

Bachelor Thesis

Optimization Based Motion Planning on the $SE(3)$ Manifold for an Omnidirectional Aerial Manipulator

Spring Term 2021



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Optimization Based Motion Planning on the SE(3) Manifold for an Omnidirectional Aerial Manipulator

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):

Näf

Bühlmann

First name(s):

Joshua

Franz

With my signature I confirm that

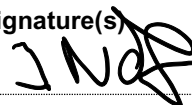
- I have committed none of the forms of plagiarism described in the '[Citation etiquette](#)' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

Olten, 17.09.2021

Signature(s)


Joshua Näf


Franz Bühlmann

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Abstract

In this bachelor thesis, a motion planner for the omnidirectional aerial manipulator *PrisMAV* is implemented. The optimization-based planner builds on the Covariant Hamiltonian Optimization for Motion Planning (CHOMP) algorithm to enable motion planning for omnidirectional micro aerial vehicles (OMAVs). CHOMP enables the computation of smooth and collision-free trajectories through the optimization of an initial (potentially infeasible) trajectory, such as a straight line path. An additional cost function on the orientation of *PrisMAV* was designed to encourage flight in energy efficient flight modes.

The planner is tested and verified in simulation under different scenarios where rapidly-exploring random trees (RRT) serve as a performance benchmark. We observe clearly longer flight periods spent in favourable orientations for trajectories produced by our planner and a noticeable decrease in trajectory length compared to RRT.

Preface

We want to express our gratitude to anyone who has supported us throughout this thesis. Special thanks go to:

- Prof. Dr. Roland Siegwart, without whom, project Griffin and this thesis would not have taken place.
- ETH Zürich and the Autonomous Systems Lab (ASL) for providing us with the necessary infrastructure and knowledge.
- Our coaches Jen Jen Chung, Julian Förster and Michael Pantic for their support and constructive feedback in all our meetings.
- The whole Griffin team who made our platform *PrisMAV* a reality.
- Our family and friends for their patience.
- Everyone who proofread our thesis.

Symbols

Symbols

ξ	Robot trajectory (mapping from time to robot configuration)
\boldsymbol{p}	Position vector in the inertial frame
$\xi_{\boldsymbol{p}}$	Trajectory of the position \boldsymbol{p}
Φ	Orientation of the MAV w.r.t. the inertial frame
\boldsymbol{R}	Rotation matrix
$\mathcal{F}(\xi)$	Cost function
\boldsymbol{K}	Finite differencing matrix
$\delta(\cdot)$	Element in the local tangent space of the specified manifold

Indices

z	z axis
-----	--------

Acronyms and Abbreviations

ASL	Autonomous Systems Lab
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
ESDF	Euclidean signed distance field
ETH	Eidgenössische Technische Hochschule
MAV	Micro Aerial Vehicle
OMAV	Omnidirectional Micro Aerial Vehicle
RRT	Rapidly-exploring Random Trees
STOMP	Stochastic Trajectory Optimization for Motion Planning

Contents

Abstract	v
Preface	vii
Symbols	ix
1 Introduction	1
1.1 Motivation	2
1.2 Goals	2
1.3 Related Work	2
1.3.1 Overview	3
2 Preliminaries	5
2.1 CHOMP: Covariant Hamiltonian Optimization for Motion Planning	5
2.2 Special Orthogonal group $SO(3)$ and Special Euclidean group $SE(3)$	5
2.2.1 Exponential and Logarithm Map	6
2.3 Optimization on Manifolds	6
2.3.1 Retraction	7
3 Method	9
3.1 Cost Functions	9
3.1.1 Smoothness Cost	9
3.1.2 Collision Cost	10
3.1.3 Orientation Cost	11
3.2 Optimization	13
3.2.1 Orthogonal projection	14
3.2.2 Gradient Descent and Update Rule	15
3.2.3 Trajectory Discretization	15
3.2.4 Cost Weight Scheduling	15
3.3 Implementation	15
4 Experiments and Results	17
4.1 Testing Environment	17
4.2 Randomly Generated Forests	18
4.2.1 Results	18
4.3 Planning in an Unstructured Environment	25
4.3.1 Results	25
4.3.2 Comparison with Randomized Forests	29
4.4 Hole in a Wall	29
4.4.1 Results	29

5	Discussion	31
5.1	Performance of <i>PrisMAV</i> planner	31
5.1.1	Randomized Forests and Unstructured Environments	31
5.1.2	Hole in a Wall	32
5.2	Parameter Choice	32
5.3	Failure Modes	33
5.3.1	Local Minima	33
5.3.2	Absence of Gradients	34
5.3.3	Excessive Smoothing	34
5.3.4	Insufficient Number of Optimization Variables	34
6	Conclusion	35
6.1	Summary of the Results	35
6.2	Outlook	35
	Bibliography	38

Chapter 1

Introduction

This bachelor thesis is an extension of focus project ‘Griffin’, during which the aerial manipulation platform *PrisMAV* (fig. 1.1) was developed. The platform is equipped with a 3 degree of freedom (DoF) parallel manipulator which was used to compensate for the positional inaccuracies of the flying platform. Due to its omnidirectional design, *PrisMAV* is able to grasp objects in any conceivable pose which was successfully demonstrated [1].



Figure 1.1: The aerial manipulator *PrisMAV*

1.1 Motivation

During Focus Project ‘Griffin’, motion planning was performed by simply linearly interpolating between the current pose of *PrisMAV* and the desired goal pose. This means that no collision checking was performed and the trajectory had to be checked manually for feasibility. This poses a problem when more complex tasks should be performed, such as navigating from a start to an end pose in a non-convex environment or performing aerial manipulation missions in a cluttered environment (for example in a warehouse).

Furthermore, the omnidirectional design of *PrisMAV* leads to an interesting motion planning challenge. Theoretically, flight in any conceivable pose is possible but certain flight modes of *PrisMAV* are more desirable as they offer higher energy efficiency.

1.2 Goals

In this thesis we want to implement a motion planner which leverages the unusual traits and design of *PrisMAV*. The planner should perform motion planning for the full 6-DoFs of the flying platform. Furthermore, the planner should produce paths which optimize the efficiency of *PrisMAV* along the trajectory. I.e. *PrisMAV* should, if possible, prefer to fly in either the tricopter or horizontal mode, as they are our preferred flight modes.

1.3 Related Work

Motion planning approaches can roughly be broken down into three main categories: graph-based algorithms, sampling-based algorithms and optimization-based algorithms. Notable examples for each are A* [2], Rapidly-exploring Random Trees (RRT)[3] and its optimal counterpart RRT* [4], and Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [5], respectively. In [6], Stochastic Trajectory Optimization for Motion Planning (STOMP) is introduced which is an extension of CHOMP that does not rely on gradient information. Instead a weighted average of noisy trajectories is used in the update step. In [7], another optimization approach, called TrajOpt, is discussed which uses a sequential convex optimization approach to break down the motion planning problem into multiple convex quadratic programs (QPs). These QPs are an approximation of the actual cost function and are easier to optimize than optimizing over the whole trajectory.

Furthermore, in the context of motion planning for MAVs, several approaches have been investigated. In [8], RRT is used with kinodynamic constraints to produce feasible paths for a blimp. Using a steer function, which steers the closest point in the tree to the newly sampled point, the dynamic constraints of the system are included in the problem formulation. In [9], a receding horizon optimal control problem is formulated in order to produce open-loop trajectories where the optimization of videographic features is used to formulate the objective function. In [10], polynomial trajectories for a quadrotor MAV are generated through keyframes (a tuple of 3D position and yaw) which are then optimized to produce minimum snap trajectories. This concept is extended in [11], where a RRT* planner is used to compute the keyframes. The advantage of polynomial trajectories is that they can represent long trajectories with a high resolution (since the polynomials are continuous functions) with a relatively low number of variables (the polynomial coefficients). In [12], polynomial trajectories are optimized subject to a smoothness cost (squared velocities along the trajectory) and a collision cost (distance to obstacles) as described in CHOMP [5]. In [13], Riemannian motion policies (RMPs) are

used as an on-line motion planning method. The planner is used to generate trajectories along a surface. RMPs generate acceleration fields in the configuration space of the robot and are therefore discretization-free. Furthermore, they can easily be extended as the resulting acceleration field is a weighted average of the acceleration fields produced by the different policies.

Although uniformly sampling orientations is straight forward, defining a probability density function on orientation which leads to the desired behaviour of preferred flight modes is less clear-cut. We chose to pursue an optimization-based approach to motion planning as this would allow us to define intuitive cost functions on the orientation of *PrisMAV*.

The mentioned works provide planning algorithms which assume the robot configuration to be in d -dimensional Euclidean space $\mathcal{C} \in \mathbb{R}^d$. This poses a problem when trying to perform optimization on the 6 DoFs of an OMAV since attitude cannot be expressed in Euclidean space without additional constraints. Therefore, the concepts discussed in this section need to be extended to allow optimization of the orientation of an OMAV.

1.3.1 Overview

The thesis is structured as follows. First, we will review the preliminaries necessary to formulate our planning algorithm in chapter 2. Then, we will discuss the motion planner in chapter 3. Validation of the planner performance is performed in chapter 4 and the results are discussed in chapter 5. Finally, in chapter 6, we conclude the thesis by summarising the results and indicating possible improvements of the planning algorithm.

Chapter 2

Preliminaries

Before we discuss the motion planner used in this thesis we first have to go over some preliminary concepts. The planning method used in this thesis is optimization-based and relies heavily on concepts which are related to CHOMP. Furthermore, we need to introduce some topological concepts in order to formulate an optimization problem on the manifold of rigid body transformations.

2.1 CHOMP: Covariant Hamiltonian Optimization for Motion Planning

CHOMP [5] is an optimization-based motion planner which generates a trajectory $\xi : [0, 1] \rightarrow \mathcal{C} \in \mathbb{R}^d$, a mapping of time to robot configuration in Euclidean space, by computing the local optimizer of the following objective function,

$$\mathcal{F}(\xi) = \mathcal{F}_{\text{col}}(\xi) + \mathcal{F}_{\text{smooth}}(\xi), \quad (2.1)$$

where $\mathcal{F}_{\text{col}}(\xi) : \mathcal{C} \rightarrow \mathbb{R}$ denotes a cost which penalizes collisions along the trajectory and $\mathcal{F}_{\text{smooth}}(\xi) : \mathcal{C} \rightarrow \mathbb{R}$ denotes a cost function which encourages smooth trajectories (for the implementation of these cost functions, please refer to [5]).

The trajectory is updated by linearizing the cost function $\mathcal{F}(\xi)$ around the current iterate ξ_i and solving the following optimization problem,

$$\xi_{i+1} = \arg \min_{\xi} \mathcal{F}(\xi_i) + (\xi - \xi_i)^\top \nabla_{\xi} \mathcal{F}(\xi_i) + \frac{\eta}{2} \|\xi - \xi_i\|_A^2, \quad (2.2)$$

with the Riemannian metric $A = K^\top K$, where K denotes the finite differencing matrix. The inclusion of the metric A means that the update will prefer trajectories that only add a minimal amount of additional velocity to the trajectory ξ [5]. Taking the gradient of eq. (2.2) and setting it to zero leads to,

$$\xi_{i+1} = \xi_i - \frac{1}{\eta} A^{-1} \nabla_{\xi} \mathcal{F}(\xi_i), \quad (2.3)$$

which we will call the *standard CHOMP update rule*.

2.2 Special Orthogonal group $SO(3)$ and Special Euclidean group $SE(3)$

To describe the pose of an OMAV, we need some way of representing its position \mathbf{p} and some way of representing its attitude Φ . Position can easily be represented

by 3D Euclidean space \mathbb{R}^3 , but things get trickier when describing the attitude of an OMAV. There are many ways of representing 3D orientation ranging from Euler angles over unit quaternions \mathbf{q} to rotation matrices \mathbf{R} (unless otherwise stated we will use $\mathbf{R} := \mathbf{R}_{BI}$, as the rotation from world frame to body frame), which have their unique up- and downsides. However, what they all have in common is that they parameterize the 3D rotation group (special orthogonal group) $SO(3)$ which can formally be expressed in terms of rotation matrices as follows:

$$SO(3) := \{ \mathbf{R} \in \mathbb{R}^{3 \times 3} \mid \det \mathbf{R} = 1, \mathbf{R}^\top \mathbf{R} = \mathbf{I} \}. \quad (2.4)$$

If parameterized by rotation matrices, $SO(3)$ is closed under multiplication but not under addition, meaning that given $\Phi_1, \Phi_2 \in SO(3)$, $\Phi_1 \circ \Phi_2 = \mathbf{R}_1 \mathbf{R}_2 \in SO(3)$ but $\Phi_1 + \Phi_2 = \mathbf{R}_1 + \mathbf{R}_2 \notin SO(3)$.

The tuple of position and attitude forms the group of rigid body transformations (special Euclidean group) $SE(3)$. Here, we define $SE(3)$ as follows:

$$SE(3) := \{ (\mathbf{p}, \Phi) \mid \mathbf{p} \in \mathbb{R}^3, \Phi \in SO(3) \}. \quad (2.5)$$

Similarly to $SO(3)$, $SE(3)$ is not closed under addition. This poses a problem since optimization methods such as gradient descent rely on addition being well defined in order to compute an update step [14].

