Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Vebjørn Johansen Rognli

Path Generation and Spline Approximation of a 3D Extended Dubins Path

Master's thesis in Cybernetics and Robotics
Supervisor: Thor Inge Fossen and Erlend Magnus Lervik Coates
June 2021

NTNU
Norwegian University of Science and Technology

Vebjørn Johansen Rognli

Path Generation and Spline Approximation of a 3D Extended Dubins Path

Master's thesis in Cybernetics and Robotics
Supervisor: Thor Inge Fossen and Erlend Magnus Lervik Coates
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

A key ability of autonomous flight for fixed-wing unmanned aerial vehicles (UAVs) is the ability to plan flights automatically. This thesis treats the problem of generating a three-dimensional path, from a sequence of waypoints, that is feasible with respect to the dynamic constraints of fixed-wing UAVs. A feasible path for a fixed-wing UAV is a path with continuous feedforward signals for the roll angle, the pitch angle, and the heading angle. The two-dimensional path generation methods in this thesis are inspired by the classical Dubins Path and the curvature continuous Extended Dubins Path, but the methods are generalized to a sequence of waypoints. The methods are based on existing work but modified for efficient computation, such as avoiding numerical integration, and improved in certain edge cases. The three-dimensional path generation is decoupled into horizontal path generation and vertical path generation. The horizontal path and vertical path are generated with two different methods since different levels of continuity are required in the horizontal plane and the vertical plane for a feasible path. The resulting three-dimensional path is approximated by cubic algebraic splines for a unified path representation which simplifies guidance and improves computational efficiency since numerical integration during guidance is avoided.

Sammendrag

En viktig egenskap som muliggjør autonom flygning av ubemannede fartøy (UAV) er automatisk baneplanlegging. Dette arbeidet tar for seg problemet med å generere en tredimensjonal bane, fra en sekvens av veipunkter, som tar hensyn til de dynamiske begrensningene som gjelder for små ubemannede fly. En hensiktsmessig bane for et lite ubemannet fly er en bane med kontinuerlige foroverkoblingssignaler for rullvinkelen, stampvinkelen og girvinkelen. Metodene for todimensjonal banegenerering er basert på den klassiske Dubins banen og den kurvatur kontinuerlige Extended Dubins banen, men metodene er generaliserte til en sekvens av veipunkter. Metodene er basert på eksisterende arbeid, men endret for å være mer beregningseffektive, ved blant annet å unngå numerisk integrasjon, og forbedret i visse spesielle tilfeller som ikke var tatt hensyn til originalt. Den tredimensjonale banegenereringen dekobles i en horisontal banegenerering og en vertikal banegenerering. Den horisontale banen og den vertikale banen generes med to forskjellige metoder siden graden av kontinuitet som er nødvendig i horisontalplanet og i vertikalplanet er forskjellig for å få en hensiktsmessig bane. Den resulterende tredimensjonale banen approksimeres med kubisk algebraiske spliner for å få en uniform parameterfremstilling av banen, som forenkler gaiding, og for å forbedre beregningseffektiviteten siden numerisk løsning av integraler under gaiding unngås.

Contents

Figures

Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background and Motivation

An unmanned aerial vehicle (UAV), often referred to as a drone, is a small aircraft without an onboard pilot. The use of UAVs is rapidly growing, especially in civil application domains [1]. The two main types of UAVs are rotary-wing and fixed-wing, where rotary-wings are most common for missions over a smaller area or with a shorter time frame, and fixed-wings for larger areas or longer time frames. The main advantages of rotary-wings are the ability for vertical take-off and the ease of hovering in place. However, the ease of hovering comes at a cost, rotary-wing UAVs use a lot of energy just to stay afloat. Fixed-wing UAVs resemble more conventional aircraft, though in a much smaller scale. The passive lift from the wings makes fixed-wings energy efficient in cruise mode. The main use cases of fixed-wing UAVs are surveillance and mapping of large areas, where energy efficiency is crucial. A hybrid vertical take-off and landing (VTOL) fixed-wing drone is a hybrid of a fixed-wing and a rotary-wing configuration.

The modeling of the dynamics of fixed-wing UAVs, from a control perspective, is treated in detail in Beard and McLain [2]. As fixed-wing UAVs resemble conventional aircraft, models of conventional aircraft from Stevens *et al.* [3] are also relevant. Fossen [4] is an extensive source for motion control of marine crafts, where the guidance chapters are relevant for aircraft due to the similar underactuated nature.

A pilot may control the UAV with a remote control system or the UAV system may be fully autonomous, or anything in between. Five levels of drone autonomy are described in Radovic [5], where the pilot is out of the loop in levels four and five. The key property of the highest autonomy levels is the ability to plan and follow flights automatically. A common model of the motion control system of autonomous or semi-autonomous systems is the guidance, navigation, and control (GNC) model [4]. A modified figure of the GNC model from Fossen [4] is shown in Figure 1.1. The guidance part of the system is responsible for guiding the system towards the desired point or along the desired

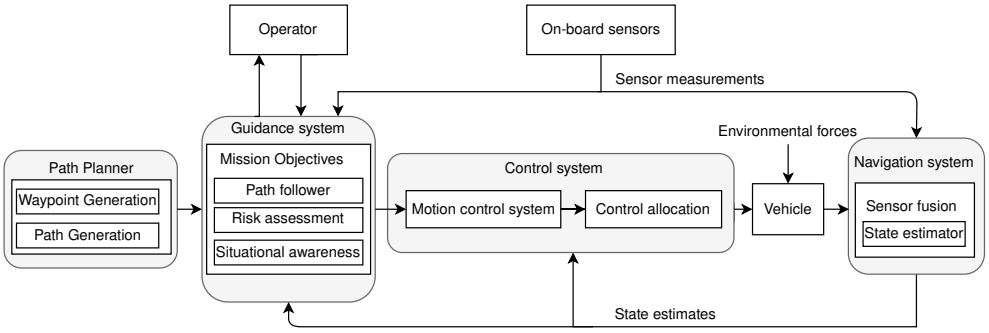


Figure 1.1: Guidance, Navigation and Control

path.

Guidance can further be divided into two systems: path planning and path following. Path following is following a time-invariant path, while trajectory tracking is following a time-varying reference trajectory [4]. Path planning constructs a desired path for the system and path following ensures that the desired path is tracked. LaValle [6] is one of the most comprehensive resources on general path planing, while Tsourdos *et al.* [7] focuses on (cooperative) path planning for (multiple) UAVs. The desired path should satisfy constraints due to the surrounding environment and constraints due to the vehicle abilities. Examples of environmental constraints are obstacles and wind. Due to the size and operating speed of fixed-wing UAVs, the wind is a significant factor and one of the primary challenges for automatic flight [2]. The vehicle constraints depend on the dynamic abilities of the vehicle. As an example, fixed-wing UAVs usually use a banking turn to change direction. A larger roll/bank angle corresponds to a smaller turning radius, but the roll/bank angle is limited to a maximum value to keep the aircraft flying. This limits the minimum turning radius of fixed-wing UAVs. The desired path must therefore be planned such that no turn has a turning radius smaller than the minimum turning radius. If the path is generated without consideration of this constraint, then the aircraft will never be able to follow the path exactly. Other constraints for fixed-wing UAVs are the maximum climbing rate and the maximum rolling motion rate.

Dividing path planning into two separate steps ensures that the environmental and vehicle constraints can be treated separately [8]. The two steps of path planning are

- the determination of waypoints,
- and the generation of a feasible path between the waypoints.

With such a split, the determination of waypoints is done with respect to wind and obstacles and the path generation with respect to the dynamic capabilities of the aircraft. This thesis is concerned with interpolating paths, which are paths that pass exactly through the waypoints.

The flight control system of a fixed-wing UAV is usually implemented as

an embedded system on a microcontroller. Due to resource constraints such as power consumption and sizing, the computational power of the system is often limited. Path planning is a real-time problem that must be able to produce a path within a deadline. The generation of a path is sometimes part of an iterative process of determining the waypoints to ensure that the generated path is feasible with respect to the environmental constraints. For that to be possible, the path generation must be computationally efficient so that a sufficient number of iterations can run before the deadline. The computational efficiency of the path generation is especially important for small systems with limited computational power.

From a path following perspective, the path generation should produce a path with continuous feedforward signals for all relevant states. For fixed-wing UAVs, the relevant states are the roll angle, pitch angle, and heading angle with the associated roll, pitch, and heading rates. Global path following stability can be achieved with only feedback-based control, but with reduced accuracy compared to path following algorithms that leverage feedforward signals, especially for paths with high curvature.

This thesis concerns feasible path generation with respect to the dynamic constraints of fixed-wing UAVs, with a focus on efficient computation, such as avoiding numerical integration and optimization methods.

1.2 Contributions

The contributions of this thesis are listed below.

- Improvements and fixes to the 2D Dubins Path and the 2D Extended Dubins Path from Dahl [9] to handle edge cases.
- A modification of the 2D Extended Dubins Path that avoids online numerical integration and instead exploits a single set of Fresnel integrals which are pre-computed offline.
- A complete and thorough description of a spline approximated three-dimensional path: from the curve parameterizations, via the path generation methods, to the spline approximation.
- A MATLAB implementation of the complete system.

1.3 Thesis Outline

This thesis is organized as follows: Chapter 2 presents the necessary building blocks for path generation such as subpaths, continuity measures, and constraints for fixed-wing UAVs. Chapter 3 presents two methods for two-dimensional path generation in detail, with improvements and fixes for edge cases. Chapter 4 combines the two two-dimensional paths into a three-dimensional path and shortly discusses the methods it is inspired by. Chapter 5 presents a spline approximation of the resulting three-dimensional path. Chapter 6 presents the

paths generated from an example sequence of waypoints and discusses the observed results. Chapter 7 summarizes the results and suggests future work.

Chapter 2

Fundamentals

This chapter introduces the central framework for path generation for fixed-wing UAVs and some requirements for the path to be feasible.

2.1 Coordinate Systems

A common three-dimensional coordinate system for local navigation is the right-handed North-East-Down ($x-y-z$) coordinate system. The x-axis points towards the true North, the y-axis points towards the East, and the z-axis points downwards normal to the Earth's surface. In the aircraft community, the downwards z-axis is sometimes substituted with an upwards h-axis for altitude.

The three-dimensional North-East-Altitude coordinate system can be decoupled into two two-dimensional coordinate systems: a horizontal coordinate system and a vertical coordinate system. The horizontal coordinate system is the North-East ($x-y$) coordinate system and the vertical coordinate system is the longitudinal Altitude-HorizontalDistance ($h-s_h$) coordinate system. The h-axis in the vertical coordinate system represents the altitude and the s_h-axis represents the horizontal distance. The work in this thesis, such as curve parameterizations, two-dimensional path generation, and spline approximations, are derived and developed in the horizontal coordinate system but are also reused in the vertical coordinate system.

To avoid confusion, a short description of the conventions in this thesis follows. The North coordinate is the first coordinate of points $\mathbf{p}$ and vectors $\mathbf{v}$ in the horizontal coordinate system, and the East coordinate is the second coordinate. Angles are measured from the x-axis positively in a clockwise direction and limited to the interval $[-\pi, \pi)$. Curvature is treated as a signed value where a positive value indicates a curve tending in a clockwise direction and a negative value indicates a counter-clockwise direction. This is opposite of the convention in Pressley [10] and other mathematical sources but is chosen to be consistent with the positive rotation in a clockwise direction as defined in Breivik and Fossen [11]. In the vertical plane, the altitude is the first coordinate and the

horizontal distance is the second coordinate. To allow for reuse of the curve parameterizations and path generation methods, angles and curvature in the vertical coordinate system follows the convention of the horizontal coordinate system.

2.2 Kinematics of Fixed-wing UAVs

The motion of a rigid-body can be described geometrically by a kinematic model [12]. The kinematic model is a set of differential equations that relate the time derivative of the configuration of the vehicle with its linear and angular velocities. The configuration of a fixed-wing UAV can be described by the position and orientation in the North-East-Down coordinate system,

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.1a}$$

$$\Theta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \tag{2.1b}$$

where the individual angles are the roll angle, the pitch angle and the heading/yaw angle, respectively [2]. The linear and angular velocities are given in a body-fixed coordinate system,

$$\mathbf{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{2.2a}$$

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.2b}$$

The kinematic model of a fixed-wing UAV is given by a rotation matrix from the linear velocities to the time derivative of position and a transformation matrix from the angular velocities to the time derivative of orientation,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.3a}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.3b}$$

where $s(\cdot)$ and $c(\cdot)$ are shorthand notations for the trigonometric functions $\sin(\cdot)$ and $\cos(\cdot)$, respectively.

A sought-after flight condition is the coordinated-turn. In a coordinated turn, the lateral acceleration in the body-frame is zero [2]. A coordinated turn

is preferred for conventional aircraft since it is more comfortable for passengers, while for fixed-wing UAVs it is preferred since it minimizes the drag force. The coordinated turn also gives a simplified relationship between the heading rate and the roll angle,

$$\dot{\psi} = \frac{g}{V_a} \tan \phi, \tag{2.4}$$

where V_a is the airspeed and g is the gravitational constant.

In the horizontal plane, the course angle is the relevant angle for path following. Due to wind and sideslip, the aircraft may not be flying in the direction it is headed at. The course angle is the actual angle of travel, which is not always equal to the heading angle. The course angle is offset from the heading angle by the crab angle,

$$\chi = \psi + \beta_c, \tag{2.5}$$

where χ is the course angle and β_c is the crab angle. The crab angle describes the offset due to both the wind and the lateral velocity, while the sideslip angle describes the offset due to lateral velocity. A sideslip controller which seeks to keep zero sideslip is beneficial for fixed-wing UAVs [2], but the sideslip angle must be estimated on small fixed-wing UAVs due to the limited sensors available [13].

In the vertical plane, the flight path angle is the relevant angle for path following. The flight path angle is the actual angle of travel in the vertical plane, while the pitch angle is the angle the aircraft is directed at. In the absence of wind, the flight path angle is offset from the pitch angle by the angle of attack,

$$\gamma = \theta - \alpha \tag{2.6}$$

where γ is the flight path angle and α is the angle of attack. The flight path angle is measured positive from s_h-axis in a counter-clockwise direction [2]. As this differs from the convention of measuring angles positive from the x-axis in a clockwise direction in the horizontal plane, the flight path angle is offset from the vertical course "course angle",

$$\gamma = \frac{\pi}{2} - \chi_v \tag{2.7}$$

where χ_v is the "course angle" in the vertical plane as given by the curve parameterizations reused from the horizontal plane. The concept of a vertical "course angle" does not really exist in the aircraft literature, but is used in this thesis to allow for reuse of the horizontal two-dimensional path generation in the vertical plane. So it is simply an artifact of the reuse of methods.

For path generation purposes, the course angle and the flight path angle, along with their derivatives, are the defining properties of the path. More details on the dynamics and modeling of fixed-wing UAVs are found in Beard and McLain [2] and Stevens *et al.* [3].

2.3 Path Representation

A path can be constructed by combining n subpaths as

$$\mathcal{P} = \bigcup_{i=1}^{n} \mathcal{P}_i, \tag{2.8}$$

which results in a piecewise-defined curve [14]. The transition points of the piecewise curve are of great importance for the properties of the overall path. Subsequent subpaths must, at least, be geometrically connected. Other requirements on subsequent subpaths are discussed later in this chapter.

A subpath can be expressed by the set

$$\mathcal{P}_i := \left\{ \mathbf{p}_i \in \mathbb{R}^2 \mid \mathbf{p}_i = \mathbf{p}_{p,i}(s) \; \forall s \in [s_{\text{lb},i}, s_{\text{ub},i}] \right\}, \tag{2.9}$$

where s is the path parameter and $\mathbf{p}_i$ is a point on the subpath [11]. This thesis uses an arc length parameterization like

$$s_{\text{lb},1} = 0, \qquad s_{\text{ub},1} = L_1, \tag{2.10a}$$
$$s_{\text{lb},2} = L_1, \qquad s_{\text{ub},2} = L_1 + L_2, \tag{2.10b}$$
$$s_{\text{lb},3} = L_1 + L_2, \qquad s_{\text{ub},3} = L_1 + L_2 + L_3, \tag{2.10c}$$
$$\vdots \qquad\qquad \vdots \tag{2.10d}$$
$$s_{\text{lb},n} = \sum_{i=1}^{n-1} L_i, \qquad s_{\text{ub},n} = \sum_{i=1}^{n} L_i, \tag{2.10e}$$

since the linear and physical relationship between the path parameter and the distance travelled is beneficial in applications where the computational load is a limiting factor [15].

All the points of the overall path are thus covered when traversing $s = [0, L]$ on $\mathcal{P}$ where $L = \sum_{i=1}^{n} L_i$ is the length of the overall path.

2.4 Mathematical Curves

A short description of curvature and arc length parameterizations of the three types of mathematical curves fundamental to the path generation in this thesis follows. The three types of curves are line segment, circular arc, and Euler spiral.

2.4.1 Curvature

The curvature κ at a point (x, y) of a curve $p(s) = (x(s), y(s))$ is the rate of change in the angle of a tangent that moves along the curve at unit speed at

that point,

$$\kappa = \frac{x'y'' - x''y'}{\left(x'^2 + y'^2\right)^{\frac{3}{2}}}. \tag{2.11}$$

Curvature is, informally, a measure of how much a curve deviates from a straight line. In other words, it describes how sharp a turn is. For differentiable curves, the curvature at a point is the curvature of the osculating circle. The osculating circle to a point on a curve is the circle that best approximates the curve around that point. The curvature of a straight line is zero and the curvature of a circle is equal to the inverse of its radius. The course angle rate is the curvature scaled by the ground speed,

$$\dot{\chi} = V_g \kappa, \tag{2.12}$$

where κ is the curvature and V_g is the ground speed. Curvature is an important value in path generation due to its connection with the turning radius, which will be discussed in detail later in this chapter.

2.4.2 Line Segment

An arc length parameterization of a line segment can be given as

$$\mathbf{p}_{\text{line}}(s) = \mathbf{p}_0 + s\frac{\mathbf{p}_1 - \mathbf{p}_0}{\|\mathbf{p}_1 - \mathbf{p}_0\|}, \tag{2.13}$$

where s is the path parameter, $\mathbf{p}_0 = [x_0, y_0]$ is the start point and $\mathbf{p}_1 = [x_1, y_1]$ is the end point. The four parameters required to uniquely define a line segment are $\{x_0, y_0, x_1, y_1\}$.

The length of a line segment is

$$L = \|\mathbf{p}_1 - \mathbf{p}_0\|. \tag{2.14}$$

Both the tangent vector and the tangent angle

$$\mathbf{v}(s) = \frac{\mathbf{p}_1 - \mathbf{p}_0}{\|\mathbf{p}_1 - \mathbf{p}_0\|}, \tag{2.15a}$$

$$\chi(s) = \text{atan2}(v_y, v_x), \tag{2.15b}$$

are constant along a line segment. The curvature is zero along a line segment,

$$\kappa(s) = 0. \tag{2.16}$$

2.4.3 Circular Arc

An arc length parameterization of a circular arc can be given as

$$\mathbf{p}_{\text{circular}}(s) = \mathbf{p}_c + R\begin{bmatrix} \cos\left(\alpha_0 + s\frac{\text{sign}(\Delta\alpha)}{R}\right) \\ \sin\left(\alpha_0 + s\frac{\text{sign}(\Delta\alpha)}{R}\right) \end{bmatrix}, \tag{2.17}$$

where s is the path parameter, $\mathbf{p}_c = [x_c, y_c]$ is the center point of the associated circle, R is the radius of the associated circle, α_0 is the start angle of the arc, and $\Delta\alpha$ is the signed central angle of the arc. See Figure 2.1 for an illustration of the parameters. The sign of the central angle defines the direction of movement along the arc. A positive value for $\Delta\alpha$ defines a clockwise movement and a negative value a counter-clockwise movement. The valid range for the start angle α_0 is $[-\pi, \pi]$, and the valid range for the central angle $\Delta\alpha$ is $[-2\pi, 2\pi]$. Note that a full circle is allowed in both directions. The radius R must be strictly positive. The five parameters required to uniquely define a circular arc are $\{x_c, y_c, R, \alpha_0, \Delta\alpha\}$.

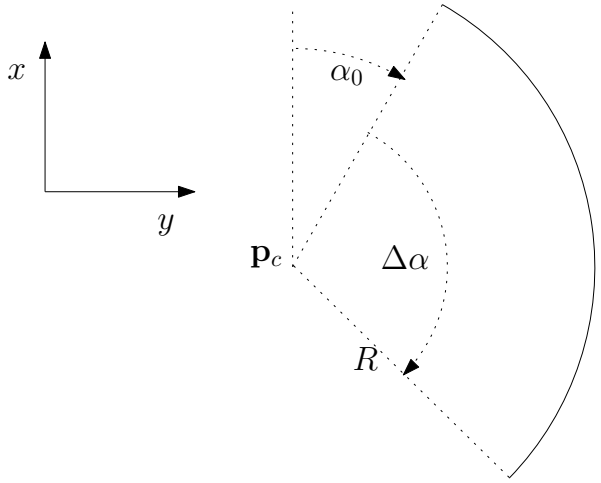


Figure 2.1: Parameters of a circular arc

The length of a circular arc is

$$L = R|\Delta\alpha|. \tag{2.18}$$

The tangent vector along a circular arc is

$$\mathbf{v}(s) = \text{sign}(\Delta\alpha) \begin{bmatrix} -\sin\left(\alpha_0 + s\frac{\text{sign}(\Delta\alpha)}{R}\right) \\ \cos\left(\alpha_0 + s\frac{\text{sign}(\Delta\alpha)}{R}\right) \end{bmatrix}. \tag{2.19}$$

The course angle along a circular arc is

$$\chi(s) = \text{ssa}\left(\alpha_0 + s\frac{\text{sign}(\Delta\alpha)}{R} + \text{sign}(\Delta\alpha)\frac{\pi}{2}\right), \tag{2.20}$$

where ssa$(\cdot)$ is the *smallest signed angle* operator defined in [4]. The ssa$(\cdot) : \mathbb{R} \to [-\pi, \pi)$ operator maps the unconstrained difference between two angles $\tilde{\beta} = \beta_1 - \beta_0 \in \mathbb{R}$ to the smallest difference between the angles, $\bar{\beta}_{\text{ssa}} = \text{ssa}\left(\tilde{\beta}\right) \in [-\pi, \pi)$.

The course angle change from the start of the circular arc to the end is given by the central angle,

$$\Delta\chi = \chi(L) - \chi(0) = \mathrm{ssa}(\Delta\alpha). \tag{2.21}$$

The curvature is constant along a circular arc and equal to the reciprocal of the radius of the associated circle, where the sign corresponds to a clockwise or counter-clockwise movement,

$$\kappa(s) = \mathrm{sign}(\Delta\alpha)\frac{1}{R}. \tag{2.22}$$

2.4.4 Euler Spiral

An Euler spiral is a curve with a linear relationship between its arc length and its curvature. Throughout history, it has been reinvented several times and is therefore also known by the names Cornu spiral or clothoid. The mathematical equations behind the curve are the Fresnel integrals. The Fresnel integrals do not have closed-form solutions, but accurate approximations exist [16]. The linear curvature change along the curve enables its usage as a transition curve from the zero curvature of a line to the constant curvature of a circular arc, and vice versa.

Inspired by Dahl [9], the Euler spiral is derived from the linear relationship between its arc length and its curvature,

$$\kappa(s) = \kappa_0 + s\frac{\Delta\kappa}{L}, \tag{2.23}$$

where κ_0 is the signed initial curvature, $\Delta\kappa$ is the signed curvature change of the spiral and L is the length of the spiral.

As the signed curvature is the rate at which the tangent vector of the curve rotates [10], the course angle is

$$\chi(s) = \mathrm{ssa}\left(\chi_0 + \int_0^s \kappa(\tau)\,d\tau\right) = \mathrm{ssa}\left(\chi_0 + s\kappa_0 + \frac{s^2}{2}\frac{\Delta\kappa}{L}\right), \tag{2.24}$$

where χ_0 is the initial course angle.

The tangent vector along the Euler spiral is given by the course angle as

$$\mathbf{v}(s) = \begin{bmatrix} \cos(\chi(s)) \\ \sin(\chi(s)) \end{bmatrix} = \begin{bmatrix} \cos\left(\chi_0 + s\kappa_0 + \frac{s^2}{2}\frac{\Delta\kappa}{L}\right) \\ \sin\left(\chi_0 + s\kappa_0 + \frac{s^2}{2}\frac{\Delta\kappa}{L}\right) \end{bmatrix}. \tag{2.25}$$

An arc length parameterization of an Euler Spiral can thus be given as

$$\mathbf{p}_{\text{spiral}}(s) = \mathbf{p}_0 + \int_0^s \mathbf{v}(\tau_1)\, d\tau_1 \tag{2.26a}$$

$$= \mathbf{p}_0 + \begin{bmatrix} \int_0^s \cos\left(\chi_0 + \tau_1\kappa_0 + \frac{\tau_1^2}{2}\frac{\Delta\kappa}{L}\right) d\tau_1 \\ \int_0^s \sin\left(\chi_0 + \tau_1\kappa_0 + \frac{\tau_1^2}{2}\frac{\Delta\kappa}{L}\right) d\tau_1 \end{bmatrix} \tag{2.26b}$$

$$= \mathbf{p}_0 + \sqrt{\frac{L}{|\Delta\kappa|}} \begin{bmatrix} \int_0^s \sqrt{\frac{|\Delta\kappa|}{L}} \cos\left(\chi_0 + \tau\kappa_0\sqrt{\frac{L}{|\Delta\kappa|}} + \frac{\tau^2}{2}\text{sign}(\Delta\kappa)\right) d\tau \\ \int_0^s \sqrt{\frac{|\Delta\kappa|}{L}} \sin\left(\chi_0 + \tau\kappa_0\sqrt{\frac{L}{|\Delta\kappa|}} + \frac{\tau^2}{2}\text{sign}(\Delta\kappa)\right) d\tau \end{bmatrix} \tag{2.26c}$$

$$= \mathbf{p}_0 + w \begin{bmatrix} \int_0^{\frac{s}{w}} \cos\left(\chi_0 + \tau\kappa_0 w + \frac{\tau^2}{2}\text{sign}(\Delta\kappa)\right) d\tau \\ \int_0^{\frac{s}{w}} \sin\left(\chi_0 + \tau\kappa_0 w + \frac{\tau^2}{2}\text{sign}(\Delta\kappa)\right) d\tau \end{bmatrix}, \tag{2.26d}$$

where s is the path parameter, $\mathbf{p}_0 = [x_0, y_0]$ is the start point, χ_0 is the initial course angle, κ_0 is the signed initial curvature, $\Delta\kappa$ is the signed curvature change of the spiral, and w is the scale factor. The scale factor w must be strictly positive.

