applied sciences

MDPI

Article

Three-Dimensional Dubins-Path-Guided Continuous Curvature Path Smoothing

Sungsu Park

Department of Aerospace Engineering, Sejong University, Seoul 05006, Korea; sungsu@sejong.ac.kr

Abstract: This paper presents an efficient three-dimensional (3D) Dubins path design and a new continuous curvature path-smoothing algorithm. The proposed 3D Dubins path is a simple extension of the 2D path and serves as a reference path generator to guide the proposed smoothing algorithm. In the smoothing algorithm, the reference path is approximated by several cubic Bezier curves to generate a parametric path that simultaneously satisfies curvature continuity and maximum curvature requirements. The algorithm also provides a criterion to select a required number of Bezier curves to strictly meet the maximum curvature constraint. Therefore, our algorithm can be used for 3D path-following applications in many areas such as aerospace and robotics. The numerical simulation results show that the proposed algorithm produces a continuous curvature path that passes through all waypoints with a slight increase in path length compared with the original 3D Dubins reference path.

Keywords: Bezier curve; continuous curvature; Dubins path; path planning; path smoothing; waypoint

check for updates

Citation: Park, S. Three-Dimensional Dubins-Path-Guided Continuous Curvature Path Smoothing. *Appl. Sci.* **2022**, *12*, 11336. https://doi.org/10.3390/app122211336

Academic Editor: Oscar Reinoso García

Received: 11 October 2022
Accepted: 7 November 2022
Published: 8 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

The path planning problem is to find a feasible obstacle-free path that connects the starting point to the goal point. Many path planning algorithms have been presented over the last two decades. Sampling-based algorithms such as probabilistic roadmaps (PRM) and rapidly exploring random tree (RRT) have received great attention because they can search for feasible paths relatively quickly, even in a high-dimensional space. An extensive comparative study and analysis of various sampling-based algorithms can be found in the literature [1–5].

Even though a sampling-based path planning algorithm is effective and computationally efficient, it typically generates an unnatural path because of its randomness. It thus requires a post-processing procedure called path pruning and smoothing. Path pruning is to remove unnecessary waypoints which are generated by a sampling-based path planning algorithm. While the number of waypoints is reduced after path pruning, the path is still piecewise linear and usually not followable for a moving vehicle with dynamic constraints. Therefore, it is necessary to smooth the path to making it suitable for a vehicle [6–10].

A smooth path is preferable since the curvature discontinuity results in a discontinuity in the vehicle's desired acceleration command, and these cause tracking errors. Although some mobile robots and flying vehicles such as multicopters can stop and hover at a certain point, this increases the driving or flight time and generally degrades the navigation performance. For a fixed-wing aircraft, it is not possible to follow a piecewise linear path without tracking errors.

A popular method to generate a smooth path is the Dubins path [11–16]. The Dubins path concatenates straight lines and circular arcs considering the vehicle's minimum turning radius. It generates the shortest path between two waypoints with the prescribed pose (or direction vector) to the path in a plane. Another method is the cubic spline method that produces a smooth path of C1 continuity, as with the Dubins path. However, C1 continuity

is not smooth enough, and even further smoothness of geometric continuity of 2nd order (G2 continuity) is required with maximum curvature constraint to be executed smoothly by moving vehicles [6]. Here, C1 continuity means continuity of the tangent vector, and G2 continuity means continuity of the curvature. Although the clothoid path provides a G2 continuous path, it is not a closed-form expression. It requires high computational cost since the evaluation of Fresnel integrals is needed to generate a path [17,18].

While three-dimensional (3D) extensions of the Dubins path have been proposed in the literature [13–16], they are very complex and difficult to use in practical applications because they are based on a 3D constant-speed point mass model or the decoupled horizontal and vertical plane model. Some G2 continuous paths are also proposed in the literature [7,17,19] based on the Bezier curve, but they generate paths that pass through only the starting and final waypoints.

Motivated by these observations, this paper proposes a 3D Dubins-path-guided continuous curvature path design algorithm. Our contribution is threefold. The first is to extend the 2D Dubins path method to 3D, which is conceptually simple and computationally efficient. The second is to use multiple cubic Bezier curves to smooth the Dubins path at the transition points where curvature and path direction discontinuities occur. The proposed smoothing algorithm can generate continuous curvature paths in 3D space that pass through all waypoints without limiting the size of the turning angle. On the other hand, existing algorithms have a limit on the size of the turning angle and do not pass through all waypoints. The third is to provide a criterion to select a required number of Bezier curves to strictly meet the maximum curvature constraint. In this way, our algorithm can generate a 3D continuous curvature path only with information of the three sequential waypoints or two successive waypoints having one direction vector.

This paper is organized as follows. In the next section, the 2D Dubins path is described, and its extension to the 3D path is proposed. Section 3 proposes a new continuous curvature path-smoothing algorithm with a two-step approach, which is a path approximation by several cubic Bezier curves after 3D Dubins reference path generation. In Section 4, numerical simulation results are provided to demonstrate the performance of the proposed algorithm. Conclusions are given in Section 5.

2. Three-Dimensional Dubins Path

The Dubins path is the shortest path between two points with a specific direction of motion in a plane. It is either a CSC or a CCC path, where C represents circular arc and S represents straight line. The CCC path is formed by three consecutive circular arcs that are tangential to each other, and the CSC is composed of a straight line and two circular arcs, where the radius of the circular arc is chosen equal to the minimum radius of rotation of a vehicle which corresponds to a given maximum curvature constraint. We only consider a CSC path since the CCC path is complex and known as suboptimal in most cases, except that the distance between the two waypoints is too close [20].

In this section, we present a new 3D Dubins path method that extends the 2D Dubins path by applying a simple constraint to the pose. Our method is straightforward for waypoint guidance problems.

Suppose that $\mathbf{p}_i, i = 1, ..., N$ is a sequence of waypoints in a 3D space, which might be generated by a path planner or pruner considering obstacle avoidance or other constraints. The problem is to fit these waypoints with the shortest path satisfying the minimum radius of rotation R.

For each waypoint $\mathbf{p}_i$, we define a pose or direction vector $\mathbf{e}_i$ as an unit vector pointing to the next waypoint $\mathbf{p}_{i+1}$.

$$\mathbf{e}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2} \tag{1}$$

At the final waypoint $\mathbf{p}_N$, a direction vector is designated as an approaching angle to the waypoint.

This definition of a direction vector is very natural in the waypoint guidance problem and is the only constraint, which makes it possible to extend a 2D to a 3D Dubins path. In this way, all three sequential waypoint tuples $\{\mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}\}$ and their associated two-direction vectors $\{\mathbf{e}_i, \mathbf{e}_{i+1}\}$ are located on the same plane $plane_i$. The consecutive waypoint tuple $\{\mathbf{p}_{i+1}, \mathbf{p}_{i+2}, \mathbf{p}_{i+3}\}$ and their associated direction vectors $\{\mathbf{e}_{i+1}, \mathbf{e}_{i+2}\}$ are also all on another plane $plane_{i+1}$. Moreover, the waypoint segment line seg_{i+1}, which connects the waypoints $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i+2}$, becomes the intersection line of two adjacent planes, $plane_i$ and $plane_{i+1}$, as shown in Figure 1.

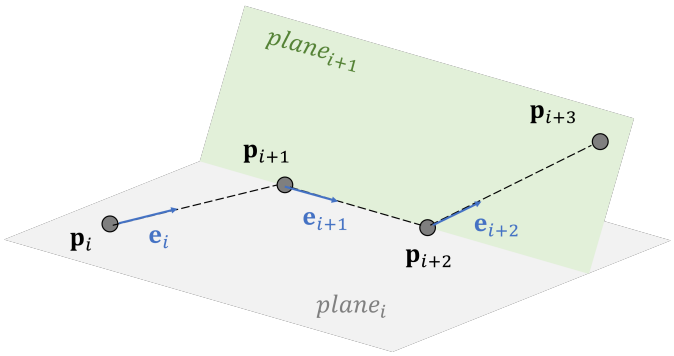

Figure 1. Waypoints and planes.

