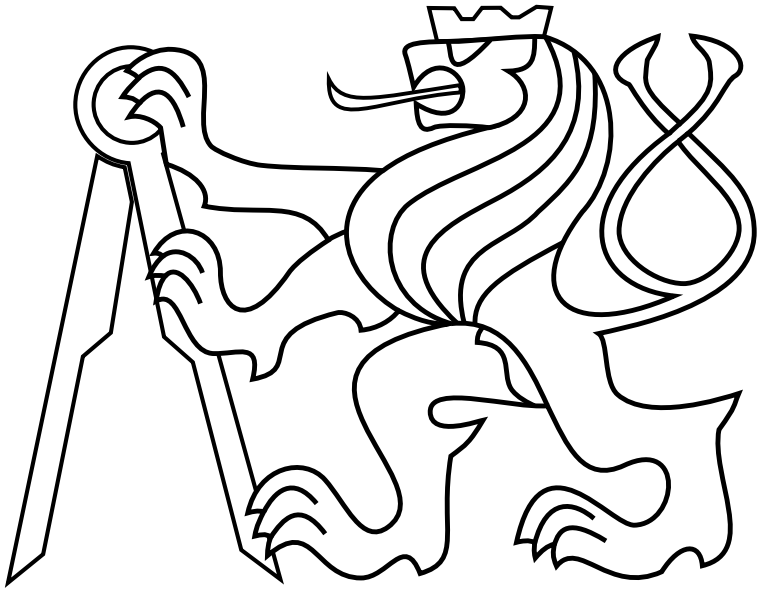
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS



Arseniy Tkachev

Multi-Goal Path Planning for Spray Writing with Unmanned Aerial Vehicle

Department of Cybernetics

Thesis supervisor: **Ing. Robert Pěnička**

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Tkachev Arseniy** Personal ID number: **464330**

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

Department / Institute: **Department of Cybernetics**

Study program: **Electrical Engineering and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Multi-Goal Path Planning for Spray Writing with Unmanned Aerial Vehicle

Bachelor's thesis title in Czech:

Plánování pohybu bezpilotního prostředku v úloze psaní textu

Guidelines:

1. Get familiar with multi-goal path planning methods suitable for spray writing with Unmanned Aerial Vehicle (UAV).
2. Based on study of geometrical representations of fonts, use exisitng or develop new font suitable for the multi-goal path planning.
3. Develop multi-goal path planning method for writing letters, words and sentences for spray writing with considered UAV.
4. Test performace of the proposed planning method on representative testing instances and in simulation environment.

Bibliography / sources:

[1] César Rego, Dorabela Gamboa, Fred Glover, Colin Osterman,"Traveling salesman problem heuristics: Leading methods, implementations and latest advances",European Jornal of Operationl Research, vol. 211, no. 3, pp 427-4, 2011.
[2] Jan Faigl, Petr Vana, "The Dubins Traveling Salesman Problem with Constrained Collectiong Maneuvers", in 2016 Acta Polytechnica CTU Proceedings, vol. 6, 2016.
[3] Jan Faigl, Petr Vana, "Surveillance Planning with Bezier Curve", IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 750 – 757, 2018.
[4] Keld Helsgaun, "Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun Algorithm", Mathematical Programming Computation, vol.7, no. 3, pp 269–287, 2015.

Name and workplace of bachelor's thesis supervisor:

Ing. Robert Pěnička, Multi-robot Systems, FEE

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

Date of bachelor's thesis assignment: **10.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

Ing. Robert Pěnička	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

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

.	
Date of assignment receipt	Student's signature

Acknowledgements

I want to thank my supervisor for his helpful advice and for showing me how to approach the problem of path planning in robotics. I thank my family for unlimited mental and material support. I can not imagine a better project for the thesis.

Abstract

This thesis describes the multi-goal path planning method for an Unmanned Aerial Vehicle (UAV) feasible for the spray writing task. The motivation is to use an autonomous UAV for precise spray writing on, e.g., roofs of industrial buildings. We formulate the writing with the UAV as a multi-goal path planning problem, and therefore, a new font suitable for the multi-goal path planning has been designed. In order to perform writing, the UAV has to travel along the input text characters. The problem can be formulated as the generalized traveling salesman problem, in which trajectories between input text segments respect the UAV constraints. We employed the Dubins vehicle to connect input text segments that allow us to traverse the final trajectory on constant speed without sharp and braking maneuvers. The implemented method has been tested in a realistic simulation environment. The experiments showed that the proposed method is feasible for the considered multirotor UAV.

Keywords: Multi-goal path planning, UAV suitable for the spray writing, Travelling Salesman Problem, Dubins vehicle.

Abstrakt

Tato práce se zabývá plánováním přes více cílů pro bezpilotní vzdušné prostředky v úloze psaní textu. Motivací je použití bezpilotní helikoptéry k preciznímu sprejování nápisů například na střechy průmyslových budov. Problém psaní textu bezpilotní helikoptérou formulujeme jako plánování přes více cílů a navrhujeme nový font vhodný pro tuto aplikaci. Helikoptéra poté musí při psaní nápisu letět podél zadaného textu s využitím navrhovaného fontu. Problém hledání cesty podél textu lze formulovat jako zobecnění problému obchodního cestujícího, kde trajektorie spojující jednotlivé segmenty písmen musí respektovat dynamická omezení helikoptéry. Na spojení segmentů písmen je použit model Dubinsova vozítka, který umožňuje průlet nalezené trajektorie konstantní rychlostí bez brzdících manévrů. Navržená metoda plánování byla otestována v realistickém simulátoru a experimenty ukazují její použitelnost pro vícerotorovou helikoptéru v úloze psaní textu.

Klíčová slova: plánování přes více cílů, bezpilotní helikoptéra pro sprejování nápisů, problém obchodního cestujícího, Dubinsovo vozítko.

Declaration

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

Prague, 22. May 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

The Unmanned Area Vehicles (UAV) or so-called drones are aircraft with remote control or autonomous operation [5]. The UAV industry has shown rapid growth in the past decade. The UAVs have become a subject of interest in numerous researches and papers. A favorable advantage of UAVs is sensing and capturing information remotely, using on-board sensors; this technique enables us to reach and explore difficult-to-access areas with low effort. Remote sensing allows us to accomplish numerous tasks without risking human life. The hexarotor UAV developed by Multi-Robot-System (MRS) [6] team is shown in the Figure 1.1.

The UAVs breakthroughs are not exaggerated, it could help modern society to develop new solutions to global problems (e.g., environmental change, wildfires, power management, and others) as well as effect and optimize industrial processes. The growth of the industry could be explained by the high potential of the UAV application in numerous areas (e.g., video production [7], agriculture, forestry [8], photogrammetry and topography [9], disaster monitoring [10] and others) and the number of applications continues to increase.

Imagine a high skyscraper, an industrial building, and another artificial structure. The exterior decoration process consists of painting, which requires scaffolding and building climbing; the methods are human power demanding and dangerous. Painting with the use of the UAV could reach an accurate graphical result in autonomous mode with lower human effort and cost. The drone with sprayer onboard can easily apply paint on difficult-to-access areas.

This kind of UAV application is relatively new and has been addressed with a limited number of approaches. Stippling with aerial robots was proposed in [11]. The presented

Figure 1.1: Hexarotor UAV developed by MRS team. Source: [1]

method uses motion capture markers for positioning with ink sponge fixed on the arm. The method was tested and could perform on planar canvas with absolute accuracy in the position of stipples. A different method was presented in [12]; the UAV demonstrated the ability to draw on both planar and 3D surfaces.

The focus in the thesis is specifically on writing with the use of the UAV. In order to accomplish writing, the UAV has to travel along the characters of the input text (e.g., word, sentence), which is requested to be written. Elements (e.g., lines, curves) could describe the character, and each element includes points (e.g., start, end, control). Therefore, to paint characters, the UAV has to traverse the segments and visit its points that describe each character. This leads to the multi-goal path planning problem, which attempts to establish a geometric path from the given start point to the endpoint visiting waypoints and respecting the constraints of the UAV. The thesis presents a multi-goal path planning method for the UAV suitable of the spray writing.

1.1 Problem Overview

The whole thesis incorporates a set of sub-problems. Typically, the problem of finding the multi-goal path for multi-rotor UAVs has precondition characterized as the set of points to be visited, defined in three-dimensional space. In this thesis, it requires an additional step to collect those points of interest. The points of interest are obtained from the input

text, which is requested to be written by the UAV. The string is served in the form of character, word, and sentence. Then the given input is represented in the chosen font type suitable for multi-goal path planning. Moreover, the characters of the input text are divided into segments (e.g., lines, curves) which incorporate the points of interest (e.g., start, end, control). The designed "A" character with the points of interest marked as blue dots is shown in the Figure 1.2. The established requirements for fonts and rigorous description of the font design process is discussed in the Chapter 2.

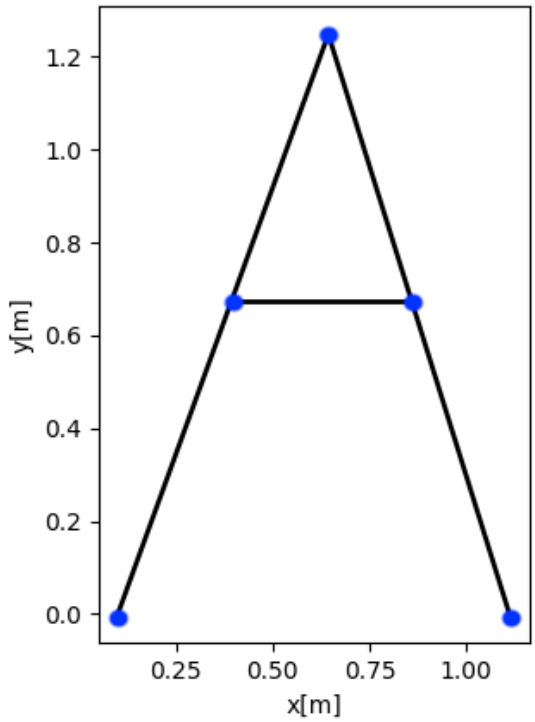


Figure 1.2: The "A" character with marked points of interest.