A variable substitution of $\tau = \sqrt{\frac{|\Delta\kappa|}{L}}\tau_1$ is done from Equation (2.26b) to Equation (2.26c), and an extraction of the scale factor $w = \sqrt{\frac{L}{|\Delta\kappa|}}$ is done from Equation (2.26c) to Equation (2.26d). The scale factor is extracted to simplify the spline approximation of the Euler spiral in later chapters.

The parameterization is a combination of the parameterization in Dahl [9] and the parameterization in Pinchetti *et al.* [15] but extended to work with signed curvature change. The parameterization in Dahl [9] uses an initial course angle and an initial curvature, while the parameterization in Pinchetti *et al.* [15] uses a scale factor to simplify the later spline approximation. Combining these, and adding the sign$(\cdot)$ operator to take care of the sign of the curvature change, gives the above parameterization. The signed curvature change enables the spiral to be used in counter-clockwise movements without multiplying it with a reflection matrix. The six parameters required to uniquely define an Euler spiral are $\{x_0, y_0, \chi_0, \kappa_0, \Delta\kappa, w\}$.

The length of an Euler spiral is

$$L = w^2|\Delta\kappa|. \tag{2.27}$$

The course angle change from the start of an Euler spiral to the end is

$$\Delta\chi = \chi(L) - \chi(0) = \text{ssa}\left(L\kappa_0 + \frac{L}{2}\Delta\kappa\right). \tag{2.28}$$

2.5 Continuity

As briefly mentioned earlier, the transition points on the piecewise curve that is the overall path are of great importance for the properties of the overall path. Continuity issues can arise when the three types of mathematical curves introduced are connected to form an overall path. Parametric continuity and geometric continuity are two different measures of continuity that relate to trajectory tracking and path following scenarios, respectively.

2.5.1 Parametric Continuity

Parametric continuity concerns both the magnitude and orientation of the derivatives of the curve. A path is C^n continuous if the first n derivatives are continuous in both magnitude and direction at all points [17].

A path is C^0 continuous if all subsequent subpaths are connected,

$$\mathbf{p}_{p,i}(s_{\text{ub},i}) = \mathbf{p}_{p,i+1}(s_{\text{lb},i+1}) \; \forall i \in [1, n-1]. \tag{2.29}$$

A path is C^1 continuous if it is C^0 continuous and the velocity vector (tangent vector) is continuous in magnitude and direction at transition points,

$$\frac{d}{ds}\mathbf{p}_{p,i}(s_{\text{ub},i}) = \frac{d}{ds}\mathbf{p}_{p,i+1}(s_{\text{lb},i+1}) \; \forall i \in [1, n-1]. \tag{2.30}$$

A path is C^2 continuous if it is C^1 continuous and the acceleration vector is continuous in magnitude and direction at transition points,

$$\frac{d^2}{d^2s}\mathbf{p}_{p,i}(s_{\text{ub},i}) = \frac{d^2}{d^2s}\mathbf{p}_{p,i+1}(s_{\text{lb},i+1}) \; \forall i \in [1, n-1]. \tag{2.31}$$

2.5.2 Geometric Continuity

Geometric continuity is a relaxed form of parametric continuity, which only concerns the orientation of the derivatives of the curve. A path is G^n continuous if the first n derivatives are continuous in direction at all points [17].

A path is G^0 continuous if all subsequent subpaths are connected,

$$\mathbf{p}_{p,i}(s_{\text{ub},i}) = \mathbf{p}_{p,i+1}(s_{\text{lb},i+1}) \; \forall i \in [1, n-1]. \tag{2.32}$$

A path is G^1 continuous if it is G^0 continuous and the tangent (course) angle is continuous at transition points,

$$\chi_{p,i}(s_{\text{ub},i}) = \chi_{p,i+1}(s_{\text{lb},i+1}) \; \forall i \in [1, n-1]. \tag{2.33}$$

A path is G^2 continuous if it is G^1 continuous and the curvature is continuous at transition points,

$$\kappa_{p,i}(s_{\text{ub},i}) = \kappa_{p,i+1}(s_{\text{lb},i+1}) \; \forall i \in [1, n-1]. \tag{2.34}$$

Geometric continuity is equal to parametric continuity when the curves are arc-length parameterized [17]. As the three types of curves in this chapter are arc-length parameterized, a path generated based on these will have equal parametric continuity and geometric continuity. However, the later spline approximation of the path is only a approximation of geometric continuity. The splines will be computed independently to improve approximation accuracy which causes the magnitude of the tangent vector to be discontinuous at transition points.

2.6 Path Constraints for Fixed-Wing UAVs

A common flight maneuver for fixed-wing UAVs is roll-to-turn, also known as bank-to-turn. It is the primary way to change the heading direction for fixed-wing aircraft. The lift acting on a fixed-wing aircraft is often modeled as a force perpendicular to the wings. If the roll angle is zero in horizontal flight, the lift force acts straight upwards. However, with a non-zero roll angle, the lift force is angled away from the vertical axis. The horizontal component of the lift force creates a centripetal acceleration which causes the aircraft to turn in a circular motion. As the roll angle increases, the horizontal component increases and the vertical component decreases. A minimum of upwards lift is needed to counteract the downwards gravitational force to keep the aircraft flying. The roll angle is therefore limited to a maximum value, where the limit depends on the properties of the aircraft. As an example, ArduPilot proposes a value of 65 degrees for fast aerobatic planes [18].

The coordinated turn Equation (2.4) gives the turn rate (course angle rate) as a function of the roll angle, if zero wind and zero sideslip is assumed,

$$\dot{\chi} = \frac{g}{V_a} \tan \phi, \tag{2.35}$$

which shows that the turn rate increases with an increasing roll angle. The limit on the roll angle limits therefore the maximum turning rate. The turning radius in horizontal flight relates to the turn rate as

$$R = \frac{V_g}{\dot{\chi}}, \tag{2.36}$$

which shows that a maximum turning rate is equivalent to a minimum turning radius. As the curvature of a circle is the reciprocal of the radius, the minimum turning radius is also equivalent to maximum curvature.

The roll angle of a fixed-wing UAV can not be changed instantly due to inertia along the roll axis. As an example, ArduPilot uses a default value of 60 deg/sec as the roll rate [18]. Since the turn rate is a function of the roll angle in a coordinated turn, the turn rate can not be changed instantly either. This limits the types of subpaths that can be connected together. The curvature

of a straight line is zero, which gives a zero turn rate, and the curvature of a circle is constant, which gives a constant turn rate. A direct transition from a line segment to a circular arc gives a discontinuous jump in curvature. The discontinuous jump in curvature gives a discontinuous turn rate which also gives a discontinuous roll angle. A fixed-wing UAV is therefore not able to directly transition from a straight line to a circle.

The lift force counteracting the downwards gravitational force is what keeps a fixed-wing aircraft flying. If the vertical component of the lift force is larger than the gravitational force, then the aircraft climbs, and vice versa for descending. A fixed-wing UAV climbs/descends by pitching up/down combined with increasing or decreasing the thrust. At a low angle of attack, an increase in angle of attack increases the lift force. However, when the angle of attack exceeds a certain value, the lift force drops drastically [2]. This effect is known as stall. The flight path angle, or the pitch angle, is therefore usually limited to a maximum value, in the same way that the roll angle is limited. The limit depends on the properties of the aircraft. As an example, ArduPilot uses a default value of 25 deg as the maximum pitch angle [18]. The climbing rate depends on the flight path angle and the ground speed,

$$\dot{h} = V_g \sin\gamma. \tag{2.37}$$

To summarize, path generation for fixed-wing UAVs must take into account the minimum turning radius, the rolling motion rate, and the climbing rate for the path to be feasible for fixed-wing UAVs. The minimum turning radius corresponds to a maximum curvature, the rolling motion rate corresponds to a continuous curvature with a certain slope, and the climbing rate corresponds to a maximum flight path angle.

Chapter 3

2D Path Generation

This chapter presents three methods for interpolating path generation from a sequence of waypoints in 2D:

- Piecewise Linear Path,
- 2D Dubins Path,
- 2D Extended Dubins Path.

The 2D Dubins Path is mainly based on a method in Dahl [9], but updated to take care of several edge cases. The core logic is unchanged, but extra logic is developed to handle waypoints with no change in course angle and to avoid special cases with complete circle turns. A simple extension to take care of an initial position with an initial course angle and a specified course angle at the last waypoint is also developed. As the method later is used in the vertical plane, where the flight path angle often does not change at waypoints and where a complete circle turn becomes a loop, it was crucial to develop the extra logic. Concise algorithmic descriptions of the method and the improvements are given in Algorithm 3.1 and Algorithm 3.2.

The 2D Extended Dubins Path is also based on a method in Dahl [9]. In the same manner, a slightly more complex extension to take care of an initial position with an initial course angle and a specified course angle at the last waypoint is developed. This thesis shows how this method is also useable for online path generation as the method is altered such that certain computations can be reused and thus avoids online numerical integration. Concise algorithmic descriptions of the method and the improvements are given in Algorithm 3.3 and Algorithm 3.4.

The chapter focuses on thorough explanations with all the necessary calculations, some supporting figures, and concise summaries of the methods in form of algorithms. As all the necessary building blocks are presented in this chapter or earlier chapters, the methods should be easy to implement. Implementations of the methods in MATLAB are also bundled with this thesis.

	Continuity	
Method	χ/G^1	$\dot{\chi}/G^2$
Piecewise Linear Path	×	×
2D Dubins Path	✓	×
2D Extended Dubins Path	✓	✓

Table 3.1: Continuity of 2D Interpolating Path Methods

3.1 Summary of 2D Path Generation

Interpolating paths passes through the waypoints exactly, in contrast to the smoothing paths in Fossen [4] and Lekkas [8], and are used when the waypoints represent locations of interest. The methods presented are compromises between complexity and desirable properties.

The piecewise linear path is the simplest method, but it generates a path with discontinuities in the course angle. From the definition of continuity in Section 2.5, the path is G^0 continuous. A discontinuous course angle requires the vehicle to instantly change course angle (e.g. by stopping and turning on the spot) to follow the path, and the piecewise linear path is therefore unsuitable for fixed-wing UAVs.

The 2D Dubins Path uses circular arcs at the waypoints to provide a continuous course angle. However, it generates a path with discontinuities in the course angle rate. The path is G^1 continuous. A discontinuous course angle rate requires the vehicle to instantly change course angle rate (e.g. by instantly changing roll angle for fixed-wing UAVs that roll-to-turn). The method only involves vector algebra and trigonometric functions. An iterative fix to avoid an edge case with complete circle turns is proposed by this thesis.

The 2D Extended Dubins Path is a more complex method where Euler spirals are placed at the entrance and exit of every circular arc to provide a continuous course angle rate. The path is G^2 continuous. The method involves solving the Fresnel integrals, which do not have closed-form solutions and are therefore solved numerically. With an assumption of constant speed, this thesis shows how the Fresnel integrals can be pre-computed offline and simply be rotated in the right direction online with rotation and reflection matrices. Since the Fresnel integrals are part of the parameterization of the Euler spirals, the integrals must still be solved online during guidance if the path is used directly as the reference path. However, with the later spline approximation, this is avoided since the splines do not have any integrals and do not need to solve any integrals to approximate the Euler spirals.

The geometric continuity of the methods is summarized in Table 3.1.

3.2 Piecewise Linear Path

The simplest path to generate from a sequence of waypoints is a piecewise linear curve connecting the waypoints together. It is computationally very efficient but is only G^0 continuous as the course angle is discontinuous at the transition points. Circles of acceptance around the waypoints can be used as a switching mechanism [4].

3.3 2D Dubins Path

The 2D Dubins Path is a G^1 continuous path combined of subpaths of line segments and circular arcs. As the path transitions directly from the zero curvature of a line segment to the constant curvature of a circular arc, and vice versa, the curvature of the path is discontinuous.

3.3.1 Dubins Path

The Dubins Path is a famous path that is the shortest curve that connects two two-dimensional poses with constant speed and a constraint on the maximum curvature [19]. It is the time-optimal curve for the simple kinematic model known as the Dubins Car [6]. The Dubins Car has a minimum turning radius which corresponds to a maximum curvature limit. The path is constructed by combining line segments and circular arcs of maximum curvature. It is known that the optimal path is either of type CLC or type CCC, or a subset of those, where C represents a circular arc and L represents a line segment [19].

For fixed-wing UAVs, this relates to finding the shortest path from a start position with a specified course angle to an end position with a specified course angle, while taking into account the minimum turning radius and where the start and end positions are at the same altitude [7]. The minimum turning radius constraint is equivalent to a maximum curvature constraint.

Tsourdos *et al.* [7] presents both an analytic geometric method and a differential geometric method for generating the Dubins Path between two poses. The analytical geometric method is extended to an interpolating path between a sequence of waypoints in Kußmaul [20], and the differential geometrical method is extended to an interpolating path between a sequence of waypoints in Dahl [9].

This thesis builds upon the differential geometric method for an interpolating path in Dahl [9] but updates it with extra logic for edge cases discovered during the work of this thesis. This thesis calls the interpolating path between a sequence of waypoints for 2D Dubins Path to differentiate it from the original Dubins Path, which is just between two poses.

3.3.2 Radius of Circular Arcs

The length of the circular arcs depends on the change in course angle during the turns,

$$L = V_g \frac{|\Delta\chi|}{\dot{\chi}_{\text{des}}}, \tag{3.1}$$

where V_g is the ground speed, $\Delta\chi$ is the change in course angle and $\dot{\chi}_{\text{des}}$ is the desired turn rate. As the length of a circular arc is $L = R|\Delta\alpha|$ and the change in course angle over a circular arc is $\Delta\chi = \text{ssa}(\Delta\alpha)$ from Equation (2.18) and Equation (2.21), respectively, the radius of the circular arcs are

$$R = \frac{V_g}{\dot{\chi}_{\text{des}}}. \tag{3.2}$$

The desired turn rate can e.g. be chosen as the maximum turn rate in a coordinated turn from Equation (2.35). With an assumption of constant speed, the radius of all the circular arcs are equal.

3.3.3 Circle Centers

The cornerstone of the method is to place circular arcs at the waypoints to turn in the direction of the next waypoint and connect the circular arcs together with line segments. The method uses a simplistic computation of the orientations at the intermediate waypoints and places the turning circular arcs based on these.

First, the tangent vectors entering and exiting the waypoints on a piecewise linear path is computed,

$$\mathbf{v}_{\text{in},i} = \frac{\mathbf{p}_{\text{wp},i} - \mathbf{p}_{\text{wp},i-1}}{\|\mathbf{p}_{\text{wp},i} - \mathbf{p}_{\text{wp},i-1}\|}, \qquad \forall i \in [2, n], \tag{3.3a}$$

$$\mathbf{v}_{\text{out},i} = \frac{\mathbf{p}_{\text{wp},i+1} - \mathbf{p}_{\text{wp},i}}{\|\mathbf{p}_{\text{wp},i+1} - \mathbf{p}_{\text{wp},i}\|}, \qquad \forall i \in [1, n-1], \tag{3.3b}$$

where $\mathbf{p}_{\text{wp},i}$ is a waypoint. The tangent vector entering the first waypoint is given by the initial course angle,

$$\mathbf{v}_{\text{in},1} = \begin{bmatrix} \cos(\chi_0) \\ \sin(\chi_0) \end{bmatrix}, \tag{3.4}$$

as the initial position is set as the first waypoint. The tangent vector exiting the last waypoint is given by the final course angle,

$$\mathbf{v}_{\text{out},n} = \begin{bmatrix} \cos(\chi_n) \\ \sin(\chi_n) \end{bmatrix}. \tag{3.5}$$

The incorporation of an initial position with an initial course angle and the final course angle is a small update this thesis makes to the method from [9].

The orientation at an intermediate waypoint is given by the tangent vector at the waypoint, referred to as the waypoint vector. The waypoint vector $\mathbf{v}_{\text{wp},i}$ is for simplicity chosen to be the mean of the entering and exiting tangent vectors on the piecewise linear path,

$$\mathbf{v}_{\text{wp},i} = \frac{\mathbf{v}_{\text{in},i} + \mathbf{v}_{\text{out},i}}{\|\mathbf{v}_{\text{in},i} + \mathbf{v}_{\text{out},i}\|}, \qquad \forall i \in [2, n-1], \tag{3.6a}$$

$$\rho_i = -\text{sign}\left(v_{\text{in},i,y} v_{\text{out},i,x} - v_{\text{in},i,x} v_{\text{out},i,y}\right), \qquad \forall i \in [2, n-1], \tag{3.6b}$$

and ρ_i is the direction of rotation from the entering tangent vector to the exiting tangent vector. When a waypoint has no change in course angle, the waypoint vector at the previous waypoint is slightly altered, more details on this is shown in Section 3.3.7. Such waypoints are redundant for a piecewise linear path and may be removed, but not for paths with turning circles since the path may not go through that waypoint if it is removed. The simplistic computation of the intermediate poses may also in some special cases cause the path to have a complete circle turn at a waypoint. To avoid this, this thesis proposes an iterative fix that alters the waypoint vector in Section 3.3.8.

The waypoint vector at the first and last waypoint is not chosen as the mean of the entering and exiting tangent vectors, but as

$$\mathbf{v}_{\text{wp},1} = \mathbf{v}_{\text{in},1}, \tag{3.7a}$$

$$\mathbf{v}_{\text{wp},n} = \mathbf{v}_{\text{out},n}, \tag{3.7b}$$

to satisfy the initial and final course angle, and with rotational directions, ρ_1 and ρ_n, given by Equation (3.6b).

The circle centers of the circular arcs are chosen such that the waypoint vector is tangential to the turning circle and rotates in the same direction as the circle. This is achieved by rotating the waypoint vector 90° in the direction of rotation and scaling it by the radius of the circular arc to find the circle center,

$$\mathbf{v}_{\text{wp}\to\text{c},i} = \mathbf{R}_z\left(\rho_i \frac{\pi}{2}\right)\mathbf{v}_{\text{wp},i}, \tag{3.8a}$$

$$\mathbf{p}_{\text{c},i} = \mathbf{p}_{\text{wp},i} + R\mathbf{v}_{\text{wp}\to\text{c},i}, \tag{3.8b}$$

where $\mathbf{p}_{\text{c},i}$ is the circle center, $\mathbf{R}_z$ is a rotation matrix and R is the radius of the circular arcs. See Figure 3.1 for an example.

Now that the circle centers are placed, the circles are connected together with line segments. A wheel-over point $\mathbf{p}_{\text{wo},i}$ defines the start of a circular arc at a waypoint and a pull-out point $\mathbf{p}_{\text{po},i}$ defines the end. A line segment is connected from the pull-out point at a waypoint to the wheel-over point at the next waypoint. The wheel-over point thus marks a transition from a line segment to a circular arc, and a pull-out point vice versa. The method computes the wheel-over and pull-out points by using the fact that a line segment must be tangential to both the circles it connects together [9]. This gives a trigonometric relationship that depends on the rotational direction at the waypoints.

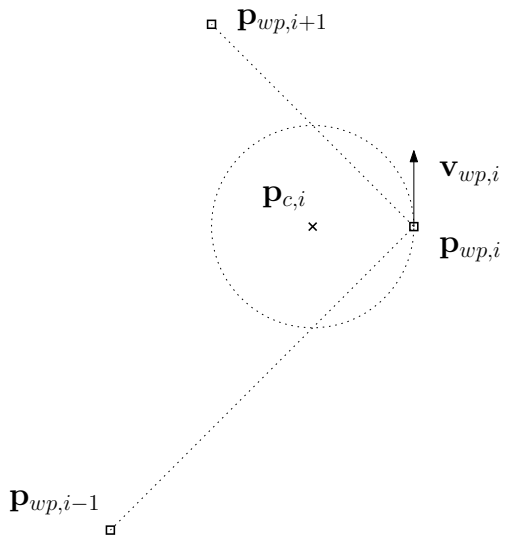


Figure 3.1: 2D Dubins Path: Intermediate pose at a waypoint

3.3.4 Different Consecutive Rotations

When the circles at two subsequent waypoints turn in opposite directions, the line segment is an internal tangent [7]. An internal tangent crosses the circle center line segment,

$$\mathbf{v}_{\mathrm{c}\to\mathrm{c},i} = \mathbf{p}_{\mathrm{c},i+1} - \mathbf{p}_{\mathrm{c},i}, \quad \forall i \in [1, n-1], \tag{3.9}$$

which gives a right triangle between the circle center, the pull-out point and the half-way point on the circle center line segment. See Figure 3.2 for an illustration. The unit vector from the circle center to the pull-out point, $\mathbf{v}_{\mathrm{c}\to\mathrm{po},i}$ is given by a rotation of the circle center line segment in the opposite direction of the turn,

$$\beta_i = \arccos\left(\frac{R}{\|\frac{1}{2}\mathbf{v}_{\mathrm{c}\to\mathrm{c},i}\|}\right), \tag{3.10a}$$

$$\mathbf{v}_{\mathrm{c}\to\mathrm{po},i} = \mathbf{R}_z(-\rho_i\beta_i)\frac{\mathbf{v}_{\mathrm{c}\to\mathrm{c},i}}{\|\mathbf{v}_{\mathrm{c}\to\mathrm{c},i}\|}, \tag{3.10b}$$

$$\mathbf{v}_{\mathrm{c}\to\mathrm{wo},i+1} = -\mathbf{v}_{\mathrm{c}\to\mathrm{po},i}, \tag{3.10c}$$

where β_i is the top angle of the right triangle. The unit vector from the next circle center to the next wheel-over point, $\mathbf{v}_{\mathrm{c}\to\mathrm{wo},i+1}$, goes in the opposite direction.

3.3.5 Equivalent Consecutive Rotations

When the circles at two subsequent waypoints turn in the same directions, the line segment is an external tangent [7]. An external tangent is parallel to the circle center line segment.

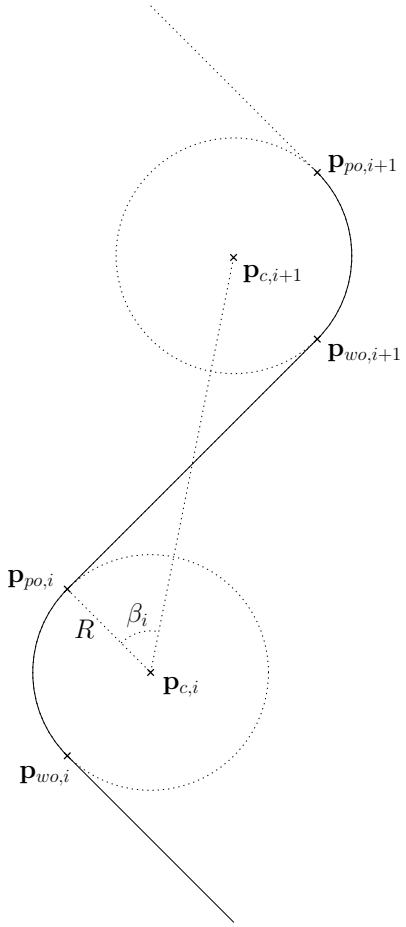


Figure 3.2: 2D Dubins Path: Different consecutive rotations

As the external tangent is parallel to the circle center line segment, the angle of rotation is simply 90°, as shown in Figure 3.3. The unit vector from the circle center to the pull-out point, $\mathbf{v}_{\text{c}\to\text{po},i}$ is given by a rotation of the circle center line segment in the opposite direction of the turn,

$$\mathbf{v}_{\text{c}\to\text{po},i} = \mathbf{R}_z\left(-\rho_i \frac{\pi}{2}\right) \frac{\mathbf{v}_{\text{c}\to\text{c},i}}{\|\mathbf{v}_{\text{c}\to\text{c},i}\|}, \tag{3.11a}$$

$$\mathbf{v}_{\text{c}\to\text{wo},i+1} = \mathbf{v}_{\text{c}\to\text{po},i}. \tag{3.11b}$$

The unit vector from the next circle center to the next wheel-over point, $\mathbf{v}_{\text{c}\to\text{wo},i+1}$, goes in the same direction.