2.1. Two-Dimensional Dubins Path

This subsection describes the 2D Dubins path. Much of this material is found elsewhere but is re-represented using unified vector notation in any plane in 3D space to account for extensions to a 3D path.

For each $plane_i$, four configurations of the conventional 2D Dubins path can be designed with two points $\mathbf{p}_i, \mathbf{p}_{i+1}$ and two directions $\mathbf{e}_i, \mathbf{e}_{i+1}$. The four configurations are RSR, RSL, LSR, and LSL, where R is a right turn, L is a left turn, and S is a straight line. The shortest path is chosen by comparing the distance along the Dubins path between two points.

We define a left rotation as a rotation about the unit vector $\mathbf{e}_{n,i}$, which is normal to $plane_i$ and points the upward direction in a reference frame, as shown in Figure 2.

$$\mathbf{e}_t = \frac{\mathbf{e}_i \times \mathbf{e}_{i+1}}{\|\mathbf{e}_i \times \mathbf{e}_{i+1}\|_2} \qquad (2)$$
$$\mathbf{e}_{n,i} = \begin{cases} \mathbf{e}_t, & \text{if } \mathbf{e}_t \cdot \mathbf{k} \geq 0 \\ -\mathbf{e}_t, & \text{else} \end{cases}$$

If $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ are parallel, the normal vector to $plane_i$ is set to the previous one as $\mathbf{e}_{n,i} = \mathbf{e}_{n,i-1}$.

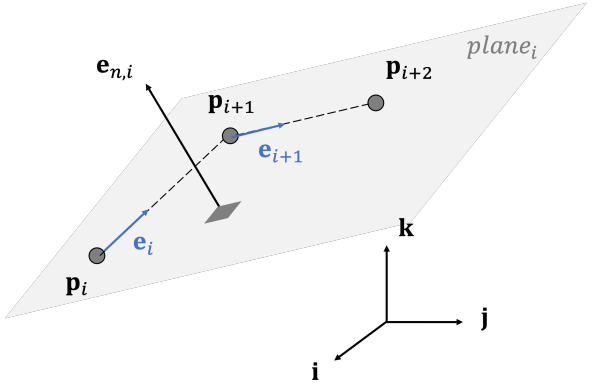

Figure 2. Normal vector to a plane.

For a given direction vector and the minimum radius of rotation R at waypoint $\mathbf{p}_i$, there are two tangent circular arcs: left and right, as shown in Figure 3. A vehicle can either turn left or right, traversing these circular arcs with the circle's rotation direction. The centers of both circular arcs are chosen such that the direction vector at waypoint $\mathbf{p}_i$ is tangential to both circular arcs and computed as follows:

$$\begin{aligned} \mathbf{c}_{l,i} &= \mathbf{p}_i + R\mathbf{e}_{n,i} \times \mathbf{e}_i \\ \mathbf{c}_{r,i} &= \mathbf{p}_i - R\mathbf{e}_{n,i} \times \mathbf{e}_i \end{aligned} \tag{3}$$

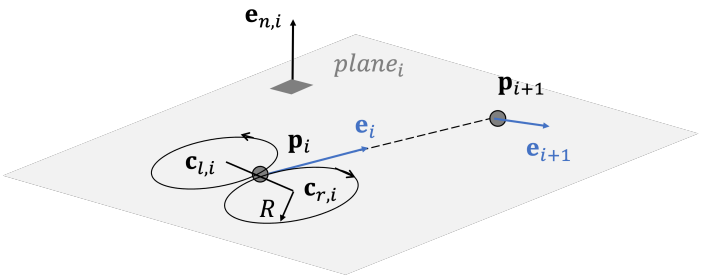

Figure 3. Left and right circle at a waypoint.

2.1.1. RSR and LSL

For the RSR case, a trajectory consists of turning right on a right circular arc at waypoint $\mathbf{p}_i$, going straight, and then turning right again on a right arc until the next waypoint $\mathbf{p}_{i+1}$ is reached. Figure 4 shows the pull-out point $\mathbf{q}_{0,i}$ at waypoint $\mathbf{p}_i$, which is a transition point from a circular arc to a straight line, and the wheel-over point $\mathbf{q}_{1,i}$, which is a transition point from a straight line to a circular arc at waypoint $\mathbf{p}_{i+1}$, and a straight line connecting them, respectively.

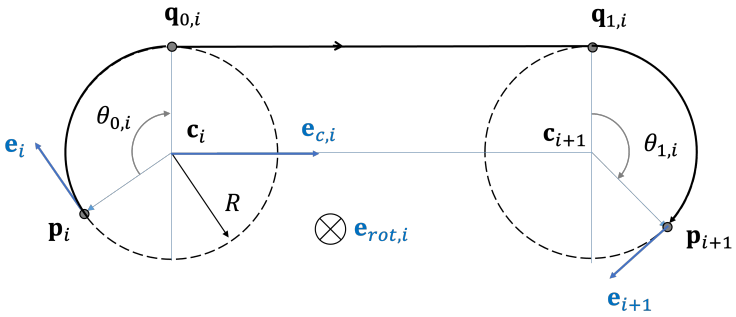

Figure 4. RSR path.

The pull-out and wheel-over points are computed as follows:

$$\begin{aligned} \mathbf{q}_{0,i} &= -R\mathbf{e}_{rot,i} \times \mathbf{e}_{c,i} + \mathbf{c}_i \\ \mathbf{q}_{1,i} &= -R\mathbf{e}_{rot,i} \times \mathbf{e}_{c,i} + \mathbf{c}_{i+1} \end{aligned} \tag{4}$$

where $\mathbf{e}_{rot,i} = -\mathbf{e}_{n,i}$ is the rotation axis of the right circular arc, $\mathbf{c}_i = \mathbf{c}_{r,i}$, $\mathbf{c}_{i+1} = \mathbf{c}_{r,i+1}$, and $\mathbf{e}_{c,i}$ is an unit direction vector from $\mathbf{c}_i$ to $\mathbf{c}_{i+1}$.

The rotation angle $\theta_{0,i}$ to traverse from point $\mathbf{p}_i$ to $\mathbf{q}_{0,i}$ is determined by

$$\begin{aligned} \phi_0 &= \cos^{-1}\left(\frac{(\mathbf{p}_i - \mathbf{c}_i) \cdot (\mathbf{q}_{0,i} - \mathbf{c}_i)}{R^2}\right) \\ \theta_{0,i} &= \begin{cases} \phi_0, & \text{if } \mathbf{e}_i \cdot \mathbf{e}_{d,0} \geq 0 \\ 2\pi - \phi_0, & \text{else} \end{cases} \end{aligned} \tag{5}$$

where

$$\mathbf{e}_{d,0} = \frac{\mathbf{q}_{0,i} - \mathbf{p}_i}{\|\mathbf{q}_{0,i} - \mathbf{p}_i\|_2}.$$

Similarly, the rotation angle $\theta_{1,i}$ to traverse from point $\mathbf{q}_{1,i}$ to $\mathbf{p}_{i+1}$ is given as follows:

$$\begin{aligned} \phi_1 &= \cos^{-1}\left(\frac{(\mathbf{p}_{i+1}-\mathbf{c}_{i+1})\cdot(\mathbf{q}_{1,i}-\mathbf{c}_{i+1})}{R^2}\right) \\ \theta_{1,i} &= \begin{cases} \phi_1, & \text{if } \mathbf{e}_{i+1}\cdot\mathbf{e}_{d,1} \geq 0 \\ 2\pi - \phi_1, & \text{else} \end{cases} \end{aligned} \tag{6}$$

where

$$\mathbf{e}_{d,1} = \frac{\mathbf{p}_{i+1} - \mathbf{q}_{1,i}}{\|\mathbf{p}_{i+1} - \mathbf{q}_{1,i}\|_2}.$$

For the LSL case, a trajectory consists of turning left on a left circular arc, going straight, and then turning left again on a left arc until the next waypoint is reached. Figure 5 shows the pull-out and wheel-over points and a straight line connecting them, respectively.

