SCHOOL OF ENGINEERING AND DESIGN
— AEROSPACE

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Aerospace

Sensor-based optimal control for an Astrobee robot on the ISS

Tommaso Faraci

SCHOOL OF ENGINEERING AND DESIGN
— AEROSPACE

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Aerospace

Sensor-based optimal control for an Astrobee robot on the ISS

Sensorgesteuerte optimale Steuerung für einen ASTROBEE-Roboter auf der ISS

Author:	Tommaso Faraci
Supervisors:	Prof. Dr. Markus Ryll, Prof. Dr. Stefan Leutenegger
Advisors:	Dr. Roberto Lampariello, M.Sc. Caroline Specht
Submission Date:	22.08.2023

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 22.08.2023 Tommaso Faraci

Acknowledgments

This endeavor would not have been possible without the guidance of Dr. Roberto Lampariello and M.Sc. Caroline Specth from the DLR. Providing their insights and expertise, they have allowed me to fuel my passion while growing professionally and personally.

I would like to express my deepest appreciation to Prof. Markus Ryll and Prof. Stefan Leutenegger, who have welcomed this thesis and have allowed me to broaden my perspective with their suggestions and feedback.

Finally, I'm extremely grateful to my girlfriend Valentina and my family, as they all sacrificed a lot for me to pursue my goals and dreams. A special thought goes to my grandmother, who would have loved to support me through this journey but couldn't.

Abstract

Autonomous robots hold the potential to revolutionize various domains of science, from space and planetary exploration to search and rescue missions. By harnessing their ability to navigate and interact within complex environments, their application has been studied to solve the increasing problem of recovering orbital debris. In the pursuit to obtain a robust and reliable methodology to service malfunctioning tumbling satellites in orbit, the challenge posed by localization and pose estimation while performing complex maneuvers has arisen.

To further the capabilities of autonomous flying robots, this thesis proposes a novel method to integrate kinodynamic path planning with perception-aware strategies. This research is driven by the aspiration to attain globally optimal solutions for autonomous guidance by capitalizing on the comprehensive understanding of a known environment.

The approach unites the constraint compliance and global optimality of gradient-based optimization and sampling-based kinodynamic path planning, with the use of visual information to obtain a kinodynamic perception-aware path planning algorithm. By fusing these paradigms, the deployed algorithm allows robots to navigate through scenarios in $SE(3)$ while exploiting the rich description of their surroundings for both obstacle avoidance and improvement of localization.

This thesis advances the landscape of visual-based planning by devising a perception-aware framework to address the problem of autonomous flight within the International Space Station (ISS). The methodologies hereby devised are not limited to space applications, but can find numerous avenues for future applications for holonomic robots in general.

In the following, the RRT*-GBO algorithm is employed to leverage the combination of sampling-based path planning with gradient-based optimization. This methodology is able to handle highly non-convex and nonlinear problems and will be used to tackle the path-planning problem in $SE(3)$ for holonomic robots, such as Astrobees and robot manipulators.

The method will be then extended by formulating a feature-density function to efficiently plan kinodynamic trajectories by accounting for vision-based cost and constraints formulation.

Both approaches are demonstrated in applications on simulated environments based on real data from the ISS. The perception-aware path planning method is then compared

with appropriate reformulation of state-of-the-art approaches in $SE(3)$, demonstrating superior performance with a simplified and more flexible approach.

Contents

1 Introduction

Space robotics has gained significant popularity in recent years. As the environment of space does not allow humans to easily operate in it, the need for autonomous vehicles is imperative to sustain the expansion that the market is showing in this field.

The increasing use of satellites and the launchers necessary to place them in orbit, leaves a substantial amount of debris, cluttering the environment and posing significant risks to any object within it. Moreover, a growing number of retired, malfunctioning or fuel depleted satellites remains in high-orbit, threatening to completely lose high-value satellites even for minor failure events.

Servicing or recovering these free-floating bodies is extremely relevant and efforts are being made to efficiently achieve this goal.

However, the challenging case of space debris tumbling uncontrollably has required additional investigation. In order to resolve this increasingly occurring and dangerous situation, the Deutsches Zentrum für Luft- und Raumfahrt (DLR) has been researching methodologies to use autonomous satellites to retrieve these spacecrafts. Often times, the motion of the freely rotating debris is unknown, thus requiring the development of a reliable pipeline to first identify the target's motion, in for the purpose of approaching it.

The pipeline has been implemented and tested in the ROAM/Tumbledock experiment [Alb+22; Alb+21]. The methodology employs visual estimation of the target motion, followed by the identification of a suitable mating point in space. A path planning step follows, in order to identify the motion the robot needs to undergo to reach the rendezvous position. Finally, the found trajectory is fed to a Tube-based Model Predictive Control (MPC) strategy, which can account for uncertainty bounds providing robustness and reliability to the method.

In order to test and validate the pipeline, it has been deployed on the ISS, using two free-flying robots on board, the Astrobees, shown in Fig. 1.1.

The experiment was designed to command motion on one of the robots, in order to simulate a tumbling target with a given inertia. This target system was to then be approached by another robot, acting as chaser. The chaser's approach and mating procedure towards the target is shown in Fig. 1.2.

During the experimental campaign, however, while in motion to approach the target, the astrobee chaser experienced localization failures. This event was later prevented by

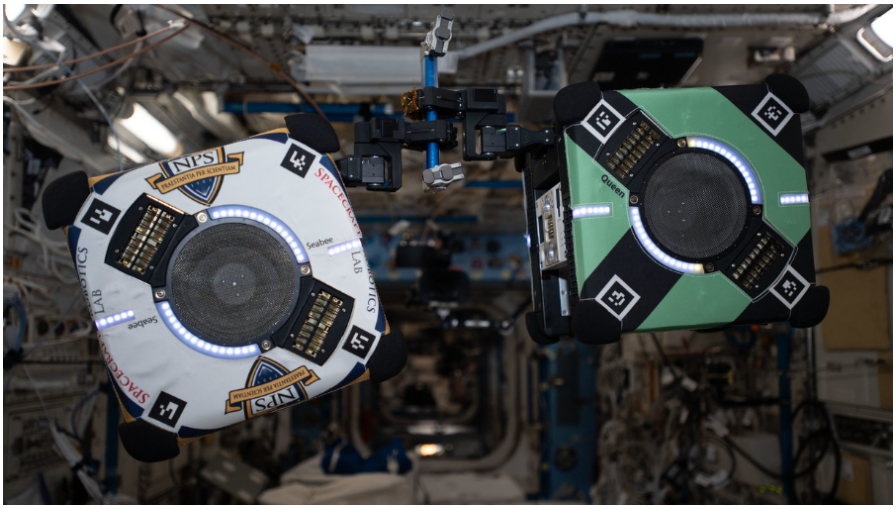


Figure 1.1: Photograph of two astrobees grasping an object. From NASA.

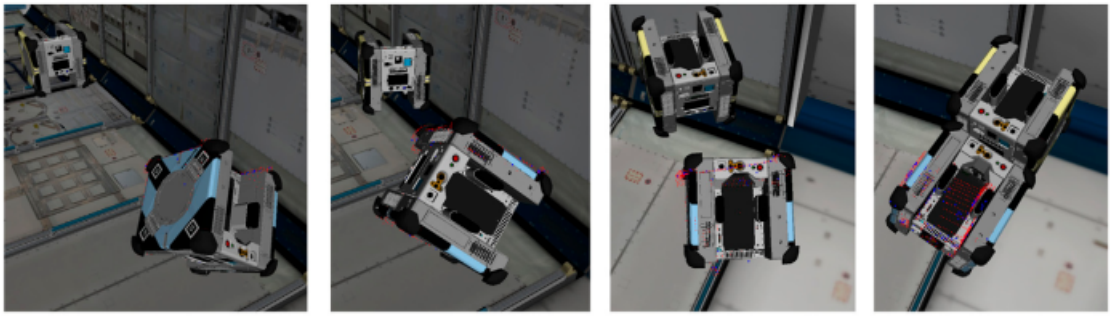


Figure 1.2: A time-lapse of the rendezvous portion of the pipeline for an arbitrary tri-axial tumble within simulation environment, from [Alb+22].

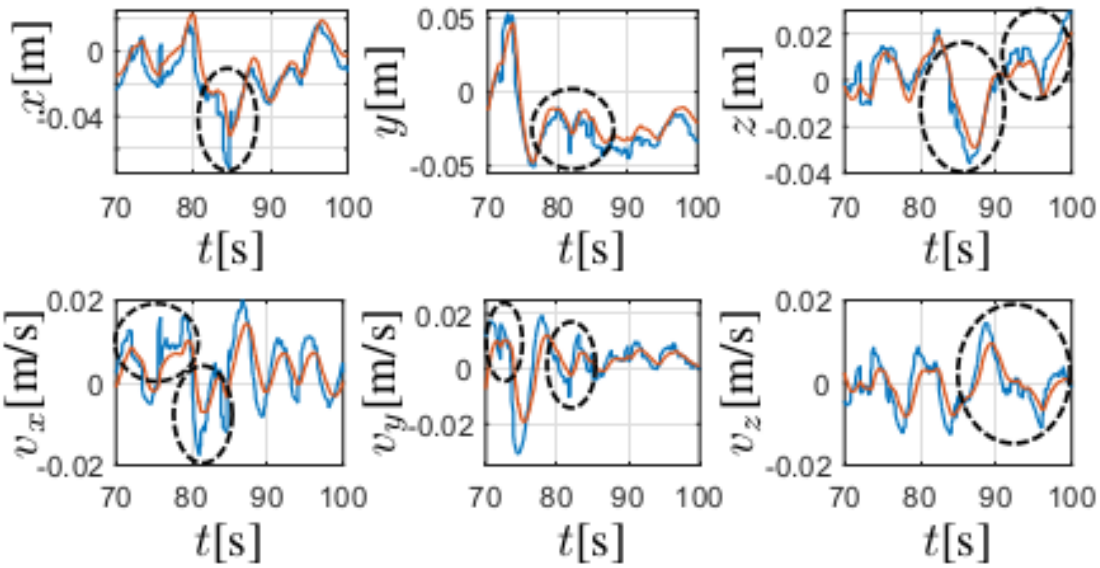


Figure 1.3: Results of estimated position and velocities during the experimental campaign. The AstroLoc [Sou+22] is represented in blue, while in red is the model-based EKF. Of notice are the discontinuities in the localization results, which are removed by the EKF. From [Alb+22].

adding a model-based Extended Kalman Filter (EKF) on top of the existing localization pipeline of the system [Sou+22]. The depiction of the difference between the localization result and the EKF for position and velocity is shown in Fig. 1.3.

While this solution has been effective, the reason for such of a relevant disruption in the localization remains unclear.

1.1 Objectives of the thesis

The purpose of this thesis is motivated by the search of a solution for the issue in the visual-based localization pointed out above. Of relevance is the $3D$ motion of a free-flying robot, as depicted in Fig. 1.1. In particular, the objective is to define a path planning pipeline that can effectively leverage a sparse description of the operational environment of the robot in order to obtain a globally optimal solution for different cost formulations, with focus on those which are perception-related.

The scope of this thesis is the implementation of a kinodynamic path planning approach for solving highly constrained planning problems, in combination with the necessity of identifying a viable metric that can be correlated to the performance of the

localization.

Finally, the identified planner has to be adapted and modified in order to allow testing on a robotic manipulator, in order to reproduce the free-flying dynamics on Earth.

As the targeted optimization problem is expected to be highly non-convex in nature, the research is directed towards kinodynamic sampling-based methodologies, which have proven to be efficient in the most general nonlinear and non-convex constrained path planning cases.

In order to provide an efficient methodology to perform sampling-based path planning on manifolds such as $SO(3)$ and $SE(3)$, the RRT*-GBO is implemented by augmenting the sampling-based method to the space of rigid body motions.

The same methodology is then applied to perception-aware path planning on sparse maps by formulating a continuous and differentiable feature-density function. The maps are assumed to be accurate and already available. These are containers for a list of $3D$ distinct and recognizable points, which represent landmarks detected during the mapping procedure.

The proposed algorithm is compared to the currently available state-of-the-art method, to showcase its capabilities and advantages.

These methods are implemented and applied to simulated environments representing real-life scenarios.

1.2 Contents of the thesis

The contents described above are laid out in the following fashion:

Chapter 2 introduces the models of the system and environment which are used throughout the work.

In Chapter 3, an analysis of the kinodynamic path planning problem is performed, studying literature and presenting a novel approach to plan optimal trajectories in $SO(3)$ and $SE(3)$ within the RRT*-GBO algorithm. Following the theoretical background for kinodynamic path planning, Chapter 4 introduces a description of the path planning problem based on visual information. Then, the problem of VIL is addressed and an available state-of-the-art method is analyzed. An alternative approach to perform perception-aware planning in $SE(3)$ is devised, in order to cope with the identified shortcomings of the method found in literature. The two are then compared after appropriately reformulating the state-of-the-art approach to be applied in $SE(3)$.

Chapter 5 presents a thorough report of the results obtained during the development of this thesis.

Finally, Chapter 6 concludes the thesis, summarizing the contents and proposing further research questions tied to the topic.

2 System Modeling

2.1 Free-Flying robot

In this section, the focus will be the definition of the system dynamics and components of the Astrobee robots on board of the ISS. In particular, the objective of this work is to design methodologies that can be applied in the context of experiments such as Tumbledock [Alb+22].

2.1.1 System Dynamics

The robot moves as a free-floating body in tridimensional space. As it is a free-floating body on-board the ISS, which orbits earth at a given rotational speed $\boldsymbol{\omega}_{ISS}$, the equations of motions in the body frame are

$$m\mathbf{a} = \mathbf{F} - \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{p}) - 2\boldsymbol{\omega} \times \mathbf{v} - \dot{\boldsymbol{\omega}} \times \mathbf{p}, \tag{2.1}$$

where $\mathbf{a}$ is the translational acceleration, $\mathbf{F}$ is the total force vector acting on the body, $\boldsymbol{\omega}$ is the angular velocity, $\mathbf{p}$ is the position with respect to the rotating frame and $\dot{\boldsymbol{\omega}}$ is the angular acceleration. In this specific case, the angular velocity comprises of the following terms $\boldsymbol{\omega} = \boldsymbol{\omega}_B + \boldsymbol{\omega}_{ISS} + \boldsymbol{\omega}_E$, where B stands for body and E stands for earth. Considering $\boldsymbol{\omega}_{ISS} = 0.0011803\frac{rad}{s}$ and $\boldsymbol{\omega}_E = 7.2921159 \times 10^{-5}\frac{rad}{s}$, the rotation of the Earth can be neglected, while the rotation of the ISS could have an effect on the robot dynamics. To be consistent with the description used by [Alb+22; WSK16], the system dynamics will be considered as described by [AEO20]:

$$\mathbf{a} = \frac{\mathbf{F}}{m} \tag{2.2}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}[\boldsymbol{\tau} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}], \tag{2.3}$$

where $\boldsymbol{\tau}$ is the torque applied to the system and $\mathbf{I}$ is the moment of inertia. In the case of the Astrobee, these quantities are:

- $m = 9.58kg$
- $\mathbf{I} = \begin{bmatrix} 0.153 & 0 & 0 \\ 0 & 0.143 & 0 \\ 0 & 0 & 0.162 \end{bmatrix} kg\,m.$

For path planning purposes, the focus of Eq. (2.3) is shifted on the actuation, therefore the *inverse dynamics* are defined as

$$\mathbf{u} = f^{-1}(\mathbf{x}, \dot{\mathbf{x}}) = \begin{bmatrix} m\mathbf{a} \\ \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} \end{bmatrix}. \tag{2.4}$$

In Eq. (2.4), the variable $\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{R}, \boldsymbol{\omega}]$ represents the state of the system and $\mathbf{u} = [\mathbf{F}, \boldsymbol{\tau}]$ is the actuation.

2.1.2 Dynamic constraints

In the context of path planning, it is desirable to define limit values for states and actuation. The following are listed in [AEO20]:

- $\mathbf{p}_x \in [0, 1.5]m$, $\mathbf{p}_y \in [0, 6.4]m$ and $\mathbf{p}_z \in [0, 1.7]m$,
- $\mathbf{v}_i \in [-0.1, 0.1]\frac{m}{s}$, for $i = x, y, z,$,
- $\mathbf{F}_x \in [-0.849, 0.849]N$, $\mathbf{F}_y \in [-0.406, 0.406]N$ and $\mathbf{F}_z \in [-0.486, 0.486]N$,
- $\boldsymbol{\omega}_i \in [-0.1, 0.1]\frac{rad}{s}$, for $i = x, y, z,$
- $\boldsymbol{\tau}_x \in [-0.0849, 0.0849]N$, $\boldsymbol{\tau}_y \in [-0.0406, 0.0406]N$ and $\boldsymbol{\tau}_z \in [-0.0486, 0.0486]N$.

The robots are actuated using a set of 12 thrusters, the forces are mapped onto the actuators through the built-in Force Allocation Module (FAM) [AEO20].

2.2 Robot manipulator

In the context of this thesis, it is interesting to also embed the free-flyer dynamics onto an holonomic robotic arm.

The robots dynamics are defined through the classical Lagrangian formulation

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{F}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \boldsymbol{\tau}, \tag{2.5}$$

where $\boldsymbol{\theta}$ corresponds to th joint angles of the manipulator, $\mathbf{M}(\boldsymbol{\theta})$ is the inertia matrix, $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ represents the Coriolis matrix and $\mathbf{F}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ contains all external forces on the system.

Additionally, the forwards kinematics can be defined as a function that maps the joint angles into a tridimensional pose $\mathbf{T}$ of the end effector, which is assumed to be the free-flyer in this case:

$$\mathbf{k} : \mathbb{R}^n \rightarrow SE(3). \tag{2.6}$$

Inverting the mapping, the inverse kinematics are obtained as the function that delivers joint angles from a tridimensional pose

$$\mathbf{k}^{-1} : SE(3) \to \mathbb{R}^n. \tag{2.7}$$

In the case of this work, it is assumed that both functions are known and available.

The robot manipulator in analysis in this thesis is the On-Orbit Servicing Simulator (OOS-SIM) [Art+15]. In particular a Kuka KR120 industrial robot, with 6 degrees of freedom.

The robot's layout is shown in Fig. 2.1, robotic arm's end effector on the left acts as the Astrobee chaser. The robot on the right can be used to move acting as a target, or positioned to increase the available space for the left arm. A light weight robot is positioned on top of the target and the camera attached to it can be therefore moved to act as an onboard fixed monocular camera.

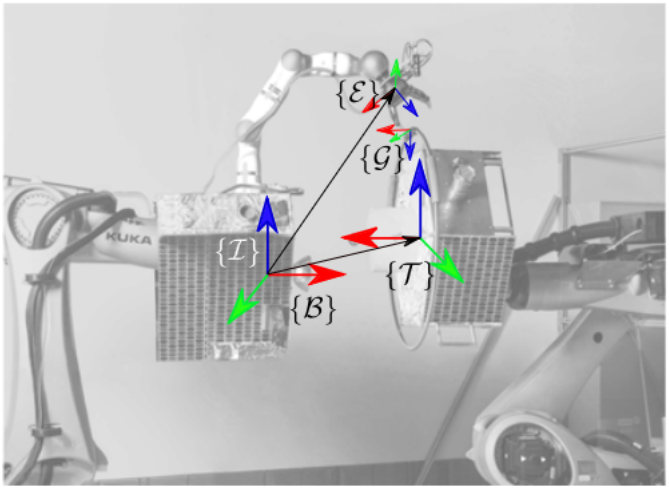


Figure 2.1: OOS-SIM arms and end effector's positions with relative position of each element's frame. From [Lam+18]. The chaser (left) carries a light weight robot, which position is assumed fixed within in this thesis. The light weight robot's frame $\mathcal{E}$ can be considered as the frame of of the camera. The chaser's body frame $\mathcal{B}$ initially coincides with the inertial frame $\mathcal{I}$. The target (right) with its frame $\mathcal{T}$ allows to be moved to simulate a rotating target and is equipped with the possibility to be grasped at the origin of frame $\mathcal{G}$.

2.3 Collision models

Within this work, it is necessary to simulate tridimensional collision between solid objects to incorporate the obstacle avoidance feature. In order to do that, the Bullet Collision library [Cou15] is employed.

The foundation of the Bullet collision model lies in the definition of shapes and objects. Shapes define the geometric form of objects, ranging from simple geometries such as spheres and boxes, to more complex geometries like convex hulls. These encapsulate the geometrical characteristics of objects, such as their size and proportions. A collision object then combines a collision shape with transformation information, such as position and orientation. This holistic representation forms a complete description of an object's collision properties, enabling it to be incorporated into any environment.

By representing robots and obstacles as collision objects, it is possible to simulate the robots pose in an environment by linking it to a specified object. In particular, the library allows to define a *Penetration depth*. This quantity defines the minimum distance that two objects have to move in order to not overlap. This quantity can be used as a measure of distance and a discriminant to define collision. The penetration depth is considered positive when the objects are colliding and negative if the objects are not in contact. Therefore the condition of obstacle avoidance can easily be formulated as a scalar inequality in the form

$$PD + |t_c| \leq 0, \tag{2.8}$$

where t_c represents a safety value that can be added to include obstacle inflation. In Figure 2.2, a sketch of the computation of the penetration depth computation is denoted. In the image, the obstacle on the right can be displaced vertically or horizontally to avoid collision. As the penetration in horizontal direction is the shorter one, that is the distance taken into consideration. This behavior determines non-linearity in the constraint, thus using simple convex shapes such as boxes or spheres is beneficial.

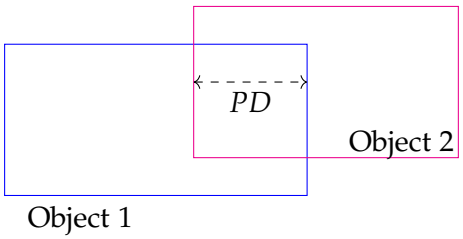


Figure 2.2: Sketch of two overlapping objects where the penetration depth is highlighted.

2.4 Camera model

As the autonomous free-flyers rely on a monocular camera to perform localization, it is important to model its parameters correctly.

The camera model used is a simple pinhole projection model [HZ03]

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}_C|\mathbf{p}_C] \begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix}, \tag{2.9}$$

where $[\mathbf{R}_C|\mathbf{p}_C]$ is a block matrix containing the camera orientation and position. The vector $[x_W, y_W, z_W, 1]$ represents the homogeneous coordinates of the point to project in the world frame. The quantities u, v corresponds to the pixel measurements in the image plane of the projected point. The matrix $\mathbf{K}$ is the intrinsic calibration matrix of the camera and in the case of the Astrobee it corresponds to

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 607.64218 & 0.0 & 624.85552 \\ 0.0 & 606.53899 & 514.57737 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, \tag{2.10}$$

where f_x, f_y are the focal lengths in pixel and (c_u, c_v) is position of the camera center in pixel coordinates.

The camera is positioned on the robot at position $\mathbf{p}_{CB} = [0.1177, -0.0422, -0.0826]$ with respect to the origin of the body frame. The orientation of the two frame is differentiated only by change of axis, such that the z-axis of the camera points in the direction of x_B. An approximate representation of the relative positions of the two frames is shown in Fig. 2.3.

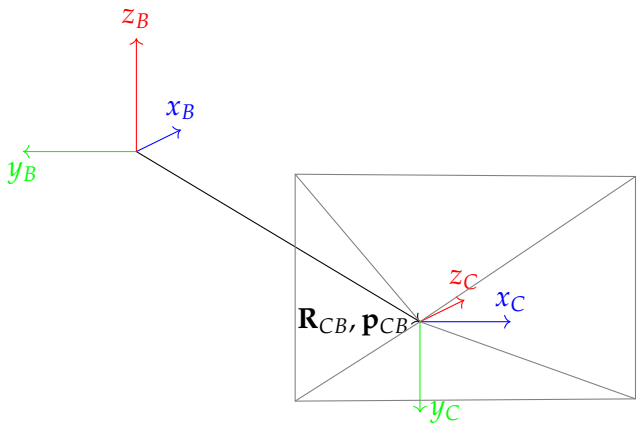


Figure 2.3: Relative representation of camera frame and body frame.

3 Kinodynamic Path Planning

As explained in Chapter 1, this work revolves around developing an efficient path planning framework on $SE(3)$ to address the problem of determining a trajectory for an Astrobee in an highly constrained environment. In particular, the Astrobee robot described in Section 2.1 moves within a small enclosure and can exploit all the 6 degrees of freedom of the system. Additionally, the robot has tight constraints with respect to dynamics, actuation and obstacle avoidance. As denoted in the Introduction, the challenges of space robotics are numerous, one of which is testing, as on-orbit testing facilities are only a few and scarcely available. Therefore, setting up tests on Earth is imperative to improve the design loop of any system to be deployed on-orbit. This calls for the necessity of extending any formulation done for the astrobee, to the OOS-SIM described in Section 2.2.

In the following sections, the problem of kinodynamic path planning is introduced formally. The numerous approaches devised over the years are analyzed and compared. Afterwards, the kinodynamic path planning problem is approached attempting to combine the common paradigms of sampling-based planning and trajectory optimization. The algorithm of RRT*-GBO is exploited in order to expand the research on its applications and extended onto the space of rigid body motions. Finally, the extension of the algorithm for robotic manipulators is discussed.

3.1 Problem Statement

3.1.1 Introduction

Kinodynamic path planning is an extremely relevant topic, as it plays a crucial role in the domain of robotics, especially related to autonomous vehicles and motion planning in general. The methods used assure finding optimal and feasible paths within a dynamic environment, in order to allow safe navigation even in highly constrained settings. Currently available methods, however, face significant challenges - particularly planning in highly dimensional spaces, handling of uncertainty, difficulty in incorporating complex dynamics, and reliance on heuristics that deliver sub-optimal results.

3.1.2 Related Work

Popular approaches to path planning in robotics can be divided in two major categories: graph-search and sampling-based methods.

Graph-search approaches that have been widely applied are Dijkstra's [Dij59] and A* [HNR68]. These methods rely on the theoretical foundations of dynamic programming [Bel54] to find exact solutions for discretized continuous problems. Due to their formulation, these algorithms are *resolution complete* and *resolution optimal*, meaning that they will find an exact and optimal solution for the given discretization in a finite time, if one exists.

A* leverages heuristics to search the graph and results in a *optimally efficient* formulation, guaranteeing that any other algorithm using the same heuristic will expand at least the same number of vertices as A*. While the solution guarantees these desirable properties, the quality of the continuous path obtained from the discretized graph is highly dependent on the discretization. Refining the discretization also results in largely extending the size of the problem and thus the computational effort required. These methods suffer the so-called *curse of dimensionality* [Bel66] when planning in highly dimensional space. Cases such as planning for robot manipulators or mobile robots in $3D$ are prime examples.

These algorithms have nonetheless been successful in kinodynamic planning tasks [Che99] when the dimensional space is limited. In [Zho+19], the authors employ an hybrid-state A* to generate a safe and feasible initial trajectory by generating motion primitives on the discretized control space. The motions are obtained by propagating the control inputs through the quadrotor dynamics represented as a Linear Time Invariant (LTI) system. As the solution of the A* algorithm is suboptimal in the continuous control space, it is further smoothed using gradient-based optimization in order to increase clearance of obstacles and smoothness. The authors from [Zho+19] employ non-uniform B-splines and apply motion constraints by leveraging their convex hull property [PBP02], as well as managing time allocation. The approach works well in confined and small spaces, especially for quadrotors which are proven to be differentially flat. As such, the system dynamics can be represented through a set of flat outputs and each of them can be treated independently. Moreover, the flat outputs are 4 parameters (position and yaw angle) which accounts for a significant reduction of the state space.

Sampling-based planners, such as Probabilistic Roadmaps (PRM) [Kav+96] and RRT [LJ01], avoid the issues related to discretization of the state space by randomly sampling the continuous domain. These approaches allows efficient planning in high dimensional spaces, but render the search probabilistic rather than exact. These methods are *probabilistically complete*- the probability of finding a solution, if that exists,

converges to one as the number of samples grows towards infinity. These methods also offer *anytime resolution*- the problem representation grows more accurate with each sampling step, allowing the user to achieve suboptimal solutions early on, which can be improved with increasing iteration numbers.

An additional step towards optimality is taken with algorithms such as RRT* and PRM* [KF11]. These variations converge to the optimal solution in infinite time, they are therefore *asymptotically optimal*. It is important to note, however, how these algorithms do not have any guarantee on the rate of convergence, as the sampling is random.

An alternative hybrid method that fuses aspects of graph-search and sampling-based methods, has been proposed, Batch Informed Trees (BIT)* [GSB14]. The algorithm has successfully been tested on $\mathbb{R}^2$ and $\mathbb{R}^8$ with robotic arms.

One of the most common path planning paradigms for mobile robots has been a decoupled approach based on initially finding a collision-free geometric path through sampling-based techniques, to then smooth it using polynomial splines and B-splines [RBR16; Ole+16]. Approaches such as the one from [MK11] have showcased the flexibility of pairing the sampling-based path planning with smoothing techniques. This approach, while efficient and effective, might initially deliver unfeasible solutions as it does not account for the dynamics of the robot.

To extend the kinodynamic properties of the original formulation of RRT*, which was only valid for holonomic robots, [WB13] devised the kinodynamic RRT*, which guarantees asymptotic optimality for any system with controllable linear dynamics and is valid in any dimension. The algorithm incorporates the system dynamics within a two-point boundary value problem. The analytical solution of the problem delivers the optimal control policy along an edge of the tree. The use of linear dynamics, however, limits the algorithm to a small class of applications which require validity of linear approximations. To further the scope of applicability of this method, [Zhi+20] devised an updated kinodynamic RRT* where the motion primitives are obtained by solving an optimal control problem. The explicit solution of the optimal duration for the motion primitives is given to optimally connect any pair of states. The methodology is effective as it outperforms the original formulation in terms of exploring more states while generating smoother trajectories.

The search to construct an all-purpose kinodynamic sampling-based methodology has been pushed forward by the RRT*-GBO [SL14]. In order to account for any kind of nonlinear system dynamics as well as integrating nonlinear constraints within the edge computation. The method outperforms the standard RRT* by generating a feasible close-to-optimal initial solution, which can then be smoothed, reaching an optimal solution in shorter time than classic two-step RRT approaches. The technique is suitable for offline applications as it has shown to be computationally expensive, thus requiring long times to converge to a solution. Although only tested in simulation with holonomic

systems, the method is not restricted by any underlying assumption and is therefore general in its applicability.

3.1.3 Problem Description

In the context of a free-flying systems, the path planning problem of interest requires the construction of a path for the full motion in the three-dimensional space. The problem is highly constrained and it requires satisfaction of constraints on position, orientation, velocities, accelerations and actuation, while searching for an optimal trajectory within a fixed time horizon.

The problem is approached by attempting to find a globally optimal solution for the path planning task of an Astrobee robot through an enclosed path with obstacles. In particular, formulating a problem which resembles the Tumbledock experiment, by considering a confined section of the ISS, planning the motion around a rotating obstacle in order to reach a defined meeting point at a selected time.

For testing purposes, it is necessary to include inverse kinematics of a robotic manipulator in order to plan the motion of its end effector as if it represented the free-flyer itself. Thus the implementation requires enough flexibility to be applicable to different robotic systems without substantially modifying the underlying optimization problem.

With the same framework, the problem is extended by introducing the kinematics constraints of a robot manipulator with 6 degrees of freedom.

Objectives

The primary objective of this chapter is to address the problem of optimal kinodynamic path planning on $SE(3)$ for a free-flying robot and for a robotic manipulator's end effector. Mathematically, the problem entails solving an optimization on a non-convex submanifold of $SE(3)$. In order to efficiently search the highly dimensional and non-convex space, we apply a sampling-based method to efficiently search it. To the best of the author's knowledge, the only methodology that allows to obtain globally optimal solutions accounting for arbitrary constraints in a one-step fashion is the RRT*-GBO. The goal within this chapter is to extend the algorithm to the six degrees of freedom necessary to describe elements of $SE(3)$. Exploiting these extension, the successive step is to include the inverse kinematics of a robotic manipulator in the problem.

Contributions

Within this chapter the following challenges will be investigated:

- Application of GBO on manifolds.
- Efficient and uniform random sampling on $SO(3)$.
- Extension of the RRT*-GBO on $SE(3)$.
- Formulation of a novel tree-edge construction to decrease the computational time for the expected use cases.
- Application of the methodology on Astrobee in simulation and on a Kuka robot in simulation and experiment.

3.2 Methodology

3.2.1 Overview

Within this section, the detailed explanation of the methods used are given. In order to reach the objectives described in section 3.1.3, a thorough investigation of the theoretical results and previous research on the topic is conducted, as well as expanded through the necessary adaptations to the case at hand. The contents which will be discussed can be summarized as:

- optimal path-planning using GBO on B-Splines,
- parametrization and optimization on manifolds,
- application of sampling-based methods on manifolds,
- formulation of RRT*-GBO on $SE(3)$,
- conceptualization of an alternative edge formulation to reduce the search space and computational time.

3.2.2 Optimal path planning on manifolds with interpolating curves

As mentioned in 3.1.2, using optimization to describe a trajectory has proven to be a very effective and valuable technique in motion planning.

The motion planning problem under consideration can be formulated as a nonlinear Optimal Control Problem (OCP) as follows:

$$\min_{\mathbf{x}(t)} \Gamma(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t), \mathbf{u}(t)) \tag{3.1a}$$

subject to

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.1b}$$
$$\mathbf{x}(t_f) = \mathbf{x}_f \tag{3.1c}$$
$$\frac{\partial^i \mathbf{x}(0)}{\partial t^i} = {\mathbf{c_0}}^{(i)} \text{ for } i = 1, \ldots, p+1 \tag{3.1d}$$
$$\frac{\partial^i \mathbf{x}(t_f)}{\partial t^i} = {\mathbf{c_{t_f}}}^{(i)} \text{ for } i = 1, \ldots, p+1 \tag{3.1e}$$
$$\mathbf{x}_{min} \leq \mathbf{x}(t) \leq \mathbf{x}_{max} \tag{3.1f}$$
$$\dot{\mathbf{x}}_{min} \leq \dot{\mathbf{x}}(t) \leq \dot{\mathbf{x}}_{max} \tag{3.1g}$$
$$\ddot{\mathbf{x}}_{min} \leq \ddot{\mathbf{x}}(t) \leq \ddot{\mathbf{x}}_{max} \tag{3.1h}$$
$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \tag{3.1i}$$
$$\mathbf{g}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t), \mathbf{u}(t)) \leq 0. \tag{3.1j}$$

where equation (3.1a) corresponds the objective function to minimize, which we consider as a general function of the state $\mathbf{x}$ and its derivatives $\dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t)$. Additionally, the function can depend on the control inputs of the system $\mathbf{u}(t)$, which are correlated to the states through the system's inverse dynamics $\mathbf{u} = f^{-1}(\mathbf{x}(t), \dot{\mathbf{x}}(t))$. Equations (3.1b) and (3.1c) correspond to boundary conditions on the state itself, while equations (3.1d) and (3.1e) are boundary constraints on state derivatives, up to order $p+1$. The values ${\mathbf{c_0}}^{(i)}$ and ${\mathbf{c_{t_f}}}^{(i)}$ are the assigned values to be assigned at the boundaries. Equations (3.1f) - (3.1i) represent bounding box constraints for the system dynamics and actuation. Finally equation (3.1j) includes additional nonlinear constraints such as collision avoidance and/or inverse kinematics, generalized in the vector-valued function $\mathbf{g}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t), \mathbf{u}(t))$.

