SAMPLING-BASED MOTION PLANNING FOR SAFE AND EFFICIENT SPACECRAFT PROXIMITY OPERATIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS & ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Joseph Alexander Starek
June 2016

This dissertation is online at: http://purl.stanford.edu/bk279gh7089

Includes supplemental files:

1. This video presents an overview of the capabilities of the robotic free-flyers of the Stanford Space Robotics Facilit... *(Free-flyer Testbed Overview and Capabilities.mp4)*
2. This video shows in detail how the Fast Marching Trees (FMT*) motion planning algorithm safely guides a hovering robo... *(Free-flyer Autonomous Docking Trials.mp4)*

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marco Pavone, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Sigrid Close

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Rock

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Autonomy has demonstrated success in many vehicle control problems, but has yet to show significant breakthroughs for spacecraft guidance during proximity operations. In part due to a costly verification and validation process as well as from limited access to formally-safe guidance algorithms, mission planners have instead had to rely on maneuver plans with straightforward, easily-verified trajectories and extensive human oversight. Unfortunately, this strategy often introduces propellant inefficiencies, adds significant labor overhead, and limits missions to Earth proximity where two-way communication times are short. This dissertation seeks to remedy these issues by developing a *provably-safe* and *propellant-efficient* sampling-based motion planning framework for fully-autonomous spacecraft proximity operations. The framework is designed for a wide range of hazardous guidance scenarios, including autonomous orbital rendezvous and inspection, pinpoint small-body descent, and on-orbit satellite servicing. Due to the dangers associated with operating near other objects, special care is taken to enable real-time guidance as well as ensure the availability of safe abort trajectories so that spacecraft can respond quickly and safely to control failures and sudden environmental changes.

In the first part of the dissertation, we motivate the need for sampling-based motion planning, introducing two algorithms particularly well-suited to complex guidance problems with many constraints. The real-time performance of these planners are then evaluated in numerical experiments and onboard a free-flying, air-bearing robot. In the second part of the dissertation, we develop a framework for real-time, propellant-optimal spacecraft proximity operations, leveraging insights from earlier experiments. Along the way, we introduce a strategy for guaranteeing abort safety, devise fast tools for improving planning solutions, and maintain real-time guidance through a unique online-offline approach that offloads many time-consuming computations to ground systems. To illustrate our ideas, techniques are tailored to impulsively-actuated guidance near circular orbits. Simulations demonstrate that the methodology can accommodate a wide variety of complex trajectory constraints, including plume impingement and obstacle avoidance, thrust allocation, and a "stuck-off" thruster fault-tolerance, all while returning low-cost solutions within seconds on modern hardware.

Through its generality, efficiency, and speed, the proposed approach offers the potential to enable entirely new capabilities for next-generation space missions, while also increasing the frequency, flexibility, and reliability of present-day operations in space.

Dedication

This work is dedicated to my father Dr. Peter J. K. Starek and my maternal grandparents Alfred J. Capoferi and Mary A. Capoferi, who each passed away during the course of my doctoral studies at Stanford. As a cardiothoracic surgeon, a mathematics teacher, and a secretary/housewife, respectively, they were the quintessence of integrity, morality, and industriousness. This thesis is a direct result of their outstanding legacy, excellent example, and endless encouragement and wisdom. They taught me in life to appreciate my education as "the greatest gift you can give yourself," and showed me the value of hard work—neither lesson of which has been more relevant or tangible than during the production of this dissertation. As my grandfather's favorite expression goes,

> "By your deeds, they shall know you."

I hope that at least a few may come to know my dad and grandparents by this work, through the immeasurable influence they have had on me. They were the greatest people I have ever known. *Ad astra, per aspera. In memoriam.*

Acknowledgments

Research Acknowledgments First and foremost, I would like to thank my advisor, Prof. Marco Pavone, who taught me what it means to do good research. This body of work is a direct result of his thoughtful counsel, keen research insights, and profound engineering acumen. I will always be grateful for his helpful advice on both my academic and non-academic pursuits, his admirable dedication to intellectual integrity and software engineering best practices from the robotics industry, and his significant patience and understanding during my times of family crisis. Thank you for your willingness to accept me as one of your first doctoral students!

A very special thank you must be said to Brent W. Barbee, Flight Dynamics engineer at NASA's Goddard Space Flight Center, aerospace engineering lecturer at the University of Maryland, and unofficial "co-advisor" during my doctoral career. His tremendous expertise in provably-safe spacecraft guidance and mission planning were indispensable in shaping the course of my research and in developing the form of my algorithmic solutions. It was a pleasure working with him and I hope that we may continue to collaborate in the future.

I would also like to acknowledge my many co-authors over the years (particularly Edward Schmerling, who was instrumental in developing many of the proofs found in this thesis), as well as Profs. Stephen Rock and Sigrid Close for their willingness to serve on my dissertation reading committee and for their constructive and valuable feedback during the final stages of my PhD.

Lastly, I cannot neglect to credit the rest of the team who helped develop the free-flyer robotic testbed of the Stanford Space Robotics Facility (see Chapter 4). This includes, in no particular order, Andrew Bylard, who was particularly instrumental in maturing the control framework and implementation (including adding the Learning-Based Model Predictive Controller), Edward Schmerling, who helped implement the planning logic and code, the team of Federico Rossi, Quinn Wu, and Ivan Marić, who designed and tested early Proportional-Integral-Derivative (PID) trajectory-following controllers and robustified much of the control logic, and finally the numerous high school and graduate students who built the mini free-flyers and collectively designed a number of docking mechanism prototypes. Special thanks go to Rick Zhang for the expertise and time he shared with me early-on in modernizing the avionics design, and to Federico Rossi for his superb cinematography skills.

Funding Acknowledgments Regarding funding, I would like to express my gratitude to the National Aeronautics and Space Administration (NASA) for its financial support of my graduate research through the NASA Space Technology Research Office (STRO). This thesis was supported by NASA Early Career Faculty (ECF) Grant #NNX12AQ43G.

Other Acknowledgments In addition, I would like to thank the many other individuals in my life who have contributed to my academic career thus far, the first and foremost being my mother, Dr. Nancy Capoferi Starek. Her unwavering support—both financial, moral, and otherwise—have been the greatest gifts I could have ever received, giving me the freedom to pursue my own research interests without burden. Thank you for always being there for me, Mom. To Prof. Sigrid Close, I would like to say thank you for advocating on my behalf for my matriculation into the Stanford Aero/Astro PhD program, and for giving me the opportunity to teach dynamics with you during my first year here—without you, it is difficult to say where my academic endeavors might have veered. I owe additional thanks to my advisor Prof. Marco Pavone and Profs. Adrian Lew and Chris Gerdes of the Mechanical Engineering department at Stanford for allowing me to teach with them as well, and for their dedication to engineering education.

To my very capable and talented peers in the Autonomous Systems Laboratory, whom I admire and respect more than they could know, it was an honor and a tremendous pleasure working with you. I will never forget our many late nights in Durand, originally desk-less back in the formative days of the lab. Nor will I forget our lab skiing trip to Lake Tahoe and our other adventures in the Bay Area. I wish you all the very best of luck in the future. Particularly Rick Zhang, Christopher Pepper, Federico Rossi, Ashley Clark, Ross Allen, Edward Schmerling, Brian Ichter, Sumeet Singh, Ben Hockman, Nathaniel Nakashima of the GSB, and Andrew C. Smith of the ARL—thank you for making these years at Stanford so memorable! To my Aunt Anna, Uncle Ike, Kostal cousins, sister Joanna, Tracy, and to Rich and Connie Walker and the entire Walker clan, thank you for making California my home away from home. To my other aunts and uncles, especially Aunt Paula, thank you for your encouragement and your continual involvement in my life. And finally, a special thank you goes to my friend Chris Buchheit, who was there for me and my family during our hardships and who has always been an honorable person and the best of friends. To my many other friends and family members who I sadly did not have the space to mention, thank you for your part in making the weights of my graduate school travels that much lighter.

Last, but certainly not least, I would be remiss if I did not mention my heartfelt thanks to my dearest Lindsay, who was continually there for me throughout the stressful times with her unyielding love, support, and encouragement. This dissertation represents your efforts as much as mine. I cannot wait to see where our adventure takes us next!

Contents

List of Tables

List of Figures

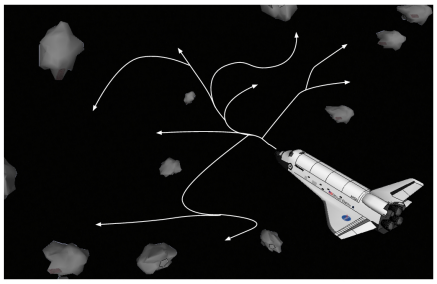
Chapter 1

Introduction

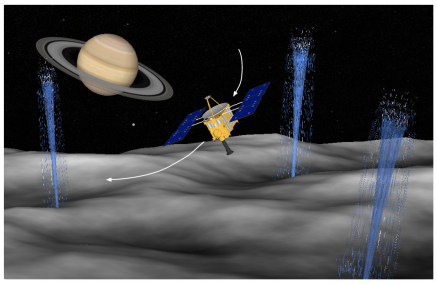
This thesis presents an algorithmic framework for the automated design of provably-safe, propellant-optimized guidance trajectories for spacecraft that are maneuvering near other objects or vehicles—a scenario generally termed *spacecraft proximity operations*. Such maneuvers arise frequently during mission planning, whether there be two spacecraft that need to be docked together or an orbiter that needs to make a soft precision landing onto the surface of an asteroid. The proposed framework relies upon the field of *sampling-based motion planning* to efficiently construct a set of feasible, low-cost trajectories through the vehicle's constrained surroundings. The approach enables a spacecraft to make quick decisions (that is, in "real-time") about how and when to apply its actuators in order to complete a point-to-point maneuver safely and with low propellant cost; in the event of control failures, actuated abort maneuvers are made available from every point on the trajectory as safe escape routes to prevent potential mission disasters. The framework is tailored to spacecraft guidance applications involving many competing constraints and mission-jeopardizing hazards, including rendezvous and docking, on-orbit satellite servicing, and small-body descent and landing.

As motivating examples, consider the scenarios illustrated in Fig. 1.1. In Fig. 1.1a, we see the Russian Progress spacecraft on its automated final approach to the International Space Station. Should thrusters fail during the final minutes of these docking maneuvers, the likelihood is high that the incoming spacecraft will collide with the station, threatening the lives of astronauts onboard. Enforcing safety constraints during operations of this kind can help maintain safety as long as possible prior to final docking. Extrapolating to many nearby objects, one might imagine an even harder scenario as illustrated in Fig. 1.1b in which we seek to maneuver safely and efficiently around a collection of dynamic, uncooperative debris (possibly created by a kinetic impact on a nearby object, or representing unexpected debris near a rubble-pile asteroid, for instance). Such a guidance feat requires either exceptional relative state sensing so that debris motion can be predicted accurately (improbable for most spacecraft applications), or otherwise necessitates a fast, high-frequency re-planning scheme (though such dynamic environments are not explicitly handled in this thesis, we

(a) A Progress M-21M spacecraft docking to the International Space Station. *Image courtesy of NASA.*



(b) Safe trajectory guidance within a heavily-cluttered orbital debris field.



(c) Artistic rendition of autonomous planning in proximity of outgassing water vapor plumes near the surface of Enceladus.

Figure 1.1: Spacecraft proximity operations applications that would benefit from autonomous relative guidance. The presence of nearby vehicles and objects adds several complex and sometimes time-varying constraints, which, if violated, could jeopardize mission safety.

handle them implicitly through real-time guidance, enabling the latter). Finally, a radically-different scenario is shown in Fig. 1.1c, depicting proximity operations about the Saturnian moon Enceladus—a target scientifically prized for its abundance of sub-surface water and thus its potential for microbial life. Occasionally, and unpredictably, the moon will emit massive plumes of this water as jets of vapor emanating from its surface fissures. We might imagine two types of unique missions to Enceladus, both requiring real-time guidance: (i) a geophysical surveying spacecraft that needs to avoid plumes in order to safely collect its data without interference, or (ii) a sample-and-return mission requiring redirection *into* the outgassing plumes in order to collect and store plume material. Such agile, opportunistic science missions have never before been demonstrated in orbital applications.

The purpose of this chapter is to provide additional context for autonomous spacecraft guidance, highlight its need as an enabling technology for next-generation space missions, and explain why it is a challenging problem. Recent autonomous demonstration missions are discussed, followed by an overview of future challenges and modern state-of-the-art techniques built to address some of them. The chapter concludes with a precise discussion of the original contributions and organization of the remainder of this thesis dissertation.

Note, much of the material presented here is also documented in book chapter [1].

1.1 Background

Before we begin, it is helpful to clarify some terminology. *Guidance* is the process of real-time planning of spacecraft state trajectories in both translational and rotational motion. This involves computing desired sets of translational and rotational states and corresponding control forces and torques as functions of time. *Control*, or more specifically feedback control, is responsible for keeping spacecraft close to these trajectories, using real-time state updates provided by a *navigation* system to inform control decisions in the presence of disturbances, measurement noise, and model uncertainties. Together they are referred to as Guidance, Navigation, and Control (GN&C, or just G&C). In this thesis, for the purposes of tractability, we will restrict our attention to guidance only; the challenging trajectory-following and navigation problems will be kept outside of our scope. As will be motivated in this chapter, the guidance problem alone is inherently difficult to solve and is by no means solved even for straight-line path planning, let alone for systems with complex dynamics.

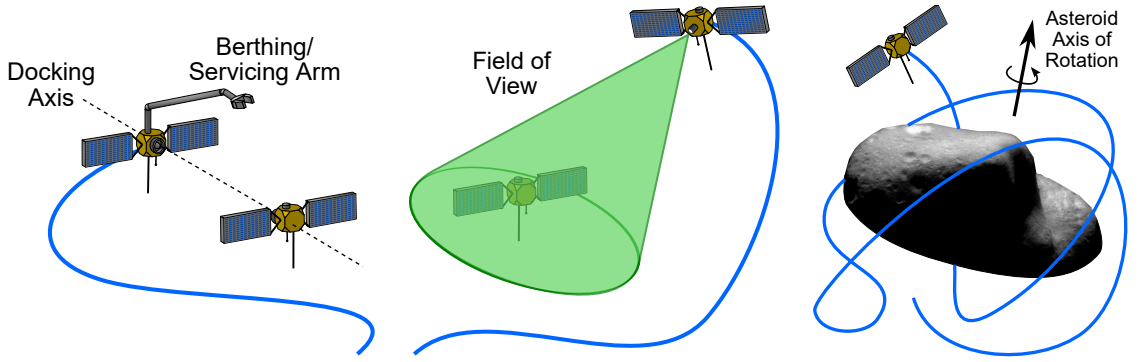
This section describes the technical details and challenges specific to *relative guidance* of autonomous spacecraft as found in the three major areas of spacecraft proximity operations:

- **Autonomous Rendezvous and Docking (AR&D)** (Fig. 1.2a): Encompasses both the far-field and near-field approach phases of rendezvous, though proximity operations are chiefly concerned with the closing and final approach portions of the near-field phase. Applications include on-orbit assembly, re-joining with deployed landers, and docking to allow material or crew exchange between spacecraft.

- **Autonomous Inspection and Servicing (AIS)** (Fig. 1.2b): Includes survey maneuvers used to visually inspect a target spacecraft's exterior, as well as missions to repair, resupply, and retrieve/harvest other spacecraft. These latter three operations, which inherently require robotic manipulation of the target or some other form of inter-cooperative planning, are also collectively referred to as *on-orbit satellite servicing*.

- **Small-Body Proximity Operations** (Fig. 1.2c): Constitutes motion near primitive solar system bodies (also called "small bodies"), including moons, asteroids, comets, and their like. In addition to their often complex shapes, small bodies are also characterized by their uncertain dynamics due to imperfect measurements of their mass distribution, soil properties (*e.g.*, albedos), and rotational states, as well as their unknown local environments, which may be cluttered with natural satellites, dust, or debris.

Regardless of the particular application, the spacecraft proximity operations guidance problem can be posed in its most general form as an *optimal control problem*, with dynamic equations describing the motion of the spacecraft as well as time-varying constraints on its associated state and control



(a) Autonomous Rendezvous and Docking (AR&D)

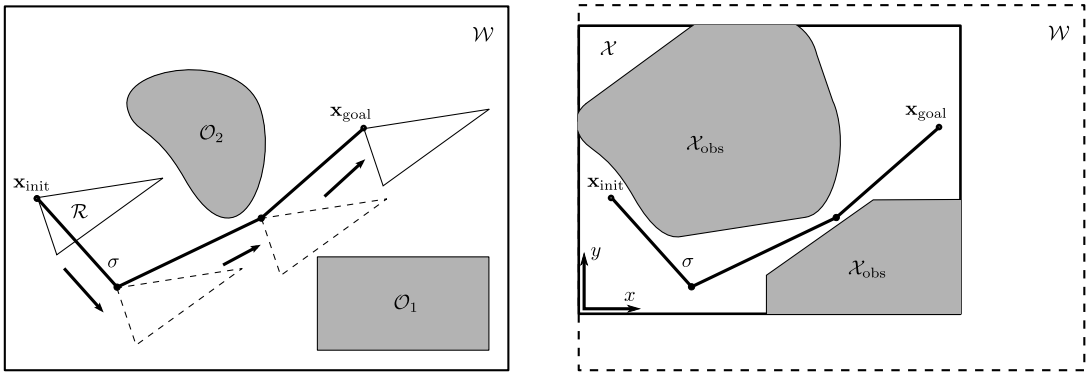
(b) Autonomous Inspection and Servicing (AIS)

(c) Small-Body Proximity Operations

Figure 1.2: Examples of spacecraft proximity operations. The solid blue curves represent artistic renditions of spacecraft guidance trajectories.

trajectories. Before proceeding further, we first need to make precise a few of the concepts behind optimal control and trajectory planning, namely states, controls, and certain topological spaces called workspaces, state spaces, and control spaces. A *state* in broad terms is the complete set of information required to mathematically model the future motion of a system; we represent a state as a collection of generalized coordinates and system parameters stacked into a vector $\mathbf{x} \in \mathbb{R}^d$. For example, states can encompass the positions, velocities, masses, or other physical properties of a system and its subcomponents. A *control* is a set of mathematical variables that the trajectory designer has direct influence over, through which a system can be driven from one state to another. Control vectors $\mathbf{u} \in \mathbb{R}^{N_u}$, which can be viewed as inputs, model physical features like actuators, thrust forces, torques, or other commands that can alter our system performance metric, called an *objective* or *cost functional* J, which may be viewed as an output. Together, a state at some initial time t_{init} (represented as $\mathbf{x}(t_{\text{init}})$) along with a control trajectory over time ($\mathbf{u}(t)$, where $t \geq t_{\text{init}}$) are entirely sufficient to simulate the future motion of any causal system through its ordinary differential dynamics equation, $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t)$. Optimal control theory seeks to find the best-possible such control trajectory, denoted by $\mathbf{u}^*(t)$, that minimizes our cost functional $J(\mathbf{x}(t), \mathbf{u}(t), t)$ subject to an array of constraints; if one can be found, the resulting state trajectory and cost are said to be *optimal*, and denoted by $\mathbf{x}^*(t)$ and J^*, respectively.

Conceptually, this motion of our system, described by states $\mathbf{x}$ and influenced by controls $\mathbf{u}$, can be interpreted in two different ways. On the one hand, the most intuitive way to view our system is as a physical vehicle moving through a 3-dimensional world (or 4-dimensional, if including time). This world in which our vehicle actually lives, encompassing the spacecraft and any nearby objects and physical barriers, is what we call the *workspace*. On the other hand, because we can parameterize the motion of our vehicle fully in terms of its d-length state vector, we can alternatively view our



(a) Workspace $\mathcal{W}$ representation (with physical obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$)

(b) State space $\mathcal{X}$ representation (with corresponding infeasible states $\mathcal{X}_{\text{obs}}$)

Figure 1.3: Comparing the workspace and state space representations of vehicle guidance. Here a triangular robot translates (without rotating) from one position to another around two nearby obstacles; if we define the robot's state as the (x, y)-position of its upper-left corner, the corresponding state space motion can be interpreted as shown on the right.

system as a particle moving within its d-dimensional *state space*, the vector space in which our states reside. This concept, first conceived by Lagrange through his notion of generalized coordinates, revolutionized the field of motion planning when it was first introduced by Lozano-Pérez [2]. Through this pivotal idea, all guidance problems, with their infinite variation and complexity when represented in the workspace, may be reduced to the universal problem of finding a feasible path for a moving particle from one point to another within the state space. (Unfortunately it is not usually practical to explicitly translate our problem from the workspace to the state space, as will be detailed later). To gain a clearer picture of this concept, we provide in Fig. 1.3 a simple visual example of this workspace/state-space equivalence for a planar triangular robot translating around two obstacles.

With this mathematical framework precisely-defined, we are now in position to express our generic autonomous spacecraft guidance problem as follows:

Problem (Autonomous Spacecraft Guidance)**.**

$$\begin{aligned}
\underset{t_{\text{f}},\mathbf{u}(t)}{\text{minimize}} \quad & J(\mathbf{x}(t),\mathbf{u}(t),t) = K(\mathbf{x}(t_{\text{f}}),t_{\text{f}}) + \int_{t_0}^{t_{\text{f}}} \ell(\mathbf{x}(t),\mathbf{u}(t),t)\,\mathrm{d}t && (1.1)\\
\text{subject to} \quad & \mathbf{x}(t_0) \in \mathcal{X}_0 && \text{Initial Condition}\\
& \mathbf{x}(t_{\text{f}}) \in \mathcal{X}_{\text{f}} && \text{Final Condition}\\
& \dot{\mathbf{x}}(t) = f(\mathbf{x}(t),\mathbf{u}(t),t) && \text{System Dynamics}\\
& \mathbf{u}(t) \in \mathcal{U}(t) && \text{Control Admissibility}\\
& \mathbf{x}(t) \in \mathcal{X}(t) \quad \text{for all } t \in [t_0,\, t_{\text{f}}] && \text{Trajectory Feasibility}
\end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the state of the spacecraft, $\mathbf{u} \in \mathbb{R}^{N_u}$ is the control input, $t \in \mathbb{R}$ is time, t_0 is the initial time, t_f is some final time (implicitly assumed to satisfy $t_\mathrm{f} > t_0$), and $\mathcal{X}_0$ and $\mathcal{X}_\mathrm{f}$ are initial and final state constraint sets. In the above representation, $J : \mathbb{R}^{d+N_u+1} \to \mathbb{R}$ is the cost-functional (which combines terminal and incremental additive cost functionals K and ℓ), $f : \mathbb{R}^{d+N_u+1} \to \mathbb{R}^d$ defines the dynamics, and $\mathcal{U} : \mathbb{R} \to \mathbb{R}^{N_u}$ and $\mathcal{X} : \mathbb{R} \to \mathbb{R}^d$ are set-valued maps defining spacecraft control and state trajectory constraints, respectively.

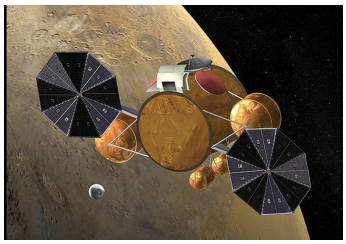
Equation (1.1) represents what is generically-called a *Two-Point Boundary Value Problem* (2PBVP), in which we seek the best decision variable values (in this case an optimal end time t_f^* and control trajectory $\mathbf{u}^*(t)$) so as to minimize some cost functional J of a function $\mathbf{x}(t)$ whose endpoints must satisfy certain conditions. Unfortunately, due to the existence of system dynamics and constraints, Eq. (1.1) cannot be solved analytically (except in very rare cases), and hence must be tackled numerically [3, 4] via an optimization algorithm after implementing a suitable decision variable discretization [5, 6]. As we will motivate in subsequent discussions, any algorithm hoping to meet the challenges of solving Eq. (1.1) for fully-autonomous spacecraft guidance will need to meet the following specifications:

- **Real-time Implementability**: Algorithms must be implemented and executed on standard processors in a reasonable amount of time.
- **Optimality**: Given that feasible solutions exist, an optimal solution $\mathbf{x}^*(t)$ is desired which minimizes (at least approximately) the cost function J.
- **Verifiability**: There must be design metrics that accurately describe the performance and robustness of GN&C algorithms, with accompanying methods for verifying these metrics.

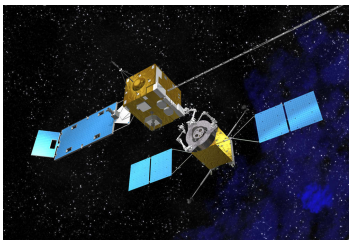
1.1.1 Need

Autonomous maneuvering, especially in proximity of artificial objects (*e.g.*, satellites, debris, *etc.*) or solar system bodies, is expected to be a key enabler for many next-generation space missions. Figure 1.4 shows a few of the more interesting examples among the many proposed mission concepts that will require serious advancements in this area, including sample return missions (expected to save greatly on mass and cost through extra-terrestrial orbital rendezvous) and satellite servicing demonstrators (which, if successful, could greatly reduce the cost of access to space). These kinds of applications are of great interest to NASA, DARPA, and the international space community at large.

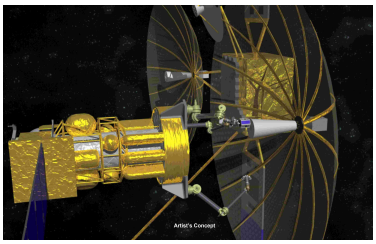
Sadly, however, these missions are beyond the capability of current approaches. The standard technique to spacecraft proximity operations maneuvering consists primarily of ad hoc planning, in which trajectory designers manually choose a sequence of waypoints interconnected by coasting arcs according to specific mission needs and to ensure collision safety following thrust failures [10]. Coasting arc interconnections are often determined using standard matrix inversion techniques [11], Lambert targeting [12], or glideslope algorithms [13]. Waypoints are usually designed as "Go/No-Go"



(a) Extra-terrestrial orbital rendezvous (Mars sample return [7]). *Image courtesy of NASA JPL.*



(b) On-orbit satellite refueling (NASA Robotic Refueling Mission [8]). *Image courtesy of NASA GSFC.*



(c) On-orbit satellite repair and harvesting (DARPA Phoenix Mission [9]). *Image courtesy of DARPA.*

Figure 1.4: Artistic concepts of next-generation space missions that are of interest to the greater space community

points, which are holding positions at which the maneuvering spacecraft must remain until given clearance from operators on the ground to proceed. The approach relies heavily on designer intuition and experience, requiring the strategic positioning of waypoints and arcs in order to verify safety at all points along the nominal trajectory (as well as under any possible perturbations).

Consider the common example of rendezvousing with the International Space Station (ISS), as depicted in Fig. 1.5. Due to the locations of relative guidance antennas, spacecraft are typically required to drift in from a lower coplanar orbit from the points S0 to S1. Next, a homing arc is followed to bring the spacecraft to a waypoint S2 within spherical communication range along the orbital path trailing behind the ISS. Once given permission to proceed, the spacecraft initiates a closing arc (carefully designed to avoid a possible collision in the event of partial or full thrust failures), which terminates at another holding point S3 before commencing a slow, steady progression along a straight-line final translation maneuver along the ISS docking axis. Other types of straight-line final approaches are also possible (*e.g.*, from radially below), but the example demonstrates the idea. Regardless of the specific choice, these paths must be rigorously verified on the ground via thorough simulations before any spacecraft is permitted to rendezvous.

Sequences such as these have been chosen through flight heritage because they work, and because they have been performed safely. However, there can be strong disadvantages to this ad hoc approach. First, straight-line trajectories (though easy to validate) are costly to follow, as they require continuous use of propellant to counteract the fictitious forces that arise from operating in the target spacecraft's rotating frame. Other arcs proposed by trajectory designers, though feasible, may also generate propellant inefficiencies due to the non-intuitive complexities of relative spacecraft motion. Second, this design method often does not admit any flexibility; small changes to the mission require complete guidance redesign. Third, one can imagine scenarios in which mission constraints are too complex to solve through intuition, or in which ad hoc reasoning would lead to such great inefficiencies as to render the solution prohibitively expensive. Fourth, the method is entirely reliant on intuition,

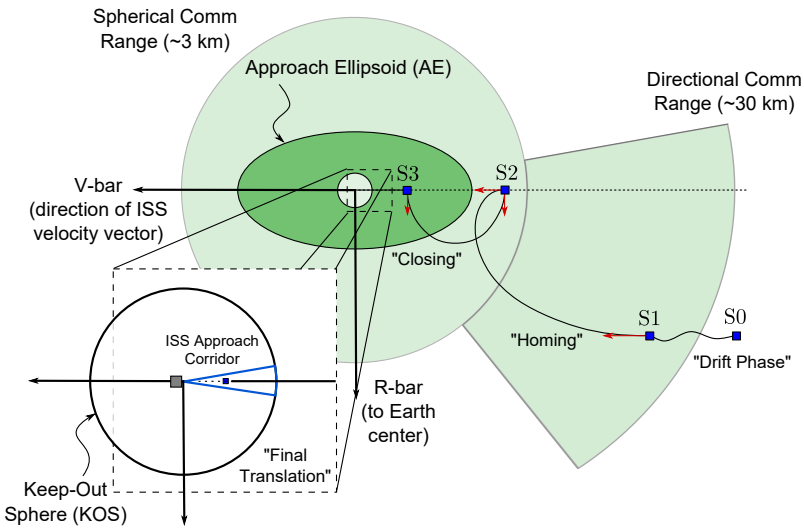


Figure 1.5: Example of ad hoc guidance, the state-of-the-art approach for most spacecraft missions. For rendezvous with the International Space Station (ISS), sequences of maneuvers are pieced together to meet mission requirements, according to trajectory designer experience and intuition. The approaching vehicle must avoid the station's approach ellipsoid (AE) and Keep-Out-Sphere (KOS) at various maneuver phases, proceeding from the points S0, S1, *etc.* (representing "Go/No-Go" points) if and only if given clearance by ground station operators.

ground infrastructure, and humans in the loop—all limitations that can be prone to error, drive up costs, and reduce access to space. Clearly, an alternative, more principled approach is needed to enable next-generation space missions in Earth orbit and beyond.

Advancements in autonomy have long been cited as the potential solution to many of these issues. NASA, for example, has repeatedly identified autonomous systems as an enabling technology over its history [14], beginning as far back as the Gemini program in the 1960's [15] and continuing up through the last decade with its previous Vision for Space Exploration [16] and its recent technological roadmaps for the future [17, 18]. One reason cited is that improved autonomy can transition space missions away from current ground-in-the-loop (geocentric) architectures towards self-sustainable and independent systems, a key requirement for improved extraterrestrial exploration [19] and for overcoming the many difficulties of interplanetary travel [20]. For example, to achieve orbital rendezvous or servicing about other planets or small bodies—feats that have to-date only been accomplished in Earth and lunar orbit—guidance decisions must be made entirely autonomously due to the large signal transmission delays to and from Earth, which can be as great as 26 minutes or more for missions near Mars and beyond (making such mission concepts as Fig. 1.4a next-to-impossible without onboard guidance and decision-making). Another reason is that autonomy has the potential to increase mission frequency, robustness, and reliability—particularly valuable for Autonomous Rendezvous and Docking (AR&D) and Autonomous Inspection and Servicing (AIS) operations about

the Earth [21, 22]. As space access improves through commercialization, the increased scheduling conflicts and labor overhead associated with ground-in-the-loop spacecraft guidance are expected to become prohibitively expensive, with the likelihood of human error heightened as well. Thankfully, provably-safe spacecraft autonomy can help circumvent these issues, as well as allow entirely new types of missions and improve the commercial and scientific return from space [23].

1.1.2 Recent Demonstration Missions

Despite its potential, autonomy within spacecraft guidance, navigation, and control has thus far been heavily restricted, with trajectory designers often opting for ground-in-the-loop architectures for maneuver planning and corrections whenever possible. This is often motivated by a predilection towards flight-proven heritage techniques, a need to reduce the costs of an expensive verification and validation process, and a lack of general guidance tools with built-in safety guarantees. Several attempts have been made, however, to introduce reliable autonomy into space applications (since as far back as the Space Race in the 1960's), especially in areas where ground-in-the-loop guidance has not been feasible. A few notable historical firsts include:

- **1st Autonomous Rendezvous and Docking**: The successful mechanical docking of Kosmos 186 and 188 in Low Earth Orbit (USSR, 1967) [10, Ch. 1.1]
- **1st Asteroid Soft Landing**: NEAR Shoemaker's descent onto Asteroid Eros (NASA, 2001) [24, 25]
- **1st Asteroid Sampling**: Several touch-and-go maneuvers by the Hayabusa spacecraft onto Asteroid Itokawa (JAXA, 2005) [26, 27]
- **1st Comet Soft Landing**: The descent and (intended) anchoring of the Rosetta mission's Philae lander onto Comet 67-P/Churyumov-Gerasimenko (ESA, 2014) [28, 29]

Other prominent examples of automated orbital operations include the undocking and re-docking of Orihime and Hikoboshi during JAXA's ETS-VII mission [30, 31], an autonomous inspection maneuver by AFRL's XSS-10 micro-satellite [32], the autonomous rendezvous, capture, berthing and Orbital Replacement Unit (ORU) transfer between DARPA's Orbital Express ASTRO and NEXTSat spacecraft [33], NASA's DART autonomous rendezvous demonstrator [34], and the highly-automated formation flying experiments of ESA's PRISMA spacecraft [35, 36]. Additional notable examples of autonomous small-body proximity operations include Russia's Phobos 1 and 2 [37] and NASA's Deep Space 1 [38] spacecraft. Finally, an early, successful example of autonomy applied to agile, opportunistic science is the NASA EO-1 spacecraft and its Continuous Activity Scheduling Planning Execution and Replanning (CASPER) system.

In spite of numerous successes, many of the missions mentioned above involved non-trivial guidance and control mishaps that resulted in significant mission degradation, and in some cases

Table 1.1: Compilation of guidance and control limitations and failures from several recent autonomous demonstration missions

Limited Autonomy	Deep Space 1, EO-1 (no collision avoidance) Orbital Express, PRISMA, XSS-10 (pre-planned or uplinked maneuvers)
Significant Hardware Failures	Deep Space 1 (star tracker failure) Hayabusa (multiple momentum wheel/RCS thruster losses) Phobos 2 (onboard computer failure)
Guidance/ Logic Errors	DART (missed guidance waypoint/navigation failure, collided with target) Deep Space 1 (anomalies in 1st experiment plan execution and support code) Hayabusa (bad descent logic/unexplained hovering/operator errors) Orbital Express (ranging software algorithm error)
Attitude Control Errors	ETS-VII (attitude thruster misfires) Orbital Express (optical sensor mishaps) PRISMA (occasional navigation error discontinuities)

catastrophic failure (see Table 1.1). The DART spacecraft, for instance, overshot one of its guidance waypoints and its collision-avoidance system failed following a navigation failure at just 200 m from its rendezvous target, the MULBCOM satellite. DART subsequently began using much more propellant than expected and initiated a series of maneuvers for departure and retirement, but eventually collided with MULBCOM at 1.5 m/s after depleting its propellant tanks [39]. The Phobos 2 mission also failed catastrophically after suffering an onboard computer failure which resulted in the loss of its two deployed landers, a mobile "hopper" and a stationary platform, above the Mars moon Phobos [37]. Finally, the Hayabusa mission was rife with guidance, logic, and hardware issues [40], though the mission was eventually salvaged after making clever adjustments. Several soft landing attempts failed due to: (i) anomalous optical navigation signals, and (ii) poor descent logic (for example, during one sampling attempt, Hayabusa experienced an unexplained 30 minute hover, then skipped its scheduled sampling sequence altogether when it entered a safe descent mode after detecting an obstacle too late for an abort and ascend). Worse still, its MINERVA minilander was lost due to operator error, when a command was sent to release the hopper during an ascent that Hayabusa had initiated autonomously to regulate its altitude [41]. To make matters more difficult, two momentum wheels malfunctioned [42] and several Reaction Control System (RCS) thrusters failed due to propellant leaks and frozen fuel lines, causing a complete loss of communication for seven weeks and a significant power outage from the resultant tumbling [40].

Unfortunately, these kinds of anomalies are not at all unique to these particular missions; one study found that 32% of 156 on-orbit spacecraft failures from 1980–2005 could be contributed to the Attitude and Orbital Control (AOCS) subsystem (though primarily from hardware failures) [43]. In addition, a number of GN&C-related anomalies have occurred during autonomous Space Shuttle operations [44] and other autonomous demonstrations. This points to the need for maturation in

autonomous spacecraft guidance, navigation, and control, suggesting that presently this field—even in static environments with well-understood dynamics—is still in its technological infancy [18, 45]. In particular, this highlights that any proposed autonomous guidance solution must be deterministically self-certifying and fault-tolerant in order to be realistically viable, or else guarantees cannot be made of mission safety. Had abort trajectories been available following the aforementioned onboard logic and hardware failures, for example, mission catastrophes like the DART spacecraft collision might have been easily avoided. This will be a key feature of the guidance solution proposed in this thesis.

1.1.3 Challenges

For autonomous maneuvering near space objects (cooperative or otherwise), the general guidance objective is to compute a state trajectory that safely brings the spacecraft as close as needed to its target object (including during docking maneuvers), while consuming as little propellant as possible (or at least, within its available budget) and simultaneously avoiding any nearby hazards. In general, this introduces many difficult, non-convex trajectory constraints into the optimal control problem given by 1.1 [46], which we represent implicitly through the time-varying sets of feasible controls $\mathcal{U}(t)$ and admissible states $\mathcal{X}(t)$. To illustrate why this is can be challenging, we provide a detailed list of specific examples in the paragraphs below.

Constraining Sensor Field-of-View For relative guidance purposes, it is often necessary to keep one or more target spacecraft, primitive bodies, or references (stars, planetary horizons, *etc.*) within the field-of-view (FOV) of onboard sensors. Assuming a radially-symmetric sensor bore/baffle, this can be represented mathematically as:

$$\hat{\mathbf{n}} \cdot \frac{\mathbf{r} - \mathbf{r}_{\mathrm{T}}}{\|\mathbf{r} - \mathbf{r}_{\mathrm{T}}\|} \geq \cos(\alpha), \tag{1.2}$$

where $\hat{\mathbf{n}}$ is the unit vector describing the sensor boresight, $\mathbf{r}$ is the position vector of the spacecraft, $\mathbf{r}_{\mathrm{T}}$ is the position vector of the target, and α is the cone half-angle defining the FOV. See Fig. 1.6a for an illustration. This constraint couples the attitude and translational dynamics through $\hat{\mathbf{n}}$, which is determined by the orientation of the spacecraft, making it a highly non-linear (and therefore very challenging) constraint to satisfy. To see this explicitly, if the position vectors are resolved in a rotating reference frame, *e.g.*, the Local-Vertical-Local-Horizontal (LVLH), and $\hat{\mathbf{n}}$ is resolved in a spacecraft body-fixed frame, then the left-hand side of Eq. (1.2) above can be re-expressed as:

$$\frac{(\mathbf{r} - \mathbf{r}_{\mathrm{T}})}{\|\mathbf{r} - \mathbf{r}_{\mathrm{T}}\|} \cdot (\mathbf{C}(\mathbf{q})\hat{\mathbf{n}}) \geq \cos(\alpha),$$

where $\mathbf{q}$ represents the attitude state of the spacecraft, and $\mathbf{C}(\mathbf{q})$ is the directional cosine matrix that takes a vector in the spacecraft body reference frame to the LVLH frame.

Avoiding Plume Impingement Impingement of thruster exhaust plumes on neighboring spacecraft poses a serious threat that can jeopardize sensitive optical devices and solar arrays, impart large disturbance forces and torques, and disrupt thermal blankets and coatings [47, 48]. Prevention requires restricting the thrusters that are currently oriented towards neighboring vehicle(s) from firing while the spacecraft is within a prescribed relative distance. Unfortunately, this imposes a loss in directional control authority and necessitates special guidance when in close proximity such that thrusters never direct exhaust towards (*i.e.*, apply thrust impulses directed away from) the target. This constraint exists for primitive bodies as well due to scientific contamination concerns, especially during sample return missions.

Represented mathematically, plume impingement constraints can be stated as, for thrusters $k = [1, \ldots, K]$:

$$u_k = 0 \quad \text{when} \quad \begin{cases} \|\mathbf{r}_k - \mathbf{r}_\mathrm{T}\| \leq R_\mathrm{plume} + R_\mathrm{T} \\ \frac{(\mathbf{r}_k - \mathbf{r})}{\|\mathbf{r}_k - \mathbf{r}\|} \cdot \left(\mathbf{C}(\mathbf{q})\hat{\mathbf{t}}_k\right) \geq \cos(\beta_\mathrm{plume}) \quad \text{for all } \mathbf{r} \in \mathbb{S}_\mathrm{T}, \end{cases}$$

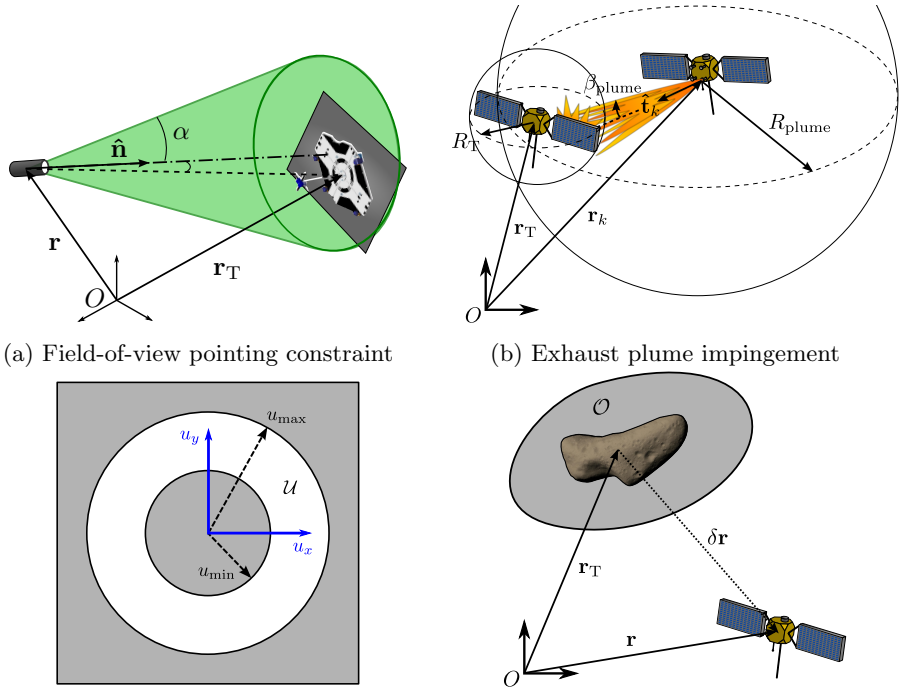
where K is the number of thrusters, u_k is the k-th thruster force command, $\hat{\mathbf{t}}_k$ is the unit vector for the k-th thruster nozzle direction in the spacecraft body-fixed frame (the negative of the thruster force direction), β_plume is the plume cone angle, R_plume is the maximum effective plume radius (*i.e.*, the plume is hazardous to the target if any part of its body lies inside this radius), R_T is the radius of the circumscribing sphere of the target, and $\mathbb{S}_\mathrm{T}$ represents all points on the surface of the target. As Fig. 1.6b attempts to illustrate, this is an extremely difficult constraint to embed in spacecraft guidance. Making sure exhaust plumes cannot hit the target requires checking for exhaust plume cone intersections with a detailed (or otherwise conservative) model of the target spacecraft body—for all but the simplest geometric representations, this is an expensive operation.

Handling Thruster Limitations (Impulse Bit Bounds) Due to propellant energy storage limitations and nozzle design constraints, all thrusters have a finite upper bound on the amount of force that they can provide. However, there is also a minimum nonzero force (called a minimum impulse bit) due to physical constraints from valving and chemical reaction times that imposes a lower bound on deliverable thrust; this means that arbitrarily small forces cannot be applied by thrusters (unless we fire them in opposition, although uncertainties in allocated thrust can make this imprecise). These bounds limit the control precision that can be achieved, which can be critical during docking and other proximity operations.

When using force commands for thrusters $k = [1, \ldots, K]$, these constraints can be expressed as:

$$u_k \in \{\{0\} \cup [u_{k,\min},\, u_{k,\max}]\},$$

where $u_{k,\min} > 0$ and $u_{k,\max} > u_{k,\min}$ are minimum and maximum thrust magnitudes. Note the



(a) Field-of-view pointing constraint

(b) Exhaust plume impingement

(c) Thruster control feasibility (2-norm bounds, *e.g.*, for gimbaled thrusters)

(d) Collision/obstacle avoidance

Figure 1.6: Illustrations of various spacecraft proximity operations guidance constraints and hazards

discontinuity in the interval of permissible thrust values. The only way to handle this constraint directly using mathematical programming is to introduce binary variables $\eta_k \in \{0, 1\}$ and continuous variables T_k, and rewrite our constraints in the form:

$$u_k = \eta_k T_k \quad \text{where} \quad \begin{cases} \eta_k \in \{0, 1\} \\ u_{k,\min} \leq T_k \leq u_{k,\max} \end{cases}$$

Unfortunately, the introduction of an additional K binary variables means we must resort to mixed-integer programming techniques, which suffer computationally when there are more than just a handful of binary constraints (especially with a non-linear optimal control problem like Eq. (1.1)). Even if we were to ignore the zero-value, the positively-valued lower bound $u_{k,\min}$ can make the problem non-convex (*e.g.*, when using gimbaled thrusters [49], as can be seen in Fig. 1.6c). For practical onboard applications, in absence of the sophisticated techniques presented in this dissertation, convex relaxations typically become necessary in such cases [50].

Avoiding Collisions Nothing can be more catastrophic to a spacecraft mission than a collision, which can damage or destroy participating vehicles and often marks an immediate mission failure.

For AR&D and AIS, a collision avoidance constraint can be described abstractly as follows:

$$\mathbf{C}(\mathbf{q})(\mathbf{r} - \mathbf{r}_{\mathrm{T}}) \notin \mathcal{O}$$

where $\delta\mathbf{r} = \mathbf{r} - \mathbf{r}_{\mathrm{T}}$ is the position vector of the spacecraft relative to a nearby obstacle (such as another spacecraft or a piece of debris), and $\mathcal{O}$ (an "obstacle" region) represents a set of relative positions in the obstacle body-fixed frame that lead to collisions. Note again the coupling between spacecraft translation and attitude caused by $\mathbf{C}(\mathbf{q})$, which is required to bring the vector $\delta\mathbf{r}$ into the obstacle's frame of reference. Refer to Fig. 1.6d for visualization.

The simplest choice for the set $\mathcal{O}$ for any given spacecraft-obstacle pair is a collision-avoidance ball $\mathcal{O} = \{\delta\mathbf{r} \mid \|\delta\mathbf{r}\| \leq R\}$, where R is some prescribed value large enough to take into account the unusual geometries of the spacecraft and obstacle. Unfortunately, "Keep Out Zone" ellipsoids are typically necessary in order to accommodate unequal dynamic uncertainties in different coordinate directions or to better approximate an elongated obstacle shape. For proximity operations around primitive bodies, this region $\mathcal{O}$ may need to be even more complicated due to the irregular and often ill-defined shapes of small bodies. Regardless of the particular choice for $\mathcal{O}$, avoidance constraints in spacecraft proximity operations are almost always non-convex, disallowing the use of convex programming to solve Eq. (1.1) unless the problem is transformed or approximated in some way.

Providing Required Thruster Silence Times When thrusters fire, large errors are introduced into state estimation due to process noise generated by sharp vehicle accelerations. As a result, it is not uncommon to require a prescribed period of thruster silence after each thruster firing (also called a "burn") to allow state estimator(s) to filter this noise and re-converge to a prescribed level of accuracy. One approach to imposing prescribed thruster silence is to force zero controls over preset time intervals during a maneuver, *i.e.*, for all thrusters k $= [1, \ldots, K]$, we require:

$$u_k(t) = 0 \quad \text{when} \quad t \in \bigcup_{j=[1,\ldots,N_s]} \mathcal{T}_j, \tag{1.3}$$

where $\mathcal{T}_j$ for $j = [1, \ldots, N_s]$ form a disjoint set of zero-thrust time intervals.

Limiting Propellant Use Every spacecraft mission is constrained by a finite supply of propellant that must be transported with the vehicle and its payload. The high cost of access to space currently inhibits the ability to refuel or resupply spacecraft, which effectively isolates them and imposes a mission lifetime synonymous with remaining propellant. This strict resource constraint also has a strong effect on mission capability. For example, AIS missions seek to maximize total inspection time, which corresponds directly to maximizing propellant efficiency. Likewise for small body operations, efficient propellant use implies not only longer observation times but also more opportunities for surface contact. This makes conserving propellant an absolute necessity during guidance.

Guaranteeing Safety Due to the high risks and economic costs associated with space missions, it goes without saying that safety is paramount during proximity operations, for both the maneuvering vehicle as well as all of its neighbors. Safety guarantees are typically partitioned into two categories: *passive safety*, in which zero-thrust coasting arcs emanating from points along the nominal guidance trajectory are certified as safe, or *active safety*, in which safe actuated abort sequences called Collision Avoidance Maneuvers (CAMs) are made available at any time [10, Ch. 4.4]. In either case, hard (deterministic) safety constraints are required to guarantee viable escape options in the event of thruster allocation errors (due to misfirings, "stuck-on" or "stuck-off" propellant valves, canted nozzles, *etc.*), unexpected environmental changes and disturbances, or even complete system shutdown. Often in practice this is achieved through ad-hoc open-loop trajectory design (guided by significant technical expertise; see Section 1.1.1 and Fig. 1.5 for details). However, an automated approach, potentially using optimal control techniques [45], positively-invariant sets [51, 53, 140], motion planning with safe samples [54], or some combination of all three [55], is needed in order to achieve truly autonomous guidance.

Handling Uncertainties Everything from imperfect orbit determination, unmodeled dynamics, and orbital perturbations to sensor drift, control inaccuracies, and signal time delays can introduce uncertainties into relative state knowledge and control implementation accuracy. For example, orbit determination errors make it very difficult to determine the initial set $\mathcal{X}_0$, in particular. Additional examples include mistimed or inaccurate thruster firings, aerodynamic drag perturbations in low Earth orbits, solar radiation pressure effects, and LIDAR or camera measurement noise. Over time, these uncertainties can induce sudden, unexpected violations in any of the aforementioned mission constraints. Hence it is extremely important to embed in autonomous guidance and control algorithms the capability to handle any expected uncertainty directly (*i.e.*, address all "known unknowns").

Unfortunately, though this is admittedly one of the most important factors in spacecraft control, it must be noted that we restrict ourselves in this dissertation to guidance without explicit disturbance modeling. By developing a real-time guidance algorithm for the undisturbed case, we expect that a certain level of robustness to disturbances can be imparted to the system by simply repeatedly refreshing guidance plans using a sufficiently-long horizon time and any new navigation information collected. Luckily, relative state accuracy typically improves as relative separation decreases (as additional sensors come into range, or as sensor signal-to-noise ratios improve); hence we expect that our results are likely to be viable on real systems, provided we restrict our attention to proximity operations (with relative separations on the order of a few hundreds of meters or less).

As can be seen, there are numerous potential sources for non-linearity and non-convexity in the autonomous spacecraft guidance problem, including collision avoidance, control feasibility, plume impingement, and sensor pointing constraints. Furthermore, coupling between translational and attitude dynamics, caused by the need to resolve vectors in and out of the spacecraft body-fixed

frame, requires including both translational and attitude states within the state transition function f of the system dynamics in 1.1. This complicates the problem further still, due to the inherent nonlinearities in the rigid-body attitude dynamic equations,

$$\mathbf{M} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) \tag{1.4a}$$

$$\dot{\mathbf{q}} = \mathbf{Q}(\mathbf{q})\boldsymbol{\omega} \tag{1.4b}$$

as seen from the cross-product term of Euler's equations and the matrix product of $\mathbf{Q}(\mathbf{q})$ with $\boldsymbol{\omega}$ (here $\mathbf{M}$ is the net external moment vector, $\mathbf{I}$ is the spacecraft inertia tensor, $\boldsymbol{\omega}$ is the spacecraft angular velocity, $\mathbf{q}$ is an attitude representation vector (*e.g.*, a quaternion/Gibbs/Rodriguez vector, exponential coordinates, *etc.*), and $\mathbf{Q}(\mathbf{q})$ is some matrix representation of $\mathbf{q}$-dependent terms). These equations lead to nonlinear equality constraints, and hence non-convex constraints, in the resulting numerical parameter optimization problem after discretization of 1.1. They also increase the number of variables required in our state vector $\mathbf{x}$, which can slow down computations and increase memory requirements.

To illustrate the difficulty this poses, if we could instead somehow express 1.1 as a convex optimization problem, then we would gain three immediate advantages: (i) global-optimality could be guaranteed [56, 57], (ii) a whole host of efficient tools, including Interior Point Methods (IPMs), would become available, and (iii) runtime execution speeds could be improved by 2-3 orders of magnitude [58]. This clearly motivates the use of real-time convex optimization for relative guidance whenever possible, either in the ideal case through lossless convexification (as in [59], for example) or through reasonable convex approximations, particularly for complex, difficult, or hazardous problems like proximity operations where the important need is a reasonably-good feasible solution obeying all mission constraints. Unfortunately, convexification approaches for spacecraft proximity operations are often unsuitable due to the errors incurred through constraint relaxation caused by the particular mathematical forms of their constraints. Hence new tools are needed. It is in this context precisely that sampling-based motion planning algorithms (as will be highlighted in Section 1.1.4 and discussed further in Chapter 2) have the potential to shine. In the next subsection, we introduce sampling-based planning and several other current state-of-the-art techniques that have been tailored specifically to handling problems with constraints of the kinds detailed here.

1.1.4 State-of-the-Art Approaches

Current state-of-the-art techniques for autonomous spacecraft proximity operations guidance include Apollo guidance (particularly phase-plane logic, glideslope, and sliding-mode controllers), Model Predictive Control (MPC) [60–64], and Artificial Potential Functions (APFs) [65–67]. Unfortunately, such techniques, while valuable in static uncluttered settings, appear to fall short in scenarios where optimization (*e.g.*, propellant minimization), logical modes (*e.g.*, safety modes), and time-varying

constraints (such as neighboring objects) become key features of the problem setup. In these cases, robotic motion planning techniques, though currently unproven in spaceflight, could serve as a valuable alternative [55, 68]. To provide a better contextual foundation for the guidance methodology proposed by this dissertation, brief synopses on each these methods are presented below.

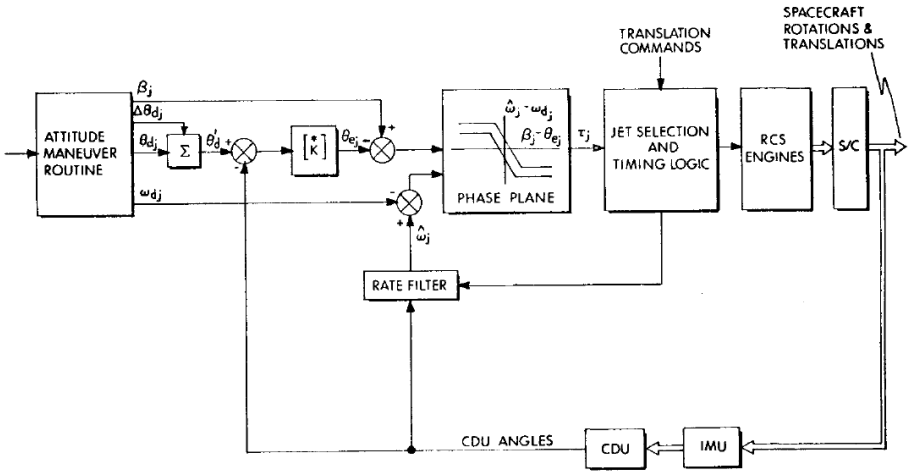
Apollo Guidance

Guidance techniques developed during NASA's Apollo Program form the basis for many of the standardized approaches to modern spacecraft guidance, still in use today some fifty years later. The techniques invented, now considered part of *classical* control, formed one of the earliest successful deployments of spacecraft autonomy. For example, the COLOSSUS Program, developed by MIT, called upon three specialized Digital AutoPilot (DAP) systems to stabilize and control the Apollo Command Service Module (CSM) as part of its Primary Guidance Navigation and Control System (PGNCS) [69]. Block-diagram schematics of the Apollo CSM control logic can be seen in Fig. 1.7. To provide context for our subsequent discussions of more sophisticated guidance techniques, we begin with a brief description of the designs of each of these Digital AutoPilot systems:

- **Orbital Re-entry Digital Autopilot (ENTRY DAP)**: Assumed control of the Command Module (CM) after separation from the Service Module (SM) and handled all Command Module flight maneuvers beginning with reorientation into Entry attitude up until drogue chute deployment. The autopilot called pairs of thrusters distributed along the rim of the base of the Command Module, as well as an additional pair near the tip for pitch-down control. The first phase of operation marked exoatmospheric mode, using various combinations of rate damping, attitude-hold, and attitude-control depending on the pitch angle value. Phase-plane logic controllers[1] (attitude rate versus attitude error) with biased deadzones drove the system to desired error tolerances. Once drag rose above 0.05g, atmospheric mode was initiated. In this regime, roll control was maintained using a complex phase plane incorporating a straight control line, maximum velocity boundaries, and constant-acceleration switching lines, while yaw and pitch reverted to rate-damping using a yaw rate versus roll rate phase plane logic and a simple relay with deadband, respectively. The purpose of ENTRY DAP was to maintain the component of lift in the trajectory plane needed to target a desired landing site given the vehicle's current position and velocity.

- **Reaction-Control System Digital Autopilot (RCS DAP)**: Responsible for controlling the attitude and attitude rates of the Command Service Module during coasting flight, both with and without the Lunar Module (LM) stage attached. The Digital AutoPilot employed

[1] Phase-plane controllers are typically used to determine stabilizing on-off control inputs for one degree-of-freedom differential systems by defining a coordinate plane of two state variables (typically a state error and its corresponding state rate error) and a set of *switching curves* with accompanying "deadband," "hysteresis," *etc.* in such a way as to partition the space into disjoint control regions that drive the system to within certain limits of the coordinate plane origin. Figure 1.7 shows a schematic of the phase-planes used by the Apollo missions.



(a) Apollo PGNCS RCS Automatic Control Logic, used by Reaction Control System thrusters to control CSM attitude. Here θ_d represents the reference attitude angle, θ_e the attitude error, β an attitude bias, ω the attitude rate, and $\hat{\omega}$ the attitude rate estimate.

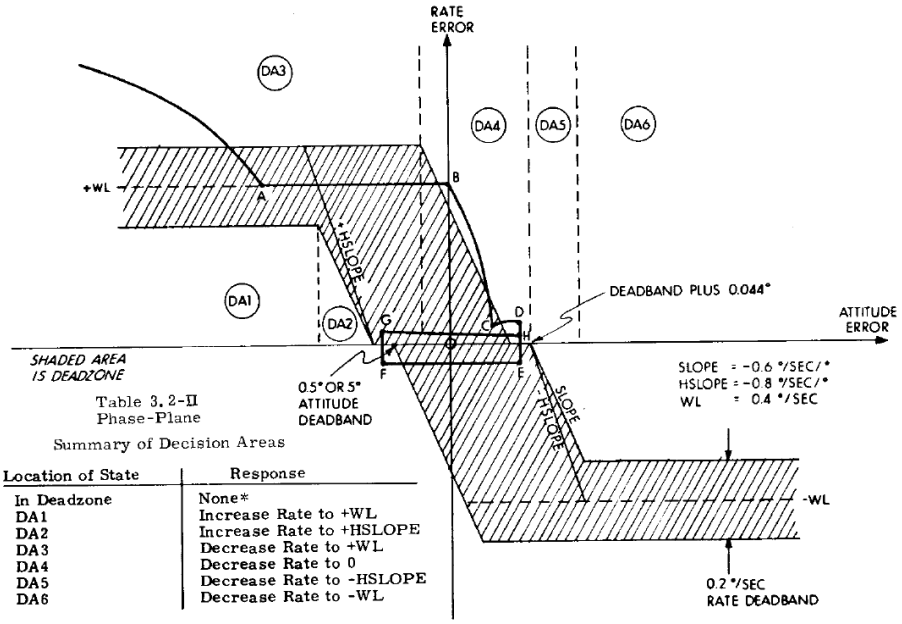


Table 3.2-II
Phase-Plane
Summary of Decision Areas

Location of State	Response
In Deadzone	None*
DA1	Increase Rate to +WL
DA2	Increase Rate to +HSLOPE
DA3	Decrease Rate to +WL
DA4	Decrease Rate to 0
DA5	Decrease Rate to -HSLOPE
DA6	Decrease Rate to -WL

(b) Apollo PGNCS Phase Plane Logic. For double-integrator models, this design can be shown to drive the rate and attitude errors plotted on the x- and y- axes to the box-like area near the origin. The logic works by breaking the plane into disjoint zones, inside of which the spacecraft is pre-programmed to torque positively or negatively (solid white areas) or coast (shaded region); horizontal lines represent zero-acceleration trajectories or "coasting arcs," while parabolas represent lines of constant acceleration.

Figure 1.7: Illustrations of one of the earliest successful spacecraft autonomous control systems, designed for the NASA Apollo Command Service Module (CSM). *Images courtesy of [69].*

four clusters, called quads, of four Reaction-Control System (RCS) thrusters each for pitch, yaw, and roll control, using a separate phase-plane logic controller for each rotational mode with nonlinear switching lines, a central deadband, and built-in hysteresis. The timing and firing commands of individual thrusters were issued by a thruster-selection logic responsible for resolving Digital AutoPilot rotation commands with translation commands and executing them as economically as possible according to the distribution of functional thrusters available. A second-order angular-rate Kalman filter was used to compute estimates of angular velocity by taking a weighted sum of: (i) extrapolated values of previous estimates, and (ii) derivations from gimbal angle measurements.

- **Thrust-Vector-Control Digital Autopilot (TVC DAP)**: Controlled the Command Service Module during powered flight, both with and without the Lunar Module attached. Pitch and yaw were adjusted by actuating the gimbal servos of the main engine, while a separate autopilot called TVC ROLL DAP controlled the Command Service Module attitude and rate about the roll axis during powered flight via the Reaction-Control System thruster quads. TVC DAP fed estimates of attitude rate and angle errors to pitch and yaw compensation filters, with various combinations of attenuation and phase stabilization depending on the configuration of the Command Service Module due to the changes in overall center-of-mass position, bending modes, and fuel slosh instabilities. TVC ROLL DAP used an adaptation to the phase-plane switching logic of RCS DAP in free flight, modified with ideal parabolic switching curves for roll axis attitude-hold within a small tolerance. A number of digital logical constraints were additionally implemented in order to conserve fuel and minimize the risk of thruster failures.

As can be inferred, significant application-specific tuning is required to achieve safe and reliable performance with phase-plane controllers and other elements of digital logic guidance systems. This makes these design approaches relatively inflexible to modifications in mission constraints and spacecraft properties, with every change requiring extensive testing and re-evaluation of the entire control system. Furthermore, these Apollo guidance controllers are tailored specifically to minimum-time (maximum-effort) cost objectives (called "bang-bang" control); heuristics for adjusting deadbands between phase-plane switching curves can encourage lower propellant use but inherently they will always be suboptimal. Though these techniques were highly successful back in the 1960's, many of the computational constraints placed on systems back then have been lifted with the advent of improved computer processing technology. As a result, we can now safely call on more sophisticated alternatives to minimize propellant and accommodate trajectory constraints in a more direct manner.

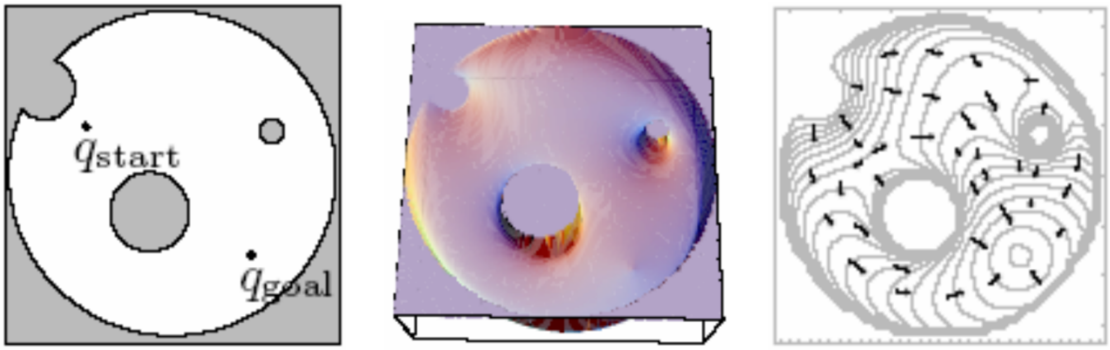
Model Predictive Control

Model predictive control (MPC) is a feedback law based on the repeated solution of an optimal control problem (as in Eq. (1.1)) that uses an assumed dynamics model f and the current state

of the spacecraft as its initial condition. This problem is solved to yield a finite-horizon control trajectory that optimizes the predicted state response over the duration of a prescribed planning period or *time horizon*. Once solved, however, only the initial control segment is actually applied, after which the problem is reinitialized and the process repeats until convergence to the goal. This characteristic renewal procedure over a repeatedly updated horizon is what gives MPC its other common names: *receding horizon* or *moving horizon* optimal control. This scheme allows the design of feedback controllers on the basis of nearly any open-loop optimal control approach, improving its robustness and imparting it the ability to handle disturbances and mitigate error growth. Even without prior disturbance modeling, one can demonstrate under appropriate assumptions that MPC can lead to closed-loop stability and state convergence to the target [70]. Other advantages of MPC include the ability to handle pointwise-in-time state and control constraints, the capability to withstand time delays, and reconfiguration in the presence of degradations and failure modes [71, 140]. As the robustness properties of MPC are contingent on fast resolvability, open-loop controllers for vehicle guidance are generally restricted to convex optimization routines. In relatively simple cases, Mixed-Integer Linear Programs (MILPs), solvers specialized for linear programs containing discrete (integral) decision variables, may also be used with MPC to accommodate simple logical constraints like mode switching and collision-avoidance [45, 46].

Artificial Potential Fields

The artificial potential field (APF) method [65–67] transforms the guidance problem into particle motion within a potential field. Attractive potentials are used for goal regions, while repulsive potentials are used for obstacles; the potential field experienced while occupying a particular state is then represented by the sum of individual terms. A gradient ascent/descent routine is often called to trace a feasible path from an initial state, which, when tuned appropriately, will safely circumnavigate neighboring obstacles and converge to a goal (see Fig. 1.8). Alternatively, an optimal control problem may be formed to plan a path that minimizes the path integral along the gradient force field (analogous to the principle of least action in physical systems). The approach benefits greatly from the ability to react in real-time to environmental changes through adjustments to individual potential functions. Some difficulty lies in tweaking potentials such that the spacecraft behaves as desired (*i.e.*, ensuring sufficient margin from obstacles, rapid convergence, *etc.*). However, the main drawback of APFs is their well-known susceptibility to converge to local minima, which cannot be avoided without additional heuristic techniques. This tendency can be mitigated by attempting random walks out of local wells, or instead relying on a global optimization routine for open-loop control, with an artificial potential function method called for closed-loop feedback (*i.e.*, trajectory-following, bubble methods [72], or real-time path modification [73], for instance). Though APFs could conceivably work well with pure state constraints like collision-avoidance and sensor field-of-view constraints, it is not clear how such methods can reasonably accommodate



(a) Creating a state-space representation (see also Fig. 1.3)

(b) Generating a potential field (low potentials are associated with goals, while high potentials represent obstacles)

(c) Obtaining a guidance solution by following potential field gradients

Figure 1.8: Visualizing the Artificial Potential Field guidance process. The guidance problem is explicitly mapped to the state space, over which potential fields are added to encourage motion towards goals and away from infeasible state regions. *Images courtesy of CMU.*

mixed state-control-time constraints like plume impingement avoidance or logical constraints like active abort safety with respect to control failures unless time and controls are incorporated into an augmented "state" space. However, due to the extremely high dimensionality introduced by such augmentation, APFs appear less suitable for spacecraft proximity operations.

