrm{m}$ and $5\,\mathrm{m}$ from the quadrotor. Resultant trajectories are depicted in Fig. 5 and the corresponding navigation times are summarized in Table III.

For case *A*, where the change occurs at the last node of the horizon at a distance of $5\,\mathrm{m}$, the quadrotor has enough time to plan a conservative maneuver – stop and descent vertically – resulting in a safe passage through the narrow gap. In scenario *B*, the tunnel modification is moved to the center of the horizon, decreasing the reaction distance to $3\,\mathrm{m}$ and forcing the quadrotor to come to a complete stop and conduct a reverse motion. Finally, in *C* the tunnel is altered just $1\,\mathrm{m}$ in front of the quadrotor, making it impossible to anticipate for the abrupt change and obliging it to skip the transition.

These results suggest that our method can account for aggressive tunnel modifications within the prediction horizon. When doing so, it has demonstrated the ability to perform not only stop-and-go maneuvers, but also backward motions. Moreover, when confronted with modification excesses, it has shown itself capable of returning to the tunnel.

TABLE III: Lap times and excursions when abruptly modifying the tunnel within the horizon. Second column values refer to the prediction node and distance from the quadrotor at which the tunnel change occurs.

Label	Abrupt change location Distance [m] – Node [/50]	Lap time [s]	No excursion
A	5 – 50	7.66	✓
B	3 – 30	5.3	✓
C	1 – 10	4.38	✗

VI. CONCLUSION

In this work, we proposed a real-time control for quadrotors, capable of approximating time-optimality within variant spatial constraints. For this purpose, a path-parametric reformulation of the dynamics has allowed for efficiently embedding the spatial properties into an NMPC-based control scheme. The presented solution has been validated in three different tracks of varying shapes and sizes, each with its own set of challenges. Experimental results demonstrate that our solution not only converges to time-optimality while navigating at high speeds – over $40\,\mathrm{m/s}$ – and performing aggressive maneuvers – up to $7\,\mathrm{g}$ –, but it also is eligible to stop or reverse if spatial bounds are abruptly modified. Lastly, the fact that the controller's tuning was kept constant throughout all evaluations, emphasizes its applicability and versatility to a wide range of scenarios and dynamic regimes.

REFERENCES

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[2] E. Kaufmann, A. Loquercio, R. Ranftl, M. Mueller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *RSS: Robotics, Science and Systems*, 2020.

[3] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *Ieee Access*, vol. 7, pp. 48 572–48 634, 2019.

[4] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.

[5] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight." *Science Robotics*, vol. 6, no. 56, 2021.

[6] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2017.

[7] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for near-time-optimal quadrotor flight," *arXiv preprint arXiv:2108.13205*, 2021.

[8] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[9] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics research*. Springer, 2016, pp. 649–666.

[10] W. Van Loock, G. Pipeleers, and J. Swevers, "Time-optimal quadrotor flight," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 1788–1792.

[11] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[12] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1476–1483.

[13] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[14] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1398–1404.

[15] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[16] D. Burke, A. Chapman, and I. Shames, "Generating minimum-snap quadrotor trajectories really fast," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1487–1492.

[17] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," *arXiv preprint arXiv:2103.08624*, 2021.

[18] J. Ji, X. Zhou, C. Xu, and F. Gao, "Cmpcc: Corridor-based model predictive contouring control for aggressive drone flight," in *Experimental Robotics*, B. Siciliano, C. Laschi, and O. Khatib, Eds. Cham: Springer International Publishing, 2021, pp. 37–46.

[19] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[20] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, 1987.

[21] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[22] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proceedings of the 11th international symposium on advanced vehicle control*, no. 2, 2012, pp. 1–6.

[23] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 4136–4141.

[24] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear mpc in real-time," in *53rd IEEE conference on decision and control*. IEEE, 2014, pp. 2505–2510.

[25] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 324–14 329, 2020.

[26] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 477–482.

[27] S. Kumar and R. Gill, "Path following for quadrotors," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 2075–2081.

[28] W. Kühnel, *Differential geometry*. American Mathematical Soc., 2015, vol. 77.

[29] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[30] R. Verschueren, N. van Duijkeren, R. Quirynen, and M. Diehl, "Exploiting convexity in direct optimal control: a sequential convex quadratic programming method," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 1099–1104.

[31] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl, "Towards a modular software package for embedded optimization," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 374–380, 2018.

[32] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[33] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[34] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[35] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *RSS: Robotics, Science and Systems*, 2020.

[36] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "Airsim drone racing lab," in *NeurIPS 2019 Competition and Demonstration Track*. PMLR, 2020, pp. 177–191.

[37] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

Part II

Formulation of Egocentric Methods

Having established the importance of egocentric decision-making, we now outline the key elements that will enable its generalization across a broad range of autonomy applications. These elements are grouped into three main areas, each explored in a separate chapter. In Chapter 4, we examine the projection of system dynamics onto the relative frame. Chapter 5 focuses on the computation of the relative frame and its underlying geometric reference. Finally, in Chapter 6, we address the parameterization of the safe or collision-free space from an egocentric perspective.

4 Embracing Egocentricity via Spatial Projections

In the previous section, we introduced the appeal of spatial coordinates for designing planning and control algorithms. These so-called spatial coordinates are central to such approaches, as they enable autonomy algorithms capable of complex behaviors.

To access these coordinates, motion must be represented relative to an appropriate reference frame. In the quadrotor example from the previous chapter, we assumed that the geometric reference was parameterized by arclength and that the relative frame followed the Frenet-Serret formulation. However, beyond the desire to remove these assumptions, the Frenet-Serret frame itself— as we will see in the next chapter— poses significant challenges. This leads us to ask:

> *How can we represent the motion of a dynamical system without relying on any path parameterization or relative frame?*

This chapter addresses this question, culminating in our first contribution: a spatial parameterization of system dynamics that is fully independent of any underlying path parameterization or reference frame.

We tackle this in three structured steps. First, we introduce a formal definition of spatial states, offering an alternative to Cartesian coordinates. Next, we derive the equations of motion for this representation without imposing any assumptions on the underlying path parameterization. This guarantees full consistency with the methods described earlier and reveals that existing formulations in the literature are merely special cases of our derivation. Finally, we examine how the parametric terms in these equations naturally align with the algorithms introduced in the preceding section.

4.1 Spatial states: An alternative to Cartesian coordinates

As introduced in (3.1), we examine time-continuous (non)linear dynamical systems expressed as

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{4.1a}$$

with state vector $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and control input vector $\boldsymbol{u} \in \mathbb{R}^{n_u}$. The system's spatial position in three-dimensional Euclidean space, represented by Cartesian coordinates $\boldsymbol{p}^W(t) \in \mathbb{R}^3$ relative to the world reference frame $(\cdot)_W$, is determined through a (non)linear transformation function h, defined as

$$\boldsymbol{p}^{\mathrm{W}}(t) = h(\boldsymbol{x}(t))\,. \tag{4.1b}$$

In order to map the system dynamics (4.1a) onto the reference curve Γ defined in (2.3), we employ *spatial coordinates* as an alternative representation to the Cartesian framework shown in (4.1b). This approach separates the system's translational movement into two

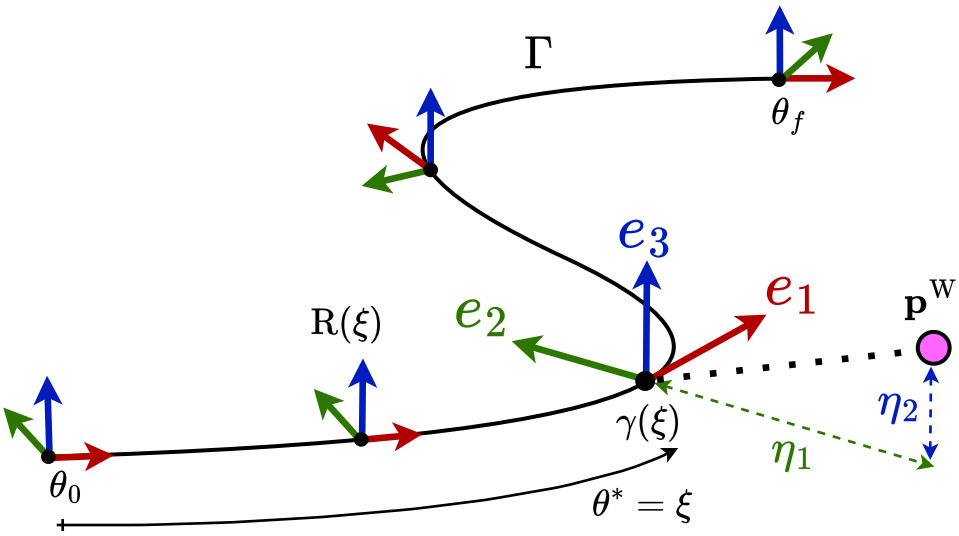


Figure 4.1 Spatial projection of the three-dimensional Cartesian coordinates $\boldsymbol{p}^W$, represented by the pink dot, onto a geometric path Γ with an associated adapted-frame $\mathrm{R}(\xi) = \{\boldsymbol{e_1}(\xi), \boldsymbol{e_2}(\xi), \boldsymbol{e_3}(\xi)\}$. The distance to the closest point on the path $\boldsymbol{\gamma}(\xi)$ is decomposed into the transverse coordinates $\boldsymbol{\eta} = [\eta_1, \eta_2]$.

distinct components: a *tangential* element that quantifies advancement along the reference path, and a *transverse* component that measures the orthogonal deviation from this path. Given the system's position $\boldsymbol{p}^W(t)$ in Cartesian space as expressed in (4.1b), we establish the *progress variable* $\xi(t)$ as the path parameter θ corresponding to the nearest point on reference curve Γ, formally defined as

$$\xi(t) = \theta^*(t) = \arg\min_{\theta} \frac{1}{2} ||\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\theta)||^2 \,. \tag{4.2}$$

With the progress variable $\xi(t)$ explicitly formulated as in (4.2), we can determine the distance between the dynamical system and the reference trajectory, both in world-frame and path-frame:

$$\boldsymbol{d}^{\Gamma}(t) = \mathrm{R}(\xi(t))^{\intercal} \boldsymbol{d}^{\mathrm{W}}(t) \,, \tag{4.3a}$$

where the displacement vector in the world frame is given by

$$\boldsymbol{d}^{\mathrm{W}}(t) = \boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t)) \,. \tag{4.3b}$$

As we have established that the path-frame $\mathrm{R}(\xi(t))$ is adapted, the initial component of $\boldsymbol{d}^{\Gamma}(t)$ vanishes, while the remaining two elements represent orthogonal distance, constituting the transverse elements $\boldsymbol{\eta}(t) = [\eta_1(t), \eta_2(t)]$. Thus, equation (4.3) can be rewritten as

$$\boldsymbol{d}^{\Gamma}(t) = [0, \boldsymbol{\eta}(t)] = \left[0, \boldsymbol{e_2}(\xi(t))\boldsymbol{d}^{\mathrm{W}}(t), \boldsymbol{e_3}(\xi(t))\boldsymbol{d}^{\mathrm{W}}(t)\right]^{\intercal} \,. \tag{4.4}$$

By integrating the progress variable $\xi(t)$ from (4.2) with the transverse coordinates $\boldsymbol{\eta}$(t) from (4.4), we define the *spatial coordinates* as an alternative description of the dynamical system's (4.1) position in Euclidean space:

$$\boldsymbol{p}^{\Gamma}(t) = [\xi(t), \boldsymbol{\eta}(t)] \in \mathbb{R}^3 \tag{4.5}$$

For a graphical representation of these spatial coordinates relative to reference path Γ, see Fig. 4.1.

4.1.1 Derivation of equations of motion

Building on the dynamical system in (4.1) and the parametric reference path Γ in (2.3), we formulate the equations governing the spatial coordinates in (4.5). This derivation is presented through two complementary approaches: one leveraging kinematic relationships and the other grounded in optimality principles.

A kinematic derivation

Starting from the distances in (4.3), the Cartesian coordinates of the dynamical system (4.1) can be formulated as

$$\boldsymbol{p}^{W}(t) = \boldsymbol{\gamma}(\xi(t)) + \mathrm{R}(\xi(t))\boldsymbol{d}^{\Gamma}(t). \tag{4.6}$$

Taking the time derivative of (4.6) results in

$$\boldsymbol{v}^{\mathrm{W}}(t) = \dot{\xi}(t)\left(\boldsymbol{\gamma}'(\xi(t)) + \mathrm{R}'(\xi(t))\boldsymbol{d}^{\Gamma}(t)\right) + \mathrm{R}(\xi(t))\dot{\boldsymbol{d}}^{\Gamma}(t). \tag{4.7}$$

Defining $\mathbf{i}^{\mathrm{W}} = [1,0,0]^{\intercal}$ as the first basis vector of the world frame and recalling from (2.2) that the parametric speed of the curve is given by $\sigma(\xi(t)) = ||\boldsymbol{\gamma}'(\xi(t))||$, we obtain

$$\boldsymbol{\gamma}'(\xi(t)) \equiv \sigma(\xi(t))\boldsymbol{e_1}(\xi(t)) \;\equiv\; \mathrm{R}(\xi(t))\mathbf{i}^{\mathrm{W}}\sigma(\xi(t)). \tag{4.8}$$

Substituting (4.8) into (4.7) and multiplying by $\mathrm{R}^{\intercal}(\xi(t))$ yields

$$0 = \dot{\xi}(t)\left(\sigma(\xi(t))\mathbf{i}^{\mathrm{W}} + \mathrm{R}(\xi(t))^{\intercal}\,\mathrm{R}'(\xi(t))\,\boldsymbol{d}^{\Gamma}(\xi(t))\right) + \dot{\boldsymbol{d}}^{\Gamma}(t) - \mathrm{R}^{\intercal}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t).$$

Utilizing the skew-symmetric matrix property from (2.6), the above equation simplifies to

$$0 = \dot{\xi}(t)\left(\sigma(\xi(t))\mathbf{i}^{\mathrm{W}} + \Omega^{\Gamma}(\xi(t))\boldsymbol{d}^{\Gamma}(\xi(t))\right) + \dot{\boldsymbol{d}}^{\Gamma}(t) - \mathrm{R}^{\intercal}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t),$$

which, in combination with (2.6) and (4.3), ultimately leads to the equations of motion for the *spatial coordinates*:

PATH-PARAMETERIZED CARTESIAN MOTION

The motion of a three-dimensional point moving at speed $\boldsymbol{v}^{\mathrm{W}}(t)$ with respect to a reference path parameterized by $\xi(t)$ with parametric speed $\sigma(\xi(t))$ and an adapted path-frame $\mathrm{R}(\xi) = [\boldsymbol{e_1}(\xi(t)), \boldsymbol{e_2}(\xi(t)), \boldsymbol{e_3}(\xi(t))]$ whose angular velocity is $\boldsymbol{\omega}(\xi(t)) = \omega_1^{\Gamma}(\xi(t))\boldsymbol{e_1}(\xi(t)) + \omega_2^{\Gamma}(\xi(t))\boldsymbol{e_2}(\xi(t)) + \omega_3^{\Gamma}(\xi(t))\boldsymbol{e_3}(\xi(t))$ is given by the following equations:

$$\dot{\xi}(t) = \frac{\boldsymbol{e_1}(\xi(t))^{\intercal}\boldsymbol{v}^{\mathrm{W}}(t)}{\sigma(\xi(t)) - \omega_3^{\Gamma}(\xi(t))\eta_1(t) + \omega_2^{\Gamma}(\xi(t))\eta_2(t)}\,, \tag{4.9a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e_2}(\xi(t))^{\intercal}\boldsymbol{v}^{\mathrm{W}}(t) + \dot{\xi}(t)\omega_1^{\Gamma}(\xi(t))\eta_2(t)\,, \tag{4.9b}$$

$$\dot{\eta}_2(t) = \boldsymbol{e_3}(\xi(t))^{\intercal}\boldsymbol{v}^{\mathrm{W}}(t) - \dot{\xi}(t)\omega_1^{\Gamma}(\xi(t))\eta_1(t)\,. \tag{4.9c}$$

An optimality derivation

Here, we demonstrate that an alternative approach also leads to the derivation of the equations of motion in (4.9). Specifically, we focus on the original definition of the progress variable $\xi(t)$ in (4.2). Since this is an unconstrained optimization problem, we utilize the first-order optimality condition:

$$0 = \frac{\mathrm{d}}{\mathrm{d}\theta}\left(\frac{1}{2}||\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\theta)||^2\right), \tag{4.10}$$

which, for the optimal path parameter $\theta^*(t) = \xi(t)$, simplifies to

$$0 = \left(\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t))\right)\boldsymbol{\gamma}'(\xi(t)).$$

Similarly, enforcing the first-order optimality condition with respect to time, we obtain

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}\left(\left(\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t))\right)\boldsymbol{\gamma}'(\xi(t))\right),$$

which expands to

$$0 = \boldsymbol{v}^{\mathrm{W}}(t)\boldsymbol{\gamma}'(\xi(t)) - \dot{\xi}(t)\boldsymbol{\gamma}'(\xi(t))\boldsymbol{\gamma}'(\xi(t)) + \dot{\xi}(t)\left(\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t))\right)\boldsymbol{\gamma}''(\xi(t)).$$

Recognizing that $\sigma^2(\xi(t)) = \boldsymbol{\gamma}'(\xi(t))\boldsymbol{\gamma}'(\xi(t))$, the equation becomes

$$\dot{\xi}(t) = \frac{\boldsymbol{v}^{\mathrm{W}}(t)\boldsymbol{\gamma}'(\xi(t))}{\sigma^2(\xi(t)) - \boldsymbol{d}^{\mathrm{W}}(t)\boldsymbol{\gamma}''(\xi(t))}. \tag{4.11}$$

Since $\boldsymbol{e_1}(\xi(t)) = \frac{\boldsymbol{\gamma}(\xi(t))}{\sigma(\xi(t))}$, it follows that

$$\boldsymbol{\gamma}''(\xi(t)) = \boldsymbol{e_1}'(\xi(t))\sigma(\xi(t)) + \boldsymbol{e_1}(\xi(t))\sigma'(\xi(t)).$$

The derivative of the first term is given by (2.6), i.e.,

$$\boldsymbol{e_1}'(\xi(t)) = \boldsymbol{e_2}(\xi(t))\omega_3^\Gamma(\xi(t)) - \boldsymbol{e_3}(\xi(t))\omega_2^\Gamma(\xi(t)),$$

while the second term vanishes because $\boldsymbol{d}^{\mathrm{W}}(t)$ is perpendicular to $\boldsymbol{e_1}(\xi(t))$. Incorporating this result into (4.11), we obtain

$$\begin{aligned}\dot{\xi}(t) &= \frac{\boldsymbol{v}^{\mathrm{W}}(t)\boldsymbol{\gamma}'(\xi(t))}{\sigma^2(\xi(t)) - \sigma(\xi(t))\boldsymbol{d}^{\mathrm{W}}(t)\beta}, \quad \text{where} \\ \beta &= (\boldsymbol{e_2}(\xi(t))\omega_3^\Gamma(\xi(t)) - \boldsymbol{e_3}(\xi(t))\omega_2^\Gamma(\xi(t))).\end{aligned} \tag{4.12}$$

Dividing both the numerator and denominator by $\sigma(\xi(t))$ and incorporating (4.4), we recover (4.9a), the equation of motion for the progress variable $\xi(t)$, previously derived using the kinematic approach. The remaining equations of motion for the transverse coordinates $\boldsymbol{\eta}(t)$, given by (4.9b) and (4.9c), can be obtained by differentiating (4.4) with respect to time and applying similar simplifications.

4.2 A universal path-parameterization

The equations of motion for the spatial states in (4.9) are general, as they do not depend on any specific path parameterization. That is, they hold for any parametric path, irrespective of its parametric speed or moving frame. To illustrate this generality, we show that the well-known Frenet-Serret-based parametric model is a particular case of the equations in (4.9). Furthermore, we demonstrate that the two-dimensional case, relevant for planar applications such as autonomous driving, follows as a straightforward simplification of the three-dimensional one.

4.2.1 A particular case: The Frenet Serret based models

The Frenet-Serret frame (FSF) has become the most widely adopted moving frame in literature [41, 42, 46] due to its analytical simplicity and ease of implementation. This popularity stems particularly from the autonomous driving community [30, 47, 35], where the planar application of the FSF enables numerical workarounds to circumvent its inherent limitations. The combination of this influence and the frame's simplicity has established the FSF as the de facto standard for path-parametric planning and control—even for three-dimensional applications [36, 1, 48], where the FSF exhibits several limitations that we will discuss in the following chapter.

Consequently, the parametric formulations available in literature are specifically tailored to the FSF. This becomes evident when we specialize Eqs. (4.9) by defining the angular velocity components in terms of curvature κ and torsion τ as $\left[\omega_1^\Gamma, \omega_2^\Gamma, \omega_3^\Gamma\right] = \sigma\,[\tau, 0, \kappa]$. Furthermore, for curves parameterized directly by arc-length $L = \xi$, the parametric speed simplifies to unit magnitude ($\sigma(\xi) = \frac{\mathrm{d}L}{\mathrm{d}\xi} = 1$), yielding:

$$\dot{\xi}(t) = \frac{\boldsymbol{e}_1(\xi(t))^\top \boldsymbol{v}^{\mathrm{W}}(t)}{1 - \kappa(\xi(t))\eta_1(t)}\,, \tag{4.13a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e}_2(\xi(t))^\top \boldsymbol{v}^{\mathrm{W}}(t) + \dot{\xi}(t)\tau(\xi(t))\eta_2(t)\,, \tag{4.13b}$$

$$\dot{\eta}_2(t) = \boldsymbol{e}_3(\xi(t))^\top \boldsymbol{v}^{\mathrm{W}}(t) - \dot{\xi}(t)\tau(\xi(t))\eta_1(t)\,. \tag{4.13c}$$

These equations of motion are formulated specifically for the FSF and are consistent with established literature, such as [41] and [42]. They also match directly with the equations in our quadrotor example presented in Section 3.4. This correspondence conclusively establishes the generality and wide applicability of our formulation in Eqs. (4.9).

4.2.2 The Planar 2D Case

For planar motions, such as those of ground vehicles, the parameterization in Eqs. (4.9) simplifies to two states, with the second transverse component vanishing ($\eta_2(t) = 0$). This reduces the equations of motion to the following planar model:

$$\dot{\xi}(t) = \frac{\boldsymbol{e}_1(\xi(t))^\top \boldsymbol{v}^{\mathrm{W}}(t)}{\sigma(\xi(t)) - \omega_3^\Gamma(\xi(t))\eta_1(t)}\,, \tag{4.14a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e}_2(\xi(t))^\top \boldsymbol{v}^{\mathrm{W}}(t)\,. \tag{4.14b}$$

Analogous to the three-dimensional case, we can specialize this planar model to the FSF by specifying the angular velocity as in Eq. (5.2) and setting $\sigma(\xi) = 1$, resulting in:

$$\dot{\xi}(t) = \frac{\boldsymbol{e}_1(\xi(t))^\intercal \boldsymbol{v}^{\mathrm{W}}(t)}{1 - \kappa(\xi(t))\eta_1(t)}\,, \tag{4.15a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e}_2(\xi(t))^\intercal \boldsymbol{v}^{\mathrm{W}}(t)\,, \tag{4.15b}$$

which is widely used in the autonomous driving community.

4.2.3 Which frame to choose?

Before concluding this section, it is important to highlight the universality of the equations derived in (4.9). These equations can be applied with any moving frame and path-parameterization technique. For instance, the previous subsection demonstrated their adaptation to the FSF case. This broad applicability naturally raises the question: *which relative frame should be chosen?* The next chapter explores this question in depth, evaluating different options and identifying the optimal choice for enabling egocentricity in autonomous systems.

4.3 Enclosed Publication

The scientific publication in which this spatial parameterization was originally derived is attached to this chapter and cited in [2]. A descriptive video accompanying the publication is available at `https://youtu.be/Sl6x8l7RJK8`, and a recording of the conference presentation can be viewed at `https://youtu.be/Vj1ofBTrNX4`.

Spatial motion planning with Pythagorean Hodograph curves

Jon Arrizabalaga and Markus Ryll

published in

IEEE Conference on Decision and Control (CDC)

2022

Contribution This paper presents a spatial path-parameterization applicable to any local frame, thereby generalizing existing state-of-the-art approaches. It also identifies Pythagorean Hodograph curves as a suitable tool for computing the parametric terms within this framework. In essence, the work extends the quadrotor-specific content discussed in the previous chapter, which is also published in [1]. I developed the theoretical formulation, implemented the method in software, and designed and executed the experiments that support the results presented.

Spatial motion planning with Pythagorean Hodograph curves

Jon Arrizabalaga[1] and Markus Ryll[1,2]

***Abstract*— This paper presents a two-stage prediction-based control scheme for embedding the environment's geometric properties into a collision-free Pythagorean Hodograph spline, and subsequently finding the optimal path within the parameterized free space. The ingredients of this approach are twofold: First, we present a novel spatial path parameterization applicable to any arbitrary curve without prior assumptions in its adapted frame. Second, we identify the appropriateness of Pythagorean Hodograph curves for a compact and continuous definition of the path-parametric functions required by the presented spatial model. This dual-stage formulation results in a motion planning approach, where the geometric properties of the environment arise as states of the prediction model. Thus, the presented method is attractive for motion planning in dense environments. The efficacy of the approach is evaluated according to an illustrative example.**

Video: https://youtu.be/Sl6x8l7RJK8

I. INTRODUCTION

Motion planning within cluttered and dynamic environments poses multiple challenges. Given a goal location and a performance criterion –duration, energy consumption or smoothness–, the underlying control scheme needs to drive the system, while remaining within a (possibly) variant free space. Thus, the optimal performance entails trading-off between the desired behavior, system constraints and spatial-awareness. To account for the latter, researchers have formulated path-parameterized control schemes allowing for a more precise embedding of the environment's geometric features. Such reformulations are based on a projection of the system dynamics from the Euclidean coordinate system to a moving frame attached to a path, located within the free space. The resulting system states –progress along the path and the orthogonal distance to it–, combined with the path's intrinsic properties –tangent, curvature and torsion– arising from the parameterization, yield very appealing attributes not only for obstacle avoidance, but also for convergence to the desired performance criterion.

Advantages of converting time-dependent dynamics into spatial-dependent were initially discussed in [1]. Applications of this spatial parameterization to planar vehicles demonstrated its ability to balance reference tracking and obstacle avoidance [2], [3]. Combining the spatial path-parameterization with advances in embedded-optimization allowed for real-time and near time-optimal Nonlinear Model Predictive Control (NMPC) applicable to miniature racing cars [4]. Subsequently, online obstacle avoidance was achieved in [5] by formulating a singularity-free parameterization of the system dynamics.

In the context of spatial dynamics, the expansion of the path-parameterization to all three dimensions has been simultaneously presented in [6]–[9]. The first two works focus on robot manipulators and leverage the spatial parameterization in a path-following and a time-optimal motion planning NMPC schemes. [8] exploits the decoupled tangential and transverse spatial states to formulate a controller for quadrotors capable of stabilizing the path following manifold, while [9] aims for minimum-time and collision-free trajectory generation. Similarly, [10] combines the time-optimality techniques from the planar methods with a complete spatial parameterization of the system dynamics to formulate a near time-optimal trajectory in real-time.

Given that the spatial path-parameterization conducted in all these works is based on the Frenet-Serret frame [11], the resultant equations of motion are not defined in inflection points, i.e., when the curvature vanishes, and thus, are only continuous for paths turning in one direction. Moreover, the undesired rotation of the Frenet-Serret frame with respect to its tangent component introduces a distortion in the representation of the environment [12].

To account for these shortcomings, in this paper we derive a spatial path-parameterization applicable to any adapted frame, and thus, resulting in a generalization of the state of the art's equations of motion. Subsequently, we present a two-stage motion planning approach, where the first stage embeds the environment's geometry into a collision-free Pythagorean Hodograph spline, compliant with the aforementioned parameterization, while the second stage finds the (local) optimal path according to a performance criterion, the parameterized system dynamics and space constraints. In particular, we make the following contributions:

1) We parameterize the three-dimensional Euclidean coordinates with respect to a path with an *arbitrary adapted frame*. To the best of the authors' knowledge, this is the first spatial path-parameterization that is independent from the Frenet-Serret frame.
2) We identify the suitability of Pythagorean Hodograph curves [13] to efficiently and continuously define the parametric-speed, adapted frame components and angular velocity needed by the aforementioned parametrization.
3) We present a hierarchical motion planning algorithm in which the spatial features of the environment are first encoded into a Pythagorean Hodograph spline and then exploited in a prediction-based optimization.

[1]Autonomous Aerial Systems, School of Engineering and Design, Technical University of Munich, Germany. E-mail: jon.arrizabalaga@tum.de and markus.ryll@tum.de

[2]Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University of Munich

Published in IEEE Conference on Decision and Control (CDC), Cancun, Mexico, 2022.

The remainder of this paper is structured as follows: Section II path parameterizes the three-dimensional Euclidean coordinates for an arbitrary adapted frame. Section III introduces Pythagorean Hodograph curves and exhibits their applicability to the derived parameterization. Section IV presents the hierarchical motion planner and the respective two optimization problems. Experimental results are shown in Section V before Section VI presents the conclusions.

Notation: We will use $\dot{(\cdot)} = \frac{\mathrm{d}(\cdot)}{\mathrm{d}t}$ for time derivatives and $(\cdot)' = \frac{\mathrm{d}(\cdot)}{\mathrm{d}\xi}$ for differentiating over path-parameter ξ. For readability we will employ the abbreviation ${}_t\xi = \xi(t)$.

II. SPATIAL PATH-PARAMETERIZATION

A. Preliminaries on space curves

Let Γ be a curve whose position and orientation are given by two rational and sufficiently continuous functions that depend on path-parameter ξ:

$$\Gamma = \{\boldsymbol{\gamma}(\xi) \in \mathbb{R}^3, \mathrm{R}(\xi) \in \mathbb{R}^{3x3} \,|\, \xi \in [0,1]\} \tag{1}$$

The rational orthonormal frame $\mathrm{R}(\xi) = [\boldsymbol{e_1}(\xi), \boldsymbol{e_2}(\xi), \boldsymbol{e_3}(\xi)]$ defines the orientation along the curve and is assumed to be *adapted*, i.e., the first frame vector coincides with the curve tangent $\boldsymbol{e_1}(\xi) = \frac{\boldsymbol{\gamma}'(\xi)}{||\boldsymbol{\gamma}'(\xi)||_2}$. The change of this frame with respect to the path-parameter is specified by

$$\mathrm{R}'(\xi) = \mathrm{R}(\xi) \overbrace{\begin{bmatrix} 0 & -\chi_3(\xi) & \chi_2(\xi) \\ \chi_3(\xi) & 0 & -\chi_1(\xi) \\ -\chi_2(\xi) & \chi_1(\xi) & 0 \end{bmatrix}}^{\mathrm{C}(\xi)} \tag{2}$$

where $\mathrm{C}(\xi)$ is the Cartan connection matrix associated to the angular velocity vector $\boldsymbol{\omega}(\xi)$

$$\boldsymbol{\omega}(\xi) = \chi_1(\xi)\boldsymbol{e_1}(\xi) + \chi_2(\xi)\boldsymbol{e_2}(\xi) + \chi_3(\xi)\boldsymbol{e_3}(\xi).$$

From (2) its components are given by

$$\chi_1(\xi) = \boldsymbol{e_2}'(\xi)\,\boldsymbol{e_3}(\xi)\,, \tag{3a}$$
$$\chi_2(\xi) = \boldsymbol{e_3}'(\xi)\,\boldsymbol{e_1}(\xi)\,, \tag{3b}$$
$$\chi_3(\xi) = \boldsymbol{e_1}'(\xi)\,\boldsymbol{e_2}(\xi)\,. \tag{3c}$$

B. Derivation of equations of motion

Let $\boldsymbol{p}_\mathrm{W}(t) \in \mathbb{R}^3$ be the location of a point-mass represented in world-frame's $(\cdot)_\mathrm{W}$ Euclidean coordinates at time t. The distance with respect to the closest point on curve Γ is given by $\boldsymbol{d}_\mathrm{W}(t) = \boldsymbol{p}_\mathrm{W}(t) - \boldsymbol{\gamma}({}_t\xi)$. Translating this distance to the curve-frame $(\cdot)_\Gamma$ results in $\boldsymbol{d}_\Gamma(t) = \mathrm{R}({}_t\xi)^\intercal \boldsymbol{d}_\mathrm{W}(t)$, and therefore, its position in the world-frame can be denoted as

$$\boldsymbol{p}_\mathrm{W}(t) = \boldsymbol{\gamma}({}_t\xi) + \mathrm{R}({}_t\xi)\boldsymbol{d}_\Gamma(t)\,. \tag{4}$$

Since we have assumed the frame $\mathrm{R}({}_t\xi)$ to be adapted, the first element of $\boldsymbol{d}_\Gamma(t)$, i.e., the tangent component is zero, while the remaining two elements are the perpendicular projections, which we will refer to as *transverse coordinates* $\boldsymbol{w}(t) = [w_1(t), w_2(t)]$. The projected distance can be observed in Fig. 1 and is expressed as

$$\boldsymbol{d}_\Gamma(t) = [0, \boldsymbol{w}(t)] = [0, \boldsymbol{e_2}({}_t\xi)\boldsymbol{d}_\mathrm{W}(t), \boldsymbol{e_3}({}_t\xi)\boldsymbol{d}_\mathrm{W}(t)]^\intercal\,. \tag{5}$$

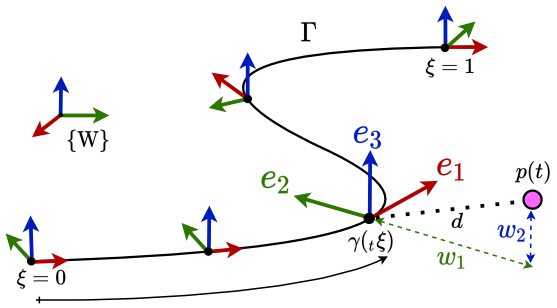


Fig. 1: Spatial representation of a point-mass, represented by the pink dot, according to path-parameter ξ and an adapted frame $\{\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}\}$ of curve Γ. The distance $d(t)$ between the point-mass located in $p(t)$ and the closest point on the curve $\gamma({}_t\xi)$ is projected onto the transverse coordinates $w_1(t)$ and $w_2(t)$. For readability, the time dependencies on the distance and transverse coordinates have been omitted.

Differentiating (4) with respect to time results in

$$\boldsymbol{v}_\mathrm{W}(t) = \dot{\xi}(t)\left(\boldsymbol{\gamma}'({}_t\xi) + \mathrm{R}'({}_t\xi)\boldsymbol{d}_\Gamma(t)\right) + \mathrm{R}({}_t\xi)\dot{\boldsymbol{d}}_\Gamma(t)\,. \tag{6}$$

Denoting $\boldsymbol{i}_\mathrm{W} = [1,0,0]^\intercal$ as the first component of the world-frame and setting the curve's parametric speed as $\sigma({}_t\xi) = ||\boldsymbol{\gamma}'({}_t\xi)||_2$ is equivalent to

$$\boldsymbol{\gamma}'({}_t\xi) \equiv \sigma({}_t\xi)\boldsymbol{e_1}({}_t\xi) \;\equiv\; \mathrm{R}({}_t\xi)\boldsymbol{i}_\mathrm{W}\sigma({}_t\xi)\,. \tag{7}$$

Introducing (7) in (6) and multiplying it with $\mathrm{R}^\intercal({}_t\xi)$ leads to

$$0 = \dot{\xi}(t)\left(\sigma({}_t\xi)\boldsymbol{i}_\mathrm{W} + \mathrm{R}({}_t\xi)^\intercal\,\mathrm{R}'({}_t\xi)\,\boldsymbol{d}_\Gamma({}_t\xi)\right) + \dot{\boldsymbol{d}}_\Gamma(t) - \mathrm{R}^\intercal({}_t\xi)\boldsymbol{v}_\mathrm{W}(t)\,.$$

Leveraging (2), the latter equation results in the following simplification

$$0 = \dot{\xi}(t)\left(\sigma({}_t\xi)\boldsymbol{i}_\mathrm{W} + \mathrm{C}({}_t\xi)\boldsymbol{d}_\Gamma({}_t\xi)\right) + \dot{\boldsymbol{d}}_\Gamma(t) - \mathrm{R}^\intercal({}_t\xi)\boldsymbol{v}_\mathrm{W}(t)\,,$$

which combined with the Cartan matrix and (5) yields

$$\dot{\xi}(t) = \frac{\boldsymbol{e_1}({}_t\xi)^\intercal \boldsymbol{v}_\mathrm{W}(t)}{\sigma({}_t\xi) - \chi_3({}_t\xi)w_1(t) + \chi_2({}_t\xi)w_2(t)}\,, \tag{8a}$$
$$\dot{w}_1(t) = \boldsymbol{e_2}({}_t\xi)^\intercal \boldsymbol{v}_\mathrm{W}(t) + \dot{\xi}(t)\chi_1({}_t\xi)w_2(t)\,, \tag{8b}$$
$$\dot{w}_2(t) = \boldsymbol{e_3}({}_t\xi)^\intercal \boldsymbol{v}_\mathrm{W}(t) - \dot{\xi}(t)\chi_1({}_t\xi)w_1(t)\,. \tag{8c}$$

These equations describe the motion of the *spatial coordinates* ξ, ω_1, ω_2 under a world-frame velocity $\boldsymbol{v}_\mathrm{W}$ with respect to curve Γ whose parametric speed, adapted frame and angular velocity are $\sigma(\xi)$, $\mathrm{R}(\xi)$ and $\boldsymbol{\omega}(\xi)$.

C. Comparison to Frenet-Serret based models

It can be easily verified that the spatial path parameterization in [6]–[9] is a particular case of equation (8), where the adapted frame matches the Frenet-Serret frame, and thus, the components of its angular velocity are $[\chi_1, \chi_2, \chi_3] = [\tau, 0, \kappa]$, τ and κ being the curve's torsion and curvature. Furthermore, if the curve is assumed to be parameterized directly by its arc-length $s = \xi$, the parametric speed reduces to a unit magnitude $\sigma(\xi) = \frac{\mathrm{d}s}{\mathrm{d}\xi} = 1$.

D. Choosing an adapted frame

The analytical simplifications of the Frenet-Serret frame come at the expense of 1) *discontinuities* when the curvature vanishes $\kappa = 0$, causing abrupt flips in the second and third components $\boldsymbol{e_2}, \boldsymbol{e_3}$ of the frame and 2) an *unnecessary twist* with respect to the first component $\boldsymbol{e_1}$, which occurs because $\chi_2 = 0$ rotates the frame such that its second component $\boldsymbol{e_2}$ points towards the center of the curvature [13].

To ensure a continuous and smooth representation of the environment we are interested in finding a frame 1) that is *defined* along the entire curve and 2) whose second and third components *rotate the minimum possible amount* to ensure that the frame remains adapted. Such frames are denoted as *Rotation Minimizing Frames* (RMF) [14] and are characterized by $\chi_1 = 0$. An illustrative comparison between a Frenet-Serret frame and an RMF is depicted in Fig. 2 .

III. PYTHAGOREAN HODOGRAPH CURVES

To leverage the derived spatial parameterization for prediction-based motion planning, we need a curve that 1) has an adapted frame which approximates an RMF, 2) has path-related functions $\sigma(\xi), \mathrm{R}(\xi), \boldsymbol{\omega}(\xi)$ that can be expressed in closed-form by C^2 equations and 3) can be reconstructed using a small amount of parameters. The first requirement ensures a *twist-free* and consistent encoding of the environment's properties, while the second and third allow for efficiently embedding (8) into a Nonlinear Program (NLP).

A Pythagorean Hodograph (PH) curve is defined by the condition that the parametric speed $\sigma(\xi)$ is a polynomial of the path-parameter ξ [13], and thus, implies that $\sigma(\xi)$ meets conditions (2) and (3). In this section, we will see how PH curves also extend these requirements to the remaining two functions $\mathrm{R}(\xi)$, $\boldsymbol{\omega}(\xi)$, while simultaneously being able to compute RMF approximations.

A. Preliminaries on Spatial PH curves

Recalling that $\sigma(\xi) = ||\boldsymbol{\gamma}'(\xi)||_2$ and revisiting the aforementioned definition of PH curves results in

$$\sigma^2(\xi) = x'^2(\xi) + y'^2(\xi) + z'^2(\xi)\,, \tag{9}$$

where $\sigma(\xi)$ is a polynomial. As proven in [15], every term in (9) can be expressed in terms of a quaternion polynomial $\boldsymbol{Z}(\xi) = u(\xi) + v(\xi)\mathbf{i} + g(\xi)\mathbf{j} + h(\xi)\mathbf{k}$, where $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ refers to the $\mathbb{R}^4$ standard basis:

$$\sigma(\xi) = u^2(\xi) + v^2(\xi) + g^2(\xi) + h^2(\xi)\,, \tag{10a}$$
$$x'(\xi) = u^2(\xi) + v^2(\xi) - g^2(\xi) - h^2(\xi)\,, \tag{10b}$$
$$y'(\xi) = 2\left[u(\xi)h(\xi) + v(\xi)g(\xi)\right]\,, \tag{10c}$$
$$z'(\xi) = 2\left[v(\xi)h(\xi) - u(\xi)g(\xi)\right] \tag{10d}$$

Each component of the quaternion is a polynomial, and thus, can be expressed according to the Bernstein form as

$$\boldsymbol{Z}(\xi) = \sum_{i=0}^{n} \binom{n}{i} \boldsymbol{\zeta}_i \,(1-\xi)^{n-i}\,\xi^i\,, \tag{11}$$

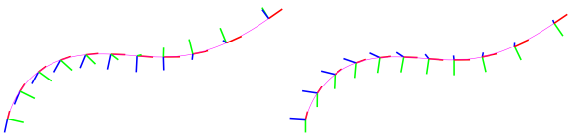


Fig. 2: A comparison between Frenet-Serret frame (left) and RMF (right) for a quintic Pythagorean Hodograph curve. Notice that the Frenet Serret frame shows an *unnecessary rotation* with respect to the tangent component. This illustration is a recreation of the first example presented in [17].

where n refers to the degree of the polynomial and $\boldsymbol{\zeta}_i = [u_i, v_i, g_i, h_i]$ are the respective Bernstein coefficients. These relate to the control points of the quaternion polynomial $\boldsymbol{Z}(\xi)$ and may be rearranged into the following matrix:

$$\boldsymbol{\zeta} = \begin{bmatrix} u_0 & v_0 & g_0 & h_0 \\ & \vdots & & \\ u_n & v_n & g_n & h_n \end{bmatrix}. \tag{12}$$

From eqs. (9), (10a) and (11) it can be concluded that a quaternion hodograph of degree n corresponds to a curve of degree $2n+1$, while relying just on $4(n+1)$ Bernstein coefficients.

Another appealing feature of PH curves is their inheritance of a continuous adapted frame that is also solely dependent on its quaternion polynomial. This frame is named *Euler Rodrigues Frame* (ERF) [16] and its respective components are described as

$$\boldsymbol{e_1}(\xi) = \boldsymbol{Z}(\xi)\mathbf{i}\boldsymbol{Z}^*(\xi)\,, \tag{13a}$$
$$\boldsymbol{e_2}(\xi) = \boldsymbol{Z}(\xi)\mathbf{j}\boldsymbol{Z}^*(\xi)\,, \tag{13b}$$
$$\boldsymbol{e_2}(\xi) = \boldsymbol{Z}(\xi)\mathbf{k}\boldsymbol{Z}^*(\xi)\,, \tag{13c}$$

with $(.)^*$ referring to the quaternion's conjugate. Combining this frame with (3) leads to the following angular velocity components:

$$\chi_1 = \frac{2\left(uv' - u'v - gh' + g'h\right)}{u^2+v^2+g^2+h^2}\,, \tag{14a}$$
$$\chi_2 = \frac{2\left(ug' - u'g + vh' - v'h\right)}{u^2+v^2+g^2+h^2}\,, \tag{14b}$$
$$\chi_3 = \frac{2\left(uh' - u'h - vg' + v'g\right)}{u^2+v^2+g^2+h^2}\,, \tag{14c}$$

where the dependency on path-parameter ξ has been omitted for clarity. Putting together eqs. (10a), (13) and (14) it becomes apparent that PH curves enable us to express all path-related functions $\sigma(\xi)$, $\mathrm{R}(\xi)$ and $\boldsymbol{\omega}(\xi)$ needed by the spatial parameterization in Section II only depending on the Bernstein coefficients $\boldsymbol{\zeta}$ of the quaternion polynomial.

B. PH curves with RMF

As said before, an RMF is characterized for not rotating with respect to its tangent $\chi_1 = 0$. In the case of PH curves, from (14a) results that the associated ERF is an RMF if the following condition holds true:

$$u(\xi)v'(\xi) - u'(\xi)v(\xi) - g(\xi)h'(\xi) + g'(\xi)h(\xi) = 0 \tag{15}$$

Given its differential nature, a generic closed form solution is very difficult to obtain [16]. Nevertheless, in the upcoming section, this condition will play a crucial role when computing PH curves whose ERFs approximate RMFs.

C. Spline of PH nonic curves

To find a collision-free path, we are interested in concatenating multiple PH curves. When doing so, it has to be ensured that the path-related functions $\sigma(\xi)$, $\mathrm{R}(\xi)$, $\omega(\xi)$ remain C^2. Out of all three, the frame's angular velocity is the one with a lowest degree of $n-1$, which is equivalent to a C^3 requirement in the quaternion polynomial. Thus, the minimum order for constructing PH splines under these conditions is $n=4$, resulting in curves of degree 9, also known as *nonics*.

Consequently, a spline with m sections depends on $20m$ Bernstein coefficients. However, when applying the C^3 condition to the quaternion polynomials, only 4 coefficients per section are independent, while the remaining 16 are linearly related to the coefficients of the previous section:

$$C^0 \rightarrow \quad \boldsymbol{\zeta}_{0,k+1} = \boldsymbol{\zeta}_{4,k}, \tag{16a}$$
$$C^1 \rightarrow \quad \boldsymbol{\zeta}_{1,k+1} = -\boldsymbol{\zeta}_{3,k} + 2\boldsymbol{\zeta}_{4,k}, \tag{16b}$$
$$C^2 \rightarrow \quad \boldsymbol{\zeta}_{2,k+1} = \boldsymbol{\zeta}_{2,k} - 4\boldsymbol{\zeta}_{3,k} + 4\boldsymbol{\zeta}_{4,k}, \tag{16c}$$
$$C^3 \rightarrow \quad \boldsymbol{\zeta}_{3,k+1} = -\boldsymbol{\zeta}_{1,k} + 6\boldsymbol{\zeta}_{2,k} - 12\boldsymbol{\zeta}_{3,k} + 8\boldsymbol{\zeta}_{4,k}\,. \tag{16d}$$

for $k = 1, ..., m$ and $\boldsymbol{\zeta}_{i,k}$ referring to the control point i – row i from the matrix in (12) – of the quaternion polynomial in section k. As a result of these equivalencies, a spline consisting of PH nonic curves depends on $20 + 4(m-1)$ Bernstein coefficients.

Finally, the curve's control points $\boldsymbol{p}_{0,\dots,9|k}$ at the spline's section k relate to the respective quaternion control points $\boldsymbol{\zeta}_{0,\dots,4|k}$ by the following expressions [18]:

$$\boldsymbol{p}_{1,k} = \boldsymbol{p}_{0,k} + \frac{1}{9}\boldsymbol{\zeta}_{0,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{0,k}\,, \tag{17a}$$
$$\boldsymbol{p}_{2,k} = \boldsymbol{p}_{1,k} + \frac{1}{18}\left(\boldsymbol{\zeta}_{0,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{1,k} + \boldsymbol{\zeta}_{1,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{0,k}\right), \tag{17b}$$
$$\boldsymbol{p}_{3,k} = \boldsymbol{p}_{2,k} + \frac{1}{126}\left(3\boldsymbol{\zeta}_{0,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{2,k} + 8\boldsymbol{\zeta}_{1,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{1,k} + 3\boldsymbol{\zeta}_{2,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{0,k}\right), \tag{17c}$$
$$\boldsymbol{p}_{4,k} = \boldsymbol{p}_{3,k} + \frac{1}{126}\left(\boldsymbol{\zeta}_{0,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{3,k} + 6\boldsymbol{\zeta}_{1,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{2,k} + 6\boldsymbol{\zeta}_{2,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{1,k} + \boldsymbol{\zeta}_{3,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{0,k}\right), \tag{17d}$$
$$\boldsymbol{p}_{5,k} = \boldsymbol{p}_{4,k} + \frac{1}{630}\left(\boldsymbol{\zeta}_{0,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{4,k} + 16\boldsymbol{\zeta}_{1,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{3,k} + 36\boldsymbol{\zeta}_{2,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{2,k} + 16\boldsymbol{\zeta}_{3,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{1,k} + \boldsymbol{\zeta}_{4,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{0,k}\right), \tag{17e}$$
$$\boldsymbol{p}_{6,k} = \boldsymbol{p}_{5,k} + \frac{1}{126}\left(\boldsymbol{\zeta}_{1,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{4,k} + 6\boldsymbol{\zeta}_{2,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{3,k} + 6\boldsymbol{\zeta}_{3,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{2,k} + \boldsymbol{\zeta}_{4,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{1,k}\right), \tag{17f}$$
$$\boldsymbol{p}_{7,k} = \boldsymbol{p}_{6,k} + \frac{1}{126}\left(3\boldsymbol{\zeta}_{2,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{4,k} + 8\boldsymbol{\zeta}_{3,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{3,k} + 3\boldsymbol{\zeta}_{4,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{2,k}\right), \tag{17g}$$
$$\boldsymbol{p}_{8,k} = \boldsymbol{p}_{7,k} + \frac{1}{18}\left(\boldsymbol{\zeta}_{3,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{4,k} + \boldsymbol{\zeta}_{4,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{3,k}\right), \tag{17h}$$
$$\boldsymbol{p}_{9,k} = \boldsymbol{p}_{8,k} + \frac{1}{9}\boldsymbol{\zeta}_{4,k}\,\mathbf{i}\,\boldsymbol{\zeta}^*_{4,k}\,, \tag{17i}$$

where $\boldsymbol{p}_{0,k}$ is a free integration constant that we set to the starting position of section k. With these control points the position function for a given section k of the spline can be expressed in Bernstein's form as

$$\boldsymbol{\gamma}_k(\xi) = \sum_{i=0}^{9} \binom{9}{i} \boldsymbol{p}_{i,k}\,(1-\xi)^{9-i}\,\xi^i\,. \tag{18}$$

IV. SPATIAL MOTION PLANNING

A. Stage 1: Computing collision-free PH nonic splines

Other than being compact and continuous, PH splines also need to be collision-free. For this purpose, in a similar manner to [19], we describe the non-convex free space $\mathcal{F}$ as the union of m convex sets, each represented by a polyhedron, i.e., $\mathcal{F} = \cup_{k=1}^{m} \mathcal{P}_k$. Considering that a curve with a Bernstein Polynomial basis is contained in the convex hull of its control points, we can ensure the given PH spline section will be inside the respective polyhedron by requiring that all control points are enclosed within it. In section k, this translates to the following condition:

$$\mathbf{A}_k \boldsymbol{p}_{i,k} \leq \mathbf{b}_k \quad \text{with} \quad i = 0, ..., 9\,, \tag{19}$$

where $\boldsymbol{p}_{i,k}$ stands for the curve control points of the PH-spline's k-th section and $\mathbf{A}_k, \mathbf{b}_k$ refer to the half-space representation of polyhedron $\mathcal{P}_k$. Under these constraints, the $20 + 4(m-1)$ free Bernstein coefficients can be used to frame the spline according to a desired criterion. Similarly to [20], acknowledging the lack of a closed-form solution to the differential condition in (15), we exploit the aforementioned degrees of freedom to find a PH nonic spline, whose ERF is as *rotation minimizing as possible*. For a given section k, this is equivalent to minimizing the functional

$$f_{\text{PH}}(\boldsymbol{\zeta}_k) = \int_0^1 \chi^2_{1,k}(\xi)\,d\xi\,. \tag{20}$$

Combining (20) with the curve's and quaternion polynomial's continuity conditions, as well as the collision-free constraints in (19), results in the following optimal control problem (OCP):

$$\min_{\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_m} \; f_{\text{PH}}(\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_m) \tag{21a}$$
$$\text{s.t.} \quad \boldsymbol{p}_{0,1} = \boldsymbol{p}_{\text{initial}} \tag{21b}$$
$$\boldsymbol{p}_{9,m} = \boldsymbol{p}_{\text{final}} \tag{21c}$$
$$\boldsymbol{p}_{0,k+1} = \boldsymbol{p}_{9,k}\,, \qquad k = 1, \cdots, m-1 \tag{21d}$$
$$\boldsymbol{Z}_{k+1}(0) = \boldsymbol{Z}_k(1)\,, \qquad k = 1, \cdots, m-1 \tag{21e}$$
$$\boldsymbol{Z}'_{k+1}(0) = \boldsymbol{Z}'_k(1)\,, \qquad k = 1, \cdots, m-1 \tag{21f}$$
$$\boldsymbol{Z}''_{k+1}(0) = \boldsymbol{Z}''_k(1)\,, \qquad k = 1, \cdots, m-1 \tag{21g}$$
$$\boldsymbol{Z}'''_{k+1}(0) = \boldsymbol{Z}'''_k(1)\,, \qquad k = 1, \cdots, m-1 \tag{21h}$$
$$\mathbf{A}_k \boldsymbol{p}_{i,k} \leq \mathbf{b}_k\,, \qquad \substack{k=1,\cdots,m \\ i=0,\cdots,9} \tag{21i}$$

where (21b) and (21c) set the starting and ending point, (21d) guarantees that the spline sections are attached to each other, from (21e) to (21h) ensure that the $\sigma(\xi)$, $\mathrm{R}(\xi)$ and $\omega(\xi)$ functions are C^2, while (21i) enforces the spline to remain in the free space.

B. Stage 2: Path-Parametric NMPC

Once the OCP in (21) is solved, the parametric functions $\sigma(\xi)$, $\mathrm{R}(\xi)$ and $\boldsymbol{\omega}(\xi)$ can be re-computed according to eqs. (10a), (13) and (14). By embedding these functions into (8), we leverage the spatial reformulation in a prediction-based controller, where the environment's geometric properties are fully embedded into the system dynamics.

To do so, we take the standard NMPC approach [21], where the input obtained from an optimization problem is applied in a receding horizon fashion. The respective OCP finds the local optimal control action over time horizon T, considering a nonlinear plant model $f(\boldsymbol{x}, \boldsymbol{u})$, a nonlinear cost function $f_{\text{MPC}}(\boldsymbol{x}, \boldsymbol{u})$, as well as nonlinear constraints $g(\boldsymbol{x}, \boldsymbol{u})$ on states $\boldsymbol{x}$ and inputs $\boldsymbol{u}$. We denote the quaternion polynomial coefficients obtained from OCP (21) as $\boldsymbol{\mathcal{Z}}$. Choosing states $\boldsymbol{x} = [\xi, \boldsymbol{w}]$ and inputs $\boldsymbol{u} = [\boldsymbol{v}_{\text{W}}]$, the plant model is given by the equations of motion in (8). As a consequence, the OCP that is solved in the motion planning's second stage is

$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} \int_0^T f_{\text{MPC}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\mathcal{Z}})\, dt \tag{22a}$$

$$\text{s.t.} \quad \boldsymbol{x}(0) = \boldsymbol{x}_{\text{initial}}\,, \tag{22b}$$

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\mathcal{Z}}), \qquad t \in [0, T] \tag{22c}$$

$$g\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\mathcal{Z}}\right) \leq 0\,, \qquad t \in [0, T] \tag{22d}$$

$$\mathbf{A}_{\boldsymbol{x}}\, \boldsymbol{p}_{\text{W}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\mathcal{Z}}) \leq \mathbf{b}_{\boldsymbol{x}}\,, \quad t \in [0, T] \tag{22e}$$

where $\mathbf{A}_{\boldsymbol{x}} = \mathbf{A}(\boldsymbol{x}(t))$ and $\mathbf{b}_{\boldsymbol{x}} = \mathbf{b}(\boldsymbol{x}(t))$ in constraint (22e) stand for the half-space matrixes of the polyhedron associated to the location of the point-mass $\boldsymbol{p}_{\text{W}}$, which can be computed from (4). With this constraint, it is guaranteed that the motion planning takes place within the free space.

C. Complete Approach

The complete motion planning scheme, alongside an exemplary application within a generic control loop, is given in Algorithm 1. After estimating the state of the system and decoupling the free space into multiple polyhedron, the motion planner solves both stages and finds the optimal path within a time horizon. In a receding horizon manner, only the first optimal input is applied.

Algorithm 1 Two-Stage Spatial Motion Planning

function MOTION PLANNING($\boldsymbol{x}$, $\mathcal{P}_1, \cdots, \mathcal{P}_m$)
  $\boldsymbol{p}_{\text{init}}, \boldsymbol{p}_{\text{final}} \leftarrow$ FindStartAndEnd($\mathcal{P}_1, \cdots, \mathcal{P}_m$)
  $\boldsymbol{\mathcal{Z}} \leftarrow$ SolveStage1($\boldsymbol{p}_{\text{init}}, \boldsymbol{p}_{\text{final}}, \mathcal{P}_1, \cdots, \mathcal{P}_m$)
  $\boldsymbol{x}^*, \boldsymbol{u}^* \leftarrow$ SolveStage2($\boldsymbol{x}, \boldsymbol{\mathcal{Z}}$)
  return $\boldsymbol{x}^*, \boldsymbol{u}^*$
end function
while controller enabled **do**
  $\boldsymbol{x} \leftarrow$ STATE ESTIMATION
  $\mathcal{P}_1, \cdots, \mathcal{P}_m \leftarrow$ ENVIRONMENT MAPPING
  $\boldsymbol{x}^*, \boldsymbol{u}^* \leftarrow$ MOTION PLANNING($\boldsymbol{x}$, $\mathcal{P}_1, \cdots, \mathcal{P}_m$)
  LOW LEVEL CONTROL($\boldsymbol{u}_0^*$)
end while

V. TUTORIAL EXAMPLE

To evaluate our approach, we generate an exemplary representation of the free space, consisting of four polyhedrons. Their sparsity along all three Euclidean axes, alongside their differences on size, allow for testing the capacity of our approach to deal with highly non-convex spaces.

A. Numerical implementation

In the first stage, we formulate OCP (21) in CasADi [22] and solve it with IPOPT [23] and MA27 [24] as the linear solver back-end. We approximate the integral in (20) with a 4th-order Runge-Kutta of step size $\Delta\xi = 0.1$. The decision variables are initialized by solving a least squares problem on the constraint-residuals with the Trust-Region-Reflective method [25].

In the second stage, we approximate the OCP in (22) by a Nonlinear Program (NLP) according to the multiple-shooting approach [26], in which time horizon T is split into N sections with constant decision variables:

$$\min_{\substack{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N, \\ \boldsymbol{u}_0, \cdots, \boldsymbol{u}_N}} \sum_{k=0}^{N-1} f_{\text{MPC}.k}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{Z}}) \tag{23a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{initial}}\,, \tag{23b}$$

$$\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{Z}}, \Delta t),\ k = 0, \cdots, N-1 \tag{23c}$$

$$g\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{Z}}\right) \leq 0\,, \qquad k = 0, \cdots, N-1 \tag{23d}$$

$$\mathbf{A}_{\boldsymbol{x}}\, \boldsymbol{p}_{\text{W}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{Z}}) \leq \mathbf{b}_{\boldsymbol{x}}\,, \quad k = 0, \cdots, N-1 \tag{23e}$$

$$h\left(\boldsymbol{x}_N, \boldsymbol{\mathcal{Z}}\right) \leq 0\,, \tag{23f}$$

with $\mathbf{A}_{\boldsymbol{x}} = \mathbf{A}(\boldsymbol{x}_k)$ and $\mathbf{b}_{\boldsymbol{x}} = \mathbf{b}(\boldsymbol{x}_k)$. The respective plant model F is obtained from discretizing $f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\mathcal{Z}})$ with a fixed time-step Δt. As an illustrative showcase, the cost function is chosen to maximize for progress:

$$f_{\text{MPC},k}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{Z}}) = -\lambda\, L(\boldsymbol{x}_k, \boldsymbol{\mathcal{Z}}) + ||\boldsymbol{u}_k||_R^2\,,$$

where $L(\boldsymbol{x}_k, \boldsymbol{\mathcal{Z}})$ is the arc-length at state $\boldsymbol{x}_k$ and can be obtained by integrating

$$L(\boldsymbol{x}_k, \boldsymbol{\mathcal{Z}}) = \int_0^{\xi_k} \sigma(\xi)\ d\xi\,.$$

Given that the parametric speed $\sigma(\xi)$ is a polynomial (see (10a)), the integral above can be expressed in closed-form as function that is solely dependent on the quaternion coefficients $\boldsymbol{\mathcal{Z}}$ computed in the first stage. Moreover, to soften the control commands we extend the states by appending the velocity to the state vector $\boldsymbol{x} = [\xi, \boldsymbol{w}, \boldsymbol{v}_{\text{W}}]$ and assigning the acceleration to the input $\boldsymbol{u} = [\boldsymbol{a}_{\text{W}}]$. As common, within (23d) we account for input constraints. Lastly, to ensure feasibility, we add an additional constraint (23f) on the states of the last shooting node.

We solve the NLP (23) with the Sequential Quadratic Programming (SQP) method in the optimal control framework ACADOS [27]. To address real-time applicability, we use its real-time iteration variant (SQP-RTI) [28] in conjunction with HPIPM [29], which efficiently solves the underlying quadratic programs. The system dynamics are integrated by an explicit 4th-order Runge-Kutta method.

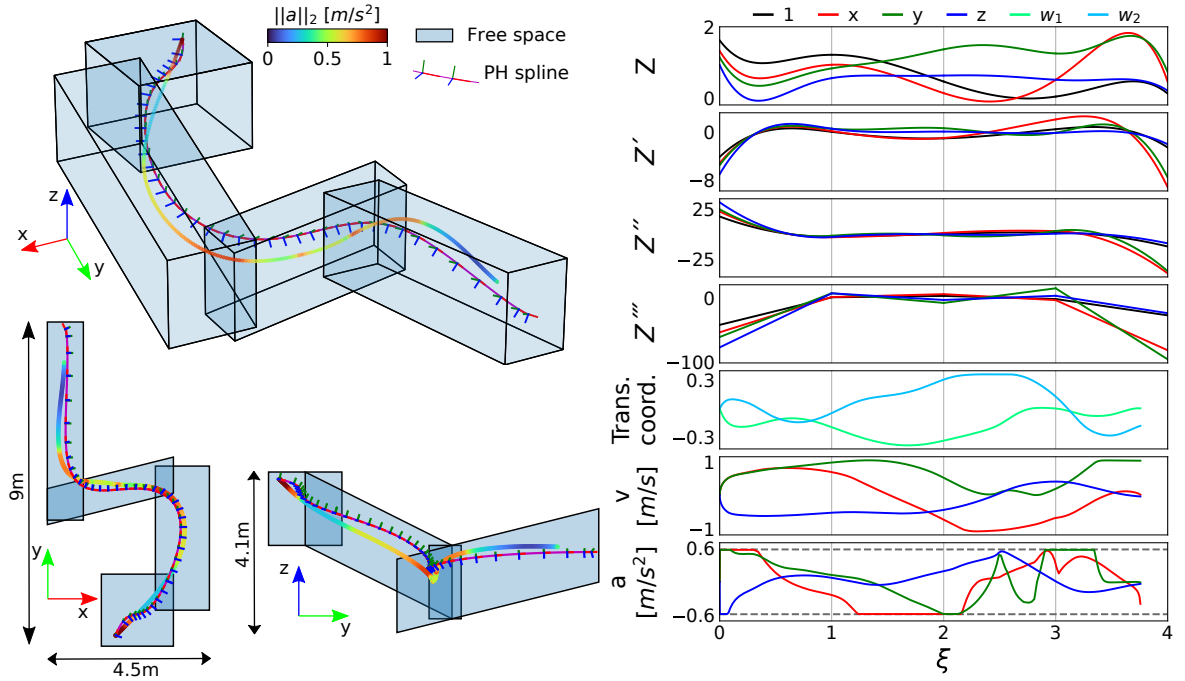


Fig. 3: Exemplary application of our two-stage spatial motion planning approach. *Left column:* Starting from the top in clock-wise direction, isometric-view, side-view and top-view. The blue polyhedrons represent the free space, the PH spline is depicted according to its adapted frame and the planned path is given by the colored line. The color mapping relates to the norm of the acceleration. *Right column:* The four divisions in the horizontal axis split the data according to the spline sections. The first four rows represent the quaternion polynomial and its derivatives, while the latter three rows show the system states (transverse coordinates and velocity), as well as the acceleration inputs.

We define a prediction horizon of $2\,\mathrm{s}$ with 40 shooting nodes, equivalent to a sampling time of 50 Hz. The weighting matrices are kept constant during all evaluations as $\lambda = 2$ and $R = 0.2I_3$, with I_3 referring to a 3×3 identity matrix. To resemble realistic motion dynamics, we limit the acceleration components to $\pm 0.58\,\mathrm{m/s^2}$, i.e., an approximated bound to $||\boldsymbol{a}_\mathrm{W}||_2 \leq 1\ \mathrm{m/s^2}$. All evaluations have been conducted on an Intel Core i7-10850H notebook.

B. Simulation results

Regarding stage 1, the spline of PH nonic curves obtained from solving the OCP in (21) is depicted on the left side of Fig. 3. Its cost function value f_PH in (20) is 3.56×10^{-5}, indicating that along the entire curve $\chi_1 \approx 0$, i.e., its adapted frame approximates an RMF. Intuitively, the moving frame portrayed in Fig. 3 appears to be free of unwanted twists along the curve's tangent. In addition, the underlying quaternion polynomials, as well as their respective derivatives, are also displayed in the first four rows at the right side of Fig. 3. Notice that these are plotted against path-parameter ξ, and thus, each vertical line depicts the transition between two successive sections of the spline. Taking this into account, these four graphs illustrate the aforementioned requirement that the quaternion polynomial $\boldsymbol{Z}(\xi)$ has to be C^3 in order for the path-related functions $\sigma(\xi)$, $\mathrm{R}(\xi)$, and $\boldsymbol{\omega}(\xi)$ to remain C^2 across the spline intersections.

TABLE I: Average computation times in milliseconds required for solving each stage. Notice that the initialization is not necessary if a prior solution is available.

Stage 1		Stage 2
Initialization*	Solve (21)	Solve (23)
27 ms	115 ms	4 ms

When it comes to stage 2, the trajectory computed from solving the NLP (23) in a receding horizon fashion is shown by the colored line in the left column of Fig. 3. The associated color mapping refers to the norm of the acceleration. The corresponding system states and inputs are attached in the last three rows of the right column. Notice that these states do not reach the end of the path, because the simulation is stopped as soon as the last node gets to the end of the free space. Given the behavior incited by the progress maximization cost function, the system fully exploits the actuation by seeking the optimal trade-off between speed and spatial bounds. This phenomenon is observable in the last graph of Fig. 3, where at least one of the acceleration components is saturated throughout the majority of the curve, as well as in the top-view and side-view, where the planned trajectory remains within the free space while (nearly) touching the apex of the curves.

The respective computation times for both stages are listed in Table I. Given that the initialization is only necessary in the first iteration, i.e., when no prior solution exists, stage 1 can run in between 5 to 10 Hz, whereas stage 2 can be

executed at up to 250 Hz. These timings allow for the deployment of our approach in a variety of motion planning tasks.

VI. CONCLUSION

In this work, we proposed a spatial motion planning control approach that allows to embed and leverage the environment's geometric properties in a prediction-based controller. For this purpose, we rely on a novel path-parameterization –agnostic to the path's adapted frame– of the three dimensional Euclidean coordinates. For an efficient usage of this model, we exploit the properties of PH curves, which allow for representing the environment in terms of smooth functions that are dependent on a small number of parameters. Taking this into account, we suggest a hierarchical scheme for geometrically constrained motion planning, where the coefficients obtained from computing a collision-free spline of PH curves are fed into an NMPC scheme, whose plant model is based on the aforementioned path-parameterization. The presented scheme has been evaluated in an illustrative example within a highly non-convex free space and a progress-maximization cost function. Results suggest that our approach not only efficiently converts the geometric properties of the environment into an approximated rotation minimizing PH spline, but it also exploits it to approximate a desired performance criterion.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Carolina Vittoria Beccari and Prof. Gudrun Albrecht, for the valuable help and discussions on spatial PH splines.

REFERENCES

[1] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, 1987.

[2] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proceedings of the 11th international symposium on advanced vehicle control*, no. 2, 2012, pp. 1–6.

[3] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 4136–4141.

[4] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear mpc in real-time," in *53rd IEEE conference on decision and control*. IEEE, 2014, pp. 2505–2510.

[5] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 324–14 329, 2020.

[6] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 477–482.

[7] R. Verschueren, N. van Duijkeren, J. Swevers, and M. Diehl, "Time-optimal motion planning for n-dof robot manipulators using a path-parametric system reformulation," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 2092–2097.

[8] S. Kumar and R. Gill, "Path following for quadrotors," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 2075–2081.

[9] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2017.

[10] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4044–4050.

[11] W. Kühnel, *Differential geometry*. American Mathematical Soc., 2015, vol. 77.

[12] W. Wang, B. Jüttler, D. Zheng, and Y. Liu, "Computation of rotation minimizing frames," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 1, pp. 1–18, 2008.

[13] R. T. Farouki, *Pythagorean—hodograph Curves*. Springer, 2008.

[14] R. L. Bishop, "There is more than one way to frame a curve," *The American Mathematical Monthly*, vol. 82, no. 3, pp. 246–251, 1975.

[15] R. Dietz, J. Hoschek, and B. Jüttler, "An algebraic approach to curves and surfaces on the sphere and on other quadrics," *Computer Aided Geometric Design*, vol. 10, no. 3-4, pp. 211–229, 1993.

[16] H. I. Choi and C. Y. Han, "Euler–rodrigues frames on spatial pythagorean-hodograph curves," *Computer Aided Geometric Design*, vol. 19, no. 8, pp. 603–620, 2002.

[17] R. Farouki, C. Giannelli, C. Manni, and A. Sestini, "Design of rational rotation–minimizing rigid body motions by hermite interpolation," *Mathematics of Computation*, vol. 81, no. 278, pp. 879–903, 2012.

[18] G. Otto, G. van Schoor, and K. R. Uren, "Geometric-dynamic trajectory: A quaternion pythagorean hodograph curves approach," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 2, pp. 283–294, 2021.

[19] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[20] G. Albrecht, C. V. Beccari, and L. Romani, "Spatial pythagorean-hodograph b–spline curves and 3d point data interpolation," *Computer Aided Geometric Design*, vol. 80, p. 101868, 2020.

[21] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[22] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[23] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[24] HSL. (2013) A collection of fortran codes for large scale scientific computation. [Online]. Available: http://www.hsl.rl.ac.uk

[25] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. SIAM, 2000.

[26] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[27] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl, "Towards a modular software package for embedded optimization," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 374–380, 2018.

[28] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[29] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

5 On the Generation of Relative Geometric References

In the previous chapter, we introduced a spatial parameterization of Cartesian motion that remains independent of both the underlying path parameterization and the chosen reference frame. This modular approach is particularly advantageous, as it allows us to decouple the motion's representation from the reference frame, offering greater flexibility in how we address egocentricity.

However, with this flexibility comes responsibility. The generalized spatial parameterization presented in the previous chapter naturally leads to the following question:

> *What is the most appropriate relative frame to use with the spatial parameterization introduced in Chapter 4?*

In this chapter, we address this question in three steps. First, we formally define the geometric reference as a parametric function with an attached moving frame. Second, we examine the problem of computing this frame, considering the available options. Finally, we present an efficient, simple, and compact algorithm for this task.

5.1 An overview of existing moving frames

The path-frame associated with the geometric reference Γ in (2.3) plays a crucial role in the formulation of path-parametric methods, as it enables decoupling the system dynamics into components tangential and orthogonal to the path. This necessitates a technique for augmenting the position function $\gamma(\theta)$ into a moving frame $\mathrm{R}(\theta)$—specifically, i.e., a method that requires only a parametric position reference to compute a frame that evolves correspondingly.

The problem of constructing such frames has been extensively studied, leading to multiple solutions in the literature, each distinguished by its choice of reference frame. The most prevalent options include the Frenet-Serret Frame (FSF) [49, 50], the Euler-Rodrigues Frame (ERF) [51, 52], and the Parallel Transport Frame (PTF) [53, 54, 55]. From a computational standpoint, FSF and ERF are analytically defined in terms of local derivatives, whereas PTF relies on global properties and typically requires numerical integration.

Among these, FSF is the most widely adopted due to its straightforward analytical formulation. However, it becomes singular when the reference path is a straight line (i.e., when curvature vanishes) and introduces unnecessary twisting along its first component, leading to nonlinearities when projecting system dynamics or defining a collision-free space around the path. An alternative analytical frame, the ERF, remains well-defined even for straight-line paths. However, it lacks a guaranteed twist-free property and depends on a

Pythagorean Hodograph curve, which is computationally demanding and often necessitates numerical methods. Consequently, the PTF is frequently preferred over FSF, as it is both singularity-free and twist-free, albeit at the cost of requiring numerical computations.

Existing methods for computing PTFs are discrete [55], limiting their ability to compute higher-order derivatives. Furthermore, they focus solely on deriving the rotation matrix while neglecting its angular velocity. However, as discussed in Chapter 3, contemporary path-parametric approaches extensively utilize learning and optimization techniques, which demand first- and second-order derivatives. To bridge this gap, this work introduces a computationally efficient and conceptually simple method for PTFs that not only ensures a singularity-free and twist-free path-frame but also explicitly accounts for its angular velocity and derivatives. To provide a comprehensive understanding, we first review alternative methods before presenting our approach.

Table 5.1 A comparison of methods for computing moving frames: Frenet Serret Frame (FSF), Parallel Transport Frame (PTF), Euler Rodrigues Frame (ERF).

Frame	Singularity-free	Twist-free
Frenet Serret	✗	✗
Euler Rodrigues	✓	✗
Parallel Transport	✓	✓

5.2 Choosing a moving frame

As Bishop famously noted, *there is more than one way to frame a curve* [53]. Building on this foundational insight, we revisit this topic with a modern perspective, focusing on three principal frames: the Frenet-Serret Frame (FSF), the Euler-Rodrigues Frame (ERF), and the Parallel Transport Frame (PTF). We examine their mathematical foundations and the distinct properties that arise from their respective formulations.

5.2.1 Frenet Serret Frame (FSF)

The Frenet-Serret frame is defined by constraining the second component to be the normalized derivative of the tangent and the third component to be orthogonal to the first two:

$$\boldsymbol{e_2}(\theta) = \frac{\boldsymbol{e_1'}(\theta)}{\|\boldsymbol{e_1'}(\theta)\|}, \quad \boldsymbol{e_3}(\theta) = \boldsymbol{e_1}(\theta) \times \boldsymbol{e_2}(\theta)\,. \tag{5.1a}$$

By incorporating the tangent component $\boldsymbol{e_1}$, the FSF can be explicitly defined in terms of the path function $\gamma(\theta)$ and its derivatives:

$$\begin{aligned} \boldsymbol{e_1}(\theta) &= \frac{\boldsymbol{\gamma}'(\theta)}{\|\boldsymbol{\gamma}'(\theta)\|}, \quad \boldsymbol{e_2}(\theta) = \frac{\boldsymbol{\gamma}'(\theta) \times \left(\boldsymbol{\gamma}''(\theta) \times \boldsymbol{\gamma}'(\theta)\right)}{\|\boldsymbol{\gamma}'(\theta)\| \cdot \|\boldsymbol{\gamma}''(\theta) \times \boldsymbol{\gamma}'(\theta)\|}, \\ \boldsymbol{e_3}(\theta) &= \frac{\boldsymbol{\gamma}'(\theta) \times \boldsymbol{\gamma}''(\theta)}{\|\boldsymbol{\gamma}'(\theta) \times \boldsymbol{\gamma}''(\theta)\|}\,. \end{aligned} \tag{5.1b}$$

Since the frame depends exclusively on derivatives at the evaluation point, the FSF is a *local frame*, meaning it can be computed using only local derivative information. However, the

second component is undefined when $\boldsymbol{\gamma}'$ and $\boldsymbol{\gamma}''$ are parallel. To interpret this, we analyze the angular velocity of the frame, $\boldsymbol{\omega}_{\text{FSF}}$. Using equations (2.5) and (5.1), we obtain:

$$\omega^{\Gamma}_{\text{FSF},1} = \boldsymbol{e}_{\boldsymbol{2}}'(\theta) \cdot \boldsymbol{e}_{\boldsymbol{3}}(\theta) = \sigma(\theta)\tau(\theta)\,, \tag{5.2a}$$

$$\omega^{\Gamma}_{\text{FSF},2} = \boldsymbol{e}_{\boldsymbol{3}}'(\theta) \cdot \boldsymbol{e}_{\boldsymbol{1}}(\theta) = 0\,, \tag{5.2b}$$

$$\omega^{\Gamma}_{\text{FSF},3} = \boldsymbol{e}_{\boldsymbol{1}}'(\theta) \cdot \boldsymbol{e}_{\boldsymbol{2}}(\theta) = \sigma(\theta)\kappa(\theta)\,. \tag{5.2c}$$

The first and third components correspond to the *torsion* $\tau(\theta)$ and *curvature* $\kappa(\theta)$, respectively, which characterize how the curve twists along the first component or bends along the third component. Mathematically, they are given by:

$$\tau(\theta) = \frac{\omega^{\Gamma}_{\text{FSF},1}(\theta)}{\sigma(\theta)} = \frac{\|(\boldsymbol{\gamma}'(\theta) \times \boldsymbol{\gamma}''(\theta)) \cdot \boldsymbol{\gamma}'''(\theta)\|}{\|\boldsymbol{\gamma}'(\theta) \times \boldsymbol{\gamma}''(\theta)\|^2}, \tag{5.3}$$

$$\kappa(\theta) = \frac{\omega^{\Gamma}_{\text{FSF},3}(\theta)}{\sigma(\theta)} = \frac{\|\boldsymbol{\gamma}'(\theta) \times \boldsymbol{\gamma}''(\theta)\|}{\|\boldsymbol{\gamma}'(\theta)\|^3}\,. \tag{5.4}$$

Since the parametric speed $\sigma(\theta)$ is decoupled from curvature and torsion, these quantities are invariant to reparameterization. Combining these results, the angular velocity of the frame is given by:

$$\begin{aligned}\boldsymbol{\omega}_{\text{FSF}}(\theta) &= \omega^{\Gamma}_{\text{FSF},1}(\theta)\boldsymbol{e}_{\boldsymbol{1}}(\theta) + \omega^{\Gamma}_{\text{FSF},3}(\theta)\boldsymbol{e}_{\boldsymbol{3}}(\theta) \\ &= \sigma(\theta)\left(\tau(\theta)\boldsymbol{e}_{\boldsymbol{1}}(\theta) + \kappa(\theta)\boldsymbol{e}_{\boldsymbol{2}}(\theta)\right).\end{aligned} \tag{5.5}$$

From (5.2), it is evident that the FSF is characterized by $\omega^{\Gamma}_{\text{FSF},2} = 0$, meaning the second component always points toward the center of curvature. This implies that (i) the frame becomes singular when curvature vanishes, and (ii) it flips when the path transitions between left and right turns. Additionally, the FSF exhibits a twist along the tangent component, leading to non-physical motion of the frame. Consequently, alternative moving frames are explored to address these limitations.