For the scope of this work, the objective function is considered to be an integral in the form:

$$\Gamma(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t), \mathbf{u}(t)) = \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t), \mathbf{u}(t)) dt \tag{3.2}$$

In this chapter, the cost formulation is related to the mechanical energy, formulated as the integral of the squared power consumption over the trajectory in [SL14] and [SL16]:

$$\mathcal{L}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t), \mathbf{u}(t)) = (\mathbf{u} \cdot \dot{\mathbf{x}})^2. \tag{3.3}$$

Numerical Optimization

The OCP defined in eq. (3.1) can be reformulated as a Nonlinear Programming (NLP) problem as in [NW06]:

$$\min_{\mathbf{x}} f(\mathbf{x}(\mathbf{p})) \tag{3.4a}$$

subject to

$$\mathbf{h}_i(\mathbf{x}(\mathbf{p})) = \mathbf{0},\ i = 1, \ldots, N \tag{3.4b}$$
$$\mathbf{g}_i(\mathbf{x}(\mathbf{p})) \leq \mathbf{0},\ i = 1, \ldots, N \tag{3.4c}$$

where $\mathbf{x}$ corresponds to the optimization variable defined with respect to a parameter $p \in \mathbb{R}^N$, f represents the objective function and $\mathbf{h}_i$ and $\mathbf{g}_i$ are equality and inequality vector constraints defined at points p, respectively.

As Equation (3.4) cannot, in general, be solved analytically, a numerical method to approximate the result is employed. Sequential Quadratic Progamming (SQP) [NW06] is an iterative method that allows to solve nonlinear constrained optimization problems efficiently. The method approximates the objective function quadratically, as well as using a first order approximation of the constraints. Considering the optimization variable $\mathbf{x} \in \mathbb{R}^n$ at an iterative step k, one obtains a Quadratic Progamming (QP) subproblem in the form:

$$\min_{\mathbf{x}} f_k + \nabla f_k^T p + \frac{1}{2} p^T \nabla^2 f_k p \tag{3.5a}$$

subject to

$$\mathbf{h}(\mathbf{x}^k) + \nabla \mathbf{h}(\mathbf{x}^k)^T p = \mathbf{0} \tag{3.5b}$$
$$\mathbf{g}(\mathbf{x}^k) + \nabla \mathbf{g}(\mathbf{x}^k)^T p \leq \mathbf{0}, \tag{3.5c}$$

where the subscript k refers to the value of the function at step k, while p represents the search direction. The solution is then formulated as $(\mathbf{x}^k + p^k, \lambda^{k+1})$ where p^k, λ^{k+1} are the solution and its corresponding Lagrange multiplier for equation (3.5).

In practice, however, the SQP method requires the solution of a constrained QP at each iteration. The cost of this approach limits the size of the solvable optimization, additionally, the use of Hessians introduces a number of issues that can jeopardize the success of the algorithm. The Hessian matrix can be computationally expensive to calculate, especially for large-scale optimization problems with numerous variables and constraints. Moreover, calculating second-order derivatives numerically can be prone to numerical instability and accuracy issues, especially when dealing with

complex or ill-conditioned functions. Finally, storing such a matrix can require large amounts of memory. To alleviate these issues, we employ the Sequential Least Squares Programming (SLSQP) [Kra94] method. Without loss of generality, as inequality constraints can be reformulated as equalities, the focus is on the equality constrained QP in the form

$$\min_{\mathbf{x}} f_k + \nabla f_k^T p + \frac{1}{2} p^T \nabla^2 f_k p \tag{3.6a}$$

subject to

$$A_k p + c_k = 0 \tag{3.6b}$$

the problem's solution (p_k, l_k) is unique and satisfies the system of Newton equations from [NW06]:

$$\begin{bmatrix} \nabla^2 f_k & -A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} p_k \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -\nabla f_k \\ -c_k \end{bmatrix} \tag{3.7}$$

in which $l_k = \lambda_k$. The new iterate (x_{k+1}, λ_{k+1}) can therefore be defined either as the solution of the quadratic program or as the iterate generated by Newton's method applied to the optimality conditions of the problem.

Examining equation (3.7), the part of the step p_k in the range space of A_k^T is completely determined by the second block row $A_k p_k = -c_k$. The Hessian only affects p_k in the null space of A_k, therefore it is useful to consider quasi-Newton methods that find approximations only to the part of $\nabla^2 f_k$ that affects the component of p_k in the null space of the constraint matrix. Defining matrices Z_k, Y_k whose columns respectively span the null space of A_k and the range space of A_k^T, the step can be rewritten as:

$$p_k = Y_k p_Y + Z_k p_Z. \tag{3.8}$$

Substituting this expression in equation 3.7, the system becomes:

$$(A_k Y_k) p_Y = -c_k \tag{3.9a}$$

$$(Z_k^T \nabla^2 f_k Z_k) p_Z = Z_k^T \nabla^2 f_k Y_k p_Y - Z_k \nabla f_k. \tag{3.9b}$$

and the Lagrange multipliers λ_{k+1} are computed as

$$(A_k Y_k)^T \lambda_{k+1} = Y_k^T (\nabla f_k + \nabla^2 f_k). \tag{3.10}$$

One can define the least-squares multipliers by choosing $Y_k = A_k$, to obtain

$$\hat{\lambda}_{k+1} = (A_k A_k^T)^{-1} A_k \nabla f_k. \tag{3.11}$$

These reformulation is equivalent to solving the problem

$$\min \|\nabla \mathcal{L}(x_k, \lambda)\|_2^2 = \|\nabla f_k - A_k^T \lambda\|. \tag{3.12}$$

One additional simplification based on the null-space can be obtained by removing the cross term $Z_k^T \nabla^2 f_k Y_k p_Y$, obtaining the simpler equation:

$$(Z_k^T \nabla^2 f_k Z_k) p_Z = -Z_k \nabla f_k. \tag{3.13}$$

As a last step of the approximation, the term $Z_k^T \nabla^2 f_k Z_k$ needs to be reformulated. Suppose having taken a step $\alpha_k p_k = x_{k+1} - x_k = \alpha_k Z_k p_Z + \alpha_k Y_k p_Y$. Considering $\nabla^2 f_{k+1} = \nabla^2 f_k(x_{k+1}, \lambda_{k+1})$, through Taylor's theorem:

$$\nabla^2 f_{k+1} \alpha_k p_k \approx \nabla \mathcal{L}(x_k + \alpha_k p_k, \lambda_{k+1}) - \nabla \mathcal{L}(x_k, \lambda_{k+1}). \tag{3.14}$$

Pre-multiplying by Z_k^T the expression becomes:

$$Z_k^T \nabla^2 f_{k+1} Z_k \alpha_k p_k \approx -Z_k^T \nabla^2 f_{k+1} Y_k \alpha_k p_Y - Z_k^T (\nabla \mathcal{L}(x_k + \alpha_k p_k, \lambda_{k+1}) - \nabla \mathcal{L}(x_k, \lambda_{k+1})). \tag{3.15}$$

Finally one can completely eliminate the Hessian by removing the cross term $Z_k^T \nabla^2 f_k Y_k p_Y$ as done above. The elimination leads to a final reformulation of the equation (3.7), which can be expressed as the secant equation [NW06]:

$$M_{k+1} s_k = y_k \tag{3.16}$$

where $s_k = \alpha_k p_k$ and $y_k = Z_k^T [\nabla \mathcal{L}(x_k + \alpha_k p_k, \lambda_{k+1}) - \nabla \mathcal{L}(x_k, \lambda_{k+1})]$. In order to obtain the matrix M_{k+1}, the problem is solved assuming the solution to be the matrix that is closest to the one at the previous step M_k, in this fashion:

$$\min_M \|M - M_k\| \tag{3.17a}$$

subject to

$$M s_k = y_k \tag{3.17b}$$

$$M = M^T \tag{3.17c}$$

where the normed difference corresponds to the Frobenius norm. The solution is obtainable using the BFGS [NW06] method, which iteratively delivers H_{k+1} assuming that

$$H_k = \nabla^2 f_k = M_k. \tag{3.18}$$

This series of considerations and approximations allow to solve an SQP without resorting to computing second derivatives. However, the is approximation leads to an extreme relevance on accurate gradient information for the solution of the optimization problem. Throughout this work, we will use the implementation made available within the NLopt [Joh07] open-source library, which is widely used and offers efficient implementation of the algorithms in C and C++.

Optimization on B-splines

The space of states on which the optimal solution is to be found is infinite in the most general case. In order to limit the size of the optimization problem to render it treatable, a generic n-dimensional trajectory $\mathbf{s}(t)$ is parametrized as a polynomial spline of selected degree p.

The B-splines are piecewise polynomial functions which are widely used for interpolation and approximation of general multivariate functions. Their formulation is advantageous as it ensures continuity and smoothness up to the $p+1$'s derivative. The function is defined on a set of knots t_i that form a knot vector $\mathbf{t}$. The spline is then defined as a linear combination of basis functions B_j^p and control points $\mathbf{p}_j$. The basis functions only depend on the knots and can be computed recursively with the De Boor's formula [Boo77]:

$$B_j^0 = \begin{cases} 1, \text{ if } t_j \leq t < t_{j+1} \\ 0, \text{ otherwise} \end{cases} \tag{3.19}$$

$$B_j^p = \frac{t - t_j}{t_{j+p} - t_j} B_j^{p-1}(t) + \frac{t_{j+p+1} - t}{t_{j+p+1} - t_{j+1}} B_{j+1}^{p-1}(t). \tag{3.20}$$

The value of the B-spline at a given value within the knot vector is:

$$\mathbf{s}(t) = \sum_{j=0}^{m} \mathbf{p}_j B_j^p(t). \tag{3.21}$$

In order to apply the algorithms defined in 3.2.2 on a trajectory optimization problem, one can introduce the state space parametrized as a B-spline, which knot vectors spans the times between t_0 to t_f. Within the optimization framework, this is equivalent as choosing the optimization variable $\mathbf{x}$ as the vector of control points $\mathbf{P} = [\mathbf{p}_0, \ldots, \mathbf{p}_m]$ of size $m+1$.

By using a clamped B-spline of order $p+1$, one can conveniently incorporate boundary conditions in the B-spline by adding knots of multiplicity $p+1$ corresponding to the starting and ending points of the trajectory. The resulting knot vector is the following sequence of size $n_{knot}+1$ where $n_{knot} = m+p+1$:

$$\mathbf{t} = \left[\underbrace{t_0, ..., t_0}_{\text{p+1}}, t_1, t_2, \ldots, t_{n_{knot}-2}, t_{n_{knot}-1}, \underbrace{t_f, \ldots, t_f}_{\text{p+1}} \right]. \tag{3.22}$$

By selecting the first and last control points to correspond to the state boundary

conditions

$$\mathbf{p}_0 = \mathbf{x}(t_0) \tag{3.23}$$
$$\mathbf{p}_m = \mathbf{x}(t_f) \tag{3.24}$$

the optimization problem can be reduced by removing a set of $2 * n$ optimization variables as well as constraints (3.1b) and (3.1c). Since B-splines are continuous and smooth polynomials of order $p + 1$, one can ensure existence and continuity of their derivatives up to the $p - k$-th one at any knot of multiplicity k. Conveniently, one can formulate the k-th derivative as a function of the same control points, through:

$$\mathbf{s}^{(k)}(t) = \sum_{j=0}^{m} \mathbf{p}_j B_j^{p(k)}(t). \tag{3.25}$$

The new basis functions are defined as:

$$B_j^{p(k)}(t) = \frac{p!}{(p-k)!} \sum_{i=0}^{k} a_{k,i} B_{j+1}^{p-k} \tag{3.26}$$

with:

$$a_{0,0} = 1 \tag{3.27a}$$
$$a_{k,0} = \frac{a_{k-1,0}}{t_{j+p-k+1} - t_j} \tag{3.27b}$$
$$a_{k,i} = \frac{a_{k-1,i} - a_{k-1,i-1}}{t_{j+p+i-k+1} - t_{j+1}} \text{ for } i = 1, \ldots, k-1 \tag{3.27c}$$
$$a_{k,k} = \frac{-a_{k-1,k-1}}{t_{j+p+1} - t_{j+k}} \tag{3.27d}$$

The relation between B-splines and their derivatives is shown in Fig. 3.1. As the order of the derivative increases, the degree of the spline decreases accordingly.

As the basis functions only depend on the value of the knots and on the degree of the derivative, they can be computed without knowledge of the control points value. As this is the case, arbitrary values can be set to the trajectory as well as its time derivative, by constructing a linear system of equations for any time t_j, with $p + 1 \leq j \leq n_{knot} - (p + 1)$:

$$\mathbf{A}\mathbf{p} = \mathbf{c} \tag{3.28}$$

where

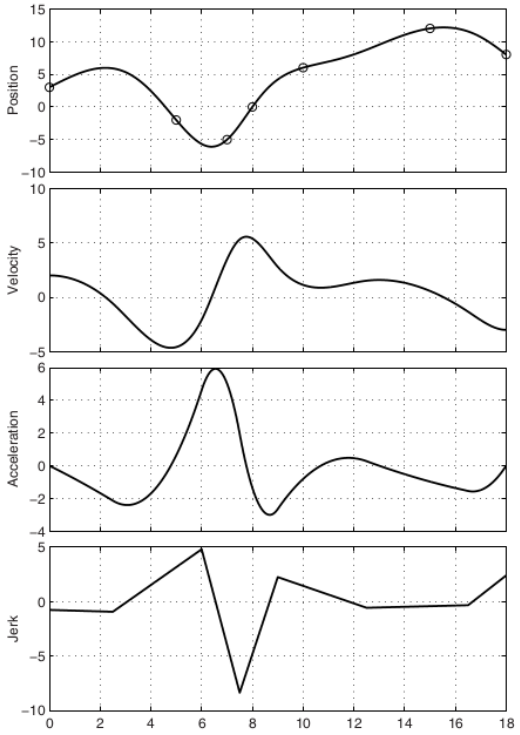


Figure 3.1: Example of the differentiability properties of a general trajectory exploiting B-splines. Using degree $p = 4$ one obtains continuity up to the jerk ($k = 4$). From [BM08].

$$\mathbf{A} = \begin{bmatrix} B_j^p(t_j), & B_{j+1}^p(t_j), & \dots, & B_{j+p+1}^p(t_j) \\ B_j^{p(1)}(t_j), & B_{j+1}^{p(1)}(t_j), & \dots, & B_{j+p+1}^{p(1)}(t_j) \\ \vdots & \vdots & \vdots & \vdots \\ B_j^{p(p+1)}(t_j), & B_{j+1}^{p(p+1)}(t_j), & \dots, & B_{j+p+1}^{p(p+1)}(t_j) \end{bmatrix} \mathbf{p} = \begin{bmatrix} p_j \\ p_{j+1} \\ \vdots \\ p_{j+p+1}. \end{bmatrix} \mathbf{c} = \begin{bmatrix} c_{t_j}^{(0)} \\ c_{t_j}^{(1)} \\ \vdots \\ c_{t_j}^{(p+1)}. \end{bmatrix} \tag{3.29}$$

In equation (3.29), the matrix $\mathbf{A}$ contains the known basis functions, the vector $\mathbf{p}$ contains the control points that are subject to the constraints and $\mathbf{c}$ is the vector of constraint values, where the superscript corresponds to the degree of the derivative that are to be fixed $c_{t_j}^{(k)} = \frac{\partial^k \mathbf{s}}{\partial t^k}$. By constructing two of such system of equations for t_0 and t_f, the vector valued boundary constraints (3.1d) and (3.1e) are integrated by defining the appropriate control points, thus reducing the number of optimization variables by

$p * n$. It is common for online motion planning formulations, to also embed constraints (3.1f)-(3.1i) directly in the formulation of the B-splines [Lam+11; Gon+16; Use+17]. Considering the derivative of a general B-spline, we can express it as a new curve with control points $\mathbf{Q} = [\mathbf{q}_0, \ldots, \mathbf{q}_{m-1}]$, defined as:

$$\frac{ds}{dt}(t) = \sum_{j=0}^{m-1} \mathbf{q}_j B_{j+1}^{p}(t) \tag{3.30}$$

where

$$\mathbf{q}_j = \frac{p+1}{t_{j+p+1} - t_{j+1}}(\mathbf{p}_{j+1} - \mathbf{p}_j). \tag{3.31}$$

Exploiting the convex hull property of the control points [Cha21], the trajectory can conservatively be defined to be confined within the kinodynamic constraints by selecting appropriate bounds. However, B-splines value pass through control points only if the multiplicity of the knots is higher than one. As the objective is to obtain the optimal trajectory given a set of constraints, the preferred approach is to enforce them directly on the spline values on a set of *via points* of size $n_{viaPoints}$, that uniformly span the knot vector (3.22). By increasing its size arbitrarily, one can more and more accurately verify that the trajectory never becomes unfeasible, at the expense of increasing computational effort.

Finally, one obtains the reformulated OCP that describes motion planning on a n-dimensional B-spline of order $p+1$, $\mathbf{s}(t)$, defined on the knot vector $\mathbf{t}$ with control points $\mathbf{P}$:

$$\min_{\mathbf{P_{opt}}} \Gamma(\mathbf{P_{opt}}) \tag{3.32a}$$

$$\textit{subject to} \tag{3.32b}$$

$$\mathbf{s}_{min} \leq \mathbf{s}(t) \leq \mathbf{s}_{max} \tag{3.32c}$$

$$\dot{\mathbf{s}}_{min} \leq \dot{\mathbf{s}}(t) \leq \dot{\mathbf{s}}_{max} \tag{3.32d}$$

$$\ddot{\mathbf{s}}_{min} \leq \ddot{\mathbf{s}}(t) \leq \ddot{\mathbf{s}}_{max} \tag{3.32e}$$

$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \tag{3.32f}$$

$$\mathbf{g}(\mathbf{s}(t), \dot{\mathbf{s}}(t), \ddot{\mathbf{s}}(t)) \leq 0 \tag{3.32g}$$

where $\mathbf{P_{opt}}$ represents the subset of control points between $\mathbf{p}_{p+1}$ and $\mathbf{p}_{m-p+1}$ that are involved in the optimization.

Optimization on the special orthogonal group $SO(3)$

To plan a sequence of attitudes and orientations and to construct smooth trajectories, applying the methodology described above is desirable. The attitude of a rigid body can be described as matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$. As not all matrices describe a rotation, the space of rotation matrices needs to be defined as $SO(3) \subset \mathbb{R}^{3\times3}$, mathematically detailed as the following:

$$SO(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times3} : \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = +1 \right\}. \tag{3.33}$$

The manifold fundamentally represents the set of matrices that are orthonormal with positive and unitary determinant.

The methods described above to perform GBO on a set of optimization variables $\mathbf{P}_{opt}$, are derived based on the underlying assumption that all variables live on the real coordinate space $\mathbb{R}^n$, thus all mathematical operations are contained within the space. In particular, when defining the numerical optimization methodology in Section 3.2.2, the solution was defined as an iterative summation of steps on a line p_k:

$$x_{k+1} = x_k + \alpha_k p_k. \tag{3.34}$$

However, when operating on the space of orientations $SO(3)$, the sum of two elements might not be an orthonormal 3×3 matrix with determinant $+1$. Trying performing a simple vector addition between two rotation matrices $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$, one has no guarantee that the solution will be contained within the original manifold. As in [DK17a], defining an increment on the manifold is achieved by leveraging the notion of the angle-axis representation. Considering a rotation axis $\bar{\omega} \in S^2$ and a rotation angle $\theta \in S^1$, a tridimensional vector $\boldsymbol{\xi} \in \mathbb{R}^3$ is obtained. The vector encapsulates both the angle and axis information by multiplying the two $\boldsymbol{\xi} \triangleq \theta\bar{\omega}$.

In order to construct a matrix from this vector, it is necessary to introduce the notion of the *hat map* $\hat{(.)}$, which operates by reformulating a vector in $\mathbb{R}^3$ into a skew-symmetric matrix, in this fashion:

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} 1 & -\xi_z & \xi_y \\ \xi_z & 1 & -\xi_x \\ -\xi_y & \xi_x & 1 \end{bmatrix}. \tag{3.35}$$

If $\boldsymbol{\xi}$ is approximately zero, the matrix is close to a member of $SO(3)$. However, it is desirable to define an exact mapping onto the manifold. Therefore the notion of matrix exponential is introduced, based on the formulation from [Hal00]:

$$\exp(\hat{\boldsymbol{\xi}}) \triangleq \sum_{k=0}^{\infty} \frac{1}{k!}\hat{\boldsymbol{\xi}}^k = I + \hat{\boldsymbol{\xi}} + \frac{1}{2!}\hat{\boldsymbol{\xi}}^2 + \frac{1}{3!}\hat{\boldsymbol{\xi}}^3 + \frac{1}{4!}\hat{\boldsymbol{\xi}}^4 + \dots \tag{3.36}$$

An ulterior definition, stemming from the matrix exponential above, is the *exponential map*. This mapping exploits a relevant feature of the group $SO(3)$. As it shares the properties of being a group, as well as a differentiable manifold, $SO(3)$ is a *Lie Group*. As such, the quantity $\hat{\boldsymbol{\xi}}$ can be described as a member of its *Lie Algebra* $\mathfrak{so}(3)$, which represents its tangent space. The exponential map is defined as the function that maps elements of the lie algebra to the lie group

$$\exp : \mathfrak{so}(3) \to SO(3). \tag{3.37}$$

This function is exact for arbitrarily large vectors and thus is very general. To construct a member of the $SO(3)$ manifold from a tridimensional vector, the matrix exponential has a convenient analytical expression, denominated Rodrigues's formula:

$$\exp(\hat{\boldsymbol{\xi}}) \triangleq \mathbf{I} + \frac{\sin(\theta)}{\theta}\hat{\boldsymbol{\xi}} + \frac{(1-\cos(\theta))}{\theta^2}\hat{\boldsymbol{\xi}}^2. \tag{3.38}$$

The exponential map is valid for any Lie group, thus the procedure hereby detailed is not constrained to $SO(3)$.

Being able to map elements of $\mathbb{R}^3$ to elements of $SO(3)$, any rotation can be interpreted as an incremental one, starting from a reference element of $SO(3)$, $\mathbf{R}_0$:

$$\mathbf{R}_1 = \mathbf{R}_0 \exp \hat{\boldsymbol{\xi}}_1. \tag{3.39}$$

Leveraging this formulation, expressing a sequence of rotations over time can be done by extending the formula above:

$$\mathbf{R}(t) = \mathbf{R}_0 \exp \hat{\boldsymbol{\xi}}(t). \tag{3.40}$$

Eq. (3.40) allows to recover the use of interpolating curves within the optimization framework.

B-splines parametrization of $SO(3)$

The main contribution of this section is the improvement of state-of-the-art path planning approaches for rotations, by means of tridimensional B-splines. In [NP16] and [BD18], the authors implement third order polynomials to construct a simple interpolation with four parameters only. The method presented in this thesis allows to use any arbitrary number of parameters to describe a trajectory in $SO(3)$. The sequence of rotations $\mathbf{R}(t)$ can be interpolated recovering the B-spline parametrization used in equation (3.21), by interpolating the vector $\boldsymbol{\xi}$:

$$\boldsymbol{\xi}(t) = \sum_{j=0}^{m} \mathbf{r}_j B_j^p(t). \tag{3.41}$$

Initial and final boundary conditions on the trajectory must be prescribed to fully define a path. However, it is not clear how to define boundary conditions on the vector $\boldsymbol{\xi}$. In particular, the objective is to constrain the following quantities:

- $\mathbf{R}_0, \mathbf{R}_{t_f}$
- $\boldsymbol{\omega}_0, \boldsymbol{\omega}_{t_f}$
- $\dot{\boldsymbol{\omega}}_0, \dot{\boldsymbol{\omega}}_{t_f}$

To this end, defining the mapping that transfers the notion of a rotation matrix into its Lie algebra becomes necessary. This mapping is denoted as the *logarithmic map*, being the inverse of the exponential one. This function is represented as $\log : SO(3) \to \mathfrak{so}(3)$. As detailed in [LP17], retrieving the angle-axis counterpart of a rotation matrix $\mathbf{R}$ requires identifying three cases:

1. if $\mathbf{R} = \mathbf{I}$ then $\theta = 0$ and $\bar{\omega}$ is undefined.
2. if $tr(\mathbf{R}) = -1$ then $\theta = \pi$ and the axis can be any of three solutions:

$$\bar{\omega} = \frac{1}{\sqrt{2(1 + r_{33})}} \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix} \tag{3.42}$$

$$\bar{\omega} = \frac{1}{\sqrt{2(1 + r_{22})}} \begin{bmatrix} r_{12} \\ 1 + r_{22} \\ r_{32} \end{bmatrix} \tag{3.43}$$

$$\bar{\omega} = \frac{1}{\sqrt{2(1 + r_{11})}} \begin{bmatrix} 1 + r_{11} \\ r_{21} \\ r_{31} \end{bmatrix} \tag{3.44}$$

   In this specific case, both $\bar{\omega}$ and $-\bar{\omega}$ are solutions.

3. if $\theta = \arccos(\frac{1}{2}(tr(\mathbf{R}) - 1)) \in [0, \pi)$, then

$$\hat{\bar{\omega}} = \frac{1}{2\sin(\theta)} \left[\mathbf{R} - \mathbf{R}^T \right] \tag{3.45}$$

With this set of formulae, the quantity $\boldsymbol{\xi}$ associated with a rotation matrix $\mathbf{R}$ is obtained by introducing the inverse of the hat operator. Defining the *reverse hat operator* as $(.)^{\vee} : \mathfrak{so}(3) \to \mathbb{R}^3$, which maps the non-diagonal elements of a skew symmetric matrix onto a three-dimensional vector.

This representation allows to parametrize any rotation matrix within $SO(3)$ as an axis-angle pair through its logarithmic map, however one cannot define relative rotations larger than 2π due to its singularity in π. None of the works of [BD18; NP16] address the singularity, nor they describe accurately how to obtain the logarithmic map of an orientation matrix. This singularity restricts the motion planning problem to relative rotations of an angle within $\theta \in [0, 2\pi]$. In the context of minimizing the mechanical energy, this formulation is advantageous. When obtaining the angle-axis representation of the final attitude, this methodology always delivers the representation with minimal angle, which can be interpreted as the shortest path on the manifold [Har+13]. This allows to define a well-posed initial solution for the path planning problem.

When it comes to defining the set of boundary conditions, the logarithmic map can be exploited to obtain the two values $\boldsymbol{\xi}_0, \boldsymbol{\xi}_{t_f}$, in particular, these values can be computed using the continuous and differentiable chart transition on $SO(3)$ as expressed in [Wat+20; Wat+18]:

$$\boldsymbol{\xi}_2 = \left[\log(\mathbf{R}_1^{-1}\mathbf{R}_2 \exp(\hat{\boldsymbol{\xi}}_1))\right]^{\vee}. \tag{3.46}$$

As the formula requires knowledge of $\boldsymbol{\xi}_1$, a possible and simple approach is to set it to $\boldsymbol{\xi}_1 = \mathbf{0}$. It is trivial to verify that the exponential map of the null vector returns the identity matrix, so $\exp(\hat{\boldsymbol{\xi}_1}) = \mathbf{I}$. This way, simplifying (3.46):

$$\boldsymbol{\xi}_2 = \left[\log(\mathbf{R}_1^{-1}\mathbf{R}_2)\right]^{\vee}. \tag{3.47}$$

Obtaining the boundary conditions means setting the values of the orientation parametrization vector as in [BD18]

$$\boldsymbol{\xi}_0 = \mathbf{0} \tag{3.48}$$

$$\boldsymbol{\xi}_{t_f} = \left[\log(\mathbf{R}_0^{-1}\mathbf{R}_{t_f})\right]^{\vee}. \tag{3.49}$$

Fixing angular velocities values can be done exploiting the notion of *right Jacobian* from [NP16]:

$$\mathbf{J}(\boldsymbol{\xi}) \triangleq \mathbf{I} + \frac{1 - \cos(\|\boldsymbol{\xi}\|)}{\|\boldsymbol{\xi}\|^2}\hat{\boldsymbol{\xi}} + \frac{(\|\boldsymbol{\xi}\| - \sin(\|\boldsymbol{\xi}\|))}{\|\boldsymbol{\xi}\|^3}\hat{\boldsymbol{\xi}}^2. \tag{3.50}$$

The matrix $\mathbf{J}(\boldsymbol{\xi})$ maps infinitesimal increments from the tangent space to the manifold, in this case it allows to map the first derivative of the interpolated vector onto the angular velocity as:

$$\boldsymbol{\omega} = \mathbf{J}(\boldsymbol{\xi})\dot{\boldsymbol{\xi}}. \tag{3.51}$$

Extending the reasoning to the angular acceleration, (3.51) can be differentiated to obtain a function of the vector $\boldsymbol{\xi}$, which can be expressed according to [NP16] as:

$$\dot{\boldsymbol{\omega}} = \mathbf{J}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \mathbf{C}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) \tag{3.52}$$

where

$$\begin{aligned}\mathbf{C}(\xi,\dot{\xi}) \triangleq & \frac{(\|\xi\| - \sin(\|\xi\|))}{\|\xi\|^3}\dot{\xi} \times (\xi \times \dot{\xi}) \\ & - \frac{(2\cos(\|\xi\|) + \|\xi\|\sin(\|\xi\|)) - 2}{\|\xi\|^4}\xi \cdot \dot{\xi}(\xi \times \dot{\xi}) \\ & + \frac{(3\sin(\|\xi\|) - \|\xi\|\cos(\|\xi\|)) - 2\|\xi\|}{\|\xi\|^5}\xi \cdot \dot{\xi}(\xi \times (\xi \times \dot{\xi})). \end{aligned} \tag{3.53}$$

With this set of formulae, the parametrization vector and its derivatives at time t_j can be determined by sequentially solving the following equations:

$$\xi_{t_j} = \left[\log(\mathbf{R}_0^{-1}\mathbf{R}_{t_j})\right]^{\vee} \tag{3.54}$$

$$\dot{\xi}_{t_j} = \mathbf{J}(\xi_{t_j})^{-1}\omega_{t_j} \tag{3.55}$$

$$\ddot{\xi}_{t_j} = \mathbf{J}(\xi_{t_j})^{-1}(\dot{\omega}_{t_j} - \mathbf{C}(\xi_{t_j}, \dot{\xi}_{t_j})). \tag{3.56}$$

The set of equations above allows to reformulate any condition $(\mathbf{R_{t_i}}, \omega_{t_i}, \dot{\omega}_{t_i})$ into a set of conditions $(\xi_{t_i}, \dot{\xi}_{t_i}, \ddot{\xi}_{t_i})$. Rewriting the angular motion variables as functions of the parametrization vector in $\mathbb{R}^3$ allows to recover the incorporation of boundary conditions within the B-spline formulation by constraining the control points, leveraging the same exact method detailed in equations (3.28)-(3.29). Moreover, exploiting equations (3.51) and (3.52), the values of angular velocity and acceleration can be computed as functions of $(\xi(t_i), \dot{\xi}(t_i), \ddot{\xi}(t_i))$. Therefore permitting to incorporate nonlinear inequality constraints on these values as well.

With these notions, an interpolation on $SO(3)$ has been constructed. The method allows to map a sequence of rotations with defined boundary constraints, by using only variables that live on $\mathbb{R}^3$. As explained in [NP16], using this representation to interpolate trajectories has the advantage of approximating the minimum acceleration profile, which implies that the underlying path is smooth and short. In particular, when $\omega_0 = \omega_{t_f} = 0$ the sequence yields the shortest path on the manifold. As done for a general B-spline, the OCP for path planning on $SO(3)$ can be expressed and resolved using GBO:

$$\min_{\mathbf{R_{opt}}} \Gamma(\mathbf{R_{opt}}) \tag{3.57a}$$

subject to

$$\boldsymbol{\xi}_{min} \leq \boldsymbol{\xi}(t) \leq \boldsymbol{\xi}_{max} \tag{3.57b}$$
$$\dot{\boldsymbol{\xi}}_{min} \leq \dot{\boldsymbol{\xi}}(t) \leq \dot{\boldsymbol{\xi}}_{max} \tag{3.57c}$$
$$\ddot{\boldsymbol{\xi}}_{min} \leq \ddot{\boldsymbol{\xi}}(t) \leq \ddot{\boldsymbol{\xi}}_{max} \tag{3.57d}$$
$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \tag{3.57e}$$
$$\mathbf{g}(\boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t), \ddot{\boldsymbol{\xi}}(t)) \leq 0 \tag{3.57f}$$

In this case, a tridimensional B-spline of degree $p = 2$ is taken into consideration, defined on knot vector $\mathbf{t}$ and control points $\mathbf{R}$, with $\mathbf{R_{opt}}$ representing the set of free control points that are subject to optimization.

Having defined the optimization parameters to be a set of vectors on $\mathbb{R}^3$, the techniques of numerical optimization detailed previously are available to approach the numerical solution.

This methodology presented allows to perform path planning efficiently on $SO(3)$. The same method can be also adapted to representations with quaternions, by using the formulas defined in [NM15]. The two share similarities as they both parametrize the rotations using an angle-axis representation and leverage the exponential map of $SO(3)$. However, the method devised by [NM15] is used to interpolate precomputed waypoint trajectories in $SO(3)$, thus not allowing to obtain optimal solutions without a precomputed set of orientations.

Finally, this method can be extended to higher derivatives such as the rotational jerk, defined in [ZKC98] as $\ddot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}} + \frac{1}{2}\boldsymbol{\omega} \times \dot{\boldsymbol{\omega}}$. By implementing the reformulation from [Som+20] one can efficiently compute time derivatives of the parameters, plus the gradients of the B-spline with respect to the control points. Due to time constraints, however, in the course of this thesis it was not possible to completely extend the formulation to incorporate this level of generality.

Optimization on the special Euclidean group $SE(3)$

Considering a rigid body in the free space, assuming an inertial frame $\{I\}$ and a frame fixed to the rigid body at point O', $\{B\}$. At any point in time, the pose of the rigid body can be described as an homogeneous transformation, which corresponds to the displacement of frame $\{B\}$ from frame $\{I\}$. This homogeneous transformation can be represented by a matrix, which is an element of the special Euclidean group of rigid body transformations in three dimension

$$SE(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} | \mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{p} \in \mathbb{R}^3, \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \right\} = \mathbb{R}^3 \times SO(3). \tag{3.58}$$

It can be shown [MLS17] that $SE(3)$ is a manifold as well as a group for matrix multiplication, which means that it is a Lie group as $SO(3)$. As for any other Lie group its Lie algebra $\mathfrak{se}(3)$ defines its tangent space

$$\mathfrak{se}(3) = \left\{ \begin{bmatrix} \mathbf{\Omega} & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix} | \mathbf{\Omega} \in \mathbb{R}^{3\times3}, \mathbf{v} \in \mathbb{R}^3, \mathbf{\Omega}^T = -\mathbf{\Omega} \right\} \tag{3.59}$$

where $\mathbf{\Omega} = \hat{\omega}$. The literature does not present results that significantly highlight an advantage in interpolating and optimizing elements of $SE(3)$ instead of using a split position and orientation $\mathbb{R}^3 \times SO(3)$ representation [NP16; Som+20]. Hence, combining two B-splines, one for the position $\mathbf{s}(t)$ and one for orientation $\boldsymbol{\xi}(t)$, to interpolate the rigid body motion is a viable option to efficiently parametrize the group without the need to modify the derivations obtained so far. Combining the methodologies described so far for $\mathbb{R}^3$ and $SO(3)$, the trajectory optimization framework on the manifold is obtained.