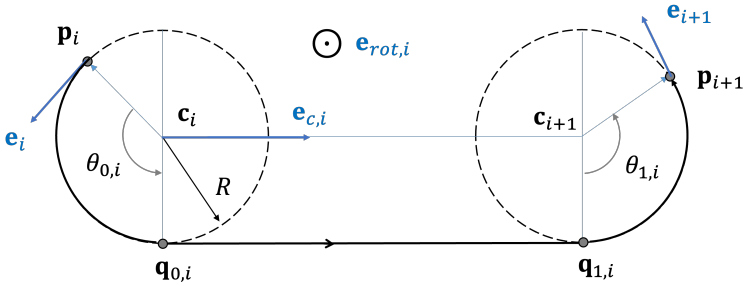

Figure 5. LSL path.

The pull-out, wheel-over points, and rotation angle are computed by the same Equations (4)–(6) with the following replacements:

$$\begin{aligned} \mathbf{e}_{rot,i} &= \mathbf{e}_{n,i} \\ \mathbf{c}_i &= \mathbf{c}_{l,i} \\ \mathbf{c}_{i+1} &= \mathbf{c}_{l,i+1} \end{aligned} \tag{7}$$

The distance s_i between two waypoints along the Dubins path is computed by summing the arc lengths along the circumference and straight line.

$$s_i = R\,\theta_{0,i} + \|\mathbf{q}_{1,i} - \mathbf{q}_{0,i}\|_2 + R\,\theta_{1,i} \tag{8}$$

2.1.2. RSL and LSR

For the RSL case, a trajectory consists of first turning right on a right circular arc at waypoint $\mathbf{p}_i$, going straight, and then turning left on a left arc until the next waypoint $\mathbf{p}_{i+1}$ is reached. Figure 6 shows the pull-out point $\mathbf{q}_{0,i}$, the wheel-over point $\mathbf{q}_{1,i}$, and a straight line connecting them, respectively.

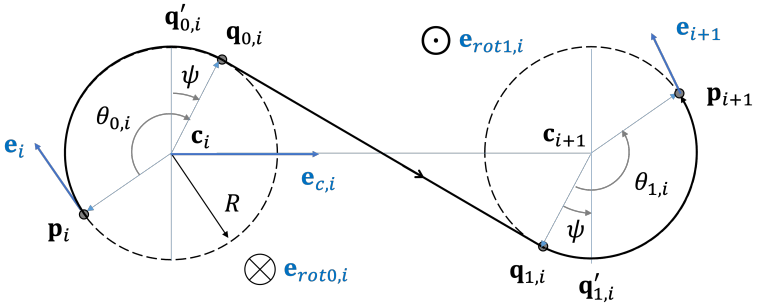

Figure 6. RSL path.

The pull-out and wheel-over points are computed as follows:

$$\begin{aligned}\mathbf{q}_{0,i} &= \mathbf{q}'_{0,i} + (\mathbf{q}'_{0,i} - \mathbf{c}_i)\cos\psi + \mathbf{e}_{rot0,i} \times (\mathbf{q}'_{0,i} - \mathbf{c}_i)\sin\psi \\ \mathbf{q}_{1,i} &= \mathbf{q}'_{1,i} + (\mathbf{q}'_{1,i} - \mathbf{c}_{i+1})\cos\psi - \mathbf{e}_{rot1,i} \times (\mathbf{q}'_{1,i} - \mathbf{c}_{i+1})\sin\psi\end{aligned} \tag{9}$$

where

$$\begin{aligned}\mathbf{q}'_{0,i} &= -R\mathbf{e}_{rot0,i} \times \mathbf{e}_{c,i} + \mathbf{c}_i \\ \mathbf{q}'_{1,i} &= -R\mathbf{e}_{rot1,i} \times \mathbf{e}_{c,i} + \mathbf{c}_{i+1} \\ \psi &= \sin^{-1}\left(\frac{2R}{\|\mathbf{c}_{i+1} - \mathbf{c}_i\|_2}\right)\end{aligned} \tag{10}$$

and $\mathbf{e}_{rot0,i} = -\mathbf{e}_{n,i}$, $\mathbf{e}_{rot1,i} = \mathbf{e}_{n,i}$ are rotation axes, and $\mathbf{c}_i = \mathbf{c}_{r,i}$, $\mathbf{c}_{i+1} = \mathbf{c}_{l,i+1}$ are centers of circular arcs.

The rotation angle $\theta_{0,i}$ about the rotation axis to traverse from point $\mathbf{p}_i$ to $\mathbf{q}_{0,i}$ can be determined by

$$\begin{aligned}\phi_0 &= \cos^{-1}\left(\frac{(\mathbf{p}_i - \mathbf{c}_i)\cdot(\mathbf{q}'_{0,i} - \mathbf{c}_i)}{R^2}\right) \\ \theta_{0,i} &= \begin{cases}\phi_0 + \psi, & \text{if } \mathbf{e}_i \cdot \mathbf{e}_{d,0} \geq 0 \\ 2\pi - \phi_0 + \psi, & \text{else}\end{cases}\end{aligned} \tag{11}$$

where

$$\mathbf{e}_{d,0} = \frac{\mathbf{q}'_{0,i} - \mathbf{p}_i}{\|\mathbf{q}'_{0,i} - \mathbf{p}_i\|_2}.$$

Similarly, the rotation angle $\theta_{1,i}$ to traverse from point $\mathbf{q}_{1,i}$ to $\mathbf{p}_{i+1}$ is given as follows:

$$\begin{aligned}\phi_1 &= \cos^{-1}\left(\frac{(\mathbf{p}_{i+1} - \mathbf{c}_{i+1})\cdot(\mathbf{q}'_{1,i} - \mathbf{c}_{i+1})}{R^2}\right) \\ \theta_{1,i} &= \begin{cases}\phi_1 + \psi, & \text{if } \mathbf{e}_{i+1} \cdot \mathbf{e}_{d,1} \geq 0 \\ 2\pi - \phi_1 + \psi, & \text{else}\end{cases}\end{aligned} \tag{12}$$

where

$$\mathbf{e}_{d,1} = \frac{\mathbf{p}_{i+1} - \mathbf{q}'_{1,i}}{\|\mathbf{p}_{i+1} - \mathbf{q}'_{1,i}\|_2}$$

For the LSR case, a trajectory consists of first turning left on a left circular arc, followed by straight, and turning right on a right arc until the next waypoint is reached. Figure 7 shows the pull-out and wheel-over points and a straight line connecting them, respectively.

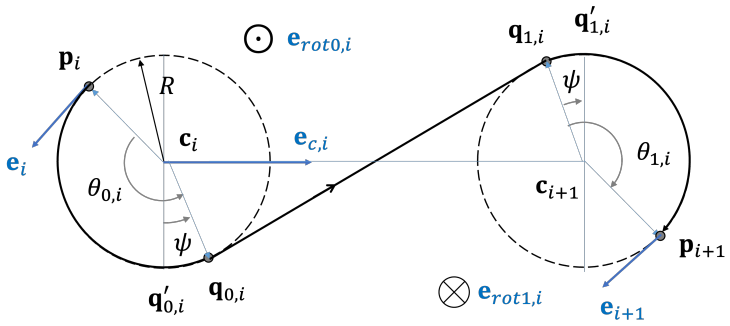

Figure 7. LSR path.