Motion Planning Algorithms

Motion planning constitutes a class of algorithms used to generate sequences of decisions, called *plans*, that safely guide robots from given initial states to a set of target states called *goals*. The framework is sufficiently general that it applies equally well to spacecraft and rovers as it does traditional robots; fortunately, many algorithmic tools are now available due to its long and rich history within the field of robotics [74]. Motion planning techniques can be broadly classified into two categories: *exact* (combinatorial) algorithms and *approximate* (sampling-based) algorithms (the latter of which we describe in great detail in Chapter 2).

Exact approaches develop a strategy based on an explicit representation of the unsafe region of the state space, which allows them to guarantee a solution if one exists (see Fig. 1.3 for a comparison of an implicit representation, *e.g.*, $\mathcal{O}_1 \cup \mathcal{O}_2$, and its corresponding explicit representation, $\mathcal{X}_{\text{obs}}$, generated by mapping the locus of unsafe vehicle configurations in the workspace $\mathcal{W}$ into the state space $\mathcal{X}$). Techniques typically involve the formation of roadmaps, which are topological graphs that efficiently capture the connectivity of points in the admissible ("free") state space. Representative examples include cellular decomposition, planning between Voronoi cell centroids, and maximum-clearance roadmaps based on free-space skeletons [75–77].

Unfortunately, due to the computational complexity of generating explicit representations of obstacle regions in the state space (to be discussed in more detail in Section 2.3.1), exact algorithms are often limited to problems of low-dimensionality, polygonally-shaped obstacles, and static environments. Sampling-based algorithms, on the other hand, forgo explicit construction of the unsafe state space and instead explore pathways via sampling procedures combined with graph search, with safety verified by a "black-box" collision-detection routine. On the one hand, this yields computational advantages (see Chapter 2) and also decouples guidance from problem geometry; on the other, it leads to the unfortunate drawback that weaker notions of correctness and completeness must be tolerated—existence of solutions can no longer be guaranteed in finite time without drawing an infinite set of samples. Prominent examples of sampling-based algorithms include Probabilistic Roadmaps (PRM) [78], the family of Rapidly-Exploring Random Tree (RRT) algorithms [68, 79], and the Fast Marching Trees (FMT*) algorithm [80] together with its kinodynamic versions [81, 82]. Sampling-based motion planning algorithms such as these have been shown under mild conditions to quickly and uniformly explore the feasible solution spaces of constrained guidance problems. Some of them (*e.g.*, RRT* [79] and FMT* [80]) have the added benefit of *asymptotic optimality*; that is, they guarantee convergence to an optimal cost solution as the number of samples taken goes to infinity. See Chapter 2 for more details.

Numerous studies have already been conducted assessing the feasibility of sampling-based planning algorithms for realistic spacecraft proximity operation scenarios [54, 55, 68, 83, 84]. Though not yet flown on spacecraft hardware, their efficacy has already been proven in real-world systems with challenging dynamics, namely for the onboard guidance of urban vehicles during the 2007 DARPA Urban Challenge. Several winning entrants to the 60 mile autonomous urban driving race used motion planning as their primary guidance logic, including CMU's winning Boss car with Anytime-D*, Stanford's 2nd-place Junior car with hybrid A*, and MIT's 4th-place Talos car with Rapidly-exploring Random Trees (RRTs) [85–89]. The ability of these algorithms to handle such diverse constraints while providing robustness certificates in real-time applications appears promising for autonomous spacecraft control. As a result, we focus exclusively on sampling-based planning techniques throughout the remainder of this dissertation.

1.2 Thesis Contributions

This thesis develops a motion planning framework for fully-autonomous spacecraft proximity operations guidance that has the following major characteristics: (i) convergence towards a propellant-minimal solution, (ii) ability to handle a wide variety of typical proximity operations constraints, (iii) real-time implementability, and (iv) deterministic certificates that can guarantee infinite-horizon mission safety with respect to losses in control authority. Many standard techniques are available that can handle one or two these specifications at a time; however, few exist that can address all of them

simultaneously. By extending the Fast Marching Trees (FMT*) sampling-based motion planning algorithm [80] (as well as its novel bi-directional variant) to propellant minimization metrics—integrating several ideas from convex optimization, orbital mechanics, and nonlinear control—we develop a general, flexible framework that is able to tackle a wide range of spacecraft proximity operations problems. Particularly novel features of the solution presented in this thesis include:

- The first bi-directional sampling-based algorithm (dubbed the Bi-directional Fast Marching Trees algorithm, or BFMT* [90]) proven to achieve asymptotic optimality with an upper bound on its convergence rate
- The ability to handle thruster plume-impingement avoidance constraints and a propellant-minimization cost function expressed as a sum of velocity impulse magnitudes
- *Active* abort safety guarantees with respect to "stuck-off" thruster failures
- The allowance of non-trivial spacecraft geometries (no point-mass model or thruster configuration assumptions are imposed)
- Fast dynamically-constrained trajectory smoothing (which can be applied to any impulsively-actuated system with Linear Time Invariant dynamics)

Though our approach is developed within the context of near-circular orbit operations, we emphasize that the techniques of this dissertation apply equally-well to other impulsively-actuated proximity operations scenarios. Furthermore, we expect many of the other proposed sampling-based planning concepts, particularly those for vehicle safety and trajectory smoothing, can be easily extended to other dynamical systems as well.

1.3 Thesis Organization

The thesis dissertation is divided into a preliminary section and two thesis parts. The exposition begins in Chapter 2 with a broad overview of sampling-based planning, defining terminology used throughout the thesis and introducing the factors that both render sampling-based planning useful for trajectory optimization and which also require careful consideration when adapting it to spacecraft guidance. The first thesis part, which includes Chapters 3–4, introduces and rigorously evaluates the two sampling-based planning algorithms, FMT* [80] and BFMT* [90], that will ultimately form the foundation of our solution method. Experiments in simulation and onboard a free-flying robotic testbed illustrate their ability to solve optimal path planning problems in real-time across a number of state spaces. The second thesis part, which includes Chapters 5–8, describes the techniques used to adapt these algorithms to the guidance of dynamically-constrained spacecraft during proximity operations, leveraging the insights developed in the first part of the dissertation. Numerical experiments demonstrate the cost and run-time performance of the approach on a realistic

near-circular orbit rendezvous scenario [55, 91, 92]. For further details, we refer the interested reader to the individual chapter summaries included below.

Chapter 2: Sampling-Based Motion Planning Before presenting any research, we start by introducing a number of key concepts behind sampling-based motion planning, including the complexity of and assumptions behind motion planning, the basics of state space sampling and exploration, and the important factors to consider when adapting sampling-based approaches, originally developed for path planning without differential constraints, to spacecraft guidance.

Chapter 3: Real-Time Sampling-Based Path Planning This chapter introduces the Fast Marching Trees (FMT*) and Bi-directional Fast Marching Trees (BFMT*) algorithms for solving the shortest-path motion planning problem, a canonical scenario useful for both visualizing motion planning algorithms as well as benchmarking them with respect to state-of-the-art methods. The chapter's principal contribution is an in-depth presentation of and proof of asymptotic-optimality for the BFMT* algorithm, ostensibly the first bi-directional sampling-based path planning algorithm with *proven* asymptotically-optimal performance. BFMT*, a lazy, continuous-space analogue to two-source dynamic programming, extends the FMT* algorithm (originally developed by Janson et al. [80]) to bi-directional search while preserving its key properties, namely its asymptotic optimality (through convergence in probability) and its upper-bounded convergence rate. In addition to extensive proofs, we also note some useful extensions that can be made to handle more general cost functionals, which we make use of in later chapters of the thesis.

Chapter 4: Benchmarking Experiments and Testbed Demonstrations This chapter serves to motivate why the FMT* and BFMT* algorithms are necessary and useful tools, demonstrating their ability to outperform other state-of-the-art methods in certain high-dimensional topological spaces much like those encountered during spacecraft guidance. Results from several numerical experiments are presented, drawn from: (i) an open-source planning library designed to provide fair algorithmic comparisons within a unified testing framework, and (ii) hardware demonstrations performed on a set of air-bearing, free-flying spacecraft simulators. The experiments illustrate that FMT* and BFMT* perform in practice at least as fast as their state-of-the-art asymptotically-optimal counterparts, and sometimes significantly faster (particularly in high-dimensional, more tightly-constrained environments).

Chapter 5: Problem Formulation The thesis transitions here from shortest-path motion planning to our original problem of real-time, fully-autonomous guidance for provably-safe, propellant-efficient spacecraft proximity operations. This chapter presents a thorough mathematical description of the problem formulation—a specialization of the generic problem presented as Eq. (1.1). Details and motivation are provided on the models employed to represent our propellant-usage cost functional,

system dynamics, and proximity operations trajectory constraints, using near-circular orbit proximity operations as a reference example in preparation for our later numerical studies (though again we reiterate that our proposed solution framework applies much more generally).

Chapter 6: Vehicle Safety Of paramount importance to any proximity operations mission is the ability to deterministically ensure the safe execution of guidance trajectories. To that end, we present in this chapter a technique for embedding fault-tolerant safety constraints into sampling-based planning for impulsively-actuated systems. The approach certifies trajectory safety by ensuring a safe actuated abort maneuver is available at every point along the returned guidance solution, with guarantees on abort control sequence admissibility under all "stuck-off" control failures up to a desired fault tolerance. Abort trajectories are designed to terminate at safe, stable regions in the free state space at which they can remain for all time; this allows for infinite-horizon safety and improves the chances of mission recoverability.

Chapter 7: Real-Time Sampling-Based Spacecraft Proximity Operations This chapter describes our algorithmic approach for autonomous spacecraft proximity operations using the sampling-based method, combining several adaptations to the path planning algorithms of Chapter 3 together with the modeling of Chapter 5 and the active-safety guarantees of Chapter 6 into one cohesive framework. Computations are carefully divided into offline and online phases such that the online phase, the critical component for imparting situationally-aware onboard autonomy, can be run in real-time under reasonable memory requirements. Putting these components together, the approach represents one of the first proximity operations guidance frameworks for impulsively-actuated, non-point-mass spacecraft that simultaneously provides provably-safe solutions with respect to control failures, promotes low-propellant usage, and enables real-time implementability. Though tailored to near-circular orbit proximity operations, the methodology is easily generalized to other scenarios. Additional fast trajectory smoothing techniques with built-in execution time upper bounds are provided to give the mission trajectory designer additional tools for rapidly improving the quality of returned planning solutions.

Chapter 8: Numerical Experiments The proposed framework of Chapter 7 is tested in this chapter against a realistic near-circular orbit near-field rendezvous case study derived from a real-world mission scenario. Both planar and non-planar single-chaser/single-target maneuvers are studied, for which we present representative motion planning solutions (illustrating together the trajectories explored, the abort trajectories computed, and ultimately the solutions selected to solve the problem). Attempts are made to compare solution costs to lower-bounds and to evaluate the built-in active-safety and asymptotically-optimal properties of our guidance approach. Extensive trade studies are performed illustrating the balance between solution quality and execution time as a function of algorithm sample count n and a neighborhood sizing parameter $\bar{J}$, demonstrating the typical

trade-offs that mission planners can take advantage of when tailoring our guidance framework to specific real-world missions. In the end, we show that for the numerical case study considered here our approach yields low propellant-cost solutions using only a few thousand samples, returning a feasible, provably-safe guidance trajectory in on the order of seconds.

Chapter 9: Conclusions We conclude with a high-level summary of our proposed planning approach, in addition to a brief synopsis of the key points made throughout the thesis. The chapter continues with a thorough discussion of future research avenues that can bring sampling-based planning closer to implementation on real-world space missions, and finally closes with a vision for the future of this technology in next-generation spacecraft mission planning.

Appendix A: The Clohessy-Wiltshire-Hill Equations To keep the thesis self-contained, we provide in this appendix a detailed derivation of the Clohessy-Wiltshire-Hill dynamics model (its system dynamics equations along with their corresponding solution equations), in support of our problem formulation presented in Chapter 5.

Appendix B: Optimal Circularization Under Impulsive CWH Dynamics A key component of our approach to providing active safety under control failures in Chapter 6 is the ability to derive abort trajectories to safe, stable regions of the state space. For near-circular orbit proximity operations, one of the simplest approaches a spacecraft might take following a failure is to abort to circular orbits located sufficiently-far from any nearby orbital objects. This appendix details an analytical, propellant-optimal, one-burn solution to designing such maneuvers under the Clohessy-Wiltshire-Hill dynamics model.

Appendix C: Intermediate Results for the FMT* Optimality Proof The final appendix includes an enumeration of several lemmas and proofs used throughout the asymptotic optimality proof for our modified FMT* solution algorithm presented in Chapter 7. We list them in an appendix to clarify the exposition in the body of the dissertation.

Chapter 2

Sampling-Based Motion Planning

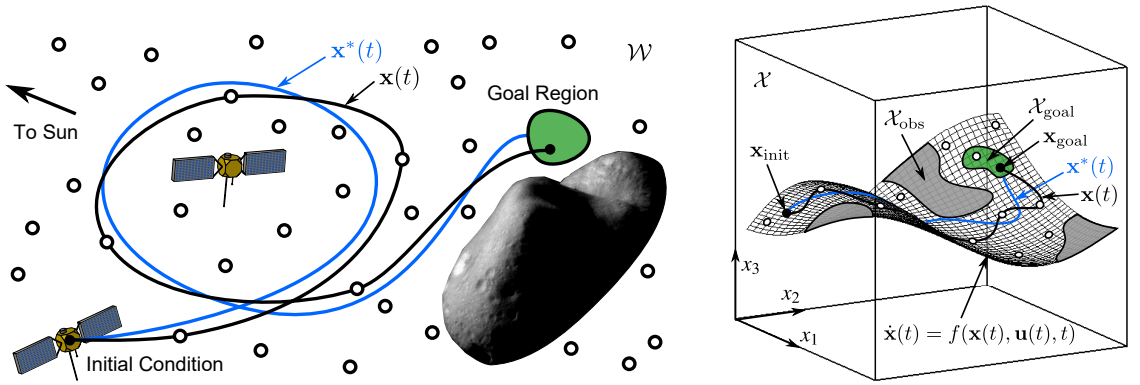
Before proceeding with the main body of the thesis, we briefly introduce a number of concepts related to standard sampling-based planning. The purpose of this chapter is to give the flavor of the methods, techniques, and issues behind sampling-based planners, in order to provide greater context for the research reported later on in this dissertation. We begin in Section 2.1 with a broad overview of the sampling-based approach and its potential advantages. We then follow with a list of major assumptions in Section 2.2 and a brief discussion of algorithm preliminaries in Section 2.3, before proceeding to sampling methods and exploration concepts in Section 2.4. We close in Section 2.5 with highlights of key considerations related to adapting sampling-based planning algorithms for real-time spacecraft control.

Note the majority of the material in this chapter is based on the seminal work of LaValle [74], to which we refer the interested reader for more details.

2.1 Overview

Sampling-based motion planning is an algorithmic process that essentially breaks down a complex continuous trajectory control problem like that of Eq. (1.1) into a series of relaxed, simpler Two-Point Boundary Value Problems (2PBVPs), which are evaluated *a posteriori* for constraint satisfaction and efficiently strung together into a graph (*i.e.*, a tree or roadmap). In this way, satisfaction of complex constraints, such as obstacle or plume impingement avoidance, is decoupled from trajectory generation (dynamic simulation), which can in many cases yield computational advantages. A simplified illustration of the process can be seen in Fig. 2.1, which uses a mock asteroid rendezvous mission for reference.

Suppose we want to maneuver the spacecraft shown in Fig. 2.1 to a relative position near the asteroid, while keeping power-positive by aligning solar arrays as much as possible with the sun vector and simultaneously avoiding plume impingement and collisions with the neighboring spacecraft



(a) Maneuvering from an initial condition to a goal region by passing through intermediate waypoints called samples. Many subtrajectories (not shown) between samples may be explored in order to discover the sequence used to generate $\mathbf{x}(t)$.

(b) Equivalent planning problem as represented in the state space. Planning must be confined to the vehicle's dynamic manifold while avoiding inadmissible state regions.

Figure 2.1: Illustrations of sampling-based motion planning in both the workspace $\mathcal{W}$ and its corresponding state space $\mathcal{X}$. Here the vehicle must plan a trajectory that avoids the nearby spacecraft and asteroid while keeping solar arrays oriented towards the sun. Due to the constraints, discovering a low-cost, feasible trajectory $\mathbf{x}(t)$ near the optimal solution $\mathbf{x}^*(t)$ in one shot is difficult; sampling-based planning pieces many local connections together in the hopes of finding solutions more quickly.

and asteroid surface. Calling a nonlinear trajectory optimization solver to solve this problem entails attempting the maneuver directly between our initial state $\mathbf{x}_{\text{init}}$ all the way to the goal region $\mathcal{X}_{\text{goal}}$—a feat that is likely to cause many failed search iterations due to the number and scope of our trajectory constraints (depending on the relative sizes of the feasible set and the full state space). From our contrived scenario, for example, it would be hard to determine that a loop around our neighboring spacecraft would be the most efficient path that satisfies our constraints while also syncing the vehicle's arrival position and velocity with the movement of the asteroid. Conceivably it could take a long time for the solver to discover this type of solution, unless the trajectory designer knew in advance how to best initialize it such that the nearest local minimum coincided with the global optimum. Though possible, assuming such intuition exists for finding a feasible solution (let alone an optimal one) is not very realistic for highly-constrained problems of type typically found in spacecraft proximity operations.

Sampling-based planning, on the other hand, constructs a series of intermediate solutions and strings them together to solve (approximately) the original full-scale problem. The main idea is to discretize the planning space, in this case the admissible state space $\mathcal{X}_{\text{free}}$, by taking a collection of well-distributed sample points, and build a graph that explores where the vehicle can travel safely. Once connection(s) have been established between our initial state $\mathbf{x}_{\text{init}}$ and some sample point $\mathbf{x}_{\text{goal}}$ within our goal region $\mathcal{X}_{\text{goal}}$, we simply take the best (lowest-cost) one available as our

guidance plan. By distributing sample points carefully, connecting only those samples that are *local* to each other, and saving only those subtrajectories (or graph "edges") between them that satisfy our constraints, the robot can eventually construct a sequence of feasible paths that terminate within the goal region. Though we started with one 2PBVP for our guidance problem and now we solve many of them, we gain an advantage by considering only local connections—most of these connections will likely be feasible given that we start and end at two neighboring feasible sample points. As a result, we can *relax* the 2PBVP (or "steering") problem that we use to interconnect sample points, ignoring our trajectory constraints and keeping only our dynamic constraints and boundary conditions. Our original trajectory constraints can then simply be checked after the fact (called *a posteriori evaluation*). This is the key behind sampling-based planning, and it yields two significant advantages:

- **A posteriori evaluation**: The most difficult-to-handle trajectory constraints can be relaxed from the subtrajectory 2PBVP and can simply be checked *a posteriori*.
- **Non-explicit free-space representation**: The problem constraints no longer need to be mapped from the workspace $\mathcal{W}$ to the state space domain $\mathcal{X}$ (that is, we need not represent $\mathcal{X}_{\text{obs}}$ explicitly), a computationally-prohibitive task for many complex planning problems (see Fig. 1.3 for a simple illustration of the process).

As a result, sampling-based algorithms can address a large variety of constraints while providing significant computational benefits with respect to traditional optimal control methods and mixed-integer programming. Furthermore, through a property called *asymptotic optimality* (AO), sampling-based algorithms can be designed to provide guarantees of optimality in the limit that the number of samples taken approaches infinity. Several decades of research in the robotics community [74] have already shown that sampling-based planning algorithms (dubbed "planners" throughout this thesis) show especially strong promise for tightly-constrained, high-dimensional optimal control problems, such as those encountered for spacecraft proximity operations. This makes sampling-based planners a strong choice for the focus of our work.

2.2 Assumptions

Suppose we wish to employ a sampling-based algorithm to solve Eq. (1.1), which we model as a *motion planning problem* represented by the tuple $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathcal{X}_{\text{goal}})$. Before we can call a sampling-based planner, we must make note of a few key assumptions:

- **Workspace/State Space Invariance**: The workspace $\mathcal{W}$ and its corresponding state space $\mathcal{X}$ are assumed to remain static in our dynamical reference frame of interest (this does not imply that the free space $\mathcal{X}_{\text{free}}$ is necessarily static, however). Problems for which this assumption is not reasonable typically define $\mathcal{W}$ and $\mathcal{X}$ by over-approximations or by static representations

derived from information available at the current time t_{init}, refreshing planning problems over time as necessary.

- **State Space Coverage**: The state space $\mathcal{X}$ is assumed to encapsulate all information relevant to the problem. This means that $\mathcal{X}$ must be large enough to include our initial state, goal region, relevant obstacles, and any intermediate maneuvers that might be required to reach the goal. Though seemingly obvious, when dealing with the relative and/or nonlinear trajectories of spacecraft and the vast array of perturbations they might experience, special care must be taken when deciding how large to make $\mathcal{X}$ so that maneuvers stay within its confines (where our problem is defined).

- **Boundary Condition Feasibility**: The initial state $\mathbf{x}_{\text{init}}$ and goal region $\mathcal{X}_{\text{goal}}$ lie entirely inside $\mathcal{X}_{\text{free}}$. If this is not the case, modifications to $\mathbf{x}_{\text{init}}$ or $\mathcal{X}_{\text{goal}}$ must be made (otherwise the planner must simply report failure and exit). This can be particularly troublesome in practice for systems with high degrees of state uncertainty, requiring appropriate smoothing and decision logic so that vehicle guidance remains unimpeded when boundary conditions lie near inadmissible state regions.

2.3 Algorithmic Concepts

With our assumptions established, we are now in position to describe the complexity of the motion planning problem, some basic limitations of the sampling-based approach, and a key property we use to guarantee a certain measure of solution quality.

2.3.1 Complexity

The first important consideration is the *complexity* of motion planning [74, Ch. 6.5]. Essentially, the question we can ask ourselves is: how much time or space, using the most efficient algorithm possible, does it take to solve all possible instances of a motion planning problem? Assuming a standard Turing machine model[1] and restricting our attention to the simpler case of path planning (that is, Eq. (1.1), but *without system dynamics* and assuming only time- and control-independent constraints), the answer turns out to be PSPACE-complete, implying it is both PSPACE and PSPACE-hard (which implies NP-hard) [93, 94]. Put into less technical terms, given a problem involving N bits of information, motion planning requires no more than a polynomial amount of time ($\mathrm{O}\big(N^k\big)$ for some integer k) to be reduced to and from a form that requires a polynomial amount of storage space to solve. The polynomial-space complexity refers to its PSPACE designation, while its polynomial-time reducability to PSPACE refers to its so-called PSPACE-hardness. These two attributes together

[1] The planning algorithm (considered a finite-state machine that writes bits to an unbounded binary string "tape") must determine whether a path exists (*i.e.*, accepts, with binary output TRUE) or not (*i.e.*, rejects, with binary output FALSE) for a given a problem instance (written as a binary string to an input "tape" fed to the algorithm).

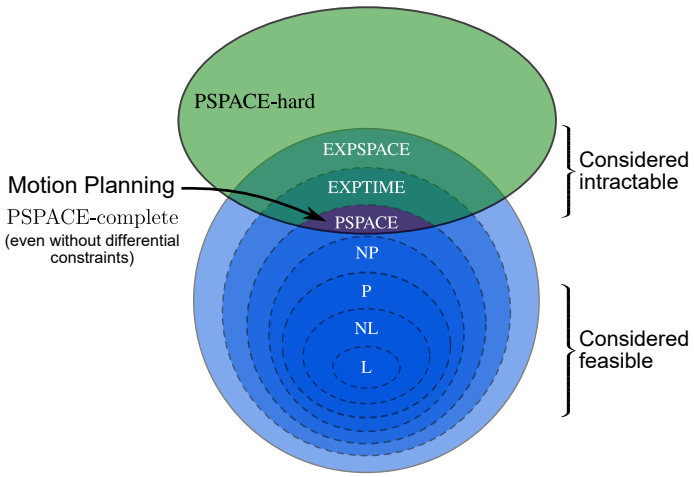


Figure 2.2: Representing algorithmic complexity class relationships with the complexity of path planning (motion planning without differential constraints)

are referred to as PSPACE-completeness. The NP-hardness implication refers to the fact that PSPACE-hard algorithms take polynomial-time to be verified on a non-deterministic Turing machine (effectively a machine that makes perfect choices while solving a problem instance).

Within the greater scope of algorithm complexity theory (see Fig. 2.2 for visual context among the various complexity classes), PSPACE-complete problems fall somewhere in between polynomial-time problems ($\mathrm{O}(N^k)$ for some integer k, designated P), considered *feasible*, and exponential-time and -space problems ($\mathrm{O}(2^{N^k})$ for some integer k, designated EXPTIME and EXPSPACE, respectively), considered *intractable*. It is not known yet whether they fall more towards feasibility or intractability, though the search has been on for decades for an efficient (polynomial-time) algorithm capable of solving these problems, so far to no avail. However, PSPACE-complete (NP-hard) problems are by general consensus considered very difficult to solve, at least through current computational techniques, because they can be verified in polynomial-time but only if the algorithm makes perfect choices along the way to finding a solution, which is impossible in practice without prior knowledge.

Hence, in short, motion planning is challenging, even for the simplest case in which we ignore system dynamics and remove time- and control-constraints entirely. This consequently puts a fundamental limitation on the performance of any sampling-based algorithm we might construct to solve a given instance of the optimal autonomous guidance problem presented as Eq. (1.1).

2.3.2 Completeness

Though sampling-based planners offer many benefits over other state-of-the-art methods, they do not come without limitations. Chief among them is that the notion of *completeness*, the ability to guarantee the return of a valid solution within a finite time if one exists or otherwise report failure, is

not possible for sampling-based algorithms. The issue stems from the fact that we are approximating a continuous planning space, $\mathcal{X}_{\text{free}}$, by a set of discrete sample points, $\mathcal{S}$. It is mathematically impossible for a sampling-based algorithm to rule out the existence of some feasible trajectory $\mathbf{x}(t)$ unless it first finds all feasible connections to every possible admissible state; because $\mathcal{X}_{\text{free}}$ contains an uncountably-infinite number of points, this requires taking the number of sample points n in $\mathcal{S}$ to infinity—a feat that is impossible in practice.

As a result, we settle for weaker notions of completeness, namely *resolution completeness* (when using deterministic sampling methods) and *probabilistic completeness* (when using randomized ones) [74, Ch. 5]. For more on sampling methods, see Section 2.4. A deterministic sampling-based algorithm that draws samples using a deterministic sampling sequence is said to be *resolution-complete* if it always returns a feasible solution in finite time provided one exists (it is not complete, however, because it may still run forever if one does not exist). It turns out that this will always be true so long as the algorithm both samples densely (in which case we can simply draw more samples to improve the effective graph resolution) and uses complete graph construction and search algorithms. Alternatively, a non-deterministic, also called probabilistic, sampling-based algorithm is said to be *probabilistically complete* if the probability of failing to return a solution, if one exists, decays to zero as the number of samples approaches infinity [95]. In other words, the probability of finding an existing solution converges to one as we draw more samples.

These two definitions of completeness prove to be much more theoretically appropriate for sampling-based planning. Furthermore, for most practical problems, calling a resolution- or probabilistically-complete sampling-based algorithm is sufficient for finding a valid solution, if one exists, so long as enough samples are taken and enough running time is allowed. It is important to keep in mind though that these algorithms always admit the chance of running forever (unless they are terminated early), as infeasible planning problems can never be verified in finite time by *any* sampling-based approach.

2.3.3 Asymptotic Optimality

Similar to the issue arising with completeness, sampling-based algorithms are required to sample an infinite number n of sample points in order to guarantee an optimal trajectory $\mathbf{x}^*(t)$. This limitation lies in the fact that the probability of sampling (either deterministically or randomly) from a set of measure zero is precisely zero; a 1-dimensional curve $\mathbf{x}^*(t)$ has no "volume" to sample from within a space $\mathcal{X}$ of dimension $d > 1$ (we neglect $d = 1$, in which case the planning problem is trivially solved). If n is sufficiently large, we can expect to find a sequence of samples within the neighborhood of $\mathbf{x}^*(t)$ but *almost surely* offset from it (off the trajectory itself); as a result, the trajectory obtained by connecting these samples together must necessarily yield a cost J_n greater than or equal to the optimal cost J^*, by definition of an optimal trajectory.

Though an infinite number of samples may be required, we can at least try to ensure that our

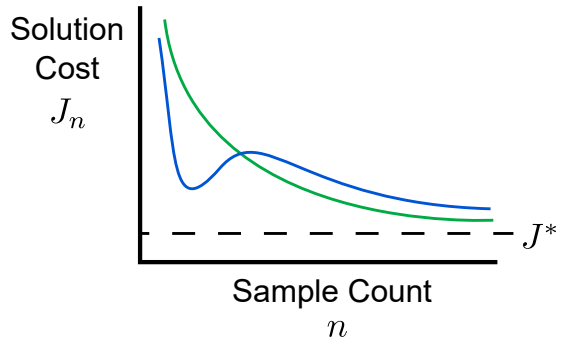


Figure 2.3: Examples of solution cost J_n convergence to the global optimum J^* for asymptotically-optimal planners as the number of samples n increases. Note asymptotic optimality does not guarantee monotonicity, only steady-state convergence.

planner converges to an optimal cost in the limit as n increases. This is captured by the following definition.

Definition 1 (Asymptotic Optimality)**.** Consider Eq. (1.1) with corresponding optimal cost J^*. Let J_n denote the cost of a trajectory returned by using n samples. A sampling-based algorithm is said to be *asymptotically optimal* if:

$$\lim_{n\to\infty} J_n = J^* \qquad \text{(Deterministic Sampling)} \tag{2.1a}$$

$$\mathrm{P}\left[\limsup_{n\to\infty} J_n = J^*\right] = 1 \qquad \text{(Random Sampling)} \tag{2.1b}$$

For an illustration of asymptotic optimality, refer to Fig. 2.3. Note that establishing this property will be critical to any approach we propose for autonomous guidance.

2.4 Sampling and Exploration

By the very nature of their name, sampling-based algorithms are highly-dependent on the particular sampling sequence used for state space exploration. In this section, we provide a brief background on sampling sequences and exploration methods, defining a number of key properties and metrics used to characterize the quality of sampling sequences and introducing a few of the more standard sampling and exploration techniques.

Note we distinguish between *sample sequences* and *sample sets*; a sample set $\mathcal{S}$ is a set of distinct vectors $\{\mathbf{x}_i\}_{i=1}^n$ (also called *points*, *samples*, or *nodes*, once incorporated into a graph) whose elements are drawn from a state space $\mathcal{X}$, while a sample sequence refers to an ordering in which a particular sample set is generated (*i.e.*, an ordering of the indices $i = 1, 2, \ldots, n$).

2.4.1 Sampling Properties and Metrics

We first examine three key properties, which will appear repeatedly throughout this thesis, that measure the quality of sampling sets and sequences.

Denseness

The first property, *denseness*, is one of the most fundamental for any valid sampling method [74, Ch. 5.2.1].

Definition 2 (Denseness)**.** Let $\mathcal{A}$ and $\mathcal{B}$ be any two subsets of a topological space (a set combined with a functional notion of neighborhoods). The set $\mathcal{A}$ is said to be *dense* in $\mathcal{B}$ if its closure $\text{cl}(\mathcal{A}) = \mathcal{B}$. A sampling sequence is said to be dense if its underlying set is dense.

In colloquial terms, this means that the elements of $\mathcal{A}$ come arbitrarily-close to the elements of $\mathcal{B}$. For a sampling-based algorithm to be complete (in terms of probabilistic or resolution completeness; see Section 2.3.2), its sampling method must produce a dense sequence in $\mathcal{X}_{\text{free}}$, or else there will exist portions of the free state space that are not covered by samples in the limit that $n \to \infty$.

Dispersion

The second property, *dispersion*, generalizes the notion of grid resolution to sample sets in continuous planning spaces [74, Ch. 5.2.3]. More specifically, it provides a measure of the spacing of discrete points within a metric space, a specific form of topological space for which the notion of distance between elements is defined through a so-called metric function ρ [96].

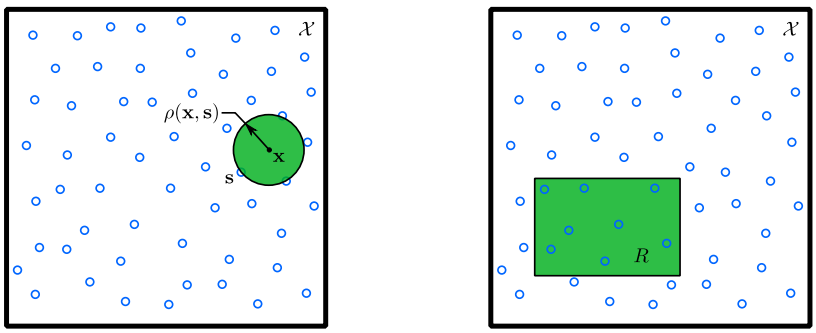
Definition 3 (Dispersion)**.** The dispersion of a finite set $\mathcal{S}$ of sample points in metric space $(\mathcal{X}, \rho)$ is:

$$D(\mathcal{S}) = \sup_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{s} \in \mathcal{S}} \rho(\mathbf{x}, \mathbf{s}) \tag{2.2}$$

In other words, the dispersion is the largest possible distance from points in $\mathcal{X}$ to their nearest neighbors in $\mathcal{S}$; *i.e.*, the radius of the largest possible empty ball we can fit in $\mathcal{X}$ without enclosing any elements of $\mathcal{S}$ (see Fig. 2.4a). As samples become more uniformly-distributed, this empty ball gets smaller (the dispersion decreases, *i.e.*, our "resolution" improves). This allows for finer exploration of trajectories in $\mathcal{X}_{\text{free}}$, which tends to yield higher quality solutions.

Discrepancy

One problem with dispersion is that it does not penalize alignments of points; for example, grids, which use a lattice of aligned, regularly-spaced samples that yield the minimum-possible dispersion, can give very high-quality solutions when they happen to align with narrow corridors in $\mathcal{X}_{\text{free}}$, or no solution at all otherwise. An alternative sampling metric that attempts to avoid this alignment-sensitivity problem is called sampling *discrepancy* [97], [74, Ch. 5.2.4].



(a) Sampling dispersion (ℓ_2-dispersion), the radius of the largest empty normed-ball

(b) Sampling discrepancy, related to the number of points that fall into subsets R in $\mathcal{X}$

Figure 2.4: Visualizing two standard measures of the denseness and uniformity of sampling sets

Definition 4 (Discrepancy)**.** Let $\mathcal{X}$ be a measure space, and define a range space $\mathcal{R}$ as some collection of subsets of $\mathcal{X}$. The *discrepancy* of a sample set $\mathcal{S}$ of n points with respect to $\mathcal{R}$ is defined as:

$$D_N(\mathcal{S}, \mathcal{R}) = \sup_{R \in \mathcal{R}} \left| \frac{|\mathcal{S} \cap R|}{n} - \frac{\mu(R)}{\mu(\mathcal{X})} \right| \tag{2.3}$$

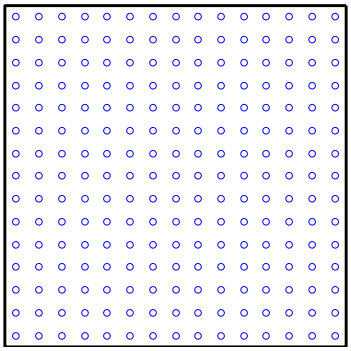
where $\mu(\cdot)$ denotes the Lebesgue-measure [96].

The idea here is to measure, given some region $R \in \mathcal{X}$, how closely the fraction of samples that fall into R, $|\mathcal{S} \cap R|/n$, comes to the actual volume fraction of R, $\mu(R)/\mu(\mathcal{X})$ (see Fig. 2.4b). It is desirable to keep these two fractions as close as possible to one another over all possible $R \in \mathcal{R}$, so that from a volumetric sense our samples are as evenly interspersed as possible. Just as for dispersion, a lower discrepancy implies a better "resolution."

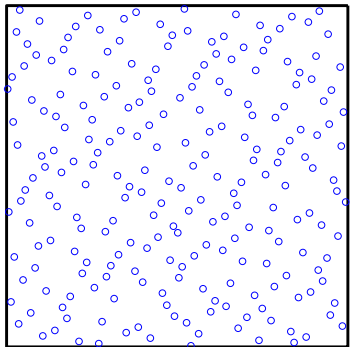
Note dispersion, a metric-based property, and discrepancy, a measure-based property, are closely related to one another. For example, consider the unit hypercube, $\mathcal{X} = [0, 1]^d$, together with the ℓ_∞-metric function for ρ. If we take $\mathcal{R}$ to be the set of all axis-aligned rectangular subsets of $\mathcal{X}$, then we have that $D(\mathcal{S}) \leq (D_N(\mathcal{S}, \mathcal{R}))^{\frac{1}{d}}$. Hence low-discrepancy implies low-dispersion (but not necessarily vice versa). In this thesis, we focus primarily on low-discrepancy sampling sequences.

2.4.2 Sampling Methods

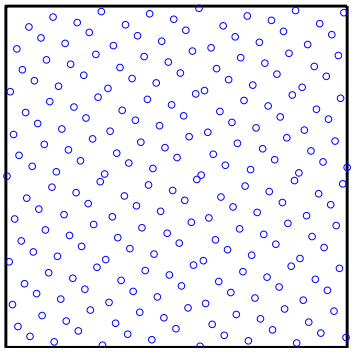
Sampling methods can be classified into two broad categories: deterministic sampling and pseudo-random sampling [74, Ch. 5.2]. The underlying sample sets produced by a handful of the more common sampling methods, both deterministic and pseudo-random, can be seen in Fig. 2.5. Deterministic methods use a repeatable algorithmic procedure to determine a sampling sequence, while pseudo-random sampling produces a sequence through repeated drawings from a probability distribution (*i.e.*,



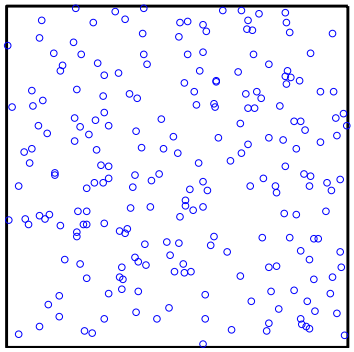
(a) Deterministic (Sukharev grid [98])



(b) Deterministic (Halton [99])



(c) Deterministic (Hammersley [100])



(d) Pseudo-random (Mersenne Twister [101])

Figure 2.5: Comparison of various sampling sequences used to discretize hyper-rectangular state spaces. All cases display exactly $n = 225$ sample points. Though sampling of 2D rectangles is shown, the techniques generalize to hyper-rectangles with an arbitrary number of dimensions; these often form the basis for sampling from other more complex topological spaces as well.

samples are realizations of the distribution's underlying random variable). Both techniques are widely-used, each sporting its own set of advantages and disadvantages. For instance, deterministic sampling offers more reliable performance than randomized sampling due to its deterministic nature, which can be important for some applications. Their sequences can also offer provably-better dispersion and/or discrepancy by their very construction [74, Chs. 5.2.3–5.2.4]. Randomized sampling, on the other hand, can make no guarantees that its sample sets will be well-spaced. One can easily show, however, that random sampling *almost surely* yields dense sample sets (with probability one) [74, Ch. 5.2.1], and often randomized methods run much faster and are much more straightforward to implement than deterministic techniques (this is especially true for spaces $\mathcal{X}$ more topologically-complex than mere hyper-rectangles or hyper-spheres).

For guidance and control verification and validation purposes, particularly regarding spacecraft mission planning, it is often imperative to ensure that guidance algorithms yield predictable, reproducible results—otherwise, there can be little certainty that an approach will work safely on-orbit.

Hence we restrict our attention primarily to deterministic methods. Throughout this thesis, unless otherwise specified, we make use of the deterministic, low-discrepancy Halton sequence [99] due to its widespread use and provably-good coverage of hyper-rectangular state spaces $\mathcal{X}$.

2.4.3 Exploration Methods

Sampling-based exploration can be very roughly divided into two basic types: tree-based search and roadmap-based search. In this subsection, we briefly describe each and weigh their relative benefits for autonomous guidance.

Tree-Based Search

Given a motion planning problem ($\mathcal{X}_{\text{free}}$, $\mathbf{x}_{\text{init}} \in \mathcal{X}_{\text{free}}$, $\mathcal{X}_{\text{goal}} \subseteq \mathcal{X}_{\text{free}}$), tree-based search involves constructing a set of search trees that expand outwards into unexplored territory in the free state space $\mathcal{X}_{\text{free}}$ until a feasible trajectory is found connecting $\mathbf{x}_{\text{init}}$ to some $\mathbf{x}_{\text{goal}} \in \mathcal{X}_{\text{goal}}$. Tree-based planning (illustrated in Figs. 2.6a–2.6c) typically proceeds as follows:

1. **Initialization**: Plant the "roots" to a number of trees, either one at $\mathbf{x}_{\text{init}}$ (called "unidirectional" search), one each at $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ (called "bi-directional" search), or several at more than two states (called "multidirectional" search).

2. **Expansion step**: Select a tree and one of its corresponding graph nodes for expansion.

3. **Local planning step**: Make connection(s) to neighboring unexplored samples. Unexplored samples(s) are either precomputed during initialization (so-called "batch" planners), or new ones are added (so-called "incremental" planners).

4. **Termination step**: Check the tree(s) for a solution. If the appropriate trees have interleaved together, solution(s) may be obtained. Select the cheapest one and exit, if desired, or continue if extra time is permitted for finding a better solution.

5. **Loop**: Repeat steps 2-4 (possibly with a different tree).

Well-known examples of tree-based algorithms include the Rapidly-Exploring Random Trees (RRT, RRT*) [68, 79], Expansive Space Trees (EST) [102], and Fast Marching Trees (FMT*) [80] algorithms.

The most difficult aspect of designing any tree-based exploration algorithm is in the local planning step. Most algorithms use state space sampling ($\mathcal{S} \subset \mathcal{X}$), in which case local planning must be able to solve a local two-point boundary value problem (2PBVP) to connect stored tree nodes to unexplored sample states. However, especially for systems with complicated dynamic constraints, it may be more useful to use control space sampling ($\mathcal{S} \subset \mathcal{U}$) instead. Here new nodes are added to the tree by applying a sample control trajectory from a given node and integrating our system

dynamics forwards (or backwards) in time. This is very easy to compute, and ensures tree edges are dynamically-feasible; however, in general, it is not clear how control trajectories should be selected to ensure adequate and rapid coverage of $\mathcal{X}_{\text{free}}$—tree branching may be much more extensive than in state-sampled planning (for reference, compare Figs. 2.6a and 2.6c), and growth could more or less take the form of a random walk if controls are not sampled properly. Additionally, if problems are too tightly-constrained, it may be too difficult to pass through narrow corridors and/or determine how to make rapid progress to the goal. Overall, the appropriate choice of local planning implementation is entirely problem-dependent; as we will see for the problem considered in this thesis, we find that state space sampling is most appropriate.

The choice of whether to use multiple trees depends on the nature of the problem. Because trees face an expected exponential number of nodes as a function of search depth (distance from the root), it can be advantageous to include additional tree roots; where to best initialize these trees, however, is a highly non-trivial problem. In some cases, extra trees can explore samples that otherwise need not have been included (see the bottom-right tree in Fig. 2.6b, for example). Similar choices must be weighed between incremental and batch sampling during the local planning step as well. As we will argue later in this thesis, batch sampling can be computationally advantageous when dealing with such complex dynamical systems as spacecraft.

Roadmap-Based Search

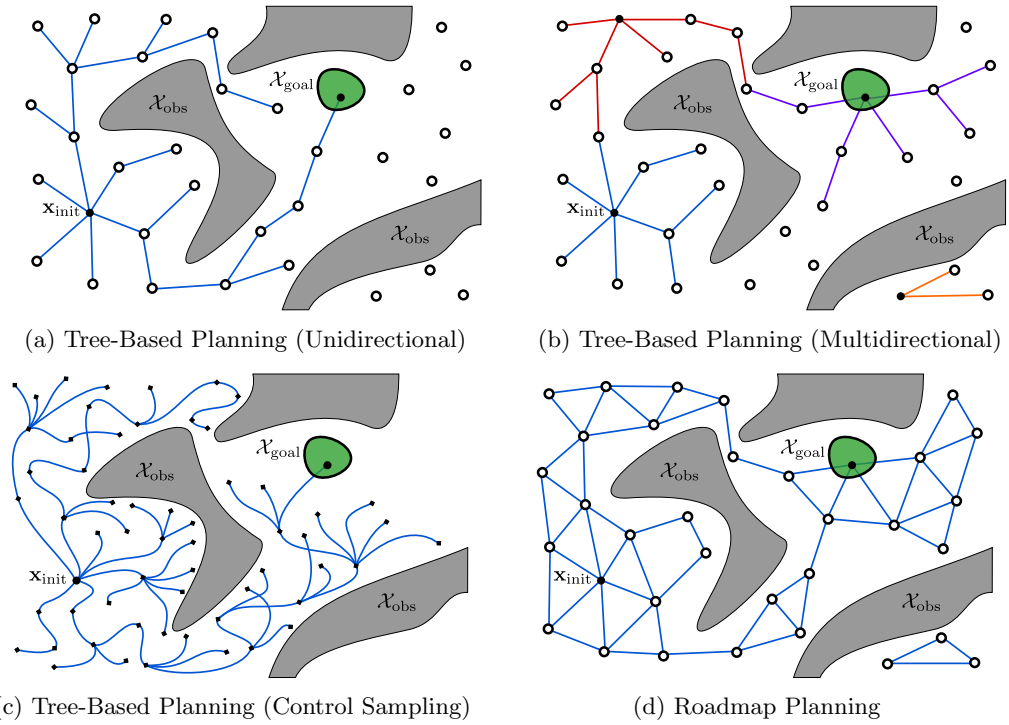
Tree-based algorithms are particularly well-suited to solving single-query problems (as is typical for maneuvering robots), in which a single initial-state/goal-state pair is given (*i.e.*, $\mathcal{X}_{\text{goal}} = \{\mathbf{x}_{\text{goal}}\}$). This is in contrast to roadmap algorithms, such as Probabilistic Roadmaps (PRM, PRM*) [78, 103], which invest substantial time constructing a full roadmap of the free state space $\mathcal{X}_{\text{free}}$, also known as a connectivity graph, so that future planning queries (which may involve boundary conditions $\mathbf{x}_{\text{init}}$ and $\mathcal{X}_{\text{goal}}$ very different from the current query) can be answered efficiently. Roadmap planning (illustrated in Fig. 2.6d) generally proceeds as follows:

1. **Precomputation phase**: Precompute a sample set and build a connectivity graph between all neighboring pairs of samples (using a local planning step similar to tree-based planners).

2. **Query phase**: Given a particular $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ pair, incorporate them into the connectivity graph using the local planner. Run a standard weighted graph search algorithm (depth-first search, breadth-first search, Dijkstra's algorithm, A*, D*, *etc.*) to determine the cheapest trajectory between them.

3. **Loop**: Repeat the previous step with new $(\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ pairs, as needed.

The connectivity graph produced by roadmap planners gives them a key advantage over tree-based exploration. Tree data structures inherently depend on their roots; if these roots change, particularly



(a) Tree-Based Planning (Unidirectional)

(b) Tree-Based Planning (Multidirectional)

(c) Tree-Based Planning (Control Sampling)

(d) Roadmap Planning

Figure 2.6: Comparison of various exploration methods commonly used by sampling-based planners

for systems with dynamic constraints, very little can be done to reuse previously-computed state space information. Roadmap planners need only recompute new local connections to the roadmap and run a new query to solve the perturbed problem. The drawback of roadmap planners is naturally in their exhaustive exploration of the state space, which can be wasteful if all one cares about is solving a single planning query. As most spacecraft typically move in time-varying environments, free space connectivity is not as useful beyond the current planning horizon (or even beyond the current time step). As a result, roadmap information is expected to quickly become irrelevant and obsolete, making such planners likely much less useful than tree-based planners for our particular guidance problem.

2.5 Adaptations for Spacecraft Proximity Operations

Despite their numerous potential benefits to spacecraft guidance, sampling-based planners cannot be applied to propellant-efficient proximity operations without modification. Several extensions must be made in order to accommodate a minimum-propellant cost functional, spacecraft system dynamics, and proximity operations mission constraints. The purpose of this section is to briefly present the difficulty of this task, which we address progressively throughout the remainder of this thesis.

Extensions Required First and foremost, the sampling-based framework must be tailored to the spacecraft guidance problem, to be detailed in Chapter 5. Necessary extensions must be made to each of the following areas:

- **Cost Functional**: Sampling-based planners were originally conceived to solve the shortest-path motion planning problem (to be presented in Chapter 3), a straight-line path planning problem which corresponds to using an arc-length (Euclidean distance) cost functional. The framework will need to be adapted to propellant-minimization. Establishing asymptotic optimality under such a cost metric will be the most challenging aspect.
- **Steering**: As described in the Chapter 2 introduction and Section 2.4.3, a technique is needed to connect two state space samples $\mathbf{x}_0$ and $\mathbf{x}_\mathrm{f}$ together with a dynamically-feasible trajectory. For path planning, a simple straight-line is drawn. For spacecraft, however, a more complex two-point boundary value problem (2PBVP) must be solved. Such problems can be computationally expensive to solve online, particularly if we plan to build an entire graph that potentially involves thousands of such connections.
- **Neighborhoods**: A key component of the local planning step discussed in Section 2.4.3 is establishing connections to local unexplored sample states. Given a point $\mathbf{x}_0$ and an unexplored set of samples $\mathcal{V}_\mathrm{unvisited}$, we must be able to efficiently identify all neighbors to $\mathbf{x}_0$. A redefinition of the concept of neighborhoods will be necessary in terms of our cost-to-go function (propellant use) from $\mathbf{x}_0$.
- **Constraint Satisfaction**: The "collision detection" module as used in traditional planners must be extended to include all trajectory constraints, including all control constraints and any mixed state-time-control constraints.

Enabling Real-Time Implementation Another extremely important consideration is the framework's real-time implementability. To better handle uncertain state information and account for the vast array of model uncertainties and control disturbances possible during spacecraft proximity operations, we must be able to produce solutions within on the order of seconds, though the faster the better, so that guidance trajectories may be repeatedly updated as state estimates improve and new environmental information (like the relative positions of nearby spacecraft) is obtained. To that end, the framework will likely require:

- **Efficient Exploration**: A fast, asymptotically-optimal, single-query planning algorithm (see Section 2.4.3) is needed to determine guidance trajectories in real-time (on the order of seconds or less, to account for fast attitude state dynamics).
- **Trajectory Smoothing**: It is often desirable to improve sampling-based planning trajectories in post-processing, if extra time allows. Trajectory smoothing algorithms improve solution

quality by locally adjusting a given feasible planning solution (often by perturbing intermediate trajectory waypoints located at graph nodes). An efficient, generalized technique for dynamically-constrained systems would be useful for reducing the propellant-cost of spacecraft planning trajectories.

- **Coding Optimizations**: Strict onboard memory limitations and run-time constraints demand a clean and efficient coding implementation for any spacecraft guidance approach. As a result, any techniques developed to improve planning run-time (such as caching) must not be to the significant detriment of memory usage, and vice versa. Clever heuristics may be needed to ensure execution times are met, such as moving computations offline, applying real-time planning [104] and neighborhood detection [105], and minimizing unnecessary constraint-checking through lazy search.

Part I

Real-Time Algorithms for Optimal Motion Planning

Conventionally, the term *motion planning* refers to the computation of paths that guide systems from an initial state to a set of goal states around nearby obstacles, while possibly optimizing a path length objective. Many algorithmic tools have been developed for such motion planning problems due to their rich and extensive history in the field of robotics (we refer the interested reader to [74] and references therein). In this part of the thesis, we examine one such set of tools that can be used to solve this special class of problems. Though different from our original spacecraft guidance problem, these path planning problems represent canonical scenarios for visualizing the planning process and for comparing algorithmic techniques. As will be seen, the insights developed here will serve as useful stepping stones towards achieving fully-autonomous spacecraft guidance.

Among the plethora of motion planning tools available to-date, *sampling-based algorithms* are arguably among the most prevalent. Notable examples include the Probabilistic Roadmap (PRM) algorithm [78], the Expansive Space Trees (EST) algorithm [102, 106], and the Rapidly-Exploring Random Tree (RRT) algorithm [68]. These planners and their many variants are well-known for finding feasible paths very quickly, especially in cluttered, high-dimensional spaces. Ever since their initial development, however, there has been interest in improving the "quality" of returned solutions; research efforts subsequently led to asymptotically-optimal (AO) versions of RRT and PRM, named RRT* and PRM*, respectively, whereby the cost of their solution paths converge almost surely to the global optimum as the number of samples approaches infinity [103, 107]. Many other asymptotically-optimal planners have since followed, including BIT* [108] and RRT$^\#$ [109], to name a few. More recently, a conceptually-different asymptotically-optimal, sampling-based motion planner has been introduced, called the Fast Marching Trees (FMT*) algorithm [80]. Numerical experiments have suggested that FMT* converges to an optimal solution faster than PRM* and RRT*, especially in very high-dimensional state spaces and in scenarios where collision-checking is expensive.

Though solution quality is of great importance in many problems, it is often improved at the expense of computational execution time. For instance, though asymptotically-optimal algorithms guarantee eventual convergence to optimal solutions, they may require increasingly-large numbers of samples before any significant cost improvements are realized. Luckily, it has long been known that incorporating *bi-directional* search (attempting connections both from the initial state towards the goal, as well as from the goal back towards the initial state) into planning algorithms can dramatically increase their convergence rate, prompting some authors [75] to advocate bi-directionality for accelerating essentially any (single-query) motion planning problem. This idea was first rigorously studied in [110] and later investigated, for example, in [111, 112]. Early algorithms include [75, 110–112], which collectively belong to the family of *non*-sampling-based approaches and are more or less closely related to bi-directional implementations of Dijkstra's algorithm [113].

More recently, and not surprisingly in light of these performance gains, bi-directional search has been merged with the sampling-based approach, with RRT-Connect and the Single-query Bi-directional Lazy collision-checking planner (SBL) representing the most notable examples [114,

115]. Though these bi-directional versions of RRT and PRM are probabilistically-complete, they unfortunately lack optimality guarantees. The next logical step in the quest for fast planning algorithms, therefore, is the design of sampling-based algorithms that are *both* bi-directional and asymptotically-optimal. To the best of our knowledge, the only available results in this context are [116] and the unpublished work [117], both of which discuss bi-directional implementations of RRT*. Neither work, however, provides a mathematically-rigorous proof of asymptotic optimality starting from first principles.

Objective Accordingly, the first objective of this thesis part is to propose and rigorously analyze such a bi-directional and asymptotically-optimal sampling-based algorithm, with the goal of providing an additional tool for real-time motion planning. For this, we propose a novel bi-directional version of the FMT* algorithm, which we choose due to its demonstrated potential relative to other state-of-the-art methods. Consequently, the second objective of this thesis part is to highlight experiments, both from numerical simulations and onboard a robotic test platform, that investigate the computational performance of these algorithms. Though development and testing are conducted within the context of shortest-path motion planning (without differential constraints), we develop a number of key insights that will help extend these algorithms to dynamically-constrained vehicle planning. This will later be tailored in Part II to the specific case of spacecraft proximity operations.

Organization The thesis part is divided into two chapters, one per objective. Chapter 3 introduces FMT* and our proposed bi-directional version, followed by a rigorous theoretical characterization of its asymptotic optimality and convergence rate properties. Chapter 4 follows with path planning simulations comparing these algorithms in several diverse planning environments to two other state-of-the-art asymptotically-optimal sampling-based algorithms, RRT* and PRM*. The chapter also illustrates how to implement these kinds of path planning algorithms for real-time autonomous guidance, evaluating FMT* onboard a free-flying robotic testbed designed to emulate planar, "deep-space" (zero-gravity) spacecraft proximity operations. Special emphasis is placed on the practical benefits and disadvantages of these planners, the limitations of using naive path planning guidance for spacecraft applications, and the factors that will need to be addressed in Part II.

Note, for the interested reader, the work presented in this part of the dissertation may also be found in conference paper [90] and its corresponding extended version [118].

Chapter 3

Real-Time Sampling-Based Path Planning

This chapter introduces the Fast Marching Tree (FMT*) and Bi-directional Fast Marching Tree (BFMT*) algorithms.[1] These algorithms are designed to solve challenging motion planning problems in real-time and with proven guarantees on solution quality and run-time performance. FMT*, originally proposed by Janson et al. [80], is essentially a "lazy" sampling-based planner analogous to a forward dynamic programming recursion over a set of well-distributed samples in the free state space. Because FMT* has been shown to work well on high-dimensional problems with many constraints, it appears to be a promising choice for spacecraft guidance. However, as a unidirectional tree-based algorithm similar to breadth-first search, FMT* faces an expected exponentially-increasing number of iterations as a function of search depth; for certain motion planning problem geometries, this may be inefficient. In an effort to provide an alternative search technique while preserving the many other desirable properties of FMT*, we propose in this chapter a novel bi-directional version of the algorithm called Bi-directional Fast Marching Trees (BFMT*), which at its core resembles a continuous (non-discrete) version of two-source dynamic programming. To the best of our knowledge, BFMT*, an original contribution of this thesis, is the first tree-based sampling-based planner that provably combines asymptotic optimality with bi-directional search (see Fig. 3.1).

The chapter opens in Section 3.1 with a detailed presentation of the shortest-path motion planning problem (a.k.a., the optimal "path planning" problem). We then formally introduce FMT* and BFMT* in Section 3.2, first providing high-level descriptions of each before presenting their pseudocode representations and corresponding implementation details. We follow in Section 3.3 with a rigorous proof of asymptotic optimality for BFMT* (derived from concepts initially developed in [80]), along with a characterization of its convergence rate. Note that though the convergence rate of

[1] The asterisk *, pronounced "star", is intended to represent asymptotic optimality much like for the RRT* and PRM* algorithms.

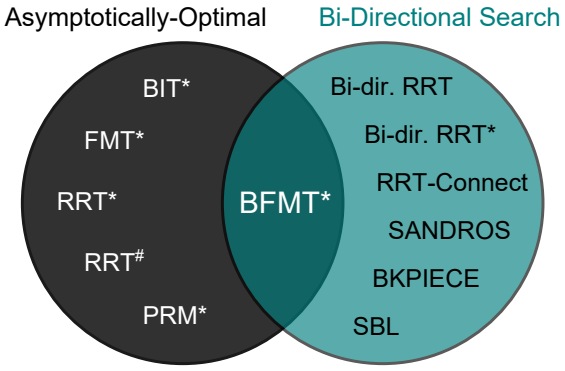


Figure 3.1: Contributions of the BFMT* algorithm in comparison to other state-of-the-art planners. To the best of our knowledge, BFMT* represents the first sampling-based planner proven to combine asymptotically-optimality with bi-directional search.

FMT* in [80] is proven only for obstacle-free state spaces, we generalize that result here to allow for the presence of obstacles. Finally, we close the chapter in Section 3.4 with a few concluding remarks.

Importantly, though the proof presented in this chapter is given with respect to probabilistic (pseudo-random) sampling, recent results demonstrating the optimality of FMT* with respect to deterministic sample sequences yield automatic extensions of deterministic asymptotic optimality to BFMT* as well [119]. These *deterministic* versions of FMT* and BFMT* will turn out to be critical features of our spacecraft guidance methodology later on in the dissertation.

3.1 The Path Planning Problem

We start by precisely defining the path planning problem, a simplified version of the generalized motion planning problem (Eq. (1.1)) that attempts to find the shortest feasible path between two points in state space $\mathcal{X}$ (a topological space, also called a *configuration* space[2] within the path planning community). Path planning effectively neglects control and time constraints, as well as system dynamics; one can view path planning as a purely state-constrained guidance problem for systems with the infinite control authority to change direction instantaneously. We examine path planning in this chapter as a useful precursor to our original vehicle guidance problem, as the principles developed here will ultimately form the foundation of our spacecraft guidance solution—as a result, we will need a real-time path planning algorithm if we have any hope of guiding vehicles in real-time. Note that the path planning problem itself is not trivial, however, due to its PSPACE-complete (NP-hard) algorithmic complexity, as described in Section 2.3.1.

To begin, let $\mathcal{X}$ be a d-dimensional state space, and let $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ be its associated obstacle

[2]We use the term "state" instead of "configuration" in this chapter to maintain consistency with the rest of this dissertation, though the term "configuration" traditionally refers to generalized *positions* and excludes velocities, mass, *etc.*, unlike states which can encompass anything. The differences in practice are purely semantic, however.

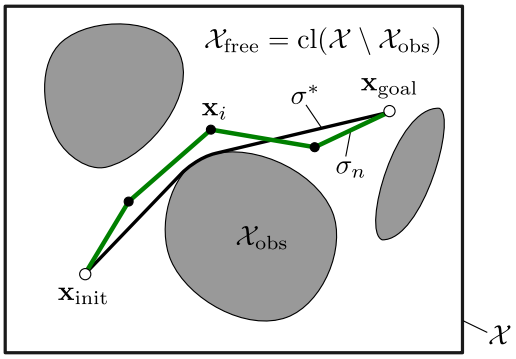


Figure 3.2: Visualizing the path planning problem. Sampling-based algorithms attempt to find waypoint states $\mathbf{x}_i$ between states $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ through which straight-line paths may be connected to approximate an optimal shortest path σ^* around infeasible or "obstacle" regions $\mathcal{X}_{\text{obs}}$ in the state space $\mathcal{X}$.

region, such that $\mathcal{X} \setminus \mathcal{X}_{\text{obs}}$ is an open set (we consider $\partial\mathcal{X} \subset \mathcal{X}_{\text{obs}}$). Denote the obstacle-free space as $\mathcal{X}_{\text{free}} = \text{cl}(\mathcal{X} \setminus \mathcal{X}_{\text{obs}})$, where $\text{cl}(\cdot)$ denotes the closure of a set. A path planning problem, denoted by a triplet $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$, seeks to maneuver from an initial state $\mathbf{x}_{\text{init}}$ to a goal state $\mathbf{x}_{\text{goal}}$ through $\mathcal{X}_{\text{free}}$. Let a continuous function of *bounded variation* $\sigma : [0, 1] \to \mathcal{X}$, called a *path*, be *collision-free* if $\sigma(s) \in \mathcal{X}_{\text{free}}$ for all $s \in [0, 1]$. A path is called a *feasible* solution to the planning problem $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ if it is collision-free, $\sigma(0) = \mathbf{x}_{\text{init}}$, and $\sigma(1) = \mathbf{x}_{\text{goal}}$. Let Σ be the set of all paths. A cost function for the planning problem $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ is a function $J : \Sigma \to \mathbb{R}_{\geq 0}$ from Σ to the nonnegative real numbers; in this chapter, we consider the cost function $J(\sigma)$ to be the *arc length* of σ with respect to the Euclidean metric in $\mathcal{X}$ (the extension to general cost functions will be briefly discussed in Section 3.3.3).

With these definitions in mind, we can finally introduce the optimal path planning problem through the following definition.

Definition 5 (Optimal Path Planning Problem)**.** Given a path planning problem $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ and an arc length function $J : \Sigma \to \mathbb{R}_{\geq 0}$, find a feasible path σ^* such that $J(\sigma^*) = \min\{J(\sigma) \mid \sigma \text{ is feasible}\}$. If no such path exists, report failure.

A depiction of the optimal path planning problem may be found in Fig. 3.2. With sampling-based planning, our goal will be to return an approximate path σ_n through n state space samples $\mathbf{x}_i \in \mathcal{X}_{\text{free}}$ whose cost comes as close as possible to that of σ^*.

3.2 Fast Marching Trees

In this section, we introduce two algorithms designed to efficiently approximate solutions for the shortest-path motion planning problem, as represented by Definition 5—the Fast Marching Trees

(FMT*) algorithm and its novel bi-directional variant, the Bi-Directional Fast Marching Trees (BFMT*) algorithm. For reference, the pseudocode representations of each are given as Algorithms 1 and 4, respectively. To begin, we start with high-level descriptions of FMT* and BFMT* in Sections 3.2.1 and 3.2.2. We then follow in Sections 3.2.3 and 3.2.4 with additional details, describing the key features of each algorithm and their implementations.

3.2.1 The FMT* Algorithm – High-Level Description

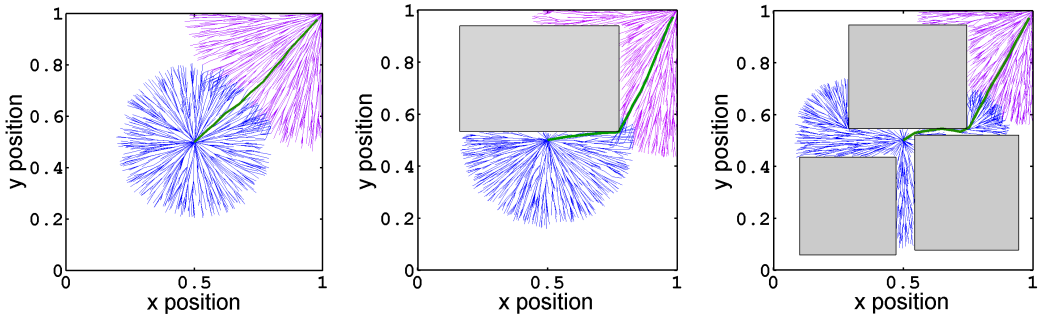
The FMT* algorithm, introduced in [80], is a unidirectional algorithm that essentially performs a forward dynamic programming recursion over a set of sampled points and correspondingly generates a tree of paths that grow steadily outward in cost-to-come space. The recursion performed by FMT* is characterized by three key features:

1. It is tailored to disk-connected graphs, where two samples are considered *neighbors* (hence connectable) if their distance is below a given bound, referred to as the *connection radius*.
2. It performs graph construction and graph search *concurrently*.
3. For the evaluation of the immediate cost in the dynamic programming recursion, one "lazily" ignores the presence of obstacles, and whenever a locally-optimal connection to a new sample (assuming no obstacles) intersects an obstacle, that sample is simply skipped and left for later (as opposed to looking for other locally-optimal connections within the neighborhood).

This last feature, which makes the algorithm "lazy," may result in *suboptimal* connections. Fortunately, a central property of FMT* is that the cases where a suboptimal connection is made become vanishingly rare as the number of samples n goes to infinity, which helps maintain the algorithm's asymptotic optimality. This manifests itself into a key computational advantage—by restricting collision detection to only locally-optimal connections, FMT* (as opposed to, *e.g.*, PRM* [103]) avoids a large number of costly collision-checks, at the price of a vanishingly small "degree" of suboptimality. For proofs and details on the other advantages of FMT*, we refer the interested reader to [80].