5.2.2 Euler Rodrigues Frame (ERF)

The Euler-Rodrigues Frame (ERF) provides a potential solution to the limitations of the FSF. It is fully defined, meaning it remains singularity-free even when the reference path is straight. Additionally, it is a local frame, as it can be computed solely from local derivative information. However, the ERF is constructed based on a Pythagorean Hodograph (PH) curve, a specialized class of polynomials characterized by the property that the parametric speed $\sigma(\theta)$ is itself a polynomial in the path parameter θ. Enforcing this condition results in a family of polynomials whose geometric properties—such as arclength, curvature, and torsion—can be computed in closed form using only rational functions. An additional advantage of PH curves is that they naturally inherit an adapted frame, the ERF, which is fully defined and straightforward to compute for a given PH curve.

Despite these advantages, converting a general reference path $\boldsymbol{\gamma}(\theta)$ into a PH curve is nontrivial and often requires numerical routines. Ensuring differentiability and continuity of both the ERF and its angular velocity further increases the complexity of the conversion. Moreover, the algebra and calculus involved in the construction of PH curves and their associated ERFs are intricate, posing a barrier to widespread adoption within the planning and control communities. For further details on PH curves and ERFs, refer to [56] and [51],

respectively. Applications of PH curves and ERFs in autonomous navigation can be found in the manuscripts attached to the previous and the following chapters, in subsections 4.3.1 and 7.4.2, respectively. Finally, a real-time, twice-differentiable algorithm for constructing PH curves and ERFs for path-parametric planning and control is presented in the manuscript accompanying this chapter in subsection 5.4.1.

5.2.3 Parallel Transform Frame (PTF)

The Parallel Transport Frame (PTF) addresses the limitations of both the Frenet–Serret Frame (FSF) and the Euler–Rodrigues Frame (ERF). Unlike these alternatives, the PTF is not only well-defined and potentially twist-free but also straightforward to implement, ensuring computational efficiency, simplicity, and reproducibility. These advantages establish PTFs as the optimal choice for path-parametric planning and control applications. While existing literature [54, 55, 39] discusses various aspects of their construction, none explicitly examines the differentiability of the moving frame or the computation of its angular velocity. To fill this gap, we introduce a concise, efficient, and easy-to-implement algorithm that, given a parametric curve $\boldsymbol{\gamma}(\theta)$, computes the corresponding PTF $\mathrm{R}_{\mathrm{PTF}}(\theta)$ and angular velocity $\boldsymbol{\omega}_{\mathrm{PTF}}(\theta)$, including their derivatives. Before detailing this method, we first review the fundamentals of the PTF.

The concept of the PTF stems from the influential paper *There is more than one way to frame a curve* [53], which highlights the flexibility in selecting the second ($\boldsymbol{e_2}$) and third ($\boldsymbol{e_3}$) components of the moving frame. These can be freely chosen within the orthogonal plane, provided they form an orthonormal basis with the tangent vector $\boldsymbol{e_1}$. By constraining the second and third components to a parallel vector field, they rotate only as necessary to maintain orthogonality with the tangent. Mathematically, this is equivalent to requiring that their derivatives align with the tangential unit vector:

$$\boldsymbol{e}_2'(\theta) = k_1(\theta)\boldsymbol{e_1}(\theta), \quad \boldsymbol{e}_3'(\theta) = k_2(\theta)\boldsymbol{e_1}(\theta). \tag{5.6a}$$

To determine the auxiliary variables k_1 and k_2, we combine (5.6a) with the orthonormality constraints:

$$0 = 1 - \boldsymbol{e}_2^{\intercal}(\theta)\boldsymbol{e_2}(\theta), \quad 0 = 1 - \boldsymbol{e}_3^{\intercal}(\theta)\boldsymbol{e_3}(\theta), \tag{5.6b}$$

$$0 = \boldsymbol{e}_1^{\intercal}(\theta)\boldsymbol{e_2}(\theta), \quad 0 = \boldsymbol{e}_1^{\intercal}(\theta)\boldsymbol{e_3}(\theta). \tag{5.6c}$$

Differentiating (5.6c) with respect to the path parameter θ, substituting (5.6a), and multiplying by $\boldsymbol{e_1}(\theta)$, we obtain:

$$k_1(\theta) = -\boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e_2}(\theta), \quad k_2(\theta) = -\boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e_3}(\theta),$$

which yields the following governing equations:

$$\begin{aligned} \boldsymbol{e}_2'(\theta) &= \left(-\boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e_2}(\theta)\right)\boldsymbol{e_1}(\theta), \\ \boldsymbol{e}_3'(\theta) &= \left(-\boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e_3}(\theta)\right)\boldsymbol{e_1}(\theta). \end{aligned} \tag{5.7}$$

Equations (5.7) provide the foundation for computing the PTF's angular velocity. By integrating them with the general moving frame angular velocity definition in (2.7), we derive:

$$\begin{aligned}
\omega^{\Gamma}_{\mathrm{PTF},1}(\theta) &= \boldsymbol{e}_2'(\theta)^{\intercal}\boldsymbol{e}_3(\theta) = \left(-\boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e}_2(\theta)\boldsymbol{e}_1(\theta)\right)^{\intercal}\boldsymbol{e}_3(\theta) = 0,\\
\omega^{\Gamma}_{\mathrm{PTF},2}(\theta) &= \boldsymbol{e}_3'(\theta)^{\intercal}\boldsymbol{e}_1(\theta) = \left(-\boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e}_3(\theta)\boldsymbol{e}_1(\theta)\right)^{\intercal}\boldsymbol{e}_1(\theta) = -\boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e}_3(\theta),\\
\omega^{\Gamma}_{\mathrm{PTF},3}(\theta) &= \boldsymbol{e}_1'(\theta)^{\intercal}\boldsymbol{e}_2(\theta),
\end{aligned} \tag{5.8a}$$

where $(\cdot)^{\Gamma}$ denotes the angular velocity expressed in the path frame Γ. Transforming to the world frame as defined in (2.5), we obtain:

$$\boldsymbol{\omega}_{\mathrm{PTF}}(\theta) = \boldsymbol{\omega}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}^{\intercal}_{\mathrm{PTF}}(\theta) = \omega^{\Gamma}_{\mathrm{PTF},2}(\theta)\boldsymbol{e}_2(\theta) + \omega^{\Gamma}_{\mathrm{PTF},3}(\theta)\boldsymbol{e}_3(\theta). \tag{5.8b}$$

From eqs. (5.8), two key observations emerge. First, unlike the FSF in (5.2), the PTF's angular velocity lacks a tangential component ($\omega^{\Gamma}_{\mathrm{PTF},1} = 0$), confirming its twist-free property. Second, $\boldsymbol{\omega}_{\mathrm{PTF}}(\theta)$ depends solely on the second and third components of $\mathrm{R}_{\mathrm{PTF}}(\theta)$ and the derivative of the tangent vector $\boldsymbol{e}_1'(\theta)$. Consequently, given an initial frame $\mathrm{R}_{\mathrm{PTF}}(\theta_0) = [\boldsymbol{e}_1(\theta_0), \boldsymbol{e}_2(\theta_0), \boldsymbol{e}_3(\theta_0)]$, the PTF at any point along the curve can be computed via forward integration:

$$\mathrm{R}_{\mathrm{PTF}}(\theta_{i+1}) = e^{\Omega_{\mathrm{PTF}}(\theta_i)\Delta\theta_i}\mathrm{R}_{\mathrm{PTF}}(\theta_i), \tag{5.9}$$

where $\Delta\theta_i = \theta_{i+1} - \theta_i$ and $\Omega_{\mathrm{PTF}}(\theta_i)$ is the skew-symmetric matrix of $\boldsymbol{\omega}_{\mathrm{PTF}}$ (see (2.6)). Since $\boldsymbol{e}_1'(\theta)$ drives the integration, the reference path $\boldsymbol{\gamma}(\theta)$ must be at least C^2-continuous. Moreover, performing the integration in $\mathrm{SO}(3)$ guarantees orthonormality of the frame, unlike conventional methods such as Euler or Runge-Kutta [57]. The complete procedure for computing the PTF and its angular velocity is outlined in Algorithm 1.

PARALLEL TRANSPORT FRAME INTEGRATION

Algorithm 1 *Parallel Transport Frame Integration* (**PTFI**):
Input: $\theta_{0,\dots,N}$, $\mathrm{R}_{\mathrm{PTF}}(\theta_0)$, $\boldsymbol{e}_1'(\theta_{0,\dots,N})$
Output: $\mathrm{R}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\boldsymbol{\omega}_{\mathrm{PTF}}(\theta_{0,\dots,N})$

function PTFI($\theta_{0,\dots,N}$, $\mathrm{R}_{\mathrm{PTF}}(\theta_0)$, $\boldsymbol{e}_1'(\theta_{0,\dots,N})$)
  for $i \in \{0, \dots, N-1\}$ **do**
    $\boldsymbol{\omega}_{\mathrm{PTF},i} \leftarrow$ AngVel($\boldsymbol{e}_{1,i}'$, $\mathrm{R}_{\mathrm{PTF},i}$) (5.8)
    $\Omega \leftarrow$ SkewMatrix($\boldsymbol{\omega}_{\mathrm{PTF},i}$) (2.6)
    $\Delta\theta = \theta_{i+1} - \theta_i$
    $\mathrm{R}_{\mathrm{PTF},i} \leftarrow$ Integr($\mathrm{R}_{\mathrm{PTF},i}, \Omega, \Delta\theta$) (5.9)
  end for
  return $\mathrm{R}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\boldsymbol{\omega}_{\mathrm{PTF}}(\theta_{0,\dots,N})$
end function

With the PTF components $\mathrm{R}_{\mathrm{PTF}}$ and angular velocity $\boldsymbol{\omega}_{\mathrm{PTF}}$ determined, we now derive their higher-order derivatives. Since path-parametric methods typically require first-order (for learning) and second-order (for optimization) gradients, we focus on computing these derivatives, though the approach can be extended to arbitrary orders through repeated application of the same procedure.

The first derivative $\mathrm{R}'_{\mathrm{PTF}}$ follows directly from (2.6) and (5.8). The angular acceleration $\boldsymbol{\alpha}_{\mathrm{PTF}}$ is obtained by differentiating (5.8):

$$\boldsymbol{\alpha}_{\mathrm{PTF}}(\theta) = \boldsymbol{\alpha}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}^{\intercal}_{\mathrm{PTF}}(\theta) + \boldsymbol{\omega}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}'^{\intercal}_{\mathrm{PTF}}(\theta), \tag{5.10a}$$

where

$$\boldsymbol{\alpha}^{\Gamma}_{\mathrm{PTF}}(\theta) = \left[0,\ -\left(\boldsymbol{e}''_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}_{\boldsymbol{3}}(\theta) + \boldsymbol{e}'_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}'_{\boldsymbol{3}}(\theta)\right),\ \left(\boldsymbol{e}''_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}_{\boldsymbol{2}}(\theta) + \boldsymbol{e}'_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}'_{\boldsymbol{2}}(\theta)\right)\right]. \tag{5.10b}$$

For the second derivative $\mathrm{R}''_{\mathrm{PTF}}$, we express the angular velocity (5.8) and acceleration (5.10) through their skew-symmetric matrices Ω_{PTF} and Ω'_{PTF}, respectively. Combining these with the derivative of (2.6) yields:

$$\mathrm{R}''_{\mathrm{PTF}}(\theta) = \Omega'_{\mathrm{PTF}}(\theta)\mathrm{R}_{\mathrm{PTF}}(\theta) + \Omega_{\mathrm{PTF}}(\theta)\mathrm{R}'_{\mathrm{PTF}}(\theta). \tag{5.11}$$

At last, to compute the second derivative of angular velocity (angular jerk $\boldsymbol{j}_{\Gamma}$), we differentiate (5.10):

$$\boldsymbol{j}_{\mathrm{PTF}}(\theta) = \boldsymbol{j}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}^{\intercal}_{\mathrm{PTF}}(\theta) + 2\boldsymbol{\alpha}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}'^{\intercal}_{\mathrm{PTF}}(\theta) + \boldsymbol{\omega}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}''^{\intercal}_{\mathrm{PTF}}(\theta), \tag{5.12a}$$

where the path-frame angular jerk components are:

$$\begin{aligned}\boldsymbol{j}^{\Gamma}_{\mathrm{PTF}}(\theta) = [0, &-\left(\boldsymbol{e}'''_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}_{\boldsymbol{3}}(\theta) + \boldsymbol{e}''_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}'_{\boldsymbol{3}}(\theta) + \boldsymbol{e}''_{\boldsymbol{3}}(\theta)^{\intercal}\boldsymbol{e}'_{\boldsymbol{1}}(\theta) + \boldsymbol{e}'_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}''_{\boldsymbol{3}}(\theta)\right),\\ &\boldsymbol{e}'''_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}_{\boldsymbol{2}}(\theta) + \boldsymbol{e}''_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}'_{\boldsymbol{2}}(\theta) + \boldsymbol{e}'_{\boldsymbol{1}}(\theta)^{\intercal}\boldsymbol{e}''_{\boldsymbol{2}}(\theta)].\end{aligned} \tag{5.12b}$$

Eqs. (5.8), (5.10), and (5.12) reveal the relationship between the continuity of the reference path $\boldsymbol{\gamma}(\theta)$ and its associated angular frame $\boldsymbol{\omega}(\xi)$:

$$\boldsymbol{e}'_{\boldsymbol{1}}(\theta),\ \mathrm{R}_{\mathrm{PTF}}(\theta) \rightarrow \boldsymbol{\omega}_{\mathrm{PTF}}(\theta), \tag{5.13a}$$

$$\boldsymbol{e}''_{\boldsymbol{1}}(\theta),\ \mathrm{R}_{\mathrm{PTF}}(\theta),\ \mathrm{R}'_{\mathrm{PTF}}(\theta) \rightarrow \boldsymbol{\alpha}_{\mathrm{PTF}}(\theta), \tag{5.13b}$$

$$\boldsymbol{e}'''_{\boldsymbol{1}}(\theta),\ \mathrm{R}_{\mathrm{PTF}}(\theta),\ \mathrm{R}'_{\mathrm{PTF}}(\theta),\ \mathrm{R}''_{\mathrm{PTF}}(\theta) \rightarrow \boldsymbol{j}_{\mathrm{PTF}}(\theta), \tag{5.13c}$$

which establishes that for a path $\boldsymbol{\gamma} \in C^n$, the resulting PTF satisfies $\mathrm{R}_{\mathrm{PTF}} \in C^{n-1}$ and $\boldsymbol{\omega}_{\mathrm{PTF}} \in C^{n-2}$. For instance, achieving $\boldsymbol{\omega}_{\mathrm{PTF}} \in C^2$ continuity requires the path to be at least C^4-continuous. The complete procedure for computing first and second-order derivatives of both the moving frame components and angular velocity is presented in Algorithm 2.

PARALLEL TRANSPORT FRAME DERIVATION

Algorithm 2 *Parallel Transport Frame Derivation* (**PTFD**):
Input: $\mathrm{R}_{\mathrm{PTF}}$, $\boldsymbol{\omega}_{\mathrm{PTF}}$, $\boldsymbol{e}_{\mathbf{1}}^{'}$, $\boldsymbol{e}_{\mathbf{1}}^{''}$, $\boldsymbol{e}_{\mathbf{1}}^{'''}$ eval. at $\theta_{0,\dots,N}$
Output: $\mathrm{R}_{\mathrm{PTF}}^{'}$, , $\mathrm{R}_{\mathrm{PTF}}^{''}$, $\boldsymbol{\alpha}_{\mathrm{PTF}}$, $\boldsymbol{j}_{\mathrm{PTF}}$ eval. at $\theta_{0,\dots,N}$

function PTFD($\mathrm{R}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\boldsymbol{\omega}_{\mathrm{PTF}}(\theta_{0,\dots,N})$)
  for $i \in \{0, \dots, N\}$ **do**
    $\mathrm{R}_{\mathrm{PTF,i}}^{'} \leftarrow$ FRAMEVEL($\mathrm{R}_{\mathrm{PTF,i}}, \boldsymbol{\omega}_{\mathrm{PTF,i}}$) (2.6)
    $\boldsymbol{\alpha}_{\mathrm{PTF,i}} \leftarrow$ ANGACC($\boldsymbol{e}_{\mathbf{1},\boldsymbol{i}}^{''}, \mathrm{R}_{\mathrm{PTF},i}, \mathrm{R}_{\mathrm{PTF,i}}^{'}$) (5.10)
    $\mathrm{R}_{\mathrm{PTF,i}}^{''} \leftarrow$ FRAMEACC($\mathrm{R}_{\mathrm{PTF,i}}, \boldsymbol{\omega}_{\mathrm{PTF},i}$ $\boldsymbol{\alpha}_{\mathrm{PTF},i}$) (5.11)
    $\boldsymbol{j}_{\mathrm{PTF},i} \leftarrow$ ANGJERK($\boldsymbol{e}_{\mathbf{1},\boldsymbol{i}}^{'''}, \mathrm{R}_{\mathrm{PTF,i}}$, $\mathrm{R}_{\mathrm{PTF},i}^{'}, \mathrm{R}_{\mathrm{PTF},i}^{''}$) (5.12)
  end for
  return $\mathrm{R}_{\mathrm{PTF}}^{'}(\theta_{0,\dots,N})$, $\mathrm{R}_{\mathrm{PTF}}^{''}(\theta_{0,\dots,N})$, $\boldsymbol{\alpha}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\boldsymbol{j}_{\mathrm{PTF}}(\theta_{0,\dots,N})$
end function

In summary, Algorithms 1 and 2 enable the efficient computation of the moving frame $\mathrm{R}_{\mathrm{RMF}}$, the angular velocity $\boldsymbol{\omega}_{\mathrm{RMF}}$, and their derivatives $\{\mathrm{R}_{\mathrm{RMF}}^{'}, \mathrm{R}_{\mathrm{RMF}}^{''}, \boldsymbol{\alpha}_{\mathrm{RMF}}, \boldsymbol{j}_{\mathrm{RMF}}\}$. This approach combines accessibility with computational efficiency, while leveraging the inherent advantages of the PTFs—namely, their singularity-free and twist-free properties. As a result, the proposed method for computing the moving frame is highly suitable for egocentric planning and control algorithms.

5.3 Modularity: Frames and Equations

While we have made every effort to select an appropriate frame and parameterization technique, I acknowledge the possibility that more effective methods may be developed in the future. It is likely that subsequent research will introduce improved approaches for defining and computing moving frames. Nevertheless, it is essential to emphasize that the equations of motion presented in eqs. (4.9), derived in the previous chapter, remain valid and relevant. Notably, the content of the current and preceding chapters is entirely independent—the path parameterization method does not rely on the specific equations of motion governing the spatial states. As a result, even if future advancements yield superior moving frames, they can still be seamlessly integrated with the equations of motion in (4.9).

5.4 Enclosed Publication

A scientific publication discussing the computation and application of the ERF through PH curves is attached to this chapter and cited in [3]. The code associated with this publication is available at `https://github.com/jonarriza96/phodcos`, and a recording of the conference talk can be viewed at `https://youtu.be/Bh2_ISXp8b4`. If you are interested in a scientific publication showcasing the usage of PTFs, please refer to Chapter 8.

PHODCOS: Pythagorean Hodograph-based Differentiable Coordinate System

Jon Arrizabalaga, Fausto Vega, Zbyněk ŠÍR, Zachary Manchester and Markus Ryll

published in

IEEE Aerospace Conference

2025

Contribution This paper introduces the Pythagorean Hodograph-based Coordinate System (PHODCOS), a moving frame that satisfies all the desirable properties discussed throughout this chapter. The interpolation algorithm extends previous work by Zbyněk Šír, and the application to the Near-Rectilinear Halo Orbit (NRHO) was proposed by Zachary Manchester. My role was to connect these contributions: I integrated the underlying concepts, designed and implemented the complete system in an open-source repository, and conducted the experiments that support the results presented.

PHODCOS: Pythagorean Hodograph-based Differentiable Coordinate System

Jon Arrizabalaga
Technical University of Munich
Carnegie Mellon University
Pittsburgh, PA 15213
jon.arrizabalaga@tum.de

Fausto Vega
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
fvega@andrew.cmu.edu

Zbyněk Šír
Faculty of Mathematics and Physic
Charles University
Prague, 186 75
zbynek.sir@mff.cuni.cz

Zachary Manchester
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
zacm@cmu.edu

Markus Ryll
Dept. of Aerospace and Geodesy
Technical University of Munich
Munich, 80333
markus.ryll@tum.de

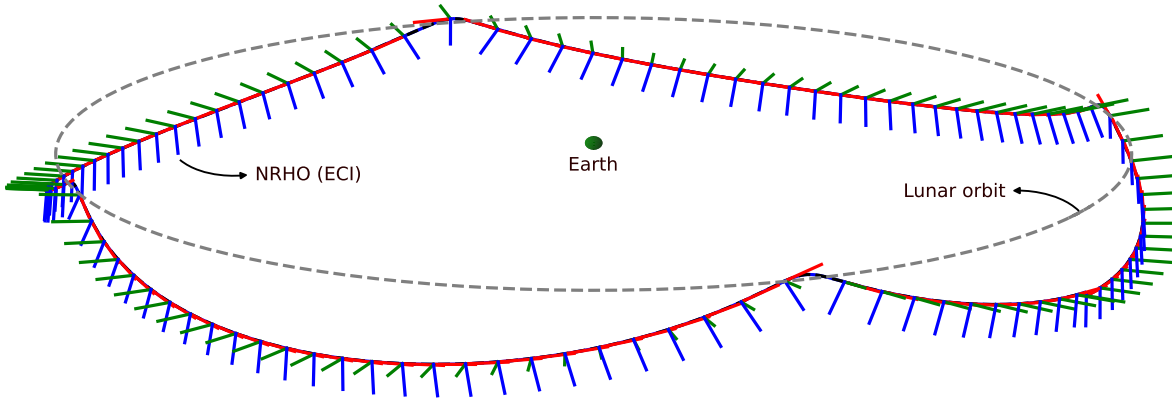


Figure 1: An illustrative example of the PHODCOS coordinate system applied to a highly nonlinear curve, specifically the Near Rectilinear Halo Orbit (NRHO) selected for the Lunar Gateway. The frame components are colored red, blue, and green, respectively. The Earth is represented by the green sphere in the middle, whereas the Moon's orbit is given by the dashed gray line. This depiction is presented in the Earth-Centered Inertial (ECI) frame.

***Abstract*— This paper presents PHODCOS, an algorithm that assigns a moving coordinate system to a given curve. The parametric functions underlying the coordinate system, i.e., the path function, the moving frame and its angular velocity, are exact – approximation free – differentiable, and sufficiently continuous. This allows for computing a coordinate system for highly nonlinear curves, while remaining compliant with autonomous navigation algorithms that require first and second order gradient information. In addition, the coordinate system obtained by PHODCOS is fully defined by a finite number of coefficients, which may then be used to compute additional geometric properties of the curve, such as arc-length, curvature, torsion, etc. Therefore, PHODCOS presents an appealing paradigm to enhance the geometrical awareness of existing guidance and navigation on-orbit spacecraft maneuvers. The PHODCOS algorithm is presented alongside an analysis of its error and approximation order, and thus, it is guaranteed that the obtained coordinate system matches the given curve within a desired tolerance. To demonstrate the applicability of the coordinate system resulting from PHODCOS, we present numerical examples in the Near Rectilinear Halo Orbit (NRHO) for the Lunar Gateway.**

Code: `https://github.com/jonarriza96/phodcos`

979-8-3503-5597-0/25/$31.00 ©2025 IEEE

TABLE OF CONTENTS

1. INTRODUCTION

Understanding and characterizing the intricate shapes and motions present in nature is a fundamental goal in both science and engineering. Although the familiar Cartesian representation excels at depicting simple lines and circles, it frequently falls short in capturing the complexities of curves and surfaces seen in the real world. Consequently, developing more adaptable tools that can represent the spatiotemporal aspects of physical phenomena has been crucial in the study of dynamic systems and their control algorithms.

Within this scope, modeling the behavior of a dynamical system relies on three design choices: 1) the *states* that parameterize the system, 2) an *evolution variable* upon which the system progresses, e.g. time or space, and 3) a *reference* state that quantifies the motions of the system in a relative fashion. In the specific case where the states describe the motion of a body's position and/or orientation, this reference becomes a *reference frame*, which can be stationary or moving. The optimal choice of these three ingredients is highly dependent on the particular system and its intended application.

Restricting ourselves to body motions, of all three ingredients, the first and second are specific to the system. Nevertheless, the third ingredient –the selection of a reference frame– is universal, as it remains independent of the governing equations of motion. For this reason, we propose a method to assign a *differentiable moving frame* to any given curve. The suggested algorithm shows attractive attributes to be used as a reference frame for dynamical systems, as well as for characterizing complex shapes and their geometrical properties.

In particular, this paper presents **PHODCOS**: a **P**ythagorean **HO**dograph-based **D**ifferentiable **CO**ordinate **S**ystem. The moving frame resulting from this algorithm is characterized by the following four properties: First, it is built upon *Pythagorean-Hodograph (PH) curves*, a class of polynomials that efficiently parameterize geometric features like arc-length, curvature, torsion, moving frame components and its angular velocity. Second, the algorithm operates using closed-form analytical expressions, avoiding numerical or iterative processes, making it *computationally efficient*. Third, the conversion *error is bounded*, ensuring that the coordinate system aligns with the curve within a defined tolerance. Fourth, the moving frame and angular velocity are *twice differentiable*, making the coordinate system compliant with algorithms based on learning or optimization. To achieve this, our method relies on four main contributions:

1. We extend the C^2 hermite interpolation method presented in [1] to C^4 and introduce a closure mechanism, thereby, ensuring that the coordinate system is twice differentiable.
2. We conduct an approximation order analysis and provide a theoretical proof that guarantees the coordinate system to be within a desired tolerance of the curve.
3. We showcase the method's applicability in a highly nonlinear exemplary curve: the Near Rectilinear Halo Orbit (NRHO) for the Lunar Gateway.
4. We open source the implementation of the algorithm, alongside the numerical analysis and examples shown in this paper.

The remainder of this manuscript is structured as follows: Section 2 provides an overview of existing methods for assigning a moving frame to a given curve. Section 3 formally defines the problem solved in this paper. After introducing the required preliminaries in Section 4, Section 5 presents the PHODCOS algorithm, by providing its derivation and a numerical analysis. Section 6 shows an illustrative example by testing PHODCOS in the Lunar Gateway before Section 7 presents the conclusions.

2. HOW TO FRAME A CURVE?

The assignment of a moving frame to a curve is a well-studied problem, with several solutions available in the existing literature. The most commonly used are the Frenet Serret Frame (FSF) [2, 3] and the Parallel Transport Frame (PTF) [4–6]. From a computational perspective, the FSF is purely analytical, as it is derived from local derivative information in closed form. In contrast, the PTF depends on global information and requires a numerical method for its computation.

The FSF is frequently chosen in the literature because of its straightforward analytical nature. However, it fails when the reference path is straight (i.e., zero curvature) and introduces an unnecessary twist to its first component. The PTF is the most recognized alternative, as it avoids both singularities and twists. Yet, current techniques for calculating PTFs [6] only focus on the rotation matrix and neglect the angular velocity. Moreover, these methods are discrete, preventing the computation of higher-order derivatives. In the age of data-driven methods, prominent decision-making algorithms increasingly rely on learning and optimization [7, 8], making this constraint particularly noteworthy.

To address these challenges, PHODCOS opts for a lesser-known yet promising alternative, the Euler Rodrigues Frame (ERF) [9]. There are two main reasons for the ERF's reduced visibility compared to the more established counterparts: First, because they depend on PH curves and their algebra, extracting an ERF from any arbitrary curve while keeping a bounded approximation error is a daunting task [10]. Broadly speaking, existing methods either rely on numerical and iterative methods, resulting in computationally expensive algorithms without any guarantees on the approximation error [11] or are given in closed-form, but require certain assumptions over the given curve, limiting their universality [12]. The second reason also relates to the ERF's inherent complex algebra, as it significantly raises the entry barrier, thus hampering its use in other scientific or engineering contexts [13–16].

In this work, we introduce a technique that addresses both major limitations. For the first drawback, which involves the universal and efficient computation of an ERF frame, we utilize a hermite interpolation method similar to that in [1]. This approach is expressed in closed form, eliminating the need for iterative or optimization procedures, and can be applied to any C^4 curve, a requirement that is easily met. Concerning the second limitation related to usability, we offer an open-source implementation of our technique, equipping practitioners with a practical toolkit and facilitating its application without requiring a deep understanding of the underlying mathematical concepts.

3. PROBLEM STATEMENT

Let Γ refer to a geometric reference and be defined as a curve whose position is given by a function $\boldsymbol{\gamma} : \mathbb{R} \to \mathbb{R}^3$ that depends on path parameter ξ

$$\Gamma = \{\xi \in [\xi_0, \xi_f] \subseteq \mathbb{R} \to \boldsymbol{\gamma}(\xi) \in \mathbb{R}^3\}, \tag{1}$$

which is assumed to be of differentiability class C^m, i.e., $\boldsymbol{\gamma} \in C^m(\xi)$, where $m \geq 2$.

Problem 1: *Given path Γ in* (1)*, design an algorithm that approximates it by*

- ***P1*** *finding a parametric path-function $\boldsymbol{p} : \mathbb{R} \to \mathbb{R}^3$ that guarantees the approximation error to remain within a desired tolerance ϵ:*

$$E = \max_{\xi \in [\xi_0, \xi_f]} ||\boldsymbol{\gamma}(\xi) - \boldsymbol{p}(\xi)|| \leq \epsilon \tag{2}$$

- ***P2*** *associating an adapted-frame[4], whose components are given by a parametric frame-function $R : \mathbb{R} \to \mathbb{R}^{3x3}$, and its angular velocity is defined by a parametric omega-function $\boldsymbol{\omega} : \mathbb{R} \to \mathbb{R}^3$,*

- ***P3*** *maintaining a minimum differentiability class of C^2 for all parametric-functions, i.e., $\{\boldsymbol{p}, R, \boldsymbol{\omega}\} \in C^m(\xi)$ with $m \geq 2$.*

4. PRELIMINARIES

The algorithm proposed for the presented in Section 3 leverages Pythagorean Hodograph (PH) curves, a subset of polynomials whose algebra relies on quaternions. To account for this, in this section we outline the required mathematical notation and background.

A. Quaternions

Quaternion algebra considers a four-dimensional vector space and is expressed in the form

$$\mathcal{A} = a + a_x\,\mathbf{i} + a_y\,\mathbf{j} + a_z\,\mathbf{k}\,, \tag{3}$$

where $\{a, a_x, a_y, a_z\} \in \mathbb{R}$ and the canonical basis $\{\mathbf{1} = (1,0,0,0), \mathbf{i} = (0,1,0,0), \mathbf{j} = (0,0,1,0), \mathbf{k} = (0,0,0,1)\}$ fulfills the following multiplication rules:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i}\,\mathbf{j}\,\mathbf{k} = -1\,,$$
$$\mathbf{i}\,\mathbf{j} = -\,\mathbf{j}\,\mathbf{i} = \mathbf{k},\ \mathbf{j}\,\mathbf{k} = -\,\mathbf{k}\,\mathbf{j} = \mathbf{i},\ \mathbf{k}\,\mathbf{i} = -\,\mathbf{i}\,\mathbf{k} = \mathbf{j}\ .$$

For a more detailed introduction, please refer to [17]. The conjugate of quaternion (3) is $\mathcal{A}^* = a - a_x\,\mathbf{i} - a_y\,\mathbf{j} - a_z\boldsymbol{z}$ and its absolute value is defined as

$$|\mathcal{A}| = \sqrt{\mathcal{A}\mathcal{A}^*} = \sqrt{\mathcal{A}^*\mathcal{A}} = \sqrt{a^2 + a_x^2 + a_y^2 + a_z^2}\,.$$

Unit quaternions, characterized by $|A| = 1$, are a singularity-free and minimal representation of SO(3). We refer to unit quaternions with vanishing $\mathbf{j}$ and $\mathbf{k}$ components as

$$Q_\phi = \cos\phi + \mathbf{i}\sin\phi\,.$$

[4]Adapted-frame: A moving rotation matrix $\mathrm{R} = [\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}]$ associated to path Γ whose first component $\boldsymbol{e_1}$ coincides with the tangent of the path $\frac{\mathrm{d}\boldsymbol{\gamma}(\xi)}{\mathrm{d}\xi}$.

Pure vector quaternions ascribe to quaternions whose real part vanishes, i.e., $a = 0$. In the sequel we will leverage the following definitions and lemmas.

Definition 1 (**Commutative operation**)**:** *The commutative multiplication on the quaternions $\mathcal{A}$ and $\mathcal{B}$ is defined as*

$$\mathcal{A} \star \mathcal{B} \coloneqq \frac{1}{2}\left(\mathcal{A}\,\mathbf{i}\,\mathcal{B}^* + \mathcal{B}\,\mathbf{i}\,\mathcal{A}^*\right) \tag{4}$$

In the case of two equivalent quaternions, the respective n-th power is represented as $\mathcal{A}^{n\star} = \underbrace{\mathcal{A} \star \mathcal{A} \star \cdots \star \mathcal{A}}_{n}$.

Remark 1. $\mathcal{A} \star \mathcal{B}$ is equal to the vector part of $\mathcal{A}\,\mathbf{i}\,\mathcal{B}$, and thus is a pure quaternion.

Later on, we will observe that the solutions to the quaternion equivalents of quadratic and linear equations play a crucial role in the development of PH hermite interpolants. In the subsequent lemmas, we provide a detailed description of these solutions.

Lemma 1: *Given a pure quaternion $\boldsymbol{a}$ and a non-zero quaternion $\mathcal{B}$, the set of all solutions to the linear equation*

$$\mathcal{X} \star \mathcal{B} = \boldsymbol{a} \tag{5a}$$

form the one-parameter family

$$\mathcal{X}_\tau = -\frac{(\tau + \boldsymbol{a})\mathcal{B}\,\mathbf{i}}{|\mathcal{B}|^2} \quad \textit{with } \tau \in \mathbb{R}\,. \tag{5b}$$

Lemma 2: *Given that $\boldsymbol{a}$ is a specific pure quaternion and not a negative multiple of $\mathbf{i}$, then every solution to the quadratic equation*

$$\mathcal{X}^{2\star} = \boldsymbol{a} \tag{6a}$$

form the one-parameter family

$$\mathcal{X}_\phi = \sqrt{|\boldsymbol{a}|}\,\frac{\frac{\boldsymbol{a}}{|\boldsymbol{a}|} + \mathbf{i}}{|\frac{\boldsymbol{a}}{|\boldsymbol{a}|} + \mathbf{i}\,|}\,\mathcal{Q}(\phi) \quad \textit{with } \phi \in [0, 2\pi)\,. \tag{6b}$$

B. Pythagorean Hodograph (PH) curves

PH curves are a subset of polynomial curves whose parametric speed is a polynomial of the path parameter ξ [10]. Decomposing the components of the path-function as $\boldsymbol{p}(\xi) = [x(\xi), y(\xi), z(\xi)]^\mathsf{T}$, the condition for a polynomial to be a PH curve is equivalent to

$$\sigma^2(\xi) = x'^2(\xi) + y'^2(\xi) + z'^2(\xi)\,, \tag{7}$$

where $\sigma(\xi)$ is a polynomial. More intuitively, as its name suggests, the curve's hodograph $\boldsymbol{h}(\xi) = \frac{\mathrm{d}\mathrm{p}(\xi)}{\mathrm{d}\xi} = [x'(\xi), y'(\xi), z'(\xi)]^\mathsf{T}$, i.e., derivative with respect to path parameter ξ, satisfies a Pythagorean condition. As stated in [18], (7) only holds true if $\{u, v, p, q\} : \mathbb{R} \to \mathbb{R}$ polynomials on path parameter ξ exist, such that

$$x'(\xi) = u^2(\xi) + v^2(\xi) - p^2(\xi) - q^2(\xi)\,, \tag{8a}$$
$$y'(\xi) = 2u(\xi)q(\xi) - 2v(\xi)p(\xi)\,, \tag{8b}$$
$$z'(\xi) = 2v(\xi)q(\xi) - 2u(\xi)p(\xi)\,, \tag{8c}$$
$$\sigma(\xi) = u^2(\xi) + v^2(\xi) + p^2(\xi) + q^2(\xi)\,. \tag{8d}$$

Given the four-dimensional structure of (8), combining it with condition (7) and converting it to quaternion algebra allows for a compact definition of the PH condition [18, 19]. This leads to the following description of PH curves.

Definition 2 (PH curves): *A space polynomial curve* $\boldsymbol{p}(\xi) = x(\xi)\,\mathbf{i} + y(\xi)\,\mathbf{j} + z(\xi)\,\mathbf{k}$ *is a PH curve if and only if there exists a quaternion polynomial* $\mathcal{A}(\xi) = u(\xi) + v(\xi)\,\mathbf{i} + p(\xi)\,\mathbf{j} + q(\xi)\,\mathbf{k}$ *such that the hodograph fulfills*

$$\boldsymbol{h}(\xi) = \mathcal{A}(\xi)\,\mathbf{i}\,\mathcal{A}^{*}(\xi) = \mathcal{A}^{2\star}(\xi)\,. \tag{9}$$

From this expression it follows that a quaternion polynomial $\mathcal{A}(\xi)$ of degree n is related to a PH curve whose hodograph $\boldsymbol{h}(\xi)$ and path-function $\boldsymbol{p}(\xi)$ are of degree $2n$ and $2n+1$, respectively. Given their polynomial nature, they can be written in the Bernstein-Bezier form as

$$\mathcal{A}(\xi) = \sum_{i=0}^{n} \mathcal{A}_i B_i^n(\xi)\,, \tag{10a}$$

$$\boldsymbol{h}(\xi) = \sum_{i=0}^{2n} \boldsymbol{h}_i B_i^{2n}(\xi)\,, \tag{10b}$$

$$\boldsymbol{p}(\xi) = \sum_{i=0}^{2n+1} \boldsymbol{p}_i B_i^{2n+1}(\xi) = \boldsymbol{p}_0 + \frac{\sum_{i=0}^{2n} \boldsymbol{h}_i B_i^{2n}(\xi)}{2n+1}\,, \tag{10c}$$

where $\mathcal{A}_i$, $\boldsymbol{h}_i$ and $\boldsymbol{p}_i$ refer to i-th control point of the polynomial functions and $B_j^n(\xi) = \binom{n}{j}\xi_j\,(1-\xi)^{n-j}$ are the Bernstein polynomials. Notice that according to eq. (10c), the control-points of the hodograph $\boldsymbol{h}_{0,\dots,2n}$ relate to the ones of the path-function $\boldsymbol{p}_{1,\dots,2n+1}$. More specifically, for a given starting point $\boldsymbol{p}_0 \in \mathbb{R}^3$,

$$\boldsymbol{p}_i = \boldsymbol{p}_0 + \frac{1}{2n+1}\sum_{j=0}^{i-1} \boldsymbol{h}_j, \quad i = 1, ..., 2n+1\,. \tag{10d}$$

Another notable attribute of PH curves is that they inherit an adapted-frame, denoted as *Euler Rodrigues Frame* (ERF), which also exclusively depends on the quaternion polynomial [9]. Its respective rotation matrix is defined as

$$\mathrm{R}(\xi) = [\boldsymbol{e_1}(\xi), \boldsymbol{e_2}(\xi), \boldsymbol{e_3}(\xi)] = \frac{[\mathcal{A}(\xi)\,\mathbf{i}\,\mathcal{A}^{*}(\xi), \mathcal{A}(\xi)\,\mathbf{j}\,\mathcal{A}^{*}(\xi), \mathcal{A}(\xi)\,\mathbf{k}\,\mathcal{A}^{*}(\xi)]}{|\mathcal{A}(\xi)|^2}\,. \tag{11}$$

and its angular velocity is given by

$$\boldsymbol{\omega}(\xi) = \chi_1(\xi)\boldsymbol{e_1} + \chi_2(\xi)\boldsymbol{e_2} + \chi_3(\xi)\boldsymbol{e_3}\,, \tag{12a}$$

where

$$\chi(\xi) = \{\boldsymbol{e_2'}(\xi)\,\boldsymbol{e_3}(\xi), \boldsymbol{e_3'}(\xi)\,\boldsymbol{e_1}(\xi), \boldsymbol{e_1'}(\xi)\,\boldsymbol{e_2}(\xi)\}\,, \tag{12b}$$

which it likewise is exclusively reliant on the quaternion polynomial. Besides that, given that the parametric speed $\sigma(\xi)$ is a polynomial defined by the condition (7), PH curves also allow for the closed form calculation for other geometrically meaningful properties, such as the arc-length $L(\xi)$, curvature $\kappa(\xi)$ and torsion $\tau(\xi)$:

$$L(\xi) = \int_{\xi_b}^{\xi_e} \sigma(\xi)\,d\xi \tag{13a}$$

$$\kappa(\xi) = \frac{||\boldsymbol{p}'(\xi) \times \boldsymbol{p}''(\xi)||}{||\boldsymbol{p}'(\xi)||^3} \tag{13b}$$

$$\tau(\xi) = \frac{\left[\boldsymbol{p}'(\xi) \times \boldsymbol{p}''(\xi)\right] \cdot \boldsymbol{p}'''(\xi)}{||\boldsymbol{p}'''(\xi)||^2} \tag{13c}$$

where ξ_b and ξ_e are the starting and ending values for the path parameter.

From all the parametric functions introduced in this subsection (10), (11), (12) and (13a), it can be stated that path parameterizing a curve as defined in Section 3 reduces to constructing a PH curve by finding an appropriate quaternion polynomial $\mathcal{A}(\xi)$, which from now onward we will refer to as *preimage*.

5. THE ALGORITHM: PHODCOS

To address problem (1), we present **PHODCOS**: a **P**ythagorean **HO**drograph-based **D**ifferentiable **CO**ordinate **S**ystem. It leverages the appealing properties of PH curves, i.e., their compact representation via the preimage and an inherited adapted-frame (**P2**), to convert a given spatial curve into a piecewise PH curve, while ensuring boundedness on the approximation error (**P1**) and continuity on the parametric functions (**P3**). In this section we provide further details on the underlying methodology and its compliance with these three features.

A. Piecewise PH curves with C^2 parametric functions

Paths parameterized by PHODCOS will only be applicable to second-order optimization methods if all parametric functions are, at least, twice differentiable i.e., $\{\boldsymbol{p}, \mathrm{R}, \boldsymbol{\omega}\} \in C^2(\xi)$ (see **P3**). As mentioned earlier, we approximate the original curve with a piecewise PH curve, and thus, special care needs to be taken across intersections. More specifically, out of all three parametric functions, the angular velocity of the adapted-frame is the one with highest preimage degree, i.e., $\boldsymbol{\omega}(\xi) = f(\mathcal{A}(\xi), \mathcal{A}'(\xi))$. Therefore, C^2 in all parametric functions is equivalent to C^3 in the preimage $\mathcal{A}(\xi)$, leading to a C^4 condition in the path-function $\boldsymbol{p}(\xi)$[2]:

$$\{\sigma,\, \mathrm{R},\, \boldsymbol{\omega}\} \in C^2(\xi) \Rightarrow \mathcal{A} \in C^3(\xi) \Rightarrow \boldsymbol{p} \in C^4(\xi) \tag{14}$$

B. PH curves of degree 17

Defining the degree

To satisfy the continuity condition (14), we split the original curve into multiple segments and approximate each of them by conducting a C^4 hermite interpolation. This is equivalent to specifying position (0), velocity (1), acceleration (2), jerk (3) and snap (4) starting at ending conditions, resulting in 30[3]

[2]Given that $\boldsymbol{\omega}(\xi)$ depends on $\mathcal{A}'(\xi)$, it will only will be C^2 if, at least, $\mathcal{A}'''(\xi)$ is continuous, implying that $\mathcal{A} \in C^3(\xi)$. The hodograph $\boldsymbol{h}(\xi)$, and consequently also the preimage $\mathcal{A}(\xi)$, relates by its derivative to the path-function $\boldsymbol{p}(\xi)$. Therefore, $\{\boldsymbol{h}, \mathcal{A}\} \in C^3(\xi)$ infers $\boldsymbol{p} \in C^4(\xi)$.

[3]Amount of interpolation constraints per interval: $30 = 2$(start and end) $\cdot$ 3(x, y and z) $\cdot$ 5(position to snap)

scalar constraints per interval. Imposing the starting position $\boldsymbol{p}_0$, related to the constant obtained from the integration of the hodograph (10b) into (10c), reduces the amount of free conditions to 27. Given the four-dimensional nature of the preimage $\mathcal{A}(\xi)$, a preimage of order 7 ($4 \cdot 7 = 28 > 27$) accounts for sufficient degrees of freedom. However, in a similar way to [1], increasing the order to 8 guarantees the existence of a solution for any interpolation data. From (10), it can be stated that defining the order of preimage $\mathcal{A}(\xi)$ as $n = 8$ leads to a hodograph $\boldsymbol{h}(\xi)$ of degree $2n = 16$ and a path-function $\boldsymbol{p}(\xi)$ of degree $2n + 1 = 17$.

Hodograph and path-function

Combining the definition of the preimage in (10a) with the quadratic expression in (9) allows for associating the control-points of the hodograph $\boldsymbol{h}_{0,\dots,16}$ to the ones of the preimage $\mathcal{A}_{0,\dots,8}$. After some simplifications[4], these are given by the following expressions:

$$\boldsymbol{h}_0 = \mathcal{A}_0^{2\star}\,, \tag{15a}$$
$$\boldsymbol{h}_1 = \mathcal{A}_0 \star A_1\,, \tag{15b}$$
$$\boldsymbol{h}_2 = \frac{1}{15}(7\mathcal{A}_0 \star A_2 + 8\mathcal{A}_1^{2\star})\,, \tag{15c}$$
$$\boldsymbol{h}_3 = \frac{1}{10}(2\mathcal{A}_0 \star A_3 + 8\mathcal{A}_1 \star A_2)\,, \tag{15d}$$
$$\boldsymbol{h}_4 = \frac{1}{65}(5\mathcal{A}_0 \star A_4 + 32\mathcal{A}_1 \star A_3 + 28\mathcal{A}_2^{2\star})\,, \tag{15e}$$
$$\boldsymbol{h}_5 = \frac{1}{39}(\mathcal{A}_0 \star A_5 + 10\mathcal{A}_1 \star A_4 + 28\mathcal{A}_2 \star A_3)\,, \tag{15f}$$
$$\boldsymbol{h}_6 = \frac{1}{143}(\mathcal{A}_0 \star A_6 + 16\mathcal{A}_1 \star A_5 + 70\mathcal{A}_2 \star A_4 + 56\mathcal{A}_3^{2\star})\,, \tag{15g}$$
$$\boldsymbol{h}_7 = \frac{1}{715}(\mathcal{A}_0 \star A_7 + 28\mathcal{A}_1 \star A_6 + 196\mathcal{A}_2 \star A_5 + 490\mathcal{A}_3 \star A_4)\,, \tag{15h}$$
$$\boldsymbol{h}_8 = \frac{1}{6435}(\mathcal{A}_0 \star A_8 + 64\mathcal{A}_1 \star A_7 + 784\mathcal{A}_2 \star A_6 + 3136\mathcal{A}_3 \star A_5 + 2450\mathcal{A}_4^{2\star})\,, \tag{15i}$$
$$\boldsymbol{h}_9 = \frac{1}{715}(\mathcal{A}_1 \star A_8 + 28\mathcal{A}_2 \star A_7 + 196\mathcal{A}_3 \star A_6 + 490\mathcal{A}_4 \star A_5)\,, \tag{15j}$$
$$\boldsymbol{h}_{10} = \frac{1}{143}(\mathcal{A}_2 \star A_8 + 16\mathcal{A}_3 \star A_7 + 70\mathcal{A}_4 \star A_6 + 56\mathcal{A}_5^{2\star})\,, \tag{15k}$$
$$\boldsymbol{h}_{11} = \frac{1}{39}(\mathcal{A}_3 \star A_8 + 10\mathcal{A}_4 \star A_7 + 28\mathcal{A}_5 \star A_6)\,, \tag{15l}$$
$$\boldsymbol{h}_{12} = \frac{1}{65}(5\mathcal{A}_4 \star A_8 + 32\mathcal{A}_5 \star A_7 + 28\mathcal{A}_6^{2\star})\,, \tag{15m}$$
$$\boldsymbol{h}_{13} = \frac{1}{10}(2\mathcal{A}_5 \star A_8 + 8\mathcal{A}_6 \star A_7)\,, \tag{15n}$$
$$\boldsymbol{h}_{14} = \frac{1}{15}(7\mathcal{A}_6 \star A_8 + 8\mathcal{A}_7^{2\star})\,, \tag{15o}$$
$$\boldsymbol{h}_{15} = \mathcal{A}_7 \star A_8\,, \tag{15p}$$
$$\boldsymbol{h}_{16} = \mathcal{A}_8^{2\star}\,. \tag{15q}$$

[4]Intermediary steps for computing the hodograph's control-points can be found in Appendix C.

Putting together these control-points with eq. (10d) and a predetermined starting point $\boldsymbol{p}_0$, the path-function's control-points $\boldsymbol{p}_{1,\dots,17}$ – and as a result, the path-function itself – are fully defined.

C. Construction of C^4 hermite interpolants

In this subsection we provide a thorough description on how each interval of the original curve is approximated by a piecewise PH curve.

Boundary conditions

At every segment, given C^4 hermite boundary data – end points $\{\boldsymbol{p}_b, \boldsymbol{p}_e\}$[5], and the first four derivative vectors $\{\boldsymbol{v}_b, \boldsymbol{v}_e, \boldsymbol{a}_b, \boldsymbol{a}_e, \boldsymbol{j}_b, \boldsymbol{j}_e, \boldsymbol{s}_b, \boldsymbol{s}_e\}$ –, we construct a spatial PH curve $\boldsymbol{p}(\xi)$. Considering the Bernstein-Bezier form of the hodograph $\boldsymbol{h}(\xi)$ in (10b) and its relationship with respect to the interpolant data, i.e., $\boldsymbol{v}(\xi) = \boldsymbol{h}(\xi)$, $\boldsymbol{a}(\xi) = \boldsymbol{h}'(\xi)$, $\boldsymbol{j}(\xi) = \boldsymbol{h}''(\xi)$, $\boldsymbol{s}(\xi) = \boldsymbol{h}'''(\xi)$, it results that

$$\boldsymbol{v}_b = \boldsymbol{h}_0\,, \tag{16a}$$
$$\boldsymbol{v}_e = \boldsymbol{h}_{16}\,, \tag{16b}$$
$$\boldsymbol{a}_b = -16\boldsymbol{h}_0 + 16\boldsymbol{h}_1\,, \tag{16c}$$
$$\boldsymbol{a}_e = -16\boldsymbol{h}_{15} + 16\boldsymbol{h}_{16}\,, \tag{16d}$$
$$\boldsymbol{j}_b = 240\boldsymbol{h}_0 - 480\boldsymbol{h}_1 + 240\boldsymbol{h}_2\,, \tag{16e}$$
$$\boldsymbol{j}_e = 240\boldsymbol{h}_{14} - 480\boldsymbol{h}_{15} + 240\boldsymbol{h}_{16}\,, \tag{16f}$$
$$\boldsymbol{s}_b = -3360\boldsymbol{h}_0 + 10080\boldsymbol{h}_1 - 10080\boldsymbol{h}_2 + 3360\boldsymbol{h}_3\,, \tag{16g}$$
$$\boldsymbol{s}_e = -3360\boldsymbol{h}_{13} + 10080\boldsymbol{h}_{14} - 10080\boldsymbol{h}_{15} + 3360\boldsymbol{h}_{16}\,. \tag{16h}$$

Moreover, introducing the interpolation endpoints on (10d) leads to

$$\boldsymbol{p}_e - \boldsymbol{p}_b = \frac{1}{17}\sum_{i=0}^{16}\boldsymbol{h}_i\,.$$

and combining it with (15), after some simplifications, results in

$$\mathcal{A}_{\boldsymbol{p}}^{2\star} = \frac{490}{21879}\,(\boldsymbol{p}_e - \boldsymbol{p}_b) - \boldsymbol{c}_{\boldsymbol{p}} \tag{17a}$$

with

$$\mathcal{A}_{\boldsymbol{p}} = \frac{1}{442}\mathcal{A}_0 + \frac{5}{663}\mathcal{A}_1 + \frac{35}{2431}\mathcal{A}_2 + \frac{49}{2431}\mathcal{A}_3 + \frac{490}{21879}\mathcal{A}_4 + \frac{49}{2431}\mathcal{A}_5 + \frac{35}{2431}\mathcal{A}_6 + \frac{5}{663}\mathcal{A}_7 + \frac{1}{442}\mathcal{A}_8 \tag{17b}$$

[5]For a given parametric function $f(\xi)$, $f_b = f(0)$ and $f_e = f(1)$.

and

$$\begin{aligned}\boldsymbol{c_p} = \frac{1}{112633092}(&147807\mathcal{A}_0^{2\star} + 144540\mathcal{A}_0 \star A_1 + \\ &61908\mathcal{A}_0 \star A_2 + 19404\mathcal{A}_0 \star A_3 - 6468\mathcal{A}_0 \star A_5 - \\ &6300\mathcal{A}_0 \star A_6 - 3636\mathcal{A}_0 \star A_7 - 1130\mathcal{A}_0 \star A_8 + \\ &+ 72732\mathcal{A}_1^{2\star} + 94248\mathcal{A}_1 \star A_2 + 38808\mathcal{A}_1 \star A_3 - \\ &17640\mathcal{A}_1 \star A_5 - 18648\mathcal{A}_1 \star A_6 - 11336\mathcal{A}_1 \star A_7 - \\ &3636\mathcal{A}_1 \star A_8 + 40572\mathcal{A}_2^{2\star} + 41160\mathcal{A}_2 \star A_3 - \\ &24696\mathcal{A}_2 \star A_5 - 28616\mathcal{A}_2 \star A_6 - 18648\mathcal{A}_2 \star A_7 - \\ &6300\mathcal{A}_2 \star A_8 + 12348\mathcal{A}_3^{2\star} - 19208\mathcal{A}_3 \star A_5 - \\ &24696\mathcal{A}_3 \star A_6 - 17640\mathcal{A}_3 \star A_7 - 6468\mathcal{A}_3 \star A_8 + \\ &12348\mathcal{A}_5^{2\star} + 41160\mathcal{A}_5 \star A_6 + d38808\mathcal{A}_5 \star A_7 + \\ &19404\mathcal{A}_5 \star A_8 + 40572\mathcal{A}_6^{2\star} + 94248\mathcal{A}_6 \star A_7 + \\ &61908\mathcal{A}_6 \star A_8 + 72732\mathcal{A}_7^{2\star} + 144540\mathcal{A}_7 \star A_8 + \\ &147807\mathcal{A}_8^{2\star}) \,. \end{aligned} \tag{17c}$$

Standard form

As discussed later, to ensure that the interpolation method is fully invariant, the interpolation data needs to be transformed into a standard form.

Definition 3 (**Standard form):** *The C^4 spatial hermite data are considered to be in standard form if $\boldsymbol{v}_b + \boldsymbol{v}_e$ can be expressed as a positive multiple of $\mathbf{i}$ and $\boldsymbol{p}_b = \mathbf{0}$.*

The interpolation algorithm

Having defined the preliminaries, boundary conditions and data's standard form, the steps to compute the preimage associated with a PH curve that interpolates C^4 hermite data are described in Algorithm 1.

Remark 2. To be more specific, in line 8 the expression to compute $\mathcal{A}_4$ is

$$\begin{aligned}\mathcal{A}_4 = \frac{21879}{490}\Bigg(&\sqrt{|\boldsymbol{c_p}|}\frac{\frac{\boldsymbol{c_p}}{|\boldsymbol{c_p}|}+\boldsymbol{i}}{|\frac{\boldsymbol{c_p}}{|\boldsymbol{c_p}|}+\boldsymbol{i}|}\mathcal{Q}(\theta_4) - (\frac{1}{442}\mathcal{A}_0 + \\ &\frac{5}{663}\mathcal{A}_1 + \frac{35}{2431}\mathcal{A}_2 + \frac{49}{2431}\mathcal{A}_3 + \frac{49}{2431}\mathcal{A}_5 + \\ &\frac{35}{2431}\mathcal{A}_6 + \frac{5}{663}\mathcal{A}_7 + \frac{1}{442}\mathcal{A}_8)\Bigg) \,. \end{aligned} \tag{18}$$

As a consequence, the preimages $\mathcal{A}_\phi(\xi)$, hodographs $\boldsymbol{h}_\phi(\xi)$ and path-functions $\boldsymbol{p}_\phi(\xi)$ of a PH curve that interpolate the given C^4 hermite data relate to a 9 parameter family $\boldsymbol{\phi} = [\theta_0, \tau_1, \tau_2, \tau_3, \theta_4, \tau_5, \tau_6, \tau_7, \theta_8]$. Similar to C1 and C2 hermite interpolation [1, 20], one of the angular parameters, e.g. θ_4, can be set to zero owing to the non-trivial fibers of the map from the preimage to the hodograph (9).As a result, all PH interpolants can be derived using parameter vectors where $\theta_4 = 0$. Considering this, for the remainder of the paper, we will adopt this specific option and remove θ_4 from the parameter vector ϕ, leading to $\phi = [\theta_0, \tau_1, \tau_2, \tau_3, \tau_5, \tau_6, \tau_7, \theta_8]$.

D. Choosing the optimal interpolants

In this section we conduct a qualitative analysis of the system of PH curve interpolants $\boldsymbol{\phi}$ in order to identify the "best" values of the parameter vector $\boldsymbol{\phi}^*$. To fix the free parameters $\boldsymbol{\phi}$, we examine the asymptotic behavior of the solutions $\boldsymbol{p}_\phi(\xi)$. Specifically, we consider the C^4 hermite data extracted from a short portion of an analytic curve and study the asymptotic properties of the solutions as the step size diminishes.

Algorithm 1 *C^4 hermite data interpolation*:
Given interpolation data $\{\boldsymbol{p}_b, \boldsymbol{p}_e, \boldsymbol{v}_b, \boldsymbol{v}_e, \boldsymbol{a}_b, \boldsymbol{a}_e, \boldsymbol{j}_b, \boldsymbol{j}_e, \boldsymbol{s}_b, \boldsymbol{s}_e\}$, compute preimage control-points $\{\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6, \mathcal{A}_7, \mathcal{A}_8\}$.

1: **function** C4INTERPOLATION($\boldsymbol{p}_b, \boldsymbol{p}_e, \boldsymbol{v}_b, \boldsymbol{v}_e, \boldsymbol{a}_b, \boldsymbol{a}_e, \boldsymbol{j}_b, \boldsymbol{j}_e, \boldsymbol{s}_b, \boldsymbol{s}_e$)
2: $\{\boldsymbol{p}_b, \cdots, \boldsymbol{s}_e\}_{sf} \leftarrow$Transform the data to the standard form in Definition 3 by a rotation and translation.
3: $\boldsymbol{h}_0, \boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, \boldsymbol{h}_{13}, \boldsymbol{h}_{14}, \boldsymbol{h}_{15}, \boldsymbol{h}_{16} \leftarrow$ Compute hodograph control-points from (16)
4: $\mathcal{A}_0, \mathcal{A}_8 \leftarrow$ Calculate $\mathcal{A}_0$ and $\mathcal{A}_8$ from eqs. (15a) and (15q), which follow Lemma 2, and thus, introduce free parameters θ_0, θ_8.
5: $\mathcal{A}_1, \mathcal{A}_7 \leftarrow$ Calculate $\mathcal{A}_1$ and $\mathcal{A}_7$ from eqs. (15b) and (15p), which follow Lemma 1, and thus, introduce free parameters τ_1, τ_7.
6: $\mathcal{A}_2, \mathcal{A}_6 \leftarrow$ Calculate $\mathcal{A}_2$ and $\mathcal{A}_6$ from eqs. (15c) and (15o), which follow Lemma 1, and thus, introduce free parameters τ_2, τ_6.
7: $\mathcal{A}_3, \mathcal{A}_5 \leftarrow$ Calculate $\mathcal{A}_3$ and $\mathcal{A}_5$ from eqs. (15d) and (15n), which follow Lemma 1, and thus, introduce free parameters τ_3, τ_5.
8: $\mathcal{A}_4 \leftarrow$Calculate $\mathcal{A}_4$ from eq. (17), which follow Lemma 2, and thus, introduces a free parameter θ_4.
9: $\boldsymbol{h}_{4,\dots,12} \leftarrow$ From eqs. (15e)- (15m), compute $\boldsymbol{h}_{4,\dots,12}$ and, by setting $\boldsymbol{p_b} = \boldsymbol{p_0}$ calculate the remaining control-points of $\boldsymbol{p}(\xi)$.
10: $\mathcal{A}_{0,\cdots,8} \leftarrow$ Return the solution to its original pose, by performing the inverse operation of step 1.
11: **return** $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6, \mathcal{A}_7, \mathcal{A}_8$
12: **end function**

Considering that the curve is represented by its Taylor series, without any loss of generality,

$$\boldsymbol{\gamma}(\Xi) = \left(\Xi + \sum_{i=2}^{\infty}\frac{x_i}{i!}\Xi^i, \sum_{i=2}^{\infty}\frac{y_i}{i!}\Xi^i, \sum_{i=2}^{\infty}\frac{z_i}{i!}\Xi^i\right)^{\mathsf{T}} , \tag{19}$$

where x_i, y_i, z_i with $i = 2, \dots, \infty$ refer to arbitrary coefficients. For a given step size of h, we pick the segment k, $\boldsymbol{\gamma}(\Xi) = \boldsymbol{\gamma}_k(h\xi)$, $\xi \in [0, 1]$, whose expansion is

$$\boldsymbol{\gamma}_k(\xi) = \left(\xi h + \sum_{i=2}^{\infty}\frac{x_i}{i!}\xi^i h^i, \sum_{i=2}^{\infty}\frac{y_i}{i!}\xi^i h^i, \sum_{i=2}^{\infty}\frac{z_i}{i!}\xi^i h^i\right)^{\mathsf{T}} . \tag{20}$$

Now we interpolate the C^4 hermite boundary data at the points $\boldsymbol{\gamma}_k(0) = \boldsymbol{\gamma}(0)$ and $\boldsymbol{\gamma}_k(1) = \boldsymbol{\gamma}(h)$. The behavior of different PH curves that interpolate the data, based on the size of the interval h, is outlined in the theorem below.

Theorem 1 (**Approximation error):** *The interpolation error of the PH*

$$E = \max_{\xi\in[0,1]} ||\boldsymbol{\gamma}_k(\xi) - \boldsymbol{p}_\phi(\xi)|| \tag{21}$$

converges to 0 for $h \to 0$ if and only if $\tau_1 = \tau_2 = \tau_3 = \tau_5 = \tau_6 = \tau_7 = 0$. Moreover, if $\theta_0 = \theta_4 = \theta_8 = 0$, then $E = \mathcal{O}\left(h^6\right)$. Otherwise, $E = \mathcal{O}\left(h^1\right)$.

Proof. The proof entails the assessment of the power series for all quantities involved in the interpolation procedure in relation to the step-size h, which can be achieved using a suitable computer algebra tool. Due to constraints in available space and the intricate nature of the expressions, we solely present the principal terms of specific quantities, aiming to demonstrate the underlying concept of our approach. For a detailed analysis of the proof, please refer to the Mathematica notebook attached alongside the released code [6]. □

Beyond ensuring the highest approximation order, choosing $\boldsymbol{\phi}^* = \mathbf{0}$ offers other noteworthy advantages, such as maintaining planarity, preserving interpolants, and being reversion invariant, as elaborated in Theorems 2, 3, and 4. These theorems and their corresponding proofs, which can be directly traced back to the C^2 predecessor [1], are formally presented in Appendix B. When combined with Theorem 1, it becomes clear that the interpolant $\boldsymbol{\phi}^* = \mathbf{0}$ is exceptionally effective for curve approximation. All these attributes collectively are encapsulated in the following statement.

Corollary 1 (Optimal interpolant $\boldsymbol{\phi}^* = \mathbf{0}$): *The C^4 hermite PH interpolant $\boldsymbol{p}_{\phi^*}(\xi)$ has an approximation order of 6 and, in addition, retains the planarity of the given data, remains invariant under any orthogonal transformations, scaling, and reversion of the input.*

E. Continuity of the preimage and the adapted-frame

In the previous subsection, we have shown the benefits of choosing $\boldsymbol{\phi}^* = \mathbf{0}$ for the C^4 interpolation algorithm in 1. Before doing so, as explained at the end of subsection C., due to non-trivial fibers between the preimage and the hodograph, we have fixed the parameter $\theta_4 = 0$. Despite being a numerically convenient choice and not influencing the algorithm's approximation error, it implies that the preimage $\mathcal{A}(\xi)$, as well as the adapted-frame $\mathrm{R}(\xi)$, of two successive segments differ by a *roll-angle offset*, and thus,

$$\mathcal{A}_k(1) \neq \mathcal{A}_{k+1}(0) \quad \text{and} \quad \mathrm{R}_k(1) \neq \mathrm{R}_{k+1}(0)\,,$$

where k and $k+1$ refer to the successive segments. To recover the continuity, we perform a *roll-rotation* α that removes the existing offset. This can be achieved by a two stage procedure: First, we calculate a quaternion that cancels the offset,

$$\mathcal{A}_\alpha = \delta_\alpha(\mathrm{R}_k, \mathrm{R}_{k+1})\,, \tag{22a}$$

where $\delta_\alpha : \{\mathrm{SO}(3), \mathrm{SO}(3)\} \to \mathrm{SO}(3)$ computes the α roll difference between two rotation matrices and outputs the respective quaternion. Second, we rotate all the preimage control-points accordingly

$$\mathcal{A}_{0,\cdots,8} = \mathcal{A}_\alpha\, \mathcal{A}_{0,\cdots,8}\,. \tag{22b}$$

Reconstructing the PH curve with the rotated preimage control-points in (22b) results in a parametric path that, besides having the appealing mathematical attributes summarized in Corollary 1, it is also offset-free, and thereby continuous, in its preimage $\mathcal{A}(\xi)$ and adapted-frame $\mathrm{R}(\xi)$.

F. Overview of the method

The pseudocode for PHODCOS – the presented path parameterization scheme – is summarized in Algorithm 2. Provided an analytical curve $\boldsymbol{\gamma}(\xi)$ and a permissible error tolerance ϵ, PHODCOS converts it into a parametric path, whose

[6]Mathematica notebook with the proof of theorem 1: `https://github.com/jonarriza96/phodcos/tree/main/theory`

Algorithm 2 *Pythagorean Hodograph-based Differentiable Coordinate System* (**PHODCOS**):
Given an analytical curve $\boldsymbol{\gamma}(\xi)$ and an admissible error tolerance ϵ, convert it into a parametric-path whose Cartesian coordinates $\boldsymbol{p}(\xi)$, adapted-frame $\mathrm{R}(\xi)$, angular velocity $\boldsymbol{\omega}(\xi)$, are given by – at least – C^2 functions.

function PHODCOS($\boldsymbol{\gamma}(\xi), \epsilon$)
  $e = \infty\,,\ n_s = 2$
  while $e \geq \epsilon$ **do**
    $\boldsymbol{\gamma}_0, ..., \boldsymbol{\gamma}_{n_s} \leftarrow$ DIVIDEINTOSEGMENTS($\boldsymbol{\gamma}, n_s$)
    for $k \in \{1,\ ..., n_s\}$ **do**
      $\boldsymbol{p}_b, \cdots, \boldsymbol{s}_e \leftarrow$ GETINTERPOLATIONDATA($\boldsymbol{\gamma}_k$)
      $\mathcal{A}_0^k, \cdots, \mathcal{A}_8^k \leftarrow$ C4HERMITE($\boldsymbol{p}_b, \cdots, \boldsymbol{s}_e$) 1
      $\mathcal{A}_0^k, \cdots, \mathcal{A}_8^k \leftarrow$ ERFCONTI($\mathcal{A}_0^k, \cdots, \mathcal{A}_8^k$) (22)
    end for
    $\boldsymbol{p}, \mathrm{R}, \boldsymbol{\omega}, \cdots \leftarrow$ CONSTRUCTCURVE($\mathcal{A}$)(10)(11)(12)
    $e \leftarrow$ CALCULATECONVERSIONERROR($\boldsymbol{\gamma}, \boldsymbol{p}$)
    $n_s \leftarrow n_s + 1$
  end while
  return $\mathcal{A}_{0,\cdots,8}$, $\boldsymbol{p}(\xi), \mathrm{R}(\xi), \boldsymbol{\omega}(\xi), L(\xi), \kappa(\xi), \tau(\xi), \cdots$
end function

Cartesian coordinates $\boldsymbol{p}(\xi)$, adapted-frame $\mathrm{R}(\xi)$, angular velocity $\boldsymbol{\omega}(\xi)$ are described by – at least – C^2 functions. Additionally, parametric functions expressing other geometric characteristics such as arc length $L(\xi)$, curvature $\kappa(\xi)$, and torsion $\tau(\xi)$ can also be computed in closed-form. Due to the approximation error of order h^6, proven in theorem 1, the algorithm iteratively runs the C^4 hermite interpolation method in 1 by augmenting the amount of segments n_s until the approximation error e is below the desired tolerance ϵ. Notice that the presented iterative scheme to find the minimum amount of segments n_s that fulfills the tolerance ϵ is exemplary and could be further tailored to the user needs, either by starting with a better initial guess – higher n_s – or by modifying the update strategy at every iteration.

G. Numerical analysis

The method described in the preceding subsection enables the transformation of any analytical curve into a PH curve. To showcase the properties of this algorithm, we perform a numerical analysis consisting of three parts. Firstly, we concentrate on an exemplary curve to replicate the approximation error as proven in theorem 1. Secondly, we examine the continuity of the resulting parametric functions. Lastly, we investigate the planarity, reversion, and invariance properties outlined in Theorems 2, 3, and 4.

Approximation error

To numerically replicate the approximation order obtained in theorem 1, as an illustrative example we select the same curve as in [1, 20]:

$$\boldsymbol{\lambda}(\xi) = \left[1.5\sin(7.2\xi), \cos(9\xi), e^{\cos(1.8\xi)}\right]\,, \tag{23}$$

for $\xi \in [0, 1]$. We split this interval into 2^n uniform segments, i.e., $h = \frac{1}{2^n}$. By means of the C^4 interpolation algorithm 1, each of them is converted into a PH curve. When doing so, aiming for the same approximation order as in theorem 1, we select $\boldsymbol{\phi}^* = \mathbf{0}$. If the error between the original analytical curve and the converted PH curve is not sufficiently

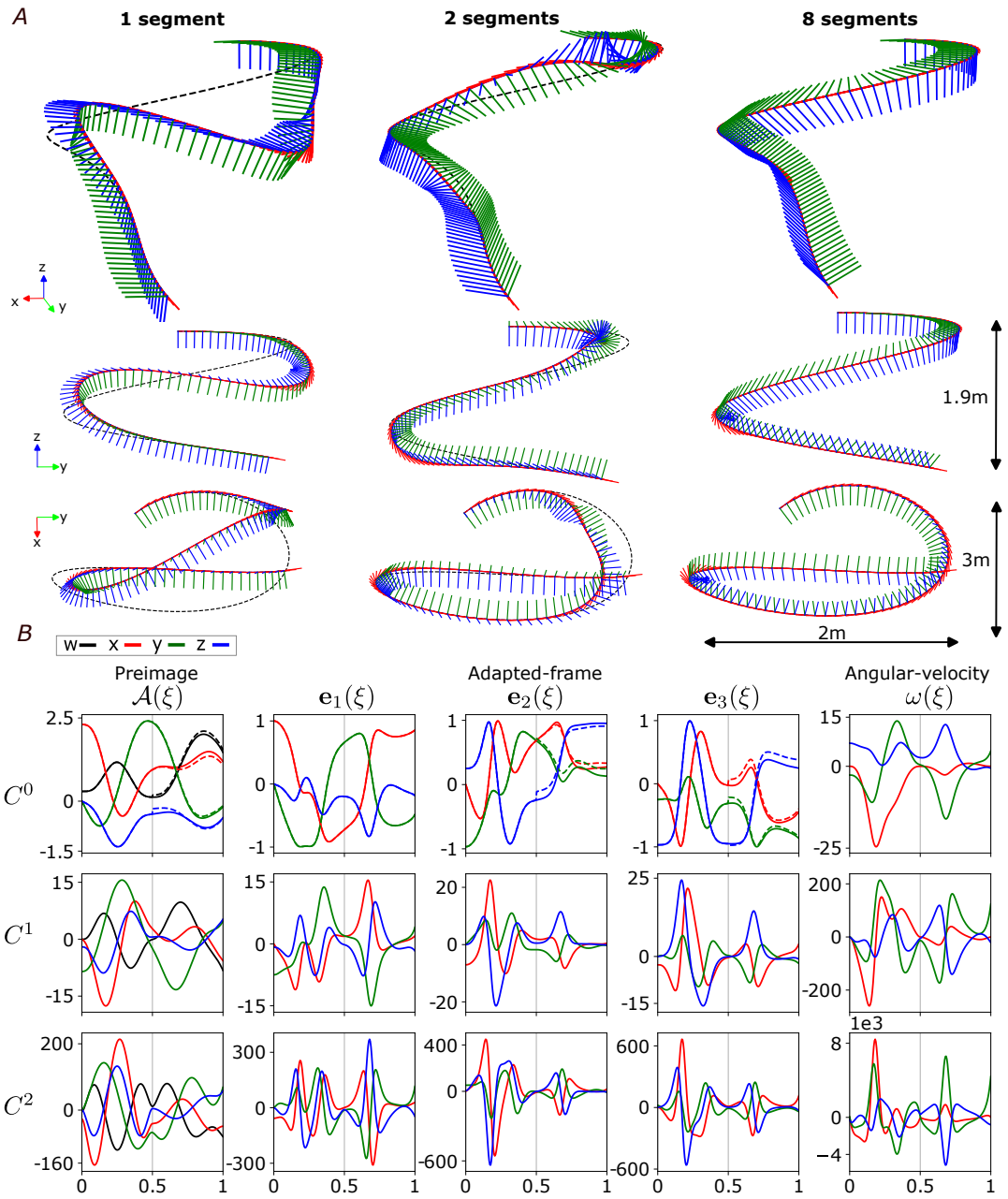


Figure 2: *A)* Comparison of the parametric paths resulting from the conversion of an exemplary analytical curve (23) by means of PHODCOS – the algorithm 2 presented in this work – for 1, 2 and 8 segments. The original analytical curve is given by the dashed black line, while the converted parametric path is depicted by its adapted frame, with the first, second and third components shown in red, green and blue, respectively. The computed parametric paths are divided into columns, each of which displays the results associated with a specific number of segments. In each column, the top row provides an isometric view, while the middle and bottom rows offer side and top views, respectively. The conversion error and improvement ratios associated to every number of segments are given in Table 1. As stated in theorem 1, the conversion error decreases as the number of segments augments, resulting in a greater similarity between the parametric path and the original curve. *B)*. Continuity analysis in the preimage $\mathcal{A}(\xi)$ (first column), adapted frame $\mathrm{R}(\xi)$ (second, third and fourth columns) and angular-velocity $\boldsymbol{\omega}(\xi)$ (fifth column) for the *2 segments* case. The transition between the first and second segments is given by the gray vertical line at $\xi = 0.5$. The first row depicts the functions themselves, while the second and third relate to the first and second derivatives. The dashed lines intend to expose the loss on continuity in the parametric functions if the angle offset cancellation (22) is not conducted.

