CTU

CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's thesis

Trajectory Planning for the 3D Dubins Vehicle

Jáchym Herynek

Supervisor: Ing. Petr Váňa

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Herynek Jáchym** Personal ID number: **483733**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Trajectory Planning for the 3D Dubins Vehicle

Bachelor’s thesis title in Czech:

Plánování trajektorií pro 3D Dubinsovo vozidlo

Guidelines:

1. Familiarize yourself with the Markov-Dubins problem [1], its extension for multi-goal trajectory planning [2], and the extension to 3D space with the pitch angle constraint [3].
2. Propose a method based on the non-linear optimization to find trajectories for the 3D Dubins vehicle.
3. Extend the proposed method for multi-goal trajectories in 3D, similarly as in [2] for the 2D Dubins vehicle.
4. Empirically evaluate the influence of the number of segments in the proposed discretization.
5. Empirically evaluate the developed approach's performance and compare the proposed approach with the available solutions [3, 4].

Bibliography / sources:

[1] BEVILACQUA, P., et al. A novel formalisation of the Markov-Dubins problem. In: 2020 European Control Conference (ECC). IEEE, 2020. p. 1987-1992.
[2] FREGO, Marco, et al. An Iterative Dynamic Programming Approach to the Multipoint Markov-Dubins Problem. IEEE Robotics and Automation Letters, 2020, 5.2: 2483-2490.
[3] VÁŇA, Petr, et al. Minimal 3D Dubins path with bounded curvature and pitch angle. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020. p. 8497-8503.
[4] WANG, Yu, et al. Real-time dynamic Dubins-Helix method for 3-D trajectory smoothing. IEEE Transactions on Control Systems Technology, 2014, 23.2: 730-736.

Name and workplace of bachelor’s thesis supervisor:

Ing. Petr Váňa, Department of Computer Science, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **27.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

Ing. Petr Váňa	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

I declare that the presented work was developed independently ant that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethiclal principles in the preparation of university theses.

Prague, May 21, 2021

. . . . . .

Jáchym Herynek

Acknowledgement

I would like to thank my supervisor Ing. Petr Váňa, who provided advice and insight and managed to guide me along the way even though we never even met in person. I would also like to thank the friends I met at the school who helped me throughout the studies more than they probably realize. The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" is also gratefully acknowledged.

Abstrakt

V této práci navrhujeme novou metodu pro nalezení co nejkratší cesty ve 3D s omezeným poloměrem zatáčení a úhlem náklonu založenou na nelineárním programování. Metoda rozděluje cestu na předem stanovený počet segmentů. Navržená formalizace umožňuje popis kruhových oblouků i rovných segmentů pomocí stejných rovnic. Navržená optimalizační metoda dosahuje zlepšení i oproti nejlepší dostupné heuristické metodě. Rozdíl vůči dolní mezi je zhruba o 25% nižší. Výpočetní čas je závislý na počtu segmentů, pro 100 segmentů činí přibližně 1 sekundu. Použitý Ipopt solver konverguje výrazně rychleji pro nižší počty segmentů, za cenu horších výsledků. Navrhovaná formulace je dále rozšířena tak, aby pokryla variantu problému, ve které je kromě vstupní a výstupní konfigurace zadána také množina bodů v prostoru, kterými musí výsledná cesta procházet. Výsledky pro tuto variantu problému jsou srovnatelné s výsledky vzorkovací metody použité jako referenční řešení, ale navrhovaná optimalizace jich dosáhne v podstatně kratším čase.

Klíčová slova: Dubinsova cesta, 3D Dubinsova cesta, Plánování trajektorie, Dubinsova cesta s více cíli

Abstract

In this thesis, we propose a novel non-linear programming approach to the problem of finding the shortest path in 3D space under the turning radius and pitch angle constraints. The path is divided into a number of segments in 3D space. The proposed formalisation enables to encode straight segments as well as circular arcs using the same equations. Even when initialized by the best-known heuristic method, the optimization is able to reduce the lower bound margin by about 25%, with computational times around one second using 100 segments. The computational time is greatly dependant on the selected segment count. The utilized Ipopt solver converges significantly faster for lower segment counts, at the cost of worse solutions. The proposed formulation is also extended to cover a variant of the problem, in which the path has to visit a number of locations in a given order. This extension achieves similar results as the sampling-based reference solution with much shorter computational times.

Keywords: Dubins path, 3D Dubins path, Trajectory planning, Multi-goal Dubins path

Contents

List of Figures

Chapter 1
Introduction

The goal of this thesis is to find the shortest path between two configurations in 3D space, under the constraints of maximum turning radius and minimal and maximal pitch angle. The original two-dimensional variant of this problem was first addressed by L. E. Dubins in [1] in 1957. Such a path is, for example, the path of a simplified model of a car that can go only forward or the path of a train. The three-dimensional variant of the problem could be used to describe the trajectory of a fixed-wing aerial vehicle, such as an airplane or unmanned aerial vehicle (UAV). The work of L. E. Dubins addresses the problem of finding the optimal path in a 2D plane. He proved that the optimal path consists of three segments and has either the form CCC, or CSC, where C stands for a circular segment, and S stands for a straight segment. Furthermore, the circular segments have curvature equal to the maximal curvature.

The 3D variant (see Figure 1) of the problem has come into focus later for the purposes of trajectory planning for unmanned aerial vehicles. Current heuristic methods can find feasible solutions, but the solutions are not optimal. The currently best heuristic method (to the best of the authors' knowledge) is the decoupled approach [2]. This method solves the horizontal and the vertical parts of the path separately and then joins the two results into one path.

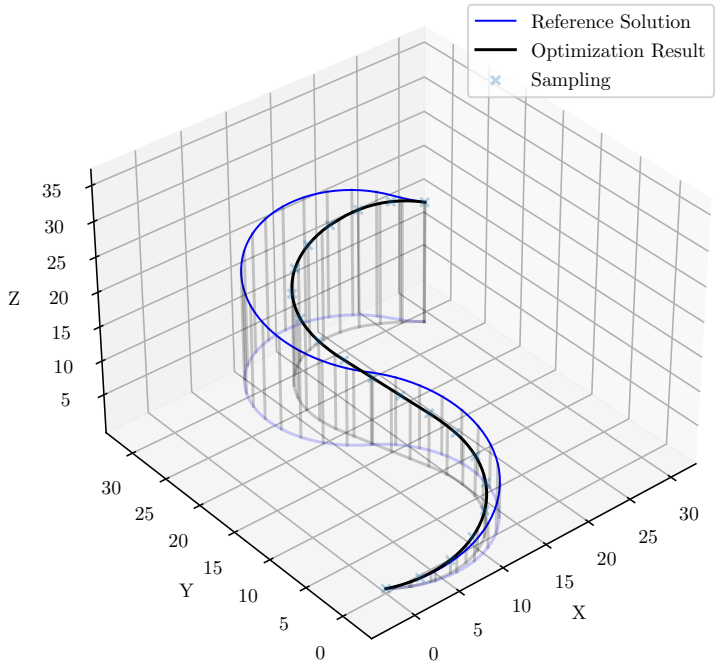


Figure 1: Example of a 3D Dubins path found by the proposed optimization (black curve) compared to reference solution (Blue curve). The proposed method is able to generate curves closer to the curvature limit.

In this work, we propose a novel non-linear programming (NLP) formulation of the problem, which is utilized to improve existing solutions. The proposed formulation divides the path into a large number of segments, where each of those segments is a circular arc in 3D space or a straight line. This

formulation can approximate any feasible curve. The precision of the approximation depends on the segment count. An example of a result of this optimization compared to a reference solution generated by the decoupled approach is shown in Figure 1. A part of this work was already submitted to the European Conference of Mobile Robots (ECMR 2021), and is currently awaiting a review [3]. The proposed NLP-based optimization approach is also extendable to cover other variants of the problem. One of those variants is presented in this work. In addition to the initial and final configurations, the curve visits additional points in a given order.

The rest of the thesis is organized as follows. Existing methods for solving this problem are presented and discussed in Chapter 2. The original 2D variant of the problem is presented. The 3D variant is presented next. Chapter 2 also covers the extension of the problem to cover multiple points and the heuristic methods used to solve this problem. Chapter 3 presents a formal formulation of the problem for the 2D and 3D variant as well as the Multipoint extension. The proposed approach is presented in Chapter 4, and a new formulation is introduced, and the formal optimization is built. Each of the constraints of this optimization is discussed more in-depth, as well as the different initialization methods. The formal optimization task is extended to cover the multipoint extension of the problem. The results of this approach are shown and discussed in depth in Chapter 5. The improvement over the reference solution is presented. Lastly, Chapter 6 offers a summary of the whole work and the results.

Chapter 2
Related Work

This chapter presents existing approaches to the Dubins path problem and its variants. The original 2D variant of the problem is presented first, followed by the extension of the problem to 3D and to cover multiple points. A short description of the optimization method used in this work is presented at the end of this chapter. The works within each section are presented in chronological order.

2.1 2D Dubins Path

The problem of finding the shortest path with a bounded curvature in a plane was first posed by Andrey Markov at the end of the nineteenth century. The optimal solution of this problem was found in 1957 by L. E. Dubins [1]. He showed that for any input configurations, the optimal path consists of three segments. Each of those segments is either a straight line or a circular arc. Moreover, the optimal path can be only one of two types: CCC or CSC (C stands for curve segment, and S stands for a straight segment). It was also shown that each of the curved segments has a turning radius equal to the minimal turning radius. An alternative proof of this using optimal control theory was presented in [4]. Other approaches to the solution of the problem were also explored. Notably, in [5], the path segments were reformulated so that both the straight segments and the circular arcs could be represented using the same formula, which enables the formulation of the problem as an optimization problem. Also in [5], a transformation of the problem was presented, which shows the symmetries of the problem.

2.2 3D Dubins Path

The Markov-Dubins problem was considered in 3D as well. With only the curvature constraint, if the two configurations are sufficiently far apart, the optimal curve has two curved segments at both ends and a straight segment between them, similarly to the 2D variant. Unlike 2D, the two curves will likely not be in the same plane. A geometric solution of this variant of the problem was presented in [6] and [7].

An extension of the Dubins car to a Dubins airplane was presented in 2007 in [8]. A new constraint for the pitch angle was introduced. This limits how steep climbs and dives a Dubins airplane can perform, which is motivated by the limitations of real-world airplanes. This work also presents an analysis of the problem. The individual input configurations are divided into three categories based on the altitude difference, and each of the categories is considered separately. The three heuristics are based on finding the necessary length of the path so that the pitch angle constraint is met and then finding a feasible two-dimensional path of that length. Note that the pitch angle of the initial and final configurations is not considered in [8] .

Later works on the subject of 3D Dubins path with the pitch angle constraint include the initial and final pitch angle as part of the problem. In [6], a numeric solution to the problem is presented. The authors claim to have found the optimal path. However, this claim is not supported by any proof. Furthermore, the computational times are very high in comparison to the other methods.

A number of heuristic approaches to the 3D Dubins path problem were proposed. In [9], the path was composed of three parts. The middle part is very similar to 2D Dubins curves (if the altitude difference is not too great), while the other two parts coped with the pitch angle change. A common method of dealing with significant altitude differences are helix curves [10], [9], [11], [8]. A helix

curve is used in order to mitigate the height difference of the two points, and a simple 2D Dubins curve can then be used for the rest of the path. In [12], a completely different approach was presented, utilizing Bézier curves to find a feasible path.

One of the most recent heuristic methods is the decoupled approach [2]. Since this method is used as a reference solution for testing the optimization presented here, it will be presented in more detail. The general idea of this method is that the final path can be split into its vertical component and its horizontal component, and those are computed separately. A 2D Dubins curve of the projection of the input configurations into the xy plane is computed as the horizontal part. Its length is then used as the x coordinate in the vertical part of the curve. The y coordinates of the two points are set to the z coordinates of the points of the initial problem. If the pitch angle constraint is met, those two paths joined together form the resulting 3D path. Otherwise, the turning radius of the horizontal path is increased, which in turn increases its length and the distance of the two points in the vertical part of the path. In [2], a lower and an upper bound for the 3D Dubins path are also presented. The results of this method are very close to the optimum, as is shown by comparison with the proposed lower bound and has low computational time. It capable of generating paths less than 1% from the lower bound when the points are far apart or (in some cases) when the altitude difference is the limiting factor. The main shortcoming of the decoupled approach is the curvature. The reason for this is that the curvature in a given point depends on both the turning radius of the horizontal part and the turning radius of the vertical part in that point. Those radii are selected and fixed for the whole curve and thus need to be selected in a way that works even for the worst case. Therefore the curvature of the path is sub-optimal and can be optimized.