A significant drawback of the UAV is battery power limitation. Typically, the UAV equipped with high-performance electronics, numerous sensors, and interconnected modules, thus are very power demanding. Careering a larger battery increases the weight, which leads to the higher power consumption of electric motors. The average operation time of the UAV is approximately ten minutes. In the case of the thesis, the UAV could be requested to write numerous letters incorporating a sentence. In order to perform writing with the UAV, a trajectory has to constitute a path of the letter segments (e.g., lines, curves). Since the UAV operation is bounded by the maximum flight time, it is desirable to generate the shortest possible multi-goal path. The core problem of the thesis could be defined as the Generalized Traveling Salesman Problem (GTSP) [13] that is an extension of the classical Traveling Salesman Problem (TSP) [14]. The formulation of the GTSP problem states that

the set of cities is divided into clusters, and every cluster is visited by a salesman exactly once. In the case of the thesis, an input text is divided into segments, such as lines and curves. The segment of the character could be traversed in two different directions; thus, the direction of traveling along the segment is described by a sequence of its points (e.g., start, end). The two sequences (two directions of traveling along a segment) then served as nodes (cities) in the GTSP. Two nodes represent two directions of traversing each segment and form individual clusters. The nodes representing the direction of traversing the segments are marked as arrows in the Figure 1.3. The nodes of the individual cluster are marked as arrows of the same color in the Figure 1.3. For example, the "C" letter in the Figure 1.3 consists of only one curve segment and could be traversed in two directions, two nodes represent the direction are marked as blue arrows and constitute an individual cluster. In order to solve the GTSP problem, we employ Lin–Kernighan–Helsgaun (LKH) [13] algorithm; the algorithm is used without modifications. The LKH algorithm provides a high-quality solution.

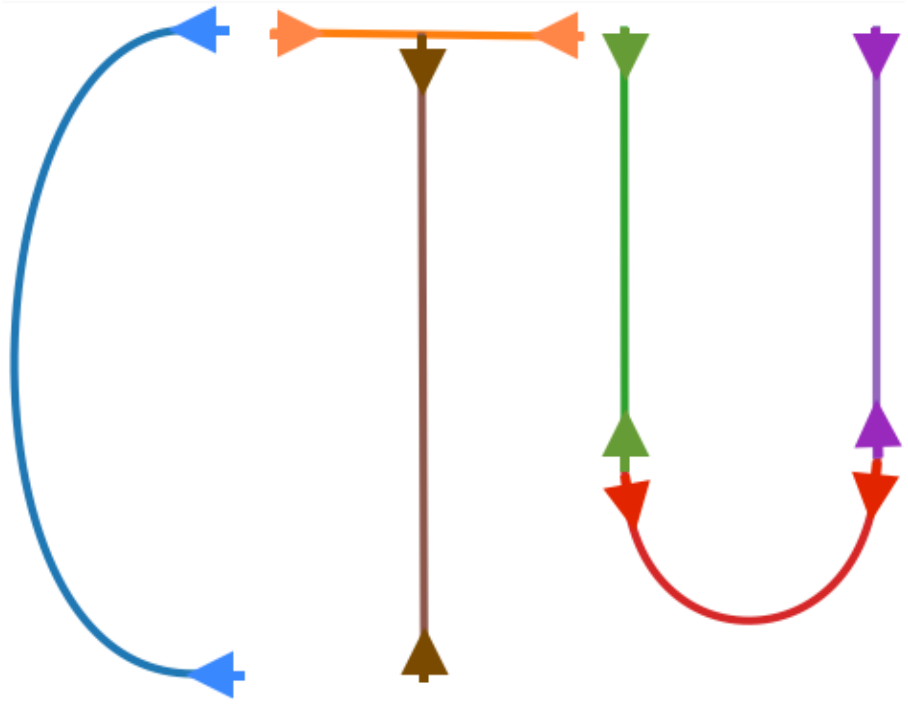


Figure 1.3: The string "CTU" with marked nodes.

GTSP establishes an Euclidean path formed from a sequence of visited nodes. The ordinary GTSP uses the Euclidean distances between nodes and thus line segments to

connect the individual nodes. Considered hexarotor UAV has, however, limited maximal acceleration and velocity. Traversing a path composed of line segments with sharp turns is therefore problematic without violating the constraints or without missing the precise positions of the nodes. Therefore, we employ the curvature-constrained Dubins vehicle [15] to model the path between individual character segments. Each Dubins maneuver is composed of three parts, where each part is either a straight line or arc. One of six possible Dubins maneuvers is then the shortest path between start (x_0, y_0, θ_0) and end (x_1, y_1, θ_1) waypoints. Since the nodes in the GTSP represent the direction of traversing the segment (i.e., cluster), both start and end waypoints are given and fixed, including its heading angle. An example of Dubins path between clusters of the "CTU" characters is shown in the Figure 1.4.

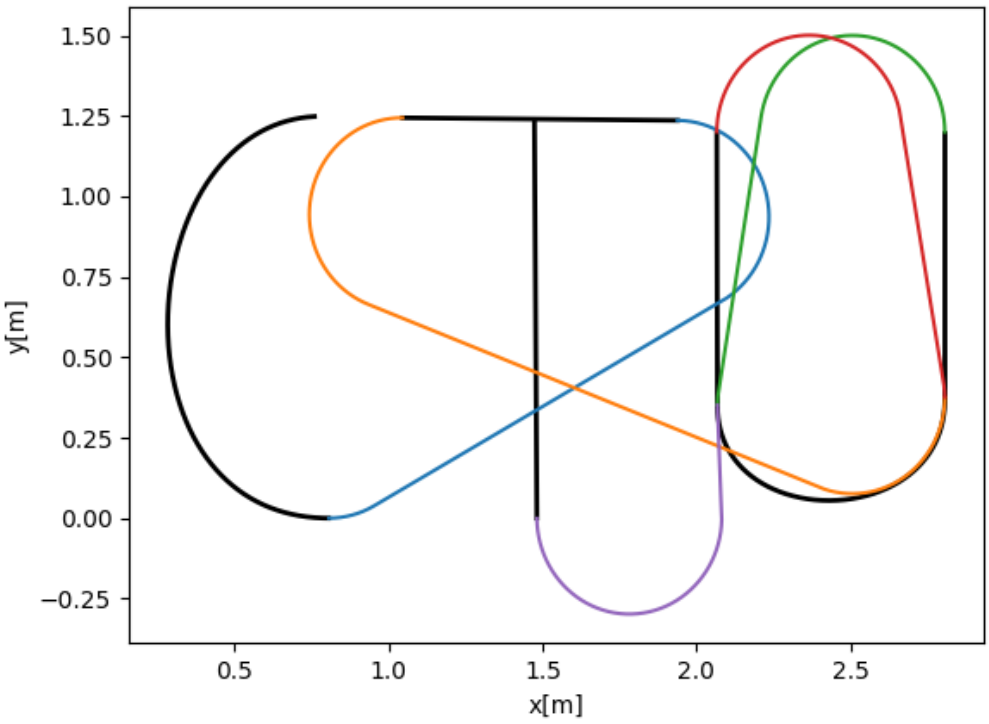


Figure 1.4: The trajectory composed of Dubins path between nodes.

The thesis is providing a complete solution in the form of a motion planning method. The method accepts a string (e.g., word, sentence) as the input parameter, the output is a continuous trajectory in 3D space where the distance between a writing surface and the UAV is a constant. An example of a smooth trajectory that respects the UAV kinematics and visits each cluster in the order; thus, the trajectory length is minimized is shown in Figure 1.4.

1.2 State of the art

The sub-problems of the thesis discussed in the Section 1.1 are the subject of numerous research studies, and those have a wide range of possible applications. The section provides an overview of those works and closely related studies. Since the thesis establishes a new font suitable for motion planning, the digital font technologies are also discussed in the section.

The font design is a significant part of the thesis. A Font is an array of visually represented letters and symbols. The implementation of characters depends on the font format. The history of the computer fonts started from Bitmap font format, which is described in the Chapter 2 and in [16]. The font format which satisfies the requirements of applications such as web, desktop, mobile, is called Outline font format. The examples of fonts in Outline format are True Type Font (TTF) [16], Open Type Font (OTF) and Scalable Vector Graphics(SVG) [17].

The Font design in the thesis has been accomplished with the use of a software called FontForge [18, 16]. It allows to style characters in TTF and OTF font formats.

Finding the close to optimal sequence of visits the character segments is formulated as the GTSP problem in the thesis. The generalized version of the TSP problem could be addressed with similar approaches as TSP. The GTSP is convertible into an asymmetric TSP (ATSP) with the same number of cities. The method of transforming GTSP into ATSP is called Noon Bean transformation, it is discussed in [19, 13]. Then the obtained ATSP could be solved with Lin-Kernighan heuristic (LKH), which is the most frequently used solver. The performance of the LKH solver has been measured and showed that it could reach an optimal solution in a reasonable time [20, 21].

The task of finding the shortest possible path between two waypoints for the curvature-constrained vehicle was addressed with numerous approaches. In 1957, Dubins [15] first proposed the solution to the problem, which is called the Dubins maneuver. He proved that the path should include at most three segments (e.g., a line denoted as S, a curve denoted as C). It was shown in [15] that the path has to be in the form of CSC or CCC. An extension of Dubins path for vehicles that could travel in both back and forward directions is studied in [22].

The DTSP is an extension of the classical TSP problem for Dubins vehicle and was addressed with numerous approaches. The DTSP incorporates two sub-problems: 1) the underlying TSP, which aims to find the most optimal sequence that visits all cities, 2) finding the shortest possible path for the curvature-constrained vehicle or Dubins vehicle that is limited by minimal turning radius [23]. The challenge of the DTSP is the heading selection within the interval $[\pi, 2\pi)$. Since the heading angle of each point of interest is fixed in the thesis, the thesis's problem is rather GTSP with a special distance function that respects the vehicle kinematics. However, both problems DTSP and GTSP with a special distance function consist of similar sub-problems such as underlying TSP. The sub-problem of finding the shortest path between two waypoints could be addressed with similar

approaches. It was attempted to solve the DTSP problem with use of numerous heuristics such as [24, 25, 26] and approximation algorithms [14, 24]. The unsupervised learning approach based on a self-organizing map [27] showed good performance. Formulation of Dubins Interval Problem (DIP) [28] is aiming to find the shortest path between two points respecting the chosen turning radius. Based on DIP, a tight lower bound of the DTSP has been presented in [10].

Chapter 2

Geometrical representation of fonts

Contents

This chapter describes the geometrical representations of the fonts. It is attempted to develop a new font suitable for multi-goal path planning. The right font choice makes text well recognizable and enhances visibility. For the thesis, the font choice is affecting the method of collecting coordinates from the input text and storing the coordinates.

A Font is a collection of visually represented letters and symbols of a similar style. The idea of a digital font is inspired by the Movable-type printing technology consisting of metal boxes filled with letters. An Example of Movable-type is shown in Figure 2.2.

The font consists of glyphs, and a glyph is defined as the image of the character [29]. Glyphs are graphical representations of characters of a chosen alphabet. For example, if the character "A" has three variations designed and displayed in different ways, thus variations of the character are considered as three glyphs. Individual font types (e.g., True Type Font, Open Type Font) use the relation when one glyph represents precisely one character. For example, the character "G" is described as "the capital Latin letter G" and consists of only one glyph. However, in some exceptional cases, a glyph could even represent more than one character, and simultaneously a character could be represented by several glyphs (e.g., Latin letter "é" build from two glyphs "e" and "´" [29]. Glyphs implementation may differ from each other depending on the font type and on the chosen alphabet set.