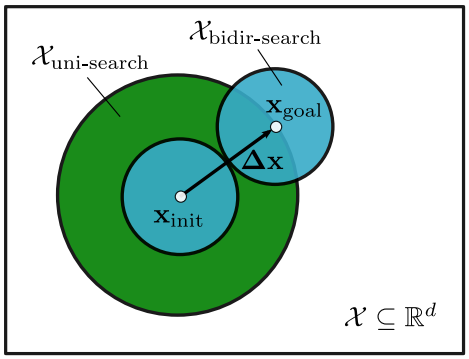
3.2.2 The BFMT* Algorithm – High-Level Description

At its core, BFMT* implements a *bi-directional* version of the FMT* algorithm by simultaneously propagating two wavefronts through the free state space (henceforth, the leaves of an expanding tree will be referred to as the tree *wavefront*). With the possible exception of propagation direction, each wavefront of BFMT* expands identically to that of FMT*. BFMT*, therefore, performs a *two-source* dynamic programming recursion over a set of sampled points, and correspondingly generates a *pair* of search trees: one in cost-to-come space from the initial state (called the *forward tree*) and another in cost-to-go space from a goal state (called the *backward tree*). Illustrations of BFMT* exploration for various path planning problem instances can be seen in Fig. 3.3.



(a) 0% Coverage (b) 25% Coverage (c) 50% Coverage

Figure 3.3: Visualizing BFMT* exploration for varying amounts of obstacle coverage. The BFMT* algorithm generates a *pair* of search trees: one in cost-to-come space from the initial state (blue) and another in cost-to-go space from the goal state (purple). The resulting solution path is given in green.



(a) Uncluttered search volumes for uni-directional and bi-directional search.

d	$\frac{\mu(\mathcal{X}_{\text{bidir-search}})}{\mu(\mathcal{X}_{\text{uni-search}})}$
1	1
2	0.5
3	0.25
$\vdots$	$\vdots$
k	$\left(\frac{1}{2}\right)^{k-1}$

(b) Comparison of uncluttered search volume ratios as a function of state space dimension, d.

Figure 3.4: Illustrating the geometric advantages of bi-directional search for straight-line path planning in relatively uncluttered state spaces.

The dynamic programming recursion performed by BFMT* is characterized by the same lazy feature of FMT* (see Section 3.2.1). However, the time it takes to run BFMT* on a given number of samples can be substantially smaller than for FMT*. Indeed, for uncluttered state spaces, the search trees grow hyperspherically, and BFMT* only has to expand about half as far (in both trees) as FMT* in order to return a solution. This is made clear in Fig. 3.3a, in which FMT* would have to expand the forward tree (shown in blue) twice as far to connect to the goal point in the upper-right corner. Since runtime scales approximately with edge number, which scales as the linear distance covered by the tree raised to the dimension of the state space, we may expect in loosely cluttered state spaces an approximate speed-up of a factor 2^{d-1} over FMT* in d-dimensional space (the -1 in the exponent is because BFMT* has to expand 2 trees, so it loses one factor of 2 advantage). This makes BFMT* an attractive alternative for some problems. See Fig. 3.4 for visualization.

3.2.3 The FMT* Algorithm – Detailed Description

To understand the FMT* algorithm, some background notation must first be introduced. Let $\mathcal{S}$ be a set of points sampled independently and identically from the uniform distribution on $\mathcal{X}_{\text{free}}$, to which $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ are added. (The extension to non-uniform sampling distributions is addressed in Section 3.3.3). Let tree $\mathcal{T}$ be the quadruple $(\mathcal{V}, \mathcal{E}, \mathcal{V}_{\text{unvisited}}, \mathcal{V}_{\text{open}})$, where $\mathcal{V}$ is the set of tree nodes, $\mathcal{E}$ is the set of tree edges, and $\mathcal{V}_{\text{unvisited}}$ and $\mathcal{V}_{\text{open}}$ are mutually exclusive sets containing the *unvisited* samples in $\mathcal{S}$ and the *wavefront* nodes in $\mathcal{V}$, respectively. To be precise, the unvisited set $\mathcal{V}_{\text{unvisited}}$ stores all samples in the sample set $\mathcal{S}$ that have not yet been considered for addition to the tree of paths, while the wavefront set $\mathcal{V}_{\text{open}}$ (also called the "frontier") tracks in sorted order (by cost from the root) only those nodes which have already been added to the tree that are near enough to tree leaves to actually form better connections.

With this notation, the FMT* algorithm can be represented in pseudocode as Algorithm 1. Before describing FMT* in detail, we list briefly the basic planning functions employed by the algorithm:

- SAMPLEFREE(n): returns a set of $n \in \mathbb{N}$ points sampled independently and identically from the uniform distribution on $\mathcal{X}_{\text{free}}$.
- PATH($\mathbf{z}, \mathcal{T}$): returns the unique path in tree $\mathcal{T}$ from its root to node $\mathbf{z}$.
- COST($\overline{\tilde{\mathbf{x}}\mathbf{x}}$): returns the cost of the straight-line path between states $\tilde{\mathbf{x}}$ and $\mathbf{x}$.
- COST($\mathbf{x}$, $\mathcal{T}$): returns the cost of the unique path in tree $\mathcal{T}$ from its root to its node $\mathbf{x}$.
- COLLISIONFREE($\mathbf{x}$, $\mathbf{y}$): returns true if the straight-line path between states $\mathbf{x}$ and $\mathbf{y}$ is collision-free (*i.e.*, feasible), or else returns false otherwise.
- NEAR($\mathcal{S}$, $\mathbf{z}$, r): returns the subset of samples $\mathcal{S}$ within a ball of radius r centered at sample $\mathbf{z}$ (*i.e.*, the set $\{\mathbf{x} \in \mathcal{S} \mid \|\mathbf{x} - \mathbf{z}\| < r\}$).
- TERMINATE(): represents an external termination criterion (*i.e.*, time-out, maximum number of samples, *etc.*) which can be used to force early termination (or prevent infinite run-time for infeasible problems).

We are now in position to describe the FMT* algorithm. We begin on line 1 with forming a sample set $\mathcal{S}$ of free-space states by drawing n samples uniformly from $\mathcal{X}_{\text{free}}$. A tree $\mathcal{T}$ is then initialized on line 2 with $\mathbf{x}_{\text{init}}$ at its root using the INITIALIZE subfunction at the bottom of Algorithm 1. Once complete, tree expansion begins on line 5 using EXPAND as shown in Algorithm 2, starting from node $\mathbf{z}$ initially set to $\mathbf{x}_{\text{init}}$. Note that, in what follows, $\mathbf{z}$ will consistently represent the node selected for expansion. The EXPAND procedure requires the specification of a *connection radius* parameter, r_n, whose selection will be discussed in Section 3.3. EXPAND implements the "lazy" dynamic programming recursion described (at a high level) in Section 3.2.1, making locally-optimal

Algorithm 1 The Fast Marching Tree (FMT*) Algorithm

Require: Query $(\mathbf{x}_{\text{init}}, \mathcal{X}_{\text{goal}})$, Search radius r_n, Sample count n
1: $\mathcal{S} \leftarrow \{\mathbf{x}_{\text{init}}\} \cup \text{SAMPLEFREE}(n)$
2: $\mathcal{T} \leftarrow \text{INITIALIZE}(\mathcal{S}, \mathbf{x}_{\text{init}})$
3: $\mathbf{z} \leftarrow \mathbf{x}_{\text{init}}$, $\sigma^* \leftarrow \varnothing$
4: **while** $\sigma^* = \varnothing$
5: $\quad \mathcal{T} \leftarrow \text{EXPAND}(\mathcal{T}, \mathbf{z}, r_n)$
6: $\quad$ **if** TERMINATE() **then return** Failure
7: $\quad$ **else if** $\mathcal{V}_{\text{open}} = \varnothing$ **then** $\mathcal{T} \leftarrow \text{INSERT}(\mathcal{T}, r_n)$
8: $\quad\quad \mathbf{z} \leftarrow \arg\min_{\tilde{\mathbf{x}} \in \mathcal{V}_{\text{open}}} \{\text{COST}(\tilde{\mathbf{x}}, \mathcal{T})\}$
9: $\quad$ **if** $\mathbf{z} \in \mathcal{X}_{\text{goal}}$
10: $\quad\quad \sigma^* \leftarrow \text{PATH}(\mathbf{z}, \mathcal{T})$
11: **return** σ^*

1: **function** INITIALIZE($\mathcal{S}$, $\mathbf{x}_0$)
2: $\quad \mathcal{V} \leftarrow \varnothing$, $\mathcal{E} \leftarrow \varnothing$, $\mathcal{V}_{\text{unvisited}} \leftarrow \mathcal{S}$, $\mathcal{V}_{\text{open}} \leftarrow \varnothing$
3: $\quad$ **return** $\mathcal{T} \leftarrow \text{ADDNODE}((\mathcal{V}, \mathcal{E}, \mathcal{V}_{\text{unvisited}}, \mathcal{V}_{\text{open}}), \mathbf{x}_0)$

1: **function** ADDNODE($\mathcal{T}, \mathbf{x}$)
2: $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{x}\}$ ▷ Add $\mathbf{x}$ to tree
3: $\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{(\mathbf{x}_{\min}, \mathbf{x})\}$ ▷ Add edge to tree
4: $\quad \mathcal{V}_{\text{unvisited}} \leftarrow \mathcal{V}_{\text{unvisited}} \setminus \{\mathbf{x}\}$ ▷ Mark $\mathbf{x}$ as visited
5: $\quad \mathcal{V}_{\text{open}} \leftarrow \mathcal{V}_{\text{open}} \cup \{\mathbf{x}\}$ ▷ Add $\mathbf{x}$ to wavefront/frontier
6: $\quad$ **return** $\mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E}, \mathcal{V}_{\text{unvisited}}, \mathcal{V}_{\text{open}})$

collision-free connections *from* unvisited samples $\mathbf{x}$ near $\mathbf{z}$ (those in set $\mathcal{V}_{\text{unvisited}}$ within search radius r_n of $\mathbf{z}$) *to* wavefront/frontier nodes $\tilde{\mathbf{x}}$ near each $\mathbf{x}$ (those in set $\mathcal{V}_{\text{open}}$ within search radius r_n of $\mathbf{x}$). See Figs. 3.5a–3.5b for visualization. Only the best locally-optimal connection is considered per unvisited sample $\mathbf{x}$; if this connection is infeasible, $\mathbf{x}$ is dropped from consideration during the current iteration even if there are other frontier nodes nearby it could possibly connect to. This "lazy" expansion in the execution of its dynamic programming recursion is the key feature of FMT* that allows such fast free state space exploration—as discussed in [80] this comes at no loss of (asymptotic) optimality (and likewise for BFMT*, as we will see in Section 3.3). The reason this does not present a significant cost detriment is because these ignored $\mathbf{x}$ sample states will often be connected to the tree frontier via alternative locally-optimal connections during future iterations of the algorithm. Once any feasible collision-free connections have been identified, they and their corresponding sample state endpoints are then added to $\mathcal{T}$ as new edges and frontier nodes, respectively, and $\mathbf{z}$ is dropped from the list of frontier nodes.

After expansion, the algorithm calls TERMINATE to check for early termination before proceeding further. If the algorithm has not terminated, it checks whether the frontier $\mathcal{V}_{\text{open}}$ of tree $\mathcal{T}$ is empty (line 7). If this is the case, the INSERT function shown in Algorithm 3 samples a new state $\mathbf{s}$ uniformly from $\mathcal{X}_{\text{free}}$ and tries to connect it to a nearest neighbor in the companion tree within radius r_n. This way, the expanding tree is ensured to have at least one state in its frontier available for expansion on subsequent iterations (the alternative would be to report failure). This mimics

Algorithm 2 Fast Marching Tree Expansion Step

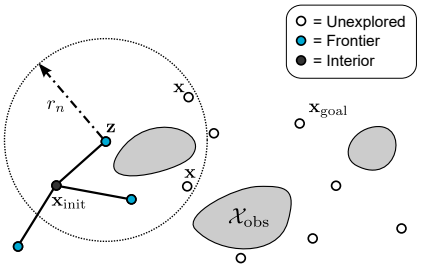
```
1: function EXPAND(𝒯, z, r_n)
2:     V_open,new ← ∅
3:     Z_near ← NEAR(V_unvisited, z, r_n)
4:     for x ∈ Z_near
5:         X_near ← NEAR(V_open, x, r_n)
6:         x_min ← arg min_{x̃ ∈ X_near} {COST(x̃, 𝒯) + COST(x̃x)}
7:         if COLLISIONFREE(x_min, x)
8:             (𝒱, ℰ, V_unvisited, V_open,new) ← ADDNODE((𝒱, ℰ, V_unvisited, V_open,new), x)
9:     V_open ← (V_open ∪ V_open,new) \ {z}          ▷ Add new nodes and drop z from the wavefront
10:    return 𝒯 ← (𝒱, ℰ, V_unvisited, V_open)
```

Algorithm 3 Insertion of New Samples

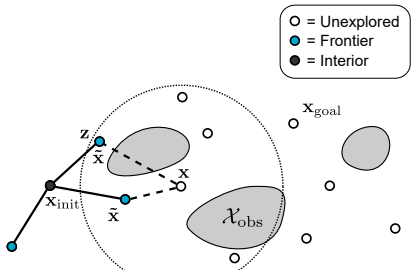
```
1: function INSERT(𝒯, r_n)
2:     while V_open = ∅ and not TERMINATE()
3:         s ← SAMPLEFREE(1)
4:         V_near ← NEAR(𝒱, s, r_n)
5:         while V_near ≠ ∅
6:             x_min ← arg min_{x ∈ V_near} {COST(x, 𝒯) + COST(xs)}
7:             if COLLISIONFREE(x_min, s)
8:                 𝒯 ← ADDNODE(𝒯, s)
9:                 break
10:            else then V_near ← V_near \ {x_min}
11:    return 𝒯 ← (𝒱, ℰ, V_unvisited, V_open)
```

anytime behavior, and by forcing samples to lie close to tree nodes we effectively "reopen" closed nodes for expansion again. Uniform re-sampling may require many attempts before finding a state $\mathbf{s}$ that can be successfully connected to $\mathcal{V}_{\text{open}}$, though we found that the impact of this on running time was negligible while running our path planning studies of Chapter 4. Still, a more effective strategy might bias re-sampling towards areas requiring expansion (*e.g.*, bottlenecks, traps) rather than uniformly within tree coverage.

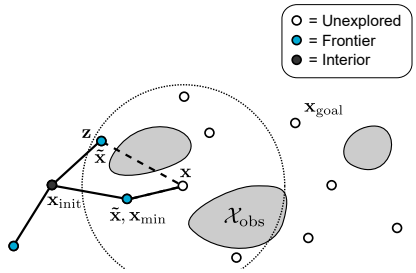
The algorithm then proceeds on line 8 with the selection of the next node for expansion. As INSERT ensures that $\mathcal{T}$ has at least one node in its frontier $\mathcal{V}_{\text{open}}$, a node is always available for subsequent expansion as the next $\mathbf{z}$. After selection, the entire process is iterated until we happen to select a node $\mathbf{z}$ that lies in the goal region $\mathcal{X}_{\text{goal}}$; at this point, we have discovered the first shortest-path connection from the root $\mathbf{x}_{\text{init}}$ that happens to end in $\mathcal{X}_{\text{goal}}$ at $\mathbf{z}$. By Bellman's principle of optimality, the path traced from $\mathbf{z}$ back to the root is therefore the optimal path σ^* to $\mathcal{X}_{\text{goal}}$ through our particular distribution of discrete samples, $\mathcal{S}$. For a graphical illustration, see Fig. 3.5d.



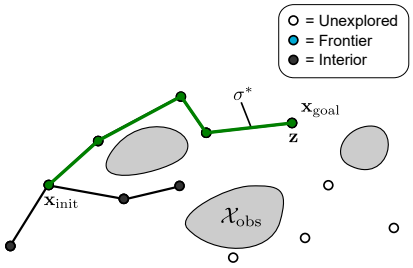
(a) Identifying unvisited samples $\mathbf{x}$ near $\mathbf{z}$



(b) Identifying frontier nodes $\tilde{\mathbf{x}}$ near one of the unvisited samples $\mathbf{x}$



(c) Lazily connecting from $\mathbf{x}_{\text{min}}$ to $\mathbf{x}$ (no edges added if infeasible)



(d) The successful terminating condition that yields solution σ^*

Figure 3.5: Various stages of the FMT* algorithm as it concurrently constructs a graph and explores feasible paths through $\mathcal{X}_{\text{free}}$

3.2.4 The BFMT* Algorithm – Detailed Description

The BFMT* algorithm, represented in Algorithm 4, iterates similarly to FMT*, using unexplored and frontier sets in exactly the same manner as FMT*. However, in this case BFMT* "grows" two trees, referred to as $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{V}_{\text{unvisited}}, \mathcal{V}_{\text{open}})$ and $\mathcal{T}' = (\mathcal{V}', \mathcal{E}', \mathcal{V}'_{\text{unvisited}}, \mathcal{V}'_{\text{open}})$, from two different locations: one from the initial state $\mathbf{x}_{\text{init}}$ and another from a goal state $\mathbf{x}_{\text{goal}}$. Initially, $\mathcal{T}$ is the tree rooted at $\mathbf{x}_{\text{init}}$, while $\mathcal{T}'$ is the tree rooted at $\mathbf{x}_{\text{goal}}$ (note, however, that the trees are exchanged during the execution of BFMT*, so $\mathcal{T}$ in Algorithm 4 is not always the tree that contains $\mathbf{x}_{\text{init}}$).

To accommodate these variations, we introduce two new planning primitives to complement the list introduced in Section 3.2.3:

- SWAP($\mathcal{T}$, $\mathcal{T}'$): swaps the two trees $\mathcal{T}$ and $\mathcal{T}'$.
- COMPANION($\mathcal{T}$): returns the companion tree $\mathcal{T}'$ to $\mathcal{T}$ (or vice versa).

We now describe the BFMT* algorithm in detail. As for FMT*, we first draw a set of samples $\mathcal{S}$ comprising n states in $\mathcal{X}_{\text{free}}$. Instead of one tree, however, we then initialize two trees using the INITIALIZE subfunction at the bottom of Algorithm 1, with a forward tree rooted at $\mathbf{x}_{\text{init}}$ and a reverse tree rooted at $\mathbf{x}_{\text{goal}}$. Once complete, state space exploration begins with the forward tree $\mathcal{T}$, setting $\mathbf{x}_{\text{init}}$ as the next node for expansion $\mathbf{z}$. Here a slightly-modified form of the EXPAND procedure,

Algorithm 4 The Bi-directional Fast Marching Tree (BFMT*) Algorithm

Require: Query $(\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$, Search radius r_n, Sample count n
1: $\mathcal{S} \leftarrow \{\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}}\} \cup \textsc{SampleFree}(n)$
2: $\mathcal{T} \leftarrow \textsc{Initialize}(\mathcal{S}, \mathbf{x}_{\text{init}})$
3: $\mathcal{T}' \leftarrow \textsc{Initialize}(\mathcal{S}, \mathbf{x}_{\text{goal}})$
4: $\mathbf{z} \leftarrow \mathbf{x}_{\text{init}}$, $\mathbf{x}_{\text{meet}} \leftarrow \varnothing$, $\sigma \leftarrow \varnothing$
5: **while** $\sigma = \varnothing$
6: $\quad \{\mathbf{x}_{\text{meet}}, \mathcal{T}\} \leftarrow \textsc{Expand}(\mathcal{T}, \mathbf{z}, r_n, \mathbf{x}_{\text{meet}})$
7: $\quad$ **if** $\mathbf{x}_{\text{meet}} \neq \varnothing$
8: $\quad\quad \sigma \leftarrow \textsc{Path}(\mathbf{x}_{\text{meet}}, \mathcal{T}) \cup \textsc{Path}(\mathbf{x}_{\text{meet}}, \mathcal{T}')$
9: $\quad\quad$ **break**
10: $\quad$ **else if** $\textsc{Terminate}()$ **then return** Failure
11: $\quad$ **else if** $\mathcal{V}'_{\text{open}} = \varnothing$ **then** $\mathcal{T}' \leftarrow \textsc{Insert}(\mathcal{T}', r_n)$
12: $\quad \mathbf{z} \leftarrow \underset{\tilde{\mathbf{x}} \in \mathcal{V}'_{\text{open}}}{\arg\min} \{\textsc{Cost}(\tilde{\mathbf{x}}, \mathcal{T}')\}$
13: $\quad \textsc{Swap}(\mathcal{T}, \mathcal{T}')$
14: **return** σ

Algorithm 5 Bi-directional Fast Marching Tree Expansion Step (Identical to Algorithm 2, except for modified function input and output signatures and the new lines 9–10)

1: **function** $\textsc{Expand}(\mathcal{T}, \mathbf{z}, r_n, \mathbf{x}_{\text{meet}})$
2: $\quad \mathcal{V}_{\text{open,new}} \leftarrow \varnothing$
3: $\quad Z_{\text{near}} \leftarrow \textsc{Near}(\mathcal{V}_{\text{unvisited}}, \mathbf{z}, r_n)$
4: $\quad$ **for** $\mathbf{x} \in Z_{\text{near}}$
5: $\quad\quad X_{\text{near}} \leftarrow \textsc{Near}(\mathcal{V}_{\text{open}}, \mathbf{x}, r_n)$
6: $\quad\quad \mathbf{x}_{\min} \leftarrow \underset{\tilde{\mathbf{x}} \in X_{\text{near}}}{\arg\min} \{\textsc{Cost}(\tilde{\mathbf{x}}, \mathcal{T}) + \textsc{Cost}(\overline{\tilde{\mathbf{x}}\mathbf{x}})\}$
7: $\quad\quad$ **if** $\textsc{CollisionFree}(\mathbf{x}_{\min}, \mathbf{x})$
8: $\quad\quad\quad (\mathcal{V}, \mathcal{E}, \mathcal{V}_{\text{unvisited}}, \mathcal{V}_{\text{open,new}}) \leftarrow \textsc{AddNode}((\mathcal{V}, \mathcal{E}, \mathcal{V}_{\text{unvisited}}, \mathcal{V}_{\text{open,new}}), \mathbf{x})$
9: $\quad\quad\quad$ **if** $\{\mathbf{x} \in \mathcal{V}'$ and $\textsc{Cost}(\mathbf{x}, \mathcal{T}) + \textsc{Cost}(\mathbf{x}, \mathcal{T}') < \textsc{Cost}(\mathbf{x}_{\text{meet}}, \mathcal{T}) + \textsc{Cost}(\mathbf{x}_{\text{meet}}, \mathcal{T}')\}$
10: $\quad\quad\quad\quad \mathbf{x}_{\text{meet}} \leftarrow \mathbf{x}$ ▷ Save $\mathbf{x}$ as best connection
11: $\quad \mathcal{V}_{\text{open}} \leftarrow (\mathcal{V}_{\text{open}} \cup \mathcal{V}_{\text{open,new}}) \backslash \{\mathbf{z}\}$ ▷ Add new nodes and drop $\mathbf{z}$ from the wavefront
12: $\quad$ **return** $\{\mathbf{x}_{\text{meet}}, \mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E}, \mathcal{V}_{\text{unvisited}}, \mathcal{V}_{\text{open}})\}$

presented in Algorithm 5, is used to conduct the same lazy dynamic programming recursion as in FMT*. As before, Expand adds any collision-free edges and newly-connected nodes to $\mathcal{T}$ and drops $\mathbf{z}$ from its list of frontier nodes $\mathcal{V}_{\text{open}}$. Now, however, we must add some bookkeeping through an additional variable, $\mathbf{x}_{\text{meet}}$, to keep track of the lowest total-cost connection candidate that joins our two search trees together.

After expansion, the algorithm checks whether a feasible path is found on line 7 by checking whether $\mathbf{x}_{\text{meet}}$ has been assigned. If unsuccessful so far, Terminate (which reports failure upon early termination) is checked before proceeding. If the algorithm has not terminated, BFMT* then prepares for a new iteration by calling Insert (shown in Algorithm 3) on its companion tree $\mathcal{T}'$ in the event that its corresponding frontier $\mathcal{V}'_{\text{open}}$ is empty (line 11). The algorithm then proceeds on lines 12–13 with the selection of the next node (and corresponding tree) for expansion. As shown,

BFMT* "swaps" the forward and backward trees on each iteration, so that each may be expanded in turn. As INSERT ensures the companion tree $\mathcal{T}'$ always has at least one node in its frontier $\mathcal{V}'_{\text{open}}$, a node is always available for subsequent expansion as the next $\mathbf{z}$. After selection, the entire process is iterated. This alternating graph search and exploration process repeats until the first connection point $\mathbf{x}_{\text{meet}}$ is established between the two trees, in which case the corresponding paths in $\mathcal{T}$ and $\mathcal{T}'$ are traced and joined together to yield solution σ.

Algorithm Variations

As for any bi-directional planner, the correctness and computational efficiency of BFMT* hinge upon two key aspects: (i) how computation is interleaved among the two trees (in other words, which wavefront/frontier at each step should be chosen for expansion), and (ii) when the algorithm should terminate. For instance, as an alternative tree expansion strategy (*i.e.*, item (i)), one could replace lines 12–13 with the "balanced trees" condition which enforces more of a balanced search, maintaining equal costs from the root within each wavefront such that the two wavefronts propagate and meet roughly equidistantly in cost-to-go from their roots:

12: $\mathbf{z}_1 \leftarrow \underset{\mathbf{x} \in \mathcal{V}_{\text{open}}}{\arg\min} \{\textsc{Cost}(\mathbf{x}, \mathcal{T})\}$
13: $\mathbf{z}_2 \leftarrow \underset{\tilde{\mathbf{x}} \in \mathcal{V}'_{\text{open}}}{\arg\min} \{\textsc{Cost}(\tilde{\mathbf{x}}, \mathcal{T}')\}$
14: $(\mathbf{z}, \mathcal{T}) \leftarrow \underset{(\mathbf{z}_1, \mathcal{T}), (\mathbf{z}_2, \mathcal{T}')}{\arg\min} \{\textsc{Cost}(\mathbf{z}_i, \mathcal{T}_i)\}$
15: $\mathcal{T}' = \textsc{Companion}(\mathcal{T})$

Similarly, as an alternative to the "first path" termination criterion (*i.e.*, item (ii)), one might replace line 7 with the "best path" criterion:

7: $\mathbf{z} \in \left(\mathcal{V}' \setminus \mathcal{V}'_{\text{open}}\right)$

Currently, line 7 returns the first available path discovered, at the moment that the two wavefronts touch at $\mathbf{x}_{\text{meet}}$ (which is not, in general, the lowest cost path). This alternative "best path" condition, on the other hand, returns the *exact optimal path* from $\mathbf{x}_{\text{init}}$ to $\mathbf{x}_{\text{goal}}$ through the given set $\mathcal{S}$ of n samples. This change terminates BFMT* when the two wavefronts have propagated sufficiently far through each other such that no better solution can be discovered. Intuitively-speaking, this occurs at the first moment where the two trees have both selected, at the current iteration or previously, the same sample as the minimum cost node $\mathbf{z}$ from their respective roots.

Though seemingly promising ideas, no appreciable differences in performance were found using the above criteria in combination or otherwise; hence we report only the simplest version of our planner as Algorithm 4.

Anytime Behavior

Because FMT* and BFMT* continually re-sample new points upon failure using the INSERT routine (Algorithm 3), each algorithm achieves a pseudo-anytime behavior (analogously to MPLB of [120]). Here they are "anytime" in the sense that samples are repeatedly added to the state space and connected to the tree until time runs out, with previous computations carefully reused; they are unlike "anytime" algorithms, on the other hand, in the sense that the first solution found is returned immediately (and not continually improved). However, due to the property that suboptimal connections vanish almost surely as the initial number of samples $n \to \infty$, this should not be a significant drawback for practical applications.

Note, by replacing INSERT on line 11 with "**return** Failure", we recover a more straightforward, non-anytime bi-directional version of the original FMT* algorithm. It is advantageous, however, to keep the INSERT routine, as it dramatically increases the success rate of the algorithm for difficult motion planning problems.

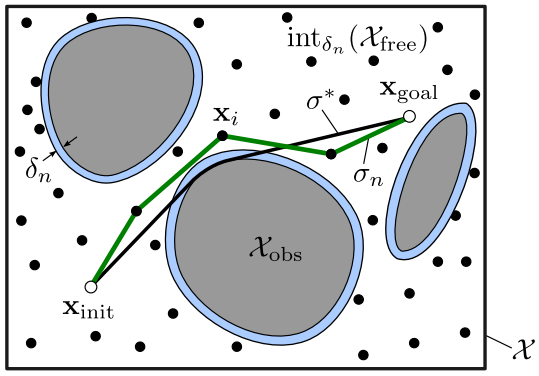
3.3 Theoretical Characterization of BFMT*

In this section, we prove the asymptotic optimality of BFMT*, and provide an upper bound on its convergence rate. We start from a result called *probabilistic exhaustivity*, which essentially states that any path in $\mathcal{X}_{\text{free}}$ may be "traced" arbitrarily well by connecting points from a sufficiently-large pseudo-random sample set covering $\mathcal{X}_{\text{free}}$. We then prove BFMT*'s asymptotic optimality by showing that it returns solutions with costs no greater than that of any tracing path. The claim is proven assuming BFMT* acts without the INSERT procedure (Algorithm 3), in place of which "Failure" is reported instead. The proof for the full algorithm then follows immediately by *a fortiori* argument.

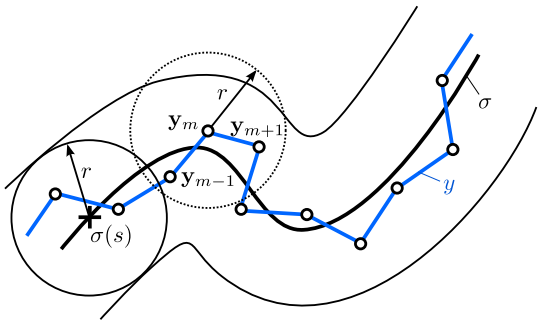
3.3.1 Mathematical Preliminaries

Before proceeding with our analysis, we first need to introduce a number of concepts that will appear recurrently throughout the proof.

Path Clearance Here we introduce some definitions concerning the *clearance* of a path, *i.e.*, its "distance" from $\mathcal{X}_{\text{obs}}$ [80]. For a given $\delta > 0$, the δ-interior of $\mathcal{X}_{\text{free}}$ is defined as the set of all points that are at least a distance δ away from any point in $\mathcal{X}_{\text{obs}}$. A collision-free path σ is said to have *strong δ-clearance* if it lies entirely inside the δ-interior of $\mathcal{X}_{\text{free}}$. This means σ is always some minimum distance $\delta > 0$ from being infeasible. As a less-stringent condition, a collision-free path σ is said to have *weak δ-clearance* if there exists a homotopy $\psi : [0, 1] \to \Sigma$, with $\psi(0) = \sigma$ and $\psi(1) = \sigma'$ for some strong δ-clear path σ', such that $\psi(\alpha)$ has strong δ_α-clearance for some $\delta_\alpha > 0$, for all $\alpha \in (0, 1]$. In other words, weak δ-clear paths σ must have a strong δ-clear paths σ' nearby (in the same "homotopy class") that can be continuously deformed into σ. All strong δ-clear paths have



(a) A robustly-feasible path planning problem, with: (i) a path σ_n with strong-δ_n clearance, and (ii) an optimal path σ^* with weak-δ_n clearance.



(b) An (ϵ, r)-tracing of a path σ. Probabilistic exhaustivity guarantees that we can always find a closely-tracing path y among samples $\mathcal{S}$ as $n \to \infty$, almost surely.

Figure 3.6: Illustrations of various path planning concepts

weak δ-clearance; however, weak δ-clearance also encompasses paths that touch obstacle boundaries but do not pass between two touching obstacles. See Fig. 3.6a for visualization.

Robust Feasibility A path planning problem with optimal solution σ^* and associated cost J^* is called *δ-robustly feasible* if: (i) σ^* has weak δ-clearance and (ii) there exists a sequence of feasible paths $\{\sigma_n\}_{n=1}^{\infty}$, with $\sigma_n(0) = \mathbf{x}_{\text{init}}$, $\sigma_n(1) = \mathbf{x}_{\text{goal}}$, and $\sigma_n(s) \neq \mathbf{x}_{\text{goal}}$ for all $s \in (0, 1)$, such that $\lim_{n\to\infty} J(\sigma_n) = J^*$ and such that σ_n has δ_n-clearance for some strictly positive sequence $\delta_n \to 0$, with $\delta_n \leq \delta$ for all $n \in \mathbb{N}$. In other words, δ-robustly-feasible problems admit arbitrarily-close δ_n-clear path approximations to optimal path solutions, where $\delta_n \leq \delta$ can be reduced to zero.

Probabilistic Exhaustivity Let $\sigma : [0, 1] \to \mathcal{X}$ be a path. Given a set of samples (referred to as waypoints) $\{\mathbf{y}_m\}_{m=1}^{M} \subset \mathcal{X}$, we associate a path $y : [0, 1] \to \mathcal{X}$ that sequentially connects the nodes $\mathbf{y}_1, \ldots, \mathbf{y}_M$ with line segments. We consider the waypoints $\{\mathbf{y}_m\}$ to (ϵ, r)-*trace* the path σ if: (i) $\|\mathbf{y}_m - \mathbf{y}_{m+1}\| \leq r$ for all m, (ii) the cost of y is bounded as $J(y) \leq (1+\epsilon)J(\sigma)$, and (iii) the distance from any point of y to σ is no more than r, *i.e.*, $\min_{s\in[0,1]} \|y(t) - \sigma(s)\| \leq r$ for all $t \in [0, 1]$. In the context of sampling-based motion planning, we may expect to find closely-tracing $\{\mathbf{y}_m\}$ as a subset of the sample set $\mathcal{S}$, provided the sample size is large and assuming a dense sampling sequence. This notion is formalized below (Theorem 6), proven as Theorem IV.5 in [81] for the general case of driftless control-affine control systems, a special case of which is path planning without differential constraints (as addressed in this chapter).

Theorem 6 (Probabilistic Exhaustivity)**.** *Define path planning problem* $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ *and let* $\sigma : [0, 1] \to \mathcal{X}_{\text{free}}$ *be a feasible path. Denote the volume of the d-dimensional Euclidean unit ball by* ζ_d.

Finally, let $\mathcal{S} = \{\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}}\} \cup \textsc{SampleFree}(n)$, $\epsilon > 0$, *and for fixed* n *consider the event* $\mathcal{A}_n$ *that there exist waypoints* $\{\mathbf{y}_m\}_{m=1}^{M} \subset \mathcal{S}$, *with* $y_1 = \mathbf{x}_{\text{init}}$ *and* $y_M = \mathbf{x}_{\text{goal}}$, *which* (ϵ, r_n)*-trace* σ, *where:*

$$r_n = 4\,(1+\eta)^{\frac{1}{d}} \left(\frac{1}{d}\right)^{\frac{1}{d}} \left(\frac{\mu(\mathcal{X}_{\text{free}})}{\zeta_d}\right)^{\frac{1}{d}} \left(\frac{\log n}{n}\right)^{\frac{1}{d}} \tag{3.1}$$

for a steering parameter $\eta \geq 0$. *Then, as* $n \to \infty$, *the probability that* $\mathcal{A}_n$ *does not occur (denoted by its complement* $\mathcal{A}_n^c$*) is asymptotically bounded as* $\mathrm{P}[\mathcal{A}_n^c] = \mathrm{O}\Big(n^{-\frac{\eta}{d}} \log^{-\frac{1}{d}} n\Big)$.

3.3.2 Asymptotic Optimality Proof

We are now in a position to prove the asymptotic optimality of BFMT*, which represents the main result of this section. In the interest of clarity, we provide a brief sketch of the proof first before delving into mathematical formalisms. Assume a δ-robustly feasible path planning problem with optimal path σ^* of cost J^* (the δ-robustly feasible property ensures that we may approach J^* arbitrarily well by approximant paths with strong δ-clearance). Consider running BFMT* to completion to solve this problem, using a set of n samples to generate one cost-to-come tree $\mathcal{T}_i$ and one cost-to-go tree $\mathcal{T}_g$ rooted at $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$, respectively. If we can show that BFMT* returns a path whose cost approaches the cost of any one of these δ-feasible approximant paths σ arbitrary well (with high probability in the limit as $n \to \infty$), then asymptotic convergence to J^* follows immediately. The proof of this "path-tracing" property begins by considering a regime of n sufficiently large so that the connection radius r_n is "small" with respect to both the obstacle clearance δ and the approximant path cost $J(\sigma)$. For large sample counts, the existence of waypoints $\{\mathbf{y}_m\}$ within the sample set which trace σ (when connected together to form path y) occurs with increasingly high probability as n increases (by our assumption of a dense sampling sequence). By choosing r_n small with respect to δ, we ensure that these waypoints will represent a feasible sequence of connections for BFMT*; moreover, it may be shown that BFMT*, like FMT*, will recover a path with cost no worse than $J(y)$—up to an additional additive factor of r_n which arises when the two trees $\mathcal{T}_i$ and $\mathcal{T}_g$ meet, motivating the second choice of r_n small with respect to $J(\sigma)$. As $J(y)$ is, by definition, close to $J(\sigma)$ (which itself is close to J^*), it follows that BFMT* returns a solution that is close to J^* with high probability as $n \to \infty$. This completes the proof.

The formal presentation of these ideas follow in the remainder of this section. We start with an important lemma, which relates the cost of the path returned by BFMT* to that of *any* feasible path. Note the proof addresses the case of running the BFMT* algorithm in the form presented as Algorithm 4 (that is, with the "first path" termination criterion and the "alternating trees" exploration strategy). It should be emphasized, however, that the results apply to *any* expansion criterion (that is, the two mentioned in this chapter and otherwise), as well as the "best path" termination criterion (see Section 3.2.4). The latter follows by *a fortiori* argument; the "best path" termination criterion returns a path at least as good as the "first path" termination criterion because

it returns the best path among a number of choices, one of which is the path produced by the "first path" criterion. See Remark 10 following this proof for further discussion.

Lemma 7 (Bi-directional FMT* Cost Comparison)**.** *Let $\sigma : [0, 1] \to \mathcal{X}_{\text{free}}$ be a feasible path with strong δ-clearance. Consider running BFMT* to completion with n samples and a connection radius r_n given by Eq.* (3.1) *with $\eta \geq 0$. Let J_n denote the cost of the path returned by BFMT*. Then for fixed $\epsilon > 0$:*

$$\mathrm{P}[J_n > (1+\epsilon)J(\sigma)] = \mathrm{O}\Big(n^{-\frac{\eta}{d}} \log^{-\frac{1}{d}} n\Big).$$

Proof. First of all, if $\mathbf{x}_{\text{init}} = \mathbf{x}_{\text{goal}}$, then BFMT* immediately terminates with $J_n = 0$, trivially satisfying the claim. Thus we assume that $\mathbf{x}_{\text{init}} \neq \mathbf{x}_{\text{goal}}$. Consider n sufficiently large so that $r_n \leq \min\big\{\, \delta/2\,, \epsilon \left\|\mathbf{x}_{\text{init}} - \mathbf{x}_{\text{goal}}\right\|/2 \,\big\}$. Apply Theorem 6 to produce, with probability at least $1 - \mathrm{O}\Big(n^{-\frac{\eta}{d}} \log^{-\frac{1}{d}} n\Big)$, a sequence of waypoints $\{\mathbf{y}_m\}_{m=1}^{M} \subset \mathcal{S}$, with $\mathbf{y}_1 = \mathbf{x}_{\text{init}}$ and $\mathbf{y}_M = \mathbf{x}_{\text{goal}}$, which $(\epsilon/2\,, r_n)$-trace σ. We claim that in the event that if such $\{\mathbf{y}_m\}$ exists, the BFMT* algorithm returns a path with cost upper bounded as $J_n \leq J(y) + r_n \leq (1 + \epsilon/2)J(\sigma) + (\epsilon/2)J(\sigma) = (1+\epsilon)J(\sigma)$. The desired result follows directly.

To prove the claim, assume the existence of an $(\epsilon/2\,, r_n)$-tracing $\{\mathbf{y}_m\}$ of σ. Let $\mathcal{B}(\mathbf{x}, r)$ represent a ball of radius r centered at a sample $\mathbf{x}$. Note that our upper bound on r_n implies that $\mathcal{B}(\mathbf{y}_m, r_n)$ intersects no obstacles. This follows from our choice of r_n and the distance bound

$$\begin{aligned}\inf_{\mathbf{s} \in \mathcal{X}_{\text{obs}}} \left\|\mathbf{y}_m - \mathbf{s}\right\| &\geq \inf_{\mathbf{s} \in \mathcal{X}_{\text{obs}}} \left\|\boldsymbol{\sigma}_m - \mathbf{s}\right\| - \left\|\mathbf{y}_m - \boldsymbol{\sigma}_m\right\| \\ &\geq 2r_n - r_n \geq r_n.\end{aligned}$$

where $\boldsymbol{\sigma}_m$ is the closest point of σ to $\mathbf{y}_m$. This fact, along with $\left\|\mathbf{y}_m - \mathbf{y}_{m+1}\right\| \leq r_n$ for all m, implies that when a connection is attempted for $\mathbf{y}_m$, both $\mathbf{y}_{m-1}$ and $\mathbf{y}_{m+1}$ will be in the search radius and no obstacles will lie within that search radius. Running BFMT* to completion generates one cost-to-come tree $\mathcal{T}_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{V}_{\text{open},i}, \mathcal{V}_{\text{unvisited},i})$ and one cost-to-go tree $\mathcal{T}_g = (\mathcal{V}_g, \mathcal{E}_g, \mathcal{V}_{\text{open},g}, \mathcal{V}_{\text{unvisited},g})$ rooted at $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$, respectively (the subscripts i and g are used to identify the root of a tree without ambiguity). The above discussion ensures that the trees will meet and the algorithm will return a feasible path when it terminates—the path outlined by the waypoints $\{\mathbf{y}_m\}$ disallows the possibility of failure.

For each sample point $\mathbf{x} \in \mathcal{S}$, let $J_i(\mathbf{x}) := \textsc{Cost}(\mathbf{x}, \mathcal{T}_i)$ denote the cost-to-come of $\mathbf{x}$ from $\mathbf{x}_{\text{init}}$ in $\mathcal{T}_i$, and let $J_g(\mathbf{x}) := \textsc{Cost}(\mathbf{x}, \mathcal{T}_g)$ denote the cost-to-go from $\mathbf{x}$ to $\mathbf{x}_{\text{goal}}$ in $\mathcal{T}_g$. If $\mathbf{x}$ is not contained in a tree $\mathcal{T}_k$, where $k = \{i, g\}$, we set $J_k(\mathbf{x}) = \infty$. When the algorithm terminates, we know there exists a sample point $\mathbf{x}_{\text{meet}} \in \mathcal{V}_i \cap \mathcal{V}_g$ where the two trees meet; indeed we select the particular meeting point $\mathbf{x}_{\text{meet}} = \arg\min_{\mathbf{x} \in \mathcal{V}_i \cap \mathcal{V}_g} J_i(\mathbf{x}) + J_g(\mathbf{x})$. Then $J_n = J_i(\mathbf{x}_{\text{meet}}) + J_g(\mathbf{x}_{\text{meet}})$. We now note a lemma bounding the costs-to-come of the waypoints $\{\mathbf{y}_m\}$, the proof of which may be found as an inductive hypothesis (Eq. 5) in Theorem VI.1 of [81].

Lemma 8. *Let $m \in \{1, \ldots, M\}$. If $J_i(\mathbf{y}_m) < \infty$, then $J_i(\mathbf{y}_m) \leq \sum_{k=1}^{m-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\|$. Otherwise if $\mathbf{y}_m \notin \mathcal{V}_i$, then $J_i(\mathbf{x}_{\text{meet}}) \leq \sum_{k=1}^{m-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\|$. Similarly if $J_g(\mathbf{y}_m) < \infty$, then $J_g(\mathbf{y}_m) \leq \sum_{k=m}^{M-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\|$; otherwise $J_g(\mathbf{x}_{\text{meet}}) \leq \sum_{k=m}^{M-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\|$.*

To bound the performance J_n of BFMT*, there are two cases to consider. Note in either case we find that $J_n \leq J(\mathbf{y}) + r_n$, thus completing the proof.

Case 1: There exists some $\mathbf{y}_m \in \mathcal{V}_i \cap \mathcal{V}_g$.
In this case, $J_n = J_i(\mathbf{x}_{\text{meet}}) + J_g(\mathbf{x}_{\text{meet}}) \leq J_i(\mathbf{y}_m) + J_g(\mathbf{y}_m) < \infty$ by our choice of $\mathbf{x}_{\text{meet}}$. Then applying Lemma 8 we see that $J_n \leq J_i(\mathbf{y}_m) + J_g(\mathbf{y}_m) \leq \sum_{k=1}^{M-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\| = J(\mathbf{y})$.

Case 2: There are no $\mathbf{y}_m \in \mathcal{V}_i \cap \mathcal{V}_g$.
Consider $\widetilde{m} = \max\{m \mid J_i(\mathbf{y}_m) < \infty\}$. Then $\mathbf{y}_{\widetilde{m}} \in \mathcal{V}_i$ and $\mathbf{y}_{\widetilde{m}}$ cannot have been the minimum-cost element of $\mathcal{V}_{\text{open},i}$ at any point during algorithm execution, or else we would have connected $\mathbf{y}_{\widetilde{m}+1} \in \mathcal{V}_i$. Let $\mathbf{z}$ denote the minimum-cost element of $\mathcal{V}_{\text{open},i}$ when $\mathbf{x}_{\text{meet}}$ was added to $\mathcal{V}_i$. We have the bound:

$$\begin{aligned} J_i(\mathbf{x}_{\text{meet}}) &\leq J_i(\mathbf{z}) + r_n \leq J_i(\mathbf{y}_{\widetilde{m}}) + r_n \\ &\leq \sum_{k=1}^{m-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\| + r_n. \end{aligned} \tag{3.2}$$

By our assumption for this case, $\mathbf{y}_{\widetilde{m}} \notin \mathcal{V}_g$. Then by Lemma 8 we know that $J_g(\mathbf{x}_{\text{meet}}) \leq \sum_{k=m}^{M-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\|$. Combining with the previous inequality yields $J_n = J_i(\mathbf{x}_{\text{meet}}) + J_g(\mathbf{x}_{\text{meet}}) \leq \sum_{k=1}^{M-1} \|\mathbf{y}_k - \mathbf{y}_{k+1}\| + r_n = J(y) + r_n$. □

Remark 9 (Tightened Bound for Connection Radius). As discussed in [81], for the sake of clarity, the constant term 4 in the expression for r_n is greater than is necessary for Theorem 6 to hold. A more careful argument along the lines of the original FMT* AO proof [80] would suffice to show that a factor of 2 satisfies the theorem as well.

Remark 10 (Alternative Termination Criteria). If we analyze the "best path" criterion discussed in Section 3.2.4 instead of the "first path" criterion used in Algorithm 4 (*i.e.*, if BFMT* continues until an element in $\mathcal{V}_i \cap \mathcal{V}_g$ in the interior of the companion tree is selected as the minimum cost element $\mathbf{z}$ of $\mathcal{V}_{\text{open},i}$ or $\mathcal{V}_{\text{open},g}$), then the r_n term may be omitted from inequality of Eq. (3.2). In that case we need only consider n sufficiently large so that $r_n \leq \delta/2$, and the proof holds for the "best path" termination condition as well.

We are now ready to show that BFMT* is asymptotically-optimal. The proof essentially follows as a corollary to Lemma 7. Assuming a δ-robustly feasible path planning problem with optimal path σ^* of cost J^*, there exist feasible approximant paths with strong δ-clearance whose costs approximate J^* arbitrarily well. By Lemma 7, the solution cost returned by BFMT* in turn closely approximates

(with high probability as $n \to \infty$) the cost of any of these approximants; asymptotic convergence to J^* then follows. The next theorem shows this formally.

Theorem 11 (BFMT* Asymptotic Optimality)**.** *Assume a δ-robustly feasible path planning problem as defined in Section 3.3.1 with optimal path σ^* of cost J^*. Then BFMT* converges* in probability *to σ^* as the number of samples $n \to \infty$. Specifically, for any $\epsilon > 0$,*

$$\lim_{n\to\infty} \mathrm{P}[J_n > (1+\epsilon)J^*] = 0$$

Proof. By our δ-robustly feasible assumption, we can find a strong δ-clearance feasible path $\sigma : [0,\, 1] \to \mathcal{X}_{\text{free}}$ that approximates σ^* with cost $J(\sigma) < (1 + \epsilon/3)J^*$ (*i.e.*, less than factor $\epsilon/3$ from J^*), for any $\epsilon > 0$. By Lemma 7, we can choose n sufficiently large such that BFMT* returns an $\epsilon/3$ cost approximation to the approximant:

$$\begin{aligned}\mathrm{P}\Big[J_n > (1+\epsilon/3)^2 J^*\Big] &< \mathrm{P}[J_n > (1+\epsilon/3)J(\sigma)] \\ &= \mathrm{O}\Big(n^{-\frac{\eta}{d}} \log^{-\frac{1}{d}} n\Big)\end{aligned}$$

To approach the optimal path, let the number of samples $n \to \infty$. It follows that, for any $\eta \geq 0$:

$$\lim_{n\to\infty} \mathrm{P}\Big[J_n > (1+\epsilon/3)^2 J^*\Big] < \lim_{n\to\infty} \mathrm{O}\Big(n^{-\frac{\eta}{d}} \log^{-\frac{1}{d}} n\Big) = 0$$

Now we relate this to the original claim. First, suppose that $\epsilon \leq 3$. From $(1 + \epsilon/3)^2 \leq 1 + \epsilon$, the event $\{J_n > (1+\epsilon)J^*\}$ is a subset of the event $\Big\{J_n > (1+\epsilon/3)^2 J^*\Big\}$, hence:

$$\lim_{n\to\infty} \mathrm{P}[J_n > (1+\epsilon)J^*] \leq \lim_{n\to\infty} \mathrm{P}\Big[J_n > (1+\epsilon/3)^2 J^*\Big] = 0.$$

Because the probability is monotone-decreasing in ϵ as ϵ increases, the statement holds for all $\epsilon > 3$ as well (to see this, apply Lemma 7 again for m sufficiently large to handle $\epsilon = 3$; then by similar argument as above $\mathrm{P}[J_m > (1+\epsilon)J^*] < \mathrm{P}[J_m > (1+3)J^*] = \mathrm{O}\Big(m^{-\frac{\eta}{d}} \log^{-\frac{1}{d}} m\Big)$ and take the limit as $m \to \infty$). Hence $\lim_{n\to\infty} \mathrm{P}[J_n > (1+\epsilon)J^*] = 0$ holds for arbitrary ϵ, and we see that BFMT* converges in probability to the optimal path cost, as claimed. □

Remark 12 (Convergence Rate)*.* Note that we can also translate the convergence rate from Lemma 7 to the setup of Theorem 11, which does not require strong δ-clearance. For any $\epsilon > 0$, the optimal path can be approximated by a strong-δ-clear path with cost less than $(1+\epsilon)J(\sigma)$ and we can focus on approximating that path to high-enough precision to still approximate the optimal path to within factor $(1+\epsilon)$. Since the convergence rate in Lemma 7 only contains ϵ in the rate's constant, the big-O convergence rate remains the same. This generalizes the convergence rate result in [80], which only applied to a specific obstacle-free state space, initial state, and goal region.

3.3.3 Sampling and Cost Generalizations

It is worth mentioning that the asymptotic optimality (AO) properties of BFMT* are not limited to uniform sampling and arc-length cost functions. For example, if one has prior information about areas that the optimal path is unlikely to pass through, it may be advantageous to consider a non-uniform sampling strategy that downsamples these regions. As long as the sampling density is lower-bounded by a *positive* number over the state space, BFMT* can be slightly altered (by merely increasing r_n by a constant factor) to ensure it stays AO. The argument is analogous to that made in [80], and essentially proceeds by making the search radius wide enough to balance out the detrimental effect of the lower sampling density (in some areas). An additional common concern is when the cost is not arc-length, but some other metric or line integral cost. In either case, BFMT* need only consider *cost* balls instead of Euclidean balls when making connections. Details on adjusting the algorithm and why the AO proof still holds can be derived from [80]. The argument basically shows that the triangle inequality either holds exactly (for metric costs) or approximately, and that this approximation goes away in the limit as $n \to \infty$.

3.4 Conclusion

In this chapter, we have introduced the Fast Marching Trees (FMT*) and Bi-directional Fast Marching Trees (BFMT*) algorithms for solving shortest-path motion planning problems with arc-length cost functionals. In particular, we presented the first proof of asymptotic optimality for a bi-directional sampling-based planner, the BFMT* algorithm, and showed that it preserves the convergence rate bounds and exploration properties of its unidirectional variant. Several modified versions of BFMT* were analyzed, and generalizations to more complex motion planning problems were discussed. Interesting avenues for future research include the use of adaptive sampling near narrow passages, sample biasing in Insert (Algorithm 3) towards failed wavefronts, and extensions of FMT* and BFMT* to dynamic environments through lazy re-evaluation (leveraging their tree-like path structures) in a way that reuses previous results as much as possible.

In the next chapter, we examine how well these algorithms perform in comparison to their state-of-the-art counterparts, and demonstrate how well they operate in practice onboard a free-flying proximity operations testbed.

Chapter 4

Benchmarking Experiments and Testbed Demonstrations

Though an asymptotically-optimal (AO) sampling-based algorithm may guarantee convergence to an optimal-cost path as the number of samples taken increases (as $n \to \infty$), in practice optimality is not possible to achieve using sampling-based planners due to constraints on memory and other computational resources. This limits any implementation to a finite number of samples, and therefore a suboptimal motion planning solution. Fortunately, for most problems, it matters little that an *exact* optimal solution be found, but rather that a feasible path be found quickly and with sufficiently-low cost (such a solution will henceforth be described as "near-optimal"). Often a trade-off exists, however, between solution path cost and computational speed, as using more samples n tends to produce cheaper paths[1] at the expense of additional processing and overhead. In this chapter, we examine this classical trade-off for the Fast Marching Tree (FMT* and BFMT*) algorithms as applied to a suite of path planning experiments, both simulated and onboard a free-flying robotic testbed. The purposes of these tests are threefold: (i) to illustrate the practical ability of FMT* and BFMT* to solve standardized planning problems, (ii) to benchmark their performance against other state-of-the-art planning algorithms, and (iii) to develop insights into how such algorithms can be used for real-time spacecraft guidance.

4.1 Open Motion Planning Library (OMPL) Experiments

In this section, we provide numerical path-planning experiments that compare the performances of FMT* and BFMT* with other sampling-based, asymptotically-optimal planning algorithms (namely,

[1] This tendency for producing cheaper paths as n increases becomes a *guarantee* for AO algorithms when using dense sampling sequences at sufficiently large sample counts – though when this occurs exactly is highly problem- and sequence-dependent.

RRT* and PRM*)[2]. Because of its introduction in this thesis, special attention is given to the relative capabilities of BFMT*, in particular. To test these algorithms, we aim to observe the quality of the solutions returned by each algorithm as a function of execution time when given the same planning workspace and query. Recall that here simulations neglect dynamic constraints and use arc-length as the solution cost. As a basis for quality comparison between incremental or "anytime" planners (such as RRT*) and non-incremental planners (such as BFMT*, which generate solutions via sample batches), we vary the total number of samples drawn by the planners during the planning process (which in essence serves as a proxy to execution time); note that *sample count* n has a different connotation depending on the planner that will not necessarily be the number of nodes stored in the constructed solution graph—for RRT* (with one sample drawn per iteration), this is the number of iterations, while for FMT*, PRM*, and BFMT*, this is the number of free space samples taken during initialization.

4.1.1 Simulation Setup

To generate simulation data for a given experiment, we query the planning algorithms once each for a series of sample counts, recording the cost of each solution returned, the planner execution time[3], and whether the planner succeeded or not, then repeat this entire process over 50 trials. To ensure a fair comparison, each planning algorithm is tested using the Open Motion Planning Library (OMPL) v1.0.0 [121], which provides high-quality C++ implementations of many state-of-the-art planners and a common framework for executing motion plans. In this way, we ensure that all algorithms employ the *exact same* primitive routines (*e.g.*, nearest-neighbor search, collision-checking, data handling, *etc.*), and thus measure their performances fairly. Regarding implementation specifics, BFMT*, FMT*, and PRM* each use a steering parameter $\eta = 0$ for the nearest-neighbor radius r_n of Eq. (3.1) (from Lemma 7) in order to satisfy the theoretical bounds provided in Theorem 6 and [103]. For RRT*, we use the default OMPL settings; namely, a 5% goal bias and a steering parameter η equal to 20% of the maximum extent of the state space (except for the α-puzzle, to be introduced below, in which case a value of 1.1 was found to work much better). For all algorithms, early termination (*e.g.*, the TERMINATE routine of FMT* and BFMT*) is suppressed by employing a 1000 second time limit, well above each planner's worst-case execution time.

Before proceeding, we emphasize that each individual marker shown on the subsequent plots of this section represents a single simulation at a fixed sample count. The points on the curves, however, represent the mean cost/time of *successful* algorithm runs *only* for a particular sample count, with

[2]State-of-the-art bi-directional sampling-based algorithms (namely, RRT-Connect and SBL) were initially also included. However, they were omitted due to high average costs (∼2-4x larger), which occluded the details of the other planners.

[3]Code for all experiments was written in C++. Corresponding programs were compiled and run on a Linux-operated PC, clocked at 2.4 GHz and equipped with 7.5 GB of RAM.

(a) $\mathbb{SE}(2)$ bug trap (b) $\mathbb{SE}(2)$ maze (c) $\mathbb{SE}(3)$ α-puzzle

Figure 4.1: Depictions of the OMPL rigid-body planning problems

error bars corresponding to one standard deviation of the 50 run sample mean.[4] Sample counts vary from about 200 to 2000 points for 2D problems, from 1000 to 30000 points for 3D problems, and from 500 to 4000 points for the hypercube examples.

4.1.2 Benchmarking Results and Discussion

Here we present benchmarking results (average solution cost versus average execution times and success rates) comparing FMT* and BFMT* to other state-of-the-art sampling-based planners. Three benchmarking test scenarios are considered: (i) a 2D "bug trap" and (ii) a 2D "maze" problem for a convex polyhedral robot in the $\mathbb{SE}(2)$ state space, as well as (iii) a challenging 3D problem called the "α-puzzle," in which we seek to untangle two loops of metal (non-convex) in the $\mathbb{SE}(3)$ state space. All problems are drawn directly from OMPL's bank of tests and are illustrated in Fig. 4.1. In each case, collision checks rely on OMPL's built-in collision-checking library, the Fast Collision Library (FCL). Additionally, to tease out the performance of BFMT* relative to FMT* in high-dimensional spaces, we also study a point mass robot moving in cluttered unit hypercubes of 5 and 10 dimensions.[5]

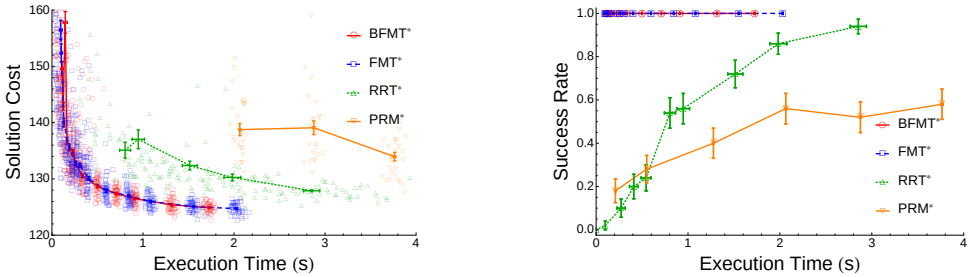
Figure 4.2 shows the results for each BFMT*, FMT*, RRT*, and PRM*. Performance here is measured by execution time on the x-axis and solution cost on the y-axis—high quality data points are therefore located in the lower-left corner (low-cost solutions that are obtained quickly). The plots reveal that both FMT* and BFMT* for the most part outperform RRT* as well as PRM*. In particular, BFMT* and FMT* achieve higher success rates[6] (always a flat 100% for the cases studied) in shorter time. To extract further information, we need to examine each test in detail.

In the Bug Trap and Maze problems, BFMT* notably generates the same cost-time curve as FMT* (meaning they return solutions of very similar cost for a given sample count), but with data

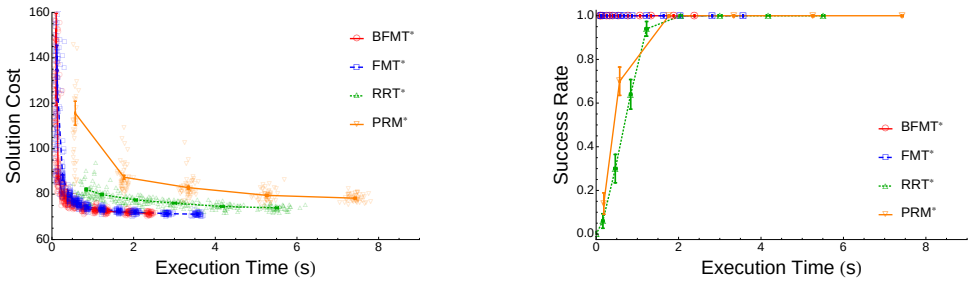
[4]Standard deviation of the mean indicates where we expect with one-σ confidence the distribution mean to lie based on the 50-run sample mean, and is related to the standard deviation of the distribution by $\sigma_\mu = \sigma \Big/ \sqrt{50}$.

[5]We populate the space to 50% obstacle coverage with randomly-sized, axis-oriented hyper-rectangles. $\mathbf{x}_{\text{init}}$ is set to the center at $[0.5, \ldots, 0.5]$, with the goal $\mathbf{x}_{\text{goal}}$ at the ones-vector (*i.e.*, $[1, \ldots, 1]$).

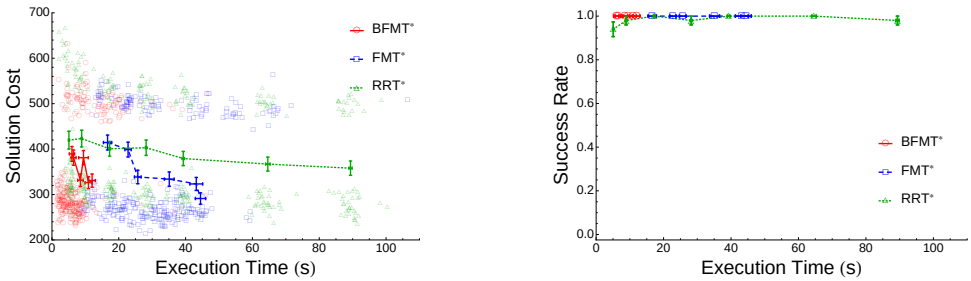
[6]Note we achieve even better success rates than in the original FMT* paper [80] due to the introduction of resampling from the INSERT algorithm.



(a) A bug trap in $\mathbb{SE}(2)$-space

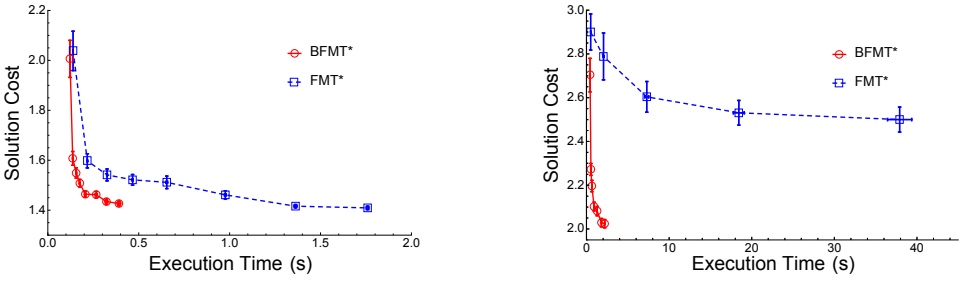


(b) A maze in $\mathbb{SE}(2)$-space



(c) The $\mathbb{SE}(3)$ "alpha" puzzle

Figure 4.2: Simulation results for the three OMPL test scenarios



(a) 5-D hypercube

(b) 10-D hypercube

Figure 4.3: FMT* and BFMT* results for 5D and 10D cluttered hypercubes (50% coverage; all success rates were 100%)

points shifted to the left (indicating they were obtained in shorter execution time). Though not shown due to slow running times for PRM* (whose results had to be truncated to clarify detail), all planners appear to tend towards similar low-cost solutions as more execution time is allocated. However, BFMT* and FMT* seem to converge to an optimum much faster, particularly for the Maze problem (on the order of 1.5 and 2.0 seconds respectively, compared to 3–4 seconds for RRT* and 5–7 seconds for PRM*). This contrast becomes even more evident for the α-puzzle. Here we see an unusual spread of solutions—one in a band at around 500 cost and another at around 275. These indicate the presence of two solution types, or *homotopy classes*: one corresponding to the true α-puzzle solution, and another less-efficient path. This appears to have yielded a "bump" in the BFMT* cost-curve, where increasing the sample count momentarily gives an increased average cost. We believe this is a result of how BFMT* trees interconnect; at this count, by unlucky circumstance, the longer homotopy seems to be found first more often than usual. But as proved in Section 3.3, the behavior disappears as $n \to \infty$. Note RRT* seems to avoid this issue through goal biasing. Despite the difficult problem structure, BFMT* finds the cheaper homotopy faster than other planners, with many more of its data points clustered in the lower-left corner, generally at lower costs and times than RRT* and of equal quality but faster times than FMT*.

These results suggest that BFMT* tends to an optimal cost at least as fast as the other planners, and sometimes much faster. To shed light on the relative performance of FMT* and BFMT* further, we compare them in higher dimensions. Results for the 5D and 10D hypercube are shown in Fig. 4.3 (success rates were again at 100%, and were thus omitted). Here BFMT* substantially outperforms FMT*, particularly as dimension increases, with convergence in roughly 0.5 and 1.4 seconds (5D), and 5 and 20 seconds (10D) on average. This suggests that *reachable volumes* play a significant role in their execution time. The relatively-small volume of reachable states around the goal at the corner implies that the reverse tree of BFMT* expands its wavefront through many fewer states than the forward tree of FMT* (which in fact needlessly expands towards the zero-vector); tree interconnection in the bi-directional case prevents its forward tree from growing too large compared to unidirectional search. This is pronounced exponentially as the dimension increases. In trap or maze-like scenarios, however, bi-directionality does not seem to change significantly the number of states explored by the marching trees, leading to comparable performance for the $\mathbb{SE}(2)$ bug-trap and maze. Note we expect a greater contrast in execution times in favor of BFMT* as the cost of collision-checking increases, such as with many non-convex obstacles or in time-varying environments.

4.2 Stanford Free-Flyer Demonstrations

Though simulations can help assess relative capabilities, the ultimate test for any planning algorithm is its performance while operating on real, application-specific hardware. In this section, we describe path planning experiments performed on a physical testbed representative of a planar spacecraft

operating in deep-space (in the absence of gravity). These demonstrations are designed to support sampling-based planning as a proof-of-concept for real-time guidance during autonomous on-orbit servicing. To begin, we describe in detail the experimental setup and justify its design as a representative simulator of deep-space spacecraft proximity operations. We then proceed to provide selective illustrative results of the testbed's planning performance, drawn from several cumulative hours of successful testing operation. Note that throughout the remainder of this chapter, we employ only the unidirectional Fast Marching Trees (FMT*) algorithm to keep the presentation simple, though arguably the Bi-directional Fast Marching Trees (BFMT*) algorithm could be applied here (to great effect) just as easily.