2.3 Multipoint 3D Dubins Path

The multipoint variant of the problem can be seen as a sequence of 3D Dubins paths. Thus, the challenge of finding the optimal heading and pitch angles for the inner points arises. Note that most of the works and algorithms presented here are dealing with the 2D variant of this problem. Since the optimal solution to the 2D Dubins path problem is known and is fast to compute, the Multipoint 2D problem can be reduced to finding the optimal heading angles for the inner points. Arguably, this could also be said about the Multipoint 3D problem, but since there is no known optimal solution for the 3D Dubins path problem, even if there was a method that could find the optimal heading and pitch angles, it could not find the optimal path.

One of the first works addressing the problem of finding a Dubins path visiting additional locations was [13]. It considers two variants of the problem: the more common variant where the sequence of the points is known in advance, and an on-line variant, where only the directly next point is known. A simple approximation of the optimal curve is presented together with a proof of its 5.03 approximation factor.

Several other heuristics were presented in other publications. In the works [14] and [15], an alternating algorithm is proposed. This algorithm selects the heading angles so that every even segment is a straight line. A similar algorithm is the mean algorithm presented in [16]. The mean algorithm tries to minimize the usage of CCC style Dubins curves. It does so by selecting the segments close enough that a CCC style curve might be necessary and setting their heading angles so that they are straight. The remaining angles are then estimated as the mean of the direction vectors from the previous point and the direction to the next. The algorithm presented in [17] is similar to the mean algorithm. The difference is that instead of using mean and setting the segments that are too short to straight lines, it uses weighted mean. The weight used is based on the distance of the points and the turning radius. In [18] and [19], two lower bounds based on relaxation of the restrictions are presented.

Another heuristic is the sampling-based method. In this method, some angles are sampled for every point, and then for every pair of consecutive points, the length of the 2D Dubins path is com-

puted. The result is then constructed as the shortest possible method from those precalculated paths. A variation of this approach utilizing dynamic programming is presented in [20]. This method has good results and is quite fast in 2D. Another variation of this method is proposed in [21]. This algorithm does not use uniform sampling but instead explores only areas which seem promising, based on the lower bounds presented in [19] and [18]. This can greatly reduce the number of samples and thus the computational time.

The mentioned algorithms were designed for 2D, but most of them could be extended to 3D as well. In terms of time requirements, the methods based on selecting the heading angles, such as the alternating algorithm or the mean algorithm, would not change at all. On the other hand, the sampling-based methods would be slower because the pitch angle needs to be sampled in addition to the heading angle. This increases the number of necessary samples and thus the computational times.

2.4 Non-Linear Programming

The goal of this work is to formulate the 3D Dubins path as a non-linear optimization problem. A non-linear programming (NLP) problem is an optimization task of the variable $x \in \mathbb{R}^n$ where the objective function $f(\boldsymbol{x}), \mathbb{R}^n \to \mathbb{R}$ or some of the constraints $g(\boldsymbol{x})$ are non-linear. The general form of the problem could look like this:

Problem 1 (General NLP)

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}) \tag{1}$$

s.t.

$$\boldsymbol{a^L} \leq g(\boldsymbol{x}) \leq \boldsymbol{a^U} \tag{2}$$

$$\boldsymbol{x^L} \leq \boldsymbol{x} \leq \boldsymbol{x^U}, \tag{3}$$

where $\boldsymbol{x}$ are the optimization variables, $\boldsymbol{x^L}$ and $\boldsymbol{x^U}$ are the lower bounds, and $g(\boldsymbol{x})$ with the values $\boldsymbol{g^U}$ and $\boldsymbol{g^U}$ are the constraints.

There exist several optimization solvers for solving NLP problems, such as Ipopt [22], Artelys Knitro [23], NLopt [24], SCIP [25]. In this work, the Ipopt solver was used. This solver uses the Interior point method to solve NLP problems. The general idea behind this method is to include the constraints of the problem in the objective function using a barrier function. The value of this function is infinity at the edge of the feasible region, and it is undefined outside of it. This results in an auxiliary problem with optimum arbitrarily close (depending on the scaling of the barrier function) to the original optimum. As this parameter approaches zero, the optimum of the auxiliary barrier problem approaches the optimum of the original problem. The whole problem is then solved as a sequence of auxiliary barrier problems with decreasing barrier parameters. Each of those problems is initialized by the result of the previous one, which is (depending on the outer step size) very close to the optimum, and thus converges quickly.

This method can be used to solve any problem in the same form as Problem 1 [22]: Any problem defined this way can be reformulated using slack variables so that it has only equality constraints and variables with only lower bounds equal to zero. This formulation can then be used in the barrier function. This method was initially developed for convex linear problems. It is capable of solving linear problems with thousands of variables and constraints. It was so successful that it was eventually modified for non-linear and non-convex problems as well.

Chapter 3
Problem Statement

A Dubins path is the shortest path between two points with the initial and the final orientation of the vehicle given, satisfying a set of constraints. This problem can be formulated in a 2D plane (as solved by L. E. Dubins, [1]) or 3D space, e.g. [10]. While the formulations are similar, the difference is not only the space dimensionality. Both 2D and 3D formulations impose a curvature constraint. In addition, the formulation in 3D then poses the pitch angle constraint.

3.1 2D Dubins Path

The 2D Dubins path is an optimal path between two configurations in a plane, with given maximal curvature. The configuration $\boldsymbol{q}$ of the vehicle is expressed using the x and y coordinates and the heading angle ϕ, i.e., $\boldsymbol{q} = [x, y, \phi]$. The configuration space is $\mathbb{C}_{2D} = \mathbb{R}^2 \times \mathbb{S}$ with two coordinates and an angle. Let $\boldsymbol{q}_I, \boldsymbol{q}_F \in \mathbb{C}_{2D}$ be the initial and the final configuration. A Dubins path in 2D is a path between configurations $\boldsymbol{q}_I$ and $\boldsymbol{q}_F$, such that the curvature of the curve is smaller than or equal to a given maximal curvature κ_{max} at every point on the path. The motion of the Dubins vehicle along a Dubins path with a given constant forward speed v can be expressed using the following equations

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = v \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \\ u_1 \end{bmatrix}, \tag{4}$$

for control input u_1 limited by the maximal curvature κ_{max}

$$|u_1| \leq \kappa_{max}. \tag{5}$$

The goal is to find a path $\boldsymbol{\gamma}(l) : [0, \lambda] \to \mathbb{C}_{2D}$, where λ is the minimized length of the path for the given speed v. The whole problem can then be formulated as follows:

Problem 2 (Dubins Path 2D)

$$\min_{\gamma} \lambda \tag{6}$$

subject to:

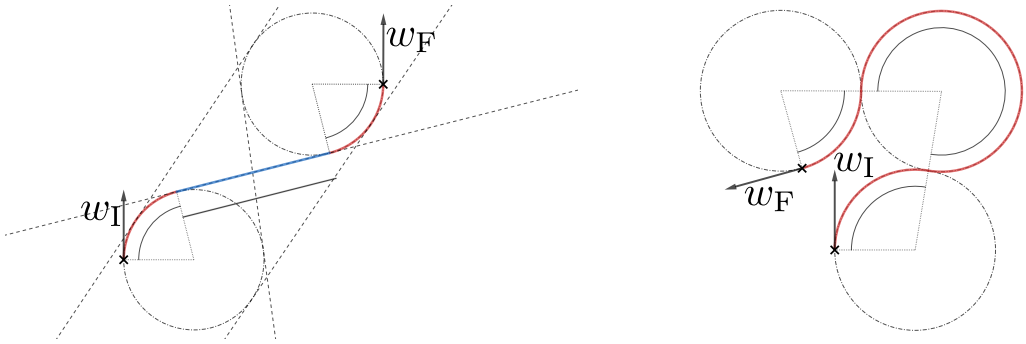
$$\gamma(0) = \boldsymbol{q}_I \tag{7}$$

$$\gamma(\lambda) = \boldsymbol{q}_F \tag{8}$$

Equations (4), (5) (9)

3.2 3D Dubins Path

The Dubins path in 3D is an extension of the Dubins path in 2D. All the constraints of the 2D Dubins path still hold, and a new constraint is introduced. In 3D, the configuration is given by three coordinates x, y and z, and two angles, heading angle ϕ, and pitch angle ψ: $\boldsymbol{q} = [x, y, z, \phi, \psi] \in \mathbb{C}_{3D}$. The 3D configuration space $\mathbb{C}_{3D} = \mathbb{R}^3 \times \mathbb{S}^2$ consists of three coordinates, the heading angle and the pitch angle. The heading angle is the angle in the xy plane, and the pitch angle is the vertical inclination of



(a) An example of a CSC 2D Dubins curve

(b) An example of a CCC 2D Dubins curve

Figure 2: Examples of Dubins curves. Curved segments are highlighted in red, straight segments in blue. The values $\boldsymbol{w}_\mathrm{I}$ and $\boldsymbol{w}_\mathrm{F}$ are the initial and final heading angles.

the vehicle. Similarly to 2D, a 3D Dubins curve is a curve from an initial configuration to a final configuration, which satisfies the same curvature constraint as a 2D Dubins curve. However, an additional constraint is introduced in the pitch angle constraint

$$\psi_k \in [\psi_{min}, \psi_{max}]. \tag{10}$$

The pitch angle constraint limits the dive/climb of the vehicle. As in 2D, we can express the motion of the vehicle given a constant forward speed v

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = v \begin{bmatrix} \cos(\phi)\cos(\psi) \\ \sin(\phi)\cos(\psi) \\ \sin(\psi) \\ u_1 \\ u_2 \end{bmatrix}, \tag{11}$$

for some control inputs u_1 and u_2 and a constant forward speed v. The two control inputs need to meet the curvature limit

$$\sqrt{u_1^2 \cos^4(\psi) + u_2^2} \leq \kappa_{max}. \tag{12}$$

Notice that the equations for x and y coordinate derivatives are very similar to the equations in 2D. The only difference is the dive/climb correction. If vertical movement is forbidden, the equations (11) and (12) are reduced to their two-dimensional equivalents, equations (4) and (5).

The resulting path is a parametric curve $\boldsymbol{\gamma}(t) : [0, \lambda] \rightarrow \mathbb{C}_{3D}$. Again, a feasible path has to have continuous first derivatives and satisfy both the turning radius constraint and the pitch angle constraint. The Dubins path is the shortest feasible path between the two input configurations. The problem can be formulated as an optimization problem, similarly to 2D:

Problem 3 (Dubins Path 3D)

$$\min_{\gamma} \lambda \tag{13}$$

subject to:

$$\gamma(0) = \boldsymbol{q}_I \tag{14}$$

$$\gamma(\lambda) = \boldsymbol{q}_F \tag{15}$$

Equations (10), (11), (12) (16)

3.3 Multipoint 3D Dubins Path

The Dubins path problem can also be extended to visit some given number of points or locations along the way. In this extension, the problem parameters are the initial and final configurations $\boldsymbol{q}_\mathrm{I}$ and $\boldsymbol{q}_\mathrm{F}$, and an ordered set of points $P = (\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}), \boldsymbol{p_1}, \ldots, \boldsymbol{p_n} \in \mathbb{R}^3$ for some $n \geq 1$ (for $n = 0$ the problem would be reduced to the 3D Dubins path). The heading and pitch angles ϕ and ψ of the points are not given. A simplified example of such a curve in 2D with one point visited is depicted in Figure 3.

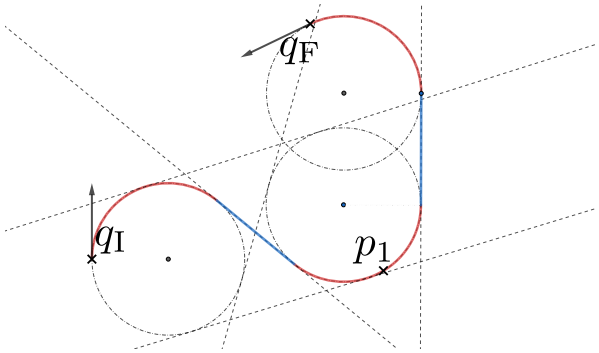


Figure 3: A simplified example of a 2D Multipoint curve, with one additional point visited. The initial and final configurations $\boldsymbol{q}_\mathrm{I}$ and $\boldsymbol{q}_\mathrm{F}$, and the visited point p_1 are labeled.