Considering the trajectory of a rigid body as $T(t) = (\mathbf{p}(t), \mathbf{R}(t))$, the OCP for trajectory optimization using the B-spline representations $\tau(t) = (\mathbf{s}(t), \boldsymbol{\xi}(t))$ can be formulated. Defining the control points as $\mathbf{Q} = (\mathbf{P}, \mathbf{R})$, where $\mathbf{Q_{opt}} = (\mathbf{P_{opt}}, \mathbf{R_{opt}})$. Finally, formulation for trajectory optimization on $SE(3)$ is

$$\min_{\mathbf{Q_{opt}}} \Gamma(\mathbf{Q_{opt}}) \tag{3.60a}$$

subject to

$$\boldsymbol{\tau}_{min} \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}_{max} \tag{3.60b}$$
$$\dot{\boldsymbol{\tau}}_{min} \leq \dot{\boldsymbol{\tau}}(t) \leq \dot{\boldsymbol{\tau}}_{max} \tag{3.60c}$$
$$\ddot{\boldsymbol{\tau}}_{min} \leq \ddot{\boldsymbol{\tau}}(t) \leq \ddot{\boldsymbol{\tau}}_{max} \tag{3.60d}$$
$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \tag{3.60e}$$
$$\mathbf{g}(\boldsymbol{\tau}(t), \dot{\boldsymbol{\tau}}(t), \ddot{\boldsymbol{\tau}}(t)) \leq 0 \tag{3.60f}$$

As a final note, Problems (3.32), (3.57) and (3.60) cannot be proven to be convex, as there are non-trivial non-linearities within the constraints and cost function. Thus, convergence to a global optimum cannot be proven and guaranteed. Most likely, the solution will converge towards a local minimum which will be highly dependent on the initial guess fed to the optimization as well as the specific problem solved.

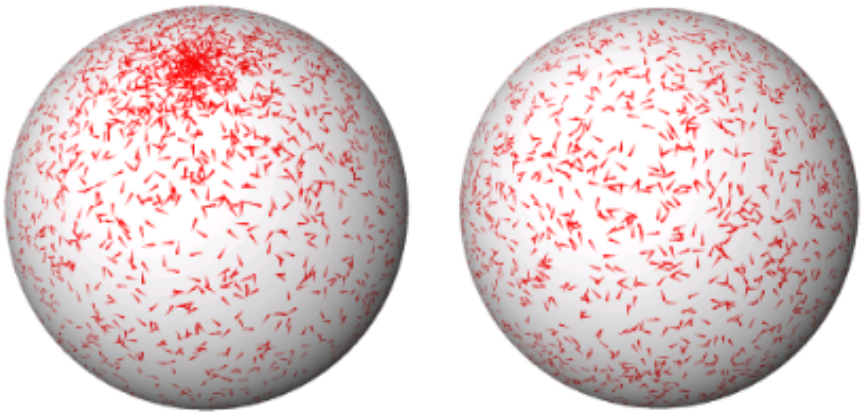
3.2.3 Sampling-based planning on manifolds

Most of the works mentioned in Section 3.1.2 avoid planning on the entire $SE(3)$. Applying simplifying reasoning on the dynamics of the robot or simplifying the

problem statement, in many cases the space of the trajectory can be conveniently reduced. The generation of randomly distributed samples on groups different from $\mathbb{R}^n$ is a non-trivial task. Mainly due to the complex topological structure of the group, which leads to difficulty in defining a distance metric on non-euclidean spaces. On n-dimensional cubes $\mathbb{R}^n$, using a pseudo-random generator delivers efficient sampling of the space. A viable alternative to improve coverage of the space, is using a deterministic sequence of quasi-random numbers. This approach offers the best results in terms of discrepancy and dispersion [LBL04]. The topology of the space $SE(3)$, induced by the rotation component of $SO(3)$, determines relevant implications for path planning algorithms which intend to efficiently represent and search the special euclidean group. Without careful implementation, these representational details can create a point of failure in the algorithm. In this section, the focus will be the sampling of $SO(3)$, as it is advantageous to consider $SE(3) = \mathbb{R}^3 \times SO(3)$.

Sampling of SO(3)

In [Kuf04], the authors point out that a naive approach when sampling Euler angles and unit quaternions would lead to samples which are strongly biased towards the polar regions. As an alternative, they propose an effective approach to randomly sample Euler angles and quaternions, as shown in Fig. 3.2.



(a) Naïve sampling (b) Uniform sampling

Figure 3.2: (a) Random sampling of Euler angles by only using a naive approach, (b) uniform sampling of $SO(3)$ with Euler angles, from [Kuf04].

Although effective, the method presented leverages rotation parametrization that are intrinsically limited. Euler angles are not natural representation of rotations due to their

singularities. On the other hand, quaternions do not allow to easily compute metrics to evaluate the quality of the sampling such as dispersion or discrepancy [Yer+10]. In [Yer+10], the authors point to the subgroup algorithm [DS87] as the most correct methodology to construct random sequences of rotations. The method exploits the property of any Lie group to be uniformly sampled by combining elements from a subgroup and its quotient, or coset space $S2$ at random. In the case of $SO(3)$, the approach can be applied by considering the subgroup $S1$ and its coset space $S2$. This approach is widely used, however it lacks deterministic uniformity guarantees, and the explicit neighborhood structure necessary to appropriately cover the entire space.

To obtain a sequence of rotations on $SO(3)$ that is uniform, covers the entire space and fulfills requirements on measures of discrepancy and distortion, the method devised by [Yer+10] is going to be followed within this work. The method requires construction of a deterministic incremental grid on $SO(3)$ by leveraging its topological structure, which allows it to locally behave as the Cartesian product of the two spaces $S1$ and $S2$. Using multiresolution grids on the 1-sphere and on the 2-sphere, they can then be lifted to the special orthogonal group by applying the Hopf coordinates. This set of coordinates permits to compute a quaternion associated with a given rotation. The quaternion in Hopf coordinates is expressed by considering the three angles (θ, ϕ, ψ), in which $\psi \in (0, 2\pi]$ parametrizes $S1$, while $\theta \in (0, \pi]$ and $\phi \in (0, 2\pi]$ represent the space $S2$. In Fig. 3.3 the effect of varying the Hopf fibration parameters on points sampled on the sphere is shown. The figure also depicts the difference between the spherical coordinates and the Hopf coordinates. This comparison allows to see why the Hopf coordinates do not present singularities, as they generate a series of non-intersecting arcs within the sphere. The figure shows the main advantages of using this parametrization, as the fibers are non-intersecting thus it presents no singularities, and the circles represented by the variable ψ are all of equal length. The rotation quaternion $\mathbf{q} = (q_1, q_2, q_3, q_4)$ from Hopf coordinates is then expressed as:

$$q_1 = \cos\frac{\theta}{2}\cos\frac{\psi}{2} \tag{3.61a}$$

$$q_2 = \cos\frac{\theta}{2}\sin\frac{\psi}{2} \tag{3.61b}$$

$$q_3 = \sin\frac{\theta}{2}\cos\left(\phi + \frac{\psi}{2}\right) \tag{3.61c}$$

$$q_3 = \sin\frac{\theta}{2}\sin\left(\phi + \frac{\psi}{2}\right). \tag{3.61d}$$

Defining a grid structure on $S1$ and $S2$ is the next step to randomly sample points from the two manifolds. Lifting this grid structures then allows to obtain a rotation

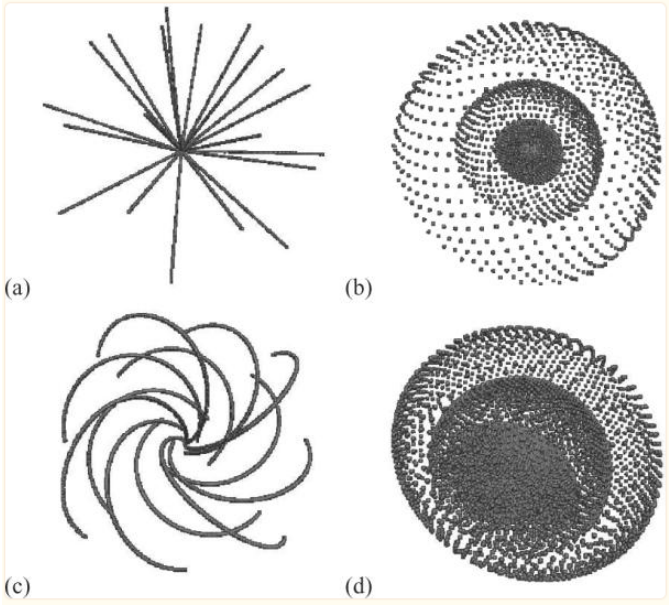


Figure 3.3: Visualization of the spherical(above) and Hopf (below) coordinates on $SO(3)$ using angle and axis representation by [Yer+10]. The depiction corresponds to a projection of the $S3$ onto the equatorial solid sphere drawn in $\mathbb{R}^3$. (a) Spherical coordinates with range of $\psi \in [0, \frac{\pi}{2}]$, where (θ, ϕ) form a discretization of size 20 in $S2$. (b) Half spheres show the full range of θ, ϕ while $\psi \in [0, \frac{\pi}{2}]$ takes 4 discrete values. (c) The range of Hopf coordinate with $\psi \in [0, 2\pi)$, where (θ, ϕ) form a discretization of size 12. (d) Half spheres showing the full range of the Hopf coordinate θ, ϕ while $\psi \in [0, 2\pi)$ takes 4 discrete values.

quaternion on $SO(3)$ which can be converted to a rotation matrix. As [Yer+10], the HEALPix package [Gor+05] is chosen to obtain a uniform grid on the 2-sphere. The package was originally developed for astrophysics and its purpose is to provide a deterministic, uniform, and multiresolution sampling method, additionally it partitions the sphere in sections of equal area. Figure 3.4 and 3.5 depict the partitioning of the baseline grid in squares in the space of Hopf coordinates, as well as the effect of subdividing the baseline grid of size $m_2 = 12$.

The HEALPix grid delivers the latitude and longitude of any point on the 2-sphere, thus delivering the Hopf coordinates (θ, ϕ). To complete the parametrization, an ordinary grid in $S1$ is constructed. Uniformly dividing the range of ψ in m_1 samples, a simple ordinary grid is obtained. [Yer+10] demonstrates that, in order to construct a grid in which cells have all equal size, the circumference of the circle $S1$ and the area of

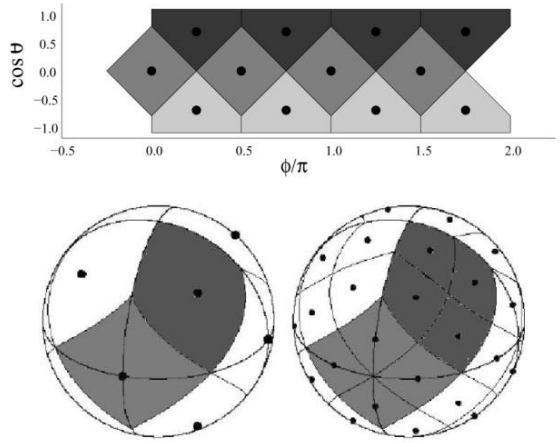


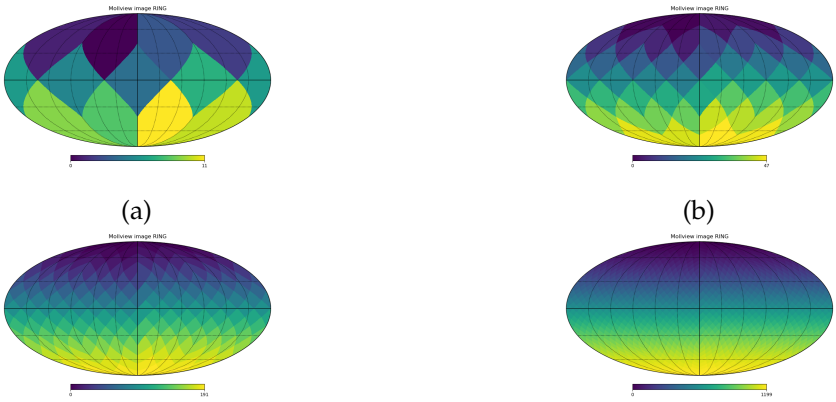
Figure 3.4: HEALPix [Gor+05] grid on the 2-sphere. Above the cylindrical projection of the sphere onto the $(\cos(\theta), \phi)$ coordinates. Below the visualization of the base grid with $m_2 = 12$ points, together with the incremental sampling procedure which subdivides each tile of the initial grid into ulterior 4 squares. From [Yer+10].

the sphere $S2$ have to relate to the number of samples with the following relation:

$$\frac{\pi}{m_1} = \sqrt{\frac{\pi}{m_2}}. \tag{3.62}$$

As the base grid of HEALPix is constructed by mapping the vertices of a icosahedron onto the sphere, the number of points correspond to 12. To satisfy equation (3.62), m_1 needs to be equal to 6. The final grid is constructed by the cartesian product of the two grids, constructing a set of m_1, m_2 samples that form the first resolution layer of the sampling procedure.

To re-introduce the randomness that characterizes sampling-based methods, a simple solution is to randomize the choice of grid elements. Within motion planning problems, this sampling method has shown to outperform classical implementations of PRM [Yer+10], and has successfully been implemented in A* on $SO(3)$ [Wat+20]. In the scope of this work, the use of the incremental property of the grid is kept to a minimum. Alternatively, the discretization is chosen such that the algorithm can rely on one sufficiently fine grid. This choice is dictated by simplicity of implementation and time constraints.



(a) (b)

(c) (d)

Figure 3.5: HEALPix [Gor+05] two-dimensional representation with different level of sampling on a Mollweide projection. (a) number of samples $m_2 = 12$, (b) $m_2 = 48$ (c) $m_2 = 192$, (d) $m_2 = 1200$

RRT*-GBO on SE(3)

In this section, the approach to obtain an optimal trajectory on the group $SE(3)$ by means of the RRT*-GBO methodology denoted in [SL14] is shown. The algorithm used comprises of three main steps:

1. Incremental building of the tree
2. Interpolation of the initial solution
3. Smoothing.

The initial solution is computed by following Algorithm 1.

For the sake of clarity, the algorithm steps are summarized in functions, which are explained in the following paragraphs.

`initialize_grid`: in this step an initial grid is constructed as in Algorithm 2. First, an origin p_o is defined in $\mathbb{R}^3$ in order to construct a grid of size (N_x, N_y, N_z) around it, with limits defined as $[(x_{min}, x_{max}), (y_{min}, y_{max}), (z_{min}, z_{max})]$. Based on this quantities, a step size is defined and the location of the grid points are stored in a container $\mathcal{G}_{\mathbb{R}^3}$. Afterwards, a similar procedure is conducted for the sampling of orientations. Once the size of the grid is defined based on the number of HEALPix and $S1$ discretization $N_{pix} = m_2, N_\psi = m_1$. The discretization of $S1$ is done through an ordinary equally spaced grid, while the $S2$ is constructed exploiting the efficient HEALPix discretization discussed above. Once the separate grids $\mathcal{G}_{S1}, \mathcal{G}_{S2}$ are constructed, one obtains the

Algorithm 1 RRT*-GBO iteration

```
initialize_grid(p_o)
connect_to_goal(x_start)
for i ← 0 to N_iter do
    x_new ← sample()
    if x_new ∉ C_obstacle then
        connect_to_goal(x_new)
        x_best, e_new ← neighborhood_query()
        if e_new is feasible then
            T ← add(x_new)
            connect_tree(x_new, x_best, e_best,new)
            rewire(x_new)
        end if
    end if
end for
```

final one through a cartesian product, obtaining $\mathcal{G}_{SO(3)}$. The grid on $SE(3)$ is finally constructed by the cartesian product $\mathcal{G}_{\mathbb{R}^3} \times \mathcal{G}_{SO(3)}$.

`sample`: this function is characteristic for sampling-based methods. As the sampling is based on incremental grids on $SE(3)$, the first step is to verify if the grid has been completely covered. In case that is true, an incremental step is performed, which means that the discretization of the grid is increased by a resolution level l. This step repeats the `initialize_grid` with updated discretization sizes. In particular, the elements of $\mathbb{R}^3$ can be re-sampled by subdividing each cube in an arbitrary even number of sub-cubes. For $SO(3)$ elements, the procedure requires increasing the size of the $S2$ grid by a factor of 4^l. The corresponding size of the $S1$ grid has to be found through Equation (3.62).

Once the grid is verified to be appropriate, an element is randomly selected from it, as denoted in Algorithm 3. The entry is then removed from the container to avoid repeatedly selecting the same elements at a given resolution level.

`neighborhood_query`: given two nodes x_i, x_j, this function computes the cost associated with the connection within the two in order to find the best node to create a new edge of the tree. Algorithm 4 shows how the function, for each node in the tree, creates an edge $e_{i,j}$ by solving the boundary value problem between the two set of states and their derivatives $(\boldsymbol{\tau}, \dot{\boldsymbol{\tau}}, \ddot{\boldsymbol{\tau}}, \dots)$, by constructing an optimization problem as in Equation (3.60). The edge is scored first based on a pruning metric, which stops the iteration if a certain threshold value $r_{threshold}$ is exceeded. This check allows to also create a neighborhood around the candidate node x_i, which allows to apply the

Algorithm 2 `initialize_grid`

$\mathcal{G}_{\mathbb{R}^3} \leftarrow p_o$
$\mathbf{N}_{\mathbb{R}^3} \leftarrow (N_x, N_y, N_z)$
$\mathbf{l}_{\mathbb{R}^3} \leftarrow [(x_{min}, x_{max}), (y_{min}, y_{max}), (z_{min}, z_{max})]$
$\mathbf{\Delta l}_{\mathbb{R}^3} \leftarrow [\Delta x, \Delta y, \Delta z]$
for $i \leftarrow 0$ to N_x **do**
  for $j \leftarrow 0$ to N_y **do**
    for $k \leftarrow 0$ to N_z **do**
      $\mathcal{G}_{\mathbb{R}^3} \leftarrow (x_o + i\Delta x, y_o + j\Delta y, z_o + k\Delta z)$
      $\mathcal{G}_{\mathbb{R}^3} \leftarrow (x_o - i\Delta x, y_o - j\Delta y, z_o - k\Delta z)$
    end for
  end for
end for
$\mathbf{N}_{SO(3)} \leftarrow [N_{pix}, N_\psi]$
$l_{S1} \leftarrow [0, 2\pi)$
$\Delta l_{S1} \leftarrow \Delta\psi$
$\mathcal{G}_{S2} \leftarrow$ `HEALPix`(N_{pix})
for $i \leftarrow 0$ to N_ψ **do**
  $\mathcal{G}_{S1} \leftarrow (i\Delta\psi)$
end for
$\mathcal{G}_{SO(3)} \leftarrow \mathcal{G}_{S1} \times \mathcal{G}_{S2}$
$\mathcal{G}_{SE(3)} \leftarrow \mathcal{G}_{\mathbb{R}^3} \times \mathcal{G}_{SO(3)}$

Algorithm 3 `sample`

if size$(\mathcal{G}_{SE(3)}) = 0$ **then**
  $\mathcal{G}_{SE(3)} \leftarrow$ `increment_grid()`
end if
$id \leftarrow$ `uniform_random_distribution`(size$(\mathcal{G}_{SE(3)})$)
$x_{new} \leftarrow \mathcal{G}_{SE(3)}[id]$
$\mathcal{G}_{SE(3)} \leftarrow$ `remove`(x_{new})

rewiring procedure later on. Afterwards the best node to connect the candidate, x_{best}, is found by simply selecting the one which would lead to the lowest cost value.

Algorithm 4 `neighborhood_query`

for x_i in $\mathcal{T}$ **do**
  $e_{i,j} \leftarrow$ `compute_edge`(x_i, x_j)
  if `prune`$(e_{i,j}) < r_{treshold}$ **then**
    neighborhood $\leftarrow x_j$
    if `cost`$(e_{i,j}) <$ `cost`$(e_{i,best})$ **then**
      $x_{best} \leftarrow x_j$
    end if
  end if
end for

`connect_tree`: this function connects two nodes x_{best}, x_{new} together by adding the new node to the existing tree. The node is added to the graph and its boundary conditions are updated by extracting them from the edge formulation $e_{best,new}$. Finally the edge is stored and added to the tree, as shown in Algorithm 5.

Algorithm 5 `connect_tree`

$\mathcal{T} \leftarrow$ `add_node`(x_{new})
$x_{new}.B \leftarrow$ `boundary_conditions`$(e_{best,new})$
$\mathcal{T} \leftarrow$ `add_edge`$(e_{best,new})$

`connect_to_goal`: this function's purpose is to connect a node of the tree x_i to the goal node x_{goal}. Following Algorithm 6, the edge $e_{i,goal}$ between the two nodes is computed. If its cost falls below the threshold r_{goal}, the procedure moves forward. If the goal has no existing connections, the new node will be connected. However, if there exists a connection from a node x_j to x_{goal}, the cost of the existing edge $e_{j,goal}$is compared with the one of the new candidate. Whenever the candidate edge's cost is lower, the existing one is removed from the tree and substituted. By proceeding this way, the solution of the edge converges to the optimal value of that section of the trajectory.

`rewire`: as in the classical RRT* formulation, this function checks checks if the latest node added to the tree x_{new} can reduce the cost of other paths that arrive into neighboring nodes x_i from $x_{incoming}$. In this case, as shown in Algorithm 7, the criteria of the rewiring is based on the cost metric. If the cost of the possible connection $e_{new,i}$ is lower than the existing $e_{incoming,i}$, then $e_{incoming,i}$ is removed and substituted by $e_{new,i}$.

This procedure delivers an initial trajectory $T_{init}(t)$, which comprises of a number

Algorithm 6 `connect_to_goal`

$e_{i,goal} \leftarrow$ `compute_edge`(x_i, x_{goal})
if `cost`$(e_{i,goal}) <$ `cost`$(e_{j,goal})$ **then**
  if x_{goal} is connected with $x_j \in \mathcal{T}$ **then**
    if `cost`$(e_{i,goal}) <$ `cost`$(e_{j,goal})$ **then**
      $\mathcal{T} \leftarrow$ `remove_edge`$(e_{j,goal})$
      $\mathcal{T} \leftarrow$ `add_edge`$(e_{i,goal})$
    end if
  else
    $\mathcal{T} \leftarrow$ `add_edge`$(e_{i,goal})$
  end if
end if

Algorithm 7 `rewire`

for all x_i in neighborhood **do**
  $e_{new,i} \leftarrow$ `compute_edge`(x_{new}, x_i)
  if `cost`$(e_{new,i}) <$ `cost`$(e_{incoming,i})$ **then**
    $\mathcal{T} \leftarrow$ `remove_edge`$(e_{incoming,i})$
    $\mathcal{T} \leftarrow$ `add_edge`$(e_{new,i})$
  end if
end for

of sub-trajectories that connect the sequence ofN_{nodes} nodes: $\{x_{start}, ..., x_{goal}\}$. Each of the sub-trajectories is kinodynamically feasible and locally optimal with respect to the boundary conditions and constraints, and spans a time $\Delta t = t_f$. This trajectory is interpolated by using a B-spline with N_{RRT} control points, this parameter can be tuned based on the problem in order to accurately express the trajectory without requiring excessive computational power and avoiding overfitting. Once the trajectory is interpolated, the resulting spline represents the motion of the system between time t_0 and time $t_0 + \Delta t(N_{nodes} - 1)$. Exploiting this result as an initial solution for a final smoothing, the quality of the end-result is increased and approximates the global optimum for any given cost metric and constraint. The sketch of the procedure is shown in Figure 3.6.

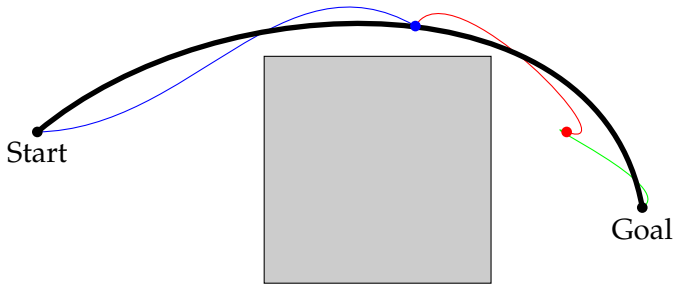


Figure 3.6: Sketch of the smoothing procedure of the RRT*-GBO. The tree delivers an initial solution that comprises of locally optimal edges (blue, red and green segments). The initial guess is fed to a smoothing procedure that delivers the optimal solution (black).

The methodology allows for path planning with arbitrary cost and constraints and delivers high-quality trajectories. One additional advantage is the time allocation aspect. The approach prevents drawing edges to nodes that cannot be reached before Δt. Rather than explicitly optimizing for time as in [Use+17], this method can add an arbitrary number of edges to reach the goal, in order to increase the duration of the trajectory and allowing for an estimate of the necessary duration of the motion. However, the RRT*-GBO presents significant drawbacks, in particular when it comes to the formulation of the edges. In the original formulation by [SL14], the sampling was conducted up to the first derivative of the state. This would result in segments with erratic behavior given by the randomness of final conditions imposed within the OCP, as shown in Figure 3.7. This effect degrades the quality of the initial solution. Additionally, it worsens the time allocation for the final trajectory by increasing the total duration to accommodate unnecessary maneuvering. The final runtime of the

algorithm is significant. The sampling of higher derivatives substantially increases the search space and, if not enough iterations are performed, the initial guess returned by the RRT*-GBO might require multiple smoothing to converge.

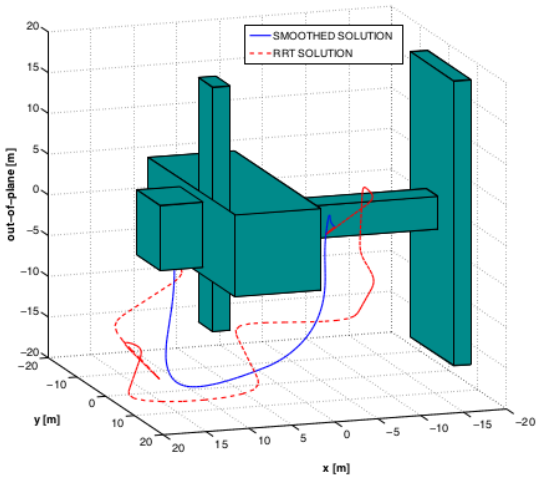


Figure 3.7: RRT*-GBO solution for a free-floating robot moving around the ISS from [SL14]. The initial solution provided to the smoother by the algorithm shows segments that are highly suboptimal.

Optimal edge construction using propagation of motion

Within this section, the problem of the random sampling within an high-dimensionally state space is investigated. In particular, an alternative edge construction is proposed, in an attempt to lower the computational effort required to find a solution of the algorithm.

In general, for a configuration space $\mathcal{C}$ of dimension n, the state space comprises of the former augmented with velocity coordinates. This augmentation leads to a two-fold increase in dimension $2n$. Since the complexity of planning algorithms usually scales exponentially with the dimension of the search space [LJ01], sampling the entire state space may be impractical, even for relatively small dimensions [SD91]. As the configuration space of the robot is $SE(3)$, it corresponds to $n = 6$ degrees of freedom, thus sampling the state-space requires searching a space of dimension $2n = 12$. In an attempt to mitigate the negative effects that the sampling approach used in [SL14] would have in $SE(3)$, within this thesis the RRT*-GBO is employed using an alternative edge construction methodology.

In [PCN13], in order to limit the search to the configuration space of the problem, rather than approaching the full state space, the authors formulate an Admissible Velocity Propagation (AVP) technique. The foundation behind the concept lies within the limited range of possible velocities that a system can assume based on its initial conditions and kinodynamic constraints. The authors first devise an algorithm to define kinodynamically feasible, time-optimal paths in the state space, by using as input a path in configuration space. The authors then construct the AVP-RRT, a kinodynamic planner that leverages the aforementioned algorithm in order to construct smooth and continuous paths. The planner limits the sampling to the configuration space and behaves like an RRT. The edges, however, are computed using the AVP formulation by propagating velocity information at the junction of any two edges. By specifying bounding-box constraints on velocity and continuity between nodes of the RRT, the methodology can produce smooth paths in the state space, by connecting two points in configuration space. The algorithm relies on considering the velocity at any origin node, which corresponds to the initial velocity for a new edge. If the value is null, the nodes are connected with a straight line, if it isn't a third order polynomial is used to connect the two. This connecting trajectory has to ensure that continuity at the junction is guaranteed and that the derivative at the final location does not violate the maximum value, which is conditioned on the velocity at the start of the segment. Once the path in configuration space is defined, the velocity is obtained through the AVP algorithm.

In [BD18], the authors explicitly formulate the construction of motion primitives for segments where a portion of the states is not fixed at the boundary conditions. In particular, they show that it is possible to define motions where no boundary state is defined, effectively freeing a number of parameters to be defined by optimization procedure. The method allows to obtain lower cost on the trajectories, as more parameters can be modified. The ability to define boundary conditions, however, is lost and the size of the optimization problem inevitably increases.

Within the context of the RRT*-GBO, these implementation choices have been the motivation behind the change. In particular, the implementation used within this thesis formulates edges such that only the configuration of the robot in $SE(3)$ is sampled. This leads to higher derivatives being the result of the optimization procedure. The quantities are later extracted from the edge and become initial conditions for new edges departing from the given nodes. This approach extends the AVP-RRT's property of propagating admissible velocities and constructing smooth paths at the junction of nodes. In particular, the edges constructed through this method are entirely smooth and differentiable up to the highest derivative embedded in the parametrization. By embedding the computation of velocity profiles within the GBO, only admissible velocities are obtained, as well as ensuring continuity of higher derivatives delivering a smooth trajectory.

A sketch of the result is presented in Figure 3.8. The resulting path can effectively be interpreted as one continuous curve as there are no jumps at the junctions. The formulation also avoids jerky motion by constructing velocity and acceleration profiles conditioned by their values at the source node, rather than originating from uninformed random samples.

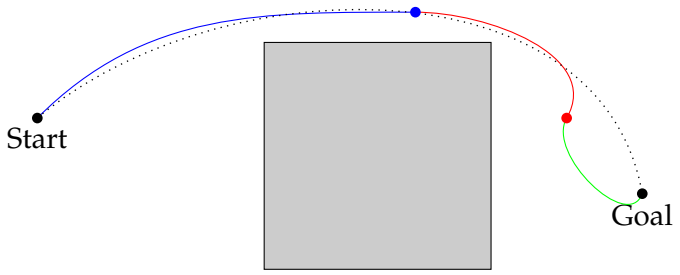


Figure 3.8: Sketch of the updated edge construction for RRT*-GBO. The tree delivers an initial solution that is continuous and differentiable at any node (blue, red and green segments). The smoothed solution is closer to the initial guess if the nodes are part of the optimal path (dotted).

The resulting algorithm may lack the probabilistic completeness attributed to sampling-based methods which sample the input space [LJ01]. In [CPN14], the authors analyze thoroughly the completeness property of various type of randomized kinodynamic planners. Considering the classical formulation of the RRT-GBO in [SL14], it is classifiable as a *state-based steering* algorithm. While [CPN14] provide proofs for probabilistic completeness of such methodologies, it is not within the scope of this work to extend it to the RRT-GBO implementation. Currently, the original formulation is not guaranteed to retain any of the properties of state-based randomized planners. Therefore, there is no evident loss in applying the edge construction defined above within the scope of this work.

3.2.4 Implementation on robotic manipulators

As a final step, the extension to the space of $SE(3)$ and the formulation of the problem allow to seamlessly apply the methodology on robotic manipulators. In particular, considering an arm with 6 degrees of freedom, the end effector moves in $3D$ space as a free-flyer would, albeit the additional limits imposed by the joints and actuation.

Assuming that the dynamic constraints applied on the free-flyer, are a subset of the limits imposed by the robotic arms dynamics and actuation, the envelope of states reachable by the end effector coincides with the one of the Astrobee robot. Defining the

set of reachable states and applicable actuation of the Astrobee as $\mathcal{X}_A$,$\mathcal{U}_A$ and the set of reachable states and applicable actuation of the end effector $\mathcal{X}_E$,$\mathcal{U}_E$, the following is assumed to be true

$$\begin{aligned} \mathcal{X}_A &\subset \mathcal{X}_E \\ \mathcal{U}_A &\subset \mathcal{U}_E. \end{aligned} \tag{3.63}$$

Thus, fulfilling the constraints placed on the free-flyer dynamics, ensures fulfillment of the constraints on the robotic manipulator as well.

Through this assumption, it is trivial to extend the problem of moving the end effector simulating the motion of an Astrobee, by just adding inverse kinematics as a constraint. In particular, in Chapter 2 it was assumed that the inverse kinematics were known, thus defining them as in Eq. (2.7)

$$\boldsymbol{\theta} = \mathbf{k}^{-1}(\mathbf{T}_E), \tag{3.64}$$

where $\boldsymbol{\theta}$ represents the joint angles of the robot and $\mathbf{T}_E$ the pose of the end-effector.

By considering a set of feasible joint angles $(\boldsymbol{\theta}_{min}, \boldsymbol{\theta}_{max})$ to account for collision with the environment and joint limits within the robotic arm itself, the inverse kinematics can be integrated in the OCP in Eq. (3.60) as a box constraint in the form:

$$\boldsymbol{\theta}_{min} \leq \mathbf{k}^{-1}(\boldsymbol{\tau}(t)) \leq \boldsymbol{\theta}_{max}. \tag{3.65}$$

4 Perception-aware path planning

This chapter's purpose is to expand the notion of kinodynamic path planning mentioned in Chapter 3. The goal within the coming sections is to leverage the notions laid out for optimal path planning, in order to formulate a cost function that combines the search for energy-efficient trajectories as well as assuring improvements of visual estimation for autonomous robots in known environments.

The problem is first studied in terms of recent approaches reported in the literature. As the purpose of this thesis is to incorporate this alternative cost formulation in the context of RRT*-GBO steering functions, it is necessary to construct a OCP to solve through the SLSQP method described in 3.2.2.

Once possible routes are identified, an analysis is performed to understand the applicability of uncertainty-aware methodologies within the context of GBO. Finally, the cost formulation selected as the most suitable is defined and compared to current state-of-the-art methodologies.