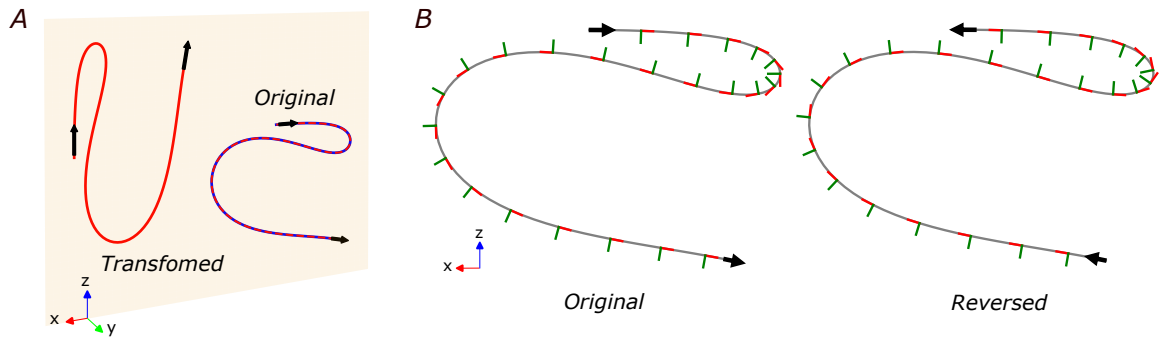


Figure 3: Numerical validation for the planarity, invariance and reversion theorems 2, 3 and 4. The original curve is a planar variant of (23), whose y component is set to 0. In both figures the interpolation data is represented by black arrows. *A)* Within the xz plane colored in yellow, the *original curve* is given in blue, the *transformed curve* in red and the curve resulting from rotating and translating back the transformed curve in dashed-red. This curve aligns with the original one, and thereby, numerically validates theorem 3. Additionally, in accordance with the planarity theorem 2, all curves remain in the same plane as the interpolation data. *B)* Planar view of the original and reversed curves, alongside their adapted frames, where the first and second components are colored in red and green, respectively. As stated by the reversion theorem 4, the curves obtained for the optimal interpolants $\phi^* = 0$ are equivalent and can only be distinguished by the direction of their adapted-frames.

small, we continue with further subdivision by increasing n. This implies that the error is expected to converge to 0 as $\mathcal{O}(h^6) = \mathcal{O}(\frac{1}{64^n})$. Please note that achieving the desired convergence in the improvement ratio necessitates performing computations with high precision. In order to accomplish this, we utilize the advanced precision capabilities offered by Mathematica and an accompanying notebook containing the relevant calculations is included along with the released code[6].

The maximum distance between the original curve and the converted path – denoted as *conversion error* – for different divisions, alongside the respective improvement ratios are given in Table 1. As shown in the third and sixth columns, the improvement ratio converges to 64. Besides that, in Fig. 2-A, a comparison of the parametric paths resulting from the conversion with 1, 2 and 8 segments, intuitively depicts how a higher number of segments correlates to a better approximation error.

Table 1: Maximum distance error and improvement ratio between the original curve and the parametric path obtained from PHODCOS. The error decreases by a ratio of 64, as stated in theorem 1. Paths for 1, 2, and 8 segments are shown in Fig. 2.

# Segments	Error [m]	Ratio
1	1.2569	-
2	0.5447	2.307
4	0.0332	16.405
8	16.080e−4	20.646
16	24.455e−6	63.173
32	1.897e−7	134.145
64	5.009e−9	37.878
128	8.009e−11	62.545
256	1.272e−12	62.986
512	1.982e−14	64.161
1024	3.105e−16	63.822
2048	4.856e−18	63.955
4096	7.588e−20	63.989
8192	1.186e−23	63.997

Continuity analysis

In order to showcase that the parametric functions resulting from PHODCOS achieve the desired continuity of C^2, we analyze their continuity up to the second derivative. For this purpose we focus on the case with two segments, namely the middle column in Fig. 2-A. Despite its high approximation error, this case allows us to clearly observe the transition between the first and second segments.

The preimage $\mathcal{A}(\xi)$ and the parametric functions for the adapted frame $\mathrm{R}(\xi)$ and its angular velocity $\boldsymbol{\omega}(\xi)$, alongside their first two derivatives, are given in Fig. 2-B. Notice that we do not analyze $\boldsymbol{p}(\xi)$ since this is apriori known to be C^4 from the underlying interpolation algorithm 1. The transition between the first and second segment occurs at $\xi = 0.5$ and is depicted by a grey line. As these plots suggest, all parametric functions are at least C^2. More specifically, following the theoretical explanations in A., the angular velocity $\boldsymbol{\omega}(\xi)$ is the parametric function with the lowest degree of continuity of C^2. This can be observed from the plot at the bottom right side of Fig. 2-B, where further derivation would result in discontinuities at the transition $\xi = 0.5$. This is not the case for the preimage $\mathcal{A}(\xi)$ and the adapted frame $\mathrm{R}(\xi)$, which are C^3, and therefore, still account for an additional continuous derivative.

Besides that, we also study the consequences of omitting the rotation – explained in subsection E. – that ensures continuity in the preimage $\mathcal{A}(\xi)$ and the adapted frame $\mathrm{R}(\xi)$ by removing the roll-offset between two successive segments. The resultant parametric functions are depicted by dashed lines in the first row of Fig. 2-B and, for clarity, we do not

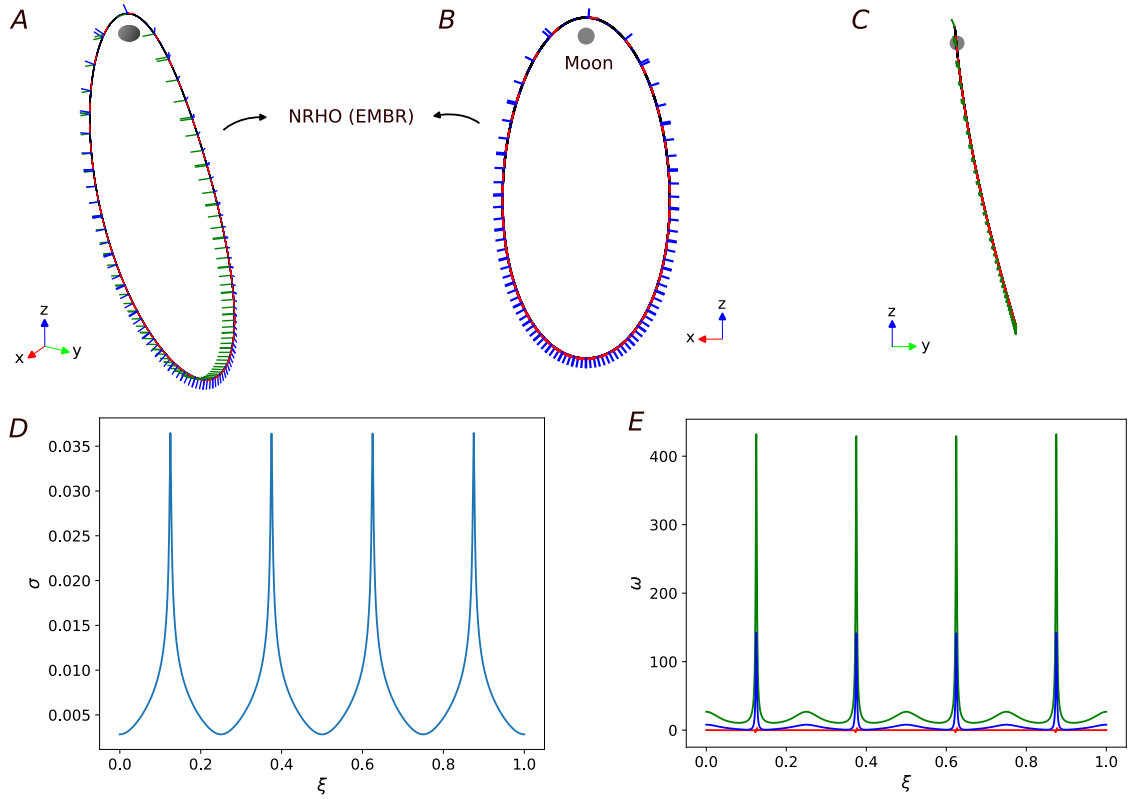


Figure 4: The PHODCOS coordinate system is applied to the Near Rectilinear Halo Orbit (NRHO) within the Earth-Moon Barycentric Rotating (EMBR) reference frame. This system is illustrated from three perspectives: A) isometric view, B) front view, and C) side view. The vector components corresponding to the coordinate system are represented in red, blue, and green, respectively. The parametric speed σ and the angular velocity $\boldsymbol{\omega}$ associated with the coordinate system are depicted in panels D) and E). For an alternative view of the coordinate system from the Earth-Centered Inertial (ECI) frame, please refer to Fig. 1.

show the higher derivatives. From the obtained results, it becomes apparent that without the angle-offset cancellation, despite having a C^4 parametric path function $\boldsymbol{p}(\xi)$, continuity in the preimage $\mathcal{A}(\xi)$ and the adapted frame $\mathrm{R}(\xi)$ is lost, thereby rendering the offset cancellation in subsection E. indispensable.

Planarity, reversion and invariance analysis

In this subsection we numerically validate the remaining theorems 2, 3, and 4. For this purpose, we reuse the exemplary curve (23) from the previous subsections. However, to validate the planarity preservation in theorem 2, we set the y component to 0. Using the planar projection, we separately study the invariance to special orthogonal motions in theorem 3 and reversion in theorem 4. In both cases, we consider the original curve (23) for a single segment.

The first study relates to Fig. 3-A, where we manipulate the hermite data of the *original curve* (depicted in blue) by employing an arbitrary special orthogonal motion, specifically a translation of $[-1, 0, 2.5]$ and a rotation of $\pi/2$ radians around the y axis. The interpolation of this data yields a *transformed curve* (given in red). According to theorem 3, it is affirmed that by translating and rotating the transformed curve back, an equivalent curve to the original one is obtained. This concept is visually depicted in Figure 3-A, where the curve resulting from translating and rotating back the transformed curve (shown in red dashed line) aligns with the original curve. Another notable outcome from this study is that in line with the planarity theorem 2, both the original and the transformed curves remain confined within the same xz-plane (colored yellow) as the interpolation data (represented by the black arrows).

The second study focuses on the reversion invariance property stated in theorem 4. Using the same hermite data as before, we show how the optimal interpolants $\phi^* = 0$ provide the exact same curve for the reversed data. This can be visualized in Fig. 3-B, where the original and reversed curves only differ by the direction of the adapted-frame.

6. Tutorial Example: Lunar Gateway

To illustrate the applicability of PHODCOS, we focus on the Near Rectilinear Halo Orbit (NRHO), a highly nonlinear geometric reference selected for the placement of the Lunar Gateway, a space station for the Artemis program aimed at

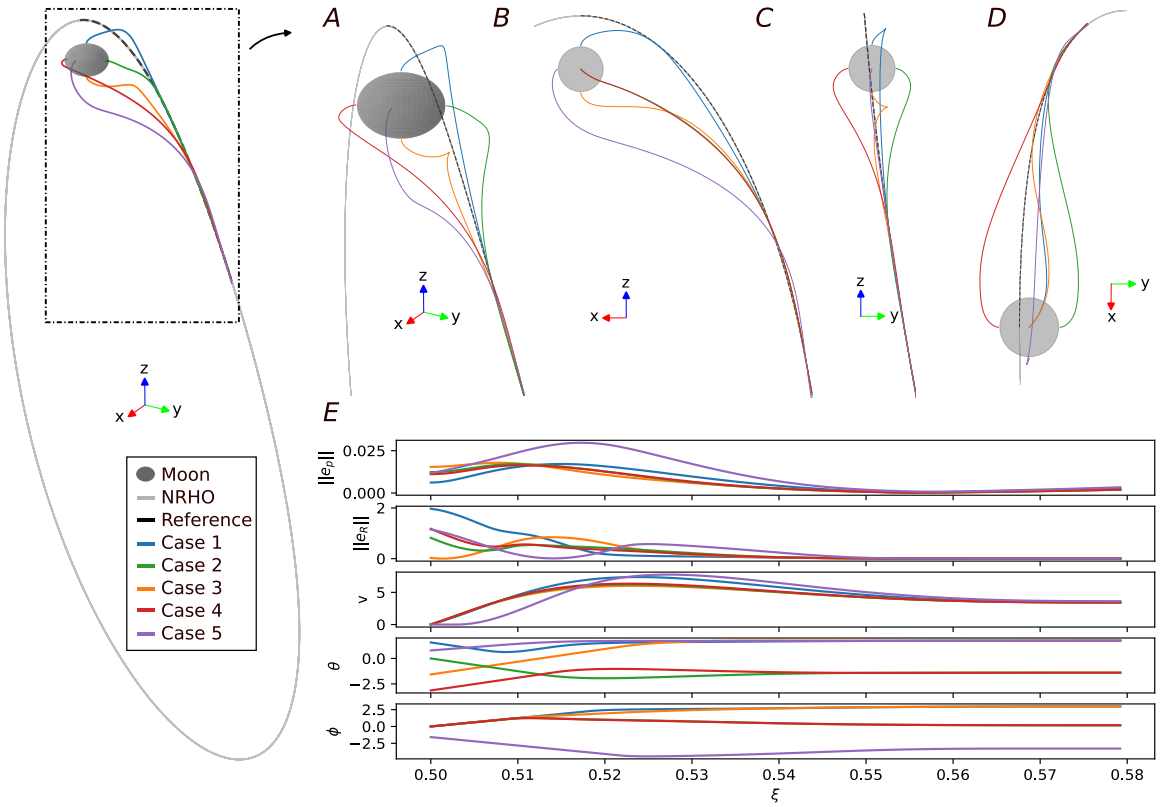


Figure 5: Ascent maneuvers that take the lunar lander from five different positions at the Moon's surface back to the NRHO. The respective motions have been computed by a prediction-based controller whose dynamics evolve according to the PHODCOS coordinate system. The Moon and NRHO are given in gray, while the five lunar lander trajectories are colored blue, green, orange, red and purple. The left column provides a broad overview of the converging maneuvers towards the orbit, while panels A), B), C), and D) present more detailed isometric, front, side, and top views respectively. Panel E) illustrates the position and orientation errors between the lunar lander and the NRHO (rows 1 and 2), its velocity magnitude (row 3) and gimbal angles (rows 4 and 5), color coded according to the different initial conditions.

establishing a permanent human presence on the Moon [21]. Unlike existing relative spacecraft maneuvering models that require the orbit to be circular or elliptical, PHODCOS does not impose constraints on shape or size, making it applicable to any arbitrary curve [22, 23].

We demonstrate the efficacy of PHODCOS by dividing the analysis into two phases. First, we use Algorithm 2 to parameterize the NRHO in both the Earth-Centered Inertial (ECI) and Earth-Moon Barycentric Rotating (EMBR) reference frames. Following that, we showcase how PHODCOS allows for the formulation of trajectory optimization algorithms, by developing a prediction-based controller for getting from the lunar surface back to orbit.

A. Parameterizing the NRHO

We obtain the NRHO data from a publicly available dataset[7]. This data is fitted to a fourth-order spline, resulting in two parametric curves $\boldsymbol{\gamma}$: one in the Earth-Centered Inertial (ECI) frame and the other in the Earth-Moon Barycentric Rotating (EMBR) frame. To assign the PHODCOS coordinate system to both curves, we apply Algorithm 2 using $n_s = 256$ segments.

The resultant coordinate systems are depicted in Fig. 1 (ECI) and Fig. 4 (EMBR). These figures confirm key characteristics of the NRHO: its high orbital eccentricity brings it very close to the lunar north pole, while it deviates significantly when passing over the south pole. This occurs approximately four times for each lunar orbit, consistent with the 9:2 orbital resonance, and results in four highly nonlinear transitions per orbital period. These transitions are clearly reflected in the parametric speed σ and the angular velocity $\boldsymbol{\omega}$ of the coordinate system, as shown in panels D and E of Fig. 4. Despite these abrupt transitions, as discussed in Section 5, PHODCOS ensures that all parametric functions maintain at least C^2 continuity, thus making them suitable for gradient-based algorithms, as will be demonstrated in the next subsection.

[7]NRHO data available at `https://ssd.jpl.nasa.gov/tools/periodic_orbits.html`. The specific settings for the selected orbit are as follows: Earth-Moon system, Halo orbit family, L2 libration point, 6-8 day period range, ID 95.

B. Predictive control for ascent of a lunar lander

To exemplify how PHODCOS can be leveraged in the design of optimization- or learning-based algorithms, we formulate a prediction-based control scheme based on the parameterization of the NRHO carried out in the previous subsection. More specifically, we design a trajectory optimization program that takes a lunar lander from the Moon's surface back to orbit.

In this model, the lunar lander is approximated as a gimbaled rocket with states defined as $\boldsymbol{x} = [\boldsymbol{p}, v, \theta, \phi] \in \mathbb{R}^6$, where $\boldsymbol{p} \in \mathbb{R}^3$ represents the position vector, and $\{v, \theta, \phi\} \in \mathbb{R}$ denote the velocity magnitude and gimbal angles, respectively. The system's control inputs are the acceleration magnitude $a \in \mathbb{R}$ and the gimbal's angular rates $\{\omega_\theta, \omega_\phi\} \in \mathbb{R}$, collectively forming the control vector $\boldsymbol{u} = [a, \omega_\theta, \omega_\phi] \in \mathbb{R}^3$. Consequently, the motions of the lunar lander are decribed by the following equations of motion:

$$\dot{\boldsymbol{p}} = v\,[\cos\phi\cos\theta,\ \sin\phi,\ \cos\phi\sin\theta] \tag{24a}$$

$$\dot{v} = a, \quad \dot{\theta} = \omega_\theta, \quad \dot{\phi} = \omega_\phi\,, \tag{24b}$$

which can be written in the common form of nonlinear dynamical system as $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$. By designating the PHODCOS coordinate system to the NRHO –as done in the previous subsection–, we can opt for a spatial parameterization, whereby the system dynamics expressed in (24) evolve according to the path parameter ξ, instead of the conventional time-based representation. To make this distinction clear and following the notation in Section 4, we rewrite the equations of motion as $\boldsymbol{x}'(\xi) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))$, where $(\cdot)' = \frac{\mathrm{d}(\cdot)}{\mathrm{d}\xi}$. Hence, rather than having a temporal horizon T, we have a spatial look-ahead $\Xi = \xi_f - \xi_0$. This implies that our prediction horizon pertains to a specific segment of the orbit, regardless of time.

Throughout this prediction interval, our objective is to find a trajectory to ascend from a given state $\boldsymbol{x_0}$ on the lunar surface back to the NRHO, while ensuring that the motion is dynamically compliant with (24) and adheres to all state and input limits, i.e., $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{u} \in \mathcal{U}$. To compute such a trajectory, we solve the subsequent Optimal Control Problem (OCP):

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \int_{\xi_0}^{\xi_f} ||\boldsymbol{p}(\xi) - \boldsymbol{p}_{\text{NRHO}}(\xi)||_Q^2 + ||\boldsymbol{u}(\xi)||_R^2\, d\xi \tag{25a}$$

$$\text{s.t.} \quad \boldsymbol{x}(\xi_0) = \boldsymbol{x_0}\,, \tag{25b}$$

$$\boldsymbol{x}'(\xi) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)), \quad \xi \in [\xi_0, \xi_f] \tag{25c}$$

$$\boldsymbol{x}(\xi) \in \mathcal{X}\,,\ \boldsymbol{u}(\xi) \in \mathcal{U}\,, \quad \xi \in [\xi_0, \xi_f]\,. \tag{25d}$$

where $\boldsymbol{p}_{\text{NRHO}}(\xi)$ is readily available from the parametric functions obtained in the previous subsection by applying the PHODCOS Algorithm 2 to the NRHO.

OCP (25) presents a basic example, as it only begins to explore the possibilities offered by parameterizing trajectory optimization problems using curvilinear differentiable coordinate systems like PHODCOS. This formulation can be extended to handle more complex behaviors, such as explicit control over the progress variable, enhancing robustness against disturbances, shaping the transverse distance to the orbit, or allocating protective funnels around the orbit [14, 16, 24]. For a comprehensive discussion of trajectory optimization strategies and applications beyond navigation, see [25]. Nevertheless, to maintain a clear and precise perspective, OCP (25) represents the simplest formulation, adequately showing that PHODCOS can be integrated into optimization- and learning-based algorithms, which require differentiating over parametric functions $\{\boldsymbol{p}_{\text{NRHO}}(\xi), \mathrm{R}_{\text{NRHO}}(\xi), \boldsymbol{\omega}_{\text{NRHO}}(\xi)\}$ by retrieving first- and second-order derivative information.

We approximate OCP (25) as a nonlinear program according to the multiple-shooting method [26] in which the optimization horizon $\Xi = 0.08$ is split into $N = 100$ segments, each with constant decision variables. To ensure the validity of the method, we consider five distinct initial locations on the lunar surface: the north and south poles, both extremes along the equator, and the region directly opposite the NRHO horizon segment.

The resulting trajectories, along with the Moon and the NRHO, are shown in Fig. 5. The segment of the orbit that falls under the prediction horizon and serves as a reference $\boldsymbol{p}_{\text{NRHO}}(\xi)$ is depicted by the dashed black line. The motions of the lunar lander are given in the left column of the figure, as well as in panels A), B), C) and D). These are colored according to the initial conditions, i.e., blue and orange for the north and south poles, green and red for the extremes along the equator, and purple for the region opposite to the horizon segment. To gain a better understanding of the convergence of these trajectories, panel E) shows the position $\boldsymbol{e}_p$ and orientation $\boldsymbol{e}_R$ errors associated to all five trajectories, as well as their respective velocity magnitudes v and gimbal angles θ, ϕ. From here, it is apparent that all five cases succeed in reaching the orbit within the prediction horizon.

7. CONCLUSIONS

In this work, we proposed PHODCOS, an algorithm to assign a differentiable coordinate system to any given curve. For this purpose, we rely on a Pythagorean Hodograph (PH) curve-based hermite interpolation scheme that parameterizes any arbitrary curve into a geometric reference with a moving frame associated with it. The conversion algorithm is characterized by three attractive properties: efficiency, bounded conversion error, and differentiability. First, all operations are provided in closed form, eliminating the need for iterative or numerical procedures. This makes PHODCOS computationally lightweight and suitable for real-time applications. Second, we have proved analytically and numerically that the algorithm achieves a sixth-order approximation, ensuring that the deviation between the coordinate system and the curve remains within a predefined tolerance. Third, the parametric functions produced by PHODCOS are at least two times differentiable, making them compatible with gradient-based numerical routines. To demonstrate the practical utility of PHODCOS, we have applied it to the Near-Rectilinear Halo Orbit (NRHO) and formulated a trajectory optimization problem to guide a lunar lander from the Moon's surface back to the NRHO. To facilitate broader use in science and engineering, we have made PHODCOS available as an open source software library, along with the examples presented.

APPENDICES

A. DERIVATION OF PREIMAGE CONTROL-POINT $\mathcal{A}_4$

$$
\begin{aligned}
\boldsymbol{c_p} = \frac{1}{112633092}(&147807\mathcal{A}_0\boldsymbol{i}\mathcal{A}_0^* + 72270\mathcal{A}_0\boldsymbol{i}\mathcal{A}_1^* + \\
&30954\mathcal{A}_0\boldsymbol{i}\mathcal{A}_2^* + 9702\mathcal{A}_0\boldsymbol{i}\mathcal{A}_3^* - 3234\mathcal{A}_0\boldsymbol{i}\mathcal{A}_5^* - \\
&3150\mathcal{A}_0\boldsymbol{i}\mathcal{A}_6^* - 1818\mathcal{A}_0\boldsymbol{i}\mathcal{A}_7^* - 565\mathcal{A}_0\boldsymbol{i}\mathcal{A}_8^* + \\
&72270\mathcal{A}_1\boldsymbol{i}\mathcal{A}_0^* + 72732\mathcal{A}_1\boldsymbol{i}\mathcal{A}_1^* + 47124\mathcal{A}_1\boldsymbol{i}\mathcal{A}_2^* + \\
&19404\mathcal{A}_1\boldsymbol{i}\mathcal{A}_3^* - 8820\mathcal{A}_1\boldsymbol{i}\mathcal{A}_5^* - 9324\mathcal{A}_1\boldsymbol{i}\mathcal{A}_6^* - \\
&5668\mathcal{A}_1\boldsymbol{i}\mathcal{A}_7^* - 1818\mathcal{A}_1\boldsymbol{i}\mathcal{A}_8^* + 30954\mathcal{A}_2\boldsymbol{i}\mathcal{A}_0^* + \\
&47124\mathcal{A}_2\boldsymbol{i}\mathcal{A}_1^* + 40572\mathcal{A}_2\boldsymbol{i}\mathcal{A}_2^* + 20580\mathcal{A}_2\boldsymbol{i}\mathcal{A}_3^* - \\
&12348\mathcal{A}_2\boldsymbol{i}\mathcal{A}_5^* - 14308\mathcal{A}_2\boldsymbol{i}\mathcal{A}_6^* - 9324\mathcal{A}_2\boldsymbol{i}\mathcal{A}_7^* - \\
&3150\mathcal{A}_2\boldsymbol{i}\mathcal{A}_8^* + 9702\mathcal{A}_3\boldsymbol{i}\mathcal{A}_0^* + 19404\mathcal{A}_3\boldsymbol{i}\mathcal{A}_1^* + \\
&20580\mathcal{A}_3\boldsymbol{i}\mathcal{A}_2^* + 12348\mathcal{A}_3\boldsymbol{i}\mathcal{A}_3^* - 9604\mathcal{A}_3\boldsymbol{i}\mathcal{A}_5^* - \\
&12348\mathcal{A}_3\boldsymbol{i}\mathcal{A}_6^* - 8820\mathcal{A}_3\boldsymbol{i}\mathcal{A}_7^* - 3234\mathcal{A}_3\boldsymbol{i}\mathcal{A}_8^* - \\
&3234\mathcal{A}_5\boldsymbol{i}\mathcal{A}_0^* - 8820\mathcal{A}_5\boldsymbol{i}\mathcal{A}_1^* - 12348\mathcal{A}_5\boldsymbol{i}\mathcal{A}_2^* - \\
&9604\mathcal{A}_5\boldsymbol{i}\mathcal{A}_3^* + 12348\mathcal{A}_5\boldsymbol{i}\mathcal{A}_5^* + 20580\mathcal{A}_5\boldsymbol{i}\mathcal{A}_6^* + \\
&19404\mathcal{A}_5\boldsymbol{i}\mathcal{A}_7^* + 9702\mathcal{A}_5\boldsymbol{i}\mathcal{A}_8^* - 3150\mathcal{A}_6\boldsymbol{i}\mathcal{A}_0^* - \\
&9324\mathcal{A}_6\boldsymbol{i}\mathcal{A}_1^* - 14308\mathcal{A}_6\boldsymbol{i}\mathcal{A}_2^* - 12348\mathcal{A}_6\boldsymbol{i}\mathcal{A}_3^* + \\
&20580\mathcal{A}_6\boldsymbol{i}\mathcal{A}_5^* + 40572\mathcal{A}_6\boldsymbol{i}\mathcal{A}_6^* + 47124\mathcal{A}_6\boldsymbol{i}\mathcal{A}_7^* + \\
&30954\mathcal{A}_6\boldsymbol{i}\mathcal{A}_8^* - 1818\mathcal{A}_7\boldsymbol{i}\mathcal{A}_0^* - 5668\mathcal{A}_7\boldsymbol{i}\mathcal{A}_1^* - \\
&9324\mathcal{A}_7\boldsymbol{i}\mathcal{A}_2^* - 8820\mathcal{A}_7\boldsymbol{i}\mathcal{A}_3^* + 19404\mathcal{A}_7\boldsymbol{i}\mathcal{A}_5^* + \\
&47124\mathcal{A}_7\boldsymbol{i}\mathcal{A}_6^* + 72732\mathcal{A}_7\boldsymbol{i}\mathcal{A}_7^* + 72270\mathcal{A}_7\boldsymbol{i}\mathcal{A}_8^* - \\
&565\mathcal{A}_8\boldsymbol{i}\mathcal{A}_0^* - 1818\mathcal{A}_8\boldsymbol{i}\mathcal{A}_1^* - 3150\mathcal{A}_8\boldsymbol{i}\mathcal{A}_2^* - \\
&3234\mathcal{A}_8\boldsymbol{i}\mathcal{A}_3^* + 9702\mathcal{A}_8\boldsymbol{i}\mathcal{A}_5^* + 30954\mathcal{A}_8\boldsymbol{i}\mathcal{A}_6^* + \\
&72270\mathcal{A}_8\boldsymbol{i}\mathcal{A}_7^* + 147807\mathcal{A}_8\boldsymbol{i}\mathcal{A}_8^*)\,.
\end{aligned}
\tag{26}
$$

Using the commutative operation declared in Definition 1, eq. (26) can be simplified into (17c).

A. More Comments

Use section and subsection keywords as usual.

B. THEOREMS FOR $\phi^* = \mathbf{0}$

Theorem 2 (**Planarity preservation**): *Given C^4 hermite data confined to a plane $P \subset \mathbb{R}^3$, the interpolant $\boldsymbol{p}_{\boldsymbol{\phi}^*}$ remains within P.*

Proof. See Theorem 3.11 in [1]. □

Theorem 3 (**Invariance of interpolants**): *The parameterization of solutions, as described in Definition 3, remains unchanged under rigid body motions, i.e., special orthogonal transformations). However, reflections invert the signs of all parameters, meaning $\boldsymbol{\phi} \to -\boldsymbol{\phi}$. Hence, $\boldsymbol{\phi}^* = \mathbf{0}$ exhibits invariance under all orthogonal transformations.*

Proof. See Lemma 3.6 in [1] □

Theorem 4 (**Reversion invariance**): *Define $\boldsymbol{p}_\phi(\xi)$ as the interpolants of the dataset $\{\boldsymbol{p}_b,\ \boldsymbol{p}_e,\ \boldsymbol{v}_b,\ \boldsymbol{v}_e,\ \boldsymbol{a}_b,\ \boldsymbol{a}_e,\ \boldsymbol{j}_b,\ \boldsymbol{j}_e,\ \boldsymbol{s}_b,\ \boldsymbol{s}_e\}$. Similarly, let $\bar{\boldsymbol{p}}_\phi(\xi)$ represent the interpolants of the corresponding "reversed" dataset $\{\bar{\boldsymbol{p}}_b = \boldsymbol{p}_e,\ \bar{\boldsymbol{v}}_b = \boldsymbol{v}_e,\ \bar{\boldsymbol{a}}_b = \boldsymbol{a}_e,\ \bar{\boldsymbol{j}}_b = \boldsymbol{j}_e,\ \bar{\boldsymbol{s}}_b = \boldsymbol{s}_e\}$ and $\{\bar{\boldsymbol{p}}_e = \boldsymbol{p}_b,\ \bar{\boldsymbol{v}}_e = \boldsymbol{v}_b,\ \bar{\boldsymbol{a}}_e = \boldsymbol{a}_b,\ \bar{\boldsymbol{j}}_e = \boldsymbol{j}_b,\ \bar{\boldsymbol{s}}_e = \boldsymbol{s}_b\}$. Consequently, we have that $\bar{\boldsymbol{p}}_{\boldsymbol{\phi}}(1-\xi) = \boldsymbol{p}_{-\phi}(\xi)$, leading to $\boldsymbol{p}_{\boldsymbol{\phi}^*} = \bar{\boldsymbol{p}}_{\boldsymbol{\phi}^*}$.*

Proof. See Theorem 3.10 in [1]. □

C. DERIVATION OF HODOGRAPH CONTROL-POINTS

Focusing the specific case of C^4 interpolation and a preimage $\mathcal{A}(\xi)$ of order 8, the extended Bernstein Bezier form in (9) is

$$
\begin{aligned}
\mathcal{A}(\xi) = &\xi^8\mathcal{A}_8 + 8(1-\xi)\xi^7\mathcal{A}_7 + 28(1-\xi)^2\xi^6\mathcal{A}_6 + \\
&56(1-\xi)^3\xi^5\mathcal{A}_5 + 70(1-\xi)^4\xi^4\mathcal{A}_4 + \\
&56(1-\xi)^5\xi^3\mathcal{A}_3 + 28(1-\xi)^6\xi^2\mathcal{A}_2 + \\
&8(1-\xi)^7\xi\mathcal{A}_1 + (1-\xi)^8\mathcal{A}_0\,.
\end{aligned}
\tag{27}
$$

Leveraging the quadratic expression in (9), the control-points of the hodograph $\boldsymbol{h}(\xi)$ are defined as:

$$\boldsymbol{h}_0 = \mathcal{A}_0\boldsymbol{i}\mathcal{A}_0^*\,, \tag{28a}$$

$$\boldsymbol{h}_1 = \frac{1}{16}(8\mathcal{A}_0\boldsymbol{i}\mathcal{A}_1^* + 8\mathcal{A}_1\boldsymbol{i}\mathcal{A}_0^*)\,, \tag{28b}$$

$$\boldsymbol{h}_2 = \frac{1}{120}(28\mathcal{A}_0\boldsymbol{i}\mathcal{A}_2^* + 64\mathcal{A}_1\boldsymbol{i}\mathcal{A}_1^* + 28\mathcal{A}_2\boldsymbol{i}\mathcal{A}_0^*)\,, \tag{28c}$$

$$
\begin{aligned}
\boldsymbol{h}_3 = \frac{1}{560}(&56\mathcal{A}_0\boldsymbol{i}\mathcal{A}_3^* + 224\mathcal{A}_1\boldsymbol{i}\mathcal{A}_2^* + 224\mathcal{A}_2\boldsymbol{i}\mathcal{A}_1^* + \\
&56\mathcal{A}_3\boldsymbol{i}\mathcal{A}_0^*)\,,
\end{aligned}
\tag{28d}
$$

$$
\begin{aligned}
\boldsymbol{h}_4 = \frac{1}{1820}(&70\mathcal{A}_0\boldsymbol{i}\mathcal{A}_4^* + 448\mathcal{A}_1\boldsymbol{i}\mathcal{A}_3^* + 784\mathcal{A}_2\boldsymbol{i}\mathcal{A}_2^* + \\
&448\mathcal{A}_3\boldsymbol{i}\mathcal{A}_1^* + 70\mathcal{A}_4\boldsymbol{i}\mathcal{A}_0^*)\,,
\end{aligned}
\tag{28e}
$$

$$
\begin{aligned}
\boldsymbol{h}_5 = \frac{1}{4368}(&56\mathcal{A}_0\boldsymbol{i}\mathcal{A}_5^* + 560\mathcal{A}_1\boldsymbol{i}\mathcal{A}_4^* + 1568\mathcal{A}_2\boldsymbol{i}\mathcal{A}_3^* + \\
&1568\mathcal{A}_3\boldsymbol{i}\mathcal{A}_2^* + 560\mathcal{A}_4\boldsymbol{i}\mathcal{A}_1^* + 56\mathcal{A}_5\boldsymbol{i}\mathcal{A}_0^*)\,,
\end{aligned}
\tag{28f}
$$

$$
\begin{aligned}
\boldsymbol{h}_6 = \frac{1}{8008}(&28\mathcal{A}_0\boldsymbol{i}\mathcal{A}_6^* + 448\mathcal{A}_1\boldsymbol{i}\mathcal{A}_5^* + 1960\mathcal{A}_2\boldsymbol{i}\mathcal{A}_4^* + \\
&3136\mathcal{A}_3\boldsymbol{i}\mathcal{A}_3^* + 1960\mathcal{A}_4\boldsymbol{i}\mathcal{A}_2^* + 448\mathcal{A}_5\boldsymbol{i}\mathcal{A}_1^* + \\
&28\mathcal{A}_6\boldsymbol{i}\mathcal{A}_0^*)\,,
\end{aligned}
\tag{28g}
$$

$$
\begin{aligned}
\boldsymbol{h}_7 = \frac{1}{11440}(&8\mathcal{A}_0\boldsymbol{i}\mathcal{A}_7^* + 224\mathcal{A}_1\boldsymbol{i}\mathcal{A}_6^* + 1568\mathcal{A}_2\boldsymbol{i}\mathcal{A}_5^* + \\
&3920\mathcal{A}_3\boldsymbol{i}\mathcal{A}_4^* + 3920\mathcal{A}_4\boldsymbol{i}\mathcal{A}_3^* + 1568\mathcal{A}_5\boldsymbol{i}\mathcal{A}_2^* + \\
&224\mathcal{A}_6\boldsymbol{i}\mathcal{A}_1^* + 8\mathcal{A}_7\boldsymbol{i}\mathcal{A}_0^*)\,,
\end{aligned}
\tag{28h}
$$

$$
\begin{aligned}
\boldsymbol{h}_8 = \frac{1}{12870}(&\mathcal{A}_0\boldsymbol{i}\mathcal{A}_8^* + 64\mathcal{A}_1\boldsymbol{i}\mathcal{A}_7^* + 784\mathcal{A}_2\boldsymbol{i}\mathcal{A}_6^* + \\
&3136\mathcal{A}_3\boldsymbol{i}\mathcal{A}_5^* + 4900\mathcal{A}_4\boldsymbol{i}\mathcal{A}_4^* + 3136\mathcal{A}_5\boldsymbol{i}\mathcal{A}_3^* + \\
&784\mathcal{A}_6\boldsymbol{i}\mathcal{A}_2^* + 64\mathcal{A}_7\boldsymbol{i}\mathcal{A}_1^* + \mathcal{A}_8\boldsymbol{i}\mathcal{A}_0^*)\,,
\end{aligned}
\tag{28i}
$$

$$
\begin{aligned}
\boldsymbol{h}_9 = \frac{1}{11440}(&8\mathcal{A}_1\boldsymbol{i}\mathcal{A}_8^* + 224\mathcal{A}_2\boldsymbol{i}\mathcal{A}_7^* + 1568\mathcal{A}_3\boldsymbol{i}\mathcal{A}_6^* + \\
&3920\mathcal{A}_4\boldsymbol{i}\mathcal{A}_5^* + 3920\mathcal{A}_5\boldsymbol{i}\mathcal{A}_4^* + 1568\mathcal{A}_6\boldsymbol{i}\mathcal{A}_3^* + \\
&224\mathcal{A}_7\boldsymbol{i}\mathcal{A}_2^* + 8\mathcal{A}_8\boldsymbol{i}\mathcal{A}_1^*)\,,
\end{aligned}
\tag{28j}
$$

$$\boldsymbol{h}_{10} = \frac{1}{8008}(28\mathcal{A}_2\boldsymbol{i}\mathcal{A}_8^* + 448\mathcal{A}_3\boldsymbol{i}\mathcal{A}_7^* + 1960\mathcal{A}_4\boldsymbol{i}\mathcal{A}_6^* + 3136\mathcal{A}_5\boldsymbol{i}\mathcal{A}_5^* + 1960\mathcal{A}_6\boldsymbol{i}\mathcal{A}_4^* + 448\mathcal{A}_7\boldsymbol{i}\mathcal{A}_3^* + 28\mathcal{A}_8\boldsymbol{i}\mathcal{A}_2^*)\,, \tag{28k}$$

$$\boldsymbol{h}_{11} = \frac{1}{4368}(56\mathcal{A}_3\boldsymbol{i}\mathcal{A}_8^* + 560\mathcal{A}_4\boldsymbol{i}\mathcal{A}_7^* + 1568\mathcal{A}_5\boldsymbol{i}\mathcal{A}_6^* + 1568\mathcal{A}_6\boldsymbol{i}\mathcal{A}_5^* + 560\mathcal{A}_7\boldsymbol{i}\mathcal{A}_4^* + 56\mathcal{A}_8\boldsymbol{i}\mathcal{A}_3^*)\,, \tag{28l}$$

$$\boldsymbol{h}_{12} = \frac{1}{1820}(70\mathcal{A}_4\boldsymbol{i}\mathcal{A}_8^* + 448\mathcal{A}_5\boldsymbol{i}\mathcal{A}_7^* + 784\mathcal{A}_6\boldsymbol{i}\mathcal{A}_6^* + 448\mathcal{A}_7\boldsymbol{i}\mathcal{A}_5^* + 70\mathcal{A}_8\boldsymbol{i}\mathcal{A}_4^*)\,, \tag{28m}$$

$$\boldsymbol{h}_{13} = \frac{1}{560}(56\mathcal{A}_5\boldsymbol{i}\mathcal{A}_8^* + 224\mathcal{A}_6\boldsymbol{i}\mathcal{A}_7^* + 224\mathcal{A}_7\boldsymbol{i}\mathcal{A}_6^* + 56\mathcal{A}_8\boldsymbol{i}\mathcal{A}_5^*)\,, \tag{28n, 28o}$$

$$\boldsymbol{h}_{14} = \frac{1}{120}(28\mathcal{A}_6\boldsymbol{i}\mathcal{A}_8^* + 64\mathcal{A}_7\boldsymbol{i}\mathcal{A}_7^* + 28\mathcal{A}_8\boldsymbol{i}\mathcal{A}_6^*)\,, \tag{28p}$$

$$\boldsymbol{h}_{15} = \frac{1}{16}(8\mathcal{A}_7\boldsymbol{i}\mathcal{A}_8^* + 8\mathcal{A}_8\boldsymbol{i}\mathcal{A}_7^*)\,, \tag{28q}$$

$$\boldsymbol{h}_{16} = \mathcal{A}_8\boldsymbol{i}\mathcal{A}_8^*\,. \tag{28r}$$

Using the commutative operation declared in Definition 1, these equations can be simplified into (15). This is the second appendix.

REFERENCES

[1] Z. Šír and B. Jüttler, "C^2 hermite interpolation by pythagorean hodograph space curves," *Mathematics of Computation*, vol. 76, no. 259, pp. 1373–1391, 2007.

[2] D. J. Struik, *Lectures on classical differential geometry*. Courier Corporation, 1961.

[3] E. Abbena, S. Salamon, and A. Gray, *Modern differential geometry of curves and surfaces with Mathematica*. Chapman and Hall/CRC, 2017.

[4] R. L. Bishop, "There is more than one way to frame a curve," *The American Mathematical Monthly*, vol. 82, no. 3, pp. 246–251, 1975.

[5] A. J. Hanson and H. Ma, "Parallel transport approach to curve framing," *Indiana University, Techreports-TR425*, vol. 11, pp. 3–7, 1995.

[6] W. Wang, B. Jüttler, D. Zheng, and Y. Liu, "Computation of rotation minimizing frames," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 1, pp. 1–18, 2008.

[7] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Acikmese, "Convex optimization for trajectory generation," *arXiv preprint arXiv:2106.09125*, 2021.

[8] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[9] H. I. Choi and C. Y. Han, "Euler–rodrigues frames on spatial pythagorean-hodograph curves," *Computer Aided Geometric Design*, vol. 19, no. 8, pp. 603–620, 2002.

[10] R. T. Farouki, *Pythagorean—Hodograph Curves*. Springer, 2008.

[11] G. Albrecht, C. V. Beccari, and L. Romani, "Spatial pythagorean-hodograph b–spline curves and 3d point data interpolation," *Computer Aided Geometric Design*, vol. 80, p. 101868, 2020.

[12] R. Farouki, C. Giannelli, C. Manni, and A. Sestini, "Design of rational rotation–minimizing rigid body motions by hermite interpolation," *Mathematics of Computation*, vol. 81, no. 278, pp. 879–903, 2012.

[13] G. Otto, G. van Schoor, and K. R. Uren, "Geometric-dynamic trajectory: A quaternion pythagorean hodograph curves approach," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 2, pp. 283–294, 2021.

[14] J. Arrizabalaga and M. Ryll, "Spatial motion planning with pythagorean hodograph curves," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2047–2053.

[15] ——, "Sctomp: Spatially constrained time-optimal motion planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 4827–4834.

[16] J. Arrizabalaga, Z. Manchester, and M. Ryll, "Differentiable collision-free parametric corridors," *arXiv preprint arXiv:2407.12283*, 2024.

[17] J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999.

[18] R. Dietz, J. Hoschek, and B. Jüttler, "An algebraic approach to curves and surfaces on the sphere and on other quadrics," *Computer Aided Geometric Design*, vol. 10, no. 3-4, pp. 211–229, 1993.

[19] R. T. Farouki, M. Al-Kandari, and T. Sakkalis, "Structural invariance of spatial pythagorean hodographs," *Computer Aided Geometric Design*, vol. 19, no. 6, pp. 395–407, 2002.

[20] Z. Šír and B. Jüttler, "Spatial pythagorean hodograph quintics and the approximation of pipe surfaces," in *Mathematics of Surfaces XI: 11th IMA International Conference, Loughborough, UK, September 5-7, 2005. Proceedings*. Springer, 2005, pp. 364–380.

[21] R. J. Whitley, D. C. Davis, L. M. Burke, B. P. McCarthy, R. J. Power, M. L. McGuire, and K. C. Howell, "Earth-moon near rectilinear halo and butterfly orbits for lunar surface exploration," in *AAS/AIAA Astrodynamics Specialists Conference, Snowbird, Utah*, 2018.

[22] J. Sullivan, S. Grimberg, and S. D'Amico, "Comprehensive survey and assessment of spacecraft relative motion dynamics models," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 8, pp. 1837–1859, 2017.

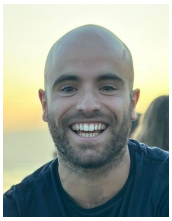
[23] J. B. Willis and Z. Manchester, "Convex optimization of relative orbit maneuvers using the kustaanheimo-stiefel transformation," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–7.

[24] J. Arrizabalaga and M. Ryll, "Pose-following with dual quaternions," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 5959–5966.

[25] ——, "A universal formulation for path-parametric planning and control," 2024.

[26] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

BIOGRAPHY



Jon Arrizabalaga *is pursuing his PhD at the Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, and is a visiting researcher at the Robotics Institute of Carnegie Mellon University. He received a BSc. in mechanical engineering from the University of Navarre (Spain, 2018) and an MSc. in mechatronics and robotics at KTH Royal Institute of Technology (Sweden, 2020). His research focuses on the convergence of optimization, control, and geometry, particularly in their applications to robotics and autonomous systems.*

Fausto Vega *is a PhD student at the Robotics Institute at Carnegie Mellon University (CMU). He received a BS in mechanical engineering from the University of Nevada, Las Vegas and an MS in Robotics at CMU. His research interests include optimal control for maneuver design small spacecraft state estimation.*

Zbyněk Šír *is an associate professor at the Mathematical Institute in the Faculty of Mathematics and Physics of Charles University in Prague. He received a DEA (Diplome d'études approfondies) from University Pierre et Marie Curie, a Master with specelization in Riemannian Geometry and Theory of Representations, and a PhD in mathematics with specialization in history of French geometry, both from Charles University of Prague. His research interestes include CAGD, theoretical differential geometry, history of geometry and other applied geometric fields.*

Zachary Manchester *is an assistant professor in the Robotics Institute at Carnegie Mellon University and founder of the Robotic Exploration Lab. He received a PhD in aerospace engineering in 2015 and a BS in applied physics in 2009, both from Cornell University. His research interests include control and optimization with applications to aerospace and robotic systems.*

Markus Ryll *is an assistant professor in the Department of Aerospace and Geodesy at the Technical University of Munich and head of the Autonomous Aerial Systems Lab. He received a PhD in 2014 from Max Planck Institute for Biological Cybernetics. Between 2014 and 2017, he was a postdoctoral researcher at the Laboratory for Analysis and Architecture of Systems (LAAS-CNRS). From 2018 to 2020, he continued his postdoctoral work in the Robust Robotics Group at the Massachusetts Institute of Technology (MIT, CSAIL). His research focuses on enabling autonomous aerial robots to interact with real-world environments.*

6 Environment Representation for Egocentric Decision-Making

Path-parametric methods provide a natural framework for defining spatial coordinates, inherently capturing the concept of progression along a trajectory. At the same time, they allow spatial constraints to be enforced as convex bounds on the orthogonal components of the spatial states (see Fig. 3.2). These characteristics make them a valuable tool in navigation, particularly for planning and control algorithms.

Consider, for instance, the quadrotor scenario in Chapter 3, where an explicit representation of the admissible region enables the system to make efficient use of the available space. Likewise, in racing, whether the goal is minimizing lap time or maximizing progress, parameterizing the track explicitly allows controllers to fully exploit its width, leading to optimal maneuvering strategies. These examples illustrate the power of using spatial coordinates to define system motion. However, they also raise a fundamental question:

> *How can one construct a spatial representation that effectively defines the admissible region around an arbitrary reference path as a function of the orthogonal spatial component, η?*

6.1 Existing methods for spatial representation

Most methods for representing space rely on convex decomposition [58, 59, 60, 61], which partitions the free space into convex polyhedra. However, this representation is fundamentally incompatible with egocentric formulations. This incompatibility arises from two key limitations. First, spatial representations based on convex decomposition are designed for *Euclidean* state formulations and therefore fail to exploit the advantages of imposing convex constraints directly on the orthogonal spatial coordinate, η. Second, discretizing free space into multiple convex sets introduces the *polyhedron allocation problem*, requiring each state to be explicitly assigned to a specific polyhedron. This discretization disrupts the smoothness of the corridor representation with respect to the progress variable, complicating the integration of collision-free corridors into optimization and learning-based frameworks.

Given this context, where conventional collision-free generation algorithms cannot be directly applied to parametric formulations, the corridors produced by studies utilizing egocentric formulations are often tailored to specific cases, making them difficult to generalize across diverse scenarios [36, 34, 1, 62]. To address these limitations, the manuscript attached to this chapter and published in [4], introduces a method for computing differentiable, continuous, and collision-free parametric corridors. This approach seamlessly integrates with planning and control algorithms that are parameterized by spatial coordinates. Therefore, we proceed by outlining the key elements of this spatial representation and explaining its connection to the broader path-parametric framework.

6.2 Differentiable Parametric Corridors

Motivated by the limitations of convex decomposition-based corridors for egocentric decision-making, the manuscript attached to this chapter introduces a method for computing corridors that are differentiable, continuous, and collision-free. In this section, we outline the key components of this approach and its pivotal role within egocentric robot auton.

6.2.1 Choosing an Off-Centered Ellipse as the Cross-Section

A parametric corridor is defined by a cross-section that sweeps along a user-defined path. Therefore, selecting a cross-section that optimally adapts to the obstacle-free space while ensuring differentiability and computational feasibility is crucial.

The simplest choice is a circle, which offers a single degree of freedom (the radius). An ellipse increases the degrees of freedom to three, and allowing it to be offset from the path increases the total degrees of freedom to five. Beyond these basic shapes, more complex options, such as polyhedra or semialgebraic sets, exist. However, these cross-sections cannot guarantee differentiability or closedness throughout the entire corridor.

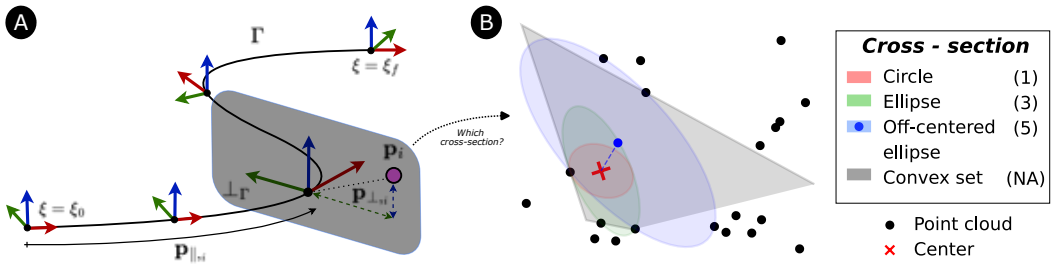


Figure 6.1 Defining the parametric corridor as a surface that sweeps along the reference path Γ simplifies spatial constraints by imposing limits on the orthogonal components, as illustrated in panel A. This raises the question of which cross-section is most effective in maximizing the corridor's coverage. Panel B provides the answer by showing different cross-section parameterizations in order of increasing degrees of freedom (DOF): a circle (red, 1 DOF), an ellipse (green, 3 DOF), and an offset ellipse (blue, 5 DOF). The gray area represents the maximum convex polygon that encompasses the center of the given point cloud. The offset for the off-centered ellipse is indicated by the blue dashed line.

Thus, the most favorable option, among feasible ones, is a rotating off-centered ellipse. Mathematically, this ellipse is defined by the matrix E and the offset $\boldsymbol{p_E}$, and its equation is given by

$$(\boldsymbol{x}_\perp - \boldsymbol{p_E})^\intercal \mathrm{E} (\boldsymbol{x}_\perp - \boldsymbol{p_E}) \leq 1\,, \tag{6.1}$$

as depicted in Figure 6.1.

6.2.2 Parameterizing the Ellipse with Polynomials

Once the cross-section is defined as a rotating off-centered ellipse, the next step is to parameterize it so that it sweeps the reference path from start to end. This implies that both the ellipse and its defining parameters, E and $\boldsymbol{p_E}$, evolve with respect to the path parameter ξ. To achieve this, we parameterize these variables using Chebyshev polynomials:

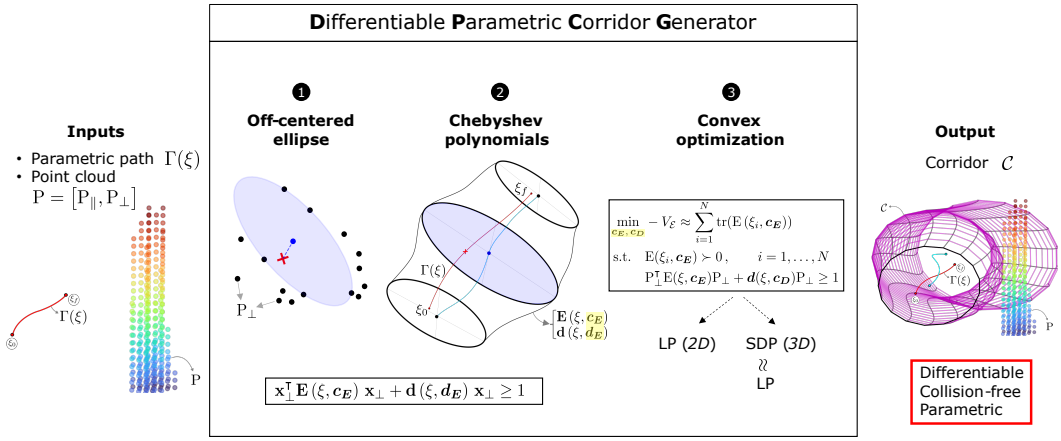


Figure 6.2 A summary of the corridor generation approach for generating differentiable, continuous, and parametric corridors that align seamlessly with egocentric decision-making. The method relies solely on a parametric path and a point cloud to construct the corridors, offering a versatile, system-independent solution that extends the scope of egocentric methods to real-world applications, even in dynamic and unstructured environments. This is accomplished through three primary steps: (1) choosing an off-centered ellipse as the cross-sectional shape of the corridor, (2) employing Chebyshev polynomials to parameterize the cross-section, and (3) casting the corridor volume maximization problem as a simple convex optimization task, enabling real-time solutions.

$$\{\mathrm{E}(\xi), \boldsymbol{p_E}(\xi)\} \rightarrow \{\mathrm{E}(\xi, \boldsymbol{c}_E), \boldsymbol{p_E}(\xi, \boldsymbol{p}_E)\} \tag{6.2}$$

where $\boldsymbol{c}_E$ and $\boldsymbol{p}_E$ are the coefficients of the polynomials. There are two key reasons for selecting Chebyshev polynomials: First, using polynomials ensures that the resulting corridors exhibit smoothness and differentiability. Second, the Chebyshev basis guarantees numerical stability, even for high-degree polynomials. Intuitively, increasing the polynomial degree enhances the corridor's ability to adapt to variations along the reference path.

By introducing the polynomial parameterization in (6.2) into the off-centered ellipse cross-section from (6.1), we obtain:

$$(\boldsymbol{x}_\perp - \boldsymbol{p_E}(\xi, \boldsymbol{c_P}))^\intercal \mathrm{E}(\xi, \boldsymbol{c_E})(\boldsymbol{x}_\perp - \boldsymbol{p_E}(\xi, \boldsymbol{c_P})) \leq 1\,.$$

Removing the nonlinearities results in:

$$\boldsymbol{x}_\perp^\intercal \mathrm{E}(\xi, \boldsymbol{c_E})\boldsymbol{x}_\perp - \boldsymbol{d}(\xi, \boldsymbol{d_E}) \leq 1\,, \tag{6.3}$$

where $\boldsymbol{d}(\xi, \boldsymbol{c_D})$ maintains the offset of the ellipse, while eliminating the nonlinearities from (6.2). From these steps, it becomes clear that the shape and size of the corridor are fully determined by the polynomial coefficients $\boldsymbol{c}_E$ and $\boldsymbol{d}_E$. This naturally leads to the question: how can these coefficients be determined? To answer this, we turn to the third and final component of the methodology.

6.2.3 Corridor volume maximization as convex optimization

From equation (6.3), it is clear that the shape and dimensions of the corridor are entirely governed by the polynomial coefficients $\boldsymbol{c_E}$ and $\boldsymbol{d_E}$. Our objective is to identify the collision-free corridor with the largest possible volume. To achieve this, we frame an optimization

problem that maximizes the corridor's volume while ensuring that the ellipse matrix E remains positive definite throughout the corridor, and that no point in the point cloud $\mathrm{P}_\perp$ lies inside the corridor. Given that the area of the ellipse in (6.1) is represented by $A_\mathcal{E} = \pi/\sqrt{\det \mathrm{E}}$, the optimization problem we solve is:

$$\max_{\boldsymbol{c_E}, \boldsymbol{d_E}} V_\mathcal{E} = \int_{\xi_0}^{\xi_f} -\det \mathrm{E}(\xi, \boldsymbol{c_E})\, d\xi \tag{6.4a}$$

$$\text{s.t.} \quad \mathrm{E}(\xi, \boldsymbol{c_E}) \succ 0\,,$$

$$\mathrm{P}_\perp^\intercal \mathrm{E}(\xi, \boldsymbol{c_E}) \mathrm{P}_\perp - \boldsymbol{d}(\xi, \boldsymbol{d_E}) \leq 1\,.$$

To make the problem computationally feasible, we employ several approximations. First, we discretize the continuous components of the optimization problem into N evaluations. Second, we approximate the nonlinear determinant of the ellipse by using the trace. Third, we add a wrapper around the reference path to guarantee that the problem remains bounded. By applying these approximations, the original problem is transformed into a convex optimization problem:

$$\min_{\boldsymbol{c_E}, \boldsymbol{d_E}} -V_\mathcal{E} \approx \sum_{i=1}^{N} \mathrm{tr}\left(\mathrm{E}(\xi_i, \boldsymbol{c_E})\right) \tag{6.5a}$$

$$\text{s.t.} \quad \mathrm{E}(\xi_i, \boldsymbol{c_E}) \succ 0\,, \tag{6.5b}$$

$$\mathrm{P}_\perp^\intercal \mathrm{E}(\xi_i, \boldsymbol{c_E}) \mathrm{P}_\perp - \boldsymbol{d}(\xi_i, \boldsymbol{d_E}) \leq 1\,. \tag{6.5c}$$

Specifically, the optimization problem in (6.5) is a Semidefinite Program (SDP). Although SDPs are convex, they are among the most challenging convex problems to solve. To simplify, we replace the linear matrix inequality in (6.5b) with a diagonal dominance condition:

$$\mathrm{E} = \begin{bmatrix} \mathrm{E}_{11} & \mathrm{E}_{12} \\ \mathrm{E}_{12} & \mathrm{E}_{22} \end{bmatrix} \succ 0 \quad \rightarrow \quad \mathrm{E}_{11}, \mathrm{E}_{22} \gg \mathrm{E}_{12}\,. \tag{6.6}$$

This approximation reduces the SDP to a Linear Program (LP), which is well-understood and computationally efficient. LPs can be solved very efficiently with standard solvers. Finally, it is worth noting that in the planar case, the diagonal dominance approximation in (6.6) is unnecessary, as the original problem is already an LP.

6.3 Enclosed Publication

The original scientific publication introducing the corridor generation method described in the previous sections is cited in [4] and attached at the end of this chapter. An open-source implementation of the method, along with the examples featured in the paper, is available at `https://github.com/jonarriza96/corrgen`. A brief promotional video can be viewed at `https://youtu.be/MvC7bPodXz8`, and a recording of the corresponding conference presentation is accessible at `https://youtu.be/l6LAugm89mQ`.

Differentiable Collision-Free Parametric Corridors

Jon Arrizabalaga, Zachary Manchester and Markus Ryll

published in

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

2024

Contribution This paper presents a spatial representation compatible with egocentric decision-making, providing a tool for representing the safe space of arbitrary environments without requiring prior identification. To achieve this, the method models collision-free space using parametric corridors that are smooth and differentiable, making them suitable for integration into gradient-based decision-making algorithms such as optimization and learning. I identified the need for an algorithm to compute such corridors, motivated primarily by the works presented in Chapters 3 and 4.3.1, while the idea of formulating the volume maximization problem originated with Zachary Manchester. I was responsible for developing this concept into a complete solution—both conceptually and in implementation—which ultimately led to the results presented in this paper.

Differentiable Collision-Free Parametric Corridors

Jon Arrizabalaga[1,2], Zachary Manchester[2], Markus Ryll[1]

***Abstract*— This paper presents a method to compute differentiable collision-free parametric corridors. In contrast to existing solutions that decompose the obstacle-free space into multiple convex sets, the continuous corridors computed by our method are smooth and differentiable, making them compatible with existing numerical techniques for learning and optimization. To achieve this, we represent the collision-free corridors as a path-parametric off-centered ellipse with a polynomial basis. We show that the problem of maximizing the volume of such corridors is convex, and can be efficiently solved. To assess the effectiveness of the proposed method, we examine its performance in a synthetic case study and subsequently evaluate its applicability in a real-world scenario from the KITTI dataset.**

Code: `https://github.com/jonarriza96/corrgen`
Video: `https://youtu.be/MvC7bPodXz8`

I. Introduction

Path-parametric methods have gained popularity in the formulation of navigation algorithms, such as high-level planners [1]–[3], reinforcement learning (RL) policies [4]–[6] or low-level model predictive controllers (MPC) [7]–[9]. The fundamental concept behind these parametric methods is to either introduce the path-parameter as an additional degree of freedom, enabling the system to regulate its progress along the path [10], [11], or to conduct a change of coordinates that project the *Euclidean states* to the *spatial states*, i.e., the progress along the path and the orthogonal distance to it [12]–[14]. These parametric formulations have proven successful for two primary reasons: firstly, they inherently capture the notion of advancement along the path, and secondly, spatial bounds manifest as convex constraints in the orthogonal terms of the spatial states.

Despite the increasing interest in path-parametric methods, existing collision-free space representation techniques proposed in the literature are not compatible with such formulations. The current solutions to this problem, often referred to as *corridor generation*, involve decomposing the free space into a collection of convex polyhedra [15]–[18]. However, applying these methods to the aforementioned parametric formulations presents two significant drawbacks: Firstly, discretizing the free space into multiple convex sets introduces the *polyhedron allocation problem*, wherein the assignment of a polyhedron to a particular state must be predetermined [19]. Essentially, this discretization disrupts the differentiability of the corridors with respect to the progress variable, making it challenging to integrate the collision-free corridors into optimization and learning routines. Secondly,

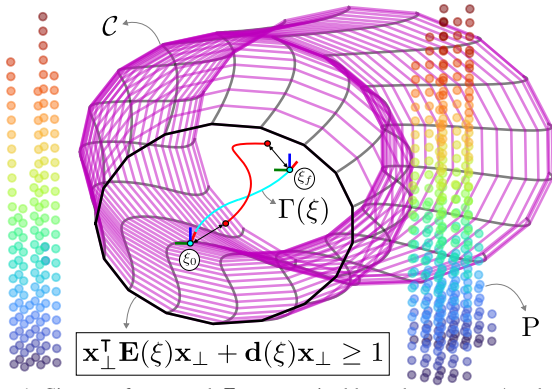


Fig. 1: Given a reference path Γ paramaterized by path-parameter ξ and a point cloud P, we present a method that computes differentiable and smooth parametric corridors $\mathcal{C}$. The reference path Γ is depicted by the blue line with a moving frame, the colored dots refer to the point cloud P and the corridor $\mathcal{C}$ is given in magenta. To generate the corridor, we optimize over a parametric ellipse $\{\mathrm{E}(\xi), \mathbf{d}(\xi) \,|\, \xi \in [\xi_0, \xi_f]\}$, where besides searching for its dimensions, we also compute the offset from the reference Γ. In other words, the center of the ellipse, shown in red, is allowed to deviate from the reference Γ. Our method can run real-time and is applicable to both, planar (2D) and spatial (3D) cases.

while representing the free space with a set of hyperplanes aligns with the *Euclidean* perspective, it neglects the ability to enforce convex constraints in the orthogonal components.

In this context, where conventional collision-free generation algorithms cannot be directly applied to parametric formulations, and where obstacle avoidance primarily involves constraining the transverse states, the development of parametric collision-free corridors has been overlooked. Consequently, the corridors generated by studies utilizing parametric formulations are often tailored to specific cases and do not generalize across different scenarios [9], [20], [21]. Due to the lack of a cross-compatible parametric corridor generation approach, most of the literature on path-parameterized methods prioritizes performance over safety, dropping the ability to impose spatial constraints, and consequently, leading to the loss of formal safety guarantees. In the MPC framework, this translates into an extremely fragile tuning process [22], heavily reliant on the sophistication and patience of an expert. Similarly, in the RL scenario this entails highly penalizing excursions out of the admissible corridors, forcing the policy to learn a trade off between optimality and safety [23], [24].

In light of these shortcomings, there is a need to properly address the generation of collision-free parametric corridors. Multiple research questions arise naturally, including how to parameterize the obstacle-free space in a differentiable way, and how to construct a lightweight program capable of computing the respective corridors in real-time.

[1]Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University of Munich (TUM)
[2]Robotics Institute (RI), Carnegie Mellon University (CMU)

Published in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, UAE, 2024.

In this paper, we address this need by presenting a method that, given a point cloud and a reference path, computes a collision-free parametric corridor around it. More specifically, the generated corridors are differentiable with respect to the path-parameter, making them compatible with existing optimization and learning pipelines; they can be larger than the corridors generated by existing tools; and they rely on a well-structured convex optimization formulation that can be solved in real-time. To the best of the authors' knowledge this is the first generic method that enables the computation of differentiable parametric corridors in real-time that are compatible with path-parametric formulations. To achieve this, our method relies on three main ingredients:

1) We represent the obstacle-free space as a parametric off-centered ellipse with Chebyshev polynomials.
2) We formulate a convex optimization problem – a Linear Program (LP) for 2D planar cases and a Semidefinite Program (SDP) for 3D spatial cases – capable of maximizing the volume of the corridor within the collision-free space.
3) We show that approximating the parametric ellipse by a diagonally-dominant matrix reduces the original SDP problem to an LP, resulting in real-time computational tractability without compromising the corridor's volumetric capacity.

The remainder of this paper is structured as follows: Section II formally introduces the parametric corridor-generation problem tackled in this work, Section III presents our solution by delving deeper into all three ingredients, Section IV shows the simulated and experimental results, and lastly, Section V presents the conclusions.

II. Problem Statement

In this section, we introduce the foundational concepts of our method, namely the reference path, the point cloud and the corridor. While the notation and problem description are presented in a generic spatial 3D form, we note that they also apply to the planar 2D case.

Let the occupied space be represented by a *point cloud* $\mathrm{P} \in \mathbb{R}^{m\times 3}$ with m points

$$\mathrm{P} = [\boldsymbol{p}_1, \dots, \boldsymbol{p}_m] \quad \text{where} \quad \boldsymbol{p}_{1,\dots,m} \in \mathbb{R}^3\,, \tag{1}$$

and assume Γ to be a *reference path* within the free space $\mathrm{P} \notin \Gamma$. Without loss of generalization, we define Γ as a parametric path with a moving frame attached to it. Its respective position and orientation are given by two functions, $\boldsymbol{\gamma} : \mathbb{R} \mapsto \mathbb{R}^3$ and $\mathrm{R}_\Gamma : \mathbb{R} \mapsto \mathbb{R}^{3\times 3} \subseteq \mathrm{SO}(3)$, dependant on path-parameter ξ:

$$\Gamma = \{\xi \in [\xi_0, \xi_f] \subseteq \mathbb{R} \mapsto \boldsymbol{\gamma}(\xi) \in \mathbb{R}^3, \mathrm{R}_\Gamma(\xi) \in \mathbb{R}^{3\times 3}\}\,. \tag{2}$$

Additionally, we define the transverse plane $\perp_\Gamma(\xi) \in \mathbb{R}^2$ as the plane orthogonal to the path, i.e. its basis is given by the second and third column of R_Γ.

The collision-free space associated to point cloud P around reference Γ is denoted as a *corridor* $\mathcal{C}_\Gamma^{\mathrm{P}}$ and encoded as an inequality on a function $\mathfrak{c}_\Gamma^{\mathrm{P}} : \mathbb{R} \mapsto \mathbb{R}^c$, such that,

$$\mathcal{C}_\Gamma^{\mathrm{P}} = \{\xi \in [\xi_0, \xi_f] \subseteq \mathbb{R}\,,\ \boldsymbol{x}_\perp \in \perp_\Gamma(\xi) \subseteq \mathbb{R}^2 \mapsto \mathfrak{c}_\Gamma^{\mathrm{P}}(\xi, \boldsymbol{x}_\perp) \leq \mathbf{0} \subseteq \mathbb{R}^c\}\,, \tag{3}$$

where c depends on the selected free space representation, and thus, will be specified in the upcoming Section III. For a visual illustration of the point cloud P in (1), the reference path Γ in (2) and the collision-free corridor $\mathcal{C}_\Gamma^{\mathrm{P}}$, please see Fig. 1.

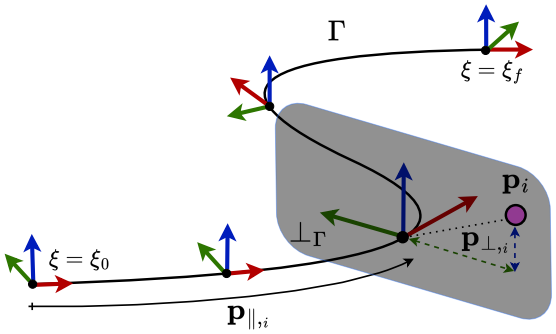


Fig. 2: Projection of a point $\boldsymbol{p}_i$, represented by the pink dot, from its Euclidean coordinates onto the reference path Γ which is parameterized by ξ and has a moving frame $\mathrm{R}_\Gamma(\xi)$ attached to it. The orthogonal plane $\perp_\Gamma$ defined by the second (green) and third (blue) components of the frame is depicted in gray. The coordinates resulting from the projection are decoupled into a tangent $\boldsymbol{p}_{\parallel,i}$ and an orthogonal $\boldsymbol{p}_{\perp,i}$ component.

Having defined the preliminaries, we can now formally state the problem addressed in this paper:

Problem 1 (Collision-free corridor generation): *Given the point cloud* P *in* (1) *and the parametric reference path* Γ *in* (2)*, compute a parametric corridor* $\mathcal{C}_\Gamma^{\mathrm{P}}$*, as defined in* (3)*, that is:*

P1.1 ***Collision-free:*** *The corridor is contained within the obstacle-free space, i.e.,* $\mathrm{P} \notin \mathcal{C}_\Gamma^{\mathrm{P}}$.

P1.2 ***Differentiable:*** *The corridor is (at least) two times differentiable and continuous with respect to the path-parameter, i.e.,* $\mathfrak{c}_\Gamma^{\mathrm{P}}(\xi)$ *is* C^2_ξ.

P1.3 ***Volume-maximizing:*** *The corridor encompasses the entire available space around the reference path* Γ *without intersecting with the point cloud* P.

Fulfilment of P1.1 and P1.3 results from the desire of maximizing navigation performance while maintaining safety. Besides that, P1.2 guarantees the existence of continuous first and second-order derivatives, and thereby, ensures the corridor's compatibility with existing numerical methods for learning and optimization. In the remainder of this manuscript, we present a method that solves this problem.

III. Solution Approach

The corridor-generation problem discussed in this paper is built upon (i) a projection of the point cloud P to the reference path Γ, (ii) a representation of the corridor as a parametric ellipse whose components are polynomials and (iii) a convex optimization problem that allows for maximizing the corridor's volume. These three elements serve as the fundamental building blocks of the proposed methodology. In

the following section, we provide a more in-depth discussion of each of these components.

A. Projecting the point cloud to the reference path

Prior to selecting a specific corridor representation for (3), a preliminary step is necessary: projecting the point cloud from Euclidean space onto the reference path Γ, i.e., $\mathrm{P} \to \mathrm{P}_\Gamma = \left[\mathrm{P}_\parallel, \mathrm{P}_\perp\right]$. This process involves decomposing the point cloud into its tangential component $\mathrm{P}_\parallel$, which corresponds to the path-parameter associated with each point in the cloud, and its orthogonal component $\mathrm{P}_\perp$, representing the projection of the distance to the reference Γ onto the transverse plane $\perp_\Gamma$. Fig. (2) provides a visual depiction of this point projection onto the reference path.

From a mathematical standpoint, the projection is conducted in two steps. First, we find the path-parameters associated to the closest point in the reference path by

$$\mathrm{P}_\parallel = \arg\min_{\xi} ||\mathrm{P} - \boldsymbol{\gamma}(\xi)||_2 \,, \tag{4a}$$

and second, we compute the orthogonal term by projecting the distance vector to the transverse plane of the reference path:

$$\mathrm{P}_\perp = \left(\mathrm{P} - \boldsymbol{\gamma}(\mathrm{P}_\parallel)\right) \mathrm{R}_\Gamma(\mathrm{P}_\parallel) \,. \tag{4b}$$

Notice that the first (tangential) term of $\mathrm{P}_\perp$ is 0, while the remaining two decompose the distance into the second and third components of the moving frame R_Γ. Besides that, the projection operation, as outlined in eqs. (4a) and (4b), is fully paralellizable, and thus, it is particularly well-suited for implementation on a GPU, enabling efficient projections regardless of the size of the point cloud.

B. Selecting a parametric cross section

Motivated by the need to formulate a corridor-generation method that is compatible with path-parameterized navigation methods and existing learning and optimization routines, we consider *parametric* corridors of the form (3). In other words, the corridors consist of a predefined cross section, whose dimensions evolve according to path-parameter ξ.

The choice of the cross section holds significant importance as it directly impacts the expressiveness and flexibility of the corridor. A cross section with fewer degrees of freedom limits the corridor's ability to accurately approximate the true free space. Hence, our objective is to select a cross section that offers maximum adaptability while ensuring differentiability[1] and computational feasibility.

The simplest cross section consists of a circle, which offers a single degree of freedom (the radius). An ellipse adds another degree of freedom by enabling different width and height parameters. Letting the ellipse rotate further increases the decision variables to 3. Additionally, allowing it to have an offset from the center raises the total degrees of freedom to 5. Ultimately, by expanding the ellipse's basis with additional polynomial terms, we can create a semialgebraic set [25], and thus, provide a high degree of expressiveness to the cross section. This concept is illustrated in Fig. 3, which compares the cross sections obtained by solving an area maximization problem. As can be seen, a higher number of degrees of freedom corresponds to a larger cross-sectional area.

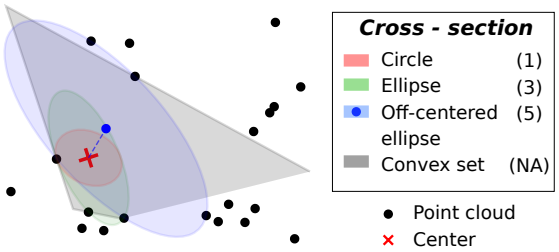


Fig. 3: Area maximization for different cross section representations centered in the red cross within the transverse point cloud $\mathrm{P}_\perp$ represented by the black dots. The depicted cross sections are: a circle with 1 degree of freedom (red), an ellipse with 3 (green) and an offset-ed ellipse with a total 5 (blue). The gray area represents the maximum convex polygon for the given point cloud encompassing the center. The offset for the off-centered ellipse is given by the blue dashed line. From this comparison, it is apparent that the higher the number the degrees of freedom, the bigger the area of the cross section.

The richness of semialgebraic sets comes at the expense of difficulties in guaranteeing the closeness and convexity of the cross section [26]. This choice makes the overall corridor generation pipeline more complicated, both from a formulation and a numerical standpoint. For this reason, we pick the next most favorable option: a rotating off-centered ellipse. Notice that, within the navigation scenario, allowing the ellipse to have an offset with respect to the reference path Γ is of major importance, since the reference might be very close to the point cloud, specially if it is obtained from search or sampling-based methods[2] [31].