The pull-out, wheel-over points, and rotation angle are computed by the same Equations (9)–(12) with the following replacements:

$$\begin{aligned} \mathbf{e}_{rot0,i} &= \mathbf{e}_{n,i} \\ \mathbf{e}_{rot1,i} &= -\mathbf{e}_{n,i} \\ \mathbf{c}_i &= \mathbf{c}_{l,i} \\ \mathbf{c}_{i+1} &= \mathbf{c}_{r,i+1} \end{aligned} \tag{13}$$

The distance s_i between two waypoints along the Dubins path is computed as follows:

$$s_i = R\,\theta_{0,i} + \|\mathbf{q}_{1,i} - \mathbf{q}_{0,i}\|_2 + R\,\theta_{1,i} \tag{14}$$

2.2. Three-Dimensional Dubins Path

A direction vector at each waypoint is defined as the direction to the next waypoint. With this definition, three sequential waypoint tuples $\{\mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}\}$ and their associated two-direction vectors $\{\mathbf{e}_i, \mathbf{e}_{i+1}\}$ are all on the same plane $plane_i$, and the next consecutive waypoint tuple $\{\mathbf{p}_{i+1}, \mathbf{p}_{i+2}, \mathbf{p}_{i+3}\}$ and their associated directions $\{\mathbf{e}_{i+1}, \mathbf{e}_{i+2}\}$ are also located on another plane $plane_{i+1}$. The $plane_{i+1}$ is tilted from $plane_i$ about the waypoint segment line as amount of $\gamma = \cos^{-1}(\mathbf{e}_{n,i} \cdot \mathbf{e}_{n,i+1})$.

Because the waypoint segment line is the intersection line of both planes, the circular arcs on $plane_i$ and $plane_{i+1}$ have the same tangent at the waypoint $\mathbf{p}_{i+1}$, which is a direction vector along the segment line. Therefore, the Dubins path on both planes is smoothly connected at the waypoint $\mathbf{p}_{i+1}$, as shown in Figure 8. This is how to extend the 2D Dubins path to a 3D path in our method.

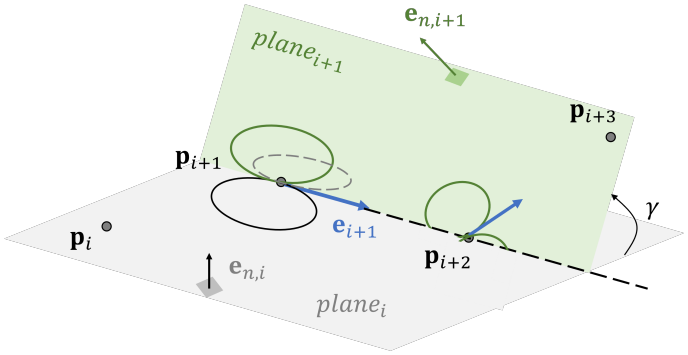

Figure 8. Two planes and intersection line.

The advantage of the proposed method is that the problem of the 3D Dubins path becomes just designing 2D Dubins paths on every plane, and the smooth connection between adjacent planes is automatically achieved. This way, it preserves the same complexity and the shortest path property of the 2D Dubins path.

3. Smoothing Methodology

3.1. Cubic Bezier Curve

A cubic Bezier curve is a continuous curvature curve with the smallest order and is given by

$$\begin{aligned} \mathbf{a}(t) =& (1-t)^3\mathbf{a}_0 + 3t(1-t)^2\mathbf{a}_1 \\ &+ 3t^2(1-t)\mathbf{a}_2 + t^3\mathbf{a}_3, \;\; 0 \le t \le 1 \end{aligned} \tag{15}$$

where $(\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ are control points. This curve passes through points $\mathbf{a}_0$ and $\mathbf{a}_3$, and passes closely through the two middle points $\mathbf{a}_1$ and $\mathbf{a}_2$. Now, suppose that all four control points are on the same plane, and three of them are on the same straight line, i.e.,

$$\begin{aligned} \mathbf{a}_1 &= \mathbf{a}_0 + d_1\mathbf{e}_{in} \\ \mathbf{a}_2 &= \mathbf{a}_1 + d_2\mathbf{e}_{in} \\ \mathbf{a}_3 &= \mathbf{a}_2 + d_3\mathbf{e}_{mid} \end{aligned} \tag{16}$$

where $\mathbf{e}_{in}$ and $\mathbf{e}_{mid}$ are unit direction vectors, and d_1, d_2, and d_3 denote the separation distances between control points. The curvature equation of the curve is given by

$$\kappa(t) = \frac{\|\mathbf{a}'(t) \times \mathbf{a}''(t)\|_2}{\|\mathbf{a}'(t)\|_2^3}. \tag{17}$$

The curvature is computed as $\kappa(0) = 0$ at $\mathbf{a}_0 = \mathbf{a}(0)$ and is monotonically increasing as t goes. For the curvature to be maximized at $\mathbf{a}_3$ with its derivative zero, the separation distance must have the following values, as in [7]:

$$\begin{aligned} &d_2 = 0.346L,\ d_1 = 0.58d_2 \\ &d_3 = 1.31d_2\cos\beta, \quad 0 < \beta < \frac{\pi}{2} \end{aligned} \tag{18}$$

where L is a design parameter that can be expressed as a function of the maximum allowed curvature, and β is the angle between $\mathbf{e}_{in}$ and $\mathbf{e}_{mid}$. The maximum curvature occurs at $\mathbf{a}_3 = \mathbf{a}(1)$ and is computed as

$$\kappa_{max} = \kappa(1) = \frac{2d_2\sin\beta}{3d_3^2} \tag{19}$$

With the Bezier curve described above, two curves can be concatenated to generate a continuous curvature path between two successive line segments in which the curvature is zero at both the start and end points located on each line segment. For two Bezier curves to be connected smoothly and to make maximum curvature occur at the connecting point, the last and first controls of both curves must coincide, and their derivatives of the curvature must be zero.

The first Bezier curve and its control points are defined by (15) and (16), and the second curve is defined symmetrically, as shown in Figure 9.

$$\begin{aligned} \mathbf{b}(t) =&(1-t)^3\mathbf{b}_3 + 3t(1-t)^2\mathbf{b}_2 \\ &+ 3t^2(1-t)\mathbf{b}_1 + t^3\mathbf{b}_0, \ \ 0 \le t \le 1 \end{aligned} \tag{20}$$

where $(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ are control points and defined by the same way to their corresponding counter parts $(\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$.

$$\begin{aligned} \mathbf{b}_1 &= \mathbf{b}_0 - d_1\mathbf{e}_{out} \\ \mathbf{b}_2 &= \mathbf{b}_1 - d_2\mathbf{e}_{out} \\ \mathbf{b}_3 &= \mathbf{b}_2 - d_3\mathbf{e}_{mid} \end{aligned} \tag{21}$$

In Figure 9, $\mathbf{e}_{in}$ and $\mathbf{e}_{out}$ are unit vectors representing the direction along each line segment, $\mathbf{e}_{mid}$ is a unit vector along the line connecting $\mathbf{a}_2$ and $\mathbf{b}_2$, $\mathbf{m}$ is the intersection point of two line segments, and θ is the turning angle between $\mathbf{e}_{in}$ and $\mathbf{e}_{out}$, which is $\theta = 2\beta$ by the geometric relationship. It is easily shown that L is a distance from intersection point $\mathbf{m}$ to $\mathbf{b}_0$ or $\mathbf{c}_0$. Points $\mathbf{a}_3$ and $\mathbf{b}_3$ are located in the center of the line connecting $\mathbf{a}_2$ and $\mathbf{b}_2$. The control points $\mathbf{a}_3$ and $\mathbf{b}_3$ are connecting points and must be equal. The maximum curvature occurs at $\mathbf{a}_3$ and $\mathbf{b}_3$, and its value is given by (19). The curvatures at $\mathbf{a}_0$ and $\mathbf{b}_0$ are, respectively, zero.

In summary, two concatenated Bezier curves produce a circular arc-like curve with maximum curvature at the middle point and zero curvature at the start and end points.