4.1 Problem Statement

4.1.1 Introduction

In recent years, the field of robotics has witnessed remarkable advancements, enabling robots to operate autonomously in complex and dynamic environments. A fundamental challenge in robotic navigation is the seamless integration of perception and planning, allowing robots to not only generate collision-free paths but also to make informed decisions based on the perception of their surroundings. This chapter delves into a critical facet of modern robotic navigation -perception-aware path planning – and explores its synergistic integration with Visual-Inertial Odometry (VIO) technology.

Traditional path planning algorithms, which have been discussed in chapter 3, have largely focused on kinematic and dynamic constraints and geometric considerations. However, as robots are increasingly deployed in real-world scenarios, the reliance on solely geometric information may be inadequate.

To navigate with efficiency and safety, robots must possess the ability to perceive and understand their environment in real-time. VIO has emerged as a powerful solution, fusing data from cameras and inertial sensors to estimate the robot's pose and motion

accurately. An extension of the methodology, Visual-Inertial Simulataneous Localization And Mapping (SLAM) (VI-SLAM) allows to both estimate the pose of the robot, while constructing an incrementally updated map of the surrounding environment. By leveraging this sensor fusion, robots can gain a deeper understanding of their surroundings, leading to more informed path planning decisions.

This paradigm shift from traditional path planning not only enhances the robot's ability to avoid collisions but also opens avenues for optimizing trajectories considering perceptual objectives, such as visibility of landmarks or minimizing uncertainty while traversing a path.

4.1.2 Related Work

Literature shows that the link between active perception and SLAM is strong, as the connection between the robot's pose and its environment [BR11; Mu+16]. The state estimation performance is highly influenced by the system's trajectory. The field of active perception and active vision [AWB88], has grown significantly in recent years in an attempt to fill the gap in between state estimation, path planning and control.

One of the first efforts in the field was presented in [DM02], the authors control the motion of two movable cameras on a ground robot, in an attempt to reduce the uncertainty during localization. The method exploits the incrementally built map to increase the localization accuracy by leveraging the cross coupling between the robot's pose and the mapped features. With a similar intent [Ata15; ACA14], integrate path-planning and control with the state estimation. Differently from [DM02], these works do not actively control the motion of the sensors alone, but rather generate paths that enhance the entire motion estimation.

Formally, the active perception path-planning problem can be formulated as a Partially Observable Markov Decision Process (POMDP) [KLC98], which delivers a framework for planning under uncertainties. Unfortunately, the complexity of such an approach motivates the search for alternative solutions.

The field of Belief-space planning has originated from the formulation of sampling-based planners that would incorporate uncertainty within the plan. The approach demonstrated to be very effective, especially tackling obstacle avoidance with uncertain position information. Methods such as belief trees [BR11] and belief roadmaps [PR09] have been successfully applied within two-dimensional exploration and mapping problems [Ach+14; Cos+16].

The most recent works address the problem of online planning through a receding-horizon approach, to combine exploration of unknown environments with belief-space planning. In [Ach+14] the authors developed a planner using random belief trees. The cost metric used is the uncertainty associated with the transition from one node

to the following. To quantify the uncertainty, the covariance matrix associated with the EKF-based estimation is propagated along an edge of the graph. Using the same approach, [Cos+16] developed a planner that can handle dense mapping scenarios using photometric information. The covariance at each node is obtained by predicting the reprojection error's uncertainty at any given point, relying on the map available at the planning stage. Both these methodologies assume straight edges as steering functions within the tree. In [ZS18], an ulterior step is proposed. The authors use a sampling-based methodology with polynomial steering functions. In particular, they construct polynomial motion primitives to reach a desired distance from the starting position. A number of sampled poses on these segments are then scored based on the quality of the expected VIO, which is obtained using an active map for that specific segment. The cost metric is obtained from the covariance matrix of the localization estimation, expressed in terms of the associated the least-squares problem. In [Wu+22], the authors improve the framework defined in [ZS18]. They first extend the formulation by formalizing the steering functions as minimum-jerk fifth-order polynomials. Each trajectory section goes through checks to verify input feasibility and velocity limits. An ulterior improvement is obtained by including a penalization on trajectories that lead to motion blur on the visual feed. The measurement covariances are computed including additional uncertainty terms. The terms are functions of the robot's velocity since they are computed using the associated pixel velocities of the mapped features.

Aside from sampling-based methodology, the path of trajectory optimization with interpolating curves has been approached as an alternative. In [Mur+19], the authors aim at defining a GBO planner for the flat outputs of a Micro Aerial Vehicle (MAV), which has to move through a set of waypoints assumed to be known. First, the planner constructs a polynomial trajectory through the set of positions. After that, the yaw angle trajectory is obtained by performing an optimization that aims at maintaining the highest number of visual landmarks in view during the motion. The authors define a relaxed differentiable function to model the visibility of a feature, in order to construct a continuous and smooth cost function to feed to the numerical optimization framework.

In [BTC20], the authors extend the approach of [Mur+19] by introducing a kinodynamic A* planner to obtain an initial guess of the trajectory. In particular, they aim at not only maintaining a large number of landmarks in view, but also selecting only areas where the features extracted are reliable. Areas which can be highly textured might be considered unreliable if the texture is transient. This is the case for water or grass moved by the wind, which does not look consistent over time although can be considered a texture-filled area. The methodology presented by [BTC20] allows to account for the reliability of an area through semantic segmentation, the information is then scored and accounted in the generation of the initial trajectory through the search-based planner. At a later time, the trajectory is optimized by using a B-spline

formulation with elastic-band cost functions. The cost accounts for smoothness and collision avoidance, as well as visibility of landmarks using the same relaxed function from [Mur+19]. Additionally, the authors introduce a co-visibility term, which penalizes large changes of direction of the camera over adjacent poses.

In [Zho+21], a methodology for replanning in the context of fast flight through unknown environments is devised. The planner uses an initial topological path searching approach, in order to obtain an initial trajectory from a number of locally optimal trajectories generated in this step. The best trajectory found through this process is refined by a risk-aware planner, which aims at increasing visibility towards unknown areas while maintaining safety guarantees with respect to unknown obstacles. Once the risk-aware trajectory is obtained, the yaw angle of the MAV is further optimized to actively explore unknown areas.

4.1.3 Problem Description

The scope of this chapter is to investigate a perception-aware path planning algorithm, that allows to plan trajectories that maximize the number of visible landmarks along them. The devised planner should seamlessly fit within the kinodynamic path planning problem described in chapter 3, therefore should allow to incorporate a number of necessary constraints. The final goal of the methodology is to define and implement an algorithm that allows to be deployed in real-life scenarios on the ISS, allowing to improve the quality of the localization algorithm deployed on the robot.

The ISS module is a known environment and as such all the visual information within it is assumed to be known. This environment is provided through a sparse map of landmarks, which is assumed to be correct and non-changing during the path planning solution.

As detailed in section 4.1.2, many of the available methodologies are constructed to be deployed on MAVs and ground vehicles. Most of these system's dynamics can be fully parametrized by considering less than 6 degrees of freedom, thus none of the methods described for these platforms is applicable to the case of a free-flyer on $SE(3)$.

State-of-the-art approaches frequently adopt a two-step process. First, a kinodynamic path is generated using search or sampling-based methods to find a feasible trajectory through the environment. Subsequently, an additional optimization step is employed, incorporating perception-aware cost functions to refine the path. However, this two-step approach may not fully exploit the potential of perception data, potentially leading to suboptimal trajectories or limited adaptability to different scenarios. Many of the currently available methods, do not provide a thorough analysis of the cost functions in use, rendering the evaluation of the solution difficult from an optimality perspective.

Objectives

The primary objective of this chapter is to construct a perception-aware GBO path planner that can efficiently exploit data from the environment. The planner should be able to cover the full parametrization of $SE(3)$ and account for flexibility in terms of cost functions and constraints. Additionally the cost function should allow to be easily interpreted within the context of numerical optimization in order to demonstrate optimality of the solution.

Contributions

In the next sections, the following challenges will be investigated:

- Analysis of uncertainty-based cost functions within GBO.
- Construction of a feature-density function in $SE(3)$.
- Application of the feature-density function within GBO.
- Extension of available GBO approaches to $SE(3)$.
- Comparison of the feature-density approach with state-of-the-art methods.

4.2 Methodology

4.2.1 Overview

In order to construct a planner that minimizes the uncertainty of the localization during operations, it is necessary to investigate the relationship between estimation approaches and the system's motion. In the following sections, the problem of localization is investigated analyzing the inner workings of available estimation methods. The analysis is performed in order to identify the applicability of uncertainty-based cost metrics in the context of this work.

Afterwards, a state-of-the-art cost function formulation from literature is extended in order to fit within the path planning problem at hand. Finally, after analyzing the shortcomings of state-of-the-art methodologies, an alternative approach is formulated and proposed.

4.2.2 Visual-Inertial Localization (VIL) for autonomous free-floating robots

A paradigm that has gained success for autonomous robots' navigation is SLAM. The methodology relies on the sensors on-board to construct and update a map of the

surrounding environment, while simultaneously estimating the system's pose. The correlation between the poses of the robot and the observed landmarks is what allows to improve the quality of the estimation and mapping over time [Thr02]. The methodology allows to reconstruct the entirety of the surrounding environment, but that also comes at a cost of increased computation.

It is common for state estimation to rely on the onboard sensors to only estimate the robot position and orientations. The most popular paradigm is VIO [SZ20; Leu+15]. The methodology combines the sensor information from cameras and IMU sensors, in order to estimate the motion of the robot over time. This approach is well suited to be used online as it is characterized by low latency. VIO leverages the high output rate of the IMU sensor (its rate can reach up to $1,000Hz$). However, since it is affected by poor signal-to-noise ratio at low accelerations and low angular velocities, the fusion with low frequency camera images allows to correct accumulated drift [SZ20].

In the context of the Astrobee robots, the system can rely on a very detailed map of the environment which is obtained through the procedure described in [Col+16a]. The localization procedure can therefore exploit the information by relying on the existing map to improve the quality of the pose estimation. The case under consideration is defined as VIL, as the robot does not need to estimate landmark positions, but rather match features extracted from an image to existing ones.

The methodology currently used on-board the Astrobee on the ISS is described in [Sou+22]. The system uses a factor-graph approach implemented in GTSAM [DC22; DK17b], which combines preintegrated IMU measurements [For+15] with visual measurements implemented as smart factors [Car+14].

Considering the motion of a system equipped with an IMU and a monocular camera, let $\mathcal{K}_k$ denote the set of all keyframes up to time k. The objective is to estimate the set of states

$$\mathcal{X}_k = \{\mathbf{x}_i\}_{i \in \mathcal{K}_k} \tag{4.1}$$

where the state vector $\mathbf{x}$ comprises of position $\mathbf{p}$, orientation $\mathbf{R}$, velocities $\mathbf{v}, \boldsymbol{\omega}$ and accelerometer and gyroscope biases $\mathbf{b}^a, \mathbf{b}^g$.

As the landmarks are not objective of the estimation, their structure is therefore not part of the optimization variables. In order to solve the problem, the measurements have to be included as inputs. Let $\mathcal{C}_i$ denote the camera measurements at keyframe i. At time i any number of landmarks l can be in view, thus the set contains a number of pixel measurements $\mathbf{z}_{i,l}$. Here $\mathcal{L}_i$ is used to denote the set of visible landmarks at time i. In the same fashion, the set $\mathcal{I}_{i,j}$ represents the set of IMU measurements acquired between two consecutive keyframes i and j. Following this notation, the set of measurements collected up to time k is represented as

$$\mathcal{Z}_k = \left\{\mathcal{C}_i, \mathcal{I}_{i,j}\right\}_{(i,j) \in \mathcal{K}_k}. \tag{4.2}$$

The estimation problem can now be formulated as a factor graph [For+15]

$$\begin{aligned} p(\mathcal{X}_k|\mathcal{Z}_k) &\propto p(\mathcal{X}_0)p(\mathcal{Z}_k|\mathcal{X}_k) = p(\mathcal{X}_0) \prod_{(i,j)\in\mathcal{K}_k} p(\mathcal{C}_i, \mathcal{I}_{i,j}|\mathcal{X}_k) \\ &= p(\mathcal{X}_0) \prod_{(i,j)\in\mathcal{K}_k} p(\mathcal{I}_{i,j}|\mathbf{x}_i, \mathbf{x}_j) \prod_{i\in\mathcal{K}_k} \prod_{l\in\mathcal{L}_i} p(\mathbf{z}_{i,l}|\mathbf{x}_i). \end{aligned} \tag{4.3}$$

The solution is the Maximum A Posteriori (MAP) estimate, corresponding to

$$\mathcal{X}_k^\star = \arg\max_{\mathcal{X}_k} \; p(\mathcal{X}_k|\mathcal{Z}_k), \tag{4.4}$$

or alternatively to the minimum of the negative log-posterior

$$\mathcal{X}_k^\star = \arg\min_{\mathcal{X}_k} -\ln \; p(\mathcal{X}_k|\mathcal{Z}_k). \tag{4.5}$$

Assuming zero-mean Gaussian noise on the measurements, equation (4.5) results in the following least-squares problem

$$\mathcal{X}_k^\star = \arg\min_{\mathcal{X}_k} \|\mathbf{r}_0\|^2_{\mathbf{\Sigma}_0} + \sum_{(i,j)\in\mathcal{K}_k} \|\mathbf{r}_{\mathcal{I}_{i,j}}\|^2_{\mathbf{\Sigma}_{i,j}} + \sum_{i\in\mathcal{K}_k} \sum_{l\in\mathcal{L}_i} \|\mathbf{r}_{\mathcal{C}_{i,l}}\|^2_{\mathbf{\Sigma}_C} \tag{4.6}$$

where $\mathbf{r}_0$, $\mathbf{r}_{\mathcal{I}_{i,j}}$ and $\mathbf{r}_{\mathcal{C}_{i,l}}$ correspond to the residuals associated to each measurements. The quantities $\mathbf{\Sigma}_0$, $\mathbf{\Sigma}_{i,j}$ and $\mathbf{\Sigma}_C$ are the respective measurement covariance matrices.

The residuals for the IMU preintegrated measurements are $\mathbf{r}_{\mathcal{I}_{i,j}} = \left[\mathbf{r}_{\Delta R_{i,j}}, \mathbf{r}_{\Delta v_{i,j}}, \mathbf{r}_{\Delta p_{i,j}}\right]^T$ as defined in [For+15]. More relevant for the current analysis are the residuals of visual measurements

$$\mathbf{r}_{\mathcal{C}_{i,l}} = \mathbf{z}_{i,l} - \pi\left(\mathbf{R}_i, \mathbf{p}_i, \boldsymbol{\rho}_l\right) \tag{4.7}$$

where $\rho_l \in \mathbb{R}^3$ denotes the position of the landmark l in the tridimensional space and $\pi(.)$ is the perspective projection which also encodes the transformation from camera to IMU frame.

In the context of perception-aware path planning, the main concern is to maximize the effectiveness of the visual component of the estimation. The analysis will thus focus on the term $\mathbf{r}_{\mathcal{C}_{i,l}}$. In particular, the aim is to exploit equation (4.6) to obtain the estimation error of the least-squares problem in terms of the associated information matrix. The focus thus shifts on the simplified problem

$$\mathcal{X}_k^\star = \arg\min_{\mathcal{X}_k} \sum_{i\in\mathcal{K}_k} \sum_{l\in\mathcal{L}_i} \|\mathbf{z}_{i,l} - \pi\left(\mathbf{T}_i, \boldsymbol{\rho}_l\right)\|^2_{\mathbf{\Sigma}_C} \tag{4.8}$$

where the pose of the body is now defined by $\mathbf{T}_i \in SE(3)$. The solution is approached through numerical optimization using the Gauss-Newton solver. The results are

achieved by operating on the manifold $SE(3)$, applying a similar methodology to the one showed in Chapter 3. The least-squares problem is reformulated through an infinitesimal increment on the Lie algebra $\mathfrak{se}(3)$, $\boldsymbol{\xi}$

$$\boldsymbol{\xi}^{\star} = \arg\min_{\boldsymbol{\xi}} \sum_{i \in \mathcal{K}_k} \sum_{l \in \mathcal{L}_i} \| \mathbf{z}_{i,l} - \pi \left(\mathbf{T}_i \exp(\boldsymbol{\xi}), \boldsymbol{\rho}_l \right) \|^2_{\boldsymbol{\Sigma}_C}. \tag{4.9}$$

The solution is obtained incrementally by updating $\mathbf{T}_i \leftarrow \mathbf{T}_i \exp(\boldsymbol{\xi})$. In practice, the problem is linearized, obtaining the following expression

$$\boldsymbol{\xi}^{\star} = \arg\min_{\boldsymbol{\xi}} \sum_{i \in \mathcal{K}_k} \sum_{l \in \mathcal{L}_i} \| \mathbf{z}_{i,l} - \pi \left(\mathbf{T}_i, \boldsymbol{\rho}_l \right) - \mathbf{J}_{\boldsymbol{\xi},i,l} \boldsymbol{\xi} \|^2_{\boldsymbol{\Sigma}_C} \tag{4.10}$$

in which the projection term has been linearized through its Jacobian $\mathbf{J}_{\boldsymbol{\xi},i,l} = \frac{\partial \pi(\mathbf{T}_i, \boldsymbol{\rho}_l)}{\partial \boldsymbol{\xi}}$. The explicit expression of the Jacobian can be obtained by applying the chain rule, using the reformulation $\boldsymbol{\rho}'_{i,l} = \mathbf{T}_i \boldsymbol{\rho}_l = [X'_{i,l}, Y'_{i,l}, Z'_{i,l}]$,

$$\mathbf{J}_{\boldsymbol{\xi},i,l} = \frac{\partial \boldsymbol{\pi}_{i,l}}{\partial \boldsymbol{\rho}'_l} \frac{\partial \boldsymbol{\rho}'_l}{\partial \boldsymbol{\xi}}, \tag{4.11}$$

where the term $\pi \left(\mathbf{T}_i, \boldsymbol{\rho}_l \right)$ has been reformulated as $\boldsymbol{\pi}_{i,l}$ for clarity. Explicitly, this quantity represents the perspective projection of the landmark $\boldsymbol{\rho}_l = [X_l, Y_l, Z_l]$ on the image plane:

$$\boldsymbol{\pi}_{i,l} = \begin{bmatrix} f_x \frac{X'_{i,l}}{Z'_{i,l}} + c_x \\ f_y \frac{Y'_{i,l}}{Z'_{i,l}} + c_y. \end{bmatrix}. \tag{4.12}$$

The derivative is obtained through derivation of the Lie algebra

$$\frac{\partial \boldsymbol{\rho}'_l}{\partial \boldsymbol{\xi}} = \left[\mathbf{I} - (\boldsymbol{\rho}'_l)^{\wedge} \right]. \tag{4.13}$$

Finally, combining equations (4.11), (4.12) and (4.13), the Jacobian can be explicitly expressed as

$$\mathbf{J}_{\boldsymbol{\xi},i,l} = \begin{bmatrix} \frac{f_x}{Z'_{i,l}} & 0 & -\frac{f_x X'_{i,l}}{{Z'_{i,l}}^2} & -\frac{f_x X'_{i,l} Y'_{i,l}}{{Z'_{i,l}}^2} & f_x + \frac{f_x {X'_{i,l}}^2}{{Z'_{i,l}}^2} & -\frac{f_x Y'_{i,l}}{Z'_{i,l}} \\ 0 & \frac{f_y}{Z'_{i,l}} & -\frac{f_y Y'_{i,l}}{{Z'_{i,l}}^2} & -f_y - \frac{f_y {Y'_{i,l}}^2}{{Z'_{i,l}}^2} & \frac{f_y X'_{i,l} Y'_{i,l}}{{Z'_{i,l}}^2} & \frac{f_y X'_{i,l}}{Z'_{i,l}} \end{bmatrix}. \tag{4.14}$$

Stacking the Jacobians $\mathbf{J}_{\boldsymbol{\xi},i,l}$ delivers $\mathbf{J}_i$, which allows to obtain the covariance of the estimated parameter [HZ03]

$$\boldsymbol{\Sigma}_i = (\mathbf{J}_i^T \boldsymbol{\Sigma}_{\mathcal{C},i}^{-1} \mathbf{J}_i)^{-1} = (\boldsymbol{\Lambda}_i)^{-1} \tag{4.15}$$

where the variable $\mathbf{\Lambda}_i$ represents the *information matrix* of the estimated pose $\mathbf{T}_i$.

As the map $\mathcal{M}$ contains only pre-estimated landmarks, which are assumed to be of high quality and fixed a priori, the poses $\{\mathbf{T}_i\}_{i \in \mathcal{K}_k}$ are conditionally independent [ZS18]. Therefore, the information matrix of the entire estimation can be written as a diagonal matrix

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_0 & 0 & \cdots & 0 \\ 0 & \mathbf{\Lambda}_1 & \cdots & 0 \\ 0 & \dots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \mathbf{\Lambda}_K. \end{bmatrix} \tag{4.16}$$

The same can be said for the covariance matrix $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$.

A representative example can be formulated by assuming a pose $\mathbf{T}_i$, from which two landmarks $l_1, l_2 \in \mathcal{L}_i$ are visible. The positions of the landmarks with respect to the camera are expressed as $\boldsymbol{\rho}'_{i,1} = [X'_{i,1}, Y'_{i,1}, Z'_{i,1}]$ and $\boldsymbol{\rho}'_{i,2} = [X'_{i,2}, Y'_{i,2}, Z'_{i,2}]$. The corresponding stacked Jacobian matrix is $\mathbf{J}_i = \begin{bmatrix} \mathbf{J}_{\xi,i,1} \\ \mathbf{J}_{\xi,i,2} \end{bmatrix}$, resulting in

$$\mathbf{J}_i = \begin{bmatrix} \frac{f_x}{Z'_{i,1}} & 0 & -\frac{f_x X'_{i,1}}{{Z'_{i,1}}^2} & -\frac{f_x X'_{i,1} Y'_{i,1}}{{Z'_{i,1}}^2} & f_x + \frac{f_x {X'_{i,1}}^2}{{Z'_{i,1}}^2} & -\frac{f_x Y'_{i,1}}{Z'_{i,1}} \\ 0 & \frac{f_y}{Z'_{i,1}} & -\frac{f_y Y'_{i,1}}{{Z'_{i,1}}^2} & -f_y - \frac{f_y {Y'_{i,1}}^2}{{Z'_{i,1}}^2} & \frac{f_y X'_{i,1} Y'_{i,1}}{{Z'_{i,1}}^2} & \frac{f_y X'_{i,1}}{Z'_{i,1}} \\ \frac{f_x}{Z'_{i,2}} & 0 & -\frac{f_x X'_{i,2}}{{Z'_{i,2}}^2} & -\frac{f_x X'_{i,2} Y'_{i,2}}{{Z'_{i,2}}^2} & f_x + \frac{f_x {X'_{i,2}}^2}{{Z'_{i,2}}^2} & -\frac{f_x Y'_{i,2}}{Z'_{i,2}} \\ 0 & \frac{f_y}{Z'_{i,2}} & -\frac{f_y Y'_{i,2}}{{Z'_{i,2}}^2} & -f_y - \frac{f_y {Y'_{i,2}}^2}{{Z'_{i,2}}^2} & \frac{f_y X'_{i,2} Y'_{i,2}}{{Z'_{i,2}}^2} & \frac{f_y X'_{i,2}}{Z'_{i,2}} \end{bmatrix}. \tag{4.17}$$

Considering a measurement covariance defined as $\mathbf{\Sigma}_{\mathcal{C}} = \text{diag}(\sigma_{\mathcal{C}}, \sigma_{\mathcal{C}})$, the resulting covariance matrix is $\mathbf{\Sigma}_{\mathcal{C},i} = \text{diag}(\sigma_{\mathcal{C}}, \sigma_{\mathcal{C}}, \sigma_{\mathcal{C}}, \sigma_{\mathcal{C}})$. Finally, the information matrix can be computed through inverting equation (4.15):

$$\mathbf{\Lambda}_i = \mathbf{J}_i^T \mathbf{\Sigma}_{\mathcal{C},i}^{-1} \mathbf{J}_i. \tag{4.18}$$

Due to the large size of the resulting matrix $\mathbf{\Lambda}_i$, it cannot be reported completely. To provide an insight of the mathematical formulation of the matrix, the trace of the information matrix is reported. The choice falls on the trace based on the notion delivered by [Cos+16]. In the paper, the authors focus on the trace of the covariance matrix, as it allows to consider the majority of the state space without excessive computation. For demonstration purposes, the trace of the information matrix in this

example reads

$$\begin{aligned}\mathrm{tr}(\mathbf{\Lambda}_i) =& \frac{f_x^2 {X'_{i,1}}^2 {Y'_{i,1}}^2}{\sigma_{\mathcal{C}} {Z'_{i,1}}^4} + \frac{f_x^2 {X'_{i,1}}^2}{\sigma_{\mathcal{C}} {Z'_{i,1}}^4} + \frac{f_x^2 {X'_{i,2}}^2 {Y'_{i,2}}^2}{\sigma_{\mathcal{C}} {Z'_{i,2}}^4} + \\ &+ \frac{f_x^2 {X'_{i,2}}^2}{\sigma_{\mathcal{C}} {Z'_{i,2}}^4} + \frac{f_x^2}{\sigma_{\mathcal{C}} {Z'_{i,2}}^2} + \frac{f_x^2}{\sigma_{\mathcal{C}} {Z'_{i,1}}^2} + \frac{f_y^2 {X'_{i,1}}^2 {Y'_{i,1}}^2}{\sigma_{\mathcal{C}} {Z'_{i,1}}^4} + \\ &+ \frac{f_y^2 {X'_{i,2}}^2 {Y'_{i,2}}^2}{\sigma_{\mathcal{C}} {Z'_{i,2}}^4} + \frac{f_y^2 {Y'_{i,1}}^2}{\sigma_{\mathcal{C}} {Z'_{i,1}}^4} + \frac{f_y^2 {Y'_{i,2}}^2}{\sigma_{\mathcal{C}} {Z'_{i,2}}^4} + \\ &+ \frac{f_y^2}{\sigma_{\mathcal{C}} {Z'_{i,2}}^2} + \frac{f_y^2}{\sigma_{\mathcal{C}} {Z'_{i,1}}^2} + \frac{\left(\frac{f_x {X'_{i,1}}^2}{{Z'_{i,1}}^2} + f_x\right)^2}{\sigma_{\mathcal{C}}} + \\ &+ \frac{\left(\frac{f_x {X'_{i,2}}^2}{{Z'_{i,2}}^2} + f_x\right)^2}{\sigma_{\mathcal{C}}} + \frac{\left(-\frac{f_y {Y'_{i,1}}^2}{{Z'_{i,1}}^2} - f_y\right)^2}{\sigma_{\mathcal{C}}} + \frac{\left(-\frac{f_y {Y'_{i,2}}^2}{{Z'_{i,2}}^2} - f_y\right)^2}{\sigma_{\mathcal{C}}}\end{aligned} \tag{4.19}$$

Equation (4.19) shows how each landmark's position appears in the expression as non-negative terms of a sum. The example shows the correlation between increasing the number of landmarks in view and increasing the information content at a given pose. The expression also allows to note that the terms within the information matrix are continuous only as long as a landmark is visible. Since the visibility of a given point is a discrete function, if one of the two landmarks fell outside of the field of view, all terms linked to it would be removed from the sum. The discontinuity of the visibility function, renders the use of metrics related to the information matrix (thus the covariance matrix as well) inappropriate within the context of GBO.

4.2.3 Continuous visibility functions

In the previous section, the correlation between camera measurements and localization uncertainty has been highlighted. This theoretical result consolidates the observations and intuitions made in most of the works from section 4.1.2, that maintaining a large number of landmarks within the field-of-view ensures reduction of localization uncertainty. Since it has been deduced that exploiting the covariance matrix of the estimation does not constitute a suitable cost metric for GBO approaches, alternative cost functions have to be investigated.

State-of-the-art approaches

In [Mur+19], the authors define a relaxed differentiable visibility function, which allows to model the discrete visibility of a given point into a continuous value.

The approach relies on the definition of five vectors defining the camera frustum. A point in space lies within the field-of-view if and only if it belongs to the viewing frustum. The representation of the frustum is the intersection of five half-spaces. Four of these represent the regions within the edges of the camera image and the optical center, while the last corresponds to the region in front of the sensor. Each of the half-spaces is expressed through a vector in $\mathbb{R}^3$. These vectors connect the optical center of the sensor to its vertices, their definition represented in the camera frame is:

$$v_{tr} = \begin{bmatrix} (w - c_x)\pi_x \\ (-c_y)\pi_y \\ f \end{bmatrix}, \tag{4.20}$$

$$v_{tl} = \begin{bmatrix} (-c_x)\pi_x \\ (-c_y)\pi_y \\ f \end{bmatrix}, \tag{4.21}$$

$$v_{lr} = \begin{bmatrix} (w - c_x)\pi_x \\ (h - c_y)\pi_y \\ f \end{bmatrix}, \tag{4.22}$$

$$v_{ll} = \begin{bmatrix} (-c_x)\pi_x \\ (h - c_y)\pi_y \\ f \end{bmatrix}, \tag{4.23}$$

$$v_z = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix}, \tag{4.24}$$

where π_x, π_y are the pixel width and height of the camera image in meters, w, h are the same width and height in pixels and f is the focal length, which is assumed to be the same in x and y direction. The cross-products of four of these five vectors define the camera frustum $v_{tr} \times v_{lr}, v_{tl} \times v_{tr}, v_{ll} \times v_{tl}$ and $v_{lr} \times v_{ll}$. In [Mur+19], the authors leverage the frustum to construct the following vector

$$\mathbf{o}(\mathbf{T}_i, \boldsymbol{\rho}_l) = \begin{bmatrix} \frac{1}{2}(1 + \tanh(v_{tr} \times v_{lr} \cdot \boldsymbol{\rho}'_{i,l})) \\ \frac{1}{2}(1 + \tanh(v_{tl} \times v_{tr} \cdot \boldsymbol{\rho}'_{i,l})) \\ \frac{1}{2}(1 + \tanh(v_{ll} \times v_{tl} \cdot \boldsymbol{\rho}'_{i,l})) \\ \frac{1}{2}(1 + \tanh(v_{lr} \times v_{ll} \cdot \boldsymbol{\rho}'_{i,l})) \\ \frac{1}{2}(1 + \tanh(v_z \cdot \boldsymbol{\rho}'_{i,l})) \end{bmatrix}. \tag{4.25}$$

The visibility function $v_s(\mathbf{T}_i, l_l)$ of landmark l_l at pose $\mathbf{T}_i$ is computed by multiplication of the terms o_j of $\mathbf{o}(\mathbf{T}_i, \boldsymbol{\rho}_l)$

$$v_s(\mathbf{T}_i, \boldsymbol{\rho}_l) = \prod_{j=0}^{4} o_j. \tag{4.26}$$

The function v_s spans the continuous range of values between 0 and 1, where 0 corresponds to complete non-visibility and 1 corresponds to the highest possible visibility score. The cost function associated to the landmarks visibility is defined as the sum over the scores of each landmarks $l \in \mathcal{L}$ at all poses $\{\mathbf{T}_i\}_{i\in\mathcal{K}}$ [BTC20; Mur+19]

$$\mathcal{F}_l = -\sum_{i\in\mathcal{K}} \sum_{l\in\mathcal{L}} v_s(\mathbf{T}_i, \boldsymbol{\rho}_l). \tag{4.27}$$

As an additional cost component, Bartolomei et al. [BTC20] introduce a term to penalize motion that steers the camera away from the previous direction, in an attempt to maintain the same set of features in view between successive keyframes. The term is denominated co-visibility, and is defined as the difference between the yaw angle at a keyframe θ_i and the direction that connects two consecutive keyframes $\theta_i^n(\mathbf{T}_i, \mathbf{T}_{i+1})$

$$F_v(\mathbf{T}_i, \mathbf{T}_{i+1}) = \begin{cases} (|\theta_i - \theta_i^n(\mathbf{T}_i, \mathbf{T}_{i+1})| - \Delta\theta^\star)^2, & \text{if } |\theta_i - \theta_i^n(\mathbf{T}_i, \mathbf{T}_{i+1})| \geq \Delta\theta^\star \\ 0, & \text{otherwise} \end{cases} \tag{4.28}$$

where $\Delta\theta^\star$ represents a limit on the maximum variation of orientation. The final cost formulation can be written

$$\mathcal{F}_v = \sum_{i\in\mathcal{K}\setminus\{K\}} F_v(\mathbf{T}_i, \mathbf{T}_{i+1}). \tag{4.29}$$

The original authors leverage their representation of the state-space by describing the orientation as a yaw angle rather than three parameters. Moreover, they constrain the system's camera direction using the direction of movement. In the case of planning on $SE(3)$, this approach does not hold anymore, as the velocity is arbitrary and the orientations cannot be simply represented by a single parameter.

Differentiable visibility function in $SE(3)$

In order to apply these methodology to the problem analyzed in this thesis, it is necessary to formulate an extension onto the special euclidean group. As mentioned above, the cost definition used in [BTC20], is not suitable for a free-flying robot such as Astrobee, nor a robot manipulator. Thus an extension of the co-visibility cost is hereby

presented, based on the assumptions that two consecutive sampled keyframes are close enough that the view does not change substantially. Under this assumption, the same landmarks are in view in two adjacent keyframes, if the variation in orientation is small. To construct a distance metric on the space of rotations $SO(3)$, the angle-axis formulation used in Section 3.2.2 can be exploited. As denoted in [Har+13], the angular distance can be defined as the the angle θ which connects two orientations $\mathbf{R}_1, \mathbf{R}_2$. The angle can always be chosen as $0 \leq \theta \leq \pi$, if necessary by reversing the axis of rotation. Under this conditions, it is possible to compute the angle through the logarithmic map of $SO(3)$. Defining the distance function as $d\angle$, the value of the angle between consecutive rotations is defined as the $2-$norm of the angle-axis vector representation of the relative rotation

$$d\angle(\mathbf{R}_1, \mathbf{R}_2) = \| \log(\mathbf{R}_1 \mathbf{R}_2^T) \|_2. \tag{4.30}$$

This distance metric expression allows to extend the co-visibility cost to $SE(3)$ by writing

$$\mathcal{F}_v = \sum_{i \in \mathcal{K} \setminus \{K\}} d\angle(\mathbf{T}_i, \mathbf{T}_{i+1}). \tag{4.31}$$

Finally, by formulating the vision-based cost function of the trajectory planning through GBO on $SE(3)$ as in [BTC20]:

$$\mathbf{\Gamma}_{\mathcal{C}}(\mathbf{T}(t)) = \lambda_l \mathcal{F}_l + \lambda_v \mathcal{F}_v \tag{4.32}$$

where λ_l, λ_v are weights used to influence the relevance of the different cost components within the optimization.

4.2.4 Feature-density function

The method described up to this point, does not allow for the flexibility desired in this thesis. The differentiable function's convexity was not thoroughly investigated by the authors, thus not providing any guarantee of optimality nor any information on how to define the weights of each cost term. Additionally, the function requires to compute a scoring function for each landmark in the map, which could lead to very high overhead in terms of computations required. The authors sample only a limited number of via points in their implementation, as increasing them would lead to a definite increase in computation. Finally, the experimental results have shown that although the offline map shows the presence of a landmark at a certain position, the landmark might not be recognized during the estimation process. That calls for the definition of a cost metric that can not only rely on texture information rather than landmarks positions, but also that allows the flexibility to give more relevance to selected areas of interest and less to others.