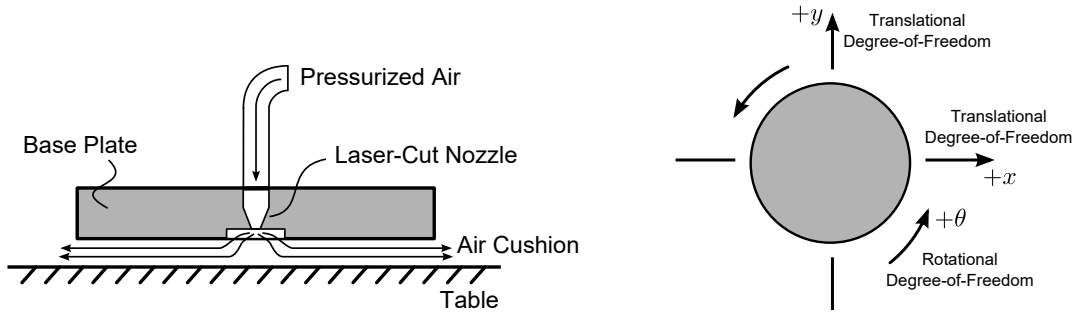
4.2.1 Experimental Setup

To study whether path planning algorithms can be realistically applied to spacecraft proximity operations guidance, it is first necessary to develop a suitable testing environment. To that end, we employ a set of three air-bearing, free-flying robots from the Stanford Space Robotics Facility. As shown in Fig. 4.4a, air bearings are thin films of pressurized air that provide a contact-free, very low friction, load-bearing interface between two surfaces. In this case, the robots generate air bearings between their lower base plates (polished flat aluminum disks) and a large, very flat granite table (see Fig. 4.5). Each air bearing is maintained by air that is drawn from onboard pressurized air tanks and then forced through laser-cut holes in the base plate bottom. The resulting system, if the robot is moving slowly enough to neglect air resistance, provides unrestricted lateral and yawing motions, which emulate zero-gravity conditions within the plane of the table. Such air-bearing simulators have been used for decades as affordable and relatively-cheap algorithmic testing platforms [122], which closely mimic 3-DOF "deep-space" spacecraft dynamics (undamped second-order dynamics with 2 translational degrees-of-freedom for planar translation plus one rotational degree-of-freedom about the plane normal; see Fig. 4.4b). It is interesting to note that this particular test facility at Stanford is the precursor to the more advanced facility shown in [123, 124]. For other representative examples of the utility of air-bearing simulators in spacecraft control experiments, see also [67, 125, 126].

In the paragraphs below, we highlight the key guidance and control features of the test facility and sketch out how hardware demonstrations are performed. We begin with a description of the localization system installed within the Stanford Space Robotics Facility, followed by details on the free-flyer robotic platform and their associated control software. In what follows, we restrict our focus to demonstrating how such previously-mentioned sampling-based path planning algorithms can be embedded into a real-time guidance and control architecture.

Localization

Localization is the process by which a robot determines its own position as well as those of any other objects and obstacles present nearby. To enable localization for the spacecraft free-flyers



(a) Schematic of an air bearing (Cross-sectional Side View)

(b) Degrees-of-Freedom enabled by an air bearing (Top View)

Figure 4.4: Conceptualizing an air-bearing simulator

Figure 4.5: The Stanford Space Robotics Laboratory testbed, including early (refurbished) versions of the air-bearing, free-flying robots. The granite table shown is 9 x 12 ft in size, with a flatness better than one micron.

and decouple its complications (beyond the scope of our work) from that of guidance, we relied on a relatively-simple and highly-accurate technique called *motion capture*. Motion capture systems cross-correlate images taken by multiple infrared cameras with overlapping fields-of-view to identify in real-time the positions of special markers called retroreflectors. Specialized algorithms are used to quickly isolate and identify common infrared reflections between camera images, from which positions can be calculated by triangulation, assuming camera coordinates remain fixed. By associating uniquely-patterned clusters of markers with objects, the motion capture system is able to estimate and broadcast to high accuracy the positions and orientations of all robots and obstacles within the testing environment, which in this case is the large granite table. This information can then be used, if needed, to obtain estimates of velocity and angular rates through appropriate filtering techniques. Though a somewhat crude comparison, one can think of motion capture here as simulating relative-GPS for our spacecraft free-flyers.

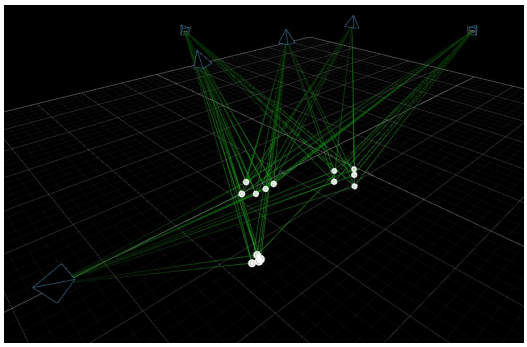


Figure 4.6: Visualizing the motion capture process for identifying the positions and orientations of test objects. The three clusters of white dots represent three groups of retro-reflectors, one for each free-flying robot. Green lines indicate the line-of-sight measurements used for triangulation that have been derived from images taken by the motion tracking cameras, illustrated here by blue pyramids.

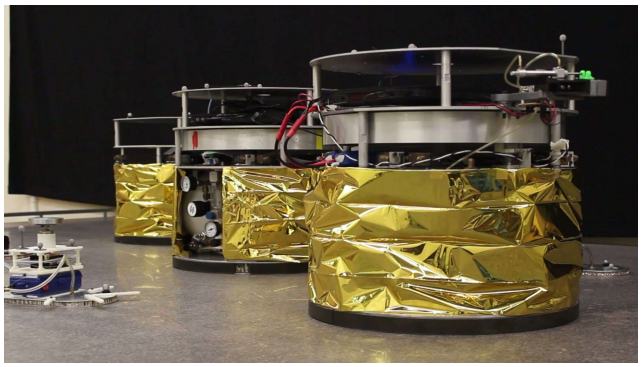
(a) An Optitrack S250e camera mounted above a table corner.

(b) Three additional cameras mounted directly above the table centerline.

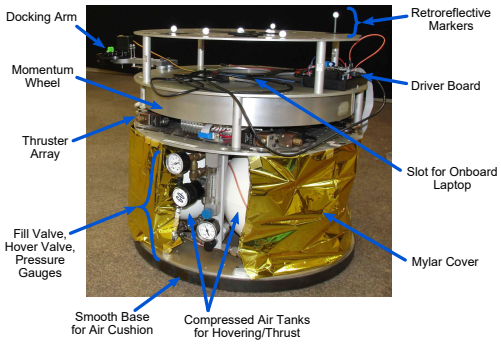
Figure 4.7: Motion capture system cameras installed within the Stanford Space Robotics Facility. A total of six cameras provide full coverage of the granite table.

The motion capture system used in the Stanford Space Robotics Facility consists of six OptiTrack S250e cameras mounted to the ceiling at several strategic vantage points, together with a centralized workstation, modem, and router for the processing and broadcasting of position and orientation data. For visualization, see Figs. 4.6 and 4.7). Though such a system is admittedly not available in any realistic spacecraft application,[7] it should be emphasized that the focus of this work is on the *guidance* aspect; testbed demonstrations are included primarily as a proof-of-concept, and hence we isolate localization from our guidance approach.

[7] One could achieve a more realistic testing simulation by emulating sensing uncertainty through artificial degradation of motion capture estimates.



(a) Free-flyer robots on the granite table. Note the rightmost robot is shown with a simple pneumatic docking mechanism, while the leftmost robot is shown without a momentum wheel and onboard avionics. Two of the three passive mini free-flyers can also be seen near the lower left and right corners.



(b) Free-flyer equipment breakdown. Note here the onboard laptop is removed for clarity.

Figure 4.8: Images of the free-flyer robots used for proximity operations demonstrations

Free-Flying Air-Bearing Robots

The three free-flying robots can be seen in Fig. 4.8. To simulate typical spacecraft actuation (within the plane of the table), each robot is outfitted with a set of eight coplanar, horizontally-oriented cold-gas thrusters for translation and a large horizontal momentum wheel for attitude control. To store thruster propellant as well as provide air pressure for their air bearings, each is also equipped with a set of three compressed-air tanks. The robots are controlled by onboard Linux-operated laptop computers, used to process localization data, plan safe paths, and issue path-following commands. Once computed, these control commands are sent from the computer over a serial connection to an MBED microcontroller, which then issues corresponding actuator commands through a custom-designed driver board to the thruster array and momentum wheel motor speed controller. Thruster solenoid valves are then opened or closed and the momentum wheel is spun-up or braked accordingly.[8] Motions for each robot are captured and broadcast by the localization system via a set of retroreflective markers or "dots" arranged in unique patterns on their top plates.

In addition to the primary free-flyer robots, a set of three mini free-flyers, designed without any actuators, are included in testbed demonstrations (see Fig. 4.8a). These smaller, passive air-bearing discs serve as additional obstacles for the primary free-flyers during proximity operations experiments. Due to their light weight, it is enough to use battery-powered air pumps to ensure their smooth levitation on the table. Like their larger counterparts, retroreflective dots are attached to their surfaces to enable their detection using the motion capture system.

[8]The hardware design reported here for the free-flying robots was determined largely during previous work at the Stanford Space Robotics Facility that dates as far back as the 1980's; the avionics and control architecture, however, were completely refurbished during the production of this thesis.

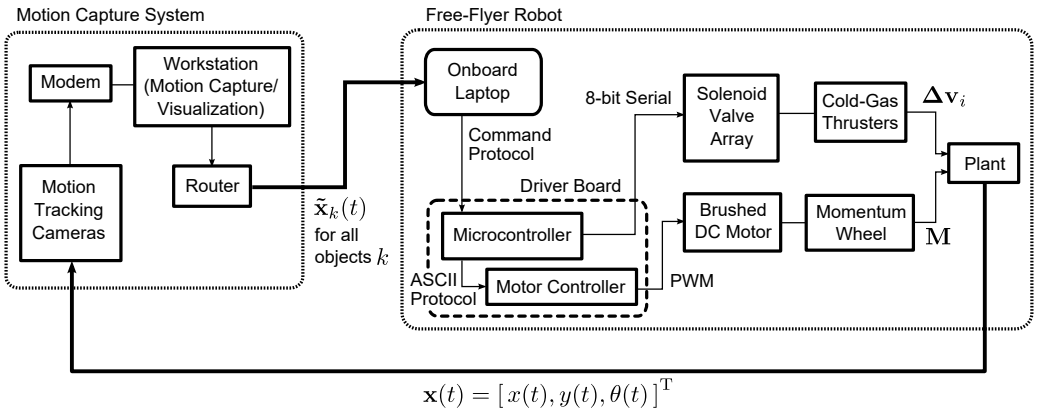


Figure 4.9: Architecture of the free-flying spacecraft testbed. Here $\tilde{\mathbf{x}}_k(t)$ represents the measured state trajectories of all objects k, $\mathbf{x}(t)$ is the actual robot state trajectory, $\mathbf{\Delta v}_i$ are the impulse vectors produced by each thruster i, and $\mathbf{M}$ is the applied momentum wheel moment vector.

Table 4.1: List of major hardware components, testing conditions, and other specifications for the free-flyer, air-bearing simulators

Base Diameter	0.49 m
Height	0.52 m
Mass (Momentum Wheel)	7.7 kg
Mass (Total)	32.4 kg
Microcontroller	mBED NXP LPC1768
Momentum Wheel Controller	Pololu Simple Motor Controller 18v15
Momentum Wheel Motor	BaneBots RS395 Brushed DC Motor
Onboard OS	Ubuntu Linux 13.04
Onboard Processor	Lenovo IdeaPad Y410p (8 GB RAM, 2.40 GHz clock speed)
Operating Pressures	500–700 PSI (Tanks), 80–100 PSI (Thrusters)
Power	3x Turnigy 4000 mAh 3S 20C Lithium-Polymer batteries
Propellant Source	3x Compressed Air Tanks (3000 PSI max)

Put together with the localization system, the overall architecture of our 3-DOF spacecraft proximity operations testbed can be seen in Fig. 4.9. More-detailed specifications on the air-bearing spacecraft simulator test beds are additionally listed in Table 4.1.

Planning

With the hardware established, we can now discuss how we embed a path planning algorithm, such as FMT*, into the guidance and control process for our spacecraft proximity operations simulator. The overall control system design, implemented entirely onboard the robot, can be seen in Fig. 4.10. Three asynchronous, independent threads are run simultaneously to manage various parts of the control process: a planning and estimation thread, a control thread, and finally an actuation thread.

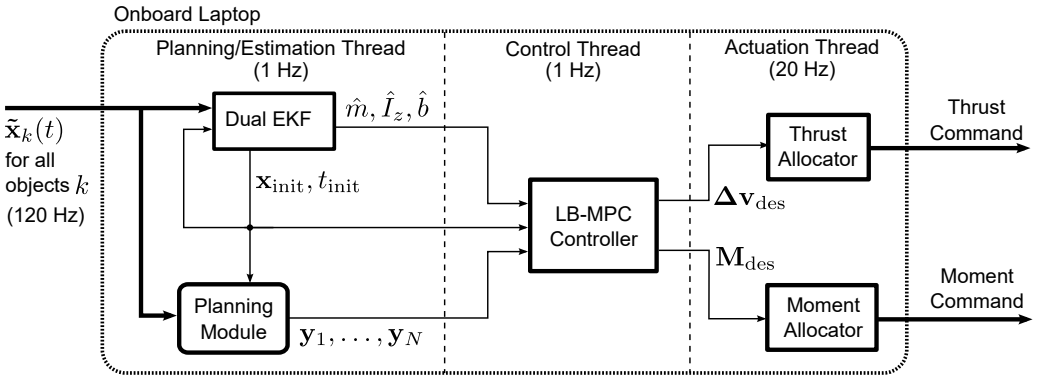


Figure 4.10: Architecture of the free-flyer on-board control system

Each of these is discussed below, in addition to a closer view of the planning thread guidance logic.

Planning and Estimation Thread Given localization measurements $\tilde{\mathbf{x}}_k(t)$ for all objects k including the robot j itself, the planning thread repeatedly updates the robot's estimate of its current table position and heading $\hat{\mathbf{x}} = \left[\hat{x}, \hat{y}, \hat{\theta}\right]$ using a full-state estimator, from which it develops a motion plan consisting of waypoints $\{\mathbf{y}_m\}_{m=1}^{N}$. In this case, we select a Dual Extended Kalman Filter (Dual EKF)—essentially two EKFs run together to jointly estimate system parameters as well as system states [127]—to provide smooth updates to the current robot state $\mathbf{x}_{\text{init}} \coloneqq \hat{\mathbf{x}}_j(t)$ at time $t_{\text{init}} \coloneqq t$, as well as the robot's estimated system mass $\hat{m}$, rotational moment of inertia $\hat{I}_z$, and a table friction parameter $\hat{b}$. These parameters update the linearized double-integrator dynamics model assumed by the Dual EKF and control thread (see below) in response to system changes over time caused by propellant expenditure, table surface irregularities, propulsive losses, *etc.* During the planning period between updates, the robot assumes all objects in its environment are "static," despite the fact that some may be moving. Provided plans are refreshed sufficiently-quickly (requiring a *real-time* planning algorithm), the robot can still respond and react to environmental changes well enough without any need for a predictive model of other agent's or obstacle's behavior; when admissible to a particular guidance problem, this *receding-horizon* type of planning approach is a simple, generalized way of handling time-varying obstacles using standard path planning algorithms (which often assume a static environment). For our slow-moving, low-thrust robots, a replanning period of 1 second was found to work well for the guidance scenarios we studied; more sophisticated methods for handling moving obstacles were deemed unnecessary.

Control Thread The control thread determines the desired net force and torque commands that efficiently drive the robot towards the discrete guidance trajectories $\{\mathbf{y}_m\}_{m=1}^{N}$ provided by the planning thread. In our implementation, we employed a Learning-Based Model Predictive Control (LB-MPC) algorithm as our trajectory-following controller [128]. This particular framework relies on

a predictive dynamics model for our system (here taken to be a simple double integrator), solves a control-effort two-point boundary-value minimization problem to determine a suitable control law, implements only the first step, and finally re-evaluates the problem after a given time interval. The "learning-based" part of the framework updates the predictive model over time—useful for accounting for uncertainties in our mass estimate $\hat{m}$, actuator uncertainties like unexpected thrust reductions caused by diminishing propellant pressurization and pressure waves in thruster tubing, and unmodeled system dynamics effects like momentum wheel friction.

Actuation Thread Once desired net force/moment commands have been given, the actuation thread parses them to individualized actuators, determining the appropriate times for thruster firings and the times and speeds at which to rotate the momentum wheel. For our particular approach, these forces and moments are first translated to the robot body-fixed frame before being broken into pulse widths (expressed in multiples of 0.05 s, the actuation loop time step), whose durations for any given pair of axis-aligned thrusters are determined by body-fixed thrust magnitudes. To improve control authority, thruster allocations are always conducted in pairs (axis-aligned pairs, in the case of translation, or counter-opposing offset pairs, in the case of rotational maneuvers). Once determined, these commands are then relayed to the driver board microcontroller to be passed on to actuators.

Guidance Logic The planning module used to generate motion plans merits special detail, for which we provide a schematic in Fig. 4.11. The first step involves generating an appropriate planning query $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$, for which we rely on the environmental measurements provided by our localization system. For automated rendezvous and docking, for example, our goal is to reach some position vector $\boldsymbol{\rho}$ relative to a target robot T. In this case, we take $\mathbf{x}_{\text{goal}}$ to be:

$$\mathbf{x}_{\text{goal}} = \tilde{\mathbf{x}}_{\text{T}}(t_{\text{init}}) + \boldsymbol{\rho}$$

Additionally, we need a mathematical representation of our workspace $\mathcal{W}$ (namely its obstacles $\mathcal{O}_k$) in order to evaluate whether states $\mathbf{x} \in \mathcal{X}$ are collision-free (*i.e.*, an *implicit* model for $\mathcal{X}_{\text{free}}$). For this, we generate obstacles from robot circumscribing circles with centers at current object positions $(x_k(t_{\text{init}}), y_k(t_{\text{init}}))$ and with radii r_k, as determined by our robot geometries (plus additional buffer room specified by the trajectory designer). Modeling the robotic free-flyer j by its own circumscribing circle with center (x, y) and radius r_j, state collisions can be evaluated by the simple check:

$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_k(t_{\text{init}}) \\ y_k(t_{\text{init}}) \end{bmatrix} \right\|^2 < (r_j + r_k)^2 \text{ for any } k \neq j$$

This example using circles demonstrates the general idea of workspace obstacle generation, though certain problems involving more complicated robot geometries may be represented better by more

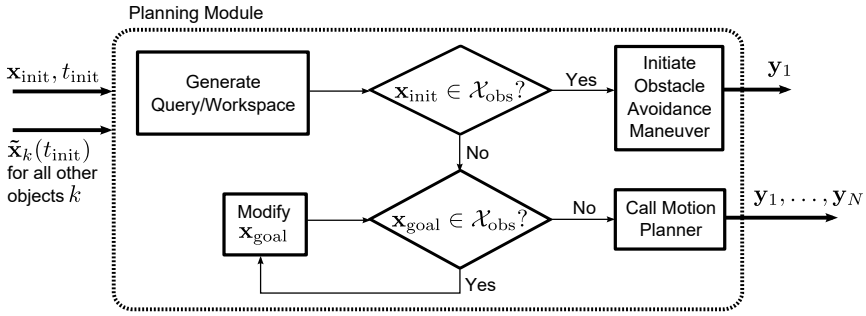


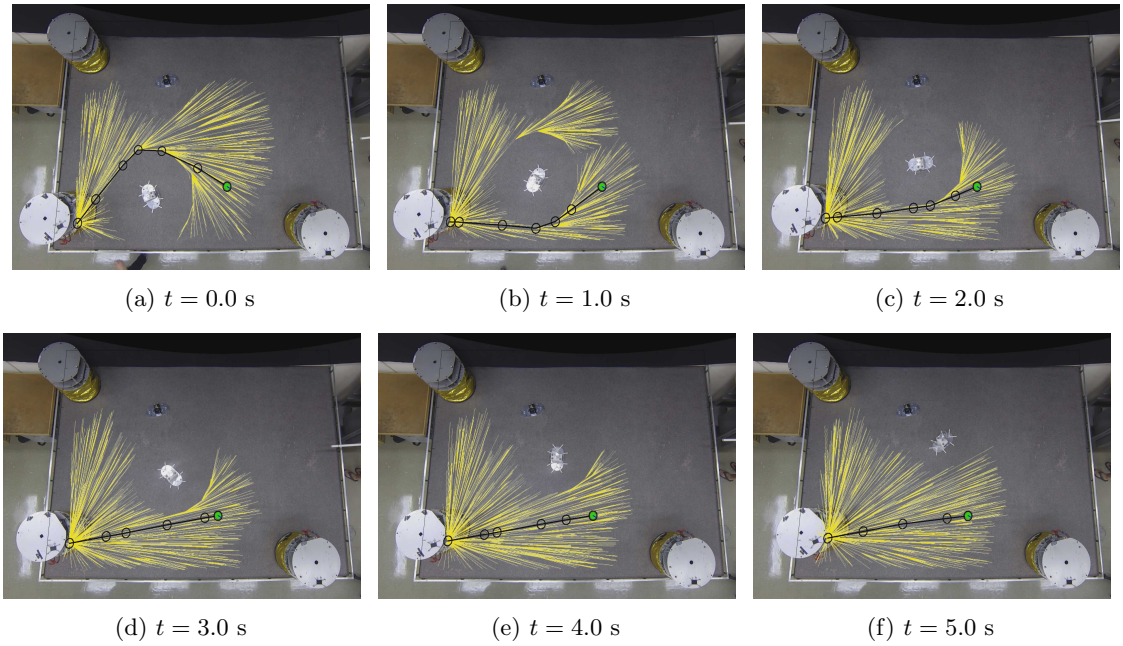
Figure 4.11: Architecture of the free-flyer guidance logic

complex or even hierarchical geometric primitives instead. See Chapter 5 for other examples. Now, recall from Definition 5 that a correct planning algorithm will report failure when no solution exists. One common such case occurs when either of the boundary conditions $\mathbf{x}_{\text{init}}$ or $\mathbf{x}_{\text{goal}}$ lie in $\mathcal{X}_{\text{obs}}$. This is particularly common for free-flying robots and other spacecraft-like systems due to the presence of drift (undamped momentum) and (often) significant state measurement uncertainty—state estimates can often drift very slightly into obstacle regions, particularly when operating near obstacle boundaries. The logic diagram shown in Fig. 4.11 illustrates the way in which we accommodate these edge cases. First, if $\mathbf{x}_{\text{init}}$ is found to be infeasible, an immediate avoidance boost is applied in the opposite direction of the target by setting our next desired path waypoint $\mathbf{y}_1$ to a point on the outside of the infeasible region on the line passing through $(x_{\text{init}}, y_{\text{init}})$ and $(x_k(t_{\text{init}}), y_k(t_{\text{init}}))$. For general spacecraft control, more sophisticated abort maneuvers would be needed (see Chapter 6), though this seemed to suffice for our purposes here. If $\mathbf{x}_{\text{init}}$ is safe but $\mathbf{x}_{\text{goal}}$ is found to be in violation, a similar projection technique of $\mathbf{x}_{\text{goal}}$ is used to find a safe goal point near our true objective. Once determined, motion planning is allowed to proceed and a sequence of waypoints $\{\mathbf{y}_m\}_{m=1}^N$ is generated for our trajectory-following controller.

When the motion planner in Fig. 4.11 is called, we run an Open Motion Planning Library (OMPL) [121] implementation of the FMT* algorithm, analogously to the numerical experiments of Section 4.1. Details on the FMT* algorithm, represented by pseudocode as Algorithm 1, may be found in Section 3.2.1. For our simulated spacecraft proximity operations testbed, we find that FMT*, in addition to producing low-cost paths, runs quickly enough to fit easily within our 1 Hz planning thread. FMT* execution times were often well below a second while using thousands of sample points—a benchmark that works well onboard our free-flyer robots.

4.2.2 Illustrative Results

With the setup described, we are now in position to present results from tests run on our air-bearing simulator. As any hardware implementation requires making application-specific design choices, which we touched on previously, the focus of this subsection is primarily on qualitative outcomes



(a) $t = 0.0$ s (b) $t = 1.0$ s (c) $t = 2.0$ s

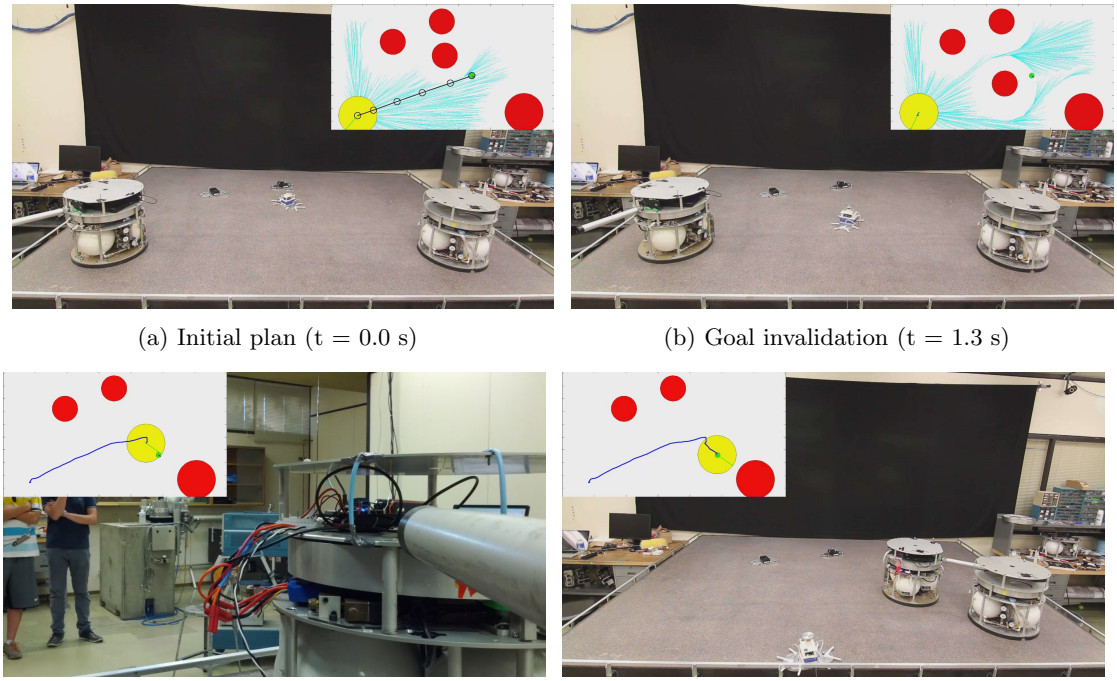
(d) $t = 3.0$ s (e) $t = 4.0$ s (f) $t = 5.0$ s

Figure 4.12: Illustration of the real-time replanning capability of the free-flyer robots. Observe how the bottom-left free-flyer's guidance plan (shown in black) and corresponding search tree (shown in yellow) adapt over time to the obstacle moving from the bottom-center upwards.

rather than quantitative data, which do not necessarily transfer from one application to another. It is hoped that the presentation here will convince the reader that the FMT* and BFMT* path planning algorithms can be used successfully for real-time robotic vehicle control when planning in tightly-constrained (and especially in slowly-changing) workspace environments.

Real-Time Capability

Let us first examine how well our guidance and control framework can handle time-varying obstacles. Figure 4.12 presents the search tree and resulting planning solution over time for a free-flyer in response to the motion of a mini free-flyer that is translating and rotating across the granite table surface. With a 1 Hz replanning rate, we see how the free-flyer adjusts its waypoint sequence according to the situation observed at the start of each planning interval. This changes the initial plan from passing in front of the mini free-flyer to one that lags behind it. One could do even better by attempting to predict the future of the mini free-flying "debris," such that no attempts would be made to build momentum along a path that may soon lead to a collision and instead saving time and propellant by immediately moving along the lower path; however, this complicates our solution and could lead to devastating results in situations where movement predictions are inaccurate. We found our more straightforward approach to work sufficiently-well for our demonstration purposes



(a) Initial plan (t = 0.0 s)

(b) Goal invalidation (t = 1.3 s)

(c) Goal convergence (t = 35.8 s)

(d) Final "docking" (t = 43.9 s)

Figure 4.13: Representative autonomous rendezvous and docking simulation with a time-varying obstacle. The chaser (yellow circle, initially lower-left) must plan around the mini free-flyers (small red circles) on its way to the target (lower-right, large red circle). The chaser's guidance plan is plotted in black (with goal state highlighted green) behind which trails its ground trace shown in blue. Safe explored paths found by FMT* are shown in teal.

given the fast replanning capability of FMT* and the low speeds of our spacecraft simulators.

Autonomous Rendezvous and Docking

From hours of cumulative operation, we present here two representative autonomous rendezvous and docking trials. The first is given in Fig. 4.13, which shows a series of testbed snapshots inlaid with the robot's corresponding motion plan and graphical representation of the workspace. The chaser robot begins in the lower-left corner of the table and seeks to maneuver to a point 1 m away from the target robot, seen in the lower-right corner, while aligning with its docking axis. This requires the chaser to implement a near-180° slew to orient its docking "arm" (represented by a straight rod) in the direction of the target's docking "port" (the slot between the two vertical blue wires seen in Fig. 4.13c). Along the way, the chaser must replan around a mini free-flyer that is dragged across its path by a string, temporarily invalidating our goal state. To complete the rendezvous maneuver, the robot initiates a straight-line "final approach" towards the target, abandoning any planning and obstacle avoidance and relying only at this point on its trajectory-following controller (note that no

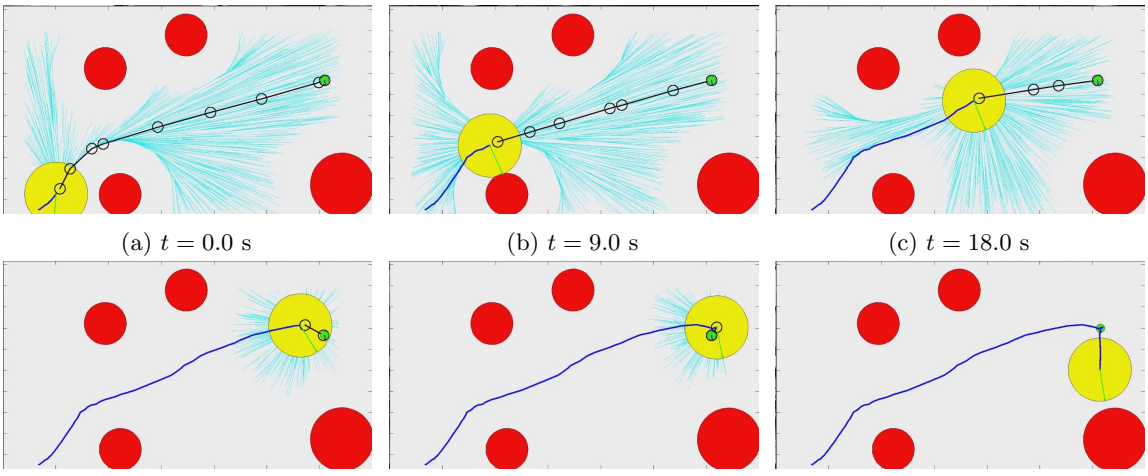
actual physical docking is performed).

As shown, the robot successfully completes its mission, moving (according to the ground trace from Fig. 4.13d) in more-or-less two straight line path segments from the initial state to the goal, and from the goal to docking. This quite closely traces the initial plan shown in Fig. 4.13a, suggesting that straight-line path planning can be a reasonable choice for approximating "deep-space" spacecraft motions, particularly for low-momentum systems like the one we study here. If you look closely, however, one can see a slight overshoot of the planning goal point just before final approach, caused by a lack of dynamic prediction in the guidance algorithm, which in fact takes quite a long time to correct. Path planning effectively assumes that infinite control authority exists, allowing the robot to change course instantaneously (ignoring momentum). This reveals that dynamic constraints will need to be explicitly accounted for in Part II of the thesis when developing our actual spacecraft planning framework, especially in scenarios where gravitational and inertial effects become more significant.

In addition, we see that despite the sudden environmental changes, the vehicle successfully ignores the mini free-flyer and converges smoothly to its goal point and final docking configuration. If the obstacle had actually been on a collision-course, however, the outcome would not have been so favorable; the low translational control authority (low thrust-to-weight ratio) of our free-flyer limits its obstacle avoidance abilities. This suggests that though real-time replanning capability is useful and necessary, it is not sufficient for ensuring future safety with respect to time-varying obstacles. In such scenarios, these kinds of dangerous situations must either be handled by using predictive modeling of obstacles or otherwise avoided entirely through appropriate mission safety constraints (a topic we will address in Chapter 6 for the case of control failures). So long as obstacles and targets are slowly-moving, the onboard architecture proposed in Fig. 4.10 appears to work quite well.

We provide the results from a second autonomous rendezvous and docking trial in Fig. 4.14. This time all mini free-flyers are held in static positions, and we focus entirely on the generated motion plans. Note the circular arc taken around the first obstacle, and the reductions in search tree size as the chaser approaches its rendezvous point. Again, due to the neglect of system dynamics, there is a slight overshoot of the goal point (which takes around 10-15 seconds to resolve as the chaser tries to reverse its linear momentum). This also causes deviations from the initial plan, though again the straight-line trajectories are followed quite closely.

All in all, it appears that our proposed control system works well using the FMT* motion planner, though it would be in our best interest to account for our system dynamics and vehicle safety directly during the guidance process. As will be seen, this will be accomplished, and more, in Part II after making appropriate modifications to the subroutines of FMT* (Algorithm 1) and BFMT* (Algorithm 4). To see more about the free-flying robots and to observe these autonomous guidance demonstrations in full detail, please refer to the supplementary videos that have been included with the electronic copy of this dissertation.



(a) $t = 0.0$ s (b) $t = 9.0$ s (c) $t = 18.0$ s

(d) $t = 27.0$ s (e) $t = 36.0$ s (f) $t = 60.1$ s

Figure 4.14: Another representative autonomous docking simulation within a purely-static obstacle field. The chaser (yellow circle, initially lower-left) must plan around the mini free-flyers (small red circles) on its way to the target (lower-right, large red circle). The chaser's guidance plan is plotted in black (with goal state highlighted green) behind which trails its ground trace shown in blue. Safe explored paths found by FMT* are shown in teal.

4.3 Conclusion

In this chapter, we evaluated the cost and run-time performance of the FMT* and BFMT* algorithms in a variety of motion planning settings. Numerical experiments in $\mathbb{R}^d$, $\mathbb{SE}(2)$, and $\mathbb{SE}(3)$ revealed that FMT* and BFMT* both tend to an optimal solution at least as fast as their state-of-the-art counterparts, and in some cases significantly faster. Experiments on a 3-DOF free-flying robotic testbed with FMT* showed that these algorithms can be used successfully on a simplified planar spacecraft proximity operations scenario, demonstrating onboard real-time capability at a 1 Hz planning rate.

According to our experience from numerical experiments, it seems the choice of whether to use FMT* and BFMT* for a particular guidance problem depends largely on workspace structure. In cases where the state space is relatively unconstrained, bi-directionality can yield significant run-time improvements. On the other hand, when goal regions have only limited entrance access (*e.g.*, through a narrow passage) but provide a large volume of exploration opportunities for the reverse tree, bi-directionality can be a hindrance; using only a forward tree in such cases skips the exploration of many unnecessary states. Both algorithms are hence useful tools for different, application-specific situations. From this point onwards in the thesis, to simplify the exposition and abstract away from any application-specific details, we will call solely upon the FMT* algorithm, with the assumed understanding that results could generally be extended to BFMT* as well.

Finally, though testbed demonstrations were successful, it is important to note their limited scope;

due to the very low thrust-to-weight ratios of our robots and the presence of table friction, dynamic motions were very slow (maneuvers took on the order of a minute over the space of a few meters) and drift effects were slight. In this limited setting, straight-line paths closely approximated state trajectories, enabling the naive use of path planning for vehicle guidance. However, this approach generally falls short during true spacecraft proximity operations, where drift effects dominate and trajectories deviate far from straight lines. In Part II of the thesis, we will modify our motion planning algorithms to directly account for the differential constraints present in real spacecraft missions, as well as a host of other complex trajectory constraints unique to spacecraft proximity operations.

Part II

Real-Time Algorithms for Spacecraft Motion Planning

With the insights developed from motion planning without differential constraints in the previous part of this dissertation, we turn our attention now to the more difficult problem of motion planning for spacecraft proximity operations (see Chapter 1), with a specific emphasis on *real-time* guidance for maneuvering near circular orbits. Real-time autonomous guidance for spacecraft is an inherently challenging task, particularly for onboard implementations where computational capabilities are limited. Many effective real-time solutions have been developed for the *unconstrained* problem, without any explicit trajectory constraints (*e.g.*, state transition matrix manipulation [11], Lambert targeting [12], glideslope methods [13], safety ellipses [51, 129], and others [130, 131]). However, the difficulty of real-time processing increases when there is a need to actively avoid nearby objects and/or incorporate some notion of propellant-optimality or control-effort minimization. In such cases, care is needed to efficiently handle collision-avoidance, plume impingement, sensor line-of-sight, and other complex guidance constraints, which are often non-convex and may depend on time and a mixture of state and control variables. State-of-the-art techniques for collision-free spacecraft proximity operations (both with and without optimality guarantees) include artificial potential function guidance [132, 133], nonlinear trajectory optimization with [50, 59] or without [134] convexification techniques, enforcing line-of-sight or approach corridor constraints [45, 135–137], maintaining relative separation [138], satisfying Keep-Out Zone (KOZ) constraints using mixed-integer (MI) programming [46], and sampling-based motion planning algorithms [54, 68, 84, 126].

Requiring hard assurances of mission safety with respect to a wide variety and number of potential failure modes [139] provides an additional challenge. Often the concept of *passive safety* (safety certifications on zero-control-effort failure trajectories) over a finite horizon is employed to account for the possibility of control failures; unfortunately, this strategy has the potential to neglect many mission-saving opportunities and fails to certify safety for all time. A less-conservative alternative that more readily adapts to infinite horizons, as we will see, is to use *active safety* in the form of positively-invariant set constraints. Positively-invariant sets are regions of the state space, such as equilibrium points and periodic trajectories, in which the vehicle remains for all future times, once entered. For instance, [45] enforces infinite-horizon active safety for a spacecraft by requiring each terminal state to lie on a collision-free orbit of equal period to the target. [54] achieves a similar effect by only planning between waypoints that lie on circular orbits (a more restrictive constraint). Likewise, [53] requires that an autonomous spacecraft maintain access to at least one safe forced equilibrium point nearby. Finally, [140] devises the Safe and Robust Model Predictive Control (MPC) algorithm, which employs invariant feedback tubes about a nominal trajectory (which guarantee resolvability) together with positively-invariant sets (taken about reference safety states) designed to be available at all times over the planning horizon.

Objective Though the aforementioned works [54, 68, 84, 126] on sampling-based planning for spacecraft proximity operations have addressed several components of the safety-constrained, propellant-optimal autonomous rendezvous problem, few if any have addressed simultaneously the aspect of

real-time implementability in conjunction with both a 2-norm propellant-cost metric and active trajectory safety with respect to control failures. The objective of the remainder of this thesis is to fill this gap, with specific emphasis on the case of near-circular orbit proximity operations, which we model using Clohessy-Wiltshire-Hill (CWH) dynamics. Our approach leverages *sampling-based motion planning algorithms* [141], specifically those described in Chapter 3, to efficiently solve a nonlinear propellant minimization problem with many non-convex, mixed state-time-control constraints and logical safety constraints. By relaxing the *online* guidance problem to a graph search problem between precomputed samples, and both (i) avoiding the *explicit* construction of the free state space and (ii) relegating the most computationally-difficult tasks to an offline precomputation phase, we can show that sampling-based planning algorithms—specifically asymptotically-optimal, batch sampling algorithms like FMT* and BFMT*—can be effective for real-time, actively-safe, propellant-efficient autonomous guidance during spacecraft proximity operations.

Organization This part's central pillar is a rigorous proof of asymptotic optimality for a particular sampling-based planner, namely a modified version of the Fast Marching Tree (FMT*) algorithm[9] [80], under impulsive CWH spacecraft dynamics with hard safety constraints. First, a description of the problem scenario is provided in Chapter 5, along with a formal definition of the sum-of-2-norms cost metric that we employ as a proxy for propellant consumption. Chapter 6 then follows with a thorough discussion of chaser/target vehicle safety, defining precisely how abort trajectories may be designed under CWH dynamics to deterministically avoid for all future times an ellipsoidal region about the CWH frame origin under a prescribed set of control failures. Next, we proceed in Chapter 7 to our proposed approach employing the modified FMT* algorithm. The chapter begins with the presentation of a conservative approximation to the propellant-cost reachability set, which characterizes the set of states that are "nearby" to a given initial state in terms of propellant use. These sets, bounded by unions of ellipsoidal balls, are then used to show that the modified FMT* algorithm maintains its (asymptotic) optimality when applied to CWH dynamics under our propellant-cost metric. From there, the chapter presents two techniques for improving motion planning solutions: (i) an analytical technique that can be called both during planning and post-processing to merge Δv-vectors between any pair of concatenated graph edges, and (ii) a continuous trajectory smoothing algorithm, which can reduce the magnitude of Δv-vectors by relaxing the implicit constraint to pass through sample points while still maintaining solution feasibility.

The combination of these tools into a unified framework provides a flexible, general technique for near-circular orbit spacecraft trajectory generation that automatically guarantees bounds on run time and solution quality (propellant cost) while handling a wide variety of (possibly non-convex) state, time, and control constraints. The methodology is demonstrated in Chapter 8 on a single-chaser,

[9] Due to BFMT*'s comparable performance with FMT* in path planning simulations from Chapter 4, attention is focused in this part of the dissertation on FMT* in order to keep the presentation simple, though results are expected to extend easily to BFMT* due to its close relationship to FMT* as described in Section 3.2.2.

single-target scenario simulating a near-field Low Earth Orbit (LEO) approach with hard constraints on total maneuver duration, relative positioning (including Keep Out Zone and antenna interference constraints), thruster plume impingement avoidance, individual and net Δv-vector magnitudes, and a two-fault thruster stuck-off failure tolerance. For the scenario posed, numerical trades are conducted studying the relative effects of the sample count n and a cost threshold parameter $\bar{J}$ on the solution cost and run time for FMT*, both with and without trajectory smoothing.

Note this part of the dissertation is also documented in the journal paper [92], as well as its associated conference papers [55] and [91].

Chapter 5

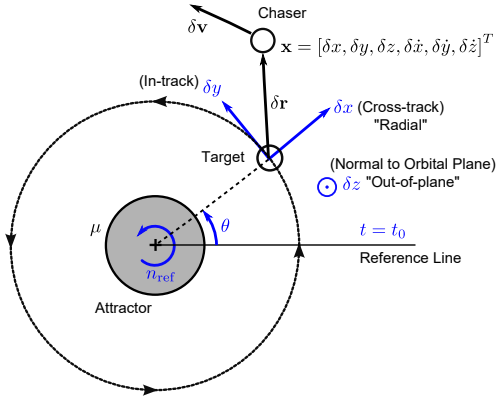
Problem Formulation

We begin our discussion on motion planning for spacecraft proximity operations by defining in this chapter the precise mathematical representation of the problem we wish to solve. For the remainder of this thesis, we model our problem as the near-field approach of a chaser spacecraft seeking to maneuver towards a single target moving in a well-defined, circular orbit (see Fig. 5.1a).

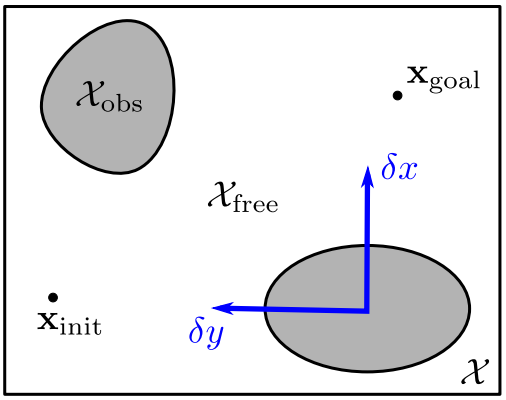
5.1 The Spacecraft Motion Planning Problem

Define our *state space* $\mathcal{X} \subset \mathbb{R}^d$ as a d-dimensional region in the target's Local Vertical, Local Horizontal (LVLH) frame, and let the *obstacle region* or $\mathcal{X}_{\text{obs}}$ be the set of states within $\mathcal{X}$ that result in immediate mission failure (*e.g.*, states outside of a specified approach corridor or which collide with the target, for example). Let the *free space* or $\mathcal{X}_{\text{free}}$ be the complement of $\mathcal{X}_{\text{obs}}$. As illustrated in Fig. 5.1b, let $\mathbf{x}_{\text{init}}$ represent the chaser's initial state relative to the target, and let $\mathbf{x}_{\text{goal}} \in \mathcal{X}_{\text{goal}}$ be a goal state inside the *goal region* $\mathcal{X}_{\text{goal}}$. Finally, define a *state trajectory* (or simply "trajectory") as a piecewise-continuous function of time $\mathbf{x}(t) : \mathbb{R} \to \mathcal{X}$, and let Σ represent the set of all state trajectories. Every state trajectory is implicitly generated by a control trajectory $\mathbf{u}(t) : \mathbb{R} \to \mathcal{U}$, where $\mathcal{U}$ is the set of controls, through the system dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$, where f is the system's state transition function. A state trajectory is called a *feasible* solution to the planning problem $(\mathcal{X}_{\text{free}}, t_{\text{init}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ if: (i) it satisfies the boundary conditions $\mathbf{x}(t_{\text{init}}) = \mathbf{x}_{\text{init}}$ and $\mathbf{x}(t_{\text{final}}) = \mathbf{x}_{\text{goal}}$ for some time $t_{\text{final}} > t_{\text{init}}$, (ii) it is *collision-free*; that is, $\mathbf{x}(t) \in \mathcal{X}_{\text{free}}$ for all $t \in [t_{\text{init}}, t_{\text{final}}]$, and (iii) it obeys all other trajectory constraints, including the system dynamics. The optimal motion planning problem can then be defined as follows:

Definition 13 (Optimal Planning Problem)**.** Given a planning problem $(\mathcal{X}_{\text{free}}, t_{\text{init}}, \mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ and a cost functional $J : \Sigma \times \mathcal{U} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$, find a feasible trajectory $\mathbf{x}^*(t)$ with associated control trajectory $\mathbf{u}^*(t)$ and time span $t = [t_{\text{init}}, t_{\text{final}}]$ for $t_{\text{final}} \in [t_{\text{init}}, \infty)$ such that $J(\mathbf{x}^*(\cdot), \mathbf{u}^*(\cdot), t) =$



(a) Schematic of CWH dynamics, which models relative guidance near a single target in a circular orbit.



(b) A representative motion planning query between feasible states $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$.

Figure 5.1: Illustration of the CWH planning scenario. Here n_{ref} is the mean motion of the target spacecraft orbit, θ is its mean anomaly, t denotes time, and $\delta\mathbf{r}$ and $\delta\mathbf{v}$ are the chaser relative position and velocity. The CWH (LVLH) frame $(\delta x, \delta y, \delta z)$ rotates with the target at rate n_{ref} as it orbits the gravitational attractor, μ. Planning takes place in state space $\mathcal{X}$, in which a safe trajectory is sought between states $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ within the feasible subspace $\mathcal{X}_{\text{free}}$ around obstacle region $\mathcal{X}_{\text{obs}}$.

$\min\{J(\mathbf{x}(\cdot), \mathbf{u}(\cdot), t) \mid \mathbf{x}(t) \text{ and } \mathbf{u}(t) \text{ are feasible}\}$. If no such trajectory exists, report failure.

Note this is much more general than the path planning problem (Definition 5) introduced in Chapter 3. For our work, we employ a control-effort cost functional J that considers only the control trajectory $\mathbf{u}(t)$ and the final time t_{final}, which we represent by the notation $J(\mathbf{u}(t), t_{\text{final}})$. Now, tailoring Definition 13 to impulsively-actuated propellant-optimal motion planning near circular orbits (where here we assume propellant optimality is measured by a sum-of-2-norms metric), the specific spacecraft motion planning problem we wish to solve is formulated as:

$$
\begin{array}{lllll}
\text{Given:} & \text{Initial state } \mathbf{x}_{\text{init}}(t_{\text{init}}), \text{Goal region } \mathcal{X}_{\text{goal}}, \text{Free space } \mathcal{X}_{\text{free}} & & & (5.1) \\
\underset{\mathbf{u}(t), t_{\text{final}}}{\text{minimize}} & J(\mathbf{u}(t), t_{\text{final}}) = \displaystyle\int_{t_{\text{init}}}^{t_{\text{final}}} \|\mathbf{u}(t)\|_2 \, \mathrm{d}t = \sum_{i=1}^{N} \|\mathbf{\Delta v}_i\|_2 & & & \\
\text{subject to} & \mathbf{x}(t_{\text{init}}) = \mathbf{x}_{\text{init}} & & & \text{Initial Condition} \\
& \mathbf{x}(t_{\text{final}}) \in \mathcal{X}_{\text{goal}} & & & \text{Terminal Condition} \\
& \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) & & & \text{System Dynamics} \\
& \mathbf{x}(t) \in \mathcal{X}_{\text{free}} & \text{for all } t \in [t_{\text{init}}, t_{\text{final}}] & & \text{Obstacle Avoidance} \\
& \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0 & \text{for all } t \in [t_{\text{init}}, t_{\text{final}}] & & \text{Other Constraints} \\
& \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) = 0 & & & \\
& \exists \text{ safe } \mathbf{x}_{\text{CAM}}(\tau), \tau > t & \text{for all } \mathbf{x}(t) & & \text{Active Safety}
\end{array}
$$

where t_{init} and t_{final} are the initial and final times and $\mathbf{x}_{\text{CAM}}(\tau)$ refers to an infinite-horizon Collision-Avoidance-Maneuver (CAM). Note, in Eq. (5.1) above, we restrict our attention to impulsive control laws $\mathbf{u}(t) = \sum_{i=1}^{N} \boldsymbol{\Delta}\mathbf{v}_i \delta(t - \tau_i)$, where $\delta(\cdot)$ denotes the Dirac delta function, which model finite sequences of instantaneous translational burns $\boldsymbol{\Delta}\mathbf{v}_i$ fired at discrete times τ_i (note that the number of burns N is not fixed *a priori*). Though possible to consider all control laws, it is both theoretically- and computationally-simpler to optimize over a finite-dimensional search space of Δv-vectors; furthermore, these represent the most common forms of propulsion systems used on-orbit, including high-impulse cold-gas and liquid bi-propellant thrusters, and they can (at least in theory) approximate continuous control trajectories in the limit that $N \to \infty$.

We now elaborate on the objective function and each constraint in turn.

5.1.1 Cost Functional

A critical component of the spacecraft rendezvous problem is the choice of cost functional. Consistent with [49], we define our cost as the L^1-norm of the ℓ_p-norm of the control. The best choice for $p \geq 1$ depends on the propulsion system geometry, and on the frame within which $\mathbf{u}(t) = \sum_{i=1}^{N} \boldsymbol{\Delta}\mathbf{v}_i \delta(t - \tau_i)$ in J is resolved. Unfortunately, minimizing propellant exactly involves resolving vectors $\boldsymbol{\Delta}\mathbf{v}_i$ into the spacecraft body-fixed frame, requiring spacecraft attitude $\mathbf{q}$ to be included in our state $\mathbf{x}$. To avoid this, a standard used throughout the literature and routinely in practical applications is to employ $p = 2$ so that each $\boldsymbol{\Delta}\mathbf{v}_i$ is as short as possible, allocating the commanded $\boldsymbol{\Delta}\mathbf{v}_i$ to thrusters in a separate control allocation step (conducted later, once the attitude is known; see Section 5.1.5 for more detail). Though this moves propellant minimization online, it greatly simplifies the guidance problem in a practical way without neglecting attitude. Because the cost of Δv-allocation can only grow from the need to satisfy torque constraints or impulse bounds (*e.g.*, necessitating counter-opposing thrusters to achieve the same net Δv-vector), we are in effect minimizing the best-case, unconstrained propellant use of the spacecraft. As we will show in our numerical experiments, however, this does not detract significantly from the technique; the coupling of J with $p = 2$ to the actual propellant use through the minimum control-effort thruster Δv allocation problem seems to promote low propellant-cost solutions. Hence (in practice) J serves as a good proxy to propellant use, with the added benefit of independence from propulsion system geometry.

5.1.2 Boundary Conditions

Planning is assumed to begin at a known initial state $\mathbf{x}_{\text{init}}$ and time t_{init} and end at a single goal state $\mathbf{x}_{\text{goal}}^{\text{T}} = \begin{bmatrix} \delta\mathbf{r}_{\text{goal}}^{\text{T}} & \delta\mathbf{v}_{\text{goal}}^{\text{T}} \end{bmatrix}$ ("exact convergence," $\mathcal{X}_{\text{goal}} = \{\mathbf{x}_{\text{goal}}\}$), where $\delta\mathbf{r}_{\text{goal}}$ is the goal position and $\delta\mathbf{v}_{\text{goal}}$ is the goal velocity. During numerical experiments, however, we sometimes permit termination at any state whose position and velocity lie within Euclidean balls $\mathcal{B}(\delta\mathbf{r}_{\text{goal}}, \epsilon_r)$ and $\mathcal{B}(\delta\mathbf{v}_{\text{goal}}, \epsilon_v)$, respectively ("inexact convergence," $\mathcal{X}_{\text{goal}} = \mathcal{B}(\mathbf{r}_{\text{goal}}, \epsilon_r) \times \mathcal{B}(\mathbf{v}_{\text{goal}}, \epsilon_v)$), where the notation $\mathcal{B}(\mathbf{r}, \epsilon) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{r} - \mathbf{x}\| \leq \epsilon\}$ denotes a ball with center $\mathbf{r}$ and radius ϵ.

5.1.3 System Dynamics

Because spacecraft proximity operations incorporate significant drift, spatially-dependent external forces, and changes on fast timescales, any realistic solution must obey dynamic constraints; we cannot in general assume straight-line trajectories (see the discussion in Section 4.2.2). In this thesis, we employ the classical Clohessy-Wiltshire-Hill (CWH) equations [142, 143] for impulsive linearized motion about a circular reference orbit at radius r_{ref} about an inverse-square-law gravitational attractor with parameter μ. This model provides a first-order approximation to a chaser spacecraft's motion relative to a rotating target-centered coordinate system (see Fig. 5.1). As can be shown from Eq. (A.8) in Appendix A, the linearized equations of motion for this scenario as resolved in the Local Vertical, Local Horizontal (LVLH) frame of the target are given by:

$$\delta\ddot{x} - 3n_{\text{ref}}^2\delta x - 2n_{\text{ref}}\delta\dot{y} = \frac{F_{\delta x}}{m} \tag{5.2a}$$

$$\delta\ddot{y} + 2n_{\text{ref}}\delta\dot{x} = \frac{F_{\delta y}}{m} \tag{5.2b}$$

$$\delta\ddot{z} + n_{\text{ref}}^2\delta z = \frac{F_{\delta z}}{m} \tag{5.2c}$$

where $n_{\text{ref}} = \sqrt{\frac{\mu}{r_{\text{ref}}^3}}$ is the orbital frequency (mean motion) of the reference spacecraft orbit, m is the spacecraft mass, $\mathbf{F} = [F_{\delta x}, F_{\delta y}, F_{\delta z}]^{\text{T}}$ is some applied force, and $(\delta x, \delta y, \delta z)$ and $(\delta\dot{x}, \delta\dot{y}, \delta\dot{z})$ represent the cross-track ("radial"), in-track, and out-of-plane relative position and velocity vectors, respectively. The CWH model is used often, especially for short-duration rendezvous and proximity operations in Low Earth Orbit (LEO) and for leader-follower formation flight dynamics.

Defining the state $\mathbf{x}$ as $[\,\delta x, \delta y, \delta z, \delta\dot{x}, \delta\dot{y}, \delta\dot{z}\,]^{\text{T}}$ and the control $\mathbf{u}$ as the applied force-per-unit-mass $\frac{1}{m}\mathbf{F}$, the CWH equations can be described by the *linear time-invariant* (LTI) system:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.3}$$

where the dynamics matrix $\mathbf{A}$ and input matrix $\mathbf{B}$ are given by:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n_{\text{ref}}^2 & 0 & 0 & 0 & 2n_{\text{ref}} & 0 \\ 0 & 0 & 0 & -2n_{\text{ref}} & 0 & 0 \\ 0 & 0 & -n_{\text{ref}}^2 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

See Appendix A.1 for a detailed derivation.

As for any LTI system (with matrices $\mathbf{A}$ and $\mathbf{B}$ of constant coefficients), there exists a *unique* solution to Eq. (5.3) given initial condition $\mathbf{x}(t_0)$ and integrable input $\mathbf{u}(t)$ that can be expressed

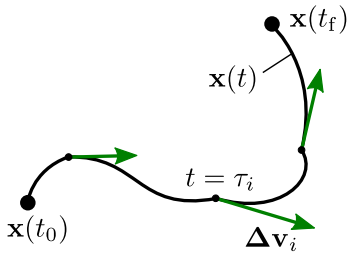


Figure 5.2: Representation of an impulsive state trajectory and its underlying control law. $\mathbf{x}(t_0)$ and $\mathbf{x}(t_\mathrm{f})$ represent the initial and final states, $\boldsymbol{\Delta}\mathbf{v}_i$ a velocity impulse (or "burn") applied at time τ_i, and $\mathbf{x}(t)$ the resultant state trajectory simulated over time interval $t \in [t_0,\, t_\mathrm{f}]$.

using superposition and the convolution integral as $\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\,\mathrm{d}\tau$ for any time $t \geq t_0$. The expression $\boldsymbol{\Phi}(t,\tau) \triangleq e^{\mathbf{A}(t-\tau)}$, called the *state transition matrix*, provides an important analytical mechanism for computing state trajectories that we rely heavily upon in simulations. Note, throughout this work, we shall sometimes represent $\boldsymbol{\Phi}(t,\tau)$ as $\boldsymbol{\Phi}$ for brevity when its arguments are understood.

We now specialize the above to the case of N impulsive velocity changes at times $t_0 \leq \tau_i \leq t_\mathrm{f}$ for $i \in [1,\ldots,N]$ (see Fig. 5.2), in which case $\mathbf{u}(\tau) = \sum_{i=1}^{N} \boldsymbol{\Delta}\mathbf{v}_i \delta(\tau - \tau_i)$, where $\delta(y) = \{1 \text{ where } y = 0, \text{ or } 0 \text{ otherwise}\}$ signifies the Dirac-delta distribution. Substituting for $\boldsymbol{\Phi}$ and $\mathbf{u}(\tau)$,

$$\begin{aligned}\mathbf{x}(t) &= \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\mathbf{B}\left(\sum_{i=1}^{N} \boldsymbol{\Delta}\mathbf{v}_i \delta(\tau-\tau_i)\right)\mathrm{d}\tau \\ &= \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \sum_{i=1}^{N}\int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\mathbf{B}\boldsymbol{\Delta}\mathbf{v}_i \delta(\tau-\tau_i)\,\mathrm{d}\tau,\end{aligned}$$

where on the second line we used the linearity of the integral operator. By the sifting property of δ, denoting N_t as the number of burns applied from t_0 up to time t, we have for all times $t \geq t_0$ the following expression for the impulsive solution to Eq. (5.3):

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \sum_{i=1}^{N_t} \boldsymbol{\Phi}(t,\tau_i)\mathbf{B}\boldsymbol{\Delta}\mathbf{v}_i \tag{5.4a}$$

$$= \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \underbrace{\begin{bmatrix} \boldsymbol{\Phi}(t,\tau_1)\mathbf{B} & \ldots & \boldsymbol{\Phi}(t,\tau_{N_t})\mathbf{B} \end{bmatrix}}_{\triangleq \boldsymbol{\Phi}_v(t,\{\tau_i\}_i)} \underbrace{\begin{bmatrix} \boldsymbol{\Delta}\mathbf{v}_1 \\ \vdots \\ \boldsymbol{\Delta}\mathbf{v}_{N_t} \end{bmatrix}}_{\triangleq \boldsymbol{\Delta}\mathbf{V}} \tag{5.4b}$$

$$= \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \boldsymbol{\Phi}_v(t,\{\tau_i\}_i)\boldsymbol{\Delta}\mathbf{V}. \tag{5.4c}$$

For a more explicit representation of this solution, refer to Appendix A.2. Throughout this thesis, the

notations $\mathbf{\Delta V}$ for the stacked Δv-vector and $\mathbf{\Phi}_v(t, \{\tau_i\}_i)$ for the aggregated impulse state transition matrix (or simply $\mathbf{\Phi}_v$ for short, when the parameters t and $\{\tau_i\}_i$ are clear) implicitly imply only those burns i occurring before time t.

5.1.4 Obstacle Avoidance

Obstacle avoidance is imposed by requiring that the spacecraft trajectory $\mathbf{x}(t)$ stay within $\mathcal{X}_{\text{free}}$ (or, equivalently, outside of $\mathcal{X}_{\text{obs}}$)—typically a difficult non-convex constraint (see Fig. 5.1b). For CWH proximity operations, $\mathcal{X}_{\text{obs}}$ might include positions in collision with a nearby object, position/velocity pairs which lie outside of a given approach corridor, *etc.* In our numerical experiments, to prevent the chaser from interfering with the target, we assume $\mathcal{X}_{\text{obs}}$ comprises an ellipsoidal Keep-Out Zone (KOZ) centered at the origin and a conical nadir-pointing region that approximates the target's antenna beam pattern.

Note, according to the definition of $\mathcal{X}_{\text{free}}$, this also requires $\mathbf{x}(t)$ to stay within the confines of $\mathcal{X}$ (CWH dynamics do not guarantee that state trajectories will lie inside $\mathcal{X}$ despite the fact that their endpoints do). Though not always necessary in practice, if $\mathcal{X}$ marks the extent of reliable sensor readings or the boundary inside which linearity assumptions hold, then this can be useful to enforce.

5.1.5 Other Trajectory Constraints

Many other types of constraints may be included to encode additional restrictions on state and control trajectories, which we represent here by a set of inequality constraints $\mathbf{g}$ and equality constraints $\mathbf{h}$ (observe that $\mathbf{g}$ and $\mathbf{h}$ denote vector functions). To illustrate the flexibility of sampling-based planning, we encode the following into constraints $\mathbf{g}$ (for brevity, we omit their exact representation, which is a straightforward exercise in vector geometry):

$$
\begin{aligned}
&T_{\text{plan,min}} \leq t_{\text{final}} - t_{\text{init}} \leq T_{\text{plan,max}} && && \text{Plan Duration Bounds} \\
&\mathbf{\Delta v}_i \in \mathcal{U}(\mathbf{x}(\tau_i)) && \text{for all } i = [1, \ldots, N] && \text{Control Feasibility} \\
&\bigcup_{k \in [1,\ldots,K]} \mathcal{P}_{ik}(-\mathbf{\Delta \hat{v}}_{ik}, \beta_{\text{plume}}, H_{\text{plume}}) \cap \mathbb{S}_{\text{target}} = \varnothing && \text{for all } i = [1, \ldots, N] && \text{Plume Impingement}
\end{aligned}
$$

Here $0 \leq T_{\text{plan,min}} < T_{\text{plan,max}}$ represent minimum and maximum motion plan durations, $\mathcal{U}(\mathbf{x}(\tau_i))$ is the admissible *control set* corresponding to state $\mathbf{x}(\tau_i)$, $\mathcal{P}_{ik}$ is the exhaust plume emanating from thruster k of the chaser spacecraft while executing burn $\mathbf{\Delta v}_i$ at time τ_i, and $\mathbb{S}_{\text{target}}$ is the target spacecraft circumscribing sphere. We motivate each constraint in turn.

Plan Duration Bounds Plan duration bounds facilitate the inclusion of rendezvous windows based on the epoch of the chaser at $\mathbf{x}_{\text{init}}(t_{\text{init}})$; such windows might be determined by illumination requirements, grounds communication opportunities, or mission timing restrictions, for example.

$T_{\text{plan,max}}$ may also be used to ensure the errors incurred by our linearized CWH model, which grow with time, do not exceed acceptable tolerances.