All the Dubins 3D path constraints, the turning radius constraint, and the pitch angle constraint, still apply. A feasible solution would then be a set of paths $\boldsymbol{\gamma_i}(l) : [0, \lambda_i] \to \mathbb{C}_{3\mathrm{D}}$, such that all the provided points were visited in the given order, and $\gamma_i(\lambda_i) = \gamma_{i+1}(0)$, the final configuration of one path is the initial configuration of the next. For a path γ_i, the initial position $\gamma_i(0)^{xyz} = \boldsymbol{p_i}$ and the final position $\gamma_i(\lambda_i)^{xyz} = \boldsymbol{p_{i+1}}$. The notation $\boldsymbol{a^{xyz}}$ used here represents the x, z and z component of the vector $\boldsymbol{a}$. Additionally, for γ_1 and γ_{n+1} the heading angles ϕ_I and ϕ_F, and the pitch angles ψ_I and ψ_F are also known: $\gamma_1(0) = \boldsymbol{q}_\mathrm{I}$ and $\gamma_{n+1}(\lambda_{n+1}) = \boldsymbol{q}_\mathrm{F}$. The goal is to minimize the total length:

Problem 4 (Multipoint Dubins 3D Path)

$$\min_{\gamma_i} \sum_{i=1}^{n+1} \lambda_i \tag{17}$$

Subject to:

$$\gamma_1(0) = \boldsymbol{q}_I \tag{18}$$

$$\gamma_{n+1}(\lambda_{n+1}) = \boldsymbol{q}_F \tag{19}$$

$$\gamma_i(\lambda_i) = \gamma_{i+1}(0), \quad i = 1, \ldots, n \tag{20}$$

$$\gamma_i(\lambda_i)^{xyz} = \boldsymbol{p_i}, \quad i = 1, \ldots, n \tag{21}$$

Equations (10), (11), (12) (22)

This results in $n + 1$ paths. The initial configuration of each path is the final configuration of the previous one, which means that the joined path is continuous and has continuous first derivatives. Those paths together then form the desired multipoint Dubins path. Note that this formulation of the problem assumes that the solution for the Problem 3 is known.

Chapter 4
Proposed Method

In this chapter, we propose a novel approach to the 3D Dubins Path problem that determines the problem as a Non-Linear Programming (NLP) optimization problem. For this, a new parametrization of the path is necessary.

The proposed parametrization divides the curve into a number of segments. Each of those segments is a circular arc in 3D or a straight line. This means that the curvature parameters, the turning radius, and the origin of the arc stay the same within each segment. A path is represented by a large number of segments. Any path can be approximated using this parametrization. The precision of the approximation increases with increasing segment counts.

Both curved and straight segments are well defined by their initial and final direction vectors w_i and w_{i+1}, and the distance between their ends. However, instead of the distance, a distance multiplicator d is used. This multiplicator is defined so that

$$p_{\mathrm{F}} = p_{\mathrm{I}} + \left(\frac{\boldsymbol{w_i} + \boldsymbol{w_{i+1}}}{2} \right) d_i, \tag{23}$$

where p_{F} and p_{I} are the initial and final points of the segment. Notice that for straight segments, the multiplicators are equal to the length of the segment. Figure 4 offers a simplified example of a curve split into segments.

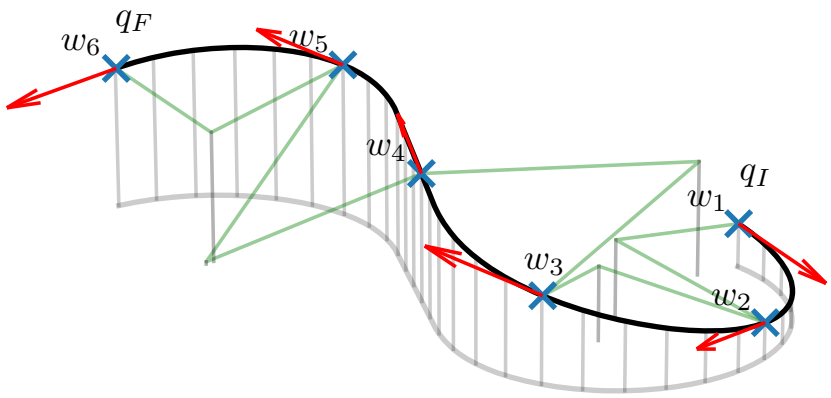


Figure 4: A simplified example of a curve between configurations $\boldsymbol{q}_{\mathrm{I}}$ and $\boldsymbol{q}_{\mathrm{F}}$ split into segments. The direction vectors w_i are shown in red. Green lines connect the ends of each segment to the origin of the turn.

From those values, the initial and final direction vectors, and the distance multiplicator, the curvature parameters of the segment can be determined. Within a segment, the path itself is then an interpolation between the two vectors situated in space. For each segment, the final direction vector is the initial direction vector of the next segment. This ensures that the curve is smooth.

The whole curve can then be defined as a sequence of such segments. This can in turn be represented as a sequence of distance multiplicators $d_i \in \mathbb{R}, d_i \geq 0, i = \quad 1, \ldots, s$ and the direction unit vectors $\boldsymbol{w_i} \in \mathbb{R}^3, \quad i = 1 \ldots, s+1$. Thus, the i-th segment is described by the vectors $\boldsymbol{w_i}$ and $\boldsymbol{w_{i+1}}$ and the distance multiplicator d_i. Given the heading angle ϕ_i and the pitch angle ψ_i, the vector $\boldsymbol{w_i}$ is

computed as follows:

$$\boldsymbol{w_i} = \begin{bmatrix} \cos(\psi_i)\cos(\phi_i) \\ \cos(\psi_i)\sin(\phi_i) \\ \sin(\psi_i) \end{bmatrix} . \tag{24}$$

The objective function is defined as the length of the path that is minimized as defined Problem (3) The length of the segment is computed from the angle change α and the curvature κ:

$$\mathcal{L}_i = \frac{\alpha_i}{\kappa_i} \tag{25}$$

The angle change α_i in the segment can be computed from the dot product of the two vectors $\boldsymbol{w_i}$ and $\boldsymbol{w_{i+1}}$:

$$\alpha_i = \arccos(\boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}) \tag{26}$$

and from α_i and the distance multiplicator d, the curvature is determined:

$$\kappa_i = 2\frac{\tan\left(\frac{\alpha_i}{2}\right)}{d_i}. \tag{27}$$

The angle change α from equation (26) can be substituted into equation (27), and the result can be transformed into

$$\kappa_i = 2\frac{\sqrt{1 - \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}}}{d_i\sqrt{1 + \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}}}. \tag{28}$$

Combining equations (25), (26), and (28), the length of a segment can be computed using the following formula:

$$\mathcal{L}_i = \frac{d_i}{2}\arccos(\boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}})\frac{\sqrt{1 + \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}}}{\sqrt{1 - \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}}}, \tag{29}$$

Notice, that the equations (25) and (29) are undefined for segments with $\kappa_i = 0$. In equation (25), no curvature would lead to division by zero. In equation (29) no curvature would lead to $\boldsymbol{w_i} = \boldsymbol{w_{i+1}}$ and $\boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}} = 1$ which would again lead to division by zero.

Based on the equations describing the individual segments, the optimization formulation can be constructed. In the following equations, the notation $\boldsymbol{w}^x$ for the x coordinate of vector $\boldsymbol{w}$ has been used. Similarly, for the y and z coordinate and the position part of a configuration $\boldsymbol{q}$: $\boldsymbol{q}^{xyz}$. Based on the previously derived equations, we can finally formulate the optimization as

Problem 5 (NLP Formulation for Dubins 3D Path)

$$\min \sum_{i=1}^{s} \mathcal{L}_i, \tag{30}$$

subject to:

$$\|\boldsymbol{w_i}\| = 1, \quad i = 1, \ldots, s+1, \tag{31}$$

$$1 - \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}} \leq \frac{d_i^2}{4}\kappa_{max}^2(1 + \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}), \quad i = 1, \ldots, s, \tag{32}$$

$$\boldsymbol{w_i^z} \in [\sin(\psi_{min}), \sin(\psi_{max})], \quad i = 1, \ldots, s+1, \tag{33}$$

$$\boldsymbol{q_I^{xyz}} + \sum_{j=1}^{s}\left(\frac{(\boldsymbol{w_j} + \boldsymbol{w_{j+1}})}{2}d_j\right) = \boldsymbol{q_F^{xyz}}, \tag{34}$$

$$\frac{d_{i-1}}{d_i} = \rho_i, \quad i = 1, \ldots, s-1, \tag{35}$$

$$\boldsymbol{w_1} = \boldsymbol{w_{init}}, \tag{36}$$

$$\boldsymbol{w_{s+1}} = \boldsymbol{w_{final}}. \tag{37}$$

The constraint (31) constrains all direction vectors to be unit vectors. This restriction does not constraint the problem in any way, and it simplifies the other constraints. The turning radius constraint (32) limits the curvature of the path so that it is less than κ_{max}, the maximal curvature given as a parameter. This constraint is derived from the equations (28). The equation was squared for convenience, and both sides were multiplied by the denominator. The pitch angle constraint (33) ensures that the pitch angle lies within the specified range, $[\psi_{min}, \psi_{max}]$.

The end point of the path is constrained by the end configuration constraint (34). It constrains the final point of the path constructed using the current values of the optimization variables. The endpoint of each segment can be computed by multiplying the mean of the two vectors by the distance multiplicator of the segment and adding that to the initial point (23). The endpoint of the whole curve is then the sum of all the means multiplied by their respective multiplicators plus the initial point of the curve $\boldsymbol{q}_\mathrm{I}$.

The ratio constraints (35) ensure that the ratios between the distance multiplicators do not change. The values ρ_i are selected upon initialization and stay constant throughout the optimization. They represent the ratios between the individual multiplicators of the segments. This constraint ensures that no segments collapse to zero length, which would effectively reduce the number of segments.

The initial and final vectors are compared to the input configurations using the initial vector constraint (36) and the final vector constraint (37). Those two constraints bind the vectors $\boldsymbol{w_1}$ and $\boldsymbol{w_{s+1}}$ to the initial and final vectors of the initial and final configurations $\boldsymbol{w}^{\boldsymbol{q}_\mathrm{I}}$ and $\boldsymbol{w}^{\boldsymbol{q}_\mathrm{F}}$, computed using the equation 24 from the input configurations $\boldsymbol{q}_\mathrm{I}$ and $\boldsymbol{q}_\mathrm{F}$.

4.1 Objective Approximation

The actual lengths of the segments as formulated above in equation (29) are undefined for straight segments and thus cannot be used in the code. One solution would be to use its Taylor series. However, for sufficiently high segment counts, a simpler approximation is also accurate enough:

$$\mathcal{L}_i \approx d_i. \tag{38}$$

With this approximation, the objective function is a sum of the distance multiplicators, which are directly used in the whole optimization. This makes it much simpler than the proper path length we are trying to minimize. The curve and the approximation of the objective function get more precise with increasing segment counts. With increasing segment counts, the angle change between consecutive vectors gets smaller. This can be expressed using the following equation:

$$\lim_{s\to\infty} \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}} = 1, \tag{39}$$

We can substitute $p = \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}$ into the equation (29). As p approaches 1, it can be shown that

$$\mathcal{L}_i \approx \lim_{p\to\,1} \frac{d_i}{2} \arccos(p) \frac{\sqrt{1+p}}{\sqrt{1-p}} = d_i. \tag{40}$$

This means that in the limit, the multiplicators d_i approach the actual length of the segments. Intuitively, this makes sense. For a straight segment, the multiplicator is equal to its length. With increasing segment count, the angle change within each segment approaches zero. Thus, its length approaches the value of the multiplicator.

4.2 Choosing Initial Optimization Values

This problem is greatly dependent on the initial values provided to the optimization. Without any initialization, the solver would very likely converge to either a sub-optimal solution or no solution at

all, and would need a lot of iterations to do so. This basically means that it is necessary to provide a high-quality solution to initialize the optimization with. If the initial solution is close to a local optimum, the optimization may converge to that point instead. Note that the initial solution does not need to be a feasible path. If some of the constraints are violated, the optimization is still likely to converge. An initialization with all the constraints violated might still converge but is much more likely to fail than one that is close to feasible.

The values of variables $d_i, i = 1 \dots, s$, and $\boldsymbol{w_i}, i = 1 \dots, s+1$ need to be initialized, and the values of constants ρ_i (the ratios between the multiplicators of the segments) found (other constants are computed from input configurations). Several options were tested: Initialization using the decoupled approach path [2], 2D Dubins path, Circle arc with or without an additional turn (left or right), and a straight line from the initial to the final point (without any consideration for the heading and pitch angles).

