Planning smooth paths for robots: generating arc-based paths for a differential drive vehicle

Junwei Zhang

Master of Science

School of Informatics

University of Edinburgh

2022

Abstract

In the context of smooth path planning in robot navigation, most of the current curve-based algorithms are conceptually complex and often involve high computational costs to acquire numerical solutions for complex application scenarios. Instead, sacrificing some smoothness, arc-based curves (biarcs or triarcs) could replace those complex curves (e.g., Bezier curves), as they are conceptually simpler and computationally cheap. Given by any two points (with Cartesian coordinates and tangent angles) in a two-dimensional space, valid triarc-based paths can be generated between any two configurations, while biarc-based paths cannot.

Accordingly, the objectives of this project were to develop a biarc-based path construction program and then extend it for developing a triarc-based program so that both of them can display alternative path candidates, whilst automatically selecting the optimal ones that have the lowest energy (energy: a measure of path smoothness). Both the biarcs and the triarcs construction program were developed based on a geometric approach where the relevant free control parameters for controlling the variations of path candidates were mathematically formulated from the initial predefined information. Both biarcs and triarcs construction programs were required to process different predefined information into different critical conditions.

The biarcs construction program can determine the optimal path candidates with the global minimum energy in biarc generic cases and biarc special cases, while any biarc edge cases that cannot be reached by biarcs construction need to be addressed by the triarcs construction program. Moreover, the triarcs construction program can output a globally optimal path candidate for all triarc special cases. However, due to the local range of one of the free control variables, the triarcs construction program could be less confident to determine whether an optimal path candidate could have a global minimum value of energy in triarc generic cases or triarc edge cases. Accordingly, further investigations could be performed on modifying this variable with a certain range or exploring a new one. Additionally, as the path construction programs are based on a two-dimensional space, they will need to be extended in a three-dimensional setting so that the modified program can be applied on a robotic car for further performance testing.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Junwei Zhang*)

Acknowledgements

I would like to express my deepest gratitude to my project supervisor who provided great support for me to complete my project.

Table of Contents

Chapter 1

Introduction

This chapter describes the context behind this project, starting with the motivation for this project in Section 1.1. Following the motivation, It then describes the overall aims and the corresponding objectives of this project in Section 1.2. Finally, the overall thesis structure will be presented.

1.1 Motivation

As significant progress has been made in hardware technologies and artificial intelligence [1, 2], there is an increasing demand that robots required to undertake various navigation tasks without human intervention (such as delivery [3], marine exploring [4] and rescuing missions [5], etc), which requires robots to have the ability of path planning between their start and end locations [6].

Robots are often simulated as a particle that could change their positions in any direction, and the shortest path for robots towards their destinations would be usually preferred if no additional constraints were applied to them [7]. As a result, the path-planning solutions could be often composed of sharp turns and straight portions, which indicates that the paths have discontinuous curvatures and tangents [8]. However, there could be some paths that those real robotic agents are physically impossible to follow in the aforementioned situation [9].

As shown in Figure 1.1, the car would need to decelerate or stop at these corners of the back path to handle the sharp turns, which is not desirable for the car's motion. If the agent was a robotic wheelchair taking disabled people along the path whilst avoiding obstacles [10], people could even get injured due to the sudden stops. Therefore, it would be more feasible for the intelligent agent to take the orange smooth path to its

goal without suffering from sudden stops as shown in Figure 1.1. In addition, the discontinuous paths are not desirable for real robots transporting items that need to avoid collision [11]. Accordingly, the focus of this project was on exploring tools for smooth path construction in real-time.

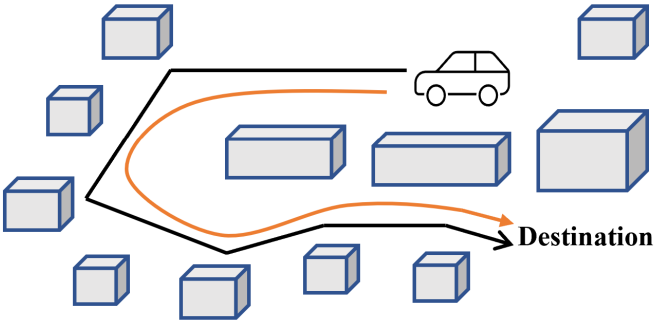


Figure 1.1: An example of path planning to avoid obstacles along a differential car's path to its destination [12]. The black path that consists of a series of corners and straight portions is assumed to be the shortest route.

Most of the current smooth-path generation techniques adopt high-derivatives curves (such as Bezier curves [13], polynomial splines [14], hodographs [15], etc.) that can construct a continuous path of high accuracy, but they are conceptually complicated [12]. With an increase in the complexity of path-planning problems (such as dynamic urban areas [16]), the path-planning problems using these techniques usually cannot be analytically solved, but require numerical solutions, which could induce relatively high computational loads [17].

However, this is not typically desirable for the robots undertaking navigation tasks as mentioned earlier, or for those equipped with limited power and computing resources, as it would be difficult for the robots' path planning module to generate sufficient path candidates for use in the selection process (e.g., selecting the shortest or the smoothest path among them) at each time step. Alternatively, there is another class of low-degree curves: circular arcs [18]. By sacrificing some smoothness, interpolating high-degree curves or paths using the arc-based method will require negligible computational loads, because the related interpolation problem can be split to find the piece-wise solution at each segment rather than considering the problem as a whole [19]. Accordingly, the motivation of this project was to explore an arc-based method for path construction that can be computationally cheap, while being smooth enough to facilitate path planning in robot navigation.

1.2 Aims and Objectives

Following the motivation of this project, the overall aim of this project was to investigate an arc-based method (biarcs [20] and triarcs [21]) for path construction between any two points in a two-dimensional (2D) space, where each point is characterised with a Cartesian position and a tangent orientation. A path is considered as valid, if the Cartesian position of its end point meets that of the goal with a correct tangent orientation. The biarcs-based method for path generation can construct valid paths for most of the pairs of starting and ending points, except for some edge cases [20], while it is assumed that the triarcs-based method can achieve valid paths generation for all the cases. Therefore, the end goal of this project was to develop programs that can construct valid paths between any two points, whilst optimising smoothness. To achieve this goal, several objectives of this project are shown below:

- Develop a program capable of constructing and displaying candidate biarc paths, whilst inspecting any edge cases that cannot be achieved by biarcs.
- By extending biarcs generation, develop a program capable of constructing and displaying candidate triarc paths, whilst ensuring any edge cases can be addressed by triarcs.
- Enable automated selection of the lowest energy candidate path from among all the alternative paths in either of biarcs or triarcs construction program.

