




Article

A Non-Smooth Numerical Optimization Approach to the Three-Point Dubins Problem (3PDP)

Mattia Piazza ^{1,†} , Enrico Bertolazzi ^{1,†}  and Marco Frego ^{2,†,*} 

¹ Department of Industrial Engineering, University of Trento, Via Sommarive 9, 38123 Trento, Italy; mattia.piazza@unitn.it (M.P.); enrico.bertolazzi@unitn.it (E.B.)

² Faculty of Engineering, Free University of Bozen-Bolzano, Piazza Università 1, 39100 Bolzano, Italy

* Correspondence: marco.frego@unibz.it

† These authors contributed equally to this work.

Abstract: This paper introduces a novel non-smooth numerical optimization approach for solving the Three-Point Dubins Problem (3PDP). The 3PDP requires determining the shortest path of bounded curvature that connects given initial and final positions and orientations while traversing a specified waypoint. The inherent discontinuity of this problem precludes the use of conventional optimization algorithms. We propose two innovative methods specifically designed to address this challenge. These methods not only effectively solve the 3PDP but also offer significant computational efficiency improvements over existing state-of-the-art techniques. Our contributions include the formulation of these new algorithms, a detailed analysis of their theoretical foundations, and their implementation. Additionally, we provide a thorough comparison with current leading approaches, demonstrating the superior performance of our methods in terms of accuracy and computational speed. This work advances the field of path planning in robotics, providing practical solutions for applications requiring efficient and precise motion planning.

Keywords: Dubins vehicle; path planning; Dubins traveling salesman problem; trigonometry; non-smooth optimization



Citation: Piazza, M.; Bertolazzi, E.; Frego, M. A Non-Smooth Numerical Optimization Approach to the Three-Point Dubins Problem (3PDP). *Algorithms* **2024**, *17*, 350. <https://doi.org/10.3390/a17080350>

Academic Editors: Sona Taheri, Kaisa Joki and Napsu Karmitsa

Received: 9 July 2024

Revised: 6 August 2024

Accepted: 8 August 2024

Published: 10 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

One of the central problems in robotics is *motion*, or *path planning*, which accounts for computing a feasible path (possibly an optimal one, for some design criterion) for a robot moving from a starting configuration to a final one. A configuration can be just the position, but can also contain additional information like the orientation or the curvature. A feasible path can be required to pass along fixed or moving obstacles or to follow internal waypoints [1–3]. Other physical limitations that restrict the path's shape are differential constraints, called *nonholonomic*, which characterize a vehicle's trajectories. An important one is the requirement of *bounded curvature*, which asks that the path does not have excessive sharp turns because the robot can steer up to a minimum turning radius ρ_{\min} , which is defined as the inverse of the maximum curvature of the path, $\kappa_{\max} = \frac{1}{\rho_{\min}}$. As an optimality criterion, minimizing the path length is very common. Another common assumption is that the vehicle travels at a constant speed; thus, the shortest path is equivalent to the other important optimization functional, the minimum time. However, this can be extended and completed by combining motion primitives [4–6].

The mathematical model containing all these design requirements is the Markov–Dubins problem, which is defined formally as follows. A *Markov–Dubins path* (MDP) [7] between two assigned points in \mathbb{R}^2 , $\mathbf{P}_i = (x_i, y_i)$ and $\mathbf{P}_f = (x_f, y_f)$, with angles θ_i and θ_f is the C^1 and piecewise C^2 path of minimum length $\gamma : [0, L] \rightarrow \mathbb{R}^2$ such that the absolute value of the curvature of the path γ , at almost every point, is not greater than $\kappa_{\max} > 0$; see [8]. The problem, originally formulated by Markov in 1889 was solved only in 1957

by Dubins [9], who proved that there are six extremal solutions to the associated optimal control problem:

$$\begin{aligned} \text{Minimize } L &= \int_0^L 1 \, d\ell \quad \text{subject to } |u(\ell)| \leq \kappa_{\max}, \\ x'(\ell) &= \cos(\theta(\ell)), \quad y'(\ell) = \sin(\theta(\ell)), \quad \theta'(\ell) = u(\ell), \\ x(0) &= x_i, \quad y(0) = y_i, \quad \theta(0) = \vartheta_i, \\ x(L) &= x_f, \quad y(L) = y_f, \quad \theta(L) = \vartheta_f. \end{aligned} \tag{1}$$

Each extremal is composed of at most three internal arcs. This sequence of curves can be written as words: *LSR, RLR, LSL*, etc., where *L*, *S*, and *R* represent a left turn, straight segment, and right turn, respectively. A more compact notation is obtained by writing *C* for an arc of a circle and *S* for a straight line segment. Not all possible combinations of curves are optimal; it has been proved that the optimal words are only [8,10] *CSC, CS, SC, S, CCC, CC*, and *C*. Thus, there are 15 possible combinations: *LSL, LSR, RSL, RSR, RLR, LRL, LS, RS, SL, SR, S, LR, RL, L, R*.

In practice, only the first six words are considered; the others can be obtained by setting some of their arcs to zero. However, there is no way to know in advance which of the six words is the shortest, and usually, the solution is found through trial-and-error (or brute force) enumeration of all six. A recent method to avoid this trial-and-error procedure employs Machine Learning to train an AI to determine the optimal solution without the need to enumerate all candidate solutions [11].

The *Multipoint Markov–Dubins Problem* (MPMDP) generalizes the classic problem to a sequence of assigned planar points $\mathbf{P}_0, \dots, \mathbf{P}_n$ using interpolation with Dubins paths [12,13]. The initial and final angles, ϑ_0 and ϑ_n , are assigned, while the other intermediate angles ϑ_j , for $j = 1, \dots, n - 1$, are free and thus are unknowns of the problem. Even for a few points, the brute force method, i.e., trying all cases for each pair of consecutive points, becomes rapidly infeasible due to the exponential growth of possible test cases. Recent solutions to the multipoint problem are based on a Non-Linear Programming approach (NLP) [12,13] or Mixed-Integer Non-Linear Programming (MINLP) [12]. The most effective technique is actually the iDPP method based on the Iterative Dynamic Programming Principle [12] and is suitable for parallel computing architectures on CPUs as well as on GPUs [14].

A special case of the MPMDP requires finding an obstacle-free path through three points, which is a problem with name and importance: 3PDP—*Three-Point Dubins Problem*. Despite being a subcase of the solved MPMDP case, the 3PDP has received special attention in the last decade because of its application as a subroutine in a high-level planner for the Dubins Traveling Salesman Problem (DTSP) [15–17]. This subroutine is used to insert a new point in an existing path; therefore, the initial, the new middle, and the final point are specified, as well as the initial and final angles. Thus, the 3PDP reduces to the one-dimensional problem of finding the angle corresponding to the central point, so that the curvature constraint is satisfied, and the minimum length is preserved. To solve this non-smooth minimization problem, several methods can be employed [18–20].

The crucial point of interest in this special instance of the MPMDP is the computational time: since it is called repeatedly as a subroutine of the DTSP, the solutions based on brute force, Dynamic Programming, or NLP/MINLP are considered too inefficient because of the considerable overhead. Therefore, a solution to this problem has been sought, as detailed in the next section, aiming at computational efficiency.

1.1. Problem Formulation

The 3PDP is formulated as follows: given an initial point $\mathbf{P}_i = (x_i, y_i)$ with orientation ϑ_i , a middle point $\mathbf{P}_m = (x_m, y_m)$, and a final point $\mathbf{P}_f = (x_f, y_f)$ with orientation ϑ_f , find the shortest concatenation of two Dubins paths that begin in \mathbf{P}_i , pass through \mathbf{P}_m , and reach \mathbf{P}_f while respecting the constraints of having the maximum absolute curvature not exceeding κ_{\max} ; see Figure 1. The problem is clearly one-dimensional, with the angle ϑ_m at

P_m being the only unknown of the problem. Once ϑ_m is determined, the solution becomes trivial, since it is just twice the application of the standard Dubins problem.

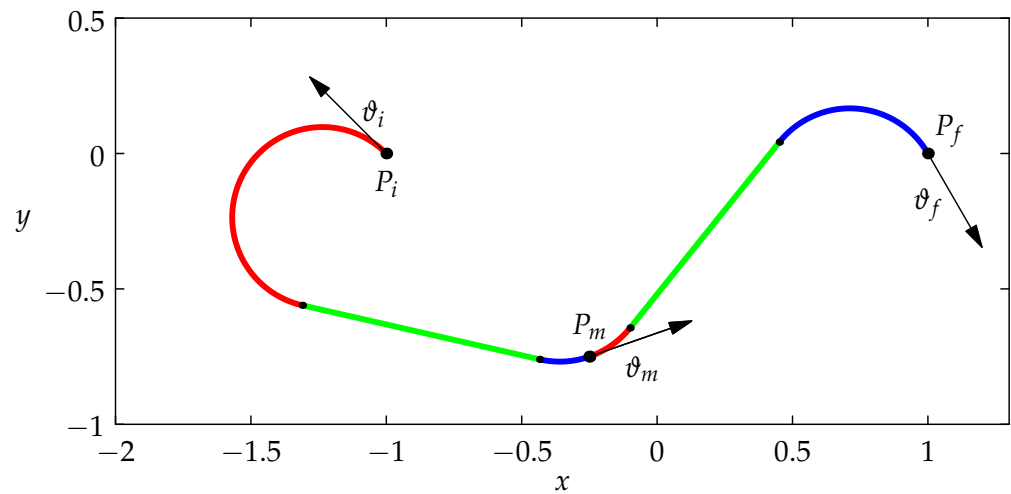


Figure 1. An example and scheme of an instance of the 3PDP. This path is an example of the LSLSR type. The three arcs of each Dubins are represented by colors red, green, and blue

Despite being formally simple, the 3PDP is quite involved: the function of the total length to be minimized is, indeed, not continuous and thus non-smooth. Therefore, classic optimization methods cannot be applied, and different methods are required [19,21].

1.2. Paper Contribution and Layout

Our research presents an efficient and compact solution to the Three-Point Dubins Problem (3PDP), addressing the non-smooth minimization challenge. Our key contributions are as follows:

- *Analytical Derivative Computation:* We formulated a compact, closed-form solution for the Markov–Dubins problem. With this formulation, we significantly reduce computational overhead and improve the efficiency. This is instrumental to finding the analytical expressions for the derivatives of the Dubins path length with respect to the initial and final angles. These expressions allow for a more precise and faster computation of path lengths and gradients, enabling a better optimization performance. Additionally, we established bounds for the existence regions of each of the six possible solutions. With this analysis, we can isolate discontinuities and provide a more accurate and efficient solution to the 3PDP.
- *Optimal Path Length and Derivative Analysis:* We computed the length and derivatives of the optimal 3PDP path, precisely framing it as a non-smooth minimization problem.
- *Introduction of Two Non-smooth Minimization Methods:* We propose two specialized non-smooth minimization algorithms tailored for the 3PDP:
 - A pattern search algorithm, which can be further enhanced with Algorithm 748 [22].
 - A trichotomy algorithm, similarly augmentable with Algorithm 748.

Algorithm 748 can be used to perform the root-finding of the first derivatives of the path length, improving the efficiency and accuracy of the solution up to machine precision.

- *Implementation and Accessibility:* We provided a comprehensive implementation of our algorithms in a C++ library with a Matlab interface, named *Clothoids* [23,24]. The library, along with clear descriptions and pseudocode of the algorithms, is freely available and open-source, facilitating further research and practical applications, bridging the gap between theoretical research and practitioners, ensuring that our work can be readily used and extended by others.

This paper is structured as follows: Section 2 gives insight into the 3PDP and presents state-of-the-art methods on the subject. Section 3 develops the solution to the standard Dubins problem and, in particular, the length of the solution as a function of the givens of the problem. Our novel derivation allows us to compute the derivatives of the lengths of the various cases, as shown in Section 4. In Section 5, we detail our solution of the 3PDP, comprising all cases. The corresponding algorithms are discussed in Section 6 and tested in Section 7. Finally, we present our conclusions in Section 8.

2. Background and State-of-the-Art Methods

Several results approximate the solution of the MPMDP under the hypothesis that the distance between consecutive points is at least $d^* := 4\rho_{\min} = 4/\kappa_{\max}$, a condition that implies that only CSC paths will constitute the solution, so that cases CCC are not considered. Under this hypothesis, in defining the angles at the waypoints as the angles of the line segment connecting the adjacent points in the sequence, the approximation yields a path that is not more than 1.91 times longer than the optimal one. Without the $d > d^*$ distance assumption, there is a method to construct a path that is at most 5.03 times longer than the optimal, a factor which decreases to 2.58 if the distance between the points is more than d^* [25,26].