To compute the segment values, both the initial and final points of the segment need to be known. For this, $s+1$ equidistant configurations are sampled from the computed path. For each of those configurations, the vectors $\boldsymbol{w_i}$ are computed using the equations (24). Those vectors are then normalized, and the variables $\boldsymbol{w_i}$ are initialized with these values. From the positions of the points on the curve, the values of the multiplicators d_i are approximated, using the following equation:

$$d_i \approx 2\frac{\|\boldsymbol{q_i^{xyz}} - \boldsymbol{q_{i+1}^{xyz}}\|}{\sqrt{\boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}}}, \tag{41}$$

where the notation $\|\cdot\|$ stands for Euclidean distance. From those multiplicators, the values of the constants $\rho_i = \frac{d_{i-1}}{d_i}$ are computed.

4.2.1 Decoupled Path

The decoupled approach path is already very close to a lower bound (both the lower bound and the results are presented in [2]), and with computational times around 0.5 ms, it is fast to compute. This means that the optimization will have a high-quality initial solution to start from and will very likely finish after a small number of iterations and find the optimum.

From the results of the decoupled approach, the optimization variables are initialized as described above. This results in an almost feasible initial configuration, which can then be optimized. The infeasibility is caused by the fact that the optimization works with only circular arcs and straight lines. While most of the found decoupled path will be representable this way, there is no guarantee that all of it will. Because the configurations are sampled evenly, it is not possible to ensure that the sampled segments have uniform curvature. Even though a 2D Dubins path can always be separated into a number of segments, which have uniform curvature, the combination of two such paths generated by the decoupled approach likely cannot.

4.2.2 2D Dubins Path

The initialization using the 2D Dubins path is very similar to the initialization using the decoupled approach path, but instead of the decoupled path, the 2D Dubins path between the projection of the configurations into the xy plane is used. The sampling from this path yields the values for the x and y coordinates and part of the vector values (the heading angle is known, the pitch angle is missing). For the z coordinate and the pitch angle, a linear interpolation is used. The z coordinate at each segment end is computed as follows:

$$z_i = (q_F^z - q_I^z)\frac{i-1}{s}, \quad i = 1 \dots s+1. \tag{42}$$

The z coordinate of the direction vectors is then set accordingly. This initialization is not feasible but can work well for cases without too significant height differences, where the optimal path is likely similar to the 2D Dubins path in that it has two circular arcs at the ends and a straight line in between.

4.2.3 Other Options

Other (even simpler) initialization options were also considered, namely using a simple arc, an arc with a left or right turn added, and a straight line. The advantage of simpler initialization is that they have lower computational times. However, it is unlikely that this would outweigh the increase in computational time of the optimization itself (the results confirmed this assumption).

The idea behind the arc initialization is that it is a very simple initialization method, which can still meet some of the constraints. The constructed path was an interpolation between the initial and final direction vectors, completely disregarding the positions of the configurations. The i-th direction vector is computed using the following equation:

$$\boldsymbol{w_i} = \frac{(i-1)}{s}\boldsymbol{w}_\mathrm{I} + \frac{(s+1-i)}{s}\boldsymbol{w}_\mathrm{F}, \quad i = 1, \ldots, s+1. \tag{43}$$

This is basically a weighted mean of the two vectors. It is also possible to compute the angle first and then convert it to the vector. This has the advantage that adding (subtracting) 2π to (from) the heading angle adds an additional turn (left or right) to the path. This meets the restriction that the final direction vector has to be the same as the direction vector of the final configuration and, in many cases, also the pitch angle and curvature constraints. The distance multiplicators were set to the Euclidean distance between the two points multiplied by $\frac{3}{2s}$ ($\frac{1}{s}$ would be a straight line, the value 1.5 walks the line between a straight line and curve). Since all the segments have the same lengths, the values of ρ_i were all set to one.

Two more variants of this initialization option were also tested, with an additional left/right turn. This can essentially force the optimization solver into a certain curvature direction or provide a helix turn. This can be useful because the optimizer cannot break or construct a loop. The general idea was that if the initialization contained an unnecessary loop, it would be countered by a loop in the other direction.

The last tested initialization option was a straight line. This initialization was included mainly for completeness, as it seems to be worse than any of the other options. The points were obtained using interpolation between the two points without taking into consideration the angles or any of the constraints.

4.3 Multipoint 3D Extension

The proposed method was also extended to cover the Problem (4). In this variant of the Dubins Path problem, additional points need to be visited. An example of such a curve is presented in Figure 5 There are three big changes necessary to extend this formulation to cover the multipoint variant of the problem. The first change is that the end configuration constraint (34) is rewritten to check that all the necessary points are visited. The second change is that the ratio constraints (35) need to allow the ratios to change between two neighboring parts. The third change is the initialization, the path generated by the decoupled approach alone will no longer be usable as initialization.

For the purposes of this optimization, the path can be seen as several shorter paths with common ending points. The sequence of the visited points is then $\boldsymbol{p_i}, i = 0, \ldots, n+1$ for n additional visited points, where $\boldsymbol{p_0} = \boldsymbol{q}_\mathrm{I}^{\boldsymbol{xyz}}$ and $\boldsymbol{p_{n+1}} = \boldsymbol{q}_\mathrm{F}^{\boldsymbol{xyz}}$. We also need to split the available segments between those path parts. The values $s_i \quad i = 1, \ldots, n$ represent the number of segments preceding the point p_i. This makes it possible to use the End configuration constraint (34) almost as it is, but use it for every middle point as well. The new optimization problem looks like this:

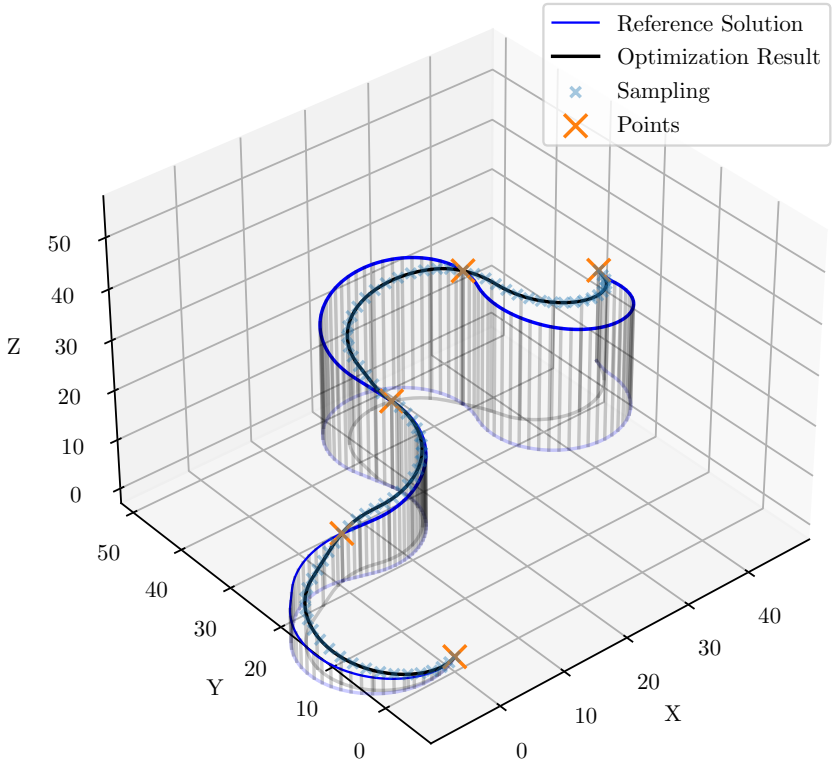


Figure 5: An example of the multipoint problem, with $n = 3$ points. Blue path is the reference solution, result of a sampling based method ran with 24 horizontal and 7 vertical samples; black line is the result of the optimization ran with 80 segments, 20 between every two consecutive points.

Problem 6 (Multipoint Dubins 3D NLP Formulation)

$$\min \sum_{i=1}^{s} \mathcal{L}_i, \tag{44}$$

subject to:

$$\|\boldsymbol{w_i}\| = 1, \quad i = 1, \ldots, s+1, \tag{45}$$

$$1 - \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}} \leq \frac{d_i^2}{4} \kappa_{max}^2 \left(1 + \boldsymbol{w_i} \cdot \boldsymbol{w_{i+1}}\right), \quad i = 1, \ldots, s, \tag{46}$$

$$\boldsymbol{w_i^z} \in [\sin(\psi_{min}), \sin(\psi_{max})], \quad i = 1, \ldots, s+1, \tag{47}$$

$$\boldsymbol{p_i} + \sum_{j=s_i}^{s_{i+1}} \left(\frac{(\boldsymbol{w_j} + \boldsymbol{w_{j+1}})}{2} d_j \right) = \boldsymbol{p_{i+1}}, \quad i = 0, \ldots, n, \tag{48}$$

$$\frac{d_{j-1}}{d_j} = \rho_j, \quad j = s_i, \ldots, s_{i+1} - 1, \quad i = 1, \ldots, n-1, \tag{49}$$

$$\boldsymbol{w_1} = \boldsymbol{q_I^w}, \tag{50}$$

$$\boldsymbol{w_{s+1}} = \boldsymbol{q_F^w}. \tag{51}$$

Only the two constraints changed: (48) and (49). The difference between (48) and the original (34) is that the equality is between two neighboring points and not the initial and final points. Similarly, within each separate path part, the ratios between the segment lengths stay the same, but the ratio of the multiplicators between two segments adjacent to a point p_i is not restricted. Note that this extension of the optimization can still be used for the 3D Dubins path problem without any visited points.

4.3.1 Multipoint Initialization

The optimization is greatly dependant on the provided initial values. Therefore it is necessary to find a suitable initialization path. The path used is a sampling-based method. This method uniformly samples k_h heading angles and k_v pitch angles. Then, all the possible combinations of the angles are computed, and the best is used. Figure 6 shows an example of such a path, with $k_h = 8$ and $k_v = 1$. Note that the values of k_h and k_v sufficient for initialization of the optimization can be significantly lower than might be otherwise used, even though better initialization obviously leads to a better solution.

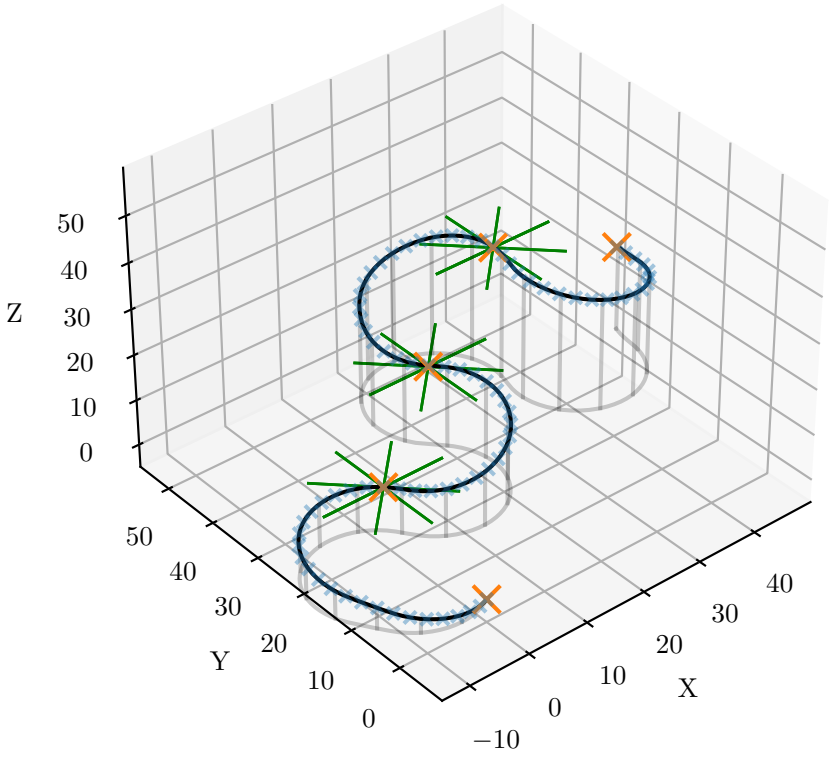


Figure 6: An example of the sampling based method used as initialization and reference. The vales used were $k_h = 8$ and $k_v = 1$. Each green ray corresponds to one sampled direction vector.

The optimization improves known solutions, which means that any other method could be used as initialization. However, for the multipoint variant of the problem, the computational time of the initialization path is usually not negligible. On the other hand, if a fast method is found, it could be used as initialization for the optimization in place of the sampling-based method and improve the results.

Chapter 5
Results

The proposed NLP-based method has been empirically evaluated, and the results are presented in this chapter. The dependences of the fail rate are further examined in Section 5.2. The results of the experiments are presented and compared with the reference solution with respect to different factors. The properties of the optimization are discussed in-depth, together with the computational times.