The aforementioned off-centered rotating ellipse is given by

$$\mathcal{E}(\mathrm{E}, \boldsymbol{p_E}) = \{\boldsymbol{x}_\perp \in \mathbb{R}^2 \,|\, (\boldsymbol{x}_\perp - \boldsymbol{p_E})^\intercal \mathrm{E} (\boldsymbol{x}_\perp - \boldsymbol{p_E}) \leq 1\} \,, \tag{5}$$

where $\mathrm{E} \in S^2_{++}$ is a symmetric positive-definite matrix and $\boldsymbol{p_E} \in \mathbb{R}^2$ is the center of the ellipse. Once the cross section is defined as a rotating off-centered ellipse in (5), the next step is to path-parameterize it to sweep along the reference path from the initial point ξ_0 to the final point ξ_f. To achieve this, we employ polynomials of degree n that govern the evolution of both the ellipse and its center with respect to the path-parameter ξ. This allows us to fully define the parametric function of the corridor introduced in (3) as:

$$\mathfrak{c}^{\mathrm{P}}_\Gamma(\xi, \boldsymbol{x}_\perp, \boldsymbol{c_E}, \boldsymbol{c_P}) = \\ (\boldsymbol{x}_\perp - \boldsymbol{p_E}(\xi, \boldsymbol{c_P}))^\intercal \, \mathrm{E}(\xi, \boldsymbol{c_E}) \, (\boldsymbol{x}_\perp - \boldsymbol{p_E}(\xi, \boldsymbol{c_P})) - 1 \,, \tag{6}$$

where $\boldsymbol{c_E} \in \mathbb{R}^{2\times 2\times n+1}$ and $\boldsymbol{c_P} \in \mathbb{R}^{2\times n+1}$ are the coefficients of the polynomials associated with the ellipse matrix E

[1]The naive way to approach this problem would be to solve a planar convex decomposition problem at the desired transverse plane. Since the number of sides associated with a polygon obtained from convex decomposition is unknown, the resulting corridor would not comply with P1.2 in Problem 1, i.e., it would be neither differentiable nor continuous.

[2]This also provides an alternative view on the corridor-generation problem, where corridors whose cross sections are allowed to be off-centered and contain a rich-enough representation are an appealing alternative to other hierarchical and compute-heavy mapping methods, such as Euclidean Signed Distance Functions (ESDFs) [27]–[30].

and its centerpoint $\boldsymbol{p_E}$, respectively. From (6), it is apparent that the dimension of the inequality in (3) is $c = 1$.

Intuitively, increasing the degree n of the polynomial enhances the corridor's ability to adapt to variations along the reference path. The utilization of polynomials offers dual advantages: Firstly, it ensures that the resulting corridors exhibit inherent smoothness and differentiability with respect to the path-parameter ξ. Secondly, this formulation aligns with the *sums of squares* framework [32], a critical aspect for formulating the volume maximization problem discussed in the subsequent subsection.

C. Corridor volume maximization

To encapsulate the entire free space, we aim to maximize the volume of the corridor by formulating an optimization problem that finds the polynomial coefficients $\boldsymbol{c_E}$ and $\boldsymbol{c_P}$ in (6). Given that the area of the parametric ellipse $\mathcal{E}$ in (5) is $A_{\mathcal{E}} = \pi/\sqrt{\det \mathrm{E}}$, the optimization problem we seek to solve is as follows:

$$\max_{\boldsymbol{c_E}, \boldsymbol{c_P}} V_{\mathcal{E}} = \int_{\xi_0}^{\xi_f} -\det \mathrm{E}(\xi, \boldsymbol{c_E})\, d\xi \tag{7a}$$

$$\text{s.t.} \quad \mathrm{E}(\xi, \boldsymbol{c_E}) \succ 0\,, \qquad \xi \in [\xi_0, \xi_f] \tag{7b}$$

$$\mathfrak{c}^{\mathrm{P}}_{\Gamma}(\mathrm{P}_{\parallel}, \mathrm{P}_{\perp}, \boldsymbol{c_E}, \boldsymbol{c_P}) \geq 0\,, \tag{7c}$$

where (7b) ensures that the ellipse matrix $\mathrm{E}(\cdot, \boldsymbol{c_E})$ remains positive-definite throughout the entire corridor and (7c) guarantees that all the points in the cloud lie outside the corridor.

The current form of the optimization problem (7) is defined in the continuous domain, rendering it infeasible for direct implementation on a computer. Moreover, it is characterized by nonlinearity and nonconvexity. To make it computationally tractable, we reformulate the problem and introduce several modifications aimed at convexifying it. In particular, we address four key modifications:

Firstly, we discretize the continuous parts of the optimization problem, namely the cost function (7a) and the positive-definite constraint (7b), into N evaluations. This discretization is valid when N is significantly larger than the order of the polynomials, i.e., $N \gg n$. Secondly, to avoid the nonlinearities associated with the determinant of the ellipse in (7a), we approximate it with the trace of the matrix, thus yielding $\det \mathrm{E} \approx \mathrm{tr}(\mathrm{E})$. Thirdly, we prevent unbounded growth in obstacle-free areas by introducing upper bounds on the dimensions of the parametric ellipse. To this end, we modify the point cloud P by introducing a wrapper around the path Γ, which not only sets an upper limit but also allows us to exclude points located further away than the wrapper itself, thereby reducing the total number of constraints in the optimization problem[3]. As a consequence, the optimization problem is always bounded and feasible. The reduced point cloud, obtained after introducing the wrapper and removing the points outside of it, is denoted as ${}_W\mathrm{P} \in \mathbb{R}^{w\times 3}$, where w is the number of remaining points. Fourth, to address the nonlinearities in (7c), we reformulate the corridor's parametric definition as

$$\mathfrak{d}^{\mathrm{P}}_{\Gamma}(\xi, \boldsymbol{x}_{\perp}, \boldsymbol{c_E}, \boldsymbol{c_D}) = \boldsymbol{x}_{\perp}^{\intercal}\mathrm{E}(\xi, \boldsymbol{c_E})\boldsymbol{x}_{\perp} + \boldsymbol{d}(\xi, \boldsymbol{c_D})\boldsymbol{x}_{\perp} - 1\,, \tag{8}$$

where $\boldsymbol{d}(\xi, \boldsymbol{c_D}) \in \mathbb{R}^2$ is a polynomial on ξ with coefficients $\boldsymbol{c_D} \in \mathbb{R}^{2\times n+1}$. This term allows for the aforementioned offset from the reference path Γ, while removing the nonlinearities of (6).

By incorporating all four modifications, the approximation to the original problem (7) that we actually solve is given by:

$$\min_{\boldsymbol{c_E}, \boldsymbol{c_D}} -V_{\mathcal{E}} \approx \sum_{i=1}^{N} \mathrm{tr}(\mathrm{E}\,(\xi_i, \boldsymbol{c_E})) \tag{9a}$$

$$\text{s.t.} \quad \mathrm{E}(\xi_i, \boldsymbol{c_E}) \succ 0\,, \qquad i = 1, \ldots, N \tag{9b}$$

$$\mathfrak{d}^{\mathrm{P}}_{\Gamma}({}_W\mathrm{P}_{\parallel}, {}_W\mathrm{P}_{\perp}, \boldsymbol{c_E}, \boldsymbol{c_D}) \geq 0\,. \tag{9c}$$

In its present state, (9) is a convex optimization problem with $5n$ decision variables and N w constraints. More specifically, it is a Semidefinite-Program (SDP) [33], a type of problem that can be efficiently solved with off-the shelf open-source solvers [34]–[38]. It is worth highlighting that, for high polynomial degrees n, we avoid the problem becoming ill-conditioned by choosing a Chebyshev polynomial basis [39].

Since the positive-definite constraint in (9b) is a two-dimensional Linear Matrix Inequality (LMI), further relaxations allow for approximating SDP (9) into a more restricted and lightweight formulation, such as a Second-Order-Cone-Program (SOCP) or a Linear-Program (LP) [40]. Aiming to maximize computational efficiency, we choose the latter option, which replaces the positive-definite constraint on E with diagonal dominance, i.e., both diagonal terms E_{11} and E_{22} need to be bigger than the off-diagonal one E_{12}. Given that all diagonally dominant matrices are positive-definite [40], we replace constraint (9b) by $\mathrm{E}_{11} > \mathrm{E}_{12}$ and $\mathrm{E}_{22} > \mathrm{E}_{12}$, and thus, reduce the original SDP problem (9) into an LP.

D. The planar 2D case

The computation of planar corridors is a subset of the problem introduced in the previous subsection. In this case, the corridor is defined by the area enclosed between two polynomials $\boldsymbol{b}_+(\xi, \boldsymbol{c_{B_+}}) \in \mathbb{R}$ and $\boldsymbol{b}_-(\xi, \boldsymbol{c_{B_-}}) \in \mathbb{R}$, situated on the positive and negative sides of the orthogonal plane $\perp_{\Gamma}$. Here, $\boldsymbol{c_{B_+}}$ and $\boldsymbol{c_{B_-}}$ represent the corresponding coefficients. The optimization problem seeks to maximize the area, which involves tailoring problem (9) as follows:

$$\min_{\boldsymbol{c_{B_+}}, \boldsymbol{c_{B_-}}} -A_{\boldsymbol{b}} \approx -\sum_{i=1}^{N} \boldsymbol{b}_+\left(\xi_i, \boldsymbol{c_{B_+}}\right) + \boldsymbol{b}_-\left(\xi_i, \boldsymbol{c_{B_-}}\right) \tag{10a}$$

$$\text{s.t.} \quad \boldsymbol{b}_+\left(\xi_i, \boldsymbol{c_{B_+}}\right) > 0\,, \quad i = 1, \ldots, N \tag{10b}$$

$$\boldsymbol{b}_-\left(\xi_i, \boldsymbol{c_{B_-}}\right) < 0\,, \quad i = 1, \ldots, N \tag{10c}$$

$$\boldsymbol{b}_+\left({}_W\mathrm{P}^{+}_{\parallel}, \boldsymbol{c_{B_+}}\right) \leq {}_W\mathrm{P}^{+}_{\perp}\,, \tag{10d}$$

$$\boldsymbol{b}_-\left({}_W\mathrm{P}^{-}_{\parallel}, \boldsymbol{c_{B_-}}\right) \leq {}_W\mathrm{P}^{-}_{\perp}\,, \tag{10e}$$

[3] An alternative method is to decouple constraint (7b) into $\mathrm{E} - \underline{\lambda}\mathrm{I} \succ 0$ and $-\mathrm{E} + \bar{\lambda}\mathrm{I} \succ 0$, where $\underline{\lambda}$ and $\bar{\lambda}$ represent the maximum and minimum eigenvalues of the ellipse matrix E, respectively. These eigenvalues correspond to the maximum $\bar{l}$ and minimum $\underline{l}$ axis sizes of the ellipse, related by $\underline{\lambda} = 2/\sqrt{\bar{l}}$ and $\bar{\lambda} = 2/\sqrt{\underline{l}}$.

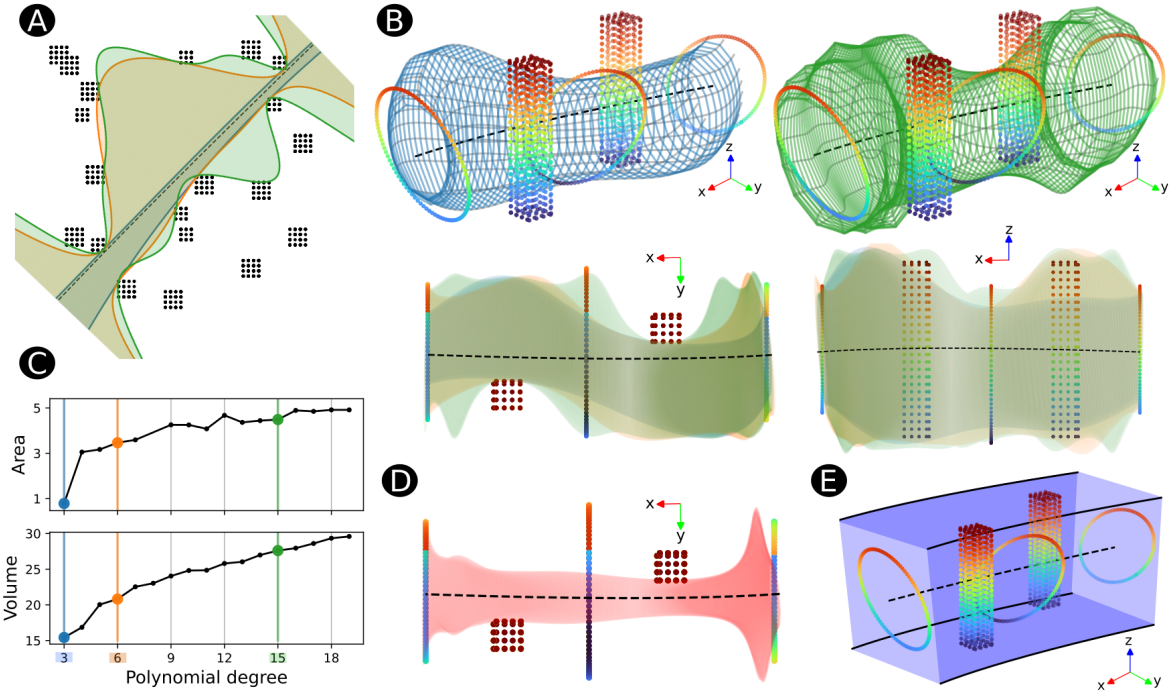


Fig. 4: 2D and 3D corridors generated by our method for different polynomial degrees: 3 in blue, 6 in orange, and 15 in green. In the planar case (A) the point cloud is depicted in black, while in the spatial case (B) it is colored according to the height. The black dashed line represents the reference path Γ. The correlation between the corridor's area – applicable to (A) – and volume – applicable to (B) – with respect to the polynomial degree is plotted in (C). As the degree increases, the respective parametric areas and volumes increase. This trend is confirmed in (A) and (B), where the expressiveness of the sampled corridors augments with the degree. Panel (D) shows the corridor associated with a parametric ellipse centered on reference Γ, making apparent the benefits of optimizing over the offset. Panel (E) depicts the wrapper that prevents the 3D corridors in (B) and (D) from becoming unbounded.

where ${}_{W}\mathrm{P}^{+}$ and ${}_{W}\mathrm{P}^{-}$ separate the point cloud in the positive and negative sides of the orthogonal plane $\perp_{\Gamma}$. The resulting problem (10) is an LP with $2n$ decision variables and $2N\,w$ constraints.

IV. Experiments

To evaluate our approach, we split the experimental analysis into two parts: First, we get a better understanding of the method's performance by focusing on two illustrative synthetic case studies. Second, we analyze its behavior in a real-world scenario obtained from the KITTI Vision Benchmark Suite [41].

Numerical implementation

In all evaluations, we compute the reference Γ by first determining a collision-free path using kinodynamic path-planning [42] and then parameterizing it with an interpolation algorithm based on Pythagorean hodograph curves [43]. It is important to note that these choices are entirely modular and independent of the presented corridor-generation method. When benchmarking against convex decomposition, we compare against [16], the implementation of which is available online[4]. Other than that, we discretize the problem into $N = 100$ evaluations and use Clarabel [36] and ProxQP [44] to solve the underlying SDPs and LPs, respectively.

[4]https://github.com/sikang/DecompUtil

A. An illustrative toy-example

To better comprehend our corridor-generation method, we examine its performance on two different 2D and 3D exemplary case studies created by randomly positioning a series of columns and rings. In particular, we compare the corridors obtained for various polynomial degrees n and we showcase the importance of optimizing over the offset with respect to reference path Γ.

The resulting solutions are illustrated in Fig. 4. In panels (A) and (B), the corridors obtained for degrees 3, 6, and 15 are presented in blue, orange, and green, respectively, for both 2D and 3D cases. From these corridors, it becomes evident that increasing the degree n enhances the corridor's expressiveness: the blue corridor is the smallest, while its capacity to adapt to the environment increases with the orange, resulting in the largest corridor for green. In panel (C), we validate this observation by demonstrating the correlation between area/volume and polynomial degree for multiple degrees. Panel (D) shows the top-down view of the corridor obtained from a centered (no offset) parametric ellipse, while panel (E) depicts the wrapper used to bound the computation of the three-dimensional corridors in (B) and (D).

The analysis of our results yields two significant conclusions: Firstly, we observe that increasing the polynomial degree leads to an expansion in the area/volume of the corridor. This augmentation is particularly notable for lower

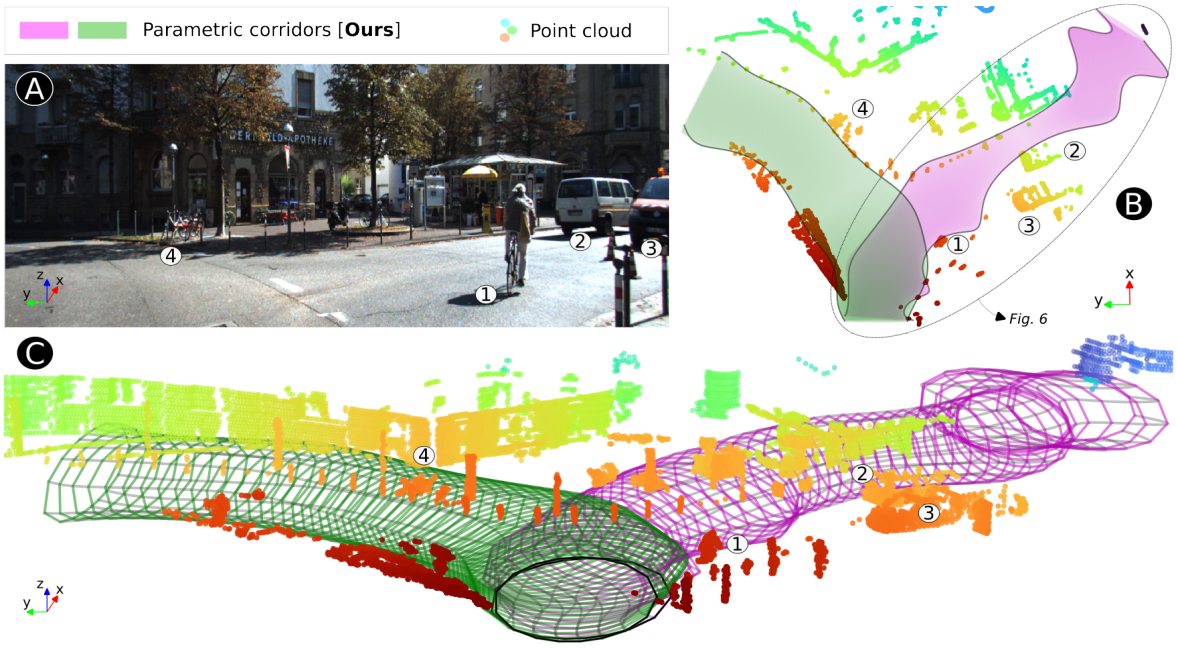


Fig. 5: Two different 3D corridors for polynomial degree $n = 9$ generated by our method in a real-world challenging driving scenario, where the road bifurcates into two narrow lanes, with other cars and cyclists. The case study is part of the KITTI Vision Benchmark Suite [41]. Panel (A) shows an RGB camera-based overview of the scene, while the respective point cloud and corridors are given in panels (B) –top view– and (C) –isometric view–. The point cloud is color-coded based on proximity to the camera's location. To facilitate the comprehension of the environment, numbered markers in white (from 1 to 4), associated with relevant features in the scene, have been added to all panels. For a detailed analysis of the corridor circled by the dashed line in Panel (B), please refer to Fig. 6.

degrees; however, it becomes less pronounced for higher degrees. Secondly, the ability to optimize over the ellipse's center point emerges as a crucial factor. As illustrated in panel (D) of Fig. 4, constraining the ellipse to be centered on the reference path Γ limits the achievable maximum volume, which is specially detrimental when the reference path is close to obstacles. However, allowing for deviations in the center point makes the corridor generation method agnostic to the underlying reference path.

B. Real-world case study

Having gained a deeper understanding on the corridor generation method, we proceed to assess its applicability to a real-world scenario taken from the KITTI dataset [41]. In particular, we focus on a roundabout intersection where a bifurcation into two lanes allows for the computation of two corridors. This intersection presents numerous challenges due to its various features, including cyclists, cars, and narrow roads.

The resulting corridors, along with the lidar-based point cloud and a raw RGB color camera view, are shown in Fig. 5. The numbers added to the image in panel (A) and the point clouds in panels (B) and (C) help to understand the landscape and locate the aforementioned features. From these visualizations it is apparent that the computed corridors encapsulate the free space, extending across the entire width of the road, and adeptly accommodating other vehicles, bicycles, and road boundaries.

To gain insight into how our method compares to the well-established convex decomposition [16], we refer to Fig. 6-A, where the top views of the corridors obtained from sequentially increasing the polynomial degree n are depicted alongside the corridors resulting from decomposing the free space into p convex sets. The results show that both methods encompass very similar spaces in all four cases. However, the hyperparameters differ significantly. In convex decomposition, the quantity and distribution of polyhedra are determined based on the number of segments within a precomputed linear path, which may compromise the corridor's volume if the reference path has few segments. In contrast, our method employs the polynomial degree n as its unique hyperparameter, while remaining agnostic to the underlying reference. Additionally, Fig. 6-A also depicts how a parametric cross section with a polynomial basis result in a continuous and smooth space representation, contrasting with the discreteness intrinsic to convex decomposition.

Besides the comparison with convex decomposition, Fig. 6-A further confirms the tendency observed in the previous synthetic case studies, namely that increasing the polynomial degree results in larger corridors. In particular, the corridor associated with $n = 24$ shows to encompass the entirety of the available space.

To reflect on the trade off between the volume gains obtained from increasing the polynomial degree n and the corresponding computational burden, we refer to Fig. 6-B. Here, we show the volumes and solve-times obtained from

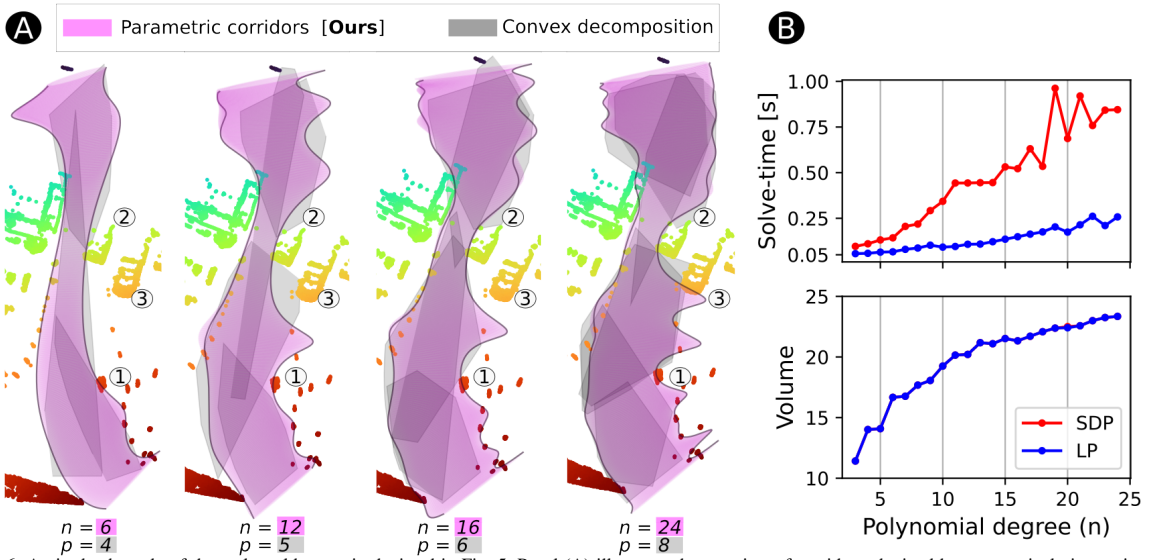


Fig. 6: An in-depth study of the real-world scenario depicted in Fig. 5. Panel (A) illustrates the top view of corridors obtained by progressively increasing the polynomial degree n, alongside the corridors resulting from decomposing the free space into p convex sets, depicted in gray. Panel (B) shows the evolution of corridor's volume and computation times with respect to the polynomial degree for both SDP and LP problems.

computing the pink corridor in Fig. 5 for various polynomial degrees, ranging from $n = 3$ up to $n = 25$, for both the original SDP and the approximated LP problems. From these results our conclusions are twofold: First, similar to the observations in the previous case studies, the increase in volume is significant for lower degrees and it becomes marginal for higher ones. Second, the approximated LP yields exactly the same corridor volumes as the original SDP, while showing major computational speedups.

We attribute this to two reasons: On the one hand, the cost function (9a) maximizes the trace, practically implying that the matrix of the resulting corridors are diagonally dominant. On the other hand, the freedom to optimize over the offset with respect to the reference Γ overcomes the limitations inherited from requiring matrix E to be diagonally dominant, resulting in corridors capable of encapsulating the available space, while maintaining diagonally dominant matrices.

These results suggest that our method is capable of computing differentiable, continuous and smooth parametric corridors at approximately 5 to 20 Hz, rendering the suggested method real-time deployable.

V. Conclusions

In this work, we proposed a corridor-generation method that allows the collision-free space to be represented in a continuous and differentiable manner. For this purpose, we predefine the cross section of the corridor as an off-centered ellipse whose components are parameterized by Chebyshev polynomials. Seeking a complete representation of the obstacle-free space, we show that the corridor volume maximization problem can be formulated as a convex optimization problem over the polynomial's coefficients. More specifically, the problem is a Linear Program (LP) in the 2D planar case and a Semidefinite Program (SDP) in the 3D spatial case. For the latter, we show that approximating the parametric ellipse by a diagonally dominant matrix also reduces it to an LP, resulting in significant computational speed improvements without compromising the corridor's volumetric capacity.

The performance of our method has been evaluated in synthetic case studies and further validated in a real-world scenario from the KITTI dataset. These results demonstrate that the corridors generated by our method achieve comparable volumetric capacity to those generated by the discrete convex decomposition method, eliminating the requirement to predefine the number of convex sets. Finally, the continuous, smooth, and differentiable nature of the obtained corridors, coupled with their real-time deployability, underscores the method's suitability for optimization and learning-based path-parametric navigation algorithms.

References

[1] R. Verschueren, N. van Duijkeren, J. Swevers, and M. Diehl, "Time-optimal motion planning for n-dof robot manipulators using a path-parametric system reformulation," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 2092–2097.

[2] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2017.

[3] J. Arrizabalaga and M. Ryll, "Sctomp: Spatially constrained time-optimal motion planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 4827–4834.

[4] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[5] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[6] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[7] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[8] T. Oelerich, F. Beck, C. Hartl-Nesic, and A. Kugi, "Boundmpc: Cartesian trajectory planning with error bounds based on model predictive control in the joint space," *arXiv preprint arXiv:2401.05057*, 2024.

[9] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4044–4050.

[10] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 6137–6142.

[11] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2015.

[12] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[13] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 477–482.

[14] J. Arrizabalaga and M. Ryll, "Spatial motion planning with pythagorean hodograph curves," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2047–2053.

[15] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2015, pp. 109–124.

[16] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[17] X. Zhong, Y. Wu, D. Wang, Q. Wang, C. Xu, and F. Gao, "Generating large convex polytopes directly on point clouds," *arXiv preprint arXiv:2010.08744*, 2020.

[18] C. Toumieh and A. Lambert, "Voxel-grid based convex decomposition of 3d space for safe corridor generation," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 4, p. 87, 2022.

[19] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[20] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 324–14 329, 2020.

[21] A. Romero, S. Govil, G. Yilmaz, Y. Song, and D. Scaramuzza, "Weighted maximum likelihood for controller tuning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1334–1341.

[22] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022.

[23] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[24] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[25] M. Shiota, *Geometry of subanalytic and semialgebraic sets*. Springer Science & Business Media, 2012, vol. 150.

[26] J. B. Lasserre, "Convexity in semialgebraic geometry and polynomial optimization," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1995–2014, 2009.

[27] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.

[28] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4423–4430.

[29] Y. Pan, Y. Kompis, L. Bartolomei, R. Mascaro, C. Stachniss, and M. Chli, "Voxfield: Non-projective signed distance fields for online planning and 3d reconstruction," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 5331–5338.

[30] J. Wang, T. Zhang, Q. Zhang, C. Zeng, J. Yu, C. Xu, L. Xu, and F. Gao, "Implicit swept volume sdf: Enabling continuous collision-free trajectory generation for arbitrary shapes," *arXiv preprint arXiv:2405.00362*, 2024.

[31] S. LaValle, "Planning algorithms," *Cambridge University Press google schola*, vol. 2, pp. 3671–3678, 2006.

[32] P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

[33] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[34] M. Yamashita, K. Fujisawa, and M. Kojima, "Implementation and evaluation of sdpa 6.0 (semidefinite programming algorithm 6.0)," *Optimization Methods and Software*, vol. 18, no. 4, pp. 491–505, 2003.

[35] M. Andersen, J. Dahl, and L. Vandenberghe, "Cvxopt: Convex optimization," *Astrophysics Source Code Library*, pp. ascl–2008, 2020.

[36] P. Goulart, "Interior-point conic optimization with clarabel.jl," https://oxfordcontrol.github.io/ClarabelDocs/stable/, accessed: 2024-02-29.

[37] B. O'Donoghue, "Operator splitting for a homogeneous embedding of the linear complementarity problem," *SIAM Journal on Optimization*, vol. 31, pp. 1999–2023, August 2021.

[38] M. Garstka, M. Cannon, and P. Goulart, "Cosmo: A conic operator splitting method for convex conic problems," *Journal of Optimization Theory and Applications*, vol. 190, no. 3, pp. 779–810, 2021.

[39] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials*. CRC press, 2002.

[40] A. A. Ahmadi and A. Majumdar, "Dsos and sdsos optimization: more tractable alternatives to sum of squares and semidefinite optimization," *SIAM Journal on Applied Algebra and Geometry*, vol. 3, no. 2, pp. 193–230, 2019.

[41] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[42] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5054–5061.

[43] Z. Šír and B. Jüttler, "c^2 hermite interpolation by pythagorean hodograph space curves," *Mathematics of Computation*, vol. 76, no. 259, pp. 1373–1391, 2007.

[44] A. Bambade, F. Schramm, S. El Kazdadi, S. Caron, A. Taylor, and J. Carpentier, "Proxqp: an efficient and versatile quadratic programming solver for real-time robotics applications and beyond," 2023.

Part III

Applications to Robot Autonomy

Following the introduction of egocentric methods and the formulation of the key components that enable their generalization to autonomous systems, we now turn to assessing these findings. This validation is conducted in two stages: first, in Chapter 7, we apply them to a diverse set of real-world problems; second, in Chapter 8, we consolidate the theoretical insights into a unified framework that standardizes the design of egocentric decision-making algorithms.

7 Egocentric Motion Planning and Control

In the previous part – Chapters 4 to 6 – we thoroughly explored the foundational elements necessary for streamlining the design of decision-making algorithms in autonomous systems. This included the core aspects of representing motion from a local frame perspective, understanding how the local frame can be parameterized, and efficiently modeling the surrounding environment or search space. These concepts were presented primarily in a theoretical manner, with little to no practical applications. As such, it is understandable if the reader finds it challenging to envision how these findings translate to real-world problems. Therefore, in this chapter, we aim to address the following question:

> *How can the findings presented in Chapters 4, 5, and 6 be applied to real-world problems?*

We approach this question in three stages, each corresponding to a different level of the decision-making hierarchy. This progression is deliberate: since the previous chapters were largely theoretical, we begin with applications that are still theory-driven, such as the design of low-level control algorithms. We then move toward more applied domains, including motion planning, and ultimately culminate in a fully application-oriented case study on slosh-free transportation.

This comprehensive range of applications – spanning from theory to practice, and encompassing multiple layers of autonomous system design – highlights the versatility, applicability, and relevance of the findings presented thus far.

7.1 Path-Parametric Control via Egocentricity

The most evident advantage of egocentric methods lies in their explicit incorporation of the notion of progress, denoted as $\sigma^* = \xi(t)$ in Chapter 3. As discussed in Section 3.3, the way this notion is embedded into the decision-making process serves as a key differentiator among egocentric approaches.

As outlined in ??, these methods are unified by the premise that *reference following* offers a more robust and flexible alternative to traditional *reference tracking*. Rather than adhering to a predefined, time-dependent reference trajectory, reference-following methods exploit velocity along the reference as an additional degree of freedom. This paradigm shift overcomes fundamental limitations of reference-tracking [23] and has since evolved into a well-established framework that has received significant attention in the literature. For comprehensive reviews, see [25, 63].

Despite their success, most existing path-following techniques primarily address translational motion, often neglecting rotational dynamics. However, in many practical applications—such as autonomous vehicles and robotic manipulators—simultaneous control of both position and orientation, collectively known as *pose*, is essential.

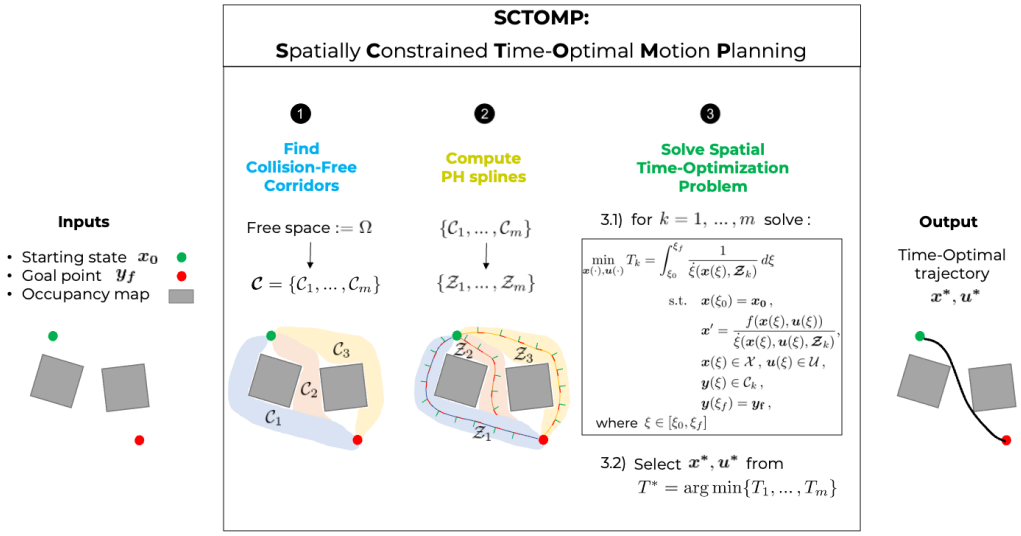


Figure 7.1 In the paper attached to Section 7.4.2 we present SCTOMP: a three-stage time-optimal motion planning method that solely requires an occupancy map of the environment, as well as the initial and goal locations: first it finds collision-free navigation corridors, second it computes an obstacle-free Pythagorean Hodograph parametric spline associated to each corridor, and third, it solves a spatially reformulated minimum-time optimization problem at each of the corridors. From the obtained trajectories, the one with the lowest navigation time is selected.

To address this limitation, we consider a geometric reference consisting of a desired path accompanied by a moving coordinate frame. We define *pose-following* as a generalization of path-following in which the system's position and orientation are both aligned with the reference. For pose representation, we adopt *unit dual quaternions*, which offer complete coverage of SE(3), favorable global stability properties, and a compact representation that efficiently encapsulates pose in a single mathematical object.

By combining the principles of reference following with the advantages of unit dual quaternions, the accompanying paper introduces a pose-following control strategy for rigid body dynamics. This formulation leverages the inherent benefits of unit dual quaternions: they are singularity-free, computationally efficient, compact, and naturally support logarithmic mappings via their underlying Lie group structure. These properties enable a seamless extension of PD-like feedback control from pose-tracking to pose-following. Consequently, the flexibility of path-following is extended to full six-degree-of-freedom motion, encompassing both translation and rotation. To the best of our knowledge, this is the first work to explicitly target pose-following in both longitudinal and angular coordinates. For details, please refer to Section 7.4.1.

7.2 Motion Planning in Constrained Environments

In the previous section, we demonstrated how egocentric components can be leveraged to design path-parametric control formulations. In this section, we ascend the autonomy hierarchy, shifting our focus from low-level control to motion planning. Unlike the pose-following control formulation discussed earlier—where the objective was to derive a control

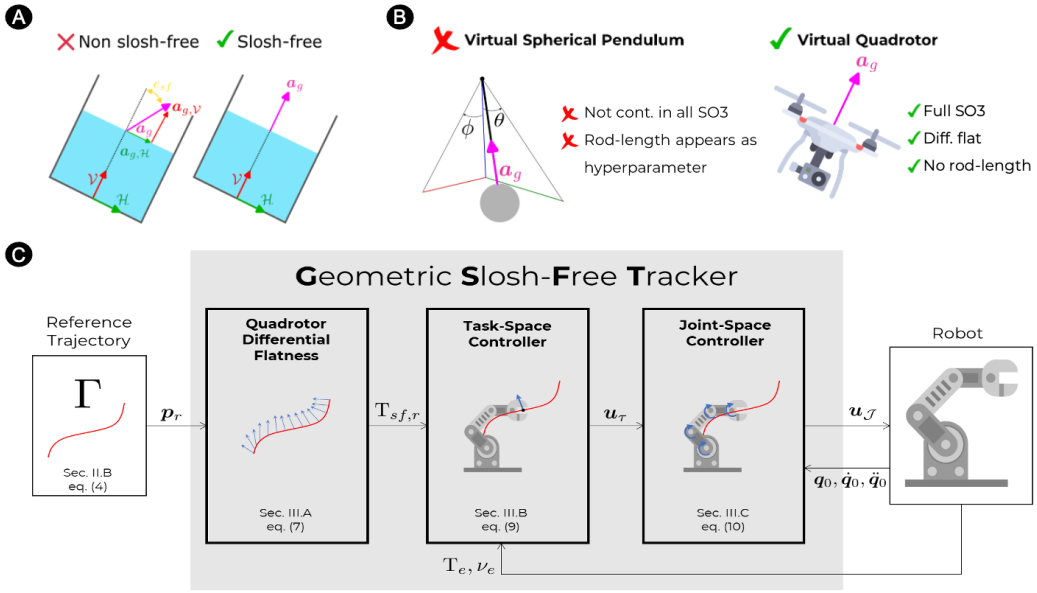


Figure 7.2 An overview of the geometric slosh-free tracking method for robotic manipulators presented in Section 7.4.3. *Panel A:* Planar diagram illustrating the slosh-free condition and its equivalence to the quadrotor model, motivating the use of a virtual quadrotor to emulate container motion. *Panel B:* Advantages of using a virtual quadrotor over a virtual pendulum model, including full SO(3) continuity, differential flatness, and the absence of a rod-length parameter. *Panel C:* Block diagram of the proposed slosh-free tracking pipeline.

law to guide the system along a predefined reference—our focus here is on computing the reference trajectory itself. In doing so, we continue to embrace the egocentric perspective, highlighting the relevance of the contributions presented in Part II.

Specifically, we introduce the **S**patially **C**onstrained **T**ime-**O**ptimal **M**otion **P**lanner (**SCTOMP**): an *offline* motion planning framework capable of computing *dynamically feasible*, *collision-free*, and *time-optimal* trajectories, given only a target location and a geometric description of the obstacle-free environment.

This is achieved by integrating several key elements: the topological search strategy from [64], the spatial transformation introduced in Section 3.3.2, the generalized spatial projection detailed in Chapter 4, and the moving frame computations presented in Chapter 5. Together, these components reformulate the time-optimal control problem into a finite-horizon spatial problem subject to convex spatial constraints.

An overview of the method and its key components is provided in Fig. 7.1. The corresponding paper is included at the end of this chapter in Section 7.4.2.

7.3 Geometric Slosh-Free Tracking

In this third section, we adopt a more systems-oriented perspective. Rather than focusing on a specific layer of the autonomy stack—such as low-level control or motion planning—we develop a method tailored to a concrete application: the transportation of liquids using robotic manipulators.

Central to this approach is the insight that if the acceleration of a motion remains orthogonal to the surface of the liquid, the motion is inherently slosh-free, thereby preventing spillage. Building on this principle, we emulate the end-effector motion of the robotic manipulator using a virtual quadrotor, which naturally satisfies the slosh-free condition by construction.

Unlike state-of-the-art techniques that model sloshing dynamics using a virtual spherical pendulum, the virtual quadrotor provides a complete representation across the full rotational envelope. This makes it well-suited for a broader range of rotational maneuvers. Moreover, the quadrotor is a differentially flat system, allowing the trajectory planning problem to be reduced from 13 dimensions to just 3. This property significantly simplifies the synthesis of slosh-free trajectories.

The proposed method harmonizes well with the egocentric framework presented in Part II, particularly with the reference generation strategies discussed in Chapter 5.

An overview of the method is shown in Fig. 7.2, and the corresponding paper is included at the end of this chapter in Section 7.4.3.

7.4 Enclosed Publications

The papers corresponding to the methods discussed in Sections 7.1, 7.2, and 7.3 are provided at the end of this chapter:

- The paper related to Section 7.1 is included in Section 7.4.1 and cited in [6]. The code implementing the content and replicating the experiments is available at `https://github.com/jonarriza96/pfdq` and the recorded version of the talk given at the conference can be viewed at `https://youtu.be/TQig2j90Ijc`.
- The paper for Section 7.2 can be found in Section 7.4.2 and cited in [5]. A short advertising video is available at `https://youtu.be/zGExvnUEfOY` and the talk given at the conference can be watched at `https://youtu.be/pOV3pGGcIqo`.
- The paper corresponding to Section 7.3 is located in Section 7.4.3 and cited in [7]. The implementation code, with the respective simulated experiments is available at `https://github.com/jonarriza96/gsft`, a short advertising video is available at `https://youtu.be/4kitqYVS9n8`, and the talk given at the conference is available at `https://youtu.be/IgoGLl_lRSw?si=MoOVsCLlmHVh_G1y`.

Pose-Following with Dual Quaternions

Jon Arrizabalaga and Markus Ryll

published in

IEEE Conference on Decision and Control (CDC)

2023

Contribution This work addresses the problem of pose-following, a generalization of path-following in which the objective is to steer both the position and orientation of a system along a path that is equipped with a moving reference frame. This formulation offers a compelling balance: it enables full-body motion control while leveraging the additional freedom to autonomously regulate progress along the path. To this end, we extend the well-established dual quaternion-based pose-tracking framework by developing a globally asymptotically stable control law for pose-following. This contribution emerged from discussions following the presentation of the work in Chapter 4, and I was the primary contributor throughout all phases of the project, including problem formulation, mathematical derivations, and the design and execution of simulated experiments.

Pose-Following with Dual Quaternions

Jon Arrizabalaga[1] and Markus Ryll[1,2]

***Abstract*— This work focuses on pose-following, a variant of path-following in which the goal is to steer the system's position and attitude along a path with a moving frame attached to it. Full body motion control, while accounting for the additional freedom to self-regulate the progress along the path is an appealing trade-off. Towards this end, we extend the well-established dual quaternion based pose-tracking method into a pose-following control law. Specifically, we derive the equations of motion for the full pose error between the geometric reference and the rigid body in the form of a dual quaternion and dual twist, and subsequently, formulate an almost globally asymptotically stable control law. The global attractivity of the presented approach is validated in a spatial example, while its benefits over pose-tracking are showcased through a planar case-study.**

Code: `https://github.com/jonarriza96/pfdq`
Video: `https://youtu.be/TQig2j90Ijc`

I. INTRODUCTION

Simultaneous attitude and position – *pose* – control of a rigid body in three-dimensional space is fundamental to many applications, such as for autonomous vehicles, spacecrafts or robotic manipulators.

The simplest approach to address the pose control problem is decoupling it into two separate subproblems [1], [2]. On the one hand, a *position controller* drives the translational motions, and on the other hand, an *attitude controller* regulates the rotational behavior. This separation relates to the de facto representation of the rigid body dynamics, in which the translational and angular motions are expressed separately (as planar and spatial examples, see eq. 7 in [3] and eq. 1 in [4]). However, such partitioning poses a challenge to effectively control the interdependence between the rotational and translational dynamics.

An alternative to this decoupling is representing the system dynamics globally on the configuration manifold of the special Euclidean group $\mathrm{SE}(3)$. Doing so allows for leveraging the group structure to first avoid singularities and second extend proportional derivative (PD) feedback controllers, for the pose-tracking problem [5]. Control methods inspired by these findings have shown very promising results within a plethora of robotic platforms, such as quadrotors [6], robotic manipulators [7], walking robots [8] and spacecrafts [9].

To represent a rigid body in $\mathrm{SE}(3)$, it is customary to combine a three-dimensional vector of the Cartesian coordinates with either a rotation matrix – resulting in a homogeneous transformation matrix (HTM) – or a unit quaternion. A less common choice are *unit dual quaternions*.

[1]Autonomous Aerial Systems, School of Engineering and Design, Technical University of Munich, Germany. E-mail: `jon.arrizabalaga@tum.de` and `markus.ryll@tum.de`
[2]Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University of Munich

Unit quaternion parameterization can be summarized into five advantages: First, when compared to HTMs, unit dual quaternions offer a more compact representation of $\mathrm{SE}(3)$, as it requires only eight parameters (against twelve) to describe the motions of a rigid body – rotations and translations – in a singularity-free and global manner [10]. Second, dual quaternion multiplications are computationally more efficient than HTM multiplications [7]. Third, the utilization of unit dual quaternions is comparatively simpler than the use of Cartesian coordinates and unit quaternions. This can be attributed to the fact that a series of rigid movements can be expressed as a sequence of dual quaternion multiplications, whereas in the other case, the computation of rotation and position is performed independently (for further details, refer to [7], [11]). Fourth, unit dual quaternions yield two closed-loop equilibrium points associated to a quaternion's double coverage of $\mathrm{SO}(3)$, both of which represent the identity rotation matrix, while rotation matrices generate a minimum of four closed-loop equilibrium points, with only one of them relating to the identity [9]. Fifth, in contrast to alternative techniques within $\mathrm{SE}(3)$, such as [6], unit dual quaternions exclusively rely on a single error function, rather than the need for two separate functions addressing position and attitude errors.

In light of these features, unit dual quaternions have been applied across a wide range of disciplines, including but not limited to inertial navigation [12], state estimation [13], inverse kinematics [14], computer graphics [15] and computer vision [16].

In the context of pose control, akin to [5], the authors in [17] and [18] broadened PD-alike feedback controllers to encompass the Lie group of unit dual quaternions using its logarithmic mapping. This resulted in a globally exponentially stable kinematic control law for pose regulation or tracking. These outcomes were subsequently expanded in [19] to also account for rigid body dynamics. Since these findings, the unit dual quaternion-based pose-tracking problem has received considerable attention in the literature. To name a few, the need for linear and angular velocity feedback was dropped in [11], a backstepping control technique to account for robustness was proposed in [20], adaptive control allowed for simultaneous pose-tracking and parameter identification in [9], formation flying was addressed in [21], and optimal control variants in the form of linear quadratic regulator (LQR) and Model Predictive Control (MPC) were formulated in [22] and [23], respectively.

Despite the achievements, all these methods exclusively focus on *pose-tracking*, i.e., they track a time-varying position and attitude reference. However, not all problems fit in such a description. For a more intuitive understanding of this concept we use an illustrative example from [24]:

Published in IEEE Conference on Decision and Control (CDC), Singapore, 2023.

When aiming for precise steering of a robot tool along a geometric reference, the primary concern is to minimize the deviation between the reference and the tool, while the velocity to move along the reference is of secondary interest and can be modified to enhance accuracy. In other words, the problem is not centered on tracking a pre-defined time-varying reference, but rather on leveraging the velocity to traverse the reference as an additional degree of freedom. Such control problems are denoted as *path-following*. Its appealing properties for a wide range of applications, accompanied by the fact that it is agnostic to the fundamental limitations of reference-tracking [25], account for the significant attention path-following has received in literature. A detailed description of existing approaches can be found in [24], [26]. Among those, most of the existing path-following methods omit the rotational dynamics and focus on path convergence of the translational dynamics.

To close this gap, we assume that the geometric reference consists of a desired path with a moving coordinate frame associated to it, and we define *pose-following* as a generalization of path-following in which the goal is to steer the system's position and attitude[1] along the reference. This begs the question of how to formulate such a pose-following method.

To answer this question, in this paper we derive a unit dual quaternion-based pose-following control approach for rigid body dynamics. To do so, we take advantage of the previously mentioned benefits of unit dual quaternions, namely singularity-free, compactness, computational efficiency and the logarithmic mapping associated with the Lie group, allowing us to extend the PD-alike feedback control from pose-tracking to pose-following. As a result, the freedom and versatility of path-following is augmented to full body motions, i.e., translations, as well as rotations. To the best of our knowledge, this is the first work that explicitly attempts to follow upon both the longitudinal and angular coordinates.

More specifically, the presented scheme consists of the following contributions:

1) We derive the equations of motion for the full pose error between the geometric reference and the rigid body in the form of a dual quaternion and dual twist.
2) We extend the original control law to account for nonlinearities that arise from introducing auxiliary states associated with pose-following. Besides that, we design the additional degree of freedom either to ensure convergence to a desired velocity profile or as feedback. Therefore, the progress along the reference is versatile in the sense that it either accepts any velocity profile or can be utilized to incite a desired behavior around the geometric reference.
3) We prove that the presented control law is almost globally asymptotically stable[2]. When doing so, we take special care of the two equilibria problem, by introducing a switch that guarantees convergence to the closest equilibrium point.

The remainder of this paper is structured as follows: Section II introduces the pose-following problem. Section III presents the solution proposed in this paper, by revisiting the unit dual quaternion-based algebra, transforming the error dynamics into dual quaternion and dual twist form, deriving the control law and conducting a stability analysis. Experimental results are shown in Section IV before Section V presents the conclusions.

Notation: We will use $\dot{(\cdot)} = \frac{\mathrm{d}(\cdot)}{\mathrm{d}t}$ for time derivatives and $\mathring{(\cdot)} = \frac{\mathrm{d}(\cdot)}{\mathrm{d}\theta}$ for differentiating over pose-parameter θ. We denote three-dimensional vectors in bold $\boldsymbol{v}$, dual numbers as $a + \epsilon \mathrm{b}$ and dual quaternions as $\hat{q}$. We define $\hat{I}$ as $[1,0,0,0] + \epsilon[0,0,0,0]$.

II. THE POSE-FOLLOWING PROBLEM

Classical path-following approaches steer a system's position, while leaving the attitude unattended. In this work, we tighten the original path-following problem by focusing on full rigid body motions, i.e, position and attitude following.

A. Rigid body dynamics

The three-dimensional rigid body dynamics in the body frame are given by

$$\ddot{\boldsymbol{p}}^b(t) = \boldsymbol{f}^b(t) m^{-1}\,, \tag{1a}$$

$$\dot{\boldsymbol{\omega}}^b(t) = \mathrm{J}^{-1}\left(\boldsymbol{\tau}^b(t) - \boldsymbol{\omega}^b(t) \times \mathrm{J}\,\boldsymbol{\omega}^b(t)\right)\,, \tag{1b}$$

where $\{\boldsymbol{p}^b, \boldsymbol{\omega}^b, \boldsymbol{f}^b, \boldsymbol{\tau}^b\} \in \mathbb{R}^3$ refer to the rigid body's position, angular velocity, control forces and control torques, while $m \in \mathbb{R}$ and $J \in \mathbb{R}^{3\times3}$ are the mass and inertia matrix. From now onward, since frame superscripts remain constant, they will be dropped. By taking position $\boldsymbol{p}$, longitudinal velocity $\dot{\boldsymbol{p}}$, attitude $q \in \mathrm{SO}(3)$ and angular velocity $\boldsymbol{\omega}$ as states $\boldsymbol{x}(t) = [\boldsymbol{p}(t), \dot{\boldsymbol{p}}(t), q(t), \boldsymbol{\omega}(t)]$ with forces $\boldsymbol{f}$ and torques $\boldsymbol{\tau}$ as inputs $\boldsymbol{u}(t) = [\boldsymbol{f}(t), \boldsymbol{\tau}(t)]$, and introducing the respective kinematic relationships, the dynamics in (1) can be written in the standard form:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))\,. \tag{2}$$

B. Geometric reference representation

Let Γ refer to a geometric reference and be defined as a path with a moving frame attached to it. Its respective desired position and attitude are given by two functions, $\boldsymbol{p}_d : \mathbb{R} \mapsto \mathbb{R}^3$ and $q_d : \mathbb{R} \mapsto \mathrm{SO}(3)$, that depend on pose-parameter θ and are at least $\mathcal{C}^2$:

$$\Gamma = \{\theta \in [\theta_0, \theta_f] \subseteq \mathbb{R} \mapsto \boldsymbol{p}_d(\theta) \in \mathbb{R}^3, q_d(\theta) \in \mathrm{SO}(3)\} \tag{3}$$

It should be noted that the C^2 requirement for q_d enables the calculation of the desired angular velocity $\boldsymbol{\omega}_d(\theta) : \mathbb{R} \mapsto \mathbb{R}^3$ from its kinematic equations.

[1] In contrast to [26], [27], where the rotational convergence is reduced to the heading $\psi \in \mathbb{R}$ and the remaining two euler angles are left unattended, we seek *true attitude convergence*, i.e. $q \in \mathrm{SO}(3)$.

[2] "almost" globally asymptotically stability is the optimal performance achievable by a continuous controller for rotational motion, owing to the fact that the group of rotation matrices SO(3) constitutes a compact manifold [28].

C. Problem statement

To incorporate the additional freedom inherited from path-following, we augment the rigid body dynamics in (2) by adding the pose-parameter $\theta(t)$ and its first time derivative $\dot{\theta}(t)$ as virtual states and assign the second time derivative $\ddot{\theta}(t)$ as a virtual input. The resulting system is denoted as

$$\dot{\boldsymbol{x}}_\Gamma(t) = f_\Gamma(\boldsymbol{x}_\Gamma(t), \boldsymbol{u}_\Gamma(t)), \tag{4}$$

where $\boldsymbol{x}_\Gamma(t) = \left[\boldsymbol{p}(t), \dot{\boldsymbol{p}}(t), q(t), \boldsymbol{\omega}(t), \theta(t), \dot{\theta}(t)\right]$ and $\boldsymbol{u}_\Gamma(t) = \left[\boldsymbol{f}(t), \boldsymbol{\tau}(t), \ddot{\theta}(t)\right]$. The augmented system f_Γ contains two additional equations of motion that relate to the integration chain of the pose-parameter $\theta(t)$, implying that the virtual input $\ddot{\theta}(t)$ is associated to its acceleration. Consequently, the time evolution $\theta(t)$, and thereby the pose reference $\{\boldsymbol{p}_d(\theta(t)), q_d(\theta(t))\}$, are controlled via the virtual input $\ddot{\theta}(t)$. This leads to the definition of the *pose-following error* as

$$\boldsymbol{e}_\Gamma(t) = \triangle\left[\{\boldsymbol{p}(t), q(t)\}, \{\boldsymbol{p}_d(\theta(t)), q_d(\theta(t))\}\right], \tag{5}$$

where $\triangle : \{\mathbb{R}^3, \mathrm{SO}(3)\} \times \{\mathbb{R}^3, \mathrm{SO}(3)\} \mapsto \mathbb{R}$ is a function that outputs the deviation between the rigid body's pose and reference pose, and will only be 0 if both are equal, i.e., $\triangle[a, b] = 0 \iff a = b$. Due to the structure of $\mathrm{SE}(3)$, this function is dependent on the control design approach, and thus, will be defined in the upcoming Section III. For the remainder of this work, we address the following problem:

Problem 1 (Pose-Following): *Given the geometric reference* Γ *in* (3) *and the augmented rigid body dynamics in* (4)*, formulate a controller* $\boldsymbol{u}_\Gamma(t) = \left[\boldsymbol{f}(t), \boldsymbol{\tau}(t), \ddot{\theta}(t)\right]$ *that fulfills:*

- P1.1 ***Pose convergence:*** *The pose-following error vanishes asymptotically* $\lim_{t\to\infty} \boldsymbol{e}_\Gamma(t) = 0$.
- P1.2 ***Convergence on pose-parameter:*** *The system converges to the end of the geometric reference* $\lim_{t\to\infty} \theta_f - \theta(t) = 0$.

For specific applications, it might be of interest to traverse the reference according to a desired velocity profile $\theta_{vd}(\theta(t))$. For example, inspection and manufacturing processes might require a lower traverse velocity at critical sections of the geometric reference. For this reason, in contrast to [24], instead of directly depending on time, our velocity profile θ_{vd} depends on the pose-parameter $\theta(t)$. Notice that this problem differs from reference tracking, for further details refer to Sec II. in [24]. Remaining consistent with the existing literature, we denote this problem as *pose-following with velocity assignment*.

Problem 2 (Pose-Following with velocity assignment): *Given the geometric reference* Γ *in* (3) *and the augmented rigid body dynamics in* (4)*, formulate a controller* $\boldsymbol{u}_\Gamma(t) = \left[\boldsymbol{f}(t), \boldsymbol{\tau}(t), \ddot{\theta}(t)\right]$ *that fulfills:*

- P2.1 ***Pose convergence:*** *P1.1 from Problem 1.*
- P2.2 ***Velocity convergence:*** *The velocity of the pose-parameter converges to a desired velocity profile* $\lim_{t\to\infty} \dot{\theta}(t) - \theta_{vd}(\theta(t)) = 0$.

III. SOLUTION APPROACH

A. Mathematical preliminaries

For the sake of making this paper self-contained we briefly recall the dual quaternion algebra. In doing so, we follow the notation and content of [19], which pioneered the use of unit dual quaternion-based pose-tracking for rigid body dynamics. To begin, we define the foundational concepts of the quaternion and the dual number, which serve as the building blocks of the dual quaternion. For more comprehensive information on these concepts, we refer the reader to [10], [18], [29].

1) Quaternions: Quaternions extend the notion of a complex number to the four-dimensional space $\mathbb{R}^4$ and can be expressed as $q = a + b\,i + c\,j + d\,k = [s, \boldsymbol{v}]$, where s is the *scalar part*, $\boldsymbol{v} \in \mathbb{R}^3$ is the *vector part* and $\{i, j, k\}$ is the standard $\mathbb{R}^4$ basis, i.e., $i^2 = j^2 = k^2 = -1$ and $ij = k$, $jk = i$, $ki = j$. When a three-dimensional vector is expressed as a quaternion with a zero scalar component, the term *vector quaternion* is used. Furthermore, quaternions fulfilling $a^2 + b^2 + c^2 + d^2 = 1$ are named *unit quaternions* and allow to describe rotations, i.e., a rotation around a unit axis $\boldsymbol{n}$ by an angle of $|\phi| < 2\pi$ can be expressed as a unit quaternion in the form of $q = [\cos(\phi/2), \sin(\phi/2)\boldsymbol{n}]$. Additionally, unit quaternions constitute a Lie group $\mathcal{Q}_u$ over multiplication, and its logarithmic map is $\ln \hat{q} = \phi/2$ with $\phi = [0, |\phi|\boldsymbol{n}]$. Limiting to $\phi \in [0, 2\pi)$ and defining v as the Lie algebra of $\mathcal{Q}_u$, i.e., all logarithmic mappings of unit quaternions, the adjoint transformation is $\mathrm{Ad}_q V = q \circ V \circ q^-1 = q \circ V \circ q^*$, where '$\circ$' is the quaternion multiplication and $V \in v$.

2) Dual numbers and vectors: Dual numbers are defined as $\hat{a} = a + \epsilon b$ with $\epsilon^2 = 0$, $\epsilon \neq 0 - \epsilon$ is nilpotent –, and $\{a, b\} \in \mathbb{R}$. In this case a and b are denoted as the *real part* and *dual part*, respectively. Dual vectors are a generalization of dual numbers, where both the real and dual part are three-dimensional vectors such that $\hat{\boldsymbol{a}} = \boldsymbol{a} + \epsilon\boldsymbol{b}$, where $\{\boldsymbol{a}, \boldsymbol{b}\} \in \mathbb{R}^3$.

3) Dual quaternions: A dual quaternion features dual components instead of its regular ones, i.e., $\hat{q} = [\hat{s}, \hat{\boldsymbol{v}}]$, where $\hat{s}$ is a dual number and $\hat{\boldsymbol{v}}$ is a dual vector. Following the introduced notation, a dual quaternion with a vanishing scalar part is named as a *dual vector quaternion*. We define the dot product of two dual vector quaternions $\hat{v} = \left[\hat{0}, \hat{\boldsymbol{v}}\right]$ with $\hat{\boldsymbol{v}} = \boldsymbol{v}_r + \epsilon\boldsymbol{v}_d$ and $\hat{k} = \left[\hat{0}, \hat{\boldsymbol{k}}\right]$ with $\hat{\boldsymbol{k}} = \boldsymbol{k}_r + \epsilon\boldsymbol{k}_d = [k_{r1}, k_{r2}, k_{r3}] + \epsilon\,[k_{d1}, k_{d2}, k_{d3}]$ as

$$\hat{k} \odot \hat{v} = \left[\hat{0}, K_r\boldsymbol{v}_r\right] + \epsilon\left[\hat{0}, K_d\boldsymbol{v}_d\right]$$

with $K_r = \mathrm{diag}(k_{r1}, k_{r2}, k_{r3})$ and $K_d = \mathrm{diag}(k_{d1}, k_{d2}, k_{d3})$. Alternatively, a dual quaternion can also be expressed as $\hat{q} = q_d + \epsilon q_r$, where q_d and q_r are quaternions. To operate with dual quaternions, we introduce the following operations:

$$\hat{q}_1 + \hat{q}_2 = [\hat{s}_1 + \hat{s}_2, \hat{\boldsymbol{v}}_1 + \hat{\boldsymbol{v}}_2] = (q_{r1} + q_{r2}) + \epsilon\,(q_{d1} + q_{d2}),$$
$$\lambda\hat{q} = [\lambda\hat{s}, \lambda\hat{\boldsymbol{v}}] = \lambda q_r + \epsilon\lambda q_d,$$
$$\hat{q}_1 \circ \hat{q}_2 = \left[\hat{s}_1\hat{s}_2 - \hat{\boldsymbol{v}}_1^T \odot \hat{\boldsymbol{v}}_2, \hat{s}_1\hat{\boldsymbol{v}}_2 + \hat{s}_2\hat{\boldsymbol{v}}_1 + \hat{\boldsymbol{v}}_1 \times \hat{\boldsymbol{v}}_2\right] = $$
$$= q_{r1} \circ q_{r_2} + \epsilon\,(q_{r1} \circ q_{d2} + q_{d1} \circ q_{r2}),$$

where $\hat{q}_1$ and $\hat{q}_2$ are dual quaternions, $\lambda \in \mathbb{R}$ and the operator '$\circ$' is the (dual) quaternion multiplication, which

is associative and distributive but not commutative. Other relevant properties of the dual quaternion are the conjugate $\hat{q}^* = [\hat{s}, -\hat{\boldsymbol{v}}]$ and the multiplicative inverse $\hat{q}^{-1} = (1/\hat{q} \circ \hat{q}^*) \circ \hat{q}^*$. It follows that dual quaternions that meet the condition $\hat{q} \circ \hat{q}^* = \hat{I}$ also satisfy $\hat{q}^{-1} = \hat{q}^*$. In such instances, the dual quaternion in question is referred to as a *unit dual quaternion*.

Just as unit quaternions permit the representation of rotations, unit dual quaternions provide a means of describing three-dimensional transformations that encompass both translation and rotation. Specifically, a transformation consisting of a translation vector $\boldsymbol{p}$ and a rotation quaternion q corresponds to a screw motion, which entails a translation along axis $\boldsymbol{n}$ by a distance d, and a rotation of angle $|\phi|$. Such a transformation is expressed as a dual quaternion in the following form:

$$\hat{q} = \left[\cos\left(\hat{\phi}/2\right), \sin\left(\hat{\phi}/2\right)\hat{\boldsymbol{n}}\right] = q + \epsilon/2\, p \circ q\,, \quad (6)$$

where $\hat{\boldsymbol{n}}$ is the dual screw axis and $\hat{\phi} = |\phi| + \epsilon d$ is the dual screw angle. Similar to unit quaternions, unit dual quaternions form a Lie group $\mathcal{DQ}_u$ with respect to the dual quaternion multiplication and its logarithmic map is also a dual quaternion given by

$$\ln \hat{q} = 1/2\,(\phi + \epsilon\, p)\ , \quad (7)$$

where $\phi = [0,\, |\phi|\, \boldsymbol{n}]$. In a similar way to unit quaternions, naming $\hat{v}$ as the Lie algebra for $\mathcal{DQ}_u$, the adjoint transformation for the dual quaternions is $\mathrm{Ad}_{\hat{q}}\hat{V} = \hat{q} \circ \hat{V} \circ \hat{q}^{-}1 = \hat{q} \circ \hat{V} \circ \hat{q}^*$, where $\hat{V} \in \hat{v}$. For additional information regarding the Lie group of unit dual quaternions, please refer to [18].

B. Unit dual quaternion dynamics

In this subsection we aim to transform the rigid body dynamics in (2) to a unit dual quaternions representation. For this purpose, in a similar way to [19], we start by derivating (6) in time, which results in the following kinematic equations:

$$\dot{\hat{q}}(t) = \frac{1}{2}\hat{\omega}(t) \circ \hat{q}(t)\,, \quad (8a)$$

$$\hat{\omega}(t) = [0, \boldsymbol{\omega}(t)] + \epsilon\,[0, \dot{\boldsymbol{p}}(t) + \boldsymbol{p}(t) \times \boldsymbol{\omega}(t)] \quad (8b)$$

where $\hat{\omega}(t)$ is the dual twist. Taking its time derivative, we get

$$\dot{\hat{\omega}}(t) = \dot{\boldsymbol{\omega}}(t) + \epsilon\left(\ddot{\boldsymbol{p}}(t) + \dot{\boldsymbol{p}}(t) \times \boldsymbol{\omega}(t) + \boldsymbol{p}(t) \times \dot{\boldsymbol{\omega}}(t)\right)\ , \quad (8c)$$

and combining it with the rigid body dynamics in (1), leads to

$$\begin{aligned}\dot{\hat{\omega}}(t) &= \left(\boldsymbol{a} + \mathrm{J}^{-1}\boldsymbol{\tau}\right) + \epsilon\left(\boldsymbol{f}/m + \dot{\boldsymbol{p}} \times \boldsymbol{\omega} + \boldsymbol{p} \times \left(\boldsymbol{a} + \mathrm{J}^{-1}\boldsymbol{\tau}\right)\right) \\ &= \underbrace{\boldsymbol{a} + \epsilon\left(\boldsymbol{p} \times \boldsymbol{a} + \dot{\boldsymbol{p}} \times \boldsymbol{\omega}\right)}_{\hat{F}} + \underbrace{\mathrm{J}^{-1}\boldsymbol{\tau} + \epsilon\left(\boldsymbol{f}/m + \boldsymbol{p} \times \mathrm{J}^{-1}\boldsymbol{\tau}\right)}_{\hat{U}} \\ &= \hat{F}(t) + \hat{U}(t) \end{aligned} \quad (8d)$$

with $\boldsymbol{a} = \mathrm{J}^{-1}\boldsymbol{\omega} \times \mathrm{J}\,\boldsymbol{\omega}$. For readability, in the first two lines of (8d) dependencies on time $(\cdot)(t)$ have been omitted. Notice that $\hat{F}(t)$ is fully defined by the rigid body's state $\boldsymbol{x}(t)$, while the force and torque control inputs $\boldsymbol{u}(t)$ only appear in $\hat{U}(t)$.

Model 1 (Unit dual quaternion-based dynamics): *Letting eq.* (6) *describe a screw motion for translation* $\boldsymbol{p}(t)$ *and rotation* $q(t)$*, and defining the linear and angular velocity as in eq.* (2)*, then the unit dual quaternion-based rigid body dynamics are*

$$\dot{\hat{q}}(t) = \frac{1}{2}\hat{\omega}(t) \circ \hat{q}(t)\,, \quad (9a)$$

$$\hat{\omega}(t) = \boldsymbol{\omega}(t) + \epsilon\left(\dot{\boldsymbol{p}}(t) + \boldsymbol{p}(t) \times \boldsymbol{\omega}(t)\right)\ , \quad (9b)$$

$$\dot{\hat{\omega}}(t) = \hat{F}(t) + \hat{U}(t) \quad (9c)$$

where $\hat{F}(t)$ *and* $\hat{U}(t)$ *are given in* (8d)*.*

C. Unit dual quaternion error dynamics

Drawing upon the dynamics obtained in the previous subsection, we proceed to derive the unit dual quaternion-based error dynamics for the pose-following problem. By applying the dual quaternion and twist definitions in eqs. (6) and (8b), it is possible to transform the geometric reference Γ in (3) into a desired dual quaternion and a desired dual twist:

$$\hat{q}_d(\theta) = q_d(\theta) + \epsilon/2\, p_d(\theta) \circ q_d(\theta)\,, \quad (10a)$$

$$\hat{\omega}_d(\theta) = [0, \boldsymbol{\omega}_d(\theta)] + \epsilon\,[0, \mathring{\boldsymbol{p}}_d(\theta) + \boldsymbol{p}_d(\theta) \times \boldsymbol{\omega}_d(\theta)]\ , \quad (10b)$$

where $\theta \in [\theta_0, \theta_f]$. Combining (10) with the kinematics in (8), the equations of motion for the desired pose are obtained:

$$\mathring{\hat{q}}_d(\theta) = \frac{1}{2}\hat{\omega}_d(\theta) \circ \hat{q}_d(\theta)\,, \quad (11a)$$

$$\begin{aligned}\mathring{\hat{\omega}}_d(\theta) = {} & [0, \mathring{\boldsymbol{\omega}}_d(\theta)] + \\ & \epsilon\,[0, \mathring{\ddot{\boldsymbol{p}}}_d(\theta) + \dot{\boldsymbol{p}}_d(\theta) \times \boldsymbol{\omega}_d(\theta) + \boldsymbol{p}_d(\theta) \times \mathring{\boldsymbol{\omega}}_d(\theta)]\ , \end{aligned} \quad (11b)$$

From (11) it is apparent that in contrast to the pose-tracking case [19], the desired pose in (10) does not evolve according to *time* t, but with respect to the *pose-parameter* θ. The error between the geometric reference and the rigid body's pose can be expressed in the form of a unit dual quaternion:

$$\hat{q}_e(t) = \hat{q}(t) \circ \hat{q}_d^*(\theta(t))\,, \quad (12)$$

Derivating the dual quaternion error (12) in time[3] leads to

$$\dot{\hat{q}}_e(t) = \dot{\hat{q}}(t) \circ \hat{q}_d^*(\theta(t)) + \dot{\theta}(t)\hat{q}(t) \circ \mathring{\hat{q}}_d^*(t)\,.$$

Combining it with (8a), (11a), (12) and the property $(\hat{q}_1 \circ \hat{q}_2)^* = \hat{q}_2^* \circ \hat{q}_1^*$ results in

$$\dot{\hat{q}}_e(t) = \frac{1}{2}\left(\hat{\omega}(t) \circ \hat{q}_e(t) + \dot{\theta}(t)\hat{q}_e(t) \circ \hat{\omega}_d^*(\theta(t))\right)\,.$$

Noticing that $\hat{q}_e \circ \hat{\omega}_d^* = (\hat{q}_e \circ \hat{\omega}_d^* \circ \hat{q}_e^*) \circ \hat{q}_e$, the equation above can be rearranged to

$$\dot{\hat{q}}_e(t) = \frac{1}{2}\left(\hat{\omega}(t) + \dot{\theta}(t)\hat{q}_e(t) \circ \hat{\omega}_d^*(\theta(t)) \circ \hat{q}_e^*(t)\right) \circ\ \hat{q}_e(t)\,,$$

which takes the same form as (8a):

$$\dot{\hat{q}}_e(t) = \frac{1}{2}\hat{\omega}_e(t) \circ\ \hat{q}_e(t)\,, \quad (13a)$$

$$\hat{\omega}_e(t) = \hat{\omega}(t) + \dot{\theta}(t)\mathrm{Ad}_{\hat{q}_e(t)}\hat{\omega}_d^*(\theta(t))\,. \quad (13b)$$

[3]Time derivations over pose-parameter θ dependent variables, such as the $\hat{q}_d(\theta(t))$ requires using the chain rule, i.e., $\frac{\mathrm{d}(\cdot)}{\mathrm{d}t} = \frac{\mathrm{d}(\cdot)}{\mathrm{d}\theta}\frac{\mathrm{d}\theta}{\mathrm{d}t} = \mathring{(\cdot)}\dot{\theta}(t)$

When compared to the pose-tracking case, the first time derivative of the pose-parameter $\dot{\theta}(t)$ appears to be multiplying the second term of the dual twist error. Similar derivations result in $p_e(t) = p(t) + \mathrm{Ad}_{q_e(t)} p_d^*(\theta(t))$ and $w_e(t) = w(t) + \dot{\theta}(t)\mathrm{Ad}_{q_e(t)}\omega_d^*(\theta(t))$. These expressions allow to ensure that the right-hand side of (13b) is equivalent to

$$\hat{\omega}_e(t) = [0, \boldsymbol{\omega}_e(t)] + \epsilon\,[0, \dot{\boldsymbol{p}}_e(t) + \boldsymbol{p}_e(t) \times \boldsymbol{\omega}_e(t)]\,. \quad (14)$$

For brevity, we omit these derivations. In case of interest, a detailed description of the respective procedure can be found in the Appendix I. Other than that, to fully define the error dynamics, we still need to compute the time derivative of the dual twist error. Towards this end, we take the time derivative of (13b) and we obtain

$$\begin{aligned}\dot{\hat{\omega}}_e(t) = \dot{\hat{\omega}}(t) + \ddot{\theta}(t)\,\mathrm{Ad}_{\hat{q}_e(t)}\hat{\omega}_d^*(\theta(t)) + \\ \dot{\theta}(t)\Big[\dot{\hat{q}}_e(t)\circ\hat{\omega}_d^*(\theta(t))\circ\hat{q}_e(t) + \\ \hat{q}_e(t)\circ\dot{\theta}(t)\mathring{\hat{\omega}}_d^*(\theta(t))\circ\hat{q}_e(t) + \\ \hat{q}_e(t)\circ\hat{\omega}_d^*(\theta(t))\circ\dot{\hat{q}}_e(t)\Big]\,.\end{aligned}$$

Model 2 (Unit dual quaternion-based error dynamics): *For a given dual quaternion state $\hat{q}(t)$ and a desired configuration $\hat{q}_d(\theta(t))$ – associated to pose-parameter $\theta(t)$ –, the dynamics of the dual quaternion error in* (12) *are*

$$\dot{\hat{q}}_e(t) = \frac{1}{2}\hat{\omega}_e(t)\circ\hat{q}_e(t)\,, \quad (15a)$$

$$\hat{\omega}_e(t) = [0, \boldsymbol{\omega}_e(t)] + \epsilon\,[0, \dot{\boldsymbol{p}}_e(t) + \boldsymbol{p}_e(t)\times\boldsymbol{\omega}_e(t)]\,, \quad (15b)$$

$$\begin{aligned}\dot{\hat{\omega}}_e(t) = \hat{F}(t) + \hat{U}(t) + \ddot{\theta}(t)\,Ad_{\hat{q}_e(t)}\hat{\omega}_d^*(\theta(t)) + \\ \dot{\theta}(t)\Big[\dot{\hat{q}}_e(t)\circ\hat{\omega}_d^*(\theta(t))\circ\hat{q}_e(t) + \\ \hat{q}_e(t)\circ\dot{\theta}(t)\mathring{\hat{\omega}}_d^*(\theta(t))\circ\hat{q}_e(t) + \\ \hat{q}_e(t)\circ\hat{\omega}_d^*(\theta(t))\circ\dot{\hat{q}}_e(t)\Big]\,,\end{aligned} \quad (15c)$$

where $\hat{q}_e(t) = \hat{q}\circ\hat{q}_d^*(\theta(t))$, $p_e(t) = p(t) + Ad_{q_e(t)}p_d^*(\theta(t))$, $w_e(t) = w(t) + \dot{\theta}(t)Ad_{q_e(t)}\omega_d^*(\theta(t))$, and $\hat{F}(t)$ and $\hat{U}(t)$ are given in (8d)*.*

The first two equations, i.e., the time derivative of the dual quaternion error and the dual twist error, show the same structure as in pose-tracking. However, differences arise in the time derivative of the dual twist error, as it involves additional terms multiplied by the first and second time derivatives of the pose-parameter $\theta(t)$.