Apart from the bounds, some exact results can be derived and computed in different ways, covering some or all the possible feasible paths.

The first exact result in the problem of visiting an ordered sequence of waypoints with a Dubins path is given in [25,27,28], under the assumption that the distance between consecutive points is at least d^* , thus considering the CSC cases only. It is shown that the problem becomes locally convex over an open region; however, this region is not even connected, and therefore, 2^{n-2} subproblems must be solved, each containing a local minimum of the original problem. In their analysis, the authors reduce the complexity to 2^k , with k being the number of sharp turns of the path.

A different approach, based on inversive geometry [26], proposes an iterative method (IM) to solve the 3PDP with a 3×3 non-linear system of equations to find the optimal angle at the middle point for paths CSC-CSC (or shorter words). This method that combines geometric arguments with the bisection algorithm is compared with the brute force method of discretizing the middle angle with 360 samples and testing all cases (called the discretization-based method—DBM [29,30]), showing a computational time improvement ranging from a factor of about 14 in the best case, for points that have a distance greater than d^* , to a worse case improvement of a factor of 5 for points that are closer than d^* . Regarding accuracy, the iterative method in [26] is generally more precise than the brute force DBM. However, it does not cover the cases where the path contains a CCC word.

An approach that covers all cases and is based on algebraic methods is the one proposed in [31], where the trigonometric relations that characterize the solution are transformed into polynomials of relatively high degree, from 4 to 20. In the paper, it is discussed that although, in principle, there are 6×6 potential path types (4 CSCs and 2 CCCs for each pair of points), of these 36 cases, only 18 are optimal candidates; see Table 1.

The algorithm requires constructing and finding the (real) roots of these 18 polynomials and, for each admissible root, testing the length of the path. The angle corresponding to the shortest length is elected as the optimal angle at the midpoint. This method is called the polynomial-based method, PBM, and requires, in the worst case, the computation of $2 \times 4 + 2 \times 6 + 10 \times 8 + 4 \times 20 = 180$ pairs of Dubins paths, plus the time to find up to their 180 roots. Depending on the distance of the points, this method achieves a speed-up ranging from 25 to 45 times the speed of the DBM with 360 samples [31].

A recent method [28] is based on geometrical observations and solves the case for points having a distance larger than d^* ; thus, the cases are CSC (but not CCC). It takes the name of the geometry-based method or GBM. This method is purely geometrical and bases the solution on the construction of a special ellipse. The solution is computed by building circles of radius ρ_{\min} with tangents on the initial and final points (four possible circles). Then, an ellipse is defined by its foci laying in the previously built circle centers. The ellipse must be tangent to the circle of radius ρ_{\min} centered in the middle point.

Table 1. The types of the 18 admissible paths; the subscript indicates the degree of the corresponding polynomial according to [31]. For instance, RSLRL₂₀ means that the path is of type CSCCC and that the resulting polynomial has degree 20.

| CCCCC | CCCSC | CSCCC | CSCSC |
|--------------------|---------------------|---------------------|--------------------|
| RLRLR ₆ | RLRSR ₈ | RSRLR ₈ | RSRSR ₄ |
| LRLRL ₆ | RLRSL ₂₀ | LSRLR ₂₀ | LSRSR ₈ |
| | LRLSL ₈ | RSLRL ₂₀ | RSRSL ₈ |
| | LRLSR ₈ | LSLRL ₈ | LSRSL ₈ |
| | | | LSLSL ₄ |
| | | | RLSL ₈ |
| | | | LSLSR ₈ |
| | | | RLSLR ₈ |

The optimal angle is given by the angle of the common tangent to the ellipse and the circle centered at the middle point, and a semi-analytical solution to the problem is provided. The computational improvement over the DBM is of a factor of 150, and it can find a shorter solution in about 92% of the instances: if the path contains a CSC word, the solution will be exact and thus better than the DBM, but not when the path contains a CCC.

3. Derivation of Dubins Path with Trigonometry

This section is devoted to deriving the solution of the classic Dubins problem to obtain analytic relations for the optimal lengths so that the next section can compute their derivatives.

Without loss of generality, the original problem (1) can be transformed into a simpler one by translating, rotating, and scaling the original configuration such that $x_i = y_i = 0$, and $x_f = d, y_f = 0$ with a unitary maximum curvature of $\kappa_{\max} = 1$. Let d be the Euclidean distance between the initial and final points scaled by the maximum curvature:

$$d = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2} / \kappa_{\max}. \tag{2}$$

Then, the aforementioned transform is the following, (3): the translation is given by $-(x_i, y_i)$ and is followed by a rotation around $(0,0)$ by an angle $-\phi$ given by $\phi = \text{atan2}(y_f - y_i, x_f - x_i)$, followed by a scaling of $\kappa_{\max}^{-1} = \rho_{\min}$, or compactly,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\kappa_{\max}} \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} x - x_i \\ y - y_i \end{bmatrix}. \tag{3}$$

Let $\alpha = \vartheta_i + \phi$ and $\beta = \vartheta_f + \phi$ be the initial and final angles in the transformed coordinate system. The new initial point is thus the origin with angle α , and the final point is on the positive x -axis at distance d , (2), from the origin and angle β with maximum unitary curvature.

The solution to a Dubins problem is a sequence of three curves (lines or arcs [32]) that can be written in the following form for $j \in 0, 1, 2$:

$$j\text{-th curve: } \begin{cases} x(\ell) = x_j + s(\sin(\vartheta_j + s\ell) - \sin \vartheta_j) + (1 - s^2)\ell \cos \vartheta_j, \\ y(\ell) = y_j + s(\cos \vartheta_j - \cos(\vartheta_j + s\ell)) + (1 - s^2)\ell \sin \vartheta_j, \\ \theta(\ell) = \vartheta_j + s\ell, \end{cases} \quad (4)$$

where ℓ is the curvilinear abscissa, $x(\ell)$ and $y(\ell)$ are the coordinates, and $\theta(\ell)$ is the orientation. Let us define $x_0 = y_0 = 0$ and $\vartheta_0 = \alpha$, corresponding to the initial point. Notice that, when the integer variable s is equal to 0, the function represents a line segment, whereas, when $s = 1$, the curve is an arc of a unitary circle turning left, and, when $s = -1$, the circle is turning right.

The Dubins problem is formulated with three internal segments (CSC or CCC) with known initial and final coordinates and headings in the transformed coordinate system. Hence, we can juxtapose three of such curves. Using the notation s_1, s_2 , and s_3 for the signs of the curvature of the three segments coming from (4), we can write the equations for the three segments:

$$\begin{aligned} x_1 &= 0 + s_1(\sin \vartheta_1 - \sin \alpha) \\ y_1 &= 0 + s_1(\cos \alpha - \cos \vartheta_1) \\ \vartheta_1 &= \alpha + s_1\ell_1 \\ x_2 &= x_1 + s_2(\sin \vartheta_2 - \sin \vartheta_1) + (1 - s_2^2)\ell_2 \cos \vartheta_1 \\ y_2 &= y_1 + s_2(\cos \vartheta_1 - \cos \vartheta_2) + (1 - s_2^2)\ell_2 \sin \vartheta_1 \\ \vartheta_2 &= \vartheta_1 + s_2\ell_2 \\ 0 &= x_2 + s_3(\sin \beta - \sin \vartheta_2) \\ d &= y_2 + s_3(\cos \vartheta_2 - \cos \beta) \\ \beta &= \vartheta_2 + s_3\ell_3. \end{aligned} \quad (5)$$

The proposed transformation simplifies the equations, eliminating $x_1, x_2, y_1, y_2, \vartheta_1$, and ϑ_2 from the general system (5). Moreover, we can separate two families of solutions based on the presence or absence of the middle singular arc (straight line segment), that is, the CSC and CCC cases.

Before proceeding, it is also convenient to define some notation and shorthand for the trigonometric functions of the angles α and β that will be used in the following sections, to specify that the expression depends on trigonometric combinations of α and β :

$$\begin{cases} S^- = \sin \alpha - \sin \beta \\ C^- = \cos \alpha - \cos \beta \\ S^+ = \sin \alpha + \sin \beta \\ C^+ = \cos \alpha + \cos \beta \end{cases} \quad \mathcal{P}_{\alpha,\beta} = \{\sin \alpha, \cos \alpha, \sin \beta, \cos \beta\} \quad (6)$$

where $\mathcal{P}_{\alpha,\beta}$ denotes a set of the trigonometric functions of the angles α and β . We remark that S^\pm and C^\pm are functions of $\mathcal{P}_{\alpha,\beta}$, but we drop the explicit dependence on $\mathcal{P}_{\alpha,\beta}$ to keep the notation light.

For all three cases, we want to compute the total length of the path, which is the sum of the lengths of the three segments.

$$\ell = \ell_1 + \ell_2 + \ell_3. \quad (7)$$

3.1. Case CSC

This case happens when $s_2 = 0$ and, depending on the sign of s_1 and s_3 , gives origin to the curves LSL, RSR, LSR, and RSL. Then, system (5) simplifies to

$$\begin{aligned} d &= s_1(\sin \vartheta_1 - \sin \alpha) + \ell_2 \cos \vartheta_1 + s_3(\sin \beta - \sin \vartheta_2) \\ 0 &= s_1(\cos \alpha - \cos \vartheta_1) + \ell_2 \sin \vartheta_1 + s_3(\cos \vartheta_2 - \cos \beta), \end{aligned} \tag{8}$$

and by imposing the continuity conditions to (8) on the angle of the line segment $\vartheta_1 = \vartheta_2 = \vartheta_S = \alpha + s_1 \ell_1 = \beta - s_3 \ell_3$, we have

$$\begin{aligned} d &= (s_1 - s_3) \sin \vartheta_S - s_1 \sin \alpha + s_3 \sin \beta + \ell_2 \cos \vartheta_S \\ 0 &= (s_3 - s_1) \cos \vartheta_S + s_1 \cos \alpha - s_3 \cos \beta + \ell_2 \sin \vartheta_S. \end{aligned} \tag{9}$$

We combine these two equations in (9), multiplying by $\sin \vartheta_S$ and $\cos \vartheta_S$, respectively, and we obtain an explicit expression for ℓ_2 and an implicit expression for ϑ_S :

$$\begin{aligned} \ell_2 &= (d - s_3 \sin \beta + s_1 \sin \alpha) \cos \vartheta_S + (s_3 \cos \beta - s_1 \cos \alpha) \sin \vartheta_S \\ 0 &= (d - s_3 \sin \beta + s_1 \sin \alpha) \sin \vartheta_S - (s_3 \cos \beta - s_1 \cos \alpha) \cos \vartheta_S + s_3 - s_1. \end{aligned} \tag{10}$$

This case can be further separated into two subcases, the first, CSC^+ , with the two circular arcs turning in the same direction (LSL and RSR), and the other, CSC^- , with circular arcs in the opposite direction (LSR and RSL). They have two different solutions, as detailed below.