3.3.6 Pull-Out and Wheel-Over Points

The pull-out point and the wheel-over point are calculated based on the radius of the circular arcs and the unit vectors given by Equation (3.10) or Equa-

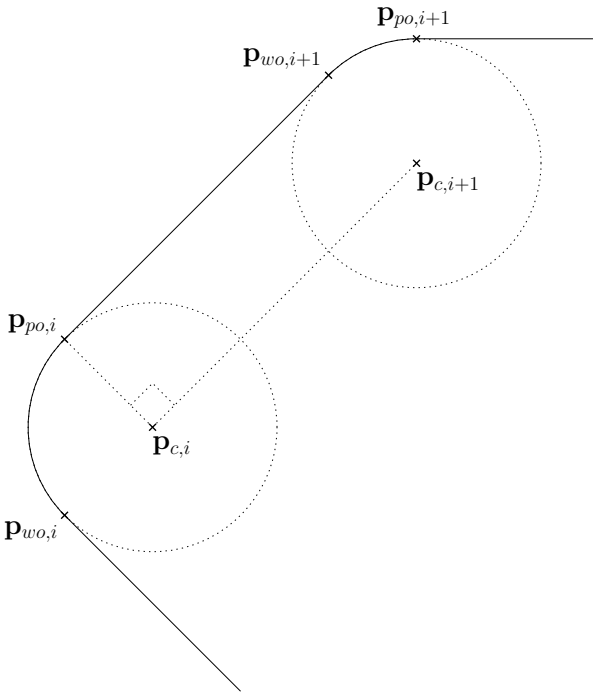


Figure 3.3: 2D Dubins Path: Equivalent consecutive rotations

tion (3.11), depending on the rotations at the waypoints,

$$\mathbf{p}_{\text{po},i} = \mathbf{p}_{\text{c},i} + R\mathbf{v}_{\text{c}\to\text{po},i}, \tag{3.12a}$$

$$\mathbf{p}_{\text{wo},i+1} = \mathbf{p}_{\text{c},i+1} + R\mathbf{v}_{\text{c}\to\text{wo},i+1}. \tag{3.12b}$$

See Figure 3.2 and Figure 3.3 for an example of where wheel-over and pull-out points are placed with two different consecutive rotations and with two equivalent consecutive rotations.

The first wheel-over point is placed at the first waypoint and the last pull-out point is placed at the last waypoint,

$$\mathbf{p}_{\text{wo},1} = \mathbf{p}_{\text{wp},1}, \tag{3.13a}$$

$$\mathbf{p}_{\text{po},n} = \mathbf{p}_{\text{wp},n}. \tag{3.13b}$$

3.3.7 Waypoints With Zero Change in Course Angle

For waypoints with zero change in course angle on the piecewise linear path, the entering and exiting tangent vectors are equal. When the entering and exiting tangent vectors are equal, there is no rotation, $\rho_i = 0$, which causes the circle centers to not be placed such that the waypoint vector is tangential to the circle, by Equation (3.8). See Figure 3.4 for an example of how the method in [9] places the circle centers and computes the pull-out and wheel-over points in such a case. The original method breaks down when there is zero change

in course angle on the piecewise linear path at a waypoint. To take care of such an edge case, this thesis proposes to change the rotational direction and waypoint vector for such waypoints. By changing the rotational direction of this waypoint to the opposite of the next waypoint and changing the previous waypoint vector to point directly towards this waypoint, the rest of method still applies to such waypoints,

$$\rho_i = -\rho_{i+1}, \quad \forall i \in \{i \mid \rho_i = 0\} \tag{3.14a}$$

$$\mathbf{v}_{\text{wp},i-1} = \mathbf{v}_{\text{out},i-1}, \quad \forall i \in \{i \mid \rho_i = 0\}. \tag{3.14b}$$

The rotational direction of such waypoints is chosen as the opposite of the next waypoint to ensure that the tangent is internal, since an external tangent would require a complete turn around the circle. The waypoints should be traversed in reverse order when applying this change to handle the edge case with several consecutive waypoints with zero course angle change.

3.3.8 Avoiding Complete Circle Turns

In some cases where the change in course angle on the piecewise linear path is very small at one waypoint and very large at the next waypoint, the generated path may contain a complete turn around the circle at a waypoint. The reason is that the wheel-over and pull-out points are placed such that the waypoint does not lie between them on the turning circle, as shown in Figure 3.5. The problem arises when the actual change in course angle required at the turning circle is very different from the one anticipated by the piecewise linear path. When the difference is large, the simplistic computation of the intermediate poses is not good enough. This thesis proposes an iterative fix for avoiding such unnecessary complete circle turns. As this method is reused in the vertical plane, where a complete circle turn is a loop, this iterative fix is crucial for the usability of the method. The idea behind the iterative fix is to use the actual change in course angle at the waypoint to determine the orientation at the waypoint and therefore place the turning circle in a better way.

First, such unnecessary complete circle turns must be detected. The wheel-over and pull-out tangent vectors are computed as

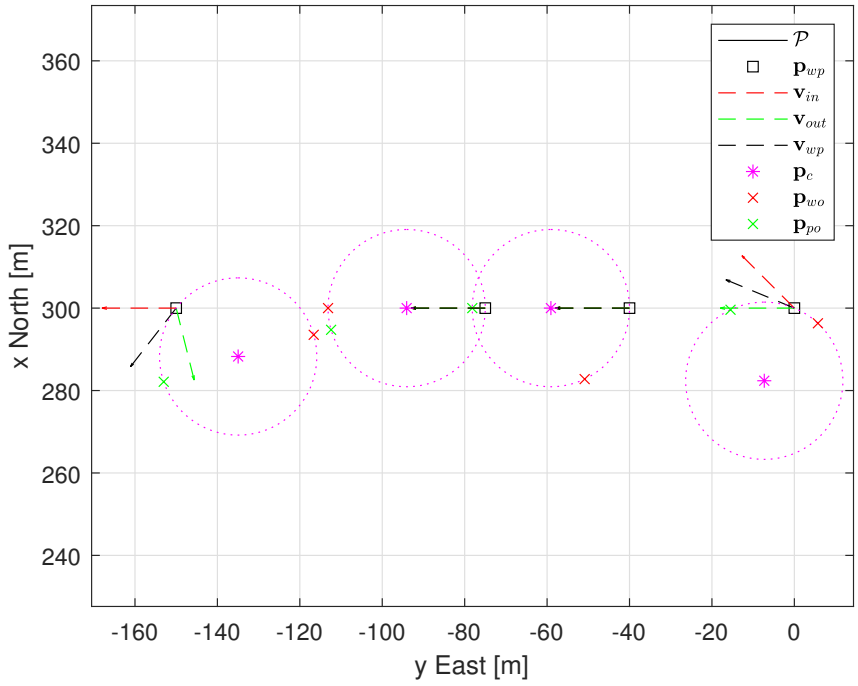
$$\mathbf{v}_{\text{po},i} = \frac{\mathbf{p}_{\text{wo},i+1} - \mathbf{p}_{\text{po},i}}{\|\mathbf{p}_{\text{wo},i+1} - \mathbf{p}_{\text{po},i}\|}, \qquad \forall i \in [1, n-1], \tag{3.15a}$$

$$\mathbf{v}_{\text{wo},i+1} = \mathbf{v}_{\text{po},i}, \tag{3.15b}$$

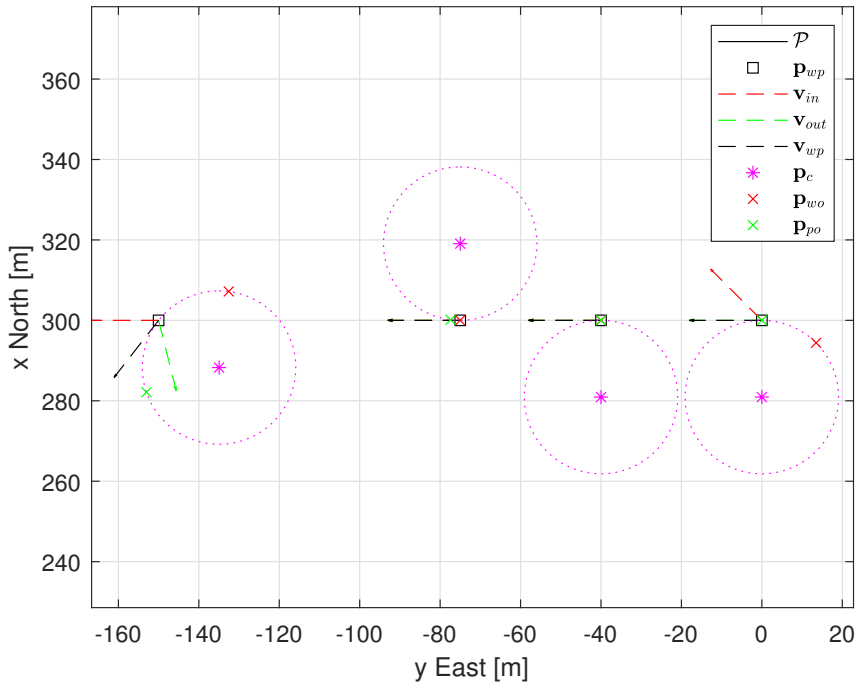
$$\mathbf{v}_{\text{wo},1} = \mathbf{v}_{\text{wp},1}, \tag{3.15c}$$

$$\mathbf{v}_{\text{po},n} = \mathbf{v}_{\text{wp},n}, \tag{3.15d}$$

where the first wheel-over and last pull-out tangent vectors are simply the waypoints vectors. If the rotational direction from the wheel-over point to the way-



(a) Original method



(b) Improved method

Figure 3.4: 2D Dubins Path: Waypoints with zero change in course angle

point or from the waypoint to the pull-out point,

$$\rho_{\text{wo}\to\text{wp},i} = -\text{sign}\left(v_{\text{wo},i,y}\, v_{\text{wp},i,x} - v_{\text{wo},i,x}\, v_{\text{wp},i,y}\right), \tag{3.16a}$$

$$\rho_{\text{wp}\to\text{po},i} = -\text{sign}\left(v_{\text{wp},i,y}\, v_{\text{po},i,x} - v_{\text{wp},i,x}\, v_{\text{po},i,y}\right), \tag{3.16b}$$

is in the opposite direction of the rotational direction at the waypoint, $\rho_{\text{wp},i}$, then there is an unnecessary complete circle turn at that waypoint. If both of the rotational directions are in the opposite direction of the turn, the turning direction must be switched.

A better computation of the waypoint vector is the mean of the wheel-over tangent vector and pull-out tangent vector,

$$\mathbf{v}_{\text{wp},i} = \frac{\mathbf{v}_{\text{wo},i} + \mathbf{v}_{\text{po},i}}{\|\mathbf{v}_{\text{wo},i} + \mathbf{v}_{\text{po},i}\|}. \tag{3.17}$$

This can not be done from the start as the wheel-over and pull-out points must be computed first. It is therefore an iterative fix. The circle center of the affected waypoint is computed again as in Equation (3.8). The wheel-over and pull-out points for this waypoint, the pull-out point for the previous waypoint, and the wheel-over point for the next waypoint must now be recomputed as the circle center is changed. This is done again with Equation (3.10) or Equation (3.11) and Equation (3.12). The waypoint vector is no longer the mean of the entering and exiting tangent vectors on the piecewise linear path, as shown in Figure 3.6.

In the worst case, this iterative fix may need several iterations. However, during the work with this thesis, one iteration was sufficient every time this special case occurred.

3.3.9 Circular Arcs

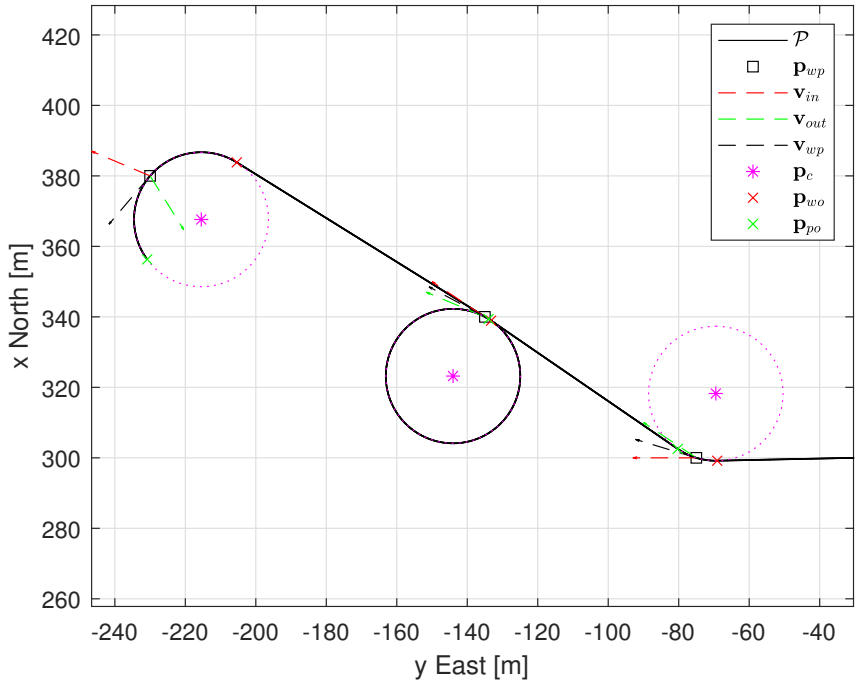
For completeness, it is shown how to compute the parameters of the circular arcs.

To simplify later methods such as three-dimensional path generation, two circular arcs are created per waypoint. One circular arc from the wheel-over point to the waypoint, and one circular arc from the waypoint to the pull-out point. Finding the distances to the waypoints is easier when there is a transition between subpaths at all waypoints. As the circle center is already computed and the radius is given directly, it is only the start angle and the central angle that needs to be computed.

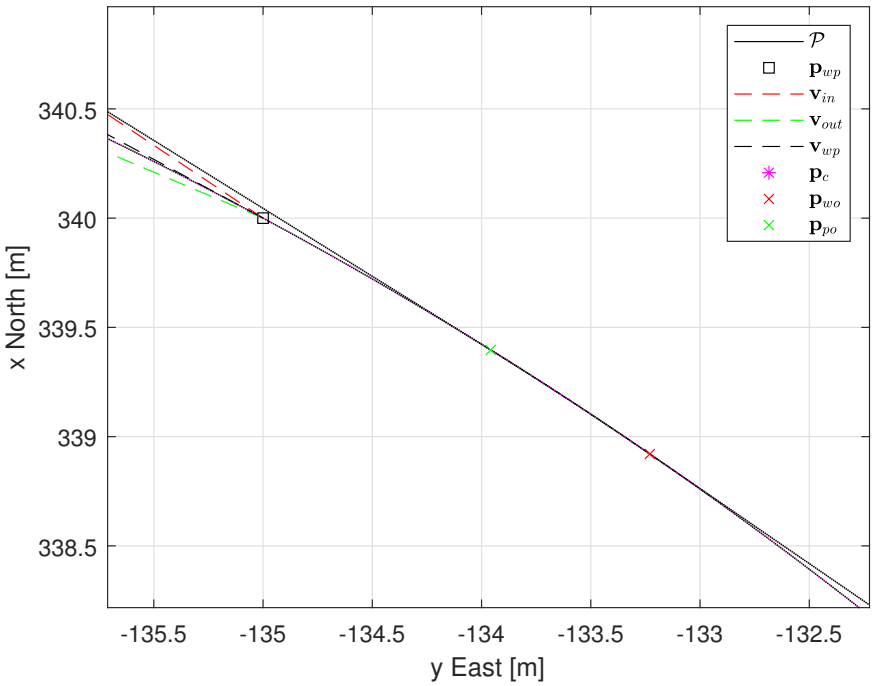
For the circular arc from the wheel-over point to the waypoint, the start angle is computed based on the vector from the circle center to the wheel-over point and the central angle is computed based on the vector from the circle center to the pull-out point and the start angle,

$$\alpha_0 = \text{atan2}(v_{\text{c}\to\text{wo},i,y}, v_{\text{c}\to\text{wo},i,x}), \tag{3.18a}$$

$$\Delta\alpha = \text{mod}\left(\text{atan2}(-v_{\text{wp}\to\text{c},i,y}, -v_{\text{wp}\to\text{c},i,x}) - \alpha_{0,i}, \rho_i 2\pi\right), \tag{3.18b}$$

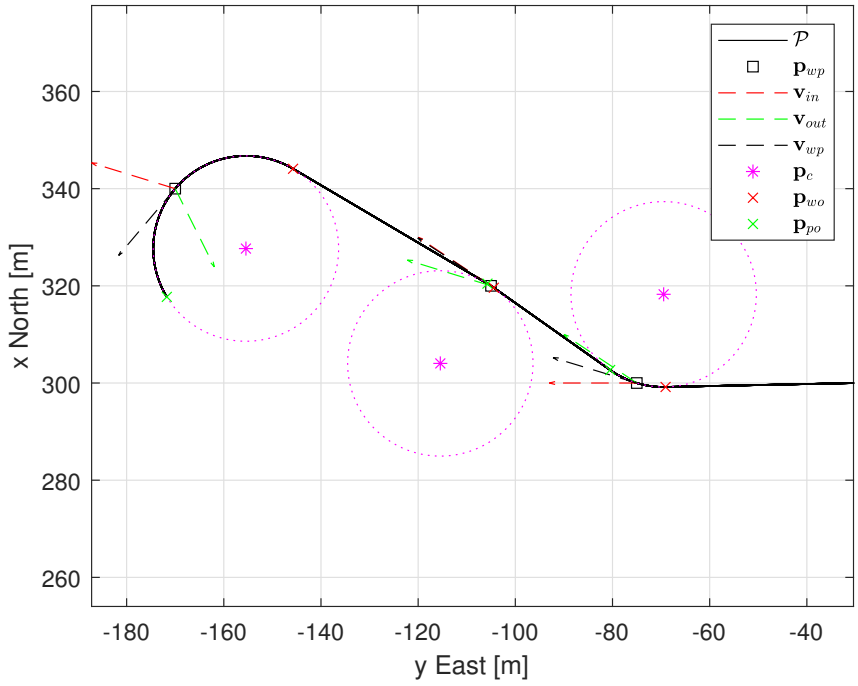


(a) A complete circle turn

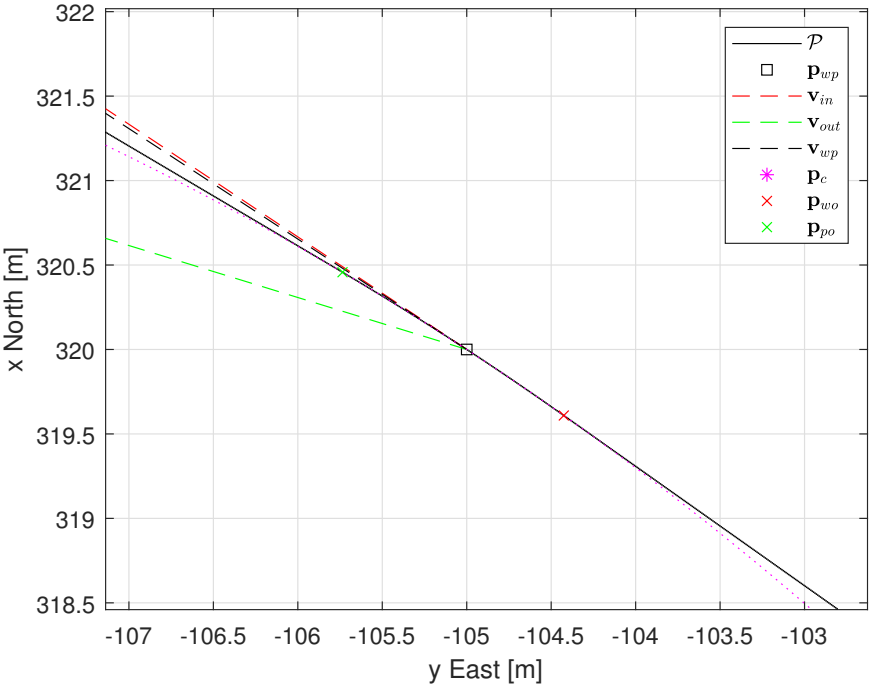


(b) Zoomed in

Figure 3.5: 2D Dubins Path: Unnecessary complete circle turn in original method



(a) No complete circle turn



(b) Zoomed in

Figure 3.6: 2D Dubins Path: Fix of unnecessary complete circle turns in improved method

where $\mathrm{mod}(\cdot)$ ensures that the central angle has the correct sign.

Likewise for the circular arc from the waypoint to the pull-out point, the start angle is computed based on the vector from the waypoint to the circle center and the central angle is computed based on the vector from the circle center to the pull-out point and the start angle,

$$\alpha_0 = \mathrm{atan2}(-v_{\mathrm{wp}\to\mathrm{c},i,y}, -v_{\mathrm{wp}\to\mathrm{c},i,x}), \tag{3.19a}$$

$$\Delta\alpha = \mathrm{mod}\left(\mathrm{atan2}(v_{\mathrm{c}\to\mathrm{po},i,y}, v_{\mathrm{c}\to\mathrm{po},i,x}) - \alpha_{0,i}, \rho_i 2\pi\right). \tag{3.19b}$$

Algorithm 3.1: 2D Dubins Path Wheel-Over and Pull-Out Points

Input: χ_0, χ_n, R, $\mathbf{p}_{\mathrm{wp},i} \quad i = 1,\ldots,n$
Result: $\mathbf{p}_{\mathrm{c}}$, ρ, $\mathbf{p}_{\mathrm{wo}}$, $\mathbf{p}_{\mathrm{po}}$

Set initial waypoint vector $\mathbf{v}_{\mathrm{wp},1}$ and rotational direction ρ_1 as in Eq. (3.7)
Set final waypoint vector $\mathbf{v}_{\mathrm{wp},n}$ and rotational direction ρ_n as in Eq. (3.7)
for $i \in [2, n-1]$ **do**
    Set waypoint vector $\mathbf{v}_{\mathrm{wp},i}$ and rotational direction ρ_i as in Eq. (3.6)
end
for $i \in [n, 1]$ **do**
    if $\rho_i = 0$ **then** `// no-course-angle-change waypoints`
        Change waypoint vector $\mathbf{v}_{\mathrm{wp},i-1}$ and rotational direction ρ_i as in Eq. (3.14)
    end
end
for $i \in [1, n]$ **do**
    Set circle center $\mathbf{p}_{\mathrm{c},i}$ as in Eq. (3.8)
end
Set first wheel-over point $\mathbf{p}_{\mathrm{wo},1}$ and last pull-out point $\mathbf{p}_{\mathrm{po},n}$ as in Eq. (3.13)
for $i \in [1, n-1]$ **do**
    if $\rho_i \neq \rho_{i+1}$ **then** `// different consecutive rotations`
        Set pull-out vector $\mathbf{v}_{\mathrm{c}\to\mathrm{po},i}$ and wheel-over vector $\mathbf{v}_{\mathrm{c}\to\mathrm{wo},i+1}$ as in Eq. (3.10)
    else `// equivalent consecutive rotations`
        Set pull-out vector $\mathbf{v}_{\mathrm{c}\to\mathrm{po},i}$ and wheel-over vector $\mathbf{v}_{\mathrm{c}\to\mathrm{wo},i+1}$ as in Eq. (3.11)
    end
    Set pull-out point $\mathbf{p}_{\mathrm{po},i}$ and wheel-over point $\mathbf{p}_{\mathrm{wo},i+1}$ as in Eq. (3.12)
end
for $i \in [1, n]$ **do**
    Set rotational direction $\rho_{\mathrm{wo}\to\mathrm{wp},i}$ and $\rho_{\mathrm{wp}\to\mathrm{po},i}$ as in Eq. (3.16)
    while $\rho_{wo\to wp,i} = \text{-}\rho_i \;||\; \rho_{wp\to po,i} = \text{-}\rho_i$ **do** `// iterative fix of 360-degree turns`
        if $\rho_{wo\to wp,i} = \text{-}\rho_i$ && $\rho_{wp\to po,i} = \text{-}\rho_i$ **then**
            Change rotational direction to $\rho_i \leftarrow -\rho_i$
        end
        Set waypoint vector $\mathbf{v}_{\mathrm{wp},i}$ as in Eq. (3.6)
        Set circle center $\mathbf{p}_{\mathrm{c},i}$ as in Eq. (3.8)
        Set pull-out point $\mathbf{p}_{\mathrm{po},i-1}$ and wheel-over point $\mathbf{p}_{\mathrm{wo},i}$ as in Eq. (3.12)
        Set pull-out point $\mathbf{p}_{\mathrm{po},i}$ and wheel-over point $\mathbf{p}_{\mathrm{wo},i+1}$ as in Eq. (3.12)
        Set rotational direction $\rho_{\mathrm{wo}\to\mathrm{wp},i}$ and $\rho_{\mathrm{wp}\to\mathrm{po},i}$ as in Eq. (3.16)
    end
end
return $\mathbf{p}_c$, ρ, $\mathbf{p}_{wo}$, $\mathbf{p}_{po}$

Algorithm 3.2: 2D Dubins Path

Input: $\mathbf{p}_\mathrm{c}$, ρ, $\mathbf{p}_\mathrm{wo}$, $\mathbf{p}_\mathrm{po}$, $\mathbf{p}_\mathrm{wp}$, R
Result: $\mathcal{P}$
for $i \in [1, n]$ **do**
  if $\mathbf{p}_{wo,i} \neq \mathbf{p}_{wp,i}$ **then** `// circular arc from wheel-over point to waypoint`
    Set start angle α_0 and (signed) central angle $\Delta\alpha$ as in Eq. (3.18)
    $\mathcal{P} \leftarrow \mathrm{CircularArc}\left(\mathbf{p}_{\mathrm{c},i}, R, \alpha_0, \Delta\alpha\right)$
  end
  if $\mathbf{p}_{wp,i} \neq \mathbf{p}_{po,i}$ **then** `// circular arc from waypoint to pull-out point`
    Set start angle α_0 and (signed) central angle $\Delta\alpha$ as in Eq. (3.19)
    $\mathcal{P} \leftarrow \mathrm{CircularArc}\left(\mathbf{p}_{\mathrm{c},i}, R, \alpha_0, \Delta\alpha\right)$
  end
  if $i \neq n$ **then** `// line segment from pull-out point to wheel-over point`
    $\mathcal{P} \leftarrow \mathrm{LineSegment}\left(\mathbf{p}_{\mathrm{po},i}, \mathbf{p}_{\mathrm{wo},i+1}\right)$
  end
end
return $\mathcal{P}$

3.4 2D Extended Dubins Path

The 2D Extended Dubins Path is a G^2 continuous path combined of subpaths of line segments, circular arcs, and Euler spirals. It is simply the 2D Dubins Path extended with Euler spirals at the entrance and exit of every circular arc. As the Euler spiral acts as a transition curve from the zero curvature of a line segment to the constant curvature of a circular arc, and vice versa, the curvature of the path is continuous. The premise of the method is to be as similar as possible to the 2D Dubins Path while keeping the curvature continuous.