Control Feasibility Control set constraints are intended to encapsulate limitations on control authority imposed by propulsive actuators and their geometric distribution about the spacecraft. For example, given the maximum burn magnitude $0 < \Delta v_{\max}$, the constraint:

$$\|\boldsymbol{\Delta}\mathbf{v}_i\|_2 \leq \Delta v_{\max} \text{ for all } i = [1, \ldots, N] \tag{5.5}$$

might represent an upper bound on the achievable impulses of a gimbaled thruster system that is free to direct thrust in all directions. In our case, we use $\mathcal{U}(\mathbf{x}(\tau_i))$ to represent all commanded net Δv-vectors that: (i) satisfy Eq. (5.5) above, and also (ii) can be successfully allocated to thrusters along trajectory $\mathbf{x}(t)$ at time τ_i according to a simple minimum-control effort thruster allocation problem (a straightforward linear program (LP) [144]). To keep the dissertation self-contained, we repeat the problem here and in our own notation. Let $\boldsymbol{\Delta}\mathbf{v}_i|_{\text{bf}}$ and $\mathbf{M}_i|_{\text{bf}}$ be the desired net Δv and moment vectors at burn time τ_i, resolved in the body-fixed frame according to attitude $\mathbf{q}(\tau_i)$ (we henceforth drop the bar, for clarity). Note the attitude $\mathbf{q}(\tau_i)$ must either be included in the state $\mathbf{x}(\tau_i)$ or be derived from it, as we assume here by imposing (along nominal trajectories) a nadir-pointing attitude profile for the chaser spacecraft. Let $\Delta v_{ik} = \|\boldsymbol{\Delta}\mathbf{v}_{ik}\|_2$ be the Δv-magnitude allocated to thruster k, which generates an impulse in direction $\boldsymbol{\Delta}\hat{\mathbf{v}}_{ik}$ at position $\boldsymbol{\rho}_{ik}$ from the spacecraft center-of-mass (both are constant vectors if resolved in the body-fixed frame). Finally, to account for the possibility of on or off thrusters, let η_{ik} be equal to 1 if thruster k is available for burn i, or 0 otherwise. Then the minimum-effort control allocation problem can be represented as:

$$\begin{aligned}
&\text{Given:} && \text{On-off flags } \eta_{ik}, \text{thruster positions } \boldsymbol{\rho}_{ik}, \text{thruster axes } \boldsymbol{\Delta}\hat{\mathbf{v}}_{ik}, \\
& && \text{commanded } \Delta v\text{-vector } \boldsymbol{\Delta}\mathbf{v}_i, \text{and commanded moment vector } \mathbf{M}_i \\
&\underset{\Delta v_{ik}}{\text{minimize}} && \sum_{k=1}^{K} \Delta v_{ik} \\
&\text{subject to} && \sum_{k=1}^{K} \boldsymbol{\Delta}\hat{\mathbf{v}}_{ik}(\eta_{ik}\Delta v_{ik}) = \boldsymbol{\Delta}\mathbf{v}_i && \text{Net } \Delta v\text{-Vector Allocation} \\
& && \sum_{k=1}^{K} (\boldsymbol{\rho}_{ik} \times \boldsymbol{\Delta}\hat{\mathbf{v}}_{ik})(\eta_{ik}\Delta v_{ik}) = \mathbf{M}_i && \text{Net Moment Allocation} \\
& && \Delta v_{\min,k} \leq \Delta v_{ik} \leq \Delta v_{\max,k} && \text{Thruster } \Delta v \text{ Bounds}
\end{aligned} \tag{5.6}$$

where $\Delta v_{\min,k}$ and $\Delta v_{\max,k}$ represent minimum and maximum impulse limits on thruster k (due to actuator limitations, minimum impulse bits, pulse-width constraints, or maximum on-time restrictions, for example). Because Δv is directly-proportional to thrust through the Tsiolkovsky rocket equation,

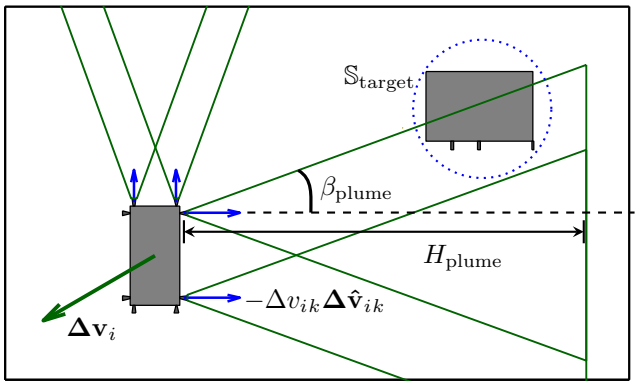


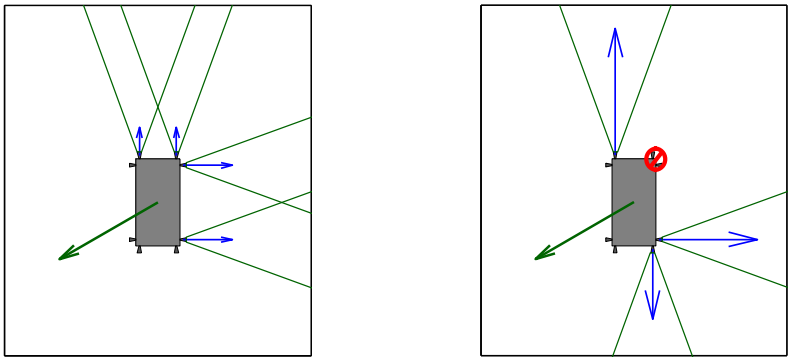
Figure 5.3: Illustration of exhaust plume impingement from thruster firings. Given commanded $\boldsymbol{\Delta}\mathbf{v}_i$, the spacecraft must successfully allocate the impulse to thrusters while simultaneously avoiding impingement of neighboring object(s).

the formulation above is directly analogous to minimum-propellant consumption; as discussed in Section 5.1.1, by using control trajectories that minimize commanded Δv-vector lengths $\|\boldsymbol{\Delta}\mathbf{v}_i\|$, we can drive propellant use downwards as much as possible subject to our thrust bounds and net torque constraints. In this work, we set $\mathbf{M}_i = \mathbf{0}$ to enforce torque-free burns and minimize disturbances to our assumed attitude trajectory $\mathbf{q}(t)$.

The value of the norm bound $\Delta v_{\max}$ may be computed from the thruster limits $\Delta v_{\min,k}, \Delta v_{\max,k}$ and knowledge of the thruster configuration. Note that we do not consider a minimum-norm constraint in Eq. (5.5) for $\boldsymbol{\Delta}\mathbf{v}_i$, as it is not necessary and would only complicate future proofs (to be seen). As discussed in Section 5.1.1, $\|\boldsymbol{\Delta}\mathbf{v}_i\|$ is only a proxy for the true propellant cost computed from the thrust allocation problem (Eq. (5.6)). Thruster allocation accounts for any propulsive limitations, firing opposing thrusters simultaneously, if necessary, to achieve a given net $\boldsymbol{\Delta}\mathbf{v}_i$ vector.

Plume Impingement As described in Section 1.1.3, impingement of thruster exhaust onto neighboring spacecraft can lead to dire consequences [47, 48]. To account for this during guidance, we first generate representative exhaust plumes at the locations of each thruster firing. For burn i occurring at time τ_i, a right circular cone with axis $-\boldsymbol{\Delta}\hat{\mathbf{v}}_{ik}$, half-angle β_{plume}, and height H_{plume} is projected from each *active* thruster k ($\eta_{ik} = 1$) whose allocated thrust Δv_{ik}^* is non-zero, as determined from Eq. (5.6). Intersections are then checked with the target spacecraft circumscribing sphere, $\mathbb{S}_{\text{target}}$, which is used as a simple conservative approximation to the exact target geometry. For an illustration of the process, refer to Fig. 5.3.

Other Constraints Other constraints may easily be added, provided we augment our state vector $\mathbf{x}$ accordingly with the spacecraft attitude $\mathbf{q}$, mass m, arrival time t, *etc.* Solar array shadowing, pointing constraints, and so forth all fit within the framework, and may be represented as additional



(a) Thruster allocation without stuck-off failures.

(b) The same allocation problem, with both upper-right thrusters "stuck off."

Figure 5.4: Changes to torque-free control allocation in response to thruster failures. As can be seen, for the same net $\mathbf{\Delta v}$ vector (the large green arrow), thruster configuration changes can have a profound impact on thruster Δv locations and magnitudes (blue arrows), and hence on plume impingement satisfaction and thereby the safety of proposed $\mathbf{\Delta v}$ trajectories.

inequality or equality constraints. For more, we refer the interested reader to Section 1.1.3.

5.1.6 Active Safety

An additional feature we include in our work is the concept of *active safety*, in which we require the target spacecraft to maintain a feasible Collision Avoidance Maneuver (CAM) to a safe higher or lower circular orbit from every point along its solution trajectory in the event that any mission-threatening control degradations take place, such as stuck-off thrusters (as in Fig. 5.4). Addressing this will be the main subject of Chapter 6.

5.2 Conclusion

As can be seen, the propellant-optimal spacecraft motion planning problem is significantly more complicated than that of path planning, combining a mixture of non-convex avoidance constraints with control constraints, mixed state-controls-time constraints, and even logical safety constraints. The purpose of this chapter was to introduce the flavor of these difficulties for the specific case of CWH guidance, while simultaneously providing details on the constraints imposed during the numerical experiments of Chapter 8. For additional discussion on other types of proximity operations constraints, please see Section 1.1.3.

Chapter 6

Vehicle Safety

In this chapter, we devise a general strategy for handling the active safety constraints introduced in Eq. (5.1) and Section 5.1.6, which we use to guarantee solution safety under potential control failures. Specifically, we examine how to ensure that safe abort trajectories are always available to the spacecraft up to a given number of thruster "stuck-off" failures, without compromising real-time guidance. As will be motivated, the idea behind our approach is to couple positively-invariant set safety constraints (introduced in Section 6.1) with escape trajectory generation (described generally in Section 6.2 and specifically for CWH dynamics in Section 6.3), and embed them into the sampling routines of deterministic sampling-based motion planners. We prioritize *active safety* measures in this chapter (which allow actuated Collision Avoidance Maneuvers or CAMs) over passive safety guarantees (which shut off all thrusters and restrict the system to zero control) in order to broaden the search space for abort trajectories. Due to the propellant-limited nature of many spacecraft proximity operations missions, emphasis is placed on finding minimum-Δv escape maneuvers in order to improve mission reattempt opportunities. In many ways, we emulate the rendezvous design process taken by Barbee et al. [145], but numerically optimize abort propellant consumption and remove much of its reliance on user intuition by automating the satisfaction of safety constraints.

Consistent with the notions proposed by Schouwenaars et al. [146], Fehse [10, Ch. 4.1.2], and Fraichard [147], our general definition for vehicle safety is taken to be the following:

Definition 14 (Vehicle Safety)**.** A vehicle state is *safe* if and only if there exists, under the worst-possible environment and failure conditions, a collision-free, dynamically-feasible trajectory satisfying the constraints that navigates the vehicle to a set of states in which it can remain indefinitely.

Note *indefinitely* (or sufficiently-long for all practical purposes under the accuracy of the dynamics model) is a critical component of the definition. Trajectories without infinite-horizon safety guarantees can ultimately violate constraints [45], thereby posing a risk that can defeat the purpose of using a hard constraint in the first place. For this reason, we impose safety constraints over an infinite-horizon

(or, as we will show using invariant sets, an *effectively* infinite horizon).

Consider the scenario described in Chapter 5 for a spacecraft with nominal state trajectory $\mathbf{x}(t) \in \mathcal{X}$ and control trajectory $\mathbf{u}(t) \in \mathcal{U}(\mathbf{x}(t))$ evolving over time t in time span $\mathcal{T} = [t_{\text{init}}, \infty)$. Let $\mathcal{T}_{\text{fail}} \subseteq \mathcal{T}$ represent the subset of potential failure times we wish to certify (for instance, a set of prescribed burn times $\{\tau_i\}$, the final approach phase $\mathcal{T}_{\text{approach}}$, or the entire maneuver span $\mathcal{T}$). When a failure occurs, control authority is lost through a reduction in actuator functionality, negatively impacting system controllability. Let $\mathcal{U}_{\text{fail}}(\mathbf{x}) \subset \mathcal{U}(\mathbf{x})$ represent the new control set, where we assume that $\mathbf{0} \in \mathcal{U}_{\text{fail}}$ for all $\mathbf{x}$ (*i.e.*, we assume that no actuation is always a feasible control option). Mission safety is commonly imposed in two different ways [10, Ch. 4.4]:

- *Passive Safety*: For all $t_{\text{fail}} \in \mathcal{T}_{\text{fail}}$, ensure that $\mathbf{x}_{\text{CAM}}(t)$ satisfies Definition 14 with $\mathbf{u}_{\text{CAM}}(t) = \mathbf{0}$ for all $t \geq t_{\text{fail}}$. For spacecraft, this means its coasting arc from the point of failure must be safe for all future time (though practically this is imposed only over a finite horizon).
- *Active Safety*: For all $t_{\text{fail}} \in \mathcal{T}_{\text{fail}}$ and failure modes $\mathcal{U}_{\text{fail}}$, design actuated collision avoidance maneuvers $\mathbf{x}_{\text{CAM}}(t)$ to satisfy Definition 14 with $\mathbf{u}_{\text{CAM}}(t) \in \mathcal{U}_{\text{fail}}$ for all $t \geq t_{\text{fail}}$, where $\mathbf{u}_{\text{CAM}}(t)$ is not necessarily restricted to $\mathbf{0}$.

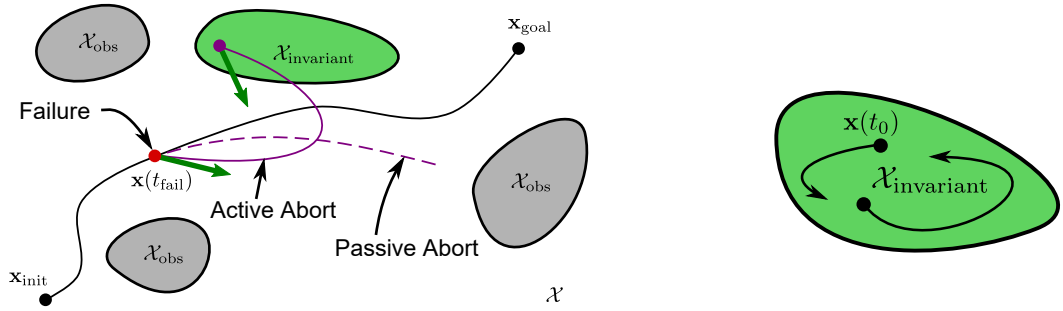
See Fig. 6.1a for an illustration. In much of the literature, only passive safety is considered out of a need for tractability (to avoid verification over a combinatorial explosion of failure mode possibilities), and in order to capture the common failure mode in which control authority is lost completely. Though considerably simpler to implement, this approach potentially neglects many mission-saving control policies. Our goal will be to provide active safety guarantees without hindering significantly the online planning and execution of vehicle guidance trajectories.

6.1 Active Safety using Positively-Invariant Sets

Instead of evaluating trajectory safety for all future times $t \geq t_{\text{fail}}$, it is more practical to consider finite-time solutions starting at $\mathbf{x}(t_{\text{fail}})$ that terminate at a point inside a safe *positively-invariant set* $\mathcal{X}_{\text{invariant}}$. If the abort maneuver is safe and the invariant set is safe for all time, then safety of the spacecraft is assured.

Definition 15 (Positively Invariant Set)**.** A set $\mathcal{X}_{\text{invariant}}$ is *positively invariant* with respect to the autonomous system $\dot{\mathbf{x}}_{\text{CAM}} = f(\mathbf{x}_{\text{CAM}})$ if and only if $\mathbf{x}_{\text{CAM}}(t_{\text{fail}}) \in \mathcal{X}_{\text{invariant}}$ implies $\mathbf{x}_{\text{CAM}}(t) \in \mathcal{X}_{\text{invariant}}$ for all $t \geq t_{\text{fail}}$.

Refer to Fig. 6.1b for visualization. Note the definition of a positively-invariant set assumes an autonomous system (a system that is uncontrolled, *i.e.*, left to propagate on its own without external interference). To extend this definition to controlled systems with dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, we simply let $\mathbf{u}$ be determined by a state-and-time-dependent control policy $\mathbf{u} = \Pi(\mathbf{x}(t), t)$. Naturally, this



(a) Comparing passive abort safety and active abort safety following the occurrence of a failure

(b) A positively-invariant set, within which state trajectories remain confined once entered

Figure 6.1: Illustrations of various vehicle safety concepts. Passive abort ensures the finite-time coasting trajectory from the point of failure is safe, while active safety reduces conservatism by allowing actuated escape maneuvers away from danger. Actively-safe maneuvers can enable infinite-horizon safety in a finite time by terminating at a safe and stable positively-invariant set.

extension includes unforced trajectories ($\mathbf{u} = 0$), constant-force trajectories ($\mathbf{u} = \text{constant}$), and purely time-dependent ("open-loop") control trajectories ($\mathbf{u}(t)$).

Under this generalization, there are many known instances of positively-invariant sets within spacecraft dynamics (see also the accompanying Fig. 6.2):

- **Equilibrium points**, $\mathbf{x}_{\text{eq}}$
  ($\mathcal{X}_{\text{invariant}} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} = \mathbf{x}_{\text{eq}}\}$)

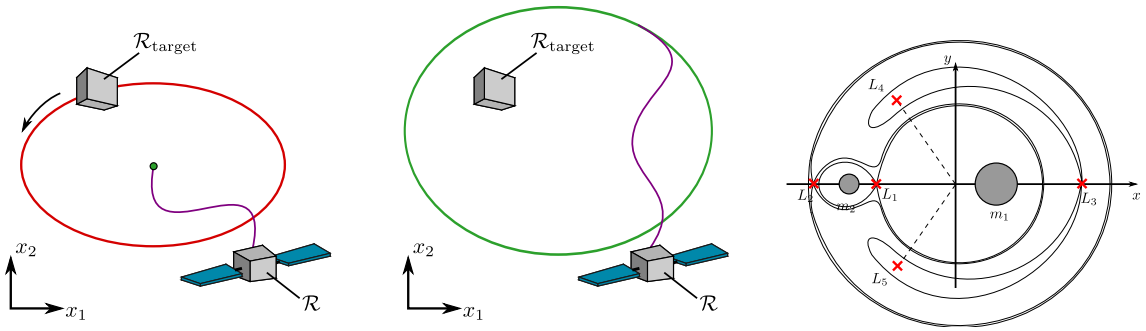
  Examples include the origins of many linearized dynamical frames of reference (since most linearizations are taken about stable equilibrium points, including Clohessy-Wiltshire-Hill dynamics—see Appendix A.1), Lagrange points (under Restricted 3-Body Problem dynamics), and hover points (dynamics-independent, though this often requires a non-zero, time-varying control trajectory).

- **Periodic solutions**, of period $\mathcal{T}_p \in \mathbb{R}_{++}$
  $\left(\mathcal{X}_{\text{invariant}} = \left\{\bigcup_{\tau \in [t,\, t+\mathcal{T}_p]} \mathbf{x}(\tau) \in \mathcal{X} \;\middle|\; \mathbf{x}(t+\mathcal{T}_p) = \mathbf{x}(t)\right\}\right)$

  Unforced examples include elliptic orbits (under Restricted 2-Body Problem dynamics), and Lagrangian (planar) and Halo (3D) orbits (under Restricted 3-Body Problem dynamics).

- **Regions of bounded relative mechanical energy**, designated as $\mathcal{E}$, with bounds $\mathcal{E}_{\min}, \mathcal{E}_{\max}$
  ($\mathcal{X}_{\text{invariant}} = \{\mathbf{x} \in \mathcal{X} \mid \mathcal{E}_{\min} \leq \mathcal{E}(\mathbf{x}) \leq \mathcal{E}_{\max}\}$)

  These are essentially the intersections of sublevel and superlevel sets of relative mechanical energy, taken from the system dynamics' frame of reference. Examples of relative mechanical energy include the total (inertial) mechanical energy for the Restricted 2-Body problem, or the



(a) Equilibrium points (forced, or otherwise)

(b) Orbits (periodic solutions of the dynamics)

(c) Regions of bounded relative mechanical energy (*e.g.*, between Jacobi constant contours)

Figure 6.2: Examples of positively-invariant sets in spacecraft dynamics. These illustrate a few of the types of state-space regions that may be used for terminating active safety maneuvers.

Jacobi constant for the Restricted 3-Body problem. Note that the constants $\mathcal{E}_{\text{min}}$ and $\mathcal{E}_{\text{max}}$ must be small and large enough, respectively, to account for the minimum- and maximum-possible relative mechanical energies of the spacecraft under control policy $\mathbf{u} = \Pi(\mathbf{x}(t), t)$ from all states in the region—this ensures our system can never escape from the set.

Of course, other examples are possible; this list is far from exhaustive. Furthermore, besides sets which satisfy the strict definition of positive invariance, there may also be many dynamic solutions that are approximately positively-invariant. For instance, quasi-periodic solutions are commonplace in spacecraft dynamics (*e.g.*, Lissajous orbits under the n-body problem) [148, 149]; their state trajectories are not strictly periodic but are functions of multiple frequencies and so behave like periodic solutions that precess or oscillate over time. The regions traced out by such trajectories, when bounded (*e.g.*, space-filling curves), may also be useful as positively-invariant sets.

Utilizing these invariant regions to terminate active abort trajectories, we obtain the following definition for finite-time verification of trajectory safety:

Definition 16 (Finite-Time Trajectory Safety Verification)**.** For all $t_{\text{fail}} \in \mathcal{T}_{\text{fail}}$ and for all $\mathcal{U}_{\text{fail}}(\mathbf{x}(t_{\text{fail}})) \subset \mathcal{U}(\mathbf{x}(t_{\text{fail}}))$, there exists $\{\mathbf{u}(t), t \geq t_{\text{fail}}\} \in \mathcal{U}_{\text{fail}}(\mathbf{x}(t_{\text{fail}}))$ and $T_h > t_{\text{fail}}$ such that $\mathbf{x}(t)$ is feasible for all $t_{\text{fail}} \leq t \leq T_h$ and $\mathbf{x}(T_h) \in \mathcal{X}_{\text{invariant}} \subseteq \mathcal{X}_{\text{free}}$,

where T_h is some finite safety horizon time. Though in principle any *safe* positively-invariant set $\mathcal{X}_{\text{invariant}}$ is acceptable, not just any will do in practice; in real-world scenarios, unstable trajectories caused by model uncertainties could cause state divergence towards configurations whose safety has not been verified. Hence care must be taken to use only *stable* positively-invariant sets. One way to determine and assess the stability of these positively-invariant sets is the well-known LaSalle Invariant Set Theorem [150]. Other tools for deriving and studying the properties of positively-invariant

sets include Poincaré maps [151, Ch. 7.3], Floquet theory [152], the Theory of Oscillations, and Kolmogorov-Arnold-Moser (KAM) theory [148].

Combining Definition 16 with our constraints in Eq. (5.1) from Chapter 5, spacecraft trajectory safety after a failure at $\mathbf{x}(t_{\text{fail}}) = \mathbf{x}_{\text{fail}}$ can be expressed in its full generality as the following optimization problem in decision variables $T_h \in [t_{\text{fail}}, \infty)$, $\mathbf{x}_{\text{CAM}}(t)$, and $\mathbf{u}_{\text{CAM}}(t)$, for $t \in [t_{\text{fail}}, T_h]$:

$$
\begin{array}{lll}
\text{Given:} & \text{Failure state } \mathbf{x}_{\text{fail}}(t_{\text{fail}}), \text{failure control set } \mathcal{U}_{\text{fail}}(\mathbf{x}_{\text{fail}}), \text{the free space } \mathcal{X}_{\text{free}}, & \\
 & \text{a safe, stable invariant set } \mathcal{X}_{\text{invariant}}, \text{and a fixed number of impulses } N & \\
\underset{\substack{\mathbf{u}_{\text{CAM}}(t)\in\mathcal{U}_{\text{fail}}(\mathbf{x}_{\text{fail}}),\\ T_h,\mathbf{x}_{\text{CAM}}(t)}}{\text{minimize}} & J(\mathbf{x}_{\text{CAM}}(t), \mathbf{u}_{\text{CAM}}(t), t) = \displaystyle\int_{t_{\text{fail}}}^{T_h} \|\mathbf{u}_{\text{CAM}}(t)\|_2 \, \mathrm{d}t = \sum_{i=1}^{N} \|\boldsymbol{\Delta}\mathbf{v}_{\text{CAM},i}\|_2 & \\
\text{subject to} & \dot{\mathbf{x}}_{\text{CAM}}(t) = f(\mathbf{x}_{\text{CAM}}(t), \mathbf{u}_{\text{CAM}}(t), t) & \text{System Dynamics} \\
 & \mathbf{x}_{\text{CAM}}(t_{\text{fail}}) = \mathbf{x}_{\text{fail}} & \text{Initial Condition} \\
 & \mathbf{x}_{\text{CAM}}(T_h) \in \mathcal{X}_{\text{invariant}} & \text{Safe Termination} \\
 & \mathbf{x}_{\text{CAM}}(t) \in \mathcal{X}_{\text{free}} \quad \text{for all } t \in [t_{\text{fail}}, T_h] & \text{Obstacle Avoidance} \\
 & \mathbf{g}(\mathbf{x}_{\text{CAM}}, \mathbf{u}_{\text{CAM}}, t) \leq 0 \quad \text{for all } t \in [t_{\text{fail}}, T_h] & \text{Other Constraints} \\
 & \mathbf{h}(\mathbf{x}_{\text{CAM}}, \mathbf{u}_{\text{CAM}}, t) = 0 &
\end{array}
\tag{6.1}
$$

This is identical to Eq. (5.1), except that now under failure mode $\mathcal{U}_{\text{fail}}(\mathbf{x}_{\text{fail}})$ we abandon the attempt to terminate at a goal state in $\mathcal{X}_{\text{goal}}$ and instead replace it with a constraint to terminate at a safe, stable positively-invariant set $\mathcal{X}_{\text{invariant}}$. We additionally neglect any timing constraints encoded in $\mathbf{g}$ as we are no longer concerned with our original rendezvous. Typically any feasible solution is sought following a failure, in which case one may use $J = 1$. However, to enhance the possibility of mission recovery, we assume the same minimum-propellant cost functional as before, but with the exception that here, as we will motivate, we use a single-burn strategy with $N = 1$.

6.2 Fault-Tolerant Safety Strategy

The difficulty of solving the finite-time trajectory safety problem lies in the fact that a feasible solution must be found for *all* possible failure times (typically assumed to be any time during the mission) as well as for *all* possible failures. To illustrate, for an F-fault tolerant spacecraft with K control components (thrusters, momentum wheels, CMGs, *etc.*) that we each model as either "operational" or "failed," this yields a total of $N_{\text{fail}} = \sum_{f=0}^{F} \binom{K}{f} = \sum_{f=0}^{F} \frac{K!}{(K-f)!f!}$ possible optimization problems that must be solved for every time t_{fail} along the nominal trajectory.[1] By any standard, this

[1] Each spacecraft configuration under f possible failures of K control components can be modeled as choosing f components out of K for failure, which we sum up to tolerance F.

is intractable, and hence explains why so often *passive* safety guarantees are selected (requiring only one control configuration check instead of N_{fail}, since we prescribe $\mathbf{u}_{\text{CAM}} = \mathbf{0}$, which must lie in $\mathcal{U}_{\text{fail}}$ given our assumption. This is analogous to setting $f = K$ with $F \triangleq K$). One idea for simplifying the problem while still satisfying safety (the constraints of Eq. (6.1)) consists of the following strategy:

Definition 17 (Fault-Tolerant Active Safety Strategy)**.** As a conservative solution to the optimization problem in Eq. (6.1), it is sufficient (but not necessary) to implement the following procedure:

1. From each $\mathbf{x}(t_{\text{fail}})$, prescribe a Collision-Avoidance Maneuver (CAM) policy Π_{CAM} that gives a horizon time T_h and escape control sequence $\mathbf{u}_{\text{CAM}} = \Pi_{\text{CAM}}(\mathbf{x}(t_{\text{fail}}))$ designed to automatically satisfy $\mathbf{u}_{\text{CAM}}(\tau) \subset \mathcal{U}$ for all $t_{\text{fail}} \leq \tau \leq T_h$ and $\mathbf{x}(T_h) \in \mathcal{X}_{\text{invariant}}$.

2. For each failure mode $\mathcal{U}_{\text{fail}}(\mathbf{x}(t_{\text{fail}})) \subset \mathcal{U}(\mathbf{x}(t_{\text{fail}}))$ up to tolerance F, determine if the control law is feasible; that is, see if $\mathbf{u}_{\text{CAM}} = \Pi_{\text{CAM}}(\mathbf{x}(t_{\text{fail}})) \subset \mathcal{U}_{\text{fail}}$ for the particular failure in question.

This effectively removes decision variables $\mathbf{u}_{\text{CAM}}$ from Eq. (6.1), allowing simple numerical integration for the satisfaction of the dynamic constraints and a straightforward *a posteriori* verification of the other trajectory constraints (inclusion in $\mathcal{X}_{\text{free}}$, and satisfaction of constraints $\mathbf{g}$ and $\mathbf{h}$). This checks if the prescribed CAM, guaranteed to provide a safe escape route, can actually be accomplished in the given failure situation. The approach is conservative due to the fact that the control law is imposed and not derived; however, the advantage is a greatly simplified optimal control problem with difficult-to-handle constraints relegated to *a posteriori* checks—exactly identical to the way that steering trajectories are derived and verified during the planning process of sampling-based planning algorithms. Note that formal definitions of safety require that this be satisfied for all possible failure modes of the spacecraft; we do not avoid the combinatorial explosion of N_{fail}. However, each instance of problem Eq. (6.1) is greatly simplified, and with F typically at most 3, the problem remains tractable. The difficult part, then, lies in computing Π_{CAM}, but this can easily be generated offline. Hence, the strategy should work well for vehicles with difficult, non-convex objective functions and constraints, as is precisely the case for CWH proximity operations.

Note, it is always possible to reduce this approach to the more conservative definition of "passive safety" that has traditionally been seen in the literature by choosing some finite horizon T_h and setting $\mathbf{u}_{\text{CAM}} = \Pi_{\text{CAM}}(\mathbf{x}(t_{\text{fail}})) = \mathbf{0}$ for all potential failure times $t_{\text{fail}} \in \mathcal{T}_{\text{fail}}$.

6.3 Safety in CWH Dynamics

We now specialize these ideas to proximity operations under impulsive CWH dynamics. Because many missions require stringent avoidance (prior to final approach and docking phase, for example), it is quite common for a "Keep-Out Zone" (KOZ) $\mathcal{X}_{\text{KOZ}}$, typically ellipsoidal in shape, to be defined about the target in the CWH frame. See the Approach Ellipsoid (AE) and Keep-Out Sphere (KOS) of Fig. 1.5 for rendezvous with the International Space Station—an example of using two hierarchical

KOZs for different stages of the rendezvous process. Throughout its approach, the chaser must certify that it will not enter this KOZ under any circumstance up to a specified thruster fault tolerance F, where here faults imply zero-output ("stuck-off") thruster failures.. To impose active safety as described in Section 6.2, we must prescribe an escape policy Π_{CAM} for each point in the trajectory and every possible failure mode, and then check for control allocation feasibility. The design of Π_{CAM} depends primarily on the safe, stable positively-invariant set $\mathcal{X}_{\text{invariant}}$, from which the escape horizon time T_h and escape maneuver $\mathbf{u}(t)$ can be derived. We derive here a number of examples for *unforced* CWH dynamics, from which we will form our CAM policy. This exercise also serves to demonstrate how safe invariant sets can be determined for the special case of systems with analytically-expressible dynamics (including all LTI systems).

In-Plane Examples ($\mathbf{x} = [\delta x, \delta y, \delta \dot{x}, \delta \dot{y}]^{\mathrm{T}}$)

- **At rest on the in-track axis, including the origin**
  $\left(\mathcal{X}_{\text{invariant}} = \left\{\mathbf{x} \in \mathbb{R}^4 \;\middle|\; \delta x_0 = \delta \dot{x}_0 = \delta \dot{y}_0 = 0\right\}\right)$

  Proof of Invariance. Substitute $\mathbf{x}(t_0) \in \mathcal{X}_{\text{invariant}}$ into the CWH state transition equations Eq. (A.14). This gives $\delta x = \delta \dot{x} = \delta \dot{y} = 0$ and $\delta y = \delta y_0$; therefore $\mathbf{x}(t > t_0) = \mathbf{x}(t_0) \in \mathcal{X}_{\text{invariant}}$. □

- **Planar, circularized orbits**
  $\left(\mathcal{X}_{\text{invariant}} = \left\{\mathbf{x} \in \mathbb{R}^4 \;\middle|\; \delta \dot{x}_0 = 0, \delta \dot{y}_0 = -\tfrac{3}{2} n_{\text{ref}} \delta x_0\right\}\right)$

  Proof of Invariance. Substitute $\mathbf{x}(t_0) \in \mathcal{X}_{\text{invariant}}$ into the CWH state transition equations Eq. (A.14). This gives $\delta x = \delta x_0$, $\delta y = \delta y_0 - \frac{3}{2}\theta \delta x_0 = \delta y_0 + \delta \dot{y}_0 (t - t_0)$, $\delta \dot{x} = 0$, and $\delta \dot{y} = -\frac{3}{2} n_{\text{ref}} \delta x_0 = \delta \dot{y}_0$; therefore, $\mathbf{x}(t > t_0) = \mathbf{x}(t_0) + \delta \dot{y}_0 (t - t_0) \hat{\boldsymbol{\delta}}\hat{\mathbf{y}} \in \mathcal{X}_{\text{invariant}}$. □

  Corollary 18 (Planar Circularization)**.** We can show that these initial conditions represent a circular orbit by converting to polar coordinates; note that the LVLH frame rotates with the polar frame, and that $\mathbf{r}(t) = (r_{\text{ref}} + \delta x(t)) \hat{\boldsymbol{\delta}}\hat{\mathbf{x}} = (r_{\text{ref}} + \delta x_0) \hat{\boldsymbol{\delta}}\hat{\mathbf{x}}$. Now $\mathbf{v}(t) = \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{r}(t)) = \frac{\mathrm{d}}{\mathrm{d}t}(r_{\text{ref}} + \delta x(t)) \hat{\boldsymbol{\delta}}\hat{\mathbf{x}} + n_{\text{ref}} \hat{\boldsymbol{\delta}}\hat{\mathbf{z}} \times \mathbf{r}(t) = \delta \dot{x}(t) \hat{\boldsymbol{\delta}}\hat{\mathbf{x}} + n_{\text{ref}} (r_{\text{ref}} + \delta x(t)) \hat{\boldsymbol{\delta}}\hat{\mathbf{y}}$. Observing that $\hat{\boldsymbol{\delta}}\hat{\mathbf{x}} = \hat{\mathbf{r}}$ and $\hat{\boldsymbol{\delta}}\hat{\mathbf{y}} = \hat{\boldsymbol{\theta}}$, we obtain $r(t) = r_{\text{ref}} + \delta x_0$ and $\dot{r}(t) = \delta \dot{x}(t) = 0$, which is the equation for a circle.

  Hence we obtain the well-known result that the initial conditions $\mathbf{x}_0 = \left[\delta x_0, \delta y_0, 0, -\frac{3}{2} n_{\text{ref}} \delta x_0\right]$ represent a circular orbit. It follows then that to circularize any planar trajectory at some state $\mathbf{x} = [\delta x, \delta y, \delta \dot{x}, \delta \dot{y}]^{\mathrm{T}}$, we must apply the impulse vector:

$$\boldsymbol{\Delta \mathbf{v}}_{\text{circ}} = \begin{bmatrix} -\delta \dot{x} \\ -\frac{3}{2} n_{\text{ref}} \delta x - \delta \dot{y} \end{bmatrix}$$

Out-of-Plane Examples ($\mathbf{x} = [\delta z, \delta \dot{z}]^{\mathrm{T}}$)

- **Unforced out-of-plane motion**
  $\left(\mathcal{X}_{\text{invariant}} = \left\{\mathbf{x} \in \mathbb{R}^2 \;\middle|\; (n_{\text{ref}}\delta z_0)^2 + \delta \dot{z}_0^2 = \text{constant}\right\}\right)$

  Proof of Invariance. Choose any out-of-plane states δz and $\delta \dot{z}$ and let their initial specific kinetic energy $\mathrm{E}_0 \geq 0$ be equal to $(n_{\text{ref}}\delta z_0)^2 + \delta \dot{z}_0^2 \triangleq \mathrm{E}_0$. Substituting into their respective CWH state transition equations Eq. (A.14), we obtain:

$$\begin{aligned}
\mathrm{E} &= (n_{\text{ref}}\delta z)^2 + \delta \dot{z}^2 \\
&= n_{\text{ref}}^2 \left[\cos^2\theta \delta z_0^2 + \frac{2}{n_{\text{ref}}} \cos\theta \sin\theta \delta z_0 \delta \dot{z} + \frac{1}{n_{\text{ref}}^2} \sin^2\theta \delta \dot{z}_0^2\right] \\
&+ \left[n_{\text{ref}}^2 \sin^2\theta \delta z_0^2 - 2n_{\text{ref}} \sin\theta \cos\theta \delta z_0 \delta \dot{z} + \cos^2\theta \delta \dot{z}_0^2\right] \\
&= n_{\text{ref}}^2 \big(\cos^2\theta + \sin^2\theta\big)\delta z_0^2 + \big(\sin^2\theta + \cos^2\theta\big)\delta \dot{z}_0^2 \\
&= n_{\text{ref}}^2 \delta z_0^2 + \delta \dot{z}^2 = \mathrm{E}_0
\end{aligned}$$

  Therefore $(n_{\text{ref}}\delta z)^2 + \delta \dot{z}^2$ is an integral of out-of-plane motion. Its level sets therefore form invariant sets. □

  Remark 19. This reiterates the well-known fact that out-of-plane CWH dynamics act as an undamped harmonic oscillator with a "spring constant" equal to the target spacecraft mean anomaly n_{ref}. However, it is not often presented in this way.

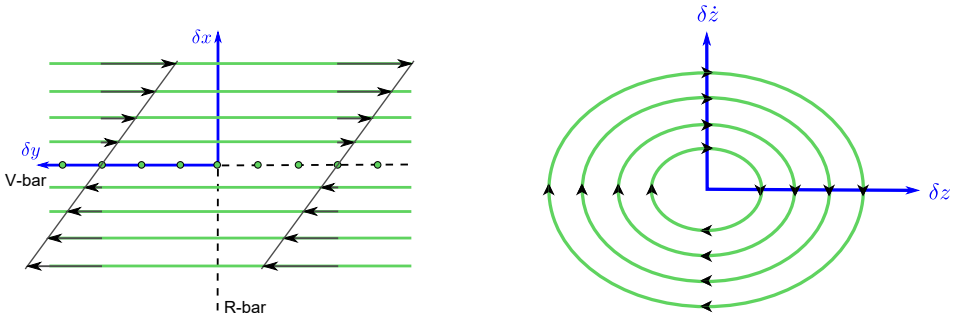
  Remark 20. As $\mathrm{E}_0 \to 0$, we approach the planar case $\delta z = \delta \dot{z} = 0$, a degenerate out-of-plane invariant set.

Full-State Examples ($\mathbf{x} = [\delta x, \delta y, \delta \dot{x}, \delta \dot{y}, \delta z, \delta \dot{z}]^{\mathrm{T}}$)

- **The product space of in-plane and out-of-plane invariant sets**
  $(\mathcal{X}_{\text{invariant}} = \mathcal{X}_{\text{invariant,in}} \times \mathcal{X}_{\text{invariant,out}})$

  Proof of Invariance. Let $[\delta x_0, \delta y_0, \delta \dot{x}_0, \delta \dot{y}_0]^{\mathrm{T}} \in \mathcal{X}_{\text{invariant,in}}$ and $[\delta z_0, \delta \dot{z}_0]^{\mathrm{T}} \in \mathcal{X}_{\text{invariant,out}}$. Then $\mathbf{x}(t_0) \in \mathcal{X}_{\text{invariant,in}} \times \mathcal{X}_{\text{invariant,out}}$. Substituting into the CWH state transition equations Eq. (A.14), the in-plane components δx_0, δy_0, $\delta \dot{x}_0$, and $\delta \dot{y}_0$ evolve independently of the out-of-plane components δz_0 and $\delta \dot{z}_0$, and vice versa, due to the decoupled nature of the transition equations. Hence each group of components remains in its respective invariant set; the in-plane components of $\mathbf{x}(t > t_0)$ remain in $\mathcal{X}_{\text{invariant,in}}$ and the out-of-plane components of $\mathbf{x}(t > t_0)$ remain in $\mathcal{X}_{\text{invariant,out}}$. Thus, $\mathbf{x}(t > t_0) \in \mathcal{X}_{\text{invariant}}$, as required. □

  Remark 21. The proof reveals that any combination of in-plane and out-of-plane invariant sets for CWH dynamics yields a full-state invariant set. Hence any in-plane invariant set (including



(a) In-plane invariant sets (at rest on the in-track axis, and circular orbits shown as horizontal lines)

(b) Out-of-plane invariant sets (any unforced motion, *i.e.*, undamped oscillations)

Figure 6.3: Examples of invariant sets under CWH dynamics. Full-state invariant sets can be made by combining (taking the product space of) these invariant set subspaces.

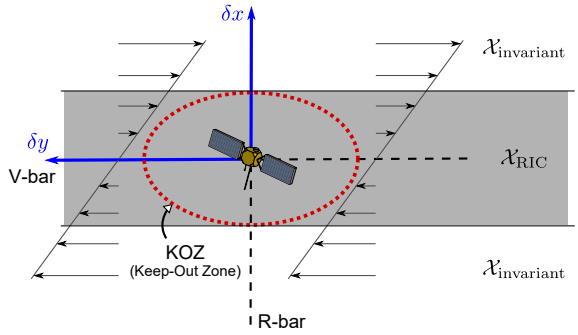
all examples presented above) extends to a full-state invariant set when considering out-of-plane motion as well, so long as the out-of-plane motion is unforced. In other words, any motions that are unforced out-of-plane and which *project* onto a planar invariant set, such as a circular orbit or an unmoving point on the in-track axis, are full-state invariant.

Illustrations of these invariant sets can be seen in Fig. 6.3. Note a more general, numerical method that uses Hamiltonians to identify invariant sets in CWH dynamics may be found in [153]. Once determined, controllers such as [154] can be used to enforce invariant manifold tracking. Throughout our implementation, due to imperfections in the CWH dynamics model, we implicitly assume that such a controller is available during the execution of a CAM.

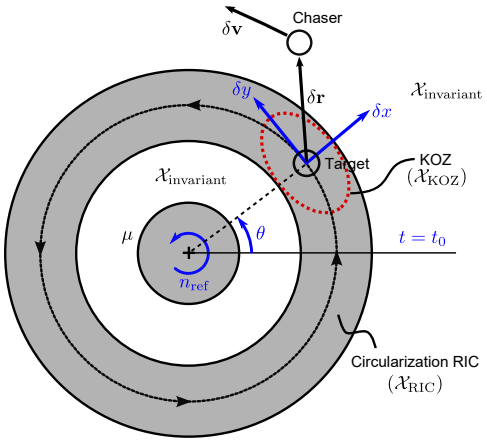
6.3.1 CAM Policy

We now have all the tools we need to formulate an active abort policy for spacecraft maneuvering under CWH dynamics. Recall from Definition 16 that for mission safety following a failure we are required to find a terminal state in an invariant set $\mathcal{X}_{\text{invariant}}$ entirely contained within the free state space $\mathcal{X}_{\text{free}}$. As will be motivated, we choose for $\mathcal{X}_{\text{invariant}}$ the set of circularized orbits whose planar projections lie outside of the radial band spanned by the KOZ. This is a subset of the product space of planar, circular orbits and unforced out-of-plane motion, and hence it is an invariant set (see Remark 21). Furthermore, because it lies outside of the KOZ, we can safely assume that $\mathcal{X}_{\text{invariant}} \subset \mathcal{X}_{\text{free}}$ (this can be verified or otherwise accounted for later on orbit, once other constraints defining $\mathcal{X}_{\text{free}}$ are known with certainty).

Now, the reasons we choose this particular set for abort termination are three-fold: circular orbits are (i) stable (assuming Keplerian motion, which is reasonable even under perturbations because the chaser and target are perturbed together and it is their *relative* state differences that matter),



(a) Safe circularization burn zones $\mathcal{X}_{\text{invariant}}$ for planar CWH dynamics. Any circularization attempts inside the zero-thrust Region of Inevitable Collision ($\mathcal{X}_{\text{ric}}$) will result in eventual penetration of the client KOZ, as indicated by the velocity arrows.



(b) Inertial view of the circularization RIC. Its complement shows the positions in $\mathcal{X}_{\text{invariant}}$ used for safe abort maneuver targeting.

Figure 6.4: Visualizing the safe and unsafe circularization regions used by the CAM safety policy

(ii) accessible (given the proximity of the chaser to the target's circular orbit), and (iii) passively safe (once reached, provided there is no intersection with the KOZ). As shown in Fig. 6.4, the set of orbital radii spanning the KOZ are excluded in order to prevent an eventual collision with the KOZ ellipsoid, either in the short-term or after nearly one full synodic period. In the event of an unrecoverable failure or an abort scenario taking longer than one synodic period to resolve, circularization within this region would jeopardize the target—a violation of Definition 14. Such a region is called a zero-thrust "Region of Inevitable Collision (RIC)," which we denote as $\mathcal{X}_{\text{ric}}$, as without additional intervention a collision with the KOZ is imminent and certain.

To summarize this mathematically,

$$\mathcal{X}_{\text{KOZ}} = \left\{ \mathbf{x} \;\middle|\; \mathbf{x}^{\mathrm{T}}\mathbf{E}\mathbf{x} \geq 1, \text{ where } \mathbf{E} = \operatorname{diag}\left(\rho_{\delta x}^{-2}, \rho_{\delta y}^{-2}, \rho_{\delta z}^{-2}, 0, 0, 0\right), \text{ with } \rho_i \text{ representing} \right. \tag{6.2}$$
$$\left. \text{the ellipsoidal KOZ semi-axis in the } i\text{-th LVLH frame axis direction.} \right\}$$

$$\mathcal{X}_{\text{ric}} = \left\{ \mathbf{x} \;\middle|\; |\delta x| < \rho_{\delta x},\, \delta\dot{x} = 0,\, \delta\dot{y} = -\frac{3}{2} n_{\text{ref}} \delta x \right\} \supset \mathcal{X}_{\text{KOZ}} \tag{6.3}$$

$$\mathcal{X}_{\text{invariant}} = \left\{ \mathbf{x} \;\middle|\; |\delta x| \geq \rho_{\delta x},\, \delta\dot{x} = 0,\, \delta\dot{y} = -\frac{3}{2} n_{\text{ref}} \delta x \right\} = \mathcal{X}_{\text{ric}}^{\mathsf{c}} \tag{6.4}$$

In short, our CAM policy to safely escape from a state $\mathbf{x}$ at which the spacecraft arrives (possibly under failures) at time t_{fail}, as visualized in Fig. 6.5, consists of the following:

1. Coast from $\mathbf{x}(t_{\text{fail}})$ to some new $T_h > t_{\text{fail}}$ such that $\mathbf{x}_{\text{CAM}}(T_h^-)$ lies at a position in $\mathcal{X}_{\text{invariant}}$.

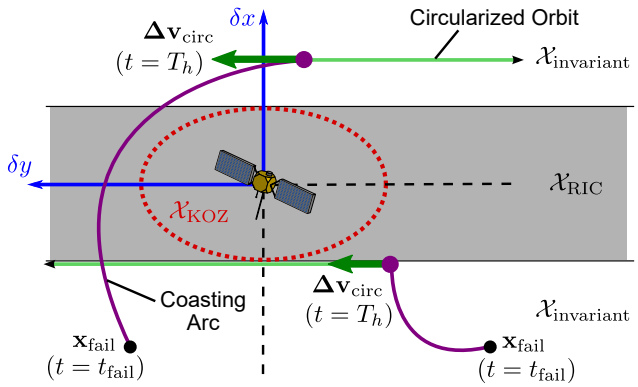


Figure 6.5: Examples of safe abort CAMs $\mathbf{x}_{\text{CAM}}$ following failures. The chaser coasts until firing a single circularization burn $\mathbf{\Delta v}_{\text{circ}}$ at horizon time T_h, a maneuver which is most propellant-efficient either at apogee or at the boundary of the circularization RIC (see Appendix B for details).

2. Circularize the (in-plane) orbit at $\mathbf{x}_{\text{CAM}}(T_h)$ such that $\mathbf{x}_{\text{CAM}}(T_h^+) \in \mathcal{X}_{\text{invariant}}$.
3. Coast along the new orbit (horizontal drift along the in-track axis in the CWH relative frame) in $\mathcal{X}_{\text{invariant}}$ until allowed to continue the mission (*e.g.*, after approval from ground operators).

6.3.2 Optimal CAM Circularization

In the event of a thruster failure at state $\mathbf{x}(t_{\text{fail}})$ that requires an emergency CAM, the time $T_h > t_{\text{fail}}$ at which to attempt a circularization maneuver becomes a degree of freedom. As we intend to maximize the recovery chances of the chaser after a failure, we choose T_h so as to minimize the cost of the circularization burn $\mathbf{\Delta v}_{\text{circ}}$, whose magnitude we denote as Δv_{circ}. Details on this problem, which can be solved analytically, may be found in Appendix B.

6.3.3 CAM Policy Feasibility

Once the circularization time T_h is determined, feasibility of the escape trajectory under every possible failure configuration at $\mathbf{x}(t_{\text{fail}})$ must be assessed in order to declare a particular CAM as actively-safe. To show this, the constraints of Eq. (6.1) must be evaluated under every combination of "stuck-off" thrusters (up to fault tolerance F), *with the exception of KOZ avoidance* as this is embedded into the CAM design process. How quickly this may be done depends on how many of these constraints may be considered "static" (unchanging, *i.e.*, independent of t_{fail}, *in the LVLH frame of reference*) or time-varying (otherwise).

Fortunately, most practical mission constraints are static (*i.e.*, imposed in advance by mission planners), allowing CAM trajectory feasibility verification to be moved offline. For example, considering our particular constraints in Section 5.1.5, if we can assume that the target remains enclosed within its KOZ near the origin and that it maintains a fixed attitude profile in the LVLH frame, then

obstacle and antenna lobe avoidance constraints become time-invariant (independent of the arrival time t_{fail}). If we further assume the attitude $\mathbf{q}(t)$ of the chaser is specified as a function of $\mathbf{x}(t)$, then control allocation feasibility and plume impingement constraints become verifiable offline as well. Better still, because of their time-independence, we need only evaluate the safety of arriving at each failure state $\mathbf{x}_{\text{fail}}$ once; this means the *active* safety of a particular state $\mathbf{x}$ can be cached—a highly useful property for reducing online computation.

Some constraints, on the other hand, cannot be defined *a priori*. These must be evaluated online, once the time t_{fail} and current environment are known. Due to the combinatorial explosion of thruster failure combinations, these constraints must either be very few or very simple to evaluate, or should otherwise be conservatively-approximated by equivalent static constraints. Alternatively, in lieu of this possibility, we may cache active safety evaluations with respect to any static constraints, per the discussion above, and leave the time-varying constraints for evaluation later once a nominal guidance plan has been completely determined. This "lazy-evaluation" approach saves significant computational effort during state space exploration, and simply requires continued planning whenever the nominal guidance plan is not cleared as actively-safe.

These are just a few useful strategies for testing active safety while maintaining real-time online guidance. As we will see in the next chapter, caching CAM trajectory evaluation with respect to static constraints, in particular, will be key to our solution approach.

6.4 Conclusion

As described, guaranteeing active safety in the face of sudden losses of control authority is a computationally-difficult task for vehicle guidance, due to a combinatorial explosion in the number of abort trajectories that must be determined over a continuum of failure states along our nominal guidance solution. In this chapter, we provided a sufficient method for handling this problem which prescribes an abort maneuver policy and checks in an *a posteriori* fashion whether given abort maneuvers are feasible under all failures of interest. To ensure infinite-horizon safety, these abort maneuvers must be terminated at positively-invariant sets, of which several were derived for the case of CWH dynamics. The proposed technique allows trajectory designer intuition to be embedded into the abort process, greatly reducing computational overhead without compromising vehicle autonomy. Furthermore, when mission constraints allow, the approach enables the active safety of potential failure states to be mostly precomputed—a fact we will exploit heavily in the design of our planning algorithm, as discussed in the next chapter.

Chapter 7

Real-Time Sampling-Based Spacecraft Proximity Operations

With the proximity operations scenario established, we are now in position to describe our approach. As previously described, the constraints that must be satisfied in Eq. (5.1) are diverse, complex, and difficult to satisfy numerically. In this chapter, we propose a guidance algorithm to solve this problem, followed by a detailed proof of its optimality with regard to the sum-of-2-norms propellant-cost metric J under impulsive CWH dynamics. As will be seen, the proof relies on an understanding of: (i) the steering connections between sampled points assuming no obstacles or other trajectory constraints, and (ii) the nearest-neighbors or *reachable* states from a given state. We hence start by characterizing these two concepts, in Sections 7.1 and 7.2 respectively. We then proceed to the algorithm presentation (Section 7.3) and its theoretical characterization (Section 7.4), before closing with a description of two smoothing techniques for rapidly reducing the costs of sampling-based solution trajectories for systems with impulsive actuation (Section 7.5).

7.1 State Interconnections: The Steering Problem

For sample-to-sample interconnections, we consider the *unconstrained* minimal-propellant 2-point boundary value problem (2PBVP) or "steering problem" between an initial state $\mathbf{x}_0$ and a final state $\mathbf{x}_\mathrm{f}$ within the CWH dynamics model. Solutions to these steering problems provide the local building blocks from which we construct solutions to the more complicated problem formulation in Eq. (5.1). Steering solutions serve two main purposes: (i) they represent a class of short-horizon controlled trajectories that are filtered online for constraint satisfaction and efficiently strung together into a state space spanning graph (*i.e.*, a tree or roadmap), and (ii) the costs of steering trajectories are used to inform the graph construction process by identifying the unconstrained "nearest neighbors"

as edge candidates. Because these problems can be expressed independently of the arrival time t_0 (as will be shown), our solution algorithm does not need to solve these problems *online*; the solutions between every pair of samples can be precomputed and stored prior to receiving a motion query. Hence the 2PBVP presented here need not be solved quickly. However, we mention techniques for speed-ups due to the reliance of our smoothing algorithm (Algorithm 7) on a fast solution method.

Substituting our boundary conditions into Eq. (5.4), evaluating at $t = t_\mathrm{f}$, and rearranging, we seek a stacked burn vector $\mathbf{\Delta V}$ such that:

$$\mathbf{\Phi}_v(t_\mathrm{f}, \{\tau_i\}_i)\mathbf{\Delta V} = \mathbf{x}_\mathrm{f} - \mathbf{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0, \tag{7.1}$$

for some number N of burn times $\tau_i \in [t_0,\, t_\mathrm{f}]$. Formulating this as an optimal control problem that minimizes our sum-of-2-norms cost functional (as a proxy for the actual propellant consumption, as described in Section 5.1.1), we wish to solve:

$$\begin{array}{lll}
\text{Given:} & \text{Initial state } \mathbf{x}_0, \text{final state } \mathbf{x}_\mathrm{f}, \text{burn magnitude bound } \Delta v_\mathrm{max}, & \\
& \text{and maneuver duration bound } T_\mathrm{max} & \\
\underset{\mathbf{\Delta v}_i, \tau_i, t_\mathrm{f}, N}{\text{minimize}} & \displaystyle\sum_{i=1}^{N} \|\mathbf{\Delta v}_i\|_2 & \\
\text{subject to} & \mathbf{\Phi}_v(t_\mathrm{f}, \{\tau_i\}_i)\mathbf{\Delta V} = \mathbf{x}_\mathrm{f} - \mathbf{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0 & \text{Dynamics/Boundary Conditions} \\
& 0 \le t_\mathrm{f} - t_0 \le T_\mathrm{max} & \text{Maneuver Duration Bounds} \\
& t_0 \le \tau_i \le t_\mathrm{f} \quad \text{for burns } i & \text{Burn Time Bounds} \\
& \|\mathbf{\Delta v}_i\|_2 \le \Delta v_\mathrm{max} \quad \text{for burns } i & \text{Burn Magnitude Bounds}
\end{array} \tag{7.2}$$

Notice that this is a relaxed version of the original problem presented as Eq. (5.1), with only its boundary conditions, dynamic constraints, and control norm bound. As it stands, due to the nonlinearity of the dynamics with respect to τ_i, t_f and N, Eq. (7.2) is non-convex and inherently difficult to solve. However, we can make the problem tractable if we make a few assumptions. Given that we plan to string many steering trajectories together to form our overall solution, let us ensure they represent the most primitive building blocks possible such that their concatenation will adequately represent any arbitrary trajectory. Set $N = 2$ (the smallest number of burns required to transfer between any pair of arbitrary states, as it makes $\mathbf{\Phi}_v(t_\mathrm{f}, \{\tau_i\}_i)$ square) and select burn times $\tau_1 = t_0$ and $\tau_2 = t_\mathrm{f}$ (which automatically satisfy our burn time bounds). This leaves $\mathbf{\Delta v}_1 \in \mathbb{R}^{d/2}$ (an intercept burn applied just after $\mathbf{x}_0$ at time t_0), $\mathbf{\Delta v}_2 \in \mathbb{R}^{d/2}$ (a rendezvous burn applied just before $\mathbf{x}_\mathrm{f}$ at time t_f), and t_f as our only remaining decision variables. The result is the scenario shown in Fig. 7.1. If we conduct a search for $t_\mathrm{f}^* \in [t_0,\, t_0 + T_\mathrm{max}]$, the relaxed-2PBVP can now be solved iteratively as a relatively simple bounded one-dimensional nonlinear minimization problem, where at each iteration one computes:

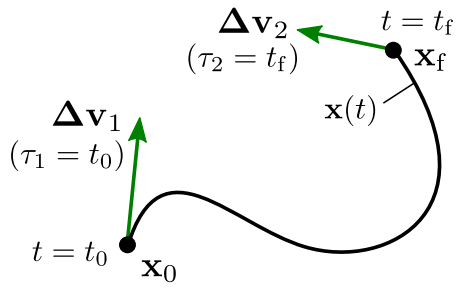


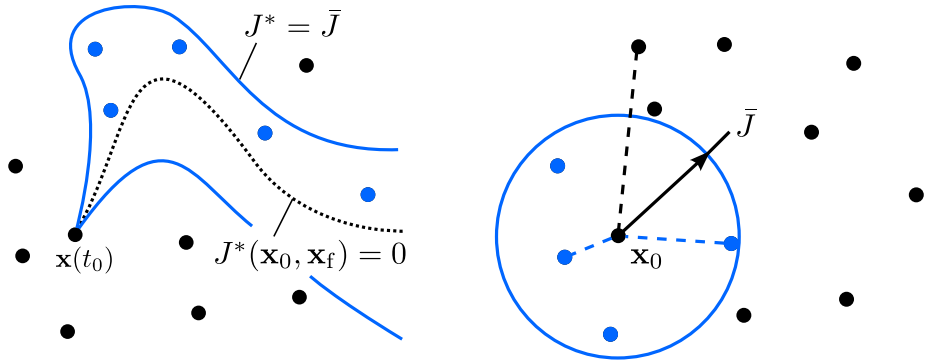
Figure 7.1: Visualizing state-to-state steering under impulsive CWH dynamics. The steering problem is solved as a one-dimensional nonlinear minimization problem for the optimal final burn time t_f^* (or equivalently, the optimal maneuver duration $T^* = t_\mathrm{f}^* - t_\mathrm{init}$).

$$\mathbf{\Delta V}(t_\mathrm{f}) = \mathbf{\Phi}_v^{-1}(t_\mathrm{f}, \{t_0, t_\mathrm{f}\})(\mathbf{x}_\mathrm{f} - \mathbf{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0),$$

where the argument t_f is shown for $\mathbf{\Delta V}$ to highlight its dependence. By uniqueness of the matrix inverse (provided $\mathbf{\Phi}_v^{-1}$ is non-singular, discussed below), we need only check that the resulting impulses $\mathbf{\Delta v}_i(t_\mathrm{f})$ satisfy the magnitude bound to declare the solution to an iteration feasible. Notice that because $\mathbf{\Phi}$ and $\mathbf{\Phi}_v^{-1}$ depend only on the difference between t_f and t_0, we can equivalently search over maneuver durations $T = t_\mathrm{f} - t_0 \in [0,\, T_\mathrm{max}]$ instead, solving the following relaxation of Eq. (7.2):

$$\begin{array}{lll}
\text{Given:} & \text{Initial state } \mathbf{x}_0, \text{final state } \mathbf{x}_\mathrm{f}, \text{burn magnitude bound } \Delta v_\mathrm{max}, & (7.3)\\
& \text{and maneuver duration bound } T_\mathrm{max} < \dfrac{2\pi}{n_\mathrm{ref}} & \\
\underset{T \in [0,\, T_\mathrm{max}]}{\text{minimize}} & \displaystyle\sum_{i=1}^{2} \|\mathbf{\Delta v}_i\|_2 & \\
\text{subject to} & \mathbf{\Delta V} = \mathbf{\Phi}_v^{-1}(T, \{0, T\})(\mathbf{x}_\mathrm{f} - \mathbf{\Phi}(T, 0)\mathbf{x}_0) & \text{Dynamics/Boundary Conditions}\\
& \|\mathbf{\Delta v}_i\|_2 \leq \Delta v_\mathrm{max} \quad \text{for burns } i & \text{Burn Magnitude Bounds}
\end{array}$$

This dependence on the maneuver duration T only (and not on the time t_0 at which we arrive at $\mathbf{x}_0$) turns out to be indispensable for precomputation, as it allows steering trajectories to be generated and stored *offline*. Observe, however, that our steering solution $\mathbf{\Delta V}^*$ requires $\mathbf{\Phi}_v$ to be invertible, *i.e.*, that $(t_\mathrm{f} - \tau_1) - (t_\mathrm{f} - \tau_2) = t_\mathrm{f} - t_0 = T$ avoids singular values (including zero, orbital period multiples, and other values longer than one period [155])—we ensure this by enforcing T_max to be shorter than one period. To handle the remaining case of $T = 0$, note a solution exists if and only if $\mathbf{x}_0$ and $\mathbf{x}_\mathrm{f}$ differ in velocity only; in such instances, we take the solution to be $\mathbf{\Delta v}_2^*$ set as this velocity difference (with $\mathbf{\Delta v}_1^* = \mathbf{0}$).



(a) Using a cost reachability set (general motion planning)

(b) Using a search radius (path planning)

Figure 7.2: Generalizing neighborhoods to optimal motion planning problems through cost reachability sets. Instead of using a search radius, neighbors to $\mathbf{x}_0$ are defined as all states $\mathbf{x}_\mathrm{f}$ whose steering cost from $\mathbf{x}_0$ to $\mathbf{x}_\mathrm{f}$ lies below the predefined threshold $\bar{J}$.

7.2 Neighborhoods: Cost Reachability Sets

Armed with a steering solution, we can now extend the path planning notion of state neighborhoods to general motion planning problems. For this, we consider the states near another state not in terms of Euclidean distance, but instead in terms of the cost-to-go. This idea is captured naturally by so-called *cost reachability sets*. In keeping with our steering solution Eq. (7.3), since $\mathbf{\Delta V}^*$ depends only on the trajectory endpoints $\mathbf{x}_\mathrm{f}$ and $\mathbf{x}_0$, we henceforth refer to the cost of a steering trajectory by the notation $J(\mathbf{x}_0, \mathbf{x}_\mathrm{f})$. We then define the forward reachability set from a given state $\mathbf{x}_0$ as follows:

Definition 22 (Forward Reachable Set)**.** The forward reachable set $\mathcal{R}$ from state $\mathbf{x}_0$ is the set of all states $\mathbf{x}_\mathrm{f}$ that can be reached from $\mathbf{x}_0$ with a cost $J(\mathbf{x}_0, \mathbf{x}_\mathrm{f})$ below a given cost threshold $\bar{J}$, *i.e.*,

$$\mathcal{R}(\mathbf{x}_0, \bar{J}) \triangleq \{\mathbf{x}_\mathrm{f} \in \mathcal{X} \mid J(\mathbf{x}_0, \mathbf{x}_\mathrm{f}) < \bar{J}\}.$$

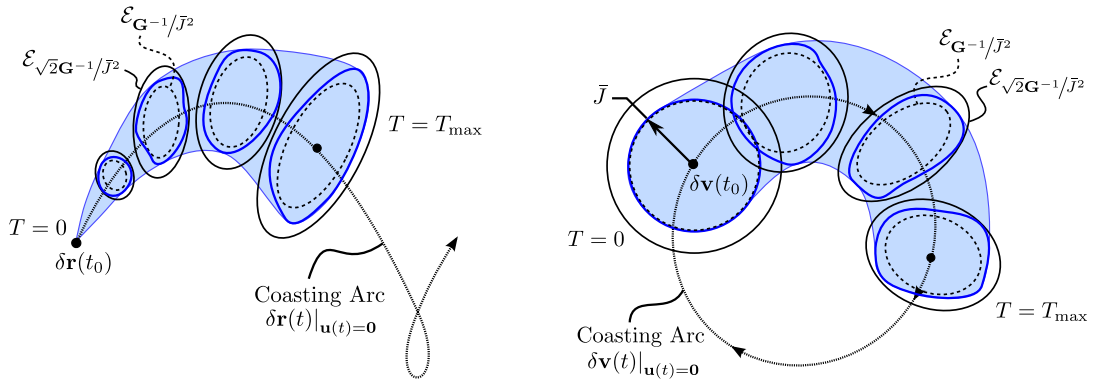
See Fig. 7.2 for visualization. Recall from Eq. (7.3) in Section 7.1 that the steering cost may be written as:

$$J(\mathbf{x}_0, \mathbf{x}_\mathrm{f}) = \|\mathbf{\Delta v}_1\| + \|\mathbf{\Delta v}_2\| = \|\mathbf{S}_1 \mathbf{\Delta V}\| + \|\mathbf{S}_2 \mathbf{\Delta V}\| \tag{7.4}$$

where $\mathbf{S}_1 = [\,\mathbf{I}_{d/2\times d/2}, \mathbf{0}_{d/2\times d/2}\,]$, $\mathbf{S}_2 = [\,\mathbf{0}_{d/2\times d/2}, \mathbf{I}_{d/2\times d/2}\,]$, and $\mathbf{\Delta V}$ is given by:

$$\mathbf{\Delta V}(\mathbf{x}_0, \mathbf{x}_\mathrm{f}) = \begin{bmatrix} \mathbf{\Delta v}_1 \\ \mathbf{\Delta v}_2 \end{bmatrix} = \mathbf{\Phi}_v^{-1}(t_\mathrm{f}, \{t_0, t_\mathrm{f}\})(\mathbf{x}_\mathrm{f} - \mathbf{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0).$$

The cost function $J(\mathbf{x}_0, \mathbf{x}_\mathrm{f})$ is difficult to gain insight on directly; however, as we shall see, we can



(a) The set of reachable positions $\delta\mathbf{r}_\mathrm{f}$ within duration $T_{\max}$ and propellant cost $\bar{J}$.

(b) The set of reachable velocities $\delta\mathbf{v}_\mathrm{f}$ within duration $T_{\max}$ and propellant cost $\bar{J}$.

Figure 7.3: Bounds on reachability sets from initial state $\mathbf{x}(t_0) = [\delta\mathbf{r}(t_0), \delta\mathbf{v}(t_0)]$ under propellant cost threshold $\bar{J}$. For cost measured as a sum of $\boldsymbol{\Delta}\mathbf{v}$ 2-norms, these are bounded by unions of ellipsoidal balls in position-velocity (phase) space over the set of all permissible maneuver durations $T \in [0, T_{\max}]$. For the special case of $T = 0$, the reachable set of states is a 3-dimensional velocity ball embedded in $\mathbb{R}^6$ (with volume 0) corresponding to states with position $\delta\mathbf{r}_\mathrm{f} = \delta\mathbf{r}(t_0)$ and velocity $\delta\mathbf{v}_\mathrm{f} \in \mathcal{B}(\delta\mathbf{v}(t_0), \Delta v_{\max})$.

work with its bounds much more easily.

Lemma 23 (Fuel Burn Cost Bounds)**.** *For the cost function in Eq.* (7.4)*, we have the following upper and lower bounds:*

$$\|\boldsymbol{\Delta}\mathbf{V}\| \leq J(\mathbf{x}_0, \mathbf{x}_\mathrm{f}) \leq \sqrt{2}\|\boldsymbol{\Delta}\mathbf{V}\|.$$

Proof. For the proof, see Appendix C. □

Now, observe that $\|\boldsymbol{\Delta}\mathbf{V}\| = \sqrt{(\mathbf{x}_\mathrm{f} - \boldsymbol{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0)^\mathrm{T}\mathbf{G}^{-1}(\mathbf{x}_\mathrm{f} - \boldsymbol{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0)}$, where $\mathbf{G}^{-1} = \boldsymbol{\Phi}_v^{-\mathrm{T}}\boldsymbol{\Phi}_v^{-1}$. This is the expression for an ellipsoid $\mathcal{E}(\mathbf{x}_\mathrm{f})$ resolved in the LVLH frame with matrix $\mathbf{G}^{-1}$ and center $\boldsymbol{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0$ (the state $T = t_\mathrm{f} - t_0$ time units ahead of $\mathbf{x}_0$ along its coasting arc). Combined with Lemma 23, we see that for a fixed maneuver time T and propellant cost threshold $\bar{J}$, the spacecraft at $\mathbf{x}_0$ can reach all states inside an area under-approximated by an ellipsoid with matrix $\mathbf{G}^{-1}/\bar{J}^2$ and over-approximated by an ellipsoid of matrix $\sqrt{2}\mathbf{G}^{-1}/\bar{J}^2$. The forward reachable set for impulsive CWH dynamics under our propellant-cost metric is therefore bounded by the union over all maneuver times of these under- and over-approximating ellipsoidal sets, respectively. To better visualize this, see Fig. 7.3 for a geometric interpretation.