3.1.1. The Subcase CSC^+ : LSL and RSR

As stated in the previous section, this solution type falls into the CSC^+ case, where $s_2 = 0$ and $s_1 = s_3 = s$. Hence, we can rewrite equations (10) in terms of the parameters $\mathcal{P}_{\alpha,\beta}$ using notation (6) as follows:

$$\begin{aligned} (d + s \mathcal{S}^-) \sin \vartheta_S + s \mathcal{C}^- \cos \vartheta_S &= 0 \\ (d + s \mathcal{S}^-) \cos \vartheta_S - s \mathcal{C}^- \sin \vartheta_S &= \ell_2. \end{aligned} \tag{11}$$

From the first equation, we can solve for ϑ_S , and the second is already solved for ℓ_2 , as a function of ϑ_S . Thus, we have

$$\vartheta_S^{CSC^+} = \text{atan2}(-\mathcal{C}^-, d + s \mathcal{S}^-) \pmod{\pi}. \tag{12}$$

We choose the only ϑ_S producing a non-negative ℓ_2 . Any ϑ_S not respecting this constraint is discarded. The length ℓ_2 is obtained from (11), knowing the solution in (12), and from the condition $\vartheta_1 = \vartheta_2 = \vartheta_S = \alpha + s \ell_1 = \beta - s \ell_3$, we can derive the other lengths ℓ_1 and ℓ_3 , which are as follows:

$$\begin{aligned} \ell_1^{CSC^+} &= s(\vartheta_S^{CSC^+} - \alpha) \pmod{2\pi} \\ \ell_2^{CSC^+} &= (d + s \mathcal{S}^-) \cos \vartheta_S^{CSC^+} - s \mathcal{C}^- \sin \vartheta_S^{CSC^+} \\ \ell_3^{CSC^+} &= s(\beta - \vartheta_S^{CSC^+}) \pmod{2\pi}. \end{aligned} \tag{13}$$

3.1.2. The Subcase CSC^- : LSR and RSL

This solution type falls into the CSC^- case, where $s_2 = 0$, $s_1 = s$, and $s_3 = -s$. It is possible to rewrite equations (10) in terms of the parameters $\mathcal{P}_{\alpha,\beta}$ using notation (6) as follows:

$$\begin{aligned} (d + s \mathcal{S}^+) \sin \vartheta_S + s \mathcal{C}^+ \cos \vartheta_S &= 2s \\ (d + s \mathcal{S}^+) \cos \vartheta_S - s \mathcal{C}^+ \sin \vartheta_S &= \ell_2. \end{aligned} \tag{14}$$

We can substitute the expression for ϑ_S to obtain a polynomial equation as a function of the auxiliary variable X .

$$\vartheta_S = \pi + 2 \arctan(X), \quad \cos \vartheta_S = \frac{X^2 - 1}{X^2 + 1}, \quad \sin \vartheta_S = \frac{-2X}{X^2 + 1}. \tag{15}$$

Therefore, using (15) in (14), multiplying by $(X^2 + 1)$, and using $s^2 = 1$, we can obtain a quadratic equation in X as follows:

$$0 = (2 - \mathcal{C}^+) X^2 + 2(s d + \mathcal{S}^+) X + (2 + \mathcal{C}^+). \tag{16}$$

Hence, once the polynomial equation is solved. We can substitute the expression for ϑ_S in (14), derive the lengths ℓ_2 , and retrieve ℓ_1 and ℓ_3 with the following:

$$\vartheta_S^{CSC^-} = \text{atan2}(-2X, X^2 - 1). \tag{17}$$

If $|2 - \mathcal{S}^+| \ll 1$, the polynomial in (16) is not well posed and ill conditioned. Therefore, we can change the coordinates to $X = 1/Y$ to reshape the polynomial, yielding the following expression:

$$\vartheta_S = \pi + 2 \arctan(1/Y), \quad \cos \vartheta_S = \frac{1 - Y^2}{Y^2 + 1}, \quad \sin \vartheta_S = \frac{-2Y}{Y^2 + 1}. \tag{18}$$

From the condition $\vartheta_1 = \vartheta_2 = \vartheta_S = \alpha + s_1 \ell_1 = \beta - s_3 \ell_3$ and (14), (15), or (18), we can derive the lengths ℓ_1 , ℓ_2 , and ℓ_3 , recalling also (17), as follows:

$$\begin{aligned} \ell_1^{CSC^-} &= s(\vartheta_S^{CSC^-} - \alpha) \pmod{2\pi} \\ \ell_2^{CSC^-} &= (d + s \mathcal{S}^+) \cos \vartheta_S^{CSC^-} - s \mathcal{C}^+ \sin \vartheta_S^{CSC^-} \\ \ell_3^{CSC^-} &= s(\vartheta_S^{CSC^-} - \beta) \pmod{2\pi}. \end{aligned} \tag{19}$$

3.1.3. Range of the Solution

The solution exists only for angles coming from real roots of polynomial (16). The discriminant \mathcal{D}_{CSC^+} of that polynomial is

$$\begin{aligned} \frac{1}{4} \mathcal{D}_{CSC^+} &= (s d + \mathcal{S}^+)^2 - (2 - \mathcal{C}^+)(2 + \mathcal{C}^+) \\ &= 2(s d(\sin \alpha + \sin \beta) + \sin \alpha \sin \beta + \cos \alpha \cos \beta) + d^2 - 2, \end{aligned} \tag{20}$$

In introducing new variables Z and W , defined by the relations $\alpha = 2 \arctan Z$ and $\beta = 2 \arctan W$, it is possible to convert the trigonometric expressions into polynomials. Substituting each of these relations into the discriminant in (20), we obtain two polynomials:

$$\begin{aligned} P_{CSC^+}(Z; \beta) &= (s d \sin \beta - \cos \beta + t) Z^2 + 2(s d + \sin \beta) Z + s d \sin \beta + \cos \beta + t, \\ P_{CSC^+}(W; \alpha) &= (s d \sin \alpha - \cos \alpha + t) W^2 + 2(s d + \sin \alpha) W + s d \sin \alpha + \cos \alpha + t, \end{aligned} \tag{21}$$

where $t = \frac{d^2}{2} - 1$. The real roots of $P_{CSC^+}(Z; \beta)$ represent the border of the range interval where α can run, to obtain a solution for the CSC^+ problem for a fixed β . The same holds for $P_{CSC^+}(W; \alpha)$ for the interval range of β .

The range detection is fundamental in the solution to the 3PDP. In general, the length of the 3PDP is a discontinuous function of the middle angle (see Figure 2). Standard direct search [33,34] and non-smooth minimization algorithms [35,36] can fail to find the solution if evaluation points are located in a small discontinuity region (see Figure 3). Hence, our proposed approach, discussed in the later sections, employs the solution of polynomial (21) to isolate the discontinuity points and treat them separately in the minimization algorithm.

The range detection is fundamental to finding the points that are candidate to be points of discontinuity of the length of the Dubins path with respect to initial or final angles α and β .

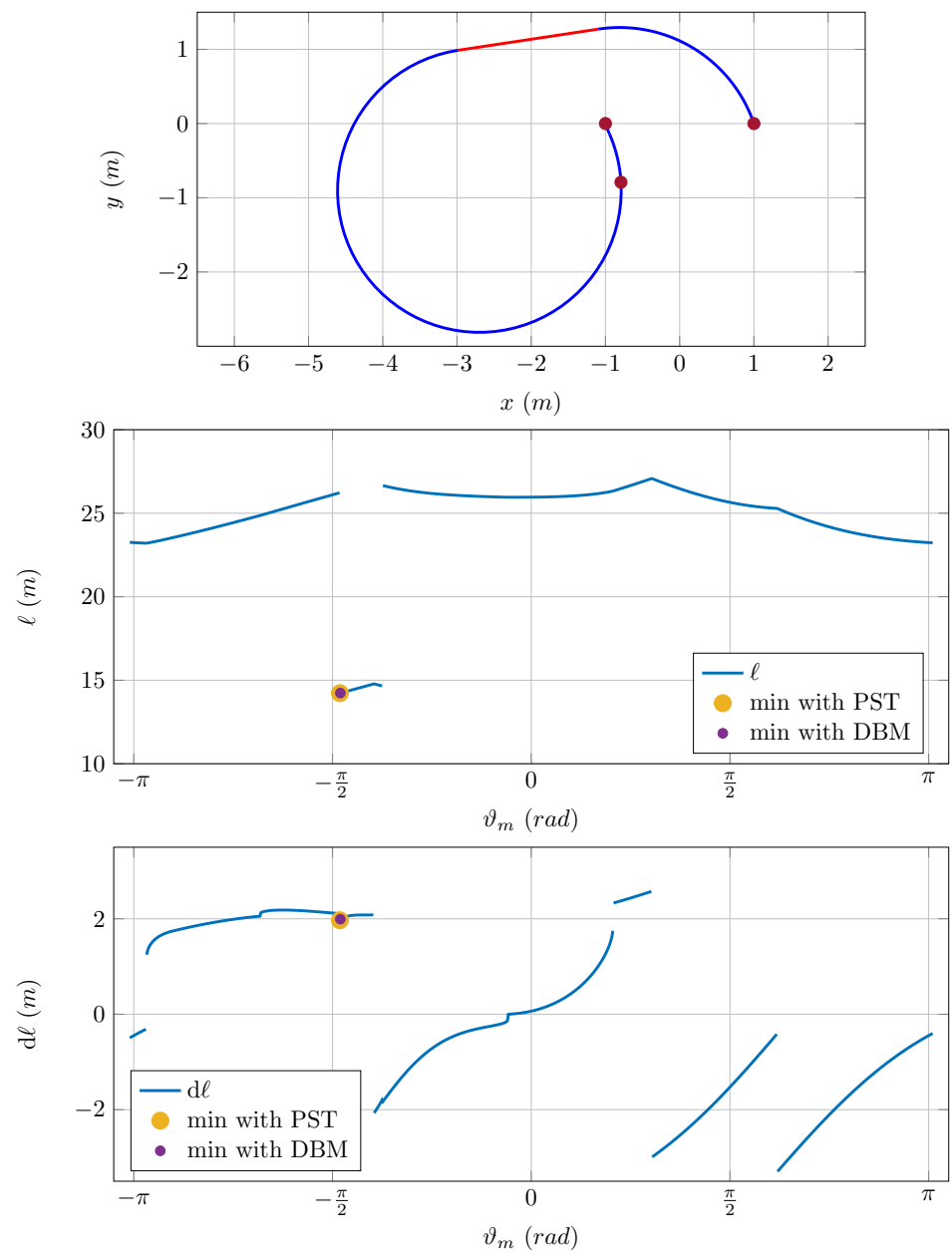


Figure 2. Example of the total length of a three-point Dubins path. Spatial coordinates are in meters, and angles are in radian. (**Top:** path result with alternating colors for each arc; **Middle:** total length ℓ as a function of middle angle ϑ_m ; **Bottom:** derivative of total length ℓ as a function of middle angle ϑ_m).

3.2. Case CCC

The CCC case, happening when $s_2 \neq 0$, comprises two paths: LRL and RLR. In this situation, we can impose $s_1 = s_3 = -s_2 = s$ and simplify (5) to obtain

$$\begin{aligned}
 s d &= (\sin \vartheta_1 - \sin \alpha) - (\sin \vartheta_2 - \sin \vartheta_1) + (\sin \beta - \sin \vartheta_2) \\
 0 &= (\cos \alpha - \cos \vartheta_1) - (\cos \vartheta_1 - \cos \vartheta_2) + (\cos \vartheta_2 - \cos \beta).
 \end{aligned}
 \tag{22}$$

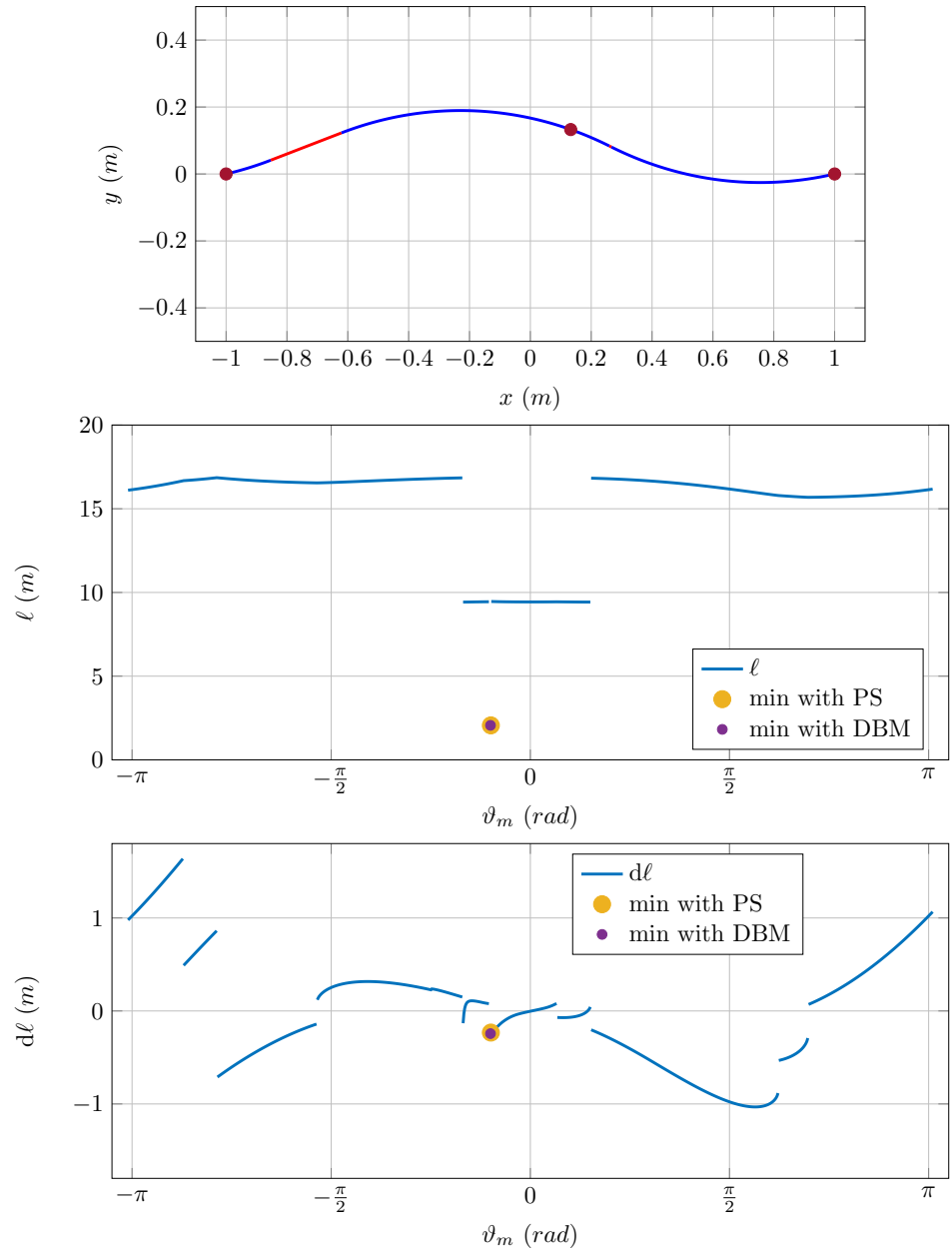


Figure 3. Example of the total length of a three-point Dubins path. Spatial coordinates are in meters, and angles are in radian. (**Top:** path result with alternating colors for each arc; **Middle:** total length ℓ as a function of middle angle ϑ_m ; **Bottom:** derivative of total length ℓ as a function of middle angle ϑ_m).