2.2.1 Exponential and Logarithm Map

Since $SO(3)$ is a Lie group, meaning it is a group but also a differentiable manifold, it is equipped with a Lie algebra. For $SO(3)$, this Lie algebra coincides with the tangent space at the identity element. The tangent space $\mathfrak{so}(3)$ is isomorphic to \mathbb{R}^3 , which implies that addition and subtraction of elements in tangent space is well defined [14].

Using the *exponential map* and the *logarithm map*, we can map an element of the tangent space $\mathfrak{so}(3)$ to the corresponding element in $SO(3)$ and vice versa:

$$\text{Exp} : \mathbb{R}^3 \rightarrow SO(3); \quad \phi \mapsto \mathbf{I} + \frac{\sin(\|\phi\|)}{\|\phi\|} \phi^\wedge + \frac{1 - \cos(\|\phi\|)}{\|\phi\|^2} (\phi^\wedge)^2 \approx \mathbf{I} + \phi^\wedge \quad (2.6)$$

$$\text{Log} : SO(3) \rightarrow \mathbb{R}^3; \quad \mathbf{R} \mapsto \left(\frac{\varphi \cdot (\mathbf{R} - \mathbf{R}^\top)}{2 \sin(\varphi)} \right)^\vee, \text{ where } \varphi = \arccos \left(\frac{\text{tr}(\mathbf{R}) - 1}{2} \right), \quad (2.7)$$

where the *vee* operator $(\cdot)^\vee : \text{Skew}_3 \rightarrow \mathbb{R}^3$ creates a vector from a skew-symmetric matrix and the *hat* operator $(\cdot)^\wedge : \mathbb{R}^3 \rightarrow \text{Skew}_3$ is the inverse of the vee operator [15]. In the following calculations, the first order approximation of the exponential map in eq. (2.6) will be used. Furthermore, we can define similar exponential and logarithm maps for $SE(3)$ but they will be left out in this discussion as they are not directly relevant for this thesis for reasons stated in section 2.3.1.

2.3 Optimization on Manifolds

In Euclidean space, a general way of solving a nonlinear optimization problem is to repeatedly solve a second order approximation of the objective function at each iteration until a solution is found. However, this cannot easily be generalized to manifolds as typically, the problem will be over-parameterized and naively applying an update step does not guarantee that the resulting iterate will again be an element of the manifold [14].

A standard way to overcome this problem is to reparameterize the optimization problem using a *retraction* $\mathcal{R}_x(\delta x) : \mathbb{R}^n \rightarrow \mathcal{M}$, which is a bijective mapping between elements in a neighbourhood of x on the manifold \mathcal{M} and elements δx in the tangent space of \mathcal{M} around x , $T_x\mathcal{M}$ [14, 16].

$$\begin{aligned} \min_{x \in \mathcal{M}} f(x) &\Rightarrow \min_{\delta x \in \mathbb{R}^n} f(\mathcal{R}_x(\delta x)) \\ \text{s.t. } &\delta x \in \Delta. \end{aligned} \quad (2.8)$$

This approach is called *lifting* as the problem is essentially ‘lifted’ into the tangent space around x , where it becomes a constrained optimization problem in Euclidean space. The trust-region Δ is necessary as the retraction is only valid in a neighbourhood around x .

Since the tangent space is isomorphic to Euclidean space, we can compute a local optimizer δx^* of the lifted cost function $f(\mathcal{R}_x(\delta x))$ using standard optimization techniques. This optimizer can then be retracted back into \mathcal{M} using the defined retraction and serves as an update to our previous optimizer x [16].

2.3.1 Retraction

For our problem, the retraction $\mathcal{R}_\phi(\delta\phi)$ of the local tangent space of rotations and the retraction $\mathcal{R}_T(\delta\tau)$ of the local tangent space of rigid body transformations are necessary. The same retractions as in [14] are chosen:

$$\mathcal{R}_\phi(\delta\phi) : \quad T_{\mathbf{R}}SO(3) \rightarrow SO(3); \quad \delta\phi \mapsto \mathbf{R} \text{Exp}(\delta\phi) \quad (2.9)$$

$$\mathcal{R}_T(\delta\tau := (\delta\mathbf{p}, \delta\phi)) : \quad T_TSE(3) \rightarrow SE(3); \quad (\delta\mathbf{p}, \delta\phi) \mapsto (\mathbf{p} + \delta\mathbf{p}, \mathbf{R} \text{Exp}(\delta\phi)), \quad (2.10)$$

where we reutilize the retraction of orientations in the retraction of rigid body transformations. We therefore never actually need to compute the exponential and logarithm maps for $SE(3)$.

The retraction lifts the cost function into the tangent space around the tuple (\mathbf{p}, \mathbf{R}) (i.e. the Lie algebra is shifted from the identity element to the current rigid body transformation). Therefore, the origin of the Lie algebra coincides with (\mathbf{p}, \mathbf{R}) . This allows us to use the first order approximation of the exponential map since we can assume $\delta\phi$ to be sufficiently small.

Chapter 3

Method

After reviewing the preliminaries in chapter 2, we can now discuss the motion planning algorithm used in this thesis. First we will examine the chosen cost functions and their respective gradients. Then, we will turn our attention to the optimization method used to generate the trajectories.

In the following we will refer to the trajectory ξ as the mapping from time to OMAV configuration. Formally: $\xi : [0, 1] \rightarrow SE(3)$ for continuous time and $\xi : \{0, 1, \dots, N + 1\} \rightarrow SE(3)$ for discrete time. Therefore, $\xi(t) = (\mathbf{p}(t), \Phi(t))$ and $\xi[i] = (\mathbf{p}[i], \Phi[i])$. Furthermore, let $\delta\phi, \delta\mathbf{p} \in \mathbb{R}^{N \times 3}$ denote the matrices consisting of row vectors which represent the elements in the local tangent spaces of orientation/translation along the trajectory. These are the optimization variables which allow us to optimize the position and attitude along the trajectory. We will denote the i -th row of $\delta\phi$ using $\delta\phi_i$, which is a vector in the local tangent space of the attitude at the i -th trajectory step. Additionally, we define $\xi_{\mathbf{p}} \in \mathbb{R}^{N \times 3}$ as the matrix consisting of column vectors $\xi_{\mathbf{p}} := [\xi_x \ \xi_y \ \xi_z]$, where each column represents the trajectory of a translational DoF without the start and end states.

3.1 Cost Functions

As an optimization-based motion planning approach was chosen, suitable cost functions $\mathcal{F}(\xi) : \xi \rightarrow \mathbb{R}$ needed to be defined. Our optimizer uses three terms in the cost function, two of which are adaptations of the smoothness and collision cost functions as used in CHOMP. The third term is used to penalize the orientation of *PrisMAV*,

$$\mathcal{F}(\xi) = \mathcal{F}_{\text{smooth}}(\xi) + \mathcal{F}_{\text{col}}(\xi) + \mathcal{F}_{\text{ori}}(\xi). \quad (3.1)$$

3.1.1 Smoothness Cost

The smoothness cost $\mathcal{F}_{\text{smooth}}(\xi)$ essentially takes the same form as in CHOMP [5]. They define the smoothness cost as the path integral over the squared norms of the velocities along the trajectory. However, due to the trajectory ξ being a mapping

from time to $SE(3)$ and not \mathbb{R}^d , some adaptations are needed:

$$\begin{aligned}\mathcal{F}_{\text{smooth}}(\xi) &= \frac{1}{2} \int_0^1 \left\| \frac{d\xi}{dt} \right\|_2^2 dt := \frac{1}{2} \int_0^1 \left\| \frac{d\mathbf{p}(t)}{dt} \right\|_2^2 + \alpha_{\text{smooth}} \left\| \frac{d\Phi(t)}{dt} \right\|_2^2 dt \\ &\propto \frac{N}{2} \sum_{i=1}^{N+1} \|\mathbf{p}[i] - \mathbf{p}[i-1]\|_2^2 + \alpha_{\text{smooth}} \|\Phi[i] \boxminus \Phi[i-1]\|_2^2 \\ &= \text{tr} \left(\frac{1}{2} \xi_{\mathbf{p}}^\top \mathbf{A} \xi_{\mathbf{p}} + \xi_{\mathbf{p}}^\top \mathbf{b} + \mathbf{c} \right) + \frac{\alpha_{\text{smooth}} N}{2} \sum_{i=1}^{N+1} \|\Phi[i] \boxminus \Phi[i-1]\|_2^2, \quad (3.2)\end{aligned}$$

where $\mathbf{A} = N \cdot \mathbf{K}^\top \mathbf{K}$, $\mathbf{b} = N \cdot \mathbf{K}^\top \mathbf{e}$, $\mathbf{c} = \frac{N}{2} \mathbf{e}^\top \mathbf{e}$ and \mathbf{K} denotes a finite differencing matrix and \mathbf{e} denotes the vector which contains the start and end states as described in [5]. $\xi_{\mathbf{p}}$ therefore only contains the free variables along the trajectory which can be optimized over. To describe the finite difference in orientation, the *boxminus operator* $\boxminus : SO(3) \times SO(3) \rightarrow \mathfrak{so}(3)$; $\Phi_1, \Phi_2 \mapsto \text{Log}(\Phi_1 \circ \Phi_2^{-1})$ is used [14]. The boxminus operator is a bilinear mapping which maps two elements in $SO(3)$ to an element in $\mathfrak{so}(3)$. The resulting element in tangent space can be interpreted as the difference in orientation between the two rotations Φ_1, Φ_2 [14].

To compute the gradient of eq. (3.2), the cost function is first lifted into tangent space and reparameterized using the retraction as defined in section 2.3.1. The gradient with respect to $\delta\mathbf{p}$ can be readily computed using matrix calculus as,

$$\nabla_{\delta\mathbf{p}} \mathcal{F}_{\text{smooth}}(\xi) = \mathbf{A}(\xi_{\mathbf{p}} + \delta\mathbf{p}) + \mathbf{b}. \quad (3.3)$$

The resulting attitude gradient $\nabla_{\delta\phi_i} \mathcal{F}_{\text{smooth}}(\xi)$ is less straightforward to calculate since several nonlinearities are introduced into the cost function by the boxminus operator and the retraction. The gradient of $\|\mathbf{R}[i] \text{Exp}(\delta\phi_i) \boxminus \mathbf{R}[i-1] \text{Exp}(\delta\phi_{i-1})\|_2^2$, with respect to $\delta\phi_{i-1}$ and $\delta\phi_i$ can be computed using a symbolic math solver such as SymPy¹ or MATLAB² and then superpositioned for $i = \{1, \dots, N+1\}$ to obtain $\delta\phi$. The closed form of the gradient $\nabla_{\delta\phi} \mathcal{F}_{\text{smooth}}(\xi)$ is omitted for brevity.

3.1.2 Collision Cost

The collision cost $\mathcal{F}_{\text{col}}(\xi)$ used in this thesis closely follows the implementation of the collision cost used in CHOMP [5]:

$$\mathcal{F}_{\text{col}}(\xi) = \int_0^1 \int_{\mathcal{B}} c(x(\xi(t), u)) \left\| \frac{d}{dt} x(\xi(t), u) \right\| du dt, \quad (3.4)$$

where $x : SE(3) \times \mathcal{B} \rightarrow \mathbb{R}^3$ denotes the position x in 3D space of a body point u on *PrisMAV*. This mapping is given by,

$$\mathbf{x}_u^I(\xi[i], u) = \mathbf{p}[i] + \mathbf{R}[i]^\top \mathbf{r}_{\text{IMU} \rightarrow u}^B. \quad (3.5)$$

The function $c : \mathbb{R}^3 \rightarrow \mathbb{R}$ is the collision potential function which assigns a collision potential to each point in 3D space. The same potential function as in [5] is used:

$$c(x) = \begin{cases} -\mathcal{D}(x) + \frac{1}{2}\epsilon & \text{if } \mathcal{D}(x) < 0, \\ \frac{1}{2\epsilon}(\mathcal{D}(x) - \epsilon)^2 & \text{if } 0 < \mathcal{D}(x) \leq \epsilon, \\ 0 & \text{otherwise,} \end{cases} \quad (3.6)$$

¹<https://www.sympy.org/en/index.html>

²<https://www.mathworks.com/products/symbolic.html>

(accessed: 31.08.21)

(accessed: 31.08.21)

where $\mathcal{D}(x)$ is an Euclidean signed distance field (ESDF). The function $c(x)$ smoothly tapers off to 0 as $\mathcal{D}(x)$ approaches the minimum clearance parameter ϵ . This prevents trajectories from diverging completely from any obstacles as the gradient becomes flat after the clearance tolerance is reached and allows us to intuitively tune the proximity of the trajectory to obstacles. The gradient $\nabla_x c(x)$ can be computed by using the chain rule and finite differencing of $\mathcal{D}(x)$.

The inclusion of the velocity of the body point in eq. (3.4) leads to an arc-length reparameterization. This is advantageous since the cost becomes invariant to the velocity of the trajectory. This prevents the gradient from encouraging the trajectory to ‘jump through obstacles’ [5].