In order to accurately analyze and then verify the validity of the perception-aware planning results, the objective of this section is to define a *Feature-density* function. The aim of this approach is to construct a function that takes a map $\mathcal{M}$ as input and returns a metric that allows to estimate the density of texture within the environment. Moreover, this function should be continuous, differentiable and possibly convex (which can be achieved through proper tuning) in order to work well in the context of GBO.

The function is obtained through a generalization of the B-splines used in the context of trajectory planning. In particular, the goal is to construct a polynomial function to interpolate the visibility of a landmark at pose $\mathbf{T}_i$.

In order to interpolate higher dimensional arrays, the formulation of tensor-product B-splines [Sch07] is exploited. A tensor-product B-spline is the combination of multiple one dimensional B-spline curves, which allows to describe d-dimensional functions $f : \mathbb{R}^d \to \mathbb{R}$ through the following formula

$$P(\boldsymbol{\chi}) = \sum_{i_1=0}^{m_1} \cdots \sum_{j_d=0}^{m_d} c_{j_1,\ldots,j_d} B_{j_1}^{p_1}(\chi_1) \ldots B_{j_d}^{p_d}(\chi_d). \tag{4.33}$$

In Eq. (4.33), $\boldsymbol{\chi} = [\chi_1, \ldots, \chi_d]$ is the vector at which the function is evaluated, $c_{i_1,\ldots,i_d}$ are elements of the coefficient matrix $\mathbf{C} \in \mathbb{R}^{m_1 \times \cdots \times m_d}$ and B are the one-dimensional basis functions defined in Eq. (3.20). The B-spline has very similar characteristics as its one-dimensional counterpart. The d splines are of degree $(p_1, \ldots, p_d)$, defined on d knot vectors $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_d)$ of sizes $(n_{knots,1} + 1, \ldots, n_{knots,d} + 1)$. The control points belong to the coefficient matrix of size $m_1 \times \cdots \times m_d$. For each dimension i, it holds that $n_{knots,i} = m_i + p_i + 1$.

With the describing power of the multivariate function, the goal is to now construct a grid of values to be interpolated. In order to do that, the grid structure constructed for sampling-based methods in Chapter 3.2.3 can be leveraged. In order to accurately model the visibility of a give landmark from a pose $\mathbf{T}_i$, the perspective projection model described in Chapter 2.4 is employed. With known camera parameters, the visibility of landmark l at position $\boldsymbol{\rho}_l$ is computed by verifying that its image coordinates (u, v) are within the width and height of the image in pixels w, h. Performing this check for all landmarks stored in the map $\mathcal{M}$, the number of visible landmarks can be stored for any pose. By iteratively performing this procedure on a grid, it is possible to span the entire environment and create a function $V(\mathbf{T}, \mathcal{M})$ which returns a single value, which will be referred to as the feature-density ρ_f. The resulting grid would require to span the space of rigid body motion, thus requiring a parametrization through 6 parameters. Through the simplifying assumption that the rotation around the camera axis z_c does not substantially affect the resulting view, therefore one variable of rotation is removed. This leaves the representation to a 5-dimensional grid that is parametrized

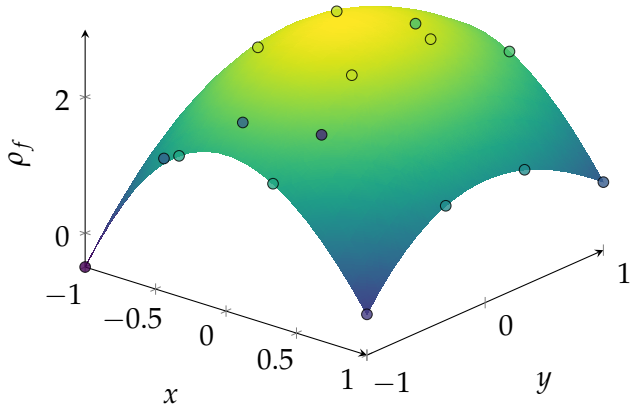


Figure 4.1: Surface interpolation through tensor-product spline of a convex function.

through position $\mathbf{p}$ and two rotational parameters Θ, Ψ. The grid is finally interpolated, obtaining the scalar value $\rho_f = P(\chi_i) \approx V(\mathbf{T}, \mathcal{M})$, with $\chi = [x_i, y_i, z_i, \Theta_i, \Psi_i]$.

The outline of the approach is described in Algorithm 8. First, the discretization of the grid is defined through the number of cells and the environment limits $mathbfl_{SE(3)}$. Once the grid is defined, the procedure spans through each of the five dimensions to construct poses covering the entire environment $\mathbf{T}_{i,j,k,l,m}$. Each pose is subsequently stored within the input grid $\mathcal{G}_{SE(3)}$. Afterwards, each of the poses is passed through the visibility function V together with the given map of the environment. The results are stored in the output grid $\mathcal{V}$. Once the sampling procedure is completed, the interpolation is conducted to obtain the interpolating tensor-product spline P.

For ease of understanding, the result of the methodology for the 2-dimensional space $\mathbb{R}^2$ is shown in Figure 4.1. The figure depicts the scattered density values obtained from the sampled grid, which are fitted through a continuous and smooth third-order polynomial curve. This methodology allows flexibility in the description of the density function based on the order of the interpolating splines and the discretization of the underlying grid. Moreover, if the map $\mathcal{M}$ used for the interpolation corresponds exactly to the one of the environment, the approximation returns the accurate number of visible landmarks seen at the interpolation point. The advantage of this approach, however, lies in the abstract description of feature density. In particular, the map used within the visibility modeling, can be arbitrarily sampled and modified, in order to remove or retain chosen areas and give preferences to certain landmarks over others. By tuning properly the spline degree, the grid and the map, the approximating function can be appropriately shaped in order to improve the performance of GBO numerical approaches. A downside of using approximating polynomial functions is the oscillatory behavior that is inherent to their formulation. Due to this effect, fictitious minima and

Algorithm 8 Feature-density function

$\mathbf{N}_{SE(3)} \leftarrow (N_x, N_y, N_z, N_\Theta, N_\Psi)$
$\mathbf{l}_{SE(3)} \leftarrow [\mathbf{l}_x, \mathbf{l}_y, \mathbf{l}_z, \mathbf{l}_\Theta, \mathbf{l}_\Psi]$
$\mathbf{\Delta l}_{SE(3)} \leftarrow [\Delta x, \Delta y, \Delta z, \Delta_\Theta, \Delta_\Psi]$
for $i \leftarrow 0$ to N_x **do**
  for $j \leftarrow 0$ to N_y **do**
    for $k \leftarrow 0$ to N_z **do**
      for $l \leftarrow 0$ to N_Θ **do**
        for $m \leftarrow 0$ to N_Ψ **do**
          $\mathbf{T}_{i,j,k,l,m} \leftarrow \texttt{pose}(x_i, y_j, z_k, \Theta_l, \Psi_m)$
          $\mathcal{G}_{SE(3)} \leftarrow \mathbf{T}_{i,j,k,l,m}$
          $\mathcal{V} \leftarrow V(\mathbf{T}_{i,j,k,l,m}, \mathcal{M})$
        end for
      end for
    end for
  end for
end for
$P \leftarrow \texttt{interpolate}(\mathcal{G}_{SE(3)}, \mathcal{V})$

maxima might appear in areas where the value of the points to be interpolated changes significantly.

Finally, the cost function associated to the feature-density along a trajectory sampled at K poses $\{\mathbf{T}_i\}_{i\in\mathcal{K}}$, formulated through the 5-parameters representation $\{\boldsymbol{\chi}_i\}_{i\in\mathcal{K}}$ is expressed as

$$\mathbf{\Gamma}_{\mathcal{C}}(\mathbf{T}(t)) = \sum_{i\in\mathcal{K}} \frac{n_{landmarks} - P(\boldsymbol{\chi}_i)}{n_{landmarks} K}, \tag{4.34}$$

where $n_{landmarks}$ indicates the total number of landmarks contained in the map $\mathcal{M}$ used to interpolate the values. Equation (4.34) represents a continuous and differentiable function, which spans the range $[0, 1]$ and reaches a minimum when the value $P(\boldsymbol{\chi})$ reaches a maximum.

4.2.5 Trajectory planning with continuous visibility functions

The cost formulation described above, is suitable to be used in the context of perception-aware trajectory planning through GBO. In the scope of this work, it is also interesting to include the cost formulation within the kinodynamic planning framework introduced in Chapter 3. What is particularly important is to introduce kinodynamic constraints, collision constraints and a smoothing component for the final trajectory. In [BTC20],

this is achieved by formulating the cost function through a series of weighted additive terms. No constraints are explicitly added, as the cost is formulated to include "soft constraints" which account for smoothness, collision avoidance, time optimization and vision constraints. In the context of this thesis, however, the objective is to construct an approach that relies on constrained optimization techniques, in order to achieve the highest optimality of the solution within the provided bounds and satisfaction of the constraints.

The trajectory planning problem formulated in Eq. (3.60) remains completely valid. The formulation allows to use any cost function formulation, thus would allow to be implemented minimizing the cost described in Eq. (4.34). This solution, however, would not be satisfactory for the requirements detailed above. The solution would have no guarantee of delivering an efficient trajectory in terms of actuation, except for not violating the constraints imposed. It is thus relevant to introduce a smoothing term in the cost formulation. Similarly to the approach of [Use+17; BTC20], an weighted sum of cost terms is implemented. As a smoothing term, the mechanical energy cost term defined in Eq. (3.3) is used.

The discrete-time implementation of Eq. (3.3) is

$$\mathbf{\Gamma}_{\mathcal{M}}(\mathbf{T}(t), \dot{\mathbf{T}}(t), \mathbf{u}(t)) \approx \frac{1}{E_{\mathcal{M}}} \sum_{i \in \mathcal{K}} (\dot{\mathbf{T}}_i \cdot \mathbf{u}_i)^2, \tag{4.35}$$

where $E_{\mathcal{M}}$ is a normalizing term that is defined, based on the initial value of the mechanical energy, or the expected maximum value, in order to maintain the cost within $[0, 1]$ for numerical efficiency.

Introducing a weighting term $\lambda_{\mathcal{M}} \in [0, 1]$, the composite cost function comprising of both perception-aware and smoothing component is defined as

$$\mathbf{\Gamma}(\mathbf{T}(t), \dot{\mathbf{T}}(t), \mathbf{u}(t)) = \lambda_{\mathcal{M}} \mathbf{\Gamma}_{\mathcal{M}}(\mathbf{T}(t), \dot{\mathbf{T}}(t), \mathbf{u}(t)) + (1 - \lambda_{\mathcal{M}}) \mathbf{\Gamma}_{\mathcal{C}}(\mathbf{T}(t)). \tag{4.36}$$

This formulation ensures that $\mathbf{\Gamma} \in [0, 1]$ and allows to define the relevance of each component of the sum within the optimization. At the extremes, the function behaves completely as mechanical energy-optimizing ($lambda_{\mathcal{M}} = 1$), or as features-density-optimizing ($lambda_{\mathcal{M}} = 0$).

Feature-density constraint

As a last formulation of interest, the implementation of a constraint based on the feature-density is investigated.

In the context of path-planning under uncertainty, it might be interesting to aim for a planner that determines optimal paths that allow to never fall below a defined threshold

of visible landmarks $t_{landmarks}$. In [Sou+22], the authors define a localization system which relies on tracking landmarks throughout motion. Within their implementation, it is explicitly stated that the localizer requires a minimum number of tracked features in view, as well as a saturation value over which tracking is not necessary.

To this end, the feature density function could easily be incorporated within a constraint in the form of Eq. (3.60f)

$$\mathbf{g}(\mathbf{T}(t)) = t_{landmarks} - P(\boldsymbol{\chi}(t)) \leq 0. \tag{4.37}$$

This constraint can then be included in the optimization problem from Eq. (3.60) without any other modification.

5 Implementation and Results

5.1 Introduction

In the following chapter, numerical results for the methodologies devised in this thesis are presented. Delving into the applications of the algorithm presented above, their practical applicability and functionality is shown. In particular, this chapter presents the numerical results of the kinodynamic and perception-aware path planning algorithms designed for holonomic robots operating in $SE(3)$. Their implementation is then employed on both the Astrobee and the robotic manipulator platforms.

5.1.1 Hardware and software environment

The translation of innovative algorithms into reality necessitates a seamless interaction between computational power and tangible hardware. The implementation of the algorithm was coded in C++ and Python and the computations have been performed on a laptop Lenovo Thinkpad T440 from 2016 with the following characteristics: processor `Intel i5-4300U` and Ram `8Gb`. The algorithms have been designed to be deployed in Robot Operating System (ROS) as well as the in-house equivalent Links and Nodes (LN). The path planning algorithms have been successfully deployed in the Astrobee simulation environment [NAS] and in simulation on the OOS-SIM.

5.1.2 Overview

This chapter is structured to provide a comprehensive understanding of the implementation process and the outcomes achieved. Each section demonstrates the effect of the methodologies developed in the previous chapter, by detailing the implementation details and showcasing the most relevant results.

The following results are hereby presented:

- optimal path planning in $SE(3)$,
- sampling in $SO(3)$,
- applications of the RRT*-GBO,

- study of maps of the ISS,
- analysis of the feature-density function,
- results of perception-aware planning in $2D$ and $3D$

5.2 Kinodynamic Path Planning

In the following, the results of the methods of path planning defined in Chapter 3 are shown.

The objective of the methodologies devised, is to implement an efficient algorithm to search the solution for highly constrained kinodynamic path planning problems in the context of free-flying robots. As such the algorithms are deployed on the Astrobee robot described in Chapter 2.1.

5.2.1 Optimization on the special Euclidean group $SE(3)$

The optimization problem described in Section 3.2.2 is numerically implemented. The problem is constructed by defining two B-splines, one for the translational trajectory on $\mathbb{R}^3$ and one for the rotational trajectory on $SO(3)$. The parameters characterizing the B-Spline parametrization of the state-space on $SE(3)$ are reported in Table 5.1.

Table 5.1: B-spline parameters on $SE(3)$.

Space	$t_0 - t_f\,[s]$	p	n_{knots}	m
$\mathbb{R}^3$	$0-60$	3	20	16
$SO(3)$	$0-60$	2	20	17

Given the parameters, the trajectory on $\mathbf{R}^3$ is represented by a piecewise third-order curve, which leads to continuity up to the third derivative (jerk). The one on $SO(3)$, instead, results in a piecewise second-order curve, meaning its derivatives result continuous up to the second derivative (angular acceleration).

Here reported is an exemplifying case of the capability of the trajectory planning through interpolating curves. In particular, a tridimensional reorientation maneuver is performed in order to truly test the ability of the algorithm.

The trajectory connects the following boundary conditions:

- $\mathbf{p}_0 = [0,0,1]^T\, m$, $\mathbf{p}_f = [3,0.5,0]^T\, m$
- $\mathbf{v}_0 = [0,0,0]^T\, \frac{m}{s}$, $\mathbf{v}_f = [0,0,0]^T\, \frac{m}{s}$

- $\mathbf{a}_0 = [0,0,0]^T \frac{m}{s^2}$ $\mathbf{a}_f = [0,0,0]^T \frac{m}{s^2}$
- $\mathbf{j}_0 = [0,0,0]^T \frac{m}{s^3}$, $\mathbf{j}_f = [0,0,0]^T \frac{m}{s^3}$
- $\mathbf{R}_0 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{R}_f = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
- $\boldsymbol{\omega}_0 = [0,0,0]^T \frac{rad}{s}$, $\boldsymbol{\omega}_f = [0,0,0]^T \frac{rad}{s}$
- $\dot{\boldsymbol{\omega}}_0 = [0,0,0]^T \frac{rad}{s^2}$ $\dot{\boldsymbol{\omega}}_f = [0,0,0]^T \frac{rad}{s^2}$.

The cost function employed is the mechanical energy defined in Eq. (3.3). The optimal path is a straight line on $SE(3)$, as it is the shortest path connecting the two poses. The velocity profiles analytically express parabolic curves while the accelerations express linear functions.

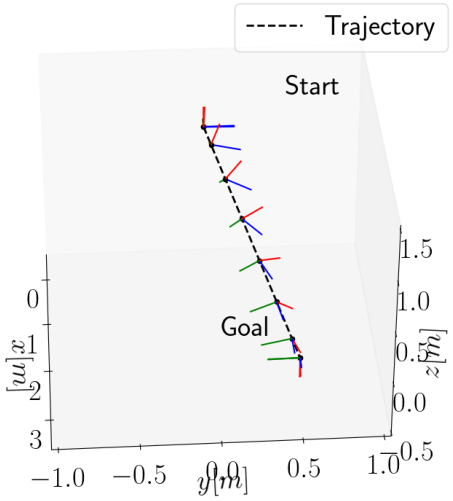
With respect to the algorithm implementation, the resolution is obtained by sampling the trajectory at $n_{viaPoints}$ waypoints. The numerical optimization is performed using the following parameters:

- relative tolerance: 10^{-8},
- absolute tolerance: 10^{-9},
- $n_{viaPoints} = 120$,
- permutation parameter for numerical gradients: $\delta c = 10^{-4}$,
- constraints tolerance: 10^{-5},
- maximum number of iterations: 200.

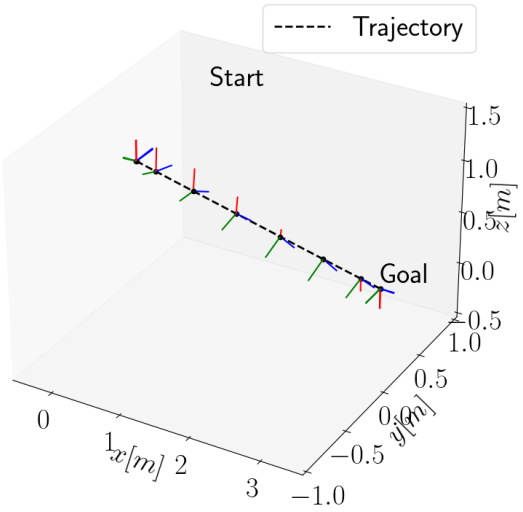
Figures 5.1 and 5.2 depict the overview of the maneuver, as well as showing the profiles of the dynamic quantities explicitly represented. In Figure 5.1, the position is represented by the dashed line (- - -), while the orientation of the body is expressed by the three axis of the body frame. Each axis is depicted as a straight solid line: x_B in blue (—), y_B in green (—) and z_B in red (—).

By observing the plot represented in Fig. 5.2, the solution approximates the parabolic shape of the theoretical solution, as described in [Lam+11].

As denoted in Eq. (3.25), B-splines derivatives are B-spline themselves. The higher the derivative's order, the lower the degree of the corresponding B-spline. As lower order B-splines have less interpolation capability, given the same knot vector [BM08], the higher derivatives cannot retain as much information. This effect is especially



(a)



(b)

Figure 5.1: Three-dimensional depiction of the optimal reorientation maneuver with three-dimensional rotation. (a) Longitudinal view, (b) lateral view.

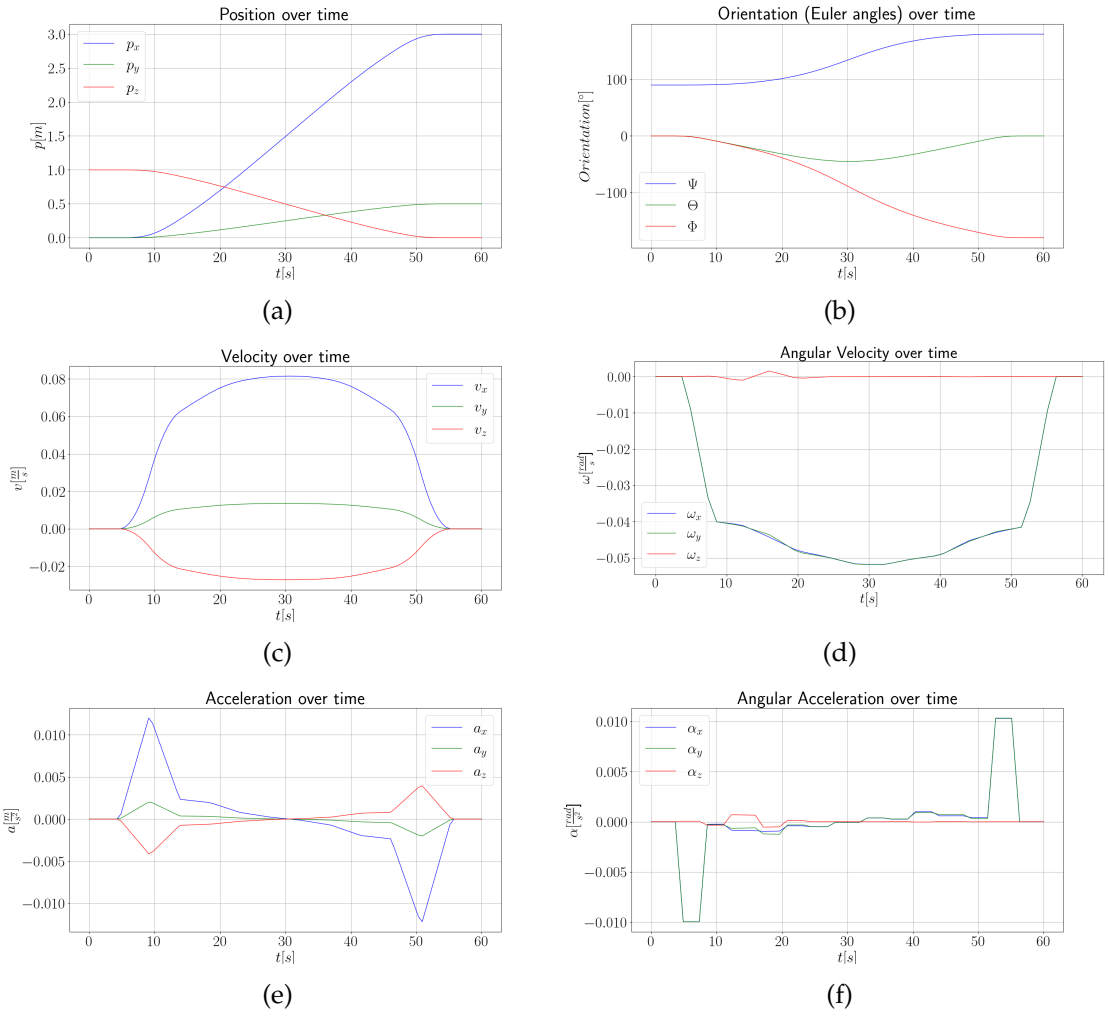


Figure 5.2: Profile of the dynamic quantities of a tridimensional reorientation maneuver. (a) position, (b) orientation, (c)-(d) velocities and (e)-(f) accelerations.

relevant in the angular motion derivatives, where the angular velocity, being a first order polynomial, can only tend towards the theoretical parabolic solution.

However, the resulting trajectory correctly determines the optimal solution. The trajectory does not present constraint violations as none of the imposed constraints on the dynamics are active, and no obstacle is present in the environment. The boundary conditions are fulfilled, as can be seen by the plots beginning and ending portions.

The performance of the resolution algorithm are described in Table 5.2.

Table 5.2: Performance of numerical optimization for the $3D$ reorientation maneuver.

Cost $[J]$	Inequality constraints	Optimization Time $[s]$	Iterations
0.0945	$16 \times n_{viaPoints}$	1.629	23

5.2.2 Sampling on $SO(3)$

Within this section, the implementation of the random sampling approach on $SO(3)$ exploited in Section 3.2.3 is depicted for ease of understanding.

In [Yer+10], the authors do not show detailed results of their methodology in the context of sampling-based planning. Here an intuitive representation of the formulated approach is proposed. In particular, similarly to the plotting method used by [Kuf04], the $SO(3)$ grid described in Section 3.2.3 is portrayed using the angle-axis representation of orientations. Each element of the manifold is represented as a vector tangent to the sphere, its position on the surface together with the direction of the vector are determined based on the angle and the axis of the corresponding element.

Figure 5.3 shows the outline of the grid structure, in particular depicting the difference between discretization levels, demonstrating the use of incremental grids. Figure 5.4 shows a set of randomly sampled elements on grids with increasing refinement, showcasing that with increasing resolution and samples, the space of $SO(3)$ is progressively covered by maintaining high degree of uniformity and discrepancy.

This sampling approach can be efficiently applied within the context of kinodynamic path planning.

5.2.3 RRT*-GBO on $SE(3)$

The implementation of the RRT*-GBO on $SE(3)$ with optimal propagation of motion, discussed in Chapter 3.2.3 is here implemented and applied within the context of

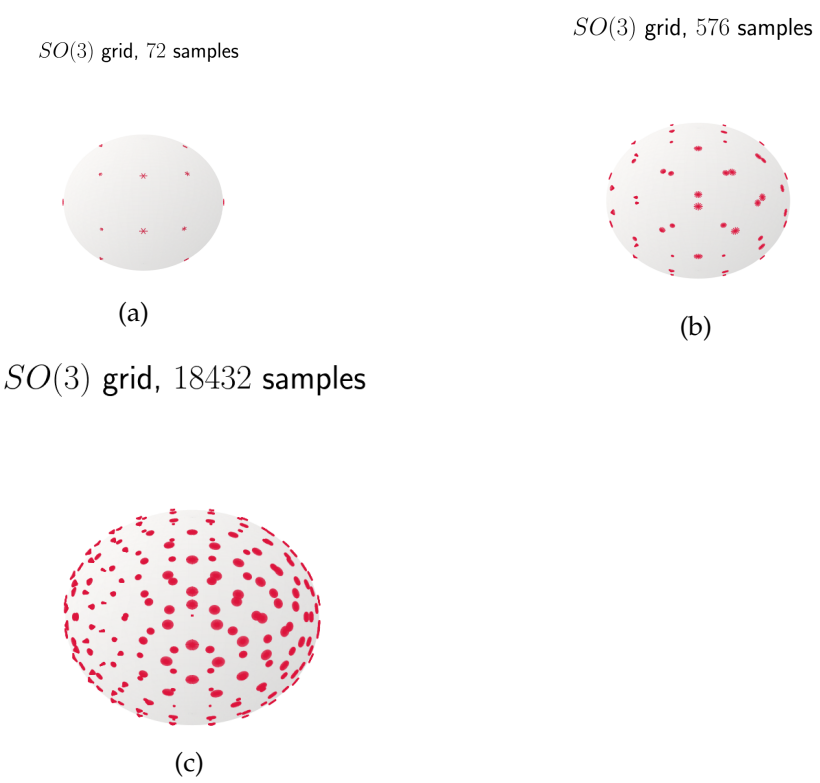


Figure 5.3: Angle-axis representation of the incremental grid structure on $SO(3)$ with increasing refinements. Depicted are 3 refinement levels l: (a) $l = 1$, $m_1 = 6, m_2 = 12$, (b) $l = 3$, $m_1 = 12, m_2 = 48$ (c) $l = 4$,$m_1 = 24, m_2 = 768$.

Figure 5.4: Angle-axis representation of random sampling of elements on $SO(3)$. (a) Random 100 samples on grid $l = 2$ (b) Random 500 samples on grid $l = 4$.

motion within constrained environments. To introduce the main characteristics of the algorithm, a series of preliminary descriptive use cases are shown.

Preliminary examples

First,the capability of the tree-search to avoid obstacles and the importance of the `connect_to_goal` functionality of the algorithm are demonstrated in Fig. 5.5 and Fig. 5.6. Starting with Fig. 5.5, the picture represents an obstacle placed to the upper right from the starting position of the robot. The problem of interest starts statically at position $[0, 0, 0]^T m$ and ends statically at $[4, 4, 4]^T m$. An obstacle is placed at $[1.5, 1, 1]$, which is in a position that would disrupt the theoretical solution connecting the `Start` and `Goal` with a straight line. The algorithm, however, immediately finds the trajectory connecting the start to the goal node due to the `connect_to_goal` call at the beginning of the path planning process. This solving edge is denoted as "`Start, Goal`" within the plot. Moreover, the algorithm samples the configuration space attempting to find other feasible solutions. The node `1` is found in the vicinity of the `Start` and similarly connected to the endpoint. As the other nodes are relatively far, the connection to the `Goal` from them is not drawn out, due to infeasibility of the edge. Once the limit of iterations is reached, the algorithms selects the most optimal solution. In this case the path `Start` $\rightarrow$ `Goal`, as it is found by slightly altering the motion to avoid the obstacle. In fact, it is mostly the area in the vicinity of the object to be affected by the constraint, and that is visible as the trajectory only slightly curves to the left side of the red box. It is clear that the alternative solution, `Start` $\rightarrow$ `1` $\rightarrow$ `Goal` requires a larger actuation effort as it performs an expensive course correction to avoid the obstacle. The colorbar allows to interpret that the smoothed trajectory's color lies in the lowest end of the spectrum, meaning that the algorithm has obtained the minimum cost trajectory between the ones it had previously found. The final solution smooths the turn near the obstacle in order to make it less aggressive, this achieves lower cost by limiting the peaks in forces and moments in favour of smooth profiles which limit power consumption.

Second, an example to showcase the construction of tree edges is reported in Fig. 5.6. The figure depicts a uncluttered view of the tree constituted previously. This case only considers three nodes: `Start`, `Goal` and node `1` at position $[0.5, 0.5, 0.5]^T m$. The depiction demonstrates how the algorithm constructs an edge stemming from the starting conditions into a sampled node. The methodology immediately finds an optimal solution for the edge (—). The obtained path is a straight line linking the two positions As there are no boundary conditions imposed to reach node `1`, the optimization delivers a set of boundary conditions dictated simply by the lowest cost achievable by fulfilling the applied constraints.

This effect has consequences on the new edge that departs from node `1`. The trajectory

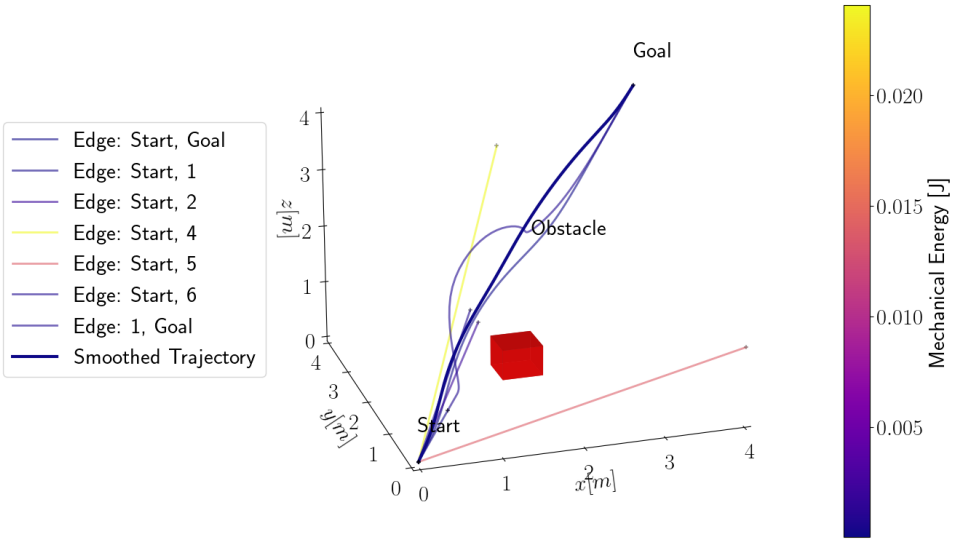


Figure 5.5: Solution of the RRT*-GBO on $SE(3)$ showing its ability to reach the `Goal` rapidly while avoiding obstacles through both sampling around it and leveraging the edge formulation.

(—) now starts with a nonzero velocity which would lead into the obstacle. Therefore, the new edge is forced to curve to avoid the collision, which leads to increased actuation effort and thus higher cost. This behavior, however, leads the obtained trajectory `Start` $\rightarrow$ `1` $\rightarrow$ `Goal` to be a possible feasible initial guess for further smoothing. Although this possible solution has not been chosen to be smoothed by the algorithm, Fig. 5.6 shows that the smoothed solution (—) and this candidate traverse a similar path in the configuration space.

Real-life scenarios

The aim of this work is to replicate real-life path planning scenarios for free-flying robots using the RRT*-GBO. In this section, two of such cases are presented.

First, a planar reorientation maneuver within an enclosed environment, by moving around a static obstacle (another Astrobee in this case). Then the same problem is augmented in order to replicate a more challenging setting, aiming to replicate a planar variation of the Tumbledock experiment. It is assumed that, at the goal position, the robot should have pre-determined boundary conditions, in order to simulate rendezvous with a rotating target.

The first of the two problems at hand is defined by the following boundary conditions:

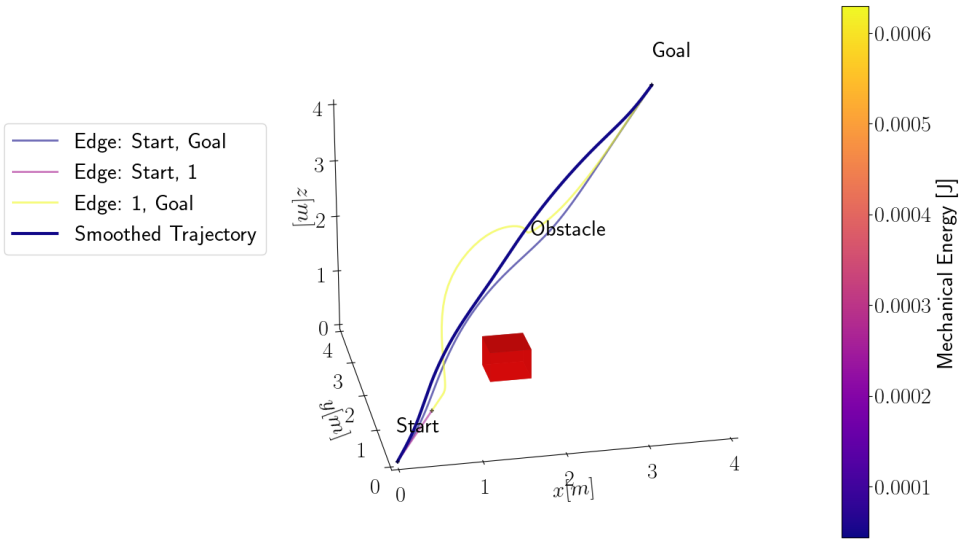


Figure 5.6: Depiction of the construction of edges through optimal propagation of motion. The composite candidate solution `Start` $\rightarrow$ `1` $\rightarrow$ `Goal` is not only continuous and differentiable, but traverses a similar path to the one of the smoothed trajectory.