Figure 2.1: An example of Movable-type. Source: [2]

Bitmap type is one of the earliest font types. The simplicity of this font type is undoubtedly a benefit. Its glyph consists of an array of white and black pixels representing each letter [16]. Bitmap type is not commonly used for rendering on high-resolution displays and printers; it would require an individual glyph to consist of thousands of pixels. However, Bitmap is still widely used for embedded boards due to its excellent performance for low-resolution and small-size displays. An example of a rendered graphical result of lowercase "a" letter in Bitmap font type is shown in Figure 2.2.

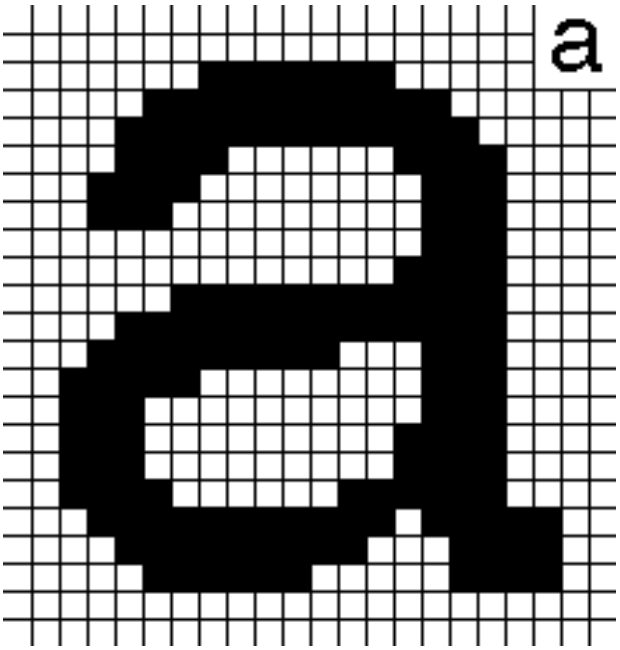


Figure 2.2: The example of rendered graphical result of lowercase "a" letter in Bitmap font type. Source: [3]

Another well-known font type is the Outline. The font type is scalable, and that is the reason it is frequently used. It has extended glyph size and multi-platform support, which is the benefit of this font type. An Outline's glyph is implemented by a chain of points which describes a particular line or a curve. Outline fonts employ Bézier curves for styling parabolic curves and arcs [16].

Bitmap font type requires a high amount of pixels for high-resolution printing (as it was mentioned earlier - the resolution depends strictly on the number of used pixels). Therefore, it is memory demanding compared to Outline font type, and that is the reason Bitmap font type is rarely applicable for web or desktop programs. Even though the Bitmap fonts are quite fast to render and easy to design, the font does not support scaling and cross-platform deployment. We decided not to use this type of font due to its disadvantages.

Outline font type has high quality of rendering, multi-platform support, and low space complexity. The Outline fonts can describe different segments of a character with the use of two, three, or four points. Therefore, the storage of the segments could be accomplished using numerous data structures (e.g., hash maps, lists) for further processing. There exist variations of tools for designing and processing Outline fonts such as FontForge, Glyph. In this thesis, we employed Outline font types as the most appropriate solution.

2.1 Bézier curves

Bézier curves are used in numerous font types (e.g., True Type Font, Open Type Font) to style the curves. It enables one to characterize the curve with two, three, or four points depending on the order of a Bézier curve. A curve could be easily changed by manipulating with control points; it makes Bézier curves applicable for numerous cases in type design.

Cubic Bézier curve of degree n is defined as follows:

$$r(t) = \sum_{i=1}^{n} b_i B_{i,n}(t), \tag{2.1}$$

$$0 \leqslant t \leqslant 1.$$

Where b_i are the control points and $B_{i,n}(t)$ are Bernstein polynomials represented as basis. The first and last control points represent the start and the end of a curve. The straight line connecting two intermediate points is called a "control polygon". An example of the Cubic Bézier curve with a control polygon is shown in Figure 2.3. The convex hull of control points never collides with a curve.

Cubic and Quadratic Bézier curves collectively could constitute a composite curve that is often used in sophisticated font graphics. However, higher degree Bézier curves have a higher amount of control points and are rarely used in font design. Composite curves substitute high degree curves in font formats such as in the True Type Font and Open Type Font.

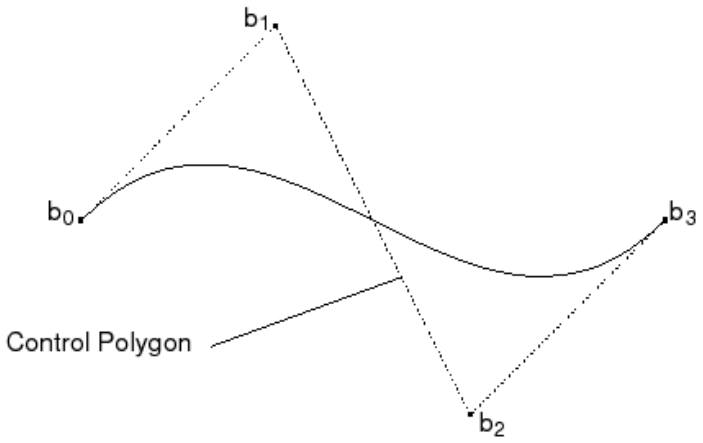


Figure 2.3: An example of Cubic Bézier curve with control polygon.

2.2 Scalable Vector Graphics (SVG)

This section describes the Scalable Vector Graphics (SVG). It is a language based on XML syntax for designing two-dimensional graphics. The SVG is employed by Cascading Style Sheets 2 (CCS2) as the main tool for font data control. There is a part of SVG called SVG font aimed to distribute font data to the user in its initial appearance. However, the glyph outlines have the same attributes and objects as SVG images. Unlike True Type Font or Open Type Font formats, SVG does not provide a table where each glyph is attached to label and description. The graphical path element [17] specifies the glyph. An example of "A" character in SVG format is shown in the Listing 2.1.

```
<svg width="57.2" height="68.2" viewBox="0 0 57.2 68.2" xmlns="http://www.
   w3.org/2000/svg">
<g id="svgGroup" stroke-linecap="round" fill-rule="evenodd" font-size="9pt
   " stroke="#000" stroke-width="0.25mm" fill="none" style="stroke:#000;
   stroke-width:0.25mm;fill:none">
<path d="M 0 68.2 L 23.4 0 L 33.8 0 L 57.2 68.2 L 48.5 68.2 L 42.5 50 L
   14.8 50 L 8.7 68.2 L 0 68.2 Z M 17.4 42 L 39.8 42 L 28.8 8 L 28.5 8 L
   17.4 42 Z" vector-effect="non-scaling-stroke"/>
</g></svg>
```

Listing 2.1: SVG script example

The "path" element in Listing 2.1 is described by "d" attribute. Every Path instruction is implemented as a single symbol in order to reduce file size, as the file size is typically one of the constraints in systems where the performance is important (e.g., web-applications).

Paths instructions are defined as:

- M - move to
- L - line to
- Z - close path
- C - curve to

It is required for the path element to start execution from the move to command. The move to (M) instruction creates a new current point; which is controlled by its parameters (x, y coordinates). If there is a sequence of coordinate pairs after M instruction, then the L command is applied to each pair as a substitute.

The Close path (Z) instruction terminates the sub-path and draws a line from the current point to the start point of the sub-path. The command is not getting any parameters.

The line to (L) instruction is drawing a line from the current point to a next point specified by the instruction parameters x and y.

The curve to (C) instruction draws a curve and takes the start, end, and control points of the curve as parameters.

2.3 True Type Font (TTF)

Another well-known font type is True Type Font. Famous Apple Inc. and Microsoft developed it. The TTF allows type developers to style the glyph and control its graphical result after rendering. There exists a language with a syntax similar to assembly language for designing glyphs in the TTF format. In TTF, each glyph is characterized by the outlines. Unlike SVG format, each glyph is representing only one character. One-to-one relation allows to process only the glyph and avoid processing of both a character and the glyph, which reduces the complexity of the scripts. Typically, a TTF file includes a table consisting of glyphs encoding format, ID, and description. Such tables allow the acceleration of a text rendering process and search for a glyph, which is considered a benefit of this font type.

2.4 Font proposal for multi-goal path planning

In this thesis, the font has to be suitable for the multi-goal path planning. In the section, we summarize the font overview and establish the requirements for the font. In order to perform writing with the UAV, a trajectory has to incorporate a path of the input text segments (e.g., lines, curves). The problem of finding the multi-goal path is formulated as the Generalized Traveling Salesman Problem (GTSP); it has precondition formulated as a set of points that are divided into clusters. The corresponding sets have to be obtained

from the input text represented in the chosen font. Each set consists of the start point and the endpoint of the particular segment.

In order to establish requirements for font, we observed a few characters represented in regular fonts, which could be converted to SVG file format using "svgpathtool" specification. For the first example, "C" character has been chosen. It consists of straight lines and Bézier curves segments. The character is shown in the Figure 2.4.

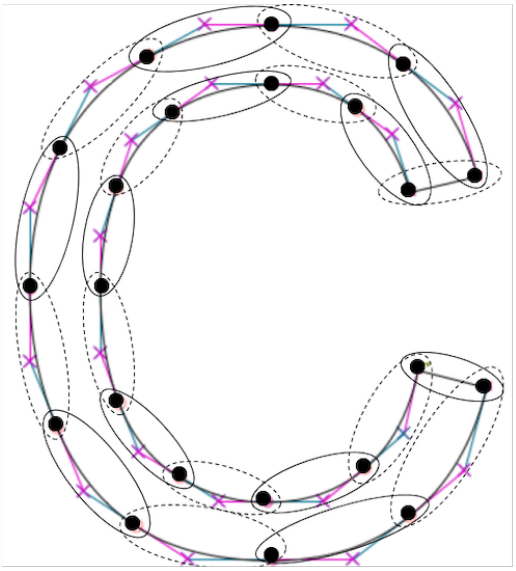


Figure 2.4: An example of "C" character in SVG format with parsed coordinates

The black points in the Figure 2.4 represent the start and the endpoints of each segment incorporating the character; the curve segments contain control points, which are marked as "×" sign. Each segment is surrounded by the elapse; segments are overlapping.