Equation (22) can be rewritten in compact form as

$$\begin{aligned} s d &= 2(\sin \vartheta_1 - \sin \vartheta_2) + \sin \beta - \sin \alpha \\ 0 &= 2(\cos \vartheta_2 - \cos \vartheta_1) + \cos \alpha - \cos \beta \\ \beta &= \alpha + s(\ell_1 - \ell_2 + \ell_3). \end{aligned} \tag{23}$$

We can state equations (23) in terms of the parameters $\mathcal{P}_{\alpha,\beta}$ using notation (6) as

$$\begin{aligned} s d &= 2(\sin \vartheta_1 - \sin \vartheta_2) - \mathcal{S}^- \\ 0 &= 2(\cos \vartheta_2 - \cos \vartheta_1) + \mathcal{C}^-. \end{aligned} \tag{24}$$

From (24) and squaring, we obtain

$$\begin{aligned} \frac{1}{4}(s d + \mathcal{S}^-)^2 &= (\sin \vartheta_1 - \sin \vartheta_2)^2 = \sin^2 \vartheta_1 + \sin^2 \vartheta_2 - 2 \sin \vartheta_1 \sin \vartheta_2 \\ \frac{1}{4}(\mathcal{C}^-)^2 &= (\cos \vartheta_2 - \cos \vartheta_1)^2 = \cos^2 \vartheta_1 + \cos^2 \vartheta_2 - 2 \cos \vartheta_1 \cos \vartheta_2 \end{aligned} \tag{25}$$

and by adding the two equalities (25) (and dividing by 2), we obtain

$$\begin{aligned} \frac{(s d + \mathcal{S}^-)^2 + (\mathcal{C}^-)^2}{8} &= 1 - \sin \vartheta_1 \sin \vartheta_2 - \cos \vartheta_1 \cos \vartheta_2 \\ &= 1 - \cos(\vartheta_2 - \vartheta_1) = 1 - \cos \ell_2, \end{aligned} \tag{26}$$

where the last step follows inserting the sixth equation of (5), that is, $\vartheta_2 - \vartheta_1 = s_2 \ell_2$, and noting that cosine is even. Thus, ℓ_2 can be obtained from (26) as follows:

$$\begin{aligned} t^{\text{CCC}} &= \frac{3 + \cos(\alpha - \beta) + s d(\sin \beta - \sin \alpha)}{4} - \frac{d^2}{8}, \\ \cos \ell_2 &= 1 - \frac{(s d + \mathcal{S}^-)^2 + (\mathcal{C}^-)^2}{8} = t^{\text{CCC}}, \\ \ell_2 &= \arccos(t^{\text{CCC}}), \quad \text{or} \quad \ell_2 = 2\pi - \arccos(t^{\text{CCC}}). \end{aligned} \tag{27}$$

The solution exists when the argument of arccos is in the interval $[-1, 1]$. Therefore, the t^{CCC} solved in (27) must be comprised between -1 and 1 , yielding the following inequality:

$$-1 \leq \frac{3 + \cos(\alpha - \beta) + s d(\sin \beta - \sin \alpha)}{4} - \frac{d^2}{8} \leq 1. \tag{28}$$

The previous expression (28) can be multiplied by 4 and moved $-d^2/2 + 3$ to the left and right side:

$$\frac{d^2}{2} - 7 \leq \cos(\alpha - \beta) + s d(\sin \beta - \sin \alpha) \leq \frac{d^2}{2} + 1. \tag{29}$$

We can obtain the other relations from the trigonometric relations furnished by (24). From $\vartheta_1 = \alpha + s\ell_1$ and $\vartheta_2 = \alpha + s(\ell_1 - \ell_2)$ and expanding the trigonometric functional, we obtain a linear system in $\cos \ell_1$ and $\sin \ell_1$, and from $\vartheta_1 = \beta + s(\ell_2 - \ell_3)$ and $\vartheta_2 = \beta - s\ell_3$ and expanding the trigonometric functions again, we obtain a linear system in $\cos \ell_3$ and $\sin \ell_3$ as follows:

$$\begin{aligned} \begin{bmatrix} \sin \alpha - \sin \tilde{\alpha} & s(\cos \alpha - \cos \tilde{\alpha}) \\ \cos \alpha - \cos \tilde{\alpha} & s(\sin \tilde{\alpha} - \sin \alpha) \end{bmatrix} \begin{bmatrix} \cos \ell_1 \\ \sin \ell_1 \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} s d + \mathcal{S}^- \\ e^- \end{bmatrix}, \\ \begin{bmatrix} \sin \tilde{\beta} - \sin \beta & s(\cos \beta - \cos \tilde{\beta}) \\ \cos \tilde{\beta} - \cos \beta & s(\sin \tilde{\beta} - \sin \beta) \end{bmatrix} \begin{bmatrix} \cos \ell_3 \\ \sin \ell_3 \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} s d + \mathcal{S}^- \\ e^- \end{bmatrix}, \end{aligned} \tag{30}$$

where $\tilde{\alpha} = \alpha - s\ell_2$ and $\tilde{\beta} = \beta + s\ell_2$. In conclusion, we can summarize the results for the CCC case as follows:

Evaluate
$$t^{\text{CCC}} = \frac{3 + \cos(\alpha - \beta) - s d \mathcal{S}^- - \frac{d^2}{8}}{4}, \tag{31}$$

If $t^{\text{CCC}} \in [-1, 1]$, then build
$$\ell_2 = \arccos(t^{\text{CCC}}) \quad \text{and} \quad \ell_2 = 2\pi - \arccos(t^{\text{CCC}}); \tag{32}$$

solve the linear system (30) and compute
$$\begin{aligned} \ell_1 &= \text{atan2}(\sin \ell_1, \cos \ell_1) \quad \text{mod} \quad 2\pi, \\ \ell_3 &= \text{atan2}(\sin \ell_3, \cos \ell_3) \quad \text{mod} \quad 2\pi. \end{aligned} \tag{33}$$

Discard the solutions that do not satisfy $\alpha + s(\ell_1^{\text{CCC}} - \ell_2^{\text{CCC}} + \ell_3^{\text{CCC}}) = \beta \quad \text{mod} \quad 2\pi$.

Range of the Solution

The solution exists only when condition (29) is satisfied. Introducing new variables X and Y , in the same fashion as before, defined from the angles $\alpha = 2 \arctan X$ and $\beta = 2 \arctan Y$, and substituting them in the inequality (29), we obtain four quadratic polynomials:

$$\begin{aligned} p &= s d \sin \beta - \frac{d^2}{2}, \quad q = 2(\sin \beta - s d) \\ P_{\text{CCC},1}(X; \beta) &= (p - \cos \beta - 1)X^2 + qX + p + \cos \beta - 1 \\ P_{\text{CCC},2}(X; \beta) &= (p - \cos \beta + 7)X^2 + qX + p + \cos \beta + 7 \end{aligned} \tag{34}$$

and

$$\begin{aligned} p &= -s d \sin \alpha - \frac{d^2}{2}, \quad q = 2(\sin \alpha + s d) \\ P_{\text{CCC},3}(X; \alpha) &= (p - \cos \alpha - 1)X^2 + qX + p + \cos \alpha - 1 \\ P_{\text{CCC},4}(X; \alpha) &= (p - \cos \alpha + 7)X^2 + qX + p + \cos \alpha + 7. \end{aligned} \tag{35}$$

The real roots of the polynomials return the border of the interval range where α can run, to yield a solution for the CCC problem with β fixed. The same holds for $P_{\text{CCC},2/3}(X; \alpha)$ for the interval range of β .

The following section uses the results derived so far to compute the derivatives of the optimal lengths so that an algorithm for the 3PDP can be constructed.

4. Derivatives of the Optimal Lengths

In this section, we present the partial derivatives of the optimal lengths with respect to the initial and final angles α and β .

4.1. Derivatives of the LSL and RSR Cases

For the CSC^+ case given in (13), we can trivially derive some partial derivatives of the lengths with respect to α or β . For instance, $\partial_\alpha(\ell_1 + \ell_3) = -s$ and $\partial_\beta(\ell_1 + \ell_3) = s$. The derivatives of ℓ_2 with respect to α and β can be computed and simplified using (11) and collecting (12):

$$\begin{aligned} \partial_\alpha \ell_2^{CSC^+} &= s \cos(\alpha - \vartheta_S^{CSC^+}) - \left(s c^- \cos \vartheta_S^{CSC^+} + (sd + \delta^-) \sin \vartheta_S \right) \partial_\alpha \vartheta_S^{CSC^+} \\ &= s \cos(\alpha - (\alpha + s \ell_1^{CSC^+})) \\ &= s \cos \ell_1^{CSC^+}, \\ \partial_\beta \ell_2^{CSC^+} &= -s \cos(\beta - \vartheta_S^{CSC^+}) - s \left(c^- \cos \vartheta_S^{CSC^+} + (sd + \delta^-) \sin \vartheta_S^{CSC^+} \right) \partial_\beta \vartheta_S^{CSC^+} \\ &= -s \cos(\beta - (\beta - s \ell_3^{CSC^+})) \\ &= -s \cos \ell_3^{CSC^+}. \end{aligned} \tag{36}$$

Therefore, we can write the partial derivatives of the sum of the lengths using the result from (36):

$$\begin{aligned} \partial_\alpha(\ell_1^{CSC^+} + \ell_2^{CSC^+} + \ell_3^{CSC^+}) &= s(\cos \ell_1^{CSC^+} - 1), \\ \partial_\beta(\ell_1^{CSC^+} + \ell_2^{CSC^+} + \ell_3^{CSC^+}) &= s(1 - \cos \ell_3^{CSC^+}). \end{aligned} \tag{37}$$

To remove the dependency of ℓ_1 and ℓ_3 from (37), we substitute their expression (13), so that we can write the partial derivatives of the sum of the lengths as follows (considering ℓ the total length, $\ell = \ell_1 + \ell_2 + \ell_3$):

$$\begin{aligned} \partial_\alpha(\ell^{CSC^+}) &= s(\cos(\vartheta_S^{CSC^+} - \alpha) - 1), \\ \partial_\beta(\ell^{CSC^+}) &= s(1 - \cos(\beta - \vartheta_S^{CSC^+})). \end{aligned} \tag{38}$$

The upper bound of the derivative in (37) and (38) and thus of the Lipschitz constant is 2.