The gradient of eq. (3.4) is shown by [5] to be,

$$\nabla_{\delta \mathbf{p}_i, \delta \phi_i} \mathcal{F}_{\text{col}}(\xi) = \int_{\mathcal{B}} \mathbf{J}^\top \|\dot{\mathbf{x}}\| \left((I - \hat{\mathbf{x}}\hat{\mathbf{x}}^\top) \nabla c - c\kappa \right) du, \quad (3.7)$$

where the time index $[i]$ and body point dependency u has been dropped for brevity. $\widehat{(\cdot)}$ denotes a normalized vector and κ is the curvature of the trajectory $\kappa = \|\dot{\mathbf{x}}\|^{-2} (I - \hat{\mathbf{x}}\hat{\mathbf{x}}^\top) \ddot{\mathbf{x}}$. Furthermore, \mathbf{J} denotes the Jacobian of $\mathbf{x}_u^I(\xi[i], u)$ with respect to $\delta \mathbf{p}_i$ and $\delta \phi_i$. Lifting eq. (3.5) into tangent space results in,

$$\mathbf{x}_u^I(\xi[i], u) = \mathbf{p}[i] + \delta \mathbf{p}_i + (\mathbf{R}[i] \text{Exp}(\delta \phi_i))^\top \mathbf{r}_{\text{IMU} \rightarrow u}^B, \quad (3.8)$$

which leads to,

$$\mathbf{J} = [\mathbf{I} \quad \alpha_{\text{col}} \mathbf{M}], \quad (3.9)$$

where,

$$\mathbf{M} = \begin{bmatrix} 0 & -\mathbf{R}_{02}\mathbf{r}_0 - \mathbf{R}_{12}\mathbf{r}_1 - \mathbf{R}_{22}\mathbf{r}_2 & \mathbf{R}_{01}\mathbf{r}_0 + \mathbf{R}_{11}\mathbf{r}_1 + \mathbf{R}_{21}\mathbf{r}_2 \\ \mathbf{R}_{02}\mathbf{r}_0 + \mathbf{R}_{12}\mathbf{r}_1 + \mathbf{R}_{22}\mathbf{r}_2 & 0 & -\mathbf{R}_{00}\mathbf{r}_0 - \mathbf{R}_{10}\mathbf{r}_1 - \mathbf{R}_{20}\mathbf{r}_2 \\ -\mathbf{R}_{01}\mathbf{r}_0 - \mathbf{R}_{11}\mathbf{r}_1 - \mathbf{R}_{21}\mathbf{r}_2 & \mathbf{R}_{00}\mathbf{r}_0 + \mathbf{R}_{10}\mathbf{r}_1 + \mathbf{R}_{20}\mathbf{r}_2 & 0 \end{bmatrix}. \quad (3.10)$$

The time index $[i]$ has been dropped for brevity. α_{col} denotes a scaling factor which allows us to tune the attitude component of the collision gradient independently from the translational gradient. In fig. 3.1, the respective influences of both the translational and attitude collision gradient are visualized. In fig. 3.1a we see the effect of $\alpha_{\text{col}} = 0$. The trajectory avoids collision by changing its translational trajectory and leaving the attitude unchanged. Figure 3.1b shows the effect of choosing a high orientation scaling factor as well as a low translational learning rate³. The trajectory avoids collision by changing its orientation and ‘rotating the rotorarms around the obstacle’.

Since the computation of the body integral in eqs. (3.4) and (3.7) over all body points is impractical, the collision mesh of the robot is decomposed into a set of collision spheres. The integral can then be evaluated for the centre of each collision sphere to obtain an approximation of the complete integral.

3.1.3 Orientation Cost

PrisMAV is capable of flight in essentially any conceivable pose. However, there are some preferred flight modes which are more energy-efficient such as horizontal flight or tricopter flight. If we parameterize $SO(3)$ as shown in fig. 3.2a, these modes can

³a low translational learning rate is necessary as otherwise the trajectory will be pushed out of collision by the translational DoFs since the translational gradient is independent of the parameter α_{col}

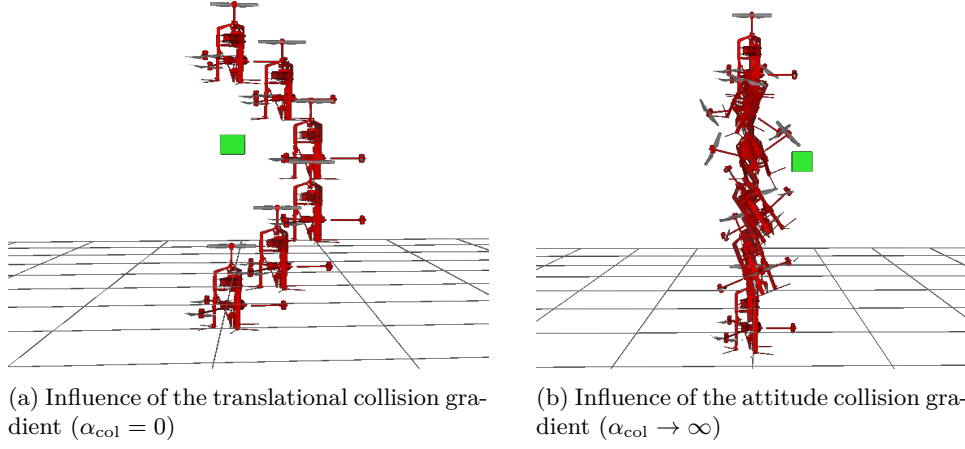


Figure 3.1: Visualizations of the influence of the collision gradient $\nabla_{\delta \mathbf{p}, \delta \phi} \mathcal{F}_{\text{col}}(\xi)$ on the trajectory ξ

be expressed by $\theta_{\text{tricopter}} = 0$ rad and $\theta_{\text{horizontal}} = \pi/2$ rad. The global minimum of the cost function was chosen in the tricopter flight mode and a local minimum was placed at the horizontal flight mode. According to these criteria, a fourth order polynomial in θ was fitted which can be seen in fig. 3.2b.

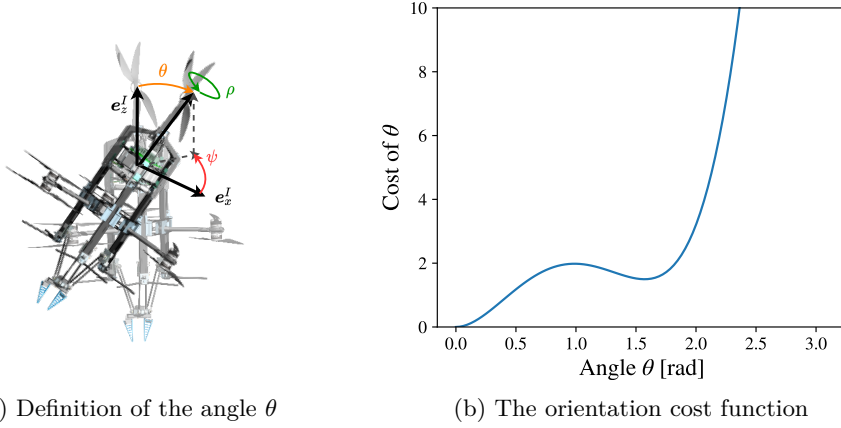


Figure 3.2: The angle θ and its corresponding cost. The global minimum of the cost is achieved when *PrisMAV* is in tricopter orientation.

The resulting orientation cost function $\mathcal{F}_{\text{ori}}(\xi)$ can be computed as the path integral over the instantaneous orientation cost along the trajectory ξ :

$$\mathcal{F}_{\text{ori}}(\xi) = \int_0^1 a \cdot \theta^4(t) + b \cdot \theta^3(t) + c \cdot \theta^2(t) dt \quad (3.12)$$

$$\propto \frac{1}{N+2} \sum_{i=0}^{N+1} a \cdot \theta^4[i] + b \cdot \theta^3[i] + c \cdot \theta^2[i] \quad (3.13)$$

with

$$\theta[i] = (\mathbf{e}_z^I)^\top \mathbf{R}[i] \mathbf{e}_z^I, \quad (3.14)$$

where a, b, c are the parameters of the polynomial, \mathbf{e}_z^I is the inertial z -axis and $\mathbf{R}[i]$ denotes the rotation matrix associated with attitude $\Phi[i]$ of *PrisMAV*.

We lift the problem into tangent space by re-parameterizing the current attitude $\mathbf{R}[i]$ by $\mathbf{R}[i] \text{Exp}(\delta\phi_i)$. The gradient of the discrete-time orientation cost can be computed using the chain rule. The resulting gradient in tangent space for time step i is given by

$$\nabla_{\delta\phi_i} \mathcal{F}_{\text{ori}}(\mathbf{R}[i], \delta\phi_i) = \frac{4a\theta^3[i] + 3b\theta^2[i] + 2c\theta[i]}{\sqrt{1 - (\mathbf{R}[i]_{33} + \mathbf{R}[i]_{31}\delta\phi_{i,2} - \mathbf{R}[i]_{32}\delta\phi_{i,1})^2}} \begin{bmatrix} \mathbf{R}[i]_{32} \\ -\mathbf{R}[i]_{31} \\ 0 \end{bmatrix} \quad (3.15)$$

The result of applying gradient descent using the orientation cost can be seen in fig. 3.3, where three iterations of gradient descent were performed on uniformly sampled rotations. We observe that the orientations converge to a tricopter orientation $\theta = 0^\circ$ or a horizontal orientation $\theta = 90^\circ$ depending on initialization. If initialized with $\theta < 56^\circ$, the orientation converges to a tricopter orientation and if $\theta > 56^\circ$, the orientation converges to a horizontal orientation

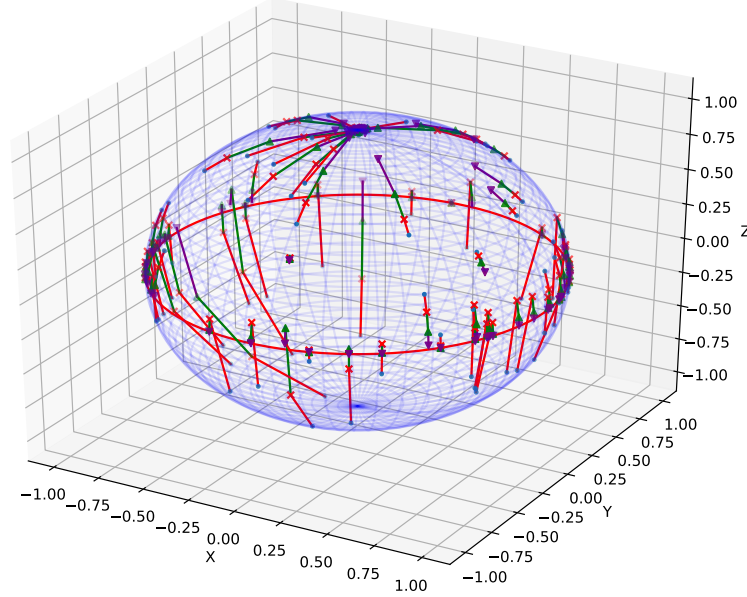


Figure 3.3: Gradient descent of the orientation cost. Each point represents an attitude of *PrisMAV*. Blue dots indicate the starting orientation and the purple triangles indicate the orientation after performing 3 iterations of gradient descent.

3.2 Optimization

The optimization problem associated with generating an optimal trajectory ξ subject to the cost functions discussed in section 3.1 can be expressed as,

$$\begin{aligned} \min_{\delta\phi, \delta\mathbf{p} \in \mathbb{R}^{N \times 3}} & w_{\text{smooth}} \mathcal{F}_{\text{smooth}}(\xi, \delta\phi, \delta\mathbf{p}) + w_{\text{col}} \mathcal{F}_{\text{col}}(\xi, \delta\phi, \delta\mathbf{p}) + w_{\text{ori}} \mathcal{F}_{\text{ori}}(\xi, \delta\phi), \\ \text{s.t. } & \delta\phi \in \Delta_\phi, \\ & \delta\mathbf{p} \in \Delta_p, \end{aligned} \quad (3.16)$$

where $w_{(\cdot)}$ denotes the weight associated with the corresponding cost function. Furthermore, $\Delta_{(\cdot)}$ denotes the trust-region corresponding to $\delta(\cdot)$. As the translational DoFs are already in Euclidean space, the trust-region Δ_p was assumed to be infinite and the trust region Δ_ϕ was chosen as a 2-norm ball with radius r . The trust region problem could therefore be formulated as an unconstrained optimization problem using orthogonal projection in the update rule.