3.4.1 Extended Dubins Path

The famous Dubins Path was in Fraichard and Scheuer [21] extended with Euler spirals to ensure curvature continuity with an upper bound on the curvature and the curvature rate.

In contrast to the Dubins Path and the Dubins Car model, the Extended Dubins Path is not a time-optimal curve for the Dubins Car model extended with curvature as an additional parameter [22]. However, when the upper bound on the curvature rate tends to infinity, the path converges to the time-optimal Dubins Path [21].

The path is constructed by combining line segments, circular arcs of maximum curvature, and Euler spirals of maximum curvature rate. For UAVs, the maximum curvature rate is equivalent to a maximum roll rate.

This thesis builds upon the method for an interpolating path between a sequence of waypoints in Dahl [9] but updates it with some extra logic for initial position and orientation and final orientation, and in addition shows how online numerical integration is avoided. In this thesis, the method is called 2D Extended Dubins Path to differentiate it from the Dubins Path between two poses extended with Euler spirals.

3.4.2 Fundamental Euler Spiral

The Euler spiral with an initial course angle of $\chi_0 = 0$, an initial curvature of $\kappa_0 = 0$ and a curvature change of $\Delta\kappa = \frac{1}{R}$ is in this thesis referred to as the fundamental Euler spiral. The fundamental Euler spiral is shown in Figure 3.7. It is a transition curve from a line segment with zero course angle to a right-turning circular arc with a radius of R. The properties of the fundamental Euler spiral will be computed once and later reused to compute all the Euler spirals needed in this method.

From Equation (2.26d), the fundamental Euler spiral is given as

$$\mathbf{p}_{\text{spiral}}(s) = \mathbf{p}_0 + w \begin{bmatrix} \int_0^{\frac{s}{w}} \cos\left(\frac{\tau^2}{2}\right) d\tau \\ \int_0^{\frac{s}{w}} \sin\left(\frac{\tau^2}{2}\right) d\tau \end{bmatrix}, \tag{3.20}$$

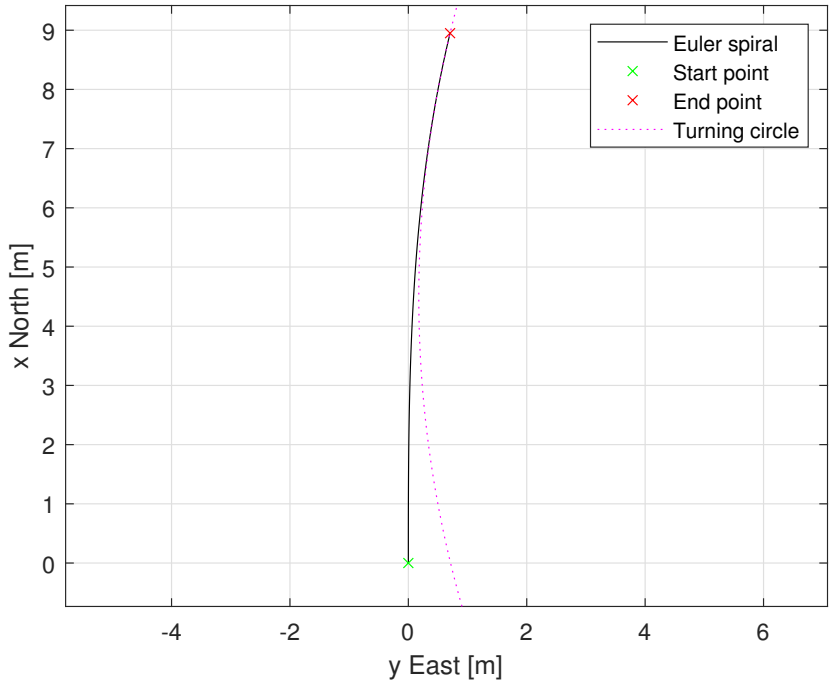


Figure 3.7: 2D Extended Dubins Path: Fundamental Euler spiral

where w is the scale factor of the fundamental Euler spiral.

As the length of an Euler spiral is $L = w^2|\Delta\kappa|$ from Equation (2.27), the scale factor of the fundamental Euler spiral is

$$w = \sqrt{L_{\text{spiral}} R}, \tag{3.21}$$

where L_{spiral} is the length of the fundamental Euler spiral.

The vector from the start point of the fundamental Euler spiral to the end point is given by the Fresnel integrals,

$$\mathbf{v}_{\text{spiral,s}\to\text{e}} = w \begin{bmatrix} \int_0^{\frac{L_{\text{spiral}}}{w}} \cos\left(\frac{\tau^2}{2}\right) d\tau \\ \int_0^{\frac{L_{\text{spiral}}}{w}} \sin\left(\frac{\tau^2}{2}\right) d\tau \end{bmatrix}. \tag{3.22}$$

From Equation (2.28), the change in course angle over the fundamental Euler spiral is

$$\Delta\chi_{\text{spiral}} = \text{sign}(\Delta\kappa)\frac{L_{\text{spiral}}}{2R}. \tag{3.23}$$

3.4.3 Length of Euler Spirals

The length of the Euler spirals depends on the change in roll angle from a line segment to a turn,

$$L = V_g \frac{|\Delta\phi|}{p_{\text{des}}}, \tag{3.24}$$

where V_g is the ground speed, $\Delta\phi$ is the change in roll angle and p_{des} is the desired roll rate. As the desired turn rate was set as the turn rate in a coordinated turn with the maximum roll angle in Section 3.3.2, the change in roll angle is $|\Delta\phi| = \phi_{\max}$. The length of the Euler spirals is therefore

$$L = V_g \frac{\phi_{\max}}{p_{\text{des}}}. \tag{3.25}$$

With an assumption of constant speed, all the Euler spirals are of equal length.

3.4.4 Euler Spirals and Fresnel Integrals

All the Euler spirals in this method are either transitions

- from a line segment to a right-turning circular arc, or
- from a line segment to a left-turning circular arc, or
- from a right-turning circular arc to a line segment, or
- from a left-turning circular arc to a line segment.

The initial course angle χ_0, the initial curvature κ_0, and the sign of the curvature change $\Delta\kappa$ are thus the only differences between the different Euler spirals. As the absolute value of the curvature change is equal to $|\Delta\kappa| = \frac{1}{R}$ in all the cases, all the Euler spirals is of the same length as the fundamental Euler spiral, L_{spiral}, and with the same scale factor w.

The original method in [9] solves the Fresnel integrals for each Euler spiral, but that is not beneficial in an embedded setting where computational efficiency is important. In this thesis, the Fresnel integrals only need to be solved once as a slightly different parameterization of the Euler spiral is used in combination with a rotation matrix and a reflection matrix.

The end point of the Euler spirals with arbitrary initial course angle χ_0, zero initial curvature $\kappa_0 = 0$ and curvature change $\Delta\kappa = \pm\frac{1}{R}$ are given by the Fresnel integrals of the fundamental Euler spiral,

$$\mathbf{p}_1 = \mathbf{p}_0 + \mathbf{R}_z(\chi_0) \begin{bmatrix} 1 & 0 \\ 0 & \text{sign}(\Delta\kappa) \end{bmatrix} \mathbf{v}_{\text{spiral,s}\to\text{e}}, \tag{3.26}$$

where $\mathbf{R}_z$ is a rotation matrix and the second matrix is either an identity matrix or a reflection matrix around the x-axis, depending on the sign of the curvature change.

For Euler spirals with non-zero initial curvature κ_0, this thesis uses a trick as it sets the end point of the Euler spirals and instead computes the start point,

$$\mathbf{p}_0 = \mathbf{p}_1 - \mathbf{R}_z(\chi_1)\begin{bmatrix} 1 & 0 \\ 0 & \text{sign}(\Delta\kappa) \end{bmatrix}\mathbf{v}_{\text{spiral,s}\to\text{e}} \tag{3.27}$$

where χ_1 is the course angle at the end point.

As is shown, when the start point of the Euler spirals with zero initial curvature is set, then the end point can be computed with the Fresnel integrals from Equation (3.22), and vice versa for the Euler spirals with non-zero initial curvature.

The Fresnel integrals do not have closed-form solutions and must therefore be solved numerically. As all the Euler spirals have the same length and scale factor, the Fresnel integrals are the same for all the Euler spirals and therefore only need to be solved once. The solution of the Fresnel integrals should be pre-computed offline so that the online complexity of the method is minimized.

3.4.5 Wheel-Over and Pull-Out Motions

The cornerstone of the method is to place Euler spirals at both the entrance and exit of the circular arcs at the waypoints to provide curvature continuity. The end of the exit Euler spiral at one waypoint is connected to the start of the Euler spiral at the next waypoint with a line segment. If the start and end points of the Euler spirals are set, then the circular arcs and line segments are implicitly defined.

The main part of the method for the 2D Dubins Path was to compute the wheel-over and pull-out points. As the curvature is now limited to being continuous, the wheel-over and pull-out motions are no longer instantaneous. The main part of this method is thus to compute the wheel-over-start points $\mathbf{p}_{\text{wo-s}}$, the wheel-over-end points $\mathbf{p}_{\text{wo-e}}$, the pull-out-start points $\mathbf{p}_{\text{po-s}}$ and the pull-out-end points $\mathbf{p}_{\text{po-e}}$. A wheel-over-start point is the start point of an Euler spiral at the entrance of a circular arc and a wheel-over-end point is the end point of such a spiral. Likewise, a pull-out-start point is the start point of an Euler spiral at the exit of a circular arc and a pull-out-end point is the end point of such a spiral.

The method uses the procedure from 2D Dubins Path for finding the wheel-over and pull-out points as an intermediate step. The makeshift wheel-over and pull-out points are computed for a circle larger than the turning circle. See Figure 3.8 for an illustration of the outer circle, the turning circle, and an Euler spiral. The radius of the outer circle is

$$R_{\text{spiral}} = R\cos\left(\Delta\chi_{\text{spiral}}\right) + v_{\text{spiral,s}\to\text{e},y}, \tag{3.28}$$

where $\Delta\chi_{\text{spiral}}$ is the change in course angle over the Euler spirals from Equation (3.23) and $v_{\text{spiral,s}\to\text{e},y}$ is the second Fresnel integral from Equation (3.22) [9].

First, the orientations at the waypoints and the circle centers are computed as in Section 3.3.3. Then, a makeshift pull-out point $\mathbf{p}_{\text{po},i}$ and a makeshift wheel-over point $\mathbf{p}_{\text{wo},i+1}$ is computed for the outer circles, with the same procedure as in Section 3.3.4, Section 3.3.5 and Section 3.3.6.

The course angle at the makeshift pull-out point and at the makeshift wheel-over point is

$$\chi_{\text{po},i} = \text{atan2}\left(v_{\text{c}\to\text{po},i,y}, v_{\text{c}\to\text{po},i,x}\right) + \rho_i \frac{\pi}{2}, \tag{3.29a}$$

$$\chi_{\text{wo},i+1} = \chi_{\text{po},i}. \tag{3.29b}$$

The wheel-over-start point must be placed an offset distance before the makeshift wheel-over point, as seen in Figure 3.8,

$$L_{\text{offset}} = v_{\text{spiral,s}\to\text{e},x} - R\sin\left(\Delta\chi_{\text{spiral}}\right), \tag{3.30}$$

where $\Delta\chi_{\text{spiral}}$ is the change in course angle over the Euler spiral from Equation (3.23) and $v_{\text{spiral,s}\to\text{e},y}$ is the first Fresnel integral from Equation (3.22) [9]. The same applies for the pull-out-end point, except that it is placed an offset distance after the makeshift pull-out point.

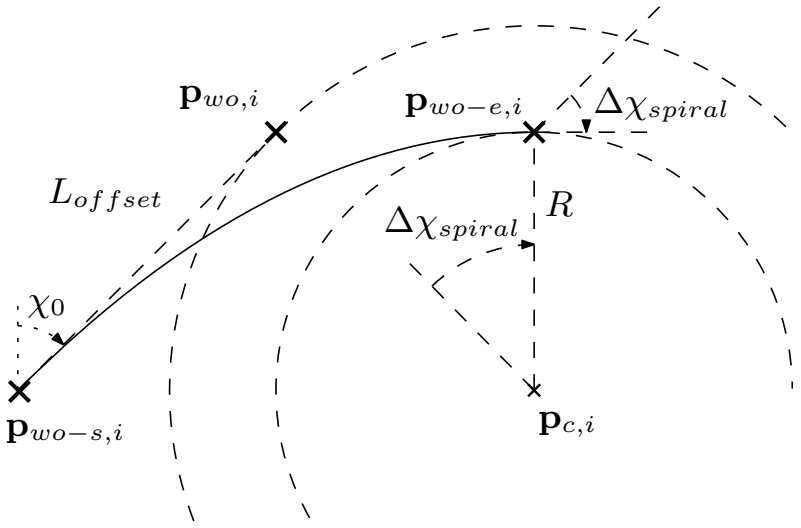


Figure 3.8: 2D Extended Dubins Path: Outer circle, turning circle, and an Euler spiral

The Euler spiral corresponding to the pull-out motion has a non-zero initial curvature, so the pull-out-start point is computed as in Equation (3.27) and the pull-out-end point is offset from the makeshift pull-out point,

$$\mathbf{p}_{\text{po-e},i} = \mathbf{p}_{\text{wo},i} + L_{\text{offset}} \begin{bmatrix} \cos\left(\chi_{\text{po},i}\right) \\ \sin\left(\chi_{\text{po},i}\right) \end{bmatrix}, \tag{3.31a}$$

$$\mathbf{p}_{\text{po-s},i} = \mathbf{p}_{\text{po-e},i} - \mathbf{R}_z\left(\chi_{\text{po},i}\right) \begin{bmatrix} 1 & 0 \\ 0 & -\rho_i \end{bmatrix} \mathbf{v}_{\text{spiral,s}\to\text{e}}. \tag{3.31b}$$

The Euler spiral corresponding to the wheel-over motion has zero initial curvature, so the wheel-over-end point is computed as in Equation (3.26) and the wheel-over-start point is offset from the makeshift wheel-over point,

$$\mathbf{p}_{\text{wo-s},i+1} = \mathbf{p}_{\text{wo},i+1} - L_{\text{offset}} \begin{bmatrix} \cos\left(\chi_{\text{wo},i+1}\right) \\ \sin\left(\chi_{\text{wo},i+1}\right) \end{bmatrix}, \tag{3.32a}$$

$$\mathbf{p}_{\text{wo-e},i+1} = \mathbf{p}_{\text{wo-s},i+1} + \mathbf{R}_z\left(\chi_{\text{wo},i+1}\right) \begin{bmatrix} 1 & 0 \\ 0 & \rho_{i+1} \end{bmatrix} \mathbf{v}_{\text{spiral,s}\to\text{e}}. \tag{3.32b}$$

3.4.6 First and Last Waypoints

As mentioned, the method uses the procedure from 2D Dubins Path for computing the orientations at the waypoints and placing the circle centers. In both the method for the 2D Dubins Path and in this method, it is assumed that the curvature at the first waypoint (initial position) and the curvature at the final waypoint is zero. In other words, it is assumed that neither the initial position nor the final position is in a turn.

For the 2D Dubins Path, both the first and the last waypoint lies on the associated turning circles as the transition from the zero curvature to a constant curvature is instantaneous. As the transition is no longer instantaneous for the 2D Extended Dubins Path, the first and last waypoint can no longer lie on their associated turning circles. The first and last circle centers must therefore be treated differently and are not computed with the procedure from Section 3.3.3.

For the first waypoint, the circle center is placed as if the wheel-over-end point is the waypoint, and the same for the pull-out-start point for the last waypoint,

$$\mathbf{p}_{\text{c},1} = \mathbf{p}_{\text{wo-e},1} + R \begin{bmatrix} \cos\left(\chi_0 + \rho_1 \Delta\chi_{\text{spiral}} + \frac{\pi}{2}\right) \\ \sin\left(\chi_0 + \rho_1 \Delta\chi_{\text{spiral}} + \frac{\pi}{2}\right) \end{bmatrix}, \tag{3.33a}$$

$$\mathbf{p}_{\text{c},n} = \mathbf{p}_{\text{po-s},n} + R \begin{bmatrix} \cos\left(\chi_n - \rho_n \Delta\chi_{\text{spiral}} + \frac{\pi}{2}\right) \\ \sin\left(\chi_n - \rho_n \Delta\chi_{\text{spiral}} + \frac{\pi}{2}\right) \end{bmatrix}, \tag{3.33b}$$

since these points lie on the turning circles.

The first wheel-over-start point is chosen as the first waypoint and the wheel-over-end point is computed based on it,

$$\mathbf{p}_{\text{wo-s},1} = \mathbf{p}_{\text{wp},1}, \tag{3.34a}$$

$$\mathbf{p}_{\text{wo-e},1} = \mathbf{p}_{\text{wo-s},1} + \mathbf{R}_z\left(\chi_{\text{wo},1}\right) \begin{bmatrix} 1 & 0 \\ 0 & \rho_1 \end{bmatrix} \mathbf{v}_{\text{spiral,s}\to\text{e}}. \tag{3.34b}$$

In the same manner, the last pull-out-end point is chosen as the last way-

point and the pull-out-start point is computed based on it,

$$\mathbf{p}_{\text{po-e},n} = \mathbf{p}_{\text{wp},n}, \tag{3.35a}$$

$$\mathbf{p}_{\text{po-s},n} = \mathbf{p}_{\text{po-e},n} - \mathbf{R}_z\left(\chi_{\text{po},n}\right)\begin{bmatrix} 1 & 0 \\ 0 & -\rho_n \end{bmatrix}\mathbf{v}_{\text{spiral,s}\to\text{e}}. \tag{3.35b}$$

3.4.7 Euler Spirals and Circular Arcs

For completeness, it is shown how to calculate the parameters of the Euler spirals and the circular arcs.

For the wheel-over Euler spiral, the start position $\mathbf{p}_{\text{wo-s},i}$, the initial course angle $\chi_{\text{wo},i}$, and the scale factor w are already computed and the initial curvature $\kappa_0 = 0$ is given. The (signed) change in curvature is

$$\Delta\kappa = \frac{\rho_i}{R}. \tag{3.36}$$

For the pull-out Euler spiral, the start position $\mathbf{p}_{\text{po-s},i}$ and the scale factor w are already computed. The initial course angle, the initial curvature and the (signed) change in curvature are

$$\chi_0 = \chi_{\text{po},i} - \Delta\chi_{\text{spiral}}, \tag{3.37a}$$

$$\kappa_0 = \frac{\rho_i}{R}, \tag{3.37b}$$

$$\Delta\kappa = -\frac{\rho_i}{R}. \tag{3.37c}$$

In addition, it is useful to save the wheel-over-end point $\mathbf{p}_{\text{wo-e},i}$ and pull-out-end point $\mathbf{p}_{\text{po-e},i}$ to avoid numerically solving the Fresnel integrals again in the later spline approximation.

The parameters for the circular arcs are computed in the same manner as in Section 3.3.9, but for wheel-over-end and pull-out-start points instead. For the circular arc from the wheel-over-end point to the waypoint,

$$\mathbf{v}_{\text{c}\to\text{wo-e},i} = \mathbf{p}_{\text{wo-e},i} - \mathbf{p}_{\text{c},i}, \tag{3.38a}$$

$$\alpha_0 = \operatorname{atan2}(v_{\text{c}\to\text{wo-e},i,y}, v_{\text{c}\to\text{wo-e},i,x}), \tag{3.38b}$$

$$\Delta\alpha = \operatorname{mod}\left(\operatorname{atan2}(-v_{\text{wp}\to\text{c},i,y}, -v_{\text{wp}\to\text{c},i,x}) - \alpha_{0,i}, \rho_i 2\pi\right), \tag{3.38c}$$

where $\operatorname{mod}(\cdot)$ ensures that the central angle has the correct sign.

Likewise, for the circular arc from the waypoint to the pull-out-start point,

$$\mathbf{v}_{\text{c}\to\text{po-s},i} = \mathbf{p}_{\text{po-s},i} - \mathbf{p}_{\text{c},i}, \tag{3.39a}$$

$$\alpha_0 = \operatorname{atan2}(-v_{\text{wp}\to\text{c},i,y}, -v_{\text{wp}\to\text{c},i,x}), \tag{3.39b}$$

$$\Delta\alpha = \operatorname{mod}\left(\operatorname{atan2}(v_{\text{c}\to\text{po-s},i,y}, v_{\text{c}\to\text{po-s},i,x}) - \alpha_{0,i}, \rho_i 2\pi\right). \tag{3.39c}$$

The parameters are computed differently for the circular arcs at the first and last waypoint as those waypoints do not lie on the turning circle. The circular

arc at those waypoints goes from the wheel-over-end point to the pull-out-start point, and the start angle and central angle is computed as

$$\alpha_0 = \text{atan2}(v_{\text{c}\to\text{wo-e},i,y}, v_{\text{c}\to\text{wo-e},i,x}), \tag{3.40a}$$

$$\Delta\alpha = \text{mod}\left(\text{atan2}(v_{\text{c}\to\text{po-s},i,y}, v_{\text{c}\to\text{po-s},i,x}) - \alpha_{0,i}, \rho_i 2\pi\right). \tag{3.40b}$$

Algorithm 3.3: 2D Extended Dubins Path Wheel-Over and Pull-Out Motions

Input: χ_0, χ_n, R, L_{spiral}, $\mathbf{p}_{\text{wp},i} \quad i = 1, \ldots, n$

Result: $\mathbf{p}_{\text{c}}$, ρ, w, χ_{wo}, χ_{po}, $\mathbf{p}_{\text{wo-s}}$, $\mathbf{p}_{\text{wo-e}}$, $\mathbf{p}_{\text{po-s}}$, $\mathbf{p}_{\text{po-e}}$

Set scale factor w as in Eq. (3.21) and $\mathbf{v}_{\text{spiral,s}\to\text{e}}$ as in Eq. (3.22)
Set radius of outer circle R_{spiral} as in Eq. (3.28) and offset value L_{offset} as in Eq. (3.30)
Set initial waypoint vector $\mathbf{v}_{\text{wp},1}$ and rotational direction ρ_1 as in Eq. (3.7)
Set final waypoint vector $\mathbf{v}_{\text{wp},n}$ and rotational direction ρ_n as in Eq. (3.7)
Set initial wheel-over-start point $\mathbf{p}_{\text{wo-s},1}$ and wheel-over-end point $\mathbf{p}_{\text{wo-e},1}$ as in Eq. (3.34)
Set final pull-out-start point $\mathbf{p}_{\text{po-s},n}$ and pull-out-end point $\mathbf{p}_{\text{po-e},n}$ as in Eq. (3.35)
Set first circle center $\mathbf{p}_{\text{c},1}$ and final circle center $\mathbf{p}_{\text{c},n}$ as in Eq. (3.33)
for $i \in [2, n-1]$ **do**
    Set waypoint vector $\mathbf{v}_{\text{wp},i}$ and rotational direction ρ_i as in Eq. (3.6)
    Set circle center $\mathbf{p}_{\text{c},i}$ as in Eq. (3.8)
end
for $i \in [1, n-1]$ **do**
    if $\rho_i \neq \rho_{i+1}$ **then** `// different consecutive rotations`
        Set makeshift pull-out vector $\mathbf{v}_{\text{c}\to\text{po},i}$ and wheel-over vector $\mathbf{v}_{\text{c}\to\text{wo},i+1}$ as in Eq. (3.10), but with R_{spiral} instead of R
    else `// equivalent consecutive rotations`
        Set makeshift pull-out vector $\mathbf{v}_{\text{c}\to\text{po},i}$ and wheel-over vector $\mathbf{v}_{\text{c}\to\text{wo},i+1}$ as in Eq. (3.11)
    end
    Set makeshift pull-out point $\mathbf{p}_{\text{po},i}$ and wheel-over point $\mathbf{p}_{\text{wo},i+1}$ as in Eq. (3.12), but with R_{spiral} instead of R
    Set pull-out course angle $\chi_{\text{po},i}$ and wheel-over course angle $\chi_{\text{wo},i+1}$ as in Eq. (3.29)
    Set pull-out-start point $\mathbf{p}_{\text{po-s},i}$ and pull-out-end point $\mathbf{p}_{\text{po-e},i}$ as in Eq. (3.31)
    Set wheel-over-start point $\mathbf{p}_{\text{wo-s},i+1}$ and wheel-over-end point $\mathbf{p}_{\text{wo-e},i+1}$ as in Eq. (3.32)
end
return $\mathbf{p}_c$, ρ, w, χ_{wo}, χ_{po}, $\mathbf{p}_{wo\text{-}s}$, $\mathbf{p}_{wo\text{-}e}$, $\mathbf{p}_{po\text{-}s}$, $\mathbf{p}_{po\text{-}e}$

Algorithm 3.4: 2D Extended Dubins Path

Input: $\mathbf{p}_{\text{c}}$, ρ, w, χ_{wo}, χ_{po}, $\mathbf{p}_{\text{wo-s}}$, $\mathbf{p}_{\text{wo-e}}$, $\mathbf{p}_{\text{po-s}}$, $\mathbf{p}_{\text{po-e}}$, $\mathbf{p}_{\text{wp}}$, R
Result: $\mathcal{P}$

for $i \in [1, n]$ **do**
  Set (signed) curvature change $\Delta\kappa$ as in Eq. (3.36)
  $\mathcal{P} \leftarrow \text{EulerSpiral}\left(\mathbf{p}_{\text{wo-s},i}, \chi_{\text{wo},i}, 0, \Delta\kappa, w, \mathbf{p}_{\text{wo-e},i}\right)$
  if $i = 1$ && $i = n$ **then** `// circular arc: wheel-over-end -> pull-out-start`
    Set start angle α_0 and (signed) central angle $\Delta\alpha$ as in Eq. (3.40)
    $\mathcal{P} \leftarrow \text{CircularArc}\left(\mathbf{p}_{\text{c},i}, R, \alpha_0, \Delta\alpha\right)$
  else `// circular arcs: wheel-over-end -> waypoint and waypoint -> pull-out-start`
    Set start angle α_0 and (signed) central angle $\Delta\alpha$ as in Eq. (3.38)
    $\mathcal{P} \leftarrow \text{CircularArc}\left(\mathbf{p}_{\text{c},i}, R, \alpha_0, \Delta\alpha\right)$
    Set start angle α_0 and (signed) central angle $\Delta\alpha$ as in Eq. (3.39)
    $\mathcal{P} \leftarrow \text{CircularArc}\left(\mathbf{p}_{\text{c},i}, R, \alpha_0, \Delta\alpha\right)$
  end
  Set initial course χ_0, initial curvature κ_0 and (signed) curvature change $\Delta\kappa$ as in Eq. (3.37)
  $\mathcal{P} \leftarrow \text{EulerSpiral}\left(\mathbf{p}_{\text{po-s},i}, \chi_0, \kappa_0, \Delta\kappa, w, \mathbf{p}_{\text{po-e},i}\right)$
  if $i \neq n$ **then** `// line segment from pull-out-end point to wheel-over-start point`
    $\mathcal{P} \leftarrow \text{LineSegment}\left(\mathbf{p}_{\text{po-e},i}, \mathbf{p}_{\text{wo-s},i+1}\right)$
  end
end
return $\mathcal{P}$

Chapter 4

3D Path Generation

This chapter presents a method for interpolating path generation from a sequence of waypoints in 3D. The 3D Extended Dubins Path reuses the 2D Extended Dubins Path and the 2D Dubins Path from Chapter 3 to generate a three-dimensional path. This thesis combines the principles of Vana *et al.* [23] and Owen *et al.* [24] in a way that is computationally efficient and therefore usable for online path generation in an embedded setting. A concise algorithmic description of the method is given in Algorithm 4.1.