4.2. Derivatives of the LSR and RSL Cases

We compute herein the partial derivatives of the optimal lengths for the CSC^- case given in (19) with respect to α or β :

$$\begin{aligned} \partial_\alpha(\ell_1^{CSC^-} + \ell_3^{CSC^-}) &= 2s \partial_\alpha \vartheta_S^{CSC^-} - s, \\ \partial_\beta(\ell_1^{CSC^-} + \ell_3^{CSC^-}) &= 2s \partial_\beta \vartheta_S^{CSC^-} - s. \end{aligned} \tag{39}$$

Then, using (6) and (14), we have

$$\begin{aligned}
 \partial_\alpha \ell_2^{\text{CSC}^-} &= \partial_\alpha \left((d + s \delta^+) \cos \vartheta_S^{\text{CSC}^-} - s C^+ \sin \vartheta_S^{\text{CSC}^-} \right) \\
 &= s \cos(\alpha - \vartheta_S^{\text{CSC}^-}) - \left((d + s \delta^+) \sin \vartheta_S^{\text{CSC}^-} + s C^+ \cos \vartheta_S^{\text{CSC}^-} \right) \partial_\alpha \vartheta_S^{\text{CSC}^-} \\
 &= s \cos \ell_1^{\text{CSC}^-} - 2s \partial_\alpha \vartheta_S^{\text{CSC}^-}, \\
 \partial_\beta \ell_2^{\text{CSC}^-} &= \partial_\beta \left((d + s \delta^+) \cos \vartheta_S^{\text{CSC}^-} - s C^+ \sin \vartheta_S^{\text{CSC}^-} \right) \\
 &= s \cos(\beta - \vartheta_S^{\text{CSC}^-}) - \left((d + s \delta^+) \sin \vartheta_S^{\text{CSC}^-} + s C^+ \cos \vartheta_S^{\text{CSC}^-} \right) \partial_\beta \vartheta_S^{\text{CSC}^-} \\
 &= s \cos \ell_3^{\text{CSC}^-} - 2s \partial_\beta \vartheta_S^{\text{CSC}^-}.
 \end{aligned} \tag{40}$$

In this case, the partial derivatives of the sum of the lengths cancel out the contribution of the partial derivative of ϑ_S , yielding a simplified expression derived from (39) and (40):

$$\begin{aligned}
 \partial_\alpha (\ell_1^{\text{CSC}^-} + \ell_2^{\text{CSC}^-} + \ell_3^{\text{CSC}^-}) &= s(\cos \ell_1^{\text{CSC}^-} - 1), \\
 \partial_\beta (\ell_1^{\text{CSC}^-} + \ell_2^{\text{CSC}^-} + \ell_3^{\text{CSC}^-}) &= s(\cos \ell_3^{\text{CSC}^-} - 1).
 \end{aligned} \tag{41}$$

Substituting the lengths into their expression from (19), we can write the partial derivatives of the sum of the lengths from (41) as

$$\begin{aligned}
 \partial_\alpha (\ell^{\text{CSC}^-}) &= s(\cos(\vartheta_S^{\text{CSC}^-} - \alpha) - 1), \\
 \partial_\beta (\ell^{\text{CSC}^-}) &= s(\cos(\vartheta_S^{\text{CSC}^-} - \beta) - 1).
 \end{aligned} \tag{42}$$

The upper bound of the derivative and thus of the Lipschitz constant is 2.

4.3. Derivatives of the LRL and RLR Cases

Using equation (33) again, we have

$$2\ell_2^{\text{CCC}} = \ell_1^{\text{CCC}} + \ell_2^{\text{CCC}} + \ell_3^{\text{CCC}} - s(\beta - \alpha) \pmod{2\pi}, \tag{43}$$

and thus, deriving (43), we obtain

$$2\partial_\alpha \ell_2^{\text{CCC}} = \partial_\alpha \ell^{\text{CCC}} + s, \quad 2\partial_\beta \ell_2^{\text{CCC}} = \partial_\beta \ell^{\text{CCC}} - s. \tag{44}$$

Then, from (44) and in recalling Equation (33) and the expression of t^{CCC} from (27),

$$\begin{aligned}
 \partial_\alpha (\ell^{\text{CCC}}) &= 2\partial_\alpha (\ell_2^{\text{CCC}}) - s = \pm \frac{\sin(\alpha - \beta) + s d \cos \alpha}{2\sqrt{1 - (t^{\text{CCC}})^2}} - s, \\
 \partial_\beta (\ell^{\text{CCC}}) &= 2\partial_\alpha (\ell_2^{\text{CCC}}) + s = s \mp \frac{\sin(\alpha - \beta) + s d \cos \beta}{2\sqrt{1 - (t^{\text{CCC}})^2}}.
 \end{aligned} \tag{45}$$

The sign depends on whether the first or the second solution for ℓ_2 is used.

The results in (38), (42) and (45) are compactly implemented in Algorithm 1, which solves the classic point-to-point Dubins problem and returns the solution and its derivatives. This is instrumental for building the algorithm that solves the 3PDP presented in the next section.

Algorithm 1 Dubin’s solver with derivatives

```

1: function DUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ )
2:   Input:  $\mathbf{P}_i = (x_i, y_i), \vartheta_i, \mathbf{P}_f = (x_f, y_f), \vartheta_f, \kappa_{max}$ ;
3:   Transform to a standard problem with rotation, translation, and scaling
4:    $\phi \leftarrow \text{atan2}(y_f - y_i, x_f - x_i); d \leftarrow \|\mathbf{P}_i - \mathbf{P}_f\|/\kappa_{max}$  (2);  $\alpha \leftarrow \vartheta_i + \phi; \beta \leftarrow \vartheta_f + \phi$ ;
5:    $l_1, l_2, l_3, s_1, s_2, s_3, \partial_\alpha l, \partial_\beta l \leftarrow \text{DUBINSCSCP}(d, \alpha, \beta)$ 
6:    $l_1, l_2, l_3, s_1, s_2, s_3, \partial_\alpha l, \partial_\beta l \leftarrow \text{DUBINSCSCM}(d, \alpha, \beta)$ 
7:    $l_1, l_2, l_3, s_1, s_2, s_3, \partial_\alpha l, \partial_\beta l \leftarrow \text{DUBINSCCC}(d, \alpha, \beta)$ 
8:   Select the solution with minimal  $l_1 + l_2 + l_3$ 
9:   return  $l_1/\kappa_{max}, l_2/\kappa_{max}, l_3/\kappa_{max}, s_1, s_2, s_3, \partial_\alpha l/\kappa_{max}, \partial_\beta l/\kappa_{max}$ ;
10: end function

1: function DUBINSCSCP( $d, \alpha, \beta$ )
2:    $S^- \leftarrow \sin \alpha - \sin \beta; C^- \leftarrow \cos \alpha - \cos \beta$ ; (6)
3:   for  $s \in \{-1, 1\}$  do
4:      $\vartheta_S \leftarrow \text{atan2}(-C^-, d + sS^-) \bmod \pi$  (12)
5:      $l_1 \leftarrow s(\vartheta_S - \alpha) \bmod 2\pi$  (13)
6:      $l_2 \leftarrow (d + sS^-) \cos \vartheta_S - sC^- \sin \vartheta_S$  (13)
7:      $l_3 \leftarrow s(\beta - \vartheta_S) \bmod 2\pi$  (13)
8:      $l \leftarrow l_1 + l_2 + l_3$  (7)
9:      $\partial_\alpha l \leftarrow s(\cos(\vartheta_S - \alpha) - 1)$  (38)
10:     $\partial_\beta l \leftarrow s(1 - \cos(\beta - \vartheta_S))$  (38)
11:   end for
12:   Select the solution with minimal  $l$  and return  $l_1, l_2, l_3, s, 0, s, \partial_\alpha l, \partial_\beta l$ ;
13: end function

1: function DUBINSCSCM( $d, \alpha, \beta$ )
2:    $S^+ \leftarrow \sin \alpha + \sin \beta; C^+ \leftarrow \cos \alpha + \cos \beta$ ; (6)
3:   for  $s \in \{-1, 1\}$  do
4:     for  $X$  real root of  $(2 - C^+)X^2 + 2(s d + S^+)X + (2 + C^+) = 0$  do
5:        $\vartheta_S \leftarrow \text{atan2}(-2X, X^2 - 1)$  (12)
6:        $l_1 \leftarrow s(\vartheta_S - \alpha) \bmod (2\pi)$  (19)
7:        $l_2 \leftarrow (d + sS^+) \cos \vartheta_S - sC^+ \sin \vartheta_S$  (19)
8:        $l_3 \leftarrow s(\vartheta_S - \beta) \bmod (2\pi)$  (19)
9:        $l \leftarrow l_1 + l_2 + l_3$  (7)
10:       $\partial_\alpha l \leftarrow s(\cos(\vartheta_S - \alpha) - 1)$  (42)
11:       $\partial_\beta l \leftarrow s(\cos(\vartheta_S - \beta) - 1)$  (42)
12:     end for
13:   end for
14:   Select the solution with minimal  $l$  or set  $l_1 = l_2 = l_3 = \infty$  in case no solution exists.
15:   return  $l_1, l_2, l_3, s, 0, -s, \partial_\alpha l, \partial_\beta l$ ;
16: end function

1: function DUBINSCCC( $d, \alpha, \beta$ )
2:    $S^- \leftarrow \sin \alpha - \sin \beta$ ; (6)
3:   for  $s \in \{-1, 1\}$  do
4:      $t \leftarrow (3 + \cos(\alpha - \beta) - s d S^-)/4 - d^2/8$  (31)
5:     if  $t \in [-1, 1]$  then
6:       for  $\ell_2 \in \{\arccos(t), 2\pi - \arccos(t)\}$  (32) do
7:         solve linear system (30) and compute
8:          $l_1 \leftarrow \text{atan2}(\sin \ell_2, \cos \ell_2) \bmod 2\pi$  (33);
9:          $l_3 \leftarrow \text{atan2}(\sin \ell_2, \cos \ell_2) \bmod 2\pi$  (33);
10:         $\partial_\alpha l \leftarrow \pm(\sin(\alpha - \beta) + s d \cos \alpha)/(2\sqrt{1 - t^2}) - s$  (45)
11:         $\partial_\beta l \leftarrow s \mp (\sin(\alpha - \beta) + s d \cos \beta)/(2\sqrt{1 - t^2})$  (45)
12:       end for
13:     end if
14:     if not  $\alpha + s(l_1 - \ell_2 + l_3) = \beta \bmod 2\pi$  (34) (35) then
15:       Discard the solution
16:     end if
17:   end for
18:   return  $l_1, l_2, l_3, s, -s, s, \partial_\alpha l, \partial_\beta l$ ;
19: end function

```

5. Three-Point Dubins Problem (3PDP)

In this section, we derive the formulation for the 3PDP. The resulting Dubins spline will be a concatenation of two Dubins paths. Hence, the total length of such a spline will be the sum of the lengths of the two elementary Dubins paths:

$$\ell_{\text{total}}(\mathbf{P}_i, \mathbf{P}_m, \mathbf{P}_f, \vartheta_i, \vartheta_m, \vartheta_f) = \ell_{\text{dub1}}(\mathbf{P}_i, \mathbf{P}_m, \vartheta_i, \vartheta_m) + \ell_{\text{dub2}}(\mathbf{P}_m, \mathbf{P}_f, \vartheta_m, \vartheta_f), \quad (46)$$

where $\mathbf{P}_i = (x_i, y_i)$, $\mathbf{P}_m = (x_m, y_m)$, and $\mathbf{P}_f = (x_f, y_f)$ are the three points defining the 3PDP path, ϑ_i and ϑ_f are the initial and final angles of the path, and finally, ϑ_m is the angle of the middle point and the only unknown variable. Therefore, we will consider the total length ℓ_{total} as a function of ϑ_m only and also consider its derivative, ℓ'_{total} . Hence, (46) can be rewritten as

$$\begin{aligned} \ell_{\text{total}}(\vartheta_m) &= \ell_{\text{dub1}}(\vartheta_m) + \ell_{\text{dub2}}(\vartheta_m), \\ \ell'_{\text{total}}(\vartheta_m) &= \ell'_{\text{dub1}}(\vartheta_m) + \ell'_{\text{dub2}}(\vartheta_m). \end{aligned} \quad (47)$$

The derivatives are computed thanks to the results of the previous sections and, given (47), we have the following:

$$\begin{aligned} \ell'_{\text{dub1}}(\vartheta_m) &= \partial_{\vartheta_m} \ell_{\text{dub1}}(\mathbf{P}_i, \mathbf{P}_m, \vartheta_i, \vartheta_m) = \partial_{\beta} \ell_{\text{dub1}} \partial_{\vartheta_m} \beta_1, \\ \ell'_{\text{dub2}}(\vartheta_m) &= \partial_{\vartheta_m} \ell_{\text{dub2}}(\mathbf{P}_m, \mathbf{P}_f, \vartheta_m, \vartheta_f) = \partial_{\alpha} \ell_{\text{dub2}} \partial_{\vartheta_m} \alpha_2. \end{aligned} \quad (48)$$

Recalling that for a single segment $\vartheta_i = \alpha - \phi$ and $x_f = d$, $y_f = 0$, $\vartheta_f = \beta - \phi$, and $\phi = \text{atan2}(y_f - y_i, x_f - x_i)$, we obtain

$$\begin{aligned} \partial_{\vartheta_m} \beta_1 &= \partial_{\vartheta_m} (\vartheta_m + \phi_1) = 1, \\ \partial_{\vartheta_m} \alpha_2 &= \partial_{\vartheta_m} (\vartheta_m + \phi_2) = 1 \end{aligned} \quad (49)$$

However, $\phi_1 = \text{atan2}(y_m - y_i, x_m - x_i)$ and $\phi_2 = \text{atan2}(y_f - y_m, x_f - x_m)$ do not depend on ϑ_m and can be considered constants. Therefore, the derivatives of the lengths with respect to ϑ_m are computed using (48) and (49):

$$\ell'_{\text{total}}(\vartheta_m) = \partial_{\beta_1} \ell_{\text{dub1}}(\mathbf{P}_i, \mathbf{P}_m, \vartheta_i, \beta_1 - \phi_1) + \partial_{\alpha_2} \ell_{\text{dub2}}(\mathbf{P}_m, \mathbf{P}_f, \alpha_2 - \phi_2, \vartheta_f).$$

However, we cannot simply find the root of the previous equation because the lengths are piecewise continuous functions of ϑ_m . Therefore, we need to use a numerical method to find the optimal angle ϑ_m that minimizes the total length of the path.