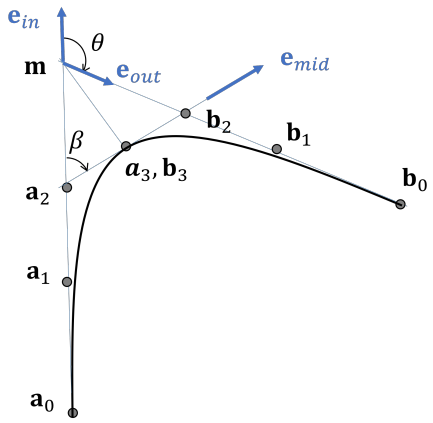

Figure 9. Two Bezier curves.

3.2. Dubins Path Approximation

Our goal is to construct a 3D path with continuous curvature for given multiple waypoints that all must be passed through. Although the proposed 3D Dubins path algorithm produces the shortest path in 3D space, its curvature is discontinuous at the transition point between a straight line and a circular arc. Since the cubic Bezier curve is a G2-continuous curvature curve, it can be used as a transition curve between a straight line and a circular arc of Dubins path to make a smooth connection.

In this paper, we propose a 3D Dubins-path-guided continuous curvature path-smoothing algorithm using the cubic Bezier curve. The proposed algorithm generates a parametric path that simultaneously satisfies curvature continuity and maximum curvature requirements.

Once the 3D Dubins path is designed, the pull-out and wheel-over points and their direction vectors are determined, which are the transition points where a straight line and a circular arc meet. The angle between two-direction vectors at the waypoint and its corresponding pull-out or wheel-over point, i.e., the turning angle θ, can also be computed.

In the case of a circular arc departure, if a waypoint $\mathbf{p}_i$ and a pull-out point $\mathbf{q}_{0,i}$ are assigned as a control point $\mathbf{a}_0$ and $\mathbf{b}_0$ in (15) and (20), respectively, the circular arc can be approximated as a Bezier curve. Similarly, if a wheel-over point $\mathbf{q}_{1,i}$ and a waypoint $\mathbf{p}_{i+1}$ are assigned as a control point $\mathbf{a}_0$ and $\mathbf{b}_0$, respectively, the circular arc can also be approximated as a Bezier curve in a case of arrival, as shown in Figure 10. Since the Bezier curve passes through points $\mathbf{a}_0$ and $\mathbf{b}_0$, all waypoints are guaranteed to pass through.

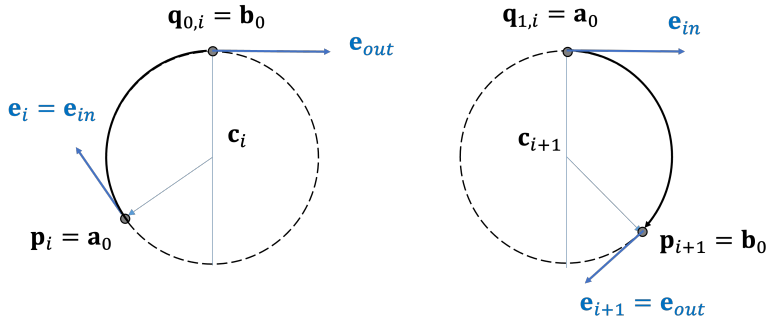

Figure 10. Circular arc approximation.

Different from the conventional Bezier curve design, a design parameter L in (18), which is selected considering maximum allowable curvature, is now computed as a function of turning angle in our method.

$$L = \frac{\|\mathbf{p}_i - \mathbf{q}_i\|_2}{2\cos\left(\frac{\theta}{2}\right)} \tag{22}$$

where $\mathbf{p}_i$ is a waypoint and $\mathbf{q}_i$ is a corresponding pull-out or wheel-over point. With the computed L, the maximum curvature κ_{max} is also determined from (19) and (22) as

$$\kappa_{max} = \frac{1.1228}{R\cos\left(\frac{\theta}{2}\right)} \tag{23}$$

The problem is that the calculated maximum curvature is always greater than the maximum permissible curvature constraint, and this tendency worsens as the turning angle provided by the Dubins path increases. Moreover, a Bezier curve is possible only when the turning angle is less than 180 degrees.

Because the maximum curvature is calculated only with a turning angle, the turning angle must be reduced to much less than 180 degrees to reduce the maximum curvature and overcome the limitation of the Bezier curve. In this paper, we propose to divide the circular arc into several parts and approximate each piece with a Bezier curve until the approximation curve does not deviate too far from the circular arc and the maximum permissible curvature constraint.

The maximum curvature and the largest deviation occur at the midpoint of the split circular arc and are, respectively, given as follows:

$$\begin{aligned} \kappa_{split} &= \frac{1.1228}{R\cos\left(\frac{\theta_s}{2}\right)} \\ \frac{\delta_{max}}{R} &= \frac{1}{\cos\left(\frac{\theta_s}{2}\right)} - 1 - 0.4533\tan\left(\frac{\theta_s}{2}\right)\sin\left(\frac{\theta_s}{2}\right) \end{aligned} \tag{24}$$

where θ_s is the split turning angle. Figure 11 represents the largest deviation δ_{max} and split turning angle θ_s.

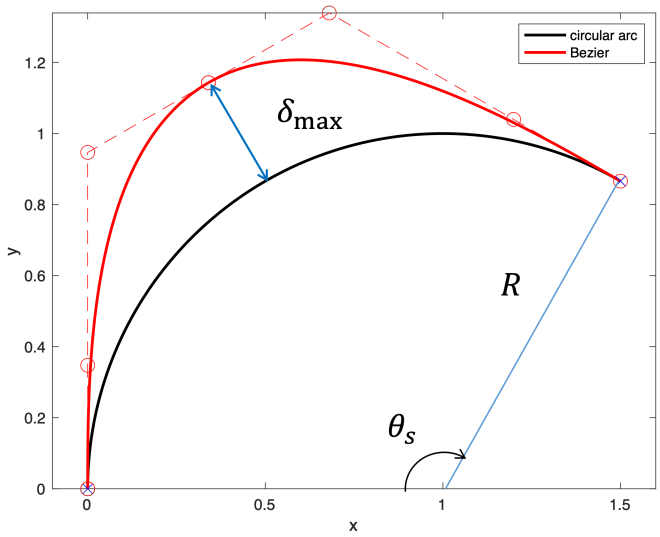

Figure 11. Circular arc and Bezier curve approximation.

The reasonable value for θ_s can be chosen based on (24). For example, κ_{split} and δ_{max} at $\theta_s = 30^0$ are computed as follows:

$$\begin{aligned} \kappa_{split} R &= 1.1624 \\ \frac{\delta_{max}}{R} &= 0.0038 \end{aligned} \tag{25}$$

Once the split turning angle is selected, the circular arc is split evenly into n segments, as shown in Figure 12, where $\mathbf{s}_{j-1}$ and $\mathbf{s}_j$ are consecutive start and end points of the

segmented arc. In the case of a circular arc departure, $\mathbf{s}_0 = \mathbf{p}_i$ and $\mathbf{s}_n = \mathbf{q}_{0,i}$, while in the case of arrival, $\mathbf{s}_0 = \mathbf{q}_{1,i}$ and $\mathbf{s}_n = \mathbf{p}_{i+1}$.

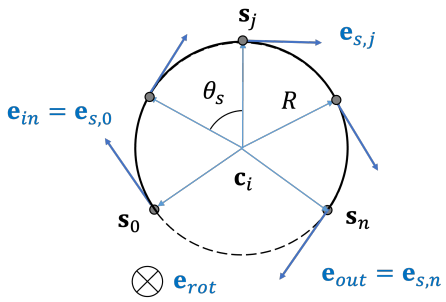

Figure 12. Split circular arc.