1.3 Thesis Structure

Following Chapter 1, Chapter 2 introduces biarcs and triarcs in terms of their backgrounds and related work, and hence highlight the importance of this study. Chapter 3 describes and discusses the related geometric approaches developed for biarc construction as well as the program to implement this. Extending from the approaches used for the biarcs construction, Chapter 4 describes and discusses the related geometric methods used for triarcs construction as well as the related triarc construction program. Chapter 5 demonstrates and discusses the results of the biarc and the triarc constructions respectively. Finally, Chapter 6 will conclude all the key information that has been mentioned in the thesis, and give relevant recommendations about the future work that could be performed for the project.

Chapter 2

Background and Related Work

This chapter describes the definitions of biarcs and triarcs in Section 2.1. Following the definitions, this chapter will then evaluate the related work for biarcs (Section 2.2.1) and triarcs (Section 2.2.2) in Section 2.2 respectively. In the last Section 2.3, this chapter reflects on the key points from the literature and hence leads to the methodology that was used for the arc-based path construction for this project.

2.1 Definitions of Biarcs and Triarcs

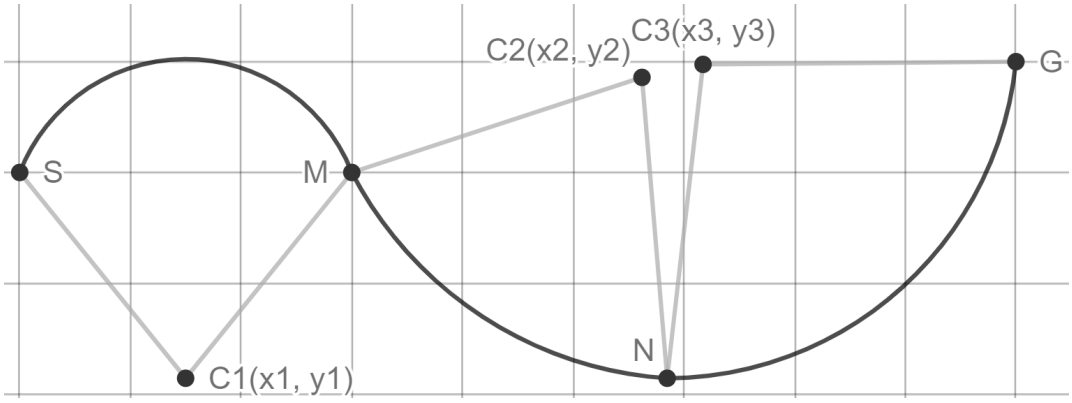


Figure 2.1: The illustration of the definitions of biarcs and triarcs

As shown in Figure 2.1, there are three single circular arcs and each arc is characterised by its arc length and its arc centre with a Cartesian coordinate (i.e., arc 1: (SM, C1), arc 2: (MN, C2) and arc 3: (NG, C3)). A biarc is a curve consisting of two arcs that join at the same tangent (e.g., the biarc SN with the joining point M or the biarc MG with the connecting point N). In contrast, a triarc is a curve consisting of three arcs and each of the two arcs shares the same tangent at their connecting points (e.g., the triarc SG that is composed of two connecting points: M and N). Accordingly, this project focused on

finding the parameters (i.e., arc length and arc centre) of each arc for the biarc or triarc candidate that has the lowest energy (energy: a measure of path smoothness) given by any two points.

2.2 Related Work Reviews

2.2.1 Biarcs

Biarcs, replacing cubic splines, were initially used as a curve-fitting tool for hull surface designs in the shipbuilding industry in 1970 [22], and one of the considerations for the replacement is that using biarcs can improve computational efficiency and hence save costs for using shipbuilding design software, while cubic splines' solutions usually involve numerical iterations that could require extra computational time. Afterward, an extensive literature has been developed on biarcs.

Biarcs have been used for approximating higher-derivative curves (such as Biezer curves [23], NURBS curves [24], B-splines [25], etc) in the CNC (computer numerical control) machining applications, as typical CNC contouring and milling machines are more capable to execute linear or circular trajectories [26]. Due to the advantage of easy computation, biarcs have also been used as an interpolation tool to specify paths for different purposes (such as computer-aided designs for manufacturing [27], initial path estimations used in path optimisations for automobiles [28], GPS data interpolation for tracking vehicle trajectories [29], trajectory planning for inverse kinematics of robotic manipulators [30], shape optimisation in reducing stress concentration [31], etc).

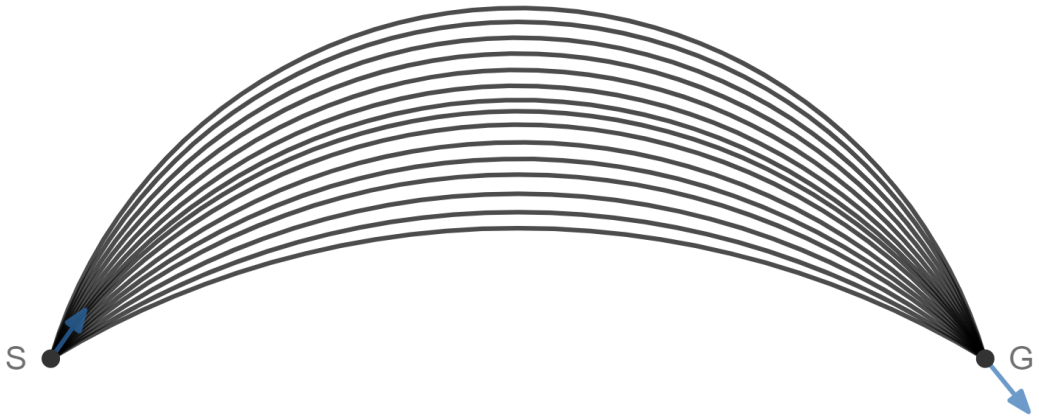


Figure 2.2: One typical example (a non-inflection case [22]) of a family of biarcs starting from the same point S and ending at the same point G

According to Figure 2.1, to determine a unique biarc between two points, the biarc problem can be solved by finding its six unknown variables (i.e., the x and y coordinates of each arc centre and the radius of each arc), but there are only five known conditions (i.e., the starting point and its tangent, the ending point and its tangent, and the common tangent of two arcs at their joint), which leaves one degree of freedom unconstrained. As a result, there will be a finite number of biarcs existing between two control points, which is considered as one family of biarcs, for example, as shown in Figure 2.2.