- $\mathbf{p}_0 = [0,2,2]^T\, m$, $\mathbf{p}_f = [3,2,2]^T\, m$
- $\mathbf{v}_0 = [0,0,0]^T\, \frac{m}{s}$, $\mathbf{v}_f = [0,0,0]^T\, \frac{m}{s}$
- $\mathbf{a}_0 = [0,0,0]^T\, \frac{m}{s^2}$ $\mathbf{a}_f = [0,0,0]^T\, \frac{m}{s^2}$
- $\mathbf{j}_0 = [0,0,0]^T\, \frac{m}{s^3}$, $\mathbf{j}_f = [0,0,0]^T\, \frac{m}{s^3}$
- $\mathbf{R}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{R}_f = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ corresponding to a rotation of 180° around the z-axis,
- $\boldsymbol{\omega}_0 = [0,0,0]^T\, \frac{rad}{s}$, $\boldsymbol{\omega}_f = [0,0,0]^T\, \frac{rad}{s}$
- $\dot{\boldsymbol{\omega}}_0 = [0,0,0]^T\, \frac{rad}{s^2}$ $\dot{\boldsymbol{\omega}}_f = [0,0,0]^T\, \frac{rad}{s^2}$.

The environment is confined within a $3m \times 3m \times 3m$ box and the obstacle is placed at $\mathbf{p}_{obj} = [2,2,2]^T\, m$. The complete graph of this case is shown in Figures 5.7 and 5.8. The pictures presents the entire graph as well as the final smoothed trajectory. In order to reach a solution in a short time, the optimizers settings are deliberately set to settle for largely suboptimal solutions, in order to leverage the constraint satisfaction and velocity propagation capabilities without requiring long computational times. Namely, the number of via points used is lowered to 60 and the relative tolerance is chosen to be 10^{-6} for the tree construction. This leads to suboptimal behavior is some of the edges, such as sudden course changes which appear as spikes. These occurrences lower the optimality of the initial guess constructed by the tree, but easily disappear when the smoothing is performed with tighter settings.

The naive straight line, which would be the optimal trajectory in this context, connecting the start to the goal would collide with the obstacle, thus the algorithm rejects it immediately and proceeds to the tree construction. After all iterations are concluded, it can be seen that the algorithm has been able to connect only one node to the goal. The formulation of the edges allows to identify very early which edges are not suitable for the solution. For this case, the solution of the tree is the path Start$\rightarrow$ 13 $\rightarrow$ Goal.

As the algorithm required two distinct segments in order to reach the goal, the final trajectory will comprise of a B-spline with $t_f = 120s$. These features allow to not only include an heuristic for final time computation, but also guarantee to fulfill constraints and reduce the final cost by reducing the required actuation.

The solution is interpolated and smoothed to obtain the trajectory shown in Fig. 5.9. The development of the state and state derivatives are reported in Fig. 5.10. The presence of the collision constraint strongly affects the solution, in particular, it forces

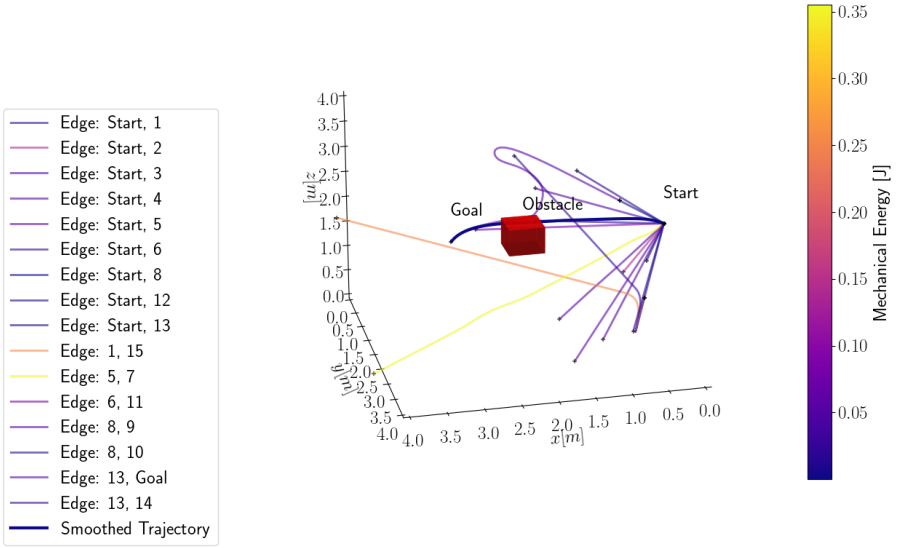


Figure 5.7: Complete tree and trajectory for the static Tumbledock simulation. The edges of the graph are shown and labeled.

the optimizer to deviate from the optimal solution and thus the parabolic shape of the velocity profiles is affected. Due to its high nonlinearity, the resulting trajectory presents a more erratic behavior, especially in the instants where the robot is closest to the obstacle. As the obstacle avoidance models the entire tridimensional object, the attitude plays a role as well. Figure 5.9 allows to see that the robot faces towards the obstacle to avoid its corners. For clarity, a sketch of the concept is presented in Fig. 5.11. If not rotated, the robot (green) would collide with the obstacle (red), while following the trajectory (—).

Nonetheless the algorithm is able to find a solution which cost is the lowest and approximates an optimal solution.

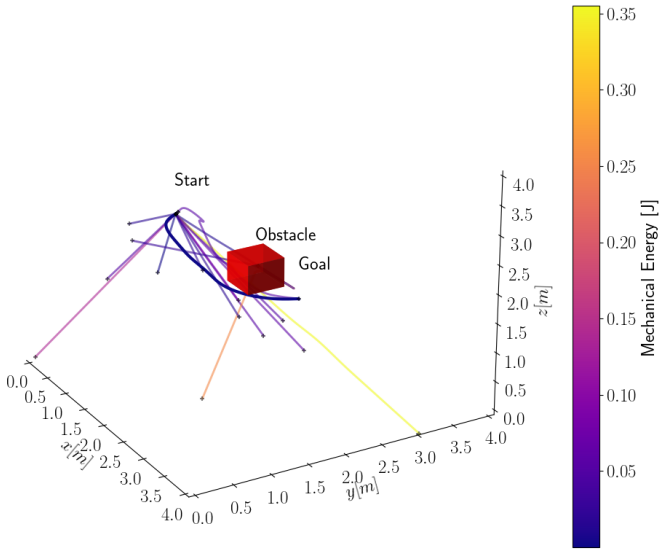
The second problem assumes a target object rotating in place with angular velocity $\boldsymbol{\omega}_{obj} = [0, 0, 0.01]^T \frac{m}{s}$. The setup of the simulation assumes that the meeting point for the robot will be at position $\mathbf{p}_f = [3, 2, 2]^T\, m$. Therefore the boundary conditions for the robot in order to synchronize its motion with the rotating object, are obtained through:

$$\mathbf{v}_f = \boldsymbol{\omega}_{obj} \times (\mathbf{p}_f - \mathbf{p}_{obj}) \tag{5.1}$$

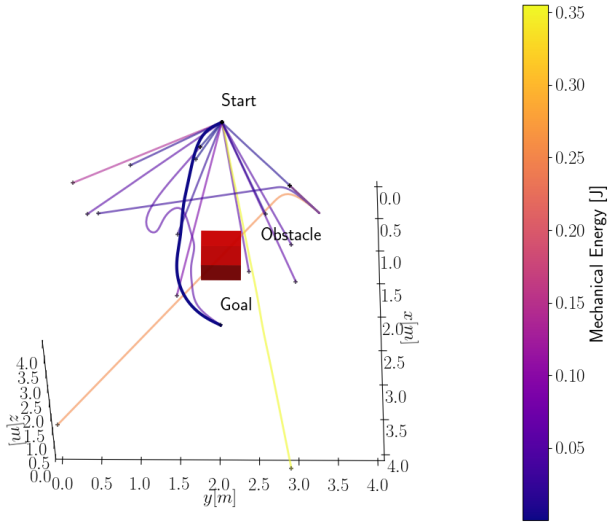
$$\mathbf{a}_f = -\boldsymbol{\omega}_{obj} \times \mathbf{v}_f \tag{5.2}$$

$$\boldsymbol{\omega}_f = \boldsymbol{\omega}_{obj}. \tag{5.3}$$

As a result, the boundary conditions for the rotating Tumbledock problem are:



(a)



(b)

Figure 5.8: Additional views for the static TumbleDock tree. (a) View from above, (b) view from the front.

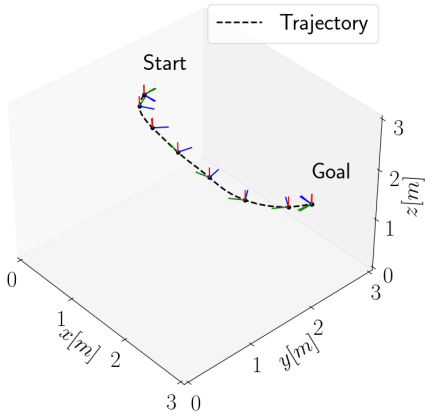


Figure 5.9: Final trajectory obtained with the RRT*-GBO in the static obstacle avoidance problem for an Astrobee.

- $\mathbf{p}_0 = [0, 2, 2]^T\, m$, $\mathbf{p}_f = [3, 2, 2]^T\, m$
- $\mathbf{v}_0 = [0, 0, 0]^T\, \frac{m}{s}$, $\mathbf{v}_f = [0, 0.01, 0]^T\, \frac{m}{s}$
- $\mathbf{a}_0 = [0, 0, 0]^T\, \frac{m}{s^2}$ $\mathbf{a}_f = [-0.001, 0, 0]^T\, \frac{m}{s^2}$
- $\mathbf{j}_0 = [0, 0, 0]^T\, \frac{m}{s^3}$, $\mathbf{j}_f = [0, 0, 0]^T\, \frac{m}{s^3}$
- $\mathbf{R}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{R}_f = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- $\boldsymbol{\omega}_0 = [0, 0, 0]^T\, \frac{rad}{s}$, $\boldsymbol{\omega}_f = [0, 0, 0.01]^T\, \frac{rad}{s}$
- $\dot{\boldsymbol{\omega}}_0 = [0, 0, 0]^T\, \frac{rad}{s^2}$ $\dot{\boldsymbol{\omega}}_f = [0, 0, 0]^T\, \frac{rad}{s^2}$.

Figures 5.12 and 5.13 represent the tree and the overview of the solution. The pictures depict how the tree changes when boundary conditions are applied. The difference is relevant as there is an increased concentration of edges on the side of the obstacle, as the introduction of a velocity component in the y-direction, biases the preferred direction of the edges connecting to the goal.

The solution of the algorithm is presented in Fig. 5.14 and the state and its derivatives are reported in 5.15. It can be seen how the boundary conditions are imprinted onto the trajectory and followed.

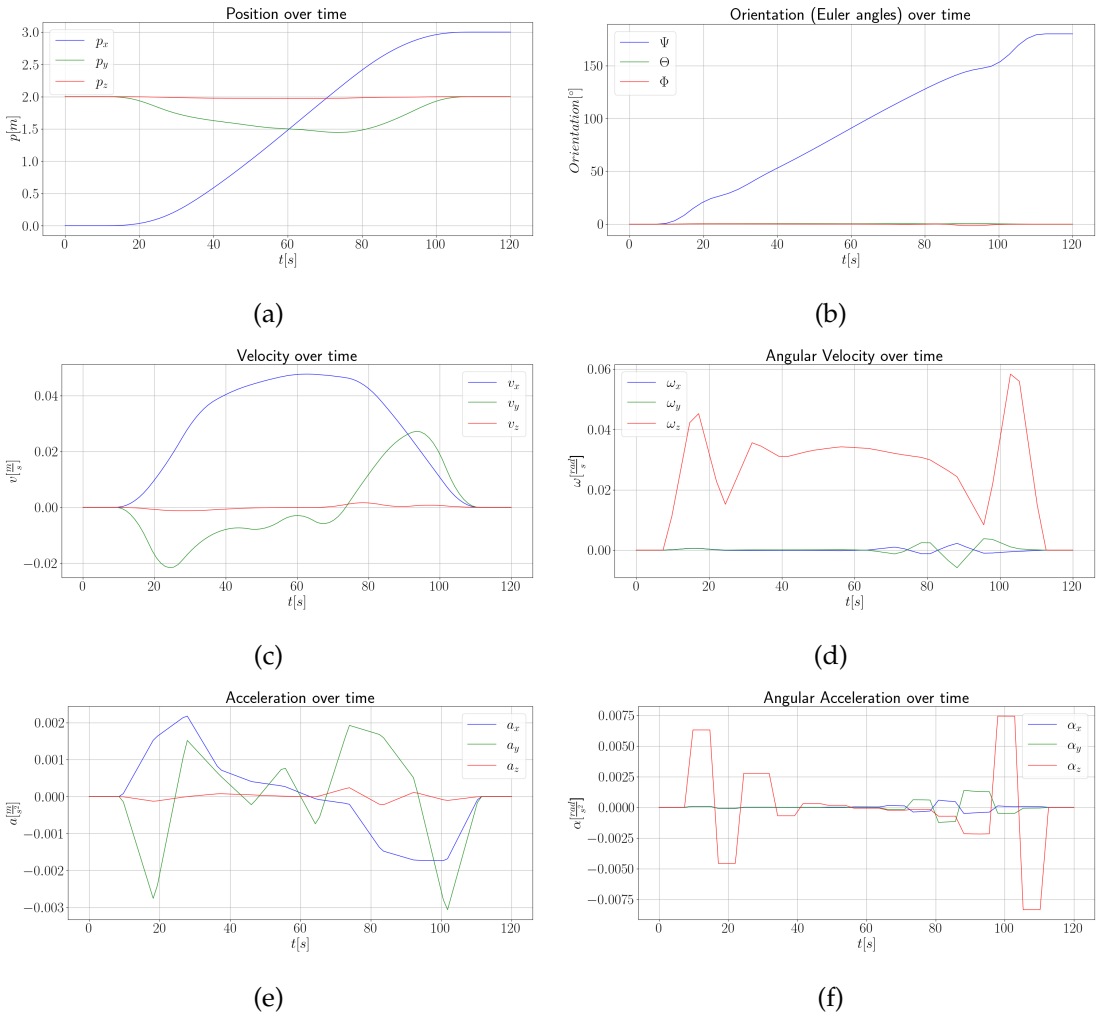


Figure 5.10: State and derivatives of the final trajectory obtained with the RRT*-GBO in the static obstacle avoidance problem for an Astrobee.

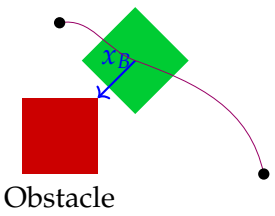


Figure 5.11: Obstacle avoidance through reorientation.

Table 5.3: Performance of RRT*-GBO for real-life scenarios

Case	Iterations	Final Cost $[J]$	RRT time $[s]$	Smoothing time $[s]$	Total time $[s]$
Static	15	0.004704	104.084	3.098	≈ 110
Rotating	20	0.001147	153.840	7.181	≈ 160

The results of the implemented algorithm for the two scenarios are reported in Table 5.3.

As the table shows, the rotating case required more iterations as finding a suitable connection to the goal presented challenges. However, the cost results lower since the robot does not need to completely decelerate when reaching the final position, but rather coasts into position by maintaining its momentum. Finally, it can be seen that the computational times are reasonable if compared with the original implementation from [SL16], although the exact performance comparison in not drawn within this work.

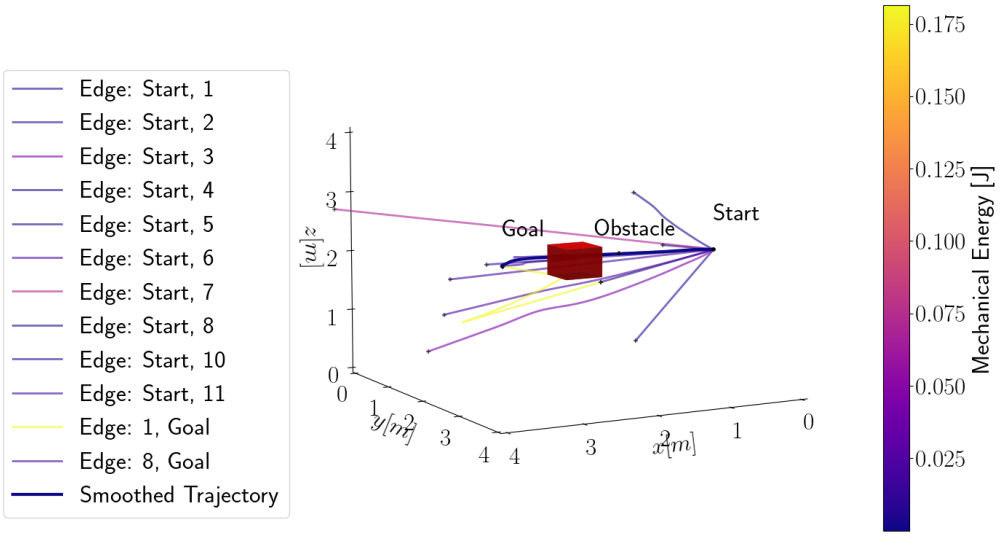
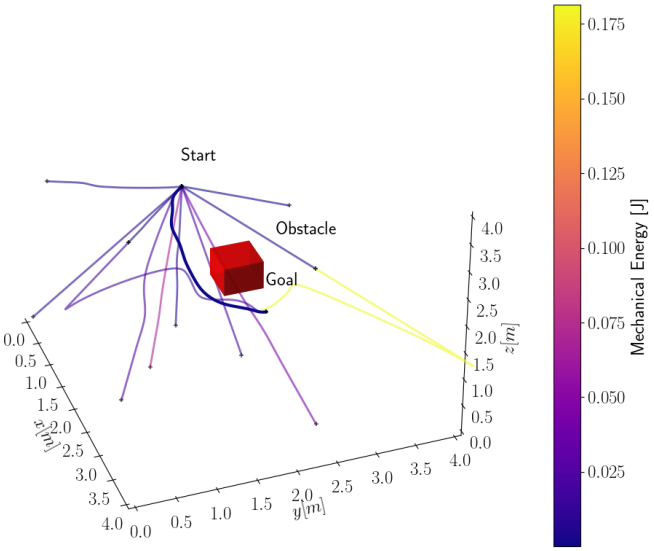


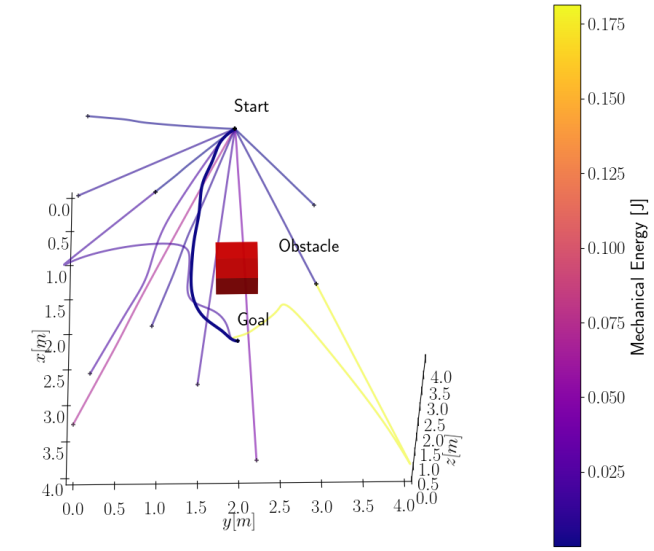
Figure 5.12: Complete tree and trajectory for the rotating Tumbledock simulation. The edges of the graph are shown and labeled.

5.2.4 RRT*-GBO for robotic manipulators

As testing on the ISS has not been possible during the course of this thesis, the results have been tested on the OOS-SIM platform available.



(a)



(b)

Figure 5.13: Additional views for the rotating TumbleDock tree. (a) View from above, (b) perspective view.

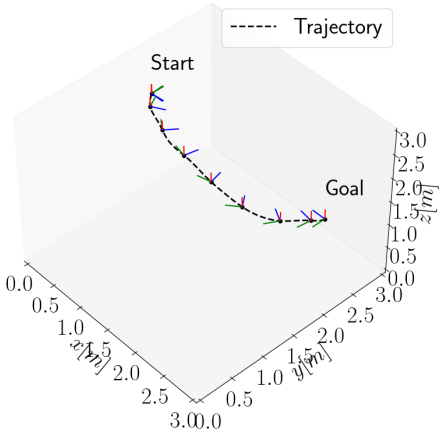


Figure 5.14: Final trajectory delivered by the RRT*-GBO in the rotating obstacle avoidance problem for an Astrobee.

The algorithm has been extended on a Kuka robot with 6 degrees of freedom and tested on a simple trajectory comprising of a 30° rotation with respect to the vertical axis, and a translational motion from $\mathbf{p}_0 = [1.64, 0, 2.245]^T\ m$ to $\mathbf{p}_f = [1.5, 0.8, 2.25]^T\ m$. The derivatives are all set to 0 in this case.

The resulting trajectory was found by the RRT*-GBO at the first step of 15 iterations since the optimal solution was reached at the first iteration by simply connecting start and goal. The algorithm's solution is reported in Fig. 5.16. The figure shows that only 2 other nodes were added, as the addition of the inverse kinematics constraint determines a strong limitation for the feasibility of many edges. The joint angles, velocities and accelerations are shown in Figures 5.17, 5.18 and 5.19. Particular attention is given to the joint angles, as their value must not exceed the thresholds described in Chapter 2.7. The constraint satisfaction is highlighted in the plots. The trajectory has been tested in simulation and a sequence of snapshot from the simulated motion is reported in Fig. 5.20.

5.3 Perception-aware Path Planning

In the following, the results related to the perception-aware path planning are reported.

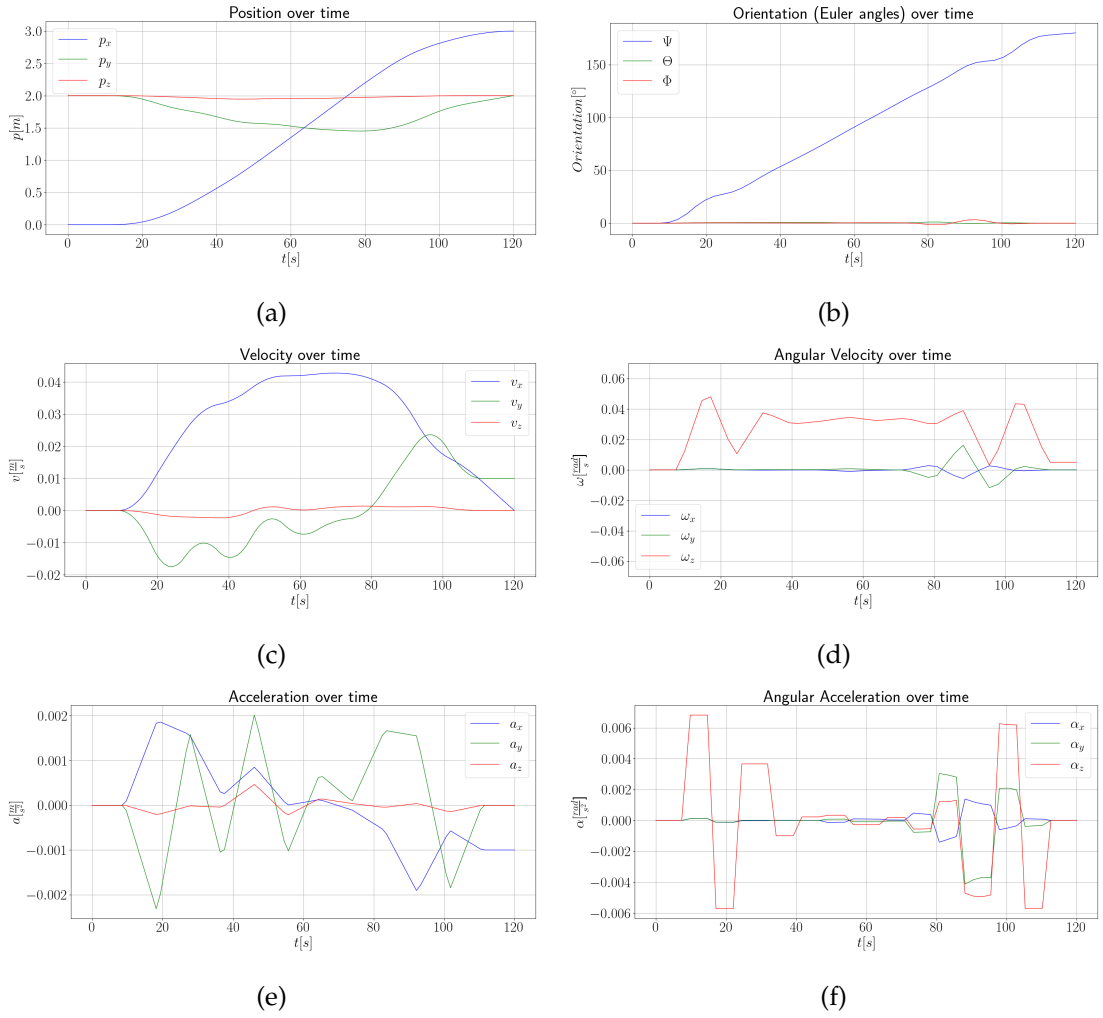


Figure 5.15: State and derivatives of the final trajectory obtained with the RRT*-GBO in the static obstacle avoidance problem for an Astrobee.

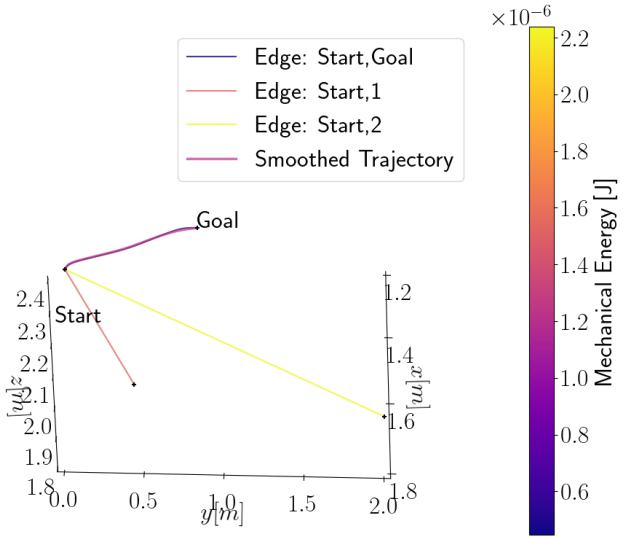


Figure 5.16: RRT*-GBO tree solution on the OOS-SIM.

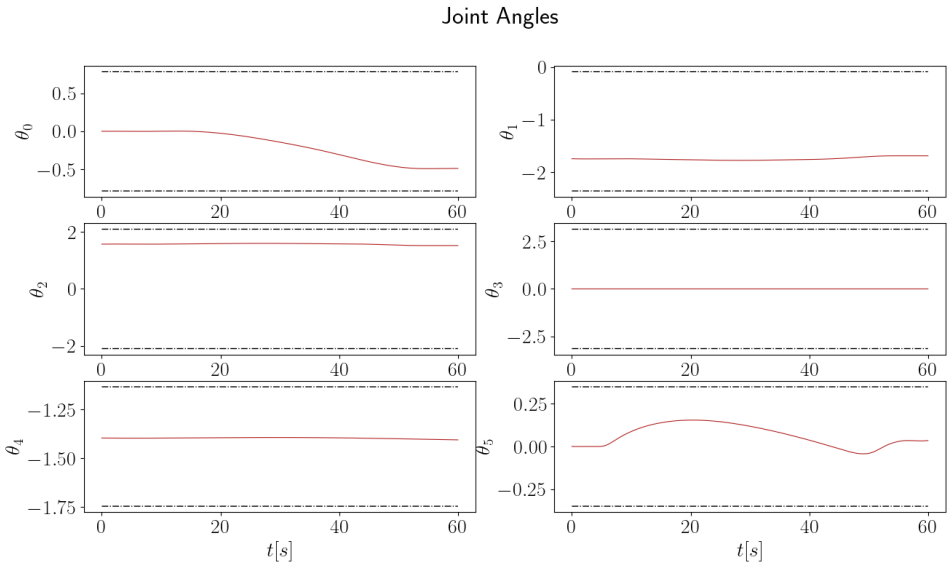


Figure 5.17: Application of the RRT*-GBO on the OOS-SIM. Joint angles with their relative upper and lower bounds.

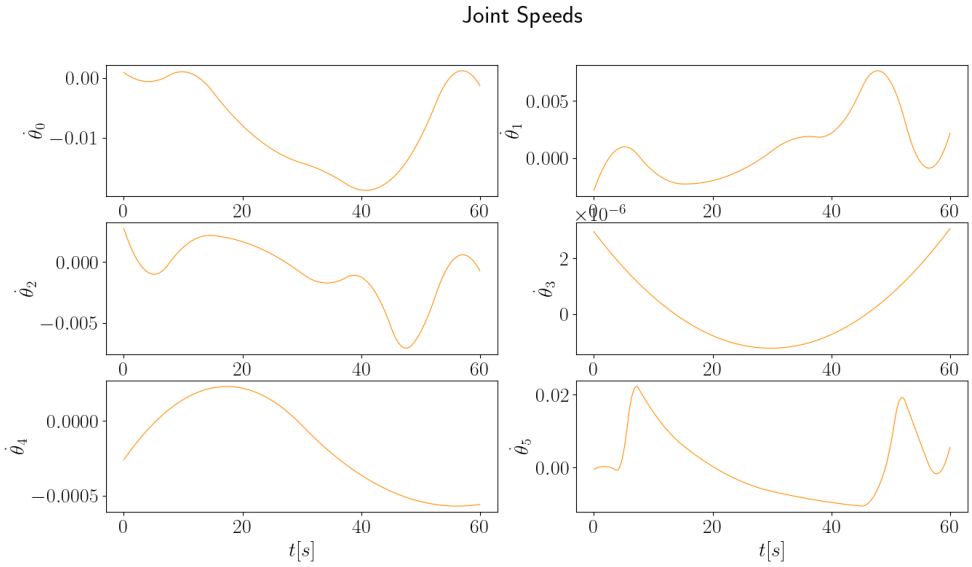


Figure 5.18: Application of the RRT*-GBO on the OOS-SIM.Joint velocities.

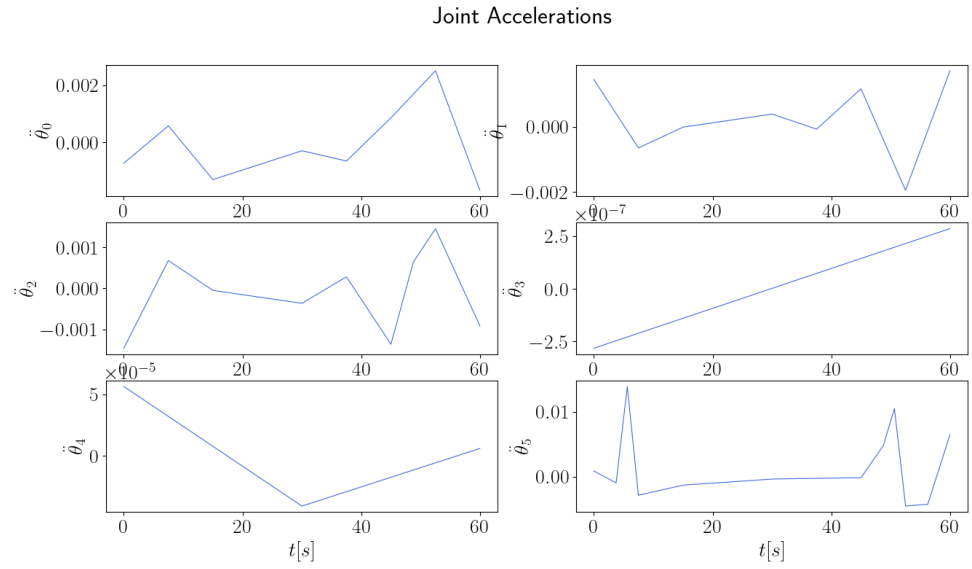
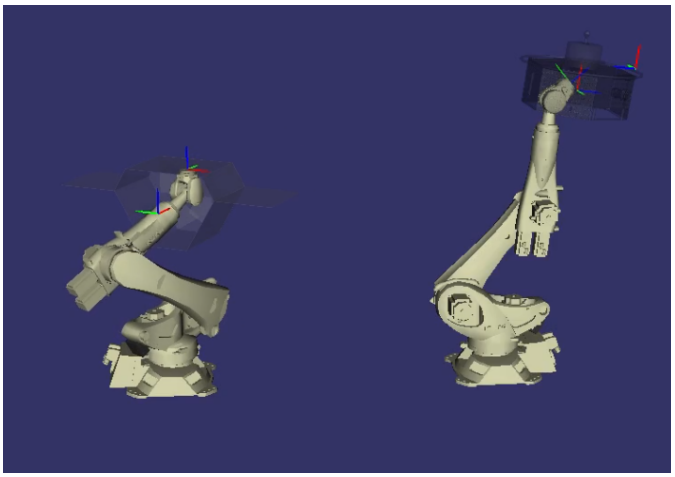
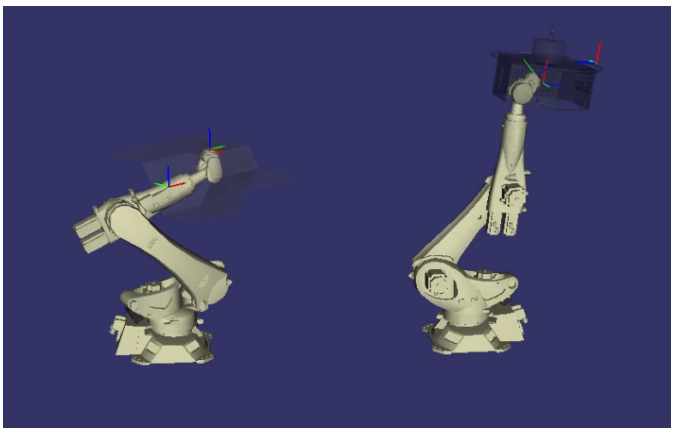


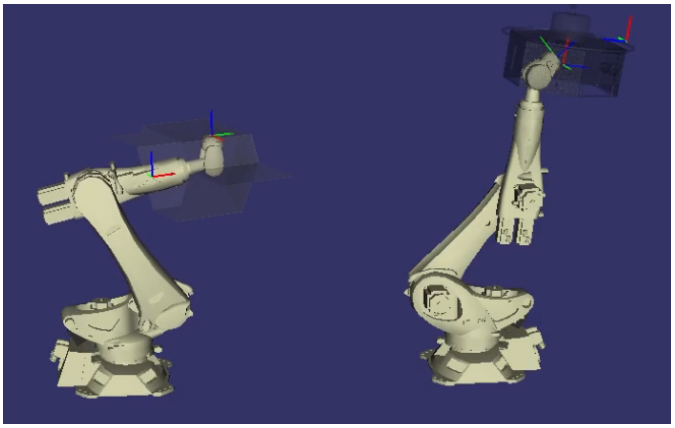
Figure 5.19: Application of the RRT*-GBO on the OOS-SIM. Joint accelerations.