The position $\mathbf{s}_j$ and direction vector $\mathbf{e}_{s,j}$ of a specific point on the circular arc in Figure 12 can be easily calculated by the following equations:

$$\begin{aligned} \mathbf{e}_{s,j} &= \mathbf{e}_{rot} \times \mathbf{e}_{s,j-1} \sin\theta_s + \mathbf{e}_{s,j-1} \cos\theta_s \\ \mathbf{s}_j &= \mathbf{e}_{rot} \times (\mathbf{s}_{j-1} - \mathbf{c}_i) \sin\theta_s \\ &\quad + (\mathbf{s}_{j-1} - \mathbf{c}_i) \cos\theta_s + \mathbf{c}_i, \quad j = 1, \ldots, n \end{aligned} \tag{26}$$

where $\mathbf{e}_{s,0} = \mathbf{e}_{in}$ and $\mathbf{e}_{s,n} = \mathbf{e}_{out}$.

Figure 13 shows an example of a circular arc with a turning angle of 150 degrees. Here, the left figure shows an approximation with one Bezier curve and the right figure one with five Bezier curves. Comparing the figure on the right with the figure on the left, it can be seen that the Bezier curve hardly differs from the circular arc, and the curvature is significantly reduced.

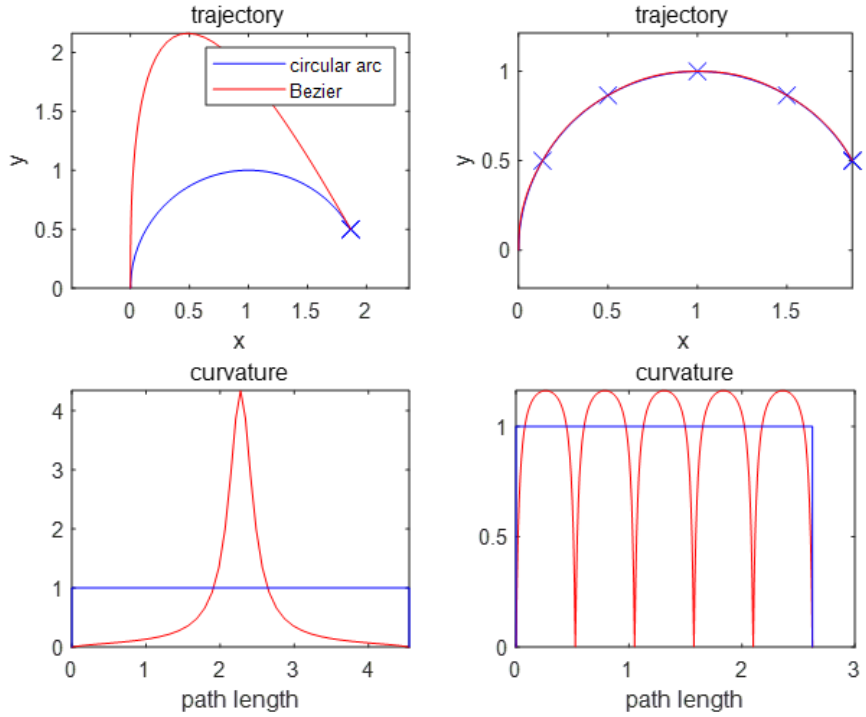

Figure 13. Bezier curve approximation for the split circular arc.

To strictly meet the maximum curvature requirement, the radius of the circular arc should be slightly increased when designing the Dubins reference path for the Bezier curve approximation.

4. Simulations

The numerical simulations are conducted to evaluate the proposed path-smoothing algorithm. In the simulations, two sequences of waypoints are considered, as shown in Tables 1 and 2. The turning radius of the vehicle is set to R = 30 m, which corresponds to the maximum curvature constraint of $\kappa_{max} = 1/30$. In the first waypoint sequence, the waypoints are far away relative to the turning radius, but the second is not. The approaching direction to the final waypoint of both sequences is assumed to be the $-y$ direction. Considering (24) or (25), when designing a Dubins reference path for Bezier curve approximation, the split turning angle is chosen as $\theta_s = 30^0$ and the base radius of the angular arc as R_{base} = 34.872 m to meet the maximum curvature constraint.

Table 1. Waypoints I.

Coordinate	Waypoints					
	#1	#2	#3	#4	#5	#6
$x(m)$	200	200	0	0	200	0
$y(m)$	0	200	200	400	400	0
$z(m)$	100	100	100	80	60	0

Table 2. Waypoints II.

Coordinate	Waypoints					
	#1	#2	#3	#4	#5	#6
$x(m)$	20	20	0	0	20	0
$y(m)$	0	20	20	40	40	0
$z(m)$	100	100	100	80	60	0

The Dubins path and Bezier curve approximation path are generated from a series of waypoints in Table 1, as shown in Figure 14, and the horizontal and vertical plan views are shown in Figure 15. From the figures, it can be seen that both paths pass through all waypoints marked with 'o'.

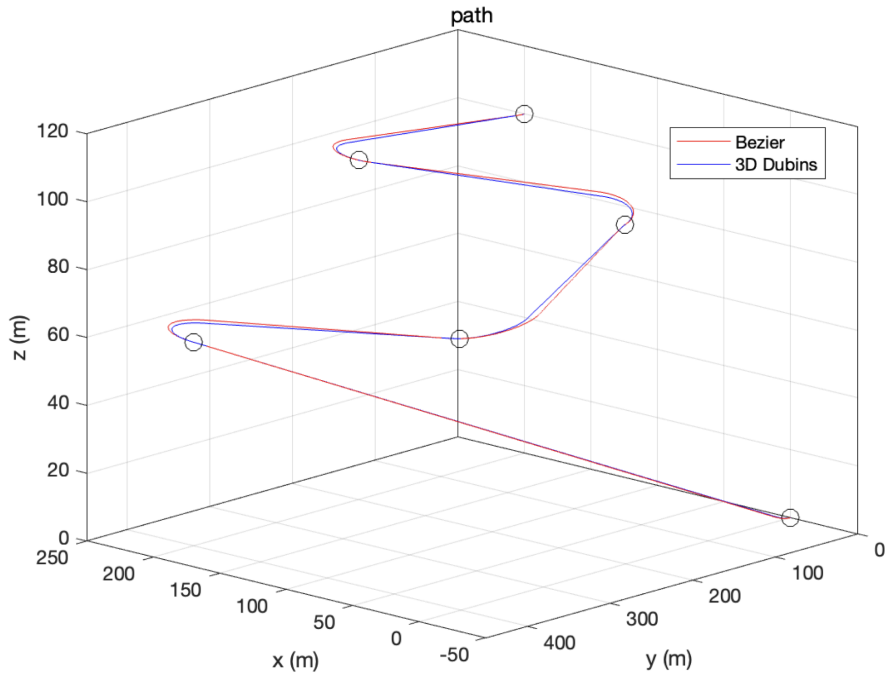

Figure 14. Three-dimensional Dubins path and Bezier approximation.

Note that the Dubins path and Dubins reference path are different. The Dubins reference path is for designing Bezier curve approximation, and it is created by adjusting the circular arc radius as R_{base}, whereas the Dubins path is designed with radius R. Therefore, the trajectory difference seen in the figures should not be interpreted as trajectory estimation error.

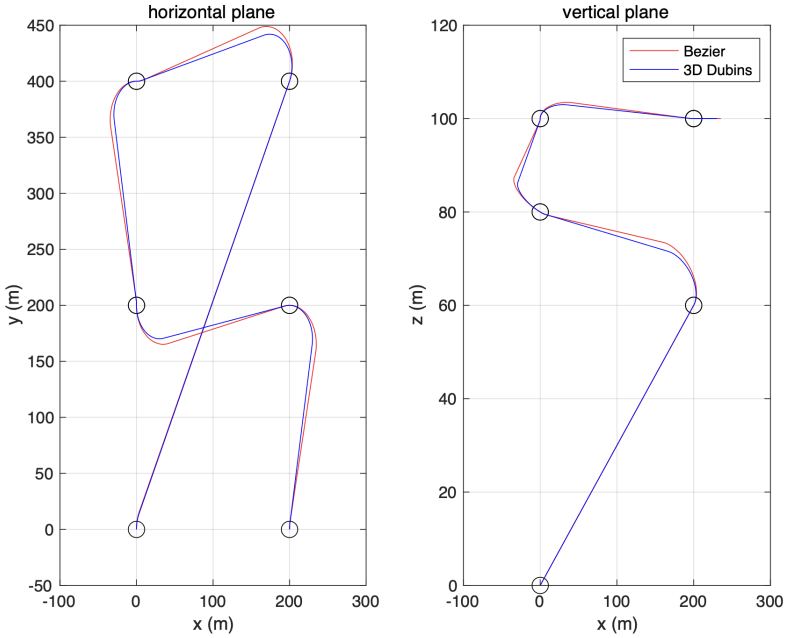

Figure 15. Horizontal and vertical plan views.