Figure 2 illustrates the optimal configuration of the three-point Dubins connection. In the second subfigure, the total length is depicted as a function of the middle-point angle. This total length, which is the sum of the two individual lengths, can exhibit discontinuities. Depending on the problem’s boundary conditions, some configurations are feasible only for a limited range of internal angles. Additionally, the third subplot displays the behavior of the analytical derivative of the total length. Notably, there are jumps in the derivative even in regions where the total length appears to be smooth.

Therefore, finding the minimum length solution cannot rely on standard minimization or root-finding algorithms [19]. These traditional methods assume a certain level of smoothness and continuity in the target function and its derivatives. Therefore, we introduce a specialized section about algorithms designed to handle such irregularities effectively as in non-smooth optimization [18–20]. Furthermore, it is important to highlight that the minimum can occur at a jumping point where the derivative may differ from zero.

6. Algorithms

In this section, we present two algorithms employed to find the optimal angle ϑ_m that minimizes the total length of the three-point Dubins path. Both algorithms are based on a mixed approach that combines a pattern search method with a non-derivative root finder.

As highlighted in the previous section, the total length of the path can have discontinuities in the derivative. Therefore, the angle ϑ_m that minimizes the total length of the path cannot be found, in general, by simply computing the root of the derivative of the total length. This is true from a global perspective. However, if some regularity conditions are met in a specific region (interval), the root of the derivative can be used to find the optimal angle ϑ_m .

6.1. Discretization-Based Method (DBM [29])

This algorithm is the baseline used in the literature to compare new methods. It is based on the observation that once the angle ϑ_m is fixed, the corresponding length of the 3PDP is readily obtained by applying the classic Dubins solution two times, first from $\mathbf{P}_i, \vartheta_i$ to $\mathbf{P}_m, \vartheta_m$ and a second time from $\mathbf{P}_m, \vartheta_m$ to $\mathbf{P}_f, \vartheta_f$. The method discretizes the round angle with N points and selects as the solution of the 3PDP the angle that returns the minimum length. This is not very accurate and not very fast. Nevertheless, it is a straightforward method used as a baseline with standard value $N = 360$ and a discretization of one degree. It is described in Algorithm 2.

Algorithm 2 Discretization-Based Method (DBM [29])

```

1: function DUBINS3PDBM( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_f, \vartheta_f, \mathbf{P}_m, \kappa_{max}, N$ )
2:    $\vartheta_m \leftarrow 0$ ;  $\Delta\vartheta \leftarrow 2\pi/N$ ;  $\ell_{min} \leftarrow \infty$ ;
3:   while  $\vartheta_m < 2\pi$  do
4:      $S \leftarrow \text{TWODUBINS}(\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m, \mathbf{P}_f, \vartheta_f, \kappa_{max})$ ;
5:     if  $S.l < \ell_{min}$  then
6:        $\ell_{min} \leftarrow S.l$ ;  $C \leftarrow S$ ;    (* select this solution *)
7:     end if
8:      $\vartheta_m \leftarrow \vartheta_m + \Delta\vartheta$ ;
9:   end while
10:  return  $C$ ;    (* the structure of the selected solution *)
11: end function

1: function TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ )
2:   $S.A \leftarrow \text{DUBINS}(\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m, \kappa_{max})$ ;
3:   $S.B \leftarrow \text{DUBINS}(\mathbf{P}_m, \vartheta_m, \mathbf{P}_f, \vartheta_f, \kappa_{max})$ ;
4:   $S.l \leftarrow S.A.l + S.B.l$ ;    (* store total length *)
5:   $S.\vartheta_i \leftarrow \vartheta_i$ ;  $S.\vartheta_m \leftarrow \vartheta_m$ ;  $S.\vartheta_f \leftarrow \vartheta_f$ ;
6:  return  $S$ ;    (* return a structure with the two curves *)
7: end function

```

6.2. Pattern Search with Root Finder

Algorithm 3 combines the potential of a pattern search method with a non-derivative root finder. The pattern search method is used iteratively to refine the interval of angles where the total length has a local minimum. The iterative search stops when the interval is small enough or whenever sufficient conditions to use the non-derivative root finder are met. The non-derivative root finder yields the optimal angle ϑ_m inside the regular interval retrieved by the pattern search method. The sufficient conditions are that the solution does not change type inside the interval. If the derivatives at the interval's endpoints have opposite signs, then, by continuity, the minimum is inside the interval. In this case, a non-derivative root finder, which searches the zero of the derivative of the length, is used. The algorithm employed to find the root is a variant of the bisection with interpolation Algorithm 748 from [22].

Algorithm 3 Pattern Search Refinement (Based on Direct Search [37])

```

1: function DUBINS3PPATTERNREFINE( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m, \mathbf{P}_f, \vartheta_f, \kappa_{max}, \vartheta_m^L, \vartheta_m^C, \vartheta_m^R$ )
2:    $L \leftarrow \text{TWODUBINS}(\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m^L, \mathbf{P}_f, \vartheta_f, \kappa_{max});$ 
3:    $C \leftarrow \text{TWODUBINS}(\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m^C, \mathbf{P}_f, \vartheta_f, \kappa_{max});$ 
4:    $R \leftarrow \text{TWODUBINS}(\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m^R, \mathbf{P}_f, \vartheta_f, \kappa_{max});$ 
5:   while  $R.\vartheta_m - L.\vartheta_m > \Delta\vartheta_{min}$  do (* refine solution *)
6:     OPTIONAL: If curve type for  $L, C$  is the same
7:     and the length derivative changes sign, use algo 748
8:     to refine the solution between  $L.\vartheta_m$  and  $C.\vartheta_m$  and return.
9:
10:    OPTIONAL: If curve type for  $C, R$  is the same
11:    and the length derivative changes sign, use algo 748
12:    to refine the solution between  $C.\vartheta_m$  and  $R.\vartheta_m$  and return.
13:
14:     $\hat{L} \leftarrow \text{TWODUBINS}(\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, (L.\vartheta_m + C.\vartheta_m)/2, \mathbf{P}_f, \vartheta_f, \kappa_{max});$ 
15:     $\hat{R} \leftarrow \text{TWODUBINS}(\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, (R.\vartheta_m + C.\vartheta_m)/2, \mathbf{P}_f, \vartheta_f, \kappa_{max});$ 
16:    if  $\hat{L}.l < \hat{R}.l$  then
17:      if  $\hat{L}.l < C.l$  then
18:         $R \leftarrow C; C \leftarrow \hat{L};$ 
19:      else
20:         $L \leftarrow \hat{L}; R \leftarrow \hat{R};$ 
21:      end if
22:    else
23:      if  $\hat{R}.l < C.l$  then
24:         $L \leftarrow C; C \leftarrow \hat{R};$ 
25:      else
26:         $L \leftarrow \hat{L}; R \leftarrow \hat{R};$ 
27:      end if
28:    end if
29:  end while
30:  return  $C;$  (* best solution found *)
31: end function

```

Algorithm 4 (Pattern Trichotomy) is similar to the previous one with a slight variation. The main drawback of the pattern search method is that it can find only one local minimum of the total length. Algorithm 4 overcomes, in part, this limitation using a pattern search method to find multiple local minima of the total length. The algorithm iteratively searches for the local minima of the total length and stores them in a list of candidate minima. Then, the algorithm explores the candidates in a neighborhood of decreasing size and stops the search in two cases. The first is when the interval is small enough. The second case is when sufficient conditions are met, where the fast bisection can be used on the total length derivative. In both cases, the solution found is stored if it is better than a previously computed one.

The previously described methods (Algorithms 3 and 4) are not always able to catch the correct minima if there are discontinuities or jumps smaller than the first sampling range $\Delta\vartheta$. Our proposed hybrid method (Algorithm 5) overcomes this problem by isolating first the discontinuity points, and then adding at least two more points (before and after the jump). Furthermore, additional points are sampled according to the interval range $\Delta\vartheta$. The discontinuity isolation is performed by computing the existence range of the solution computed in (21), (34) and (35). The algorithm loops over all the sampled points, comparing the left and right samples. If the length at the central point is less or equal to the left and right lengths, we refine the solution. The algorithm returns the best solution found. The refinement phase is conducted by either the DUBINS3PPATTERNREFINE or DUBINS3PTRICHOTOMYREFINE functions described in Algorithms 3 and 4.

Algorithm 4 Trichotomy Refine (Based on Trichotomy Method [36])

```

1: function DUBINS3PTRICHOTOMYREFINE( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m, \mathbf{P}_f, \vartheta_f, \kappa_{max}, \vartheta_m^L, \vartheta_m^C, \vartheta_m^R$ )
2:    $L \leftarrow$  TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m^L, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ );
3:    $C \leftarrow$  TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m^C, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ );
4:    $R \leftarrow$  TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m^R, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ );
5:   while  $R.\vartheta_m - L.\vartheta_m > \Delta\vartheta_{min}$  do (* refine solution *)
6:     OPTIONAL: If curve type for  $L, C$  is the same
7:       and the length derivative change signs, use algo 748
8:       to refine the solution between  $L.\vartheta_m$  and  $C.\vartheta_m$  and return.
9:
10:    OPTIONAL: If curve type for  $C, R$  is the same
11:     and the length derivative changes sign, use algo 748
12:     to refine the solution between  $C.\vartheta_m$  and  $R.\vartheta_m$  and return.
13:
14:     $\widehat{L} \leftarrow$  TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, (L.\vartheta_m + 2C.\vartheta_m)/3, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ );
15:    if  $\widehat{L}.\ell \leq C.\ell$  then
16:       $\widetilde{L} \leftarrow$  TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, (2L.\vartheta_m + C.\vartheta_m)/3, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ );
17:      if  $\widetilde{L}.\ell \leq \widehat{L}.\ell$  then
18:         $C \leftarrow \widetilde{L}; R \leftarrow \widehat{L};$ 
19:      else
20:         $L \leftarrow \widetilde{L}; R \leftarrow C; C \leftarrow \widehat{L};$ 
21:      end if
22:    else
23:       $\widehat{R} \leftarrow$  TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, (R.\vartheta_m + 2C.\vartheta_m)/3, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ );
24:      if  $\widehat{R}.\ell \leq C.\ell$  then
25:         $\widetilde{R} \leftarrow$  TWODUBINS( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, (2R.\vartheta_m + C.\vartheta_m)/3, \mathbf{P}_f, \vartheta_f, \kappa_{max}$ );
26:        if  $\widetilde{R}.\ell \leq \widehat{R}.\ell$  then
27:           $L \leftarrow \widetilde{R}; C \leftarrow \widehat{R};$ 
28:        else
29:           $L \leftarrow C; C \leftarrow \widehat{R}; R \leftarrow \widetilde{R};$ 
30:        end if
31:      else
32:         $L \leftarrow \widehat{L}; R \leftarrow \widehat{R};$ 
33:      end if
34:    end if
35:  end while
36:  return  $C;$  (* best solution found *)
37: end function

```

Algorithm 5 Hybrid Search

```

1: function DUBINS3PHYBRIDSEARCH( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_f, \vartheta_f, \mathbf{P}_m, \kappa_{max}, \Delta\vartheta$ )
2:    $\vartheta_d \leftarrow$  Compute the candidate discontinuity points
3:    $\vartheta_s \leftarrow$  sample points (at least 2 close to  $\vartheta_d$  with distance less or equal to  $\Delta\vartheta$ )
4:   for all  $\vartheta_m \in \vartheta_c$  do
5:      $\vartheta_m^L, \vartheta_m^R \leftarrow$  left and right sample points of  $\vartheta_m$ 
6:     if  $\ell(\vartheta_m) \leq \ell(\vartheta_m^L)$  and  $\ell(\vartheta_m) \leq \ell(\vartheta_m^R)$  then
7:        $C \leftarrow$  REFINE( $\mathbf{P}_i, \vartheta_i, \mathbf{P}_m, \vartheta_m, \mathbf{P}_f, \vartheta_f, \kappa_{max}, \vartheta_m^L, \vartheta_m, \vartheta_m^R$ )
8:     end if
9:   end for
10:  return  $C;$  (* the structure of the best computed solution *)
11: end function

```

where REFINE can be DUBINS3PPATTERNREFINE or DUBINS3PTRICHOTOMYREFINE

7. Numerical Tests and Results

In this section, we briefly present the numerical results and performance achievements. The tests were conducted on a commercial laptop with an Apple M2 Max processor. The test was performed on a sampled pool of 10,000 random three-point Dubins problems, which is the standard benchmark test for the 3PDP found in the literature. The results are summarized in Tables 2–5.