4.1 Summary

The 3D Extended Dubins Path is an interpolating path between waypoints in 3D. The path is generated from two decoupled paths: one in the horizontal plane and one in the vertical plane. The 2D Extended Dubins Path is used in the horizontal plane to provide a continuous course angle rate, while the 2D Dubins Path is used in the vertical plane to provide a continuous flight path angle. The horizontal path is G^2 continuous and the vertical path is G^1 continuous. As to not exceed the constraint on the flight path angle, the length of the horizontal path is increased by adding complete circle turns where needed. This corresponds to helix spirals in the three-dimensional path. The expansion of the horizontal path is an iterative procedure.

4.2 Dubins Airplane Path

The Dubins Airplane Path is an extension of the famous Dubins Path with an altitude component. The path connects two poses in 3D, with a constraint on the maximum roll angle and a constraint on the maximum flight path angle. It is the time-optimal curve for the kinematic model known as the Dubins Airplane [25]. The constraint on the roll angle limits the minimum turning radius, while the constraint on the pitch angle limits the maximum climbing rate. The simple kinematic model has both the roll angle and the pitch angle as inputs. In other

words, the kinematic model assumes that both the roll angle and the pitch angle can be changed instantly.

The Dubins Path is the basis of the Dubins Airplane Path, but it is modified to satisfy the climbing rate constraint. As the path is three-dimensional, it has a horizontal projection and a vertical projection. The horizontal projection of the path consists of line segments and circular arcs, same as the Dubins Path, while the vertical projection of the path only consists of line segments.

The Dubins Airplane Path modifies the Dubins Path in three different ways: low-altitude, medium-altitude, and high-altitude. The modification depends on the relative altitude difference between the start pose and the end pose.

The simplest case is the low-altitude case. In the low-altitude case, the altitude difference between the start pose and the end pose is relatively small compared to the length of the Dubins Path. As the altitude difference is relatively small, the altitude gain/loss can be achieved by simply flying the Dubins Path with a non-zero flight path angle. The flight path angle is computed as

$$\gamma = \operatorname{atan}\left(\frac{h_1 - h_0}{L_{\text{Dubins}}}\right), \tag{4.1}$$

where h_1 and h_0 are the altitude of the start pose and the end pose, respectively, and L_{Dubins} is the length of the Dubins Path between the horizontal projection of the poses.

The computed flight path angle can exceed the maximum allowed flight path angle when the altitude difference between the start pose and the end pose is too large relative to the horizontal distance. A longer horizontal path length will make the flight path angle smaller as the altitude gain/loss happens over a longer distance. In the medium-altitude and high-altitude cases, the horizontal projection of the path is expanded to be just long enough for the altitude gain/loss to be achievable by flying with a maximum flight path angle. Owen *et al.* [24] distinguishes between two different ways of expanding the horizontal path.

In the high-altitude case, the horizontal path is expanded by adding a certain number of complete turns at the beginning or at the end of the horizontal path. In order to have most of the path at a high altitude, the turns are added at the start of the path if the end altitude is higher than the start altitude, or at the end of the path if the end altitude is lower than the start altitude. First, the number of turns is computed based on the minimum turning radius, and then a root-finding algorithm (bisection) is used to find the turning radius which gives a maximum flight path angle.

In the medium-altitude case, the altitude gain/loss can not be achieved by flying the Dubins Path with maximum flight path angle, but expanding the horizontal path with a complete turn is not necessary either. Instead, the horizontal path is expanded to be just long enough by adding a circular arc of a certain length with minimum turning radius from the start pose. A Dubins Path is computed from the end of the circular arc to the end pose. The length

of the added circular arc is found with a bisection algorithm such that it is just long enough for the altitude difference to be achievable with a maximum flight path angle.

The resulting path is time-optimal for the Dubins Airplane. The horizontal projection of the path is either the Dubins Path or an expanded version of the Dubins Path which is just long enough to make the altitude difference achievable by flying the path with a maximum flight path angle.

The assumption of the ability to instantly change roll angle and pitch angle makes the Dubins Airplane a simplistic model. As the vertical path only consists of line segments, it is G^0 continuous by the definition in Section 2.5. The flight path angle is therefore discontinuous for the Dubins Airplane Path. As the horizontal path is a version of the Dubins Path, it has the same continuity issues as the Dubins Path. The horizontal path is G^1 continuous, which entails that the course angle is continuous, but the course angle rate is discontinuous. The roll angle, which is a function of the course angle rate for UAVs that roll-to-turn, experiences step changes for the Dubins Airplane Path. The Dubins Airplane path is therefore not directly flyable by aircraft. Summarized, the reference roll angle and the reference pitch angle will be discontinuous with the Dubins Airplane Path.

In contrast to the Dubins Path, the Dubins Airplane Path also uses an iterative method in the generation of the path, as the bisection method is an iterative root-finding method. The computational complexity of the Dubins Airplane Path is therefore a lot higher than the simple Dubins Path.

4.3 Decoupled 3D Dubins Path

A three-dimensional path can be seen as a combination of two separate paths: a horizontal path and a vertical path. The horizontal path lies in the $x-y$ plane where x is the north coordinate and y is the east coordinate. The vertical path lies in the $h-s_h$ plane where h is the altitude and s_h is the horizontal distance. As the horizontal distance is one of the coordinates of the vertical plane, the vertical path is directly affected by the horizontal path.

One of the main drawbacks of the Dubins Airplane Path is that the flight path angle is discontinuous as the vertical path is only G^0 continuous. If circular arcs are used as transitions between the line segments in the vertical plane, then the vertical path would be G^1 continuous. As discussed earlier in this thesis, the Dubins Path is a simple and G^1 continuous path between two poses. A continuous flight path angle can therefore be achieved by also generating a Dubins Path in the vertical plane. Vana *et al.* [23] proposes a method where a Dubins Path is generated in both the horizontal plane and the vertical plane. The resulting three-dimensional path is a combination of the two separate paths.

As with the Dubins Airplane Path, the length of the horizontal path must still be expanded in some cases to satisfy the climbing rate constraint. Vana *et*

al. [23] expands the horizontal path by using a larger turning radius for the Dubins Path in the horizontal plane, as opposed to the extra circular arcs in the Dubins Airplane Path. First, a feasible solution is found by iteratively doubling the horizontal turning radius, and then local hill-climbing optimization is performed to minimize the length of the path. The final result is a horizontal Dubins Path and a vertical Dubins Path,

$$\mathcal{P}^{3D} = \{\mathcal{P}^{x-y}_{\text{Dubins}}, \mathcal{P}^{h-s_h}_{\text{Dubins}}\}. \tag{4.2}$$

4.4 3D Extended Dubins Path

This thesis combines the concepts of the Dubins Airplane Path and the decoupled 3D Dubins Path to generate an interpolating path between three-dimensional waypoints, with focus on computational efficiency. The 3D Extended Dubins Path uses the 2D Extended Dubins Path as the basis for the horizontal path and the 2D Dubins Path as the vertical path. Both of these methods are described in detail in Chapter 3. The horizontal path is G^2 continuous and the vertical path is G^1 continuous. The premise of the method is to generate a path with a continuous roll angle and a continuous flight path angle, with a low computational cost. The method itself is quite simple since it mostly reuses the two horizontal path generation methods.

4.4.1 Decoupled 3D Path

Inspired by Vana *et al.* [23], this thesis simplifies the generation of a three-dimensional path by combining a horizontal path and a vertical path. The 2D Extended Dubins Path is used in the horizontal plane, in contrast to Vana *et al.* [23] which uses a Dubins Path,

$$\mathcal{P}^{3D} = \{\mathcal{P}^{x-y}_{\text{2D Extended Dubins}}, \mathcal{P}^{h-s_h}_{\text{2D Dubins}}\}. \tag{4.3}$$

First, the horizontal path in the $x-y$ plane is computed using the horizontal projection of the 3D waypoints. Then, the vertical path in the $h-s_h$ plane is computed using the vertical projection of the 3D waypoints. Note that the vertical projection of the waypoints depends on the horizontal path as the second coordinate of the vertical plane is the horizontal path length.

The 2D Extended Dubins Path is a more complex method, but it generates a path with continuous curvature. The continuous horizontal curvature gives a continuous course angle rate which in turn gives a continuous roll angle. In the vertical plane, continuity of the curvature is not necessary as a continuous vertical course angle gives a continuous flight path angle. The 2D Dubins Path is therefore sufficient in the vertical plane.

The radius of the turning circles in the vertical plane is chosen as

$$R_{hs} = \frac{V_g}{q_{\text{des}}} \tag{4.4}$$

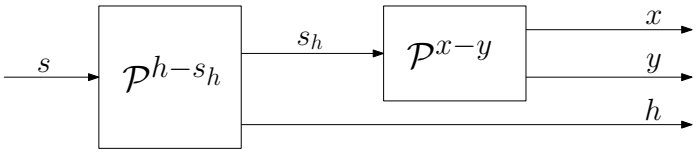


Figure 4.1: Combination of two two-dimensional paths into a three-dimensional path

where V_g is the ground speed and q_{des} is the desired maximum pitch rate.

When the paths are constructed with the arc length parameterized curves described in Chapter 2, the combination of the horizontal path and the vertical path into a three-dimensional path is straightforward. The input path parameter of the vertical path is the distance along the path, $s \in [0, L_{3D}]$, and the input parameter of the horizontal path is the horizontal distance along the path, $s_h \in [0, L_h]$. As the horizontal distance along the path is the second coordinate of the vertical path, the final three-dimensional path is evaluated in the following way,

$$\begin{bmatrix} h \\ s_h \end{bmatrix} = \mathcal{P}^{H-S_h}(s), \tag{4.5}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathcal{P}^{X-Y}(s_h), \tag{4.6}$$

where the second coordinate of the vertical path is the input path parameter to the horizontal path and the result is the three-dimensional coordinate $[x, y, h]$.

The length of the final three-dimensional path is equal to the length of the two-dimensional path in the vertical plane,

$$\mathcal{L}(\mathcal{P}^{3D}) = \mathcal{L}(\mathcal{P}^{h-s_h}). \tag{4.7}$$

4.4.2 Flight Path Angle

The 2D Dubins Path method from Chapter 3 is reused in the vertical plane in this method. As it is derived in the $x-y$ plane, some transformation is needed when it is used in the $h-s_h$ plane. The course angle in the $x-y$ plane is measured positively from the x-axis in a clockwise direction, while the flight path angle in the $h-s_h$ plane is measured positively from the s_h-axis in a counter-clockwise direction. The transformation between the vertical course angle, given by the path parameterizations due to reuse of the horizontal path generation, and the flight path angle is

$$\gamma = \frac{\pi}{2} - \chi_v, \tag{4.8}$$

where χ_v is the course angle in the vertical plane.

The easiest way to check if the 2D Dubins Path in the vertical plane violates the maximum flight path angle is to check the flight path angle of the line

segments. The flight path angle of circular arcs starts with the same flight path angle as the previous line segment and ends with the same flight path angle as the next line segment. If none of the line segments violates the maximum flight path angle constraint, then the vertical 2D Dubins Path satisfies the constraint.

4.4.3 Expanding The Horizontal Path

If one of the line segments violates the flight path angle constraint, then the horizontal path must be expanded. This thesis simplifies the expansion of the horizontal path to a simplified version of the high-altitude case in the Dubins Airplane Path. Both Vana *et al.* [23] and Owen *et al.* [24] change the turning radius of the horizontal path with an optimization technique to find a horizontal path just long enough. This is a problem when the 2D Extended Dubins Path is used as the horizontal path. The Euler Spiral, which is used as a transition curve, depends on the turning radius of the circular arc it transitions to/from. As the Euler Spiral does not have a closed-form solution, the Fresnel integrals must be solved numerically. When each circular arc has a different turning radius, each Euler Spiral will be different and offline computation of the Fresnel integrals is no longer an option. This thesis chooses therefore to only expand the horizontal path with extra complete turns with the same minimum turning radius as all other circular arcs. An extra complete circle turn is added to the waypoint before the line segment which breaks the flight path angle constraint. This increases the distance between the two waypoints significantly which most likely makes the flight path angle constraint satisfied. This should be done iteratively to handle the cases with large altitude difference where several complete circle turns is necessary.

Algorithm 4.1: 3D Extended Dubins Path

Input: χ_0, χ_n, R_{xy}, L_{spiral}, R_{hs}, $\gamma_{\max}$, $\mathbf{p}_{\text{wp}}$
Result: $\mathcal{P}^{X-Y}$, $\mathcal{P}^{H-S_h}$

$\mathcal{P}^{X-Y} = \text{ExtendedDubins2D}(\chi_0, \chi_n, R_{\text{xy}}, L_{\text{spiral}}, \mathbf{p}_{\text{wp}}^{\text{xy}})$
Compute waypoints in the vertical plane $\mathbf{p}_{\text{wp}}^{\text{hs}}$
$\mathcal{P}^{H-S_h} = \text{Dubins2D}(0, 0, R_{\text{hs}}, \mathbf{p}_{\text{wp}}^{\text{hs}})$
Compute flight path angle γ_i of all line segments as in Eq. (4.8)
while $|\gamma_i| > \gamma_{max}$ **do**
    $\mathcal{P}^{X-Y} = \mathcal{P}_{1:i}^{X-Y} + \text{CircularArc}(\mathbf{p}_{c,i}, R, \alpha_{0,i}2\pi) + \mathcal{P}_{i:n}^{X-Y}$
    Compute waypoints in the vertical plane $\mathbf{p}_{\text{wp}}^{\text{hs}}$
    $\mathcal{P}^{H-S_h} = \text{Dubins2D}(0, 0, R_{\text{hs}}, \mathbf{p}_{\text{wp}}^{\text{hs}})$
    Compute flight path angle γ_i of all line segments as in Eq. (4.8)
end
return $\{\mathcal{P}^{X-Y}, \mathcal{P}^{H-S_h}\}$

Chapter 5

Spline Approximation

This chapter presents an approximation with algebraic splines of the three-dimensional path generated in Chapter 4. The path is used as a template and the approximation should accurately approximate the key properties of each subpath. The approximation is done with cubic algebraic splines. The algebraic splines give a unified path representation of the path, which simplifies guidance as it does not have to distinguish between the different types of subpaths.

5.1 Algebraic Spline

An algebraic spline is a polynomial parametric curve [26]. This thesis uses two-dimensional cubic splines,

$$\mathbf{s}(l) = \begin{bmatrix} a_{x,0} + a_{x,1}l + a_{x,2}l^2 + a_{x,3}l^3 \\ a_{y,0} + a_{y,1}l + a_{y,2}l^2 + a_{y,3}l^3 \end{bmatrix}, \tag{5.1}$$

where $\mathbf{a}_x$ and $\mathbf{a}_y$ are the spline coefficients for the two coordinates and l is the parameter of the spline. The splines are used both in the horizontal $x-y$ plane and in the vertical $h-s_h$ plane. In the $h-s_h$ plane, the $\mathbf{a}_x$ coefficients are used for the h coordinate and the $\mathbf{a}_y$ coefficients for the s_h coordinate.

Splines are often used as a form of interpolation between data points, where each spline is a piecewise polynomial [27]. This thesis uses splines slightly differently. The subpaths of the three-dimensional path are templates for the splines. So a spline should not only interpolate the start and end points of a subpath but also approximate the properties of the subpath. For Euler spirals, this means that the spline should approximate the linear curvature profile. Each subpath is approximated by an algebraic cubic spline and the approximation of the complete path is piecewise-defined by the splines.

5.2 Unified Path Representation

The main reason for approximating the path generated in Chapter 4 with splines is to get a unified representation of the path. The path consists of three different types of subpaths: line segments, circular arcs, and Euler spirals. Each subpath has a different parameterization and the guidance algorithm must therefore take into account what kind of subpath it is tracking. As an example, Beard and McLain [2] provides a guidance algorithm for line segments and another guidance algorithm for circular arcs. The Euler spirals would complicate the guidance algorithm further.

The guidance module also needs some kind of switching logic to keep track of the current subpath type and what type the next subpath can be. See Beard and McLain [2] or Owen *et al.* [24] for examples where the switching logic is realized as a state machine. With a spline-based path, a homogeneous guidance algorithm is sufficient and the switching logic is significantly simplified. As each spline only differs in the value of its coefficients, the splines provide a unified representation of the path. This reduces the complexity of the guidance module.

The guidance module usually employs some kind of root finding method to find the closest point on the path,

$$l^* = \operatorname{argmin} \|\mathbf{S}(l) - \mathbf{p}\| \tag{5.2}$$

where $\mathbf{p}$ is the position of the aircraft. The minimization problem can be solved with a combination of a quadratic minimization algorithm and the Newton method [15].

In an embedded setting where the computational load is a limiting factor, an approximation of the Euler spiral is necessary for it to be useful. The Fresnel integrals, which do not have a closed-form solution and therefore must be solved numerically, are part of the parameterization of the Euler spiral. Each evaluation of a point along an Euler spiral would involve solving the integrals numerically. During the above-mentioned root finding, the path is evaluated several times per time step in the guidance algorithm. Numerically solving the integrals several times per time step is not feasible in an embedded setting with limited computational power. The Euler spiral must therefore be approximated by splines to be useful.

5.3 Arc Length Parameterization

An arc-length parameterization of the splines is beneficial in applications with limited computational power due to the linear relationship between the physical distance traveled and the path parameter [15]. With an arc length parameterization, a good initial guess for the root-finding problem can be made based on the speed of the aircraft and the last solution. For the spline to be

properly parameterized by arc length, the length of each spline must be computed. As the length of a spline segment does not have a closed-form solution, this requires a numerical computation of an integral for each spline,

$$L = \int_0^1 \mathbf{S}'(l)dl, \tag{5.3}$$

if the splines are first created with a unit domain parameterization. Numerically solving an integral for each spline is not practical in an embedded setting with limited computation power. So an arc length parameterization is beneficial in applications with limited computational power, but creating an arc length parameterization is not practical in applications with limited computational power. However, the splines in this thesis uses approximates the known subpaths. The length of the subpaths is known and can be used as estimates of the arc length of the splines. If the spline approximation of the subpath is sufficiently accurate, then the spline will be close to arc length parameterized. Using this estimate of the arc length to create parameterizations of the splines gives pseudo-parameterizations by arc length. The spline parameter l is limited to the interval $[0, \hat{L}]$ where $\hat{L}$ is an estimate of the arc length of the spline segment.

5.4 Spline Approximation of Line Segments and Circular Arcs

The spline approximation of line segments and circular arcs is based on cubic Hermite splines [15]. Cubic Hermite splines, for data interpolation, are given by the function value and the value of the first derivative at the data points [27]. This is equivalent to the position and the tangent vector at the start and end of the subpath since this thesis approximates subpaths. The most important properties of the line segments and the circular arcs are the position and the course angle. Cubic Hermite splines are therefore useful for approximating the line segments and circular arcs.

The spline approximation of the line segments and the circular arcs are derived with a requirement on the position,

$$\mathbf{S}(0) = \mathbf{p}_0, \tag{5.4a}$$
$$\mathbf{S}(L) = \mathbf{p}_1, \tag{5.4b}$$

and the tangent vector,

$$\mathbf{S}'(0) = \mathbf{v}_0, \tag{5.5a}$$
$$\mathbf{S}'(L) = \mathbf{v}_1, \tag{5.5b}$$

at the start and end of the spline.

Coefficient	x	y
a_0	$p_{0,x}$	$p_{0,y}$
a_1	$\frac{1}{L}v_{0,x}$	$\frac{1}{L}v_{0,y}$
a_2	0	0
a_3	0	0

Table 5.1: Spline coefficients for line segment

Coefficient	x	y
a_0	$p_{0,x}$	$p_{0,y}$
a_1	$\frac{1}{L}v_{0,x}$	$\frac{1}{L}v_{0,y}$
a_2	$\frac{1}{L^2}\left(3p_{1,x}-3p_{0,x}-2v_{0,x}-v_{1,x}\right)$	$\frac{1}{L^2}\left(3p_{1,y}-3p_{0,y}-2v_{0,y}-v_{1,y}\right)$
a_3	$\frac{1}{L^3}\left(v_{0,x}+v_{1,x}+2p_{0,x}-2p_{1,x}\right)$	$\frac{1}{L^3}\left(v_{0,y}+v_{1,y}+2p_{0,y}-2p_{1,y}\right)$

Table 5.2: Spline coefficients for circular arc

The generated path from Chapter 4 provides the position values and the course angles at the start and end of each subpath. The orientation of the tangent vectors are therefore given, but the magnitude can be chosen at will. The tangent vector magnitude for the line segments is chosen as

$$\|\mathbf{v}_0\| = \|\mathbf{v}_1\| = \|\mathbf{p}_1 - \mathbf{p}_0\|, \tag{5.6}$$

and the tangent vector magnitude for the circular arcs is chosen

$$\|\mathbf{v}_0\| = \|\mathbf{v}_1\| = \frac{2\|\mathbf{p}_1 - \mathbf{p}_0\|}{1+\cos(0.5\Delta\alpha)}, \tag{5.7}$$

as they give the best spline approximation [28].

The cubic Hermite spline with a unit domain parameterization from a start point $\mathbf{p}_0$ with tangent vector $\mathbf{v}_0$ to an end point $\mathbf{p}_1$ with tangent vector $\mathbf{v}_1$ is

$$\mathbf{p}(t) = (2t^3-3t^2+1)\mathbf{p}_0 + (t^3-2t^2+t)\mathbf{v}_0 + (-2t^3+3t^2)\mathbf{p}_1 + (t^3-t^2)\mathbf{v}_1 \tag{5.8}$$

where $t \in [0,1]$ is the spline parameter [29].

For line segments, the spline coefficients are simplified since $\mathbf{v}_0 = \mathbf{v}_1$. The spline coefficients for an arc-length parameterization of a line segment are summarized in Table 5.1. For circular arcs, the spline coefficients for an arc length parameterization are a scaled version of the above cubic Hermite spline and are summarized in Table 5.2.

The spline approximation of a line segment is trivial as a line segment can be exactly parameterized by arc length [30] and it is therefore not an approximation.

The curvature of a spline is computed as

$$\kappa(l) = \frac{S'_x(l)S''_y(l) - S'_y(l)S''_x(l)}{\left(S'_x(l)^2 + S'_y(l)^2\right)^{\frac{3}{2}}}. \tag{5.9}$$

The curvature of the spline approximation of a circular arc with central angle $\Delta\alpha = \pi$ and a circular arc with central angle $\Delta\alpha = \frac{\pi}{4}$ is shown in figure Figure 5.1. As the curvature of a circular arc is constant, it is easy to see how accurate the spline approximation is. The figure shows that the approximation is worst at the start and end of the circular arc. The central angle of the circular arc directly affects the spline approximation as the tangent vector magnitude depends on the central angle. The figure shows that the approximation error in curvature is significantly smaller with a smaller central angle. The average approximation errors in position, course angle, and curvature for the two circular arcs are also summarized in Table 5.3 and Table 5.4. A more accurate approximation is achieved by splitting up circular arcs with a large central angle into several circular arcs with smaller central angles. In this thesis, all circular arcs with a central angle greater than $\Delta\alpha_{\max} = \frac{\pi}{4} = 45°$ are split up to improve approximation accuracy. It should be noted that this creates more spline segments, which increases the number of transition points. Small jumps in curvature are expected at the transition points due to the approximations not being equal to the constant value at the start and end. So splitting up the circular arcs significantly lowers the approximation errors, but comes with a small cost of more small jumps in curvature.