The character from the Figure 2.4 consists of an unnecessarily large amount of Bézier curve segments. For comparison, the "C" character from the Figure 2.4 could be represented in such a way that the numerous segments could be replaced by as little as one cubic Bézier curve. It is usually required for the font developers to include numerous Bézier curves segments in order to resolve designing issues and accomplish a superior visual result. The proposed multi goal-path planning method, however, connects segments by distance function based on Dubins maneuvers. The high number of segments arranged by regular fonts would require an arrangement of additional Dubins paths connecting the segments. It could lead to an extension of the final trajectory. For the multi-goal path planning, the font which can describe the characters using several segments or a reasonable number of segments is preferable.

To study the effect of the wide strokes on multi-goal path planning more deeply, we observed another example. The character "B" presented in the Figure 2.5, its coordinates marked as red points and captured with the use of "svgpathtool".

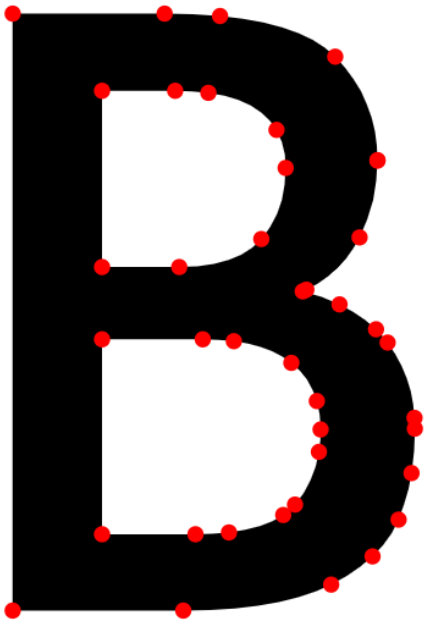


Figure 2.5: An example of "B" character in SVG format with parsed coordinates.

The red points in Figure 2.5 are the start and the endpoints of segments. It can be seen that the stroke width of the character segments differs from each other. The spray writing of the character, which includes segments of different width, would require changing the height of the UAV from the writing surface; it would change the thesis problematic into a three-dimensional problem. The height of the UAV from the surface is a constant in this thesis. We decided not to use fonts with the strokes due to the complexity of the solution.

Observations made from the characters represented in regular font format allow us to establish requirements for the font used in multi-goal path planning. The requirements could be defined as:

- The font should not include strokes.
- The font should use only necessary amount of segments.
- Each character of the font should be originally captured in SVG format or converted to SVG format.

An attempt to find a font that fits all listed requirements was unsuccessful due to the high complexity of regular fonts. The rights for the application of specific fonts are also a subject of concern. Therefore, we decided to design a new font suitable for the thesis task.

2.5 Design of font

This section discusses the process of the font design, tools, and data structures being used to store the font and its application in implemented spray writing planning method. The challenge of the font design is to establish a new variate of existing character. Font design combining numerous concepts and principals (e.g., Horizontal weight, width regularity, variability). Essential characteristics and rules are applied in the thesis, but exceptional ones are avoided due to strokes nonexistence. The font requirements arranged earlier in the chapter were followed in the designing process.

There exists a professional solution widely used by the type developers known as "FontForge". The environment enables us to create a font in TTF or OPT formats and export its characters and glyphs to SVG format, which allows further processing in planning method implementation. "FontForge" provides a table for the comfortable allocation of characters. The table is shown in the figure 2.6.

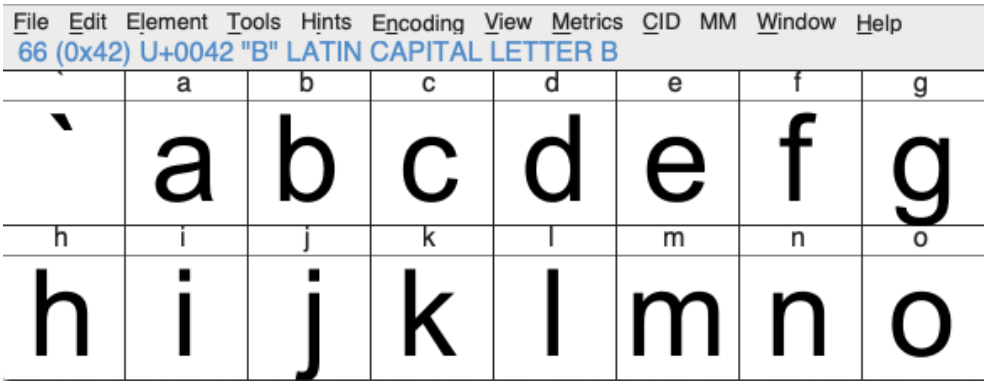


Figure 2.6: Table of characters.

Each character in the font has its encapsulating space box, which is called the *em Square* [18]. The *em Square* is a two-dimensional space where x coordinate represents a value on the horizontal axis, and y coordinate represents a value on the vertical axis. The height of the *em Square* consists of the character height plus additional space, which is needed to avoid collisions between lines of text. However, numerous font types (e.g., True Type Font, Open Type Font) are supporting the execution of the character, which is partially crossing the boundaries of the *em Square*.

Each point in a grid is described by FUnit, which is the smallest unit of measure of the *em Square*. The font developer should customize the quantity points per *em Square*. It is only natural that the specified *em Square* is not infinite on x or y-axis and must be within a certain range from -16384 to +1638 FUnits [30]. The size of the specified *em Square* is a constant which is valid for every outline in the font. An example of designed "T" letter encapsulated by the typical *em Square* is shown in the Figure 2.7.

An essential principle in the type design is to create a set of characters such that every character inherits the common style. The concept could be accomplished by using

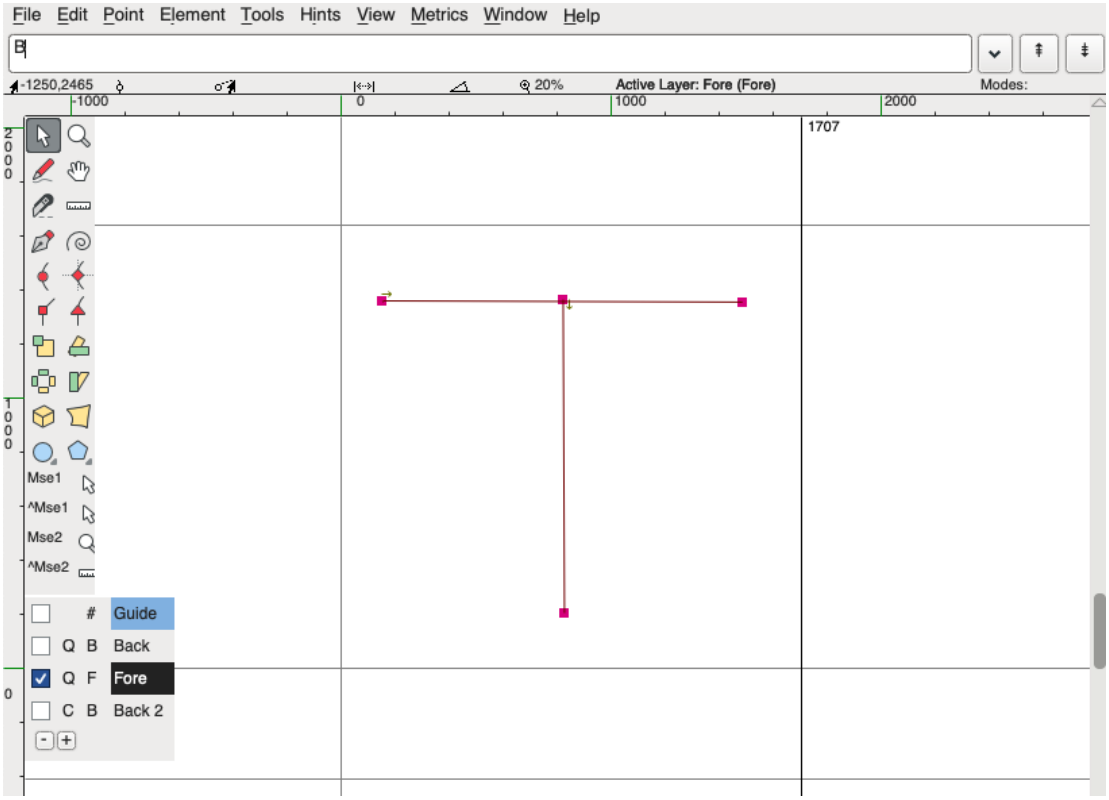


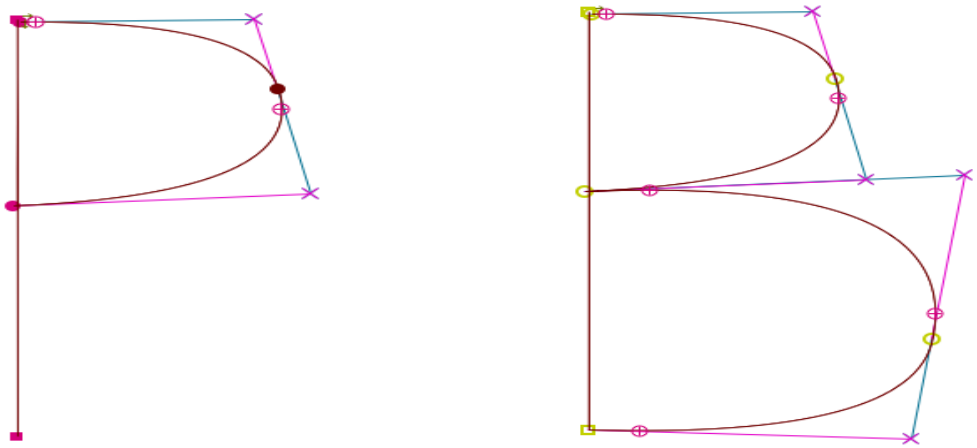
Figure 2.7: Example of letter in the *em Square*.

similar segments(e.g., curves, lines, and arcs) in different characters. The letters "B" and "P" composed of the identical curve and the line. The designed letters are shown in the Figure 2.8.

In the case of the thesis, the designed font must be functional for multi-goal path planning. It could be seen in the Figure 2.8 that the letters dose not include strokes and consist only of the necessary amount of segments.

The designing process is followed by parsing and storing the characters. The parsing of points (e.g., start, end, control) is accomplished by "svgpathtool" tool for Python 3 programming language. Then storing the parsed points in the Python application is performed by hashmap data structure. An example of the representation of the character in the Python 3 program is shown in the Listing 2.2.

```
start_end_points = {
    'A': [[[412.0, 672.0], [860.0, 672.0]], [[128.0, 0.0], [644.0,
   1248.0]], [[644.001, 1248.001], [1112.0, 0.0]]],
    'C': [[[807.0 , 81.0], [759.0 , 1248.0]]],
    }
control_points = {
    ((807.0 , 81.0), (759.0 , 1248.0)): [[80.0 , 90.0], [155 , 1215]],}
```



(a) “P” glyph (b) “B” glyph

Figure 2.8: The letters “B” and “P” composed of the identical curve and the line.