7.3 Motion Planning: The Modified FMT* Algorithm

We now have all the tools we need to adapt sampling-based motion planning algorithms to optimal vehicle guidance under impulsive CWH dynamics, as represented by Eq. (5.1). Recall from Chapter 2

that sampling-based planning [68, 74, 77] essentially breaks down a continuous trajectory optimization problem into a series of relaxed, local steering problems (as in Section 7.1) between intermediate waypoints (called *samples*) before piecing them together to form a global solution to the original problem. This framework can yield significant computational benefits if: (i) the relaxed subproblems are simple enough, and (ii) the *a posteriori* evaluation of trajectory constraints is fast compared to a single solution of the full-scale problem. Furthermore, provided samples are sufficiently dense in the free state-space $\mathcal{X}_{\text{free}}$ and graph exploration is spatially-symmetric, sampling-based planners can closely approximate global optima without fear of convergence to local minima. Though many candidate planners could be used here, we rely on the asymptotically-optimal (AO) Fast Marching Trees (FMT*) and Bi-directional Fast Marching Trees (BFMT*) algorithms, originally presented in Section 3.2. These algorithms are chosen for their proven efficiency as well as their compatibility with *deterministic* batch sampling [119], a key benefit that leads to a number of algorithmic simplifications (including use of offline knowledge). Note, however, that because differences between both algorithms are mainly workspace-specific (see Chapter 4), we modify and implement only the FMT* algorithm in order to simplify the following exposition. Analogous modifications may be made to BFMT* as well, due to its straightforward inheritance from FMT*.

The FMT* algorithm, adapted from Algorithm 1 and tailored to our application, is reiterated here as Algorithm 6 (we shall henceforth refer to our modified version of FMT* as simply FMT*, for brevity). Like its path planning variant, our modified FMT* efficiently expands a tree of feasible trajectories from an initial state $\mathbf{x}_{\text{init}}$ to a goal state $\mathbf{x}_{\text{goal}}$ around nearby obstacles; unlike in path planning, "obstacles" now represent infeasible state regions that do not necessarily arise from physical obstacles in the workspace.

The algorithm begins by taking a set of samples distributed in the free state space $\mathcal{X}_{\text{free}}$ using the SampleFree routine, which restricts state sampling to actively-safe feasible states (which lie outside of $\mathcal{X}_{\text{obs}}$ and have access to a safe Collision Avoidance Maneuver (CAM) as described in Section 6.3). In our implementation, we assume samples are taken using the Halton sequence [99], though any deterministic, *low-discrepancy* sampling sequence may be used [119] (see Section 2.4.2). Selecting $\mathbf{x}_{\text{init}}$ first for further expansion as the minimum cost-to-come node $\mathbf{z}$, the algorithm then proceeds to look at reachable unvisited samples or "neighbors" (samples that can be reached with less than a given propellant cost threshold $\bar{J}$, as described in the previous section), and attempts to connect those with the cheapest cost-to-come back to the tree (using Steer). The cost threshold $\bar{J}$ is a free parameter whose value can have a significant effect on performance; see Theorem 28 for a theoretical characterization and Chapter 8 for a representative numerical trade study. Those trajectories satisfying the constraints of Eq. (5.1), as determined by CollisionFree, are saved. As feasible connections are made, the algorithm relies on adding and removing nodes (saved waypoint states) from three sets: a set of unexplored samples $\mathcal{V}_{\text{unvisited}}$ not yet connected to the tree, a frontier $\mathcal{V}_{\text{open}}$ of nodes likely to make efficient connections to unexplored neighbors, and an interior $\mathcal{V}_{\text{closed}}$ of

Algorithm 6 The Fast Marching Tree Algorithm (FMT*). Computes a minimal-cost trajectory from an initial state $\mathbf{x}(t_0) = \mathbf{x}_{\text{init}}$ to a target state $\mathbf{x}_{\text{goal}}$ through a fixed number n of samples $\mathcal{S}$.

1: Add $\mathbf{x}_{\text{init}}$ to the root of the tree $\mathcal{T}$, as a member of the frontier set $\mathcal{V}_{\text{open}}$
2: Generate samples $\mathcal{S} \leftarrow \textsc{SampleFree}(\mathcal{X}, n, t_0)$ and add them to the unexplored set $\mathcal{V}_{\text{unvisited}}$
3: Set the minimum cost-to-come node in the frontier set as $\mathbf{z} \leftarrow \mathbf{x}_{\text{init}}$
4: **while true**
5: **for** each neighbor $\mathbf{x}$ of $\mathbf{z}$ in $\mathcal{V}_{\text{unvisited}}$
6: Find the neighbor $\mathbf{x}_{\text{min}}$ in $\mathcal{V}_{\text{open}}$ of cheapest cost-to-go to $\mathbf{x}$
7: Compute the trajectory between them as $[\mathbf{x}(t), \mathbf{u}(t), t] \leftarrow \textsc{Steer}(\mathbf{x}_{\text{min}}, \mathbf{x})$ (*Section 7.1*)
8: **if** $\textsc{CollisionFree}(\mathbf{x}(t), \mathbf{u}(t), t)$
9: Add the trajectory from $\mathbf{x}_{\text{min}}$ to $\mathbf{x}$ to tree $\mathcal{T}$
10: Remove all $\mathbf{x}$ from the unexplored set $\mathcal{V}_{\text{unvisited}}$
11: Add any new connections $\mathbf{x}$ to the frontier $\mathcal{V}_{\text{open}}$
12: Remove $\mathbf{z}$ from the frontier $\mathcal{V}_{\text{open}}$ and add it to $\mathcal{V}_{\text{closed}}$
13: **if** $\mathcal{V}_{\text{open}}$ is empty
14: **return** Failure
15: Reassign $\mathbf{z}$ as the node in $\mathcal{V}_{\text{open}}$ with smallest cost-to-come from the root ($\mathbf{x}_{\text{init}}$)
16: **if** $\mathbf{z}$ is in the goal region $\mathcal{X}_{\text{goal}}$
17: **return** Success, and the unique trajectory from the root ($\mathbf{x}_{\text{init}}$) to $\mathbf{z}$

nodes that are no longer useful for exploring the state space $\mathcal{X}$. More details on FMT* and BFMT* can be found in their original works [80, 90] and in Sections 3.2.1–3.2.4.

To make FMT* and its variants amenable to a real-time implementation, we consider an online-offline approach that relegates as much computation as possible to a pre-processing phase. To be specific, we precompute the sample set $\mathcal{S}$ (line 2), nearest-neighbor sets (used in lines 5 and 6), and steering trajectory solutions (line 7), provided the planning problem satisfies the following conditions:

1. the state space $\mathcal{X}$ is known *a priori* (typical for most missions; note we do not impose this on the obstacle space $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$, which must generally be identified online using onboard sensors),

2. steering solutions are independent of sample arrival times t_0, as we show in Section 7.1.

Here Item 1 allows samples to be precomputed, while Item 2 enables steering trajectories to be stored onboard or uplinked from the ground up to the spacecraft, since their values remain relevant regardless of the times at which the spacecraft actually follows them during the mission. Once online, any samples invalidated from new or changing obstacle regions can simply be pruned from $\mathcal{S}$ before initiating the planning algorithm, or otherwise left to be omitted from consideration during collision checks. This leaves only collision-checking, graph construction and search, and termination checks as parts of the online phase, greatly improving the online run time and leaving the more intensive work to offline resources where running time is less important. This breakdown into online and offline components (inspired by [156]) is a valuable technique for imbuing kinodynamic motion planning

problems with real-time online solvability using fast batch-planners like FMT*.

7.4 Theoretical Characterization of FMT*

It remains to show that FMT* provides similar asymptotic optimality guarantees under the sum-of-2-norms propellant-cost metric and impulsive CWH dynamics (which enter into Algorithm 6 under lines 6–7), as has already been shown for kinematic (straight-line path planning) problems [80]. As a reminder to the reader, *asymptotic optimality* refers to the property that as the number of samples $n \to \infty$, the cost of the trajectory (a.k.a. "path") returned by the planner approaches that of the optimal cost (see Section 2.3.3). Here a proof is presented showing asymptotic optimality for the planning algorithm and problem setup used in this thesis. We note that while CWH dynamics are the primary focus of this work, the following proof methodology extends to any general linear system controlled by a finite sequence of impulsive actuations, whose fixed-duration 2-impulse steering problem is uniquely determined (*e.g.*, a wide array of second-order control systems).

The proof proceeds analogously to that in [80] and Section 3.3 by showing that it is always possible to construct an approximate path using points in $\mathcal{S}$ that closely follows the optimal path. Similarly to [80], we will make use here of the ℓ_2-*dispersion* of a set of points (a specialization of Eq. (2.2)), which upper bounds how far away a point in $\mathcal{X}$ can be from its nearest point in $\mathcal{S}$ as measured by the ℓ_2-norm (see Fig. 2.4a).

Definition 24 (ℓ_2-dispersion)**.** For a finite, non-empty set $\mathcal{S}$ of points in a d-dimensional compact Euclidean subspace $\mathcal{X}$ with positive Lebesgue measure, its ℓ_2-dispersion $D(\mathcal{S})$ is defined as:

$$\begin{aligned} D(\mathcal{S}) &\triangleq \sup_{\mathbf{x}\in\mathcal{X}} \min_{\mathbf{s}\in\mathcal{S}} \|\mathbf{s}-\mathbf{x}\| \\ &= \sup\{R>0 \mid \exists \mathbf{x}\in\mathcal{X} \text{ with } \mathcal{B}(\mathbf{x},R)\cap\mathcal{S}=\varnothing\}, \end{aligned}$$

where $\mathcal{B}(\mathbf{x}, R)$ is a Euclidean ball with radius R centered at state $\mathbf{x}$.

We also require a means for quantifying the deviation that small endpoint pertubations can bring about in the 2-impulse steering control. This result is necessary to ensure that the particular placement of the points of $\mathcal{S}$ is immaterial; only its low-dispersion property matters.

Lemma 25 (Steering with Perturbed Endpoints)**.** *For a given steering trajectory $\mathbf{x}(t)$ with initial time t_0 and final time t_f, let $\mathbf{x}_0 \coloneqq \mathbf{x}(t_0)$, $\mathbf{x}_\mathrm{f} \coloneqq \mathbf{x}(t_\mathrm{f})$, $T \coloneqq t_\mathrm{f} - t_0$, and $J \coloneqq J(\mathbf{x}_0, \mathbf{x}_\mathrm{f})$. Consider now the perturbed steering trajectory $\tilde{\mathbf{x}}(t)$ between perturbed start and end points $\tilde{\mathbf{x}}_0 = \mathbf{x}_0 + \delta\mathbf{x}_0$ and $\tilde{\mathbf{x}}_\mathrm{f} = \mathbf{x}_\mathrm{f} + \delta\mathbf{x}_\mathrm{f}$, and its corresponding cost $J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f})$.*
Case 1: $T = 0$. There exists a perturbation center $\delta\mathbf{x}_c$ (consisting of only a position shift) with $\|\delta\mathbf{x}_c\| = \mathrm{O}(J^2)$ such that if $\|\delta\mathbf{x}_0\| \le \eta J^3$ and $\|\delta\mathbf{x}_\mathrm{f} - \delta\mathbf{x}_c\| \le \eta J^3$, then $J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f}) \le J(1 + 4\eta + \mathrm{O}(J))$ and the spatial deviation of the perturbed trajectory $\tilde{\mathbf{x}}(t)$ from $\mathbf{x}(t)$ is $\mathrm{O}(J)$.

Case 2: $T > 0$. *If $\|\delta\mathbf{x}_0\| \leq \eta J^3$ and $\|\delta\mathbf{x}_\mathrm{f}\| \leq \eta J^3$, then $J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f}) \leq J\big(1 + \mathrm{O}\big(\eta J^2 T^{-1}\big)\big)$ and the spatial deviation of the perturbed trajectory $\tilde{\mathbf{x}}(t)$ from $\mathbf{x}(t)$ is $\mathrm{O}(J)$.*

Proof. For the proof, see Appendix C. □

We are now in a position to prove that the cost of the trajectory returned by FMT* approaches that of an optimal trajectory as the number of samples $n \to \infty$. The proof proceeds in two steps. First, we establish that there is a sequence of waypoints in $\mathcal{S}$ that are placed closely along the optimal path and approximately evenly-spaced in cost. Then we show that the existence of these waypoints guarantees that FMT* finds a path with a cost close to that of the optimal cost. The theorem and proof combine elements from Theorem 1 in [80] and Theorem IV.6 from [82].

Definition 26 (Strong δ-Clearance)**.** A trajectory $\mathbf{x}(t)$ is said to have *strong* δ-clearance if, for some $\delta > 0$ and all t, the Euclidean distance between $\mathbf{x}(t)$ and any point in $\mathcal{X}_\mathrm{obs}$ is greater than δ.

Theorem 27 (Existence of Waypoints near an Optimal Path)**.** *Let $\mathbf{x}^*(t)$ be a feasible trajectory for the motion planning problem Eq. (5.1) with strong δ-clearance, let $\mathbf{u}^*(t) = \sum_{i=1}^{N} \boldsymbol{\Delta}\mathbf{v}_i^* \cdot \delta(t - \tau_i^*)$ be its associated control trajectory, and let J^* be its cost. Furthermore, let $\mathcal{S} \cup \{\mathbf{x}_\mathrm{init}\}$ be a set of $n \in \mathbb{N}$ points from $\mathcal{X}_\mathrm{free}$ with dispersion $D(\mathcal{S}) \leq \gamma n^{-1/d}$. Let $\epsilon > 0$, and choose $\bar{J} = 4\big(\gamma n^{-1/d}/\epsilon\big)^{1/3}$. Then, provided that n is sufficiently large, there exists a sequence of points $\{\mathbf{y}_k\}_{k=0}^{K}$, $\mathbf{y}_k \in \mathcal{S}$ such that $J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq \bar{J}$, the cost of the path $\mathbf{y}(t)$ made by joining all of the steering trajectories between $\mathbf{y}_k$ and $\mathbf{y}_{k+1}$ is $\sum_{k=0}^{K-1} J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq (1+\epsilon)J^*$, and $\mathbf{y}(t)$ is itself strong $(\delta/2)$-clear.*

Proof. We first note that if $J^* = 0$ then we can pick $\mathbf{y}_0 = \mathbf{x}^*(t_0)$ and $\mathbf{y}_1 = \mathbf{x}^*(t_\mathrm{f})$ as the only points in $\{\mathbf{y}_k\}$ and the result is trivial. Thus assume that $J^* > 0$. Construct a sequence of times $\{t_k\}_{k=0}^{K}$ and corresponding points $\mathbf{x}_k^* = \mathbf{x}^*(t_k)$ spaced along $\mathbf{x}^*(t)$ in cost intervals of $\bar{J}/2$. We admit a slight abuse of notation here in that $\mathbf{x}^*(\tau_i^*)$ may represent a state with any velocity along the length of the impulse $\boldsymbol{\Delta}\mathbf{v}_i^*$; to be precise, pick $\mathbf{x}_0^* = \mathbf{x}_\mathrm{init}$, $t_0 = 0$, and for $k = 1, 2, \ldots$ define $j_k = \min\Big\{ j \;\Big|\; \sum_{i=1}^{j} \|\boldsymbol{\Delta}\mathbf{v}_i^*\| > k\frac{\bar{J}}{2} \Big\}$ and select t_k and $\mathbf{x}_k^*$ as:

$$
\begin{aligned}
t_k &= \tau_{j_k}^* \\
\mathbf{x}_k^* &= \lim_{t \to t_k^-} \mathbf{x}^*(t) + \left(k\frac{\bar{J}}{2} - \sum_{i=1}^{j_k - 1} \|\boldsymbol{\Delta}\mathbf{v}_i^*\| \right) \mathbf{B} \frac{\boldsymbol{\Delta}\mathbf{v}_i^*}{\|\boldsymbol{\Delta}\mathbf{v}_i^*\|}.
\end{aligned}
$$

Let $K = \lceil J^* \rceil / \big(\bar{J}/2\big)$ and set $t_K = t_\mathrm{f}$, $\mathbf{x}_K^* = \mathbf{x}^*(t_\mathrm{f})$. Since the trajectory $\mathbf{x}^*(t)$ to be approximated is fixed, for sufficiently small $\bar{J}$ (equivalently, sufficiently large n) we may ensure that the control applied between each $\mathbf{x}_k^*$ and $\mathbf{x}_{k+1}^*$ occurs only at the endpoints. In particular this may be accomplished by choosing n large enough so that $\bar{J} < \min_i \|\boldsymbol{\Delta}\mathbf{v}_i^*\|$. In the limit $\bar{J} \to 0$, the vast majority of the 2-impulse steering connections between successive $\mathbf{x}_k^*$ will be zero-time maneuvers (arranged along the length of each burn $\boldsymbol{\Delta}\mathbf{v}_i^*$) with only N positive-time maneuvers spanning the regions of $\mathbf{x}^*(t)$

between burns. By considering this regime of n, we note that applying 2-impulse steering between successive $\mathbf{x}_k^*$ (which otherwise may only approximate the performance of a more complex control scheme) requires cost no greater than that of $\mathbf{x}^*$ itself along that step, *i.e.*, $\bar{J}/2$.

We now inductively define a sequence of points $\{\hat{\mathbf{x}}_k^*\}_{k=0}^K$ by $\hat{\mathbf{x}}_0^* = \mathbf{x}_0^*$ and for each $k > 0$: (i) if $t_k = t_{k-1}$, pick $\hat{\mathbf{x}}_k^* = \mathbf{x}_k^* + \delta\mathbf{x}_{c,k} + (\hat{\mathbf{x}}_{k-1}^* - \mathbf{x}_{k-1}^*)$, where $\delta\mathbf{x}_{c,k}$ comes from Lemma 25 for zero-time approximate steering between $\mathbf{x}_{k-1}^*$ and $\mathbf{x}_k^*$ subject to perturbations of size ϵJ^3; (ii) otherwise if $t_k > t_{k-1}$, pick $\hat{\mathbf{x}}_k^* = \mathbf{x}_k^* + (\hat{\mathbf{x}}_{k-1}^* - \mathbf{x}_{k-1}^*)$. The reason for defining these $\hat{\mathbf{x}}_k^*$ is that the process of approximating each $\mathbf{\Delta v}_i^*$ by a sequence of small burns necessarily incurs some short-term position drift. Since $\delta\mathbf{x}_{c,k} = \mathrm{O}(\bar{J}^2)$ for each k, and since $K = \mathrm{O}(\bar{J}^{-1})$, the maximum accumulated difference satisfies $\max_k \|\hat{\mathbf{x}}_k^* - \mathbf{x}_k^*\| = \mathrm{O}(\bar{J})$.

For each k consider the Euclidean ball centered at $\hat{\mathbf{x}}_k^*$ with radius $\gamma n^{-\frac{1}{d}}$, *i.e.*, let $\mathcal{B}_k := \mathcal{B}\left(\hat{\mathbf{x}}_k^*, \gamma n^{-\frac{1}{d}}\right)$. By Definition 24 and our restriction on $\mathcal{S}$, each $\mathcal{B}_k$ contains at least one point from $\mathcal{S}$. Hence for every $\mathcal{B}_k$ we can pick a waypoint $\mathbf{y}_k$ such that $\mathbf{y}_k \in \mathcal{B}_k \cap \mathcal{S}$. Then $\|\mathbf{y}_k - \hat{\mathbf{x}}_k^*\| \leq \gamma n^{-\frac{1}{d}} = \epsilon(\bar{J}/2)^3/8$ for all k, and thus by Lemma 25 (with $\eta = \epsilon/8$) we have that:

$$J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq \frac{\bar{J}}{2}\left(1 + \frac{\epsilon}{2} + \mathrm{O}(\bar{J})\right) \leq \frac{\bar{J}}{2}(1+\epsilon)$$

for sufficiently large n. In applying Lemma 25 to Case (ii) for k such that $t_{k+1} > t_k$, we note that the T^{-1} term is mitigated by the fact that there are only a finite number of burn times τ_i^* along $\mathbf{x}^*(t)$. Thus, for each such k, $t_{k+1} - t_k \geq \min_j(t_{j+1} - t_j) > 0$, so in every case we have $J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq (\bar{J}/2)(1+\epsilon)$. That is, each steering segment connecting $\mathbf{y}_k$ to $\mathbf{y}_{k+1}$ approximates the cost of the corresponding $\mathbf{x}_k^*$ to $\mathbf{x}_{k+1}^*$ segment of $\mathbf{x}^*(t)$ up to a multiplicative factor of ϵ, and thus:

$$\sum_{k=0}^{K-1} J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq (1+\epsilon)J^*.$$

Finally, to establish that $\mathbf{y}(t)$, the trajectory formed by steering through the $\mathbf{y}_k$'s in succession, has sufficient obstacle clearance, we note that its distance from $\mathbf{x}^*(t)$ is bounded by $\max_k \|\hat{\mathbf{x}}_k^* - \mathbf{x}_k^*\| = \mathrm{O}(\bar{J})$ plus the deviation bound from Definition 24, again $\mathrm{O}(\bar{J})$. For sufficiently large n, the total distance, $\mathrm{O}(\bar{J})$, will be bounded by $\delta/2$, and thus $\mathbf{y}(t)$ will have strong $(\delta/2)$-clearance. □

We now prove that FMT* is asymptotically optimal in the number of points n, provided the conditions required in Theorem 27 hold; note the proof is heavily based on Theorem VI.1 from [81].

Theorem 28 (Asymptotic Performance of FMT*)**.** *Let $\mathbf{x}^*(t)$ be a feasible trajectory satisfying Eq.* (5.1) *with strong δ-clearance and cost J^*. Let $\mathcal{S} \cup \{\mathbf{x}_0\}$ be a set of $n \in \mathbb{N}$ samples from $\mathcal{X}_{\text{free}}$ with dispersion $D(\mathcal{S}) \leq \gamma n^{-1/d}$. Finally, let J_n denote the cost of the path returned by FMT* with n points in $\mathcal{S}$ while using a cost threshold $\bar{J}(n) = \omega(n^{-1/3d})$ and $\bar{J} = \mathrm{o}(1)$. (That is, $\bar{J}(n)$ asymptotically dominates $n^{-1/3d}$ and is asymptotically dominated by 1.) Then $\lim_{n\to\infty} J_n \leq J^*$.*

Proof. Let $\epsilon > 0$. Pick n sufficiently large so that $\delta/2 \geq \bar{J} \geq 4\big(\gamma n^{-1/d}/\epsilon\big)^{1/3}$ such that Theorem 27 holds. That is, there exists a sequence of waypoints $\{\mathbf{y}_k\}_{k=0}^{K}$ approximating $\mathbf{x}^*(t)$ such that the trajectory $\mathbf{y}(t)$ created by sequentially steering through the $\mathbf{y}_k$ is strong $\delta/2$-clear, whose connection costs satisfy $J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq \bar{J}$, and whose total cost satisfies $\sum_{k=0}^{K-1} J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq (1+\epsilon)J^*$. We show that FMT* recovers a path with cost at least as good as $\mathbf{y}(t)$; that is, we show that $\lim_{n\to\infty} J_n \leq J^*$.

Consider running FMT* to completion, and for each $\mathbf{y}_k$, let $c(\mathbf{y}_k)$ denote the cost-to-come of $\mathbf{y}_k$ in the generated graph (with value ∞ if $\mathbf{y}_k$ is not connected). We show by induction that:

$$\min(c(\mathbf{y}_m), J_n) \leq \sum_{k=0}^{m-1} J(\mathbf{y}_k, \mathbf{y}_{k+1}) \tag{7.5}$$

for all $m \in [1, \ldots, K]$. For the base case $m = 1$, we note by the initialization of FMT* on line 1 of Algorithm 6 that $\mathbf{x}_{\text{init}}$ is in $\mathcal{V}_{\text{open}}$; therefore, by the design of FMT* (per lines 5–9), every possible feasible connection is made between the first waypoint $\mathbf{y}_0 = \mathbf{x}_{\text{init}}$ and its neighbors. Since $J(\mathbf{y}_0, \mathbf{y}_1) \leq \bar{J}$ and the edge $(\mathbf{y}_0, \mathbf{y}_1)$ is collision-free, it is also in the FMT* graph. Then $c(\mathbf{y}_1) = J(\mathbf{y}_0, \mathbf{y}_1)$. Now assuming that Eq. (7.5) holds for $m-1$, one of the following statements holds:

1. $J_n \leq \sum_{k=0}^{m-2} J(\mathbf{y}_k, \mathbf{y}_{k+1})$

2. $c(\mathbf{y}_{m-1}) \leq \sum_{k=0}^{m-2} J(\mathbf{y}_k, \mathbf{y}_{k+1})$ and FMT* ends before considering $\mathbf{y}_m$.

3. $c(\mathbf{y}_{m-1}) \leq \sum_{k=0}^{m-2} J(\mathbf{y}_k, \mathbf{y}_{k+1})$ and $\mathbf{y}_{m-1} \in \mathcal{V}_{\text{open}}$ when $\mathbf{y}_m$ is first considered

4. $c(\mathbf{y}_{m-1}) \leq \sum_{k=0}^{m-2} J(\mathbf{y}_k, \mathbf{y}_{k+1})$ and $\mathbf{y}_{m-1} \notin \mathcal{V}_{\text{open}}$ when $\mathbf{y}_m$ is first considered.

We now show for each case that our inductive hypothesis holds.

Case 1: $J_n \leq \sum_{k=0}^{m-2} J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq \sum_{k=0}^{m-1} J(\mathbf{y}_k, \mathbf{y}_{k+1})$.

Case 2: Since at every step FMT* considers the node that is the endpoint of the path with the lowest cost, if FMT* ends before considering $\mathbf{y}_m$, we have $J_n \leq c(\mathbf{y}_m) \leq c(\mathbf{y}_{m-1}) + J(\mathbf{y}_{m-1}, \mathbf{y}_m) \leq \sum_{k=0}^{m-1} J(\mathbf{y}_k, \mathbf{y}_{k+1})$.

Case 3: Since the neighborhood of $\mathbf{y}_m$ is collision-free by the clearance property of $\mathbf{y}$, and since $\mathbf{y}_{m-1}$ is a possible parent candidate for connection, $\mathbf{y}_m$ will be added to the FMT* tree as soon as it is considered with $c(\mathbf{y}_m) \leq c(\mathbf{y}_{m-1}) + J(\mathbf{y}_{m-1}, \mathbf{y}_m) \leq \sum_{k=0}^{m-1} J(\mathbf{y}_k, \mathbf{y}_{k+1})$.

Case 4: When $\mathbf{y}_m$ is considered, it means there is a node $\mathbf{z} \in \mathcal{V}_{\text{open}}$ (with minimum cost-to-come through the FMT* tree) and $\mathbf{y}_m \in \mathcal{R}(\mathbf{z}, \bar{J})$. Then $c(\mathbf{y}_m) \leq c(\mathbf{z}) + J(\mathbf{z}, \mathbf{y}_m)$. Since $c(\mathbf{y}_{m-1}) < \infty$, $\mathbf{y}_{m-1}$ must be added to the tree by the time FMT* terminates. Consider the path from $\mathbf{x}_{\text{init}}$ to $\mathbf{y}_{m-1}$ in the final FMT* tree, and let $\mathbf{w}$ be the last vertex along this path, which is in $\mathcal{V}_{\text{open}}$ at the

time when $\mathbf{y}_m$ is considered. If $\mathbf{y}_m \in \mathcal{R}(\mathbf{w}, \bar{J})$, *i.e.*, $\mathbf{w}$ is a parent candidate for connection, then:

$$\begin{aligned} c(\mathbf{y}_m) &\leq c(\mathbf{w}) + J(\mathbf{w}, \mathbf{y}_m) \\ &\leq c(\mathbf{w}) + J(\mathbf{w}, \mathbf{y}_{m-1}) + J(\mathbf{y}_{m-1}, \mathbf{y}_m) \\ &\leq c(\mathbf{y}_{m-1}) + J(\mathbf{y}_{m-1}, \mathbf{y}_m) \\ &\leq \sum_{k=0}^{m-1} J(\mathbf{y}_k, \mathbf{y}_{k+1}). \end{aligned}$$

Otherwise if $\mathbf{y}_m \notin \mathcal{R}(\mathbf{w}, \bar{J})$, then $J(\mathbf{w}, \mathbf{y}_m) > \bar{J}$ and:

$$\begin{aligned} c(\mathbf{y}_m) &\leq c(\mathbf{z}) + J(\mathbf{z}, \mathbf{y}_m) \\ &\leq c(\mathbf{w}) + \bar{J} \\ &\leq c(\mathbf{w}) + J(\mathbf{w}, \mathbf{y}_m) \\ &\leq c(\mathbf{w}) + J(\mathbf{w}, \mathbf{y}_{m-1}) + J(\mathbf{y}_{m-1}, \mathbf{y}_m) \\ &\leq c(\mathbf{y}_{m-1}) + J(\mathbf{y}_{m-1}, \mathbf{y}_m) \\ &\leq \sum_{k=0}^{m-1} J(\mathbf{y}_k, \mathbf{y}_{k+1}). \end{aligned}$$

where we used the fact that $\mathbf{w}$ is on the path of $\mathbf{y}_{m-1}$ to establish $c(\mathbf{w}) + J(\mathbf{w}, \mathbf{y}_{m-1}) \leq c(\mathbf{y}_{m-1})$. Thus, by induction, Eq. (7.5) holds for all m. Taking $m = K$, we finally have that $J_n \leq c(\mathbf{y}_K) \leq \sum_{k=0}^{K-1} J(\mathbf{y}_k, \mathbf{y}_{k+1}) \leq (1+\epsilon)J^*$, as desired. □

Remark 29 (Asymptotic Optimality of FMT*). If the planning problem at hand admits an optimal solution that does not itself have strong δ-clearance, but is arbitrarily approximable both pointwise and in cost by trajectories with strong clearance (see [81] for additional discussion on why such an assumption is reasonable), then Theorem 28 implies the asymptotic optimality of FMT*.

7.5 Trajectory Smoothing

Due to the discreteness caused by using a *finite* number of samples, sampling-based solutions will necessarily be approximations to true optima. In an effort to compensate for this limitation, we offer in this section two techniques to improve the quality of solutions returned by our planner from Section 7.3. We first describe a straightforward method for reducing the sum of Δv-vector magnitudes along concatenated sequences of edge trajectories that can also be used to improve the search for propellant-efficient trajectories in the feasible state space $\mathcal{X}_{\text{free}}$. We then follow with a fast post-processing algorithm for further reducing propellant cost after a solution has been reported.

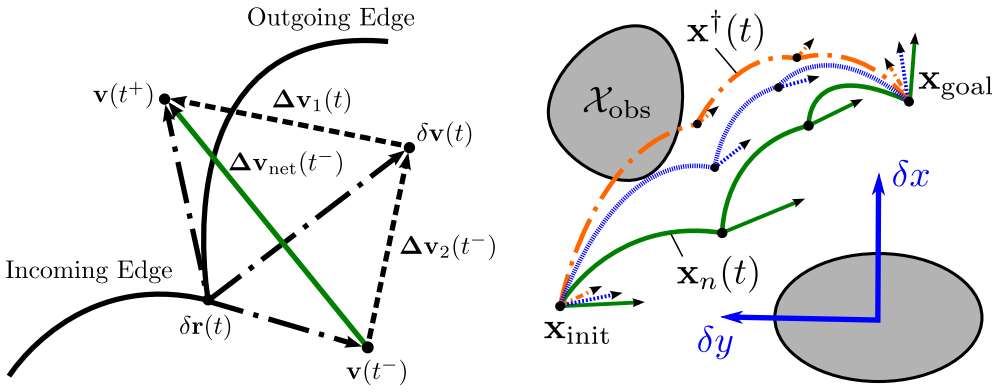
The first technique removes unnecessary Δv-vectors that occur when joining sub-trajectories

(edges) in the planning graph. Consider merging two edges at a node with position $\delta\mathbf{r}(t)$ and velocity $\delta\mathbf{v}(t)$ as in Fig. 7.4a. A naive concatenation would retain both $\mathbf{\Delta v}_2(t^-)$ (the rendezvous burn added to the incoming velocity $\mathbf{v}(t^-)$) and $\mathbf{\Delta v}_1(t)$ (the intercept burn used to achieve the outgoing velocity $\mathbf{v}(t^+)$) individually within the combined control trajectory. Yet, because these impulses occur at the same time, a more realistic approach should merge them into a single net Δv-vector $\mathbf{\Delta v}_{\text{net}}(t^-)$. By the triangle inequality, we have that:

$$\left\|\mathbf{\Delta v}_{\text{net}}\left(t^-\right)\right\| = \left\|\mathbf{\Delta v}_2\left(t^-\right) + \mathbf{\Delta v}_1(t)\right\| \leq \left\|\mathbf{\Delta v}_2\left(t^-\right)\right\| + \left\|\mathbf{\Delta v}_1(t)\right\|.$$

Hence, merging edges in this way guarantees Δv-savings for solution trajectories under our propellant-cost metric. Furthermore, incorporating net Δv's into the cost-to-come during graph construction can make exploration of the search space more efficient; the cost-to-come $c(\mathbf{z})$ for a given node $\mathbf{z}$ would then reflect the cost to rendezvous with $\mathbf{z}$ from $\mathbf{x}_{\text{init}}$ through a series of intermediate intercepts rather than a series of rendezvous maneuvers (as a trajectory designer might normally expect). Note, on the other hand, that these new net Δv-vectors may be larger than either individual burn, which may violate control constraints; control feasibility tests (allocation feasibility to thrusters, plume impingement, *etc.*) must thus be reevaluated for each new impulse. Furthermore, observe that the velocity $\mathbf{v}(t)$ is no longer achieved at edge endpoints when two edges as in Fig. 7.4a are merged in this fashion; the state $\mathbf{x}(t)$ is skipped altogether. This can be problematic for our active safety policy from Section 6.3 for states along the incoming edge. This is because our actively-safe abort maneuver relies on rendezvousing with the endpoint $\mathbf{x} = [\,\delta\mathbf{r}(t), \delta\mathbf{v}(t)\,]$ *exactly* before executing its one-burn circularization maneuver. To compensate for this, care must be taken to ensure that the burn $\mathbf{\Delta v}_2(t^-)$ that is eliminated during merging is appropriately appended to the front of the escape control trajectory and verified for all possible failure configurations. Hence we see the price of smoothing in this way is: (i) re-evaluating control-dependent constraints at edge endpoints before accepting smoothing, and (ii) our original one-burn policy now requires an extra burn, which may not be desirable in some applications.

The second technique attempts to reduce solution cost by adjusting the magnitudes of Δv-vectors in the trajectory returned by FMT* (denoted by $\mathbf{x}_n(t)$ with associated stacked impulse vector $\mathbf{\Delta V}_n$). By relaxing FMT*'s constraint to pass through state samples, strong cost improvements may be gained. The main idea is to deform our low-cost, feasible solution $\mathbf{x}_n(t)$ as much as possible towards the unconstrained minimum-propellant solution $\mathbf{x}^*(t)$ between $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$, as determined by the 2-point Boundary Value Problem (Eq. (7.2)) solution from Section 7.1 (in other words, use a homotopic transformation from $\mathbf{x}_n(t)$ to $\mathbf{x}^*(t)$). However, a naive attempt to solve Eq. (7.2) in its full generality would be too time-consuming to be useful, and would threaten the real-time capability of our approach. Assuming our sampling-based trajectory is near-optimal (or at least, in a low-cost solution homotopy), we can relax Eq. (7.2) by keeping the number of burns N, end time $t_{\text{f}} \coloneqq t_{\text{final}}$, and burn times τ_i fixed from our planning solution, and solve for an approximate unconstrained



(a) Smoothing during graph construction (merges Δv-vectors at edge endpoints).

(b) Smoothing during post-processing (see Algorithm 7).

Figure 7.4: Improving sampling-based solutions under minimal-propellant impulsive dynamics. Figure 7.4a can be used to merge Δv-vectors between edge endpoints during and after graph construction, while Fig. 7.4b illustrates the post-processing smoothing algorithm given in Algorithm 7 (the original trajectory $\mathbf{x}_n(t)$ is solid, the approximate unconstrained optimum $\mathbf{x}^\dagger(t)$ is dash-dotted, and the resulting "smoothed" trajectory derived from their combination is shown dashed).

minimum-propellant solution $\mathbf{\Delta V}^\dagger$ with associated state trajectory $\mathbf{x}^\dagger(t)$ via:

$$\begin{aligned}
&\underset{\mathbf{\Delta v}_i}{\text{minimize}} && \sum_{i=1}^{N} \|\mathbf{\Delta v}_i\|_2 && && (7.6)\\
&\text{subject to} && \mathbf{\Phi}_v(t_{\text{final}}, \{\tau_i\}_i)\mathbf{\Delta V} = \mathbf{x}_{\text{goal}} - \mathbf{\Phi}(t_{\text{final}}, t_{\text{init}})\mathbf{x}_{\text{init}} && && \text{Dynamics/Boundary Conditions}\\
& && \|\mathbf{\Delta v}_i\|_2 \leq \Delta v_{\max} \quad \text{for all burns } i && && \text{Burn Magnitude Bounds}
\end{aligned}$$

(see Section 5.1.3 for definitions). It can be shown that Eq. (7.6) is a second-order cone program (SOCP), and hence quickly solved using standard convex solvers. As the following theorem shows explicitly, we can safely deform the trajectory $\mathbf{x}_n(t)$ towards $\mathbf{x}^\dagger(t)$ without violating our dynamics and boundary conditions if we use a convex combination of our two control trajectories $\mathbf{\Delta V}_n$ and $\mathbf{\Delta V}^\dagger$. This follows from the principle of superposition, given that the CWH equations are Linear Time-Invariant (LTI), and the fact that both solutions already satisfy the boundary conditions.

Theorem 30 (Dynamic Feasibility of CWH Trajectory Smoothing)**.** *Suppose* $\mathbf{x}_n(t)$ *and* $\mathbf{x}^\dagger(t)$ *with respective control vectors* $\mathbf{\Delta V}_n$ *and* $\mathbf{\Delta V}^\dagger$ *are two state trajectories which satisfy the impulsive CWH steering problem Eq.* (7.1) *between states* $\mathbf{x}_{\text{init}}$ *and* $\mathbf{x}_{\text{goal}}$*. Then the trajectory* $\mathbf{x}(t)$ *generated by the convex combination of* $\mathbf{\Delta V}_n$ *and* $\mathbf{\Delta V}^\dagger$ *is itself a convex combination of* $\mathbf{x}_n(t)$ *and* $\mathbf{x}^\dagger(t)$*, and hence also satisfies Eq.* (7.1)*.*

Algorithm 7 "Trajectory smoothing" algorithm for impulsive CWH dynamics. Given a trajectory $\mathbf{x}_n(t), t \in [t_{\text{init}}, t_{\text{goal}}]$ between initial and goal states $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ satisfying Eq. (5.1) with impulses $\boldsymbol{\Delta\mathbf{V}}_n$ applied at times $\{\tau_i\}_i$, returns another feasible trajectory with reduced propellant-cost.

Initialize the smoothed trajectory $\mathbf{x}_{\text{smooth}}(t)$ as $\mathbf{x}_n(t)$, with $\boldsymbol{\Delta\mathbf{V}}_{\text{smooth}} = \boldsymbol{\Delta\mathbf{V}}_n$
Compute the unconstrained optimal control vector $\boldsymbol{\Delta\mathbf{V}}^\dagger$ by solving Eq. (7.6)
Compute the unconstrained optimal state trajectory $\mathbf{x}^\dagger(t)$ using Eq. (5.4) *(Section 5.1.3)*
Initialize weight α and its lower and upper bounds as $\alpha \leftarrow 1$, $\alpha_\ell \leftarrow 0$, $\alpha_u \leftarrow 1$
while true
    $\mathbf{x}(t) \leftarrow (1-\alpha)\mathbf{x}_n(t) + \alpha\mathbf{x}^\dagger(t)$
    $\boldsymbol{\Delta\mathbf{V}} \leftarrow (1-\alpha)\boldsymbol{\Delta\mathbf{V}}_n + \alpha\boldsymbol{\Delta\mathbf{V}}^\dagger$
    if CollisionFree$(\mathbf{x}(t), \boldsymbol{\Delta\mathbf{V}}, t)$
        $\alpha_\ell \leftarrow \alpha$
        Save the smoothed trajectory $\mathbf{x}_{\text{smooth}}(t)$ as $\mathbf{x}(t)$ and control $\boldsymbol{\Delta\mathbf{V}}_{\text{smooth}}$ as $\boldsymbol{\Delta\mathbf{V}}$
    else
        $\alpha_u \leftarrow \alpha$
    if $\alpha_u - \alpha_\ell$ is less than tolerance $\delta\alpha_{\min} \in (0, 1)$
        break
    $\alpha \leftarrow (\alpha_\ell + \alpha_u)/2$
return the smoothed trajectory $\mathbf{x}_{\text{smooth}}(t)$, with $\boldsymbol{\Delta\mathbf{V}}_{\text{smooth}}$

Proof. Let $\boldsymbol{\Delta\mathbf{V}} = \alpha\boldsymbol{\Delta\mathbf{V}}_n + (1-\alpha)\boldsymbol{\Delta\mathbf{V}}^\dagger$ for some value $\alpha \in [0, 1]$. From our dynamics equation,

$$\begin{aligned}
\mathbf{x}(t) &= \boldsymbol{\Phi}(t, t_{\text{init}})\mathbf{x}_{\text{init}} + \boldsymbol{\Phi}_v(t, \{\tau_i\}_i)\boldsymbol{\Delta\mathbf{V}} \\
&= [\alpha + (1-\alpha)]\boldsymbol{\Phi}(t, t_{\text{init}})\mathbf{x}_{\text{init}} + \boldsymbol{\Phi}_v(t, \{\tau_i\}_i)\Big[\alpha\boldsymbol{\Delta\mathbf{V}}_n + (1-\alpha)\boldsymbol{\Delta\mathbf{V}}^\dagger\Big] \\
&= \alpha[\boldsymbol{\Phi}(t, t_{\text{init}})\mathbf{x}_{\text{init}} + \boldsymbol{\Phi}_v(t, \{\tau_i\}_i)\boldsymbol{\Delta\mathbf{V}}_n] + (1-\alpha)\Big[\boldsymbol{\Phi}(t, t_{\text{init}})\mathbf{x}_0 + \boldsymbol{\Phi}_v(t, \{\tau_i\}_i)\boldsymbol{\Delta\mathbf{V}}^\dagger\Big] \\
&= \alpha\mathbf{x}_n(t) + (1-\alpha)\mathbf{x}^\dagger(t)
\end{aligned}$$

which is a convex combination, as required. Substituting $t = t_{\text{init}}$ or $t = t_{\text{goal}}$, we see that $\mathbf{x}(t)$ satisfies the boundary conditions given that $\mathbf{x}_n(t)$ and $\mathbf{x}^\dagger(t)$ do. This completes the proof. □

We take advantage of this fact for trajectory smoothing. Our algorithm, reported as Algorithm 7 and illustrated in Fig. 7.4b, computes the approximate unconstrained minimum-propellant solution $\mathbf{x}^\dagger(t)$ and returns it (if feasible) or otherwise conducts a bisection line search on α, returning a convex combination of our original planning solution $\mathbf{x}_n(t)$ and $\mathbf{x}^\dagger(t)$ that comes as close to $\mathbf{x}^\dagger(t)$ as possible without violating trajectory constraints. Note because $\boldsymbol{\Delta\mathbf{V}}_n$ lies in the feasible set of Eq. (7.6), the algorithm can only improve the final propellant cost. By design, Algorithm 7 is geared towards reducing our original solution propellant-cost as quickly as possible while maintaining feasibility; the most expensive computational components are the calculation of $\boldsymbol{\Delta\mathbf{V}}^\dagger$ and collision-checking (consistent with our sampling-based algorithm). Fortunately, the number of collision-checks is limited

by the maximum number of iterations $\left\lceil \log_2\left(\frac{1}{\delta\alpha_{\min}}\right)\right\rceil + 1$, given tolerance $\delta\alpha_{\min} \in (0, 1)$. As an added bonus, for strictly time-constrained applications that require a solution in a fixed amount of time, the algorithm can be easily modified to return the α_ℓ-weighted trajectory $\mathbf{x}_{\text{smooth}}(t)$ when time runs out, as the feasibility of this trajectory is maintained as an algorithm invariant.

7.6 Conclusion

In this chapter, a solution methodology has been presented which enables the FMT* sampling-based motion planning algorithm (or BFMT*, by extension) to be used for provably-safe propellant-optimized spacecraft guidance. The approach is divided into two parts: (i) an offline phase, which is used to precompute a set of actively-safe samples along with their associated interconnecting steering trajectories and nearest neighbors (*i.e.*, reachability sets), and (ii) an online phase, which calls FMT* to search for a low propellant-cost solution between an initial state and a goal region. Time permitting, a fast homotopic trajectory smoothing technique based on bisection may also be used in post-processing to further improve solution quality.

The framework presented solves Eq. (5.1) in the limit as $n \to \infty$, assuming a low-dispersion sampling sequence, and holds for any impulsively-actuated, Linear Time-Invariant (LTI) system that is steered by endpoint-impulse maneuvering under the sum-of-2-norms propellant-cost metric (including impulsive Clohessy-Wiltshire-Hill dynamics). The result is a general, flexible framework that can accommodate a multitude of trajectory constraints, without compromising real-time implementability (particularly in cases where most constraints are static and known ahead of time). In the next chapter, we test the performance of our framework on a realistic near-field rendezvous scenario.

Chapter 8

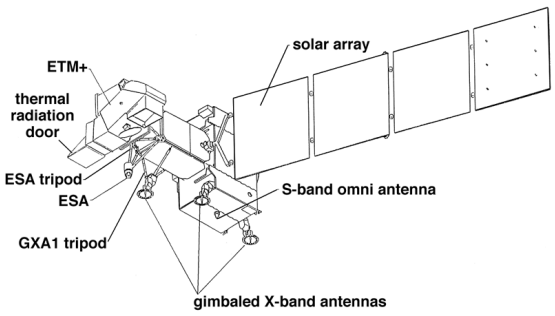
Numerical Experiments

In this chapter, we present a detailed numerical case study demonstrating the capabilities of our sampling-based motion planning approach for near-circular orbit spacecraft proximity operations. The chapter revolves around Section 8.1, which provides quantitative results for a realistic guidance problem encountered frequently during Low Earth Orbit (LEO) operations. Trade-offs are considered between solution cost and execution time as the number of samples n and reachability set cost threshold $\bar{J}$ are varied. These studies also serve to illustrate how autonomous mission design and planning can be conducted using the tools provided in Chapter 7. The chapter closes in Section 8.2 with a brief summary of our solution framework, its overall performance results, and interesting avenues for future research.

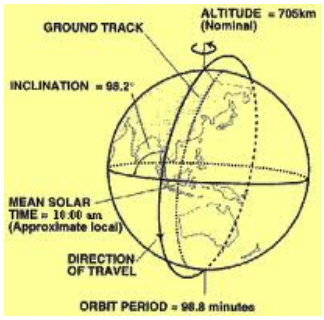
8.1 Near-Circular Orbit Rendezvous Simulations

To begin, consider the two scenarios shown in Fig. 8.2 modeling both planar and non-planar near-field approaches of a chaser spacecraft in close proximity [1] to a target moving on a circular LEO trajectory (as in Fig. 5.1). We imagine the chaser, which starts in a circular orbit of lower radius, must be repositioned through a sequence of pre-specified CWH waypoints (*e.g.*, required for equipment checks, surveying, *etc.*) to a coplanar position located radially above the target, arriving with zero relative velocity in preparation for a final radial ("R-bar") approach. Throughout the maneuver, as described in detail in Chapter 5, the chaser must avoid entering the elliptic target KOZ, enforce hard safety constraints with regard to a two-fault tolerance to stuck-off thruster failures, and otherwise avoid interfering with the target. This includes avoiding the target's nadir-pointing communication lobes (represented by truncated half-cones), and preventing exhaust plume impingement on its surfaces. For context, we use the Landsat-7 spacecraft and orbit as a reference [157, Sec. 3.2] (see Fig. 8.1).

[1] *Close proximity* in this context implies that any higher-order terms of the linearized relative dynamics are negligible, *e.g.*, within a few percent of the target orbit mean radius.

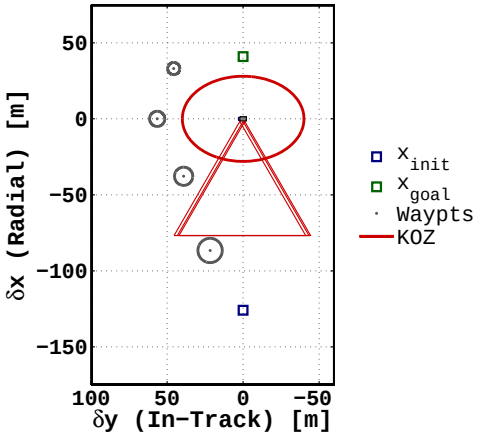


(a) Landsat-7 schematic (Nadir (-δx direction) points down, while the in-track ($+\delta y$) direction points left)

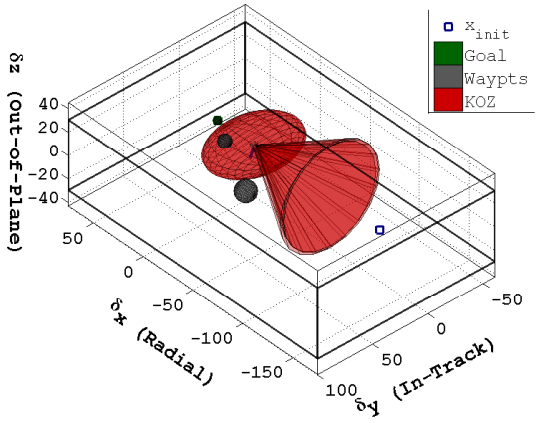


(b) Landsat-7 orbit (Courtesy of the Landsat-7 Handbook)

Figure 8.1: Target spacecraft geometry and orbital scenario used in numerical experiments

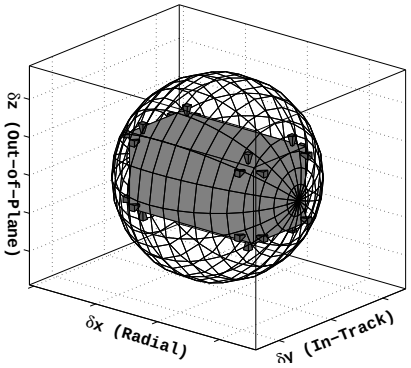


(a) Planar motion planning query

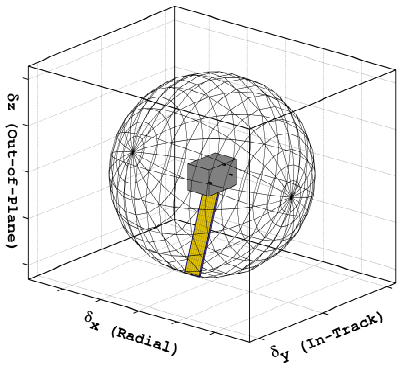


(b) 3-dimensional motion planning query

Figure 8.2: Illustrations of the planar and 3D motion plan queries in the LVLH frame. The chaser must track a series of guidance waypoints to the goal state, located radially above the target. Positional tolerances are visualized as circles around each waypoint, which successively shrink in size.



(a) Chaser spacecraft



(b) Target spacecraft

Figure 8.3: Models of the chaser and target, together with their circumscribing spheres

8.1.1 Simulation Setup

Before proceeding to results, we first outline some key features of our setup. Taking the prescribed query waypoints one at a time as individual goal points $\mathbf{x}_{\text{goal}}$, we solve the given scenario as a series of motion planning problems (or "subplans") linked together into an overall solution, calling FMT* from Section 7.3 once for each subplan. For this multi-plan problem, we take the solution cost to be the sum of individual subplan costs (when using trajectory smoothing, the endpoints between two plans are merged identically to two edges within a plan, as described in Section 7.5).

As our steering controller from Section 7.1 is attitude-independent, states $\mathbf{x} \in \mathbb{R}^d$ are either $\mathbf{x}^{\mathrm{T}} = [\,\delta x, \delta y, \delta \dot{x}, \delta \dot{y}\,]$ with $d = 4$ (planar case) or $\mathbf{x}^{\mathrm{T}} = [\,\delta x, \delta y, \delta z, \delta \dot{x}, \delta \dot{y}, \delta \dot{z}\,]$ with $d = 6$ (non-planar case). This omission of the attitude $\mathbf{q}$ from the state is achieved by employing an attitude policy (assuming a stabilizing attitude controller), which produces $\mathbf{q}(t)$ from the state trajectory $\mathbf{x}(t)$. For illustration purposes, a simple nadir-pointing attitude profile is chosen during nominal guidance, representing a mission requiring constant communication with the ground; for actively-safe abort, we assume a simple "turn-burn-turn" policy, which orients the closest-available thruster under each failure mode as quickly as possible into the direction required for circularization (see Section 6.3).

Given the hyper-rectangular shape of the state-space, we call upon the deterministic, low-dispersion d-dimensional Halton sequence [99] to sample positions and velocities. To improve sampling densities, each subplan uses its own sample space defined around its respective initial and goal waypoints, with some arbitrary threshold space added around them. Additionally, extra samples n_{goal} are taken inside each waypoint ball to facilitate convergence.

Finally, we make note of three additional implementation details. First, for clarity, we list all relevant simulation parameters in Table 8.1. Second, all position-related constraint-checks regard the chaser spacecraft as a point at its center of mass, with all other obstacles artificially inflated by the radius of its circumscribing sphere. Third and finally, all trajectory constraint-checking is implemented by point-wise evaluation with a fixed time-step resolution Δt, using the analytic state transition equations Eq. (A.14) together with steering solutions from Section 7.1 to propagate graph edges; for speed, the line segments between points are excluded. Except near very sharp obstacle corners, this approximation is generally not a problem in practice (obstacles can always be inflated further to account for this). To improve performance, each obstacle primitive (ellipsoid, right-circular cone, hypercube, *etc.*) employs hierarchical collision-checking using hyper-spherical and/or hyper-rectangular bounding volumes to quickly prune points from consideration.

8.1.2 Planar Motion Planning Solution

A representative solution to the posed planning scenario, both with and without the trajectory smoothing algorithm (Algorithm 7), is shown in Fig. 8.4. As shown, the planner successfully finds safe trajectories within each subplan, which are afterwards linked to form an overall solution. The state space of the first subplan shown at the bottom is essentially unconstrained, as the chaser at

Table 8.1: List of parameters used during near-circular orbit rendezvous simulations

Chaser plume half-angle, β_{plume}	$10\,^\circ$
Chaser plume height, H_{plume}	16 m
Chaser thruster fault tolerance, F	2
Cost threshold, $\bar{J}$	0.1–0.4 m/s
Dimension, d	4 (planar), 6 (non-planar)
Goal sample count, n_{goal}	$0.04n$
Goal position tolerance, ϵ_r	3–8 m
Goal velocity tolerance, ϵ_v	0.1–0.5 m/s
Max. allocated thruster Δv magnitude, $\Delta v_{\max,k}$	∞ m/s
Max. commanded Δv-vector magnitude $\|\boldsymbol{\Delta}\mathbf{v}_i\|$, $\Delta v_{\max}$	∞ m/s
Max. plan duration, $T_{\text{plan,max}}$	∞ s
Min. plan duration, $T_{\text{plan,min}}$	0 s
Max. steering maneuver duration, $T_{\max}$	$0.1 \cdot (2\pi/n_{\text{ref}})$
Min. steering maneuver duration, $T_{\min}$	0 s
Sample count, n	50–400 per plan
Simulation timestep, Δt	$0.0005 \cdot (2\pi/n_{\text{ref}})$
Smoothing tolerance, $\delta\alpha_{\min}$	0.01
Target antenna lobe height	75 m
Target antenna beamwidth	$60\,^\circ$
Target KOZ semi-axes, $[\rho_{\delta x}, \rho_{\delta y}, \rho_{\delta z}]$	$\begin{bmatrix}35 & 50 & 15\end{bmatrix}$ m

this point is too far away from the target for plume impingement to come into play. This means every edge connection attempted here is added, so the first subplan illustrates well a discrete subset of the reachable states around $\mathbf{x}_{\text{init}}$ and the unrestrained growth of FMT*. As the second subplan is reached, the effects of the Keep-Out Zone position constraints come in to play, and we see edges begin to take more leftward loops. In subplans 3 and 4, plume impingement begins to play a role. Finally, in subplan 5 at the top, where it becomes very cheap to move between states (as the spacecraft can simply coast to the right for free), we see the initial state connecting to nearly every sample in the subspace, resulting in a straight shot to the final goal. As is evident, straight-line path planning would not approximate these trajectories well—particularly near coasting arcs, along which our dynamics allow the spacecraft to transfer for free.

To understand the smoothing process, examine Fig. 8.5. Here we see how the discrete trajectory sequence from our sampling-based algorithm may be smoothly and continuously deformed towards the unconstrained minimal-propellant trajectory until it meets trajectory constraints (as outlined in Section 7.5); if these constraints happen to be inactive, then the exact minimal-propellant trajectory is returned, as Fig. 8.5a shows. This computational approach is generally quite fast, assuming a well-implemented convex solver is used, as will be seen in the results of the next subsection.

The net Δv costs of the two reported trajectories in this example come to 0.835 m/s (unsmoothed) and 0.811 m/s (smoothed). Compare this to 0.641 m/s, the cost of the unconstrained direct solution that intercepts each of the goal waypoints on its way to rendezvousing with $\mathbf{x}_{\text{goal}}$ (this trajectory

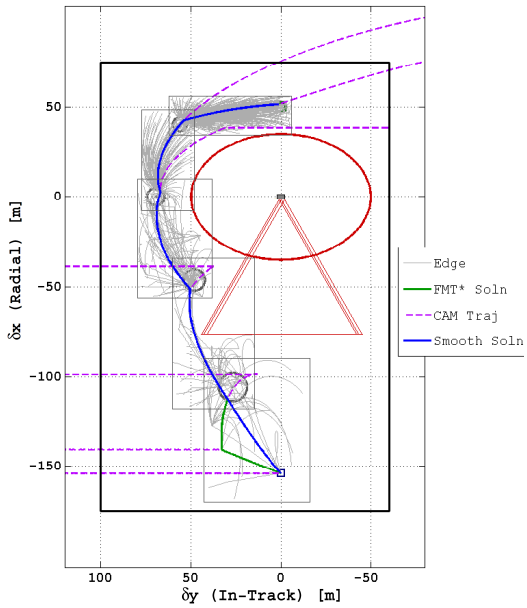
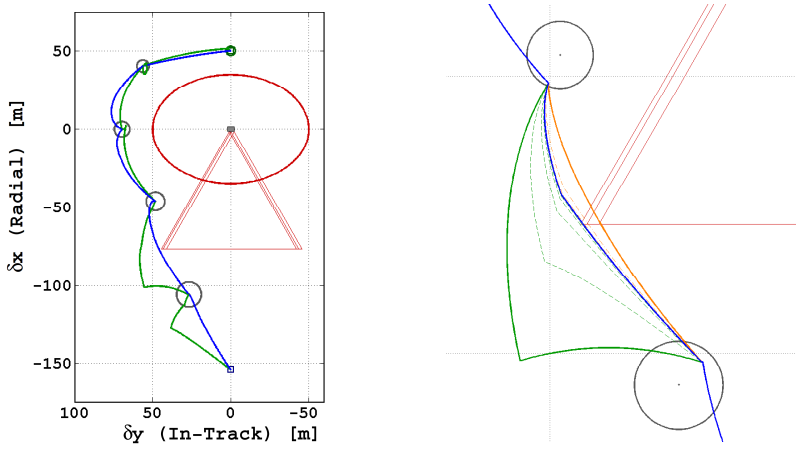


Figure 8.4: Representative planar motion planning solution using the FMT* algorithm (Algorithm 6) with $n = 2000$ (400 per subplan), $\bar{J} = 0.3$ m/s, and relaxed waypoint convergence. The output from FMT* is shown in green, while the trajectory combined with post-processing smoothing is shown in blue. Explored trajectories found to be safe are shown in grey. Actively-safe, minimum-propellant abort trajectories are shown as purple dashed lines (one for each burn $\mathbf{\Delta v}_i$ along the trajectory).



(a) Paths before and after smoothing ($n = 2000$, $\bar{J} = 0.2$ m/s, exact waypoint convergence).

(b) Smoothing algorithm iterations ($n = 1500$, $\bar{J} = 0.3$ m/s, inexact waypoint convergence).

Figure 8.5: Visualizing trajectory smoothing (Algorithm 7) for the solution shown in Fig. 8.4, zoomed in on the second plan. The original plan is shown in green (towards the bottom-left), along with various iterates attempted while converging to the smoothed trajectory shown in blue (in the center). Invalid trajectories, including the lower propellant-cost trajectory used to guide the process, are shown in orange (towards the right).

exits the state-space along the positive in-track direction, a violation of our proposed mission; hence its cost represents an under-approximation to the true optimal cost J^* of the constrained problem). This suggests that our solutions are quite close to the constrained optimum, and certainly on the right order of magnitude. Particularly with the addition of smoothing at lower sample counts, the approach appears to be a viable one for spacecraft planning.

If we compare the net Δv costs to the actual measured propellant consumption given by the sum total of all allocated thruster Δv magnitudes (which equal 1.06 m/s (unsmoothed) and 1.01 m/s (smoothed), respectively), we find increases of 27.0% and 24.5%—as expected, our sum-of-2-norms propellant-cost metric under-approximates the true propellant cost. For point-masses with isotropic control authority (*e.g.*, a steerable or gimbaled thruster that is able to point freely in any direction), our cost metric would be exact. Though inexact for our distributed attitude-dependent propulsion system (see Fig. 8.3a), it is clearly a reasonable proxy for allocated propellant use, returning values on the same order of magnitude. Though we cannot make a strong statement about our proximity to the propellant-optimal solution without directly optimizing over thruster Δv allocations, our solution clearly seems to promote low propellant consumption.

8.1.3 Non-Planar Motion Planning Solution

For the non-planar case, representative smoothed and unsmoothed FMT* solutions can be found in Fig. 8.6. Here the spacecraft is required to move out-of-plane to survey the target from above before reaching the final goal position located radially above the target. The first subplan involves a long re-route around the conical region spanned by the target's communication lobes. Because the chaser begins in a coplanar circular orbit at $\mathbf{x}_{\text{init}}$, most steering trajectories require a fairly large cost to maneuver out-of-plane to the first waypoint. Consequently, relatively few edges are added that both lie in the reachable set of $\mathbf{x}_{\text{init}}$ and safely avoid the large conical obstacles. As we progress to the second and third subplans, the corresponding trees become denser (more steering trajectories are both safe and within our cost threshold $\bar{J}$) as the state space becomes more open. Compared with the planar case, the extra degree-of-freedom associated with the out-of-plane dimension appears to allow more edges ahead of the target in the in-track direction than before, likely because now the exhaust plumes generated by the chaser are well out-of-plane from the target spacecraft. Hence the spacecraft smoothly and tightly curls around the ellipsoidal KOZ to the goal.

The net Δv costs for this example come to 0.611 m/s (unsmoothed) and 0.422 m/s (smoothed). Counter-intuitively, these costs are on the same order of magnitude and slightly cheaper than the planar case; the added freedom given by the out-of-plane dimension appears to outweigh the high costs typically associated with inclination changes and out-of-plane motion. These cost values correspond to total thruster Δv allocation costs of 0.893 m/s and 0.620 m/s, respectively—increases of 46% and 47% above their counterpart cost metric values. Again, our cost metric appears to be a reasonable proxy for actual propellant use.

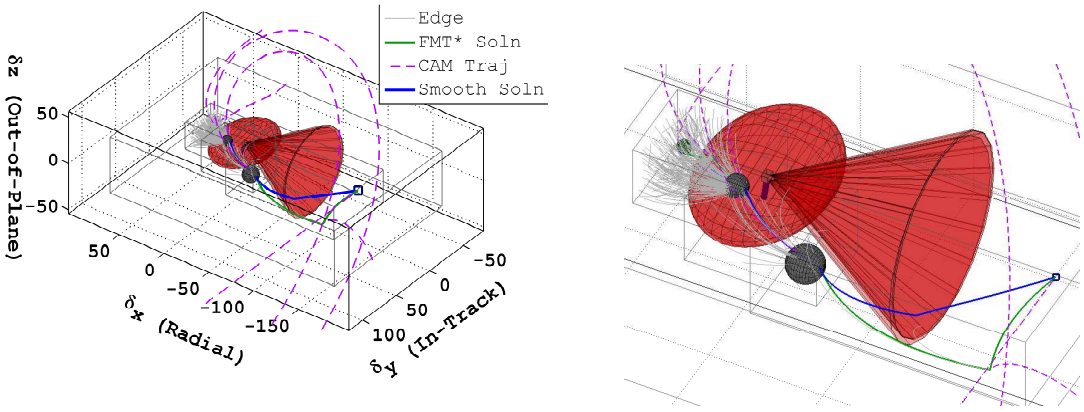
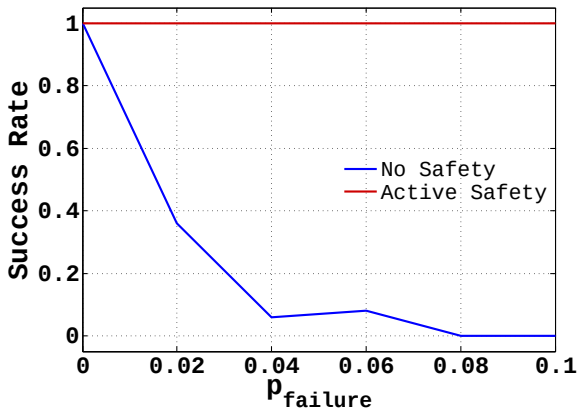


Figure 8.6: Representative non-planar motion planning solution using the FMT* algorithm (Algorithm 6) with $n = 900$ (300 per subplan), $\bar{J} = 0.4$ m/s, and relaxed waypoint convergence. The output from FMT* is shown in green, while the trajectory combined with post-processing smoothing is shown in blue. Explored trajectories found to be safe are shown in grey. Actively-safe minimum-propellant abort trajectories are shown as purple dashed lines (one for each burn $\mathbf{\Delta v}_i$ along the trajectory).

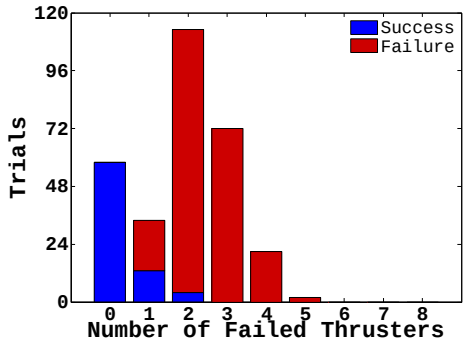
8.1.4 Active Safety Evaluation

A natural question one might ask is just how effective our active-safety policy is, and whether it truly tolerates the control failures it was designed to withstand. In this subsection, we attempt to evaluate through simulation the robustness of our F-fault tolerant abort safety logic (see Section 6.3; note in this study we take $F = 2$). Here safety is measured by repeatedly executing the same motion plan under probabilistic "stuck-off" thruster actuation losses and plotting the resulting average success rates as a function of thruster failure likelihood. We compare two different versions of the planar guidance scenario—one with embedded active-safety constraints, and one without. Failures are simulated at each burn location. Before every scheduled burn, a Bernoulli trial is conducted to assign thruster availabilities; the Δv-allocation algorithm (which solves Eq. (5.6)) then attempts to allocate the burn vector with whichever functional thrusters remain, continuing to follow the nominal trajectory if successful or otherwise activating a CAM (if including actively-safe constraints) or else concluding failure (due to the lack of knowledge of a safe abort decision). If the entire nominal trajectory can be followed safely to the final goal (satisfying Eq. (5.1)), then nominal mission success is declared. If a safe abort CAM is executed (satisfying Eq. (6.1)), then abort success is declared. The overall "success rate" is thus defined as the mean number of nominal or abort mission successes conducted over a set of 50 trials, repeated for each thruster failure probability (ranging from 0% to 10% chance of failure per thruster firing).