3.2.1 Orthogonal projection

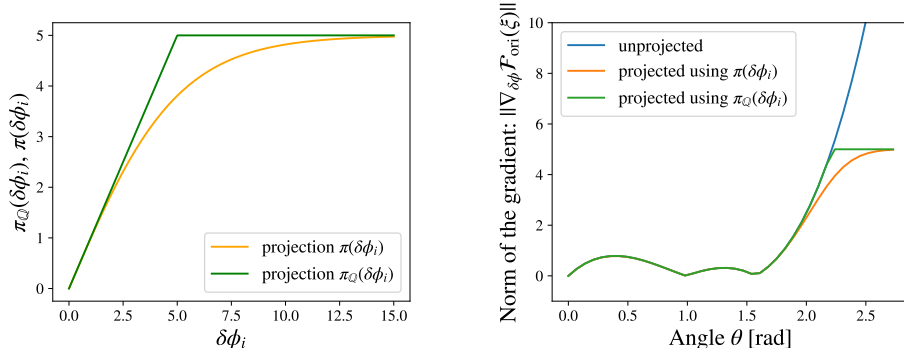
As discussed above, orthogonal projection is used to transform the constrained optimization problem into an unconstrained optimization problem. The chosen orthogonal projection is related to the orthogonal projection used for 2-norm balls $\mathbb{Q} := \{ \delta\phi_i \in \mathbb{R}^n \mid \|\delta\phi_i\| < r \}$ with $r > 0$. The corresponding projection is

$$\pi_{\mathbb{Q}}(\delta\phi_i) = \begin{cases} \frac{\delta\phi_i}{\|\delta\phi_i\|} r & \text{if } \|\delta\phi_i\| > r, \\ \delta\phi_i & \text{otherwise.} \end{cases} \quad (3.17)$$

We employ what can be considered a smooth approximation of eq. (3.17) which is given by,

$$\pi(\delta\phi_i) = \delta\phi_i \cdot r \cdot \tanh\left(\frac{\|\delta\phi_i\|}{r}\right). \quad (3.18)$$

A comparison between the two projections can be seen in fig. 3.4a.



(a) Both projections for an arbitrary choice of $r = 5$. Green: $\pi_{\mathbb{Q}}(\delta\phi_i)$, Yellow: $\pi(\delta\phi_i)$

(b) A comparison of the unprojected and projected gradients of the mentioned example with $r = 5$

Figure 3.4: Comparison of the described projections

Using the projection in eq. (3.18) ensures that the orientation gradients along the trajectory are ‘well-behaved’. For instance, if we consider a trajectory ξ which rotates from $\theta(0) = 3\pi/4$ rad to $\theta(1) = 0$ rad we can imagine that the gradient of the ‘instantaneous’ orientation cost $\mathcal{F}_{\text{ori}}(\theta)$ can become relatively large for ‘big’ $\theta(t)$. If we use the projection as stated in eq. (3.17) the orientation gradient along the trajectory will be at its maximum value r if the magnitude of the gradient is bigger than the allowed value r . Using eq. (3.18), we ensure that the gradient is continuous along the trajectory as the projection $\pi(\delta\phi_i)$ is continuous. The attitude gradient resulting from this hypothetical problem can be inspected in fig. 3.4b.

3.2.2 Gradient Descent and Update Rule

After formulating the gradient in tangent space, $\nabla_{\delta \mathbf{p}, \delta \phi} \mathcal{F}(\xi)$, we can compute an update step in tangent space in the direction of steepest descent,

$$\begin{bmatrix} \delta \mathbf{p}_{i,j+1} \\ \delta \phi_{i,j+1} \end{bmatrix} = \begin{bmatrix} \delta \mathbf{p}_{i,j} \\ \delta \phi_{i,j} \end{bmatrix} - \begin{bmatrix} \eta_{\text{trans}} \nabla_{\delta \mathbf{p}_i} \mathcal{F}(\xi_j) \\ \eta_{\text{rot}} \nabla_{\delta \phi_i} \mathcal{F}(\xi_j) \end{bmatrix}, \quad (3.19)$$

and then update our trajectory point $(\mathbf{p}_{\text{new}}, \Phi_{\text{new}})$ by retracting the update step back onto $SE(3)$ using the projection in eq. (3.18) and the retraction in eq. (2.10). Setting $[\delta \mathbf{p}_{i,j}^\top \ \delta \phi_{i,j}^\top]^\top = [\mathbf{0}^\top \ \mathbf{0}^\top]^\top$, the update equation becomes,

$$\xi_{\mathbf{p},j+1} = \xi_{\mathbf{p},j} - \eta_{\text{trans}} \mathbf{A}^{-1} \nabla_{\delta \mathbf{p}} \mathcal{F}(\xi_j) \quad (3.20)$$

$$\mathbf{R}_{i,j+1} = \mathbf{R}_{i,j} \text{Exp} \left(\pi \left(\eta_{\text{rot}} \nabla_{\delta \phi_i} \mathcal{F}(\xi_j) \right) \right)^\top \forall i \in \{1, 2, \dots, N\}, \quad (3.21)$$

where we adapted the update rule for the translational DoFs to include the Riemannian metric \mathbf{A} . The translational update eq. (3.20) therefore becomes the standard CHOMP update rule seen in eq. (2.3). In eq. (3.21), we exploited the fact that $\text{Exp}(-\delta \phi_i) = \text{Exp}(\delta \phi_i)^\top$.

3.2.3 Trajectory Discretization

The trajectory discretization N was performed by taking the weighted sum of the absolute Euclidean distance and the absolute angular distance between start and end state.

$$N = c_{\text{trans}} \cdot \|\mathbf{p}_{\text{end}} - \mathbf{p}_{\text{start}}\| + c_{\text{rot}} \cdot \angle(\Phi_{\text{end}}, \Phi_{\text{start}}), \quad (3.22)$$

where $\angle(\cdot)$ is a function which returns the absolute angle between two rotations⁴. The two coefficients c_{trans} , c_{rot} can be interpreted as steps per meter and steps per radian, respectively. This allows us to have approximately constant trajectory step spacing for different trajectory lengths and therefore the planner parameters do not need to be retuned for new problems.

3.2.4 Cost Weight Scheduling

To increase the success rate and performance of our planner, we opted to implement cost weight scheduling as described in [5]. During the initial ‘exploratory-phase’ of the planning algorithm, the planner tries to find a feasible, collision-free trajectory. This is encouraged by choosing a low w_{smooth} relative to w_{col} . If a collision-free path is found, we increase w_{smooth} in the following ‘smoothing-phase’ in order to smooth out the trajectory.

3.3 Implementation

The implementation of our planner was performed as a plugin for the motion planning framework MoveIt [17]. MoveIt comes equipped with all the necessary components for motion planning, such as collision checking, a modular planner interface, sensor integration for on-line replanning and inverse kinematic solvers. This allowed us to focus on the implementation of our planner and use the off-the-shelf collision checkers provided by MoveIt. The CHOMP implementations in the `ethz-asl/mav_local_avoidance`⁵ and the `ros-planning/moveit`⁶ repositories served as a starting point. The complete planning algorithm is demonstrated in algorithm 1.

⁴<http://boris-belousov.net/2016/12/01/quat-dist/>

(accessed: 01.09.21)

⁵https://github.com/ethz-asl/mav_local_avoidance

(accessed: 10.09.2021)

⁶<https://github.com/ros-planning/moveit>

(accessed: 10.09.2021)

Algorithm 1 *PrisMAV* Motion Planning Algorithm

Require: Valid start and end states $(\mathbf{p}_{\text{start}}, \Phi_{\text{start}})$, $(\mathbf{p}_{\text{end}}, \Phi_{\text{end}})$

Require: ESDF of the planning environment $\mathcal{D}(x)$

```

1:  $\xi_0 \leftarrow \text{lerp}((\mathbf{P}_{\text{end}}, \Phi_{\text{end}}))$ 
2: while stopping criterion not reached do
3:   if collision-free trajectory found then
4:     performWeightScheduling()
5:      $\eta_j \leftarrow \eta_0 / \sqrt{1+j}$  ▷ Iteratively decrease learning rate
6:   end if
7:    $\nabla_{\delta \mathbf{p}} \mathcal{F}(\xi_j), \nabla_{\delta \Phi} \mathcal{F}(\xi_j) \leftarrow \text{computeGradient}(\xi_j)$ 
8:    $\xi_{\mathbf{p}, j+1} \leftarrow \xi_{\mathbf{p}, j} - \eta_{\text{tans}} \mathbf{A}^{-1} \nabla_{\delta \mathbf{p}} \mathcal{F}(\xi_j)$  ▷ Update the translational DoFs
9:   for  $i=\{1, 2, \dots, N\}$  do ▷ Update the attitude DoFs
10:     $\mathbf{R}_{i, j+1} \leftarrow \mathbf{R}_{i, j} \text{Exp}(\pi(\eta_{\text{rot}} \nabla_{\delta \Phi_i} \mathcal{F}(\xi_j)))^\top$  ▷  $\mathbf{R}_{i, j}$ : Rot. for it.  $j$ , step  $i$ 
11:   end for
12:   cost  $\leftarrow \text{evaluateCostFunction}(\xi_{j+1})$ 
13: end while
14: return  $\xi_{\text{final}}$ 

```

Chapter 4

Experiments and Results

To validate our proposed planner (in the following also denoted as *PrisMAV* planner), several experiments were conducted. In the following, we will introduce the testing environments as well as the experiments and their results.

4.1 Testing Environment

To evaluate the performance of our planner, three test worlds were constructed. The first world was meant to simulate a random forest, the second world was implemented to test the planner's capability to plan around various objects in an unstructured 3D-space and the third environment consisted of a hole in a wall. The trees of the forest were sampled uniformly on a 2D-grid in the form of cylinders while the obstacles in the 3D-space were generated as cubes with a side length of 1 m. The hole in the third world was generated by projecting a scaled version of *PrisMAV*'s collision mesh onto the wall.

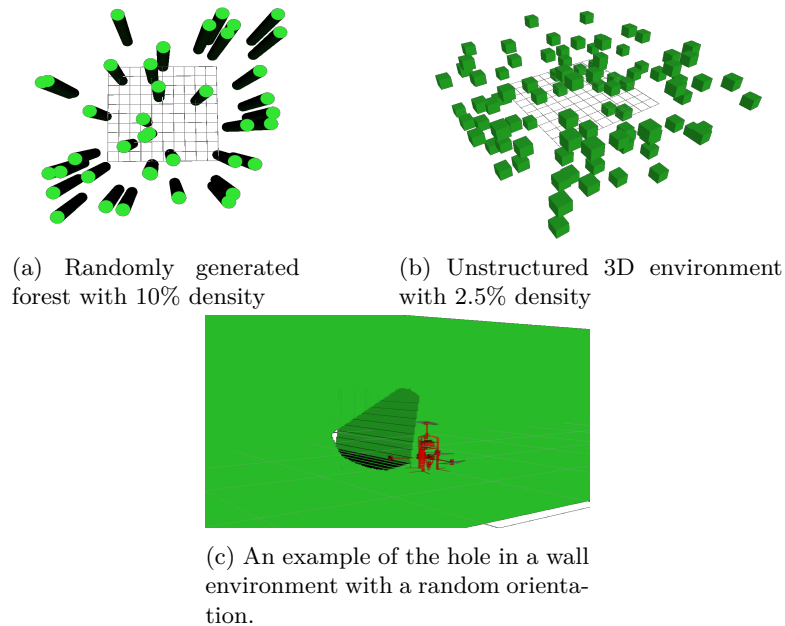


Figure 4.1: The testing worlds which were used for the planner evaluation

As a benchmark the `ompl`¹ implementation of RRT was used. The performance was evaluated using the following metrics:

- Success Rate
- Planning Time
- Translational length of the trajectory
- Angular length of the trajectory
- Orientation cost of the trajectory

Since we want *PrisMAV* to prefer flight in either the horizontal or tricopter mode, we need some metric which expresses how much time is spent in ‘good’ orientations (tricopter or horizontal flight) vs. ‘bad’ orientations. Evaluation of this metric is done using the orientation cost $\mathcal{F}_{\text{ori}}(\xi)$ as described in section 3.1.3. A high orientation cost therefore indicates long flight periods in unfavourable orientations, whereas a low orientation cost indicates more time spent in tricopter or horizontal mode.

4.2 Randomly Generated Forests

To test the performance of our planner, randomly generated forests with densities of 5%, 7.5% and 10% were generated. The allocated planning time was 5 s for both RRT and *PrisMAV* planner. Furthermore, we extracted the trajectory produced by our planner as soon as it first became feasible. This allows us to compare both the first feasible as well as the optimized trajectory to RRT.

4.2.1 Results

Presented here are the results for the randomized forests. A selection of trajectories can be seen in fig. 4.2. We notice that *PrisMAV* trajectories generally appear to be shorter and smoother.

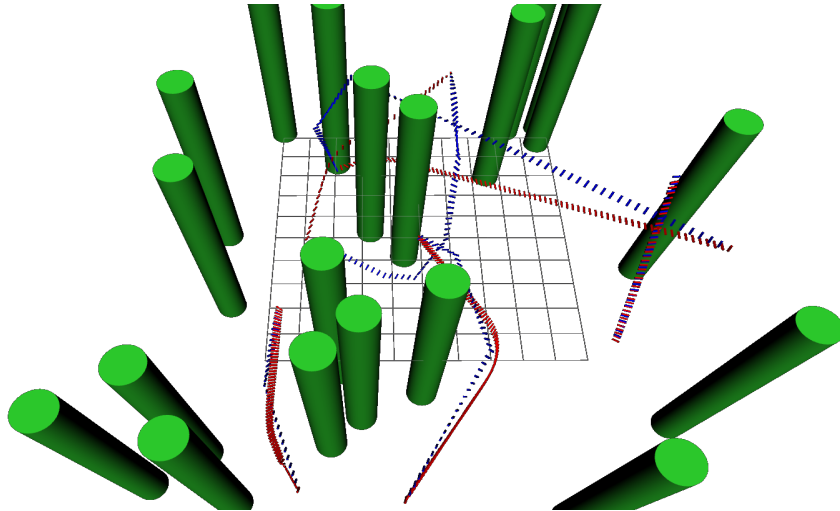


Figure 4.2: A selection of trajectories produced by RRT (blue) and our planner (red) in 5% forest density. Dashes indicate position and orientation of *PrisMAV* along the trajectory.