Various conditions have been adopted to make a unique biarc solution due to different needs (such as using minimum difference between two arcs' radii [22], middle point embedding [32], or identifying the region of the boundary of a family of biarcs [33], etc). However, this is not always the same situation for some biarcs-related optimisation problems (such as [27]), as Yang et al., aimed to determine an optimal control parameter (from a family of biarcs) that can minimise the biarcs fitting errors and hence address the related gouging issue in a CNC contouring application.

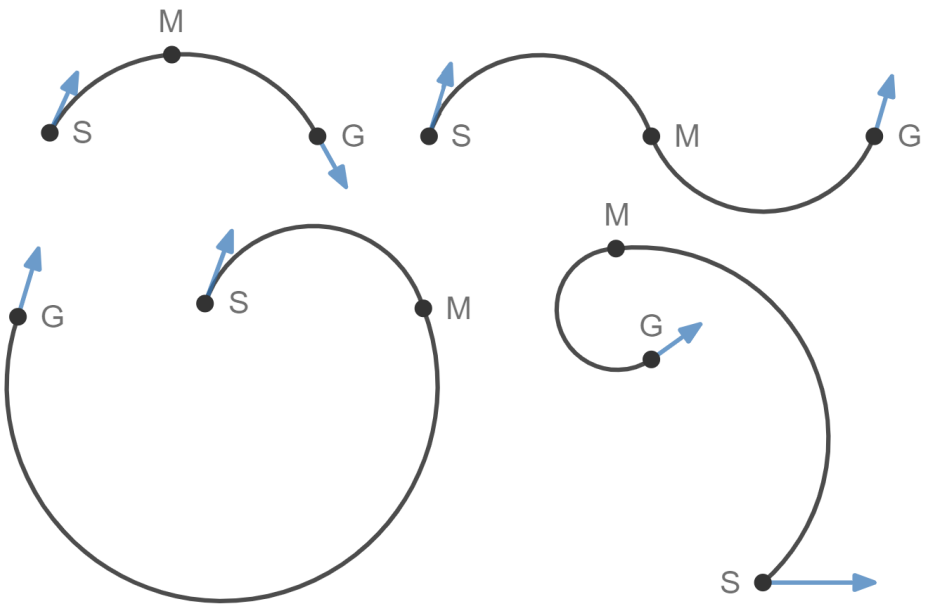


Figure 2.3: Several typical geometrical cases of biarcs: the C-shaped [34] and the S-shaped [27, 35] cases as shown in the first row, and the J-shaped ones [33] as shown in the second row. It should be noted that each biarc starts from point S, goes through point M, and ends in point G. The C-shaped and S-shaped biarcs are classified as short spirals, while the J-shaped ones are considered as long spirals [20].

Moreover, the commonly used biarcs construction approaches usually create a biarc by using 'equal chords' (equal length of the two connected arcs), or 'parallel tangent' (the tangent at the joining points is parallel to those at the starting and ending

points) [36]. In most cases, biarcs are constructed in a geometrical way (It has been verified that the connecting point of a biarc must exist on the same circle passing through its start and end points [20, 36]). However, since biarcs can be configured into different geometric shapes as shown in Figure 2.2, the current geometric-based techniques can only construct a biarc or a family of biarcs of one certain shape at a time.

Bolton [22] adopted two completely different construction methods to build biarcs of different shapes (i.e., build an S-shaped biarc with the condition of minimum difference of the radii of the two connected arcs, while constructing a C-shaped biarc with the condition that the radius of each arc is proportional to the angles between it and the known tangent vector adjacent to it). Bolton's work is aligned with another recent study conducted by Kurnosenko [20]. He proposed a theoretical framework for an entire categorisation of biarcs where a one-dimensional family of biarcs of a certain shape can be constructed based on the associated Möbius transformations, but each geometrical case would require distinct conditions for its construction [20].

Recently, Bertolazzi, instead, proposed a numerical approach that adopted the sinc function estimation and Taylor's approximations, without considering mutually exclusive geometric cases [37]. However, this biarcs construction algorithm would not stop searching for the variables that satisfy the condition of biarc formation, until the condition was met, which could be less computationally efficient compared to those geometrical techniques.

2.2.2 Triarcs

Similarly, triarcs have been also used as a tool for curve fitting and curve interpolation, but in only a few applications (e.g., curve fitting for CNC tools to machine cam profiles [38], curve fitting for planar osculating arc splines [39], aiding to evaluate the geometrical properties of airfoil leading and trailing edges [40], etc). As for triarc construction, a triarc problem would require solving nine unknown variables (i.e., x and y coordinates, and the radius of each arc) for a certain triarc solution. If only the start and the end points were given for use as constraints, there would be three free conditions left, rather than only one free condition left in a biarc solution.

To simplify the triarcs construction, one certain triarc is usually built by connecting a predefined biarc to an additional arc at the same tangent [20, 21]. Instead, without using biarcs as a connection base, Yang et al., did work on an independent geometric approach to construct a triarc by adding three more conditions to obtain a certain triarc

solution [41], however, the conditions that were used are customised for their own curve-fitting applications, which is difficult for generalisation.

2.3 Reflection

Arc-based paths (i.e., biarcs or triarcs) have existed, which are mainly used as a curve-fitting or interpolation tool in engineering designs, which is the current research trend. Besides the two end point conditions, most of the studies tend to use extra constraints to fix any extra degrees of freedom existing in arc solutions to construct unique biarcs or triarcs as required for their applications. However, this operation would not be suitable and hence not used in this project, as there is a need to find a group of candidate paths under free conditions, hence selecting an optimal path among them.

Most of the current geometrical approaches for biarc-path constructions are not generalisable, as each of them were developed and customised for a certain application that usually requires a certain shape of biarc configurations. However, since the biarc-path problem would need to leave one degree of freedom unconstrained for this project, biarc solutions with different shapes (as shown in Figure 2.3) could exist simultaneously between two end points, which indicates that the previous existing geometric-based approaches would not be feasible for use in this project. Moreover, some attempted to approximate biarc solutions with a numerical approach without splitting biarc problems into different geometric cases. Accordingly, it is possible to numerically (e.g., brute force search [42]) construct arc-based paths for this project, which would require searching for the combinations of different arc radii and arc angles.