The Dubins path consists of the following sequence: RSL, LSR, LSR, LSR, and RSL, and has a length of 1351.5 m. The length of the Bezier curve approximation is 1371.0 m, which is a 19.5 m increase. However, the curvature of the Dubins path is discontinuous at the transition points, since the path directly transitions from line segments to circular arcs. In contrast, the Bezier curve approximation is a continuous curvature path satisfying maximum curvature constraints, as shown in Figure 16. The horizontal dashed line in the figure indicates the maximum curvature limit. The Bezier curve approximation is, therefore, a feasible path.

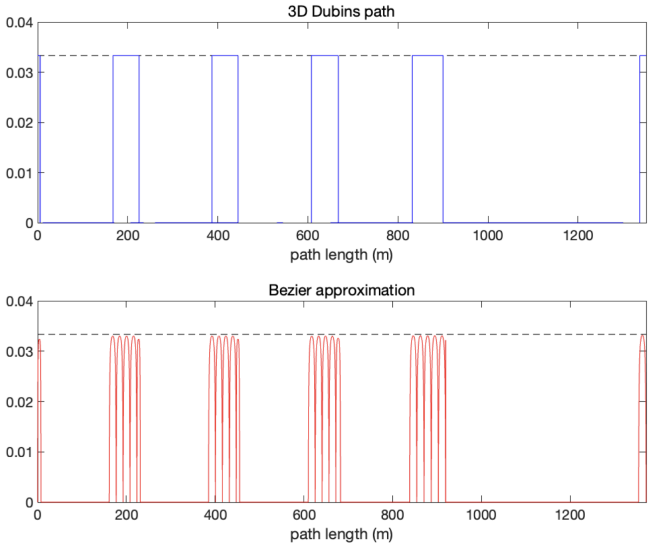

Figure 16. Curvature.

Figure 17 is a partially enlarged view of Figure 16. It shows that the circular arc segment of the Dubins path is approximated by several Bezier curves to meet the maximum curvature requirement.

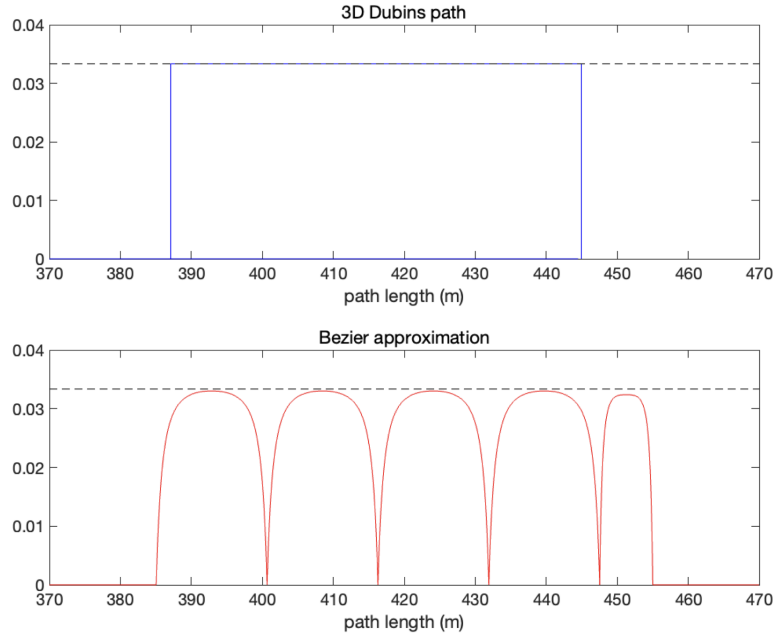

Figure 17. Partially enlarged view of Figure 16.

For the sequence of waypoints in Table 2, the Dubins path and its Bezier curve approximation are shown in Figures 18 and 19. The Dubins path and its approximated Bezier paths look complicated because the waypoints are relatively close to the turning radius. However, both paths pass through all waypoints. The Dubins path consists of the following sequence: LSL, RSR, RSL, RSL, and RSL, and has a length of 1042.6 m. The length of the Bezier curve approximation is 1196.8 m, which is a 154.2 m increase. The length of the Bezier curve approximation is a bit longer due to the large portion of the circular arc segment of the Dubins path.

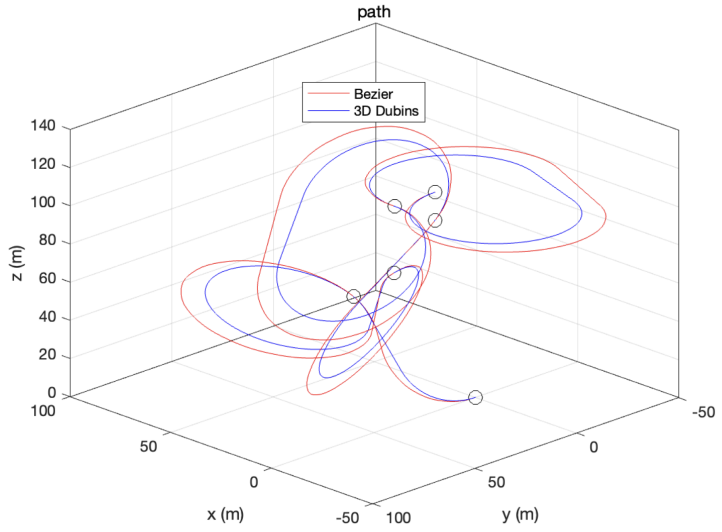

Figure 18. Three-dimensional Dubins path and Bezier approximation.

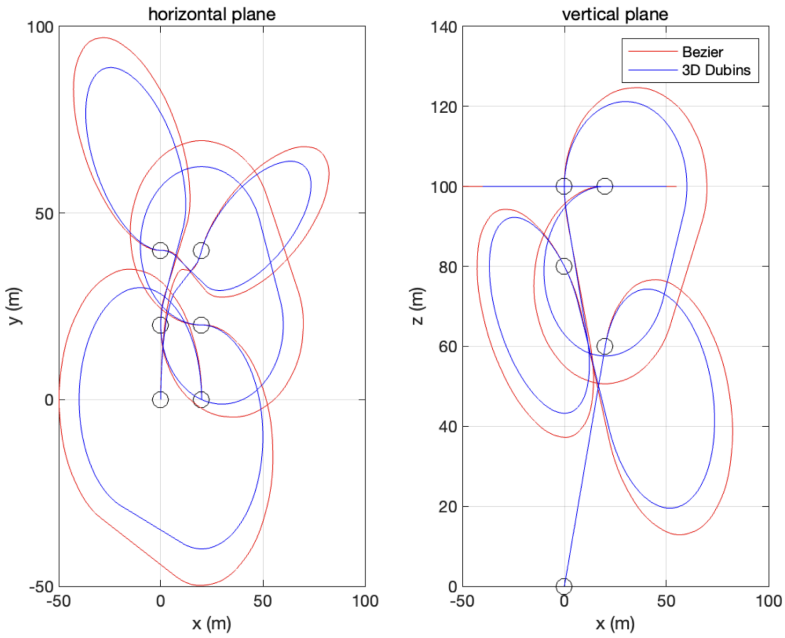

Figure 19. Horizontal and vertical plan views.

Figure 20 is the curvature plot of both paths, showing that only the Bezier curve approximation has continuous curvature and satisfies the maximum curvature requirement.