The optimization was implemented in the julia programming language [26] using the Ipopt optimization solver [22]. All the results were computed using a single core of the Intel Xeon Scalable Gold 6146 processor. The results are considered with respect to a reference solution, the decoupled approach [2], and the lower bound proposed in [2]. This heuristic was selected as the reference solution because it is currently the best-known method (to the best of the authors' knowledge). The real-time dynamic Dubins-Helix (RDDH) [10] method was considered as well, but the evaluation in [2] shows that the decoupled approach offers better solutions in most cases. Furthermore, the source code of the RDDH method is not available, which means that the results could be compared on only a handful of instances.

5.1 Methodology of Generating Random Instances

The proposed method was empirically evaluated on 1250 randomly generated instances. Every instance is determined by its initial and final configurations $\boldsymbol{q_I}$ and $\boldsymbol{q_F}$. For every pair of configurations, their positions on the xy plane were fixed at a selected value, while the z coordinate of the final configuration was generated with a slight variation. Heading angle ϕ and pitch angle ψ were generated randomly with uniform distribution within their respective restrictions, $\phi \in [0, 2\pi]$ and $\psi \in [-\psi_{min}, \psi_{max}]$.

For the generation of an instance, two input values were used: the end point distance E and the altitude difference multiplicator Z. From those two values the instance was generated: $\boldsymbol{q}_\mathrm{I} = [0, 0, 0, \phi_\mathrm{I}, \psi_\mathrm{I}]$ and $\boldsymbol{q}_\mathrm{F} = [x_\mathrm{F}, 0, z_\mathrm{F}, \phi_\mathrm{F}, \psi_\mathrm{F}]$. The distance between the two points on the xy plane is given in multiples of the turning radius: $x_\mathrm{F} = \frac{1}{\kappa_{max}} E$. The height was given in multiples of the height achievable by a straight line between the two points:

$$z_{max} = \tan(\psi_{max}) \, \|\boldsymbol{q}_\mathrm{I}^{\boldsymbol{xy}} - \boldsymbol{q}_\mathrm{F}^{\boldsymbol{xy}}\| Z. \tag{52}$$

A slight variation was then added to this value: $z_f \in [0.9 z_{max}, 1.1 z_{max}]$. The reason for this is that the level of limitation the pitch angle constraint poses is still subject to randomness due to the random generation of the two angles. This means that there is no value in having the same values of z_f for every instance, unlike the xy distance, which is always limiting. Adding slight variation provides a slightly wider range of possible instances while keeping the same properties.

Any instance generated this way can be characterized using the two values: the end point distance E and the height difference Z. For the experiments presented here, the values for E and Z were chosen from predetermined sets: $E \in [0.5, 1, 1.5, 2, 3, 5, 7, 10, 15, 20]$ and $Z \in [0.5, 1, 2, 3, 5]$. For every combination of the two values, 25 instances were generated, resulting in 1250 instances total. The values were chosen to cover as many cases as possible and provide data for illustration of the properties of the optimization. The range for the values of Z was chosen smaller because its influence is to an extent binary: either the pitch angle constraint is limiting, or it is not.

This way of generating the instances was selected because it allows splitting the results by one of the two properties. It allows for selection of all the results with end point distance equal to one and

have different height differences and vice versa. In most graphs the values are split by one of the two values. That means that within each column, all instances with the one value are shown, covering the whole range of the other value. For example, a column with end point distance $E = 1$ has instances with Z values from the whole set. Notice that the results for $E = 7$ and $E = 15$ are not shown in graphs split by the end point distance. Their results did not offer any new conclusions, and the inclusion rendered the figures hard to read. Note, however, that they are still taken into account in graphs split by the height difference.

Notice that since the maximal height was calculated without taking the heading and pitch angles into consideration, additional curves to increase the path length so that the pitch angle constraint is met might not have been necessary at z difference of three, in some cases even four. The threshold where the altitude difference starts to be a limiting factor is around three but is dependant on the distance of the points as well as the angles. If the points are further apart, it will take a relatively shorter distance to compensate for the heading angle, which in turn decreases the threshold.

For all of the evaluations, the minimal turning radius was set to $\rho = 10$, and the pitch angle constraints were set to $\psi_{min} = -\frac{pi}{10}$ and $\psi_{max} = \frac{pi}{10}$. Note, however, that since all the distances are directly dependant on the turning radius, any other value could have been chosen with the same results. All the values were intentionally left without units because they have no influence over the results.

5.2 Fail Rate

In some cases, the solver did not finish successfully. This section addresses those cases and studies the fail rate of different initializations and with respect to the segment count.

A result was classified as a failure if it did not, for some reason, finish successfully. The most common causes for this are converging to a point of local infeasibility, and exceeding the maximum number of iterations. Converging to a point of local infeasibility should mostly be eliminated by proper initialization but can still occur even when the path is initialized by the decoupled approach. The maximum number of iterations was set to 500 for all the test cases. This number was chosen because most of the experiments initialized with the decoupled approach finish within 100 iterations, so 500 should give enough space for the optimization to finish and still give enough space for unusually long computations. Tests that finished with a relative length of more than one (that means tests with results worse than the decoupled approach used as initialization) were considered successful, even though they did not improve the initial solution in any way. Note that in all of the figures describing improvement rate and computational times, all the failed computations were disregarded.

The fail rate was studied with respect to the initialization method, see Figure 7, and with respect to the segment count, see Figure 8. Seven different initialization methods were considered. The six presented in 4.2 and a method called here ”Best”. This method is a combination of all the other initialization methods in that it takes the results from all the other different initialization methods and uses the best of them. For most of the instances, at least one of the initialization methods led to a successful result, which is shown by the fact that the method ”Best” finished. Excluding the ”Best” initialization method, initialization using the decoupled approach path has by far the best fail rate, with an overall fail rate under 10% and under 5% for most input configurations. Initialization using 2D Dubins path can also have good results for the shorter instances, however mostly just worse than the decoupled approach path. The initialization with a straight line has surprisingly good fail rate as well, likely because it provides a path that leads to the final configuration and meets the curvature constraint. For most of the initialization options, there were instances (even though only a few) in which all other initialization options failed. This means that all of the tested initialization options offer something the other options do not.

Since the decoupled approach initialization has a significantly lower fail rate than the other initial-

ization methods, further on, the other options will be disregarded. For all the results presented here, the decoupled approach initialization was used.

The segment count seems to have no effect on the fail rate of the optimization. However, Figure 8 suggests that the fail rate of the decoupled approach initialization is slightly worse for the shorter paths.

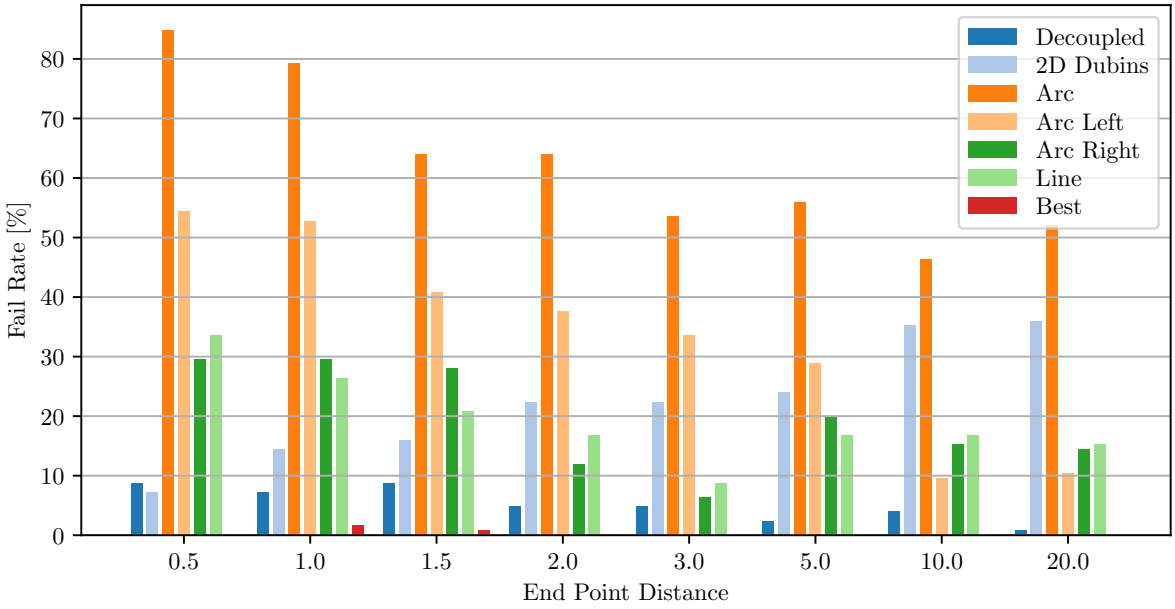


Figure 7: The fail rate of different initialization paths

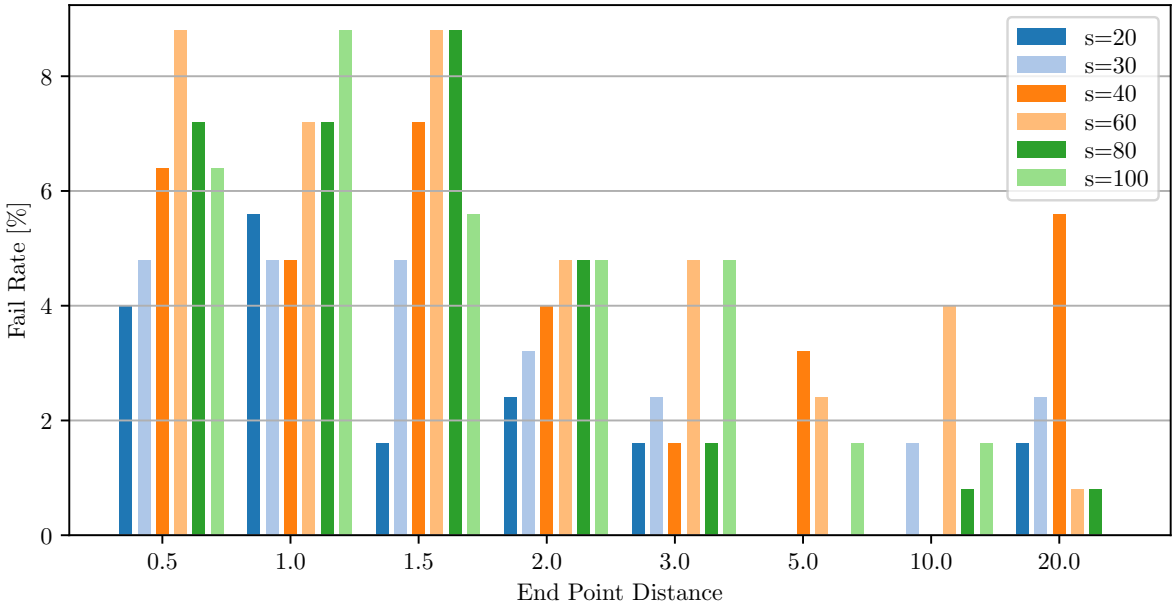


Figure 8: The fail rate of the optimization using decoupled approach for initialization with relation to the segment counts

5.3 Path Length Improvement

The generated path was compared to a reference solution generated by the decoupled approach [2]. Even though all the initialization options were tested, only the results of the optimization initialized by the decoupled approach are presented, even though in some cases, some of the other initialization methods resulted in a better solution.

The 3D Dubins path can have two main cases (that can be further distinguished). The first case is when the distance and the curvature constraints are the limiting factors. The other case is limited by the altitude difference of the two points and the pitch angle constraint. The properties of the optimization will be considered for both of these cases separately.

5.3.1 Relative to the Point Distance

This section discusses the improvement rate with respect to the distance of the two points. Distances from 1/2 up to 20 were tested (given in the multiples of the turning radius, for details on the instances see Section 5.1).

In general, the improvement is much more significant for shorter paths. This is clear from the nature of the Dubins path. In the case when the limiting factor is the distance (the height difference is low), the path mostly looks very similar to the 2D Dubins path. This means that there are two possible forms: CCC and CSC. The straight part cannot be optimized. This means that any improvement must come from the curved parts. This naturally leads to greater improvement in cases where the straight part of the path is short or missing altogether, which happens for instances with the points close to each other. This can be seen in Figure 9.