Listing 2.2: The characters representation in Python application

The key of “start end points” dictionary is a letter, the value is an array of segments which consists of two points. The “control points” dictionary include a segment as a key and the segment control points as a value.

Chapter 3

Proposed multi-goal path planning method

Contents

This chapter describes the proposed multi-goal path planning method for the spray writing with UAV. It discusses the approaches and algorithms employed by the method. The path planning method in the thesis is addressed with the approach where the sequence of visiting segments constituting the input text is determined before generating a continuous trajectory. The problem is formulated as the Generalized Traveling Salesman Problem (GTSP) [13, 31], which divides nodes into clusters and visits each cluster exactly once. However, the ordinary GTSP uses the Euclidean trajectory composed of only straight line segments connecting the nodes. The hexarotor UAV, which has maximal acceleration and velocity constraints, is considered in this thesis. It would be problematic for the UAV to travel along the Euclidean trajectory without violating the constraints or missing the nodes' positions. In contrast to the Euclidean solution, we propose a modified distance function, which is based on Dubins maneuvers. The function connects nodes earlier established by the GTSP sequential order and considers the constraints of the Dubins model. The MRS team [6] developed the considered UAV, which is equipped with MPC trajectory following controller. It accepts the trajectory in the form of samples (i.e., set of points) [32]. The Diagram of the components incorporating the proposed motion planning method is shown in the Figure 3.1.

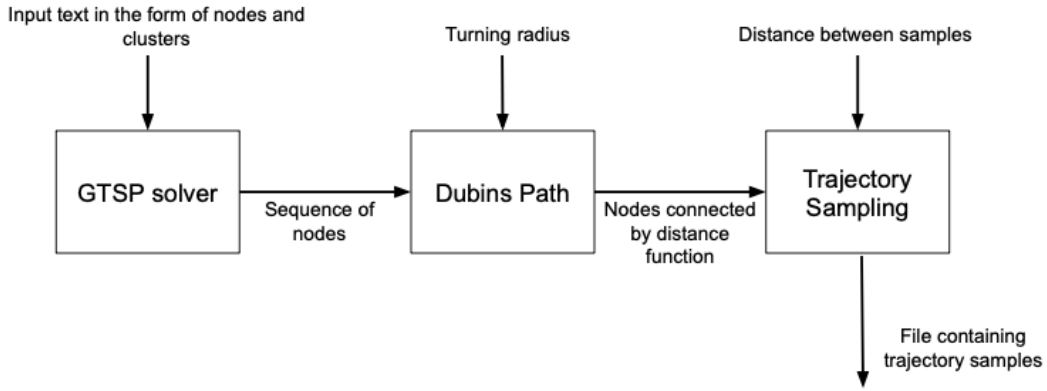


Figure 3.1: The diagram presents components and their input parameters constituting the multi-goal path planning method.

3.1 Determining a sequence of character segments

Finding the close to optimal sequence of visiting segments is based on the GTSP solution. It is hard to speak about GTSP without mentioning the underlying TSP problem [14] where the salesman aims to visit every node in the given set precisely once. The distance between cities is initially known. The TSP determines the shortest tour, which minimizes the total distance of the path. The largest problem optimally solvable by Concorde algorithm [33] includes 85,900 cities. The problem can be addressed with numerous heuristics, which allows us to find solutions close to optima or the optimal one.

The TSP problem is widely used in UAV data collection planning since the problem has a similar formulation to the TSP (i.e., the given set of target locations has to be visited by the UAV); thus, the problem has the same formulation as the TSP. In the case of the thesis, the UAV has to visit each segment of the input text (e.g., line, curve), and the direction of traversing individual segments represent the node. However, the final trajectory has to visit only one node (i.e., direction) of each segment. We employed the Generalized variant of TSP problem, which allows to group nodes into clusters and visit precisely one node in each cluster. An example of the tour between clusters is shown in Figure 3.2.

The character segments are formed from two points (e.g., start, end), consequently the segment could be entered by the UAV from both points. Two directions of traversing the segment could be represented as two points in two different orders. Then, the sequences or directions of the segments represent nodes incorporating individual clusters. The labeled points of "H" character are shown in the Figure 3.3; the points constituting the nodes and clusters are presented in the Table 3.1.

Each point constituting the character "H" in the Figure 3.3 is marked with a label. The character consists of three straight line segments incorporating the points (e.g., (1,3), (2,5), and (4,6)). It is shown in the Table 3.1 that each segment from the Figure 3.3 could

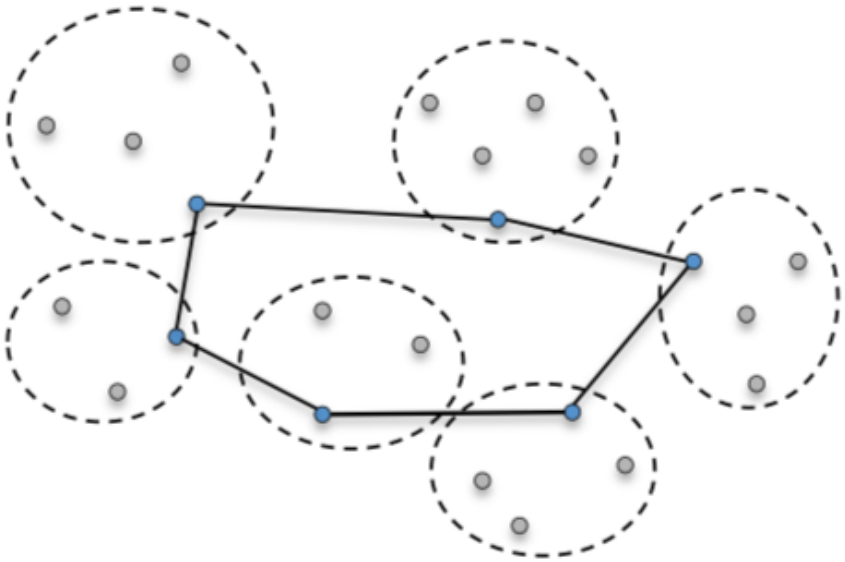


Figure 3.2: An example of GTSP euclidean tour. The source: [4]

Table 3.1: The table of points constituting nodes and clusters.

Cluster	Node (start point, end point)	Node (start point, end point)
1	a(1,3)	b(3,1)
2	c(2,5)	d(5,2)
3	e(6,4)	f(4,6)

be executed in two sequences (e.g., (1,3), (3,1)), those sequences or directions represent nodes (e.g., a, b). Moreover, two nodes of the individual segment are forming a cluster. The GTSP problem requires the distance between all possible pairs of nodes in two-dimensional Cartesian coordinates. Since each node consists of two points, the distance can be primitively calculated between the endpoint of the current node and the start point of the next node. For example, distance between nodes "a" and "c" from the Table 3.1 has to be determined between "a" endpoint - "3" and "c" start point - "2". The distance could be calculated with the use of the formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{3.1}$$

The distance matrix of nodes from the Table 3.1 is illustrated in the Table 3.2. The first column and line consists of nodes. The values are the points of the corresponding pair of nodes between which the distance has to be calculated. Those points are the horizontal plane node endpoint and the vertical plane node start point.

The final sequence of nodes is obtained from the solver based on Lin-Kernighan-Helsgaun (LKH) Algorithm. The LKH solver has two input files: 1) problem file in GT-

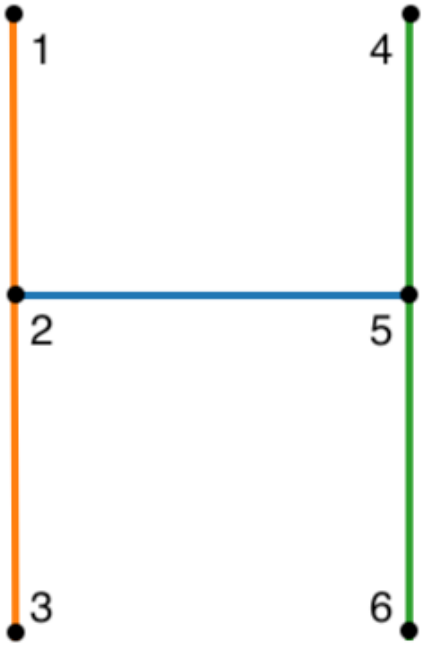


Figure 3.3: The "H" character with labeled points.

Table 3.2: The distance matrix between nodes of the string "H".

Node	a	b	c	d	e
a	(3,1)	(1,1)	(5,1)	(2,1)	(4,1)
b	(3,3)	(1,3)	(5,3)	(2,3)	(4,3)
c	(3,2)	(1,2)	(5,2)	(2,2)	(4,2)
d	(3,5)	(1,5)	(5,5)	(2,5)	(4,5)
e	(3,6)	(1,6)	(5,6)	(2,6)	(4,6)

SPLIB format containing established set of clusters and distance matrix between nodes, 2) parameter file that consists of the name of problem file and parameters which control the solution. The Python program generates the files. The solver transforms the GTSP problem to asymmetric TSP by Noon Bean transformation and provides a solution determined with the use of LKH heuristic function. The LKH solver output is a file that contains a sequence of nodes and length of the Euclidean trajectory. For clarification, the sequence of nodes is visualized in the Figure 3.4.

In the Figure 3.4, nodes are marked as arrows and labeled with its periodical number

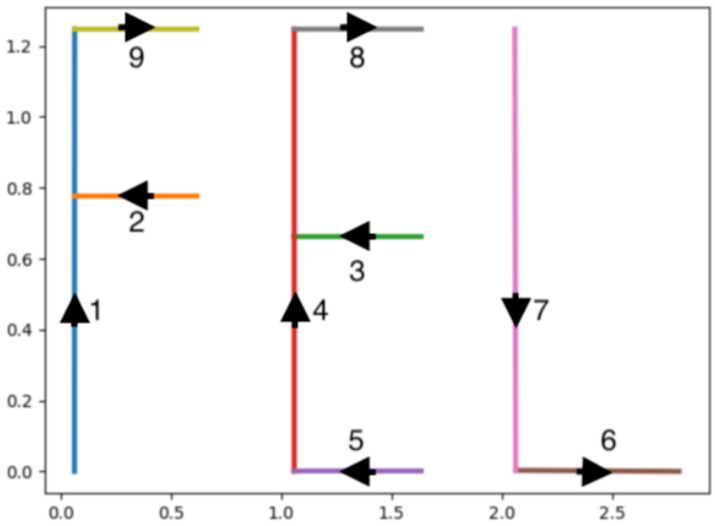


Figure 3.4: The sequence of nodes of the string "FEL".

in the earlier established sequence by the LKH solver. For example, node "1" is first in the course; thus, it is first to be visited by the UAV.