However, when searching, both arc radii and arc angles would need to be set as continuous variables to ensure the solution accuracy so that the search space would tend to infinity, which would be not feasible. Although their incremental precision can be set properly (e.g., 0.1), there would still be a huge search space for the biarc path construction, which would cause computational inefficiency compared to previous geometric approaches. In addition, compared to any geometric approaches, this search approach could cause a lower density of valid path solutions with respect to the predefined search space, which obeys the objectives of this project. Accordingly, this project focuses on a novel geometric method for biarc path construction that would consistently involve biarc solutions of different shapes, whilst yielding a sufficiently high density of valid biarc path alternatives, and being generalisable. Constructing triarc path candidates would be extended by the biarc construction method, but in a generalisable manner.

Chapter 3

Biarc Construction

This chapter describes how a valid biarc candidate can be constructed and the way of controlling the shapes of biarcs in two different groups: standard cases and special cases through a series of calculations. Biarc candidates with almost all the different end configurations can be constructed under the standard approach (Section 3.1), while those with only two certain groups of end configurations need to be handled in two special approaches (Section 3.2). Then, this chapter describes the program design and developments for biarc generation, as well as the approach of selecting the biarc candidate with the minimum energy (Section 3.3). One edge case that cannot be achieved through either the standard approach or the special approaches is identified (Section 3.3.2), as separating those edge cases from any standard cases and special cases contributes to developing a complete program for biarc construction.

3.1 Standard Cases

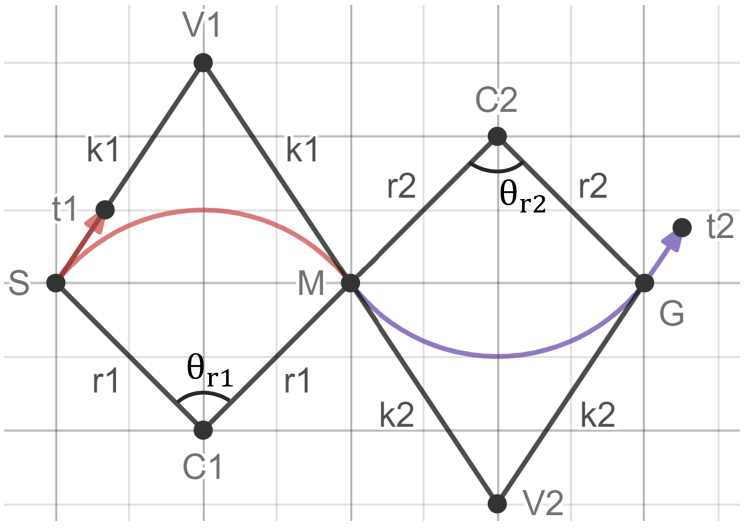


Figure 3.1: One generic example of a biarc candidate defined with various parameters. It is noted that every named point shown in this diagram has a two-dimensional Cartesian coordinate and all the relevant derivations would involve vector-based calculations.

Given by the start configuration at point S (S_x, S_y, tangent angle 1: θ_{t1}) and the end configuration at point G (G_x, G_y, tangent angle 2: θ_{t2}), the objective was able to construct different shapes of biarcs (e.g., a biarc (SG)) starting from point S with an initial tangent ($\vec{t_1}$) and ending at point G with an ending tangent ($\vec{t_2}$) as shown in Figure 3.1. The SG consists of two unique arcs (SM) and (MG), sharing the same tangent at the joint point (M). The arc SM is defined by radius (r_1) and centre (C_1), and the line extending from S along $\vec{t_1}$ intersects with the line extending from M at the point (V_1), where the magnitude of SV_1 or MV_1 is k_1. As for arc MG, it is defined by radius (r_2) and centre (C_2), and V_2 is another intersection of the line extending from G along t_2 with the line extending from M in another direction, where k_2 is the magnitude of $\vec{MV}_2$ or $\vec{GV}_2$.

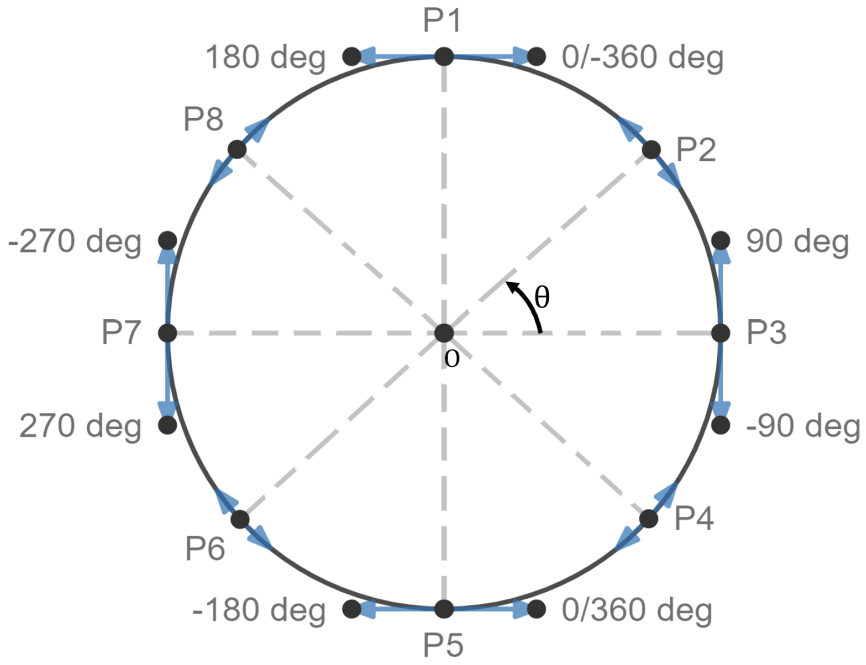


Figure 3.2: A diagram of a circular panel for use in defining any tangent or rotating angles of arcs in this project. It is noted that the panel consists of eight points from P_1 to P_8, and each point corresponds to two tangent angles of opposite directions.

According to Figure 3.2, an arc's tangent angle is defined by the tangent angle at a certain point on the circle of the centre O, while an arc's angle of rotation is defined by the angle (e.g., θ) between two points on the circle and it will be positive when rotating

in an anticlockwise direction and vice versa. Moreover, the circular panel is divided into four quadrants: quadrant 1 (from P_3 to P_1), quadrant 2 (from P_1 to P_7), quadrant 3 (from P_7 to P_5), and quadrant 4 (from P_5 to P_3). Given by any tangent angles of an arc, the position where the arc starts to rotate, can be determined. Therefore, the start point S of the biarc SG is nearly at P_8 within quadrant 2 or at P_4 within quadrant 4.