D. Control law

Considering the error dynamics in (15), the *pose-convergence* definition in P1.1 can be reformulated as $\lim_{t\to\infty}\hat{q}_e(t) = \pm\hat{I}$ and $\lim_{t\to\infty}\hat{\omega}_e(t) = \hat{0}$. To accomplish this, in this subsection we derive a control law for the term $\hat{U}$ in equation (15c). Its design is significantly influenced by the forthcoming stability analysis. Once the control law is determined, we will be able calculate the command forces $\boldsymbol{f}$ and torques $\boldsymbol{\tau}$ from eq. (8d). In a similar way to [5] and [19], we decouple it into a feedforward (FF) and a feedback term (FB). The former eliminates nonlinearities in (15c) and the latter ensures stability.

$$\hat{U} = \hat{U}_{\mathrm{FF}} + \hat{U}_{\mathrm{FB}} \quad (16)$$

From (15c) it is apparent that the first and the last term can readily be cancelled out by the feedforward compensation. However, this does not hold true for the second adjoint term, which is multiplied by the virtual input $\ddot{\theta}(t)$. To account for this, we choose $\ddot{\theta}(t) = U_\theta(x_\Gamma(t))$, where $U_\theta(\cdot)$ is the – yet to be defined – *pose-parameter control law* dependent on the augmented state vector $x_\Gamma(t)$ in (4). This choice allows us to also include the adjoint term into the feedforward[4]:

$$\begin{aligned}\hat{U}_{\mathrm{FF}}(t, U_\theta) = -\hat{F}(t) - U_\theta(x_\Gamma(t))\,\mathrm{Ad}_{\hat{q}_e(t)}\hat{\omega}_d^*(\theta(t)) - \\ \dot{\theta}(t)\Big[\dot{\hat{q}}_e(t)\circ\hat{\omega}_d^*(\theta(t))\circ\hat{q}_e(t) + \\ \hat{q}_e(t)\circ\dot{\theta}(t)\mathring{\hat{\omega}}_d^*(\theta(t))\circ\hat{q}_e(t) + \hat{q}_e(t)\circ\hat{\omega}_d^*(\theta(t))\circ\dot{\hat{q}}_e(t)\Big]\,.\end{aligned} \quad (17)$$

Regarding the feedback term, following the pose-tracking formulation in [19], we leverage the logarithmic mapping associated to the Lie group of unit dual quaternions $\mathcal{DQ}_u$ to design a proportional derivative feedback as

$$\hat{U}_{\mathrm{FB}}(t) = -2\hat{\boldsymbol{k}}_p \odot \ln\lambda\hat{q}_e(t) - \hat{\boldsymbol{k}}_v \odot \hat{\omega}_e(t)\,, \quad (18)$$

where $\hat{\boldsymbol{k}}_p$ and $\hat{\boldsymbol{k}}_v$ are vector dual quaternion control gains and $\lambda \in \{-1, 1\}$ is a switching parameter to account for both equilibrium points $\pm\hat{I}$. This is defined as $\lambda = 1$, if $\hat{q}_{e_1}(t) >= 0$ and -1 otherwise, where $\hat{q}_{e_1}$ refers to the first component of $\hat{q}_e(t)$.

E. Stability analysis

In the present subsection, we establish the necessary conditions for control laws $\hat{U}$ and U_θ to almost global asymptotic stability[5].

Theorem 1 (Stability of pose-following): *Consider the geometric reference* (3)*, the augmented system* (4)*, the control law $\hat{U}$ in* (16) *with the feedforward and feedback terms in* (17) *and* (18)*, and suppose that the following conditions are satisfied:*

i *The dual quaternion control gains are chosen as $\hat{\boldsymbol{k}}_p > \hat{0}$ with $k_{pd1} = k_{pd2} = k_{pd3}$, i.e., equivalent terms in the dual part of $\hat{\boldsymbol{k}}_p$, and $\hat{\boldsymbol{k}}_v > \hat{0}$.*

ii *The pose-parameter control law ensures that the velocity of the pose-parameter is positive, i.e., $U_\theta(x_\Gamma(t)) \implies \dot{\theta}(t) > 0,\ \forall\,\theta \in [\theta_0, \theta_f]$.*

Then, the closed-loop control scheme defined by system (2) *and control law* (16) *solves the pose-following Problem 1.*

Proof. Starting with *pose convergence* in P1.1, since the feedforward term (17) was designed to eliminate all the nonlinearities in (15c), substituting (16) in (15c) results in

[4]Model 1 in (9) and Model 2 in (15) enable the conversion of $x_\Gamma(t)$ into a dual quaternion error and a dual twist error, as well as the conversion of these errors back into $x_\Gamma(t)$.

[5]We assume perfect and instantaneous state measurements, and thus, the presented global attractivity might be jeopardized by noises and delays that arise in practical applications. This can formally be addressed by combining the proposed method with robust control.

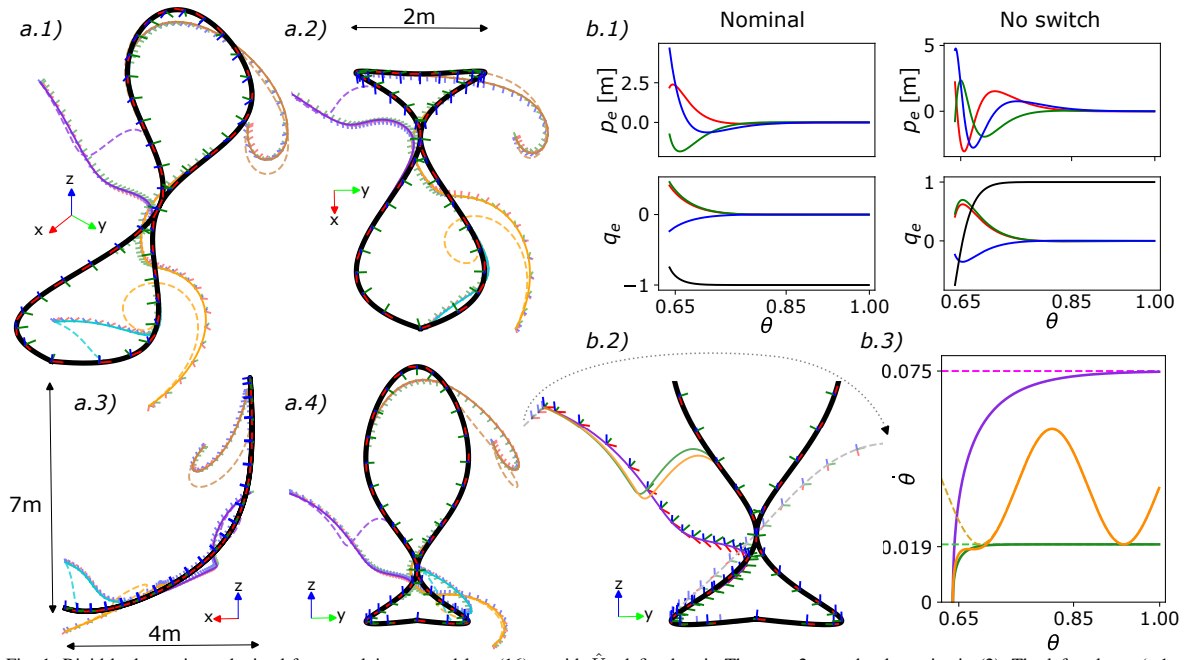


Fig. 1: Rigid body motions obtained from applying control law (16) – with $\hat{U}_\theta$ defined as in Theorem 2 – to the dynamics in (2). The left column (a.1 - a.4) shows the almost global asymptotic stability of the presented pose-following control law by starting from different initial poses (purple, orange, yellow, cyan) from different perspectives. Each starting point is evaluated according to two different constant velocity profiles. The motions associated to the fast profiles are depicted by a continuous line, as well as their orientations, while the motions related to the slow profiles are given by dashed lines. The plots at the top right (b.1), together with the gray dashed line in (b.2), showcase the consequences of deactivating the λ switch. The figure (b.3) shows that the pose-parameter's velocity (continuous line) converges to the desired velocity profile (dashed line) by evaluating three different cases (fast in purple, slow in green and sinusoidal in orange).

$\dot{\hat{\omega}}_e = \hat{U}_{\text{FB}}$. In addition, considering that (14) also remains true for pose-following, the stability analysis in [19] holds. This implies that Model 2 converges to the closest equilibrium point $\{\pm\hat{I}, \hat{0}\}$ asymptotically, which directly translates to the fulfillment of pose convergence.

Regarding *convergence on pose-parameter* in P1.2, combining the Lyapunov function $V = ||\theta(t) - \theta_f||^2$ with $\dot{\theta}(t) > 0$ – from (ii) – shows that θ_f is an asymptotically stable equilibrium point. □

Theorem 2 (Stability of pose-following with velocity assignment): *Consider the geometric reference* (3)*, the augmented system* (4)*, the control law* $\hat{U}$ *in* (16) *with the feedforward and feedback terms in* (17) *and* (18)*, and suppose that the following conditions are satisfied:*

i *The dual quaternion control gains are chosen as* $\hat{\boldsymbol{k}}_p > \hat{0}$ *with* $k_{pd1} = k_{pd2} = k_{pd3}$ *and* $\hat{\boldsymbol{k}}_v > \hat{0}$.

ii *The pose-parameter control law is given by* $U_\theta(x_\Gamma(t)) = -k_\theta \left(\dot{\theta}(t) - \theta_{vd}(\theta(t))\right) + \dot{\theta}(t)\mathring{\theta}_{vd}(\theta(t))$*, where* $k_\theta \in \mathbb{R}_{>0}$.

Then, the closed-loop control scheme defined by system (2) *and control law* (16) *solves the pose-following with velocity assignment Problem 2.*

Proof. The proof for *pose convergence* in P2.1 remains the same as P1.1 in Theorem 1. When it comes to *velocity convergence* in P2.2, the utilization of the Lyapunov function $V = ||\dot{\theta}(t) - \theta_{vd}(\theta(t))||^2$ in conjunction with $U_\theta(x_\Gamma(t))$ as given in (iii), and the recognition that $\ddot{\theta}(t) = U_\theta(x_\Gamma(t))$, indicates that the velocity of the pose-parameter asymptotically converges to the desired velocity profile. □

IV. NUMERICAL EXPERIMENTS

In order to assess the effectiveness of our methodology, we focus on two case studies. The first one focuses on the key properties of the derived control law, including its almost global asymptotic stability and its capability to converge to a predetermined velocity profile. In the second case-study, we demonstrate the advantages of the proposed pose-following control law in comparison to its predecessor, the pose-tracking control law.

Numerical implementation: For all evaluations, the parameters are kept constant as $m = 1\text{kg}$, $J = \text{diag}(0.01, 0.01, 0.01)\,\text{Kg/m}^{-2}$, $\hat{\boldsymbol{k}}_p = \hat{\boldsymbol{k}}_v = \hat{\boldsymbol{3}}$ and $k_\theta = 1$.

A. Almost global asymptotic stability on pose-following with velocity assignment

The primary focus of this study is to validate the outcome of Theorem 2: almost global asymptotic stability for pose-following with velocity assignment. When doing so, we intend to verify that, regardless of the initial state, the pose of the rigid body converges to the geometric reference. To this end, we initialize the system at four distinct poses. In addition, we also want to show that this convergence is upheld irrespective of the velocity assignment. To achieve this goal, we evaluate each starting condition according to

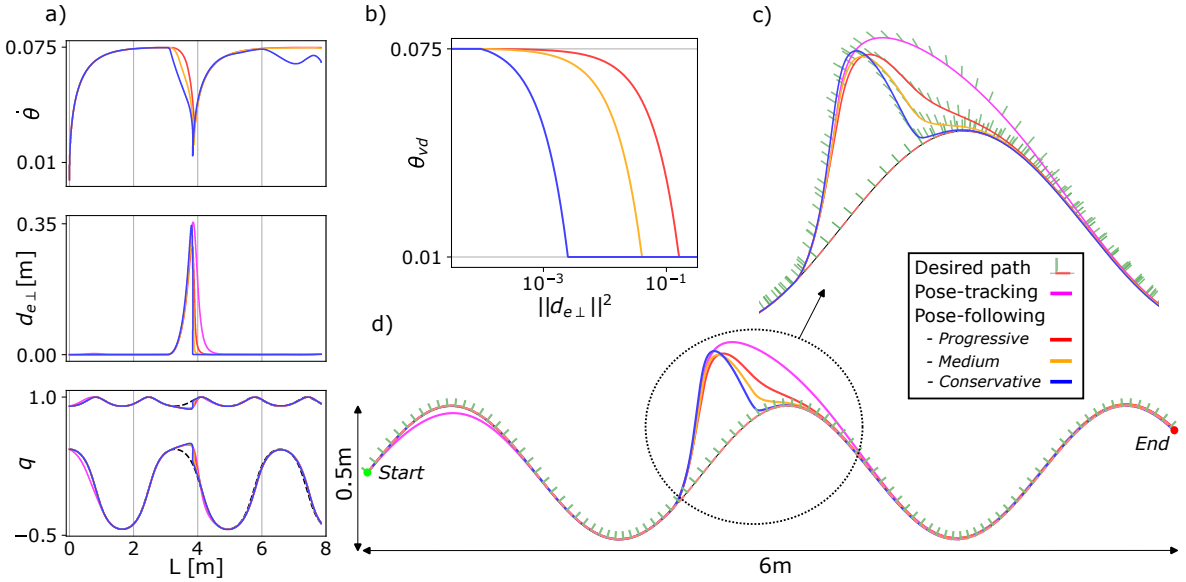


Fig. 2: A comparison between pose-tracking [19] (magenta) and three variants of the presented pose-following control approach (progressive as red, medium as orange and conservative as blue) in the presence of a disturbance. The geometric reference is given by a thin black line and its moving frame. For the rest of the motions, only the normal components are shown for clarity. a) From top to bottom, velocity of pose-parameter, transverse distance to the geometric reference and first and last components of the unit quaternion, where the dashed black line refers to the geometric reference's orientation. b) The desired velocity profile as a mapping from transverse distance to the reference. c) Zoomed comparison in the location of the disturbance and d) Overview of the case-study.

two distinct profiles, namely a slow one with $\theta_{vd,\text{slow}} = 0.019$ and a fast one with $\theta_{vd,\text{fast}} = 0.075$. As an exemplary geometric reference, we select the same three-dimensional curve as in [30] and to conform to the requirements of (3), we assign a moving frame to it. For this purpose, akin to [31], [32], we rely on *Pythagorean Hodograph curves*, allowing us to overcome the singularities and discontinuities of the well-known *Frenet-Serret* frame.

The motions resulting from applying the control law (16) – with the pose-parameter control $\hat{U}_\theta$ defined as in Theorem 2 – to the rigid body dynamics in (2) are depicted in the left side of Fig. 1. The motions respective to the runs with the faster velocity profiles are shown by a continuous line and their respective orientations, while the ones related to the slow profiles are represented by dashed lines.

These motions manifest two noteworthy characteristics. The first one being that all of them demonstrate asymptotic convergence towards the geometric reference. The second characteristic, which aligns with common intuition, is that motions corresponding to the slower velocity profiles attain convergence at an earlier stage.

For a more comprehensive analysis, we direct attention to the purple case-study, which refers to the motion located at the top-left corner of Fig. 1(a.1). A magnified view of this case is presented on the right-hand side of Fig. 1(b.2). We hereby validate that the velocity of the pose-parameter, $\dot{\theta}(t)$, achieves convergence with the desired velocity profile, θ_{vd}. To accomplish this, as demonstrated in Fig. 1(b.3), we examined the convergence in not only slow (green) and fast (purple) constant velocity profiles but also in a sinusoidal profile (orange).

Lastly, we showcase the importance of taking care of the existence of two equilibrium points $\pm\hat{I}$, which in our approach is handled by the switching term λ (18). Within the same case-study as in the previous paragraph, we show that if this switching term is deactivated, the control law only converges to $\hat{I}$ resulting in unnecessarily lengthy and large motion. This can be visualized in the position and quaternion errors, as well as in the resultant motions colored in light gray in Fig.1(b.1-2).

B. Comparison to pose-tracking

Having analyzed the most relevant properties of the presented control law, in this second case-study we compare the performance of the proposed pose-following approach against pose-tracking [19]. For this purpose, we pick a planar sinusoidal curve with a moving frame attached to it as a geometric reference. The task at hand consists of traversing the geometric reference from a zero-velocity pose. However, at the middle of the navigation a longitudinal and angular disturbance is introduced. To ensure a fair comparison, both the pose-tracking and pose-following have been tuned to ensure that the navigation time is the same if no disturbance occurs.

In this experiment the desired velocity profile function is chosen to be dependent on the distance to the geometric reference: $U_\theta(x_\Gamma(t)) = -k_\theta \left(\dot{\theta}(t) - \theta_{vd}(d_{e\perp}(x_\Gamma(t))\right)$[6] Intuitively, if the system is far away from the reference, it slows down until it is close enough to increase the speed. This mapping is regarded as a tuning parameter that the user can tailor based on system properties and task at hand. In an illustrative manner, we design three variants:

[6] $d_{e,\perp}(x_\Gamma(t))$ is the transverse distance to the geometric reference.

progressive (red), medium (orange) and conservative (blue). These velocity profiles alongside their associated motions can be visualized in Fig. 2.

When compared to pose-tracking (in magenta), two differences can be spotted. First, at the very beginning of the trajectory, the tracking method shows a small deviation from the reference. This is due to the fact that the rigid body initially is standing still and needs to catch up with the moving time-reference. In contrast the presented pose-following is aware of its initial state and progressively increases its velocity along the reference. Second, as soon as the disturbance is over, the additional degree of freedom inherited from augmenting the system allows all three variants to slow down and converge back to the geometric reference. This can clearly be visualized in the evolution of $\dot{\theta}$. As expected, the convergence rate directly correlates to how conservative the desired velocity profile mapping is. On the other hand, pose-tracking lacks this additional degree of freedom and has no choice but to catch up with the time-based reference, causing a large deviation error.

V. CONCLUSIONS

In this paper we have formulated a unit dual quaternion-based pose-following control approach for rigid body dynamics. Initially, we have derived the equations of motion for the full pose error between the rigid body and the geometric reference in the form of a dual quaternion and dual twist. Subsequently, we have extended the original control law to account for nonlinearities arising from the introduction of auxiliary states associated with pose-following and designed the additional degree of freedom either to achieve convergence to a desired velocity profile or as a feedback mechanism. When doing so, we have also established almost global asymptotic stability. Lastly, we have numerically validated our findings with two illustrative simulations.

REFERENCES

[1] O.-E. Fjellstad and T. I. Fossen, "Position and attitude tracking of auv's: a quaternion feedback approach," *IEEE Journal of Oceanic Engineering*, vol. 19, no. 4, pp. 512–518, 1994.

[2] D. T. Stansbery and J. R. Cloutier, "Position and attitude control of a spacecraft using the state-dependent riccati equation technique," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 3. IEEE, 2000, pp. 1867–1871.

[3] J. Arrizabalaga, N. van Duijkeren, M. Ryll, and R. Lange, "A caster-wheel-aware mpc-based motion planner for mobile robotics," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 613–618.

[4] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4044–4050.

[5] F. Bullo and R. M. Murray, "Proportional derivative (pd) control on the euclidean group," 1995.

[6] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[7] L. Figueredo, B. V. Adorno, J. Y. Ishihara, and G. A. Borges, "Robust kinematic control of manipulator robots using dual quaternion representation," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1949–1955.

[8] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[9] N. Filipe and P. Tsiotras, "Adaptive position and attitude-tracking controller for satellite proximity operations using dual quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 566–577, 2015.

[10] A. T. Yang, *Application of quaternion algebra and dual numbers to the analysis of spatial mechanisms*. Columbia University, 1963.

[11] N. Filipe and P. Tsiotras, "Rigid body motion tracking without linear and angular velocity feedback using dual quaternions," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 329–334.

[12] Y. Wu, X. Hu, D. Hu, T. Li, and J. Lian, "Strapdown inertial navigation system algorithms based on dual quaternions," *IEEE transactions on aerospace and electronic systems*, vol. 41, no. 1, pp. 110–132, 2005.

[13] Y. Zu, U. Lee, and R. Dai, "Distributed motion estimation of space objects using dual quaternions," in *AIAA/AAS Astrodynamics Specialist Conference*, 2014, p. 4296.

[14] D. Gan, Q. Liao, S. Wei, J. Dai, and S. Qiao, "Dual quaternion-based inverse kinematics of the general spatial 7r mechanism," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 222, no. 8, pp. 1593–1598, 2008.

[15] L. Kavan, S. Collins, C. O'Sullivan, and J. Zara, "Dual quaternions for rigid transformation blending," *Trinity College Dublin, Tech. Rep. TCD-CS-2006–46*, 2006.

[16] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[17] D.-P. Han, Q. Wei, and Z.-X. Li, "Kinematic control of free rigid bodies using dual quaternions," *Int. J. Autom. Comput.*, vol. 5, no. 3, pp. 319–324, 2008.

[18] X. Wang, D. Han, C. Yu, and Z. Zheng, "The geometric structure of unit dual quaternion with application in kinematic control," *Journal of Mathematical Analysis and Applications*, vol. 389, no. 2, pp. 1352–1364, 2012.

[19] X. Wang and C. Yu, "Unit dual quaternion-based feedback linearization tracking problem for attitude and position dynamics," *Systems & Control Letters*, vol. 62, no. 3, pp. 225–233, 2013.

[20] F. Zhang and G. Duan, "Robust integrated translation and rotation finite-time maneuver of a rigid spacecraft based on dual quaternion," in *Aiaa guidance, navigation, and control conference*, 2011, p. 6396.

[21] X. Wang, C. Yu, and Z. Lin, "A dual quaternion solution to attitude and position control for rigid-body coordination," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1162–1170, 2012.

[22] M. M. Marinho, L. Figueredo, and B. V. Adorno, "A dual quaternion linear-quadratic optimal controller for trajectory tracking," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4047–4052.

[23] U. Lee and M. Mesbahi, "Constrained autonomous precision landing via dual quaternions and model predictive control," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 292–308, 2017.

[24] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2015.

[25] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic, "Path-following for nonminimum phase systems removes performance limitations," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 234–239, 2005.

[26] N. Hung, F. Rego, J. Quintas, J. Cruz, M. Jacinto, D. Souto, A. Potes, L. Sebastiao, and A. Pascoal, "A review of path following control strategies for autonomous robotic vehicles: Theory, simulations, and experiments," *Journal of Field Robotics*, no. 3, pp. 747–779, 2023.

[27] J. Płaskonka, "Different kinematic path following controllers for a wheeled mobile robot of (2, 0) type," *Journal of Intelligent & Robotic Systems*, vol. 77, pp. 481–498, 2015.

[28] S. P. Bhat and D. S. Bernstein, "A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon," *Systems & control letters*, vol. 39, no. 1, pp. 63–70, 2000.

[29] B. Kenwright, "A beginners guide to dual-quaternions: what they are, how they work, and how to use them for 3d character hierarchies," 2012.

[30] S. Kumar and R. Gill, "Path following for quadrotors," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 2075–2081.

[31] J. Arrizabalaga and M. Ryll, "Spatial motion planning with pythagorean hodograph curves," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2047–2053.

[32] J. Arrizabalaga and M. Ryll, "SCTOMP: Spatially constrained time-optimal motion planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023.

APPENDIX I
DERIVATION OF DUAL TWIST ERROR

Let $\hat{q}_e$ and $\hat{\omega}_e$, defined in eqs. (12) and (13b), be the dual quaternion and twist errors between the pose of rigid body (1) and the parametric geometric reference (3). Further derivating the dual twist error as in (14) requires from the upcoming three lemmas:

Lemma 1: *Let q_e and $\boldsymbol{\omega}_e$ refer to the quaternion and twist errors. Then,*

$$\dot{q}_e(t) = \frac{1}{2}\boldsymbol{\omega}_e(t) \circ q_e(t)\,,$$
$$\boldsymbol{\omega}_e(t) = \omega(t) + \dot{\theta}(t) Ad_{q_e(t)}\boldsymbol{\omega}_d^*(\theta(t))\,.$$

Proof. Defining the quaternion error as $q_e(t) = q(t) \circ q_d^*(\theta(t))$, its time derivative is given by

$$\dot{q}_e(t) = \dot{q}(t) \circ q_d^*(\theta(t)) + q(t) \circ \dot{q}_d^*(\theta(t))$$

Leveraging that $\dot{q}(t) = \frac{1}{2}\boldsymbol{\omega}(t) \circ q(t)$ and $\dot{q}^*(t) = \frac{1}{2}q^*(t) \circ \boldsymbol{\omega}^*(t)$, results in

$$\begin{aligned}\dot{q}_e(t) =& \frac{1}{2}(\boldsymbol{\omega}(t) \circ q(t) \circ q_d^*(\theta(t)) + \\ & \dot{\theta}(t) q(t) \circ q_d^*(\theta(t)) \circ \boldsymbol{\omega}_d^*(\theta(t)))\,,\end{aligned}$$

and further simplifies into

$$\dot{q}_e(t) = \frac{1}{2}\left(\boldsymbol{\omega}(t) \circ q_e(t) + \dot{\theta}(t) q_e(t) \circ \boldsymbol{\omega}_d^*(\theta(t)))\right)\,,$$

which is equivalent to

$$\dot{q}_e(t) = \frac{1}{2}\underbrace{\left(\boldsymbol{\omega}(t) + \dot{\theta}(t) q_e(t) \circ \boldsymbol{\omega}_d^*(\theta(t)) \circ q_e^*(t)\right)}_{\boldsymbol{\omega}_e(t)} \circ q_e(t)\,.$$

□

Lemma 2: *Let $\boldsymbol{p}_e(t)$ refer to the position error as*

$$\boldsymbol{p}_e(t) = \boldsymbol{p}(t) + Ad_{q_e(t)}\boldsymbol{p}_d^*(\theta(t))\,,$$

then

$$\begin{aligned}\dot{\boldsymbol{p}}_e(t) =& \dot{\boldsymbol{p}}(t) + \boldsymbol{\omega}_e(t) \times Ad_{q_e(t)}\boldsymbol{p}_d^*(\theta(t)) + \\ & \dot{\theta}(t) Ad_{q_e(t)}\mathring{\boldsymbol{p}}_d^*(\theta(t))\,.\end{aligned}$$

Proof. Derivating the position error in time

$$\begin{aligned}\dot{\boldsymbol{p}}_e(t) =& \boldsymbol{p}(t) + \dot{q}_e(t) \circ \boldsymbol{p}_d^*(\theta(t)) \circ q_e^*(t) + \\ & q_e(t) \circ \dot{\theta}(t)\mathring{\boldsymbol{p}}_d^*(\theta(t)) \circ q_e^*(t) + \\ & q_e(t) \circ \boldsymbol{p}_d^*(\theta(t)) \circ \dot{q}_e^*(t)\,,\end{aligned}$$

which combined with Lemma 1 leads to

$$\begin{aligned}\dot{\boldsymbol{p}}_e(t) =& \boldsymbol{p}(t) + \frac{1}{2}\left(\boldsymbol{\omega}_e(t) \circ q_e(t) \circ \boldsymbol{p}_d^*(\theta(t)) \circ q_e^*(t) + \right. \\ & q_e(t) \circ \boldsymbol{p}_d^*(\theta(t)) \circ q_e^*(t) \circ \boldsymbol{\omega}_e^*(t)) + \\ & \left. q_e(t) \circ \dot{\theta}(t)\mathring{\boldsymbol{p}}_d^*(\theta(t)) \circ q_e^*(t)\right)\,,\end{aligned}$$

and noticing that $\boldsymbol{\omega}_e^*(t) = -\boldsymbol{\omega}_e(t)$

$$\begin{aligned}\dot{\boldsymbol{p}}_e(t) =& \boldsymbol{p}(t) + \frac{1}{2}\left(\boldsymbol{\omega}_e(t) \circ \mathrm{Ad}_{q_e(t)}\boldsymbol{p}_d^*(\theta(t)) - \right. \\ & \left. \mathrm{Ad}_{q_e(t)}\boldsymbol{p}_d^*(\theta(t)) \circ \boldsymbol{\omega}_e(t)\right) + \\ & \dot{\theta}(t)\mathrm{Ad}_{q_e(t)}\mathring{\boldsymbol{p}}_d^*(\theta(t))\,.\end{aligned}$$

Given that for vector quaternions $q_1 \circ q_2 = [-q_1 q_2,\, q_1 \times q_2]$ and noticing that $\boldsymbol{\omega}_e(t)$ is perpendicular to $\mathrm{Ad}_{q_e(t)}\boldsymbol{p}_d^*(\theta(t))$,

$$\begin{aligned}\dot{\boldsymbol{p}}_e(t) =& \boldsymbol{p}(t) + \boldsymbol{\omega}_e(t) \circ \mathrm{Ad}_{q_e(t)}\boldsymbol{p}_d^*(\theta(t)) + \\ & \dot{\theta}(t)\mathrm{Ad}_{q_e(t)}\mathring{\boldsymbol{p}}_d^*(\theta(t))\,,\end{aligned}$$

or equivalently,

$$\begin{aligned}\dot{\boldsymbol{p}}_e(t) =& \boldsymbol{p}(t) + \boldsymbol{\omega}_e(t) \times \mathrm{Ad}_{q_e(t)}\boldsymbol{p}_d^*(\theta(t)) + \\ & \dot{\theta}(t)\mathrm{Ad}_{q_e(t)}\mathring{\boldsymbol{p}}_d^*(\theta(t))\,.\end{aligned}$$

□

Lemma 3: *The following statement is true*

$$\begin{aligned}\boldsymbol{p}_e(t) \times \boldsymbol{\omega}_e(t) =& \boldsymbol{p}(t) \times \boldsymbol{\omega}(t) + \\ & \dot{\theta}(t) Ad_{q_e(t)}\boldsymbol{\omega}_d^*(\theta(t)) + \\ & Ad_{q_e(t)}\boldsymbol{\omega}_d^*(\theta(t)) \times \boldsymbol{\omega}_e(t)\end{aligned}$$

Proof. Expanding the cross product with the definitions in Lemmas 1 and 2,

$$\begin{aligned}\boldsymbol{p}_e(t) \times \boldsymbol{\omega}_e(t) =& \left[\boldsymbol{p}(t) + \mathrm{Ad}_{q_e(t)}\boldsymbol{p}_d^*(\theta(t))\right] \times \\ & \left[\boldsymbol{\omega}(t) + \dot{\theta}(t)\mathrm{Ad}_{q_e(t)}\boldsymbol{\omega}_d^*(\theta(t))\right]\,,\end{aligned}$$

which further simplifies into

$$\begin{aligned}\boldsymbol{p}_e(t) \times \boldsymbol{\omega}_e(t) =& \boldsymbol{p}(t) \times \boldsymbol{\omega}(t) + \\ & \dot{\theta}(t)\boldsymbol{p}(t) \times \mathrm{Ad}_{q_e(t)}\boldsymbol{\omega}_d^*(\theta(t)) + \\ & \mathrm{Ad}_{q_e(t)}\boldsymbol{p}_d^*(\theta(t)) \times \boldsymbol{\omega}_e(t)\,.\end{aligned}$$

□

Using these three Lemmas we state the following theorem:

Theorem 3: *%labellemma:lemma3 The dual twist error is given by*

$$\hat{\omega}_e(t) = [0, \boldsymbol{\omega}_e(t)] + \epsilon\,[0, \dot{\boldsymbol{p}}_e(t) + \boldsymbol{p}_e(t) \times \boldsymbol{\omega}_e(t)]\,.$$

Proof. The proof consists on combining the dual twist error in (13b)

$$\hat{\omega}_e(t) = \hat{\omega}(t) + \dot{\theta}(t)\mathrm{Ad}_{\hat{q}_e(t)}\hat{\omega}_d^*(\theta(t))\,.$$

with Lemmas 1, 2 and 3. First, expanding the second term follows as

$$\begin{aligned}\mathrm{Ad}_{\hat{q}_e(t)}\hat{\omega}_d^*(\theta(t)) =& \begin{bmatrix} q_e(t) \\ \frac{1}{2}\boldsymbol{p}_e(t) \circ q_e(t)\end{bmatrix} \circ \\ & \begin{bmatrix} \omega_d^*(\theta(t)) \\ (\mathring{\boldsymbol{p}}_d(\theta(t)) + \boldsymbol{p}_d(\theta(t)) \times \boldsymbol{\omega}_d(\theta(t)))^*\end{bmatrix} \circ \begin{bmatrix} q_e^*(t) \\ \frac{1}{2}q_e^*(t) \circ \boldsymbol{p}_e^*(t)\end{bmatrix}\,,\end{aligned}$$

which, after some derivations, can be further simplified into

$$\begin{aligned}&\mathrm{Ad}_{\hat{q}_e(t)}\hat{\omega}_d^*(\theta(t)) = \\ &\begin{bmatrix} \mathrm{Ad}_{\hat{q}_e(t)}\boldsymbol{\omega}_d^*(\theta(t)) \\ \boldsymbol{p}(t) \times \mathrm{Ad}_{\hat{q}_e(t)}\boldsymbol{\omega}_d^*(\theta(t)) + \mathrm{Ad}_{\hat{q}_e(t)}\mathring{\boldsymbol{p}}_d^*(\theta(t))\end{bmatrix}\,.\end{aligned}$$

Getting back to the dual twist error in (13b) and replacing its second term by the expression above, and the first one by the definition of the dual twist in (8b), leads to

$$\begin{aligned}&\hat{\omega}_e(t) = \boldsymbol{\omega}(t) + \dot{\theta}(t)\mathrm{Ad}_{\hat{q}_e(t)}\boldsymbol{\omega}_d^*(\theta(t)) + \\ &\epsilon[\dot{\boldsymbol{p}}(t) + \boldsymbol{p}(t) \times \boldsymbol{\omega}(t) + \\ &\dot{\theta}(t)\left(\boldsymbol{p}(t) \times \mathrm{Ad}_{\hat{q}_e(t)}\boldsymbol{\omega}_d^*(\theta(t)) + \mathrm{Ad}_{\hat{q}_e(t)}\mathring{\boldsymbol{p}}_d^*(\theta(t))\right)]\,.\end{aligned}$$

Combining the real part with Lemma 1 and leveraging the following statement resulting from Lemmas 2 and 3 within the dual part results in

$$\dot{\boldsymbol{p}}_e(t) + \boldsymbol{p}_e(t) \times \boldsymbol{\omega}_e(t) = \dot{\boldsymbol{p}}(t) + \boldsymbol{p}(t) \times \boldsymbol{\omega}(t) + \dot{\theta}(t) \left(\mathrm{Ad}_{q_e(t)} \hat{\boldsymbol{p}}_d^*(\theta(t)) + \boldsymbol{p}(t) \times \mathrm{Ad}_{q_e(t)} \boldsymbol{\omega}_d^*(\theta(t)) \right) ,$$

and therefore, the dual twist error can be expressed as

$$\hat{\omega}_e(t) = [0, \boldsymbol{\omega}_e(t)] + \epsilon \left[0, \dot{\boldsymbol{p}}_e(t) + \boldsymbol{p}_e(t) \times \boldsymbol{\omega}_e(t)\right] .$$

□

SCTOMP: Spatially Constrained Time-Optimal Motion Planning

Jon Arrizabalaga and Markus Ryll

published in

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

2024

Contribution This work explores spatial time-optimal motion planning, an extension of the exact time-optimal path-following problem that enables planning over a defined spatial domain rather than along a fixed path. The motivation for this development stemmed from the need for a motion planner capable of generating reference trajectories for the tunnel predictive controller presented as a case study in Chapter 3. I was responsible for the full development of this work, from identifying the core problem, through the mathematical formulation and derivations, to the design and implementation of the simulated experiments.

SCTOMP: Spatially Constrained Time-Optimal Motion Planning

Jon Arrizabalaga[1] and Markus Ryll[1,2]

***Abstract*— This work focuses on spatial time-optimal motion planning, a generalization of the exact time-optimal path following problem that allows a system to plan within a predefined space. In contrast to state-of-the-art methods, we drop the assumption of a given collision-free geometric reference. Instead, we present a three-stage motion planning method that solely relies on start and goal locations and a geometric representation of the environment to compute a time-optimal trajectory that is compliant with system dynamics and constraints. The proposed scheme first finds collision-free navigation corridors, second computes an obstacle-free Pythagorean Hodograph parametric spline along each corridor, and third, solves a spatially reformulated minimum-time optimization problem at each of these corridors. The spline obtained in the second stage is not a geometric reference, but an extension of the free space associated with its corridor, and thus, time-optimality of the solution is guaranteed. The validity of the proposed approach is demonstrated by a well-established planar example and benchmarked in a spatial system against state-of-the-art methodologies across a wide range of scenarios in highly congested environments.**

Video: `https://youtu.be/zGExvnUEfOY`

I. INTRODUCTION

Time-optimal motion planning within cluttered environments poses multiple challenges. The underlying motion planning scheme needs to compute a set of input commands that drive the system from its current state to a goal location in minimum-time, without compromising system constraints and spatial bounds. Thus, its solution implies a trading-off between time-optimality and spatial-awareness.

The de facto approach to solve this problem has been to decouple it into two stages [1]–[4]. First, the *path planning* stage determines a collision-free geometric path according to high level – task related – commands [5]. Second, the predefined path is (exactly) tracked either by *path tracking* or *path following*. The former computes a dynamically feasible timing law for traversing along the predetermined geometric path – *when* to be *where* –, while the latter introduces the timing law and the (bounded) distance to the path as control freedoms [6]–[8].

When tackling time-optimality, previous work mainly focused on the second stage. *Time-optimal path-tracking* for robotic manipulators is a long studied problem [9]–[11]. Its convexity was proven in [6] and in [12] it was extended to a wider range of systems. Enhanced numerical implementations to exploit this convexity were presented in [13], [14]. Regarding *time-optimal path-following*, [15], [16] leveraged a spatial reformulation of the system dynamics, allowing to compute minimum-time trajectories *around* the predefined path. Similarly, assuming waypoints to be predetermined, [17] made use of Complementary Progress Constraints (CPC) to compute time-optimal trajectories for quadrotors. Lastly, exploiting optimal control, nonlinear model predictive control (NMPC) methods based on the aforementioned spatial reformulation [18], [19] and contouring control [20], [21] have shown to approximate time-optimal performance in race-alike scenarios, i.e., *around* a predefined geometric path (the centerline of the track). However, the optimality of all these approaches is upper bounded by the geometric reference predetermined by the path planning stage. In other words, only if *both* the path planning and path tracking/following stages are optimal, will the resultant planned trajectory be optimal, implying that the decoupled nature of these methods jeopardizes the time-optimality of the planned trajectories.

To overcome this conceptual shortcoming, [22] presented a hierarchical sampling-based method capable of computing time-optimal trajectories to fly a quadrotor over a set of waypoints within cluttered environments. This was further extended in [23], where planning and control were simultaneously solved with a deep Reinforcement Learning (deep RL) approach. Despite the ability to roughly approximate time-optimal trajectories in congested scenarios, none of these methods can guarantee that the resultant trajectory is the true-optimal. On the one hand, the sampling-based nature of [22] renders it nondeterministic, introducing randomness into the solution. On the other hand, the additional tracking capabilities brought by the end-to-end learning approach in [23] come at the expense of 1) system-specificity – changes in the system dynamics require retraining – and 2) sub-optimal trajectories resulting from learning the trade-off between flying safe and fast.

This raises the question on how to formulate a motion planning scheme, applicable to any system and constrained environment, that guarantees to compute the time-optimal trajectory compliant with system dynamics, state/input constraints and spatial bounds. To answer this question, in this paper we present a **S**patially **C**onstrained **T**ime-**O**ptimal **M**otion **P**lanner (**SCTOMP**): an *offline* motion planning approach capable of *computing dynamically feasible, collision-free and time-optimal trajectories*, by solely relying on a goal location and a geometric representation of the obstacle-free environment.

For this purpose, we formulate a three-stage approach,

[1]Autonomous Aerial Systems, School of Engineering and Design, Technical University of Munich, Germany. E-mail: `jon.arrizabalaga@tum.de` and `markus.ryll@tum.de`

[2]Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University of Munich

Published in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, USA, 2023.

where, firstly collision-free navigation corridors are found, secondly parametric paths within all corridors are computed, and thirdly a spatially reformulated time minimization per corridor is solved. In contrast to the aforementioned decoupled approaches, the parametric paths obtained in the second stage are not a geometric reference, but an extension of the free space associated with the respective corridor, and consequently, have no effect on the optimality of the time-minimizing problem in the third stage.

The presented scheme consists of three main ingredients: 1) Through the use of a spatial reformulation presented in [24], we perform a spatial transformation of the time-based system states to path coordinates. The resultant system dynamics evolve according to a *path parameter* ξ instead of *time* t. 2) We leverage Pythagorean Hodograph curves to efficiently and analytically compute the parametric functions required by this spatial reformulation. 3) Using the first and second ingredients, the time minimization problem is reduced into a finite horizon spatial problem with convex spatial bounds.

More specifically, we make the following contributions:

1) We extend the applicability of the spatial reformulation introduced in [24], by showing how it allows to conduct a singularity-free spatial transformation of the dynamics for any arbitrary system.
2) We identify a computationally tractable methodology to compute Pythagorean Hodograph splines in a closed form, without the need for additional optimizations.
3) We derive a motion planning methodology that, given a nonlinear system and constrained environment, finds the collision-free and dynamically feasible time-optimal trajectory.

The remainder of this paper is structured as follows: Section II introduces the spatially constrained time-optimal motion planning problem. Section III presents the solution proposed in this paper, by revisiting the aforementioned spatial reformulation, its compatibility with PH curves, and performing a spatial transformation of the time minimization problem. Experimental results are shown in Section IV before Section V presents the conclusions.

Notation: We will use $\dot{(\cdot)} = \frac{\mathrm{d}(\cdot)}{\mathrm{d}t}$ for time derivatives and $(\cdot)' = \frac{\mathrm{d}(\cdot)}{\mathrm{d}\xi}$ for differentiating over path parameter ξ. For readability we will employ the abbreviation ${}_t\xi = \xi(t)$.

II. PROBLEM STATEMENT

We consider continuous time, nonlinear systems of the form

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x_0}\,, \tag{1a}$$
$$\boldsymbol{y}(t) = h(\boldsymbol{x}(t))\,, \tag{1b}$$

where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ define state and input constraints. Function $f : \mathbb{R}^{n_x} \times \mathbb{R}^m \mapsto \mathbb{R}^{n_x}$ refers to the equations of motion of an arbitrary dynamic system, while the map $h : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ defines the output of the system $\boldsymbol{y} \in \mathbb{R}^{n_y}$. Both f and h functions are assumed to be sufficiently continuously differentiable. The initial state is contained in a subset of the constrained state set, i.e., $\boldsymbol{x}_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$.

To ensure that the system remains in the obstacle free space, the geometrically constrained environment is described as

$$\Omega = \{g(h(\boldsymbol{x})) < 0\,, \quad \forall \boldsymbol{x} \in \mathcal{X}\} \in \mathbb{R}^{n_y}\,, \tag{2}$$

where we also assume the function $g : \mathbb{R}^{n_y} \mapsto \mathbb{R}^{n_y}$ to be sufficiently continuously differentiable.

Spatially Constrained Time-Optimal Motion Planning refers to the problem of planning a trajectory that steers system (1) from its initial state $\boldsymbol{x_0}$ to a goal output state $\boldsymbol{y_f} \in \mathbb{R}^{n_y}$ in minimum-time, without compromising the system dynamics in (1a), ensuring the integrity of state and input constraints – $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{u} \in \mathcal{U}$ – and guaranteeing that output (1b) remains within the free space in (2), i.e., $\boldsymbol{y} \in \Omega$. This is equivalent to solving the following optimal control problem (OCP):

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} T = \int_0^T dt \tag{3a}$$
$$\text{s.t.} \quad \boldsymbol{x}(0) = \boldsymbol{x_0}\,, \tag{3b}$$
$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \qquad t \in [0, T] \tag{3c}$$
$$\boldsymbol{x}(t) \in \mathcal{X}\,,\, \boldsymbol{u}(t) \in \mathcal{U}\,, \quad t \in [0, T] \tag{3d}$$
$$\boldsymbol{y}(t) \in \Omega\,, \qquad t \in [0, T] \tag{3e}$$
$$\boldsymbol{y}(T) = \boldsymbol{y_f}\,. \tag{3f}$$

As mentioned earlier, *decoupled* approaches separate this problem into a path planning and path tracking/following problem. The computational tractability advantages brought by these methods come at the expense of an inherited suboptimality in the planned trajectories. We seek to close this gap by formulating a method that *directly* solves the minimum-time problem (3), eliminating the need for the path planning stage, and thus leveraging the entire free space to compute the time-optimal trajectory.

III. SOLUTION APPROACH

The motion planning scheme presented in this paper leverages (i) a *spatial transformation of the system dynamics* with respect to (ii) an *arbitrary parametric curve with an associated adapted frame* to perform (iii) a *spatial reformulation of the time-minimizing problem* (3). These three ingredients are the main building blocks of the proposed methodology. In this section, we present further details on each of them.

A. SPATIAL TRANSFORMATION OF SYSTEM DYNAMICS

Let Γ refer to a geometric reference and be defined as a path with an associated adapted-frame, whose position and orientation are given by two sufficiently continuous functions, $\boldsymbol{\gamma} : \mathbb{R} \mapsto \mathbb{R}^3$ and $\mathrm{R} : \mathbb{R} \mapsto \mathbb{R}^{3\times3}$, that depend on path parameter ξ:

$$\Gamma = \{\xi \in [\xi_0, \xi_f] \subseteq \mathbb{R} \mapsto \boldsymbol{\gamma}(\xi) \in \mathbb{R}^3, \mathrm{R}(\xi) \in \mathbb{R}^{3\times3}\} \tag{4}$$

Decomposing the adapted-frame into its components $\mathrm{R}(\xi) = \{\boldsymbol{e_1}(\xi), \boldsymbol{e_2}(\xi), \boldsymbol{e_3}(\xi)\}$ allows to define its angular velocity

vector as $\boldsymbol{\omega}(\xi) = \chi_1(\xi)\boldsymbol{e_1}(\xi) + \chi_2(\xi)\boldsymbol{e_2}(\xi) + \chi_3(\xi)\boldsymbol{e_3}(\xi)$, where $\chi(\xi) = \{\chi_1(\xi), \chi_2(\xi), \chi_3(\xi)\} \in \mathbb{R}^3 \mapsto \mathbb{R}$ is also sufficiently continuous.

In [24] we demonstrated that the equations of motion of the spatial coordinates – progress along the path ξ and the orthogonal distance to it $\boldsymbol{w} = [w_1, w_2]$ – associated to a point-mass moving at velocity $\boldsymbol{v} \in \mathbb{R}^3$ with respect to path Γ are given by

$$\dot{\xi}(t) = \frac{\boldsymbol{e_1}({}_t\xi)^\intercal \boldsymbol{v}(t)}{\sigma({}_t\xi) - \chi_3({}_t\xi)w_1(t) + \chi_2({}_t\xi)w_2(t)}\,, \tag{5a}$$

$$\dot{w}_1(t) = \boldsymbol{e_2}({}_t\xi)^\intercal \boldsymbol{v}(t) + \dot{\xi}(t)\chi_1({}_t\xi)w_2(t)\,, \tag{5b}$$

$$\dot{w}_2(t) = \boldsymbol{e_3}({}_t\xi)^\intercal \boldsymbol{v}(t) - \dot{\xi}(t)\chi_1({}_t\xi)w_1(t)\,, \tag{5c}$$

where $\sigma(\cdot)$ stands for the parametric speed of path Γ. For a better understanding of the spatial states, see Fig. 1. These states, alongside their respective equations of motion (5), allow for a projection of the Euclidean coordinates into the geometric path Γ, resulting in a change of coordinates. Doing so, embeds the geometric properties *around path* Γ into the system dynamics, and thus, it has become a very popular approach among recent path following methods [16], [18], [19], [25].

Given that our solution aims to be agnostic from a geometric reference, we perform a spatial transformation of the system dynamics in (1). To do so, we leverage the chain rule as follows:

$$\boldsymbol{x}' \coloneqq \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\xi} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\xi} \tag{6}$$

Noticing that $\frac{\mathrm{d}t}{\mathrm{d}\xi} = 1/\dot{\xi}$ and assuming that $\dot{\xi} \neq 0$, (6) is simplified into

$$\boldsymbol{x}' = \frac{1}{\dot{\xi}} f(\boldsymbol{x}, \boldsymbol{u})\,, \quad \forall \dot{\xi} \neq 0\,.$$

The resultant spatially transformed equations of motion for the dynamic system (1) are

$$\boldsymbol{x}'(\xi) = \frac{f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))}{\dot{\xi}(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))}, \quad \boldsymbol{x}(\xi_0) = \boldsymbol{x_0}\,, \tag{7a}$$

$$\boldsymbol{y}(\xi) = h(\boldsymbol{x}(\xi))\,, \tag{7b}$$

where $\dot{\xi}(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))$ is obtained from (5a). Comparing the *spatially transformed* system (7) with respect to the original system (1), it becomes apparent that the dynamics evolve with respect to *path parameter* ξ instead of *time* t.

Transforming the system dynamics according to the chain rule (6) is a mature technique [6], [9]. Nevertheless, to the best of the authors' knowledge, this is the first time that it is applied to the recently derived spatial reformulation (5). When doing so, the benefits of this reformulation are inherited by the transformed system (7). As opposed to state-of-the-art Frenet-Serret based reformulations [15], [16], [25], (5) does not take any assumptions in its adapted frame, and as result, it overcomes the two major drawbacks of the Frenet-Serret frame: (i) *singularities* when the curvature vanishes and (ii) an *undesired twist* with respect to its tangent component.

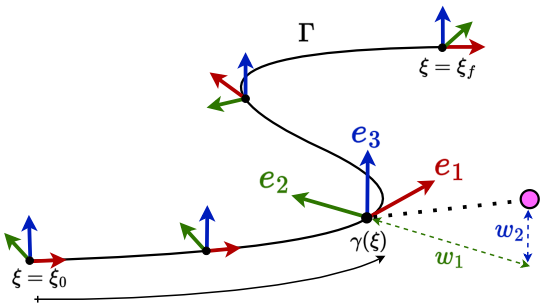


Fig. 1: Spatial projection of the three-dimensional Euclidean coordinates, represented by the pink dot, onto a geometric path Γ with an associated adapted-frame $\mathrm{R}(\xi) = \{\boldsymbol{e_1}(\xi), \boldsymbol{e_2}(\xi), \boldsymbol{e_3}(\xi)\}$. The distance to the closest point on the path $\boldsymbol{\gamma}(\xi)$ is decomposed into the transverse coordinates $\boldsymbol{w} = [w_1, w_2]$.

B. PYTHAGOREAN HODOGRAPH SPLINES

For a complete definition of the spatially transformed system (7), the *parametric functions* associated to ξ – the parametric speed $\sigma(\xi)$ and the adapted frame's rotation matrix $\mathrm{R}(\xi)$ and angular velocity $\boldsymbol{\omega}(\xi)$ – need to be expressed in closed form. To this end, we employ Pythagorean Hodograph (PH) curves.

1) Preliminaries on PH curves

PH curves are a subset of polynomial curves whose parametric speed is a polynomial of the path parameter ξ [26]. Decomposing the components of $\boldsymbol{\gamma}(\xi)$ in (4) into $x(\xi), y(\xi), z(\xi)$, the condition for a polynomial to be a PH curve is equivalent to

$$\sigma^2(\xi) = x'^2(\xi) + y'^2(\xi) + z'^2(\xi)\,, \tag{8}$$

where $\sigma(\xi)$ is a polynomial. As proven in [27], every term in (8) can be expressed in terms of a *quaternion polynomial* $\boldsymbol{Z}(\xi) = u(\xi) + v(\xi)\mathbf{i} + g(\xi)\mathbf{j} + h(\xi)\mathbf{k}$ [1], whose components $u(\xi), v(\xi), g(\xi), h(\xi)$ are also polynomial functions of the path parameter ξ. Consequently, the parametric speed can be reformulated as

$$\sigma(\xi) = u^2(\xi) + v^2(\xi) + g^2(\xi) + h^2(\xi)\,. \tag{9}$$

Another notable benefit of PH curves is that they inherit a continuous adapted frame, denoted as *Euler Rodrigues Frame* (ERF), which also exclusively depends on its quaternion polynomial [28]. Its respective rotation matrix is given by

$$\mathrm{R}(\xi) = \frac{[\boldsymbol{Z}(\xi)\mathbf{i}\boldsymbol{Z}^*(\xi), \boldsymbol{Z}(\xi)\mathbf{j}\boldsymbol{Z}^*(\xi), \boldsymbol{Z}(\xi)\mathbf{k}\boldsymbol{Z}^*(\xi)]}{|\boldsymbol{Z}(\xi)|^2}\,, \tag{10}$$

where $(\cdot)^*$ refers to the quaternion's conjugate. Finally, considering that the components of its angular velocity are expressed as

$$\chi(\xi) = \{\boldsymbol{e_2}'(\xi)\,\boldsymbol{e_3}(\xi), \boldsymbol{e_3}'(\xi)\,\boldsymbol{e_1}(\xi), \boldsymbol{e_1}'(\xi)\,\boldsymbol{e_2}(\xi)\}\,, \tag{11}$$

the adapted frame's angular velocity $\boldsymbol{\omega}(\xi)$ is likewise exclusively reliant on the quaternion polynomial.

[1] $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ refers to the $\mathbb{R}^4$ standard basis

From (9), (10) and (11), it can be stated that all *parametric functions* – $\sigma(\xi), \mathrm{R}(\xi), \boldsymbol{\omega}(\xi)$ – solely depend on the quaternion polynomial $\boldsymbol{Z}(\xi)$, and consequently, they are fully defined by the coefficients of the underlying polynomial functions $u(\xi), v(\xi), g(\xi), h(\xi)$. For convenience we will refer to these as *PH coefficients*, $\boldsymbol{\zeta} \in \mathbb{R}^{n_\zeta}$. Finally, notice that (9) entails that a quaternion polynomial $\boldsymbol{Z}(\xi)$ of degree n corresponds to a curve $\boldsymbol{\gamma}(\xi)$ of degree $2n+1$, while relying just on $4(n+1)$ PH coefficients. This implies that all three parametric functions, and hence the spatial reformulation underpinning system (7), are effectively encoded into a handful of coefficients.

2) Efficiently computing collision-free PH splines

To ensure planning capabilities within highly non-convex and complex environments, we concatenate multiple PH curves into a *PH spline*. As mentioned before, the PH spline is not a geometric reference, but a parametric path that encodes the geometric properties of the environment into the spatially transformed system dynamics (7). Thus, as proven in the upcoming Section IV-A, it has no influence on the time-optimality of the computed trajectory. However, when choosing the PH spline the requirements are twofold: First, the parametric functions – $\sigma(\xi), \mathrm{R}(\xi), \boldsymbol{\omega}(\xi)$ – need to remain sufficiently often continuously differentiable, and second, it must be located within the obstacle-free space.

For this purpose, [24] reduces the amount of free PH coefficients according to the continuity conditions, and subsequently, tailors the shape of the PH spline based on a desired criterion by solving an optimization problem on the remaining coefficients. To ensure the integrity of the spatial bounds it performs a convex decomposition of the free space Ω and constraints the control points of each segment of the PH spline to be encompassed inside the associated convex set. These control points can readily be obtained from a function that takes the starting location of the spline and the PH coefficients as inputs [26]. However, given that the coefficients relate to the hodograph, these constraints, as well as the cost function, are highly nonlinear and nonconvex, resulting in a large and numerically involved nonlinear program.

To alleviate this burden, rather than executing a convex decomposition and directly computing a PH spline, the proposed methodology in the **first-stage** identifies all navigation corridors within the free-space $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\} \subseteq \Omega$, and in the **second-stage** locates an arbitrary curve, enclosed within each of these corridors $\lambda_{1,\dots,m}$, and transforms each of these curves into a PH spline whose coefficients are $\boldsymbol{\mathcal{Z}}_{1,\dots,m}$.

The presented hierarchical methodology exhibits modularity in that the finding of the corridors, computation of the curves and the successive conversion algorithm are entirely decoupled. Specifically, the former can be carried out effectively by employing state-of-the-art planning algorithms [29]–[31], whereas for the latter, we expand upon the C^2 hermite interpolation algorithm outlined in [32] to C^4. Given that the conversion algorithm is presented as a closed-form solution, it entails no computational cost, and the burden of solving the aforementioned intractable nonlinear program is diminished to identifying a collision-free curve.

C. TIME MINIMIZATION: A SPATIAL APPROACH

In order to encompass the entirety of the free space Ω and ensure time-optimality of the calculated trajectory, the **third-stage** of SCTOMP involves solving a time-minimizing OCP for all corridors $k = 1, , ..., m$ and selecting the one with the minimum navigation time.

At a given corridor k, the PH coefficients $\boldsymbol{\mathcal{Z}}_k$ associated to the PH spline computed in the second-stage explicitly describe the parametric coefficients, and therefore, allow to rewrite the equation of motion of the spatially transformed system (7a) as

$$\boldsymbol{x}'(\xi, \boldsymbol{\mathcal{Z}}_k) = \frac{f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))}{\dot{\xi}(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi), \boldsymbol{\mathcal{Z}}_k)}\,. \tag{12}$$

This equation represents the spatially transformed dynamics model of a system whose temporal dynamics are given by f, with respect to a PH spline defined by PH coefficients $\boldsymbol{\mathcal{Z}}_k$. In contrast to (7), the system in (12) accounts for a method to compute the underlying parametric functions – based on PH splines –, and thus, it is fully defined.

Revisiting the fact that $\frac{\mathrm{d}t}{\mathrm{d}\xi} = 1/\dot{\xi}$ and leveraging the spatially transformed system dynamics in (12), the time-minimizing problem within the free space associated to corridor k can be reformulated as

$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} T_k = \int_{\xi_0}^{\xi_f} \frac{1}{\dot{\xi}(\boldsymbol{x}(\xi), \boldsymbol{\mathcal{Z}}_k)}\, d\xi \tag{13a}$$

$$\text{s.t.} \quad \boldsymbol{x}(\xi_0) = \boldsymbol{x_0}\,, \tag{13b}$$

$$\boldsymbol{x}' = \frac{f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))}{\dot{\xi}(\boldsymbol{x}(\xi), \boldsymbol{\mathcal{Z}}_k)}, \qquad \xi \in [\xi_0, \xi_f] \tag{13c}$$

$$\boldsymbol{x}(\xi) \in \mathcal{X}\,,\ \boldsymbol{u}(\xi) \in \mathcal{U}\,, \quad \xi \in [\xi_0, \xi_f] \tag{13d}$$

$$\boldsymbol{y}(\xi) \in \mathcal{C}_k\,, \tag{13e}$$

$$\boldsymbol{y}(\xi_f) = \boldsymbol{y_f}\,. \tag{13f}$$

The resultant OCP (13) is a finite horizon problem, and unlike the original problem (3), the integration interval on which we solve the optimization is independent of the decision variables. Similar spatial reformulations of the time minimization problem have already been conducted previously [15], [16]. However, these are system-specific and inherit the limitations to Frenet-Serret based reformulations. Both shortcomings are solved by (13), which is compatible with any system whose dynamics are given by f and applicable to a geometric path whose adapted frame can be tailored by PH coefficients $\boldsymbol{\mathcal{Z}}_k$. Besides that, in a similar manner to [16], [18], combining the spatial transformation in (12) with the change of coordinates from the Euclidean position to the spatial coordinates, i.e., $[p_x, p_y, p_z] \rightarrow [\xi, w_1, w_2]$, allows to reformulate the spatial bounds in (13e) as convex constraints. This can be tailored to the corridor's definition, e.g., either as $||\boldsymbol{w}||_2^2 < \delta(\xi)^2$ or $A(\xi)\boldsymbol{w} - b(\xi) \leq 0$, where $\boldsymbol{w} = [w_1, w_2]$, and $\{A(\xi), b(\xi)\}$, $\delta(\xi)$ are the halfspace representation and the radius of the corridor's cross section at ξ, respectively.

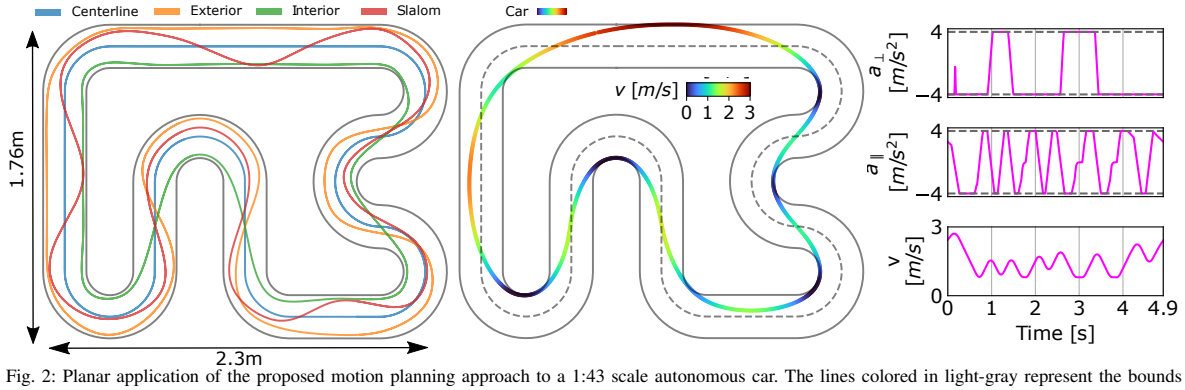


Fig. 2: Planar application of the proposed motion planning approach to a 1:43 scale autonomous car. The lines colored in light-gray represent the bounds of the race-track. *Left:* The four PH splines used to demonstrate the invariance of the solution with respect to PH spline modifications. For clarity, their respective adapted frames are omitted. *Center:* The resultant trajectory obtained for all four PH splines is given by the colored line. The color mapping relates to the car's longitudinal velocity. *Right:* The first two rows represent the lateral and longitudinal accelerations, while the third shows the car's longitudinal velocity, all with respect to time.

TABLE I: States, inputs and constraints of the planar (car) and spatial (quadrotor) systems.

System	States $\boldsymbol{x}$	Inputs $\boldsymbol{u}$	Constraints
Car	$\{\xi, w, \psi, v, D, \delta\}$	$\{\dot{D}, \dot{\delta}\}$	$D \in [-1,1]\,, \delta \in [-0.4, 0.4]\,, \dot{D} \in [-10, 10]\,, \dot{\delta} \in [-2, 2]\,, \{a_\parallel, a_\perp\} \in [-4, 4]$
Quadrotor	$\{\xi, w_1, w_2, \boldsymbol{v}, \boldsymbol{q}\}$	$\{f_c, \boldsymbol{\omega}\}$	$f_c \in [0, 27.52]\,, \{\omega_x, \omega_y\} \in [-15, 15]\,, \omega_z \in [-0.3, 0.3]$

A summary of the proposed motion planning scheme, showing the first, second- and third-stages is depicted in Algorithm 1.

Algorithm 1 *Spatially Constrained Time-Optimal Motion Planning* (**SCTOMP**): Given state $\boldsymbol{x_0}$, goal $\boldsymbol{y_f}$ and environment Ω, find the time-optimal set of states and inputs $\boldsymbol{x}^*, \boldsymbol{u}^*$

function SCTOMP($\boldsymbol{x_0}$, $\boldsymbol{y_f}$, Ω)
  $\boldsymbol{\mathcal{C}} = \mathcal{C}_1, \ldots, \mathcal{C}_m \leftarrow$ CORRIDORS($\boldsymbol{x_0}, \boldsymbol{y_f}, \Omega$) [29]
  $T^* \leftarrow \infty$
  for $k \in \{1, \ldots, m\}$ **do**
    $\lambda_k \leftarrow$ FIND COLLISION-FREE PATH($\boldsymbol{x_0}, \boldsymbol{y_f}, \mathcal{C}_k$)
    $\boldsymbol{\mathcal{Z}}_k \leftarrow$ CONVERT TO PH SPLINE(λ_k) [32]
    $\boldsymbol{x}_k, \boldsymbol{u}_k, T_k \leftarrow$ TIME MIN.($\boldsymbol{x_0}, \boldsymbol{y_f}, \mathcal{C}_k, \boldsymbol{\mathcal{Z}}_k$) (13)
    if $T_k < T^*$ **then**
      $\boldsymbol{x}^*, \boldsymbol{u}^* \leftarrow \boldsymbol{x}_k, \boldsymbol{u}_k$
    end if
  end for
  return $\boldsymbol{x}^*, \boldsymbol{u}^*$
end function

IV. EXPERIMENTS

To evaluate our approach, we divide the experimental analysis into two parts, including systems with different dynamics. First, we test SCTOMP by analyzing its performance in a well-known single-corridor planar scenario (2D, Sec. IV-A), demonstrating that the proposed scheme remains time-optimal regardless of the underlying PH spline. Secondly, the evaluation is expanded to a spatial system (3D, Sec. IV-B) and compared against existing state-of-the-art approaches in a range of case-studies across two different cluttered environments.

Numerical implementation: The OCP (13) is modeled in CasADi [33] and solved by IPOPT [34] after being approximated by a multiple-shooting approach [35], where the optimization horizon $\xi_f - \xi_0$ is discretized into sections with constant decision variables. The integration routines respective to the dynamic constraints (13c) are approximated by a 4th-order Runge-Kutta method.

A. PLANAR SYSTEM IN SINGLE-CORRIDOR SCENARIO

We assess the performance of our method in a state-of-the-art race track for 1:43 scale autonomous cars. Given that this case-study has been the research focus of previous work [19], [36], its time-optimality is well-understood.

It is worth highlighting that our method shows its full potential in unstructured environments, where imposing a geometric reference might be detrimental for planning time-optimal trajectories. However, race-alike scenarios, consist of well-structured – single-corridor– environments in which the progress along the track and its boundaries are perfectly referenced by the centerline, and thus are better suited to path following methods, such as [19], [21], [24]. Nevertheless, the proposed case-study allows for (i) performing an *approximated benchmark* on time-optimality, (ii) demonstrating that the underlying PH splines have no effect on the computed trajectory and (iii) showing the method's applicability to constrained planar systems.

To this end, in a similar way to [19] and [36], we employ the dynamic bicycle model with states $\boldsymbol{x} = [p_x, p_y, \psi, v, D, \delta] \in \mathbb{R}^6$ and inputs $\boldsymbol{u} = \left[\dot{D}, \dot{\delta}\right] \in \mathbb{R}^2$, where $\{p_x, p_y, \psi, v, D, \delta\} \in \mathbb{R}$, refer to the position, yaw, longitudinal velocity, throttle and steering angle. For the respective equations of motion and model coefficients, please refer to eq. 3 in [19]. To exploit the aforementioned convexity in the spatial constraints (13e), we adopt a similar approach presented in [16] and [18], where the Euclidean position

TABLE II: A comparison of the four cases employed for analyzing the sensitivity of the time-optimal solution with respect to PH spline variations. The features of the resultant splines – curvature energy and arc-length – are given in the second and third column, while the respective navigation times – computed from (13) – are shown in the fourth column. All four PH splines, as well as the equivalent trajectory obtained for all the splines are shown in Fig. 2.

PH spline	Energy [-]	Arc-length [m]	Time [s]
Centerline	39.16	8.71	4.924
Exterior	35.65	9.98	4.920
Interior	49.50	7.25	4.925
Slalom	46.16	9.25	4.923

coordinates $[p_x, p_y]$ are projected onto their corresponding spatial counterparts $[\xi, w]$. In addition, to guarantee the validity of the first-principles model's approximation, we impose constraints on the throttle, steering angle, their respective change rates, as well as the longitudinal and lateral accelerations $\{a_\parallel, a_\perp\} \in \mathbb{R}$. The numerical values associated with these constraints can be found in Table I.

Following Algorithm 1, and given the single-corridor nature of the present case-study, the first-stage results in the race-track itself. Subsequently, before solving the time-minimizing problem, in the second-stage we need to compute a PH spline that lies within the race-track. As discussed in Section III-B.2, the definition of the collision-free curve upon which the spline is converted does not influence the computation of the time-optimal trajectory. Putting it another way, any curve that lies within the race-track is eligible to be embedded as a PH spline into the time optimization problem (13). Nonetheless, because the obstacle-free space remains fixed, the time-optimal solution is invariant regardless of the PH spline used. To demonstrate the non-sensitivity of SCTOMP with respect to PH splines, in the second-stage we compute four different versions, according to four different criteria: the track's center, outer and inner-lines, as well as an intermediary slalom. Their geometrical locations are visualized in the left of Fig. 2 and their characteristics are given in the second and third columns of Table II. Comparing the PH splines against each other, the one denoted as *interior* is the shortest and sharpest, followed by *centerline*, while *exterior* and *slalom* are smoother, longer and more curvy. Their differences on shape and size allow for challenging our methodology's robustness with respect to PH spline variations.

Upon solving the third-stage for all four splines, a closely-matched series of lap times are obtained, exhibiting an average duration of 4.922 ± 0.002s and a maximum gap of $0.005\,$s. These timings are presented in the fourth column of Table II and the minor differences in time are attributed to numerical implementation. In fact, all four solutions result in the same trajectory as the one stated in [19], thereby rendering the aforementioned differences negligible and demonstrating that the obtained solution is agnostic of the underlying PH spline.

The resultant trajectory is illustrated in the center column of Fig. 2. As common in racing scenarios, the time-optimal trajectory enters a corner from its inner side and exists from the outside. Moreover, the longitudinal and angular accelerations attached in the right column of Fig. 2 show that the car's actuation is fully exploited by always remaining on its physical limit.

B. SPATIAL SYSTEM IN MULTI-CORRIDOR SCENARIOS

After analyzing SCTOMP's ability to compute time-optimal trajectories in a planar case-study with a single-corridor, we study its applicability to spatial systems within more challenging, i.e., multi-corridor, scenarios.

To this end, we focus on a benchmark introduced in [22], and further extended in [23], which entails the navigation of a quadrotor between start and goal states in two intricate and congested environments: a "forest" characterized by scattered columns, and an indoor "office". To compute time-optimal trajectories for a quadrotor, it is necessary to select single rotor thrust commands as control inputs, as this approach enables full exploitation of the system's physical limits. However, to ensure a fair comparison, we adopt the same input modality as in [22], [23], namely, collective thrust and body rates. It is worth noting that this is the preferred control choice for professional human pilots and has recently been demonstrated to be the most successful input selection for training policies on quadrotors [37].

The states and inputs of the quadrotor are denoted as $\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{q}] \in \mathbb{R}^{10}$, $\boldsymbol{u} = [f_c, \boldsymbol{\omega}] \in \mathbb{R}^4$, where $\{\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{\omega}\} \in \mathbb{R}^3$ refer to the position, velocity and body rates, $\boldsymbol{q} \in \mathbb{R}^4$ is the unit quaternion and $f_c \in \mathbb{R}$ is the collective thrust. The equations of motion of this system can be found in [18]. For the specific coefficients we use the same racing quadrotor as in [17], [22], [23]. As conducted in the preceding planar study, to capitalize the convexity of the spatial constraints (13e), we perform a spatial reformulation from the Euclidean coordinates $\boldsymbol{p}$ to the spatial ones $[\xi, w_1, w_2]$. Moreover, to ensure the integrity of the first-principles model, the collective thrust and the angular velocities are bounded to the values in Table I.

We compare the performance of SCTOMP against three baseline algorithms. Among these, the first one is a variant of CPC [17], a methodology capable of computing theoretically time-optimal trajectories according to the physical limits of the quadrotor, extended by the authors in [22] to make it applicable in cluttered environments. The second baseline is the hierarchical sampling-based method [22], while the third refers to the deep Reinforcement Learning approach [23]. Each of these methods is tested in three and four case-studies, i.e., different start and goal positions in the forest and office

TABLE III: A comparison between the navigation times computed by baseline algorithms and our proposed method. The sampling-based timings show the average of 30 runs and the best in parenthesis.

Environment	Test case	CPC [17]	Sampling-based [22]	RL [23]	**SCTOMP (Ours)**
Forest	0	0.95	1.10 ± 0.13 (0.96)	0.98	**0.94**
	2	0.95	0.98 ± 0.01 (0.96)	1.00	**0.94**
	3	-	1.50 ± 0.17 (1.30)	1.28	**1.19**
Office	0	-	2.38 ± 0.28 (1.93)	1.62	**1.53**
	1	-	1.74 ± 0.06 (1.69)	1.64	**1.54**
	2	-	2.20 ± 0.13 (1.93)	**1.56**	1.63
	3	-	1.81 ± 0.11 (1.58)	1.40	**1.32**

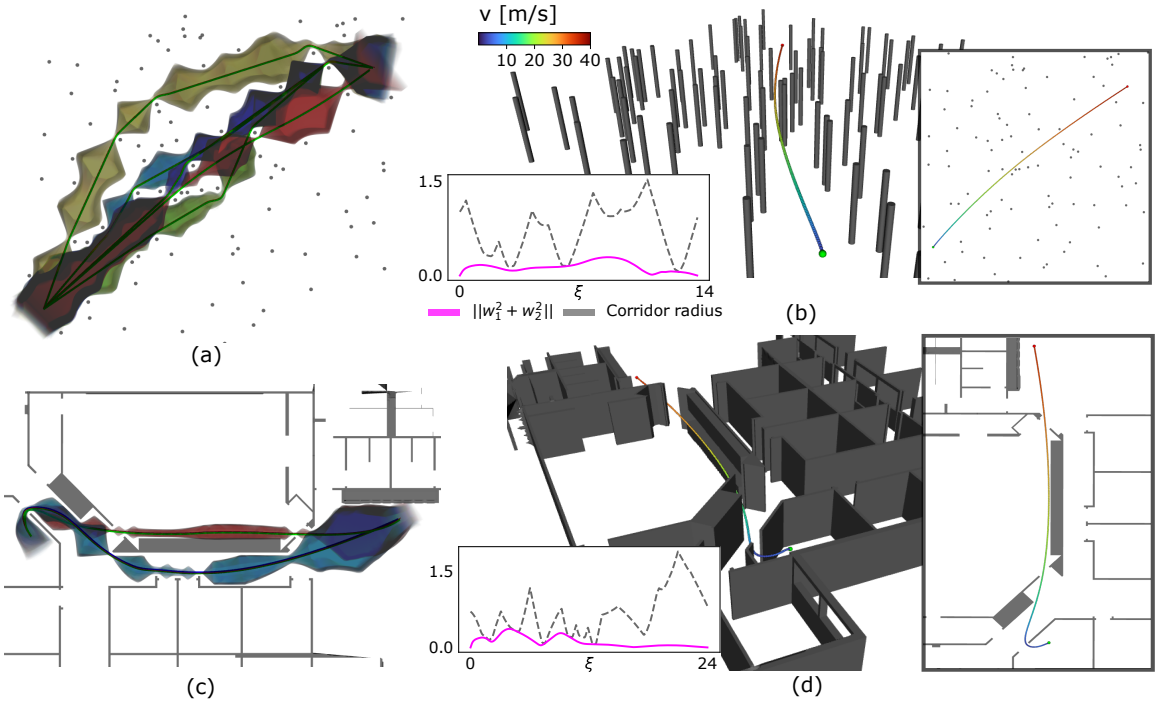


Fig. 3: Two 3D case-studies conducted to compare SCTOMP to baseline algorithms, with the numerical results presented in Table III. The top row (a-b) pertains to test-case 3 in a forest environment, while the bottom row (c-d) corresponds to test-case 0 in an office environment. The left column (a, c) displays a top-view of the various corridors obtained from the first-stage, each of which is represented by the adapted frames respective to the underlying PH splines. The right column (b, d) shows the minimum-time trajectory of the quadrotor, denoted by a colored mapping that is indicative of the velocity norm. At the lower-left side, we plot the distance between the corridor's PH spline and the quadrotor –the norm of the transverse coordinates w_1 and w_2–, alongside the radius of the corridor, showcasing how the computed trajectory fully exploits the available space by tangentially touching its borders.

environments, respectively. Notice that these case-studies are identical to the ones in [22], [23] and are listed in the second column of Table III.

Leveraging the aforementioned modularity of the first-stage, we find the collision-free corridors by defining a tunnel around the topological paths computed in [22]. The radius of the tunnel is specified by the distance to the closest point in an inflated voxel grid of the occupancy map, resulting in overly conservative corridors. As already mentioned, this stage is modular in such a way that it could be replaced by other methods, e.g., [30], [31]. After computing all corridors and solving the second- and third-stages for each of them, the trajectory with lowest navigation time is picked. The corridors and trajectories computed for two different case-studies can be visualized in Fig 3.

The obtained navigation times are reported in Table III. Due to the randomness associated to the sampling-based method, we give the average duration for 30 runs, alongside the best solution in parentheses. The results show that SCTOMP is able to compute the trajectories with the minimum navigation time for all test-cases, except for one. As mentioned in [22], the extended CPC method only finds solutions for the easiest forest case-studies. Reductions of SCTOMP with respect to these theoretical lower bounds are attributed to avoiding the singularity $\dot{\xi} = 0$; instead of starting from a stationary position, we initialize the quadrotor with a minimal velocity $||\boldsymbol{v}||_2 \neq 0\,\mathrm{m\,s^{-1}}$. In the remaining more complex test cases, SCTOMP shows capable of computing shorter time solutions than the baseline methods. On the one hand, the excessively hierarchical procedure of the sampling-based algorithm jeopardizes its time-optimality. On the other hand, aiming to account for tracking disturbances, the RL method learns to prioritize safety over time-optimality by avoiding to fly too close to obstacles, and thus, rendering the obtained trajectories sub-optimal. In contrast, SCTOMP fully exploits the free-space within a given corridor, resulting in trajectories that tangentially touch its borders. This feature can be visualized at the graphs attached to the lower-left side of Figs 3b and 3d. As a consequence, once the corridor associated to the optimal trajectory is found, SCTOMP guarantees to compute the time-optimal trajectory. This relates to the second office case-study, where SCTOMP shows higher navigation times than the RL due to the fact that none of the corridors obtained in first-stage contained the optimal trajectory. This can be overcome by implementing more detailed sampling-based method in stage 1.