3.2 Distance function

We proposed to substitute Euclidean distance between nodes by length of shortest Dubins path. The solution proposed by Dubins introduces the new type of planer curves which could be executed by curvature-constrained vehicle. The state of the vehicle is defined as $q = (x, y, \theta)$; it consists of position coordinates on the plane (x, y), and the heading angle θ. The non-holonomic vehicle model has a minimal turning radius denoted as p, which affects the length of the Dubins path. The Dubins model could be described as:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{u}{p} \end{bmatrix}, u \in [1, 1]. \tag{3.2}$$

The Formula (3.2) includes a constant forward velocity v and a control input u. It was proved that the shortest path for the Dubins model (3.2) should consist of at most three segments, such as line (denoted as S) and arcs (left turn - L, right turn - R) [15]. The optimal path is an element from the set of possible maneuvers LSL, LSR, RSL, RSR, LRL, RLR. An example of the RSR Dubins path between two points is shown in the Figure 3.5, where arrows indicate the heading angle of the vehicle.

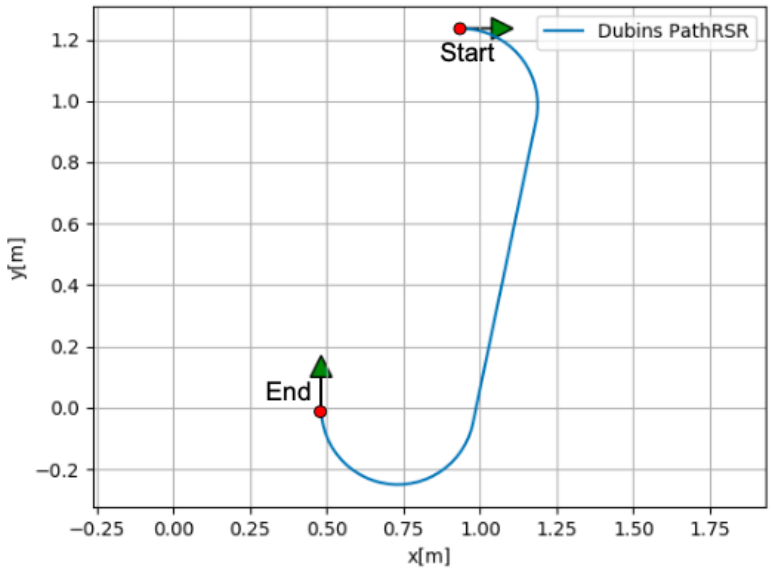


Figure 3.5: Dubins path between two points with marked heading angles.

The heading angles of the nodes are fixed since it is dependent on the topology of the segment locations. Since each node in the established GTSP problem is entered and exited by the UAV exactly once, the heading parameter has to be determined for each point constituting individual node. The headings of individual nodes depend on the direction of the node. There are two types of nodes, such as line and curve. The θ parameter of nodes of both types could be calculated with the use of the Formula (3.3).

$$\theta = \arctan\left(\frac{y_1 - y_0}{x_1 - x_0}\right) \tag{3.3}$$

The θ of entry point of a line node could be calculated by substituting the node exit point as $[x_1, y_1]$ and entry point as $[x_0, y_0]$ into the Formula (3.3). The letter "T" in the Figure 3.6 provides an illustration of the headings of nodes. The character in the Figure 3.6 consists of two nodes marked with blue arrows and labeled as "a" and "b", nodes are composed of points "P1", "P2", "P3", "P4", each point has its θ which is labelled as "θ1", "θ2", "θ3", "θ4". For example, "θ1" could be calculated by applying "P2" as $[x_1, y_1]$ and "P1" as $[x_0, y_0]$ into the Formula (3.3). If the node "b" would be pointing in the opposite direction (i.e., "P2" is the entry point, "P1" is the exit point), both headings (i.e., "θ1", "θ"2) have to be reversed as well. In this case, "θ1" could be calculated by applying "P1" as $[x_1, y_1]$ and "P2" as $[x_0, y_0]$ into the Formula (3.3).

The curve nodes that use Cubic Bézier curves include start, end, and two control points. The θ parameter of the node point could be calculated by substituting the point and the control point into the Formula (3.3). Letter "C" displayed in the Figure 3.7 consists

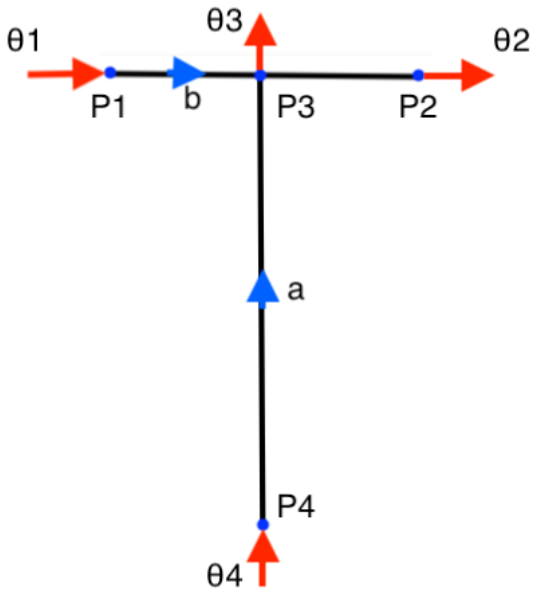


Figure 3.6: Two line nodes "a" and "b" with marked control points and heading angles.

of "P1", "P2" points, and control points "C1" and "C2". For example, "$\theta 1$" could be calculated by substituting "P1" as $[x_1, y_1]$ and "C1" as $[x_0, y_0]$ into the Formula (3.3). Another node of the "C" latter's cluster is pointing in the opposite direction; thus, its headings are reversed accordingly (i.e., "P1" is the entry point, "P2" is the exit point).

Since the Dubins path is widely used in motion planning, there are numerous implementations in various programming languages. In this thesis, we used mainly the Python language. We employed a proper quality implementation of the Dubins path in the form of Python library from [34]. It allows us to specify the desired turning radius. The output of the "dubins" library is sampled Dubins path between two points. The Dubins path then connects nodes in sequential order established by the GTSP solver, which constitutes a smooth trajectory. The examples of the final trajectory with $0.3(m)$ turning radius are presented in the Figure 3.8 and 3.9.

3.3 Trajectory sampling

The considered UAV in the thesis is equipped with MPC trajectory following controller [32], which takes the trajectory in the form of a file containing samples. The MPC receives the set of points and attempts to execute it with the equal time gap. The time between samples is predefined as $t = 0.2s$. The time distance between samples selects the speed of the vehicle. Longer distance between samples would mean higher velocity of the UAV. Equidistant sampling allows the UAV to traverse the trajectory with a constant ve-

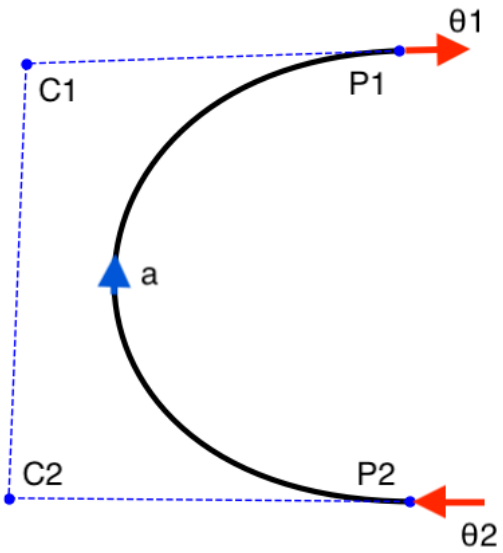


Figure 3.7: Curve node "a" with marked control points and heading angles.

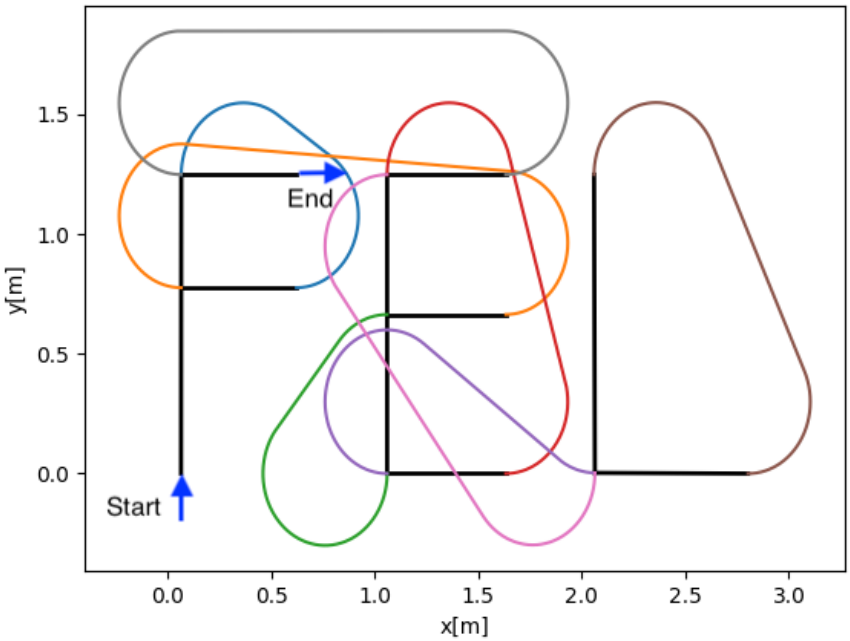


Figure 3.8: The final trajectory with the input text "FEL" composed of Dubins path connecting nodes. Blue arrows indicate the start point and endpoint of the trajectory.

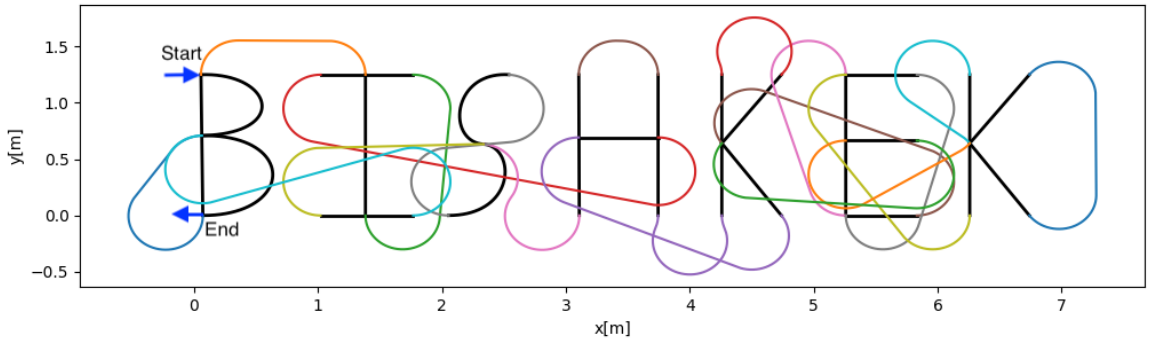


Figure 3.9: The final trajectory composed of nodes connected by Dubins path.

locity. We experimentally determined that arc length $s = 0.1m$ between samples relatively good describes geometrical features of the input text letters in the selected size, and we adopted this distance between samples for every trajectory. The speed of the vehicle inside the interval could be primitively calculated with the Formula (3.4).

$$v = \frac{s}{t} \tag{3.4}$$

The input text can be defined as the set of points on the text segments. The Dubins path between the input text segments is calculated using the "dubins" Python library as a set of equidistant samples. The letter segments and Dubins path are thus independent fragments in the sequence. The proposed naive approach is to sample each fragment independently with the distance between points $0.001m$, then gather obtained points into a set and determine the equidistant points from the set. This allows us to change the distance between samples and provide control over the UAV's velocity during trajectory tracking. In the Figure 3.10, equidistant samples are marked as green points.