¹<https://ompl.kavrakilab.org/>

(accessed: 03.09.21)

Success Rate

An important metric in motion planning is the success rate of the planner. The influence of the forest density on the success rate can be seen in fig. 4.3. We note that only valid planning problems are plotted²

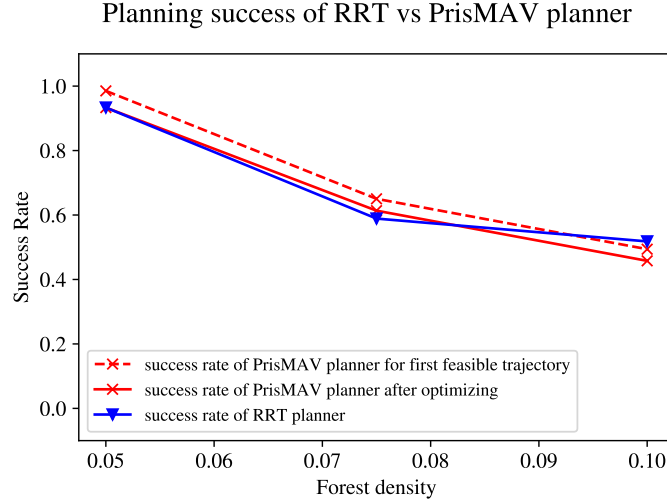


Figure 4.3: Success rate in a random forest as a function of forest density. We observe that the success rate decreases with increasing forest density

In 5% forest density, we observe that our planner has a probability of 98% to find a feasible solution if we terminate the planner as soon as the first feasible trajectory is found while RRT only has a 93% success rate. However, due to the optimization of the cost functions we lose some solutions and end up with a success rate of 93%. Reasons for this will be discussed later in section 5.3.3. For denser forests, the success rate of our planner decreases to 49% for the first feasible paths and to 46% for the optimized paths. RRT achieves a success rate of 51%.

Planning time

In fig. 4.4, the results of the planning time metric are presented. The first plot indicates the planning times of RRT in the respective environment. The second plot shows the planning times for *PrisMAV* planner until a first feasible trajectory is found as well as the time until an optimized trajectory is returned. The last plot shows the difference in planning time between RRT and our planner for the same start and end states. A positive difference indicates that our planner finds a solution quicker than RRT and vice versa.

On average, RRT takes 9 ms while our planner only needs 2 ms to find a feasible path in a 5% forest density. With increasing density the variance of planning time increases more for our planner than for the RRT planner. However, from the boxplot we can interpret that our planner finds a feasible solution faster than RRT if we terminate as soon as a feasible trajectory is found. In the case of a 10% density environment we have an 82% probability of finding a feasible path faster than RRT. If we compare planning times for the optimized trajectories, we observe that RRT outperforms our planner as the distributions are skewed towards negative differences. This becomes more pronounced with increasing obstacle density.

²To assess whether a problem was valid, RRTConnect was run for 10 s. If neither RRTConnect nor any of the other planners found a feasible solution we assumed the problem to be infeasible.

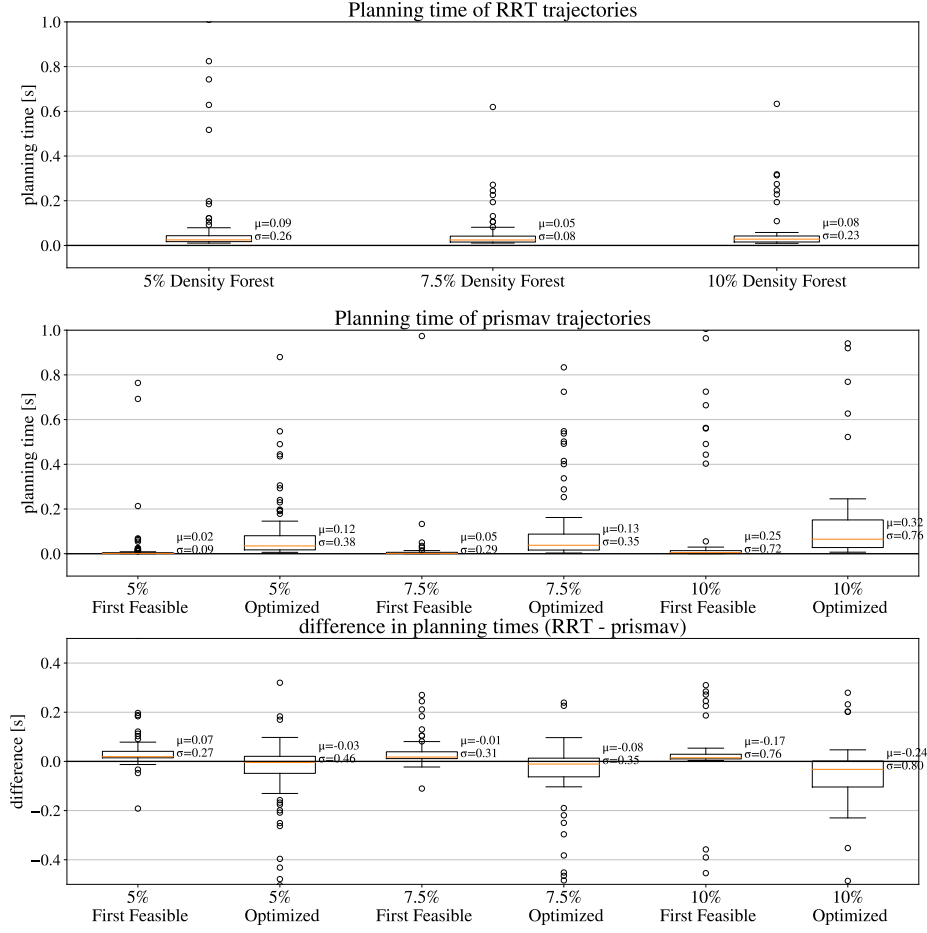


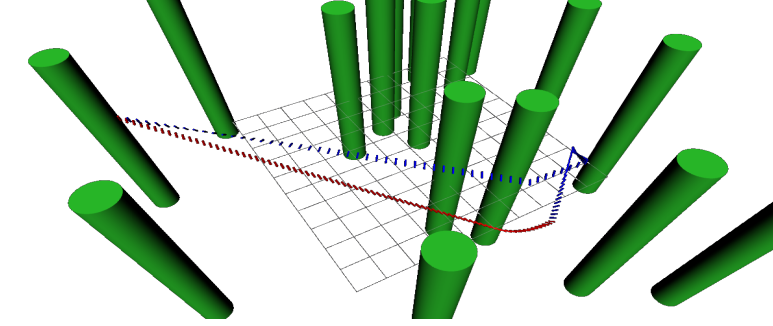
Figure 4.4: Box-plots of the distribution of used planning time using either the RRT planner, our planner and the time used to get a first feasible path with our planner

Translational Length

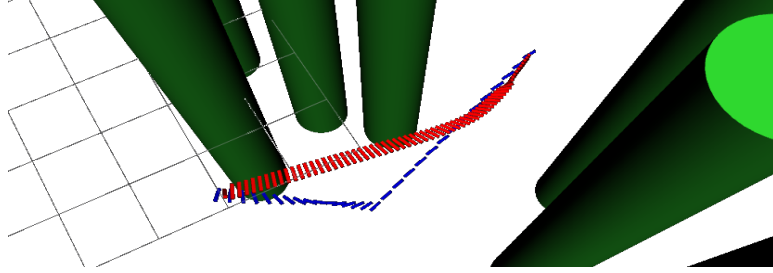
Qualitative and quantitative analysis of the output trajectories show that already the first feasible paths calculated by our planner perform better regarding this metric compared to RRT. In general, we can claim that our planner has roughly a 80% probability of calculating a shorter path. Due to the fact that RRT samples uniformly across the space, the resulting path can include clearly suboptimal detours. In fig. 4.5 two pairs of trajectories are shown which exhibit the mentioned behaviour.

In fig. 4.6 we plot the difference in translational length between RRT and our planner. We first compute the arc length integral along the translational trajectory ξ_p for both RRT and our proposed planner for each start/end tuple. Then, we subtract one from the other. Here, a positive difference indicates that our planner obtains a shorter trajectory.

We observe that our planner consistently produces lower path lengths compared to RRT. In a 5% density forest, the trajectories produced by our planner are shorter with a probability of 76% which drops to 60% as the forest density increases.



(a) Example where RRT produces a significantly longer path than our proposed planner. We can clearly see that the RRT trajectory takes a detour into the wrong direction at first.



(b) Another example where our planner produces a more optimal path.

Figure 4.5: Qualitative analysis of trajectory length of the RRT planner (blue) and our proposed planner (red).

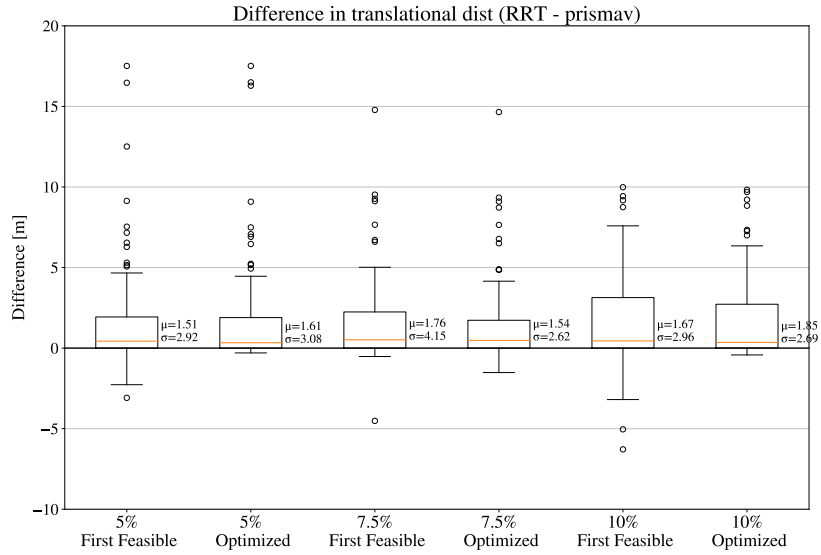


Figure 4.6: Difference in translational distances between RRT and *PrisMAV* planner trajectories.

Angular Length

The same analysis as for the translational trajectory length was done for the angular length along the trajectory. Again the difference between a RRT path and a path produced by our planner is computed. The results can be seen in fig. 4.7.

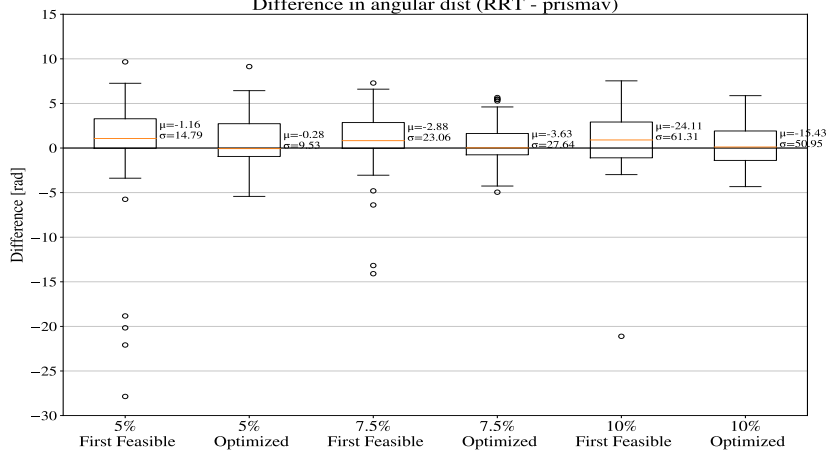


Figure 4.7: Difference in angular length of RRT trajectories and trajectories produced by our planner.

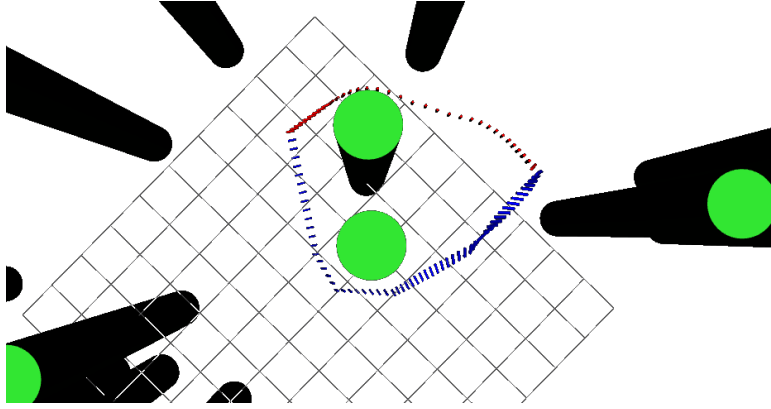
Quantitative analysis shows that $P(\text{difference} > 0) \in [50\%, 64\%]$, which indicates that our planner produces paths with shorter angular length in approximately half of the cases. We therefore cannot conclude that our planner performs significantly better in this metric. A reason for this is that we optimize the orientation according to the orientation cost in our planner which can lead to trajectories with increased angular length.