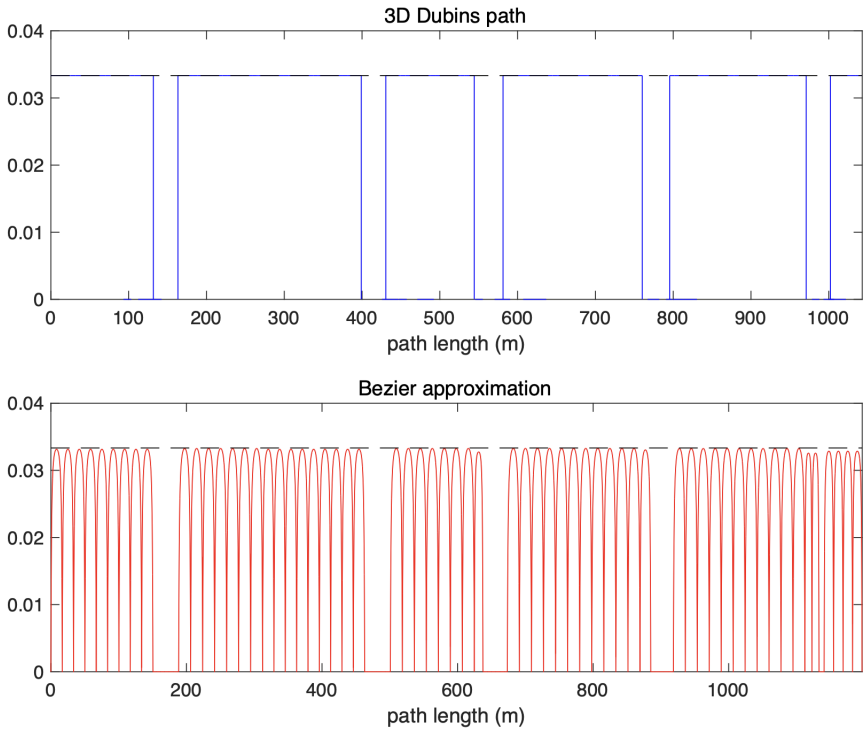

Figure 20. Curvature.

5. Conclusions

In this paper, we present a novel smoothing algorithm generating a three-dimensional continuous curvature path from a sequence of waypoints. Our algorithm is composed of two stages: one is a reference path generation by the 3D Dubins path, and the other is a reference path approximation by several cubic Bezier curves.

For the 3D Dubins reference path, we propose a novel method to extend a 2D Dubins path to a 3D path by applying a simple constraint that the pose is a unit vector pointing to the next waypoint to the pose of each waypoint. Our method is straightforward for use in waypoint guidance problems. Its advantages include preserving the same methodology and complexity of the 2D Dubins path design.

Since the curvature of the resulting 3D Dubins reference path is discontinuous at the transition point between a straight line and a circular arc, we propose to divide the circular arc into several parts and approximate each piece using a Bezier curve to avoid curvature discontinuity. This method also overcomes the limitation on the size of the turning angle of the Bezier curve and allows all waypoints to pass through. We also provide a criterion to select a required number of Bezier curves to strictly meet the maximum curvature constraint.

The numerical simulations are conducted to evaluate the proposed path-smoothing algorithm. In the simulations, two different sequences of waypoints are considered. The results show that the proposed algorithm produces a continuous curvature path that passes through all waypoints with a slight increase in path length compared with the Dubins reference path.

Our path-smoothing algorithm can be used for path following applications in many areas, such as aerospace and robotics, as it generates a 3D parametric path that simultaneously satisfies curvature continuity and maximum curvature requirements. However, since the proposed algorithm is tested only in simulation and not in a real environment, future work will be to perform experiments in a real environment.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2021R1F1A1063780).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Elbanhawi, M.; Simic, M. Sampling-Based Robot Motion Planning: A Review. *IEEE Access* **2014**, *2*, 56–77. [CrossRef]
2. Amato, N.; Bayazit, O.; Dale, L.; Jones C.; Vallejo, D. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Autom.* **2000**, *16*, 442–447. [CrossRef]
3. Ahmad, Z.; Ullah, F.; Tran, G.; Lee, S. Efficient Energy Flight Path Planning Algorithm Using 3-D Visibility Roadmap for Small Unmanned Aerial Vehicle. *Int. J. Aerosp. Eng.* **2017**, *2017*, *2849745*. [CrossRef]
4. Lee, D.; Shim, D. Spline-RRT* Based Optimal Path Planning of Terrain Following Flights for Fixed-Wing UAVs. In Proceedings of the International Conference on URAI, Kuala Lumpur, Malaysia, 12–15 November 2014.
5. Tsai, Y.; Lee, C.; Lin, C.; Huang, C. Development of Flight Path Planning for Multirotor Aerial Vehicles. *Aerospace* **2015**, *2*, 171–188. [CrossRef]
6. Yang, Y.; Sukkarieh, S. 3D Smooth Path Planning for a UAV in Cluttered NaturalEnvironments. In Proceedings of the International Conference on Intelligent Robotics and Systems, Nice, France, 22–26 September 2008.
7. Yang K.; Sukkarieh, S. An Analytical Continuous-Curvature Path-Smoothing Algorithm. *IEEE Trans. Robot.* **2010**, *26*, 561–568. [CrossRef]
8. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Hoshino, Y.; Peng, C. Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges. *Sensors* **2018**, *18*, 3170. [CrossRef] [PubMed]
9. Cimurs R.; Suh, I. Time-optimized 3D Path Smoothing with Kinematic Constraints. *Int. J. Control Autom. Syst.* **2020**, *18*, 1277–1287. [CrossRef]
10. Ahmed, A.; Soliman, A.; Maged, A.; Gaafar, M.; Magdy, M. Path Smoothing Algorithm Using Thin-Plate Spline. In Proceedings of the 7th International Conference on Control, Automation and Robotics, Singapore, 3–26 April 2021.
11. Dubins, L. 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. [CrossRef]
12. Cardenas, I.; Flores, G.; Salazar, S.; Lo, R. Dubins Path Generation for a Fixed Wing UAV. In Proceedings of the International Conference on Unmanned Aircraft Systems, Orlando, FL, USA, 27–30 May 2014.

13. Tsourdos, A.; White, B.; Shanmugavel, M. Path Planning in Three Dimensions. In *Cooperative Path Planning of Unmanned Aerial Vehicles*; Wiley: Chichester, West Sussex, UK, 2011; pp. 65–78.
14. Vana, P.; Neto, A.; Faigl, J.; Macharet, D. Minimal 3D Dubins Path with Bounded Curvature and Pitch Angle. In Proceedings of the IEEE International Conference on Robotics and Automation, Paris, France, 31 May 2020.
15. Rognli, V. Path Generation and Spline Approximation of a 3D Extended Dubins Path. Master's Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2021.
16. Cai W.; Zhang, M. Smooth 3D Dubins Curves Based Mobile Data Gathering in Sparse Underwater Sensor Networks. *Sensors* **2018**, *18*, 2105. [CrossRef] [PubMed]
17. Choi, J.; Curry, R.; Elkaim, G. Continuous Curvature Path Generation Based on Bezier Curves for Autonomous Vehicles. *Int. J. Appl. Math.* **2010**, *40*, 91–101.
18. Lekkas, A.; Dahl, A.; Breivik, M.; Fossen, T. Continuous-Curvature Path Generation Using Fermat's Spiral. *Model. Identif. Control* **2013**, *34*, 183–198. [CrossRef]
19. Li, F-F.; Du, Y.; Jia, K-J. Path planning and smoothing of mobile robot based on improved artifcial fish swarm algorithm. *Sci. Rep.* **2022**, *12*, 659. [CrossRef]
20. Shkel, A.; Lumelsky, V. Classification of the Dubins set. *Robot. Auton. Syst.* **2001**, *34*, 179–202. [CrossRef]