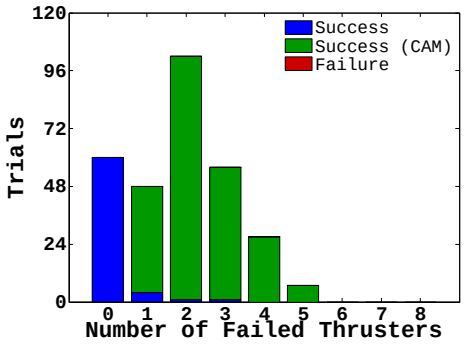
Results from our simulation experiments are presented in Fig. 8.7. Immediately evident is that the actively-safe motion planning strategy works exactly as designed, improving mission success rates to 100% over the safety-unconstrained case. We further note that even though the spacecraft



(a) Success rate comparison vs thruster failure probability, computed from 50 trials for each data point



(b) Success distribution for the nominal strategy (no active-safety constraints)



(c) Success distribution for the safety strategy with CAMs (with active-safety constraints)

Figure 8.7: Measuring the robustness of our CWH active-safety guidance policy. Observe that the active-safety strategy guarantees safety up to two stuck-off thruster failures, as designed.

cannot typically finish its nominal mission once one or more failures have occurred, actively-safe maneuvers are always available, even for cases involving greater than $F = 2$ failures. This indicates that our particular CAM policy (aborting to higher or lower circularized orbits, using a one-burn circularization maneuver coupled with a "turn-burn-turn" attitude slew policy) happens to work quite robustly for the given motion planning scenario, exceeding our design constraints and enabling safety for even higher numbers of failures than anticipated.

8.1.5 Performance Evaluation

To evaluate the performance of our approach, an assessment of solution quality is necessary as a function of planning parameters—*i.e.*, the number of samples n taken and the reachability set cost threshold $\bar{J}$. As proven in Section 7.4, the solution cost will eventually reduce to the optimal value

as we increase the sample size n. Alternatively, we can attempt to reduce cost by increasing the cost threshold $\bar{J}$ used for nearest-neighbor identification, so that more connections are explored. Both, however, work at the expense of running time. To understand the effects of these changes on quality, especially at finite sample counts where the asymptotic guarantees of FMT* do not necessarily imply cost improvements, we measure the cost versus computation time for the planar planning scenario parameterized over several values of n and $\bar{J}$.

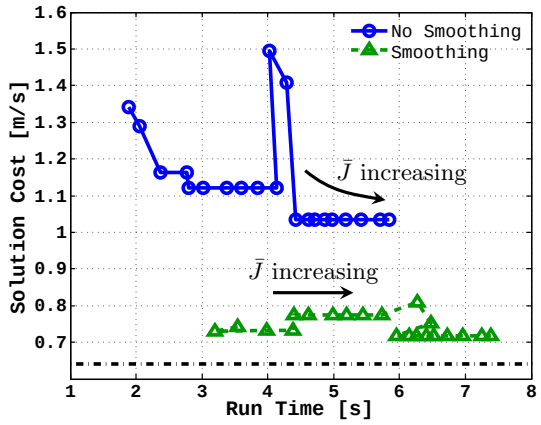
Results are reported in Figs. 8.8–8.9. Here we call FMT* once each for a series of sample count/cost threshold pairs, plotting the total cost of *successful* runs at their respective run times[2] as measured by wall clock time. Note only the *online* components of each FMT* call—*i.e.*, graph search/construction, constraint-checking, and termination evaluations—constitute the run times reported; everything else may either be stored onboard prior to mission launch or otherwise computed offline on ground computers and later uplinked to the spacecraft. See Section 7.3 for details. Samples are stored as a $d \times n$ array, while inter-sample steering controls $\mathbf{\Delta v}_i^*$ and times τ_i are precomputed as $n \times n$ arrays of $d/2 \times N$ and $N \times 1$ array elements, respectively. Steering state and attitude trajectories $\mathbf{x}^*(t)$ and $\mathbf{q}(t)$, on the other hand, are generated online through Eq. (5.4) and our nadir-pointing attitude policy, respectively. This reduces memory requirements, though nothing precludes them from being generated and stored offline as well, to save additional computation time.

Figure 8.8 reports the effects on solution cost from varying the cost threshold $\bar{J}$ while keeping n fixed. As described in Section 7.2, increasing $\bar{J}$ implies a larger reachability set size, and hence increases the number of candidate neighbors evaluated during graph construction. Generally, this gives a cost improvement at the expense of extra processing; though there are exceptions, as in Fig. 8.8a at $\bar{J} \approx 0.3$ m/s. Likely this arises from a single new neighbor (connected at the expense of another, since FMT* only adds one edge per neighborhood) that readjusts the entire graph subtree, ultimately increasing the cost of *exact* termination at the goal. Indeed, we see that this does not occur where inexact convergence is permitted, given the same sample distribution.

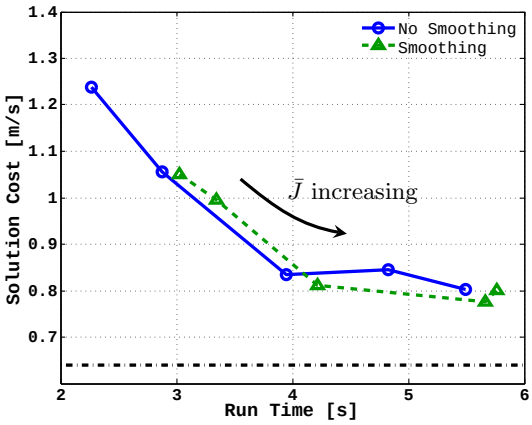
We can also vary the sample count n while holding $\bar{J}$ constant. From Figs. 8.8a–8.8b, we select $\bar{J} = 0.22$ m/s and 0.3 m/s, respectively, for each of the two cases (the values which suggest the best solution cost per unit of run time). Repeating the simulation for varying sample count values, we obtain Fig. 8.9. Note the general downward trend as run time increases (corresponding to larger sample counts)—a classical trade-off in sampling-based planning. However, there is bumpiness. Similar to before, this is likely due to new connections previously unavailable at lower sample counts that cause a slightly different graph with an unlucky jump in propellant cost. This reinforces that n and $\bar{J}$ should always be tuned before applying FMT*.

As the figures show, the utility of trajectory smoothing is clearly affected by the fidelity of the planning simulation. In each, trajectory smoothing yields a much larger improvement in cost at

[2]All simulations are implemented in MATLAB 2012b and run on a PC operated by Windows 10, clocked at 4.00 GHz, and equipped with 32.0 GB of RAM. CVXGEN and CVX [158], disciplined convex programming solvers, are used to implement Δv-allocation and trajectory smoothing, respectively.

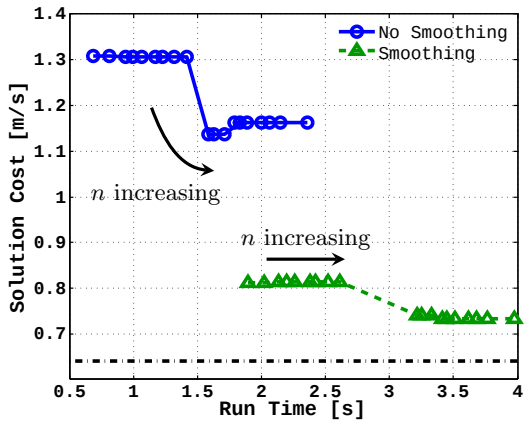


(a) Exact waypoint convergence ($n = 2000$)

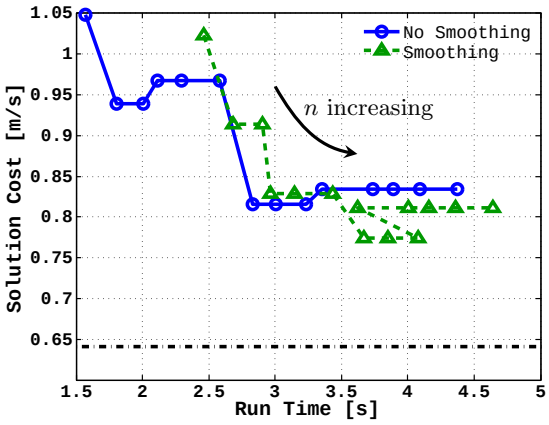


(b) Inexact waypoint convergence ($n = 2000$)

Figure 8.8: Algorithm performance for the given LEO proximity operations scenario as a function of cost threshold ($\bar{J} \in [0.2, 0.4]$) with n held constant (lowering $\bar{J}$ at these n yields failure). Results are reported for both: (i) trajectories constrained to rendezvous exactly with pre-specified waypoints, and (ii) trajectories free to terminate anywhere in $\mathcal{X}_{\text{goal}}$ (within position/velocity tolerances).



(a) Exact waypoint convergence ($\bar{J} = 0.22$ m/s).



(b) Inexact waypoint convergence ($\bar{J} = 0.3$ m/s).

Figure 8.9: Algorithm performance for the given LEO proximity operations scenario as a function of sample count ($n \in [650, 2000]$) with $\bar{J}$ held constant (lowering n further at these $\bar{J}$ yields failure). Results are reported for trajectories both with and without exact waypoint convergence.

modest increases in computation time when we require exact waypoint convergence. It provides little improvement, on the other hand, when we relax these waypoint tolerances; FMT* (with goal region sampling) seems to return trajectories with costs much closer to the optimum in such cases, making the additional overhead of smoothing less favorable. This conclusion is likely highly problem-dependent; these tools must always be tested and tuned to the particular application.

Note that the overall run times for each simulation are on the order of 1-5 seconds, including smoothing. This clearly indicates that FMT* can return high-quality solutions in real-time for spacecraft proximity operations. Though run on a computer currently unavailable to spacecraft, we hope that our examples serve as a reasonable proof-of-concept; we expect that with a more efficient coding language and implementation, our approach would be competitive on spacecraft hardware.

8.2 Conclusion

A technique has been presented for efficiently automating minimum-propellant guidance during near-circular orbit proximity operations, enabling the computation of near-optimal feasible trajectories in real time (on the order of 1–5 seconds for our numerical examples). The approach uses a modified version of the FMT* sampling-based motion planning algorithm to approximate the solution to the minimal-propellant trajectory control problem Eq. (5.1) under impulsive Clohessy-Wiltshire-Hill (CWH) dynamics. Our method begins by discretizing the feasible space of Eq. (5.1) through state space sampling in the CWH Local-Vertical Local-Horizontal (LVLH) frame. Next, state samples and their forward *reachability sets*, which we have shown comprise sets bounded by unions of ellipsoids taken over steering maneuver duration, are precomputed offline and stored onboard the spacecraft together with all pairwise steering solutions. Finally, the FMT* algorithm is called online to efficiently construct a tree of trajectories through the feasible state space towards a goal region, returning a solution that satisfies a broad range of trajectory constraints (*e.g.*, plume impingement and obstacle avoidance, control allocation feasibility, *etc.*) or else reporting failure. If desired, trajectory smoothing using the techniques outlined in Section 7.5 can be employed to reduce solution propellant cost.

The proposed planning framework for impulsively-actuated spacecraft offers several interesting avenues for future research. For example, though nothing in the methodology forbids it outside of computational limitations, it would be interesting to revisit the problem with attitude states included in the planning process (instead of abstracted away, as we have done here by assuming an attitude profile). This would allow direct inclusion of attitude constraints into maneuver planning (*e.g.*, enforcing line-of-sight, keeping solar panels oriented towards the Sun to stay power positive, maintaining a communication link between the chaser antenna and ground, *etc.*). Also of interest are other actively-safe policies that relax the need to circularize escape orbits (potentially costly in terms of propellant use) or which mesh better with trajectory smoothing, without the need to add compensating impulses (see Section 7.5). Extensions to dynamic obstacles (such as debris or

maneuvering spacecraft, which are unfixed in the LVLH frame), elliptical target orbits, higher-order gravitation, curvilinear coordinates, or dynamics under relative orbital elements also represent key research areas useful for extending applicability to more general maneuvers. Finally, memory and run time performance evaluations of our algorithms on space-like hardware are vital to assessing our method's true benefit to spacecraft planning in practice.

Chapter 9

Conclusions

The sampling-based planning framework proposed in this dissertation provides a new general, flexible way of designing provably-safe, propellant-efficient trajectories for impulsively-actuated spacecraft operating in close proximity to other objects. The approach consists of seven overall steps:

1. **Problem Formulation**: Formulate the mission trajectory constraints required for nominal and active Collision Avoidance Maneuver (CAM) trajectories.

2. **Steering Design**: Develop a steering function (*i.e.*, a Two-Point Boundary Value Problem solver) that is able to generate optimal (or at least, propellant-efficient) point-to-point reference state trajectories for the system dynamics under a *relaxed* version of the guidance problem. The fewer relaxations that are made, the faster constraint evaluation will be online and the better guided trajectory exploration will be; on the other hand, the advantage of the sampling-based framework is that the steering function can simply ignore any constraints that are too hard to solve directly or which cannot be known ahead of time (such as the positions of orbiting debris). If precomputation of actively-safe abort trajectories is desirable, the steering function should include impulses at only the initial and target states (so that safety can be evaluated purely at graph nodes instead of graph edges).

3. **CAM Policy Design**: Find a safe, stable positively-invariant set to use for termination of active abort maneuvers, preferably something that the designer knows cannot be invalidated during the mission by newly-discovered or time-varying obstacles or constraints. Re-evaluation of the set's safety may be needed online if this cannot be ensured.

4. **Precomputation Phase**: Precompute a set of samples from the state space and their corresponding pairwise steering trajectories; if abort safety mission constraints (such as Keep-Out Zones) are prescribed and time-invariant, then the *active safety* of abort trajectories under failures can be cached as well.

5. **Online Phase**: Storing only the steering trajectory costs, steering control trajectories, and abort safety control trajectories onboard the vehicle, compute online a feasible, low-cost trajectory using an asymptotically-optimal batch sampling-based planner like FMT* or BFMT*. Full state trajectories associated with steering and abort trajectories may be cached as well for additional speed-ups, if memory allows.

6. **Post-Processing Phase**: Smooth the resulting solution trajectory if further processing time is available.

7. **Feedback**: To account for state uncertainty, dynamic perturbations, and control disturbances during execution, follow the returned guidance trajectory using a stable feedback controller, possibly recomputing the solution in a receding-horizon fashion as one would do using Model Predictive Control (MPC).

The framework's major contribution is its ability to return online trajectories in a matter of seconds (or conceivably much faster, with improved coding implementations and/or parallelization), while simultaneously tending towards the minimum-propellant solution (in the limit that we take our state space discretization of n samples to infinity) and deterministically guaranteeing safety with respect to control failures. The key breakthrough of our solution for autonomous spacecraft guidance is its judicious distribution of computations; in essence, only what *must* be computed onboard, such as collision-checking and graph construction, is computed online—everything else, including the most intensive computations, are relegated to the ground where computational effort and execution time are less critical. Furthermore, only minimal information (steering problem control trajectories, costs, and nearest-neighbor set memberships) requires storage onboard the spacecraft. Though we have illustrated through numerical simulations the ability to tackle a particular minimum-propellant LEO near-field rendezvous problem, it should be noted that the methodology applies equally well to other objectives, such as the minimum-time problem, and can be generalized to other dynamic models and environments. The approach is flexible enough to handle non-convexity and mixed state-control-time constraints without compromising real-time implementability, so long as constraint function evaluation is relatively efficient. In short, the proposed approach appears to be useful for automating the mission planning process for spacecraft proximity operations, enabling real-time computation of low cost trajectories.

9.1 Summary

The dissertation began in Chapter 1 with a detailed study of the spacecraft proximity operations guidance problem, including a discussion of its importance to the science and engineering community at large and a simple analysis of some of its unique challenges. From there, we motivated our choice for sampling-based planning algorithms as a solution to this problem through highlights of recent

autonomous demonstration missions and a brief review of other state-of-the-art techniques. A broad introduction to sampling-based planning followed, setting the stage for the remainder of the thesis.

In Part I of the thesis, we designed and tested a new family of efficient batch sampling-based motion planning algorithms. In Chapter 3, we extended the unidirectional FMT* algorithm to bi-directional search and showed by our construction that this new algorithm, dubbed the Bi-directional Fast Marching Trees (BFMT*) algorithm, preserves the probabilistic exhaustivity, asymptotic optimality and convergence rate guarantees of its counterpart—arguably firsts in the field of bi-directional sampling-based planning. Numerical experiments in $\mathbb{R}^d$, $\mathbb{SE}(2)$, and $\mathbb{SE}(3)$ revealed that BFMT* tends to an optimal solution at least as fast as its state-of-the-art counterparts, and in some cases significantly faster. The addition of BFMT* gives trajectory designers an additional efficient tool for solving motion planning problems. We then demonstrated the performance of FMT* on a working robotic testbed, showcasing the algorithm's ability to solve a simulated autonomous rendezvous and docking problem on a planar, 3-DOF analogue to deep-space flight.

In Part II of the thesis, we described our proposed methodology for provably-safe, propellant-efficient autonomous guidance. Through Chapters 5–7, using near-circular orbit guidance as a running example, we demonstrated each of the steps of our solution framework as outlined in the chapter introduction, progressing in sequence through the formulation of our specific guidance problem, the design of our steering controller and one-burn CAM policy, and finally the presentation of our solution framework built from a modified version of the FMT* algorithm. Along the way, we characterized for the first time the form of forward-reachable sets under the sum-of-2-norms propellant-cost metric under Clohessy-Wiltshire-Hill dynamics, revealing that these sets are tightly upper- and lower-bounded by unions of ellipsoids taken over maneuver duration. Furthermore, we designed a general, "anytime"-capable algorithm for sampling-based trajectory smoothing that efficiently reduces solution trajectory costs in post-processing using a bisection technique. This smoothing algorithm can be applied to *any* Linear Time-Invariant (LTI) system with impulsively-actuated controls. Finally, we ended in Chapter 8 with a detailed trade study of FMT*'s online performance as applied to both planar and non-planar examples of a realistic, single-chaser single-target near-field rendezvous mission in near-circular orbit. Ultimately, we showed that the approach can return efficient, low-cost solutions in real-time, while handling a bevy of constraints like plume impingement and collision avoidance, control allocation feasibility with respect to a collection of distributed thrusters, and a 2-fault tolerance to "stuck-off" thruster failures.

9.2 Future Work

Though this dissertation has addressed a number of difficulties associated with real-time guidance for spacecraft proximity operations, the complexity of the field and the wide variety of imaginable missions leaves many avenues open for further study. Furthermore, as so few autonomous guidance

missions have been demonstrated at the time of this writing, the number of practical directions available for development are effectively limitless. In this section, we outline just a few of these important areas for future work.

Accounting for Disturbances and Uncertainty One of the most important elements of any GN&C system is the rigorous handling of dynamic perturbations together with actuation, localization, and measurement uncertainties. Throughout this work, we have neglected the instability and performance issues brought about by an uncertain state; we assume the initial and goal states $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ can be determined exactly once on orbit, along with the positions and velocities of any nearby objects, such as the states of any neighboring spacecraft or debris. In reality, orbit determination is highly uncertain and at best requires a certain amount of time for navigation filters to converge once targets (if cooperative) are in range to broadcast their relative state estimates. Addressing this state uncertainty, possibly by extending sampling-based planners to allow formal guarantees of optimality with respect to a finite-measure initial state set (such as an uncertainty ellipse), would be of critical importance for both theoretical and practical purposes. In addition, solving the question of how to handle discrepancies in control allocation due to thruster misfirings, canted thruster nozzles, imprecise manufacturing and assembly, uncertain burn times, friction, and so forth, would be vital before such planners could be used onboard any real-world application. Finally, and not unrelated to the concerns above, a formalized approach to imparting closed-loop feedback to our guidance algorithm, backed by theoretical analysis regarding its robustness and regions of convergence, would go a long way towards validating the technique with respect to dynamic perturbations and model uncertainties. A reasonable approach might be development of a dynamic version of FMT* and BFMT*, as we discuss in detail below, which efficiently reuses previous solutions upon repeated calls, which one might embed within a receding-horizon controller.

Extensions to Other Scenarios The numerical experiments in this thesis have demonstrated the performance of our approach on a simulated single-chaser, single-target near-field rendezvous maneuver in a near-circular orbit scenario. Though little impedes it from being applied to other scenarios, it remains to be seen how well the framework extends to other common proximity operations missions. Examples include propellant-optimized relative guidance near elliptic orbits, during final approach/on-orbit servicing, in deep-space (neglecting the effects of external gravitational influences), or in close proximity of celestial small bodies (such as comets, asteroids, and small moons) [159]. The latter case is particularly interesting, especially with regard to CAM policy design. The microgravitational environment near small bodies allows entirely new types of positively-invariant sets for termination of CAMs, including forced equilibrium points (*e.g.*, through body-fixed hovering[1]), low-Δv escape trajectories, or even near-inertial hovering maneuvers that take solar gravitation into

[1]Though forced equilibria are technically not positively-invariant under a limited propellant budget, they may be well-approximated as such because so little propellant is needed to maintain them per unit time.

account [160]. Because our abort safety policy design approach relies so heavily on designer intuition and mission specifics (CAM policies must be achievable under all possible failures of interest, or else actively-safe samples will never be discovered for online planning), studies are needed to determine which types of CAMs and corresponding positively-invariant sets are most appropriate to these kinds of new dynamic scenarios.

It could also be of interest to extend our guidance scheme from Linear Time-Invariant (LTI) dynamics to higher-fidelity dynamic models, to improve accuracy and allow longer guidance maneuver durations. Ideally this means accounting for perturbations like attractor oblateness, atmospheric drag, and third-body gravitational effects during the planning process, so that guidance algorithms can better predict true propellant costs (and therefore enable propellant savings). Alternatively, this could be achieved using an appropriate feedback strategy.

Finally, it would be useful to remove some of the limitations of our approach, where some of the theory breaks down. For example, the instantaneous thrust assumption made throughout our work greatly simplifies our dynamic equations, and in particular allows us to precompute our active safety evaluations if we restrict impulses to samples (edge endpoints) only—all edges then become coasting arcs that we know are passively-safe. Unfortunately, the instantaneous thrust assumption can be inaccurate during very short duration proximity operations maneuvers, or for systems with low thrust-to-weight ratios. Extensions to finite-time or continuous-time propulsion models would greatly improve simulation accuracies. The ability to accommodate low-thrust propulsion systems, including nuclear-electric and solar-electric propulsion, Hall/ion/plasma thrusters, and solar sails, would be particularly constructive for small-body proximity operations [161, 162]. In addition, though the theory allows multi-spacecraft planning (in which we embed the states that lead to collision with another maneuvering spacecraft within the obstacle region $\mathcal{X}_{\text{obs}}$), in practice this is extremely difficult to handle using planning alone; often game-theoretic principles are needed to predict other vehicles' motions. This problem coordinating fleets of independent spacecraft is a rich area for further investigation. Though collision avoidance has been examined extensively in the formation flying and fleet planning communities [163], little appears to have been done to simultaneously address fleet propellant-minimization, pointing constraints, plume impingement avoidance, control feasibility, *etc.*, which suggests that perhaps our sampling-based scheme could be beneficial.

Other Challenging Trajectory Constraints We have demonstrated in this thesis the ability to handle a number of difficult constraints, including collision and plume impingement avoidance, control allocation with impulse bit bounds, and a "stuck-off" thruster fault tolerance. However, it would also be of interest to introduce other constraints of the like mentioned in Section 1.1.3, including slew rate, relative pointing, solar array shadowing, and illumination or communication window timing constraints, all of which fortunately fall within the framework as is, but have yet to be demonstrated. One possibility for addressing attitude constraints might be full-state guidance (incorporating attitude and attitude rate with position and velocity in the state vector), as mentioned

in Section 8.2. Additionally, path integral constraints, of the form:

$$\mathbf{g}(\mathbf{x}, \mathbf{u}, t) = \int\limits_{\mathbf{x}(\tau), \tau \in [t_0, t]} \mathcal{L}(\mathbf{y}) \cdot \, \mathrm{d}\mathbf{y} = \int_{t_0}^{t} \mathcal{L}(\mathbf{x}(\tau)) \cdot \dot{\mathbf{x}}(\tau) \, \mathrm{d}\tau \leq 0$$

(where $\mathbf{u}$ is embedded implicitly in the state dynamics $\dot{\mathbf{x}}(\tau) = f(\mathbf{x}, \mathbf{u}, \tau)$) fit within our methodology as well, but likely require special handling within state-to-state steering in order to avoid wasted exploration along trajectories that accumulate path costs $\mathcal{L}$ too quickly. Accounting for such constraints is important, however, because they can represent momentum wheel saturation and power consumption limitations [83], both of which are critical factors in many spacecraft missions.

In addition, time-varying constraints, such as orbiting debris or suddenly-introduced obstacles like hazardous outgassing plumes, will require particularly special care [164], as our proposed methodology is specifically tailored for static environments. As we saw, when mission objectives and constraints are static and known beforehand—for example, when obstacles are not moving in the dynamic reference frames used to describe our states $\mathbf{x} \in \mathcal{X}$ and controls $\mathbf{u} \in \mathcal{U}$—more computations, like abort trajectory safety verification, may be precomputed in advance of deployment. This can save significant online computation time. However, precomputation may not always be possible. In some contexts, it may suffice to assume a static environment and re-plan sufficiently often using FMT* or BFMT* as is [165] (see also Section 4.2), or it may be appropriate to make conservative approximations, such as modeling moving obstacles as "velocity obstacles" [166, 167] (static obstacles constructed from the volume of the physical obstacle swept out over time). In other cases, using a dynamic version of FMT* and BFMT* that is able to revisit previously-computed trees and efficiently prune newly-invalidated edges may be the more functional, less-conservative solution (see [168–170] for notable examples of dynamic algorithms). Investigations are needed to determine which techniques are most effective in combination with our unique solution approach.

Another direction of important research is how to best accommodate constraints that are currently out of the scope of our framework, including logical constraints (different from our active safety constraints, which we accounted for explicitly) and risk constraints. Logical constraints encode high-level specifications into the mission planning process, such as "achieve all of the following once, in any order: refuel the target, charge batteries before Earth eclipse, and broadcast health information to ground." Risk constraints are probabilistic inequality expressions that allow trajectory designers to balance performance with uncertainty [171, 172]—a particularly useful tool in strictly risk-averse applications like spacecraft planning. In general, these kinds of constraints cannot currently be handled by the proposed framework because their evaluation cannot be broken-up piece-wise into a sequence of local subproblems; evaluating them over a sequence of connected graph edges cannot predict whether they will ultimately be able to meet the requirements of each risk or logical constraint until we have the full solution trajectory all the way to the goal.

Different Sampling Schemes In all of our numerical experiments, only the standard, low-discrepancy Halton sequence was used to generate sample sets. Though the sequence works well enough in our examples, performance gains could be made if sampling were able to take better account of our propellant-related cost functional. Low-discrepancy sequences excel at covering state spaces uniformly in terms of Euclidean distance, but it would be interesting and much more efficient if we could uniformly cover our cost-to-go space instead. A simple heuristic might be to *bias* our sample distribution towards states near coasting arcs, which may be far apart physically but which cost effectively nothing to reach. How to best achieve this remains an open question.

Different Cost Functionals This thesis has been primarily concerned with minimizing propellant consumption. It would be interesting to analyze the approach's performance under other types of cost functionals, such as the minimum-time problem or a mixed propellant and total burn count objective (to promote fewer numbers of burns, which can wreak havoc on navigation systems). Extensions to pareto-optimal guidance and multi-objective optimization would also be useful, as space mission trajectory designers must often trade-off between several guidance objectives at once.

Realistic Vehicle Simulations Finally, the true test for any algorithm is its performance on realistic space hardware, due to their strict computational constraints. As such, it is critical that these kinds of new, theoretical guidance strategies be verified through hardware-in-the-loop testing, including thorough investigations of their memory usage and run-time performance. Experiments are also needed to estimate state error magnitudes under typical trajectory-following controllers, to determine whether guidance trajectories are truly representative of dynamic feasibility. Though difficult to run within the gravitational well of Earth, possible testing options include: air-bearing robots [122, 173, 174] (see also Section 4.2), gantry systems [175], 6-DOF dynamic simulators driven by manipulator arms or hexapod platforms [176, 177], reduced-gravity aircraft, commercial suborbital flights (hopefully a reality in the near-future), and, finally, on-orbit demonstrations [178, 179].

9.3 Vision for the Future

As can be seen, much work remains to bring truly autonomous guidance and control into future spacecraft missions. It is hoped though that this thesis presents a significant incremental step in the right direction. With the flexibility and modularity of our real-time sampling-based planning framework, we envision a new *comprehensive* mission planning paradigm in which every phase of future spacecraft missions—from launch, to orbital operations, to planetary exploration—is guided by the same algorithmic framework. This would significantly ease the validation and verification process without compromising solution cost, safety, or computational efficiency, hopefully one day enabling entirely new mission capabilities never before achieved in spaceflight.

Appendix A

The Clohessy-Wiltshire-Hill Equations

Throughout this work, spacecraft proximity operations are specialized to the canonical scenario of a spacecraft (called the *chaser*) moving in the near-vicinity of another spacecraft (called the *target*) that is constrained to a circular orbit about a large gravitational attractor. Though the methods presented in this thesis may be easily generalized to other scenarios, we restrict our attention to this case due to its frequent occurrence in orbital mission planning. In this appendix, we derive the dynamic equations that govern the translational motions of such spacecraft during near-circular orbit proximity operations (Appendix A.1), and present their closed-form solution (Appendix A.2). These equations, known as the Clohessy-Wiltshire-Hill (CWH) equations [142, 143], are well-known and quite pervasive within the spacecraft control community. Though the derivation here is a reproduction, we include it to keep the thesis self-contained.

A.1 Derivation of the CWH Equations

To begin, let us make a few assumptions:

- There is only a single gravitational attractor that yields a spherically-symmetric gravitational field (inverse-square law gravitation).
- There are no forces exerted between the chaser and target (that is, we neglect their mutual gravitation and assume the spacecraft do not influence one another).
- The target spacecraft moves in a perfect circle about the attractor (a constraint which requires that the target exert zero net translational thrust at all times).

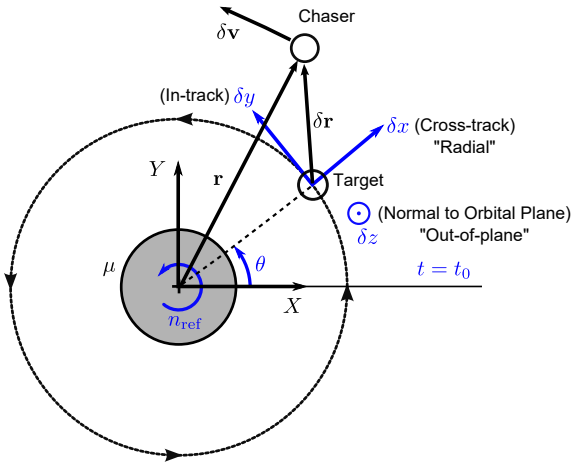


Figure A.1: Setup used to derive the Clohessy-Wiltshire-Hill (CWH) equations. The orbital motion of the chaser spacecraft is linearized about the circular orbit of the target spacecraft, and expressed in the Local-Vertical, Local-Horizontal (LVLH) frame associated with the moving target.

The scenario can be visualized in Fig. A.1. In essence, we assume that each spacecraft moves according to Restricted Two-Body Problem (Keplerian) dynamics (in the Earth-Centered Inertial (ECI) X-Y-Z frame), with the target constrained to perfect circular orbit (to which we attach a moving Local-Vertical, Local-Horizontal (LVLH) δx-δy-δz frame). The accuracy of these assumptions is limited by orbit insertion/determination errors and by orbital perturbations, including atmospheric drag, attractor oblateness, and third-body effects. For these reasons, the CWH equations are only useful for modeling short duration maneuvers, where "short" implies a sufficiently-small duration (typically a few target orbital periods, or less) such that these uncertainties and perturbations do not have time to drive predicted motions significantly far from truth.

To describe this mathematically, we have for the chaser spacecraft that:

$$\dot{\mathbf{p}} = -\frac{GMm}{r^3}\mathbf{r} + \mathbf{F}$$

using Newton's Law of Gravitation (and neglecting relativistic effects), where G is the universal gravitational constant, M is the attractor mass, m is the chaser mass, $\mathbf{p}$ is its linear momentum, and $\mathbf{F}$ is its applied thrust. Writing $\dot{\mathbf{p}}$ as $\dot{(m\mathbf{v})} = \dot{m}\mathbf{v} + m\dot{\mathbf{v}}$, let us assume that m is approximately constant (*i.e.*, that propellant usage during proximity operations is small compared to the total spacecraft mass, such that the term $\dot{m}\mathbf{v} \ll m\dot{\mathbf{v}}$, where $\ll$ is applied component-wise). Then we can re-express our dynamics as:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} + \frac{\mathbf{F}}{m} \tag{A.1}$$

where $\mu = GM$ is the gravitational parameter associated with the gravitational attractor.

It follows analogously for the target (*i.e.*, the LVLH frame origin, or reference point) that:

$$\ddot{\mathbf{r}}_{\text{ref}} = -\frac{\mu}{r_{\text{ref}}^3}\mathbf{r}_{\text{ref}} \tag{A.2}$$

Now, by our circular orbit assumption, r_{ref} is equal to a constant and the LVLH frame angular velocity satisfies $\boldsymbol{\omega}_{\text{ref}} = \dot{\theta}\hat{\mathbf{Z}} = \dot{\theta}\hat{\boldsymbol{\delta}}\mathbf{z}$. Combined with Fig. A.1, it follows that:

$$\begin{aligned}
\mathbf{r}_{\text{ref}} &= r_{\text{ref}}\hat{\mathbf{r}}_{\text{ref}} = r_{\text{ref}}\hat{\boldsymbol{\delta}}\mathbf{x} \\
\dot{\mathbf{r}}_{\text{ref}} &= \left(r_{\text{ref}}\hat{\boldsymbol{\delta}}\mathbf{x}\right)^{\cdot} = \cancelto{0}{\dot{r}_{\text{ref}}}\hat{\boldsymbol{\delta}}\mathbf{x} + r_{\text{ref}}\dot{\hat{\boldsymbol{\delta}}}\mathbf{x} = r_{\text{ref}}\dot{\theta}\hat{\boldsymbol{\delta}}\mathbf{y} \\
\ddot{\mathbf{r}}_{\text{ref}} &= \left(r_{\text{ref}}\ddot{\hat{\boldsymbol{\delta}}}\mathbf{x}\right)^{\cdot} = \cancelto{0}{\ddot{r}_{\text{ref}}}\hat{\boldsymbol{\delta}}\mathbf{x} + r_{\text{ref}}\ddot{\hat{\boldsymbol{\delta}}}\mathbf{x} = -r_{\text{ref}}\dot{\theta}^2\hat{\boldsymbol{\delta}}\mathbf{x}
\end{aligned}$$

where we used the facts that $\dot{\hat{\boldsymbol{\delta}}}\mathbf{x} = \boldsymbol{\omega}_{\text{ref}} \times \hat{\boldsymbol{\delta}}\mathbf{x} = \dot{\theta}\hat{\boldsymbol{\delta}}\mathbf{z} \times \hat{\boldsymbol{\delta}}\mathbf{x} = \dot{\theta}\hat{\boldsymbol{\delta}}\mathbf{y}$ and that $\ddot{\hat{\boldsymbol{\delta}}}\mathbf{x} = \boldsymbol{\omega}_{\text{ref}} \times \dot{\hat{\boldsymbol{\delta}}}\mathbf{x} = \dot{\theta}\hat{\boldsymbol{\delta}}\mathbf{z} \times \dot{\theta}\hat{\boldsymbol{\delta}}\mathbf{y} = -\dot{\theta}^2\hat{\boldsymbol{\delta}}\mathbf{x}$. Substituting into Eq. (A.2),

$$-r_{\text{ref}}\dot{\theta}^2\hat{\boldsymbol{\delta}}\mathbf{x} = -\frac{\mu}{r_{\text{ref}}^3}\left(r_{\text{ref}}\hat{\boldsymbol{\delta}}\mathbf{x}\right)$$

we see immediately that $\dot{\theta}^2 = \frac{\mu}{r_{\text{ref}}^3}$, which yields (taking only the positive solution):

$$n_{\text{ref}} \triangleq \dot{\theta} = \sqrt{\frac{\mu}{r_{\text{ref}}^3}} \tag{A.3}$$

where n_{ref}, a constant, is the *mean motion* of the target spacecraft orbit. Integrating, we find that:

$$\theta(t) = \int_{t_0}^{t} n_{\text{ref}}\,\mathrm{d}t = n_{\text{ref}}(t - t_0) \tag{A.4}$$

for the polar angle (true/eccentric/mean anomaly) of the target spacecraft.

Turning our attention to the chaser spacecraft, let us express the inertial dynamics of Eq. (A.1) in the target LVLH frame. These *relative dynamics* will give us the equations necessary for linearization. From Fig. A.1, note that $\mathbf{r} = \mathbf{r}_{\text{ref}} + \delta\mathbf{r}$. It follows that $\ddot{\mathbf{r}} = \ddot{\mathbf{r}}_{\text{ref}} + \ddot{\delta\mathbf{r}}$, which means we now need only find an expression for $\ddot{\delta\mathbf{r}}$. Let $\dot{(\cdot)}$ and $\mathring{(\cdot)}$ denote time derivatives in the inertial ECI and moving LVLH frames, respectively. Then we have, relating the two derivatives to one another, that:

$$\begin{aligned}
\dot{\delta\mathbf{r}} &= \mathring{\delta\mathbf{r}} + \boldsymbol{\omega}_{\text{ref}} \times \delta\mathbf{r} \\
\ddot{\delta\mathbf{r}} &= \left(\mathring{\delta\mathbf{r}} + \boldsymbol{\omega}_{\text{ref}} \times \delta\mathbf{r}\right)^{\cdot} = \dot{\mathring{\delta\mathbf{r}}} + (\boldsymbol{\omega}_{\text{ref}}^{\cdot} \times \delta\mathbf{r}) = \left(\mathring{\mathring{\delta\mathbf{r}}} + \boldsymbol{\omega}_{\text{ref}} \times \mathring{\delta\mathbf{r}}\right) + \left(\cancelto{\mathbf{0}}{\dot{\boldsymbol{\omega}}_{\text{ref}}} \times \delta\mathbf{r} + \boldsymbol{\omega}_{\text{ref}} \times \dot{\delta\mathbf{r}}\right) \\
&= \left(\mathring{\mathring{\delta\mathbf{r}}} + \boldsymbol{\omega}_{\text{ref}} \times \mathring{\delta\mathbf{r}}\right) + \left(\boldsymbol{\omega}_{\text{ref}} \times \mathring{\delta\mathbf{r}} + \boldsymbol{\omega}_{\text{ref}} \times (\boldsymbol{\omega}_{\text{ref}} \times \delta\mathbf{r})\right) \\
&= \mathring{\mathring{\delta\mathbf{r}}} + 2\boldsymbol{\omega}_{\text{ref}} \times \mathring{\delta\mathbf{r}} + \boldsymbol{\omega}_{\text{ref}} \times (\boldsymbol{\omega}_{\text{ref}} \times \delta\mathbf{r})
\end{aligned}$$

where we used that $\dot{\boldsymbol{\omega}}_{\text{ref}} = \left(n_{\text{ref}}\dot{\hat{\boldsymbol{\delta}}}\mathbf{z}\right)^{\cdot} = \dot{n}_{\text{ref}}\hat{\boldsymbol{\delta}}\mathbf{z} + n_{\text{ref}}\dot{\hat{\boldsymbol{\delta}}}\mathbf{z} = \mathbf{0}$, since both n_{ref} and $\hat{\boldsymbol{\delta}}\mathbf{z}$ are constant.

Resolving into the LVLH coordinate frame,

$$\delta\mathbf{r} = \begin{bmatrix}\delta x\\ \delta y\\ \delta z\end{bmatrix} \quad \delta\mathbf{v} = \mathring{\delta\mathbf{r}} = \begin{bmatrix}\dot{\delta x}\\ \dot{\delta y}\\ \dot{\delta z}\end{bmatrix} \quad \mathring{\mathring{\delta\mathbf{r}}} = \begin{bmatrix}\ddot{\delta x}\\ \ddot{\delta y}\\ \ddot{\delta z}\end{bmatrix} \quad \boldsymbol{\omega}_{\text{ref}} = \begin{bmatrix}0\\ 0\\ n_{\text{ref}}\end{bmatrix} \quad \mathbf{r} = \begin{bmatrix}r_{\text{ref}} + \delta x\\ \delta y\\ \delta z\end{bmatrix} \quad \mathbf{F} = \begin{bmatrix}F_{\delta x}\\ F_{\delta y}\\ F_{\delta z}\end{bmatrix}$$

we find that:

$$\begin{aligned}
\ddot{\mathbf{r}} &= \ddot{\mathbf{r}}_{\text{ref}} + \ddot{\delta\mathbf{r}} \\
&= \begin{bmatrix}-r_{\text{ref}}n_{\text{ref}}^2\\ 0\\ 0\end{bmatrix} + \left(\begin{bmatrix}\ddot{\delta x}\\ \ddot{\delta y}\\ \ddot{\delta z}\end{bmatrix} + 2\left(\begin{bmatrix}0\\ 0\\ n_{\text{ref}}\end{bmatrix} \times \begin{bmatrix}\dot{\delta x}\\ \dot{\delta y}\\ \dot{\delta z}\end{bmatrix}\right) + \begin{bmatrix}0\\ 0\\ n_{\text{ref}}\end{bmatrix} \times \left(\begin{bmatrix}0\\ 0\\ n_{\text{ref}}\end{bmatrix} \times \begin{bmatrix}\delta x\\ \delta y\\ \delta z\end{bmatrix}\right)\right) \\
&= \begin{bmatrix}-r_{\text{ref}}n_{\text{ref}}^2\\ 0\\ 0\end{bmatrix} + \left(\begin{bmatrix}\ddot{\delta x}\\ \ddot{\delta y}\\ \ddot{\delta z}\end{bmatrix} + 2\begin{bmatrix}-n_{\text{ref}}\dot{\delta y}\\ n_{\text{ref}}\dot{\delta x}\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 0\\ n_{\text{ref}}\end{bmatrix} \times \begin{bmatrix}-n_{\text{ref}}\delta y\\ n_{\text{ref}}\delta x\\ 0\end{bmatrix}\right) \\
&= \begin{bmatrix}\ddot{\delta x} - 2n_{\text{ref}}\dot{\delta y} - n_{\text{ref}}^2(r_{\text{ref}} + \delta x)\\ \ddot{\delta y} + 2n_{\text{ref}}\dot{\delta x} - n_{\text{ref}}^2\delta y\\ \ddot{\delta z}\end{bmatrix}
\end{aligned}$$

Equating with Eq. (A.1), and rearranging:

$$\begin{bmatrix}\ddot{\delta x}\\ \ddot{\delta y}\\ \ddot{\delta z}\end{bmatrix} = -\frac{\mu}{r^3}\begin{bmatrix}r_{\text{ref}} + \delta x\\ \delta y\\ \delta z\end{bmatrix} + \begin{bmatrix}2n_{\text{ref}}\dot{\delta y} + n_{\text{ref}}^2(r_{\text{ref}} + \delta x)\\ -2n_{\text{ref}}\dot{\delta x} + n_{\text{ref}}^2\delta y\\ 0\end{bmatrix} + \frac{1}{m}\begin{bmatrix}F_{\delta x}\\ F_{\delta y}\\ F_{\delta z}\end{bmatrix} \tag{A.5}$$

we obtain the non-linear equations of motion for the chaser spacecraft in the LVLH frame, where $r = \sqrt{(r_{\text{ref}} + \delta x)^2 + \delta y^2 + \delta z^2}$.

To linearize these equations of motion, we must first put them into first-order form. Define our state $\mathbf{x}$ as the relative position and velocity vector, $\mathbf{x} \triangleq \left[\,\delta\mathbf{r}^{\mathrm{T}}, \delta\mathbf{v}^{\mathrm{T}}\,\right]^{\mathrm{T}}$, and let our control $\mathbf{u}$ be the specific force (force per unit mass), $\mathbf{u} \triangleq \mathbf{F}/m$. Then:

$$\dot{\mathbf{x}} = \begin{bmatrix}\delta\mathbf{v}\\ \mathbf{f}(\mathbf{x}, \mathbf{u})\end{bmatrix} \tag{A.6}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is the right-hand side of Eq. (A.5). We aim to linearize Eq. (A.6) about the equilibrium point at the LVLH origin ($\mathbf{x}_{\text{eq}} = \mathbf{0}$ with $\mathbf{u}_{\text{eq}} = \mathbf{0}$), corresponding to the unforced circular orbit of the target. Restricting our attention to only the velocity equations (as the position dynamics are already

linear),

$$\mathbf{f}(\mathbf{x},\mathbf{u}) \cong \mathbf{f}(\mathbf{x}_{\text{eq}},\mathbf{u}_{\text{eq}}) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{eq}},\mathbf{u}_{\text{eq}}} (\mathbf{x}-\mathbf{x}_{\text{eq}}) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{x}_{\text{eq}},\mathbf{u}_{\text{eq}}} (\mathbf{u}-\mathbf{u}_{\text{eq}}) \tag{A.7}$$

Using Eq. (A.5), it is a straightforward exercise to show that the Jacobian matrices $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$ can be expressed in the LVLH frame as:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} -\frac{\mu}{r^3} + \frac{3\mu(r_{\text{ref}}+\delta x)^2}{r^5} + n_{\text{ref}}^2 & \frac{3\mu(r_{\text{ref}}+\delta x)\delta y}{r^5} & \frac{3\mu(r_{\text{ref}}+\delta x)\delta z}{r^5} & 0 & 2n_{\text{ref}} & 0 \\ \frac{3\mu\delta y(r_{\text{ref}}+\delta x)}{r^5} & -\frac{\mu}{r^3} + \frac{3\mu\delta y^2}{r^5} + n_{\text{ref}}^2 & \frac{3\mu\delta y\delta z}{r^5} & -2n_{\text{ref}} & 0 & 0 \\ \frac{3\mu\delta z(r_{\text{ref}}+\delta x)}{r^5} & \frac{3\mu\delta z\delta y}{r^5} & -\frac{\mu}{r^3} + \frac{3\mu\delta z^2}{r^5} & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Substituting into Eq. (A.7) and evaluating at $\mathbf{x}_{\text{eq}} = \mathbf{u}_{\text{eq}} = \mathbf{0}$, we find that $\mathbf{f}(\mathbf{x}_{\text{eq}},\mathbf{u}_{\text{eq}}) = \mathbf{0}$ and hence:

$$\mathbf{f}(\mathbf{x},\mathbf{u}) \cong \begin{bmatrix} 3n_{\text{ref}}^2 & 0 & 0 & 0 & 2n_{\text{ref}} & 0 \\ 0 & 0 & 0 & -2n_{\text{ref}} & 0 & 0 \\ 0 & 0 & -n_{\text{ref}}^2 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u}$$

Substituting into Eq. (A.6), we obtain, finally, the linear *Clohessy-Wiltshire-Hill (CWH) equations*:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n_{\text{ref}}^2 & 0 & 0 & 0 & 2n_{\text{ref}} & 0 \\ 0 & 0 & 0 & -2n_{\text{ref}} & 0 & 0 \\ 0 & 0 & -n_{\text{ref}}^2 & 0 & 0 & 0 \end{bmatrix}}_{\triangleq \mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\triangleq \mathbf{B}} \mathbf{u} \tag{A.8}$$

where $\mathbf{x} = [\delta x, \delta y, \delta z, \delta\dot{x}, \delta\dot{y}, \delta\dot{z}]^{\text{T}}$ are the position and velocity coordinates of the chaser as expressed in the LVLH frame, $\mathbf{u} = \frac{1}{m}[F_{\delta x}, F_{\delta y}, F_{\delta z}]^{\text{T}}$ is the specific thrust of the chaser resolved in the LVLH frame, and n_{ref} is the mean motion of the target spacecraft orbit, as given by Eq. (A.3).

A.2 Analytical Solutions to the CWH Equations

The CWH equations represented by Eq. (A.8) admit a closed-form solution. To derive it, we make use of the Laplace Transform $\mathcal{L}$ as well as its inverse $\mathcal{L}^{-1}$, defined for causal systems as:

$$\mathcal{L}(f(t)) = F(s) = \int_{t_0}^{\infty} e^{-st} f(t)\,\mathrm{d}t \tag{A.9a}$$

$$\mathcal{L}^{-1}(F(s)) = f(t) = \frac{1}{2\pi j} \lim_{T\to\infty} \int_{\gamma-jT}^{\gamma+jT} e^{st} F(s)\,\mathrm{d}s \tag{A.9b}$$

where γ is a real number large enough to ensure that the vertical line of integration $\mathrm{Re}(s) = \gamma$ in the complex plane lies to the right of any singularities of $F(s)$. As we will see, these two operations turn out to be very useful in simplifying ordinary differential equation (ODE) expressions.

We begin by applying the Laplace transform to both sides of Eq. (A.8):

$$\begin{aligned}
\mathcal{L}(\dot{\mathbf{x}}(t)) &= \mathcal{L}(\mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)) \\
s\,\mathcal{L}(\mathbf{x}(t)) - e^{-st_0}\mathbf{x}(t_0) &= \mathbf{A}\,\mathcal{L}(\mathbf{x}(t)) + \mathbf{B}\,\mathcal{L}(\mathbf{u}(t)) \\
(s\mathbf{I} - \mathbf{A})\mathbf{X}(s) &= e^{-st_0}\mathbf{x}(t_0) + \mathbf{B}\mathbf{U}(s) \\
\mathbf{X}(s) &= (s\mathbf{I} - \mathbf{A})^{-1} e^{-st_0}\mathbf{x}(t_0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)
\end{aligned}$$

where on the second line we used integration-by-parts (for the left-hand-side) and the linearity of the Laplace Transform (for the right-hand-side). If we now apply the Inverse Laplace Transform, we find:

$$\mathbf{x}(t) = \mathcal{L}^{-1}(\mathbf{X}(s)) = \mathcal{L}^{-1}\big((s\mathbf{I} - \mathbf{A})^{-1} e^{-st_0}\big)\mathbf{x}(t_0) + \mathcal{L}^{-1}\big((s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)\big)$$

where we again rely on linearity to simplify the right-hand-side. Now, let $\boldsymbol{\varphi}(s)$ represent the Laplace Transform of some time-domain function $\boldsymbol{\Phi}(t)$. We make use of two well-known facts: (i) the Inverse Laplace Transform of the exponential e^{-st_0} times $\boldsymbol{\varphi}(s)$ creates a shift in the time-domain $\mathcal{L}^{-1}(\boldsymbol{\varphi}(s)e^{-st_0}) = \boldsymbol{\Phi}(t - t_0)$, and (ii) the Inverse Laplace Transform of the product of two s-domain functions $\boldsymbol{\varphi}(s)$ and $\mathbf{G}(s)$ is given by $\mathcal{L}^{-1}(\boldsymbol{\varphi}(s)\mathbf{G}(s)) = \boldsymbol{\Phi}(t) * \mathbf{g}(t)$, a convolution integral in the time domain (verification of both properties is a straightforward exercise in calculus that we do not show here). Setting $\boldsymbol{\varphi}(s) = (s\mathbf{I} - \mathbf{A})^{-1}$ and $G(s) = B\mathbf{U}(s)$ in this case, we obtain:

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t - t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t - \tau)\mathbf{B}\mathbf{u}(\tau)\,\mathrm{d}\tau \tag{A.10}$$

Equation (A.10) reveals that to find the solution $\mathbf{x}(t)$ to our dynamic equations, we need only determine $\boldsymbol{\Phi}(t) = \mathcal{L}^{-1}\big((s\mathbf{I} - \mathbf{A})^{-1}\big)$. Note that the argument, $(s\mathbf{I} - \mathbf{A})^{-1}$, called the *resolvent* of $\mathbf{A}$, is defined at all $s \in \mathbb{C}$ except for the eigenvalues of $\mathbf{A}$. If we re-express the resolvent in terms of its

power series,

$$(s\mathbf{I}-\mathbf{A})^{-1}=\frac{1}{s}\left(\mathbf{I}-\frac{\mathbf{A}}{s}\right)^{-1}=\frac{1}{s}\left(\mathbf{I}+\frac{\mathbf{A}}{s}+\left(\frac{\mathbf{A}}{s}\right)^{2}+\ldots\right)=\frac{\mathbf{I}}{s}+\frac{\mathbf{A}}{s^2}+\frac{\mathbf{A}^2}{s^3}+\ldots$$

(valid at all s values with $|s|$ sufficiently-large and far from the eigenvalues of $\mathbf{A}$), we obtain an expression that is easier to evaluate inside the Inverse Laplace Transform:

$$\mathbf{\Phi}(t)=\mathcal{L}^{-1}\left((s\mathbf{I}-\mathbf{A})^{-1}\right)=\mathbf{I}+t\mathbf{A}+\frac{(t\mathbf{A})^2}{2!}+\ldots=\sum_{k=0}^{\infty}\frac{(t\mathbf{A})^k}{k!}\triangleq e^{t\mathbf{A}} \tag{A.11}$$

where $e^{t\mathbf{A}}$ is called the *matrix exponential* of $\mathbf{A}$. Putting Eq. (A.10) and Eq. (A.11) together, the solution to our system dynamics (applicable, in fact, to any LTI system) is therefore given by:

$$\mathbf{x}(t)=e^{\mathbf{A}(t-t_0)}\mathbf{x}(t_0)+\int_{t_0}^{t}e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\,\mathrm{d}\tau \tag{A.12}$$

Hence all that remains to compute a closed-form expression for the CWH dynamic equations is to derive an analytical expression for the matrix $\mathbf{\Phi}(t)=e^{\mathbf{A}t}=\mathcal{L}^{-1}\left((s\mathbf{I}-\mathbf{A})^{-1}\right)$, called the *state transition matrix* of our system. Rather than evaluating the infinite sum in Eq. (A.11), it actually turns out to be much simpler to use the inverse Laplace Transform directly on the elements of the resolvent of $\mathbf{A}$. Substituting the CWH system matrix $\mathbf{A}$, shown in Eq. (A.8),

$$(s\mathbf{I}-\mathbf{A})=\begin{bmatrix} s & 0 & 0 & -1 & 0 & 0\\ 0 & s & 0 & 0 & -1 & 0\\ 0 & 0 & s & 0 & 0 & -1\\ -3n_{\text{ref}}^2 & 0 & 0 & s & -2n_{\text{ref}} & 0\\ 0 & 0 & 0 & 2n_{\text{ref}} & s & 0\\ 0 & 0 & n_{\text{ref}}^2 & 0 & 0 & s \end{bmatrix}$$

$$(s\mathbf{I}-\mathbf{A})^{-1}=\begin{bmatrix} \frac{4n_{\text{ref}}^2+s^2}{s\left(n_{\text{ref}}^2+s^2\right)} & 0 & 0 & \frac{1}{n_{\text{ref}}^2+s^2} & \frac{2n_{\text{ref}}}{s\left(n_{\text{ref}}^2+s^2\right)} & 0\\ \frac{-6n_{\text{ref}}^3}{s^2\left(n_{\text{ref}}^2+s^2\right)} & \frac{1}{s} & 0 & \frac{-2n_{\text{ref}}}{s\left(n_{\text{ref}}^2+s^2\right)} & \frac{-3n_{\text{ref}}^2+s^2}{s^2\left(n_{\text{ref}}^2+s^2\right)} & 0\\ 0 & 0 & \frac{s}{n_{\text{ref}}^2+s^2} & 0 & 0 & \frac{1}{n_{\text{ref}}^2+s^2}\\ \frac{3n_{\text{ref}}^2}{n_{\text{ref}}^2+s^2} & 0 & 0 & \frac{s}{n_{\text{ref}}^2+s^2} & \frac{2n_{\text{ref}}}{n_{\text{ref}}^2+s^2} & 0\\ \frac{-6n_{\text{ref}}^3}{s\left(n_{\text{ref}}^2+s^2\right)} & 0 & 0 & \frac{-2n_{\text{ref}}}{n_{\text{ref}}^2+s^2} & \frac{-3n_{\text{ref}}^2+s^2}{s\left(n_{\text{ref}}^2+s^2\right)} & 0\\ 0 & 0 & \frac{-n_{\text{ref}}^2}{n_{\text{ref}}^2+s^2} & 0 & 0 & \frac{s}{n_{\text{ref}}^2+s^2} \end{bmatrix}$$

it now becomes a straightforward matter of applying the inverse Laplace Transform to each element of $(s\mathbf{I}-\mathbf{A})^{-1}$ (easily achieved via a combination of partial fraction decompositions and a table of elementary Laplace transforms). Introducing $\theta(t)=n_{\text{ref}}t$ as the polar angle (true anomaly) of the

target spacecraft, the result (once all the dust settles) turns out to be:

$$\mathbf{\Phi}(t) = \begin{bmatrix} 4-3\cos\theta & 0 & 0 & \frac{1}{n_{\text{ref}}}\sin\theta & \frac{2}{n_{\text{ref}}}(1-\cos\theta) & 0 \\ 6\sin\theta-6\theta & 1 & 0 & \frac{2}{n_{\text{ref}}}(\cos\theta-1) & \frac{1}{n_{\text{ref}}}(4\sin\theta-3\theta) & 0 \\ 0 & 0 & \cos\theta & 0 & 0 & \frac{1}{n_{\text{ref}}}\sin\theta \\ 3n_{\text{ref}}\sin\theta & 0 & 0 & \cos\theta & 2\sin\theta & 0 \\ 6n_{\text{ref}}(\cos\theta-1) & 0 & 0 & -2\sin\theta & 4\cos\theta-3 & 0 \\ 0 & 0 & -n_{\text{ref}}\sin\theta & 0 & 0 & \cos\theta \end{bmatrix} \tag{A.13}$$

To make our solution explicit for impulsive dynamics resolved in the CWH frame (the rotating Local-Vertical, Local-Horizontal frame of the target spacecraft), suppose that we apply the control trajectory $\mathbf{u}(t) = \sum_{i=1}^{N} \mathbf{\Delta v}_i \delta(t-\tau_i)$ to our chaser spacecraft, comprising N impulses $\mathbf{\Delta v}_i = [\,\Delta v_{\delta x,i}, \Delta v_{\delta y,i}, \Delta v_{\delta z,i}\,]^{\mathrm{T}}$ at burn times τ_i for $i \in [1,\ldots,N]$. Define the chaser initial state as $\mathbf{x}(t_0) = \mathbf{x}_0 = [\,\delta x_0, \delta y_0, \delta z_0, \delta\dot{x}_0, \delta\dot{y}_0, \delta\dot{z}_0\,]^{\mathrm{T}}$. Then the chaser state transition equations are given by:

$$\delta x(t) = (4-3\cos\theta)\delta x_0 + \left(\frac{1}{n_{\text{ref}}}\sin\theta\right)\delta\dot{x}_0 + \left(\frac{2}{n_{\text{ref}}}(1-\cos\theta)\right)\delta\dot{y}_0 + \sum_{i=1}^{N_t}\left[\left(\frac{1}{n_{\text{ref}}}\sin\theta_i\right)\Delta v_{\delta x,i} + \left(\frac{2}{n_{\text{ref}}}(1-\cos\theta_i)\right)\Delta v_{\delta y,i}\right] \tag{A.14a}$$

$$\delta y(t) = (6\sin\theta-6\theta)\delta x_0 + \delta y_0 + \left(\frac{2}{n_{\text{ref}}}(\cos\theta-1)\right)\delta\dot{x}_0 + \left(\frac{1}{n_{\text{ref}}}(4\sin\theta-3\theta)\right)\delta\dot{y}_0 + \sum_{i=1}^{N_t}\left[\left(\frac{2}{n_{\text{ref}}}(\cos\theta_i-1)\right)\Delta v_{\delta x,i} + \left(\frac{1}{n_{\text{ref}}}(4\sin\theta_i-3\theta_i)\right)\Delta v_{\delta y,i}\right] \tag{A.14b}$$

$$\delta z(t) = (\cos\theta)\delta z_0 + \left(\frac{1}{n_{\text{ref}}}\sin\theta\right)\delta\dot{z}_0 + \sum_{i=1}^{N_t}\left(\frac{1}{n_{\text{ref}}}\sin\theta_i\right)\Delta v_{\delta z,i} \tag{A.14c}$$

$$\delta\dot{x}(t) = (3n_{\text{ref}}\sin\theta)\delta x_0 + (\cos\theta)\delta\dot{x}_0 + (2\sin\theta)\delta\dot{y}_0 + \sum_{i=1}^{N_t}[(\cos\theta_i)\Delta v_{\delta x,i} + (2\sin\theta_i)\Delta v_{\delta y,i}] \tag{A.14d}$$

$$\delta\dot{y}(t) = (6n_{\text{ref}}(\cos\theta-1))\delta x_0 + (-2\sin\theta)\delta\dot{x}_0 + (4\cos\theta-3)\delta\dot{y}_0 + \sum_{i=1}^{N_t}[(-2\sin\theta_i)\Delta v_{\delta x,i} + (4\cos\theta_i-3)\Delta v_{\delta y,i}] \tag{A.14e}$$

$$\delta\dot{z}(t) = (-n_{\text{ref}}\sin\theta)\delta z_0 + (\cos\theta)\delta\dot{z}_0 + \sum_{i=1}^{N_t}(\cos\theta_i)\Delta v_{\delta z,i} \tag{A.14f}$$

where $\theta = n_{\text{ref}}(t-t_0)$, $\theta_i = n_{\text{ref}}(t-\tau_i)$, and $N_t = \sum_{i=1}^{N} \mathbb{1}[\tau_i \le t]$ is the number of burns occurring at or before time t.

These equations represent the expanded form of Eq. (5.4), originally presented in Chapter 5 during the introduction of the CWH equations. Having such closed-formed solutions is useful for

propagating state trajectories $\mathbf{x}(t)$ given steering control laws $\mathbf{u}(t)$, such as those returned by Eq. (7.3) or Eq. (7.6), without requiring numerical computation of the state transition matrix $\mathbf{\Phi}(t)$. Better still, the expressions in Eq. (A.14) are exact and can be computed very efficiently—and even faster when the trigonometric terms $\cos\theta$, $\sin\theta_i$, *etc.* are cached or tabulated. Note that we employ Eq. (A.14) in Chapter 8 to quickly propagate trajectory edges in the FMT* graph (see Algorithm 6) as well as to generate the unconstrained minimal-propellant solution trajectory required for our smoothing technique (see Algorithm 7).

Appendix B

Optimal Circularization Under Impulsive CWH Dynamics

As detailed in Section 6.3.1, a vital component of our Collision Avoidance Manuever (CAM) policy is the generation of one-burn minimal-propellant transfers to circular orbits located radially above or below the target. Assuming we need to abort from some state $\mathbf{x}(t_{\text{fail}}) = \mathbf{x}_{\text{fail}}$, the problem we wish to solve in order to assure safety (per Definition 16 and as seen in Fig. 6.5) is:

Given:	Failure state $\mathbf{x}_{\text{fail}}$, and CAM $\mathbf{u}_{\text{CAM}}\big(t_{\text{fail}} \leq t < T_h^-\big) \triangleq \mathbf{0}, \mathbf{u}_{\text{CAM}}(T_h) \triangleq \mathbf{\Delta v}_{\text{circ}}(\mathbf{x}(T_h))$	
$\underset{T_h}{\text{minimize}}$	$\Delta v_{\text{circ}}^2(T_h)$	
subject to	$\mathbf{x}_{\text{CAM}}(t_{\text{fail}}) = \mathbf{x}_{\text{fail}}$	Initial Condition
	$\mathbf{x}_{\text{CAM}}(T_h^+) \in \mathcal{X}_{\text{invariant}}$	Invariant Set Termination
	$\dot{\mathbf{x}}_{\text{CAM}}(t) = f(\mathbf{x}_{\text{CAM}}(t), \mathbf{0}, t)$, for all $t_{\text{fail}} \leq t \leq T_h$	System Dynamics
	$\mathbf{x}_{\text{CAM}}(t) \notin \mathcal{X}_{\text{KOZ}}$, for all $t_{\text{fail}} \leq t \leq T_h$	KOZ Collision Avoidance

where T_h is the future time at which to implement an abort circularization burn $\mathbf{\Delta v}_{\text{circ}}$, Δv_{circ} is the burn magnitude, $\mathcal{X}_{\text{KOZ}}$ is the target Keep-Out Zone (KOZ) ellipsoid (Eq. (6.2)), and $\mathcal{X}_{\text{invariant}}$ is the set of orbits whose projections are circles with radii either above or below $\mathcal{X}_{\text{KOZ}}$ (Eq. (6.4)).