We employed two different ranges of sampling, the first coming from the literature and representing a standard benchmark case, whereas the second was more representative of the solution-type distribution. Without loss of generality, we kept the initial and final points fixed at $(-1, 0)$ and $(1, 0)$, respectively.

In the first sampling, we took the middle point uniformly distributed in $[-10, 10] \times [-10, 10]$. The initial and final angles were in the interval $[-\pi, \pi]$, and the maximum curvature was kept constant at 1. The same test was conducted in [26,28,31]. In summary,

$$(x_m, y_m) \in [-10, 10] \times [-10, 10], \quad \theta_i, \theta_f \in [-\pi, \pi], \quad \text{and} \quad \kappa_{max} = 1, \quad (50)$$

The second test sampled the middle point randomly inside the region $[-2, 2] \times [-2, 2]$. As before, the initial and final angles were randomly sampled in the interval $[-\pi, \pi]$, and we randomly generated the maximum curvature inside the interval $[0.1, 1.5]$. In brief,

$$(x_m, y_m) \in [-2, 2] \times [-2, 2], \quad \theta_i, \theta_f \in [-\pi, \pi], \quad \text{and} \quad \kappa_{max} \in [0.1, 1.5]. \quad (51)$$

We proposed the second test because, when the middle point is distant from the initial and the final points, we lose two potential solutions of the 3PDP. When the distance is greater than the maximum curvature inverse, we can only observe 4 of the 18 possible solutions, which are the ones with the straight line in the middle.

In Tables 2 and 4, we present the mean and standard deviation of both the number of iterations and computational time for the sampling in (50) and (51), respectively. The last columns of the tables present the number of calls to the Dubins function, on average. It is important to highlight that the number of iterations is linked to the number of Dubins calls (twice). The information is redundant, but most of the available state-of-the-art publications report the number of Dubins calls. The computational time is measured in milliseconds. It is important to stress that the computational time depends on the implementation.

Tables 3 and 5 present a comparison between different approaches scaling the results with the DBM benchmark. The ratio is computed as the number of iterations and time of the algorithm divided by the number of iterations and time of the DBM algorithm. The results are presented for the sampling in (50) and (51), respectively.

The four tables (Tables 2–5) showcase the strong performance of our algorithm when the middle point is distant, on average. In fact, in this occurrence, both the length and the derivative do not jump; hence, non-smooth root-finding algorithms can be used to obtain the minimum length curve.

The results show that the proposed algorithms outperform the benchmark algorithms in terms of the number of iterations and time. The pattern search with the root finder and the pattern trichotomy algorithms are more efficient than the brute force (DBM) and refinement (iDDP) algorithms by a factor ranging between 4.27 and 6.67 times depending on the proposed sampling. Moreover, it should be noted that these methods do not achieve the same precision: indeed, the DBM evaluates the problem with a 1-degree precision, while the other methods have a precision of 10^{-2} degrees. However, the threshold precision can be set to a custom value in our proposed methods.

The Pattern Trichotomy algorithm is the most efficient one. It requires the fewest iterations and the least amount of time. The pattern search algorithm is slightly less efficient than the Pattern Trichotomy algorithm. However, it is still more efficient than the DBM, IM, PBM, and iDDP algorithms. The pattern search algorithm combined with the root finder (algorithm 748) requires fewer iterations on average. However, the trichotomy algorithm implementation allows a speed-up in the computational time. Hence, the pattern

trichotomy algorithm equipped with a root finder is the fastest and most efficient. Furthermore, when root finding is employed, both pattern search with algorithm 748 and Pattern Trichotomy with algorithm 748 yield the same exact result up to machine precision.

Looking at the comparison in Table 5, we can see that the pattern search and Pattern Trichotomy algorithms require approximately 4 times fewer iterations than the DBM algorithm, and 2.8 times with respect to the iDPP. However, if we compare the time required by the algorithms, the pattern search exhibits a factor of 270, and the Pattern Trichotomy, a factor of 320. Other speed-up factors (compared to the brute force DBM) declared in the literature are about 3 for the iDPP [12], which is a general method capable of solving the MPMDP with n points, adapted for the case $n = 3$; about 5 for the inversive geometry iterative method (IM) [26]; between 25 and 45 for the polynomial-based method (PBM) [31]; and about 150 for the geometry-based method (GBM) [28], (which solves exactly the case of points at distance $d > d^*$ only, and gives a good approximation otherwise). Therefore, we can conclude that our methods are performant.

Figure 3 illustrates the optimal configuration of the three-point Dubins connection in a challenging configuration. In the second subfigure, the total length showcases strong discontinuities resulting in a tiny interval of existence of the retrieved minimum. Furthermore, the derivative displays many noncontinuous points, hence demonstrating the non-applicability of derivative methods, at least globally.

Table 2. The evaluation of the proposed algorithms with the benchmark test with mean μ and standard deviation σ of iterations and computational times for the sampling in (50).

| Algorithm | Iter Number | | Time (ms) | | N. Dubins Calls |
|--------------------------|-------------|----------|-----------|----------|-----------------|
| | μ | σ | μ | σ | |
| Pattern search | 85.54 | 10.66 | 0.174 | 0.036 | ~171 |
| Pattern search + 748 | 46.27 | 10.72 | 0.122 | 0.031 | ~92 |
| Pattern Trichotomy | 73.69 | 7.97 | 0.150 | 0.026 | ~147 |
| Pattern Trichotomy + 748 | 54.99 | 8.74 | 0.115 | 0.025 | ~109 |
| DBM – Algorithm 2 | 360 | 0.0 | 54.846 | 3.210 | 720 |
| iDPP [12] | 128 | 0.0 | 19.455 | 0.740 | 256 |

Table 3. Comparison between number of iterations and time with respect to brute force DBM for sampling in (50). (n/a means not available.)

| Algorithm | Iter Number Ratio | Time Ratio |
|--------------------------|-------------------|------------|
| Pattern search | 4.27 | 325.66 |
| Pattern search + 748 | 6.57 | 471.78 |
| Pattern Trichotomy | 4.94 | 374.73 |
| Pattern Trichotomy + 748 | 6.67 | 493.50 |
| iDPP [12] | 2.81 | 2.82 |
| IM [26] | n/a | 5 |
| PBM [31] | n/a | [25, 45] |
| GBM [28] | n/a | 150 |

We compared the example proposed in [28,31], where they tested their implementation versus the DBM. The test example has the following definition:

$$\mathbf{P}_i = (0,0), \quad \theta_i = \frac{\pi}{3}, \quad \mathbf{P}_m = (10,5), \quad \mathbf{P}_f = (15,20), \quad \theta_f = \frac{\pi}{6}. \quad (52)$$

The DBM yields the following solution:

$$\ell_{\text{total}} = 27.11279494, \quad \theta_M = 0.8575935377. \quad (53)$$

This is a suboptimal solution, as highlighted in [28]. Their work claims to obtain a solution that is 1.08×10^{-6} % better than that of the DBM. Our approach provides solutions with the following values, reported in Table 6 and Figure 4, representing our best solution. The DBM performs 360 iterations and will need to perform 100 times more to achieve the same precision as in [28]. Moreover, the pattern search and Pattern Trichotomy algorithms stop the iteration when a custom tolerance is reached (1/100 degree). On the other hand, when combined with Algorithm 748, our approach retrieves the solution to machine precision, yielding the true numerical minimum.

Table 4. The evaluation of the proposed algorithms with the benchmark test with mean μ and standard deviation σ of iterations and computational times for the sampling in (51).

| Algorithm | Iter Number | | Time (ms) | | N. Dubins Calls |
|--------------------------|-------------|----------|-----------|----------|-----------------|
| | μ | σ | μ | σ | |
| Pattern search | 88.35 | 20.57 | 0.217 | 0.047 | ~176 |
| Pattern search + 748 | 77.70 | 23.55 | 0.187 | 0.031 | ~155 |
| Pattern Trichotomy | 74.78 | 13.34 | 0.179 | 0.032 | ~149 |
| Pattern Trichotomy + 748 | 68.54 | 14.81 | 0.163 | 0.034 | ~137 |
| DBM – Algorithm 2 | 360 | 0.0 | 55.577 | 3.357 | 720 |
| iDPP [12] | 128 | 0.0 | 19.720 | 0.845 | 256 |

Table 5. Comparison between number of iterations and time with respect to brute force DBM for sampling in (51). (n/a means not available.)

| Algorithm | Iter Number Ratio | Time Ratio |
|--------------------------|-------------------|------------|
| Pattern search | 4.27 | 265.66 |
| Pattern search + 748 | 5.02 | 315.19 |
| Pattern Trichotomy | 4.95 | 318.01 |
| Pattern Trichotomy + 748 | 5.47 | 354.27 |
| iDPP [12] | 2.81 | 2.82 |
| IM [26] | n/a | 5 |
| PBM [31] | n/a | [25, 45] |
| GBM [28] | n/a | 150 |

Table 6. Comparison of result with the best result (pattern search + 748 and Pattern Trichotomy + 748).

| Algorithm | Length | θ_M |
|--------------------------|--------------------------|-------------------------|
| Pattern search + 748 | 27.1127934 | 0.8556738609 |
| Pattern Trichotomy + 748 | +0.0 | +0.0 |
| Pattern search | $+1.552 \times 10^{-11}$ | $+3.711 \times 10^{-5}$ |
| Pattern Trichotomy | $+8.985 \times 10^{-13}$ | $+8.929 \times 10^{-6}$ |
| DBM – Algorithm 2 | $+5.672 \times 10^{-8}$ | $+2.243 \times 10^{-3}$ |
| iDPP [12] | +0.0 | $+1.510 \times 10^{-8}$ |

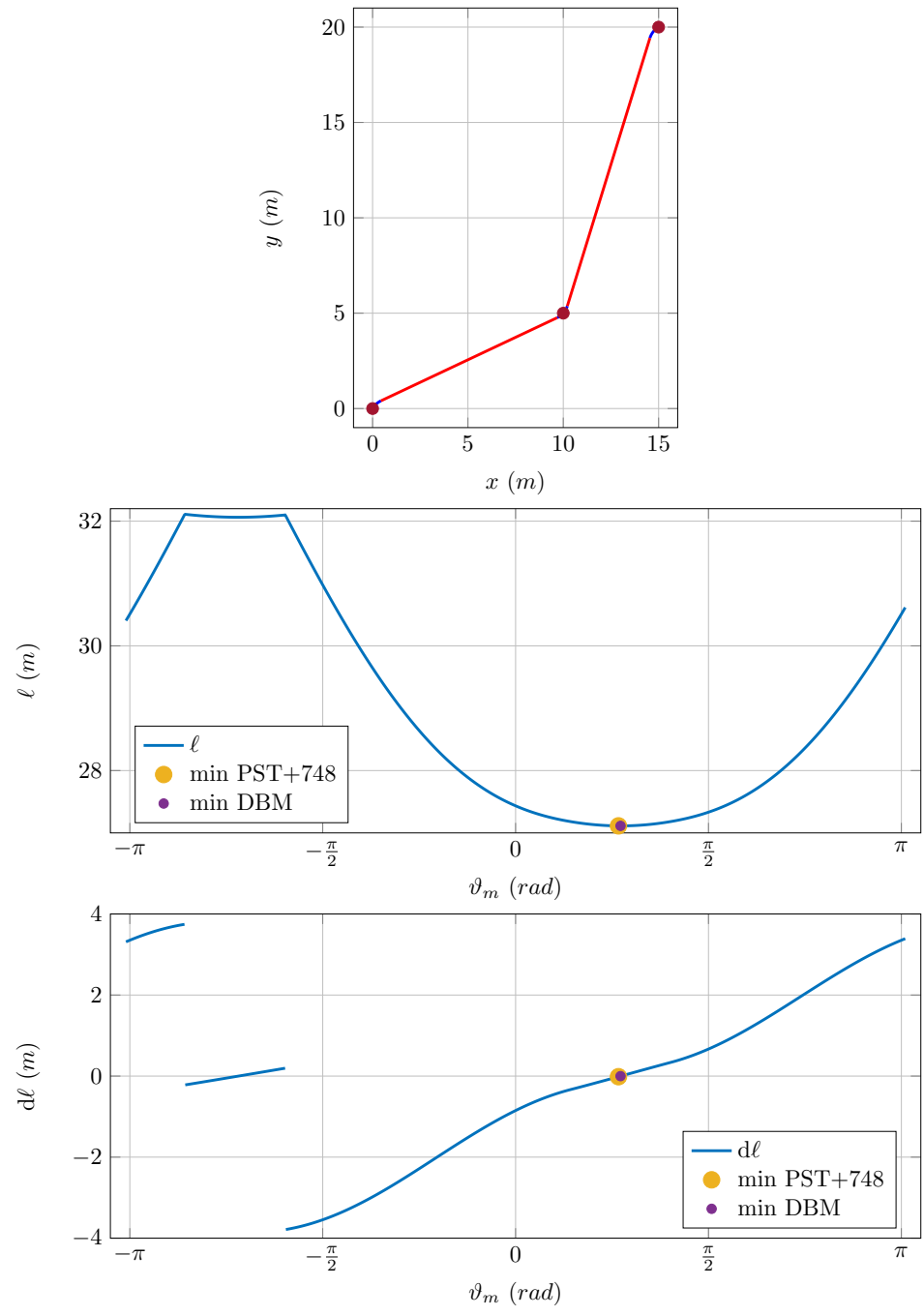


Figure 4. An example of the total length of a three-point Dubins path for comparison with [28]. (**Top:** path result with alternating colors for each arc; **Middle:** total length ℓ as a function of the middle angle ϑ_m ; **Bottom:** derivative of total length ℓ as a function of the middle angle ϑ_m .)