Qualitative analysis of the produced paths does not lead to the conclusion that RRT performs systematically better with respect to this metric. As can be seen in the selection of paths shown in fig. 4.8, RRT trajectories can include significant unnecessary changes in orientation along the trajectories.

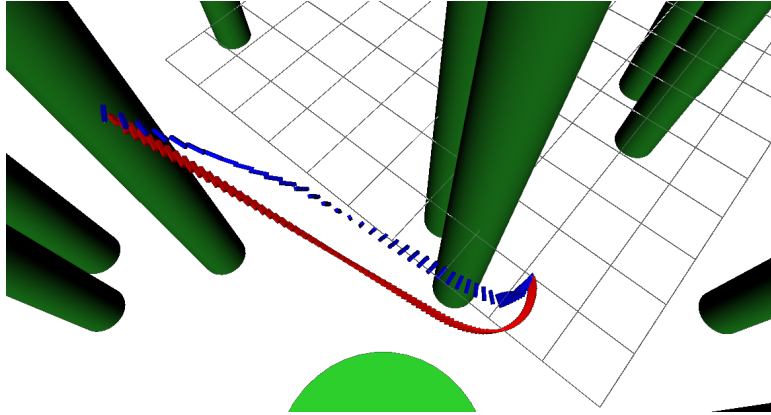
Orientation Cost

Last but not least, the orientation cost for the trajectories was computed. The results are presented in fig. 4.9. Our planner performs significantly better in this metric compared to RRT as all medians, as well as the 25% quartiles, are greater than 0 for all tested forest densities. We can expect 88% of trajectories in a 5% forest and 82% of trajectories in a 10% forest to yield lower orientation cost compared to RRT. This metric directly implies that our planner prefers flight in orientations we deem ‘good’. Even the barely optimized first feasible trajectories have a probability of over 70% to have a lower orientation cost than the RRT planner. The result is unsurprising since we directly optimize this metric in our planner and therefore expect path produced by *PrisMAV* planner to yield lower orientation cost.

In fig. 4.10, a RRT and a *PrisMAV* planner trajectory can be seen which both solve the same planning problem. The RRT trajectory passes through the inverse tricopter mode which was deemed a high cost region, whereas the *PrisMAV* planner trajectory stays in the horizontal flight mode along the trajectory. The resulting difference in orientation cost is 5.7.



(a) An example where a RRT path includes unnecessary change in orientation.



(b) A further example showcasing the same effect.

Figure 4.8: Qualitative analysis of angular trajectory length of the RRT planner (blue) and our proposed planner (red)

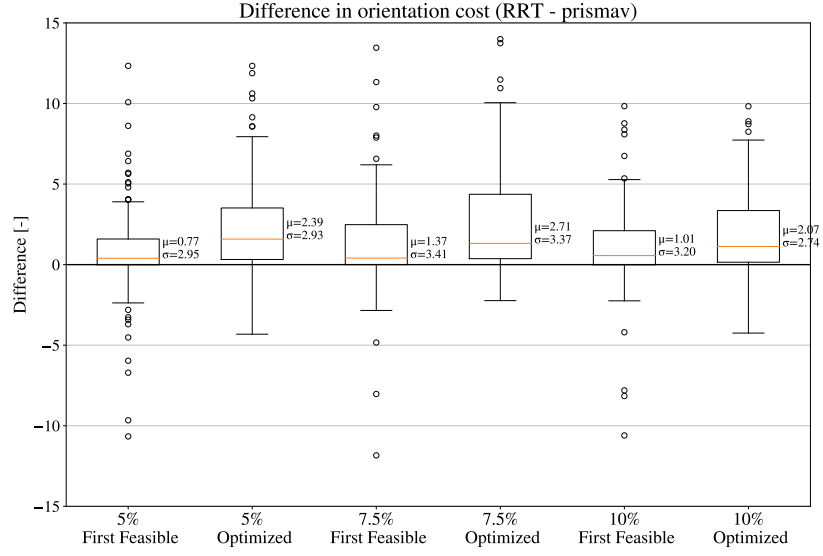
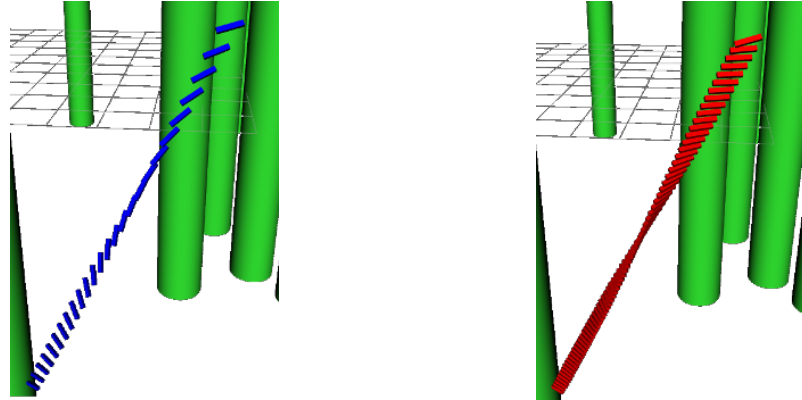


Figure 4.9: Difference in orientation cost between RRT and our proposed planning algorithm



(a) RRT trajectory passes through inverse tricopter (high orientation cost region).

(b) Trajectory produced by our planner stays in the horizontal flight mode along the trajectory.

Figure 4.10: A comparison of a RRT trajectory (blue) with its *PrisMAV* planner counterpart (red). The resulting difference in orientation cost is 5.7.

4.3 Planning in an Unstructured Environment

Testing in this environment was performed to assess whether our planner also works in more complex scenes since the forest test can be considered a test in a 2D plane (changes in height along the z -axis does not change the environment).

The testing was performed in environments with 2.5% and 5% obstacle densities respectively. We extracted a trajectory from *PrisMAV* planner once it found a feasible path and again after it converged to a solution. The planning time limit was set to 5 s.

4.3.1 Results

In this section we will discuss the results obtained for the unstructured environment tests. We generally observe similar results as in the randomized forests. A selection of trajectories can be seen in fig. 4.11

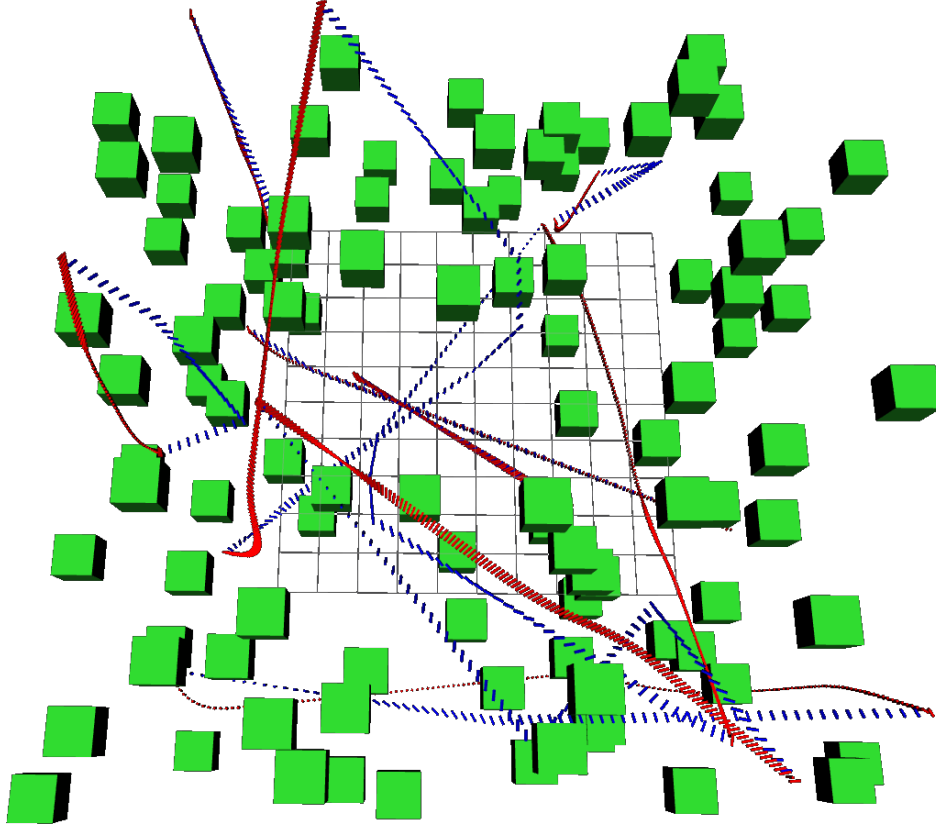


Figure 4.11: A selection of RRT trajectories (blue) and trajectories produced by our planner (red) in an unstructured environment with 2.5% density

Success Rate

Figure 4.12 shows the obtained success rate in this testing environment. We observe that our planner has a 100% success rate when returning the first feasible path. This success rate drops to 95% and 90% in the 2.5%, respectively 5% obstacle density environment if the trajectories are optimized further. We also observe that RRT achieves a 90-92% success rate.

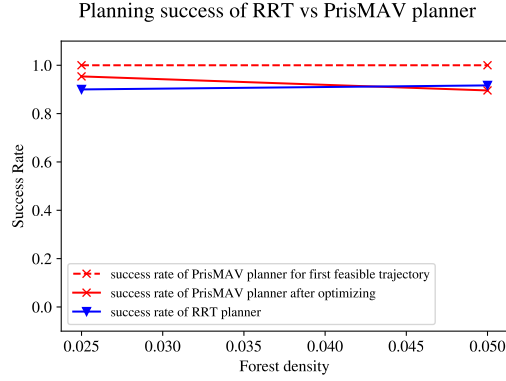


Figure 4.12: Success rate of RRT and *PrisMAV* planner in different obstacle densities.

Planning Time

In fig. 4.13, the planning times in unstructured environments are presented. We observe that the difference is positive if we terminate the planner as soon as a feasible trajectory is found. On average, our planner finds a feasible solution 0.15s and 0.27s faster than RRT in 2.5%, respectively 5% obstacle density. We observe that the distribution shifts towards negative differences and becomes more heavy tailed if we compare the planning times of the optimized solutions against RRT.

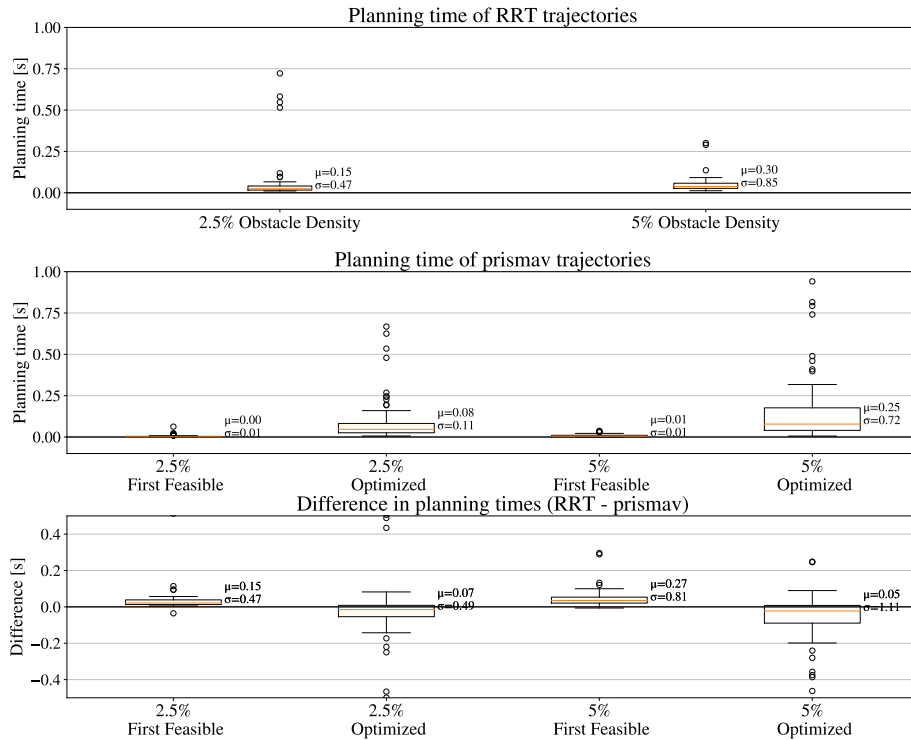


Figure 4.13: Planning times until a first feasible trajectory or an optimized trajectory is found.

Translational Length

In fig. 4.14, the differences in translational distances of RRT trajectories compared to *PrisMAV* planner trajectories are visualized. Again a positive difference indicates that our planner performed better. We observe little change from the 25-th percentile upwards between the difference statistics for the first feasible and the optimized trajectories in a 2.5% environment. This indicates that the first feasible trajectories in this environment are already close to an optimal solution and only a minor amount of trajectories experiences a decrease in translational length.