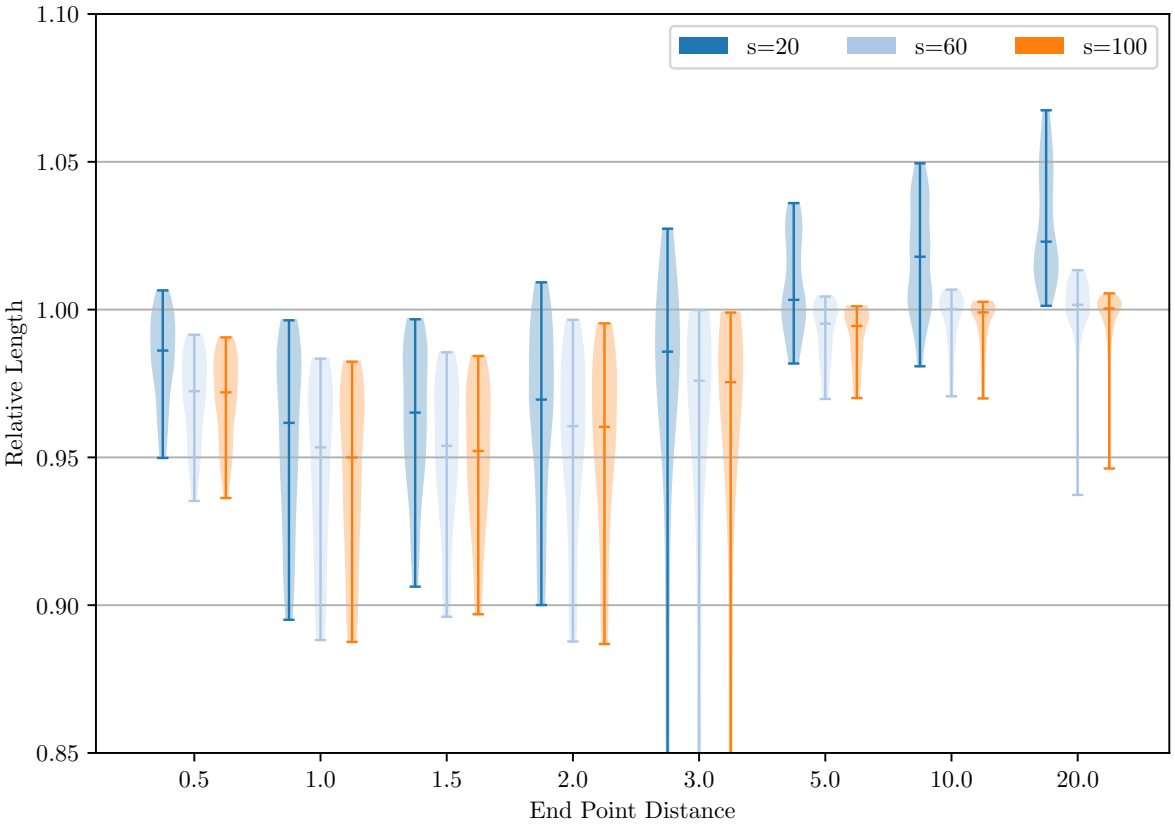


Figure 9: The results of the optimization initialized using the decoupled approach path separated by the distance between the two points (in multiples of the turning radius); horizontal bar represents median

The reason why the curved parts of the path can be improved is that the optimization approach can generate paths with curvature exactly equal to the maximal curvature. This means that the necessary direction change occurs on a shorter curve. Other methods can struggle with the curvature, especially when the pitch angle is non-zero.

Another reason for this is also a side-effect of the segmentation of the curve. When the curve is very long, the relative segment count of the curved part gets lower (in comparison to the segment count of the straight part). This means that fewer segments are used to represent the parts where they are actually useful. As a result, the computation of paths with longer straight parts will be inefficient. Furthermore, the curvature within a segment is constant. This means that if the optimal curve would end in a segment, the optimization has to find a longer curve that covers the whole segment.

A special case of the path limited by the distance occurs when the points are very close to each other (and the initial and final directions differ significantly). This means all instances which lead to a CCC style curve or to a CSC style curve with close-to-zero straight parts. In this case, the path is limited not by the distance of the points but solely by the curvature constraint. In this particular case, the improvement is greater than most other cases. Since the most improvement happens on the curved parts, the whole path gets improved. Furthermore, since the whole curve gets shorter, if there was a reverse turn, this turn might become shorter or even unnecessary, which further improves the length.

For points close to each other, the median improvement rate can be at around 4%. At a distance of 10 radii, the median of the improvement is close to zero (for the higher segment counts), and further increasing the distance of the points decreases it even more. This is, however, most likely not caused by the properties of the optimization. The shortcoming of the decoupled approach is the curvature (and some edge cases). Therefore when most of the path is a straight line, there is not much room for improvement. The decoupled approach has results very close to optimal in those cases.

5.3.2 Relative to the Height Difference

The other splitting property of the instances is the height difference. This section discusses the results with respect to this property.

Overall, according to Figure 10, the instances with medium height difference seem to yield the best results. This is caused by the fact that the segments generated by the optimization can have the maximal possible curvature regardless of the slope, which is something the decoupled approach struggles with. If the height difference is small enough so that the pitch angle constraint does not come into effect, the optimization will be able to find shorter curves that are still within the constraints.

When the height difference is the limiting factor and prolonging the path is necessary to meet the pitch angle constraint, the results get worse. There are two big reasons for this. The first is that the decoupled approximate is able to approximate the height profile of the optimal path and thus can have nearly optimal results in those cases. Outside of some edge cases, there is very small (or even no) room for improvement. The second reason is that when the height difference is the limiting factor, there are infinitely many feasible curves with the optimal length, and there may not be a single curve the optimization can converge to. Furthermore, for instances with the initial and final points far from each other, the curves have a relatively small turning radius, which results in even more options.

On the other hand, when the height difference is too small, the decoupled approach can have results close to the optimum as well. The shortcoming of the decoupled approach is the approximation of the curvature, which cannot incorporate the slope into the turning radius. When the slope is smaller, the problem is lesser as well.

An interesting edge case occurs when the altitude difference is very small, and the points are close to each other. For instances with very low or zero height difference, a 2D Dubins curve could be almost feasible. However, that does not mean that this path is the optimal solution. If the points are close to each other, there might be a path that can utilize a dive to perform the necessary turn without going around. The optimization can find this path, but the initialization using the decoupled approach

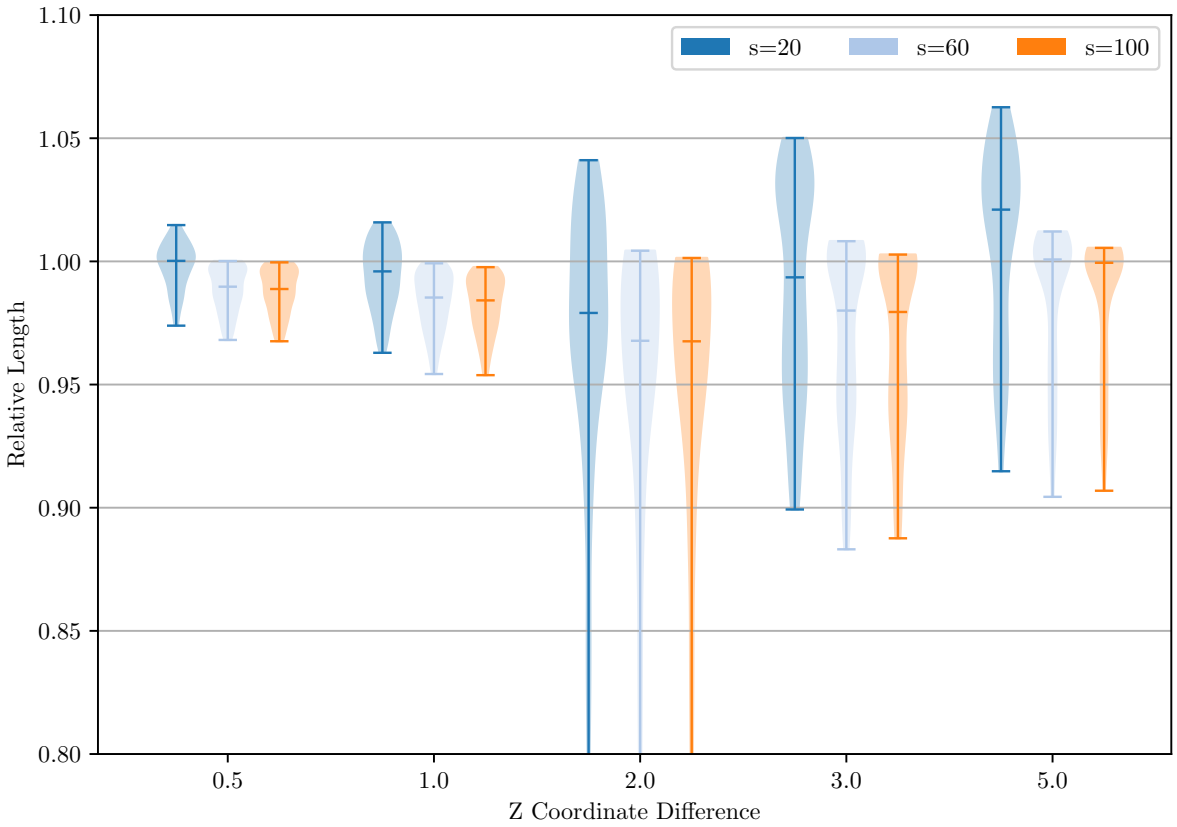
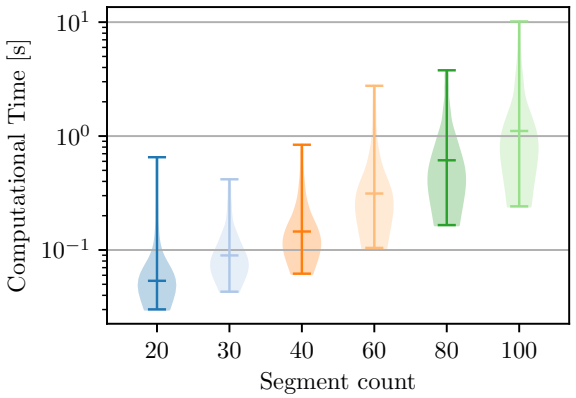
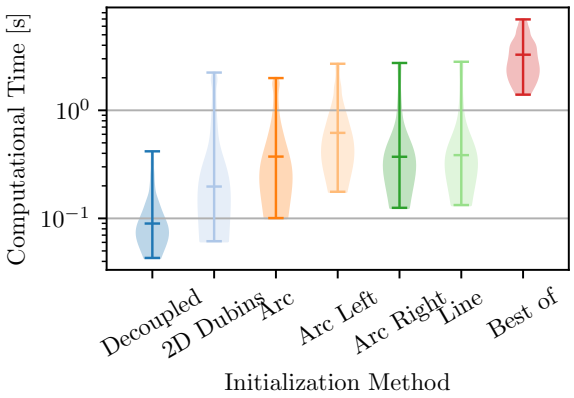


Figure 10: The improvement rate of the optimization initialized using the decoupled approach with relation to the height difference between the two points (see section Data 5.1)



(a) The computational time of the optimization initialized using the decoupled approach with with different segment counts



(b) The comparison of computational times of the different initialization methods across all segment counts

Figure 11: The computational times of the optimization

may land in the 2D Dubins path. This seems to be a local minimum, which the optimization solver cannot improve. In that case, it is necessary to use another initialization.

5.4 Computational Time

At around 0.1s and up to over 1s for higher segment counts, the optimization is much slower than the decoupled approach reference method (which has times around 0.5 ms), but is still quite fast, at least for lower segment counts.

As Figure 11b shows, out of all the different tested initialization methods, the initialization using the decoupled approach is clearly the fastest. This is no surprise, as the optimization is initialized with an almost feasible path that is very close to (a local) optimum already. The other initialization options are all similarly slow, with the Dubins path initialization being slightly faster than the others.

The main influence over the computational time is the segment count. Figure 11a shows that increasing the segment count leads to an exponential computational time increase. This is not surprising, as each segment adds more variables and more constraints to the optimization task.

5.5 The Influence of the Segment Count

This section discusses how the properties of the optimization change with different segment counts. The segment count is the number of segments used to approximate the curve. The results show (see Figure 9 and Figure 10) that increasing the segment count leads to more optimal paths. This is expectable since the segment count is directly responsible for the precision of the used approximations. Increasing segment count helps to smooth out the areas where the optimization struggles. Especially longer paths, where the fact that within a segment, the curvature is fixed, leads to curves with suboptimal curvature because the limited segment count forces longer turns than necessary. The difference between lower segment counts (around 20) and higher (60 or more) is rather significant, at more than 1%. The differences between the higher segment counts are much lower than between the lower ones. This offers a trade-off. While increasing the segment count improves the results, it also causes an exponential increase in computational time.

5.6 Multipoint 3D Extension

An extension of the optimization was also tested, the multipoint problem. This version of the problem is harder to test for several reasons. One reason is that the multipoint version (and especially the reference solution) take much more time to compute. The second reason is data generation. While the single segment had the point distance and height difference to work with, the multipoint adds the point count and mixing the height differences and point distances between the individual points of the path. In the results presented here, the points were generated with equal distance and given order. This approach has the advantage of clearly showing the properties of the optimization with respect to the distance between the points.