Since the two unit tangent vectors $\vec{t_1}$ and $\vec{t_2}$ would not be directly given at the beginning, they need to be obtained from the two corresponding tangent angles θ_1 and θ_2 as specified, which is formulated in (3.1.1) and (3.1.2)

$$\vec{t_1} = [\cos(\theta_{t1}) \;\; \sin(\theta_{t1})] \tag{3.1.1}$$

$$\vec{t_2} = [\cos(\theta_{t2}) \;\; \sin(\theta_{t2})] \tag{3.1.2}$$

where $\vec{t_1}$ and $\vec{t_2}$ are two unit tangent vectors of magnitude equal to 1.

3.1.1 Calculations of free variables and joint points

Since the joint point M is not a predefined parameter, its position could be varied. As M varied, either k_1 or k_2 would be changed simultaneously, which means either of the two variables can be considered as a free variable. Since only one free variable is enough to control the position of M and hence the shapes of biarcs as mentioned in Section 2.2.1, only k_1 was set as free control variable, while k_2 would be derived as a solution based on k_1 given start and end information. Therefore, the derivations of k_2 and the joint point M involving vector calculations are described below. Firstly, the two intersection points V_1 and V_2 are expressed in (3.1.3) and (3.1.4).

$$\vec{V_1} = \vec{S} + k_1\vec{t_1} \tag{3.1.3}$$

$$\vec{V_2} = \vec{G} - k_2\vec{t_2} \tag{3.1.4}$$

Moreover, to associate the given information with k_1 and k_2 , the expressions are shown below.

$$|\vec{V_2} - \vec{V_1}| = k_1 + k_2 \;\; \Rightarrow \;\; (\vec{V_2} - \vec{V_1})^2 = (k_1 + k_2)^2 \tag{3.1.5}$$

To determine the function of k_2 against k_1 and the given information, the (3.1.5) are expanded and shown in (3.1.6) (see appendix (A.0.1) for the full derivation).

$$\vec{P}\vec{P} - 2k_1\vec{P}\vec{t_1} - k_2[2\vec{P}\vec{t_2} - 2k_1(\vec{t_1}\vec{t_2} - 1)] = 0 \tag{3.1.6}$$

where $\vec{P}$ = $\vec{G}$-$\vec{S}$. Therefore, the final solution of k_2 given by k_1 and the predefined parameters is expressed in (3.1.7).

$$k_2 = \frac{\vec{P}\vec{P} - 2k_1\vec{P}\vec{t_1}}{2\vec{P}\vec{t_2} - 2k_1(\vec{t_1}\vec{t_2} - 1)} \quad \Rightarrow \quad k_2 = \frac{(\vec{G}-\vec{S})(\vec{G}-\vec{S}) - 2(\vec{G}-\vec{S})k_1\vec{t_1}}{2(\vec{G}-\vec{S})\vec{t_2} - 2k_1(\vec{t_1}\vec{t_2} - 1)} \tag{3.1.7}$$

The joint point M can then be derived given by k_1, k_2 and the known parameters as expressed in (3.1.8) (see appendix (A.0.2) for the full derivation).

$$\vec{M} = \frac{k_2}{k_1 + k_2}(\vec{S} + k_1\vec{t_1}) + \frac{k_1}{k_1 + k_2}(\vec{G} - k_2\vec{t_2}) \tag{3.1.8}$$

Based on the formula (3.1.7), given a real number for k_1, it is possible for the denominator to be equal to 0 so that k_2 would tend to infinity. In this case, the connecting point M cannot be found through (3.1.8), and this singularity problem needs to be manipulated separately.

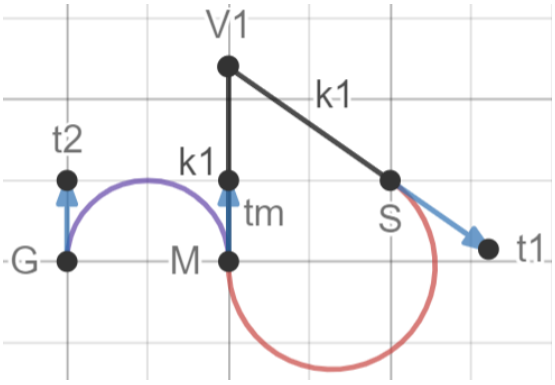


Figure 3.3: An example of a biarc candidate where k_2 tends to infinity

As shown in Figure 3.3, k_2 approaching to infinity indicates that $\vec{t_2}$ is parallel to the tangent vector $\vec{t_m}$ of M, which makes the second arc MG as a semicircle. Since the point M must be on the line extending from V_1 with the tangent $\vec{t_m}$, it can be formulated below,

$$\vec{M} = \vec{V_1} + b\vec{t_m} \;\; \& \;\; \vec{M} = \vec{V_1} + b\vec{t_2} \tag{3.1.9}$$

Where b is a scalar indicating the distance between $\vec{M}$ and $\vec{V_1}$. It is noted that $\vec{t_m}$ needs to be substituted with $\vec{t_2}$, as the former vector is parallel to $\vec{t_2}$, but it would not be given at the beginning. Moreover, since the vector $\vec{GM}$ should be perpendicular to $\vec{t_2}$, the dot product of the two vectors can be formulated below.

$$(\vec{M} - \vec{G}) \cdot \vec{t_2} = 0 \tag{3.1.10}$$

Substituting (3.1.9) into (3.1.10), the value of b can be calculated as shown in (3.1.11).

$$\begin{aligned}(\vec{M}-\vec{G})\cdot\vec{t_2}=0 \;\Rightarrow\; (\vec{V_1}+b\vec{t_2}-\vec{G})\cdot\vec{t_2}=0 \;\Rightarrow\; \vec{V_1}\vec{t_2}+b\vec{t_2}\vec{t_2}-\vec{G}\vec{t_2}=0 \\ b\vec{t_2}\vec{t_2}=(\vec{G}-\vec{V_1})\vec{t_2} \;\Rightarrow\; b=(\vec{G}-\vec{V_1})\vec{t_2}\end{aligned} \tag{3.1.11}$$

Substituting (3.1.3), (3.1.4) and (3.1.11) back into (3.1.9), the new point M for the singularity case based on Figure 3.3 can be finally obtained as shown in (3.1.12).

$$\begin{aligned}\vec{M}=\vec{V_1}+b\vec{t_2} \;\Rightarrow\; \vec{M}=\vec{V_1}+[(\vec{G}-\vec{V_1})\vec{t_2}]\vec{t_2} \\ \Rightarrow\; \vec{M}=(\vec{S}+k_1\vec{t_1})+[(\vec{G}-\vec{S}-k_1\vec{t_1})\vec{t_2}]\vec{t_2} \\ \Rightarrow\; \vec{M}=(\vec{S}+k_1\vec{t_1})+(\vec{G}\vec{t_2}-\vec{S}\vec{t_2}-k_1\vec{t_1}\vec{t_2})\vec{t_2}\end{aligned} \tag{3.1.12}$$

Given the joint point M in (3.1.8) or (3.1.12) and other relevant parameters, the radius and the centre of each arc can then be determined as shown in Section 3.1.2.