The splitting of a circular arc into several circular arcs is easy since the only parameters that differ between the circular arcs are the start angle and the central angle, the radius and the circle center are the same. An easy way to ensure that the circular arcs does not violate the constraint, is to split up each circular arc into m splits, where $m = \left\lceil \frac{|\Delta\alpha|}{\Delta\alpha_{\max}} \right\rceil$. The new circular arcs has the following parameters

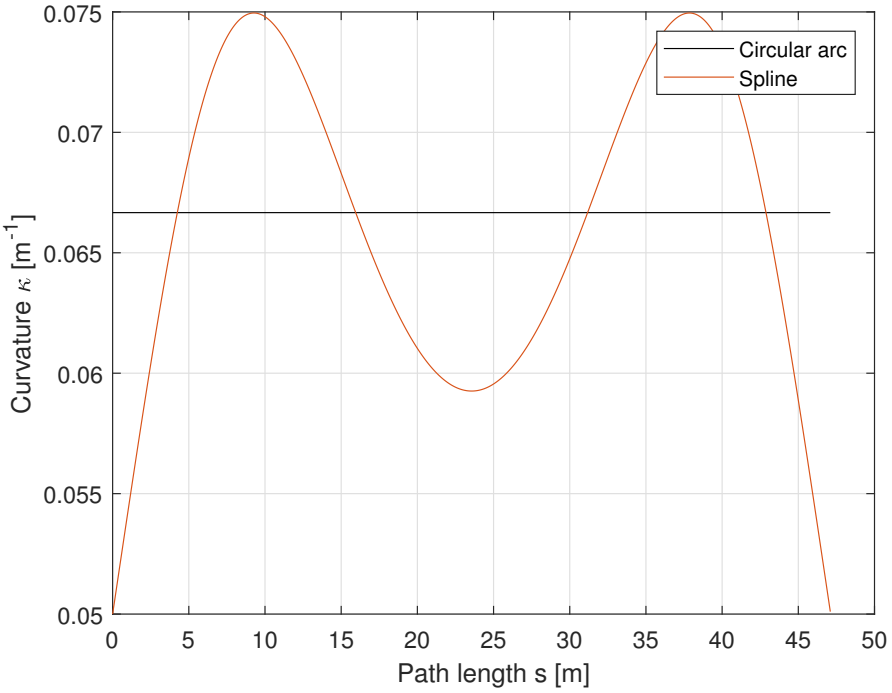
$$\alpha_{0,k} = \operatorname{ssa}\left(\alpha_{0,k-1} + \Delta\alpha_{k-1}\right), \tag{5.10a}$$

$$\Delta\alpha_k = \frac{\Delta\alpha}{m}, \quad k = 1, \ldots, m, \tag{5.10b}$$

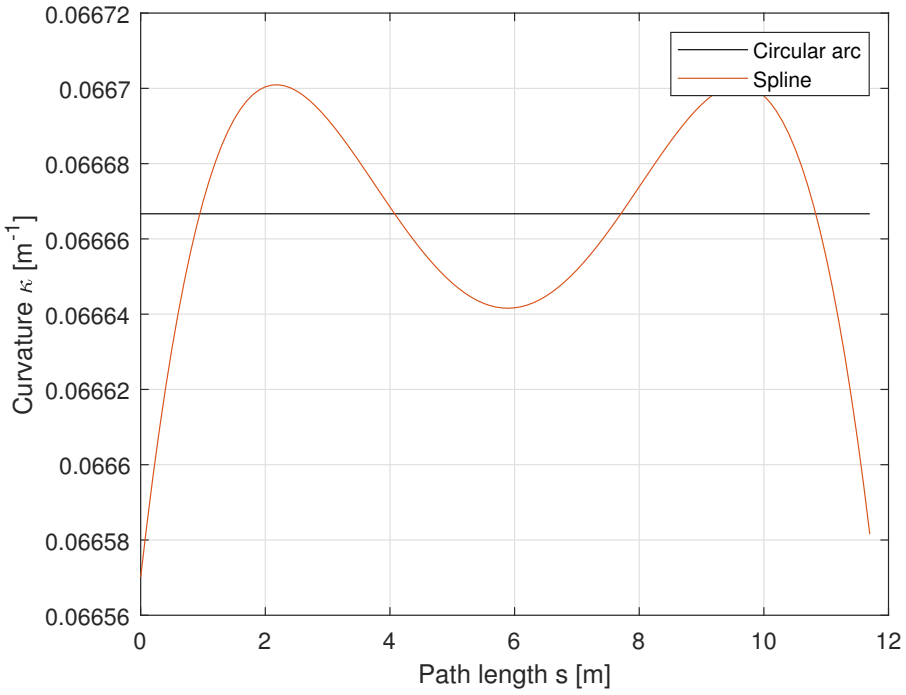
where the center point $\mathbf{p}_c$ and the radius R are equal to the original circular arc.

5.5 Spline Approximation of Euler Spirals

A Cubic Hermite spline is not useful for approximating an Euler spiral as it only provides G^1 continuity [31]. Instead, Pinchetti *et al.* [15] proposes a modifica-



(a) Circular arc with central angle π



(b) Circular arc with central angle $\frac{\pi}{4}$

Figure 5.1: Curvature of spline approximations of circular arcs with different central angles

Parameter	Average error
$\|\mathbf{p}_{\text{path}} - \mathbf{p}_{\text{spline}}\|$	0.5008 m
$\|\chi_{\text{path}} - \chi_{\text{spline}}\|$	0.0335 rad
$\|\kappa_{\text{path}} - \kappa_{\text{spline}}\|$	0.0056 m^{-1}

Table 5.3: Average spline approximation error of circular arc with central angle $\Delta\alpha = \pi$

Parameter	Average error
$\|\mathbf{p}_{\text{path}} - \mathbf{p}_{\text{spline}}\|$	0.0093 m
$\|\chi_{\text{path}} - \chi_{\text{spline}}\|$	0.00062023 rad
$\|\kappa_{\text{path}} - \kappa_{\text{spline}}\|$	0.000023503 m^{-1}

Table 5.4: Average spline approximation error of circular arc with central angle $\Delta\alpha = \frac{\pi}{4}$

tion of the Cubic Hermite spline to approximate an Euler spiral. Since the Euler spiral is employed for its curvature properties, it is important that the spline approximation accurately approximates those properties.

The requirement on the start and end position remains the same,

$$\mathbf{S}(0) = \mathbf{p}_0, \tag{5.11a}$$
$$\mathbf{S}(L) = \mathbf{p}_1, \tag{5.11b}$$

but the requirement on the tangent vector is relaxed to a requirement on the course angle,

$$\frac{S_y'(0)}{S_x'(0)} = \tan(\chi_0), \tag{5.12a}$$
$$\frac{S_y'(L)}{S_x'(L)} = \tan(\chi_1). \tag{5.12b}$$

This relaxation gives room for a requirement on the curvature,

$$\frac{S_x'(0)S_y''(0) - S_y'(0)S_x''(0)}{\left(S_x'(0)^2 + S_y'(0)^2\right)^{\frac{3}{2}}} = \kappa_0, \tag{5.13a}$$
$$\frac{S_x'(L)S_y''(L) - S_y'(L)S_x''(L)}{\left(S_x'(L)^2 + S_y'(L)^2\right)^{\frac{3}{2}}} = \kappa_1. \tag{5.13b}$$

Finding the solution to these equations is not straightforward. Pinchetti *et al.* [15] simplifies the requirements by assuming that $\chi_0 = 0$ and $\kappa_0 = 0$,

Coefficient	x	y
a_0	$p_{0,x}$	$p_{0,y}$
a_1	$\frac{1}{L}\left(3x_{\text{spiral}} - 4.5y_{\text{spiral}}^2 \frac{\Delta\kappa}{(\sin(\Delta\chi))^3} - 3y_{\text{spiral}} \frac{1}{\tan(\Delta\chi)}\right)$	0
a_2	$\frac{1}{L^2}\left(-3x_{\text{spiral}} + 9y_{\text{spiral}}^2 \frac{\Delta\kappa}{(\sin(\Delta\chi))^3} + 3y_{\text{spiral}} \frac{1}{\tan(\Delta\chi)}\right)$	0
a_3	$\frac{1}{L^3}\left(x_{\text{spiral}} - 4.5y_{\text{spiral}}^2 \frac{\Delta\kappa}{(\sin(\Delta\chi))^3}\right)$	$\frac{1}{L^3} y_{\text{spiral}}$

Table 5.5: Spline coefficients for Euler spiral with zero initial course angle and zero initial curvature

which corresponds to the fundamental Euler spiral this thesis uses under path generation and is described in Section 3.4.2. The spline coefficients for Euler spirals with zero initial course angle and zero initial curvature are summarized in Table 5.5. As this thesis uses a slightly different parameterization of the Euler spiral, the coefficients are slightly different from Pinchetti *et al.* [15]. This thesis also gives the coefficients in terms of the solution of the Fresnel integrals from Section 3.4.2. As the both the start and end point of each Euler spiral were saved during the path generation, the solution of the Fresnel integrals is easily recovered as

$$\begin{bmatrix} x_{\text{spiral}} \\ y_{\text{spiral}} \end{bmatrix} = \mathbf{R}_z^\top(\chi)(\mathbf{p}_1 - \mathbf{p}_0), \tag{5.14}$$

where $\chi = \chi_0$ if the initial curvature is zero and $\chi = \chi_1$ otherwise. The spline approximation does therefore not need to solve the Fresnel integrals again.

The assumption on the initial course angle is easily corrected for with a rotation matrix,

$$\begin{bmatrix} a_{i,x} \\ a_{i,y} \end{bmatrix} = \mathbf{R}_z(\chi_0) \begin{bmatrix} \tilde{a}_{i,x} \\ \tilde{a}_{i,y} \end{bmatrix} \quad \forall i = 1, 2, 3 \tag{5.15}$$

where $\tilde{a}$ are the coefficients from Table 5.5 and a are the corrected coefficients.

The coefficients derived only work for Euler spirals at the entrance of circular arcs because of the assumption of zero initial curvature. On the other hand, an assumption on a non-zero initial curvature and a zero final curvature leads to a different set of coefficients. The spline coefficients for Euler spirals with zero initial course angle and non-zero initial curvature are summarized in Table 5.6. The coefficients must also in this case be corrected with a rotation matrix. Note that the coefficients in Pinchetti *et al.* [15] have some typos.

The curvature of the spline approximation of an Euler spiral with zero initial curvature is shown in Figure 5.2. The spline approximates the linear change in curvature of an Euler spiral well. In contrast to the circular arcs, the approximation error is worst halfway through the spline. The average approximation errors in position, course angle and curvature are also summarized in

Coefficient	x	y
a_0	$p_{0,x}$	$p_{0,y}$
a_1	$3\frac{1}{L}y_{\text{spiral}}\frac{1}{\tan(\Delta\chi)}$	$3\frac{1}{L}y_{\text{spiral}}$
a_2	$\frac{1}{L^2}\left(4.5y_{\text{spiral}}^2\frac{\Delta\kappa}{(\sin(\Delta\chi)^3)}-3y_{\text{spiral}}\frac{1}{\tan(\Delta\chi)}\right)$	$-3\frac{1}{L^2}y_{\text{spiral}}$
a_3	$\frac{1}{L^3}\left(x_{\text{spiral}}-4.5y_{\text{spiral}}^2\frac{\Delta\kappa}{(\sin(\Delta\chi))^3}\right)$	$\frac{1}{L^3}y_{\text{spiral}}$

Table 5.6: Spline coefficients for Euler spiral with non-zero initial curvature

Parameter	Average error
$\|\mathbf{p}_{\text{path}}-\mathbf{p}_{\text{spline}}\|$	0.0105 m
$\|\chi_{\text{path}}-\chi_{\text{spline}}\|$	0.00023510 rad
$\|\kappa_{\text{path}}-\kappa_{\text{spline}}\|$	0.00010384 m^{-1}

Table 5.7: Average spline approximation error of fundamental Euler spiral with curvature change $\Delta\kappa = 0.0524$ and scale factor $w = 13.1025$

Table 5.3 and Table 5.7. The average approximation errors are about the same magnitude as the average approximation errors of the circular arc with central angle $\Delta\alpha = \frac{\pi}{4}$.

5.6 Spline Approximation of Path

The spline approximations of the three types of subpath derived are combined to approximate two-dimensional paths. Each subpath of the two-dimensional path is approximated with the coefficients from either Table 5.1, Table 5.2, Table 5.5 or Table 5.6, depending on the type of the subpath. The approximation of the path is piecewise-defined by the spline approximations of the subpaths,

$$\mathcal{S} = \bigcup_{i=1}^{n} \mathcal{S}_i, \tag{5.16}$$

in the same way as the path is piecewise-defined by the subpaths. An algorithmic description of the approximation of a two-dimensional path is given in Algorithm 5.1. As discussed in Section 5.4, the circular arcs are split up for greater accuracy.

The three-dimensional path generated in Chapter 4 is approximated by approximating the horizontal path and the vertical path separately. The horizontal and vertical spline paths are combined into a three-dimensional ap-

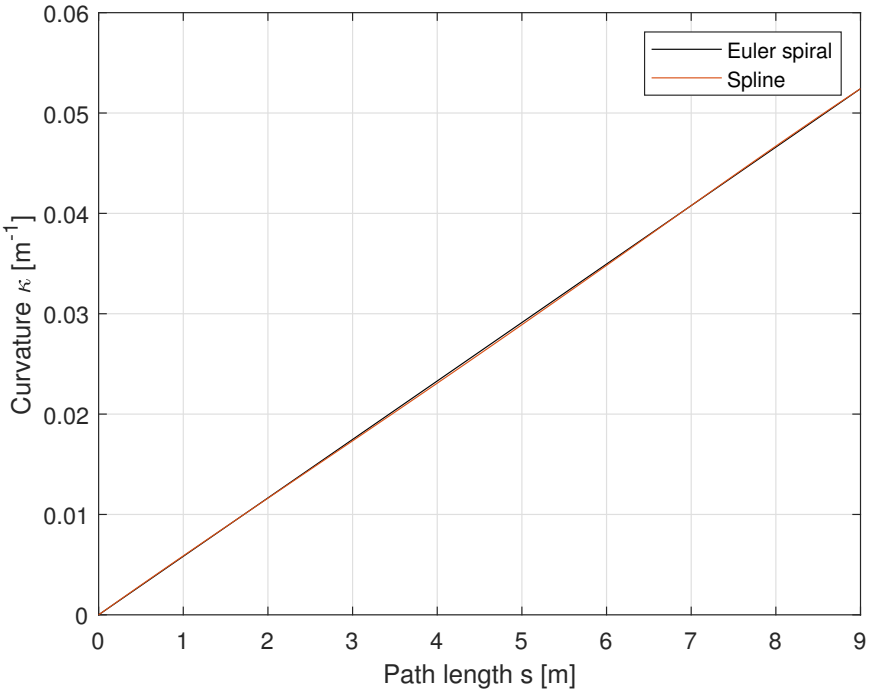


Figure 5.2: Curvature of spline approximations of Euler spiral with zero initial curvature

proximation in the same way as in Chapter 4,

$$\begin{bmatrix} h \\ s_h \end{bmatrix} = \mathcal{S}^{h-s_h}(s), \tag{5.17}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathcal{S}^{x-y}(s_h), \tag{5.18}$$

where the second coordinate of the vertical spline path is the input path parameter to the horizontal spline path and the result is the three-dimensional coordinate $[x, y, h]$.

Algorithm 5.1: Spline Approximation of 2D Path

Input: $\mathcal{P}$
Result: $\mathcal{S}$

for $k \in [1, size(\mathcal{P})]$ **do**
  if $\mathcal{P}_k == LineSegment$ **then**
    Compute spline coefficients $\mathbf{a}_x$ and $\mathbf{a}_y$ as in Table 5.1
    $\mathcal{S} \leftarrow \mathcal{S}_i(\mathbf{a}_x, \mathbf{a}_y)$
  end
  if $\mathcal{P}_k == CircularArc$ **then**
    Split up circular arcs as in Equation (5.10)
    for $k \in [1, m]$ **do**
      Compute spline coefficients $\mathbf{a}_x$ and $\mathbf{a}_y$ as in Table 5.2
      $\mathcal{S} \leftarrow \mathcal{S}_i(\mathbf{a}_x, \mathbf{a}_y)$
    end
  end
  if $\mathcal{P}_k == EulerSpiral$ **then**
    if $\kappa_0 == 0$ **then**
      Compute spline coefficients $\mathbf{a}_x$ and $\mathbf{a}_y$ as in Table 5.5
      Rotate the spline coefficients as in Eq. (5.15)
    end
    else
      Compute spline coefficients $\mathbf{a}_x$ and $\mathbf{a}_y$ as in Table 5.6
      Rotate the spline coefficients as in Eq. (5.15)
    end
    $\mathcal{S} \leftarrow \mathcal{S}_i(\mathbf{a}_x, \mathbf{a}_y)$
  end
end
return $\mathcal{S}$

5.7 Reference Signals

Feedforward values for the Euler angles and the angular rates can be computed from the splines.

Another advantage with algebraic splines is that it is easy to evaluate the derivatives,

$$\mathbf{S}'(l) = \begin{bmatrix} a_{x,1} + 2a_{x,2}l + 3a_{x,3}l^2 \\ a_{y,1} + 2a_{y,2}l + 3a_{y,3}l^2 \end{bmatrix}, \tag{5.19}$$

$$\mathbf{S}''(l) = \begin{bmatrix} 2a_{x,2} + 6a_{x,3}l \\ 2a_{y,2} + 6a_{y,3}l \end{bmatrix}, \tag{5.20}$$

$$\mathbf{S}'''(l) = \begin{bmatrix} 6a_{x,3} \\ 6a_{y,3} \end{bmatrix}, \tag{5.21}$$

which makes it easy to compute the course angle χ and the flight path angle γ,

$$\chi = \text{atan2}(S_y', S_x'), \tag{5.22}$$

$$\gamma = \frac{\pi}{2} - \text{atan2}(S_{s_h}', S_h'), \tag{5.23}$$

and the horizontal curvature κ_h and the vertical curvature κ_v,

$$\kappa_h = \frac{S_x' S_y'' - S_y' S_x''}{\left(S_x'^2 + S_y'^2\right)^{\frac{3}{2}}}, \tag{5.24}$$

$$\kappa_v = \frac{S_h' S_{s_h}'' - S_{s_h}' S_h''}{\left(S_h'^2 + S_{s_h}'^2\right)^{\frac{3}{2}}}. \tag{5.25}$$

The heading angle and the pitch angle are offset from the course angle and the flight path angle,

$$\theta = \gamma + \alpha, \tag{5.26}$$

$$\psi = \chi - \beta_c, \tag{5.27}$$

and the roll angle is given by the coordinated turn equation [15],

$$\phi = \arctan\left(\frac{V_g^2 \kappa_h}{g}\right). \tag{5.28}$$

According to Pinchetti *et al.* [15], the angular rates can be computed as

$$p = \frac{g V_g^2 \kappa_h'}{g^2 + V_g^4 \kappa_h^2}, \tag{5.29}$$

$$q = \dot{\theta} + \sin(\phi) V_g \kappa_h, \tag{5.30}$$

$$r = \cos(\phi) V_g \kappa_h, \tag{5.31}$$

where the time derivative of the pitch angle is

$$\dot{\theta} = -\kappa_v V_g \tag{5.32}$$

and the derivative of the horizontal curvature is

$$\kappa_h' = \frac{S_x' S_y''' - S_y' S_x'''}{\left(S_x'^2 + S_y'^2\right)^{\frac{3}{2}}} - 3\frac{\left(S_x' S_x'' + S_y' S_y''\right)\left(S_x' S_y'' - S_y' S_x''\right)}{\left(S_x'^2 + S_y'^2\right)^{\frac{5}{2}}}. \tag{5.33}$$

Chapter 6

Results & Discussion

6.1 Parameters and Waypoints

The paths are generated with the fixed-wing UAVs parameters in Table 6.1 for the waypoints in Table 6.2. The two-dimensional paths, 2D Dubins Path and 2D Extended Dubins Path, are generated for the x and y coordinate for the waypoints.

Parameter	Value
V_g	18 m/s
$\lvert\phi_{\max}\rvert$	60°
$\lvert\theta_{\max}\rvert$	30°
p_{des}	$120°/s$
q_{des}	$60°/s$

Table 6.1: Fixed-wing UAV parameters

	Waypoints						
	#1	#2	#3	#4	#5	#6	#7
x	-10	100	200	300	250	300	400
y	-1	0	100	0	-100	-150	-100
h	100	100	100	200	100	70	100
χ	−45°						90°
γ	0°						0°

Table 6.2: Waypoints for example path

6.2 2D Dubins Path

A 2D Dubins Path generated from the sequence of waypoints in Table 6.2 is shown in Figure 6.1. The length of the path is 701.5854 meters. This is a 2.10% increase from the piecewise linear path between the same waypoints, which is 687.1647 meters long. The MATLAB Profiler gives a run time in the range 50*ms* to 100*ms* on a standard desktop computer for the path generation. The profiler is not usable to get an exact run time value since it is sensitive to disturbances such as background processes, but it can be useful to get a ballpark value.

The properties of the path are shown in Figure 6.2. The vertical dashed lines in the figure indicate transitions from line segments to circular arcs and vice versa. The course angle is continuous with a linear increase/decrease between the different constant values due to the circular arcs at the waypoints. However, the curvature of the path is discontinuous at the transition points since the path directly transitions from line segments to circular arcs. The consequence of a discontinuous curvature is a discontinuous course angle rate, which implies a discontinuous roll angle. The 2D Dubins Path is therefore not a feasible path for fixed-wing UAVs due to the discontinuous curvature.

The path generation, summarized in Algorithm 3.1 and Algorithm 3.2, is solely based on a set of trigonometric computations per waypoint. The path generation is therefore deterministic and suitable for real-time implementations. However, this thesis proposes an iterative fix to avoid unnecessary complete circle turns, which may be a potential problem in a real-time setting. The iterative fix only needs to run in certain edge cases. During the work with this thesis, a single iteration was sufficient each time such an edge case arose. The iterative fix also only affects the neighboring subpaths. The rare occurrences combined with the small extra computational load hopefully make this is a small problem.

A computational efficient generation of a path with a continuous course angle, and with only a small increase in path length, makes the 2D Dubins Path a better choice than the too simple piecewise linear path. The discontinuous roll angle is however a problem for fixed-wing UAVs, especially for those with a slow roll rate. In practice, the direct transition from line segments to circular arcs may not be a large problem for fast and acrobatic aircraft. A fast roll rate gives a fast transient in the roll angle which approximates the discontinuous step. A conservative value for the maximum roll angle, which implies a conservative minimum turning radius, should be chosen so that the aircraft has some room to recover. However, no matter how fast the roll rate is, a fixed-wing UAV will never be able to follow the path exactly.

6.2.1 Existence of Solution

The 2D Dubins Path has some underlying assumptions that are not discussed in the original source [9]. If the rotational direction of subsequent waypoints

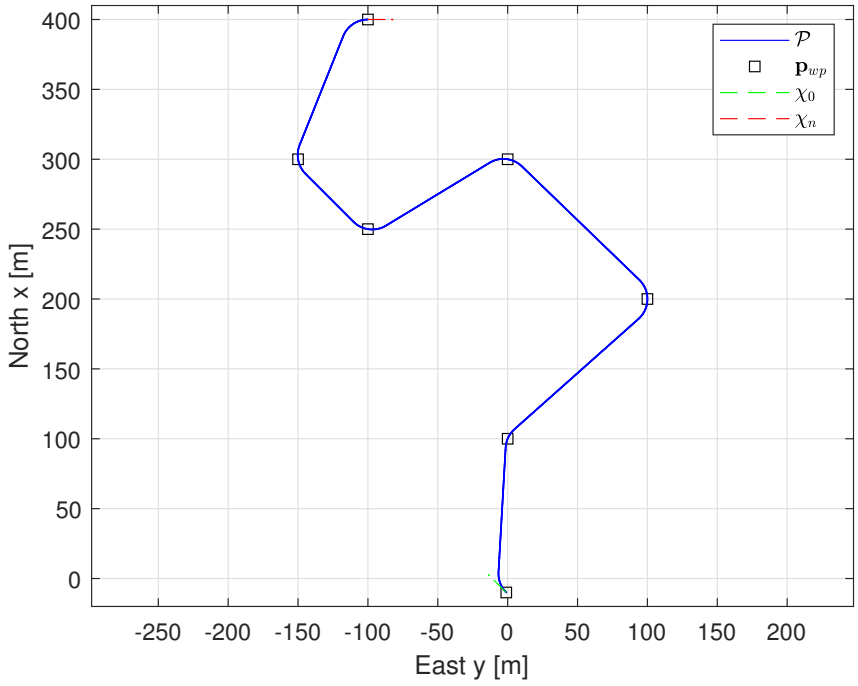


Figure 6.1: 2D Dubins Path: Example path

differs, then an internal tangent, which crosses the line between the circle centers, connects the turning circles together. In that case, the turning circles can not overlap as it makes it impossible to construct a line segment that is tangential to both circles. This can also be verified by looking at the calculation of the wheel-over and pull-out points, Equation (3.10). It uses the arccos function on the ratio between the radius and half the distance between circle centers. As the domain of $\arccos(\cdot)$ for a real result is $[-1,1]$, the distance between the circle centers must be more than two times the radius of the circles,

$$\|\mathbf{v}_{\mathrm{c}\to\mathrm{c},i}\| \geq 2R, \quad \forall i \in [1, n-1], \tag{6.1}$$

to get a real angle as the result. In other words, the circles cannot overlap. This problem only exists for internal tangents and not external tangents, but the type of tangents and the circle centers of waypoints are not known in advance. The above condition is therefore not suitable to determine in advance if a solution exists for a sequence of waypoints. However, a sufficient condition can be found by ensuring that the circles can not overlap no matter how the circles are placed. A sufficient condition on the distance between waypoints is that subsequent waypoints are more than four times the radius apart,

$$\|\mathbf{p}_{\mathrm{wp},i+1} - \mathbf{p}_{\mathrm{wp},i}\| \geq 4R, \quad \forall i \in [1, n-1], \tag{6.2}$$

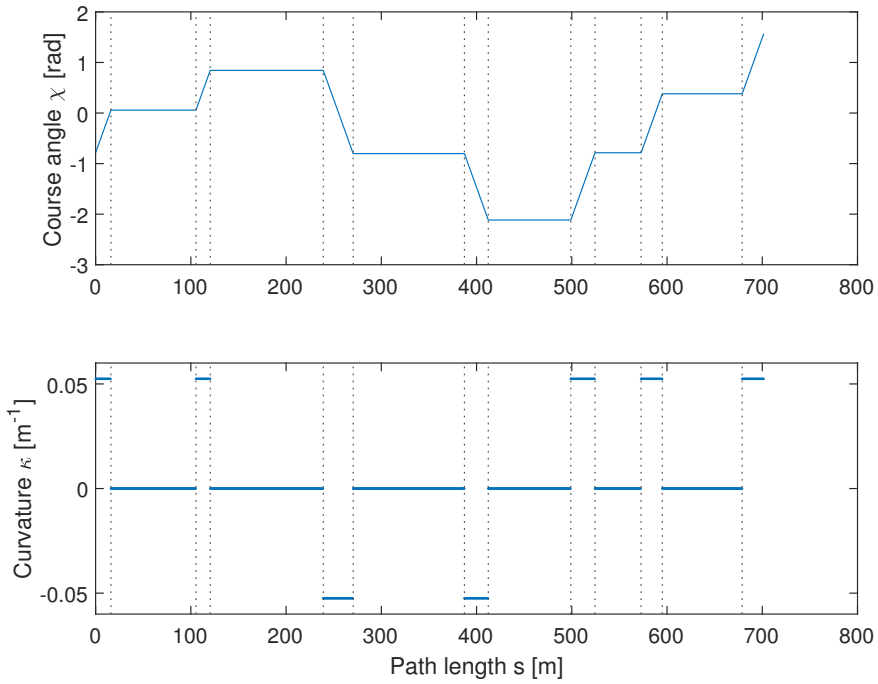


Figure 6.2: 2D Dubins Path: Discontinuous curvature

which ensures that the circles will never overlap. This is a sufficient condition on the distance between waypoints, but not a necessary condition.