For the 5% density environment, we observe that the difference distribution for the first feasible trajectories is almost symmetric with a mean of $\mu = -3.4$ m, standard deviation $\sigma = 8.59$ m and $P(\text{difference} > 0) = 0.25$. If the trajectories are optimized further, the distribution becomes skewed towards positive differences and we obtain $P(\text{difference} > 0) = 0.77$.

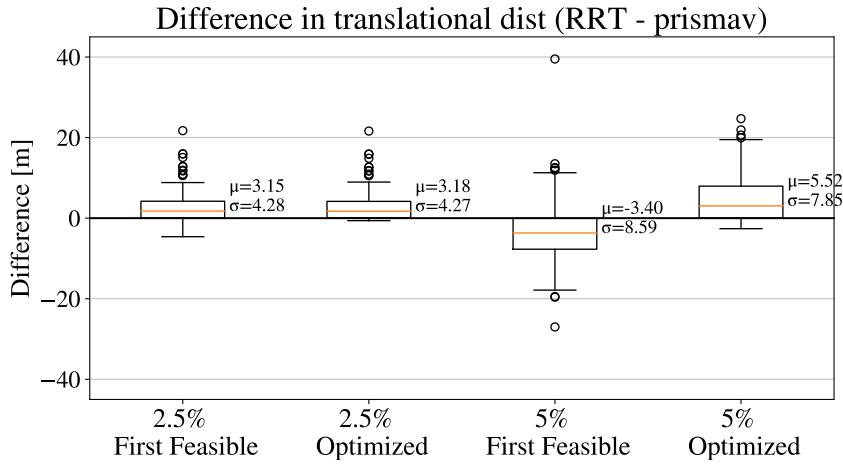


Figure 4.14: Difference in translational trajectory length between RRT trajectories and trajectories produced by our planner. A positive difference indicates that our planner performed better in this metric.

Angular Length

In fig. 4.15, the difference in angular distances of RRT trajectories compared to trajectories produced by *PrisMAV* planner are plotted. We observe that the expected value of the differences is more positive for the unoptimized trajectories. This is due to the fact that we are optimizing the orientation according to the orientation cost which can introduce more angular distance into the trajectory.

Orientation cost

Figure 4.16 shows the difference in orientation cost between RRT and *PrisMAV* planner trajectories. We observe that the mean is positive for all four cases and is more skewed towards positive differences for the optimized trajectories. This is unsurprising since we directly optimize this metric and therefore expect *PrisMAV* planner to perform better. Next to the increasing mean, we also notice that the standard deviation decreases from first feasible trajectories to the optimized trajectories. This indicates that a higher fraction of trajectories possesses a positive difference. Specifically, we can expect 90% of trajectories in 2.5% obstacle density and 93% of trajectories in 5% obstacle density to have a positive difference.

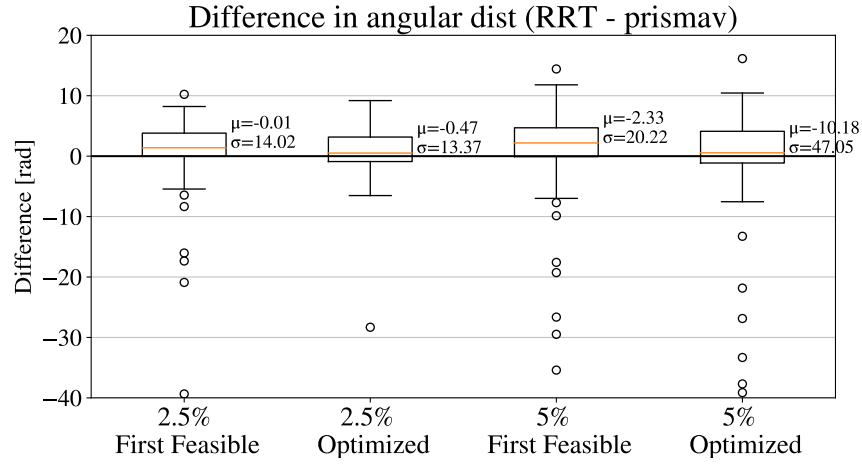


Figure 4.15: Difference in angular trajectory length between RRT trajectories and trajectories produced by our planner. A positive difference indicates that our planner performed better in this metric.

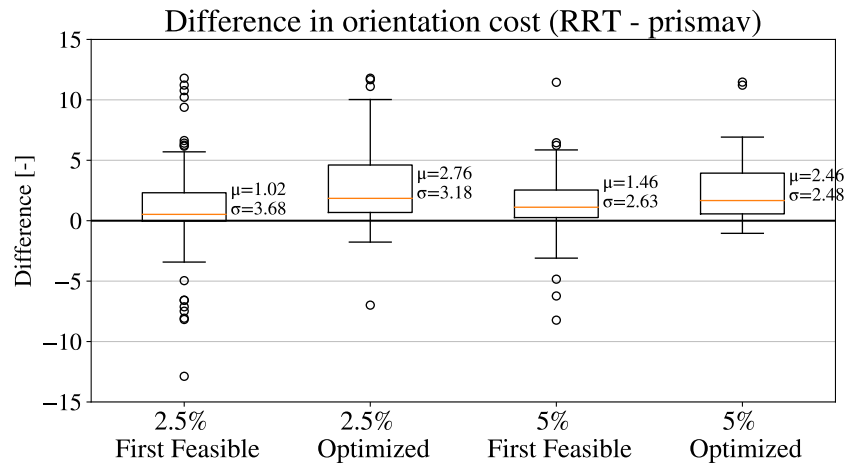


Figure 4.16: Difference in orientation cost between RRT trajectories and trajectories produced by our planner. A positive difference indicates that our planner performed better in this metric.

4.3.2 Comparison with Randomized Forests

Overall, the results obtained in this test are similar to the randomized forests. We observe the same trends in the quantitative analysis of this test as was also observed in the randomized forests.

There is however one apparent difference. In a 5% density environment, we observe that the translational distance for the first feasible trajectories seems to be significantly worse in an unstructured environment compared with the randomized forests. In the randomized forests, the distribution of the differences does not appear to change drastically between the first feasible and the optimized trajectory, indicating that the first feasible trajectories are already close to a local optimizer of the cost function. In the unstructured environment however, we observe that the distribution for the first feasible trajectories has a negative mean and median whereas the distribution for the optimized paths is skewed towards positive differences. This could indicate that the first feasible trajectories take a longer path to avoid obstacles, whereas further optimization leads to the optimizer finding short-cuts which decrease the trajectory length significantly.

4.4 Hole in a Wall

The main goal of this test was to evaluate the performance of our planner compared to a sampling-based planner in the case where there is a critical section along the trajectory where only one orientation leads to a feasible solution. The testing was conducted as follows. Each planner was tested ten times where each trial was done on a different orientation of the hole. In the unsupported test, the orientation of *PrisMAV* was chosen in the tricopter mode in both the start and end states. In the supported test, the orientation of *PrisMAV* was adjusted to either the tricopter mode or the horizontal mode depending on which orientation was closer to the orientation of the hole.

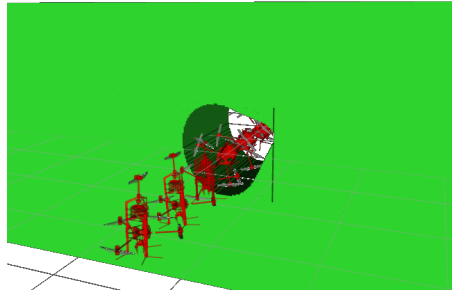
4.4.1 Results

Table 4.1 shows the success rates for the different planning problems. We observe that RRT finds no feasible path if the orientation is not adjusted to the orientation of the hole. If the orientation is adjusted, the success rate rises to 40%. On the other hand, our planner manages a success rate of 60% without adjustment of the orientation and 80% success rate with adjustment of the orientation.

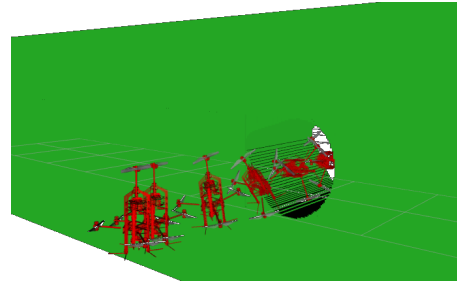
Planner	RRT	RRT adjusted orientation	<i>PrisMAV</i> planner	<i>PrisMAV</i> planner adjusted orientation
Success Rate	0/10	4/10	6/10	8/10

Table 4.1: Success rates for RRT and *PrisMAV* planner for the hole in a wall test.

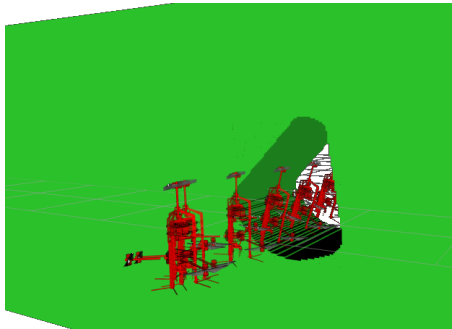
Figure 4.17 shows some trajectories produced by *PrisMAV* planner which solve the motion planning problem. Figures 4.17a to 4.17c show results for the unadjusted test where both the start and end states were left in the tricopter mode, whereas fig. 4.17d shows a path where the start and end orientations were adjusted to horizontal flight mode. The resulting initial straight line trajectory therefore is in a neighbourhood of the feasible trajectory which prevents our planner from getting stuck in a local minimum.



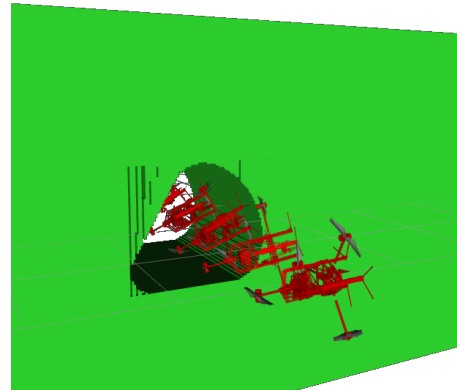
(a) *PrisMAV* has to tilt forwards in order to fit through the generated hole.



(b) Similar scenario to fig. 4.17a but in the other direction



(c) *PrisMAV* has to tilt slightly to the right in order to fit through the hole starting from the tricopter mode



(d) An exmaple where the orientation of *PrisMAV* was adjusted to be in a neighbourhood to a feasible path

Figure 4.17: A selection of trajectories produced by our planner which plan through the generated hole.

Chapter 5

Discussion

In this chapter, a discussion of the results obtained in chapter 4 is presented. First, the performance of the planner in randomly generated forests, unstructured 3D environments and the hole in a wall is discussed. Furthermore, we will provide an overview of some failure modes where our planner fails to produce a feasible path.

5.1 Performance of *PrisMAV* planner

5.1.1 Randomized Forests and Unstructured Environments

With a success rate of over 90% in both a 5% forest and in the unstructured environments we are confident that our planner works as expected. Furthermore, given that our planner finds a feasible path, we can report that our proposed planner performs comparably if not better than RRT in all tested metrics apart from planning time. If planning time is the main metric to be optimized, the stopping criterion could be relaxed to terminate the planner as soon as a feasible trajectory is found. This will however decrease the optimality of the trajectory since less time is spent in favourable orientations.

The comparable performance in the angular distance metric can be explained by the fact that our planner optimizes the orientation of the OMAV to prefer flight in tricopter or horizontal modes. For instance, if both start and end states are tilted by $\theta_{\text{start}} = \theta_{\text{end}} = 10^\circ$, our planner will push the attitude to $\theta = 0^\circ$ along the trajectory, whereas RRT does not care about the attitude of the OMAV along the trajectory.

This directly leads us to the orientation cost metric. As can be expected, our planner performs significantly better in this metric as we directly use this metric as a cost to optimize over. Optimizing this cost might lead to trajectories with increased angular length as discussed above, however this is not directly of interest as we want *PrisMAV* to prefer flight in either tricopter or horizontal mode and therefore care more about the minimization of the orientation cost as compared to minimization of angular length along the trajectory. Transition manoeuvres between flight modes are therefore constrained to take place at certain places along the trajectory instead of being stretched out over the whole trajectory. An example of this can be seen in fig. 5.1, where the trajectory transitions between the two flight modes. Also noteworthy is the fact that the RRT implementation uses uniform sampling in the space of orientations. One could develop a heuristic which prefers sampling in the horizontal or tricopter mode, but without rewiring of the random tree the resulting trajectory might jump between the tricopter and horizontal flight mode unless a steer function is used which makes sure that the orientations of neighbouring nodes are ‘close to each other’. This can however lead to a significant bottleneck of the

planner if we consider the hole in a wall test where sampling of a specific orientation is necessary.