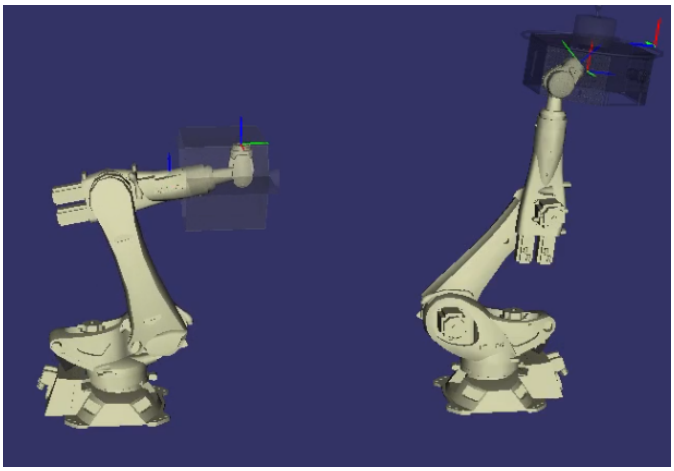
(a)



(b)



(c)



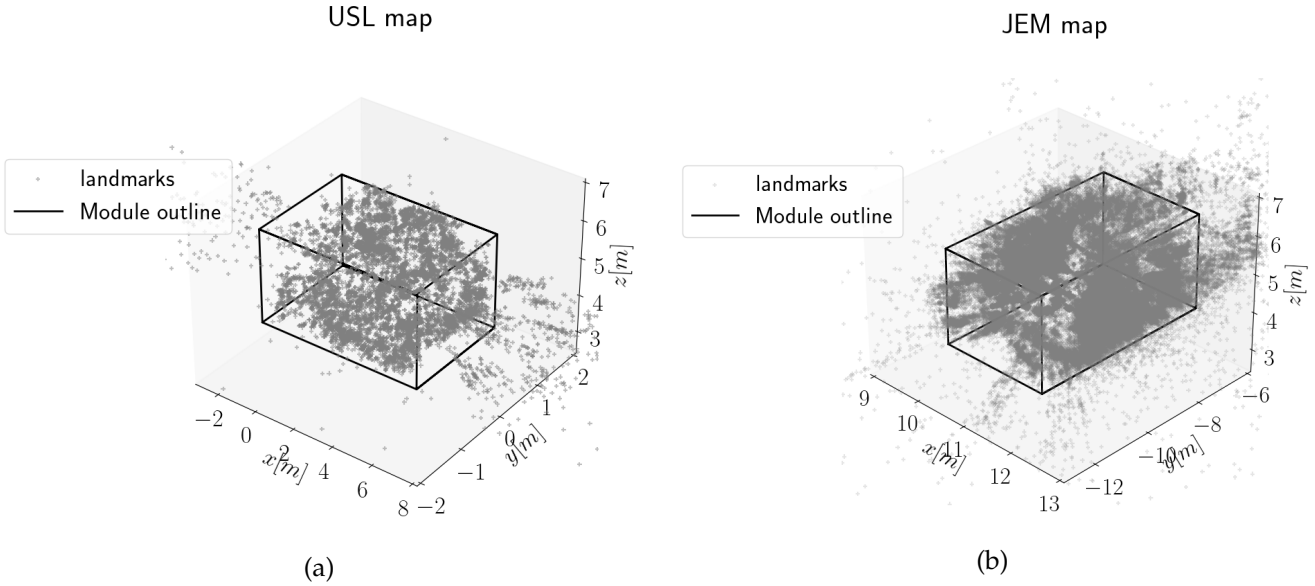
(d)

Figure 5.20: Snapshots of the trajectory on the OOS-SIM. While the robot on the right is positioned to limit possible collisions, the end effector of the robot on the left acts as an Astrobee, translating and rotating. The motion is depicted from (a)-(d) where the first is the initial condition and the last is the end condition.

5.3.1 Environment Maps

As a first step examples of maps of the ISS are shown in Fig. 5.21. Both maps contain a large amount of landmarks, amounting to approximately 10000 for the U.S. Laboratory Module (USL) and 100000 for the Japanese Experiment Module (JEM).

These results show that using methods such as the one from [BTC20] would require iterating over a large number of landmarks for every via point, thus requiring a huge number of numerical operations.



(a) (b)

Figure 5.21: NASA maps obtained through bundle adjustment, from [Col+16b]. (a) USL module, (b) JEM module.

The use of the feature-density function, allows to reduce the computations associated to the map by downsampling it beforehand. By applying random downsampling the maps can be reduced in size without losing the underlying density information. If the downsampling factor is too aggressive, however, the map might lose information and thus the path-planning problem would be affected. The results of downsampling with different factors are shown in Fig. 5.22.

5.3.2 Feature-density function

It is now possible to analyze the feature-density interpolating function described in Chapter 4.2.3. The results discussed here are obtained using the map shown in Fig. 5.22 (d), which corresponds to the map of the JEM downsampled by a factor $f = 100$.

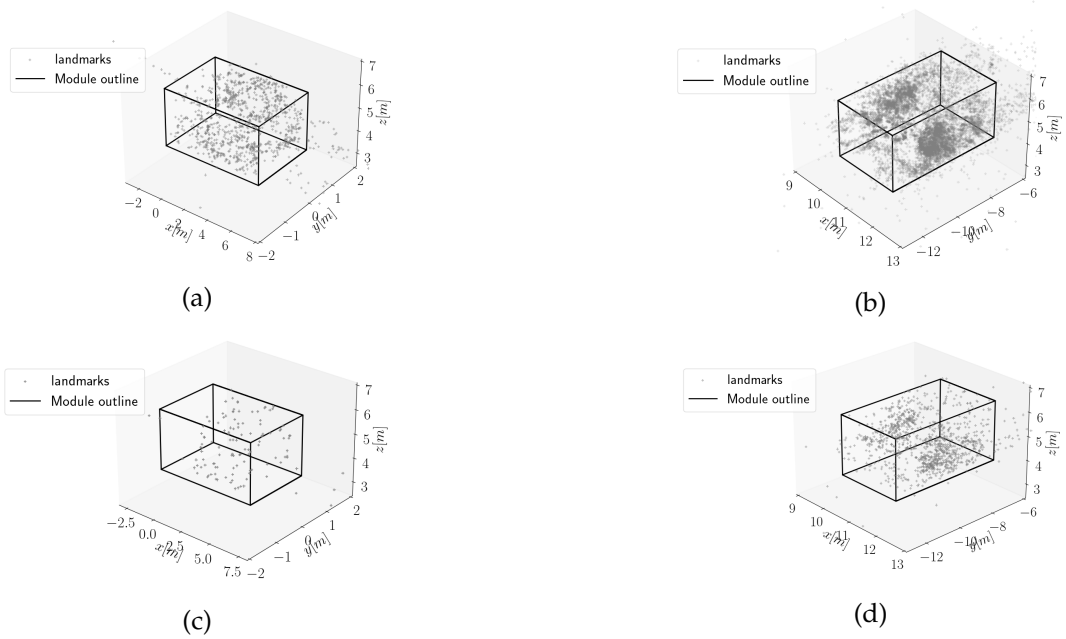


Figure 5.22: Effect of random downsampling on maps. (a) USL map downsampled by a factor $f = 10$, (b) JEM map downsampled by a factor $f = 10$, (c)USL map downsampled by a factor $f = 100$, (d) JEM map downsampled by a factor $f = 100$.

The map reflects the experimental results showing that most of the visible features are located at the further end of the module at $y \approx -10m$ and $z \approx 4.6m$, which corresponds to the airlock area.

The discrete grid is constructed with the parameters defined in Table 5.4.

Table 5.4: Grid parameters for interpolation of feature-density function

Quantity	N	Lower limit	Upper limit
x	5	$10m$	$12.5m$
y	5	$-12m$	$-6m$
z	5	$2.5m$	$7.5m$
Θ	15	$-2\pi rad$	$2\pi rad$
Ψ	15	$-2\pi rad$	$2\pi rad$

The grid is intentionally loose, as the objective is to avoid overfitting in order to obtain a smooth function that doesn't require long computation to be obtained. The variables which are discretized more thoroughly are the ones related to rotation, as it has been noticed that these need higher refinement. Large steps translate to very large changes in orientation (changes of π when using a discretization of 5), which does not allow to produce suitable interpolations as the values change abruptly.

The interpolation is performed using third-order tensor product splines, provided by the Scipy library [Vir+20], leveraging the `scipy.interpolate.RegularGridInterpolator`, which allows to interpolate in arbitrary n-dimensional regular grids.

The results of the interpolation are shown in Fig. 5.23. The pictures report the view of the 4 walls of the JEM as interpolated functions, as well as the effect of a rotation around the z-axis. The plots clearly show that with a suitable parametrization of the grid and interpolation functions, the cost function can be shaped in order to be approximately convex, thus making it very efficient to be used within the GBO approach.

5.3.3 Trajectory planning with continuous visibility functions

In this section, the cost formulations from Eq. (4.34) and (4.36) are discussed and compared to the formulation from [BTC20].

Feature-density function

As a test case, the path planning algorithm is first applied in the context of a two-dimensional motion parallel to the front wall. The trajectory is deliberately started in an area outside of the map, in order to show that at the initial stage of the solution, the

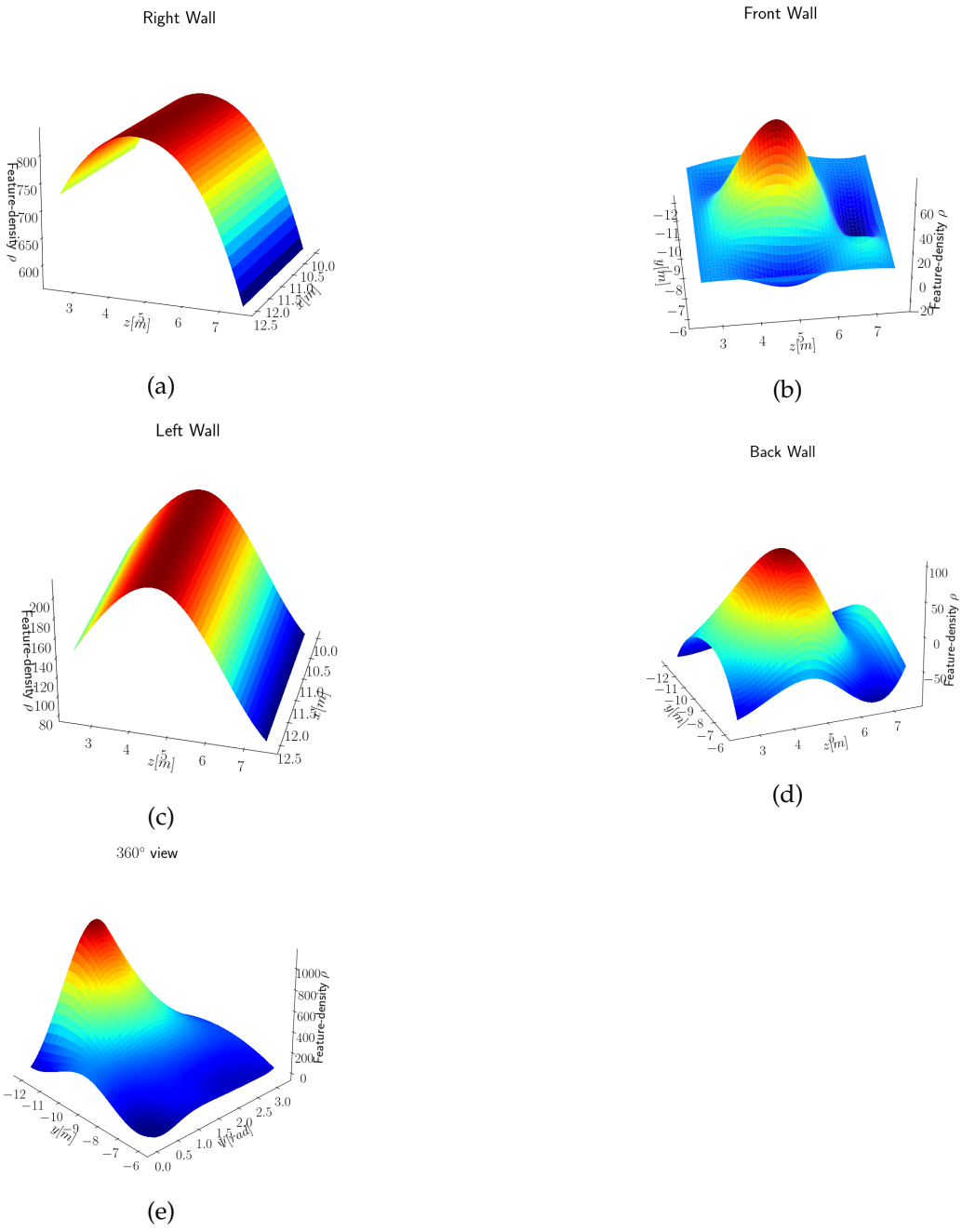


Figure 5.23: Interpolation function on the walls of the JEM. (a)-(c) Sidewalls perpendicular to the y-direction. (b)-(d) Longitudinal walls perpendicular to the x-axis, (e) view obtained rotating around the z-axis.

number of visible features is small. Figure 5.24 shows all the OCP formulations discussed in Chapter 4, providing the mechanical energy-based solution as a benchmark.

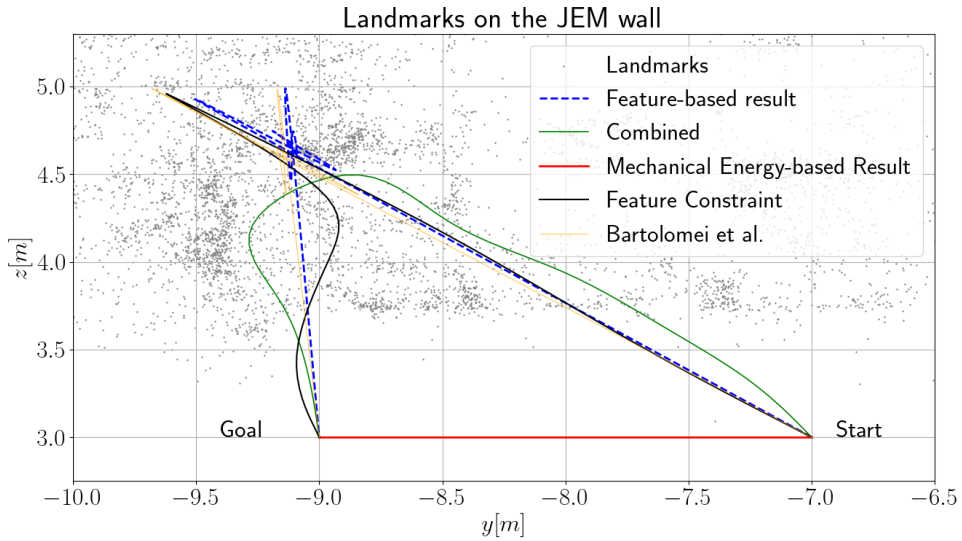


Figure 5.24

The solution from a purely mechanical energy perspective (—) results in a straight ght line between the starting and goal position. The one from the purely feature-based cost function (- - -) , results in the smoothed approximation of a step function, which immediately approaches the area with the highest concentration of visible features and then reaches the goal. The combined cost formulation delivers a smooth trajectory (—) that approaches the more textured area in a more efficient manner. Additionally, the result from the feature constraint formulation of the OCP is reported. The result (—) is a trajectory that closely resembles the one where the cost function is purely feature-based when the constraint is active, while also deviating to cross an additional area of locally higher concentration than the one crossed by the purely feature-based formulation, as the latter attempts to reach the goal position in the shortest amount of time. Towards the end, the trajectory becomes smoother and more efficient. Finally, a comparison is drawn with the cost formulation used [Mur+19; BTC20]. The obtained path (—) is almost identical to the one obtained with the feature-density function. This results is extremely relevant as it demonstrates that in the $2D$ case, the feature-density cost formulation achieves the same results as state-of-the-art methods by exploiting a downsampled map rather than a full description. Additionally, the formulation devised in this thesis can act as a benchmarking value to verify the optimality of the

methodology defined in [BTC20], allowing to discuss questions that the authors did not address in the paper.

All these methodologies show that the formulation steers the robot towards textured areas. In order to showcase the optimality of the solution, the interpolating function can be easily represented. Figure 5.25 depicts the development of the cost value along the trajectory. The picture shows not only how the function reaches the peak of the interpolating function, but that it attempts to remain in the maximum for as long as possible. The picture also shows the effect of the blending parameter $\lambda_{\mathcal{M}}$ on the balance between actuation-effort and feature visibility. At the limits of the value, the function collapses on either of the two extreme solutions.

The same results can be see in more details in Figure 5.26. The plot represents the feature density over the time span of the trajectory. Again, the plot highlights the blending effect of the combined cost formulation. Based on the blending parameter, the number of features can be increased. The picture also presents the behavior of the planner when introducing a feature visibility constraint with a high threshold. The constraint is only enforced in the time span between 20 and 40 seconds, and it is clear that it severely affects the solution even in the segments in which it is not enforced. The graph also shows that, for the purely feature-based cost formulation, the result approximates a step function. As the underlying spline is intrinsically smooth, it cannot accurately represent function with noncontinuous derivatives.

As the methodology has proven to be effective in the two-dimensional case, the path planning is extended to the full space of $SE(3)$. In particular, the problem is lateral motion parallel to the front wall, described by the following boundary conditions:

- $\mathbf{p}_0 = [11, -7, 5]^T\, m$, $\mathbf{p}_f = [11, -9, 5]^T\, m$
- $\mathbf{v}_0 = [0, 0, 0]^T\, \frac{m}{s}$, $\mathbf{v}_f = [0, 0, 0]^T\, \frac{m}{s}$
- $\mathbf{a}_0 = [0, 0, 0]^T\, \frac{m}{s^2}$ $\mathbf{a}_f = [0, 0, 0]^T\, \frac{m}{s^2}$
- $\mathbf{j}_0 = [0, 0, 0]^T\, \frac{m}{s^3}$, $\mathbf{j}_f = [0, 0, 0]^T\, \frac{m}{s^3}$
- $\mathbf{R}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{R}_f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- $\boldsymbol{\omega}_0 = [0, 0, 0]^T\, \frac{rad}{s}$, $\boldsymbol{\omega}_f = [0, 0, 0]^T\, \frac{rad}{s}$
- $\dot{\boldsymbol{\omega}}_0 = [0, 0, 0]^T\, \frac{rad}{s^2}$ $\dot{\boldsymbol{\omega}}_f = [0, 0, 0]^T\, \frac{rad}{s^2}$.

The overview of the solution is depicted in Figure 5.27. The problem is solved on the downsampled map with a blending factor $\lambda_{\mathcal{M}} = 0.4$. The robot starts in a pose that

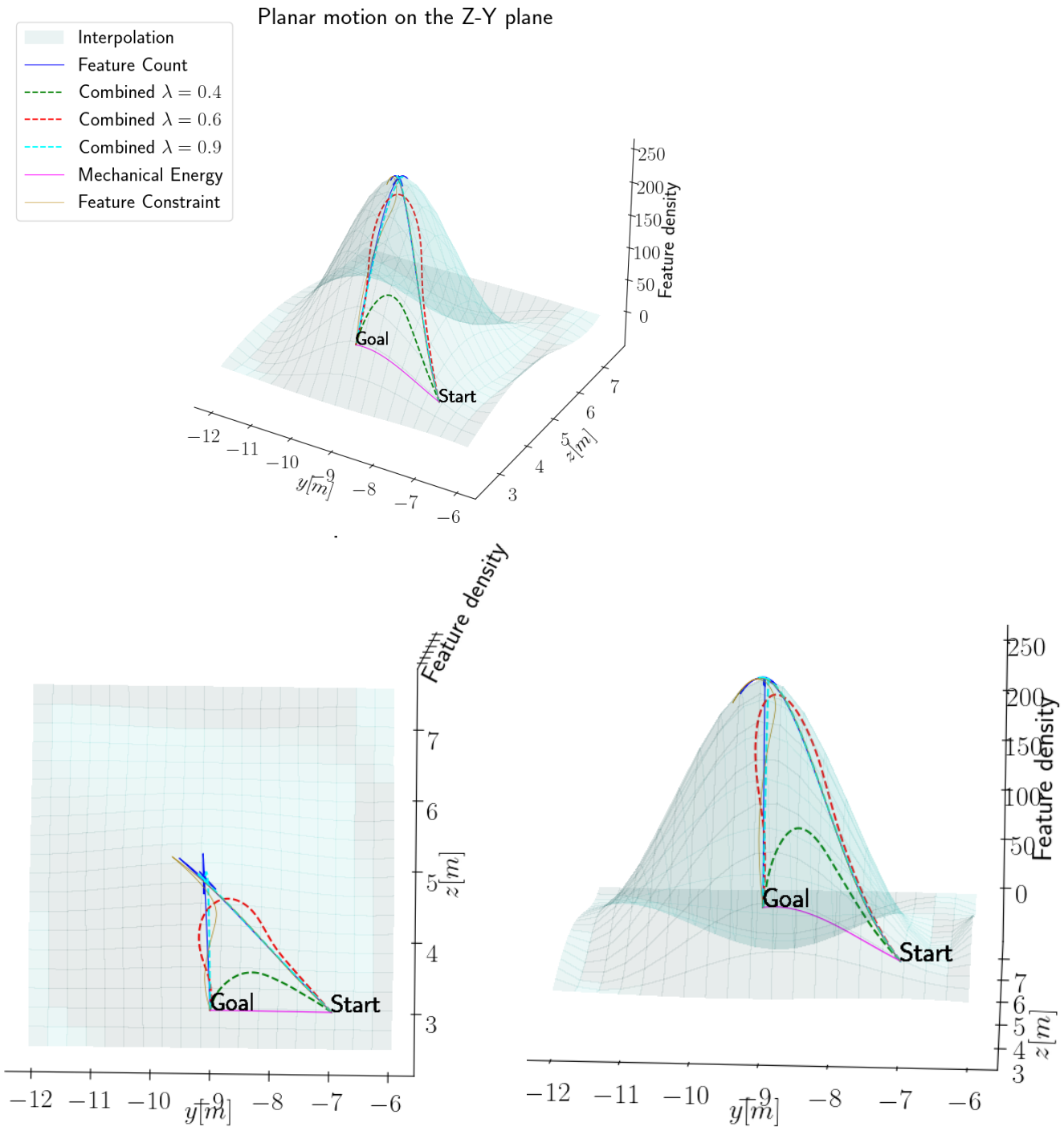


Figure 5.25: 3*D* representation of the cost value of different trajectories for the 2*D* motion case.

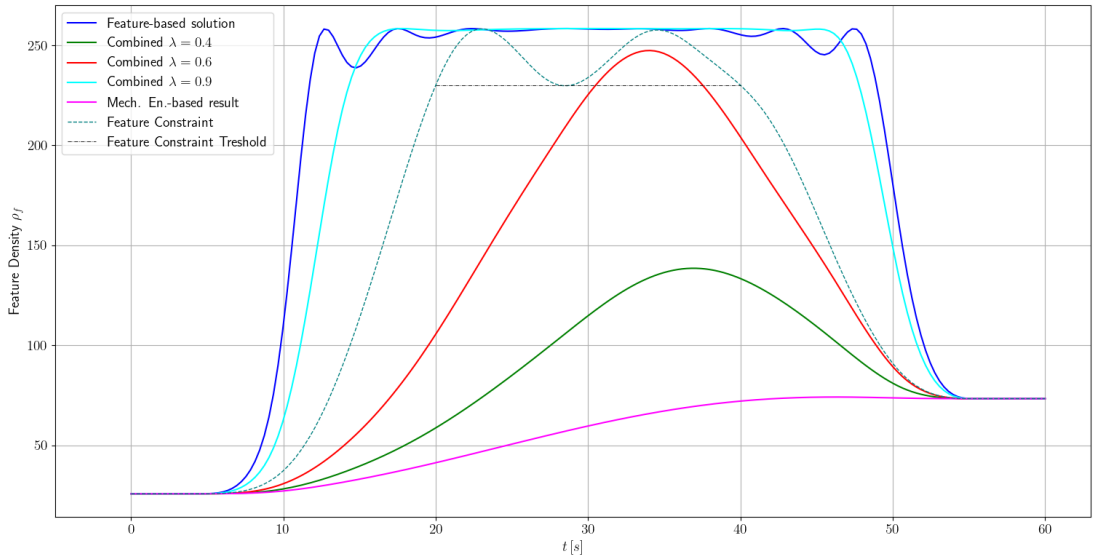


Figure 5.26: Feature density over time for different perception-aware planning problem formulation in $2D$.

does not allow to view the highest number of landmarks, thus it immediately starts turning towards the most textured area. Additionally, the robot slightly translates away from the wall, as that allows to fit a bigger portion of the wall within the field-of-view.

The detailed representation of the state and its derivatives is reported in Figure 5.28. The plots demonstrate how the robot moves smoothly towards the goal position, while slowly adjusting its position and orientation in order to view a bundle of landmarks located at $[11, -9, 4.6]\,m$. In particular, the motion the system undertakes is a reorientation combined with moving away from the area of interest, as that allows to not only increase the features in view, but also to reduce the necessary angular velocity.

The motion is an extremely relevant result obtained with the approach proposed in this thesis. As Fig. 5.28 shows, the methodology exploits fully the given environment, in fact, the obtained trajectory pushes the robot to the limits given by the constraints. In particular, the position and angular velocity constraints are active, with the latter experiencing a slight violation of $\approx 0.005\frac{rad}{s}$ (allowed by a selected threshold of $0.03\frac{rad}{s}$).

The algorithm performance is reported in Table 5.5.

From an implementation perspective, the use of a Scipy interpolation within C++ is detrimental for the performance, as can be seen by the long computation time needed. However, it can be expected that an implementation offline the entire pipeline in C++ would lead to an approximate reduction in time by a factor of 10.

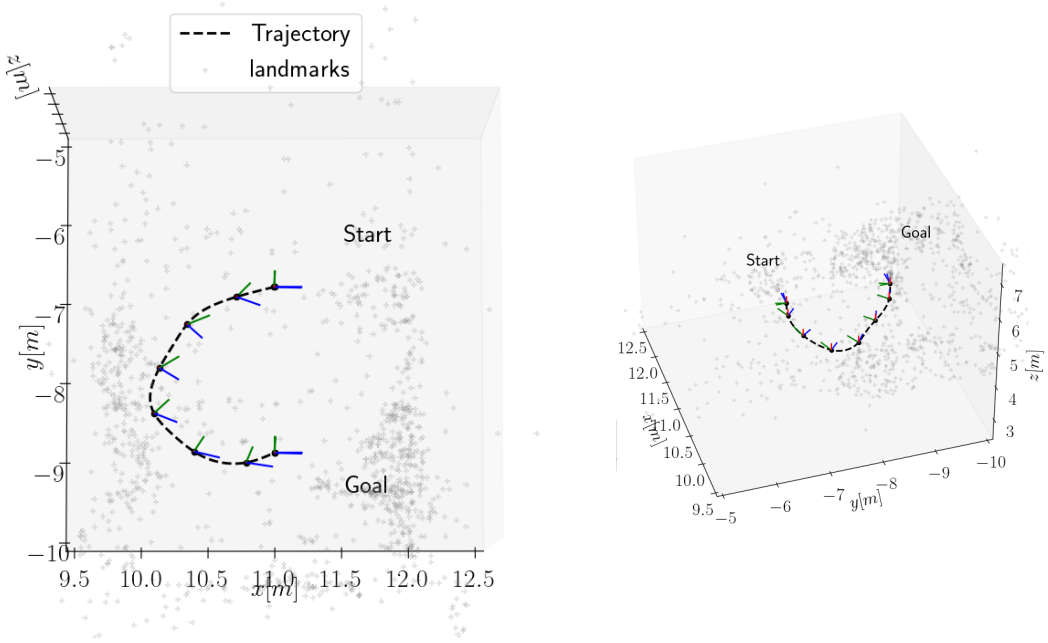
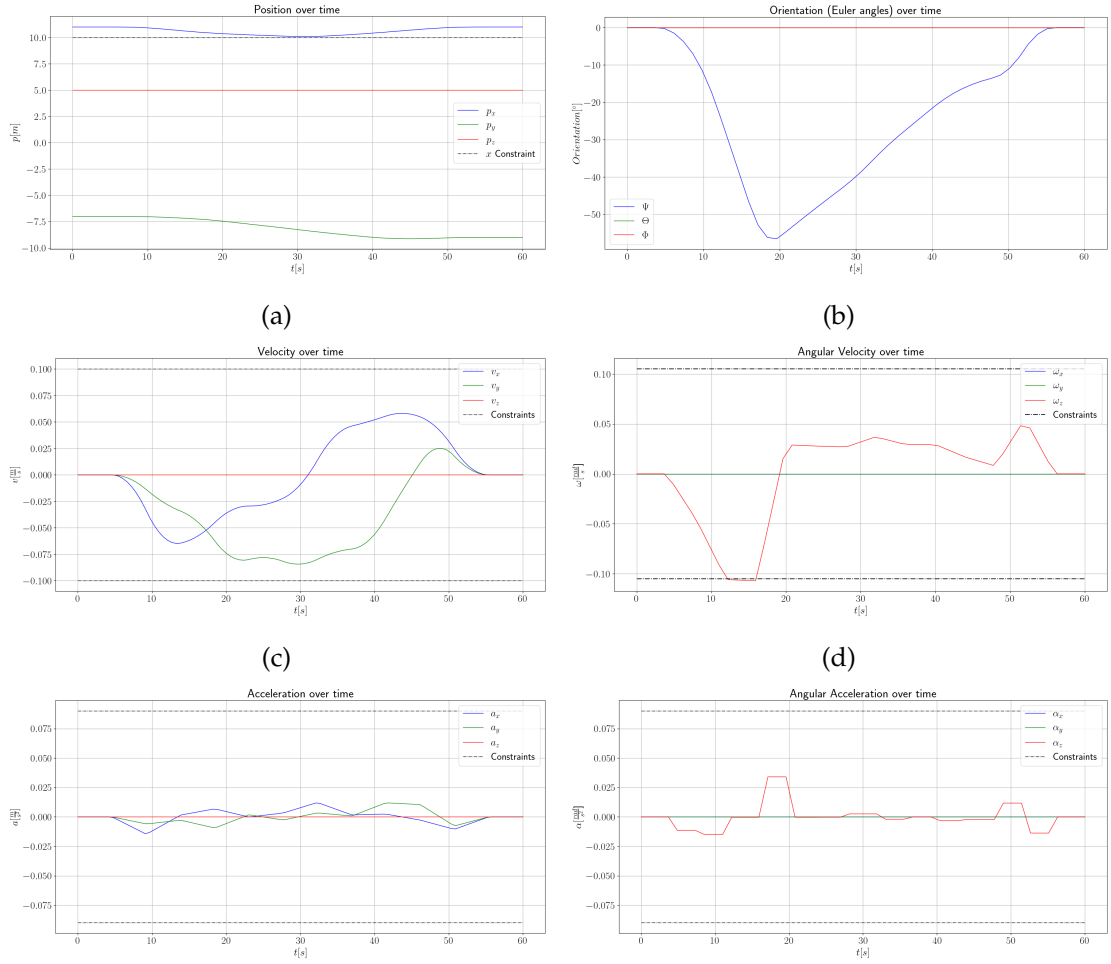


Figure 5.27: 3*D* representations of the trajectory obtained using the feature-density cost formulation, blended with mechanical energy. The trajectory is shown within the sparse map of the JEM environment.

Table 5.5: Performance of path planner with the feature-density cost function.

Case	Cost	Implementation language	Time $[s]$
Feature-density function	0.7677815	Python embedded in C++	6.53994



(a) (b) (c) (d) (e) (f)

Figure 5.28: State and derivatives of the 3D trajectory obtained using the feature-density cost formulation, blended with mechanical energy. The plots additionally report the constraints imposed within the OCP (black dashed-dotted).

Relaxed visibility function in $SE(3)$ on a downsampled map

One of the contributions of this work has been the reformulation of the relaxed visibility function found in the literature, into the case of motion on $SE(3)$. The detailed explanation of the method and implementation within the context of kinodynamic path planning is denoted in Section 4.2.3.

In order to draw a comparison, the function is tested on the same downsampled map of the JEM. The cost is formulated using the mechanical energy as smoothing component, while the perception cost comprises of both the feature visibility function and the co-visibility term, as derived in 4.32.

The resulting trajectory is shown in Figures 5.29 and 5.30.

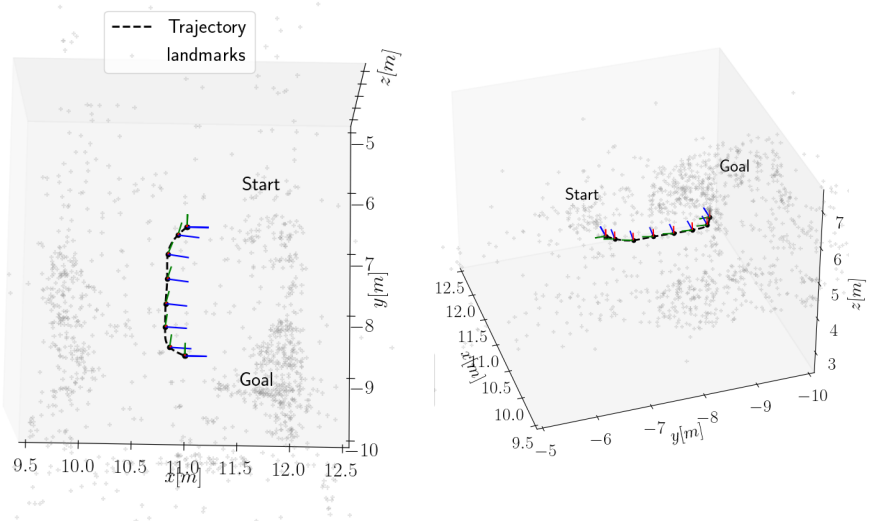


Figure 5.29: 3*D* representations of the trajectory obtained using the relaxed visibility cost formulation, blended with mechanical energy. The trajectory is shown within the sparse map of the JEM environment.

The solution behaves similarly to the one shown previously, albeit the planned motion is requires less translation and rotation. The trajectory appears less smooth, while not showing as much redirection towards the bundle of most landmarks.

Finding the correct tuning parameters is non-trivial and requires trial and error, in this case the weights used are $\lambda_l = 10^{-1}$ and $\lambda_v = 10^{+1}$. The performance of the algorithm is reported in Table 5.6.