3.1.2 Calculations of arcs' radii and centres

As for arc SM, the centre C_1 is along the line SC_1 that is perpendicular to $\vec{t_1}$. To determine the position of C_1, a unit normal vector corresponding to $\vec{t_1}$ needs to be defined as shown in (3.1.13).

$$\begin{aligned}\vec{n_1}\cdot\vec{t_1}=0 \;\Rightarrow\; n_{1x}t_{1x}+n_{1y}t_{1y}=0 \\ n_{1x}=-t_{1y};\;\; n_{1y}=t_{1x} \;\Rightarrow\; \vec{n_1}=[-t_{1y}\;\; t_{1x}]\end{aligned} \tag{3.1.13}$$

Accordingly, the centre C_1 is along $\vec{n_1}$ obtained from (3.1.13) at equal distance away from S and M, which can be formulated in (3.1.14) and (3.1.15).

$$\vec{C_1}=\vec{S}+r_1\vec{n_1} \tag{3.1.14}$$

$$|\vec{S}-\vec{C_1}|=|\vec{M}-\vec{C_1}| \tag{3.1.15}$$

By expanding (3.1.15) and substituting (3.1.14) into the expanded expression of (3.1.15), r_1 can be determined as shown in (3.1.16) (see appendix (A.0.4) for the full derivation).

$$r_1=\left|\frac{(\vec{M}-\vec{S})(\vec{M}-\vec{S})}{2\vec{n_1}(\vec{M}-\vec{S})}\right|;\quad \vec{C_1}=\vec{S}+\frac{(\vec{M}-\vec{S})(\vec{M}-\vec{S})}{2\vec{n_1}(\vec{M}-\vec{S})}\vec{n_1} \tag{3.1.16}$$

Substituting r_1 into (3.1.14), the final expression of C_1 can be determined as shown in (3.1.16). Following the same approach of the derivations of r_1 and $\vec{C_1}$, r_2 and hence $\vec{C_2}$ can be determined as shown in (3.1.17).

$$r_2=\left|\frac{(\vec{M}-\vec{G})(\vec{M}-\vec{G})}{2\vec{n_2}(\vec{M}-\vec{G})}\right|;\quad \vec{C_2}=\vec{G}+\frac{(\vec{M}-\vec{G})(\vec{M}-\vec{G})}{2\vec{n_2}(\vec{M}-\vec{G})}\vec{n_2} \tag{3.1.17}$$

3.1.3 Calculations of arcs' angles of rotation

An arc's angle of rotation is considered as a vector in this project, as it contains the information about both the direction and magnitude of rotation. As shown in Figure 3.1, $\vec{\theta}_{r1}$ is the angle of rotation sweeping from the vector $\vec{C_1S}$ to the vector $\vec{C_1M}$ about C_1, while $\vec{\theta}_{r2}$ is the one sweeping from $\vec{C_2M}$ to $\vec{C_2G}$ about C_2.

$$\hat{C_1S} = \frac{\vec{S} - \vec{C_1}}{|r_1|}; \quad \hat{C_1M} = \frac{\vec{M} - \vec{C_1}}{|r_1|} \tag{3.1.18}$$

$$\hat{C_2M} = \frac{\vec{M} - \vec{C_2}}{|r_2|}; \quad \hat{C_2G} = \frac{\vec{G} - \vec{C_2}}{|r_2|} \tag{3.1.19}$$

The direction of $\vec{\theta}_{r1}$ can be determined based on the sign of the cross product of the two unit vectors $\hat{C_1S}$ and $\hat{C_1M}$, and the sign is negative, indicating that $\vec{\theta}_{r2}$ has a clockwise direction of rotation. Similarly, the direction of $\vec{\theta}_{r2}$ can be obtained by inspecting the sign of the cross product of $\hat{C_2M}$ and $\hat{C_2G}$, and the sign is positive, meaning that $\vec{\theta}_{r2}$ has an anti-clockwise direction of rotation.

$$\vec{\theta_{r1}} = -\arccos(\hat{C_1S} \cdot \hat{C_1M}); \quad \vec{\theta_{r2}} = -\arccos(\hat{C_2G} \cdot \hat{C_2M}) \tag{3.1.20}$$

Given the direction of $\vec{\theta}_{r1}$, $\vec{\theta}_{r1}$ can be obtained by taking an arccosine function to the dot product of $\hat{C_1S}$ and $\hat{C_1M}$ as shown in (3.1.20), and similarly, $\vec{\theta}_{r2}$ can be calculated through (3.1.21), given the direction of $\vec{\theta}_{r2}$. However, the aforementioned approach of calculating arc's angles of rotation can only be used for one specific group (e.g., the biarc SG composed of two short spirals as shown in Figure 3.1) where both k_1 and k_2 are greater than 0 in this case, but not appropriate for other cases (e.g., the biarc candidate as shown in Figure 3.4).

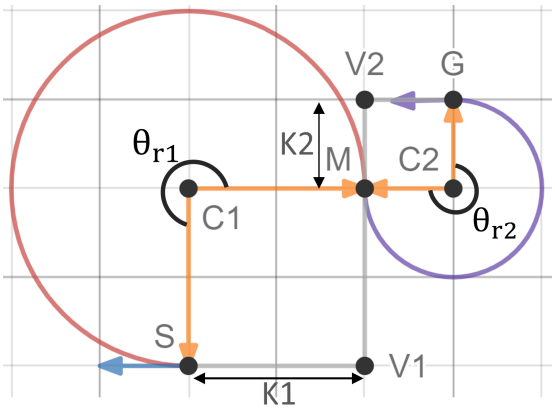


Figure 3.4: One example of a biarc candidate with negative k_1 and negative k_2

As shown in Figure 3.4, $\vec{\theta}_{r1}$ should represent the angle of rotation sweeping more than 180° from $\vec{C_1S}$ to $\vec{C_1M}$ clockwise. if taking the previous approach, the direction of

$\vec{\theta}_{r1}$ would be calculated as the anti-clockwise, while absolute value of $\vec{\theta}_{r1}$ would not be greater than 180° which is not correct. The same issue also occurred at the calculation of $\vec{\theta}_{r2}$ of this new case. Accordingly, only using the technique of cross product would not be enough to correctly determine the direction of rotation, and hence the correct angle of rotation.