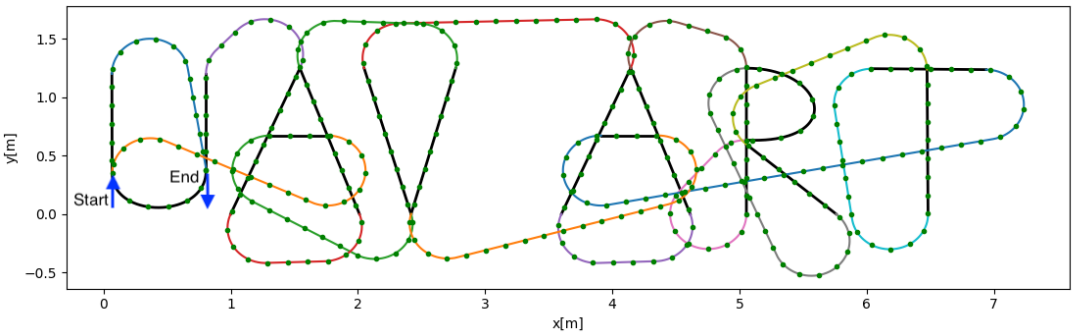


Figure 3.10: Example of the trajectory with equidistant samples, which are marked as green dots. The start point and endpoint of the trajectory are marked with blue arrows.

Chapter 4

Results

A set of experiments presented in this chapter is aimed to show the established trajectory suitability for the UAV traversing. The chapter also provides the GTSP solver performance benchmark with different instances. The system developed by MRS team, based on ROS and Gazebo simulation [35], is used for testing. The proposed motion planning method accepts turning radius of Dubins maneuvers parameter which could affect the execution of the trajectory by the UAV. A relatively low turning radius can cause missing the precise location of nodes by the UAV.

After testing numerous trajectories with different turning radius, we observed that the UAV precisely follows the trajectory with $r = 0.3m$. The first trajectory tested in simulation is shown in the Figure 4.1; Figure 4.2 is a snapshot taken while the UAV was traversing the trajectory. The UAV is marked with blue, red, and green big arrows; equidistant samples are marked with blue arrows.

For the second example, we attempted to test a trajectory composed of multiline text. The trajectory is shown in the Figure 4.3. The performance of the UAV in the simulation environment is presented in the Figure 4.4, where the UAV is marked with blue, red, and green arrows and equidistant samples are marked as blue arrows.

The Table 4.1 provides a benchmark of trajectories from the Figure 4.1 denoted as "1" and the Figure 4.2 denoted as "2". In the Table 4.1, the solution time of the GTSP solver is labeled as t_g.

Table 4.1: The table of points constituting nodes and clusters.

Trajectory	Number of segments	Number of nodes	$t_g(sec)$	Trajectory length (m)
1	36	72	3.7	47.7
2	32	64	4.1	48.7

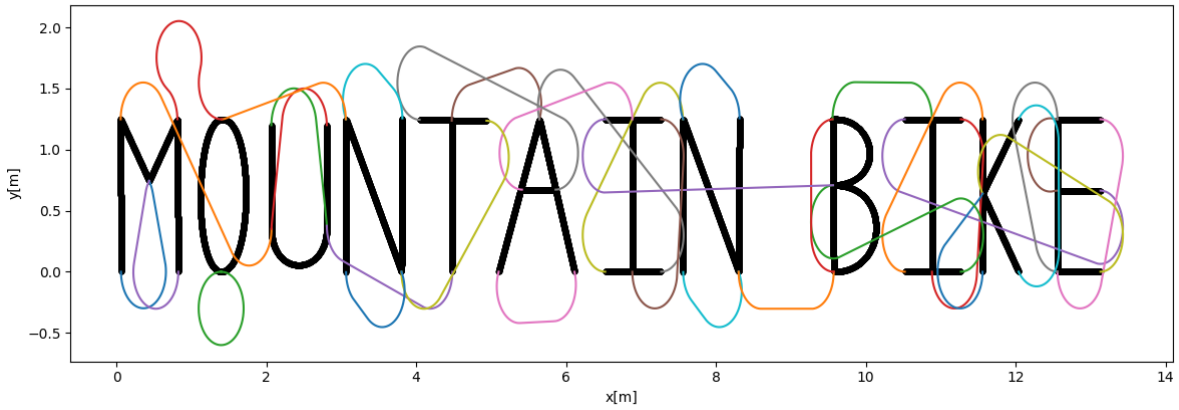


Figure 4.1: Trajectory established by the proposed method with input text "MOUNTAIN BIKE".

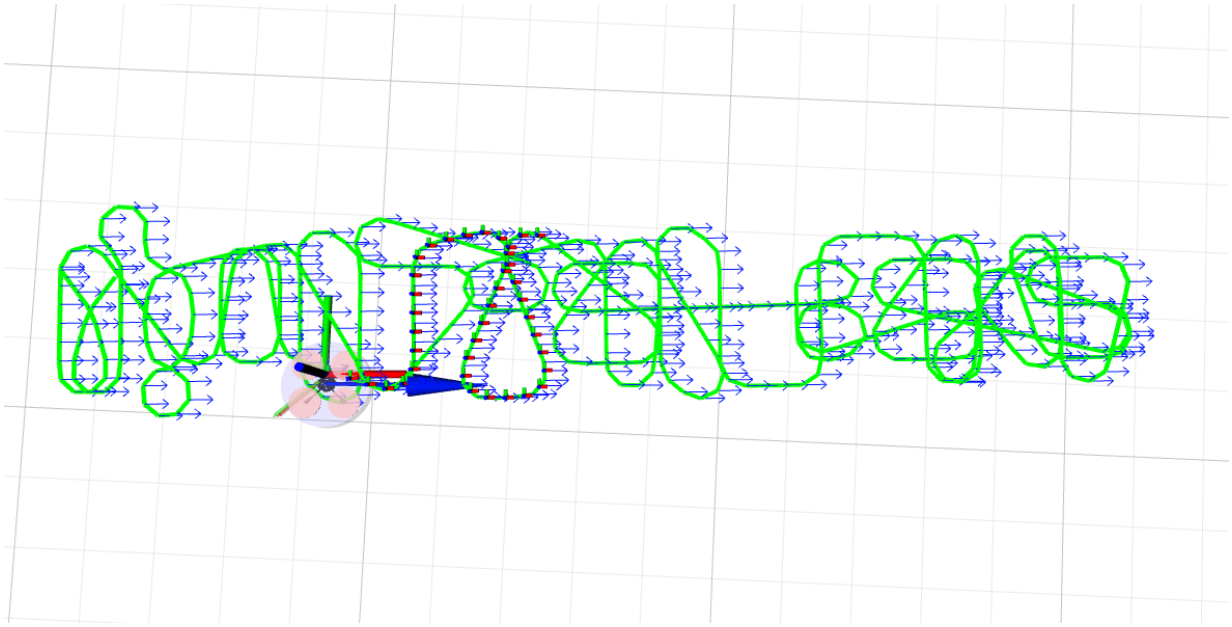


Figure 4.2: Snapshot of the UAV traversing the trajectory from the Figure 4.1 in the simulation.

The set of experiments showed that the UAV precisely followed the trajectories established by the proposed motion planning method. The video of the simulation will be presented during the defense of the thesis.

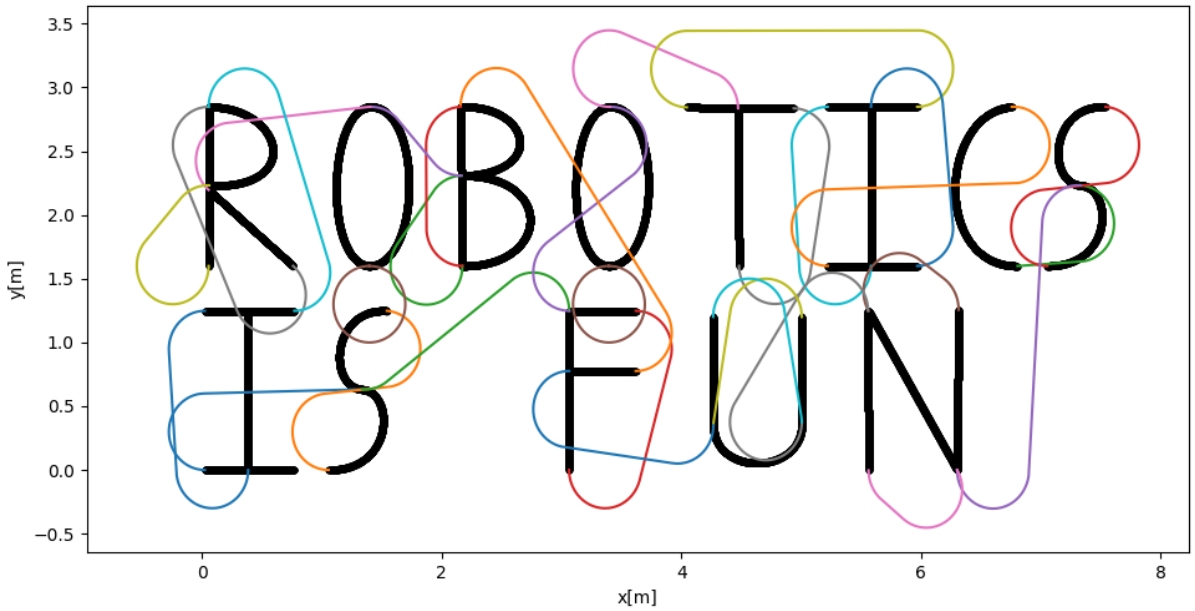


Figure 4.3: Trajectory established by the proposed method incorporating multiline text "ROBOTICS IS FUN".