V. CONCLUSION

In this work, we presented a motion planning approach capable of computing time-optimal trajectories in spatially constrained environments. The time-optimal trajectories obtained by our method, not only exploit the system's actuation, but also the available free space. For this purpose, we rely on a spatial reformulation that allows for perform-

ing a singularity-free spatial transformation of the system dynamics, as well as the time minimization problem. To compute the underlying parametric functions required by this reformulation, we leverage Pythagorean Hodograph splines. This results in a three-stage scheme, where after finding collision-free corridors with PH splines enclosed in them, their respective coefficients are fed into a spatially reformulated time minimization problem. Experiments on a single-corridor planar case-study show that the solution's time-optimality is agnostic to the underlying PH spline. Moreover, an extensive benchmark against baseline algorithms account for the method's time-optimal capabilities. Lastly, the versatility of the systems – a planar bicycle model and a spatial quadrotor – and differences in the scale of the environments upon which the experiments were conducted emphasize the motion planner's generality and applicability to a wide range of systems and scenarios.

REFERENCES

[1] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[2] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.

[3] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Past, present, and future of autonomous drone racing: A survey," *arXiv preprint arXiv:2301.01755*, 2023.

[4] J. Arrizabalaga, N. van Duijkeren, M. Ryll, and R. Lange, "A caster-wheel-aware mpc-based motion planner for mobile robotics," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 613–618.

[5] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and operation planning of robotic systems*, pp. 3–27, 2015.

[6] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[7] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2015.

[8] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 6137–6142.

[9] Z. Shiller and S. Dubowsky, "Robot path planning with obstacles, actuator, gripper, and payload constraints," *The International Journal of Robotics Research*, vol. 8, no. 6, pp. 3–18, 1989.

[10] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.

[11] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[12] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *International Journal of Control*, vol. 87, no. 6, pp. 1297–1311, 2014.

[13] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, 2018.

[14] L. Consolini, M. Locatelli, A. Minari, A. Nagy, and I. Vajk, "Optimal time-complexity speed planning for robot manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 790–797, 2019.

[15] R. Verschueren, N. van Duijkeren, J. Swevers, and M. Diehl, "Time-optimal motion planning for n-dof robot manipulators using a path-parametric system reformulation," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 2092–2097.

[16] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2017.

[17] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[18] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4044–4050.

[19] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 324–14 329, 2020.

[20] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[21] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, 2022.

[22] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[23] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[24] J. Arrizabalaga and M. Ryll, "Spatial motion planning with pythagorean hodograph curves," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2047–2053.

[25] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 477–482.

[26] R. T. Farouki, *Pythagorean—hodograph Curves*. Springer, 2008.

[27] R. Dietz, J. Hoschek, and B. Jüttler, "An algebraic approach to curves and surfaces on the sphere and on other quadrics," *Computer Aided Geometric Design*, vol. 10, no. 3-4, pp. 211–229, 1993.

[28] H. I. Choi and C. Y. Han, "Euler–rodrigues frames on spatial pythagorean-hodograph curves," *Computer Aided Geometric Design*, vol. 19, no. 8, pp. 603–620, 2002.

[29] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[30] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[31] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[32] Z. Šír and B. Jüttler, "c^2 hermite interpolation by pythagorean hodograph space curves," *Mathematics of Computation*, vol. 76, no. 259, pp. 1373–1391, 2007.

[33] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[34] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[35] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[36] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear mpc in real-time," in *53rd IEEE conference on decision and control*. IEEE, 2014, pp. 2505–2510.

[37] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," *arXiv preprint arXiv:2202.10796*, 2022.

Geometric Slosh-Free Tracking for Robotic Manipulators

Jon Arrizabalaga, Lukas Pries, Riddhiman Laha, Runkang Li, Sami Haddadin and Markus Ryll

published in

IEEE International Conference on Robotics and Automation (ICRA)

2024

Contribution This work introduces a real-time slosh-free tracking technique that generates kinematically feasible joint-space commands based solely on the reference trajectory and the robot's kinematic constraints, enabling agile liquid transportation using robotic manipulators. The pivotal insight—mimicking the end-effector's motion with a quadrotor rather than a pendulum—emerged from extensive discussions with Riddhiman Laha and Rafael Cabral. Lukas Pries contributed valuable ideas on leveraging the differential flatness property of quadrotors to integrate this concept effectively. I was responsible for synthesizing these insights, from recognizing the unique opportunity presented by the quadrotor analogy to developing the complete conceptual and implementation framework. I also carried out the simulated experiments, which were later validated through physical experiments conducted by Runkang Li.n

Geometric Slosh-Free Tracking for Robotic Manipulators

Jon Arrizabalaga[1], Lukas Pries[1], Riddhiman Laha[2], Runkang Li[2], Sami Haddadin[2], Markus Ryll[1,2]

***Abstract*— This work focuses on the agile transportation of liquids with robotic manipulators. In contrast to existing methods that are either computationally heavy, system/container specific or dependant on a singularity-prone pendulum model, we present a real-time slosh-free tracking technique. This method solely requires the reference trajectory and the robot's kinematic constraints to output kinematically feasible joint space commands. The crucial element underlying this approach consists on mimicking the end-effector's motion through a virtual quadrotor, which is inherently slosh-free and differentially flat, thereby allowing us to calculate a slosh-free reference orientation. Through the utilization of a cascaded proportional-derivative (PD) controller, this slosh-free reference is transformed into task space acceleration commands, which, following the resolution of a Quadratic Program (QP) based on Resolved Acceleration Control (RAC), are translated into a feasible joint configuration. The validity of the proposed approach is demonstrated by simulated and real-world experiments on a 7 DoF Franka Emika Panda robot.**

Code: `https://github.com/jonarriza96/gsft`
Video: `https://youtu.be/4kitqYVS9n8`

I. Introduction

Slosh-free liquid handling is of great importance in various fields. For example, it plays a pivotal role in the design of earthquake-resistant watertanks or transportation systems, particularly when handling fuel in planes, propellants in spacecrafts or navigating through waves in ships that have tanks on their decks [1]. The use of slosh-free motions is also critical within industrial assembly lines or the healthcare sector, where robotic manipulators are often required to seamlessly mix, transport, or pour liquids [2].

In such manipulation scenarios achieving agile slosh-free maneuverability poses a significant challenge. Humans require exceptional skill to perform such motions effectively. For instance, only the most skilled waiters and waitresses can transport multiple dishes and glasses without spilling any food or liquid. Nevertheless, by utilizing advanced control methods and leveraging the enhanced agility of robotic systems, there exists potential to realize super-human slosh-free motions. Doing so, offers an appealing incentive to the aforementioned applications. Driven by this goal, in this work we focus on the problem of slosh-free tracking with robotic manipulators.

Starting with a purely-scientific interest that led to the formulation of the Navier-Stokes equations [3], and further motivated by the economic benefits of multiple industries [1], [2], slosh-free motion control is a well-studied and long-sought problem. In the field of robotics, the existing literature

[1]Autonomous Aerial Systems, School of Engineering and Design, Technical University of Munich, Germany. E-mail: `jon.arrizabalaga@tum.de` and `markus.ryll@tum.de`
[2]Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University of Munich

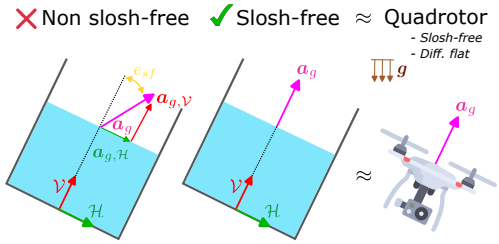


Fig. 1: A planar diagram showcasing the slosh-free condition and its equivalence to the quadrotor. Similar to the existing literature our method couples the longitudinal and rotational accelerations in the container to ensure that the resultant translational acceleration acting on the liquid (in magenta) is perpendicular to the liquid's surface. This implies that the slosh-free angle error e_{sf} (in yellow) is desired to be 0. For this purpose, we emulate the motion of the container with a virtual quadrotor, whose resultant acceleration aligns with the vertical component (in red), and thus, guarantees to be slosh-free.

on slosh-free tracking methods with robotic manipulators can be divided into four categories.

The first category relates to the white-box *fluid analysis / modelling* approach, in which the dynamics of the liquid are approximated by models created using complex methods such as Computational Fluid Dynamics (CFD) [4], [5] or particle-based approaches [6], [7]. Once generated, these models can be used for slosh-free trajectory planning and control [6], [8]. The computational demands involved in generating such detailed models not only hinder their immediate deployment but also render them case-specific. These models rely on various parameters, including liquid viscosity, density, and vessel size and shape. The second category takes the opposite black-box *learning based* approach, where instead of relying on physics to model the liquid's behavior, the slosh-free motions are Learned from Demonstrations (LfD) [9], [10]. These methods are also system-specific since they demand extensive training and are exclusively applicable to the system they were trained for.

The third category encompasses the *motion / smoothness minimization* methods. These rely on a first-principles model built from the Navier-Stokes equations, and aim to minimize the slosh-generating oscillations either by using polynomials to shape the input commands [11]–[13], designing smooth velocity profiles [14] or formulating filtering feedback controllers in the frequency domain [15]. Notice that the first two approaches are feedforward, making them vulnerable to unmodeled dynamics and disturbances, whereas the latter requires feedback from the motion of the liquid, thereby creating a dependency on highly specialized sensors designed to measure the state of the liquid. Additionally, these techniques exclusively pertain to translational movements and, as such, are unsuitable for agile motions, as they do not permit tilting of the vessel.

Published in IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024.

The fourth category overcomes this limitation by coupling the container's translation and rotation by means of a *spherical pendulum model*. These methods cancel out the lateral accelerations acting on the container by emulating the liquid's motion as a virtual pendulum model. Since the original presentation of this technique in [16], [17], further studies have allowed for a better understanding on why the augmented pendulum model results in slosh-free motions [18], [19].

Out of the extensive literature that falls under the fourth category, two works have shown very appealing results. First, in [20] a two-stage plug-in allows to modify the task space commands before being fed to the joint space controller: After passing the desired motion commands through a filter that ensures its smoothness and continuity – similar to the third category–, a spherical pendulum model is used for calculating the vessel's desired orientation, allowing for compensation of the slosh-generating lateral accelerations. In a similar manner, the authors in the second work [21] linearize the spherical pendulum model in its lower equilibrium part, resulting in an optimization-based real-time task-space controller for planar (2D) trajectories. Despite these achievements, these two works share three main drawbacks: First, the analytical simplicity of the pendulum's angular parameterization comes at the expense of singularities, preventing it from being applicable to all $\mathrm{SO}(3)$ [22]. Second, the tracking quality is dependant on the controller design, i.e., the damping-frequency in the first work and the pendulum's rod-length in the second, rendering the overall performance very sensitive to parameter selection. Third, these methods do not tackle the joint space control, leaving it up to inverse kinematics, which is computationally expensive.

This raises the question on how to design a slosh-free tracking method that is (i) singularity-free within all SE3, i.e., spatial (3D) end-effector trajectories in $\mathbb{R}^3$ with a continuous coverage of SO3, (ii) outputs joint commands that fulfill the robot's constraints, (iii) runs online in real-time and (iv) is generic with respect to the liquid properties, cup shape or robot specific parameters.

To address this question, we present a slosh-free tracking algorithm that, similar to [20], [21], leverages the tilt compensation mechanism. However, instead of relying on the spherical pendulum model, we emulate the end-effector's motion by utilizing the model of a quadrotor. When doing so, we can leverage the differential flatness property of a quadrotor to compute a slosh-free reference pose, i.e. position and orientation. This reference is converted to the joint space by two consecutive controllers, which are not only computationally lightweight but also enforce the robot's kinematic constraints. To the best of the authors' knowledge this is the first method that enables real-time slosh-free tracking of any 3D trajectory with robotic manipulators.

To achieve this, our method comprises three main ingredients: 1) Using the quadrotor's differential flatness [23], we compute a slosh-free translation and orientation reference. 2) Subsequently, we implement a cascaded proportional derivative (PD) controller to track the desired reference in the task space [24]. 3) Last but not least, we utilize Resolved-Acceleration Control (RAC) [25] to formulate a convex Quadratic Program (QP) that maps the desired task space accelerations to joint space while satisfying the robot's kinematic constraints.

More in detail, we make the following contributions:

1) We propose a novel perspective on the problem of slosh-free tracking by identifying the appropriateness of differential-flatness based trajectories from the domain of mobile robotics.
2) We formulate an entire pipeline for slosh-free tracking from a desired reference to joint space commands that is (i) capable of tracking any 3D references – even if infeasible –, (ii) deployable in real-time, (iii) system agnostic and (iv) compliant with the robot's kinematic constraints.

The reminder of this paper is organized as follows: Section II formally introduces the slosh-free tracking problem tackled in this work and, subsequently, Section III presents our solution by delving deeper into all three ingredients. Experimental results are shown in Section IV before Section V presents the conclusions.

II. The Slosh-Free Tracking Problem

A. Robotic manipulator model

The forward kinematics of a robotic manipulator are given by a nonlinear mapping between the joint space $\mathcal{J}$ and the task space $\mathcal{T}$, expressed as:

$$\mathrm{T}_e(t) = \begin{bmatrix} \boldsymbol{p}_e(t) & \mathrm{R}_e(t) \\ \mathbf{0} & 1 \end{bmatrix} = \Upsilon(\boldsymbol{q}(t)) \,. \tag{1}$$

Here, $\mathcal{J} \coloneqq \boldsymbol{q}(t) \in \mathcal{Q} \in \mathbb{R}^n$ is the vector of the joint angles, where n is the number of joints, $\mathcal{Q}$ represents the set of kinematically feasible joint configurations and $\mathcal{T} \coloneqq \mathrm{T}_e(t) \in \mathbb{R}^{4x4} \in \mathrm{SE3}$ refers to the homogeneous transformation matrix that represents the end-effector's pose. Its position and orientation are denoted as $\boldsymbol{p}_e(t) \in \mathbb{R}^3$ and $\mathrm{R}_e(t) \in \mathrm{SO3}$, respectively. The nonlinear mapping $\Upsilon(\boldsymbol{q}(t)) : \mathbb{R}^n \mapsto \mathrm{SE3}$ can either be computed from Denavit-Hartenberg parameters or from elementary transform sequences [26].

Derivating (1) on time allows to compute the end effector's longitudinal and angular velocities $\{\boldsymbol{v}_e(t), \boldsymbol{\omega}_e(t)\} \in \mathbb{R}^3$ as:

$$\boldsymbol{\nu}_e(t) = [\boldsymbol{v}_e(t), \boldsymbol{\omega}_e(t)] = J(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) \,, \tag{2}$$

where $J(\boldsymbol{q}(t)) \in \mathbb{R}^{6\times n}$ is the robot's jacobian. Further derivating (2) on time leads to the end-effector's longitudinal and angular accelerations $\{\boldsymbol{a}_e(t), \dot{\boldsymbol{\omega}}_e(t)\} \in \mathbb{R}^3$:

$$\boldsymbol{\alpha}_e(t) = \begin{bmatrix} \boldsymbol{a}_e(t) \\ \dot{\boldsymbol{\omega}}_e(t) \end{bmatrix} = J(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + \dot{\boldsymbol{q}}(t) \otimes H(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) \,, \tag{3}$$

where $H(\boldsymbol{q}(t)) \in \mathbb{R}^{n\times 6\times n}$ is the robot's hessian and $\otimes$ denotes the tensor contraction.

B. Reference trajectory

Let Γ refer to a geometric reference whose Cartesian coordinates are given by a time-based reference $\boldsymbol{p}_r : \mathbb{R} \mapsto \mathbb{R}^3$ that is, at least, $\mathcal{C}^4$:

$$\Gamma = \{t \in [t_0, t_f] \subseteq \mathbb{R} \mapsto \boldsymbol{p}_r(t) \in \mathbb{R}^3\} \,. \tag{4}$$

The position difference between the end-effector and the reference will be denoted as *position error* $e_p(t) \in \mathbb{R}$:

$$e_p(t) = ||\boldsymbol{p}_r(t) - \boldsymbol{p}_e(t)|| \,. \tag{5}$$

C. Problem statement

As initially proposed in [16], [17] and further discussed in [18], [19], sloshing can be prevented by suppressing horizontal accelerations acting on the container. This can be achieved by maintaining the surface of the liquid perpendicular to the resultant acceleration at all times. Notice that this resultant acceleration $\boldsymbol{a}_g(t) \in \mathbb{R}^3$ includes all accelerations acting on the liquid, and thus the gravity $\boldsymbol{g} \in \mathbb{R}^3$ also needs to be considered, i.e., $\boldsymbol{a}_g(t) = \boldsymbol{a}_e(t) + \boldsymbol{g} = \boldsymbol{a}_{g,\mathcal{H}}(t) + \boldsymbol{a}_{g,\mathcal{V}}(t)$, where $\boldsymbol{a}_{g,\mathcal{V}}(t)$ and $\boldsymbol{a}_{g,\mathcal{H}}(t)$ refer to the vertical and horizontal components, respectively. In addition, to quantify the slosh-freeness of a given acceleration, we define the *slosh-free error angle*, $e_{sf} \in \mathbb{R}$, as

$$\tan e_{sf}(t) = \frac{\boldsymbol{a}_{g,\mathcal{H}}(t)}{\boldsymbol{a}_{g,\mathcal{V}}(t)} \,. \tag{6}$$

For a better understanding of these terms, please see the illustrative planar diagram in Fig. 1.

Having defined the robotic manipulator in (1), the geometric reference Γ in (4), as well as the respective position and slosh-free angle errors in eqs. (5) and (6), we can now formally state the problem addressed in this paper:

Problem 1 (Slosh-free tracking for robotic manipulators): *Given a robotic manipulator whose forward kinematics are expressed in* (1)*, and the geometric reference* Γ *in* (4)*, formulate a control method that achieves:*

- P1.1 ***Reference tracking:*** *The end-effector tracks the moving reference within a desired tolerance* ϵ_p*, i.e.,* $e_p(t) \leq \epsilon_p \,; \forall t \in [t_0, t_f]$.
- P1.2 ***Slosh-free motions:*** *The tracking motions are conducted in a slosh-free manner, i.e.,* $|e_{sf}(t)| \leq \epsilon_{sf} \,; \forall t \in [t_0, t_f]$*, where* ϵ_{sf} *is sufficiently small not to generate any slosh.*
- P1.3 ***Constraint satisfaction:*** *The kinematic constraints on the robotic manipulator's joints are satisfied for all times, i.e.,* $\{q(t), \dot{q}(t), \ddot{q}(t), \dddot{q}(t)\} \in \mathcal{Q} \,; \forall t \in [t_0, t_f]$.

III. Methodology

The slosh-free tracking scheme presented in this paper leverages (i) a quadrotor differential flatness based reference generator, (ii) a cascaded proportional derivative task space controller and (iii) a joint space controller based on a convex quadratic programme. These three ingredients are the main building blocks of the proposed methodology. In this section, we present further details in each of them.

A. Quadrotor inspired slosh-free reference generation

As mentioned earlier, in the last few years, a significant amount of the existing slosh-free control literature has focused on formulating novel architectures that leverage the dynamics of the spherical pendulum model. However, as we have already mentioned in Section I, this choice has two main drawbacks: First, the commonly chosen angular parameterization of this model is not applicable to all $\mathrm{SO}(3)$, and second, the pendulum's rod-length introduces a sensitive parameter, leading to a trade-off between tracking quality and slosh-freeness.

In this work, we take a step back and replace the underlying spherical pendulum model with a virtual quadrotor, a system whose configuration inherently fulfills the slosh-free condition presented in Section II. The benefits of this change are fourfold: First, the quadrotor's differential flatness allows us to compute slosh-free orientation references for any 3D trajectory, while also being singularity-free and continuous. Second, the dependency on the pendulum's rod-length is dropped. Third, given that the differential flatness mappings are expressed by closed-form non-linear equations, the calculation of a slosh-free reference does not add any computational burden. Fourth, the capacity to map back and forth from the flat outputs to the quadrotor states allows for computing the reference's longitudinal and angular velocity and acceleration, which might render very appealing for the design of task space controllers.

In the following, we explain how the quadrotor's differential flatness enables the calculation of slosh-free motion references. More specifically, assuming that the end-effector exactly tracks the geometric reference Γ in (4), $e_p(t) = 0$, while mimicking the motions of a (virtual) quadrotor, we proceed to derive the expressions that facilitate the expansion of the position-reference $\boldsymbol{p}_r(t) \in \mathbb{R}^3$ from Γ into a full slosh-free pose-reference $\mathrm{T}_{sf,r} = [\boldsymbol{p}_r(t), \boldsymbol{R}_r(t)] \in \mathrm{SE3}$.

The quadrotor's differential flatness property implies that all its states can be written as algebraic functions of four flat outputs $\boldsymbol{\sigma}(t) = [\boldsymbol{p}_r(t), \psi(t)]$ and their time derivatives $\dot{\boldsymbol{\sigma}}(t)$, $\ddot{\boldsymbol{\sigma}}(t)$, $\dddot{\boldsymbol{\sigma}}(t)$, $\ddddot{\boldsymbol{\sigma}}(t)$. Here $\boldsymbol{p}_r(t)$ is the reference's position, which is obliged to emulate the motion of a (virtual) quadrotor, and $\psi(t)$ is the yaw angle, which is not relevant to the slosh-free problem, and thus, will be fixed to a constant value $\tilde{\psi}$. As a consequence, the flat outputs are fully defined by the geometric reference Γ, i.e., $\boldsymbol{\sigma}(t) = \left[\boldsymbol{p}_r(t), \tilde{\psi}\right] \equiv \Gamma$. This signifies that all the quadrotor or better named the end-effector states can be computed from reference Γ. Out of all these states, the successive task space controller solely requires position and orientation slosh-free states $[\boldsymbol{p}_r(t), \boldsymbol{R}_r(t)]$, and therefore, we will only show the derivations for these two states. For those interested in the computation of the remaining states, please refer to [23].

Commencing with the position reference $\boldsymbol{p}_r(t)$, it becomes evident that it inheretly serves as a flat output that is readily accessible from the reference Γ. Regarding the orientation reference $\boldsymbol{R}_r(t)$, as denoted in the previous paragraph, we parameterize it by a rotation matrix. Its derivation starts by aligning its third component with respect to the resultant acceleration acting on the liquid:

$$\boldsymbol{z}_r(t) = \frac{\boldsymbol{a}_g(t)}{||\boldsymbol{a}_g(t)||} \text{ with } \boldsymbol{a}_g = [\ddot{\sigma}_1(t), \ddot{\sigma}_2(t), \ddot{\sigma}_3(t)] + \boldsymbol{g}. \tag{7a}$$

Next, we define an auxiliary vector

$$\tilde{x}_r = [\cos\sigma_4, \sin\sigma_4, 0]^\intercal \,, \tag{7b}$$

that allows us to compute the first and second components of the rotation matrix by

$$\boldsymbol{y}_r = \frac{\boldsymbol{z}_r(t) \times \tilde{\boldsymbol{x}}_r}{||\boldsymbol{z}_r(t) \times \tilde{\boldsymbol{x}}_r||}\,, \quad \boldsymbol{x}_r(t) = \boldsymbol{y}_r(t) \times \boldsymbol{z}_r(t)\,.$$

If $\tilde{\boldsymbol{x}}_r(t) \times \boldsymbol{z}_r(t) \neq 0$, which can easily be ensured by choosing $\tilde{\psi}$ appropriately, we can stack all three components to finally obtain the reference's slosh-free rotation matrix:

$$\mathrm{R}_r(t) = [\boldsymbol{x}_r(t), \boldsymbol{y}_r(t), \boldsymbol{z}_r(t)]\ . \tag{7c}$$

From eqs. (7a) and (7b) it is apparent that this matrix exclusively depends on the flat outputs and its derivatives. This allows us to compute the reference's slosh-free orientation for the end-effector by simply evaluating the rotation matrix in (7).

It is worth to highlight that, as per (7a), the computed slosh-free orientation matrix is dependent on the reference's translational accelerations $\boldsymbol{a}_r(t)$. At the same time, the end-effector's orientation is defined by the forward kinematics in (1). Putting both facts together, it results that the reference's acceleration profile $\boldsymbol{a}_r(t)$, if tracked by the associated slosh-free orientation $\mathrm{R}_r(t)$ in (7), is correlated to the robotic manipulator's joint angles $q(t)$. Intuitively, or following the derivations in [23], the joint velocities $\dot{q}(t)$ correlate to the jerk profile $\boldsymbol{j}_r(t)$ of the reference, while the joint accelerations $\ddot{q}(t)$ correlate to the snap profile $\boldsymbol{s}_r(t)$.

B. Task space control: A Cascaded Proportional Derivative

The slosh-free pose-reference $\mathrm{T}_{sf,r} = [\boldsymbol{p}_r(t), \boldsymbol{R}_r(t)]$ derived in the previous subsection, is now used as the input for a task space controller that outputs longitudinal and angular acceleration commands, i.e., $\boldsymbol{u}_{\mathcal{T}}(t) = [\boldsymbol{a}_{\mathcal{T}}(t)\,, \boldsymbol{\omega}_{\mathcal{T}}(t)] \in \mathbb{R}^6$. This controller hinges on an error vector which represents the difference between the desired and current poses:

$$\boldsymbol{e}_{\mathcal{T}}(t) = [\boldsymbol{p}_r(t) - \boldsymbol{p}_e(t),\, e_{\mathrm{R}}(\mathrm{R}_r(t), \mathrm{R}_e(t))] \in \mathbb{R}^6\,, \tag{8}$$

where $e_{\mathrm{R}}(\mathrm{R}_r(t), \mathrm{R}_e(t))$ is the error function between the reference and current orientation. To express this function, in this work we adopt the same approach as in [26]. Alternatively, one could also consider utilizing the more compact variant proposed in [27]. As mentioned before, a cascaded PD controller determines the task space acceleration commands. First, given the pose-error $\boldsymbol{e}_{\mathcal{T}}(t)$ defined in (8), the outer loop computes the desired longitudinal and angular end-effector velocities:

$$\boldsymbol{\nu}_{\mathcal{T}}(t) = [\boldsymbol{v}_{\mathcal{T}}(t), \boldsymbol{\omega}_{\mathcal{T}}(t)] = \boldsymbol{k}_{\mathcal{T}} \odot \boldsymbol{e}_{\mathcal{T}}(t) \in \mathbb{R}^6\,, \tag{9a}$$

where $\odot$ refers to the element-wise product and $\boldsymbol{k}_{\mathcal{T}} \in \mathbb{R}^6$ is a proportional gain that controls the convergence rate in the task space. Subsequently, the computed velocities are fed to the inner loop, which, in conjunction with the feedback acquired from the end-effector's longitudinal and angular velocity defined in $\boldsymbol{\nu}_e(t)$ (2), proceeds to compute the longitudinal and angular acceleration commands:

$$\boldsymbol{u}_{\mathcal{T}}(t) = [\boldsymbol{a}_{\mathcal{T}}(t)\,, \boldsymbol{\omega}_{\mathcal{T}}(t)] = \boldsymbol{k}_{\nu} \odot (\boldsymbol{\nu}_{\mathcal{T}}(t) - \boldsymbol{\nu}_e(t))\ , \tag{9b}$$

where $\boldsymbol{k}_{\nu} \in \mathbb{R}^6$ is also a proportional gain.

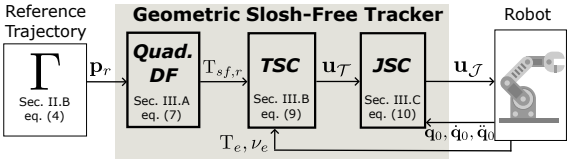


Fig. 2: Block diagram of the presented slosh-free tracking method. "Quad. DF." stands for Quadrotor based differential flatness, "TSC" for task space control and "JSC" for joint space control

C. Joint space control: Optimal Resolved Acceleration

The joint space controller maps the task space acceleration commands $\boldsymbol{u}_{\mathcal{T}}(t)$ to the joint space, $\boldsymbol{u}_{\mathcal{J}}(t) = [\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t)]$. For this purpose, we rely on a well-established method denoted as Resolved-Acceleration Control (RAC) [25], a direct application of the second order differential equation in (3). Unlike the conventional RAC approach, which computes joint accelerations $\ddot{\boldsymbol{q}}(t)$ via a nonlinear inversion of (3) and leaves the solution's feasibility unattended, we solve a constrained optimal control problem at each iteration, ensuring the kinematic feasibility of the obtained joint configuration.

When formulating the optimal control problem, two design choices are particularly noteworthy due to their pivotal roles in enhancing the methodology's robustness and real-time applicability: First, inspired by [28], we ensure that the optimal control problem always remains feasible by adding slack variables, $\boldsymbol{\delta} \in \mathbb{R}^6$, to the accelerations commanded by the task space controller. These are penalized in the cost function and therefore only get activated if the acceleration commanded by the task space controller is infeasible with respect to the kinematic constraints. Second, to facilitate real-time implementation, we make an approximation by assuming that the Jacobian in (2) and the second-order term of (3) remain constant. It's worth noting that this assumption holds true when the optimization problem is solved at a high rate, which is the underlying rationale for this approximation. Following this simplification, the optimal control problem becomes a convex Quadratic Programme (QP), and thus, can be very efficiently solved [29].
In particular, given an initial joint space configuration $[\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \ddot{\boldsymbol{q}}_0]$, acceleration commands from the task space controller $\boldsymbol{u}_{\tau}$ and the robot's kinematic constraints, the QP that we solve at every iteration is:

$$\min_{\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\boldsymbol{\delta}} [\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\delta}]\, P\, [\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\delta}]^{\intercal} \tag{10a}$$

$$\text{s.t.} \quad \boldsymbol{q} = \boldsymbol{q}_0 + \dot{\boldsymbol{q}}\Delta t\,, \tag{10b}$$

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_0 + \ddot{\boldsymbol{q}}\Delta t\,, \tag{10c}$$

$$\boldsymbol{u}_{\mathcal{T}} + \boldsymbol{\delta} = J_0\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}_0 \otimes H_0\, \dot{\boldsymbol{q}}_0\,, \tag{10d}$$

$$\left[\underline{\boldsymbol{q}}, \underline{\dot{\boldsymbol{q}}}, \underline{\ddot{\boldsymbol{q}}}\right] \leq [\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}] \leq \left[\overline{\boldsymbol{q}}, \overline{\dot{\boldsymbol{q}}}, \overline{\ddot{\boldsymbol{q}}}\right]\ , \tag{10e}$$

$$\underline{\ddot{\boldsymbol{q}}} \leq (\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_0)\,/\Delta t \leq \overline{\ddot{\boldsymbol{q}}}\,, \tag{10f}$$

where $P \in \mathbb{R}^{27\times 27}$ is a diagonal positive definite matrix, Δt is the time-step, $J_0 = J(\boldsymbol{q_0})$ and $H_0 = H(\boldsymbol{q_0})$. It's worth noting that the resulting QP is convex, with just 27 variables and all constraints being linear. This illustrates the problem's lightweight nature, and thereby, its real-time applicability.

An overview of the full slosh-free tracking scheme is shown in Fig. 2, where each block within the highlighted

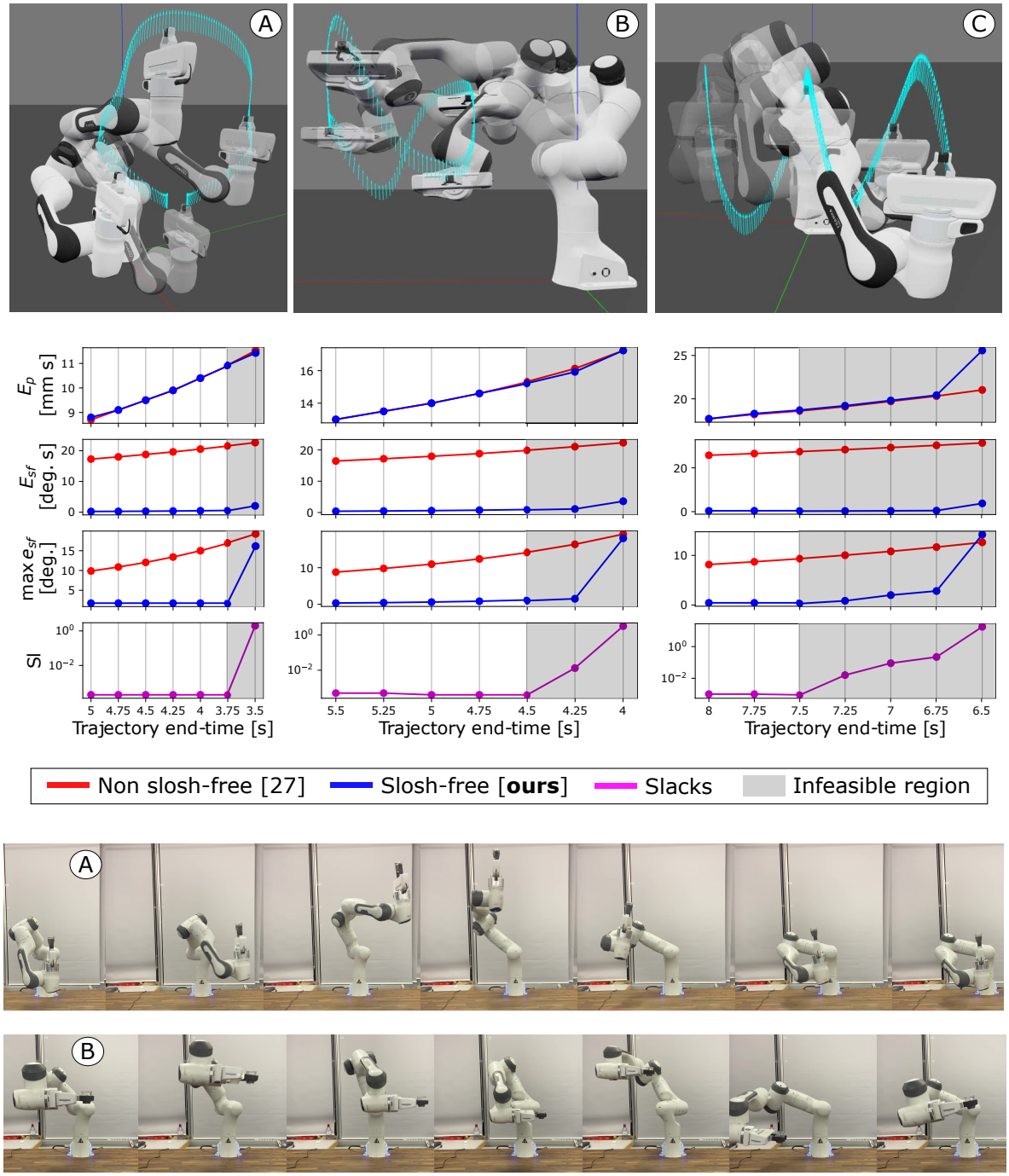


Fig. 3: *Top (above legend box):* A comparison between the presented geometric slosh-free tracker (blue) against a non-slosh-free standard tracker (red) with a Franka Emika Panda robot for three different case-studies: A) Loop (left col.), B) Lissajous (middle col.), C) Helix (right col.). The motions of the robot along these trajectories are shown in the first row. The cyan arrows result from the differential flatness based reference generation and refer to the required acceleration's direction to ensure the motion to be slosh-free. Each case-study has been evaluated for different trajectory execution times. The respective position errors E_p, slosh-free angle errors E_{sf} and maximum slosh-free angle errors $\max e_{sf}$ are depicted in the second, third and fourth rows respectively. The slacks resulting from the slosh-free motions are shown in the fifth row. The gray areas refer to the infeasible regions, i.e, the cases where QP (10) would fail to find a solution if slacks would not be activated. *Bottom (below legend box):* The motions (left to right) resulting from running the proposed slosh-free tracker in a real Franka Emika Panda robot for trajectories (A) and (B) with a cup filled of water attached to its end-effector.

area represents a different component of the proposed slosh-free tracking scheme.

IV. Experiments

To assess the validity of our methodology, we conduct simulated and real-world experiments on a 7 DoF Franka Emika Panda robot arm [30].

A. Simulated analysis

Aiming to understand the performance of our *slosh-free* tracker, we start by comparing it against a *non slosh-free* variant similar to [28], where the quadrotor-based differential flatness part is deactivated, and thus, the resulting end-effector's motions are not slosh-free. The comparison is conducted according to four criterion: First, we evaluate the tracking performance by computing the area below position error $E_p = \int_{t_0}^{t_f} \boldsymbol{e}_p(t)\, dt \in \mathbb{R}$. Second, we analyze the motion's overall slosh-freeness by computing the area below the slosh-free angle error $E_{sf} = \int_{t_0}^{t_f} e_{sf}(t)\, dt \in \mathbb{R}$. Third, we study the extreme slosh-prone cases by also computing the maximum slosh-free angle error $\max \boldsymbol{e}_{sf}(t) \in \mathbb{R};\ \forall t \in [t_0, t_f]$. Fourth, we evaluate the feasibility of the motions by computing the sum of the areas below all slack variables $Sl = \sum_{i=0}^{6} \int_{t_0}^{t_f} \boldsymbol{\delta}_i(t)\, dt \in \mathbb{R}$.

For a better understanding of the solution's performance, we generalize the evaluation by running the comparison in three different case-studies: A) A *loop*-alike trajectory that widely expands the XY plane, B) a *Lissajous* trajectory that requires fast end-effector motions and C) a *helix* that fully exploits the reachable task space. Their differences on shape and scale allow for testing the versatility of the presented method. Moreover, to observe the behavior of the solution in different dynamic regimes, each case-study is evaluated for different navigation times, i.e., starting with low and feasible durations, up to infeasible ones. This allows us to evaluate the performance of the presented slosh-free tracker in slow-moving, as well as in agile and aggressive maneuvers.

To conduct the numerical simulations, we use the Robotics Toolbox for Python [31], which provides a convenient interface to the Franka Emika Panda robot. As a solver for the QP in (10) we use quadprog[1], an open-source software that efficiently solves convex QPs by relying on the Goldfarb/Idnani dual algorithm [32]. Regarding the selection of gains and weights, in the task space control we heuristically choose $\boldsymbol{k}_{\mathcal{T}} = [10, 10, 10, 10, 10, 10]$ and $\boldsymbol{k}_{\nu} = 10\boldsymbol{k}_{\mathcal{T}}$. In the joint space control, we define the weights as $P = \text{blkdiag}(1\text{e}{-8}\, I_7, I_7, 1\text{e}{-8}\, I_7, 1\text{e}3, I_6)$. All of these values are held constant across all the upcoming evaluations.

The robot motions and the respective benchmarks resulting from running the presented slosh-free tracking method are shown in Fig. 3. The first row depicts the evolution of the robot when navigating along the reference trajectories. The cyan arrows refer to the third component of the slosh-free orientation obtained from the quadrotor-based differential flatness. The second, third and fourth rows depict the aforementioned benchmarking metrics E_p, E_{sf}, Sl evaluated for different navigation times. The lower the navigation time, the more aggressive the trajectory. The gray areas within these graphs refer to the infeasible region, where the reference's excessive agility implies that if the slacks would not have been included, the accelerations commanded by the task space controller $\boldsymbol{u}_{\mathcal{T}}(t)$ in (10d) would be infeasible with respect to the kinematic constraints in (10e) and (10f), thereby, causing QP (10) to fail.

The computed motions exhibit three noteworthy characteristics. Firstly, both methods demonstrate equivalent tracking accuracy across all navigation times and case studies. However, the slosh-free variant stands out by delivering a substantial enhancement in the elimination of sloshing. Secondly, in all three cases, the slosh-free variant consistently maintains its slosh-free nature, as indicated by the conditions $e_sf \approx 0$ and $\max e_sf < \epsilon_{sf}$. This holds true unless the reference trajectory becomes excessively infeasible. Thirdly, when the navigation time is exceptionally short, implying an overly agile reference trajectory, adjustments in the form of significantly increased slacks are necessary to render the QP (10) feasible. These adjustments, in turn, result in a substantial modification of the accelerations generated by the task space accelerator, ultimately leading to the generation of sloshing in the motion.

B. Real-world validation

Having understood the behavior of the presented slosh-free tracking method, we proceed in performing real-world experiments. For this purpose, we attach a cup of water to the robotic arm's end-effector and test our solution on two of the three case-studies introduced in the simulations: A) the *loop*-alike trajectory evaluated with a navigation time of 3.75 s and B) the *Lissajous* trajectory with a navigation time of 4.5 s. The remaining *helix* trajectory is not compatible with the laboratory's specific robot setup[2], and thus, will be dropped. The conducted motions are depicted at the lower part of Fig. 3 by image sequences. For a better insight, we encourage readers to view the accompanying video for this paper. As shown by these, and akin to the simulation results, water spillage is successfully avoided.

V. Conclusion

In this work we presented a real-time slosh-free tracker for robotic manipulators. Within the proposed framework, the emulation of the end-effector's motion by a virtual quadrotor has allowed for efficiently calculating a slosh-free reference, which, after going through task and joint space controllers, is converted into a feasible joint space configuration. To validate our findings, first, we have conducted a simulation analysis. The results indicate that our solution approximates slosh-free behaviour without compromising tracking performance, even when dealing with infeasible trajectories. Second, we have run some real-world experiments by attaching a cup of water to a Franka Emika Panda robot and showing that the resulting end-effector motions are spill-free.

[1]quadprog is available in `https://github.com/quadprog/quadprog.git`

[2]The *helix* trajectory requires moving the end-effector below its base's height, which would cause a collision with the large table that holds the robot

REFERENCES

[1] H. Abramson, "Dynamic behavior of liquids in moving containers with applications to propellants in space vehicle fuel tanks," in *Technical Report*, vol. 36, no. 1, 1966, pp. 44–67.

[2] S. a. d. Wiesche, "Computational slosh dynamics: theory and industrial application," *Computational mechanics*, vol. 30, no. 5, pp. 374–387, 2003.

[3] P. G. Lemarié-Rieusset, *The Navier-Stokes problem in the 21st century*. CRC press, 2018.

[4] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Citeseer, 2003, pp. 154–159.

[5] M. H. Djavareshkian and M. Khalili, "Simulation of sloshing with the volume of fluid method," *Fluid Dynamics & Materials Processing*, vol. 2, no. 4, pp. 299–308, 2006.

[6] Z. Pan and D. Manocha, "Motion planning for fluid manipulation using simplified dynamics," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4224–4231.

[7] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, "Gomp-fit: Grasp-optimized motion planning for fast inertial transport," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5255–5261.

[8] Y. Kuriyama, K. Yano, and M. Hamaguchi, "Trajectory planning for meal assist robot considering spilling avoidance," in *2008 IEEE International Conference on Control Applications*. IEEE, 2008, pp. 1220–1225.

[9] J. Hartz, "Adaptive pouring of liquids with a robotic arm," 2018.

[10] L. Rozo, P. Jiménez, and C. Torras, "Force-based robot learning of pouring skills using parametric hidden markov models," in *9th International Workshop on Robot Motion and Control*, 2013, pp. 227–232.

[11] M. Grundelius and B. Bernhardsson, "Motion control of open containers with slosh constraints," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 6142–6147, 1999.

[12] A. Aboel-Hassan, M. Arafa, and A. Nassef, "Design and optimization of input shapers for liquid slosh suppression," *Journal of Sound and Vibration*, vol. 320, no. 1-2, pp. 1–15, 2009.

[13] B. Pridgen, K. Bai, and W. Singhose, "Shaping container motion for multimode and robust slosh suppression," *Journal of Spacecraft and Rockets*, vol. 50, no. 2, pp. 440–448, 2013.

[14] A. Y. S. Wan, Y. D. Soong, E. Foo, W. L. E. Wong, and W. S. M. Lau, "Waiter robots conveying drinks," *Technologies*, vol. 8, no. 3, p. 44, 2020.

[15] K. Yano, S. Higashikawa, and K. Terashima, "Liquid container transfer control on 3d transfer path by hybrid shaped approach," in *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01)(Cat. No. 01CH37204)*. IEEE, 2001, pp. 1168–1173.

[16] M. W. Decker, "Active acceleration compensation for transport of delicate objects," Ph.D. dissertation, School of Civil and Environmental Engineering, Georgia Institute of Technology, 2000.

[17] A. X. Dang and I. Ebert-Uphoff, "Active acceleration compensation for transport vehicles carrying delicate objects," *IEEE transactions on robotics*, vol. 20, no. 5, pp. 830–839, 2004.

[18] J. Han, "A study on the coffee spilling phenomena in the low impulse regime," *Achievements in the Life Sciences*, vol. 10, no. 1, pp. 87–101, 2016.

[19] H. Guang, S. Bazzi, D. Sternad, and N. Hogan, "Dynamic primitives in human manipulation of non-rigid objects," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3783–3789.

[20] L. Biagiotti, D. Chiaravalli, L. Moriello, and C. Melchiorri, "A plug-in feed-forward control for sloshing suppression in robotic teleoperation tasks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5855–5860.

[21] R. I. C. Muchacho, R. Laha, L. F. Figueredo, and S. Haddadin, "A solution to slosh-free robot trajectory optimization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 223–230.

[22] T. Lee, M. Leok, and N. H. McClamroch, "Global formulations of lagrangian and hamiltonian dynamics on manifolds," *Springer*, vol. 13, p. 31, 2017.

[23] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[24] K. J. Åström and T. Hägglund, *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[25] J. Luh, M. Walker, and R. Paul, "Resolved-acceleration control of mechanical manipulators," *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 468–474, 1980.

[26] J. Haviland and P. Corke, "Manipulator differential kinematics: Part i: Kinematics, velocity, and applications," *IEEE Robotics & Automation Magazine*, pp. 2–12, 2023.

[27] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[28] J. Haviland and P. Corke, "A purely-reactive manipulability-maximising motion controller," *arXiv preprint arXiv:2002.11901*, 2020.

[29] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[30] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jähne, L. Hausperger, and S. Haddadin, "The franka emika robot: A reference platform for robotics research and education," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 46–64, 2022.

[31] P. Corke and J. Haviland, "Not your grandmother's toolbox–the robotics toolbox reinvented for python," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 357–11 363.

[32] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical programming*, vol. 27, no. 1, pp. 1–33, 1983.

8 A Holistic View on Egocentric Decision-Making for Robot Autonomy

In the previous chapter, we explored how the egocentric paradigm can be applied to real-world problems. This was done through multiple perspectives, ranging from low-level control design and motion planning in constrained environments to application-oriented autonomous robot behaviors. Although we established the foundational concepts of egocentric methods in Part II, the connections among existing egocentric approaches remain largely unexplored. Motivated by this gap, in this chapter we focus on the following question:

> *Are existing egocentric methods related to each other? If so, can we formulate a unified approach that encompasses them all?*

8.1 A Universal Formulation for Egocentric Decision-Making

In this chapter, we demonstrate that all existing egocentric approaches discussed in the literature—and introduced in Chapter 3—are not isolated methods, as often portrayed, but rather specific instances of a more general approach that can be unified under a framework.

Formulating this framework does not require introducing any new components. Instead, it involves organizing and synthesizing concepts already presented throughout this dissertation. Our main contribution lies in recognizing that all egocentric—or path-parametric—methods share two essential ingredients. When these elements are appropriately defined, their formulation reduces to two key design choices: constructing a suitable cost or reward function, and selecting an appropriate representation of system dynamics.

As previously discussed, these two ingredients are: (i) a spatial projection of the Cartesian system dynamics onto a parametric path, formulated without imposing prior assumptions on the moving frame, introduced in Chapter 4; and (ii) the path-parameterization technique for computing moving frames, presented in Chapter 5. Together, these components form the foundation of a *universal framework* that standardizes the design of egocentric decision-making algorithms. To validate this framework, we accompany its presentation with a diverse set of illustrative examples that highlight its versatility.

8.2 Enclosed Publication

The scientific manuscript associated with this chapter was recently submitted (March 2025) and is currently under review. The code used in this work is available at `https://github.com/jonarriza96/PathParam`. A companion website summarizing the key aspects of the paper and featuring animated examples of the simulations can be accessed at `https://path-parametric.github.io/`.

A Universal Formulation for Path-Parametric Planning and Control

Jon Arrizabalaga, Zbyněk ŠÍR, Zachary Manchester and Markus Ryll

published in

Under Review

2025

Contribution This paper presents a unified framework for egocentric decision-making that consolidates existing methods in the literature. The foundational elements of this framework were introduced in Part II of this dissertation. The motivation for this manuscript arose from a frustration with the prevailing tendency to treat established methods as independent formulations, despite their shared underlying principles. This work aims to demonstrate that many of these approaches are, in fact, specific instances of the broader concepts developed throughout this thesis. As such, the manuscript is driven by a desire to offer a holistic perspective on the material presented herein and to highlight the deep connections between existing egocentric approaches. I was responsible for the entire research pipeline, from synthesizing the ideas to conducting the accompanying simulated examples.

Path-parametric methods have become increasingly popular in navigation algorithm design, spanning high-level planners [1], [2], [3], reinforcement learning (RL) policies [4], [5], [6], and low-level model predictive controllers (MPC) [7], [8], [9]. The core idea behind these methods is to either introduce the path parameter as an additional degree of freedom—allowing the system to regulate its progress along the path [10], [11]—or to perform a coordinate transformation that projects the *Euclidean states* onto *spatial states*, i.e., the progress along the path and the orthogonal distance from it [12], [13], [14]. These parametric formulations have proven effective for three key reasons: (1) they naturally capture the concept of advancement along the path, (2) they embed the path's geometric properties, such as curvature and torsion, into the system dynamics, and (3) spatial constraints become convex in the orthogonal components of the spatial states.

Given the broad range of problems encompassing path-parametric approaches, existing methods remain detached from each other and are frequently presented as independent work. This has resulted in a disjointed body of literature, where these techniques are viewed as distinct methods. Consequently, the reader is left with a fragmented view of the path-parametric problem, making it difficult to understand the interplay between the different techniques. To close this gap, in this paper we show how all these approaches are interconnected by presenting a universal formulation for path-parametric planning and control.

Summary

We present a unified framework for path-parametric planning and control. This formulation is universal as it standardizes the entire spectrum of path-parametric techniques – from traditional path following to more recent contouring or progress-maximizing Model Predictive Control and Reinforcement Learning – under a single framework. The ingredients underlying this universality are twofold: First, we present a compact and efficient technique capable of computing singularity-free, smooth and differentiable moving frames. Second, we derive a spatial path parameterization of the Cartesian coordinates for any arbitrary curve without prior assumptions on its parametric speed or moving frame, and that perfectly interplays with the aforementioned path parameterization method. The combination of these two ingredients leads to a planning and control framework that unites existing path-parametric techniques in literature.

Website: `https://path-parametric.github.io/`

Digital Object Identifier 10.1109/MCS.2025.000000
Date of current version: XXXXXX

To address this, the path-parametric problem is analyzed from three different yet interconnected perspectives (i-iii): We study the (i) *interplay of existing parametric techniques* and show how they can be unified under a single framework consisting of two ingredients: (ii) a *path-parameterization technique* and (iii) a *spatial representation of the system dynamics*. Aiming to exemplify the utility of this framework, we implement state-of-the-art path-parametric methods in a two-link robotic manipulator and a miniature race-car. Finally, we extend the framework's applicability to real-world cases by computing safety corridors that are compliant with unstructured spaces.

PATH-PARAMETRIC PLANNING AND CONTROL

Before presenting a universal formulation for path-parametric planning and control, we first formally define the problems we aim to address. We then establish the conceptual connections between state-of-ther-art methods and, ultimately, demonstrate how they can be unified within a single framework.

An overview of path-parametric methods

Path-parametric methods are those that require a reference path parameterized either by its arc-length or an arbitrary variable. This variable is referred to as the *path-parameter* and it inherently captures the notion of progress along the path. The difference on how this path parameter is leveraged in the design of the planning and control algorithms is what distinguishes the different path-parametric methods. Conceptually, these differences can be grouped according to two standards: the system states and the navigation criterion. The system states refer to the way the path parameter is introduced in the system dynamics, while the navigation criterion relates to how the path parameter is used for planning and control.

Within the different system state representations, we distinguish three main categories: (i) *augmented states*, where the path parameter is introduced as an additional (virtual) degree of freedom in the system dynamics (ii) *projected states*, where the system dynamics are projected onto a coordinate system that is aligned with the reference path, and (iii) *transformed states*, where the system dynamics are transformed from the temporal domain to the spatial domain, causing them to evolve according to the path parameter instead of time.

The navigation criteria can also be grouped into three categories: (i) *progress or velocity profile regulation*, where the path-parameter's speed is desired to follow a predefined speed profile, (ii) *time minimization*, where the aforementioned transformation from the temporal to the spatial domain is leveraged to convert the time minimization into a finite horizon problem, and (iii) *progress maximization*, where the path parameter is used to maximize the progress along the path.

Existing path-parametric methods in the literature result from combining one of the categories corresponding to the state representation with those within the navigation criteria. This is shown in Table 1, which provides an overview of the most relevant path-parametric methods in literature, alongside the system states and navigation criteria they use. The literature attached in the table is sorted by year of publication, and the specific method employed to implement the path-parametric representation: Control Law (CL), Optimization-based Planner (OP), Model Predictive Control (MPC) and Reinforcement Learning (RL). This wide spectrum demonstrates how path-parametric methods have evolved over time, becoming more sophisticated as the planning and control research fields have advanced.

A brief history: From Path Following to MPC and RL

The description of the complex shapes and motions encountered in the world is a fundamental pursuit in the sciences and engineering. While traditional Cartesian equations are adept at representing simple lines and circles, they often prove inadequate for capturing the intricacies of curves and surfaces observed in nature and human-made objects. This limitation necessitates the development of more versatile tools, and parametric equations emerge as a powerful solution. For example, parametric equations can be employed to express complex motions and paths, such as the trajectory of a projectile or the outline of a shape. For this reason, the concept of parameterizing a path by its arc length (or a proxy variable) has captivated the interest of humanity for centuries. Equally significant is the related idea of defining a curve by its curvature, a measure of "how much it bends." The fascination with these methods of defining curves can be traced back to ancient Greece, where philosophers such as Aristotle and mathematicians like Archimedes explored these concepts. The advent of differential calculus in the 17th century introduced new tools and rekindled interest in this problem, attracting the attention of some of history's greatest mathematicians, including Newton, Descartes, Leibniz, Euler, and Gauss. For a comprehensive account of the fascinating history of curve parameterization, please refer to [15].

In the realm of planning and control, the inception of path-parametric methods emerged in the 1980s, notably within the domain of robotic manipulators. Early endeavors identified the advantages of incorporating the path parameter in planning and control laws, such as rescaling infeasible trajectories [16] or computing end-effector motions for traversing a path in minimum time [17], [18]. These works were the first leveraging the path parameter to transform the temporal dynamics into spatial dynamics, as explained in the previous subsection and depicted in

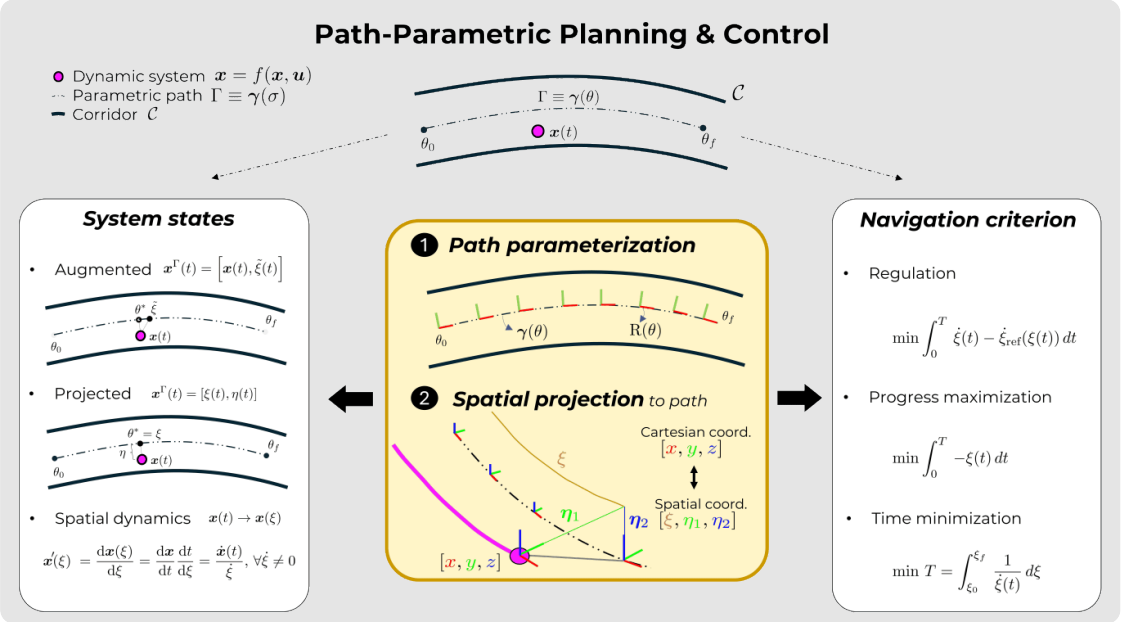


FIGURE 1: Path-parametric methods rely on a reference path parameterized by an auxiliary variable, σ. These methods can be classified based on two key aspects: the system states and the navigation criterion (as indicated by the white boxes). However, the literature on parametric methods remains fragmented, with existing approaches often presented in isolation. To unify these methods, we introduce a universal formulation that highlights their underlying connections. This formulation consists of two main components (shown in the yellow box): (i) a path-parameterization technique for computing moving frames, and (ii) a spatial projection of the Cartesian system dynamics onto the parametric path, formulated without imposing prior assumptions on the moving frame.

Fig. 1. Soon thereafter, applications in the context of navigation for mobile robots appeared in [19], [20], [21], where the error to the path was decoupled into a tangent and an orthogonal component. This decoupling allowed for the design of control laws that regulated the progress along the path, while ensuring the robot would always converge to it. These findings sowed the seed for upcoming more sophisticated formulations, such as the works under the spatial projections category (column 7 in Table 1 and Fig. 1). The subsequent years served for establishing the theoretical foundations and getting a better understanding of these formulations [22], [23], resulting in the articulation of *path-following* as a distinct and superior alternative to path-tracking [24], [25], [26], [11].

These findings ushered in a new era, shifting path-parametric methods from classical closed-form solutions to optimization-based formulations. This transition is constituted by two separate lines of work which laid the foundations for multiple applications and extensions, ultimately becoming the top contributors for the attention that path-parametric methods receive today. On the one hand, in the seminal work [12], it was shown that traversing a path in minimum time with a robotic manipulator – originally tackled in [17], [18] – can be formulated as a convex optimization problem. On the other hand, [35], [10] shed some light on the appealing attributes that result from embedding the tangent and orthogonal error components – originally introduced in [19], [20], [21] – into an optimization problem that is solved in a receding horizon (or MPC) fashion. For the first time, this provided the capacity to deviate from the path, allowing the user to trade-off between traversability and tracking accuracy. Despite the technical differences between [10] and [35], this line of work is commonly referred to as Contouring Control, or MPCC in the specific context of MPC.

These two lines of works laid the foundations that, combined with advances in numerical solvers and increases in computational power, led to a boom in the applicability of path-parametric methods, especially in the context of receding horizon optimization-based control algorithms (MPC). Most of the successful applications of path-parametric methods in real-world systems can

An overview of path-parametric formulations and their applications

TABLE 1: Path-parametric approaches sorted by the year, system applicability, methodology, capacity to account for deviations from the path, system state representation and navigation criterion. The methods are "Control Law" (CL), "Optimization-based Planner" (OP), "Model Predictive Control" (MPC), "Reinforcement Learning" (RL), and for those that do not fit into any of the previous, "Other" (OT).

Ref.	Year	System	Method	Dev.	System states			Navigation criterion		
					Augm.	Proj.	Transf.	Reg.	Time	Progr.
[16]	1984	Rob. Mani.	OT	✗			•	•		
[17]	1985	Rob. Mani.	OP	✗			•		•	
[18]	1987	Rob. Mani.	OT	✗			•		•	
[19]	1988	UGV	CL	✗	•			•		
[20]	1990	UGV	CL	✗	•			•		
[21]	1991	UGV	CL	✗	•			•		
[22]	1995	Any	CL	✗	•			•		
[23]	2004	Any	CL	✗	•			•		
[24]	2004	Ship	CL	✗	•			•		
[25]	2005	Any	CL	✗	•			•		
[27]	2004	Planar formation	CL	✗		•		•		
[28]	2004	Boundary Tracking	CL	✗		•		•		
[29]	2005	Formation Flight	CL	✗		•		•		
[30]	2005	Guidance	CL	✗		•		•		
[31]	2006	Motion Camouflage	CL	✗		•		•		
[32]	2006	Pursuit	CL	✗		•		•		
[33]	2006	Any (planar)	CL	✗		•		•		
[34]	2006	UAV	CL	✗		•		•		
[26]	2007	Underactuated	CL	✗	•			•		
[12]	2009	Rob. Mani.	OP	✗			•		•	
[35]	2009	Any	MPC	✓	•			•		
[10]	2010	Any	MPC	✗		•				•
[36]	2011	Car	MPC	✓			•		•	
[37]	2012	Car	MPC	✓			•		•	
[38]	2013	Rob. Mani.	MPC	✗	•			•		
[39]	2013	Car	MPC	✓			•	•		
[40]	2013	Crane	MPC	✗	•			•		
[7]	2015	Car	MPC	✓	•					•
[11]	2015	Any	MPC	✓	•			•		
[13]	2016	Rob. manip.	MPC	✓			•	•		
[1]	2016	Rob. manip.	OP	✓			•		•	
[41]	2017	Quadrotor	CL	✗		•		•		
[2]	2017	Quadrotor	OP	✓			•		•	
[42]	2019	Car	MPC	✓	•					•
[43]	2020	Car	MPC	✓		•		•		
[44]	2021	Car	MPC	✓		•		•		
[45]	2021	Quadrotor	MPC	✓	•					•
[9]	2022	Quadrotor	MPC	✓		•		•		
[46]	2022	Quadrotor	MPC	✗	•					•
[14]	2022	Particle	MPC	✓		•				•
[5]	2022	Quadrotor	RL	✓	•					•
[47]	2023	Particle	CL	✗	•			•		
[3]	2023	Car, Quadrotor	OP	✓			•		•	
[48]	2023	Car	MPC	✓		•		•		
[49]	2023	Quadrotor	OP	✓			•		•	
[6]	2023	Quadrotor	RL	✓	•					•
[8]	2024	Rob. manip.	MPC	✓		•		•		
[50]	2024	Quadrotor	MPC	✓	•					•

be found in this family of works. Additionally, the advent of Reinforcement Learning (RL) brought yet another use to the path-parametric problem, where the notion of progress and other geometric features inherent to the path-parametric methods became very appealing for the design of the reward function. Compared to the previous solutions, most of these methods allowed deviations from the path by relying on collision-free tunnels, tubes or corridors. This led to a paradigm shift, where instead of committing to a given path, any trajectory within the admissible space is valid, a very attractive feature for case studies with constrained task spaces. As a consequence, the resulting literature broadened across a wide range of applications, such as autonomous driving [36], [37], [39], [7], [42], [43], [48], autonomous agile flight [2], [45], [9], [46], [49], [6], robotic manipulators [1], [13] or more exotic cases like cranes [40].

From the overview in this subsection it is apparent that path-parametric methods have progressed significantly, spanning across various applications and methodologies. To illustrate this clearly, we have categorized all the aforementioned works in Table 1. From this table, we characterize path-parametric methods according to three observations: First, they are a combination of the system state representation and the navigation criteria explained in the previous subsection, and thus, can be grouped accordingly. Second, the rise of new gradient-based techniques within planning and control, such as optimization and learning, have driven the evolution of path-parametric methods. Third, these methods have become more sophisticated over time, allowing for deviations from the path, and thereby, increasing the applicability to a wider range of problems.

A universal formulation for all path-parametric methods

As discussed earlier, path-parametric methods have evolved to meet the diverse needs of the planning and control communities. This evolution has significantly expanded their application across various domains, resulting in a wide array of real-world implementations. However, this has resulted in a more disjointed literature, with each method often presented in isolation. This fragmented approach has created a scattered body of knowledge where these techniques are perceived as distinct entities, making it challenging for readers to grasp their interconnectedness and broader implications.

To address this gap, this paper aims to demonstrate the unified nature of path-parametric planning and control methods. We show how these approaches are inherently linked by introducing a universal formulation. Central to this formulation are two fundamental components that underpin all existing works in the literature. These components are: (i) the method of path parameterization, which defines how the desired geometric reference is articulated, and (ii) the representation of system dynamics relative to this path parameterization, crucial for understanding how the system behaves along the specified path.

Currently, the literature lacks a generic treatment of these components, contributing to its fragmented nature. By establishing a generalized approach that transcends specific methodologies, we aim to unify path-parametric techniques into a single framework that encompasses everything from traditional path following control laws to more advanced optimization and learning-based methods. In the remainder of the manuscript, we delve into the details necessary to formulate these two foundational ingredients, while ensuring that they meet the demands outlined earlier. Through this unified framework, we seek to not only consolidate existing knowledge but also facilitate the future development and application of path-parametric methods across diverse fields.

Notation: We denote the path parameter by θ, while $\xi(t)$ represents the progression of the dynamical system along the path. Temporal and geometric derivatives are written as $\dot{(\cdot)} = \frac{\mathrm{d}(\cdot)}{\mathrm{d}t}$ and $(\cdot)' = \frac{\mathrm{d}(\cdot)}{\mathrm{d}\theta}$, respectively.

PATH PARAMETERIZING THE REFERENCE PATH

In this section, we focus on the *first ingredient* of the universal framework, namely the method used to transcribe the reference path into a parametric function. We do this in three separate steps: First, we formally define the geometric reference as a parametric function with a moving frame attached to it. Second, we refer to the problem of computing this frame, by considering the existing options, and third, we present an efficient, simple and compact algorithm for this task.

Assigning a path parameter to the reference path

Existing autonomous navigation systems define the reference path either as a set of waypoints [51] or as a parametric function determined by a higher-level planner [52], [53]. In the first scenario, where the reference is characterized by a collection of waypoints $\boldsymbol{wp} = [\boldsymbol{wp}_1, \boldsymbol{wp}_2, \ldots, \boldsymbol{wp}_n] \in \mathbb{R}^{3\times n}$, the path parameterization is done by interpolating these waypoints with a smooth curve, as in [54]. In the second scenario, this interpolation step is unnecessary because the higher-level planner directly provides a predetermined parametric function formulation, such as B-splines. In either case, the reference path is ultimately expressed as a smooth parametric function $\boldsymbol{\gamma} : \mathbb{R} \mapsto \mathbb{R}^3$, dependent on the *path parameter* θ. This implies that the *arc-length* of the reference is given by

$$l(\theta) \coloneqq \int_{\theta_0}^{\theta} ||\boldsymbol{\gamma}'(\theta)||\, d\theta, \tag{1}$$

highlighting a distinction often misunderstood in literature, namely, that the arc-length $l(\theta)$ is distinct from the path parameter θ. To better understand this concept, we

define the term inside the integral as the *parametric speed*:

$$\sigma(\theta) \coloneqq ||\boldsymbol{\gamma}'(\theta)||. \tag{2}$$

Intuitively, the parametric speed captures the variation of the path parameter θ with respect to the arc-length $l(\theta)$. Hence, only when $\sigma = 1$ do the arc-length and the path parameter coincide, resulting in a reference path parameterized by its arc-length[1].

Assigning a moving frame to the reference path

The ability to decouple the system dynamics into tangential and orthogonal components relative to the reference path is fundamental to path-parametric formulations. This decoupling facilitates the design of algorithms that guide the system along the path while limiting deviations from it. To achieve this, it is necessary to define a local frame that evolves with the path. As such, we augment the position function $\boldsymbol{\gamma}(\theta)$ by attaching a moving frame whose rotation matrix is given by another parametric function $\mathrm{R} : \mathbb{R} \mapsto \mathbb{R}^{3\times3}$ also parameterized by θ. Combining it with the position function $\boldsymbol{\gamma}(\theta)$ introduced previously formally defines the *geometric reference* as:

$$\Gamma \coloneqq \{\theta \in [\theta_0, \theta_f] \subseteq \mathbb{R} \mapsto \boldsymbol{\gamma}(\theta) \in \mathbb{R}^3, \mathrm{R}(\theta) \in \mathbb{R}^{3x3}\}. \tag{3}$$

We refer to the moving frame $\mathrm{R}(\theta) \coloneqq [\boldsymbol{e_1}(\theta), \boldsymbol{e_2}(\theta), \boldsymbol{e_3}(\theta)]$ as the *path-frame* $(\cdot)^\Gamma$ and is assumed to be *adapted*, i.e., the first component of the frame coincides with the path's tangent $\boldsymbol{e_1}(\theta) \coloneqq \frac{\boldsymbol{\gamma}'(\theta)}{||\boldsymbol{\gamma}'(\theta)||} = \frac{\boldsymbol{\gamma}'(\theta)}{\sigma(\theta)}$. The change of this frame with respect to the path parameter is determined by the angular velocity $\boldsymbol{\omega}(\theta) \coloneqq [\omega_1(\theta), \omega_2(\theta), \omega_3(\theta)]^\intercal$, which can also be represented in the path-frame as $\boldsymbol{\omega}^\Gamma(\theta) \coloneqq [\omega_1^\Gamma(\theta), \omega_2^\Gamma(\theta), \omega_3^\Gamma(\theta)]^\intercal$:

$$\boldsymbol{\omega}(\theta) = \boldsymbol{\omega}^\Gamma(\theta)\,\mathrm{R}^\intercal(\theta) = \omega_1^\Gamma(\theta)\boldsymbol{e_1}(\theta) + \omega_2^\Gamma(\theta)\boldsymbol{e_2}(\theta) + \omega_3^\Gamma(\theta)\boldsymbol{e_3}(\theta). \tag{4}$$

In either case, the motion of the moving frame $\mathrm{R}(\theta)$ is given by

$$\mathrm{R}'(\theta) = \overbrace{\begin{bmatrix} 0 & -\omega_3(\theta) & \omega_2(\theta) \\ \omega_3(\theta) & 0 & -\omega_1(\theta) \\ -\omega_2(\theta) & \omega_1(\theta) & 0 \end{bmatrix}}^{\Omega(\theta)} \mathrm{R}(\theta) \equiv \mathrm{R}(\theta) \underbrace{\begin{bmatrix} 0 & -\omega_3^\Gamma(\theta) & \omega_2^\Gamma(\theta) \\ \omega_3^\Gamma(\theta) & 0 & -\omega_1^\Gamma(\theta) \\ -\omega_2^\Gamma(\theta) & \omega_1^\Gamma(\theta) & 0 \end{bmatrix}}_{\Omega^\Gamma(\theta)} \tag{5}$$

where $\Omega(\theta)$ and $\Omega^\Gamma(\theta)$ are the skew symmetric matrices associated to the angular velocity vectors in world-frame $\boldsymbol{\omega}(\theta)$ and path-frame $\boldsymbol{\omega}^\Gamma(\theta)$, respectively[2]. From (5), it follows that the components of the angular velocity in the moving frame are given by

$$\left[\omega_1^\Gamma(\theta), \omega_2^\Gamma(\theta), \omega_3^\Gamma(\theta)\right] = \left[\boldsymbol{e_2'}^\intercal(\theta)\,\boldsymbol{e_3}(\theta), \boldsymbol{e_3'}^\intercal(\theta)\,\boldsymbol{e_1}(\theta), \boldsymbol{e_1'}^\intercal(\theta)\,\boldsymbol{e_2}(\theta)\right] \tag{6}$$

As will become apparent in the upcoming section, the angular velocities of the moving frames correspond to the *curvature* κ and *torsion* τ of the path, and their computation is highly dependant on the choice of the moving frame. Subsequently, we will delve into the different options for describing the moving frame, and ultimately present an efficient, simple and compact algorithm for their computation.

Choosing a smooth moving frame

The path-frame associated to the geometric reference Γ in (3) plays a crucial role in the formulation of path-parametric methods, since it allows for decoupling the system dynamics into components tangential and orthogonal to the path. This raises the need for a technique to augment the position function $\gamma(\theta)$ into a moving frame $\mathrm{R}(\theta)$, i.e., a method that solely requires a parametric position reference to compute a frame that evolves with it.

This is a well-studied problem with multiple solutions available in the existing literature, each differing based on the choice of the underlying frame. The most common options are the Frenet-Serret Frame (FSF) [63], [64], the Euler-Rodrigues Frame (ERF) [58], [65] and the Parallel Transport Frame (PTF) [56], [60], [61]. From a computation perspective, the first two frames are analytical, as they are given in closed form by local derivative information, while the latter requires global information, and thus, its computation relies on a numerical routine.

The most common choice in the literature is the FSF due to its analytical simplicity. However, it is undefined when the reference path is a straight line (i.e., when the curvature vanishes) and introduces an unnecessary twist over its first component, causing undesired nonlinearities when projecting system dynamics to the path frame or representing the collision-free space around the path. Within analytical frames, an alternative is the ERF, which remains defined even if the reference path is straight. However, the ERF cannot be guaranteed to be twist-free and relies on a Pythagorean Hodograph curve, whose computation is complex and may require numerical routines. Consequently, the most common alternative to the FSF is the PTF, which is both singularity-free and twist-free but requires numerical computation.

Existing methods to compute PTFs are discrete [61], and thus do not allow for the computation of higher derivatives. Additionally, they focus exclusively on the computation of the rotation matrix and neglect its angular velocity. However, as explained in Section , current path-parametric

[1]For further details on how to path parameterize the reference path by its arc-length, please refer to [54]

[2]For a detailed proof of the equivalence in eq. (5), please see in [55]

An overview of moving frames: Frenet-Serret, Euler Rodrigues, and Parallel Transport

As highlighted in the seminal work of Bishop, *there is more than one way to frame a curve* [56]. Aiming to provide an updated perspective on this topic, we focus on the most prominent frames: the Frenet-Serret Frame (FSF), the Euler-Rodrigues Frame (ERF), and the Parallel Transport Frame (PTF). We will explore their underlying mathematical foundations and the properties they inherit as a result of their respective formulations.

Frame	Singularity-free	Twist-free
Frenet-Serret	✗	✗
Euler-Rodrigues	✓	✗
Parallel Transport	✓	✓

Frenet-Serret Frame (FSF)

The Frenet-Serret frame is defined by imposing the second component to be the normalized derivative of the tangent, and the third component othogonal to the first and second, i.e.,

$$\boldsymbol{e_2}(\theta) = \frac{\boldsymbol{e_1'}(\theta)}{||\boldsymbol{e_1'}(\theta)||}, \quad \boldsymbol{e_3}(\theta) = \boldsymbol{e_1}(\theta) \times \boldsymbol{e_2}(\theta)\,. \tag{S7a}$$

Combining these with the tangent component $\boldsymbol{e_1}$, we can explicitly define the FSF by the path function $\gamma(\theta)$ and its derivatives:

$$\boldsymbol{e_1}(\theta) = \frac{\gamma'(\theta)}{||\gamma'(\theta)||}, \quad \boldsymbol{e_2}(\theta) = \frac{\gamma'(\theta) \times (\gamma''(\theta) \times \gamma'(\theta))}{||\gamma'(\theta)|| \cdot ||\gamma''(\theta) \times \gamma'(\theta)||},$$
$$\boldsymbol{e_3}(\theta) = \frac{\gamma'(\theta) \times \gamma''(\theta)}{||\gamma'(\theta) \times \gamma''(\theta)||}\,. \tag{S7b}$$

Given that the frame is exclusively dependant on the derivatives at the evaluating point, FSF is a *local frame*, i.e., the frame can be evaluated by only relying on local derivative information. Notice that the second component is not defined when γ' and γ'' are parallel. To understand the meaning of this, we need to calculate the angular velocity of the frame $\boldsymbol{\omega}_{\text{FSF}}$. To compute it, we combine the eqs. in (4) with (S7), resulting in

$$\omega^{\Gamma}_{\text{FSF},1} = \boldsymbol{e_2'}^{\mathsf{T}}(\theta)\boldsymbol{e_3}(\theta) = \sigma(\theta)\tau(\theta)\,, \tag{S8a}$$
$$\omega^{\Gamma}_{\text{FSF},2} = \boldsymbol{e_3'}^{\mathsf{T}}(\theta)\boldsymbol{e_1}(\theta) = 0\,, \tag{S8b}$$
$$\omega^{\Gamma}_{\text{FSF},3} = \boldsymbol{e_1'}^{\mathsf{T}}(\theta)\boldsymbol{e_2}(\theta) = \sigma(\theta)\kappa(\theta)\,. \tag{S8c}$$

where the first and third components relate to the *torsion* $\tau(\theta)$ and *curvature* $\kappa(\theta)$, respectively. Intuitively, these express how the curve twists (corridor) over the first component or bends over the third component (road). Mathematically, they are given by:

$$\tau(\theta) = \frac{\omega^{\Gamma}_{\text{FSF},1}(\theta)}{\sigma(\theta)} = \frac{||(\gamma'(\theta) \times \gamma''(\theta)) \cdot \gamma'''(\theta)||}{||\gamma'(\theta) \times \gamma''(\theta)||^2}, \tag{S8d}$$

$$\kappa(\theta) = \frac{\omega^{\Gamma}_{\text{FSF},3}(\theta)}{\sigma(\theta)} = \frac{||\gamma'(\theta) \times \gamma''(\theta)||}{||\gamma'(\theta)||^3}\,. \tag{S8e}$$

Notice that the parametric speed $\sigma(\theta)$ is decoupled from the curvature $\kappa(\theta)$ and torsion $\tau(\theta)$, proving that they are agnostic to the underlying path-parameterization. Putting all together, we can compute the angular velocity of the frame as:

$$\begin{aligned}\boldsymbol{\omega}_{\text{FSF}}(\theta) &= \omega^{\Gamma}_{\text{FSF},1}(\theta)\boldsymbol{e_1}(\theta) + \omega^{\Gamma}_{\text{FSF},3}(\theta)\boldsymbol{e_3} \\ &= \sigma(\theta)\left(\tau(\theta)\boldsymbol{e_1}(\theta) + \kappa(\theta)\boldsymbol{e_2}(\theta)\right)\,.\end{aligned} \tag{S8f}$$

From (S8), it is apparent that the FSF is characterized by $\omega^{\Gamma}_{\text{FSF},2} = 0$ (second component always pointing to the center of the curve). This implies that (i) it has singularities when curvature vanishes and (ii) the frame flips when the path transitions from left to right turn. Additionally, it contains a twist along the tangent component, which results in a non-realistic motion of the frame. Therefore we look into alternative moving frames.