In fact, a positive k (k_1 or k_2) means that the related arc is a short spiral having a rotating angle less than 180°, while a negative k indicates that the corresponding arc is a long spiral with a rotating angle greater than 180°. Therefore one more parameter (i.e., k) would need to involve in correcting the improper calculations, and hence a more complete approach is to use a piece-wise function to group $\vec{\theta}_{r1}$ at various conditions (see appendix (A.0.5)), and use another piece-wise function for the $\vec{\theta}_{r2}$ classification (see appendix (A.0.6)). The classifications of $\vec{\theta}_{r1}$ and $\vec{\theta}_{r2}$ are only applicable for the standard cases, as any cases of biarc construction corresponding to $\hat{C_1S} \times \hat{C_1G} = 0$ or $\hat{C_2M} \times \hat{C_2G} = 0$ were not be considered here and they are grouped as special cases which is described in Section 3.2.

3.1.4 Calculations of tangent angles at the joint point

Calculating the tangent angles at the joint point enables the linear interpolation between any start and end tangent angles for each arc in the subsequent program, hence generating a series of intermediate points at their corresponding tangent angles in order for the arc formation. As the joint point M is the connection between arc SM and arc MG in Figure 3.5, there are two tangent angles at this point, but they are not always equal to each other based on the rotation mechanism as shown in Figure 3.2.

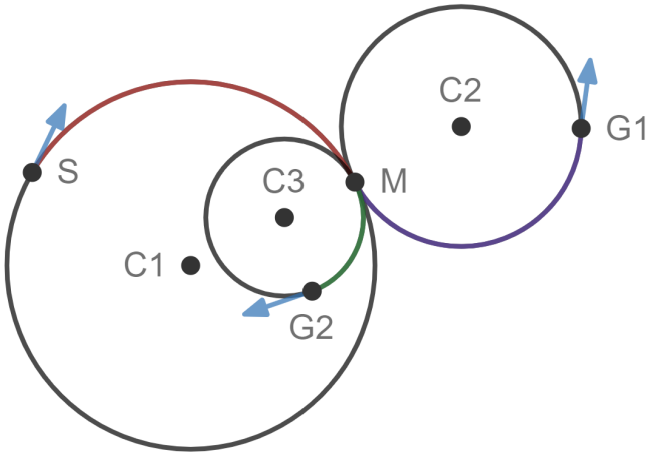


Figure 3.5: An illustration of calculating the tangent angles at the joint point M under different conditions. It should be noted that there are three arc segments: SM taken from the circle at the centre C_1, MG_1 taken from the circle at the centre C_2, and MG_2 taken from the circle at the centre C_2. The end tangent angle of the first biarc segment (e.g., arc SM) is denoted as θ_{m1}, while the start tangle angle of the second biarc segment (e.g., arc MG_1 or arc MG_2) is denoted as θ_{m2}.

As shown in Figure 3.5, there are two biarcs: the biarc SG_1 composed of SM and MG_1 connecting at M, and the biarc SG_2 composed of SM and MG_2 connecting at M. According to the initial rotating direction at S, arc SM can be considered as the curve that was obtained by a clockwise rotation starting from P_8 in quadrant 2 (Q_2) and ending at P_2 in quadrant 1 (Q_1) about C_1 based on Figure 3.2. Meanwhile, arc MG_2 was also obtained by a clockwise rotation. Therefore, as for biarc SG_2, θ_{m1} is equal to θ_{m2}.

However, since arc MG_1 was obtained due to an anticlockwise rotation, for the MG_1 construction, M would need to be changed from P_2 to P_6 in quadrant 3 (Q_3), which means θ_{m1} is not equal to θ_{m2} for SG_1. Accordingly, as for a biarc, whether its θ_{m1} could be equal to its θ_{m2} depends on the rotating directions of their $\vec{\theta}_{r1}$ and $\vec{\theta}_{r2}$ (i.e.,sign($|\vec{\theta}_{r1}|$) = sign($|\vec{\theta}_{r2}|$), indicating $\theta_{m1} = \theta_{m2}$, while sign($|\vec{\theta}_{r1}| \neq |\vec{\theta}_{r2}|$), indicating $\theta_{m1} \neq \theta_{m2}$). As for the unequal condition, the relation between θ_{m1} and θ_{m2} depends on the position of M on the rotating panel, which is formulated in appendix A (A.07).

After performing the linear interpolation given by the start and end tangent angles of each arc, each corresponding intermediate point constituting an arc can be determined, which is described in Section 3.1.5.

3.1.5 Calculations of Cartesian points along arcs' paths

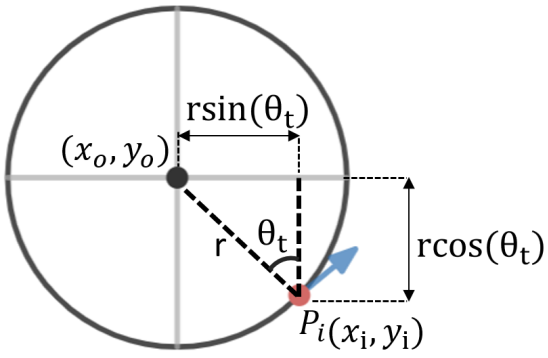


Figure 3.6: An illustration of calculating an arbitrary point (P_i) at its tangent angle (θ_t) about the corresponding arc's centre (x_o, y_o) within quadrant 4 based on Figure 3.2

As shown in Figure 3.6, the Cartesian coordinate of P_i (x_i, y_i) is (x_o+rsin(θ_t), y_o+rcos(θ_t)), however, the calculations of an arc's point would be different at different quadrants, and all of them are summarised within a piece-wise function as shown in Appendix A (A.0.8).

3.2 Special Cases

3.2.1 Type I: both k_1 and k_2 approaching to infinity

As illustrated in Figure 3.3, when k_2 tends to infinity, the tangent t_m will be parallel to t_2. Similarly, when t_1 is parallel to t_m, this condition also indicates that k_1 will tend to infinity. Accordingly, when both k_1 and k_2 tends to infinity, the connecting point cannot be found through (3.1.8) or (3.1.12) as mentioned in Section 3.1.1, and as a result, all the calculations mentioned from Section 3.1.2 to Section 3.1.3 are not applicable for this special case (e.g., the one as shown in Figure 3.7).