8. Conclusions

We presented two novel algorithms for solving the 3PDP, a non-smooth constrained minimization problem and an optimal control problem. Our methods leverage the pattern search technique, integrating analytic derivatives within continuous-length function intervals to enhance computational efficiency. This improvement is crucial for the 3PDP and may also be used for the Dubins Traveling Salesman Problem. In these examples, high performance is essential due to the routine’s frequent invocation.

Extensive numerical tests were conducted on a standard benchmark pool of 10,000 random Three-Point Dubins Problems. Two different sampling ranges were employed to reflect standard benchmarks and more representative scenarios with a middle point close to the

initial and final points. The results demonstrate that our proposed algorithms significantly outperform existing methods in terms of both iteration count and computational time.

For the first sampling range, our pattern search and Pattern Trichotomy algorithms achieved an iteration count reduction factor of approximately 4.27 to 6.674 times compared to the DBM algorithm, and a time reduction factor ranging from ~ 325 to ~ 493 times. For the second sampling range, the iteration count reduction factors were similar with a range of 4.27 to 5.47 times, with the time reduction factors being 265 to 354 times compared to the DBM algorithm. Additionally, when combined with Algorithm 748, both the pattern search and Pattern Trichotomy methods consistently yielded results with machine precision, demonstrating their robustness and accuracy.

These performance improvements are notable when compared to other state-of-the-art methods, such as the polynomial-based method (PBM) in [31], which achieves a 25–45 times speed-up over the DBM, and the geometry-based method (GBM) in [28], which offers a speed-up of approximately 150 times. Our methods achieve a computational speed-up factor exceeding 300 times compared to the DBM and less than half the time declared in [28].

Overall, our methods represent a significant advancement in solving the 3PDP efficiently, making them highly suitable for applications where computational performance is critical.

9. Future Works

Our research is ongoing, and this work is part of a broader project. We are working on the generalization of the 3PDP to the Multipoint Markov–Dubins problem (MPMDP). The goal is to provide a general and fast method to solve these Dubins-related challenging problems encountered in robotics. Our research aims to find a solution that does not rely on finite meshing exploration.

Moreover, we are working on domain-specific applications for the 3PDP. We are including our implementation into an autonomous parking framework [38]. The solution to the 3PDP is used to efficiently compute an exploration trajectory with obstacle collision checking. Furthermore, we are extending the framework developed in [4] to include low-speed maneuvers with the 3PDP. The goal is to employ a piecewise constant curvature profile for *valet* maneuvers.

Author Contributions: Conceptualization, M.P., E.B. and M.F.; Methodology, M.P., E.B. and M.F. The authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, C.; Zhong, Z.; Xiang, X.; Zhu, Y.; Wu, L.; Yin, D.; Li, J. UAV Path Planning in Multi-Task Environments with Risks through Natural Language Understanding. *Drones* **2023**, *7*, 147. <https://doi.org/10.3390/drones7030147>.
2. Perez-Ramos, J.L.; Ramirez-Rosales, S.; Canton-Enriquez, D.; Diaz-Jimenez, L.A.; Xicotencatl-Ramirez, G.; Herrera-Navarro, A.M.; Jimenez-Hernandez, H. Algorithm Based on Morphological Operators for Shortness Path Planning. *Algorithms* **2024**, *17*, 184. <https://doi.org/10.3390/a17050184>.
3. Latif, E.; Parasuraman, R. On the Intersection of Computational Geometry Algorithms with Mobile Robot Path Planning. *Algorithms* **2023**, *16*, 498. <https://doi.org/10.3390/a16110498>.
4. Piazza, M.; Piccinini, M.; Taddei, S.; Biral, F. MPTREE: A Sampling-based Vehicle Motion Planner for Real-time Obstacle Avoidance. *IEAC-PapersOnLine* **2024**, *58*, 146–153. <https://doi.org/10.1016/j.ifacol.2024.07.332>.
5. Chitsaz, H.; LaValle, S.M. Time-optimal paths for a Dubins airplane. In Proceedings of the 2007 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007; pp. 2379–2384. <https://doi.org/10.1109/CDC.2007.4434966>.

6. Scharff Willners, J.; Gonzalez-Adell, D.; Hernández, J.D.; Pairet, È.; Petillot, Y. Online 3-Dimensional Path Planning with Kinematic Constraints in Unknown Environments Using Hybrid A* with Tree Pruning. *Sensors* **2021**, *21*, 1152. <https://doi.org/10.3390/s21041152>.
7. Markov, A.A. Some examples of the solution of a special kind of problem on greatest and least quantities. *Soobshch. Karkovsk. Mat. Obshch* **1887**, *1*, 250–276.
8. Kaya, C.Y. Markov–Dubins path via optimal control theory. *Comput. Optim. Appl.* **2017**, *68*, 719–747. <https://doi.org/10.1007/s10589-017-9923-8>.
9. Dubins, L.E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *Am. J. Math.* **1957**, *79*, 497–516.
10. Park, S. Three-Dimensional Dubins-Path-Guided Continuous Curvature Path Smoothing. *Appl. Sci.* **2022**, *12*, 11336. <https://doi.org/10.3390/app122211336>.
11. Consonni, C.; Brugnara, M.; Bevilacqua, P.; Tagliaferri, A.; Frego, M. A new Markov–Dubins hybrid solver with learned decision trees. *Eng. Appl. Artif. Intell.* **2023**, *122*, 106166.
12. Frego, M.; Bevilacqua, P.; Saccon, E.; Palopoli, L.; Fontanelli, D. An Iterative Dynamic Programming Approach to the Multipoint Markov–Dubins Problem. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2483–2490.
13. Kaya, C.Y. Markov–Dubins interpolating curves. *Comput. Optim. Appl.* **2019**, *73*, 647–677. <https://doi.org/10.1007/s10589-019-00076-y>.
14. Saccon, E.; Bevilacqua, P.; Fontanelli, D.; Frego, M.; Palopoli, L.; Passerone, R. Robot motion planning: Can GPUs be a game changer? In Proceedings of the 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), Virtual, 12–16 July 2021; pp. 21–30.
15. Isaacs, J.T.; Hespánha, J.P. Dubins Traveling Salesman Problem with Neighborhoods: A Graph-Based Approach. *Algorithms* **2013**, *6*, 84–99. <https://doi.org/10.3390/a6010084>.
16. Nayak, A.; Rathinam, S. Heuristics and Learning Models for Dubins MinMax Traveling Salesman Problem. *Sensors* **2023**, *23*, 6432. <https://doi.org/10.3390/s23146432>.
17. Li, L.; Shi, D.; Jin, S.; Yang, S.; Zhou, C.; Lian, Y.; Liu, H. Exact and Heuristic Multi-Robot Dubins Coverage Path Planning for Known Environments. *Sensors* **2023**, *23*, 2560. <https://doi.org/10.3390/s23052560>.
18. Mäkelä, M.M. On the methods of nonsmooth optimization. In *System Modelling and Optimization*; Sebastian, H.J., Tammer, K., Eds.; Berlin/Heidelberg, Germany, 1990; pp. 177–186.
19. Bagirov, A.; Karmitsa, N.; Mäkelä, M.M. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 12.
20. Bagirov, A.M.; Karmitsa, N.; Taheri, S. *Partitional Clustering via Nonsmooth Optimization*; Springer Nature: Cham, Switzerland, 2020.
21. Holmes, M.H. *Introduction to Numerical Methods in Differential Equations*; Springer: Berlin/Heidelberg, Germany, 2007.
22. Alefeld, G.E.; Potra, F.A.; Shi, Y. Algorithm 748: Enclosing zeros of continuous functions. *ACM Trans. Math. Softw. (TOMS)* **1995**, *21*, 327–344. <https://doi.org/10.1145/210089.210111>.
23. Bertolazzi, E.; Bevilacqua, P.; Frego, M. Clothoids: A C++ library with MATLAB interface for the handling of clothoid curves. *Rend. Del Semin. Mat.* **2018**, *76*, 47–56.
24. Bertolazzi, E.; Frego, M. Clothoids: A C++ Library with Matlab Interface. 2019. Available online: <https://github.com/ebertolazzi/Clothoids> (accessed on 1 June 2024).
25. Goac, X.; Kim, H.S.; Lazard, S. Bounded-Curvature Shortest Paths through a Sequence of Points. Research Report RR-7465, INRIA, HAL ID:inria-00539957. 2010. Available online: <https://inria.hal.science/inria-00539957> (accessed on 1 June 2024).
26. Sadeghi, A.; Smith, S.L. On efficient computation of shortest Dubins paths through three consecutive points. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 6010–6015. <https://doi.org/10.1109/CDC.2016.7799192>.
27. Goac, X.; Kim, H.S.; Lazard, S. Bounded-Curvature Shortest Paths through a Sequence of Points Using Convex Optimization. *SIAM J. Comput.* **2013**, *42*, 662–684. <https://doi.org/10.1137/100816079>.
28. Parlange, G.; De Palma, D.; Attanasi, R. A novel approach for 3PDP and real-time via point path planning of Dubins’ vehicles in marine applications. *Control Eng. Pract.* **2024**, *144*, 105814. <https://doi.org/10.1016/j.conengprac.2023.105814>.
29. Chen, Z.; Shima, T. Relaxed Dubins problems through three points. In Proceedings of the 2019 27th Mediterranean Conference on Control and Automation (MED), Akko, Israel, 1–4 July 2019; pp. 501–506.
30. Cohen, I.; Epstein, C.; Isaiah, P.; Kuzi, S.; Shima, T. Discretization-based and look-ahead algorithms for the dubins traveling salesperson problem. *IEEE Trans. Autom. Sci. Eng.* **2016**, *14*, 383–390.
31. Chen, Z.; Shima, T. Shortest Dubins paths through three points. *Automatica* **2019**, *105*, 368–375. <https://doi.org/10.1016/j.automatica.2019.04.007>.
32. Bertolazzi, E.; Frego, M. A Note on Robust Biarc Computation. *Comput.-Aided Des. Appl.* **2019**, *16*, 822–835. <https://doi.org/10.14733/cadaps.2019.822-835>.
33. Hooke, R.; Jeeves, T.A. “Direct Search” Solution of Numerical and Statistical Problems. *J. ACM* **1961**, *8*, 212–229. <https://doi.org/10.1145/321062.321069>.
34. Custódio, A.L.; Madeira, J.A. GLODS: Global and local optimization using direct search. *J. Glob. Optim.* **2015**, *62*, 1–28.

35. Audet, C. *A Survey on Direct Search Methods for Blackbox Optimization and Their Applications*; Springer: Berlin/Heidelberg, Germany, 2014.
36. Antonova, A.; Ibryaeva, O.L. A new zero-order 1-D optimization algorithm: Trichotomy method. *arXiv* **2019**, arXiv:1903.07117.
37. Audet, C.; Dennis, J.E., Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.* **2006**, *17*, 188–217.
38. Pagot, E.; Piccinini, M.; Bertolazzi, E.; Biral, F. Fast Planning and Tracking of Complex Autonomous Parking Maneuvers With Optimal Control and Pseudo-Neural Networks. *IEEE Access* **2023**, *11*, 124163–124180.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.