The instances were generated in a similar manner as the instances for the 3D Dubins path. A first point was created with random heading angle ϕ and pitch angle ψ. For every additional point, the direction from the previous point was selected randomly. The distance was set to be the same for all the points, and the height difference was similar for all the points (generated in the same way as the height of the instances for the single segment variant, with a possibility of being negative). This results in instances split by the same properties as the instances for the 3D Dubins path.

For the multipoint reference, a simple implementation (using dynamic programming) of a sampling-based approach with 24 samples for the heading angle and 7 for the pitch angle was used. Those values were chosen because they are big enough to yield good results but small enough so that the computation of the reference does not take too long. The heuristic is the same as the one used for the initialization ran with higher sample counts. The sample counts used for the initialization were 8 samples for the heading angle and 1 for the pitch angle. With this low sample counts, the resulting curve is obviously sub-optimal, but as initialization for the optimization, it is sufficient.

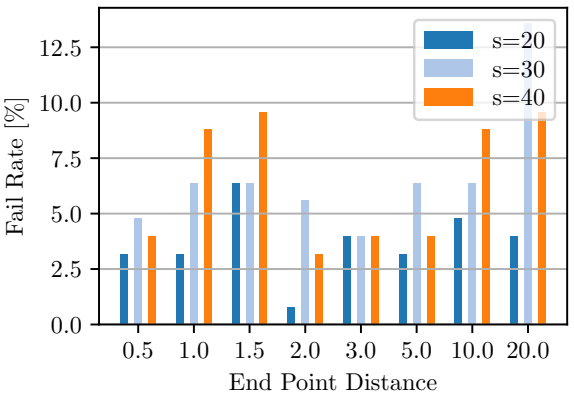
The results for the multipoint extension look very similar to the 3D Dubins path results, with two big differences. The first is that the multipoint is way more likely to end up with a solution worse than the reference. This is caused by the fact that since the optimization does not start in the reference solution, there is a possibility of converging to a local minimum and stopping the optimization. This could obviously be solved by using the reference as the initial solution, but the computational time of the reference is by no means negligible. Despite this, as Figure 12b shows, the median relative length of the results of the optimization is below one for the short paths and around one for the longer ones. Note that the optimization has a much higher fail rate for longer paths.

The second big difference is the computational times. As Figure 14 shows, for the cases with more than one point the optimization managed to finish much faster than the reference in most cases. The reference is much faster for the single point, because there is only a single configuration to sample. Similarly to the 3D Dubins paths, the computational time grows exponentially with increasing segment counts. The computational time seems to have a linear dependence on the visited point count for both the reference and the optimization.

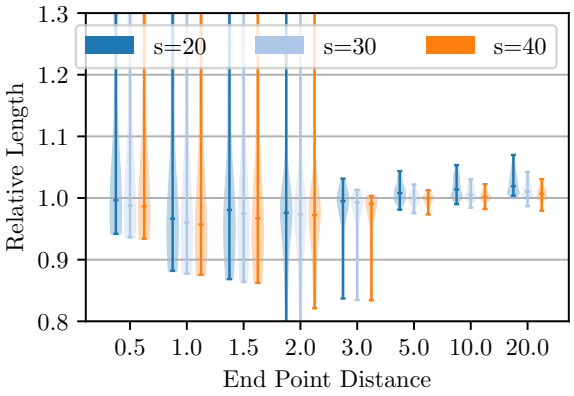
Altogether, while the optimization has mostly similar results as the used reference (with outliers in both directions), it achieves those results much faster than the sampling-based method. It is important to note that those results are highly dependant on the initialization method. The same method as the reference could have been used for initialization. That would very likely lead to much fewer results worse than the initialization, but the optimization would then be slower than the reference.

5.7 Discussion

Overall, this non-linear optimization approach can achieve results much closer to the lower bound (the lower bound presented in [2] was used) than the reference solution, by up to 25% of the margin. The improvement is about 4%, in terms of the relative path lengths, for favorable paths. As Figure 16 shows, the decoupled approach can, in some cases, generate paths that are provably optimal, and those cannot get improved. On the other hand, for the cases with which the decoupled approach struggles,



(a) The fail rate of the multipoint extension with respect to the point distance of the individual points



(b) The improvement rate of the multipoint extension with respect to the point distance of the individual points

the optimization achieved a consistent improvement with reasonably low fail rates. If only the cases when the reference solution has a margin bigger than 1% are considered, the optimization offers a significant improvement (in terms of the lower bound margin) in almost all the cases. Additionally, the optimization managed to get reasonable results for the edge cases with which the decoupled approach struggles. Also, note that the lower bound is by no means perfect. This means that a margin of 5% does not necessarily imply a possibility of improvement.

The results of the decoupled approach are close to the optimum, when either the distance, or the height difference are the limiting factor. In those cases, the longer paths and the paths with a big height difference, there is little room for improvement. However, the optimization is able to use the available curvature much more efficiently than the decoupled approach. Thanks to this, for paths where the height difference is not too constraining and the initial and final configurations are not too far apart, the optimization can compute paths that are shorter by 3-5% or more. The Figure 15a and Figure 15b illustrate this very well. (Note that the examples were handpicked to illustrate this point, the improvement rate in both the figure is above average) The curvature plots compare the curvatures of the optimization generated paths and the paths generated by the decoupled approach.

Figure 15a shows an example where the height difference was the limiting factor. The curvature plot of the optimization in Figure 15c seems, at first glance, a bit strange, but it makes sense. The middle part of the curve can be any path long enough to satisfy the pitch angle constraint. The optimization found a path with the form CSCSC, while the decoupled approach had to use a 2D Dubins path, which (in this particular case) was suboptimal.

The second figure, Figure 15b, illustrates the case where the limiting factors are the distance and the curvature. The Figure 15d and Figure 15f clearly show the main advantage of the optimization. The decoupled approach can achieve the full curvature only when the curvature is maximal on both parts of the path (the horizontal and the vertical), while the optimization can maximize the curvature at any point necessary.

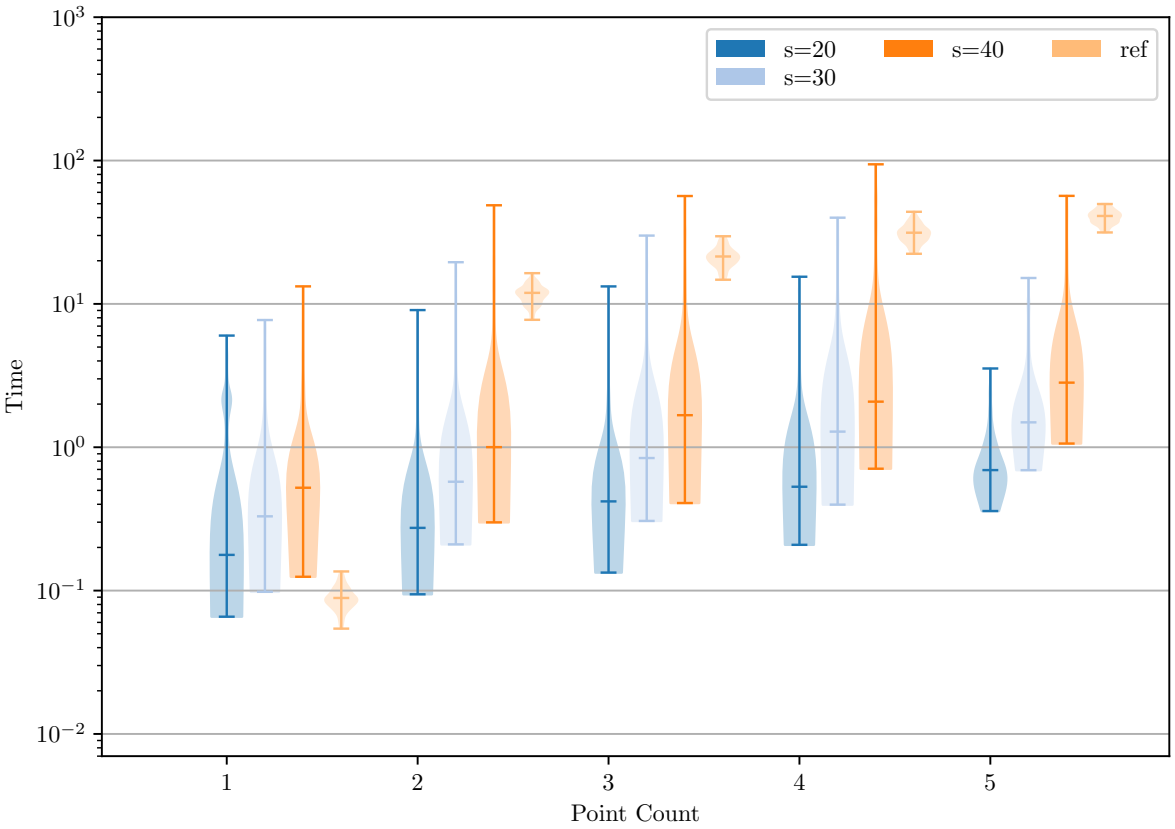
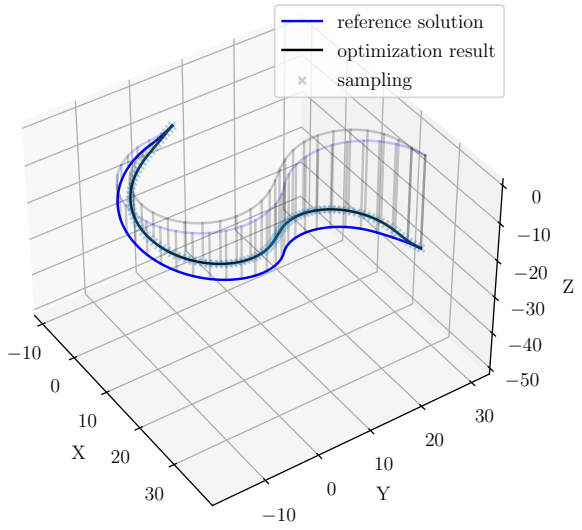
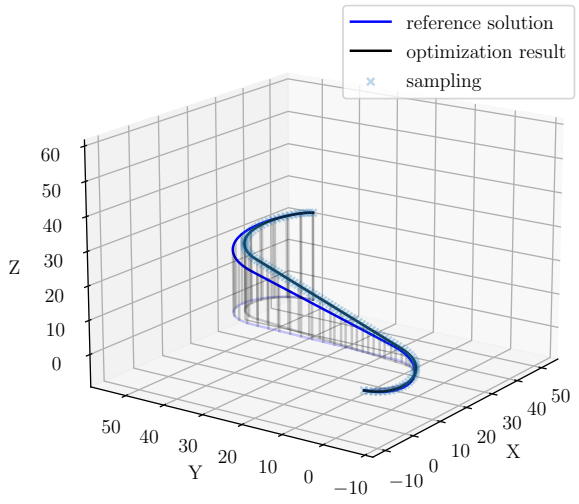


Figure 13: The computational time with respect to the number of points

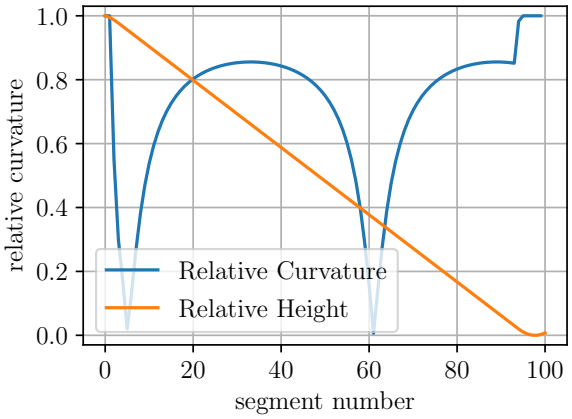
Figure 14: The parameters of the multipoint extension: the fail rate and the improvement rate



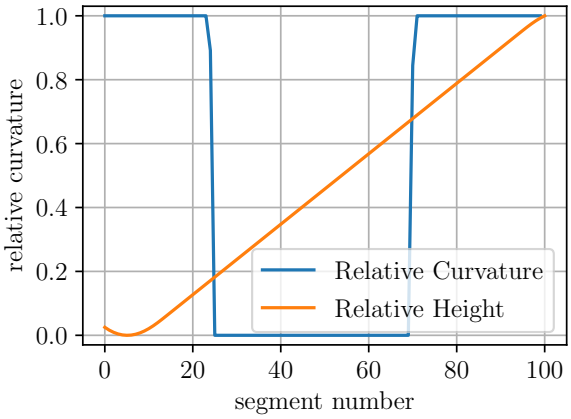
(a) Comparison of the two paths, the optimization result and the decoupled approach, path limited by the height difference, for a curve limited by the altitude difference.