6.2.2 Optimality

Even though the method is based on the Dubins Path, which is the shortest curve between two poses with a constraint on the curvature, the 2D Dubins Path is not the shortest path connecting the initial pose and final pose via the sequence of intermediate waypoints with a constraint on the turning radius. The simplistic computation of the poses at the intermediate waypoints makes the path suboptimal.

The optimal solution of the problem depends on if CCC types of Dubins Paths are taken into account or not. The above assumption on the distance between subsequent waypoints ensures that CCC types of Dubins Paths are not part of the optimal solution [32]. For the case with a single intermediate waypoint, iterative methods exist in both Sadeghi and Smith [33] and Parlangeli [32] which computes the optimal solution. For several intermediate waypoints, the problem becomes a convex optimization problem that needs to be solved several times because of all the possible combinations of maneuvers [34]. The number of possible combinations for n waypoints are 2^{n-2}. As this thesis focuses on computationally efficient methods, solving potentially exponentially

many optimization problems is not a feasible approach. An approximation algorithm for several intermediate waypoints is given in Rathinam and Khargonekar [35]. The general case with no assumption on the distance between subsequent waypoints remains unsolved, but tight lower bounds on an optimal solution are given in Manyam *et al.* [36].

All the mentioned alternatives give a better solution in terms of path length, but with a significantly higher computational cost than the simple method chosen in this thesis. Apart from the simplistic computation of the intermediate poses, the rest of the method is optimal with the above assumption. In other words, if the distance between waypoints is at least $4R$, then it is only the computation of the intermediate poses that need to be changed if the method is to be improved. As the simplicity and efficiency of the method is prioritized in this thesis, the suboptimal computation of the intermediate poses is deemed sufficient.

6.3 2D Extended Dubins Path

A 2D Extended Dubins Path generated from the sequence of waypoints in Table 6.2 is shown in Figure 6.3. The length of the path is 705.8922 meters. Compared to the 2D Dubins Path, it is a 0.61% increase. The MATLAB Profiler gives a run time in the range $80ms$ to $120ms$ on a standard desktop computer, which is slightly higher than the 2D Dubins Path. The exact values should not be trusted, but it is a sign that the method is in the same ballpark as the 2D Dubins Path.

The properties of the path are shown in Figure 6.4. The vertical dashed lines in the figure indicate transitions between the line segments, circular arcs, and Euler spirals. The curvature is continuous due to the linear curvature change of the Euler spirals, which act as transition curves between the line segments and the circular arcs. The continuous curvature gives a continuous course angle rate, which again implies a continuous roll angle. The 2D Extended Dubins Path is therefore a feasible path for the kinematic model in Chapter 2. A key assumption on the feasibility is that the angular rates such as the roll rate can change instantly, which is not possible for a real fixed-wing UAV. However, the rates are usually fast enough that such an assumption is warranted.

The Euler spirals make the path generation more complex, compared to the 2D Dubins Path. However, as this thesis uses a trick to compute the start and end point of all Euler spirals from a pre-computed solution of the Fresnel integrals, the runtime of the method is not significantly affected. With online numerical integration, the method would be in a completely different ballpark concerning the runtime. The trick is based on an assumption of constant speed and that all Euler spirals are transitions to/from a circular arc of minimum turning radius, which enables all the Euler spirals to be computed based on the same Fresnel integrals. The method is therefore almost as computationally efficient as the 2D Dubins Path.

A problem with the 2D Extended Dubins Path, which is not discussed in the original source [9], is the above assumption that all Euler spirals are transition curves to/from a circular arc of minimum turning radius. This assumption also entails an assumption on a minimum change of course at each waypoint since each waypoint will have an entering Euler spiral, a circular arc of variable length, and an exiting Euler spiral. No matter how short the circular arc is, the Euler spirals cause a minimum of change in course at each waypoint since they are all of equal length. With the 2D Dubins Path, an arbitrarily small change in course angle could be achieved by following the turning circle for an arbitrarily small distance. This is not the case with 2D Extended Dubins Path due to the equal-length Euler spirals. When the change in course at a waypoint is less than two times the change in course of an Euler spiral, a complete circle turn is needed to compensate, as shown in Figure 6.5. The problem is that the method assumes that the change in course angle at each waypoint is so large that the turning circle must be followed for at least a small distance. A solution to the problem could be to not have a turning circle at waypoints with small change in course angle and only use the Euler spirals to change course. The Euler spirals could then be adjusted to just achieve the course change necessary. However, this would most likely require online numerical integration due to the different parameters of the Euler spirals, which complicates the method drastically. This thesis did not manage to find a solution to this that did not involve online numerical integration.

If the problem with the complete circle turns is solved or the mission is such that the course change at waypoints is always large enough, then the 2D Extended Dubins Path is a better choice than the 2D Dubins Path. The method is computationally efficient while providing a continuous roll angle for UAVs that roll-to-turn. However, the possibility of having to make complete circle turns at some waypoints is a significant downside.

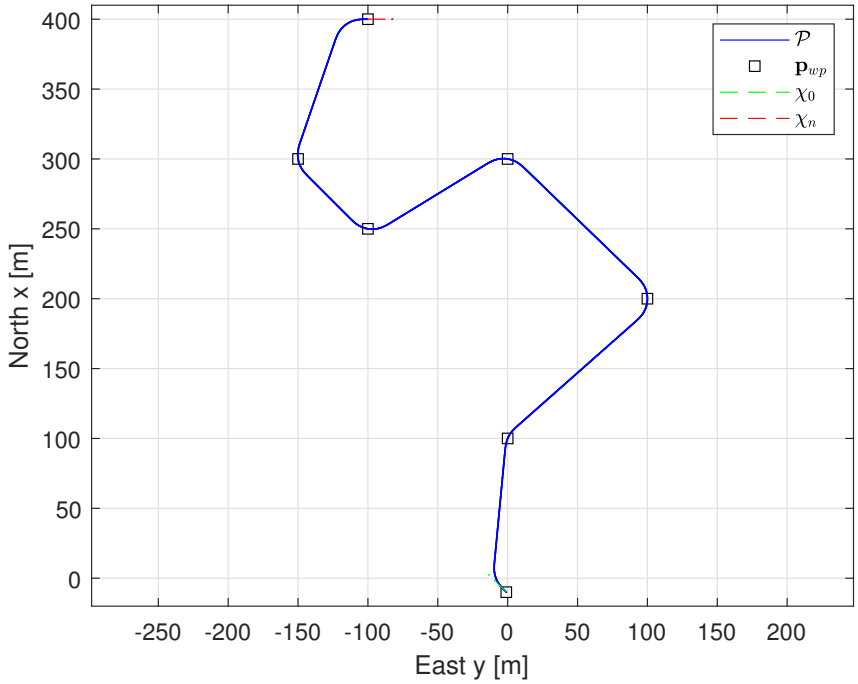


Figure 6.3: 2D Extended Dubins Path: Example path

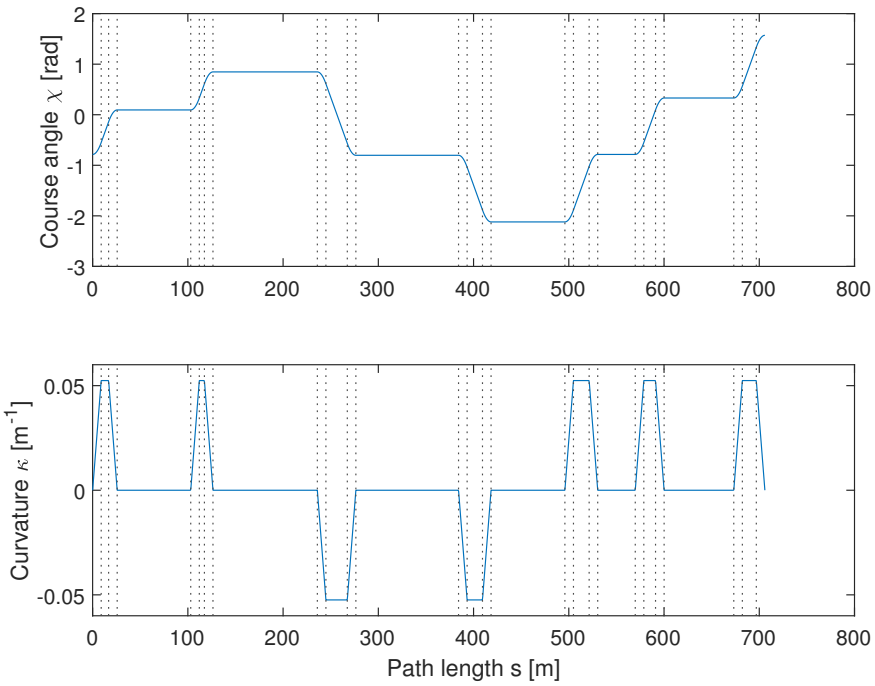
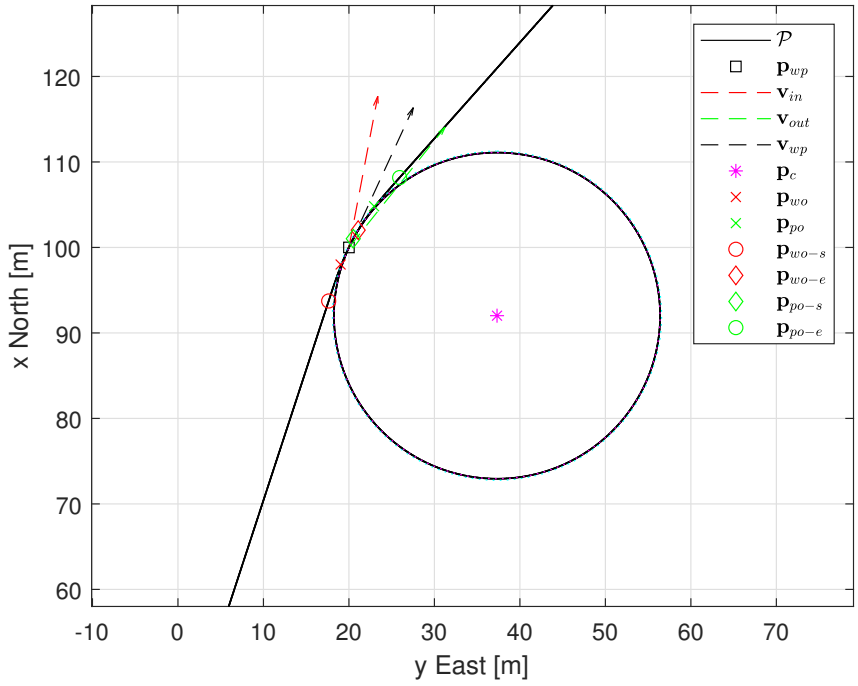
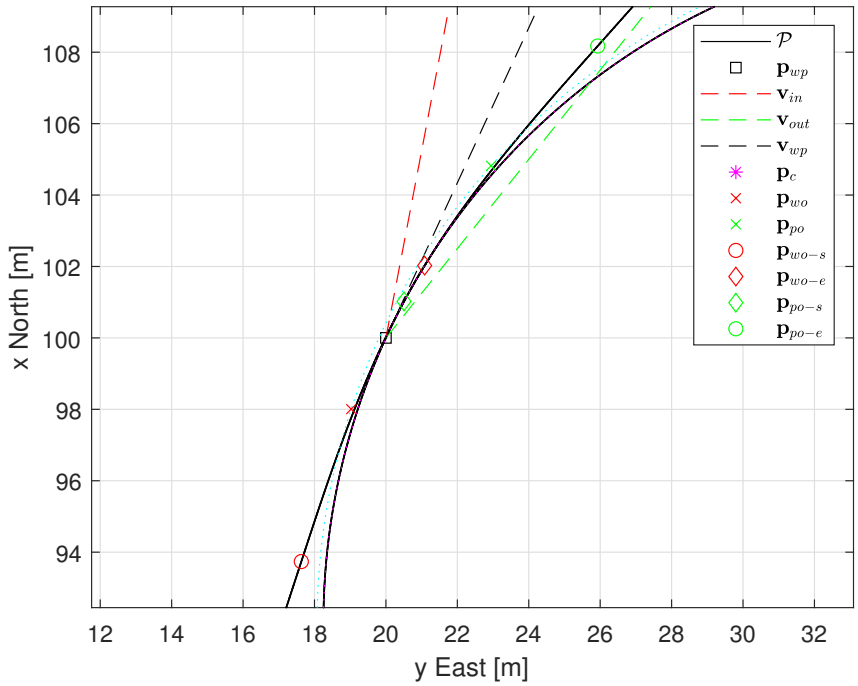


Figure 6.4: 2D Extended Dubins Path: Continuous curvature



(a) Full circle turn



(b) Zoomed in

Figure 6.5: 2D Extended Dubins Path: Unnecessary complete circle turn

6.4 3D Extended Dubins Path

A 3D Extended Dubins Path generated from the sequence of waypoints in Table 6.2 is shown in Figure 6.6. The 2D Extended Dubins Path in the horizontal plane and the 2D Dubins Path in the vertical plane is shown in Figure 6.7. The MATLAB Profiler gives a run time in the range 200*ms* to 250*ms* on a standard desktop computer. Since the 3D Extended Dubins Path calls the two-dimensional path generation methods to create the path, it is expected that the runtime is in a higher ballpark than the two-dimensional methods. However, as the two-dimensional methods are called several times due to the iterative expanding of the horizontal path, it is surprising that the runtime values are not higher.

The properties of the horizontal path are shown in Figure 6.8 and the properties of the vertical path are shown in Figure 6.9. The change in flight path angle is relatively small because of the limit on its maximum value. Since the change in flight path angle is relatively small during the vertical circular arcs, the segments with non-zero vertical curvature are very short. As expected due to the reuse of the 2D Extended Dubins Path in the horizontal plane and the 2D Dubins Path in the vertical plane, the horizontal curvature is continuous and the vertical curvature is discontinuous. A continuous vertical curvature is not needed since it is the continuity of the flight path angle that is important in the vertical plane. The 3D Extended Dubins Path is therefore a feasible path for the kinematic model in Chapter 2. As with the 2D Extended Dubins Path, a key assumption on the feasibility is that the angular rates such as the roll rate and the pitch rate can change instantly.

The path generation is simple since it mostly reuses the two-dimensional path generation methods. The vertical path is created iteratively since the horizontal path is expanded iteratively. By only expanding the horizontal path when the vertical path breaks the flight path angle constraint, optimization techniques to find just the right horizontal path length are avoided. However, this comes at a cost. The length of the path is not at all optimal. As the horizontal path is always expanded with complete circle turns, the length of the path is in some cases significantly longer than necessary. This is a sacrifice made to keep the method simple and avoids optimization techniques such as the bisection algorithm. The complexity of the method is therefore comparable to the complexity of the 2D Extended Dubins Path but comes with a significant cost at the path length.

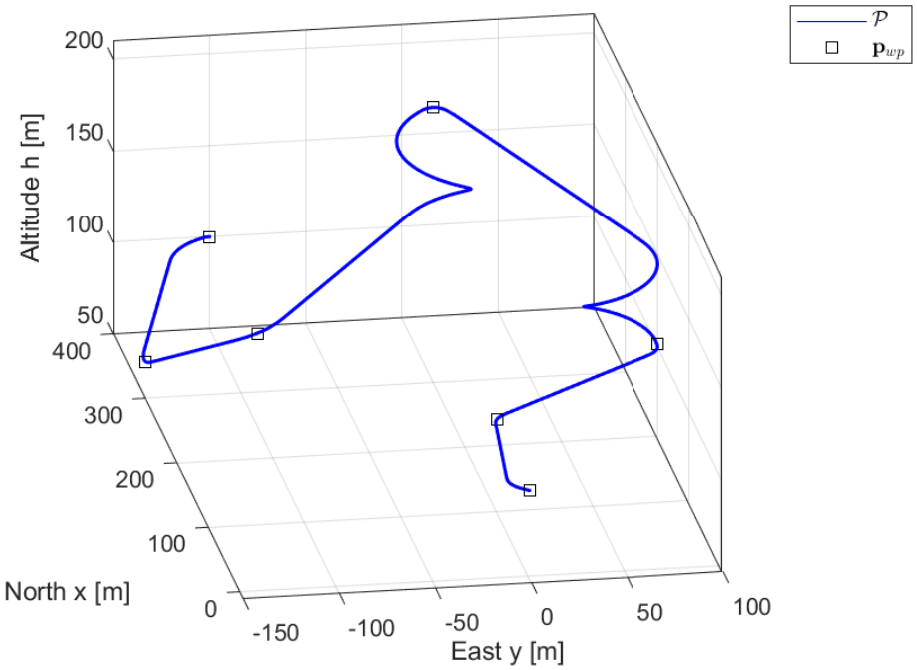
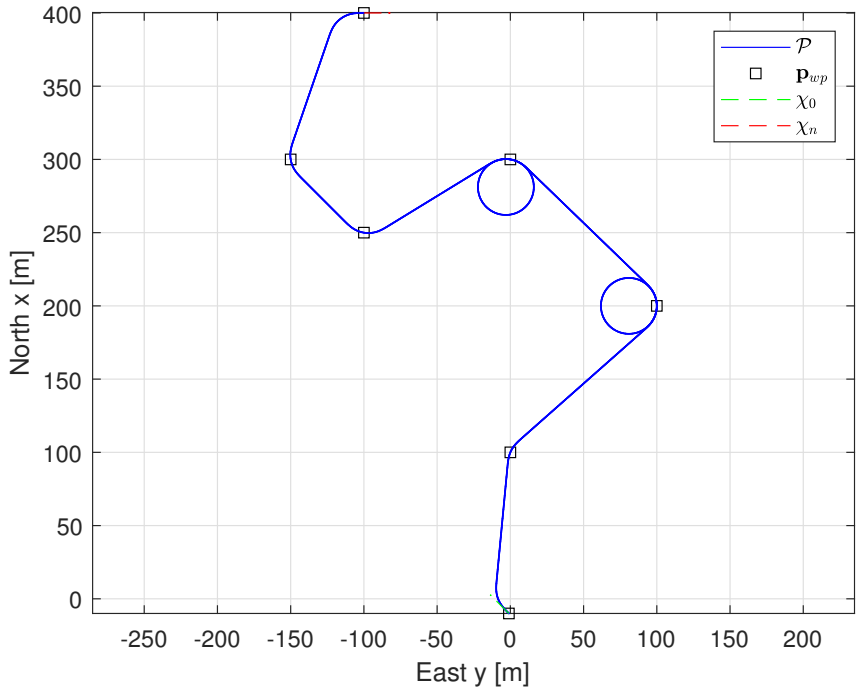
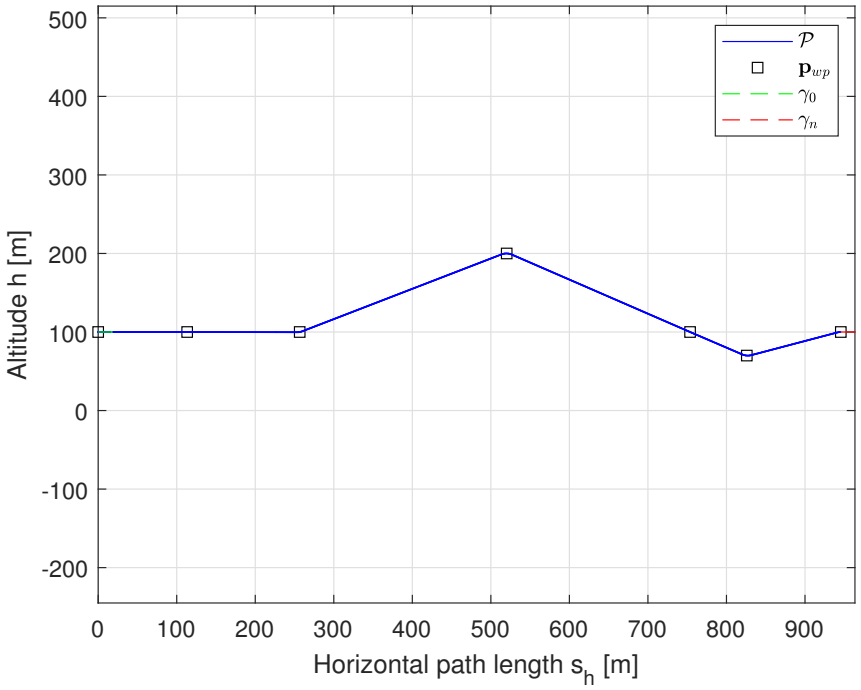


Figure 6.6: 3D Extended Dubins Path: Example path



(a) Path in $X-Y$ plane



(b) Path in $H-S_h$ plane

Figure 6.7: 3D Extended Dubins Path: Decoupled example path

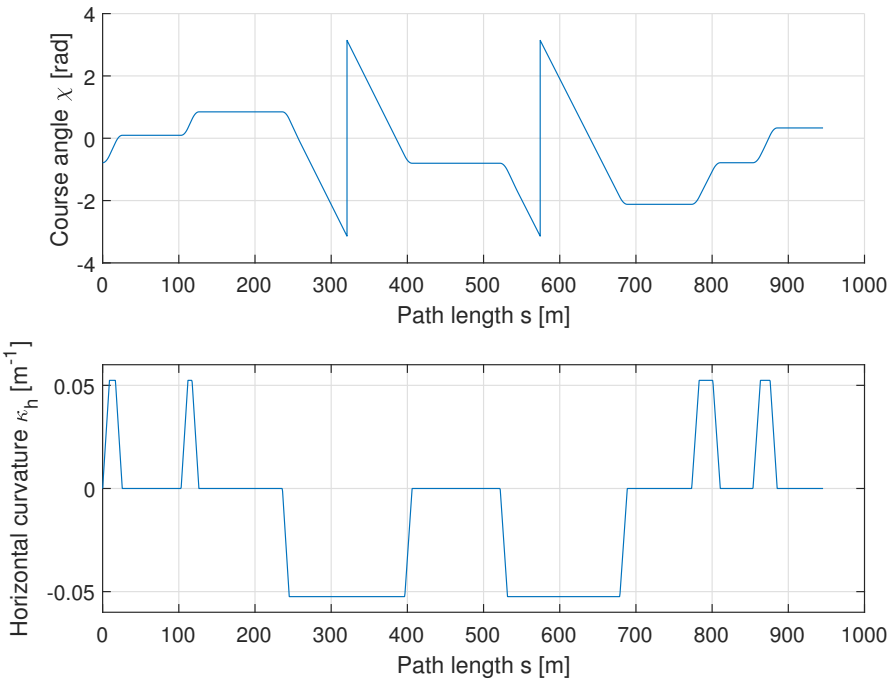


Figure 6.8: 3D Extended Dubins Path: Continuous horizontal curvature

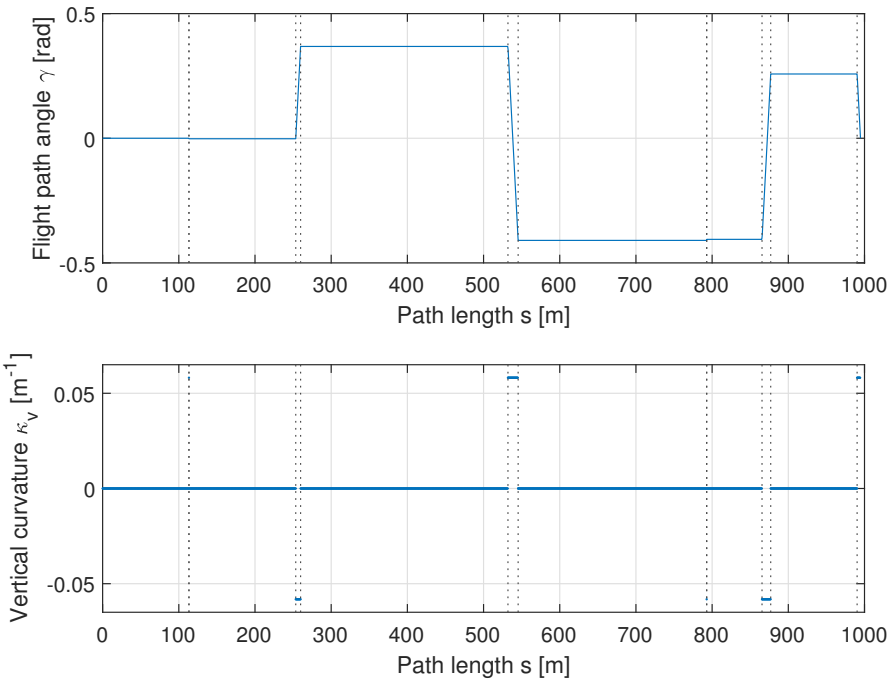


Figure 6.9: 3D Extended Dubins Path: Discontinuous vertical curvature

6.5 Spline Approximation

The course angle and horizontal curvature of the spline approximation of the three-dimensional example path are shown in Figure 6.10 and the flight path angle and vertical curvature in Figure 6.11. As expected, the approximations resembles the actual values in Figure 6.8 and Figure 6.9. However, a zoomed-in version on the curvatures in Figure 6.12 shows the effect of the approximation. Both the horizontal and vertical curvature in the zoomed-in plots should be a constant line but are small waves around the constant values. There are also jumps in value between the waves. This effect is recognized from the approximation of the circular arcs, Figure 5.1. The small jumps arise due to joining the approximations directly together. However, notice the small scale of the axis in the zoomed-in figure. The jumps are of very small values, which most likely will not have any effect in a real system. An approximation with fewer jumps can be produced by not splitting up the circular arcs, but the approximation error will be significantly larger then.