Euler-Rodrigues Frame (ERF)

The ERF presents a potential solution to the limitations of the FSF. It is fully defined, i.e., has no singularities even when the reference path is straight, and it is a local frame, implying that it can be calculated by exclusively relying on local derivative information. However, the ERF is constructed upon a Pythagorean Hodograph (PH) curve, a subset of polynomials characterized by the property that the parametric speed $\sigma(\theta)$ is a polynomial on the path parameter θ. Imposing this condition, results in a family of polynomials whose geometric features, such as the arclength, curvature, torsion, etc. can be computed in closed form, by exclusively relying on rational functions. Another benefit of these polynomials is that they inherit an adapted frame, the ERF, which is fully defined and, for a given PH curve, their computation is straightforward.

However, converting a given reference path $\gamma(\theta)$ into a PH curve is not a trivial task, and may require numerical routines. Moreover, guaranteeing differentiability and continuity of the ERF and its angular velocity further increases the complexity of the conversion. Besides that, the algebra and calculus involved in the computation of PH curves and their corresponding ERFs are highly involved, introducing an additional layer of complexity and making them less prone to be reproduced by practitioners within the planning and control communities. Given the amount of detail required to derive and compute these curves and frames, they will be omitted in this manuscript. For further information on PH curves and ERFs, see [57] and [58], respectively. A more applied perspective with navigation applications leveraging PH curves and ERFs can be found in [14], [3], [47]. Lastly, if you are interested in the construction of PH curves and ERFs for path-parametric planning and control algorithms, a real-time and two times differentiable algorithm is given in our accompanying manuscript [59].

Parallel Transport Frame (PTF)

The PTF offers a solution to the shortcomings of the FSF, as well as the ERF. Not only it is fully defined and can be twist-free, but its implementation is also very simple, making it lightweight, efficient and easy to reproduce. These reasons make the PTFs the most suitable choice for path-parametric planning and control algorithms. Their construction is spread out across the literature [60], [61], [49], but none of them addresses the computation of the moving frame's differentiability and angular velocity. Therefore, in this subsection we present an efficient, simple and compact algorithm that given a parametric curve $\gamma(\theta)$, finds the associated PTF and angular velocity $\mathrm{R}_{\mathrm{PTF}}(\theta)$, $\omega_{\mathrm{PTF}}(\theta)$, while also accounting for the respective derivatives. Before presenting this algorithm, we provide an overview of the PTF.

The PTF originates from the seminal work *There is more than one way to frame a curve* [56], which raises the realization that the second e_2 and third e_3 components of the moving frame can be chosen freely, as long as they remain in the orthogonal plane and form an orthonormal frame with the remaining tangent component e_1. This allows for choosing the second and third components to form a parallel vector field that only rotates by the necessary amount, so that it is always perpendicular to the tangent component. This is equivalent to imposing the derivative of the second and third components to point in the direction of the tangential unit vector:

$$e_2'(\theta) = k_1(\theta)e_1(\theta), \quad e_3'(\theta) = k_2(\theta)e_1(\theta)\,. \tag{S9a}$$

To compute the auxiliary variables k_1 and k_2, we combine (S9a) with the orthonormality conditions

$$0 = 1 - e_2^{\mathsf{T}}(\theta)e_2(\theta), \quad 0 = 1 - e_3^{\mathsf{T}}(\theta)e_3(\theta) \tag{S9b}$$

$$0 = e_1^{\mathsf{T}}(\theta)e_2(\theta), \quad 0 = e_1^{\mathsf{T}}(\theta)e_3(\theta)\,. \tag{S9c}$$

Differentiating (S9c) with path parameter θ, combining it with (S9a) and multiplying with $e_1(\theta)$, we get

$$k_1(\theta) = -e_1'^{\mathsf{T}}(\theta)e_2(\theta)\,, \quad k_2(\theta) = -e_1'^{\mathsf{T}}(\theta)e_3(\theta)\,,$$

which ultimately leads to

$$\begin{aligned} e_2'(\theta) &= \left(-e_1'^{\mathsf{T}}(\theta)e_2(\theta)\right)e_1(\theta), \\ e_3'(\theta) &= \left(-e_1'^{\mathsf{T}}(\theta)e_3(\theta)\right)e_1(\theta)\,. \end{aligned} \tag{S10}$$

Eqs. (S10) provide the key insight required to compute the angular velocity of the PTF. By merging them with the definitions of the angular velocity for a generic moving frame in (6), we get

$$\begin{aligned} \omega^{\Gamma}_{\mathrm{PTF},1}(\theta) &= e_2'^{\mathsf{T}}(\theta)e_3(\theta) = \left(-e_1'^{\mathsf{T}}(\theta)e_2(\theta)e_1(\theta)\right)^{\mathsf{T}}e_3(\theta) = 0\,, \\ \omega^{\Gamma}_{\mathrm{PTF},2}(\theta) &= e_3'^{\mathsf{T}}(\theta)e_1(\theta) = \left(-e_1'^{\mathsf{T}}(\theta)e_3(\theta)e_1(\theta)\right)^{\mathsf{T}}e_1(\theta) \\ &= -e_1'^{\mathsf{T}}(\theta)e_3(\theta)\,, \\ \omega^{\Gamma}_{\mathrm{PTF},3}(\theta) &= e_1'^{\mathsf{T}}(\theta)e_2(\theta)\,. \end{aligned} \tag{S11a}$$

where $(\cdot)^{\Gamma}$ indicates that the angular velocity is expressed in the path-frame Γ. When translated to the world-frame as defined in (4), it becomes

$$\begin{aligned} \omega_{\mathrm{PTF}}(\theta) &= \omega^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}^{\mathsf{T}}_{\mathrm{RMF}}(\theta) \\ &= \omega^{\Gamma}_{\mathrm{PTF},2}(\theta)e_2(\theta) + \omega^{\Gamma}_{\mathrm{PTF},3}(\theta)e_3(\theta)\,. \end{aligned} \tag{S11b}$$

From eqs. (S11), there are two key insights to be drawn. First, in contrast to the FSF in (S8), the angular velocity of the PTF does not have a component along the tangent e_1, i.e., $\omega^{\Gamma}_{\mathrm{PTF},1} = 0$, and thus, the PTF is twist-free. Second, the angular velocity $\omega_{\mathrm{PTF}}(\theta)$ is exclusively dependant on the second and third components of the moving frame $\mathrm{R}_{\mathrm{PTF}}(\theta)$ and the derivative of its tangent $e_1'(\theta)$. Thus, if the initial frame is given $\mathrm{R}_{\mathrm{PTF}}(\theta_0) = [e_1(\theta_0), e_2(\theta_0), e_3(\theta_0)]$, we can compute the PTF at any point along the curve by forward integrating it as

$$\mathrm{R}_{\mathrm{PTF}}(\theta_{i+1}) = e^{\Omega_{\mathrm{PTF}}(\theta_i)\Delta\theta_i}\mathrm{R}_{\mathrm{PTF}}(\theta_i)\,, \tag{S12}$$

where $\Delta\theta_i = \theta_{i+1} - \theta_i$ and $\Omega_{\mathrm{PTF}}(\theta_i)$ refers to the skew symmetric matrix associated with the angular velocity ω_{PTF}, as defined in (5). Notice that $e_1'(\theta)$ drives the integration forward, and thus, the reference path $\gamma(\theta)$ needs to be at least C^2. Additionally, conducting the integration within $\mathrm{SO}(3)$, ensures that the frame remains orthonormal, in contrast to other standard methods, such as Euler or Runge-Kutta [62]. Algorithm 1 summarizes the steps required to compute the PTF and its angular velocity.

Algorithm 1 *Parallel Transport Frame Integr.* (**PTFI**):
Input: $\theta_{0,\dots,N}$, $\mathrm{R}_{\mathrm{PTF}}(\theta_0)$, $e_1'(\theta_{0,\dots,N})$
Output: $\mathrm{R}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\omega_{\mathrm{PTF}}(\theta_{0,\dots,N})$

function PTFI($\theta_{0,\dots,N}$, $\mathrm{R}_{\mathrm{PTF}}(\theta_0)$, $e_1'(\theta_{0,\dots,N})$)
  for $i \in \{0, \dots, N-1\}$ **do**
    $\omega_{\mathrm{PTF},i} \leftarrow \textsc{AngVel}(e_{1,i}', \mathrm{R}_{\mathrm{PTF},i})$ (S11)
    $\Omega \leftarrow \textsc{SkewMatrix}(\omega_{\mathrm{PTF},i})$ (5)
    $\Delta\theta = \theta_{i+1} - \theta_i$
    $\mathrm{R}_{\mathrm{PTF},i} \leftarrow \textsc{Integr}(\mathrm{R}_{\mathrm{PTF},i}, \Omega, \Delta\theta)$ (S12)
  end for
  return $\mathrm{R}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\omega_{\mathrm{PTF}}(\theta_{0,\dots,N})$
end function

Having obtained the PTF components $\mathrm{R}_{\mathrm{PTF}}$ and angular velocity ω_{PTF}, we can now proceed to computing the respective higher derivatives. Given that differentiable path-parametric methods require from first order (learning) and second order (optimization) gradients, we will focus on the derivation of the first two derivatives. However, the suggested methodology can be extended to higher order derivatives by following the same procedure.

Starting with the first order derivatives, $\mathrm{R}'_{\mathrm{PTF}}$ is readily available from (5) and (S11), while the angular acceleration $\boldsymbol{\alpha}_{\mathrm{PTF}}$ can be obtained by derivating (S11):

$$\boldsymbol{\alpha}_{\mathrm{PTF}}(\theta) = \boldsymbol{\alpha}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}^{\intercal}_{\mathrm{PTF}}(\theta) + \boldsymbol{\omega}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}'^{\intercal}_{\mathrm{PTF}}(\theta) \tag{S13a}$$

with

$$\boldsymbol{\alpha}^{\Gamma}_{\mathrm{PTF}}(\theta) = \Big[0, -\Big(\boldsymbol{e}_1''^{\intercal}(\theta)\boldsymbol{e}_3(\theta) + \boldsymbol{e}_1'^{\intercal}(\theta)\boldsymbol{e}_3'(\theta)\Big), \Big(\boldsymbol{e}_1''^{\intercal}(\theta)\boldsymbol{e}_2(\theta) + \boldsymbol{e}_1'^{\intercal}(\theta)\boldsymbol{e}_2'(\theta)\Big)\Big]\,. \tag{S13b}$$

To derive the second derivative of the moving frame $\mathrm{R}''_{\mathrm{PTF}}$, we express the angular velocity (S11) and acceleration (S13) by their skew-symmetric matrices Ω_{PTF}, Ω'_{PTF} and combine them with the derivative of (5) over θ, which results in:

$$\mathrm{R}''_{\mathrm{PTF}} = \Omega'_{\mathrm{PTF}}(\theta)\mathrm{R}_{\mathrm{PTF}}(\theta) + \Omega_{\mathrm{PTF}}(\theta)\mathrm{R}'_{\mathrm{PTF}}(\theta) \tag{S14}$$

Finally, to calculate the second derivative of the angular velocity –denoted as angular jerk $\boldsymbol{j}_{\Gamma}$– we derivate (S13):

$$\boldsymbol{j}_{\mathrm{PTF}}(\theta) = \boldsymbol{j}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}^{\intercal}_{\mathrm{PTF}}(\theta) + \boldsymbol{\alpha}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}'^{\intercal}_{\mathrm{PTF}}(\theta) + \boldsymbol{\alpha}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}'^{\intercal}_{\mathrm{PTF}}(\theta) + \boldsymbol{\omega}^{\Gamma}_{\mathrm{PTF}}(\theta)\mathrm{R}''^{\intercal}_{\mathrm{PTF}}(\theta)\,, \tag{S15a}$$

where

$$\boldsymbol{j}^{\Gamma}_{\mathrm{PTF}}(\theta) = [0, -\Big(\boldsymbol{e}_1'''^{\intercal}(\theta)\boldsymbol{e}_3(\theta) + \boldsymbol{e}_1''^{\intercal}(\theta)\boldsymbol{e}_3'(\theta) + \boldsymbol{e}_1'^{\intercal}(\theta)\boldsymbol{e}_3''(\theta) + \boldsymbol{e}_1'^{\intercal}(\theta)\boldsymbol{e}_3''(\theta)\Big), \boldsymbol{e}_1'''^{\intercal}(\theta)\boldsymbol{e}_2(\theta) + \boldsymbol{e}_1''^{\intercal}(\theta)\boldsymbol{e}_2'(\theta) + \boldsymbol{e}_1'^{\intercal}(\theta)\boldsymbol{e}_2''(\theta)\Big]\,. \tag{S15b}$$

Eqs. (S11), (S13), (S15) show how the continuity of the reference path $\gamma(\theta)$ and angular frame $\boldsymbol{\omega}(\xi)$ are related:

$$\boldsymbol{e}_1'(\theta),\, \mathrm{R}_{\mathrm{PTF}}(\theta) \to \boldsymbol{\omega}_{\mathrm{PTF}}(\theta)\,, \tag{S16a}$$

$$\boldsymbol{e}_1''(\theta),\, \mathrm{R}_{\mathrm{PTF}}(\theta),\, \mathrm{R}'_{\mathrm{PTF}}(\theta) \to \boldsymbol{\alpha}_{\mathrm{PTF}}(\theta)\,, \tag{S16b}$$

$$\boldsymbol{e}_1'''(\theta),\, \mathrm{R}_{\mathrm{PTF}}(\theta),\, \mathrm{R}'_{\mathrm{PTF}}(\theta), \mathrm{R}''_{\mathrm{PTF}}(\theta) \to \boldsymbol{j}_{\mathrm{PTF}}(\theta)\,, \tag{S16c}$$

i.e., a path with continuity degree $\gamma \in C^n$ relates to $\mathrm{R}_{\mathrm{PTF}} \in C^{n-1}$ and $\boldsymbol{\omega}_{\mathrm{PTF}} \in C^{n-2}$. For example, in the case of continuous dynamical systems, if we desire $\boldsymbol{\omega}_{\mathrm{PTF}}$ to be C^2, γ needs to be (at least) C^4. The specific steps necessary for computing the first and second order derivatives of the moving frame components and angular velocity are detailed in Algorithm 2.

Algorithm 2 *Parallel Transport Frame Deriv.* (**PTFD**):
Input: $\mathrm{R}_{\mathrm{PTF}}$, $\boldsymbol{\omega}_{\mathrm{PTF}}$, $\boldsymbol{e}_1'$, $\boldsymbol{e}_1''$, $\boldsymbol{e}_1'''$ eval. at $\theta_{0,\dots,N}$
Output: $\mathrm{R}'_{\mathrm{PTF}}$, $\mathrm{R}''_{\mathrm{PTF}}$, $\boldsymbol{\alpha}_{\mathrm{PTF}}$, $\boldsymbol{j}_{\mathrm{PTF}}$ eval. at $\theta_{0,\dots,N}$

function PTFD($\mathrm{R}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\boldsymbol{\omega}_{\mathrm{PTF}}(\theta_{0,\dots,N})$)
  for $i \in \{0, \dots, N\}$ **do**
    $\mathrm{R}'_{\mathrm{PTF},i} \leftarrow \textsc{FrameVel}(\mathrm{R}_{\mathrm{PTF},i}, \boldsymbol{\omega}_{\mathrm{PTF},i})$ (5)
    $\boldsymbol{\alpha}_{\mathrm{PTF},i} \leftarrow \textsc{AngAcc}(\boldsymbol{e}''_{1,i}, \mathrm{R}_{\mathrm{PTF},i}, \mathrm{R}'_{\mathrm{PTF},i})$ (S13)
    $\mathrm{R}''_{\mathrm{PTF},i} \leftarrow \textsc{FrameAcc}(\mathrm{R}_{\mathrm{PTF},i}, \boldsymbol{\omega}_{\mathrm{PTF},i}, \boldsymbol{\alpha}_{\mathrm{PTF},i})$ (S14)
    $\boldsymbol{j}_{\mathrm{PTF},i} \leftarrow \textsc{AngJerk}(\boldsymbol{e}'''_{1,i}, \mathrm{R}_{\mathrm{PTF},i}, \mathrm{R}'_{\mathrm{PTF},i}, \mathrm{R}''_{\mathrm{PTF},i})$ (S15)
  end for
  return $\mathrm{R}'_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\mathrm{R}''_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\boldsymbol{\alpha}_{\mathrm{PTF}}(\theta_{0,\dots,N})$, $\boldsymbol{j}_{\mathrm{PTF}}(\theta_{0,\dots,N})$
end function

In summary, Algorithms 1 and 2 facilitate the computation of the moving frame $\mathrm{R}_{\mathrm{RMF}}$, angular velocity $\boldsymbol{\omega}_{\mathrm{RMF}}$ and its derivatives $\{\mathrm{R}'_{\mathrm{RMF}}, \mathrm{R}''_{\mathrm{RMF}}, \boldsymbol{\alpha}_{\mathrm{RMF}}, \boldsymbol{j}_{\mathrm{RMF}}\}$. This combination of accessibility and efficiency, along with the inherent advantages of the PTFs — namely, being singularity-free and twist-free — renders the proposed moving frame computation method highly suitable for path-parametric planning and control algorithms.

methods heavily rely on learning and optimization algorithms, which require first and second order derivatives. To address this, in this manuscript we present an efficient, simple, and compact method based on PTFs that computes the components and angular velocity of a singularity-free and twist-free path frame, while also accounting for the computation of the respective derivatives[3]. Aiming to make this manuscript self-contained, before presenting this algorithm, we also provide an overview of the alternative methods.

Numerical tests

To showcase the concepts presented in this section, we provide two numerical tests: First, we compare the FSF to the PTF for a two-dimensional planar curve and a three-dimensional spatial curve. Second, we observe how the continuity of the reference path and the angular velocity of the moving frames are related.

In Fig. 2 we show a 2D (top row) and 3D (bottom row) comparison between the FSF (first column) and the PTF (second column). The planar comparison is based on the curve $\gamma(\theta) = \sin(2\pi\theta)$ and clearly depicts how the FSF presents a singularity in the inflection point, where the frame is not defined and its normal component abruptly flips. The spatial curve is given by

$$\gamma(\theta) = [(0.6 + 0.3\cos(\theta))\cos(2\theta), (0.6 + 0.3\cos(\theta))\sin(2\theta), 0.3\sin(7\theta)]\,,$$

and showcases how the FSF presents an undesired rotation over its first component. In contrast, the PTF does not suffer from this twist, ultimately leading to a moving frame with a lower angular velocity magnitude (third column).

[3]While presented for continuous dynamical systems, if discreteness is properly addressed, these findings also apply to hybrid systems, e.g. legged robots. For further details, see [66].

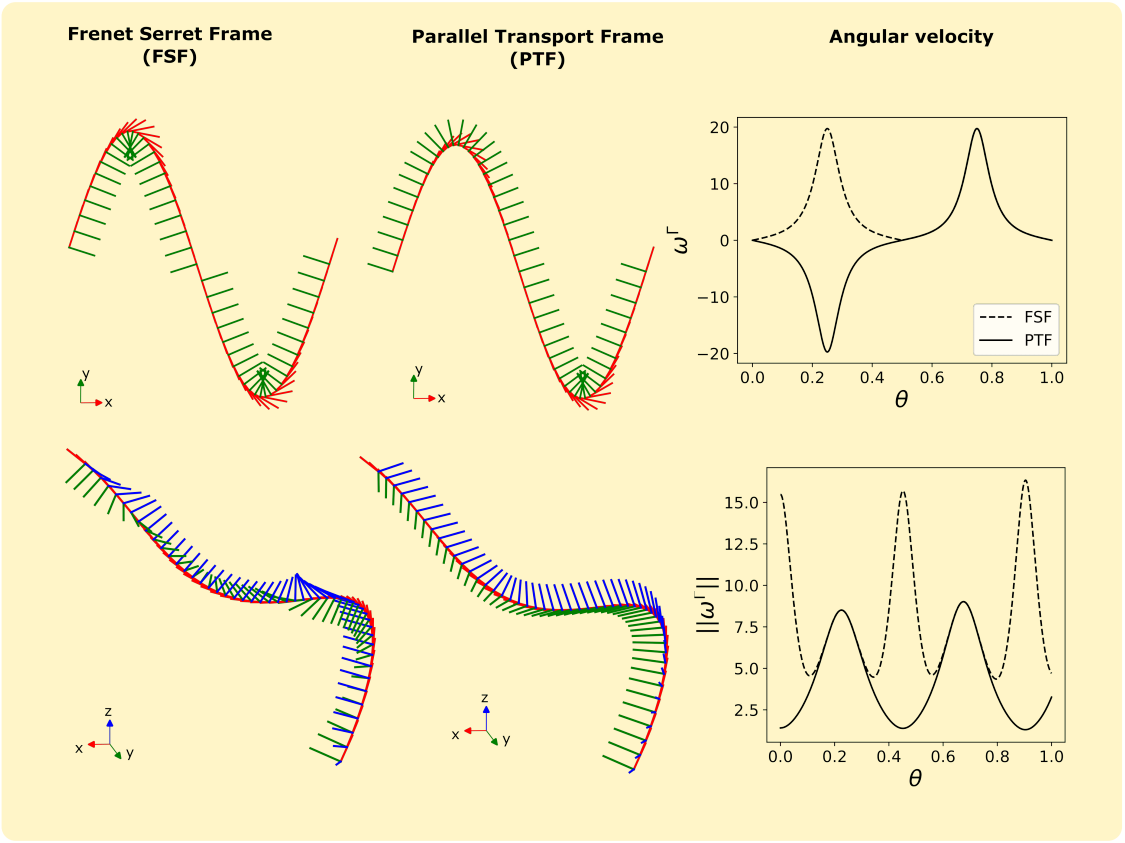


FIGURE 2: Comparison of Frenet-Serret Frame (FSF, first column) and Parallel Transport Frame (PTF, second column) for a two-dimensional planar curve (first row) and three-dimensional spatial curve (second row). The first, second and third components of the moving frame are shown in red, green and blue, respectively. The third column shows the angular velocity of the moving frames.

Regarding the continuity analysis, in eqs. (S16) it was concluded that a reference path γ that is C^n relates to an angular velocity $\boldsymbol{\omega}_{\text{PTF}}$ that is C^{n-2}. To numerically validate this statement, we divide an illustrative curve $\gamma(\theta) = \left[0.5\cos(9\theta), e^{\cos(1.8\theta)}\right]$ into two sections, conduct interpolations of different degrees and observe the continuity of the angular velocity. The obtained results are shown in Fig. 3 and confirm the theoretical analysis: When the interpolation is C^2, the angular velocity is C^0 (only $\boldsymbol{\omega}_{\text{PTF}}$ is continuous); when the interpolation is C^3, the angular velocity is C^1 ($\boldsymbol{\omega}_{\text{PTF}}, \boldsymbol{\alpha}_{\text{PTF}}$ are continuous); and when the interpolation is C^4, the angular velocity is C^2 ($\boldsymbol{\omega}_{\text{PTF}}, \boldsymbol{\alpha}_{\text{PTF}}, \boldsymbol{j}_{\text{PTF}}$ are continuous).

PATH PARAMETERIZING THE CARTESIAN COORDINATES

In this section, we concentrate on the *second ingredient* of the universal framework, specifically a parametric reformulation that projects the Cartesian system dynamics into the spatial states associated with a parametric path. This is addressed in three distinct steps:

First, we formally define the spatial states, as an alternative representation of the Cartesian coordinates. Second, we derive the equations of motion for this representation without making any assumptions regarding the underlying path parameterization. This ensures full compliance with the methods outlined in the previous section and demonstrates that existing formulations in the literature are specific cases of the presented derivation. Lastly, we discuss how the parametric terms required by the derived equations of motion seamlessly integrate with the algorithms introduced in the previous section.

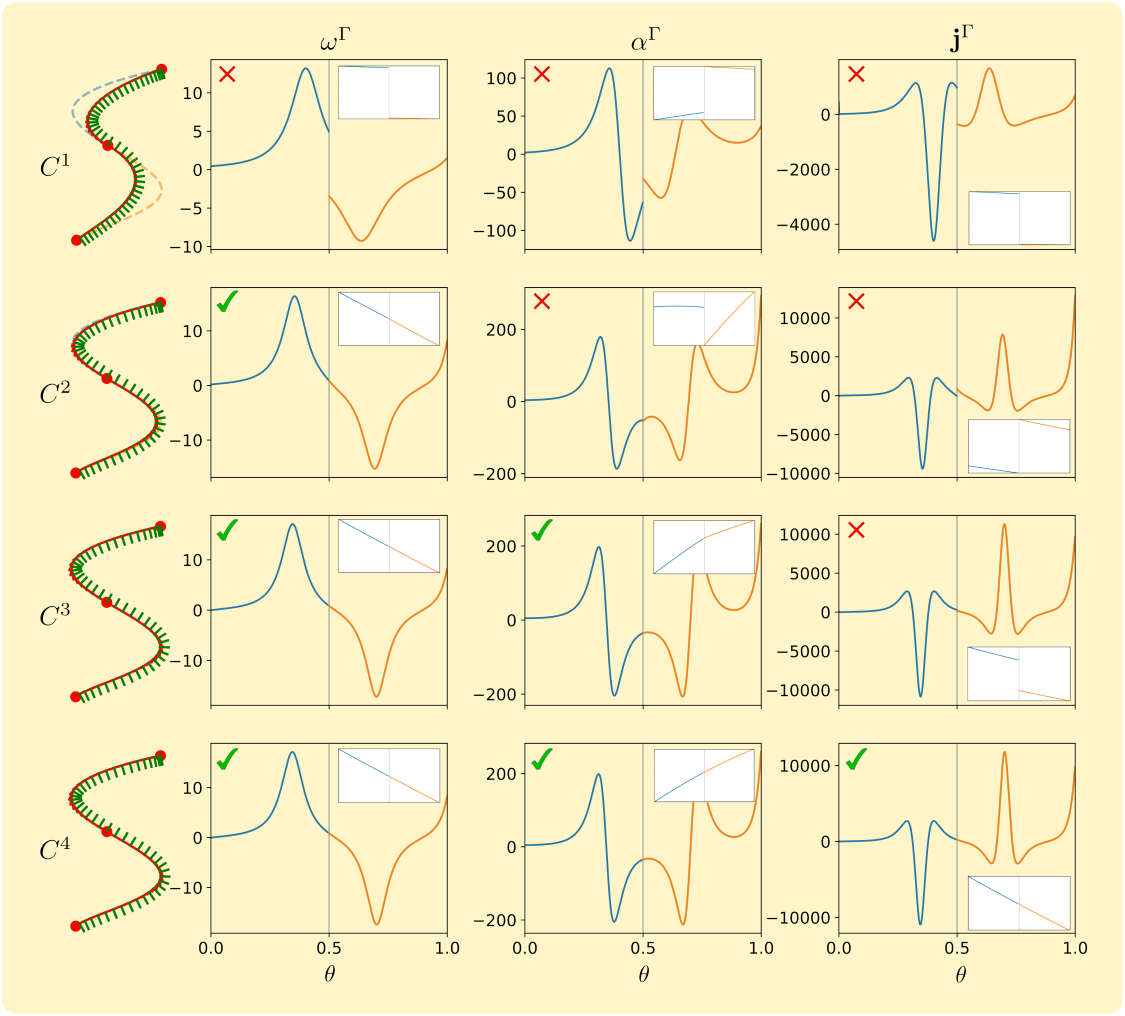


FIGURE 3: Numerical validation of the continuity analysis conducted in eqs. (S16), namely that a reference curve γ that is C^n relates to an angular velocity $\boldsymbol{\omega}_{\text{PTF}}$ that is C^{n-2}. For this purpose we divide an exemplary curve into two sections and interpolate with various continuity degree (C^0 to C^4 from top to bottom). The left column shows the exemplary curve, while the remaining columns depict the angular velocity $\boldsymbol{\omega}^\Gamma$, acceleration $\boldsymbol{\omega}^\Gamma$ and jerk $\boldsymbol{\omega}^\Gamma$. The intersection is given by the red dot located in the middle of the curve at $\theta = 0.5$. The evaluations associated to the first and second sections are depicted in blue and orange. The boxes in the upper right side of each plot provide a more detailed look into the intersection, allowing us to differ the continuous and discontinuous cases. Additionally, the continuous cases have been labelled by a green tick, while the discontinuous ones are marked by a red cross.

Spatial states: An alternative to Cartesian coordinates

We consider continuous time, (non)linear dynamic systems of the form

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{17a}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and $\boldsymbol{u} \in \mathbb{R}^{n_u}$ define the system states and inputs, respectively. We assume that the Cartesian coordinates $\boldsymbol{p}^W(t) \in \mathbb{R}^3$ associated to the system's longitudinal location with respect to a world-frame $(\cdot)_W$ in the Euclidean space are given by a (non)linear mapping h, such that

$$\boldsymbol{p}^{\text{W}}(t) = h(\boldsymbol{x}(t))\,. \tag{17b}$$

To project the system dynamics (17a) onto the reference Γ in (3), we introduce the *spatial coordinates* as an alternative

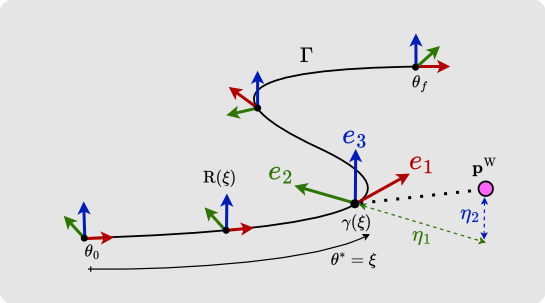


FIGURE 4: Spatial projection of the three-dimensional Cartesian coordinates $\boldsymbol{p}^W$, represented by the pink dot, onto a geometric path Γ with an associated adapted-frame $\mathrm{R}(\xi) = \{\boldsymbol{e_1}(\xi), \boldsymbol{e_2}(\xi), \boldsymbol{e_3}(\xi)\}$. The distance to the closest point on the path $\boldsymbol{\gamma}(\xi)$ is decomposed into the transverse coordinates $\boldsymbol{\eta} = [\eta_1, \eta_2]$.

to the Cartesian representation in (17b). For this purpose, we decouple the system's translational motion into two terms: a *tangent* element – describing the progress along the path – and a *transverse* component – representing the distance perpendicular to the path –.

Given the dynamical system's Cartesian location $\boldsymbol{p}^W(t)$ in (17b), we define the *progress variable* $\xi(t)$ as the path parameter θ of the closest point to the reference Γ, i.e.,

$$\xi(t) \coloneqq \theta^*(t) = \arg\min_{\theta} \frac{1}{2}||\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\theta)||^2 . \tag{18}$$

An explicit representation of the progress variable $\xi(t)$ as in (18) allows for expressing the distance between the dynamical system and the reference path, in both the world-frame and the path-frame:

$$\boldsymbol{d}^{\Gamma}(t) \coloneqq \mathrm{R}^{\intercal}(\xi(t))\boldsymbol{d}^{\mathrm{W}}(t) , \tag{19a}$$

where

$$\boldsymbol{d}^{\mathrm{W}}(t) \coloneqq \boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t)) . \tag{19b}$$

Since we have assumed the path-frame $\mathrm{R}(\xi(t))$ to be adapted, the first element of $\boldsymbol{d}^{\Gamma}(t)$, is zero, while the remaining two are the perpendicular projections, which refer to the aforementioned transverse component $\boldsymbol{\eta}(t) = [\eta_1(t), \eta_2(t)]$. Consequently, eq. (19) simplifies into

$$\boldsymbol{d}^{\Gamma}(t) = [0, \boldsymbol{\eta}(t)]^{\intercal} = \left[0, \boldsymbol{e_2}(\xi(t))\boldsymbol{d}^{\mathrm{W}}(t), \boldsymbol{e_3}(\xi(t))\boldsymbol{d}^{\mathrm{W}}(t)\right]^{\intercal} . \tag{20}$$

Finally, combining the progress variable $\xi(t)$ in (18) with the transverse coordinates $\boldsymbol{\eta}$(t) in (20), we define the *spatial coordinates* as an alternative representation for the location of the dynamical system (17) in the Euclidean space:

$$\boldsymbol{p}^{\Gamma}(t) \coloneqq [\xi(t), \boldsymbol{\eta}(t)] \in \mathbb{R}^3 \tag{21}$$

For an illustrative visualization of the spatial coordinates associated to the reference path Γ, please refer to Fig. 4.

Derivation of equations of motion

Given the dynamical system in (17) and the parametric reference path Γ in (3), we derive the equations of motion for the spatial coordinates in (21). To conduct this derivation, we offer two distinct approaches: one based on kinematics and another rooted in optimality principles.

A kinematic derivation

From the distances in (19) the Cartesian coordinates of the dynamical system (17) can be expressed as

$$\boldsymbol{p}^W(t) \coloneqq \boldsymbol{\gamma}(\xi(t)) + \mathrm{R}(\xi(t))\boldsymbol{d}^{\Gamma}(t) \tag{22}$$

Differentiating (22) with respect to time results in

$$\boldsymbol{v}^{\mathrm{W}}(t) \coloneqq \dot{\xi}(t)\left(\boldsymbol{\gamma}'(\xi(t)) + \mathrm{R}'(\xi(t))\boldsymbol{d}^{\Gamma}(t)\right) + \mathrm{R}(\xi(t))\dot{\boldsymbol{d}}^{\Gamma}(t) . \tag{23}$$

Denoting $\boldsymbol{e}_1^{\mathrm{W}} \coloneqq [1,0,0]^{\intercal}$ as the first component of the world-frame and, recalling from (2) that the curve's parametric speed is $\sigma(\xi(t)) = ||\boldsymbol{\gamma}'(\xi(t))||$, we can derive:

$$\boldsymbol{\gamma}'(\xi(t)) \equiv \sigma(\xi(t))\boldsymbol{e_1}(\xi(t)) \equiv \mathrm{R}(\xi(t))\boldsymbol{e}_1^{\mathrm{W}}\sigma(\xi(t)) . \tag{24}$$

Introducing (24) in (23) and multiplying it with $\mathrm{R}^{\intercal}(\xi(t))$ leads to

$$0 = \dot{\xi}(t)\left(\sigma(\xi(t))\boldsymbol{e}_1^{\mathrm{W}} + \mathrm{R}(\xi(t))^{\intercal}\mathrm{R}'(\xi(t))\,\boldsymbol{d}^{\Gamma}(\xi(t))\right) + \dot{\boldsymbol{d}}^{\Gamma}(t) - \mathrm{R}^{\intercal}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t) .$$

By leveraging the skew-symmetric matrix in (5), the equation simplifies to

$$0 = \dot{\xi}(t)\left(\sigma(\xi(t))\boldsymbol{e}_1^{\mathrm{W}} + \Omega^{\Gamma}(\xi(t))\boldsymbol{d}^{\Gamma}(\xi(t))\right) + \dot{\boldsymbol{d}}^{\Gamma}(t) - \mathrm{R}^{\intercal}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t);,$$

which, when combined with (5) and (19), leads to the equations of motion for the *spatial coordinates* in (S25), highlighted in the yellow rubric on the following page.

An optimality derivation

In here we show that an alternative approach also allows for the derivation of the equations of motion in (S25). More specifically, we focus on the original definition of the progress variable $\xi(t)$ in (18). Given that this is an unconstrained optimization, we leverage the first order optimality condition, so that

$$0 = \frac{\mathrm{d}}{\mathrm{d}\theta}\left(\frac{1}{2}||\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\theta)||^2\right) , \tag{26}$$

which for the optimal path parameter $\theta^*(t) = \xi(t)$ results in $0 = \left(\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t))\right)\boldsymbol{\gamma}'(\xi(t))$. Similarly, we enforce the first optimality condition with respect to time, i.e., $0 = \frac{\mathrm{d}}{\mathrm{d}t}\left(\left(\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t))\right)\boldsymbol{\gamma}'(\xi(t))\right)$, whose expansion is

$$0 = \boldsymbol{v}^{\mathrm{W}}(t)\boldsymbol{\gamma}'(\xi(t)) - \dot{\xi}(t)\boldsymbol{\gamma}'(\xi(t))\boldsymbol{\gamma}'(\xi(t)) + \dot{\xi}(t)\left(\boldsymbol{p}^W(t) - \boldsymbol{\gamma}(\xi(t))\right)\boldsymbol{\gamma}''(\xi(t)) .$$

Path-parameterized Cartesian motion

The motion of a three-dimensional point moving at speed $v^{\mathrm{W}}(t)$ with respect to a reference path parameterized by $\xi(t)$ with parametric speed $\sigma(\xi(t))$ and an adapted path-frame $\mathrm{R}(\xi) = [\boldsymbol{e_1}(\xi(t)), \boldsymbol{e_2}(\xi(t)), \boldsymbol{e_3}(\xi(t))]$ whose angular velocity is $\boldsymbol{\omega}(\xi(t)) = \omega_1^{\Gamma}(\xi(t))\boldsymbol{e_1}(\xi(t)) + \omega_2^{\Gamma}(\xi(t))\boldsymbol{e_2}(\xi(t)) + \omega_3^{\Gamma}(\xi(t))\boldsymbol{e_3}(\xi(t))$ is given by the following equations:

$$\dot{\xi}(t) = \frac{\boldsymbol{e_1^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t)}{\sigma(\xi(t)) - \omega_3^{\Gamma}(\xi(t))\eta_1(t) + \omega_2^{\Gamma}(\xi(t))\eta_2(t)}, \tag{S25a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e_2^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t) + \dot{\xi}(t)\omega_1^{\Gamma}(\xi(t))\eta_2(t), \tag{S25b}$$

$$\dot{\eta}_2(t) = \boldsymbol{e_3^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t) - \dot{\xi}(t)\omega_1^{\Gamma}(\xi(t))\eta_1(t). \tag{S25c}$$

Noticing that $\sigma^2(\xi(t)) = \gamma'(\xi(t))\gamma'(\xi(t))$ the equation above simplifies to

$$\dot{\xi}(t) = \frac{\boldsymbol{v}^{\mathrm{W}}(t)\gamma'(\xi(t))}{\sigma^2(\xi(t)) - \boldsymbol{d}^{\mathrm{W}}(t)\gamma''(\xi(t))}. \tag{27}$$

Recalling that $\boldsymbol{e_1}(\xi(t)) = \frac{\gamma(\xi(t))}{\sigma(\xi(t))}$, follows that $\gamma''(\xi(t)) = \boldsymbol{e_1'}(\xi(t))\sigma(\xi(t)) + \boldsymbol{e_1}(\xi(t))\sigma'(\xi(t))$, whose derivative in the first term is known from (5), i.e., $\boldsymbol{e_1'}(\xi(t)) = \boldsymbol{e_2}(\xi(t))\omega_3^{\Gamma}(\xi(t)) - \boldsymbol{e_3}(\xi(t))\omega_2^{\Gamma}(\xi(t))$ and second term gets cancelled because $\boldsymbol{d}^{\mathrm{W}}(t)$ is perpendicular to $\boldsymbol{e_1}(\xi(t))$. Incorporating this information into (27) results in

$$\dot{\xi}(t) = \frac{\boldsymbol{v}^{\mathrm{W}}(t)\gamma'(\xi(t))}{\sigma^2(\xi(t)) - \sigma(\xi(t))\boldsymbol{d}^{\mathrm{W}}(t)\beta}, \quad \text{where}$$
$$\beta = (\boldsymbol{e_2}(\xi(t))\omega_3^{\Gamma}(\xi(t)) - \boldsymbol{e_3}(\xi(t))\omega_2^{\Gamma}(\xi(t))). \tag{28}$$

Dividing both the numerator and denominator by $\sigma(\xi(t))$ and combining it with (20), coincides with (S25a), the equation of motion for the progress variable $\xi(t)$ derived by the previous kinematic approach. The remaining equations of motions for the transverse coordinates $\boldsymbol{\eta}(t)$ (S25b) and (S25c), can easily be obtained from derivating (20) on time and following similar simplifications as above.

A universal path-parameterization

The equations of motion for the spatial states derived in (S25) are universal, as they do not rely on any assumptions regarding the underlying path parameterization. In other words, they are applicable to any parametric path, regardless of its parametric speed and moving frame. To showcase this universality, we demonstrate how the well-known Frenet-Serret based parametric model is a particular instance of the equations of motion in (S25). Additionally, we show how the two-dimensional case, relevant for planar application such as autonomous driving, is a trivial simplification of the three-dimensional one.

A particular case: The Frenet-Serret based models

The analytical simplicity and ease of implementation make the FSF the most widely applied moving frame in literature [1], [13], [41]. This popularity is rooted in the autonomous driving community [37], [44], [48], where the planar application of the FSF allows for numerical tricks to dodge its fundamental limitations. This influence, combined with the aforementioned simplicity, have made the FSF the de facto standard for path-parametric planning and control, even for three-dimensional applications [2], [9], [50], where the FSF suffers from all the limitations discussed in Fig. 2. As a consequence, the parametric formulations available in the literature are specific to the FSF. To show this, it is sufficient to tailor eqs. (S25) by specifying the angular velocity as in (S8) i.e., $[\omega_1^{\Gamma}, \omega_2^{\Gamma}, \omega_3^{\Gamma}] = \sigma[\tau, 0, \kappa]$. Furthermore, if the curve is assumed to be parameterized directly by its arc-length $L = \xi$, the parametric speed reduces to a unit magnitude $\sigma(\xi) = \frac{\mathrm{d}L}{\mathrm{d}\xi} = 1$:

$$\dot{\xi}(t) = \frac{\boldsymbol{e_1^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t)}{1 - \kappa(\xi(t))\eta_1(t)}, \tag{29a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e_2^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t) + \dot{\xi}(t)\tau(\xi(t))\eta_2(t), \tag{29b}$$

$$\dot{\eta}_2(t) = \boldsymbol{e_3^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t) - \dot{\xi}(t)\tau(\xi(t))\eta_1(t). \tag{29c}$$

The resultant equations of motion are specific to the FSF and match the ones available in literature, e.g. [1], [13], showcasing the generality of our equations in (S25).

The planar 2D case

In the case of planar motions, such as ground vehicles, the parameterization in (S25) smplifies to two states, i.e., the second component of the transverse components gets cancelled $\eta_2(t) = 0$. Consequently, the equations of motion in (S25) reduce to the following planar model:

$$\dot{\xi}(t) = \frac{\boldsymbol{e_1^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t)}{\sigma(\xi(t)) - \omega_3^{\Gamma}(\xi(t))\eta_1(t)}, \tag{30a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e_2^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t). \tag{30b}$$

In a similar way as for the three-dimensional case, the planar model can be tailored to the FSF by specifying the angular velocity as in (S8) and the parametric speed as $\sigma(\xi) = 1$. This results in the following planar model:

$$\dot{\xi}(t) = \frac{\boldsymbol{e_1^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t)}{1 - \kappa(\xi(t))\eta_1(t)}, \tag{31a}$$

$$\dot{\eta}_1(t) = \boldsymbol{e_2^{\mathsf{T}}}(\xi(t))\boldsymbol{v}^{\mathrm{W}}(t), \tag{31b}$$

which is commonly employed in the autonomous driving community.

Modularity: Frames and Equations

Before concluding this section, there are two points that we would like to emphasize: First, the equations derived in (S25) are universal, in the sense that they can be used alongside any moving frame and path-parameterization technique. For example, in the previous subsection we have tailored them for the FSF case. To shed some light on the choice of the moving frame, we suggested the PTF as the most suitable candidate. It goes without saying that Algorithms 1 and 2 interplay perfectly with the equations of motion in (S25). The fusion of the PTF and the universal equations of motion is a powerful formulation, that enjoys the benefits of both ingredients.

The second point is to recognize that, despite our best efforts, we – the authors – might have failed to identify the most appropriate frame and parameterization technique. It is very likely that future researchers will come up with more appropriate methods to define and compute moving frames. Despite this, it is important to insist that the equations of motion in (S25) still remain relevant. The presented universal framework is modular in the sense that the underlying ingredients are completely decoupled, i.e., the path parameterization technique is agnostic to the equations of motion of the spatial states. Therefore, even if future research leads to the development of more suitable moving frames, they can still be used alongside the equations of motion in (S25).

WHY PATH-PARAMETRIC?

To highlight the appeal, practicality and universality of the presented path-parametric framework in designing planning and control algorithms, our analysis is structured into two parts. First, we delve into low-level control, exploring the foundational reasons that led to the development of path-parametric methods. Second, we demonstrate how these core ideas extend to broader motion planning scenarios, where the desired trajectories are intended to fully exploit the available free space.

An illustrative case-study: A two-link robotic manipulator

To perform these experiments, we utilize a two-link *robotic manipulator*, which serves as a baseline system. This platform allows us to analyze the motions of a nonlinear system subject to constraints in both configuration and task spaces. The state vector, defined as $\boldsymbol{x} = [p_x, p_y, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$, captures the end-effector position, joint angles, and their respective velocities. The control

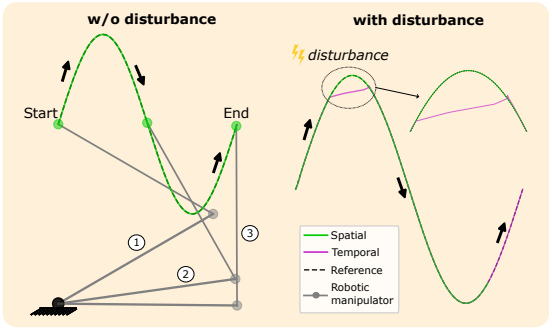


FIGURE 5: Comparison between a *temporal* and *spatial* reference-tracking for a two joint robotic manipulator when traversing a sinusoidal path. The right side depicts three successive sequences for motions of the robotic manipulator in a nominal scenario, without disturbances. In the right, we show the trajectories obtained with both methods in the presence of a disturbance. In the left, the temporal reference keeps on progressing while the disturbance is happening, forcing it to *catch-up*, resulting in a large deviation. In comparison, the spatial reference only depends on its location, and thus, is able to resume without generating a large error.

inputs, represented as $\boldsymbol{u} = [\ddot{\theta}_1, \ddot{\theta}_2]$, correspond to the joint accelerations. The equations of motion for the end-effector's position are given by

$$\dot{p}_x = -L_1\dot{\theta}_1 \sin(\theta_1) - L_2\dot{\theta}_2 \sin(\theta_2), \tag{32a}$$

$$\dot{p}_y = L_1\dot{\theta}_1 \cos(\theta_1) + L_2\dot{\theta}_2 \cos(\theta_2), \tag{32b}$$

where $L_1 = 1$ and $L_2 = 1$ are the lengths of the links. Additionally, the geometric reference used along the upcoming examples is a planar sinusoidal curve parameterized by ξ and given by the following equation:

$$\gamma(\xi) = 1 + 0.5\sin(2\,\pi\,\xi) \tag{33}$$

Temporal vs Spatial reference-tracking

We begin by analyzing a simple yet illustrative example that demonstrates the benefits of path-parametric methods: a comparison between temporal and spatial references. The task involves guiding the robotic manipulator's end-effector along the geometric reference defined in (33) using the dynamics described in (32). A 2 s disturbance at the first maximum point temporarily obstructs the end-effector's progress along the path. To ensure a fair comparison, both the *temporal* and *spatial* formulations are calibrated to achieve the same navigation time in the absence of disturbances.

In particular, the controllers are formulated in a Nonlinear Model Predictive Control (NMPC) fashion, where an Nonlinear Program (NLP) is solved at a receding hori-

zon fashion. Specifically, we solve the NLP in a multiple shooting fashion with N nodes as:

$$\min_{\substack{\boldsymbol{x}_{0,\dots,N} \\ \boldsymbol{u}_{0,\dots,N-1}}} ||\boldsymbol{x}_N - \boldsymbol{x}_{\text{ref}}(\cdot)||^2_{Q_E} + \sum_{k=0}^{N} ||\boldsymbol{x}_k - \boldsymbol{x}_{\text{ref}}(\cdot)||^2_Q + ||\boldsymbol{u}_k||^2_R \tag{34a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_i, \tag{34b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, dt), \quad k = [0, \dots, N-1] \tag{34c}$$

$$c(\boldsymbol{x}_k, \boldsymbol{u}_k) \geq 0, \quad k = [0, \dots, N-1] \tag{34d}$$

where the reference in (34a) is time-dependent – $x_{\text{ref}}(t)$ – for the tracking case, while it is path-parameter dependent – $x_{\text{ref}}(\xi)$ – in the spatial case. The continuity constraints in (34c) account for the equations of motion of the robotic manipulator given in (32) with a horizon T and a time step $dt = T/N$. The state and input constraints are expressed with (34d).

We solve the NLP (34) using the optimal control framework ACADOS [67], which employs a sequential quadratic programming (SQP) method. The underlying quadratic programs leverage their multi-stage structure and are solved using HPIPM [68]. For dynamics integration, an explicit 4th-order Runge-Kutta method is applied with $T = 1$ and $N = 20$. To navigate the reference path, the weights are designed to prioritize position tracking, with $Q = Q_E = \text{diag}\,(1, 1, 0, 0, 0, 0)$ and $R = \text{diag}\,(1\text{e}-4, 1\text{e}-4)$.

The resulting motions for the temporal and spatial references are illustrated in Fig. 5. The comparison clearly shows that the motion associated with the temporal reference deviates from the desired path. This occurs because, while the end-effector is blocked, the temporal reference continues to advance. Consequently, once the 2 s disturbance ends, the end-effector must "*catch up*", leading to the observed deviation. In contrast, the spatial reference remains stationary during the disturbance, allowing the motion to resume seamlessly without any deviations. This realization underpins the rise of path-following methods, a paradigm shift that overcomes the limitations of the temporal dependency. The following subsection delves deeper into this concept.

Path-following: A superior alternative to path-tracking

We have observed that tracking a spatial reference offers superior robustness against disturbances. This principle forms the foundation of path-following formulations, which shift the focus from a time-varying reference dictating *where to be when* to minimizing deviations from the reference path. In path-following, the velocity along the path becomes a secondary concern, adjustable to improve performance. In other words, the problem is no longer about adhering to a predefined time schedule but instead treats velocity as an additional degree of freedom for traversing the reference. The versatility of path-following

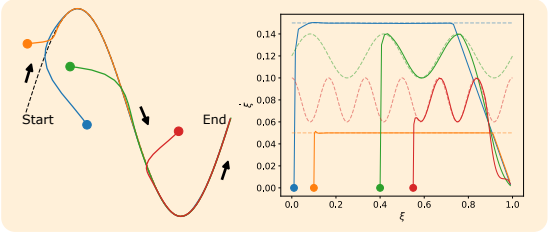


FIGURE 6: End-effector motions of a two-link robotic manipulator computed with an optimization-based path-following approach, trading off between orthogonal convergence to the path and achieving a predefined velocity profile. Trajectories are color-coded by initial conditions. The plot on the right shows traversal velocities, with desired reference velocities as dashed lines.

across a wide range of applications, combined with its independence from the inherent limitations of traditional reference tracking [25], explains the substantial attention it has garnered in the literature. Comprehensive overviews of existing approaches can be found in [11], [69].

To showcase the relevance of path-following, we build upon the previous reference-tracking example by formulating a path-following method. Instead of tracking a position reference, this approach tracks a desired velocity profile $\dot{\xi}_{\text{ref}}$ that determines how the end-effector traverses the reference path. To achieve this, we project the Cartesian coordinates of the end effector $[p_x, p_y]$ to the spatial coordinates $[\xi, \eta]$, namely the progress along the path and the orthogonal distance to it. As a consequence the new states of the robotic manipulator are $\boldsymbol{x} = [\xi, \eta, \dot{\xi}, \dot{\eta}, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$. The corresponding equations of motion for the spatially projected robotic manipulator are derived by taking the first derivative and combining eqs. (32) with the path-parameterized Cartesian motions in (S25).

For this case-study, we reuse the same NMPC formulation as before. However, the weights associated to the cost function of the NLP (34a) are chosen to trade-off between converging to the reference path $\lim_{t\to\infty} ||\eta(t)||_2 = 0$ and tracking a desired velocity profile $\lim_{t\to\infty} ||\dot{\xi}(t) - \dot{\xi}_{\text{ref}}(t)||_2 = 0$, such that $Q = Q_E = \text{diag}\,(0, 1\text{e}-1, 1, 1\text{e}-4, 0, 0, 0, 0)$ and $R = \text{diag}\,(1\text{e}-4, 1\text{e}-4)$.

To test the performance of the optimization-based path-following formulation, we initialize the end-effector at four distinct positions, each associated with a different velocity profile, characterized by different magnitudes, shapes and frequencies. The results depicted in Fig. 6 verify that, regardless of the initial state and the desired velocity profile, the position of the robotic manipulator converges to the geometric reference, both in the task space and the traversing speed.

This example raises the question of what the optimal

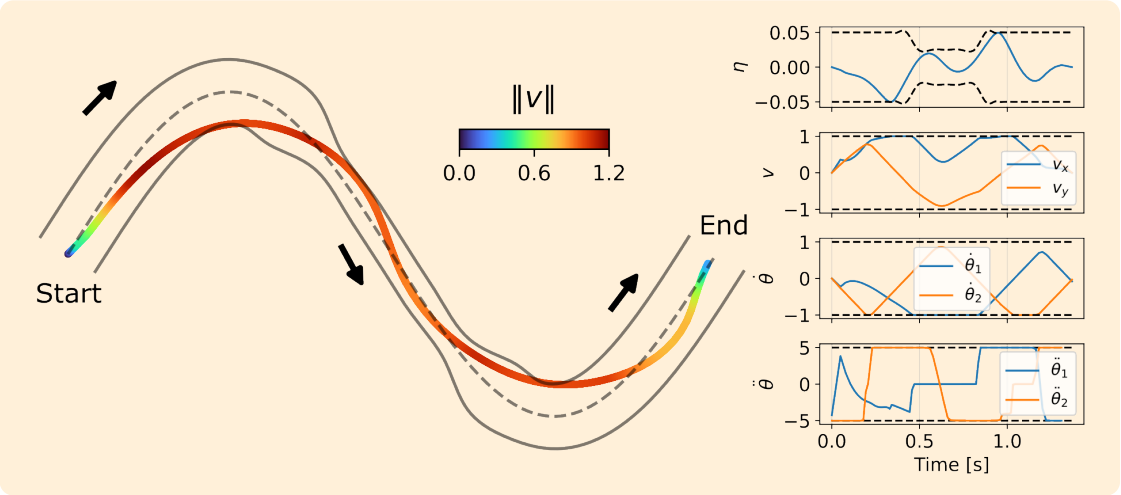


FIGURE 7: End-effector motions of a two-link robotic manipulator traversing a reference path within a corridor, representing the admissible deviation region, color-coded by velocity norm. On the right, from top to bottom: orthogonal spatial coordinate, end-effector velocity components, and joint angle velocities and accelerations, with bounds shown as black dashed lines.

velocity profile $\dot{\xi}_{\text{ref}}$ is and how it can be computed. To answer these questions, we turn to the upcoming subsection, where we showcase the capabilities enabled by path-parametric methods in a more generic context of motion planning, ultimately expanding the range of methods that benefit from the presented framework.

Motion planning: Spatial-awareness

In the previous subsections, we demonstrated how the path-parametric framework enables the design of controllers capable of achieving complex navigation behaviors, such as explicitly controlling the convergence rate to a reference path or maintaining a predetermined traversal velocity profile. Here, we aim to extend these examples to a more general motion planning scenario, providing a comprehensive view of the full potential of path-parametric methods.

To this end, we extend the previous two-link robotic manipulator case study by assuming that the sinusoidal reference path (33) is enclosed within a corridor $\mathcal{C}$, allowing the end-effector to navigate within this region. This scenario is commonly encountered in industrial robotics, where exact tracking of the reference is not required, and approaching the surrounding area within a desired tolerance is sufficient. Specifically, our goal is to find the time-optimal motion that takes advantage of this admissible region to move the robotic manipulator's end-effector from the start to the end of the reference path, while respecting both dynamical and spatial constraints.

We approach this as an offline motion planning problem, where — similar to [1], [2], [3] — we utilize the transformation from the temporal to the spatial domain, $\boldsymbol{x}(t) \to \boldsymbol{x}(\xi)$, to convert the time minimization problem into a finite horizon problem. In particular, the Optimal Control Problem (OCP) we solve is:

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \; T = \int_{\xi_0}^{\xi_f} \frac{1}{\dot{\xi}(\boldsymbol{x}(\xi))}\, d\xi \tag{35a}$$

$$\text{s.t.} \quad \boldsymbol{x}(\xi_0) = \boldsymbol{x_0}\,, \quad \boldsymbol{x}(\xi_f) = \boldsymbol{x}_f\,, \tag{35b}$$

$$\boldsymbol{x}'(\xi) = \frac{f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))}{\dot{\xi}(\boldsymbol{x}(\xi))}, \qquad \xi \in \left[\xi_0, \xi_f\right] \tag{35c}$$

$$\boldsymbol{x}(\xi) \in \mathcal{X}\,,\, \boldsymbol{u}(\xi) \in \mathcal{U}\,, \qquad \xi \in \left[\xi_0, \xi_f\right] \tag{35d}$$

$$\boldsymbol{x}(\xi) \in \mathcal{C}\,. \tag{35e}$$

Due to the transformation from the temporal to the spatial domain $\boldsymbol{x}(t) \to \boldsymbol{x}(\xi)$, the system dynamics in (35c) evolve according to path-parameter ξ, and thus, the resulting OCP (35) is a finite horizon problem, as opposed to the original time minimization problem. Besides that, we represent the end-effector by the same projected states as before, i.e., $\boldsymbol{x} = [\xi, \eta, \dot{\xi}, \dot{\eta}, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$ and $\boldsymbol{u} = [\ddot{\theta}_1, \ddot{\theta}_2]$. This enables to formulate the corridor bounds in (35e) as convex constraints in the orthogonal coordinate. For the specific planar case at hand, this simplifies to

$$\underline{\eta}(\xi) \leq \eta(\xi) \leq \overline{\eta}(\xi)\,, \tag{35f}$$

where $\underline{\eta} : \mathbb{R} \mapsto \mathbb{R}$ and $\overline{\eta} : \mathbb{R} \mapsto \mathbb{R}$ are C^2 parametric functions on ξ that describe the upper and lower bounds of the corridor. Readers interested in how such a parametric corridor example extends to 3D and unknown environments should refer to the following rubric, where we provide an in-depth explanation on how such corridors can be computed.

Beyond the spatial limitations inherent to the corridor geometry, we introduce further constraints on the configuration and task spaces, as defined in equation (35d), to ensure a more accurate representation of a realistic scenario. Specifically, these constraints are

$$-1 \leq \dot{\theta}(\xi) \leq 1\,, \quad -5 \leq \ddot{\theta}(\xi) \leq 5\,, \tag{35g}$$

$$-1 \leq v(\xi) \leq 1\,. \tag{35h}$$

Following a similar approach to the one described in (34), we transform the optimal control problem (OCP) defined in (35) into a nonlinear program (NLP) by discretizing it with the multiple shooting approach into N nodes. We then solve the resulting NLP using the IPOPT solver [70].

The computed end-effector motions are illustrated in the left panel of Fig. 9, where the color gradient corresponds to the magnitude of the velocity. The right panel displays the evolution of the associated states and inputs. Given the behavior incentivized by the time-optimal formulation, the trajectory fully exploits the actuation by seeking the optimal trade-off between speed and spatial bounds. This phenomenon is observable from two different perspectives: Firstly, both the end-effector's trajectory and the orthogonal spatial coordinate η dynamically adapt to the narrowing section of the corridor, showcasing the end-effector's ability to stay within the confines while leveraging the available space. Secondly, the bottom plot reveals that at least one joint acceleration remains saturated during most of the navigation, akin to the bang-bang behavior associated with time-optimal motions.

The specific time-optimal cost function, robotic manipulator dynamics, and corridor formulations chosen for this example serve as a proof of concept, illustrating how path-parametric methods offer a compelling framework for achieving agile motion while ensuring safety guarantees.

MINIMIZING TIME OR MAXIMIZING PROGRESS?

After exploring the fundamental advantages that have propelled the rise of the path-parametric paradigm, we shift our focus to one of its most notable applications: time-optimal navigation. Time-optimal control tackles a fundamental challenge in control theory: steering a dynamic system from an initial state to a desired final state in minimal time [71]. During the past few decades, this problem has attracted extensive research attention due to its wide-ranging applications—from military operations like missile interception to robotic navigation, rapid manufacturing changeovers, and optimized logistics routing [72]. Despite decades of theoretical advancement, a comprehensive mathematical framework for solving minimum-time control problems remains elusive.

The evolution of minimum-time trajectory optimization can be viewed as a progression from conceptual insight to theoretical formalization and, finally, to computational realization [73]. It began with the brachistochrone problem, which introduced the principle of time-optimality through the calculus of variations. This was later generalized by Pontryagin's Maximum Principle (PMP), an *indirect method* that provides necessary conditions for optimality but often leads to difficult two-point boundary value problems. To overcome these challenges, *direct optimization* methods emerged, replacing analytical optimality conditions with discretization and large-scale nonlinear programming, thereby enabling tractable solutions for complex, constrained systems. This trajectory from the brachistochrone to PMP and ultimately to direct methods highlights the minimum-time problem's shift from theoretical elegance to practical applicability.

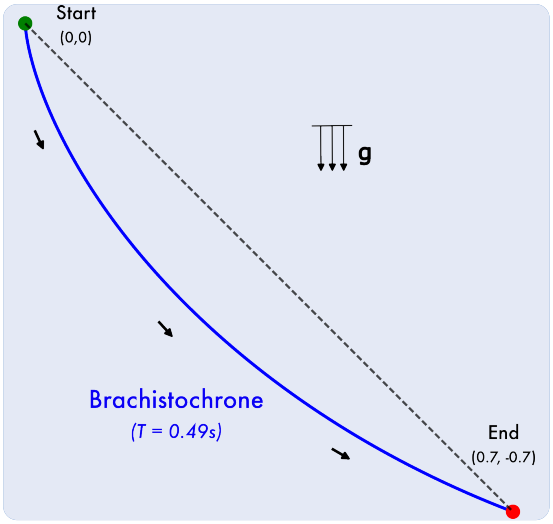


FIGURE 8: The brachistochrone problem asks for the curve of fastest descent between two points under gravity. As the first minimum-time problem, it also marks the earliest use of path-parameterization for trajectory calculation. Its solution is the cycloid, shown in blue.

Building on this perspective, we argue that the interplay between *time minimization* and *progress maximization* has been present since the inception of the field and remains central to its development. To make this connection clear, we organize our discussion into three stages: the origin of the problem (the brachistochrone), which first introduced the principle of time-optimal navigation; its formal standardization through Pontryagin's Maximum Principle, which generalized the problem into a rigorous theoretical framework; and the rise of modern direct methods, which provide computational tools capable of addressing complex dynamical systems. Together, these stages illustrate how the core dilemma of minimizing time versus maximizing progress has guided both the historical trajectory and present-day practice of trajectory optimization.

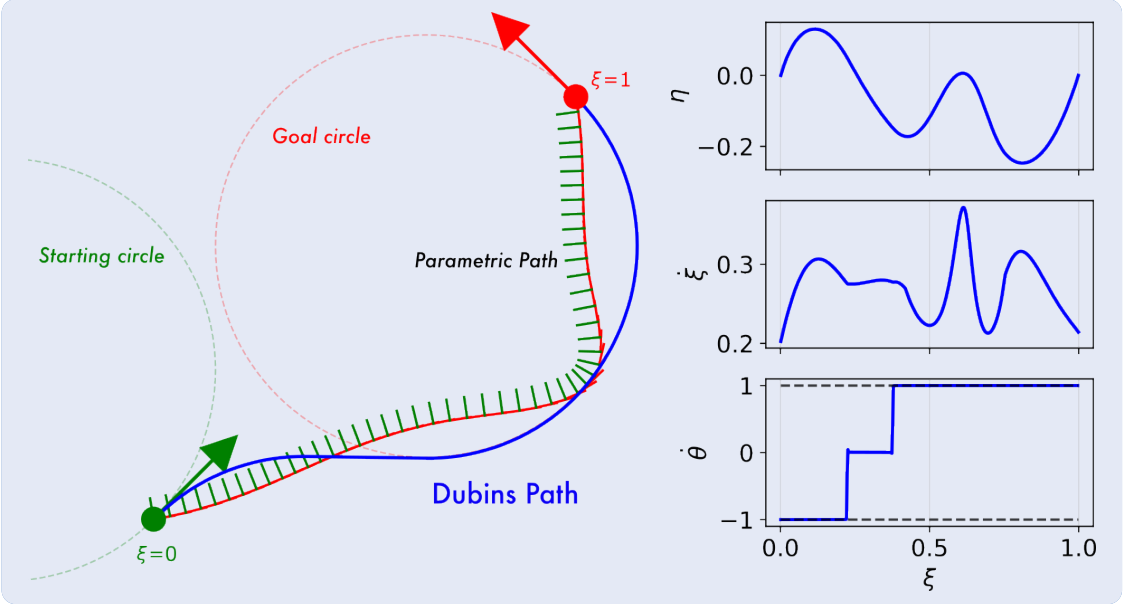


FIGURE 9: Path-parametric solution of the Dubins minimum-time problem. The framework exactly recovers the analytical Dubins trajectory (blue), with states and inputs shown on the right.

Brachistrochrone or fastest descent: The origins of the Minimum Time Problem

Brachistochrone—from the Greek for "shortest time"—is regarded as the first minimum-time problem formally addressed in a scientific setting. Posed as a challenge by Johann Bernoulli in 1696, it was solved independently by several leading scientists of the era. The presented solutions shaped the development of the calculus of variations and laid important foundations for classical mechanics and beyond. We highlight it here for two reasons. First, it represents the earliest instance of minimum-time motion planning, making it the conceptual ancestor of virtually all subsequent approaches, including the path-parametric framework discussed in this work. Second, it can be seen as the first method to exploit the idea of projecting motion into a parametric representation, thereby simplifying the analysis and enabling a closed-form solution.

The brachistochrone problem consists in determining the curve along which a particle slides, under gravity, from one point to another at a lower height in the shortest possible time. Within the path-parametric framework introduced in this manuscript, we can interpret the sought curve as the parametric path $\gamma(\xi)$ that governs the particle's motion. The tangent $\boldsymbol{e}1(\xi(t))$ specifies the local slope, while the particle's velocity is $\boldsymbol{v}(t) = v(t), \boldsymbol{e}1(\xi(t))$. By conservation of energy, the velocity magnitude is given in closed form as $v(t) = \sqrt{2gy(t)}$, where $y(t)$ denotes the vertical displacement from the initial height. Combining this relation with the path-parameterization of Cartesian coordinates in (S25a), and assuming arc-length parameterization ($\sigma(\xi) = 1$) with no deviation ($\eta_1 = \eta_2 = 0$), we obtain $\dot{\xi} = \sqrt{2gy(t)}$. Transforming the problem into the temporal domain as in (35), the task of finding the optimal surface reduces to solving the unconstrained problem

$$\min y(\cdot) T = \int_{\xi_0}^{\xi_f} \frac{1}{\dot{\xi}(\boldsymbol{y}(\xi))}, d\xi = \int_{\xi_0}^{\xi_f} \frac{1}{\sqrt{2gy(\xi)}}, d\xi,$$

whose closed-form solution is the well-known cycloid trajectory shown in Fig. 8. In this sense, the classical derivation of the brachistochrone—originally obtained via classical mechanics and the calculus of variations—can be seen as an early and elegant instance of the path-parametric perspective developed in this work.

An alternative approach is to disregard the physical structure imposed by the previous solution and instead formulate the problem as a numerical optimization. This resembles (35), where the nominal path is a straight line between the two points, and the objective is to compute the optimal deviation $\eta(\xi)$ that minimizes travel time. While this may appear unnecessarily elaborate for the brachistochrone case, it highlights a powerful paradigm for more complex problems, where embedding physical knowledge is far less straightforward than in the brachistochrone. This capability to recast intricate minimum-time problems as optimization programs is the foundation of the direct methods introduced later, with path-parametric representations playing a key enabling role.

Pontyangrin's Maximum Principle: Standardizing the Minimum Time Problem

Following the conceptual birth of the minimum-time problem with the brachistochrone, the field transitioned from isolated case-studies to a unified theory with the advent of *Pontryagin's Maximum Principle (PMP)* in the 1950s [73], [74]. Where the brachistochrone exemplified the first instance of trajectory optimization under gravity, PMP generalized this reasoning to arbitrary dynamical systems, establishing the first systematic recipe to address minimum-time problems across aerospace, robotics, and control. In this sense, PMP served as the historical bridge between the classical analytical formulations of the 17th–18th centuries and the modern optimization-based approaches that dominate today.

In generic terms, the PMP recipe prescribes the following steps: one defines the system dynamics $\dot{x} = f(x, u)$ with the objective of minimizing the final time T; constructs the Hamiltonian $H(x, u, \lambda) = \lambda^\top f(x, u) + 1$; derives the costate dynamics $\dot{\lambda} = -\partial H / \partial x$; enforces the maximum principle by choosing at each instant $u^*(t) = \arg\min_u H(x, u, \lambda)$; and finally solves the resulting two-point boundary value problem in (x, λ).

To illustrate the equivalence –in the context of minimum time problems– between PMP and the path-parametric framework developed in this paper, we consider the *Dubins path problem* [75], [76]. The system state is $\boldsymbol{x} = [p_x, p_y, \theta]$, where (p_x, p_y) denote the Cartesian coordinates and θ the heading. The control input is $u = \dot{\theta}$, and the dynamics are given by $\dot{\boldsymbol{x}} = [v\cos\theta, v\sin\theta, \dot{\theta}]$ with constant forward speed v. The objective is to minimize travel time subject to the curvature constraint $|\dot{\theta}| \leq \dot{\theta}_{\max}$. Applying PMP shows that the optimal solution consists of piecewise-constant controls that switch between straight motion and maximum-rate turns, yielding circular arcs connected by straight segments.

In the path-parametric formulation, the vehicle's progression is instead expressed relative to a parametric path, with the state written as $\boldsymbol{x} = [\xi, \eta, \theta]$. This reformulation, analogous to (35), converts the infinite-horizon problem into a finite-horizon one amenable to numerical optimization. The resulting trajectory is illustrated in Fig. 9, coinciding with the analytical Dubins solution. Moreover, the bottom plot confirms that the recovered control profile in $\dot{\theta}$ is again bang–bang, precisely as predicted by PMP.

These observations demonstrate that the path-parametric framework faithfully reproduces the classical PMP solutions while offering a more flexible computational route. Whereas PMP set the theoretical standard for analyzing minimum-time problems, the path-parametric perspective provides a unifying and practical means to recover these solutions, and it readily extends to systems where a direct application of PMP may be conceptually cumbersome or numerically involved.

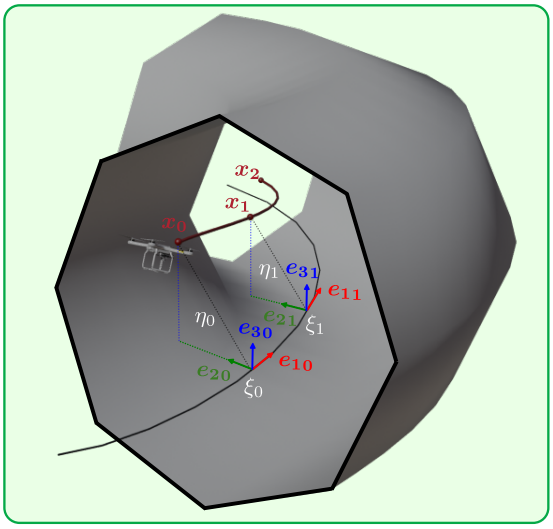


FIGURE 10: Path-parametric methods use a moving frame relative to a reference path, employing spatial coordinates (ξ and η) to create *spatially aware* planning and control algorithms. These exploit environmental geometry, ensuring safety by constraining the orthogonal coordinate η within an admissible *corridor*. We highlight how the corridor generation method from [77] integrates with this framework.

Direct Optimization for Minimum-Time Control: A dilemma between tractability and optimality

Unlike other approaches, precomputing time-optimal trajectories offline –as done in the previous two subsections– for subsequent online tracking proves inadequate. Real-world implementations inevitably encounter noise and model uncertainties, causing deviations from the reference trajectory. Once the system deviates, the remaining path loses its time-optimality property. Continuing to follow such a trajectory not only compromises time efficiency but also risks system failure, as time-optimal motions operate at the physical limits of the system where any overshoot can have severe consequences [51].

To overcome these limitations, researchers have developed low-level controllers that directly compute approximately time-optimal motions without relying on offline references. These controllers employ progress maximization—a technique that optimizes the system's advancement along a geometric reference path [7], [46], [43], [9]. As shown in *Time Minimization vs. Progress Maximization* this approach closely approximates true time-optimal motions while offering significant advantages in problem formulation and practical implementation. The path-parametric framework presented in this manuscript provides the necessary tools to implement these progress maximization techniques, ultimately, reducing the barrier to solve complex minimum-time problems.

Time-Minimization vs Progress-Maximization

Progress maximization has emerged as a leading approach for minimum-time motion control. Driven by advancements in numerical optimization and embedded solvers, progress maximization based prediction-based controllers have shown very promising results in real-world applications [7], [46], [43], [9]. These controllers are built on path-parametric methods, allowing systems to operate near their performance limits and achieve behavior that closely approximates time-optimality. Additionally, these formulations leverage the capacity to easily impose collision-free constraints, which would otherwise be non-convex or difficult to enforce without the path-parametric structure [48]. The combination of these attributes has made progress maximization the preferred approach for achieving agile performance along a designated reference path. However, while progress maximization serves as an approximation to time minimization, the precise quantification of the gap between these two methods remains unresolved. In this study, we aim to shed some light on this question by performing an experimental comparison of both approaches.

An illustrative case-study: A miniature racing car

As an exemplary system upon which we can test both control formulations, we choose a 1:43 miniature racecar. The state vector, defined as $\boldsymbol{x} = [X, Y, \varphi, v_x, v_y, r, d, \delta]$, represents the car's position, orientation, linear and angular velocities, throttle position, and steering angle, respectively. The control inputs, throttle rate and steering rate, are denoted by $\boldsymbol{u} = [\dot{d}, \dot{\delta}]$. The equations of motion for these states are given by

$$\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi), \tag{S36a}$$
$$\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi), \tag{S36b}$$
$$\dot{\varphi} = r, \tag{S36c}$$
$$\dot{v}_x = \frac{1}{m}\left(F_{r,x} + F_{\text{fric}} - F_{f,y}\sin(\delta) + m v_y r\right), \tag{S36d}$$
$$\dot{v}_y = \frac{1}{m}\left(F_{r,y} - F_{f,y}\cos(\delta) - m\, v_x r\right), \tag{S36e}$$
$$\dot{r} = \frac{1}{I_z}\left(F_{f,y} l_f \cos(\delta) - F_{r,y} l_r\right), \tag{S36f}$$

whose tire forces are expressed as

$$F_{f,y} = D_f \sin(C_f \arctan(B_f \alpha_f)), \tag{S37a}$$
$$F_{r,y} = D_r \sin(C_r \arctan(B_r \alpha_r)), \tag{S37b}$$
$$F_{r,x} = (C_{m1} - C_{m2} v_x) d, \tag{S37c}$$
$$F_{\text{fric}} = -C_r - C_d v_x^2, \tag{S37d}$$

where α_f and α_r are the front and rear slip angles:

$$\alpha_f = -\arctan\left(\frac{r\, l_f + v_y}{v_x}\right) + \delta, \tag{S37e}$$
$$\alpha_r = \arctan\left(\frac{r\, l_r - v_y}{v_x}\right). \tag{S37f}$$

All parameters involved in this model are summarized in the following table:

Parameter	*Value*	*Description*
m	Mass	$0.041\,\text{kg}$
I_z	Inertia	$27.8 \times 10^{-6}\,\text{kg}\,\text{m}^2$
$[C_{m1}, C_{m2}]$	Motor params.	$[0.287, 0.545]$
$[B_f, C_f, D_f]$	Front tire params.	$[2.579, 1.269, 0.192]$
$[B_r, C_r, D_r]$	Rear tire params.	$[3.385, 1.269, 0.174]$
$[l_r, l_f]$	Dist. to front/rear axle	$[0.029, 0.033]\text{cm}$

Time-Minimization

To compute time-optimal motions that are dynamically feasible, we formulate a predictive controller that solves an NLP with a cost function that solely minimizes time. We implement this approach using a multiple shooting method, where time is incorporated as a decision variable, as detailed below:

$$\min_{\substack{x_{0,\dots,N} \\ u_{0,\dots,N-1} \\ dt_{0,\dots,N}}} \; T = \sum_{k=0}^{N} dt_k \tag{S38a}$$
$$\text{s.t.} \quad x_0 = x_i,\; x_N = x_f, \tag{S38b}$$
$$x_{k+1} = f(x_k, u_k, dt_k), \quad k = [0, \dots, N-1] \tag{S38c}$$
$$c(x_k, u_k) \geq 0, \quad k = [0, \dots, N-1] \tag{S38d}$$
$$dt_k \geq 0, \quad k = [0, \dots, N-1] \tag{S38e}$$

where f in (S38c) are the nonlinear dynamics in (S36) In contrast to standard MPC, the horizon shrinks with each step, meaning that N decreases as the system progresses along the trajectory. To achieve the *true* time-optimal solution, as defined in equation (S38a), we focus exclusively on minimizing time without incorporating any additional stabilizing or regularizing terms. Developing a solver capable of reliably addressing the nonlinear programming problem (NLP) defined in equation (S38) at a receding horizon and at high rates necessitates careful attention to the underlying numerical methods and requires considerable expertise. Despite recent advancements in the field [78], [79], there is no reliable solver available to tackle NLP (S38) in its most general form. For this experiment, we employ a Sequential L1 Quadratic Programming (SL1QP) approach [80], akin to the methodology used in [81].