Due to the analytical descriptions of state transitions, as given by Eq. (5.4) (or more explicitly by Eq. (A.14)), it is a straightforward task to express the decision variable T_h, invariant set constraint, and objective function analytically in terms of $\theta(t) = n_{\text{ref}}(t - t_{\text{fail}})$, the polar angle of the target spacecraft. Using Corollary 18, keeping in mind we need only circularize the planar projection of our

orbit per Remark 21, we find that:

$$\begin{aligned}\Delta v_{\text{circ}}^2(\theta) = &\left[(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})^2 + \frac{1}{4}\delta\dot{x}_{\text{fail}}^2 + n_{\text{ref}}^2\delta z_{\text{fail}}^2\right]\sin^2\theta \\ &+ \left[\frac{1}{4}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})^2 + \delta\dot{x}_{\text{fail}}^2 + \delta\dot{z}_{\text{fail}}^2\right]\cos^2\theta \\ &+ \left[\frac{3}{4}\delta\dot{x}_{\text{fail}}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}}) - n_{\text{ref}}\delta\dot{z}_{\text{fail}}\delta z_{\text{fail}}\right]\sin 2\theta\end{aligned} \tag{B.1}$$

The problem is therefore one-dimensional in terms of θ. We can reduce the invariant set termination constraint to an invariant set positioning constraint if we ensure the spacecraft ends up at a position inside $\mathcal{X}_{\text{invariant}}$ and circularize the orbit, since $\mathbf{x}(\theta_{\text{circ}}^+) = \mathbf{x}(\theta_{\text{circ}}^-) + \left[\begin{smallmatrix}\mathbf{0}\\ \boldsymbol{\Delta}\mathbf{v}_{\text{circ}}(\theta_{\text{circ}})\end{smallmatrix}\right] \in \mathcal{X}_{\text{invariant}}$. Denote $\theta_{\text{circ}} = n_{\text{ref}}(T_h - t_{\text{fail}})$ as the target anomaly at which we enforce circularization. Now, suppose the failure state $\mathbf{x}_{\text{fail}}$ lies outside of the KOZ (otherwise there exists no safe CAM, and we conclude that $\mathbf{x}_{\text{fail}}$ is unsafe). We can define a new variable $\theta_{\min} = n_{\text{ref}}(t - t_{\text{fail}})$ and integrate the coasting dynamics forward from time t_{fail} until the chaser touches the boundary of the KOZ ($\theta_{\max} = \theta_{\text{collision}}^-$) or until we have reached one full orbit ($\theta_{\max} = \theta_{\min} + 2\pi$) such that, between these two bounds, the CAM trajectory satisfies the dynamics and contains only the coasting segment outside of the KOZ. Replacing the dynamics and collision avoidance constraints with the bounds on θ as a box constraint, the problem becomes:

$$\begin{aligned}\underset{\theta_{\text{circ}}}{\text{minimize}} \quad & \Delta v_{\text{circ}}^2(\theta_{\text{circ}}) && \\ \text{subject to} \quad & \theta_{\min} \le \theta_{\text{circ}} \le \theta_{\max} && \text{Theta Bounds} \\ & \delta x^2\left(\theta_{\text{circ}}^-\right) \ge \rho_{\delta x}^2 && \text{Invariant Set Positioning}\end{aligned} \tag{B.2}$$

Restricting our search range to $\theta \in [\theta_{\min}, \theta_{\max}]$, our problem reduces to minimizing a function of one variable subject to one constraint, something we can easily optimize analytically using the method of Lagrange multipliers. To solve, we seek to minimize the Lagrangian, $\mathcal{L} = \Delta v_{\text{circ}}^2 + \lambda g_{\text{circ}}$, where $g_{\text{circ}}(\theta) = \rho_{\delta x}^2 - \delta x^2\left(\theta_{\text{circ}}^-\right)$. There are two cases to consider:

Case 1: Inactive Invariant Set Positioning Constraint

We set $\lambda = 0$ such that $\mathcal{L} = \Delta v_{\text{circ}}^2$, corresponding to the case that circularization takes place radially above or below the radial span of the target KOZ. Candidate optimizers θ^* must satisfy $\nabla_\theta\mathcal{L}(\theta^*) = 0$. Taking the gradient of $\mathcal{L}$,

$$\begin{aligned}\nabla_\theta\mathcal{L} = \frac{\partial\Delta v_{\text{circ}}^2}{\partial\theta} = &\left[\frac{3}{4}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})^2 - \frac{3}{4}\delta\dot{x}_{\text{fail}}^2 + n_{\text{ref}}^2\delta z_{\text{fail}}^2 - \delta\dot{z}_{\text{fail}}^2\right]\sin 2\theta \\ &+ \left[\frac{3}{2}\delta\dot{x}_{\text{fail}}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}}) - 2n_{\text{ref}}\delta\dot{z}_{\text{fail}}\delta z_{\text{fail}}\right]\cos 2\theta\end{aligned}$$

and setting $\nabla_\theta \mathcal{L}(\theta^*) = 0$, we find that:

$$\tan 2\theta^* = \frac{-\left(\frac{3}{2}\delta\dot{x}_{\text{fail}}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}}) - 2n_{\text{ref}}\delta\dot{z}_{\text{fail}}\delta z_{\text{fail}}\right)}{\frac{3}{4}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})^2 - \frac{3}{4}\delta\dot{x}_{\text{fail}}^2 + n_{\text{ref}}^2\delta z_{\text{fail}}^2 - \delta\dot{z}_{\text{fail}}^2}$$

Denote the set of candidate solutions that satisfy Case 1 by Θ_1^*. In the general case, this does not admit an analytical solution (its roots must be found numerically). Interestingly, however, there are special cases:

- **Circularizing a coast maneuver from rest**: Setting $\delta\dot{x}_{\text{fail}} = \delta\dot{y}_{\text{fail}} = 0$, we find $\tan 2\theta^* = 0$, hence $\theta^* = k\frac{\pi}{2}$ for all non-negative integers $k \in \mathbb{Z}_+$. This means the best time to circularize our orbit is at any multiple of quarter-orbit periods after our failures occurs.

- **Circularizing a planar coast maneuver**: Setting $\delta z_{\text{fail}} = \delta\dot{z}_{\text{fail}} = 0$, we obtain:

$$\tan 2\theta^* = \frac{-\left(\frac{3}{2}\delta\dot{x}_{\text{fail}}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})\right)}{\frac{3}{4}(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})^2 - \frac{3}{4}\delta\dot{x}_{\text{fail}}^2} = \frac{2(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})(-\delta\dot{x}_{\text{fail}})}{(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})^2 - \delta\dot{x}_{\text{fail}}^2}$$

  If we let $a \triangleq (3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})$ and $b \triangleq -\delta\dot{x}_{\text{fail}}$, then from the double-angle identity $\tan^{-1}\left(\frac{2ab}{a^2-b^2}\right) = 2\tan^{-1}\left(\frac{b}{a}\right)$ it follows that $\theta^* = \tan^{-1}\left(\frac{-\delta\dot{x}_{\text{fail}}}{(3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})}\right)$. This has a geometric interpretation. First, notice that the coefficients of $\delta\dot{x}(\theta)$ in Eq. (A.14) for planar motion match exactly the numerator and denominator inside the argument of $\tan^{-1}$ in θ^*. Now, let us look for a moment for values of θ at which $\delta\dot{x} = 0$:

$$\delta\dot{x}(\theta) = \delta\dot{x}_{\text{fail}}\cos\theta + (3n_{\text{ref}}\delta x_{\text{fail}} + 2\delta\dot{y}_{\text{fail}})\sin\theta \triangleq 0 \qquad \Longrightarrow \qquad \theta = \theta^*$$

  Thus θ^* represents the true anomalies (times) at which the chaser satisfies $\delta\dot{x} = 0$. In other words, the minimum-cost circularization Δv in the planar case occurs where the chaser has zero radial velocity, at apoapse or periapse (as we might expect from two-body orbital mechanics).

Case 2: Active Invariant Set Positioning Constraint

Here the chaser attempts to circularize its orbit at the boundary of the zero-thrust RIC shown in Fig. 6.4a. The positioning constraint is active, and therefore $g_{\text{circ}}(\theta) = \rho_{\delta x}^2 - \delta x^2\left(\theta_{\text{circ}}^-\right) = 0$. This is equivalent to finding where the coasting trajectory from $\mathbf{x}(t_{\text{fail}}) = \mathbf{x}_{\text{fail}}$ crosses $\delta x(\theta) = \pm\rho_{\delta x}$ for $\theta \in [\theta_{\min}, \theta_{\max}]$. This can be achieved using standard root-finding algorithms. Denote the set of candidate solutions that satisfy Case 2 by Θ_2^*.

Solution to the Minimal-Cost Circularization Burn

The global optimizer θ^* either lies on the boundary of the box constraint, at an unconstrained optimum ($\theta \in \Theta_1^*$), or at the boundary of the zero-thrust RIC ($\theta \in \Theta_2^*$), all of which are economically

obtained through either numerical integration or a root-finding solver. Therefore, the minimal-cost circularization burn time T_h^* satisfies:

$$\theta^* = n_{\text{ref}}(T_h^* - t_{\text{fail}}) = \underset{\theta \in \{\theta_{\min}, \theta_{\max}\} \bigcup \Theta_1^* \bigcup \Theta_2^*}{\arg\min} \Delta v_{\text{circ}}^2(\theta)$$

where $\Delta v_{\text{circ}}^2(\theta)$ is given by Eq. (B.1). If no solution exists (which can happen if and only if $\mathbf{x}_{\text{fail}}$ starts inside the KOZ), there is no safe circularization CAM and we therefore declare $\mathbf{x}_{\text{fail}}$ unsafe. Otherwise, the CAM is saved for future trajectory feasibility verification. If later we find that we can achieve the one-burn abort maneuver from $\mathbf{x}_{\text{fail}}$ given by $\boldsymbol{\Delta}\mathbf{v}_{\text{circ}}(\theta^*)$ under all failure cases of interest (either offline if mission constraints are known *a priori* and time invariant, *i.e.*, independent of the arrival time t_{fail} at $\mathbf{x}_{\text{fail}}$, or online otherwise), then $\mathbf{x}_{\text{fail}}$ can be declared an *actively-safe* state (see Section 6.3.3). This forms the basis for our actively-safe sampling routine in Algorithm 6, as described in detail in Section 7.3.

Appendix C

Intermediate Results for the FMT* Optimality Proof

We report here a number of useful lemmas concerning bounds on the trajectory costs between samples, which are used throughout the asymptotic optimality proof for FMT* in Section 7.4. We begin with the proof of Lemma 23, which relates the propellant-burn cost function Eq. (7.4) between points $\mathbf{x}_0$ and $\mathbf{x}_\mathrm{f}$ to the norm of the stacked Δv-vector $\|\mathbf{\Delta V}\| = \|\mathbf{x}_\mathrm{f} - \mathbf{\Phi}(t_\mathrm{f}, t_0)\mathbf{x}_0\|_{\mathbf{G}^{-1}}$. We then provide a lemma bounding the sizes of the minimum and maximum eigenvalues of $\mathbf{G}$, useful for bounding reachable volumes from $\mathbf{x}_0$. Finally, we prove Lemma 25 which forms the basis of our asymptotic optimality analysis for FMT*. Here $\mathbf{\Phi}(t_\mathrm{f}, t_0) = e^{\mathbf{A}T}$ is the state transition matrix, $T = t_\mathrm{f} - t_0$ is the maneuver duration, and $\mathbf{G}$ is the $N = 2$ impulse Gramian matrix:

$$\mathbf{G}(T) = \mathbf{\Phi}_v \mathbf{\Phi}_v^{-1} = \left[\begin{array}{cc} e^{\mathbf{A}T}\mathbf{B} & \mathbf{B} \end{array}\right] \left[\begin{array}{cc} e^{\mathbf{A}T}\mathbf{B} & \mathbf{B} \end{array}\right]^{\mathrm{T}}, \tag{C.1}$$

where $\mathbf{\Phi}_v(t, \{\tau_i\}_i)$ is the aggregate Δv transition matrix corresponding to burn times $\{\tau_i\}_i = \{t_0, t_\mathrm{f}\}$.

Lemma 23 (Fuel Burn Cost Bounds)**.** *For the cost function in Eq.* (7.4)*, we have the following upper and lower bounds:*

$$\|\mathbf{\Delta V}\| \leq J(\mathbf{x}_0, \mathbf{x}_\mathrm{f}) \leq \sqrt{2}\|\mathbf{\Delta V}\|.$$

Proof. For the upper bound, note that by the Cauchy-Schwarz inequality:

$$J = \|\mathbf{\Delta v}_1\| \cdot 1 + \|\mathbf{\Delta v}_2\| \cdot 1 \leq \sqrt{\|\mathbf{\Delta v}_1\|^2 + \|\mathbf{\Delta v}_2\|^2} \cdot \sqrt{1^2 + 1^2} = \sqrt{2}\|\mathbf{\Delta V}\|.$$

Similarly, for the lower bound, note that:

$$J = \sqrt{\left(\|\boldsymbol{\Delta}\mathbf{v}_1\| + \|\boldsymbol{\Delta}\mathbf{v}_2\|\right)^2} \geq \sqrt{\|\boldsymbol{\Delta}\mathbf{v}_1\|^2 + \|\boldsymbol{\Delta}\mathbf{v}_2\|^2} = \|\boldsymbol{\Delta}\mathbf{V}\|.$$

This completes the proof. □

Lemma 31 (Bounds on Gramian Eigenvalues)**.** *Let $T_{\max}$ be less than one orbital period for the system dynamics of Section 5.1.3, and let $\mathbf{G}(T)$ be defined as in Eq.* (C.1)*. Then there exist constants $M_{\min}, M_{\max} > 0$ such that $\lambda_{\min}(\mathbf{G}(T)) \geq M_{\min}T^2$ and $\lambda_{\max}(\mathbf{G}(T)) \leq M_{\max}$ for all $T \in (0,\, T_{\max}]$.*

Proof. We bound the maximum eigenvalue of $\mathbf{G}$ through norm considerations, yielding $\lambda_{\max}(\mathbf{G}(T)) \leq \left(\left\|e^{\mathbf{A}^T}\mathbf{B}\right\| + \|\mathbf{B}\|\right)^2 \leq \left(e^{\|\mathbf{A}\|T_{\max}} + 1\right)^2$, and take $M_{\max} = \left(e^{\|\mathbf{A}\|T_{\max}} + 1\right)^2$. As long as $T_{\max}$ is less than one orbital period, $\mathbf{G}(T)$ only approaches singularity near $T = 0$ [155]. Explicitly Taylor-expanding $\mathbf{G}(T)$ about $T = 0$ reveals that $\lambda_{\min}(\mathbf{G}(T)) = T^2/2 + \mathrm{O}\left(T^3\right)$ for small T, and thus $\lambda_{\min}(\mathbf{G}(T)) = \Omega\left(T^2\right)$ for all $T \in (0,\, T_{\max}]$. □

Lemma 25 (Steering with Perturbed Endpoints)**.** *For a given steering trajectory $\mathbf{x}(t)$ with initial time t_0 and final time t_f, let $\mathbf{x}_0 \coloneqq \mathbf{x}(t_0)$, $\mathbf{x}_\mathrm{f} \coloneqq \mathbf{x}(t_\mathrm{f})$, $T \coloneqq t_\mathrm{f} - t_0$, and $J \coloneqq J(\mathbf{x}_0, \mathbf{x}_\mathrm{f})$. Consider now the perturbed steering trajectory $\tilde{\mathbf{x}}(t)$ between perturbed start and end points $\tilde{\mathbf{x}}_0 = \mathbf{x}_0 + \delta\mathbf{x}_0$ and $\tilde{\mathbf{x}}_\mathrm{f} = \mathbf{x}_\mathrm{f} + \delta\mathbf{x}_\mathrm{f}$, and its corresponding cost $J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f})$.*
Case 1: $T = 0$. There exists a perturbation center $\delta\mathbf{x}_c$ (consisting of only a position shift) with $\|\delta\mathbf{x}_c\| = \mathrm{O}\left(J^2\right)$ such that if $\|\delta\mathbf{x}_0\| \leq \eta J^3$ and $\|\delta\mathbf{x}_\mathrm{f} - \delta\mathbf{x}_c\| \leq \eta J^3$, then $J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f}) \leq J(1 + 4\eta + \mathrm{O}(J))$ and the spatial deviation of the perturbed trajectory $\tilde{\mathbf{x}}(t)$ from $\mathbf{x}(t)$ is $\mathrm{O}(J)$.
Case 2: $T > 0$. If $\|\delta\mathbf{x}_0\| \leq \eta J^3$ and $\|\delta\mathbf{x}_\mathrm{f}\| \leq \eta J^3$, then $J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f}) \leq J\left(1 + \mathrm{O}\left(\eta J^2 T^{-1}\right)\right)$ and the spatial deviation of the perturbed trajectory $\tilde{\mathbf{x}}(t)$ from $\mathbf{x}(t)$ is $\mathrm{O}(J)$.

Proof. For bounding the perturbed cost $J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f})$, we consider the two cases separately.
Case 1: $T = 0$
Here 2-impulse steering degenerates to a single net impulse $\boldsymbol{\Delta}\mathbf{v}$; that is, $\mathbf{x}_\mathrm{f} = \mathbf{x}_0 + \mathbf{B}\boldsymbol{\Delta}\mathbf{v}$ with $\|\boldsymbol{\Delta}\mathbf{v}\| = J$. To aid in the ensuing analysis, denote the position and velocity components of states $\mathbf{x} = \left[\mathbf{r}^\mathrm{T}, \mathbf{v}^\mathrm{T}\right]^\mathrm{T}$ as $\mathbf{r} = [\,\mathbf{I}, \mathbf{0}\,]\mathbf{x}$ and $\mathbf{v} = [\,\mathbf{0}, \mathbf{I}\,]\mathbf{x}$. Since $T = 0$, we have $\mathbf{r}_\mathrm{f} = \mathbf{r}_0$ and $\mathbf{v}_\mathrm{f} = \mathbf{v}_0 + \boldsymbol{\Delta}\mathbf{v}$. We pick the perturbed steering duration $\widetilde{T} = J^2$ (which will provide an upper bound on the optimal steering cost) and Taylor-expand the steering system (Eq. (5.4)) for small durations $\widetilde{T}$ as:

$$\mathbf{r}_\mathrm{f} + \delta\mathbf{r}_\mathrm{f} = (\mathbf{r}_0 + \delta\mathbf{r}_0) + \widetilde{T}(\mathbf{v}_0 + \delta\mathbf{v}_0 + \widetilde{\boldsymbol{\Delta}\mathbf{v}}_1) + \mathrm{O}\left(\widetilde{T}^2\right) \tag{C.2}$$

$$\mathbf{v}_\mathrm{f} + \delta\mathbf{v}_\mathrm{f} = (\mathbf{v}_0 + \delta\mathbf{v}_0) + \widetilde{\boldsymbol{\Delta}\mathbf{v}}_1 + \widetilde{\boldsymbol{\Delta}\mathbf{v}}_2 + \widetilde{T}\left(\mathbf{A}_{21}(\mathbf{r}_0 + \delta\mathbf{r}_0) + \mathbf{A}_{22}(\mathbf{v}_0 + \delta\mathbf{v}_0 + \widetilde{\boldsymbol{\Delta}\mathbf{v}}_1)\right) + \mathrm{O}\left(\widetilde{T}^2\right) \tag{C.3}$$

where $\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1$ and $\widetilde{\boldsymbol{\Delta}\mathbf{v}}_2$ denote the perturbed steering trajectory's intercept and rendezvous impulse vectors, respectively, and $\mathbf{A}_{21} = \begin{bmatrix} 3n_\mathrm{ref}^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -n_\mathrm{ref}^2 \end{bmatrix}$ and $\mathbf{A}_{22} = \begin{bmatrix} 0 & 2n_\mathrm{ref} & 0 \\ -2n_\mathrm{ref} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

If we solve Eq. (C.2) for $\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1$ to first order, we find:

$$\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1 = \widetilde{T}^{-1}(\delta\mathbf{r}_\mathrm{f} - \delta\mathbf{r}_0) - (\mathbf{v}_0 + \delta\mathbf{v}_0) + \mathrm{O}\big(\widetilde{T}\big).$$

By selecting $\delta\mathbf{x}_c = \begin{bmatrix}\widetilde{T}\mathbf{v}_0^\mathrm{T} & \mathbf{0}^\mathrm{T}\end{bmatrix}^\mathrm{T}$ (note: $\|\delta\mathbf{x}_c\| = J^2\|\mathbf{v}_0\| = \mathrm{O}\big(J^2\big)$) and supposing that $\|\delta\mathbf{x}_0\| \leq \eta J^3$ and $\|\delta\mathbf{x}_\mathrm{f} - \delta\mathbf{x}_c\| \leq \eta J^3$, we have that:

$$\|\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1\| \leq J^{-2}(\|\delta\mathbf{x}_0\| + \|\delta\mathbf{x}_\mathrm{f} - \delta\mathbf{x}_c\|) + \|\delta\mathbf{x}_0\| + \mathrm{O}\big(J^2\big) = 2\eta J + \mathrm{O}\big(J^2\big).$$

Now solving Eq. (C.3) for $\widetilde{\boldsymbol{\Delta}\mathbf{v}}_2 = \boldsymbol{\Delta}\mathbf{v} + (\delta\mathbf{v}_\mathrm{f} - \delta\mathbf{v}_0) - \widetilde{\boldsymbol{\Delta}\mathbf{v}}_1 + \mathrm{O}\big(J^2\big)$, and taking norms of both sides:

$$\|\widetilde{\boldsymbol{\Delta}\mathbf{v}}_2\| \leq \|\boldsymbol{\Delta}\mathbf{v}\| + (\|\delta\mathbf{x}_0\| + \|\delta\mathbf{x}_\mathrm{f} - \delta\mathbf{x}_c\|) + 2\eta J + \mathrm{O}\big(J^2\big) \leq J + 2\eta J + \mathrm{O}\big(J^2\big).$$

Therefore the perturbed cost satisfies:

$$\begin{aligned} J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f}) &\leq \|\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1\| + \|\widetilde{\boldsymbol{\Delta}\mathbf{v}}_2\| \\ &\leq J(1 + 4\eta + \mathrm{O}(J)). \end{aligned}$$

Case 2: $T > 0$

We pick $\widetilde{T} = T$ to compute an upper bound on the perturbed cost. Applying the explicit form of the steering control $\boldsymbol{\Delta}\mathbf{V}$ (see Eq. (7.3)) along with the norm bound $\big\|\boldsymbol{\Phi}_v^{-1}\big\| = \lambda_{\min}(\mathbf{G})^{-1/2} \leq M_{\min}^{-1/2}T^{-1}$ from Lemma 31, we have:

$$\begin{aligned} J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f}) &\leq \big\|\boldsymbol{\Phi}_v^{-1}(t_\mathrm{f}, \{t_0, t_\mathrm{f}\})(\tilde{\mathbf{x}}_\mathrm{f} - \boldsymbol{\Phi}(t_\mathrm{f}, t_0)\tilde{\mathbf{x}}_0)\big\| \\ &\leq \big\|\boldsymbol{\Phi}_v^{-1}(\mathbf{x}_\mathrm{f} - \boldsymbol{\Phi}\mathbf{x}_0)\big\| + \big\|\boldsymbol{\Phi}_v^{-1}\delta\mathbf{x}_\mathrm{f}\big\| + \big\|\boldsymbol{\Phi}_v^{-1}\boldsymbol{\Phi}\delta\mathbf{x}_0\big\| \\ &\leq J + M_{\min}^{-1/2}T^{-1}\|\delta\mathbf{x}_\mathrm{f}\| + M_{\min}^{-1/2}T^{-1}e^{\|\mathbf{A}\|T_{\max}}\|\delta\mathbf{x}_0\| \\ &\leq J\big(1 + \mathrm{O}\big(\eta J^2 T^{-1}\big)\big). \end{aligned}$$

In both cases, the deviation of the perturbed steering trajectory $\tilde{\mathbf{x}}(t)$ from its closest point on the original trajectory is bounded (quite conservatively) by the maximum propagation of the difference in initial conditions; that is, the initial disturbance $\delta\mathbf{x}_0$ plus the difference in intercept burns $\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1 - \boldsymbol{\Delta}\mathbf{v}_1$, over the maximum maneuver duration $T_{\max}$. Thus,

$$\begin{aligned} \|\tilde{\mathbf{x}}(t) - \mathbf{x}(t)\| &\leq e^{\|\mathbf{A}\|T_{\max}}\Big(\|\delta\mathbf{x}_0\| + \|\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1\| + \|\boldsymbol{\Delta}\mathbf{v}_1\|\Big) \\ &\leq e^{\|\mathbf{A}\|T_{\max}}\big(\eta J^3 + 2J + \mathrm{o}(J)\big) = \mathrm{O}(J) \end{aligned}$$

where we have used $\|\boldsymbol{\Delta}\mathbf{v}_1\| \leq J$ and $\|\widetilde{\boldsymbol{\Delta}\mathbf{v}}_1\| \leq J(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_\mathrm{f}) \leq J + \mathrm{o}(J)$ from our above arguments. $\square$

Bibliography

[1] J. A. Starek, B. Açıkmeşe, et al. "Spacecraft Autonomy Challenges for Next Generation Space Missions". In: *Advances in Control System Technology for Aerospace Applications*. Ed. by E. Feron. Vol. 460. Lecture Notes in Control and Information Sciences. Springer, Sept. 2015. Chap. 1, pp. 1–48. DOI: 10.1007/978-3-662-47694-9_1.

[2] T. Lozano-Pérez and M. A. Wesley. "An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles". In: *Communications of the ACM* 22.10 (Oct. 1979), pp. 560–570. DOI: 10.1145/359156.359164.

[3] J. T. Betts. "Survey of Numerical Methods for Trajectory Optimization". In: *AIAA Journal of Guidance, Control, and Dynamics* 21.2 (1998), pp. 193–207. DOI: 10.2514/2.4231.

[4] F. Fahroo and I. M. Ross. "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method". In: *AIAA Journal of Guidance, Control, and Dynamics* 25.1 (2002), pp. 160–166. DOI: 10.2514/2.4862.

[5] D. G. Hull. "Conversion of Optimal Control Problems into Parameter Optimization Problems". In: *AIAA Journal of Guidance, Control, and Dynamics* 20.1 (1997), pp. 57–60. DOI: 10.2514/2.4033.

[6] J. Vlassenbroeck and R. V. Dooren. "A Chebyshev Technique for Solving Nonlinear Optimal Control Problems". In: *IEEE Transactions on Automatic Control* 33.4 (1988), pp. 333–340. DOI: 10.1109/9.192187.

[7] R. Mattingly and L. May. "Mars Sample Return as a Campaign". In: *IEEE Aerospace Conference*. Big Sky, MT, Mar. 2011, pp. 1–13. DOI: 10.1109/AERO.2011.5747287.

[8] B. Roberts and J. Pellegrino. "Robotic Servicing Technology Development". In: *AIAA SPACE Conferences & Exposition*. 5339. San Diego, CA, 2013, pp. 1–10. DOI: 10.2514/6.2013-5339.

[9] C. G. Henshaw. "The DARPA Phoenix Spacecraft Servicing Program: Overview and Plans for Risk Reduction". In: *i-SAIRAS*. Montreal, Canada, June 2014.

[10] W. Fehse. *Automated Rendezvous and Docking of Spacecraft*. Vol. 16. Cambridge University Press, 2003.

[11] D. P. Dannemiller. "Multi-Maneuver Clohessy-Wiltshire Targeting". In: *AAS Astrodynamics Specialist Conference*. Girdwood, AK, July 2011, pp. 1–15.

[12] E. T. Pitkin. "A General Solution of the Lambert Problem". In: *Journal of Astronautical Sciences* 15 (Sept. 1968), p. 270.

[13] H. B. Hablani, M. L. Tapper, et al. "Guidance and Relative Navigation for Autonomous Rendezvous in a Circular Orbit". In: *AIAA Journal of Guidance, Control, and Dynamics* 25.3 (2002), pp. 553–562. DOI: 10.2514/2.4916.

[14] C. Cruzen and J. T. Thompson. "Advancing Autonomous Operations Technologies for NASA Missions". In: *IEEE Aerospace Conference*. 2080. Big Sky, MT, Mar. 2013, pp. 1–10.

[15] M. E. Polites. *An Assessment of the Technology of Automated Rendezvous and Capture in Space*. Tech. rep. NASA/TP–1998-208528. NASA, July 1998.

[16] NASA. *The Vision for Space Exploration*. http://www.nasa.gov/pdf/55583main_vision_space_exploration2.pdf. Feb. 2004.

[17] N. OCT. *NASA Space Technology Roadmaps*. Tech. rep. TA04 - Robotics, Tele-Robotics and Autonomous Systems. NASA, Dec. 2013.

[18] S. C. for NASA Technology Roadmaps. *NASA Space Technology Roadmaps and Priorities: Restoring NASA's Technological Edge and Paving the Way for a New Era in Space*. Tech. rep. Available at http://www.nap.edu/openbook.php?record_id=13354. Washington, D.C.: NRC, 2012.

[19] L. M. Major, T. M. Brady, et al. "Apollo Looking Forward: Crew Task Challenges". In: *IEEE Aerospace Conference*. Big Sky, MT, Mar. 2009, pp. 1–8. DOI: 10.1109/AERO.2009.4839353.

[20] W. F. Truszkowski, M. G. Hinchey, et al. "Autonomous and Autonomic Systems: A Paradigm for Future Space Exploration Missions". In: *IEEE Transactions on Systems, Man, & Cybernetics. Part C: Applications & Reviews* 36.3 (May 2006), pp. 279–291. DOI: 10.1109/TSMCC.2006.871600.

[21] B. R. Sullivan and D. L. Akin. "A Survey of Serviceable Spacecraft Failures". In: *AIAA SPACE Conferences & Exposition*. Vol. 1. 4540. Albuquerque, NM, Aug. 2001, p. 4540. DOI: 10.2514/6.2001-4540.

[22] A. Flores-Abad, O. Ma, et al. "A Review of Space Robotics Technologies for On-Orbit Servicing". In: *Progress in Aerospace Sciences* 68 (July 2014), pp. 1–26. DOI: 10.1016/j.paerosci.2014.03.002.

[23] D. A. Whelan, E. A. Adler, et al. "The DARPA Orbital Express Program: Effecting a Revolution in Space-Based Systems". In: *SPIE Small Payloads in Space*. Vol. 4136. San Diego, CA, July 2000, pp. 48–56. DOI: 10.1117/12.406656.

[24] P. G. Antreasian, S. R. Chesley, et al. "The Design and Navigation of the NEAR Shoemaker Landing on Eros". In: *AAS Astrodynamics Specialist Conference*. 372. Quebec City, Quebec, Canada, July 2001, pp. 1–27.

[25] M. E. Holdridge. "NEAR Shoemaker Spacecraft Mission Operations". In: *Johns Hopkins APL Technical Digest* 23.1 (Jan. 2002), pp. 58–70.

[26] A. Fujiwara, J. Kawaguchi, et al. "The Rubble-Pile Asteroid Itokawa as Observed by Hayabusa". In: *Science* 312.5778 (June 2006), pp. 1330–1334. DOI: 10.1126/science.1125841.

[27] H. Yano, T. Kubota, et al. "Touchdown of the Hayabusa Spacecraft at the Muses Sea on Itokawa". In: *Science* 312.5778 (June 2006), pp. 1350–1353. DOI: 10.1126/science.1126164.

[28] S. Ulamec and J. Biele. "Surface Elements and Landing Strategies for Small Bodies Missions – Philae and Beyond". In: *Advances in Space Research* 44.7 (2009), pp. 847–858. DOI: 10.1016/j.asr.2009.06.009.

[29] P. Muñoz, F. Budnik, et al. "Rosetta Navigation During Lander Delivery Phase and Reconstruction of Philae Descent Trajectory and Rebound". In: *International Symposium on Space Flight Dynamics*. Munich, Germany, Oct. 2015, pp. 1–20.

[30] I. Kawano, M. Mokuno, et al. "Result of Autonomous Rendezvous Docking Experiment of Engineering Test Satellite-VII". In: *AIAA Journal of Spacecraft and Rockets* 38.1 (Jan. 2001), pp. 105–111. DOI: 10.2514/2.3661.

[31] M. Oda. "ETS-VII: Achievements, Troubles and Future". In: *i-SAIRAS*. ESA. Montreal, Quebec, Canada, June 2001, pp. 1–7.

[32] T. M. Davis and D. Melanson. "XSS-10 Microsatellite Flight Demonstration Program Results". In: *Proc. of SPIE*. Ed. by P. T. Jr. and M. Wright. Vol. 5419. SPIE Spacecraft Platforms and Infrastructure. Orlando, FL, Apr. 2004, pp. 16–25. DOI: 10.1117/12.544316.

[33] R. T. Howard, A. F. Heaton, et al. "Orbital Express Advanced Video Guidance Sensor". In: *IEEE Aerospace Conference*. Big Sky, MT, Mar. 2008, pp. 1–10. DOI: 10.1109/AERO.2008.4526518.

[34] T. E. Rumford. "Demonstration of Autonomous Rendezvous Technology (DART) Project Summary". In: *Proc. of SPIE*. Ed. by P. T. Jr. and J. Shoemaker. Vol. 5088. SPIE Space Systems Technology and Operations. Orlando, FL, Apr. 2003, pp. 10–19. DOI: 10.1117/12.498811.

[35] E. Gill, O. Montenbruck, et al. "Autonomous Formation Flying for the PRISMA Mission". In: *AIAA Journal of Spacecraft and Rockets* 44.3 (May 2007), pp. 671–681. DOI: 10.2514/1.23015.

[36] S. D'Amico, J.-S. Ardaens, et al. "Spaceborne Autonomous Formation-Flying Experiment on the PRISMA Mission". In: *AIAA Journal of Guidance, Control, and Dynamics* 35.3 (May 2012), pp. 834–850. DOI: 10.2514/1.55638.

[37] R. Z. Sagdeev and A. V. Zakharov. "Brief History of the Phobos Mission". In: *Nature* 341.6243 (Oct. 1989), pp. 581–585. DOI: 10.1038/341581a0.

[38] D. Bernard, R. Doyle, et al. "Autonomy and Software Technology on NASA's Deep Space One". In: *IEEE Intelligent Systems and their Applications* 14.3 (May 1999), pp. 10–15. DOI: 10.1109/5254.769876.

[39] D. M. I. Board. *Overview of the DART Mishap Investigation Results*. Tech. rep. Available at http://www.nasa.gov/pdf/148072main_DART_mishap_overview.pdf. NASA, May 2006.

[40] M. Yoshikawa, J. Kawaguchi, et al. "Hayabusa Sample Return Mission". In: *Asteroids IV*. University of Arizona Press, 2015, pp. 397–418.

[41] T. Yoshimitsu, T. Kubota, et al. "MINERVA Rover Which Became a Small Artificial Solar Satellite". In: *Proc. of the AIAA/USU Conf. on Small Satellites*. 06-IV-4. Logan, UT, Aug. 2006, pp. 1–6.

[42] M. Uo, K. Shirakawa, et al. "Attitude Control Challenges and Solutions for Hayabusa Spacecraft". In: *AAS Astrodynamics Specialist Conference*. 6534. Keystone, CO, Aug. 2006, pp. 1–10. DOI: 10.2514/6.2006-6534.

[43] M. Tafazoli. "A Study of On-orbit Spacecraft Failures". In: *Acta Astronautica* 64.2 (Jan. 2009), pp. 195–205. DOI: 10.1016/j.actaastro.2008.07.019.

[44] J. L. Goodman. *Lessons Learned From Seven Space Shuttle Missions*. Tech. rep. NASA/CR-2007-213697. Houston, TX: United Space Alliance, Jan. 2007.

[45] L. Breger and J. P. How. "Safe Trajectories for Autonomous Rendezvous of Spacecraft". In: *AIAA Journal of Guidance, Control, and Dynamics* 31.5 (2008), pp. 1478–1489. DOI: 10.2514/1.29590.

[46] A. Richards, T. Schouwenaars, et al. "Spacecraft Trajectory Planning With Avoidance Constraints Using Mixed-Integer Linear Programming". In: *AIAA Journal of Guidance, Control, and Dynamics* 25.4 (2002), pp. 755–765. DOI: 10.2514/2.4943.

[47] G. Dettleff. "Plume Flow and Plume Impingement in Space Technology". In: *Progress in Aerospace Sciences* 28.1 (1991), pp. 1–71. DOI: 10.1016/0376-0421(91)90008-R.

[48] J. L. Goodman. "History of Space Shuttle Rendezvous and Proximity Operations". In: *AIAA Journal of Spacecraft and Rockets* 43.5 (Sept. 2006), pp. 944–959. DOI: 10.2514/1.19653.

[49] I. M. Ross. "How to Find Minimum-Fuel Controllers". In: *AIAA Conf. on Guidance, Navigation and Control*. Providence, RI, Aug. 2004, pp. 1–10. DOI: 10.2514/6.2004-5346.

[50] B. Açıkmeşe and L. Blackmore. "Lossless Convexification of a Class of Optimal Control Problems with Non-convex Control Constraints". In: *Automatica* 47.2 (Feb. 2011), pp. 341–347. DOI: 10.1016/j.automatica.2010.10.037.

[51] D. E. Gaylor and B. W. Barbee. "Algorithms for Safe Spacecraft Proximity Operations". In: *AAS Meeting*. Vol. 127. Advances in the Astronautical Sciences 107. Seattle, WA, Jan. 2007, pp. 1–20.

[52] J. M. Carson, B. Açikmeşe, et al. "A Robust Model Predictive Control Algorithm with a Reactive Safety Mode". In: *IFAC World Congress*. Ed. by M. J. Chung and P. Misra. Vol. 17. 1. Gangnam-gu Seoul, South Korea, July 2008, pp. 13175–13181. DOI: 10.3182/20080706-5-KR-1001.02232.

[53] A. Weiss, M. Baldwin, et al. "Spacecraft Constrained Maneuver Planning for Moving Debris Avoidance Using Positively Invariant Constraint Admissible Sets". In: *American Control Conference*. Washington, DC, June 2013, pp. 4802–4807. DOI: 10.1109/ACC.2013.6580581.

[54] E. Frazzoli. "Quasi-Random Algorithms for Real-Time Spacecraft Motion Planning and Coordination". In: *Acta Astronautica* 53.4–10 (Aug. 2003), pp. 485–495. DOI: 10.1016/S0094-5765(03)80009-7.

[55] J. A. Starek, B. W. Barbee, et al. "A Sampling-Based Approach to Spacecraft Autonomous Maneuvering with Safety Specifications". In: *AAS GN&C Conference*. Vol. 154. Advances in the Astronautical Sciences. Breckenridge, CO, Feb. 2015, pp. 1–13.

[56] Y. Nesterov and A. Nemirovskii. *Interior-point Polynomial Methods in Convex Programming*. Philadelphia, PA: SIAM, 1994. DOI: 10.1137/1.9781611970791.

[57] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[58] J. Mattingley and S. Boyd. "Real-time Convex Optimization in Signal Processing". In: *IEEE Signal Processing Magazine* 27.3 (May 2010), pp. 50–61. DOI: 10.1109/MSP.2010.936020.

[59] M. W. Harris and B. Açıkmeşe. "Lossless Convexification of Non-Convex Optimal Control Problems for State Constrained Linear Systems". In: *Automatica* 50.9 (2014), pp. 2304–2311. DOI: 10.1016/j.automatica.2014.06.008.

[60] S. Nolet, E. Kong, et al. "Design of an Algorithm for Autonomous Docking with a Freely Tumbling Target". In: *Proc. of SPIE*. Ed. by P. Motaghedi. Vol. 5799. SPIE Modeling, Simulation, and Verification of Space-based Systems. Orlando, FL, Mar. 2005, pp. 123–134. DOI: 10.1117/12.603178.

[61] B. Açikmeşe, J. M. Carson, et al. "A Robust Model Predictive Control Algorithm for Incrementally Conic Uncertain/Nonlinear Systems". In: *International Journal on Robust and Nonlinear Control* 21.5 (2011), pp. 563–590. DOI: 10.1002/rnc.1613.

[62] R. Bevilacqua, M. Romano, et al. "Guidance Navigation and Control for Autonomous Multiple Spacecraft Assembly: Analysis and Experimentation". In: *Int. Journal of Aerospace Engineering* (Dec. 2011), pp. 1–18. DOI: 10.1155/2011/308245.

[63] J. M. Carson III, B. Açıkmeşe, et al. "Capabilities of Convex Powered-Descent Guidance Algorithms for Pinpoint and Precision Landing". In: *IEEE Aerospace Conference*. Big Sky, MT, Mar. 2011, pp. 1–8. DOI: 10.1109/AERO.2011.5747244.

[64] H. Park, S. Di Cairano, et al. "Model Predictive Control for Spacecraft Rendezvous and Docking with a Rotating/Tumbling Platform and for Debris Avoidance". In: *American Control Conference*. San Francisco, CA, July 2011, pp. 1922–1927. DOI: 10.1109/ACC.2011.5991151.

[65] C. R. McInnes. "Potential Function Methods for Autonomous Spacecraft Guidance and Control". In: *AAS Astrodynamics Specialist Conference*. Halifax, Nova Scotia, Canada, Aug. 1995, pp. 2093–2109.

[66] A. Badawy and C. R. McInnes. "On-Orbit Assembly Using Superquadric Potential Fields". In: *AIAA Journal of Guidance, Control, and Dynamics* 31.1 (Jan. 2008), pp. 30–43. DOI: 10.2514/1.28865.

[67] R. Bevilacqua, T. Lehmann, et al. "Development and Experimentation of LQR/APF Guidance and Control for Autonomous Proximity Maneuvers of Multiple Spacecraft". In: *Acta Astronautica* 68.7-8 (Apr. 2011), pp. 1260–1275. DOI: 10.1016/j.actaastro.2010.08.012.

[68] S. M. LaValle and J. J. Kuffner. "Randomized Kinodynamic Planning". In: *International Journal of Robotics Research* 20.5 (May 2001), pp. 378–400. DOI: 10.1177/02783640122067453.

[69] W. S. Widnall. *Apollo Guidance Navigation and Control: Guidance System Operations Plan for Manned CM Earth Orbital and Lunar Missions Using Program COLOSSUS I and Program COLOSSUS IA*. Tech. rep. R-577 Section 3. Cambridge, MA: MIT Instrumentation Laboratory, Dec. 1968.

[70] D. Mayne, J. Rawlings, et al. "Constrained Model Predictive Control: Stability and Optimality". In: *Automatica* 36.6 (2000), pp. 789–814. DOI: 10.1016/S0005-1098(99)00214-9.

[71] E. F. Camacho and C. Bordons. "Nonlinear Model Predictive Control: An Introductory Review". In: *Assessment and Future Directions of Nonlinear Model Predictive Control*. Ed. by R. Findeisen, F. Allgöwer, et al. Vol. 358. Lecture Notes in Control and Information Sciences. Springer, 2007, pp. 1–16. DOI: 10.1007/978-3-540-72699-9_1.

[72] S. Quinlan and O. Khatib. "Elastic Bands: Connecting Path Planning and Control". In: *Proc. IEEE Conf. on Robotics and Automation*. Vol. 2. Atlanta, GA, May 1993, pp. 802–807. DOI: 10.1109/ROBOT.1993.291936.

[73] O. Brock and O. Khatib. "Real-time Re-planning in High-Dimensional Configuration Spaces Using Sets of Homotopic Paths". In: *Proc. IEEE Conf. on Robotics and Automation*. Vol. 1. San Francisco, CA, Apr. 2000, pp. 550–555. DOI: 10.1109/ROBOT.2000.844111.

[74] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[75] Y. K. Hwang and N. Ahuja. "Gross Motion Planning — A Survey". In: *ACM Computing Surveys* 24.3 (Sept. 1992), pp. 219–291. DOI: 10.1145/136035.136037.

[76] O. Takahashi and R. J. Schilling. "Motion Planning in a Plane Using Generalized Voronoi Diagrams". In: *IEEE Transactions on Robotics and Automation* 5.2 (1989), pp. 143–150. DOI: 10.1109/70.88035.

[77] M. Sharir. "Algorithmic Motion Planning". In: *Handbook of Discrete and Computational Geometry*. Ed. by J. E. Goodman and J. O'Rourke. CRC Press, 1997. Chap. 40, pp. 733–754.

[78] L. E. Kavraki, P. Švestka, et al. "Probabilistic Roadmaps for Path Planning in High-Dimensional Spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (Aug. 1996), pp. 566–580. DOI: 10.1109/70.508439.

[79] S. Karaman and E. Frazzoli. "Optimal Kinodynamic Motion Planning using Incremental Sampling-based Methods". In: *Proc. IEEE Conf. on Decision and Control*. 2010, pp. 7681–7687. DOI: 10.1109/CDC.2010.5717430.

[80] L. Janson, E. Schmerling, et al. "Fast Marching Tree: A Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions". In: *International Journal of Robotics Research* 34.7 (2015), pp. 883–921. DOI: 10.1177/0278364915577958.

[81] E. Schmerling, L. Janson, et al. "Optimal Sampling-Based Motion Planning under Differential Constraints: the Driftless Case". In: *Proc. IEEE Conf. on Robotics and Automation*. Seattle, WA, May 2015, pp. 2368–2375. DOI: 10.1109/ICRA.2015.7139514.

[82] E. Schmerling, L. Janson, et al. "Optimal Sampling-Based Motion Planning under Differential Constraints: the Drift Case with Linear Affine Dynamics". In: *Proc. IEEE Conf. on Decision and Control*. 2015.

[83] E. Frazzoli, M. A. Dahleh, et al. "A Randomized Attitude Slew Planning Algorithm for Autonomous Spacecraft". In: *AIAA Conf. on Guidance, Navigation and Control*. Montreal, Quebec, Canada, Aug. 2001, pp. 1–8. DOI: 10.2514/6.2001-4155.

[84] J. M. Phillips, L. E. Kavraki, et al. "Spacecraft Rendezvous and Docking with Real-Time, Randomized Optimization". In: *AIAA Conf. on Guidance, Navigation and Control*. Austin, TX, Aug. 2003, pp. 1–11. DOI: 10.2514/6.2003-5511.

[85] C. Urmson, J. Anhalt, et al. "Autonomous Driving in Urban Environments: Boss and the Urban Challenge". In: *Journal of Field Robotics* 25.8 (July 2008), pp. 425–466. DOI: 10.1002/rob.20255.

[86] J. Leonard, J. P. How, et al. "A Perception-Driven Autonomous Urban Vehicle". In: *Journal of Field Robotics* 25.10 (2008), pp. 727–774. DOI: 10.1002/rob.20262.

[87] Y. Kuwata, J. Teo, et al. "Real-Time Motion Planning With Applications to Autonomous Urban Driving". In: *IEEE Transactions on Control Systems Technology* 17.5 (Sept. 2009), pp. 1105–1118. DOI: 10.1109/TCST.2008.2012116.

[88] M. Buehler, K. Iagnemma, et al., eds. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. 1st ed. Vol. 56. Tracts in Advanced Robotics. Springer, 2010, p. 628.

[89] M. Montemerlo, J. Becker, et al. "Junior: The Stanford Entry in the Urban Challenge". In: *Journal of Field Robotics* 25.9 (Aug. 2008), pp. 569–597. DOI: 10.1002/rob.20258.

[90] J. A. Starek, J. V. Gomez, et al. "An Asymptotically-Optimal Sampling-Based Algorithm for Bi-directional Motion Planning". In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. Hamburg, Germany, Sept. 2015, pp. 2072–2078. DOI: 10.1109/IROS.2015.7353652.

[91] J. A. Starek, E. Schmerling, et al. "Real-Time, Propellant-Optimized Spacecraft Motion Planning under Clohessy-Wiltshire-Hill Dynamics". In: *IEEE Aerospace Conference*. In Press. Big Sky, MT, Mar. 2016, pp. 1–16.

[92] J. A. Starek, E. Schmerling, et al. "Fast, Safe, and Propellant-Efficient Spacecraft Planning under Clohessy-Wiltshire-Hill Dynamics". In: *AIAA Journal of Guidance, Control, and Dynamics, Special Issue on Computational Guidance and Control* (Jan. 2016). Submitted.

[93] J. H. Reif. "Complexity of the Mover's Problem and Generalizations". In: *20th Annual IEEE Symposium on Foundations of Computer Science*. 1979, pp. 421–427.

[94] J. F. Canny. *Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[95] J. Barraquand, L. Kavraki, et al. "A Random Sampling Scheme for Path Planning". In: *International Journal of Robotics Research* 16.6 (Dec. 2000), pp. 759–774. DOI: 10.1177/027836499701600604.

[96] R. Johnsonbaugh and W. E. Pfaffenberger. *Foundations of Mathematical Analysis*. Dover Books on Mathematics. Mineola, New York: Dover Publications, 2012.

[97] H. Niederreiter. *Random number generation and Quasi-Monte Carlo methods*. CBMS-NSF Regional Conference Series in Applied Mathematics 63. Society for Industrial & Applied Mathematics, 1992.

[98] A. G. Sukharev. "Optimal Strategies of the Search for an Extremum". In: *USSR Computational Mathematics and Mathematical Physics* 11.4 (1971), pp. 119–137. DOI: 10.1016/0041-5553(71)90008-5.

[99] J. H. Halton. "On the Efficiency of Certain Quasirandom Sequences of Points in Evaluating Multidimensional Integrals". In: *Numerische Mathematik* 2.1 (1960), pp. 84–90. DOI: 10.1007/BF01386213.

[100] J. M. Hammersley. "Monte Carlo Methods for Solving Multivariable Problems". In: *Annals of the Ney York Academy of Sciences* 86.3 (May 1960), pp. 844–874. DOI: 10.1111/j.1749-6632.1960.tb42846.x.

[101] M. Matsumoto and T. Nishimura. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-random Number Generator". In: *ACM Transactions on Modeling and Computer Simulation* 8.1 (Jan. 1998), pp. 3–30. DOI: 10.1145/272991.272995.

[102] J. M. Phillips, N. Bedrossian, et al. "Guided Expansive Spaces Trees: A Search Strategy for Motion- and Cost-Constrained State Spaces". In: *Proc. IEEE Conf. on Robotics and Automation*. Vol. 4. New Orleans, LA, Apr. 2004, pp. 3968–3973. DOI: 10.1109/ROBOT.2004.1308890.

[103] S. Karaman and E. Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning". In: *International Journal of Robotics Research* 30.7 (June 2011), pp. 846–894. DOI: 10.1177/0278364911406761.

[104] R. Allen and M. Pavone. "Toward a Real-Time Framework for Solving the Kinodynamic Motion Planning Problem". In: *Proc. IEEE Conf. on Robotics and Automation*. Seattle, WA, May 2015, pp. 928–934. DOI: 10.1109/ICRA.2015.7139288.

[105] R. Allen, A. Clark, et al. "A Machine Learning Approach for Real-Time Reachability Analysis". In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. Chicago, IL, Sept. 2014, pp. 2202–2208. DOI: 10.1109/IROS.2014.6942859.

[106] D. Hsu, J.-C. Latombe, et al. "Path Planning in Expansive Configuration Spaces". In: *International Journal of Computational Geometry & Applications* 9.4 (Feb. 1999), pp. 495–512. DOI: 10.1142/S0218195999000285.

[107] J. Luo and K. Hauser. "An Empirical Study of Optimal Motion Planning". In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. Sept. 2014, pp. 1761–1768. DOI: 10.1109/IROS.2014.6942793.

[108] J. D. Gammell, S. S. Srinivasa, et al. *BIT*: Batch Informed Trees for Optimal Sampling-based Planning via Dynamic Programming on Implicit Random Geometric Graphs*. Available at http://arxiv.org/abs/1405.5848. 2014.

[109] O. Arslan and P. Tsiotras. "Use of Relaxation Methods in Sampling-Based Algorithms for Optimal Motion Planning". In: *Proc. IEEE Conf. on Robotics and Automation*. Karlsruhe, Germany, May 2013, pp. 2421–2428. DOI: 10.1109/ICRA.2013.6630906.

[110] I. Pohl. *Bi-directional and heuristic search in path problems*. 104. Department of Computer Science, Stanford University., 1969.

[111] M. Luby and P. Ragde. "A Bidirectional Shortest-Path Algorithm with Good Average-Case Behavior". In: *Algorithmica* 4.1 (June 1989), pp. 551–567. DOI: 10.1007/BF01553908.

[112] A. Goldberg, H. Kaplan, et al. "Reach for A*: Efficient Point-to-Point Shortest Path Algorithms". In: *Proc. of the 8th Workshop on Alg. Engin. and Experiments (ALENEX)*. Springer, 2006. Chap. 12, pp. 129–143.

[113] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271. DOI: 10.1007/BF01386390.

[114] J. J. Kuffner and S. M. LaValle. "RRT-Connect: An Efficient Approach to Single-Query Path Planning". In: *Proc. IEEE Conf. on Robotics and Automation*. San Francisco, CA, Apr. 2000, pp. 995–1001. DOI: 10.1109/ROBOT.2000.844730.

[115] G. Sánchez and J.-C. Latombe. "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking". In: *Robotics Research*. Vol. 6. Springer, 2003. Chap. 9, pp. 403–417. DOI: 10.1007/3-540-36460-9_27.

[116] B. Akgun and M. Stilman. "Sampling Heuristics for Optimal Motion Planning in High Dimensions". In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. IEEE. San Francisco, CA, Sept. 2011, pp. 2640–2645. DOI: 10.1109/IROS.2011.6095077.

[117] M. Jordan and A. Perez. "Optimal Bidirectional Rapidly-Exploring Random Trees". http://people.csail.mit.edu/aperez/obirrt/csailtech.pdf. Aug. 2013.

[118] J. A. Starek, J. V. Gomez, et al. "An Asymptotically-Optimal Sampling-Based Algorithm for Bi-Directional Motion Planning (Extended Version)". Available at http://arxiv.org/abs/1507.07602/. July 2015.

[119] L. Janson, B. Ichter, et al. "Deterministic Sampling-Based Motion Planning: Optimality, Complexity, and Performance". In: *International Symposium on Robotics Research*. In Press. Sestri Levante, Italy, Sept. 2015.

[120] O. Salzman and D. Halperin. *Asymptotically-Optimal Motion Planning using Lower Bounds on Cost*. Available at http://arxiv.org/abs/1403.7714v2/. 2014.

[121] I. A. Şucan, M. Moll, et al. "The Open Motion Planning Library". In: *IEEE Robotics and Automation Magazine* 19.4 (Dec. 2012), pp. 72–82. DOI: 10.1109/MRA.2012.2205651.

[122] J. L. Schwartz, M. A. Peck, et al. "Historical Review of Air-Bearing Spacecraft Simulators". In: *AIAA Journal of Guidance, Control, and Dynamics* 26.4 (July 2003), pp. 513–522. DOI: 10.2514/2.5085.

[123] E. A. LeMaster, D. B. Schaechter, et al. "Experimental Demonstration of Technologies for Autonomous On-Orbit Robotic Assembly". In: *AIAA SPACE Conferences & Exposition*. Vol. 2006. 7428. San Jose, CA, Sept. 2006, pp. 1–13. DOI: 10.2514/6.2006-7428.

[124] E. A. LeMaster. "Optical Pathlength Control Between Formation-Flying Space Vehicle Emulators". In: *AIAA SPACE Conferences & Exposition*. 6488. Pasadena, CA, Sept. 2009, pp. 1–12. DOI: 10.2514/6.2009-6488.

[125] M. Romano, D. A. Friedman, et al. "Laboratory Experimentation of Autonomous Spacecraft Approach and Docking to a Collaborative Target". In: *AIAA Journal of Spacecraft and Rockets* 44.1 (Jan. 2007), pp. 164–173. DOI: 10.2514/1.22092.

[126] M. Kobilarov and S. Pellegrino. "Trajectory Planning for CubeSat Short-Time-Scale Proximity Operations". In: *AIAA Journal of Guidance, Control, and Dynamics* 37.2 (Mar. 2014), pp. 566–579. DOI: 10.2514/1.60289.

[127] L. Ljung. "Asymptotic Behavior of the Extended Kalman Filter as a Parameter Estimator for Linear Systems". In: *IEEE Transactions on Automatic Control* 24.1 (Feb. 1979), pp. 36–50. DOI: 10.1109/TAC.1979.1101943.

[128] A. Aswani, H. Gonzalez, et al. "Provably Safe and Robust Learning-Based Model Predictive Control". In: *Automatica* 49.5 (May 2013), pp. 1216–1226. DOI: 10.1016/j.automatica.2013.02.003.

[129] B. J. Naasz. "Safety Ellipse Motion with Coarse Sun Angle Optimization". In: *Proc. of the NASA GSFC Flight Mechanics Symposium*. Greenbelt, MD, Oct. 2005, pp. 1–13.

[130] M. Develle, Y. Xu, et al. "Fast Relative Guidance Approach for Autonomous Rendezvous and Docking Control". In: *Proc. of SPIE*. Vol. 8044. Sensors and Systems for Space Applications IV. Orlando, FL, Apr. 2011, 80440F.1–80440F.15. DOI: 10.1117/12.885943.

[131] S. D'Amico. "Autonomous Formation Flying in Low Earth Orbit". PhD thesis. Delft University of Technology, Mar. 2010.

[132] I. Lopez and C. R. McInnes. "Autonomous Rendezvous using Artificial Potential Function Guidance". In: *AIAA Journal of Guidance, Control, and Dynamics* 18.2 (Mar. 1995), pp. 237–241. DOI: 10.2514/3.21375.

[133] J. D. Muñoz and N. G. Fitz-Coy. "Rapid Path-Planning Options for Autonomous Proximity Operations of Spacecraft". In: *AAS Astrodynamics Specialist Conference*. Toronto, Canada, Aug. 2010, pp. 1–24. DOI: 10.2514/6.2010-7667.

[134] N. Martinson. "Obstacle Avoidance Guidance and Control Algorithms for Spacecraft Maneuvers". In: *AIAA Conf. on Guidance, Navigation and Control*. 5672. Chicago, IL, Aug. 2009, pp. 1–9. DOI: 10.2514/6.2009-5672.

[135] R. Vazquez, F. Gavilan, et al. "Trajectory Planning for Spacecraft Rendezvous with On/Off Thrusters". In: *IFAC World Congress*. Vol. 18. 1. Milano, Italy, Aug. 2011, pp. 8473–8478. DOI: 10.3182/20110828-6-IT-1002.02445.

[136] P. Lu and X. Liu. "Autonomous Trajectory Planning for Rendezvous and Proximity Operations by Conic Optimization". In: *AIAA Journal of Guidance, Control, and Dynamics* 36.2 (Mar. 2013), pp. 375–389. DOI: 10.2514/1.58436.

[137] X. Liu and P. Lu. "Solving Nonconvex Optimal Control Problems by Convex Optimization". In: *AIAA Journal of Guidance, Control, and Dynamics* 37.3 (May 2014), pp. 750–765. DOI: 10.2514/1.62110.

[138] Y.-Z. Luo, L.-B. Liang, et al. "Quantitative Performance for Spacecraft Rendezvous Trajectory Safety". In: *AIAA Journal of Guidance, Control, and Dynamics* 34.4 (July 2011), pp. 1264–1269. DOI: 10.2514/1.52041.

[139] M. Haught and G. Duncan. *Modeling Common Cause Failures of Thrusters on ISS Visiting Vehicles*. Tech. rep. Available at http://ntrs.nasa.gov/search.jsp?R=20140004797. NASA, Jan. 2014.

[140] J. M. Carson, B. Açikmeşe, et al. "A Robust Model Predictive Control Algorithm with a Reactive Safety Mode". In: *Automatica* 49.5 (May 2013), pp. 1251–1260. DOI: 10.1016/j.automatica.2013.02.025.

[141] S. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

[142] W. H. Clohessy and R. S. Wiltshire. "Terminal Guidance System for Satellite Rendezvous". In: *AIAA Journal of the Aerospace Sciences* 27.9 (Sept. 1960), pp. 653–658. DOI: 10.2514/8.8704.

[143] G. W. Hill. "Researches in the Lunar Theory". In: *JSTOR American Journal of Mathematics* 1.1 (1878), pp. 5–26. DOI: 10.2307/2369430.

[144] M. Bodson. "Evaluation of Optimization Methods for Control Allocation". In: *AIAA Journal of Guidance, Control, and Dynamics* 25.4 (July 2002), pp. 703–711. DOI: 10.2514/2.4937.

[145] B. W. Barbee, J. R. Carpenter, et al. "A Guidance and Navigation Strategy for Rendezvous and Proximity Operations with a Noncooperative Spacecraft in Geosynchronous Orbit". In: *AIAA Journal of the Aerospace Sciences* 58.3 (July 2011), pp. 389–408. DOI: 10.1007/BF03321176.

[146] T. Schouwenaars, J. P. How, et al. "Decentralized Cooperative Trajectory Planning of Multiple Aircraft with Hard Safety Guarantees". In: *AIAA Conf. on Guidance, Navigation and Control*. Providence, RI, Aug. 2004, pp. 1–14. DOI: 10.2514/6.2004-5141.

[147] T. Fraichard. "A Short Paper about Motion Safety". In: *Proc. IEEE Conf. on Robotics and Automation*. Roma, Italy, Apr. 2007, pp. 1140–1145. DOI: 10.1109/ROBOT.2007.363138.

[148] A. Celletti and S. S. Ferraz-Mello, eds. *Periodic, Quasi-Periodic and Chaotic Motions in Celestial Mechanics: Theory and Applications*. Norwell, MA: Springer, 2006, p. 438. DOI: 10.1007/978-1-4020-5325-2.

[149] E. Kolemen, N. J. Kasdin, et al. "Quasi-Periodic Orbits of the Restricted Three-Body Problem Made Easy". In: *New Trends in Astrodynamics and Applications III*. Vol. 886. 68. Princeton, NJ: AIP, Aug. 2007, pp. 68–77. DOI: 10.1063/1.2710044.

[150] J. P. LaSalle. "Some Extensions of Liapunov's Second Method". In: *IRE Transactions on Circuit Theory* 7.4 (Dec. 1960), pp. 520–527. DOI: 10.1109/TCT.1960.1086720.

[151] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Ed. by J. E. Marsden, L. Sirovich, et al. 1st ed. Vol. 10. New York, NY: Springer Science & Business Media, Apr. 2013, p. 668. DOI: 10.1007/978-1-4757-3108-8.

[152] T. A. Burton. *Stability & Periodic Solutions of Ordinary & Functional Differential Equations*. Mineola, NY: Courier Corporation, June 2014, p. 352.

[153] J. D. Biggs, V. M. Becerra, et al. "A Search for Invariant Relative Satellite Motion". In: *Int. Workshop on Satellite Constellations and Formation Flying*. Sao Jose dos Campos, Brazil, 2005, pp. 203–213.

[154] J. W. Mitchell and D. L. Richardson. "Invariant Manifold Tracking for First-Order Nonlinear Hill's Equations". In: *AIAA Journal of Guidance, Control, and Dynamics* 26.4 (July 2003), pp. 622–627. DOI: 10.2514/2.5090.

[155] K. Alfriend, S. R. Vadali, et al. *Spacecraft Formation Flying: Dynamics, Control and Navigation*. Vol. 2. Butterworth-Heinemann, 2009.

[156] R. Allen and M. Pavone. "A Real-Time Framework for Kinodynamic Planning with Application to Quadrotor Obstacle Avoidance". In: *AIAA Conf. on Guidance, Navigation and Control*. San Diego, CA, Jan. 2016, pp. 1–18. DOI: 10.2514/6.2016-1374.

[157] S. N. Goward, J. G. Masek, et al. "The Landsat 7 Mission: Terrestrial Research and Applications for the 21st Century". In: *Remote Sensing of Environment* 78.1–2 (2001), pp. 3–12. DOI: 10.1016/S0034-4257(01)00262-0.

[158] M. Grant and S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming, Version 2.1*. http://cvxr.com/cvx. June 2015.

[159] H.-X. Baoyin and J. Li. "A Survey on Orbital Dynamics and Navigation of Asteroid Missions". In: *Acta Mechanica Sinica* 30.3 (May 2014), pp. 282–293. DOI: 10.1007/s10409-014-0035-8.

[160] D. J. Scheeres. "Close Proximity Operations for Implementing Mitigation Strategies". In: *Planetary Defense Conference*. Orange County, CA: AIAA, Feb. 2004, pp. 1–11. DOI: 10.2514/6.2004-1445.

[161] E. Morrow, D. J. Scheeres, et al. "Solar Sail Orbit Operations at Asteroids". In: *AIAA Journal of Spacecraft and Rockets* 38.2 (Mar. 2001), pp. 279–286. DOI: 10.2514/2.3682.

[162] M. Leomanni, E. Rogers, et al. "Explicit Model Predictive Control Approach for Low-Thrust Spacecraft Proximity Operations". In: *AIAA Journal of Guidance, Control, and Dynamics* 37.6 (Aug. 2014), pp. 1780–1790. DOI: 10.2514/1.G000477.

[163] D. P. Scharf, F. Y. Hadaegh, et al. "A Survey of Spacecraft Formation Flying Guidance and Control, Part II: Control". In: *American Control Conference*. Vol. 4. Boston, MA, June 2004, pp. 2976–2985.

[164] J. Reif and M. Sharir. "Motion Planning in the Presence of Moving Obstacles". In: *Journal of the Association for Computing Machinery* 41.4 (July 1994), pp. 764–790. DOI: 10.1145/179812.179911.

[165] K. I. Tsianos and L. E. Kavraki. "Replanning: A Powerful Planning Strategy for Hard Kinodynamic Problems". In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. Nice, France, Sept. 2008, pp. 1667–1672. DOI: 10.1109/IROS.2008.4650965.

[166] P. Fiorini and Z. Shiller. "Motion Planning in Dynamic Environments Using Velocity Obstacles". In: *International Journal of Robotics Research* 17.7 (July 1998), pp. 760–772. DOI: 10.1177/027836499801700706.

[167] J. Van Den Berg, D. Ferguson, et al. "Anytime Path Planning and Replanning in Dynamic Environments". In: *Proc. IEEE Conf. on Robotics and Automation*. Orlando, FL, May 2006, pp. 2366–2371. DOI: 10.1109/ROBOT.2006.1642056.

[168] J. Bruce and M. Veloso. "Real-time Randomized Path Planning for Robot Navigation". In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. Vol. 3. Lausanne, Switzerland, Oct. 2002, pp. 2383–2388. DOI: 10.1109/IRDS.2002.1041624.

[169] M. Likhachev, D. I. Ferguson, et al. "Anytime Dynamic A*: An Anytime, Replanning Algorithm." In: *Int. Conference on Automated Planning and Scheduling*. Monterey, CA, June 2005, pp. 262–271.

[170] D. Ferguson, N. Kalra, et al. "Replanning with RRTs". In: *Proc. IEEE Conf. on Robotics and Automation*. Orlando, FL, May 2006, pp. 1243–1248. DOI: 10.1109/ROBOT.2006.1641879.

[171] M. Ono and B. C. Williams. "An Efficient Motion Planning Algorithm for Stochastic Dynamic Systems with Constraints on Probability of Failure." In: *Proc. AAAI Conf. on Artificial Intelligence*. Chicago, IL, July 2008, pp. 1376–1382.

[172] Y. Chow and M. Pavone. "Stochastic Optimal Control with Dynamic, Time-Consistent Risk Constraints". In: *American Control Conference*. 2013, pp. 390–395. DOI: 10.1109/ACC.2013.6579868.

[173] D. Gallardo, R. Bevilacqua, et al. "Advances on a 6 Degrees of Freedom Testbed for Autonomous Satellites Operations". In: *AIAA Conf. on Guidance, Navigation and Control*. 6591. Portland, OR, Aug. 2011, pp. 1–17. DOI: 10.2514/6.2011-6591.

[174] S. P. Viswanathan, A. Sanyal, et al. "Dynamics and Control of a Six Degrees of Freedom Ground Simulator for Autonomous Rendezvous and Proximity Operation of Spacecraft". In: *AIAA Conf. on Guidance, Navigation and Control*. 4926. Minneapolis, MN, Aug. 2012, pp. 1–19. DOI: 10.2514/6.2012-4926.

[175] M. Wilde, B. Kaplinger, et al. "ORION: A Teaching and Research Platform for Simulation of Space Proximity Operations". In: *AIAA SPACE Conferences & Exposition*. 4427. Pasadena, CA, Aug. 2015, pp. 1–10. DOI: 10.2514/6.2015-4427.

[176] J. Doebbler, J. J. Davis, et al. "Mobile Robotic System for Ground Testing of Multi-Spacecraft Proximity Operations". In: *AIAA Modeling and Simulation Technologies Conference*. Vol. 6548. Honolulu, HI, Aug. 2008, pp. 1–8. DOI: 10.2514/6.2008-6548.

[177] G. Gaias, J.-S. Ardaens, et al. "Formation Flying Testbed at DLR's German Space Operations Center (GSOC)". In: *Int. ESA Conf. on Guidance, Navigation & Control Systems*. Karlovy Vary, Czech Republic, June 2011, pp. 1–14.

[178] S. Nolet, E. Kong, et al. "Autonomous Docking Algorithm Development and Experimentation Using the SPHERES Testbed". In: *SPIE Spacecraft Platforms and Infrastructure*. Orlando, FL, Apr. 2004, pp. 1–15. DOI: 10.1117/12.547430.

[179] S. Nolet. "The SPHERES Navigation System: From Early Development to On-Orbit Testing". In: *AIAA Conf. on Guidance, Navigation and Control*. 6354. Hilton Head, SC, Aug. 2007, pp. 1–17. DOI: 10.2514/6.2007-6354.