The angular rates computed from the spline approximation are shown in Figure 6.13. The pitch rate and the heading rate looks as expected. The heading rate is continuous since it is a function of the roll angle and the curvature, which both are continuous by design. The pitch rate has some discontinuous jumps when the flight path angle is changed, due to the discontinuous vertical curvature. On the other hand, the roll rate does not look right. A mostly constant value of the roll rate is expected when it is non-zero. The figure shows an almost linear roll rate when the roll angle is changed. It could be an effect of the approximation or a bug in the code. This thesis did not manage to find out the reason behind this odd roll rate.

The angular rates are expected to be discontinuous since the G^2 continuity of the horizontal path and the G^1 continuity of the vertical path only ensure continuity of the roll angle and the pitch angle. A higher level of continuity is required if continuity of the angular rates are desired. Creating a horizontal path with higher continuity is a complex task as it would require continuity of the derivative of the curvature, which requires other subpaths than those in this thesis. The vertical path could be of higher continuity if the 2D Extended Dubins Path was used in the vertical plane instead of the 2D Dubins Path. However, as discussed earlier in this chapter, the 2D Extended Dubins Path has some significant drawbacks, mainly a longer path in some cases, which was not solved.

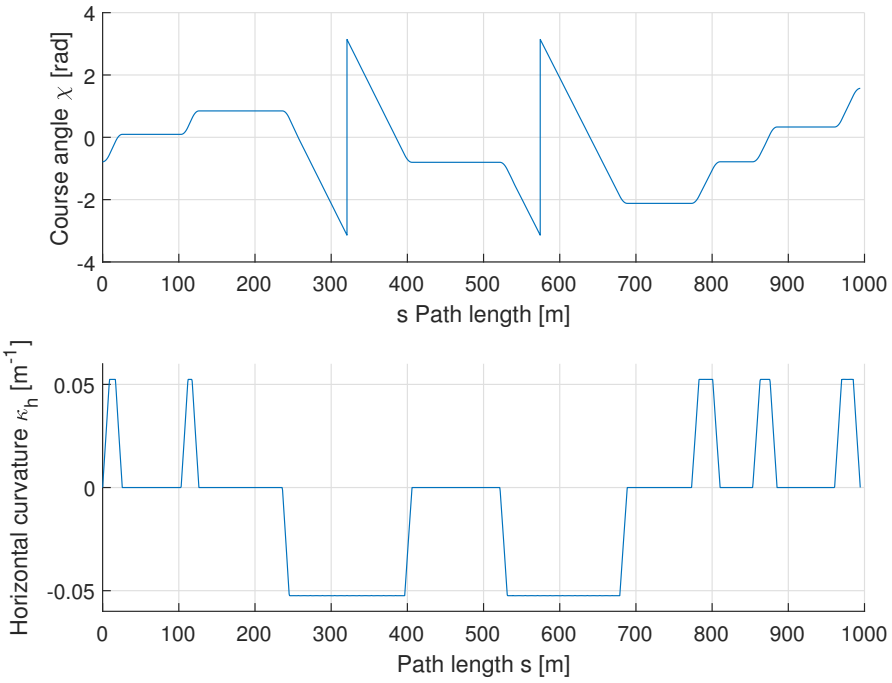


Figure 6.10: Course angle and horizontal curvature of spline approximation of example path

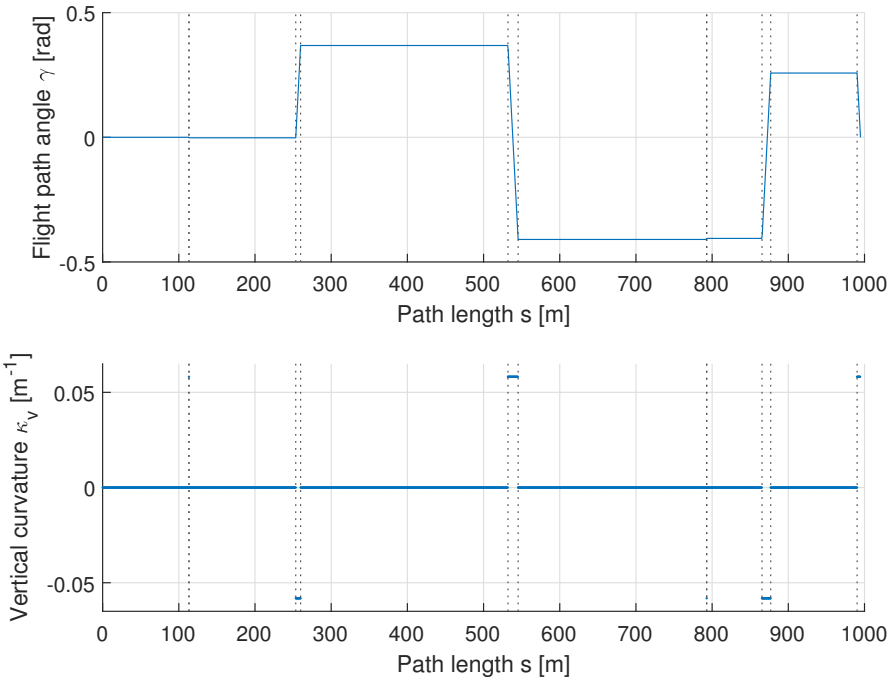
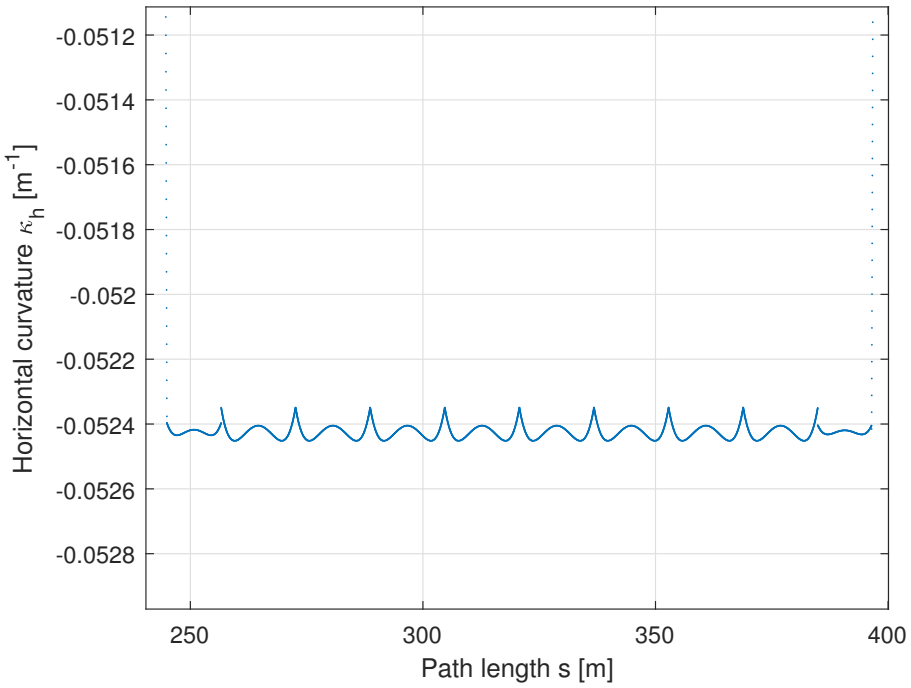
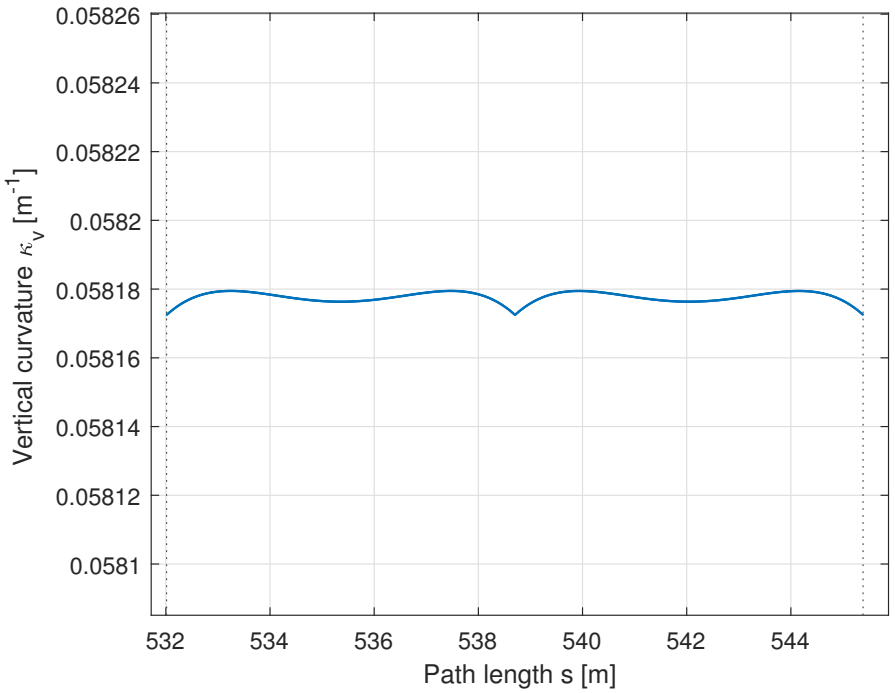


Figure 6.11: Flight path angle and vertical curvature of spline approximation of example path



(a) Zoomed in on horizontal curvature



(b) Zoomed in on vertical curvature

Figure 6.12: Zoomed in on horizontal and vertical curvature of spline approximation of example path

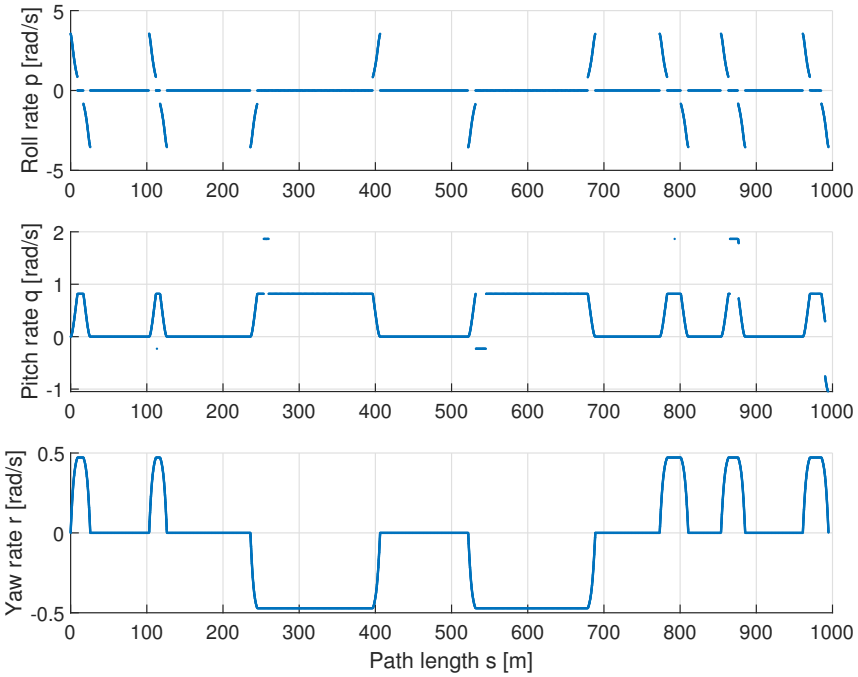


Figure 6.13: Angular rates from spline approximation

Chapter 7

Conclusion

The overall goal of this thesis was to develop a method for efficient three-dimensional path generation from a sequence of waypoints, which was feasible with respect to the dynamic constraints of a fixed-wing unmanned aerial vehicle (UAV). The three-dimensional path was generated by combining a horizontal two-dimensional path and a vertical two-dimensional path together.

The horizontal path used circular arcs to turn at the waypoints and Euler spirals at the entrance and exit of every turn for a linear curvature change. The Euler spirals were necessary for the path to have continuous curvature, which ensured that the feedforward roll angle was continuous. In contrast, the vertical path only used circular arcs at the waypoints since it was sufficient for a continuous flight path angle. When the two paths were combined into a three-dimensional path, the horizontal path was expanded with complete circle turns where necessary to satisfy the maximum flight path angle constraint. This thesis avoided numerical integration, due to the Euler spirals, during the path generation by use of a specific Euler spiral as a base and modified it by rotation and mirroring where needed. The feedforward roll angle, pitch angle, and heading angle of the resulting three-dimensional path were continuous. However, there were some unresolved issues with the Euler spirals. In some cases, an unnecessary complete turn around the turning circle was generated, which made the horizontal path significantly longer than necessary.

The resulting three-dimensional path was approximated by algebraic cubic splines to get a unified path representation and to avoid numerical integration during guidance. The splines approximated the key properties of the subpaths well, such as the linear curvature change of the Euler spirals. Furthermore, the average approximation error in position was less than 0.015 meters. The roll angle, pitch angle, and heading angle computed from the splines were approximations of the continuous signals. However, there was an issue with the roll rate computed from the splines.

Overall, a method for efficient feasible three-dimensional path generation was proposed, but it still needs some work to iron out the issues.

7.1 Future Work

The suggestions for future work are:

- Find a way to avoid the edge case of unnecessary complete circle turns with Euler spirals without online numerical integration.
- Find out why the roll rate computed from the spline approximation does not work as expected and propose a fix.
- Investigate efficient path generation which gives continuous angular rates.

Bibliography

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah and M. Guizani, 'Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges,' *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019. [Online]. Available: https://doi.org/10.1109/access.2019.2909530.

[2] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft*. Princeton University Press, Dec. 2012. [Online]. Available: https://doi.org/10.1515/9781400840601.

[3] B. L. Stevens, F. L. Lewis and E. N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. John Wiley & Sons, Inc, Nov. 2015. [Online]. Available: https://doi.org/10.1002/9781119174882.

[4] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2nd ed. John Wiley & Sons, Ltd, 2021.

[5] M. Radovic, *Unraveling 5 levels of drone autonomy*, https://dronelife.com/2019/03/11/droneii-tech-talk-unraveling-5-levels-of-drone-autonomy/, Online; accessed 2 June 2021, 2019.

[6] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: https://doi.org/10.1017/cbo9780511546877.

[7] A. Tsourdos, B. White and M. Shanmugavel, *Cooperative Path Planning of Unmanned Aerial Vehicles*. American Institute of Aeronautics and Astronautics, Inc., Nov. 2010. [Online]. Available: https://doi.org/10.2514/4.867798.

[8] A. M. Lekkas, 'Guidance and path-planning systems for autonomous vehicles,' Ph.D. dissertation, NTNU, 2014. [Online]. Available: http://hdl.handle.net/11250/261188.

[9] A. R. Dahl, 'Path planning and guidance for marine surface vessels,' M.S. thesis, NTNU, 2013. [Online]. Available: http://hdl.handle.net/11250/260957.

[10] A. Pressley, *Elementary Differential Geometry*. Springer London, 2010. [Online]. Available: https://doi.org/10.1007/978-1-84882-891-9.

[11] M. Breivik and T. I. Fossen, 'Guidance laws for autonomous underwater vehicles,' in *Underwater Vehicles*, A. V. Inzartsev, Ed., Rijeka: IntechOpen, 2009, ch. 4. [Online]. Available: https://doi.org/10.5772/6696.

[12] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2002, ISBN: 9788292356012.

[13] A. Wenz and T. A. Johansen, 'Estimation of wind velocities and aerodynamic coefficients for UAVs using standard autopilot sensors and a moving horizon estimator,' in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, Jun. 2017. [Online]. Available: https://doi.org/10.1109/icuas.2017.7991443.

[14] J. Haugen, 'Guidance algorithms for planar path-based motion control scenarios,' M.S. thesis, NTNU, 2010. [Online]. Available: http://hdl.handle.net/11250/259954.

[15] F. Pinchetti, A. Joos and W. Fichter, 'Efficient continuous curvature path generation with pseudo-parametrized algebraic splines,' *CEAS Aeronautical Journal*, vol. 9, no. 4, pp. 557–570, May 2018. [Online]. Available: https://doi.org/10.1007%2Fs13272-018-0306-3.

[16] R. Levien, 'The euler spiral: A mathematical history,' Electrical Engineering and Computer Sciences, University of California at Berkeley, Tech. Rep., 2008. [Online]. Available: https://www.levien.com/phd/euler_hist.pdf.

[17] B. A. Barsky and T. D. DeRose, 'Geometric continuity of parametric curves: Three equivalent characterizations,' *IEEE Computer Graphics and Applications*, vol. 9, no. 6, pp. 60–69, 1989. [Online]. Available: https://doi.org/10.1109/38.41470.

[18] *Ardupilot*, http://ardupilot.org, 2009-2021.

[19] L. E. Dubins, 'On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents,' *American Journal of Mathematics*, vol. 79, no. 3, p. 497, Jul. 1957. [Online]. Available: https://doi.org/10.2307/2372560.

[20] R. Kußmaul, 'A spline-based approach to path planning and path tracking for a highly-automated ga-aircraft,' M.S. thesis, University of Stuttgart, 2012.

[21] T. Fraichard and A. Scheuer, 'From reeds and shepp's to continuous-curvature paths,' *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, Dec. 2004. [Online]. Available: https://doi.org/10.1109/tro.2004.833789.

[22] J. Boissonnat, A. Cérézo and J. Leblond, 'A note on shortest paths in the plane subject to a constraint on the derivative of the curvature,' 1994.

[23] P. Vana, A. A. Neto, J. Faigl and D. G. Macharet, 'Minimal 3d dubins path with bounded curvature and pitch angle,' in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2020. [Online]. Available: https://doi.org/10.1109/icra40945.2020.9197084.

[24] M. Owen, R. W. Beard and T. W. McLain, 'Implementing dubins airplane paths on fixed-wing UAVs,' in *Handbook of Unmanned Aerial Vehicles*, Springer Netherlands, Aug. 2014, pp. 1677–1701. [Online]. Available: https://doi.org/10.1007/978-90-481-9707-1_120.

[25] H. Chitsaz and S. M. LaValle, 'Time-optimal paths for a dubins airplane,' in *2007 46th IEEE Conference on Decision and Control*, IEEE, 2007. [Online]. Available: https://doi.org/10.1109/cdc.2007.4434966.

[26] L. Schumaker, 'Introduction,' in *Spline Functions: Basic Theory*, Cambridge University Press. [Online]. Available: https://doi.org/10.1017/cbo9780511618994.003.

[27] S. Dyer and J. Dyer, 'Cubic-spline interpolation. 1,' *IEEE Instrumentation & Measurement Magazine*, vol. 4, no. 1, pp. 44–46, Mar. 2001. [Online]. Available: https://doi.org/10.1109/5289.911175.

[28] J. R. Manning, 'Continuity conditions for spline curves,' *The Computer Journal*, vol. 17, no. 2, pp. 181–186, Feb. 1974. [Online]. Available: https://doi.org/10.1093/comjnl/17.2.181.

[29] E. Kreyszig, H. Kreyszig and E. J. Norminton, *Advanced Engineering Mathematics*, Tenth. Wiley, 2011, ISBN: 0470458364.

[30] R. T. Farouki, *Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable*. Springer Berlin Heidelberg, 2008. [Online]. Available: https://doi.org/10.1007/978-3-540-73398-0.

[31] R. T. Farouki, 'Construction of rounded corners with pythagorean-hodograph curves,' *Computer Aided Geometric Design*, vol. 31, no. 2, pp. 127–139, Feb. 2014. [Online]. Available: https://doi.org/10.1016/j.cagd.2014.02.002.

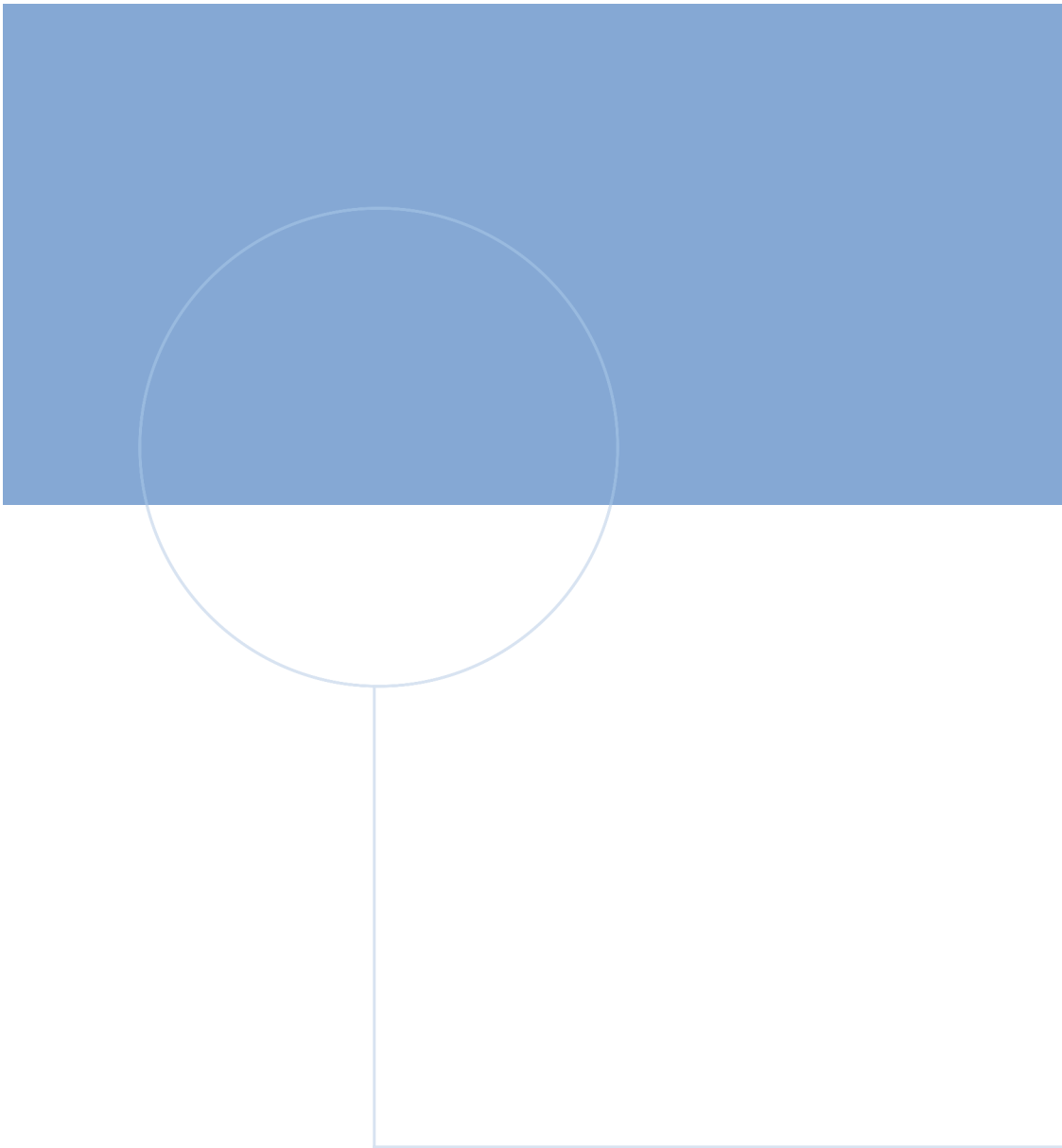
[32] G. Parlangeli, 'Shortest paths for dubins vehicles in presence of via points,' *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 295–300, 2019. [Online]. Available: https://doi.org/10.1016/j.ifacol.2019.08.086.

[33] A. Sadeghi and S. L. Smith, 'On efficient computation of shortest dubins paths through three consecutive points,' in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, Dec. 2016. [Online]. Available: https://doi.org/10.1109/cdc.2016.7799192.

[34] X. Goaoc, H.-S. Kim and S. Lazard, 'Bounded-curvature shortest paths through a sequence of points using convex optimization,' *SIAM Journal on Computing*, vol. 42, no. 2, pp. 662–684, Jan. 2013. [Online]. Available: https://doi.org/10.1137/100816079.

[35] S. Rathinam and P. Khargonekar, 'An approximation algorithm for a shortest dubins path problem,' American Society of Mechanical Engineers, Oct. 2016. [Online]. Available: https://doi.org/10.1115%2Fdscc2016-9798.

[36] S. G. Manyam, S. Rathinam, D. Casbeer and E. Garcia, 'Tightly bounding the shortest dubins paths through a sequence of points,' *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 495–511, Jan. 2017. [Online]. Available: https://doi.org/10.1007/s10846-016-0459-4.



NTNU

Norwegian University of
Science and Technology