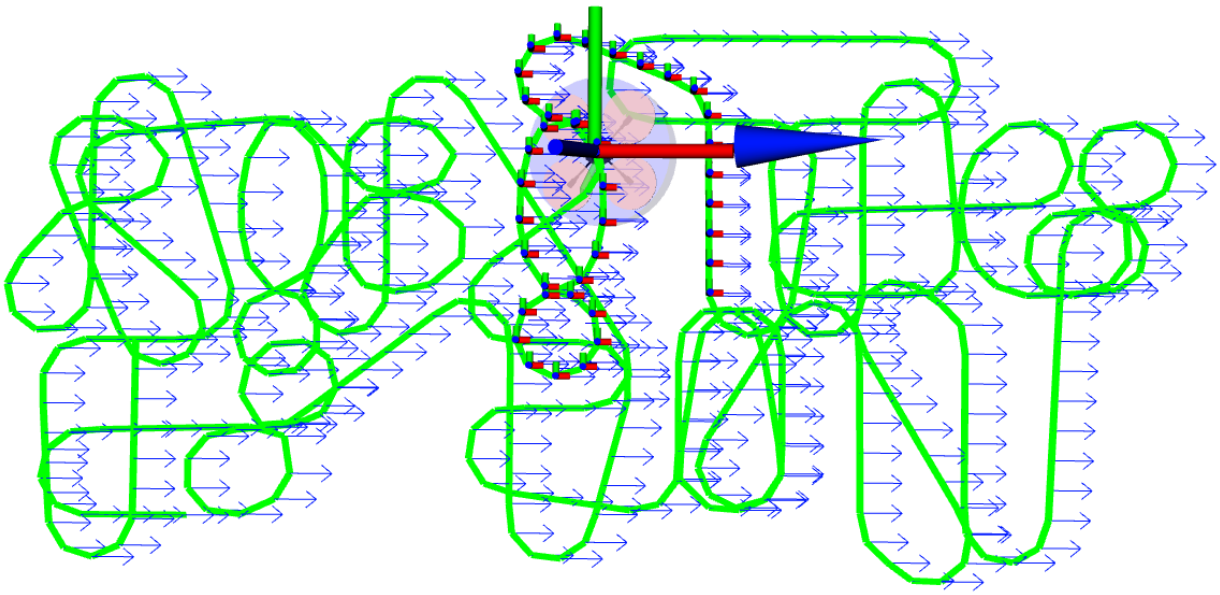


Figure 4.4: Snapshot of the UAV traversing the multiline text trajectory from the Figure 4.3 in the simulation environment.

Chapter 5

Conclusion

Spray painting is a relatively new application for the Unmanned Aerial Vehicles. In some cases, painting with the UAV could achieve better graphical results than painting with classical methods. This was a motivation for this thesis. We proposed a novel planning method for spray writing with UAVs. The thesis incorporates following sub-tasks:

- Font Design for the task of multi-goal path planning.
- Development of the path planning method for spray writing.
- Testing the performance of the proposed planning method in the simulation environment.

The focus of the thesis is specifically on writing with the use of the UAV. In order to find a suitable font for multi-goal path planning, we researched different font types and their geometrical representations. The font requirements for multi-goal path planning have been established in the Chapter 2. Unfortunately, an attempt to find a font which satisfies the requirements was not successful. Therefore, a new font for multi-goal path planning has been designed in Section 2.5.

The proposed multi-goal path planning method involves tasks of determining close to optimal sequence of visiting the input text segments and connecting the segments with the distance function, which considers the Dubins model. The problem of finding the sequence is addressed with the GTSP formulation. The final trajectory uses a constant speed of the UAV without braking maneuvers, which is an advantage of the method. Chapter 3 contains a rigorous description of the proposed method.

The continuous trajectories established by the method have been tested in ROS/Gazebo simulation environment. Overall experiments showed that the proposed motion planning method is applicable for the real UAV. The results are described in the Chapter 2.

The method is feasible for drawing geometrical shapes since the letters contain similar segments (e.g., lines, curves). The sophisticated graphical images could consist of numerous geometrical shapes filled with color. Future work could be focused on extending the existing method with an approach of painting inside individual elements. Developing the device to deliver paint from UAV to the surface would also be a good improvement.

Bibliography

[1] Czech Technical University. "uav developed by mrs team.". Cited on: 10.04.2020. [Online]. Available: http://mrs.felk.cvut.cz

[2] Morgan W. Movable type. Cited on: 14.04.2020. [Online]. Available: https://www.thinglink.com/scene/781507739854569473

[3] user:Krokofant. Letter in bitmap font. Cited on: 11.04.2020. [Online]. Available: https://cs.wikipedia.org/wiki/Soubor:Bitmapfont.png

[4] K. Helsgaun. (2014, 05) Solving the equality generalized traveling salesman problem using the lin-kernighan-helsgaun algorithm. [Online]. Available: http://akira.ruc.dk/~keld/research/GLKH/GLKH_Report.pdf

[5] T. J. Q. Julian Tan Kok Ping, Ang Eng Ling and C. Y. Dat, "Generic unmanned aerial vehicle (uav) for civilian application-a feasibility assessment and market survey on civilian application for aerial imaging," *IEEE Conference on Sustainable Utilization and Development in Engineering and Technology*, pp. 289–294, 2012.

[6] V. Spurný, T. Báča, M. Saska, R. Pěnička, T. Krajník, J. Thomas, D. Thakur, G. Loianno, and V. Kumar, "Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles," *Journal of Field Robotics*, 10 2018.

[7] S. Chen, D. Laefer, and E. Mangina, "State of technology review of civilian uavs," *Recent Patents on Engineering*, vol. 10, pp. 1–1, 07 2016.

[8] C. Yuan, Z. Liu, and Y. Zhang, "Fire detection using infrared images for uav-based forest fire surveillance," *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 567–572, 2017.

[9] J. Gonçalves and H. Renato, "Uav photogrammetry for topographic monitoring of coastal areas," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 104, 06 2015.

[10] T.-Y. Chou, M.-L. Yeh, Y. C. Chen, and Y. H. Chen, "Disaster monitoring and management by the unmanned aerial vehicle technology," 2010.

[11] A. S. Vempati, M. Kamel, N. Stilinovic, Q. Zhang, D. Reusser, I. Sa, J. Nieto, R. Siegwart, and P. Beardsley, "Paintcopter: An autonomous uav for spray painting on three-dimensional surfaces," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2862–2869, 2018.

[12] B. Galea and P. G. Kry, "Tethered flight control of a small quadrotor robot for stippling," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1713–1718, 2017.

[13] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the lin-kernighan-helsgaun algorithm," *Mathematical Programming Computation*, vol. 7, 05 2014.

[14] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *IEEE robotics & automation magazine*, vol. 17, no. 4, pp. 70–77, 2010.

[15] 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, pp. 497–516, 1957.

[16] Y. Haralambous, "Fonts and encodings," *O'Reilly Media*, p. 1040, 2009. [Online]. Available: http://shop.oreilly.com/product/9780596102425.do

[17] B. John, C. Milt, C. Richard, D. David, D. Andrew, and D. David, "Scalable vector graphics (svg) 1.0 specification," *W3C*, 1999. [Online]. Available: http://www.w3.org/TR/1999/12/WD-SVG-19991203/

[18] FontForge authors. (2020) Design with fontforge. Cited on: 10.04.2020. [Online]. Available: http://designwithfontforge.com

[19] E. N. Charles and C. B. James, "An efficient transformation of the generalized traveling salesman problem,," *Operations Research*, vol. 21, no. 10.1080/03155986.1993.11732212, pp. 39–44, 1993.

[20] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, pp. 106–130, 10 2000.

[21] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

[22] A. S. Jeanne and T. Guoqing, "Shortest paths for the reeds-shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control. 1," *Rutgers Center for Systems and Control Technical Report*, vol. 10, pp. 1–71, 1991.

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

[24] P. Váňa and J. Faigl, "On the dubins traveling salesman problem with neighborhoods," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4029–4034, 2015.

[25] J. Faigl, "Self-organizing map for orienteering problem with dubins vehicle," *International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*, pp. 1–8, 2017.

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

[27] J. Faigl, R. Pěnička, and G. Best, "Self-organizing map-based solution for the orienteering problem with neighborhoods," *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 001 315–001 321, 2016.

[28] G. M. Satyanarayana, R. Sivakumar, C. David, and G. Eloy, "Tightly bounding the shortest dubins paths through a sequence of points." *Journal of Intelligent Robotic Systems,*, no. 1007/s10846-016-0459-4,, pp. 1–17,, 2017.

[29] B.-R. Amelia, B. Bogdan, L. Chris, and W. Eric, "Scalable vector graphics(svg) 2," *W3C*, 2018, cited on: 11.04.2020. [Online]. Available: https://www.w3.org/TR/SVG/Overview.html

[30] "Microsoft typography documentation," cited on: 10.04.2020. [Online]. Available: https://docs.microsoft.com/en-gb/typography/

[31] M. Fischetti, J. J. S. González, and P. Toth, "The symmetric generalized traveling salesman polytope," *Networks*, vol. 26, no. 2, pp. 113–123, 1995.

[32] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6753–6760, 2018.

[33] D. L. Applegate, R. E. Bixby, V. Chvátal, W. Cook, D. G. Espinoza, M. Goycoolea, and K. Helsgaun, "Certification of an optimal tsp tour through 85,900 cities," *Operations Research Letters*, vol. 37, no. 1, pp. 11–15, 2009.

[34] A. Walker *et al.*, "Hard real-time motion planning for autonomous vehicles," *Hard Real-Time Motion Planning for Autonomous Vehicles PhD thesis, Swinburne University*, 2011.

[35] C. Bernardeschi, A. Fagiolini, M. Palmieri, G. Scrima, and F. Sofia, "Ros/gazebo based simulation of co-operative uavs," *Modelling and Simulation for Autonomous Systems*, pp. 321–334, 2019.

Appendices

CD Content

Table 1: CD Content

Directory name	**Description**
thesis	the thesis in pdf format
thesis_sources	latex source codes
splines	proposed method source codes
splines/get_cluster_matrix.py	takes input text and establishes a distance matrix
splines/run_TSP.py	writes files required for the GTSP solver and runs the solver
splines/DUB.py	connects nodes with Dubins path
splines/font.py	stores designed font
splines/main.py	samples the trajectory, specifies the path to the final trajectory file
splines/GLKH-1.0	GTSP solver

List of abbreviations

In Table 2 are listed abbreviations used in this thesis.

Abbreviation	Meaning
UAV	Unmanned Aerial Vehicle
2D	Two-dimensional
3D	Three-dimensional
TSP	Traveling Salesman Problem
GTSP	Generalized Traveling Salesman Problem
LKH	Lin–Kernighan–Helsgaun
TTF	True Type Font
OTF	Open Type font
SVG	Scalable Vector Graphics
ATSP	Asymmetric Traveling Salesman Problem

Table 2: Lists of abbreviations