This algorithm's implementation in C++ is approximately ten times faster than the feature-density function. The thus they are comparable in numerical performance. However, the cost formulation presented in [BTC20] is not numerically efficient, as it

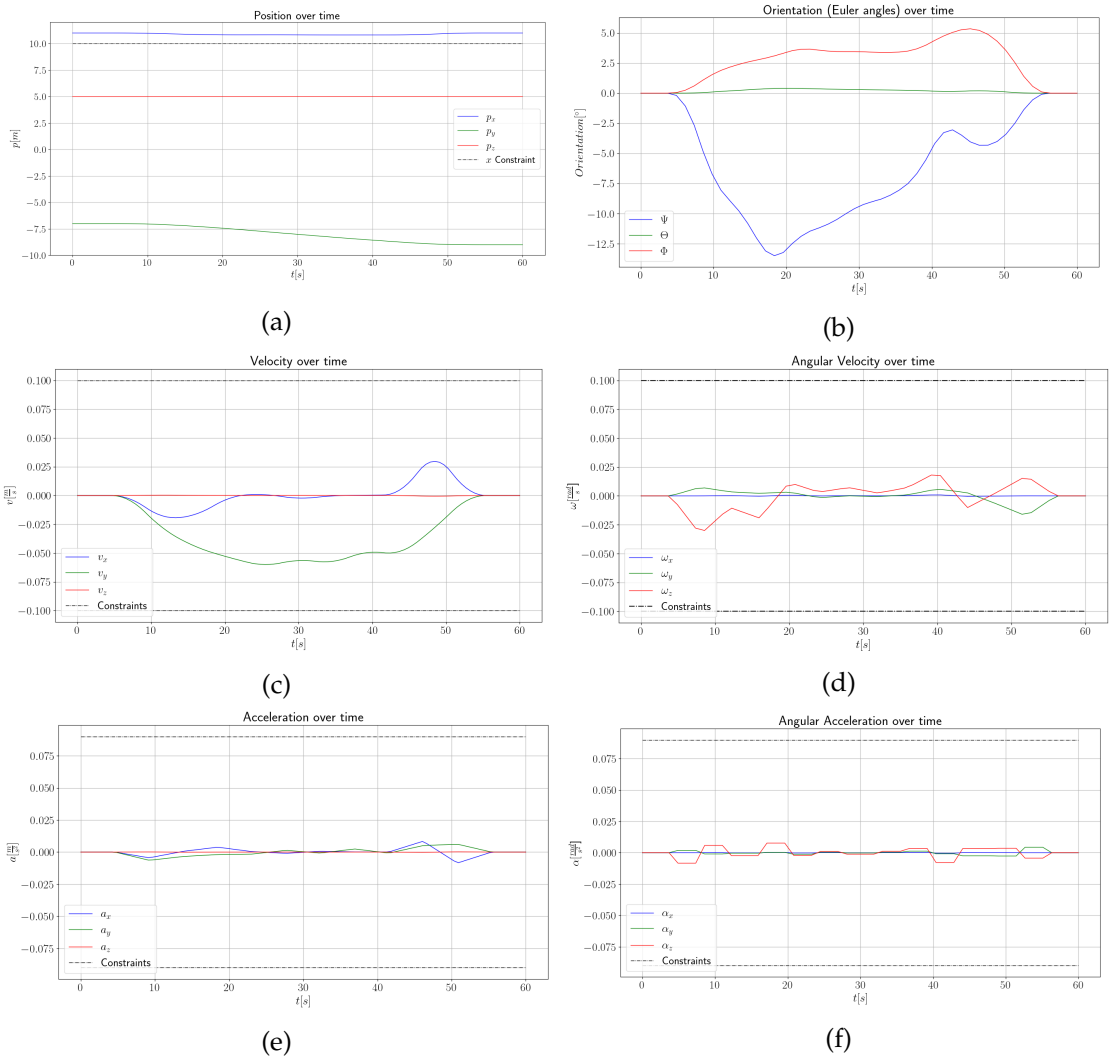


Figure 5.30: State and derivatives of the $3D$ trajectory obtained using the relaxed visibility cost formulation, blended with mechanical energy. The plots additionally report the constraints imposed within the OCP.

can be seen the cost takes negative values, which could jeopardize the quality of the solution in numerical solvers.

Table 5.6: Performance of the path planner with the relaxed visibility cost function on the downsampled JEM map.

Case	Cost	Implementation language	Time $[s]$
Differentiable visibility function, $f = 100$	-19.9258	C++	0.6174

Relaxed visibility function in $SE(3)$ on a full map

Finally, the same test case just described for the differentiable visibility function from Eq. 4.32, is deployed on the full JEM map.

Figure 5.31 depicts the overview of the trajectory. The number of features in view comes from a downsampled map for clarity of visualization. Since the number of landmarks is so high, they would obstruct the view completely.

The resulting path presents the behavior that has been common in all cases of $3D$ perception-aware planning discussed in this work. The robot travels backwards from the most textured areas, to include as many features as possible within the field of view.

Both Fig. 5.31 and 5.32 show that the resulting path has resemblances to the two cases described above with respect to position and attitude. However, in this case the resulting plots portray a very erratic behavior with strong accelerations and changes in velocities.

In particular it can be seen that many of the constraints are being violated. This effect can be associated to numerical issues with the cost formulation described by [BTC20]. Specifically, referring to Eq. (4.32), the cost formulation depends on the negative sum of a score associated with each landmark at each via point. In cases such as the one presented here, the environment is a small and enclosed space, as described in Section 2.1. Even motions of small entity can result in large changes in cost, which leads to high gradient values (up to $\approx 10^7$). As detailed in Section 3.2.2, the gradient information reflects directly on the numerical optimization method by influencing the Hessian matrix. The mathematical properties of the Hessian matrix of the Lagrangian, define the convexity and convergence of the problem, as well as constraint satisfaction [NW06].

Moreover, the large number of landmarks in the map, directly influence the number of operations to perform at any iteration of the solution process. Referring to Section 4.2.3,

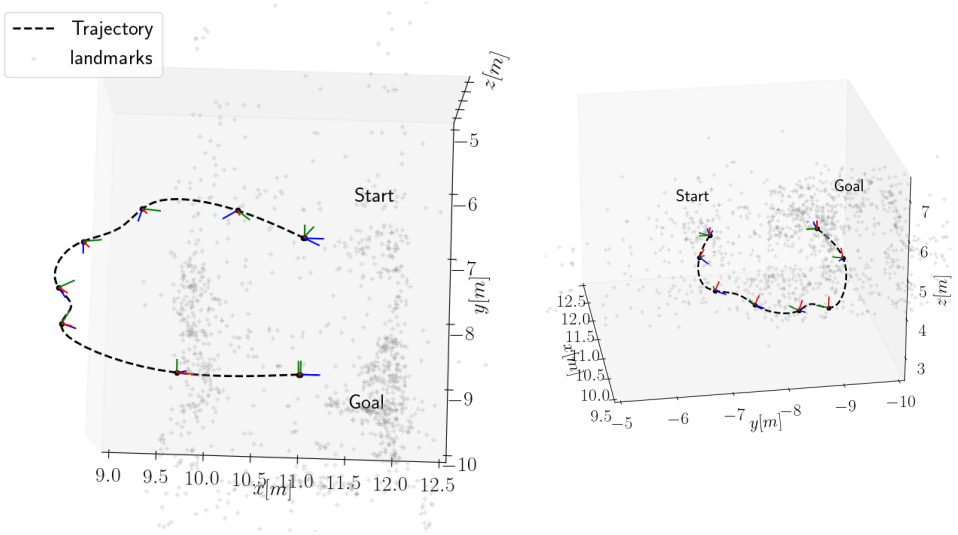


Figure 5.31: 3*D* representations of the trajectory obtained using the relaxed visibility cost formulation, blended with mechanical energy. The trajectory is shown within the sparse map of the JEM environment. For clarity only a limited amount of landmarks are shown.

the computation of the cost function at any given pose requires numerous computations, which scale poorly with an increasing number of landmarks.

This "curse of dimensionality" is clearly showcased in Table 5.7, which reports both the cost value and the extremely long computation time required. With an increase in landmarks of a factor of 100, the computational time has increased by ≈ 4000 times.

Table 5.7: Performance of the path planner with the relaxed visibility cost function on the downsampled JEM map.

Case	Cost	Implementation language	Time $[s]$
Differentiable visibility function, $f = 1$	-3.14×10^4	C++	2463.62

5.4 Discussion

Kinodynamic path planning: The method improves on sampling-based path planning methodologies by extending its formulation to the sampling and planning of orientations, leveraging incremental grid structures and GBO on manifolds. The RRT*-GBO

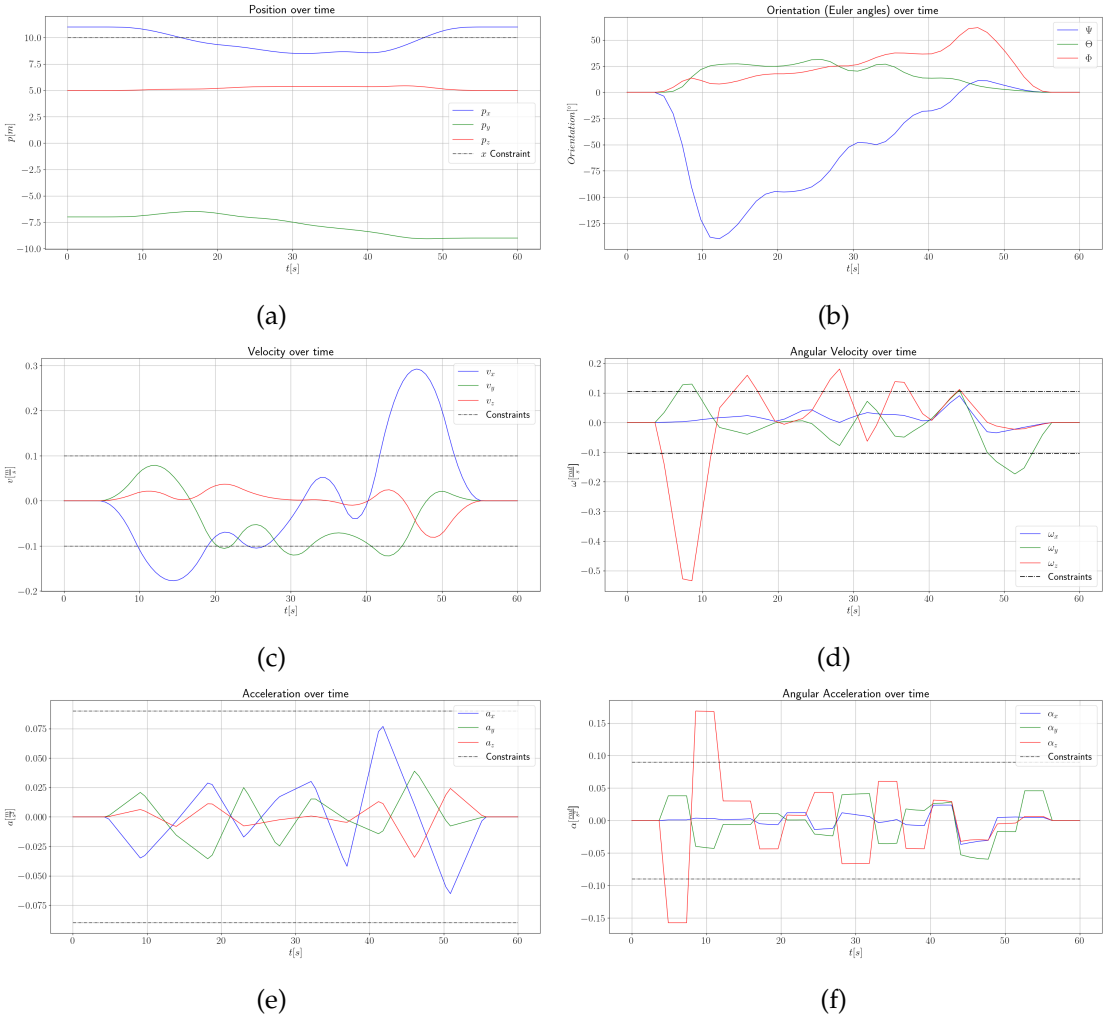


Figure 5.32: State and derivatives of the 3*D* trajectory obtained using the relaxed visibility cost formulation, blended with mechanical energy in the full map case. The plots additionally report the constraints imposed within the OCP.

method was modified from its original formulation by introducing an improved edge construction through propagation of motion. The modified edge formulation allows to limit the sampling manifold to the configuration space, thus alleviating the computational load related to sampling on a 12-dimensional state space, by only sampling a 6-dimensional one on $SE(3)$. In the context of sampling-based path planning, avoiding to sample the complete state-space could lead to losing the probabilistic completeness of the RRT* algorithm. The issue, however, has not been addressed in this thesis.

A sampling technique on $SO(3)$ has been proposed based on state-of-the-art approaches. Additionally, the grid and the resulting samples have been shown, in order to qualitatively demonstrate the mathematical properties described in the literature, but never showed visually before.

The kinodynamic path planning algorithm has demonstrated to deliver the expected solutions for tridimensional reorientation maneuvers inspired by the TumbleDock scenario, which entails obtaining a kinodynamic path that reaches a spinning object with boundary conditions that allow to synchronize the motion of target and chaser Astrobees. Additionally, the algorithm has proven to achieve the necessary flexibility to include the inverse kinematics constraint of a robotic manipulator. This extension allows, with the proper assumptions, to simulate the Astrobee dynamics using a robotic manipulator on ground. A demonstrative example solution has been obtained. The path has shown to allow compliance with the joint limits constraint for a simple reorientation maneuver within the OOS-SIM simulation environment at DLR.

While no guarantee of global optimality is available, the RRT*-GBO has shown to consistently deliver the solution with the lowest cost with respect to the generated tree.

Perception-aware path planning: In the second part of the thesis, an extended analysis of perception-aware path planning using GBO has been conducted. First a novel approach for perception-aware path planning in $SE(3)$ has been devised. The resulting formulation leverages a feature-density function to embed the environment's information in a continuous and two-times differentiable function. Differently from current state-of-the-art methods, the approach is not solely reliant on full environment descriptions, but rather allows to lower the complexity of the problem by exploiting landmarks' density measures rather than accurate $3D$ positions. By randomly downsampling an initially given map, the algorithm can deliver a density function which retains the information content without the need to query large amount of landmark data online. The result is achieved by performing a multivariate interpolation of a discrete visibility function obtained from a grid on $SE(3)$. This step allows for high degree of flexibility, as the discretization of the grid, the discrete visibility function and the interpolating curves can all be modified and adapted arbitrarily.

First the interpolation results have been showcased and demonstrated by plotting the feature density of the JEM walls. Afterwards, the function has been tested by

constructing different OCPs and applying them to a $2D$ scenario within the ISS.

The results have demonstrated that using a feature-density cost formulation delivers trajectories which maximize the number of features as well as the duration of this visibility condition.

While this solution demonstrates the functionality of the path planning algorithm obtained, the obtained path is highly inefficient and impractical from the standpoint of the actuation. Therefore, a blended cost function has been proposed and employed. The cost, comprising of a visual term (feature-density) and a smoothing term (mechanical energy), has proven to deliver smoother trajectories that still steer towards highly textured areas. Moreover, the obtained cost function allows to appropriately tune a blending parameter to favour either of the two cost components, adding a degree of flexibility and tuning to the resulting trajectory planner.

Finally, since localization algorithms require a limited number of features to remain in view, rather than purely maximizing the number of visible features, it is possible to specify a threshold of density in the form of a nonlinear constraint. This approach fits seamlessly within the GBO approach and has demonstrated to deliver feasible trajectories. Using a constraint allows the algorithm to autonomously discern between optimizing for mechanical energy of feature visibility based on the current conditions of the robot. Unfortunately, the method requires the knowledge to define a suitable threshold such that the trajectory cannot become unfeasible due to constraint violation.

To complete the analysis of perception-aware path planners, the state-of-the-art methods defined in the literature for $\mathbf{R}^3$ have been extended to work in the space of rigid body motions. In particular, the method using a relaxed visibility function has been chosen as a suitable candidate to draw a significant comparison.

Both the feature-density function and the relaxed visibility functions have been deployed in $3D$ scenarios using the downsampled JEM map. Both algorithm have shown very similar results in both the $3D$ and $2D$ cases. In $3D$, however, the feature-based methodology has demonstrated to plan trajectories that more efficiently leverage the available environment and constraints.

Attempting to apply the relaxed visibility methodology onto the full map, however, has delivered largely unfeasible results which violated many of the constraints. Thus, the effectiveness of the methodology requires a more thorough analysis. The result acts as an indicator of a possible shortcoming of the state-of-the-art methodology, which might not be well suited to be applied in cases where the number of landmarks is high.

6 Conclusions

In this thesis, a novel methodology for kinodynamic and perception-aware path planning in $SE(3)$ is proposed.

Kinodynamic path planning: The RRT*-GBO method was successfully expanded to be applied to cases of path planning on $SE(3)$. The increased state-space to sample requires the implementation of an improved edge construction methodology, which has successfully delivered smooth edges even across tree nodes, while reducing the sampling to the configuration space. The modification of the underlying RRT* approach, however, might compromise the optimality of the algorithm, although these properties have not been investigated in this work.

The algorithm leverages the mathematical structure of the manifold $SE(3)$ to sample it and perform GBO using B-splines.

The algorithm has been deployed on simulated scenarios using Astrobee robots as well as a robotic manipulator on ground. Both cases have demonstrated the algorithm's effectiveness in delivering optimal smooth trajectories in a limited number of iterations for highly constrained problems.

Perception-aware path planning: In the second part of the thesis, perception-aware path planning has been addressed.

First a novel approach for perception-aware path planning in $SE(3)$ has been devised. By using smooth and two-times differentiable interpolating B-splines, a feature-density function is obtained from downsampled landmarks map of the ISS.

The approach showcased its functionality in simplified cases on $\mathbb{R}^2$ as well as in $SE(3)$. The results have demonstrated that using a feature-density cost formulation delivers trajectories which maximize the number of features as well as the duration of this visibility condition.

In two-dimensional cases, the method has shown to obtain the same results as the relaxed visibility method described in [BTC20; Mur+19].

To alleviate the issue of obtaining impractical trajectories, two approaches were proposed: a blended cost combining mechanical energy and feature-density, and a feature constraint. Both the formulations deliver smooth trajectories that steer the system towards more textured areas.

Finally, the blended cost formulation was chosen to be employed in the context of

kinodynamic perception-aware planning on $SE(3)$. The method has delivered optimal results that leverage the dynamic properties of the Astrobee platform to its maximum within the constraints. The feature-density blended cost formulation proposed in this thesis has outperformed the extension to $SE(3)$ of the state-of-the-art methodology on a downsampled map of the JEM.

To complete the investigation, the extended relaxed visibility cost formulation was tested with a full map, demonstrating that the method's performance drastically degrades when the visibility of many landmarks has to be computed online.

6.1 Future work

Due to time limitations within this thesis, a series of interesting research questions have been left unanswered, while many others have arisen.

Resolution of localization failures: it should be investigated if the proposed perception-aware methodology translates to a real improvement of the localization for the astrobee case. Constructing a sparse map of the OOS-SIM environment and implementing locally a VIL pipeline would allow to gather significant results.

Perception-aware path planning: the feature-density path planning algorithm should be integrated within the RRT*-GBO to verify the quality of the solution for non-convex problems. The extension to $SE(3)$ of the relaxed visibility function should be further tested and analyzed, to verify its performance in highly textured maps. In order to account for obstruction of features, introducing ray-tracing techniques during the visibility grid construction could improve the functionality of the algorithm. Moreover, implementing the interpolation routine in C++ rather than Python should deliver a substantial increment in computational speed. Finally, by restricting the interpolation to confined areas, the online feasibility of the feature-density method could be analyzed.

Machine learning: With the rising use of supervised and unsupervised learning approaches, the problem of path planning with active perception could be tackled using alternative cost functions, system modeling or, alternatively, entirely learned through reinforcement learning.

Abbreviations

DLR Deutsches Zentrum für Luft- und Raumfahrt

MPC Model Predictive Control

ISS International Space Station

JEM Japanese Experiment Module

USL U.S. Laboratory Module

ROS Robot Operating System

LN Links and Nodes

OOS-SIM On-Orbit Servicing Simulator

FAM Force Allocation Module

SLAM Simulataneous Localization And Mapping

RRT Rapidly Exploring Random Trees

GBO Gradient based Optimization

EKF Extended Kalman Filter

OCP Optimal Control Problem

VIO Visual-Inertial Odometry

LTI Linear Time Invariant

PRM Probabilistic Roadmaps

BIT Batch Informed Trees

NLP Nonlinear Programming

SQP Sequential Quadratic Progamming

QP Quadratic Progamming

SLSQP Sequential Least Squares Programming

AVP Admissible Velocity Propagation

VI-SLAM Visual-Inertial SLAM

VIL Visual-Inertial Localization

POMDP Partially Observable Markov Decision Process

MAV Micro Aerial Vehicle

MAP Maximum A Posteriori

List of Figures

List of Tables

Bibliography

[ACA14] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato. "FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements." In: *The International Journal of Robotics Research* 33.2 (2014), pp. 268–304.

[Ach+14] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart. "Motion- and Uncertainty-aware Path Planning for Micro Aerial Vehicles." In: *Journal of Field Robotics* 31.4 (2014), pp. 676–698. DOI: https://doi.org/10.1002/rob.21522. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21522.

[AEO20] K. Albee, M. Ekal, and C. Oestreich. "A brief guide to Astrobee's flight software." In: (2020).

[Alb+21] K. Albee, C. Oestreich, C. Specht, A. Terán Espinoza, J. Todd, I. Hokaj, R. Lampariello, and R. Linares. "A Robust Observation, Planning, and Control Pipeline for Autonomous Rendezvous with Tumbling Targets." In: *Frontiers in Robotics and AI* 8 (2021). ISSN: 2296-9144. DOI: 10.3389/frobt.2021.641338.

[Alb+22] K. Albee, C. Specht, H. Mishra, C. Oestreich, B. Brunner, R. Lampariello, and R. Linares. "Autonomous Rendezvous with an Uncertain, Uncooperative Tumbling Target: The TumbleDock Flight Experiments." In: (2022).

[Art+15] J. Artigas, M. De Stefano, W. Rackl, R. Lampariello, B. Brunner, W. Bertleff, R. Burger, O. Porges, A. Giordano, C. Borst, and A. Albu-Schaeffer. "The OOS-SIM: An on-ground simulation facility for on-orbit servicing robotic operations." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2854–2860. DOI: 10.1109/ICRA.2015.7139588.

[Ata15] N. A. Atanasov. *Active information acquisition with mobile robots*. University of Pennsylvania, 2015.

[AWB88] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. "Active vision." In: *International journal of computer vision* 1 (1988), pp. 333–356.

[BD18] D. Brescianini and R. D'Andrea. "Computationally Efficient Trajectory Generation for Fully Actuated Multirotor Vehicles." In: *IEEE Transactions on Robotics* 34.3 (2018), pp. 555–571. DOI: 10.1109/TRO.2018.2813373.

[Bel54] R. E. Bellman. *The Theory of Dynamic Programming*. Santa Monica, CA: RAND Corporation, 1954.

[Bel66] R. Bellman. "Dynamic Programming." In: *Science* 153.3731 (1966), pp. 34–37. DOI: 10.1126/science.153.3731.34. eprint: https://www.science.org/doi/pdf/10.1126/science.153.3731.34.

[BM08] L. Biagiotti and C. Melchiorri. *Trajectory Planning for Automatic Machines and Robots*. 1st. Springer Publishing Company, Incorporated, 2008. ISBN: 3540856285.

[Boo77] C. de Boor. "Package for Calculating with B-Splines." In: *SIAM Journal on Numerical Analysis* 14.3 (1977), pp. 441–472. ISSN: 00361429.

[BR11] A. Bry and N. Roy. "Rapidly-exploring random belief trees for motion planning under uncertainty." In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 723–730.

[BTC20] L. Bartolomei, L. Teixeira, and M. Chli. "Perception-aware Path Planning for UAVs using Semantic Segmentation." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5808–5815. DOI: 10.1109/IROS45743.2020.9341347.

[Car+14] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert. "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors." In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 4290–4297. DOI: 10.1109/ICRA.2014.6907483.

[Cha21] A. Chaudhuri. "B-Splines." In: *CoRR* abs/2108.06617 (2021). arXiv: 2108.06617.

[Che99] M. Cherif. "Kinodynamic motion planning for all-terrain wheeled vehicles." In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 1. 1999, 317–322 vol.1. DOI: 10.1109/ROBOT.1999.769998.

[Col+16a] B. Coltin, J. Fusco, Z. Moratto, O. Alexandrov, and R. Nakamura. "Localization from visual landmarks on a free-flying robot." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 4377–4382.

[Col+16b] B. Coltin, J. Fusco, Z. Moratto, O. Alexandrov, and R. Nakamura. "Localization from visual landmarks on a free-flying robot." In: (2016).

[Cos+16] G. Costante, C. Forster, J. A. Delmerico, P. Valigi, and D. Scaramuzza. "Perception-aware Path Planning." In: *CoRR* (2016).

[Cou15] E. Coumans. "Bullet physics simulation." In: *ACM SIGGRAPH 2015 Courses* (2015).

[CPN14] S. Caron, Q.-C. Pham, and Y. Nakamura. "Completeness of randomized kinodynamic planners with state-based steering." In: *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 5818–5823.

[DC22] F. Dellaert and G. Contributors. *borglab/gtsam*. Version 4.2a8. May 2022. DOI: 10.5281/zenodo.5794541.

[Dij59] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs." In: *Numerische Mathematik* (1959).

[DK17a] F. Dellaert and M. Kaess. *Factor Graphs for Robot Perception*. Now Publishers Inc., Aug. 2017.

[DK17b] F. Dellaert and M. Kaess. *Factor Graphs for Robot Perception*. Foundations and Trends in Robotics, Vol. 6, 2017.

[DM02] A. J. Davison and D. W. Murray. "Simultaneous localization and map-building using active vision." In: *IEEE transactions on pattern analysis and machine intelligence* 24.7 (2002), pp. 865–880.

[DS87] P. Diaconis and M. Shahshahani. "The subgroup algorithm for generating uniform random variables." In: *Probability in the engineering and informational sciences* 1.1 (1987), pp. 15–32.

[For+15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. *IMU preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation*. 2015.

[Gon+16] D. González, J. Pérez, V. Milanés, and F. Nashashibi. "A Review of Motion Planning Techniques for Automated Vehicles." In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2016), pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841.

[Gor+05] K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. "HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere." In: *The Astrophysical Journal* 622.2 (2005), p. 759.

[GSB14] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. "BIT*: Batch Informed Trees for Optimal Sampling-based Planning via Dynamic Programming on Implicit Random Geometric Graphs." In: *CoRR* abs/1405.5848 (2014). arXiv: 1405.5848.

[Hal00] B. C. Hall. *An Elementary Introduction to Groups and Representations*. 2000. arXiv: math-ph/0005032 [math-ph].

[Har+13] R. Hartley, J. Trumpf, Y. Dai, and H. Li. "Rotation averaging." In: *International journal of computer vision* 103 (2013), pp. 267–305.

[HNR68] P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[HZ03] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[Joh07] S. G. Johnson. *The NLopt nonlinear-optimization package*. https://github.com/stevengj/nlopt. 2007.

[Kav+96] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580. DOI: 10.1109/70.508439.

[KF11] S. Karaman and E. Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning." In: *CoRR* abs/1105.1186 (2011). arXiv: 1105.1186.

[KLC98] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. "Planning and acting in partially observable stochastic domains." In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[Kra94] D. Kraft. "Algorithm 733: TOMP–Fortran modules for optimal control calculations." In: *ACM Transactions on Mathematical Software* 20 (1994), pp. 262–281. DOI: 10.1145/192115.192124.

[Kuf04] J. Kuffner. "Effective sampling and distance metrics for 3D rigid body path planning." In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 4. 2004, 3993–3998 Vol.4. DOI: 10.1109/ROBOT.2004.1308895.

[Lam+11] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters. "Trajectory planning for optimal robot catching in real-time." In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3719–3726. DOI: 10.1109/ICRA.2011.5980114.

[Lam+18] R. Lampariello, H. Mishra, N. Oumer, P. Schmidt, M. De Stefano, and A. Albu-Schäffer. "Tracking Control for the Grasping of a Tumbling Satellite With a Free-Floating Robot." In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3638–3645. DOI: 10.1109/LRA.2018.2855799.

[LBL04] S. M. LaValle, M. S. Branicky, and S. R. Lindemann. "On the Relationship between Classical Grid Search and Probabilistic Roadmaps." In: *The International Journal of Robotics Research* 23.7-8 (2004), pp. 673–692. DOI: 10.1177/0278364904045481. eprint: https://doi.org/10.1177/0278364904045481.

[Leu+15] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. "Keyframe-based visual–inertial odometry using nonlinear optimization." In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334. DOI: 10.1177/0278364914554813. eprint: https://doi.org/10.1177/0278364914554813.

[LJ01] S. M. LaValle and J. James J. Kuffner. "Randomized Kinodynamic Planning." In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400. DOI: 10.1177/02783640122067453.

[LP17] K. M. Lynch and F. C. Park. *Modern Robotics: Mechanics, Planning, and Control*. 1st. USA: Cambridge University Press, 2017. ISBN: 1107156300.

[MK11] D. Mellinger and V. R. Kumar. "Minimum snap trajectory generation and control for quadrotors." In: *2011 IEEE International Conference on Robotics and Automation* (2011), pp. 2520–2525.

[MLS17] R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[Mu+16] B. Mu, M. Giamou, L. Paull, A.-a. Agha-mohammadi, J. Leonard, and J. How. "Information-based active SLAM via topological feature graphs." In: *2016 IEEE 55th Conference on decision and control (Cdc)*. IEEE. 2016, pp. 5583–5590.

[Mur+19] V. Murali, I. Spasojevic, W. Guerra, and S. Karaman. "Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness." In: *2019 American Control Conference (ACC)*. 2019, pp. 3936–3943. DOI: 10.23919/ACC.2019.8814697.

[NAS] NASA. *NASA Astrobee Robot Software*. https://github.com/nasa/astrobee.

[NM15] M. Neubauer and A. Müller. "Smooth orientation path planning with quaternions using B-splines." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 2087–2092. DOI: 10.1109/IROS.2015.7353654.

[NP16] H. Nguyen and Q.-C. Pham. "Time-Optimal Path Parameterization of Rigid-Body Motions: Applications to Spacecraft Reorientation." In: *Journal of Guidance, Control, and Dynamics* 39.7 (2016), pp. 1667–1671. DOI: 10.2514/1.G001600. eprint: https://doi.org/10.2514/1.G001600.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2e. New York, NY, USA: Springer, 2006.

[Ole+16] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran. "Continuous-time trajectory optimization for online UAV replanning." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 5332–5339. DOI: 10.1109/IROS.2016.7759784.

[PBP02] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Jan. 2002. ISBN: 978-3-642-07842-2. DOI: 10.1007/978-3-662-04919-8.

[PCN13] Q.-C. Pham, S. Caron, and Y. Nakamura. "Kinodynamic Planning in the Configuration Space via Velocity Interval Propagation." In: 2013.

[PR09] S. Prentice and N. Roy. "The belief roadmap: Efficient planning in belief space by factoring the covariance." In: *The International Journal of Robotics Research* 28.11-12 (2009), pp. 1448–1465.

[RBR16] C. Richter, A. Bry, and N. Roy. "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments." In: *Robotics Research: The 16th International Symposium ISRR*. Springer. 2016, pp. 649–666.

[Sch07] L. Schumaker. *Spline functions: basic theory*. Cambridge university press, 2007.

[SD91] Z. Shiller and S. Dubowsky. "On computing the global time-optimal motions of robotic manipulators in the presence of obstacles." In: *IEEE Transactions on Robotics and Automation* 7.6 (1991), pp. 785–797. DOI: 10.1109/70.105387.

[SL14] S. Stoneman and R. Lampariello. "Embedding nonlinear optimization in RRT* for optimal kinodynamic planning." In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 3737–3744. DOI: 10.1109/CDC.2014.7039971.

[SL16] S. Stoneman and R. Lampariello. "A Nonlinear Optimization Method to Provide Real-Time Feasible Reference Trajectories to Approach a Tumbling Target Satellite." In: *ISAIRAS 2016*. Vol. 13. 2016.

[Som+20] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers. "Efficient Derivative Computation for Cumulative B-Splines on Lie Groups." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[Sou+22] R. Soussan, V. Kumar, B. Coltin, and T. Smith. "AstroLoc: An Efficient and Robust Localizer for a Free-flying Robot." In: (2022), pp. 4106–4112. DOI: 10.1109/ICRA46639.2022.9811919.

[SZ20] D. Scaramuzza and Z. Zhang. "Aerial Robots, Visual-Inertial Odometry of." In: *Encyclopedia of Robotics*. Ed. by M. H. Ang, O. Khatib, and B. Siciliano. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, pp. 1–9. ISBN: 978-3-642-41610-1. DOI: 10.1007/978-3-642-41610-1_71-1.

[Thr02] S. Thrun. "Probabilistic robotics." In: *Communications of the ACM* 45.3 (2002), pp. 52–57.

[Use+17] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers. "Real-Time Trajectory Replanning for MAVs using Uniform B-splines and 3D Circular Buffer." In: *CoRR* abs/1703.01416 (2017). arXiv: 1703.01416.

[Vir+20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[Wat+18] M. Watterson, S. Liu, K. Sun, T. Smith, and V. Kumar. "Trajectory Optimization On Manifolds with Applications to $SO(3)$ and $\boldsymbol{R}^3 \times S^2$." In: *Proc. Rob. Sci. Systems*. 2018.

[Wat+20] M. Watterson, S. Liu, K. Sun, T. Smith, and V. Kumar. "Trajectory optimization on manifolds with applications to quadrotor systems." In: *The International Journal of Robotics Research* 39.2-3 (2020), pp. 303–320. DOI: 10.1177/0278364919891775. eprint: https://doi.org/10.1177/0278364919891775.

[WB13] D. J. Webb and J. van den Berg. "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics." In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 5054–5061. DOI: 10.1109/ICRA.2013.6631299.

[WSK16] M. Watterson, T. Smith, and V. Kumar. "Smooth trajectory generation on SE (3) for a free flying space robot." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 5459–5466.

[Wu+22] X. Wu, S. Chen, K. Sreenath, and M. W. Mueller. "Perception-Aware Receding Horizon Trajectory Planning for Multicopters With Visual-Inertial Odometry." In: *IEEE Access* 10 (2022), pp. 87911–87922. DOI: 10.1109/ACCESS.2022.3200342.

[Yer+10] A. Yershova, S. Jain, S. M. LaValle, and J. C. Mitchell. "Generating Uniform Incremental Grids on SO(3) Using the Hopf Fibration." In: *The International Journal of Robotics Research* 29.7 (2010). PMID: 20607113, pp. 801–812. DOI: 10.1177/0278364909352700. eprint: https://doi.org/10.1177/0278364909352700.

[Zhi+20] T. Zhiling, B. Chen, R. Lan, and S. Li. "Vector Field Guided RRT* Based on Motion Primitives for Quadrotor Kinodynamic Planning." In: *Journal of Intelligent & Robotic Systems* (Dec. 2020). DOI: 10.1007/s10846-020-01231-y.

[Zho+19] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen. *Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight*. 2019. arXiv: 1907.01531 [cs.RO].

[Zho+21] B. Zhou, J. Pan, F. Gao, and S. Shen. "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight." In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1992–2009.

[ZKC98] M. Zefran, V. Kumar, and C. Croke. "On the generation of smooth three-dimensional rigid body motions." In: *IEEE Transactions on Robotics and Automation* 14.4 (1998), pp. 576–589. DOI: 10.1109/70.704225.

[ZS18] Z. Zhang and D. Scaramuzza. "Perception-aware receding horizon navigation for MAVs." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2534–2541.