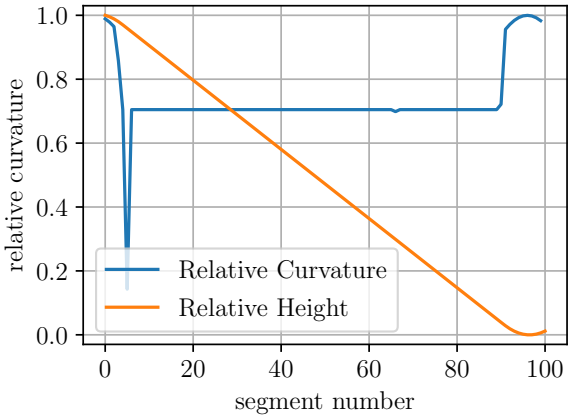
(b) Comparison of the two paths, the optimization result and the decoupled approach, path limited by the point distance, for a curve limited by the distance of the points.



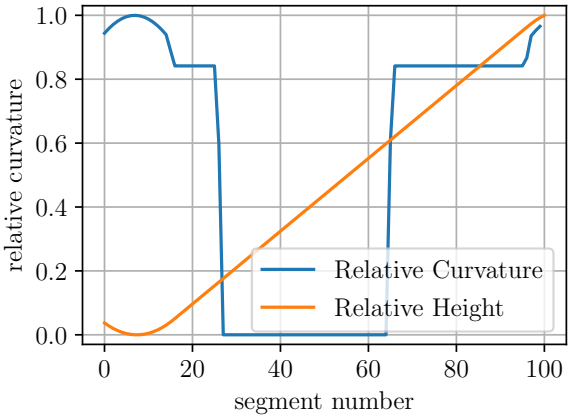
(c) The curvature of the result of the optimization



(d) The curvature of the result of the optimization

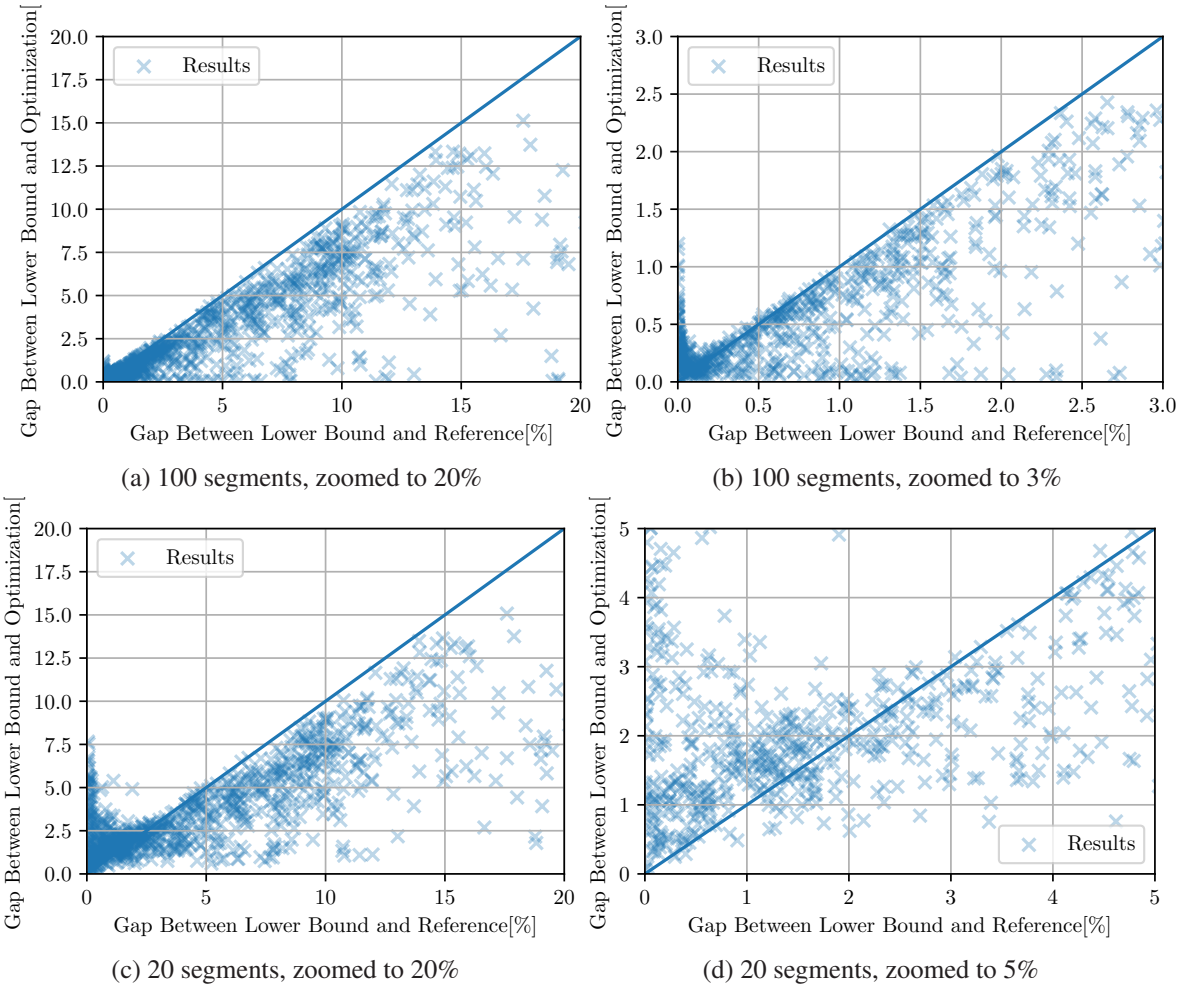


(e) The curvature of the path generated by the decoupled approach



(f) The curvature of the path generated by the decoupled approach

Figure 15: Examples of the results. For both examples, the comparison of the path generated by the proposed method and the reference solution are presented, together with the curvature profiles (the blue curve) and the height profiles (the orange curve) are presented, for both, the proposed method, and the decoupled approach reference solution.



(a) 100 segments, zoomed to 20%

(b) 100 segments, zoomed to 3%

(c) 20 segments, zoomed to 20%

(d) 20 segments, zoomed to 5%

Figure 16: The results with respect to the margin to the lower bound. Every cross represents a single solution, its position on the x axis is the margin of the reference solution, its position on the y axis represents the margin of the Optimization results. Every point below the diagonal has better Optimization results than the reference solution.

Chapter 6
Conclusion

In this work, a novel non-linear programming formulation of the 3D Dubins path problem was presented. The final curve was modeled as a sequence of straight or circular segments. This can be represented using only the heading vectors at the two endpoints of the segments and the distance of those two points. The curvature and pitch angle constraint can then be formulated based on the geometric properties of the vectors. An extension of the optimization to cover the Multipoint variant of the problem was also presented. Thanks to the nature of the optimization formulation of the original problem, the extension differs slightly and can be used to compute both. In some sense, it is more an improved version rather than an extension.

The proposed optimization formulation was implemented in the julia programming language using the Ipopt optimization solver. The implementation was then empirically evaluated on randomly generated instances and compared with known methods: the decoupled approach for the 3D Dubins path and a sampling-based method utilizing dynamic programming for the Multipoint variant of the problem. It was shown that for the 3D Dubins path problem, the paths generated by the optimization can follow the constraints more closely than the heuristic methods, which leads to more optimal paths. The results were considered with respect to the different properties of the random instances. It was shown that while the decoupled approach can generate paths very close to the optimum, the optimization can consistently reduce the optimality gap by an additional 25%, which corresponds to an improvement of 3-4% path length for favorable paths. Similar relations hold for the multipoint variant of the problem. There are two big differences: greater variance and decreased computational time, relatively to the used reference.

In general, the proposed method can improve known solutions in that it can construct curves that have the maximal possible curvature and thus are shorter. However, the optimization is greatly dependant on proper initialization. Several initialization options were tested, and in most cases, the decoupled approach initialization produced the best results. There are, however, cases when a simpler initialization resulted in shorter paths. Initialization is even more important to the Multipoint variant of the problem, where the computational times for the heuristic methods, and thus the computational time of the initialization, are much larger.

The extension of the proposed method to the Multipoint variant of the problem generated paths shorter by about 2-3% but has significantly lower computational times compared to the sampling-based reference. For instances with five additional points visited, the median computational time of the proposed method was around two seconds, while the sampling-based method needed around twenty seconds.

References

[1] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[2] P. Váňa, A. Alves Neto, J. Faigl, and D. G. Macharet. Minimal 3d dubins path with bounded curvature and pitch angle. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8497–8503, 2020.

[3] J. Herynek, P. Váňa, and J. Faigl. Finding dubins path with pitch angle constraint using non-linear optimization (in review). In *2021 European Conference on Mobile Robots (ECMR)*, 2021.

[4] J. . Boissonnat, A. Cerezo, and J. Leblond. Shortest paths of bounded curvature in the plane. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2315–2320 vol.3, 1992.

[5] P. Bevilacqua, M. Frego, D. Fontanelli, and L. Palopoli. A novel formalisation of the markov-dubins problem. In *2020 European Control Conference (ECC)*, pages 1987–1992, 2020.

[6] S. Hota and D. Ghose. Optimal path planning for an aerial vehicle in 3d space. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4902–4907, 2010.

[7] S. Hota and D. Ghose. Optimal geometrical path in 3d with curvature constraint. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 113–118, 2010.

[8] H. Chitsaz and S. M. LaValle. Time-optimal paths for a dubins airplane. In *2007 46th IEEE Conference on Decision and Control*, pages 2379–2384, 2007.

[9] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. De Lellis, and A. Pironti. Path generation and tracking in 3-d for uavs. *IEEE Transactions on Control Systems Technology*, 17(4):980–988, 2009.

[10] Y. Wang, S. Wang, M. Tan, C. Zhou, and Q. Wei. Real-time dynamic dubins-helix method for 3-d trajectory smoothing. *IEEE Transactions on Control Systems Technology*, 23(2):730–736, 2015.

[11] Mark Owen, Randal W. Beard, and Timothy W. McLain. *Implementing Dubins Airplane Paths on Fixed-Wing UAVs**, pages 1677–1701. Springer Netherlands, Dordrecht, 2015.

[12] A. A. Neto, D. G. Macharet, and M. F. M. Campos. 3d path planning with continuous bounded curvature and pitch angle profiles using 7th order curves. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4923–4928, 2015.

[13] Jae-ha Lee, Otfried Cheong, Woo-Cheol Kwon, Sung Shin, and Kyung-Yong Chwa. Approximation of curvature-constrained shortest paths through a sequence of points. In *Algorithms - ESA 2000*, volume 1879, 09 2000.

[14] K. Savla, E. Frazzoli, and F. Bullo. On the point-to-point and traveling salesperson problems for dubins' vehicle. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 786–791 vol. 2, 2005.

[15] K. Savla, E. Frazzoli, and F. Bullo. Traveling salesperson problems for the dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391, 2008.

[16] D. G. Macharet, A. A. Neto, V. F. da Camara Neto, and M. F. M. Campos. Data gathering tour optimization for dubins' vehicles. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[17] Douglas Macharet and Mario Campos. An orientation assignment heuristic to the dubins traveling salesman problem. In *IBERAMIA*, pages 457–468, 11 2014.

[18] S. Manyam, S. Rathinam, and D. Casbeer. Dubins paths through a sequence of points: Lower and upper bounds. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 284–291, 2016.

[19] Satyanarayana Manyam, Sivakumar Rathinam, David Casbeer, and Eloy García. Tightly bounding the shortest dubins paths through a sequence of points. *Journal of Intelligent & Robotic Systems*, 88, 12 2017.

[20] M. Frego, P. Bevilacqua, E. Saccon, L. Palopoli, and D. Fontanelli. An iterative dynamic programming approach to the multipoint markov-dubins problem. *IEEE Robotics and Automation Letters*, 5(2):2483–2490, 2020.

[21] J. Faigl, P. Váňa, M. Saska, T. Báča, and V. Spurný. On solution of the dubins touring problem. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2017.

[22] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[23] Byrd Richard H., Nocedal Jorge, Waltz Richard A., and Roma M. *Knitro: An Integrated Package for Nonlinear Optimization*, pages 35–59. Springer US, Boston, MA, 2006.

[24] Steven G. Johnson. The nlopt nonlinear-optimization package. `https://nlopt.readthedocs.io/en/latest/`. Accessed: 2021-5-21.

[25] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. Technical report, Optimization Online, March 2020.

[26] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.