Progress-Maximization

A widely used approach to approximate time-optimal trajectories is to maximize progress along the path. To achieve this, the progress needs to be represented as a system state. As illustrated in Fig. 1, this can be done by either *augmenting* the state vector – $x^{\Gamma}(t) = [x(t), \tilde{\xi}(t)]$ – or *projecting* it – $x^{\Gamma}(t) = [\tilde{\xi}(t), \eta(t)]$ [7], [9]. Including progress as a state variable enables the design of an optimization-based predictive controller, where the cost function seeks to maximize progress along the path while satisfying dynamic and feasibility constraints. This setup encourages the system to traverse the path as quickly as possible within feasible limits, resulting in highly agile motions that closely approximate time-optimality.

Despite numerous attempts, the trajectories produced by these methods are limited in optimality due to several key issues:

1) Conceptually, minimizing time and maximizing progress are not equivalent goals, and thus lead to fundamentally different trajectories.
2) The absence of a Hessian in the cost function necessitates a stabilizing or regularizing term to ensure numerical stability. A common approach is to lightly penalize the derivative of the inputs, but this ultimately compromises the primary objective.
3) When progress is introduced as a virtual variable to augment the state space, a synchronization term is needed to align the system dynamics with the virtual kinematic chain, thereby penalizing lag errors. This further detracts from achieving the true objective.

Since these limitations are inherent across this family of methods, we select contouring control as the representative approach for progress maximization [7], [46]. This choice is justified by its success as the first formulation to achieve near time-optimal motion control in racing contexts. Although it is based on *state augmentation*, the insights from our forthcoming experiments are also applicable to other progress maximization methods, including those that employ *projection* techniques [43], [9]. In particular, the NLP problem underlying the contouring controller is:

$$\min_{\substack{x_{0,\dots,N},u_{0,\dots,N-1}\\ \xi_{0,\dots,N},\dot{\xi}_{0,\dots,N-1}}} \sum_{k=0}^{N-1} -\dot{\xi}_k + g(x_k,u_k,\xi_k,\dot{\xi}_k) \tag{S39a}$$

$$\text{s.t.} \quad x_0 = x_i\,, \quad \xi_0 = \xi_i\,, \tag{S39b}$$

$$x_{k+1} = f(x_k,u_k,dt_k), \quad k=[0,\dots,N-1] \tag{S39c}$$

$$\xi_{k+1} = \Xi(\xi_k,\dot{\xi}_k), \quad k=[0,\dots,N-1] \tag{S39d}$$

$$c(x_k,u_k) \geq 0, \quad k=[0,\dots,N-1] \tag{S39e}$$

$$\xi_f \geq \xi_k \geq \xi_i, \quad k=[0,\dots,N-1] \tag{S39f}$$

$$\dot{\xi}_k \geq 0, \quad k=[0,\dots,N-1] \tag{S39g}$$

where g in (S39a) is a representative function for bringing numerical stability to the problem, either with some regularization or smoothening of the inputs and Ξ is a kinematic integration that relates the progress variable with its velocity.

To incite agile behavior, we formulate the cost function (S39a) akin to the aforementioned contouring control formulation [7] by maximizing the velocity of the progress variable, i.e., $\min -\dot{\xi}$. However, it is important to note that the approach may vary based on the particular formulation used. Alternatives include directly maximizing the progress variable $-\xi$ [14], or regulating over a target velocity profile by minimizing $(\xi - \xi_{\text{ref}})^2$ [43], [9].

Comparison without model mismatch

We begin by comparing the ***time-minimization*** and ***progress-maximization*** controllers in an ideal scenario, where the model used by the controller perfectly matches the real system. Fig. 11 presents the trajectories generated by both controllers. While both trajectories appear quite similar, the time-minimization formulation completes the task $0.437\,\text{s}$ faster.

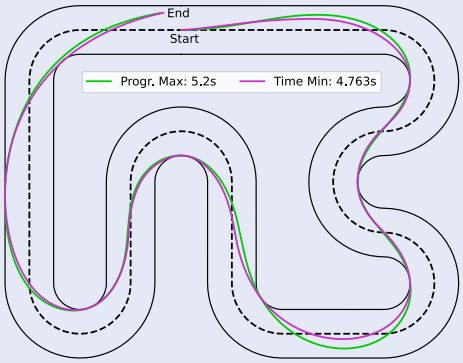


FIGURE 11: Comparison between time-minimization and progress-maximization without model mismatch.

To better understand the source of this discrepancy, we refer to Fig. 12, which shows the rear tire forces. These forces represent the interaction between the car and the road, serving as a key indicator of how close the car is to its dynamic limits. Specifically, we examine the rear tire's friction circle. The dashed line marks the tire's grip limit; if the applied force exceeds this limit, the car will lose traction and begin to slide. It is clear that the minimum time controller, unlike the progress maximization controller, pushes the tire to this limit, maximizing the available force without causing a slide. This aggressive utilization of grip explains why the minimum time controller achieves faster performance.

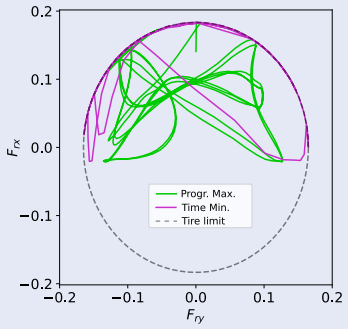


FIGURE 12: Friction circle for the rear tire without model mismatch. Tire limit is given by the dashed line.

Comparison with deterministic model mismatch

In the previous subsection, we observed that, without model mismatch, the minimum-time controller outperforms progress maximization. However, real-world systems are affected by disturbances and uncertainties. In this subsection, we explore whether this finding holds when model mismatch is introduced.

We begin by introducing a deterministic mismatch in the tire model by reducing the maximum lateral force the tire can generate. This reduction is quantified by the parameter r, representing the percentage decrease in the coefficient D of the tire model (S37), such that $\tilde{D} = (1 - r)\, D$. This mismatch can be applied to either the front or the rear tire. The following figure illustrates the lateral tire forces for various values of r:

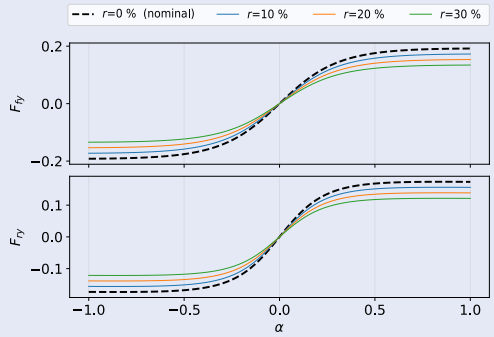


FIGURE 13: Tire model with reduction term r.

Given a slip angle α, the force the tire can generate decreases as r increases. Introducing this mismatch in the front tire induces understeer, while applying it to the rear tire causes oversteer. To illustrate these effects, Fig. 14 shows the understeering and oversteering behavior resulting from the mismatch reduction of 20%:

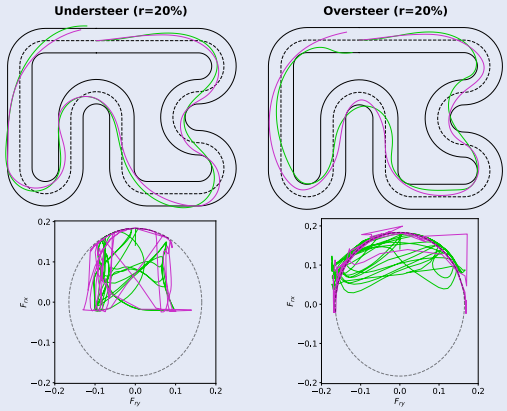


FIGURE 14: Understeering and oversteering trajectories and lateral rear tire forces with a mismatch of $r = 20\%$ in *time-minimization* and *progress-maximization*.

From Fig. 14, we observe a clear deviation from the nominal scenario without model mismatch in Fig. 11. In the understeering case, the rear tires slip in every corner, causing the car to drift outward, while in the oversteering case, the front wheels slip, leading the car to drift through the corner. This behavior is also reflected in the rear tire friction circles, shown in the bottom row of Fig. 14. In the understeering scenario, the front tires reach their limit before the rear tires, preventing the rear tires from achieving maximum grip. Conversely, in the oversteering scenario, the rear tires slip at each corner, pushing forces beyond the friction circle. For a clearer understanding of these dynamics, we encourage the reader to view the accompanying video, which animates these trajectories

With the impact of mismatch on the tire model understood, we now aim to quantify its effect on the controllers. To do this, we analyze the lap times achieved by both controllers under various reductions in the front and rear tires:

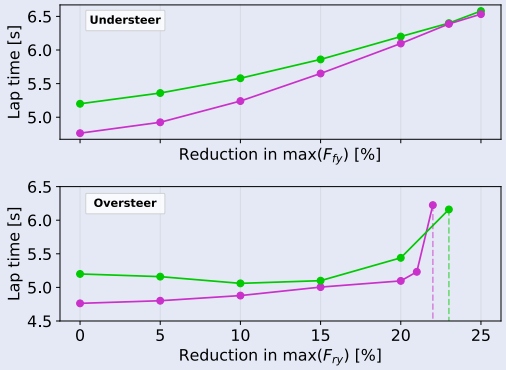


FIGURE 15: Laptimes for multiple tire mismatch reduction values r for understeering and oversteering with *time-minimization* and *progress-maximization*. The dashed lines represent the tire reduction upon which the car spins.

In both scenarios, the minimum-time controller outperforms progress maximization; however, this performance difference narrows as the mismatch increases. The minimum-time controller drives the system toward its theoretical limits, but as mismatch grows, the gap between the theoretical model and the actual system widens, causing greater discrepancies between the planned and actual trajectories. In contrast, the progress maximization controller's cost function balances lag error, regularization, and progress maximization—all encapsulated in the g term in the cost function (S39a)—which inherently makes it more conservative. This suboptimality becomes beneficial as mismatch increases, as its commands are less aggressive. This effect is particularly evident in the oversteer scenario: while the minimum-time controller outperforms progress maximization up to a critical mismatch level ($r_r = 21.5\%$), it eventually causes the car to spin, indicated by the green dashed line. In contrast, the progress maximization controller's less aggressive maneuvers delay the onset of spinning to a higher mismatch level ($r_r = 23\%$), shown by the magenta dashed line.

Comparison with stochastic model mismatch

In the previous subsection, we examined each controller's performance under a deterministic mismatch. However, in real-world scenarios, the discrepancies between theoretical and actual models are typically non-deterministic and arise from multiple sources, each with unique behaviors.

To address this, we evaluate the controllers in the presence of a non-deterministic model mismatch. For this purpose, we sample the reduction factor from a normal distribution, defined as $r = \mathcal{N}(0, \sigma)$, where σ represents the standard deviation of the distribution. In addition, we leverage the insights from the previous subsection to prevent non-representative edge cases where the car spins. We do this by bounding the maximum reduction to $r \leq 20\%$. Ultimately, the resulting model of the tire forces is as follows:

$$\tilde{D}_f = 1 - \min(|\mathcal{N}_f(0, \sigma)|, 0.2) D_f \,, \tag{S40a}$$

$$\tilde{D}_r = 1 - \min(|\mathcal{N}_r(0, \sigma)|, 0.2) D_r \,. \tag{S40b}$$

To assess the performance of the controllers under the stochastic model mismatch described in equation (S40), we evaluated the controllers for various standard deviations, σ, ranging from 0.1 to 0.5. For each value of σ, we conducted 15 experiments, providing a thorough representation of the controllers' performance under stochastic mismatch.

The resulting lap times are depicted in Fig. 16. In both controllers, lap times increase as σ increases. Notably, the time gap between the two methods remains constant. Furthermore, since the reduction is constrained to the admissible range—where the car is not at risk of spinning—the time-minimization controller consistently outperforms the progress-maximization controller.

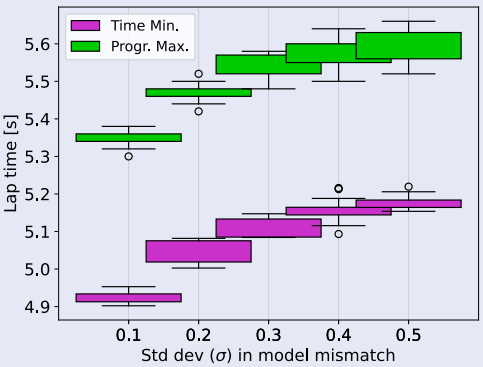


FIGURE 16: Lap times for a stochastic mismatch, where the reductions at the front and rear tires are obtained from a normal distributions, evaluated for multiple standard deviations. Each case study consists of 15 runs.

Discussion

Generally, minimizing time is preferable, particularly when the model mismatch is restricted to a permissible range. However, if the mismatch extends beyond this range, maximizing progress proves to be more reliable. Additionally, the numerical implementation of progress maximization aligns with existing numerical solvers, avoiding the complex numerics needed for solving the min time problem, which demands a specially designed solver.

PATH-PARAMETRIC CORRIDORS: A CONTINUOUS SPATIAL REPRESENTATION

As discussed throughout this manuscript, spatial coordinates derived from path-parametric methods inherently capture the concept of advancement along a path while enabling the imposition of spatial bounds as convex constraints in the orthogonal components of the spatial states (see Fig. 10). These features make path-parametric methods a compelling toolset for planning and control algorithms in navigation. For instance, in the motion planning example of the robotic manipulator shown in Fig. 9, explicitly representing the admissible region allows the system to efficiently utilize the available space.. Similarly, in the racing scenario, where minimizing time and maximizing progress were compared, an explicit representation of the racetrack enabled controllers to fully exploit the road's width for optimal performance.

These examples highlight the potential of parameterizing a system's motion using spatial coordinates. However, they also raise a critical question: *how can one formulate and compute a spatial representation that effectively describes the admissible region around an arbitrary reference path as a function of the orthogonal spatial component, η?* This question is central to bringing path-parametric methods beyond controlled environments into the real world.

Existing methods based on convex decomposition [82], [83], [84], [85], which partition free space into convex polyhedra, are incompatible with such formulations. These approaches adhere to a *Euclidean* perspective and fail to leverage the capability of imposing convex constraints specifically in the orthogonal spatial coordinate η. Moreover, discretizing free space into multiple convex sets introduces the *polyhedron allocation problem*, where each state must be preassigned to a specific polyhedron. This discretization disrupts the differentiability of the corridors, making it challenging to incorporate collision-free corridors into optimization and learning frameworks.

To overcome these limitations, [77] presented a method for generating *differentiable, continuous and collision-free corridors*. We proceed to detail its key components.

Differentiable Parametric Collision-Free Corridors

Motived by the incompatibility of convex decomposition based corridors for parametric formulations, [77] proposed a method for computing corridors that are differentiable, continuous and collision-free. We now outline its main ingredients and crucial role in the path-parametric framework.

Choosing an off-centered ellipse as the cross-section

A parametric corridor is defined as a predefined cross-section that sweeps the parametric path given by the user. For this reason, choosing a cross section that offers maximum adaptability to the obstacle-free space and ensures differentiability and computational feasibility is of utmost importance.

The simplest option is a circle, which offers a single degree of freedom (the radius). An ellipse increases this number to three, and allowing it to have an offset from the path further raises the total degrees of freedom to 5. Beyond the circular or elliptical choices, there are more elaborate options, such as polyhedrons or semialgebraic sets. However, these cross sections cannot guarantee to remain differentiable or closed throughout the entire corridor.

For this reason, the most favorable option within the feasible ones is a rotating off-centered ellipse. Mathematically, this ellipse is defined by the matrix E and the offset $\boldsymbol{p}_E$, and its equation is given by

$$(\boldsymbol{x}_\perp - \boldsymbol{p}_E)^\mathsf{T}\mathrm{E}(\boldsymbol{x}_\perp - \boldsymbol{p}_E) \leq 1\,. \tag{S41}$$

Parameterizing the ellipse with polynomials

Having defined the cross section as a rotating off-centered ellipse, the next step is to parameterize it, so that it sweeps the reference path from the beginning to the end. This implies that the ellipse, and thereby the matrix E and the offset $\boldsymbol{p}_E$, evolve according to path-parameter ξ. For this purpose, these variables are chosen to be parameterized by Chebyshev polynomials:

$$\{\mathrm{E}(\xi), \boldsymbol{p}_E(\xi)\} \to \{\mathrm{E}(\xi, \boldsymbol{c}_E), \boldsymbol{p}(\xi, \boldsymbol{c}_P)\} \tag{S42}$$

where $\boldsymbol{c}_E$ and $\boldsymbol{p}_E$ are the coefficients of the polynomials. There are two reasons underpinning the choice of Chebyshev polynomials: Firstly, utilizing polynomials ensures that the resulting corridors exhibit inherent smoothness and differentiability. Secondly, the Chebyshev basis guarantees that our method is numerically stable, even for high polynomial degrees. Intuitively, increasing the degree of the polynomial enhances the corridor's ability to adapt to variations along the reference path.

Introducing the polynomial parameterization in (S42) into the off-centered ellipse cross section in (S41)

$$(\boldsymbol{x}_\perp - \boldsymbol{p}_E(\xi, \boldsymbol{c}_P))^\mathsf{T}\mathrm{E}(\xi, \boldsymbol{c}_E)(\boldsymbol{x}_\perp - \boldsymbol{p}_E(\xi, \boldsymbol{c}_P)) \leq 1\,.$$

and removing the nonlinearities leads to

$$\boldsymbol{x}_\perp^\mathsf{T}\mathrm{E}(\xi, \boldsymbol{c}_E)\boldsymbol{x}_\perp - \boldsymbol{d}(\xi, \boldsymbol{d}_E)\boldsymbol{x}_\perp \leq 1\,, \tag{S43}$$

where $\boldsymbol{d}(\xi, \boldsymbol{c}_D)$ maintains the offset of the ellipse, while removing the nonlinearities of (S42). From these decisions, it becomes evident that the shape and size of the corridor are entirely determined by the polynomial coefficients $\boldsymbol{c}_E$ and $\boldsymbol{d}_E$. This naturally raises the question: how can these coefficients be determined? To address this, we turn to the third and final component of the methodology.

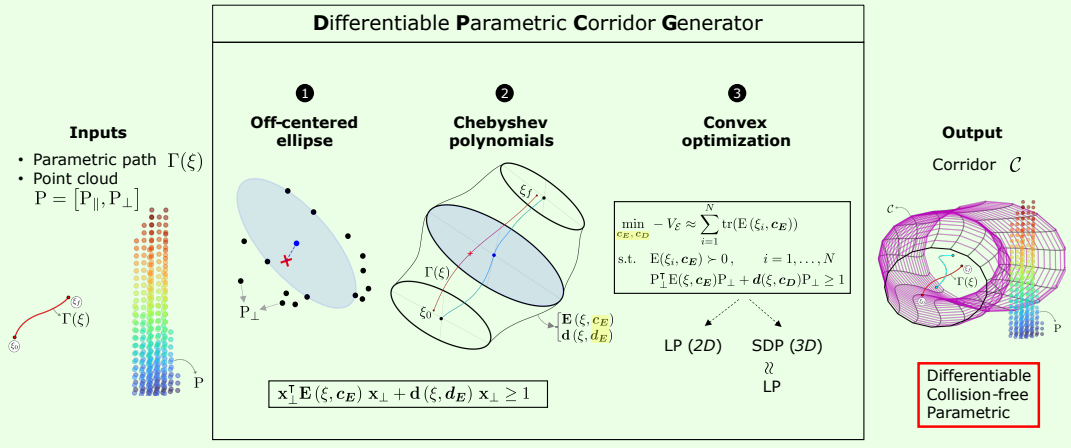


FIGURE 17: An overview of the method introduced in [77] for computing differentiable, continuous, and parametric corridors fully compliant with the presented path-parametric framework. The method requires only a parametric path and a point cloud to generate the corridors, offering a system- and environment-agnostic tool that extends the applicability of path-parametric methods to real-world scenarios beyond well-defined and controlled environments. It achieves this through three key components: (1) selecting an off-centered ellipse as the corridor's cross-section, (2) parameterizing the cross-section using Chebyshev polynomials, and (3) formulating the corridor volume maximization problem as a lightweight convex optimization task solvable in real-time.

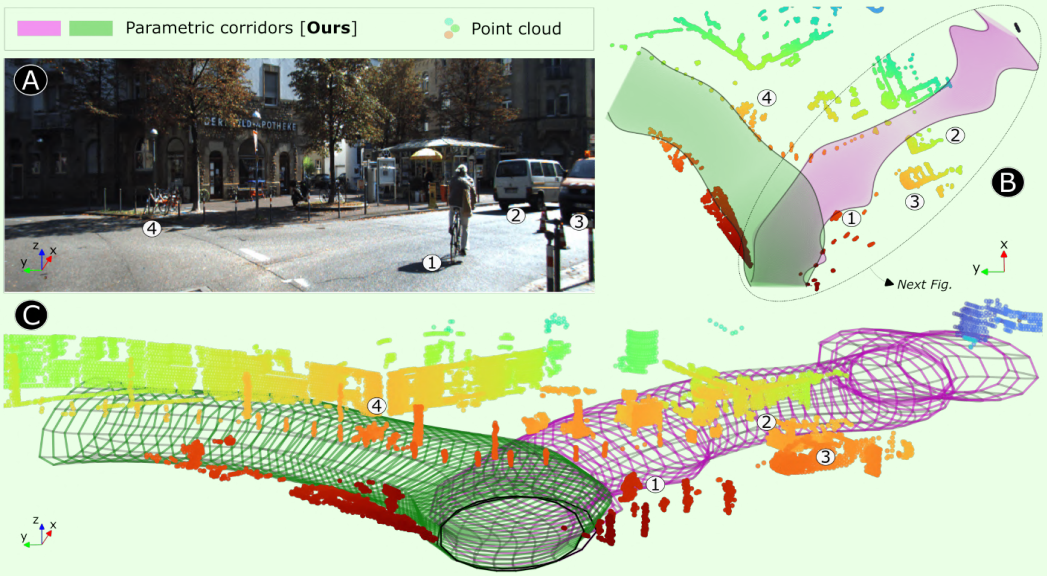


FIGURE 18: 3D path-parametric corridors (degree 9) from [77] in a KITTI Vision Benchmark [86] scene with narrow bifurcating lanes, vehicles, and cyclists. Panels: (A) RGB view, (B) top-down, (C) isometric. Point cloud and corridors are color-coded by camera proximity; white markers (1–4) denote key features. Dashed corridor in (B) detailed in Fig. 19.

Corridor volume maximization as convex optimization

From eq. (S43), it is apparent that the corridor's shape and size is exclusively dependent on the polynomial coefficients $\boldsymbol{c}_E$ and $\boldsymbol{d}_E$. From the available collision-free corridors, we aim to identify the one with the largest volume. To determine the coefficients of this corridor, we formulate an optimization problem that maximizes the volume, ensuring that the ellipse matrix E remains positive definite throughout the corridor, and that all points in the point cloud $\mathrm{P}_\perp$ lie outside of it. Given that the area of the ellipse in (S41) is given by $A_\mathcal{E} = \pi/\sqrt{\det \mathrm{E}}$, the optimization problem that we solve is:

$$\max_{\boldsymbol{c}_E, \boldsymbol{d}_E} V_\mathcal{E} = \int_{\xi_0}^{\xi_f} -\det \mathrm{E}(\xi, \boldsymbol{c}_E)\, d\xi \tag{S44a}$$

$$\text{s.t.} \quad \mathrm{E}(\xi, \boldsymbol{c}_E) \succ 0\,,$$

$$\mathrm{P}_\perp^\intercal \mathrm{E}(\xi, \boldsymbol{c}_E)\mathrm{P}_\perp - \boldsymbol{d}(\xi, \boldsymbol{d}_E) \leq 1\,.$$

To make this problem computationally tractable, we conduct the following approximations. Firstly, we discretize the continuous parts of the optimization problem into N evaluations. Secondly, we avoid the nonlinearities associated with the determinant of the ellipse by approximating it with the trace. Thirdly, we introduce a wrapper around the reference path, ensuring that the problem always remains bounded. By incorporating all three modifications, the original problem becomes a convex optimization problem:

$$\min_{\boldsymbol{c}_E, \boldsymbol{d}_E} -V_\mathcal{E} \approx \sum_{i=1}^{N} \mathrm{tr}(\mathrm{E}\,(\xi_i, \boldsymbol{c}_E)) \tag{S45a}$$

$$\text{s.t.} \quad \mathrm{E}(\xi_i, \boldsymbol{c}_E) \succ 0\,, \tag{S45b}$$

$$\mathrm{P}_\perp^\intercal \mathrm{E}(\xi_i, \boldsymbol{c}_E)\mathrm{P}_\perp - \boldsymbol{d}(\xi_i, \boldsymbol{d}_E) \leq 1 \tag{S45c}$$

More specifically, the optimization problem in (S45) is a Semidefinite-Program (SDP). Despite being a convex problem, SDPs are the most difficult convex problems to solve. To overcome this burden, the linear matrix inequality in (S45b) is replaced with diagonal dominance, i.e.,

$$\mathrm{E} = \begin{bmatrix} \mathrm{E}_{11} & \mathrm{E}_{12} \\ \mathrm{E}_{12} & \mathrm{E}_{22} \end{bmatrix} \succ 0 \quad \rightarrow \quad \mathrm{E}_{11}, \mathrm{E}_{22} >> \mathrm{E}_{12}\,. \tag{S46}$$

This approximation reduces the SDP into a Linear Program (LP). LPs are very well understood and extremely lightweight, and thus, can be very efficiently solved with off-the shelf solvers. As a final remark, notice that, in the planar case there is no need for the diagonal dominance approximation in (S46), since the original problem is already an LP.

Accounting for uncertainty in the point cloud

The derivations above assume an exact point cloud. In practice, however, point clouds are inherently uncertain. Here, we emphasize that parametric corridors are not only well suited for path-parametric methods but also exhibit particularly favorable properties under uncertainty. This is due to two main reasons. First, by parameterizing the cross-section as an ellipse, the construction naturally accommodates uncertainty in the data. Second, in high-uncertainty settings where the point cloud is very noisy, the corridor generation problem in (S45) may become infeasible. Fortunately, for convex problems, infeasibility can be detected efficiently through the homogeneous self-dual embedding method [87], a feature exploited by most off-the-shelf solvers [88].

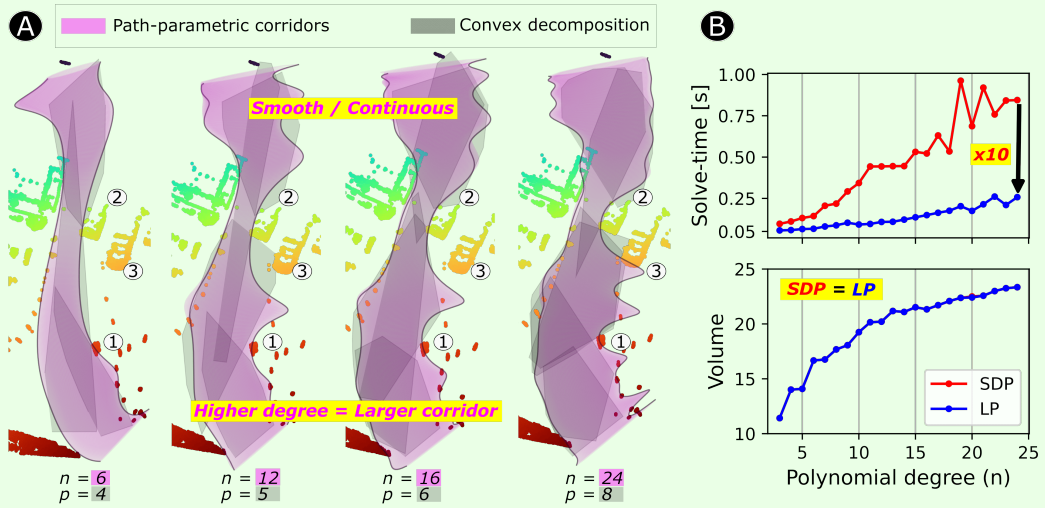


FIGURE 19: A detailed analysis of the real-world scenario shown in Fig. 18. Panel (A) presents a top view of the corridors, generated by incrementally increasing the polynomial degree n, alongside the corridors obtained by decomposing the free space into p convex sets, represented in gray. Panel (B) illustrates the evolution of the corridor volume and computation times as a function of the polynomial degree, for both the SDP and LP problems.

Experiment: An autonomous driving case-study

The method's real-world applicability is demonstrated through an autonomous driving case study of a roundabout, where two corridors are computed. Figure 18 shows these corridors alongside a lidar-generated point cloud and a raw RGB camera image. The corridors effectively capture the free space, spanning the entire road width while accommodating vehicles, bicycles, and road boundaries. The analysis examines three key aspects of the method.

First, to better understand the influence of polynomial degree on the shape and size of the corridors, we conduct evaluations across various degrees. The resulting solutions are presented in Fig. 19-A, where top-down views of the corridors, obtained by sequentially increasing the polynomial degree n, are shown. It is clear from these images that higher polynomial degrees enhance the expressiveness of the corridors. This trend is further illustrated in the plot at the bottom of panel (B), which demonstrates that an increase in polynomial degree corresponds to a growth in the corridor's volume. Second, to illustrate the comparison between the path-parametric corridor generation method and the well-established convex decomposition, Fig. 19-A is considered once more. The path-parametric corridors are depicted alongside the corridors resulting from decomposing the free space into p convex sets. The results show that both methods encompass very similar spaces in all four cases. However, the hyperparameters differ significantly. In convex decomposition, the quantity and distribution of polyhedra are determined based on the number of segments within a precomputed linear path, which may compromise the corridor's volume if the reference path has few segments. In contrast, this method employs the polynomial degree n as its unique hyperparameter, while remaining agnostic to the underlying reference. Furthermore, Figure 19-A also illustrates how a parametric cross section with a polynomial basis results in a continuous and smooth space representation, contrasting with the discreteness intrinsic to convex decomposition.

Third, to assess the trade-off between volume gains and computational cost associated with increasing the polynomial degree (n), Figure 19-B is examined. This figure illustrates the volumes and solve times for the pink corridor depicted in Figure 18, across polynomial degrees ranging from $n = 3$ to $n = 25$. It compares both the original SDP formulation and the approximated LP relaxation. Notably, the results demonstrate that the LP relaxation achieves corridor volumes identical to the original SDP while yielding computational speedups of up to a factor of 10. This enhanced performance stems from two key factors. First, the cost function (S45a), which maximizes the trace, inherently promotes diagonally dominant matrices in the resulting corridors. Second, optimizing the offset relative to a reference overcomes the constraint of requiring a diagonally dominant matrix E. This allows the resulting corridors to fully encompass the available space while preserving the diagonally dominant structure within their underlying matrices.

These results demonstrate that this corridor generation method can compute differentiable, continuous, and smooth parametric corridors at 5–20 Hz in unstructured arbitrary environments, making it suitable for real-time deployment alongside the various planning and control techniques of the presented path-parametric framework.

CONCLUSIONS

Path-parametric methods have emerged as a cornerstone in egocentric decision-making, owing to three key advantages: they inherently model progress along a path, incorporate geometric features such as curvature and torsion into system dynamics, and enable spatial bounds to be expressed as convex constraints on orthogonal spatial states. These attributes have made path-parametric formulations highly effective in planar scenarios, such as autonomous driving. However, extending these methods to real-world three-dimensional cases—where curves are defined by both curvature and torsion—has proven challenging. This subtle yet critical difference has limited their applicability to complex 3D problems, including aerial navigation and robotic manipulator control.

Existing approaches to path-parametric problems are often presented as isolated works, resulting in a fragmented literature where techniques appear disconnected. This disjointed presentation obscures the underlying relationships between methods, leaving readers with an incomplete understanding of the field. To address this, we proposed a universal formulation for path-parametric planning and control, demonstrating how these approaches are fundamentally interconnected.

For this purpose, we analyzed the path-parametric problem from multiple yet interrelated perspectives. First, we examined the *interplay of existing parametric techniques* and showed how they can be unified under a single framework composed of two key components: (i) a *path-parameterization technique* and (ii) a *spatial representation of system dynamics*.

To illustrate how these components enable the formulation of planning and control strategies, we applied them to a two-link robotic manipulator in two distinct contexts. First, we focused on *low-level control*, exploring the foundational motivations behind the development of path-parametric methods. Second, we demonstrated how this framework extends to more versatile optimization-based *motion planning* scenarios.

Next, we delved into one of the most popular instantiations of this framework: the minimum-time problem. Following its origins in the Brachistochrone challenge and its standardization through Pontryagin's Maximum Principle—demonstrated numerically with Dubins' path—we focused on *progress maximization*, examining it as an approximation to time-optimal behavior.. Using a miniature racing car example, we highlighted the differences between these two formulations. Although progress maximization serves as a proxy for the original time-minimization problem, experimental results showed that it achieves comparable lap times while offering additional advantages. Notably, it simplifies implementation and facilitates the incorporation of spatial constraints by imposing convex bounds on the orthogonal distance to the road's centerline.

Finally, to generalize this approach for navigating safe corridors in diverse environments—beyond cases where a road is explicitly defined—we focused on representing the admissible environment around the parametric path. Specifically, we introduced a method for generating differentiable, continuous, and collision-free *corridors*. These corridors, thanks to their properties, are fully compatible with any gradient-based path-parametric method, whether used as constraints or as environmental information to guide the optimization algorithm.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Rudolf Reiter, Ángel Romero, and Jelena Trisovic for their invaluable insights and engaging discussions on path-parametric methods and their diverse applications in robotics and autonomous systems. Special thanks also go to Jon Santamaría for his assistance in creating the graphics for the cover figure.

AUTHOR INFORMATION

Jon Arrizabalaga (jon.arrizabalaga@tum.de) is pursuing his PhD at the Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, and is a visiting researcher at the Robotics Institute of Carnegie Mellon University. His research focuses on the convergence of numerical optimization and control theory, particularly in their applications to robotics and autonomous systems.

Zbyněk Šír (zbynek.sir@mff.cuni.cz) is an associate professor at the Mathematical Institute in the Faculty of Mathematics and Physics of Charles University in Prague. He received a PhD in mathematics with specialization in history of French geometry, both from Charles University of Prague. His research interests include CAGD, theoretical differential geometry, history of geometry and other applied geometric fields.

Zachary Manchester (zmanches@andrew.cmu.edu) is an assistant professor in the Robotics Institute at Carnegie Mellon University and founder of the Robotic Exploration Lab. He received a PhD in aerospace engineering in 2015 and a BS in applied physics in 2009, both from Cornell University. His research interests include control and optimization with applications to aerospace and robotic systems.

Markus Ryll (markus.ryll@tum.de) is an assistant professor in the Department of Aerospace and Geodesy at the Technical University of Munich and head of the Autonomous Aerial Systems Lab. He received a PhD in 2014 from Max Planck Institute for Biological Cybernetics. Between 2014 and 2017, he was a postdoctoral researcher at the Laboratory for Analysis and Architecture of Systems (LAAS-CNRS). From 2018 to 2020, he continued his postdoctoral work in the Robust Robotics Group at the Massachusetts Institute of Technology (MIT, CSAIL). His

research focuses on enabling autonomous aerial robots to interact with real-world environments.

REFERENCES

[1] R. Verschueren, N. van Duijkeren, J. Swevers, and M. Diehl, "Time-optimal motion planning for n-dof robot manipulators using a path-parametric system reformulation," in *2016 American Control Conference (ACC)*, pp. 2092–2097, IEEE, 2016.

[2] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2017.

[3] J. Arrizabalaga and M. Ryll, "Sctomp: Spatially constrained time-optimal motion planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4827–4834, IEEE, 2023.

[4] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1205–1212, IEEE, 2021.

[5] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[6] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[7] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[8] T. Oelerich, F. Beck, C. Hartl-Nesic, and A. Kugi, "Boundmpc: Cartesian trajectory planning with error bounds based on model predictive control in the joint space," *arXiv preprint arXiv:2401.05057*, 2024.

[9] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 4044–4050, IEEE, 2022.

[10] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 6137–6142, IEEE, 2010.

[11] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2015.

[12] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[13] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC)*, pp. 477–482, IEEE, 2016.

[14] J. Arrizabalaga and M. Ryll, "Spatial motion planning with pythagorean hodograph curves," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 2047–2053, IEEE, 2022.

[15] "The history of curvature." https://web.archive.org/web/20071106083431/http://www3.villanova.edu/maple/misc/history_of_curvature/k.htm. Accessed: 2024-05-27.

[16] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," in *1983 American Control Conference*, pp. 752–756, IEEE, 1983.

[17] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.

[18] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, 1987.

[19] W. L. Nelson and I. J. Cox, "Local path control for an autonomous vehicle," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 1504–1510, IEEE, 1988.

[20] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 384–389, IEEE, 1990.

[21] I. J. Cox, "Blanche-an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Transactions on robotics and automation*, vol. 7, no. 2, pp. 193–204, 1991.

[22] J. Hauser and R. Hindman, "Maneuver regulation from trajectory tracking: Feedback linearizable systems," *IFAC Proceedings Volumes*, vol. 28, no. 14, pp. 595–600, 1995.

[23] R. Skjetne, T. I. Fossen, and P. V. Kokotović, "Robust output maneuvering for a class of nonlinear systems," *Automatica*, vol. 40, no. 3, pp. 373–383, 2004.

[24] K. D. Do, Z.-P. Jiang, and J. Pan, "Robust adaptive path following of underactuated ships," *Automatica*, vol. 40, no. 6, pp. 929–944, 2004.

[25] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic, "Path-following for nonminimum phase systems removes performance limitations," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 234–239, 2005.

[26] A. P. Aguiar and J. P. Hespanha, "Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty," *IEEE transactions on automatic control*, vol. 52, no. 8, pp. 1362–1379, 2007.

[27] E. W. Justh and P. Krishnaprasad, "Equilibria and steering laws for planar formations," *Systems & control letters*, vol. 52, no. 1, pp. 25–38, 2004.

[28] F. Zhang, E. W. Justh, and P. S. Krishnaprasad, "Boundary following using gyroscopic control," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 5, pp. 5204–5209, IEEE, 2004.

[29] E. W. Justh and P. Krishnaprasad, "Natural frames and interacting particles in three dimensions," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 2841–2846, IEEE, 2005.

[30] M. Breivik and T. I. Fossen, "Principles of guidance-based path following in 2d and 3d," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 627–634, IEEE, 2005.

[31] E. W. Justh and P. Krishnaprasad, "Steering laws for motion camouflage," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 462, no. 2076, pp. 3629–3643, 2006.

[32] P. Reddy, E. W. Justh, and P. Krishnaprasad, "Motion camouflage in three dimensions," in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 3327–3332, IEEE, 2006.

[33] L. Lapierre, D. Soetanto, and A. Pascoal, "Nonsingular path following control of a unicycle in the presence of parametric modelling uncertainties," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 16, no. 10, pp. 485–503, 2006.

[34] R. Rysdyk, "Unmanned aerial vehicle path following for target observation in wind," *Journal of guidance, control, and dynamics*, vol. 29, no. 5, pp. 1092–1100, 2006.

[35] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 8642–8647, IEEE, 2009.

[36] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager, "Optimal control of formula 1 race cars in a vdrift based virtual environment," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11907–11912, 2011.

[37] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proceedings of the 11th international symposium on advanced vehicle control*, no. 2, pp. 1–6, 2012.

[38] T. Faulwasser, J. Matschek, P. Zometa, and R. Findeisen, "Predictive path-following control: Concept and implementation for an industrial robot," in *2013 IEEE International Conference on Control Applications (CCA)*, pp. 128–133, IEEE, 2013.

[39] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *2013 European Control Conference (ECC)*, pp. 4136–4141, IEEE, 2013.

[40] M. Böck and A. Kugi, "Real-time nonlinear model predictive path-following control of a laboratory tower crane," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1461–1473, 2013.

[41] S. Kumar and R. Gill, "Path following for quadrotors," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 2075–2081, IEEE, 2017.

[42] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive

contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[43] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14324–14329, 2020.

[44] R. Reiter, M. Kirchengast, D. Watzenig, and M. Diehl, "Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 99–106, 2021.

[45] J. Ji, X. Zhou, C. Xu, and F. Gao, "Cmpcc: Corridor-based model predictive contouring control for aggressive drone flight," in *Experimental Robotics: The 17th International Symposium*, pp. 37–46, Springer, 2021.

[46] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022.

[47] J. Arrizabalaga and M. Ryll, "Pose-following with dual quaternions," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 5959–5966, IEEE, 2023.

[48] R. Reiter, A. Nurkanović, J. Frey, and M. Diehl, "Frenet-cartesian model representations for automotive obstacle avoidance within nonlinear mpc," *European Journal of Control*, vol. 74, p. 100847, 2023.

[49] T. Fork and F. Borrelli, "Euclidean and non-euclidean trajectory optimization approaches for quadrotor racing," *arXiv preprint arXiv:2309.07262*, 2023.

[50] M. Krinner, A. Romero, L. Bauersfeld, M. Zeilinger, A. Carron, and D. Scaramuzza, "Time-optimal flight with safety constraints and data-driven dynamics," *arXiv preprint arXiv:2403.17551*, 2024.

[51] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[52] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 1934–1940, IEEE, 2019.

[53] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[54] H. Wang, J. Kearney, and K. Atkinson, "Arc-length parameterized spline curves for real-time simulation," in *Proc. 5th International Conference on Curves and Surfaces*, vol. 387396, 2002.

[55] F. L. Markley and J. L. Crassidis, "Correction to: Fundamentals of spacecraft attitude determination and control," in *Fundamentals of spacecraft attitude determination and control*, pp. C1–C7, Springer, 2014.

[56] R. L. Bishop, "There is more than one way to frame a curve," *The American Mathematical Monthly*, vol. 82, no. 3, pp. 246–251, 1975.

[57] R. T. Farouki, *Pythagorean—Hodograph Curves*. Springer, 2008.

[58] H. I. Choi and C. Y. Han, "Euler–rodrigues frames on spatial pythagorean-hodograph curves," *Computer Aided Geometric Design*, vol. 19, no. 8, pp. 603–620, 2002.

[59] J. Arrizabalaga, F. Vega, Z. Šír, Z. Manchester, and M. Ryll, "Phodcos: Pythagorean hodograph-based differentiable coordinate system," in *2025 IEEE Aerospace Conference*, pp. 1–15, IEEE, 2025.

[60] A. J. Hanson and H. Ma, "Parallel transport approach to curve framing," *Indiana University, Techreports-TR425*, vol. 11, pp. 3–7, 1995.

[61] W. Wang, B. Jüttler, D. Zheng, and Y. Liu, "Computation of rotation minimizing frames," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 1, pp. 1–18, 2008.

[62] G. Wanner and E. Hairer, *Solving ordinary differential equations II*, vol. 375. Springer Berlin Heidelberg New York, 1996.

[63] D. J. Struik, *Lectures on classical differential geometry*. Courier Corporation, 1961.

[64] E. Abbena, S. Salamon, and A. Gray, *Modern differential geometry of curves and surfaces with Mathematica*. Chapman and Hall/CRC, 2017.

[65] Z. Šír and B. Jüttler, "c^2 hermite interpolation by pythagorean hodograph space curves," *Mathematics of Computation*, vol. 76, no. 259, pp. 1373–1391, 2007.

[66] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[67] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl, "Towards a modular software package for embedded optimization," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 374–380, 2018.

[68] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[69] N. Hung, F. Rego, J. Quintas, J. Cruz, M. Jacinto, D. Souto, A. Potes, L. Sebastiao, and A. Pascoal, "A review of path following control strategies for autonomous robotic vehicles: Theory, simulations, and experiments," *Journal of Field Robotics*, vol. 40, no. 3, pp. 747–779, 2023.

[70] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[71] M. Athans and P. L. Falb, *Optimal control: an introduction to the theory and its applications*. Courier Corporation, 2007.

[72] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[73] H. J. Sussmann and J. C. Willems, "300 years of optimal control: from the brachystochrone to the maximum principle," *IEEE Control Systems Magazine*, vol. 17, no. 3, pp. 32–44, 2002.

[74] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[75] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[76] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[77] J. Arrizabalaga, Z. Manchester, and M. Ryll, "Differentiable collision-free parametric corridors," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1839–1846, IEEE, 2024.

[78] D. Kiessling, C. Vanaret, A. Astudillo, W. Decre, and J. Swevers, "An almost feasible sequential linear programming algorithm," *arXiv preprint arXiv:2401.13840*, 2024.

[79] L. Yang, T. Marcucci, P. A. Parrilo, and R. Tedrake, "A new semidefinite relaxation for linear and piecewise-affine optimal control with time scaling,"

[80] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[81] D. Kiessling, A. Zanelli, A. Nurkanović, J. Gillis, M. Diehl, M. Zeilinger, G. Pipeleers, and J. Swevers, "A feasible sequential linear programming algorithm with application to time-optimal path planning problems," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 1196–1203, IEEE, 2022.

[82] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, pp. 109–124, Springer, 2015.

[83] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[84] X. Zhong, Y. Wu, D. Wang, Q. Wang, C. Xu, and F. Gao, "Generating large convex polytopes directly on point clouds," *arXiv preprint arXiv:2010.08744*, 2020.

[85] C. Toumieh and A. Lambert, "Voxel-grid based convex decomposition of 3d space for safe corridor generation," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 4, p. 87, 2022.

[86] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[87] Y. Ye, M. J. Todd, and S. Mizuno, "An $o(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm," *Mathematics of operations research*, vol. 19, no. 1, pp. 53–67, 1994.

[88] P. J. Goulart and Y. Chen, "Clarabel: An interior-point solver for conic programs with quadratic objectives," *arXiv preprint arXiv:2405.12762*, 2024.

Part IV

Résumé

We conclude the dissertation by reflecting on the journey undertaken and outlining the path forward. In Chapter 9, we revisit the key contributions of this work, highlighting the central ideas and results developed throughout the dissertation. Chapter 10 identifies open questions and promising directions for future research, emphasizing challenges that are particularly promising and important for further exploration. Finally, Chapter 11 distills the overarching insights of this work into a set of concluding remarks that underscore its broader implications and potential impact.

9 Summary

In this dissertation, we focus on the development of decision-making algorithms for autonomous systems from an egocentric perspective. This inquiry is motivated by the persistent gap in agility, reliability, and efficiency between biological and robotic systems. While various aspects of this disparity have received attention, a crucial but often overlooked distinction is that animals inherently make decisions from a first-person—egocentric—viewpoint, whereas most robotic systems operate from a third-person, fixed reference frame. This dissertation addresses this discrepancy by systematically examining egocentricity: how to define a local viewpoint, how to transform motion representations from a fixed to a local frame, and how to exploit this perspective in designing decision-making algorithms such as low-level controllers and motion planners.

Before delving into these technical contributions, we first evaluate whether adopting an egocentric perspective is indeed beneficial. To this end, in Part I, we introduce a representative problem that serves as a testbed to validate our hypothesis: that decision-making algorithms rooted in egocentricity yield superior performance compared to their allocentric counterparts. Specifically, we examine the problem of minimum-time traversal through a tunnel using a quadrotor. This task is deliberately chosen for its alignment with the key properties we wish to evaluate. First, the tunnel represents a safe, collision-free corridor, providing the opportunity to assess the spatial exploitation enabled by egocentric awareness. Second, the time-optimal objective enables us to quantify the agility of the resulting control strategies. Third, the choice of a quadrotor—a highly agile platform—amplifies the relevance of spatial representation. These characteristics collectively provide a robust experimental setting to assess the efficacy of egocentric design. To this end, we develop an egocentric MPC strategy based on progress maximization tailored to quadrotors.

The results of this case study reveal three key advantages of the egocentric MPC approach. First, despite relying entirely on real-time computation, the controller consistently produces trajectories that closely approximate offline time-optimal solutions, even across diverse tunnel geometries. This is achieved by embedding spatial features directly into the optimization process via egocentric encoding. Second, the controller demonstrates strong robustness in dynamic environments: it maintains high performance even when tunnel disturbances occur outside the predictive horizon and adapts flexibly when changes fall within it, provided feasible solutions exist. Third, the same control law proves effective over a wide range of dynamic regimes—from high-speed motion exceeding 40 m/s to precise stop-and-go maneuvers—without requiring re-tuning. This versatility highlights its potential for broad, plug-and-play deployment in autonomy tasks. Together, these findings support the viability of the egocentric paradigm in real-world robotics applications.

Encouraged by these results, we then explore how to generalize the egocentric approach beyond this specific case study. Although the tunnel-traversal task demonstrates the paradigm's promise, it is narrowly scoped: it assumes a specific robot morphology, a predefined environment, and a time-optimal cost function. Extending egocentric methods to other domains introduces conceptual challenges that must be addressed.

In Part II, we tackle these challenges by identifying and formalizing the key components necessary for generalization. First, we derive a framework for representing the motion of Cartesian coordinates along a path with an associated arbitrary local frame, making no assumptions about the parameterizations of either. Next, we present a method for parameterizing the local frame itself, overcoming the singularities and unwanted twists inherent to the Frenet-Serret frame used in the initial quadrotor example. These contributions render the egocentric formulation agnostic to system morphology and reference path structure, thereby making it applicable to a wide array of systems. All proposed methods for local frame construction—notably the selected PTF—are at least twice differentiable, ensuring compatibility with gradient-based decision-making tools such as optimization and learning. Finally, we develop a differentiable representation of the free space surrounding the reference path using parametric corridors. This real-time spatial representation complements the previous contributions by enabling egocentric reasoning about environmental constraints.

In Part III, we demonstrate the practicality and versatility of these methods through three case studies addressing different layers of the autonomy stack. These case studies range from theoretical to highly application-oriented. The first example illustrates how singularity- and twist-free local frames (as introduced in Chapter 5) enable reference-following control strategies for simultaneous position and orientation—i.e., pose—control. The second example combines the spatial transformation techniques of Section 3.3, the generalized spatial parameterization from Chapter 4, and local frame generation from Chapter 5 to develop a time-optimal motion planner for navigating cluttered environments. The third example integrates local frame computation with domain-specific insights to produce a full-stack pipeline—from planning to control—capable of transporting liquids with robotic manipulators in a slosh-free manner, thereby preventing spillage.

Following these examples, we unify egocentric methods from the literature under a common framework built on the concepts developed in Part II. This unified perspective reveals that egocentric strategies, rather than being a set of isolated techniques, can be understood through a general framework comprising two essential ingredients: a method to represent system dynamics in a local frame, and a method to parameterize the motion of that frame. Both components are comprehensively addressed in Part II, and thus provide a solid foundation for designing decision-making systems that exploit egocentric representations. The resulting framework, supported by a diverse set of examples, demonstrates the broad applicability and conceptual coherence of egocentricity as a guiding principle for robotic autonomy.

10 Outlook

This dissertation has established a foundation for incorporating egocentricity into decision-making processes. Nonetheless, several important questions remain unresolved. In this chapter, we outline some of the most pressing challenges, with the hope that, through the collective efforts of the broader research community, these questions will ultimately be addressed.

10.1 Numerical Solvers for Egocentric Decision-Making

Which numerical methods best embed egocentricity into trajectory optimization?

The advantages of incorporating egocentricity come with the challenge of highly nonlinear projections that map system states from a global frame to a moving relative frame. As a consequence, the resulting optimization problems no longer benefit from the appealing properties of convex optimization—such as solution guarantees, independence from initial guesses, and bounded solve times. There are two principal avenues for carefully addressing these non-convexities:

On one hand, leveraging tools from differential geometry [65] and manifold optimization [66] appears promising, as the search process is naturally conducted within the SE(3) space where the local frame resides. This approach may help avoid infeasible local solutions—commonly encountered in sequential optimization methods—and could even facilitate the discovery of global optima.

On the other hand, recent developments in Sum-of-Squares (SoS) [67] and Semidefinite Programming (SDP) [68], which suggest that certain non-convex problems can be approximated by larger convex programs, offer an alternative path. These techniques open up the possibility of bringing the benefits of convex optimization into egocentric trajectory planning. In particular, exploring the intersection of Semidefinite Relaxation (SDR) techniques with the proposed egocentric optimization framework presents an exciting research direction. Such a synthesis could combine the performance advantages of egocentric formulations with the theoretical guarantees of convex optimization, paving the way for trajectory optimization methods that offer both extreme agility and provable safety guarantees.

10.2 Combined discrete and continuous search

How can egocentric trajectory optimization methods be combined with motion planning under a single formulation?

The convexification of the egocentric trajectory optimization approaches outlined in the previous paragraph raises the question of how these methods could be merged with a higher

level motion planning formulation. In fact, defining a single formulation that simultaneously searches for a geometrically feasible path and computes a dynamically feasible (optimal) trajectory to traverse it is central to the grand mission of unifying perception, planning and control.

Until very recently, the standard way to do this was in two separate stages – such in Chapter 7–, resulting in natural issues that arise from decoupling. This is not longer necessary, since a recent algorithmic breakthrough –denoted as Graph of Convex Sets (GCS)–, has unified the forever divided fields of discrete topological sampling-based planning and continuous trajectory optimization [69, 70, 71].

Merging the advantages of egocentric approaches with the capabilities of GCS to simultaneously solve path-planning and trajectory optimization problems enables the integration of a global perspective into traditionally local egocentric methods. Not only can egocentric methods benefit from GCS, but the rich environmental representation inherent to egocentricity might also enhance GCS, potentially leading to significant speed improvements over current implementations. This intersection is central to another question in the following subsection, and further details will be provided there.

10.3 Continuous State Space Representations

Spatial representation underpins decision-making algorithms by defining the feasible state space, essential for ensuring safety and improving algorithmic efficiency. This space is typically the union of the collision-free workspace and the dynamically feasible state space. The choice of representation is critical, as it affects how effectively algorithms explore the solution space. The standard approach involves decomposing this region into convex polyhedra [58, 59, 72, 73], creating a discrete representation and introducing the *polyhedron allocation problem*, where a system state must be assigned to its corresponding polyhedron. However, this discretization disrupts the spatial representation's differentiability, complicating its integration into optimization and learning frameworks. Additionally, while hyperplane-based representations suit Cartesian coordinates, they fail to enforce collision-free convex constraints in the egocentric domain.

To overcome these limitations, in Chapter 6 we have proposed differentiable parametric collision-free corridors [4], a novel mathematical formulation that represents the space in a continuous and smooth fashion, without the need for discretization. Consequently, this method eliminates the need for polyhedron allocation and seamlessly integrates with gradient-based optimization and learning techniques. The preliminary findings in Chapter 6 only scratch the surface of this representation's potential. Consequently, there is a clear need to conduct an in depth exploration of the autonomy capabilities –from safety and interpretability to performance– enabled by representing the space in a continuous and smooth manner. In particular, the required research effort could be organized into the following research questions:

What is the most suitable space representation for such egocentric navigation formulations?

Egocentric approaches define the feasible region as the area surrounding a reference parametric path, making the proper definition of this space crucial. Computing this region presents several challenges, requiring lightweight formulations that can be updated faster than the rate of dynamic environmental changes while ensuring compatibility with the underlying

numerical methods and the available sensor suite. To address this, it would be of great interest exploring mathematical spatial representations—such as convex decomposition [58], polytopic action spaces [74], and parametric corridors [4]—that integrate effectively with the egocentric methods.

Can these corridors provide transparency and safety guarantees for AI-driven control?

In the context of learning-based methods –such as RL or Behavior Cloning–, I identify two interesting research directions worth pursuing: First, these corridors can be used to enforce safety by constraining policy-predicted actions to remain within them. Since they explicitly represent high-reward areas, they not only provide hard safety guarantees, but also have the potential to accelerate the training process, leading to more efficient training pipelines. Second, beyond safety enforcement and training speed, these corridors can be used to imbue the policy with deep spatial awareness. For instance, the policy could be explicitly rewarded for navigating corridors with a desired minimum volume or curvature. Motivated by these possibilities, it is of great interest exploring methods –such as the Implicit Function Theorem [75]– that allow for embedding these corridors into neural networks. More importantly, there is a need to understand the role that these corridors play in enhancing the transparency and safety of learning-based methods for autonomous and robotic systems.

Which are the most appropriate numerical methods for computing such corridors?

Representing space via corridors requires not only selecting an appropriate mathematical formulation but also solving an optimization problem to maximize the corridor volume. Since this problem must be solved faster than the environment changes, computational speed is critical. To address this, the corridor generation method proposed in Chapter 6 was designed as a convex optimization problem, benefiting from the desirable properties of convexity described before. Specifically, the volume maximization problem was initially formulated as a Semidefinite Program (SDP), for which we developed a relaxation that reduced it to a Linear Program (LP). Building on this foundation, I am now interested in exploring two key challenges:

- *Understanding the tightness of the relaxation*: While the relaxation demonstrated experimental tightness—showing no optimality gap in all tested cases—a rigorous theoretical analysis is needed to confirm this property. Establishing the theoretical tightness would not only deepen our understanding but also provide the assurances necessary for deploying the method in safety-critical applications.
- *Parallelizing the computation of the corridors*: Motivated by the application of these corridors in motion planning, there is a need to investigate how parallelization can enable the simultaneous computation of multiple corridors. This involves reformulating the volume maximization problem in such a way that it is compatible with recent numerical methods capable of solving optimization problems efficiently on GPUs [76, 77, 78].

How can we leverage differentiable parametric corridors to unify planning and control?

As mentioned earlier, one promising direction is to explore how these continuous corridors can be applied within the Graph of Convex Sets (GCS). Despite being a paradigm shift in the fields of planning and motion control, most of GCS's potential remains to be unlocked. Its main reason is its reliance on a convex set representation, whose computation with current convex decomposition techniques is cumbersome and computationally heavy [58, 73], reducing the method's applicability to tasks where the environment is time-invariant.

I believe that the continuous space representation in Chapter 6 can be a key enabler to overcome this burden. Specifically, these corridors have a higher capacity to accurately express the environment, which could reduce the number of necessary convex sets. Additionally, a proper formulation of the volume maximization problem, could be faster than the currently used convex decomposition methods. The combination of these two factors could enable the widespread adoption of GCS into more dynamic and versatile robotic applications, ultimately leading to enhanced robotic skills in navigation, locomotion, and manipulation.

11 Conclusion

In this dissertation, I have presented the essential ingredients for formulating decision-making algorithms for autonomous systems from an egocentric—first-person—perspective. The original motivation behind this work has been to mimic the reasoning processes of humans and animals, with the hope of achieving similar levels of reliability, agility, and capability in autonomous robots.

While the foundations for this goal have been laid from an algorithmic standpoint, the true bottlenecks to reaching—or surpassing—the abilities of biological counterparts remain elusive. It is likely that these capabilities extend beyond the scope of low-level decision-making, as addressed in this work.

Some may argue that the main limitation lies not in execution, but in intelligence: the system's ability to develop a deep, adaptive understanding of the world. Such understanding could enable effective behavior without the need to explicitly encode structure or domain-specific knowledge. In fact, certain phenomena may be inherently difficult, if not impossible, to represent explicitly. For instance, a quadruped robot that could anticipate the behavior of loose gravel beneath its feet might exhibit greater agility when descending uneven terrain. Likewise, a robotic manipulator attempting to pick up an open bag of M&M's faces the challenge of dealing with a large number of interacting elements—modeling the motion of each candy, or even their collective dynamics, is infeasible and prone to failure. Similar challenges arise in embodiments that involve fluids, such as flight or swimming, where complex interactions persist until we fully solve the Navier-Stokes equations. Contact-heavy tasks, like manipulation or legged locomotion, further introduce friction and uncertainty. These examples highlight the difficulty of capturing physical phenomena in a decision-making framework and suggest that egocentric representations, while valuable, may not be sufficient on their own.

Others might argue that the fundamental bottleneck lies not in algorithms or intelligence, but in hardware. Current electric motors are a rudimentary proxy for biological muscles, which combine soft tissue, tendons, and other dynamic components. Similarly, the dexterity of the human hand far surpasses the simplistic two-fingered grippers often used in robotics. Efforts in soft robotics, tactile sensing, and novel actuators are promising, but it remains unclear whether replicating biological form is truly necessary. After all, airplanes achieve flight without flapping wings. This raises the open question: must robotic embodiments mirror biology to succeed, are the real limitations purely algorithmic, or is the real secret sauce found in the interplay between hardware and algorithms? The answer remains uncertain—and only time will tell.

This thesis is a humble step toward the ambitious vision of achieving autonomous systems with agility, reliability, and versatility comparable to that of animals. It builds upon centuries of scientific discovery and engineering progress. In that spirit, I hope this work contributes a small yet meaningful piece to that ongoing journey, paving the way for future research to continue the pursuit.

Part V

Appendix

Bibliography

[1] Jon Arrizabalaga and Markus Ryll. “Towards time-optimal tunnel-following for quadrotors”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 4044–4050.

[2] Jon Arrizabalaga and Markus Ryll. “Spatial motion planning with pythagorean hodograph curves”. In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE. 2022, pp. 2047–2053.

[3] Jon Arrizabalaga, Fausto Vega, Zbyněk ŠÍR, Zachary Manchester, and Markus Ryll. “PHODCOS: Pythagorean Hodograph-based Differentiable Coordinate System”. In: *IEEE Aerospace Conference (AERO)* (2025).

[4] Jon Arrizabalaga, Zachary Manchester, and Markus Ryll. “Differentiable collision-free parametric corridors”. In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024, pp. 1839–1846.

[5] Jon Arrizabalaga and Markus Ryll. “Sctomp: Spatially constrained time-optimal motion planning”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 4827–4834.

[6] Jon Arrizabalaga and Markus Ryll. “Pose-following with dual quaternions”. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE. 2023, pp. 5959–5966.

[7] Jon Arrizabalaga, Lukas Pries, Riddhiman Laha, Runkang Li, Sami Haddadin, and Markus Ryll. “Geometric Slosh-Free Tracking for Robotic Manipulators”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 1226–1232.

[8] Jon Arrizabalaga, Zbyněk Šír, Zachary Manchester, and Markus Ryll. “A Universal Formulation for Path-Parametric Planning and Control”. In: *Under review* (2025).

[9] Philipp Foehn, Angel Romero, and Davide Scaramuzza. “Time-optimal planning for quadrotor waypoint flight”. In: *Science robotics* 6.56 (2021), eabh1221.

[10] Jesus Tordesillas, Brett T Lopez, and Jonathan P How. “Faster: Fast and safe trajectory planner for flights in unknown environments”. In: *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2019, pp. 1934–1940.

[11] Boyu Zhou, Jie Pan, Fei Gao, and Shaojie Shen. “Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1992–2009.

[12] Hongling Wang, Joseph Kearney, and Kendall Atkinson. “Arc-length parameterized spline curves for real-time simulation”. In: *Proc. 5th International Conference on Curves and Surfaces*. Vol. 387396. 2002.

[13] Shiyu Zhao. “Time derivative of rotation matrices: A tutorial”. In: *arXiv preprint arXiv:1609.06088* (2016).

[14] John M Hollerbach. “Dynamic scaling of manipulator trajectories”. In: *1983 American Control Conference*. IEEE. 1983, pp. 752–756.

[15] Kang Shin and Neil McKay. “Minimum-time control of robotic manipulators with geometric path constraints”. In: *IEEE Transactions on Automatic Control* 30.6 (1985), pp. 531–541.

[16] Friedrich Pfeiffer and Rainer Johanni. “A concept for manipulator trajectory planning”. In: *IEEE Journal on Robotics and Automation* 3.2 (1987), pp. 115–123.

[17] Winston L Nelson and Ingemar J Cox. “Local path control for an autonomous vehicle”. In: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. IEEE. 1988, pp. 1504–1510.

[18] Yutaka Kanayama, Yoshihiko Kimura, Fumio Miyazaki, and Tetsuo Noguchi. “A stable tracking control method for an autonomous mobile robot”. In: *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE. 1990, pp. 384–389.

[19] Ingemar J Cox. “Blanche-an experiment in guidance and navigation of an autonomous robot vehicle”. In: *IEEE Transactions on robotics and automation* 7.2 (1991), pp. 193–204.

[20] John Hauser and Rick Hindman. “Maneuver regulation from trajectory tracking: Feedback linearizable systems”. In: *IFAC Proceedings Volumes* 28.14 (1995), pp. 595–600.

[21] Roger Skjetne, Thor I Fossen, and Petar V Kokotović. “Robust output maneuvering for a class of nonlinear systems”. In: *Automatica* 40.3 (2004), pp. 373–383.

[22] Khac Duc Do, Zhong-Ping Jiang, and Jie Pan. “Robust adaptive path following of underactuated ships”. In: *Automatica* 40.6 (2004), pp. 929–944.

[23] A Pedro Aguiar, Joao P Hespanha, and Petar V Kokotovic. “Path-following for non-minimum phase systems removes performance limitations”. In: *IEEE Transactions on Automatic Control* 50.2 (2005), pp. 234–239.

[24] A Pedro Aguiar and Joao P Hespanha. “Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty”. In: *IEEE transactions on automatic control* 52.8 (2007), pp. 1362–1379.

[25] Timm Faulwasser and Rolf Findeisen. “Nonlinear model predictive control for constrained output path following”. In: *IEEE Transactions on Automatic Control* 61.4 (2015), pp. 1026–1039.

[26] Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl. “Time-optimal path tracking for robots: A convex optimization approach”. In: *IEEE Transactions on Automatic Control* 54.10 (2009), pp. 2318–2327.

[27] Timm Faulwasser, Benjamin Kern, and Rolf Findeisen. “Model predictive path-following for constrained nonlinear systems”. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE. 2009, pp. 8642–8647.

[28] Denise Lam, Chris Manzie, and Malcolm Good. “Model predictive contouring control”. In: *49th IEEE Conference on Decision and Control (CDC)*. IEEE. 2010, pp. 6137–6142.

[29] Florian Kehrle, Janick V Frasch, Christian Kirches, and Sebastian Sager. “Optimal control of formula 1 race cars in a VDrift based virtual environment”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 11907–11912.

[30] Yiqi Gao, Andrew Gray, Janick V Frasch, Theresa Lin, Eric Tseng, J Karl Hedrick, and Francesco Borrelli. “Spatial predictive control for agile semi-autonomous ground vehicles”. In: *Proceedings of the 11th international symposium on advanced vehicle control*. 2. 2012, pp. 1–6.

[31] Janick V Frasch, Andrew Gray, Mario Zanon, Hans Joachim Ferreau, Sebastian Sager, Francesco Borrelli, and Moritz Diehl. “An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles”. In: *2013 European Control Conference (ECC)*. IEEE. 2013, pp. 4136–4141.

[32] Alexander Liniger, Alexander Domahidi, and Manfred Morari. “Optimization-based autonomous racing of 1: 43 scale RC cars”. In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647.

[33] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. “Model predictive contouring control for collision avoidance in unstructured dynamic environments”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4459–4466.

[34] Daniel Kloeser, Tobias Schoels, Tommaso Sartor, Andrea Zanelli, Gianluca Prison, and Moritz Diehl. “NMPC for racing using a singularity-free path-parametric model with obstacle avoidance”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 14324–14329.

[35] Rudolf Reiter, Armin Nurkanović, Jonathan Frey, and Moritz Diehl. “Frenet-cartesian model representations for automotive obstacle avoidance within nonlinear mpc”. In: *European Journal of Control* 74 (2023), p. 100847.

[36] Sara Spedicato and Giuseppe Notarstefano. “Minimum-time trajectory generation for quadrotors in constrained environments”. In: *IEEE Transactions on Control Systems Technology* 26.4 (2017), pp. 1335–1344.

[37] Jialin Ji, Xin Zhou, Chao Xu, and Fei Gao. “Cmpcc: Corridor-based model predictive contouring control for aggressive drone flight”. In: *Experimental Robotics: The 17th International Symposium*. Springer. 2021, pp. 37–46.

[38] Angel Romero, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. “Model predictive contouring control for time-optimal quadrotor flight”. In: *IEEE Transactions on Robotics* 38.6 (2022), pp. 3340–3356.

[39] Thomas Fork and Francesco Borrelli. “Euclidean and non-Euclidean Trajectory Optimization Approaches for Quadrotor Racing”. In: *arXiv preprint arXiv:2309.07262* (2023).

[40] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. “Champion-level drone racing using deep reinforcement learning”. In: *Nature* 620.7976 (2023), pp. 982–987.

[41] Robin Verschueren, Niels van Duijkeren, Jan Swevers, and Moritz Diehl. “Time-optimal motion planning for n-dof robot manipulators using a path-parametric system reformulation”. In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 2092–2097.

[42] Niels van Duijkeren, Robin Verschueren, Goele Pipeleers, Moritz Diehl, and Jan Swevers. “Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation”. In: *2016 European Control Conference (ECC)*. IEEE. 2016, pp. 477–482.

[43] Martin Böck and Andreas Kugi. “Real-time nonlinear model predictive path-following control of a laboratory tower crane”. In: *IEEE Transactions on Control Systems Technology* 22.4 (2013), pp. 1461–1473.

[44] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[45] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. “Outracing champion Gran Turismo drivers with deep reinforcement learning”. In: *Nature* 602.7896 (2022), pp. 223–228.

[46] Sant Kumar and Rajan Gill. “Path following for quadrotors”. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE. 2017, pp. 2075–2081.

[47] Rudolf Reiter, Martin Kirchengast, Daniel Watzenig, and Moritz Diehl. “Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards”. In: *IFAC-PapersOnLine* 54.6 (2021), pp. 99–106.

[48] Maria Krinner, Angel Romero, Leonard Bauersfeld, Melanie Zeilinger, Andrea Carron, and Davide Scaramuzza. “Time-Optimal Flight with Safety Constraints and Data-driven Dynamics”. In: *arXiv preprint arXiv:2403.17551* (2024).

[49] Dirk Jan Struik. *Lectures on classical differential geometry*. Courier Corporation, 1961.

[50] Elsa Abbena, Simon Salamon, and Alfred Gray. *Modern differential geometry of curves and surfaces with Mathematica*. Chapman and Hall/CRC, 2017.

[51] Hyeong In Choi and Chang Yong Han. “Euler–Rodrigues frames on spatial Pythagorean-hodograph curves”. In: *Computer Aided Geometric Design* 19.8 (2002), pp. 603–620.

[52] Zbyněk Šír and Bert Jüttler. “C^2 Hermite interpolation by Pythagorean hodograph space curves”. In: *Mathematics of Computation* 76.259 (2007), pp. 1373–1391.

[53] Richard L Bishop. “There is more than one way to frame a curve”. In: *The American Mathematical Monthly* 82.3 (1975), pp. 246–251.

[54] Andrew J Hanson and Hui Ma. “Parallel transport approach to curve framing”. In: *Indiana University, Techreports-TR425* 11 (1995), pp. 3–7.

[55] Wenping Wang, Bert Jüttler, Dayue Zheng, and Yang Liu. “Computation of rotation minimizing frames”. In: *ACM Transactions on Graphics (TOG)* 27.1 (2008), pp. 1–18.

[56] Rida T Farouki. *Pythagorean—Hodograph Curves*. Springer, 2008.

[57] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*. Vol. 375. Springer Berlin Heidelberg New York, 1996.

[58] Robin Deits and Russ Tedrake. “Computing large convex regions of obstacle-free space through semidefinite programming”. In: *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer. 2015, pp. 109–124.

[59] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J Taylor, and Vijay Kumar. "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments". In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695.

[60] Xingguang Zhong, Yuwei Wu, Dong Wang, Qianhao Wang, Chao Xu, and Fei Gao. "Generating large convex polytopes directly on point clouds". In: *arXiv preprint arXiv:2010.08744* (2020).

[61] Charbel Toumieh and Alain Lambert. "Voxel-grid based convex decomposition of 3D space for safe corridor generation". In: *Journal of Intelligent & Robotic Systems* 105.4 (2022), p. 87.

[62] Angel Romero, Shreedhar Govil, Gonca Yilmaz, Yunlong Song, and Davide Scaramuzza. "Weighted maximum likelihood for controller tuning". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 1334–1341.

[63] Nguyen Hung, Francisco Rego, Joao Quintas, Joao Cruz, Marcelo Jacinto, David Souto, Andre Potes, Luis Sebastiao, and Antonio Pascoal. "A review of path following control strategies for autonomous robotic vehicles: Theory, simulations, and experiments". In: *Journal of Field Robotics* 3 (2023), pp. 747–779.

[64] Robert Penicka and Davide Scaramuzza. "Minimum-time quadrotor waypoint flight in cluttered environments". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5719–5726.

[65] Heinrich W Guggenheimer. *Differential geometry*. Courier Corporation, 2012.

[66] Jiang Hu, Xin Liu, Zai-Wen Wen, and Ya-Xiang Yuan. "A brief introduction to manifold optimization". In: *Journal of the Operations Research Society of China* 8 (2020), pp. 199–248.

[67] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

[68] Shucheng Kang, Xiaoyang Xu, Jay Sarva, Ling Liang, and Heng Yang. "Fast and Certifiable Trajectory Optimization". In: *arXiv preprint arXiv:2406.05846* (2024).

[69] Tobia Marcucci, Jack Umenberger, Pablo Parrilo, and Russ Tedrake. "Shortest paths in graphs of convex sets". In: *SIAM Journal on Optimization* 34.1 (2024), pp. 507–532.

[70] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. "Motion planning around obstacles with convex optimization". In: *Science robotics* 8.84 (2023), eadf7843.

[71] Tobia Marcucci. "Graphs of Convex Sets with Applications to Optimal Control and Motion Planning". PhD thesis. MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2024.

[72] Mark Petersen and Russ Tedrake. "Growing convex collision-free regions in configuration space using nonlinear programming". In: *arXiv preprint arXiv:2303.14737* (2023).

[73] Peter Werner, Thomas Cohn, Rebecca H Jiang, Tim Seyde, Max Simchowitz, Russ Tedrake, and Daniela Rus. "Faster Algorithms for Growing Collision-Free Convex Polytopes in Robot Configuration Space". In: *arXiv preprint arXiv:2410.12649* (2024).

[74] Akshay Jaitly and Siavash Farzan. “PAAMP: Polytopic Action-Set And Motion Planning For Long Horizon Dynamic Motion Planning via Mixed Integer Linear Programming”. In: *arXiv preprint arXiv:2403.10924* (2024).

[75] Sadatoshi Kumagai. “An implicit function theorem: Comment”. In: *Journal of Optimization Theory and Applications* 31 (1980), pp. 285–288.

[76] Brian Plancher and Scott Kuindersma. “A performance analysis of parallel differential dynamic programming on a gpu”. In: *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*. Springer. 2020, pp. 656–672.

[77] Emre Adabag, Miloni Atal, William Gerard, and Brian Plancher. “Mpcgpu: Real-time nonlinear model predictive control through preconditioned conjugate gradient on the gpu”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 9787–9794.

[78] Arun L Bishop, John Z Zhang, Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester. “Relu-qp: A gpu-accelerated quadratic programming solver for model-predictive control”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 13285–13292.

List of Figures

Supplementary Material of Enclosed Publications

Towards Time-Optimal Tunnel-Following for Quadrotors [1]

Paper https://arxiv.org/pdf/2110.01351

Video https://youtu.be/Apc8MCu7Yvo

Spatial Motion-Planning with Pythagorean Hodograph Curves [2]

Paper https://arxiv.org/pdf/2209.01673

Video https://youtu.be/Sl6x8l7RJK8

Talk https://youtu.be/Vj1ofBTrNX4

Pythagorean Hodograph-based Differentiable Coordinate System [3]

Paper https://arxiv.org/pdf/2410.07750

Code https://github.com/jonarriza96/phodcos

Talk https://youtu.be/Bh2_ISXp8b4

Differentiable Collision-Free Parametric Corridors [4]

Paper https://arxiv.org/pdf/2407.12283

Code https://github.com/jonarriza96/corrgen

Video https://youtu.be/MvC7bPodXz8

Talk https://youtu.be/l6LAugm89mQ

SCTOMP: Spatially Constrained Time-Optimal Motion Planning [5]

Paper https://arxiv.org/pdf/2210.02345

Video https://youtu.be/zGExvnUEfOY

Talk https://youtu.be/pOV3pGGcIqo

Pose-Following with Dual Quaternions [6]

Paper https://arxiv.org/pdf/2308.09507

Code https://github.com/jonarriza96/pfdq

Talk https://youtu.be/TQig2j90Ijc

Geometric Slosh-Free Tracking for Robotic Manipulators [7]

Paper https://arxiv.org/pdf/2402.05197

Code https://github.com/jonarriza96/gsft

Video https://youtu.be/4kitqYVS9n8

Talk https://youtu.be/IgoGLl_lRSw?si=MoOVsCLlmHVh_G1y

A Universal Framework for Path-Parametric Planning & Control [8]

Paper https://arxiv.org/pdf/2410.04664

Website https://path-parametric.github.io/

Code https://github.com/jonarriza96/PathParam