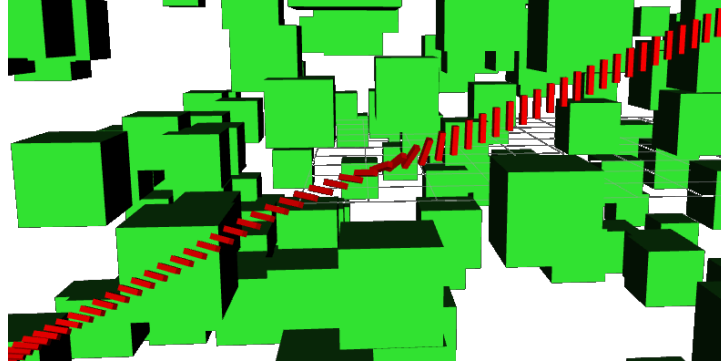


Figure 5.1: our planner (red) performs a transition manoeuvre from horizontal flight to tricopter flight

The statistics of the unstructured environment reinforce the findings from the forest test. We can conclude that our planner systematically produces paths with lower orientation cost and therefore encourages flight in our preferred flight modes. This is further verified by qualitative inspection of the trajectories which show significantly higher portions of path length in favourable flight modes with a clearly visible transition manoeuvre between said flight modes. The test in this environment allows us to conclude that our planner also produces desirable results in more difficult environments.

5.1.2 Hole in a Wall

In this test we were clearly able to show that our optimization-based planner outperforms the sampling-based planner. This is mainly due to the characteristics of a sampling-based planner. In this specific test case, there were only a very limited number of samples that would render a feasible path. Therefore, if the state space is sampled uniformly, only a small fraction of samples will drive the trajectory towards the goal state. This problem is better suited for our optimization-based planner since our planner leverages the collision gradient to push the trajectory into a feasible orientation.

5.2 Parameter Choice

Since our planner has 13 different tuneable parameters, finding the ideal combination is not straight forward. We defined an arbitrary smoothness weight and then increased the collision and orientation weights until we obtained trajectories which, by inspection, seemed feasible (i.e. no jumps in orientation, ‘smooth enough’, not in collision, etc.). The choice of parameters can drastically affect the quality of the planned paths, as well as the necessary planning time. For instance, a low stopping tolerance or a small step size can lead to long runtimes. In fig. 5.2, we can see how the choice of collision weight w_{col} affects the produced trajectory. A high collision weight can lead to longer path lengths since the collision gradient pushes the trajectory further away from obstacles.

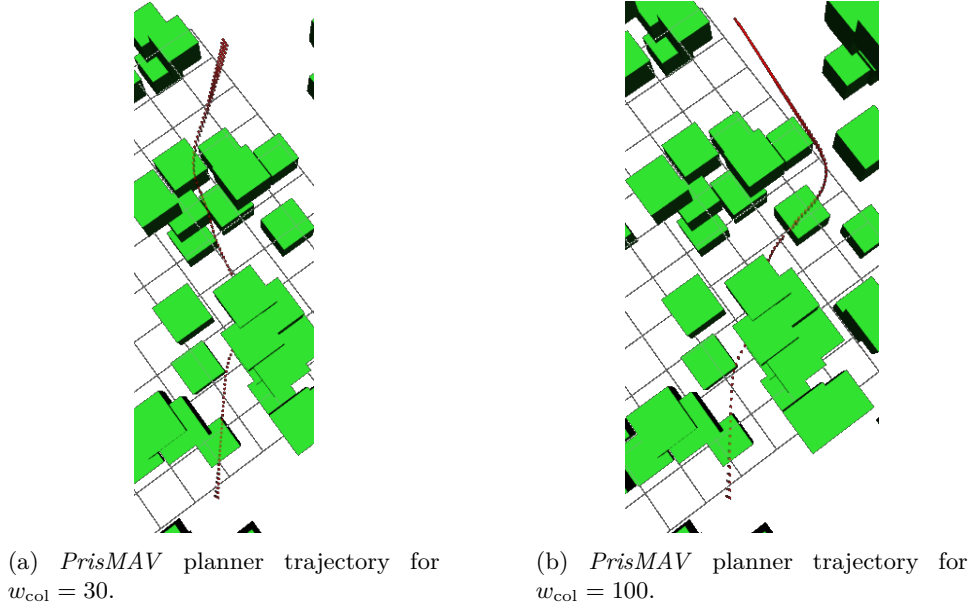


Figure 5.2: Influence of parameter choice on the planned path.

5.3 Failure Modes

Since optimization-based planners are generally not probabilistically complete, they might not find a solution even though the planning problem is feasible. Some of the failure modes which can be encountered are local minima, flat gradients or excessive smoothing. Illustrative examples are given in fig. 5.3.

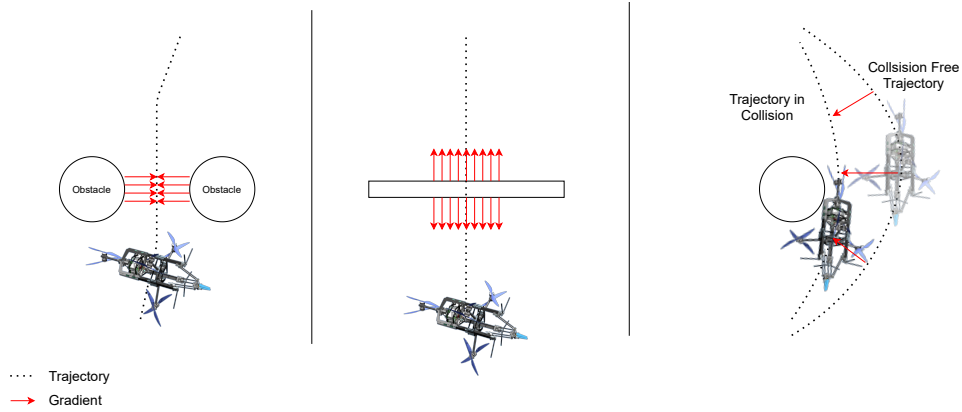


Figure 5.3: Some failure modes encountered during testing.

5.3.1 Local Minima

A local minimum encountered by an optimization-based planner can have many different forms. The ones we encountered during testing can be understood as a trajectory being stuck between two cylinders, where the trajectory is not collision-free. Since the trajectory passes through the space between the cylinders, the collision gradient tries to maximize the distance between the trajectory and the obstacles. This minimum is found in the center between the two obstacles, yet this

space is not wide enough to allow *PrisMAV* to pass through. This essentially traps the planner in a local minimum making it unable to produce a feasible path.

5.3.2 Absence of Gradients

Another example of a failure mode encountered by optimization-based planners is the absence of gradients. If we consider a trajectory which is perpendicular to a wall, there is essentially no gradient present which pushes the trajectory out of collision as the gradient is also perpendicular to the wall. In the absence of stochastic additions to the trajectory (e.g. by adding normally distributed noise to the trajectory as seen in [6]), the planner will fail to find a feasible path.

5.3.3 Excessive Smoothing

Another source of failure which we encountered is due to the cost weight scheduling as described in section 3.2.4. If we increase the smoothness cost as soon as a feasible trajectory is found we risk the trajectory becoming infeasible by choosing a smoothness weight that is too large. This leads to excessive smoothing which again brings the trajectory into collision. This can be avoided by choosing the smoothness weight in such a way that the smoothness gradient never becomes bigger than the collision gradient when the ESDF becomes 0. Then, the collision gradient will always be able to push the trajectory out of collision.

5.3.4 Insufficient Number of Optimization Variables

Another issue arises from the fact that the actual optimal trajectory length is unknown. This can pose an issue for longer trajectories, as the gradient is dependent on the chosen parameterization of the trajectory. If we consider an extreme example of $N = 3$, we only have one optimization variable. The smoothness optimum for this optimization variable obviously lies on the straight line path between the start and end point. However, let us assume that the straight line path is in collision. If the trajectory is sufficiently long (i.e. the start and end states are sufficiently far apart), the smoothness gradient will outweigh the collision gradient and we therefore will not obtain a feasible path. This effect can be mitigated to some extent by allocating more points along the trajectory. Therefore, the quality of the path is a function of the trajectory steps N , which might become computationally intractable for long trajectories as computation time is approximately of order $\mathcal{O}(N^2)$ [12]. This can be resolved by choosing a lower dimensional representation of the trajectory, such as polynomial segments as proposed in [11], which reduces the number of optimization variables.

Chapter 6

Conclusion

6.1 Summary of the Results

In this thesis, a trajectory planner for the omnidirectional aerial manipulation system *PrisMAV* was developed and verified. The planner is based on optimization-based motion planning and extends the concepts of CHOMP from robot configurations in \mathbb{R}^d to robot configurations in $SE(3)$ by means of lifting the cost function into the local tangent space of the current iterate. After computing an update step in the local tangent space, the new iterate is retracted back onto the $SE(3)$ manifold, guaranteeing that the update step produces a valid trajectory. This enables CHOMP to be used for 6-DoF planning for OMAVs. The planner can easily be extended by formulating new cost functions on position and orientation of the OMAV. This was verified by the use of an orientation cost function which penalizes flight in unfavourable flight modes and prefers flight in the horizontal and tricopter modes. The planning algorithm was tested in random forests, in unstructured 3D environments of various obstacle densities as well as tests where a trajectory through a hole in a wall had to be found. In random forests with 5% obstacle densities, a success rate of up to 93% could be achieved which drops to 46% if the density is increased to 10%.

The proposed planning algorithm consistently (probability bigger than 60% in all tested forests) produces trajectories with lower translational length. Furthermore, our planner also produces paths with lower orientation cost in over 80% of tested trajectories. The paths produced by our planner are therefore significantly better suited to our platform than the paths produced by RRT.

6.2 Outlook

In order to increase the success rate of the planner, several modifications could be made. Instead of generating an initial straight line trajectory, the planner could be initialized by a set of waypoints. This could be achieved by pre-processing the motion planning problem using a RRT-based planner to generate a set of waypoints as done in [11]. A different approach could be to decompose the problem into a lower dimensional representation using cells which are either free or in collision and then running a grid-based search algorithm such as A^* to find a set of cells which connect the start and end points. The center of these cells could then be used as waypoints for our planner.

To overcome the problem of the planner getting stuck in local minima, the planning algorithm could be extended using concepts as seen in STOMP [6], where update steps are computed as the weighted average of noisy trajectories. A difficulty here

would be to find a suitable algorithm to compute the weighted sum of orientations since averaging rotations is itself an optimization problem [18].

In this thesis, we assumed the environment to be completely known and static. In order for the planner to be usable in a real-world environment, the planner should be extended with online replanning capabilities. This would allow the planner to react to new incoming map information or avoid unforeseen events.

Furthermore, due to time constraints and no readily available planning environment, the planner could only be tested in simulation. The planner should be tested on actual hardware to ensure that the produced trajectories are also feasible in a real-world setting.

Bibliography

- [1] P. Brigger, D. Gisler, N. Ospelt, F. Bühlmann, J. Näf, M. Inauen, M. Hüsler, and M. Rubio, “Griffin Aerial Manipulation Final Report,” ETH Zürich, Zürich, Focus Project Final Report, Jun. 2021.
- [2] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968, conference Name: IEEE Transactions on Systems Science and Cybernetics.
- [3] S. LaValle, “Rapidly-exploring random trees : a new tool for path planning,” *undefined*, 1998.
- [4] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods,” in *49th IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 7681–7687, iSSN: 0191-2216.
- [5] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “CHOMP: Covariant Hamiltonian optimization for motion planning,” *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, Aug. 2013.
- [6] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574, iSSN: 1050-4729.
- [7] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequential convex optimization and convex collision checking,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014.
- [8] J. Kim and J. Ostrowski, “Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sep. 2003, pp. 2200–2205 vol.2, iSSN: 1050-4729.
- [9] T. Nägele, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, “Real-Time Motion Planning for Aerial Videography With Dynamic Obstacle Avoidance and Viewpoint Optimization,” *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, Jul. 2017, conference Name: IEEE Robotics and Automation Letters.
- [10] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525, iSSN: 1050-4729.

- [11] C. Richter, A. Bry, and N. Roy, “Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments,” in *Robotics Research*, M. Inaba and P. Corke, Eds. Cham: Springer International Publishing, 2016, vol. 114, pp. 649–666, series Title: Springer Tracts in Advanced Robotics.
- [12] 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)*, Oct. 2016, pp. 5332–5339, iSSN: 2153-0866.
- [13] M. Pantic, L. Ott, C. Cadena, R. Siegwart, and J. Nieto, “Mesh Manifold based Riemannian Motion Planning for Omnidirectional Micro Aerial Vehicles,” *arXiv:2102.10313 [cs]*, Feb. 2021, arXiv: 2102.10313.
- [14] M. Bloesch, H. Sommer, T. Laidlow, M. Burri, G. Nuetzi, P. Fankhauser, D. Bellicoso, C. Gehring, S. Leutenegger, M. Hutter, and R. Siegwart, “A Primer on the Differential Calculus of 3D Orientations,” *arXiv:1606.05285 [cs]*, Oct. 2016, arXiv: 1606.05285.
- [15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-Manifold Preintegration for Real-Time Visual-Inertial Odometry,” *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, Feb. 2017, arXiv: 1512.02363.
- [16] P.-A. Absil, C. Baker, and K. Gallivan, “Trust-Region Methods on Riemannian Manifolds,” *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 303–330, Jul. 2007.
- [17] I. A. Sucas and S. Chitta. Moveit. [Online] Available at moveit.ros.org.
- [18] M. Moakher, “Means and Averaging in the Group of Rotations,” *SIAM Journal on Matrix Analysis and Applications*, vol. 24, no. 1, pp. 1–16, Jan. 2002.