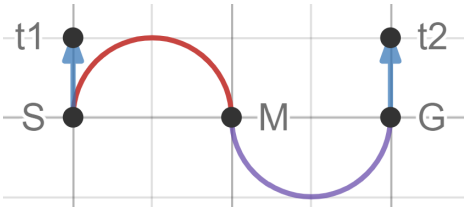


Figure 3.7: One example of a biarc candidate composed of two semicircles where both k_1 and k_2 tends to infinity

Accordingly, the connecting point M as shown in Figure 3.7 would need to be derived with a new free variable denoted as m. The position of the M can then be formulated in (3.2.1).

$$\vec{M} = \vec{S} + \frac{|\vec{G} - \vec{S}|}{m} \tag{3.2.1}$$

Where m could range from the negative infinity to the positive infinity. Accordingly, the centres and the radii of the two connecting arcs are formulated in (3.2.2) and (3.2.3) respectively.

$$r_1 = \frac{|\vec{M} - \vec{S}|}{2}; \quad \vec{C_1} = \frac{\vec{M} + \vec{S}}{2} \tag{3.2.2}$$

$$r_2 = \frac{|\vec{G} - \vec{S}|}{2}; \quad \vec{C_2} = \frac{\vec{M} + \vec{S}}{2} \tag{3.2.3}$$

Moreover, according to Figure 3.7, it is clear that the value of $\vec{\theta}_r$ for each arc is equal to 180°, while the related rotating direction can be determined based on the sign

of the cross product of the two vectors: $\vec{G}$-$\vec{S}$ and $\vec{t}_2$. Therefore, the angle of rotation of each arc can be formulated in Appendix A (A.09) and (A.010) respectively.

Furthermore, the calculations of the tangents angles at the joint point, and the related Cartesian points of an arc follow exactly the same methods as mentioned in Section 3.1.5 and Section 3.1.6 respectively.

3.2.2 Type II: $V_1 = V_2$

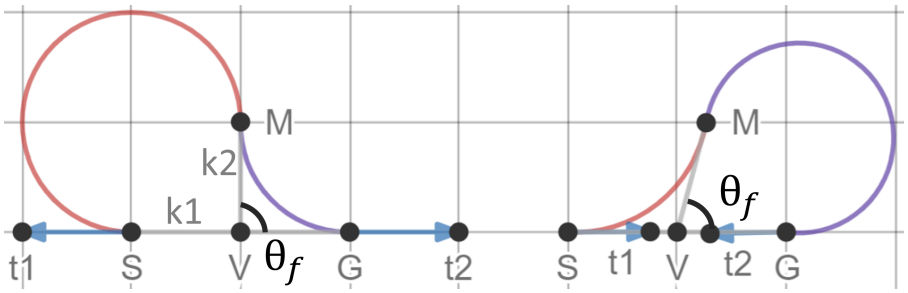


Figure 3.8: Two examples of the biarc candidates of the type II special cases. It is noted that the biarc at the left side has two end tangents that are facing away to each other, while the biarc at the right side has two end tangents that are facing towards each other.

As shown in Figure 3.8, in each example, the lines extending from S, M and G intersect at the same point V, which means $V_1 = V_2$ according to Figure 3.1. As a result, formula (3.1.5) will not be applicable for use in deriving the relation between k_1 and k_2, as it will be equal to zero. The intersection point V will always be positioned at the middle position between S and G, as k_1 will always remain the same and always equal to k_2 in their absolute values, regardless of how a biarc candidate could be varied given by S and G. Therefore, k_1 cannot be considered as a free control variable for biarc variations when $V_1 = V_2$. Accordingly, a new free variable (θ_f) needs to be defined in order to vary the position of the connecting point M. This new variable is a local angle rotating from G to M about V, and it is not related to the angle rotation panel based on Figure 3.2. Accordingly, the calculations of M for the condition of $V_1 = V_2$ is shown in (3.2.4).

$$\begin{aligned} \vec{M} &= [M_x,\, M_y] \\ \Rightarrow\ M_x &= V_x + \cos(\theta_f)(G_x - V_x) - \sin(\theta_f)(G_y - V_y) \\ \Rightarrow\ M_y &= V_y + \sin(\theta_f)(G_x - V_x) + \cos(\theta_f)(G_y - V_y) \end{aligned} \tag{3.2.4}$$

Where M_x and M_y are the x and y coordinates of $\vec{M}$, V_x and V_y are the coordinates of $\vec{V}$ and G_x and G_y are the x and y coordinates of $\vec{G}$. θ_f ranges from -360° to 360°, but

it is not equal to the absolute values of either 0°, 180° or 360°, which will make one of the arcs infinite and hence there will be a straight line between S and G. Based on (A.08) and (A.09), both k_1 and k_2 can decide the angles of rotation. However, in these special cases, the values of either k_1 and k_2 are fixed once S and G are given. Accordingly, they need to be reformulated for the type II special cases, as shown in (3.2.5).

$$(k_1,\ k_2) = \begin{cases} (-\frac{|\vec{G}-\vec{S}|}{2},\ \frac{|\vec{G}-\vec{S}|}{2}) & \vec{S}+|\vec{G}-\vec{S}|\vec{t_1} \neq \vec{G} \\ (\frac{|\vec{G}-\vec{S}|}{2},\ -\frac{|\vec{G}-\vec{S}|}{2}) & \vec{S}+|\vec{G}-\vec{S}|\vec{t_1} = \vec{G} \end{cases} \tag{3.2.5}$$

Where the first condition indicates that S and G are facing away from each other, while the second one indicates that the two points are facing towards each other. Then, the subsequent computation procedures from radius and centre calculations to the arc points calculations will still follow the methods as mentioned from Section 3.1.2 to Section 3.1.5.

3.3 Program Design and Developments

3.3.1 Automatic initialisation of the free variables

To meet the objectives of this project, the program would need to take an arbitrary user-defined inputs (a start and an end configurations as mentioned in Section 3.1) only, and then find all the biarc candidates for that certain input information, ultimately finding the one with the lowest energy. Since different values of k_1 (for standard cases) or m (for special cases) correspond to different connecting points M for a certain input information, varying the free variables can product different biarc candidates. However, not all the values of k_1 or m can produce a valid biarc solution (e.g., only one arc exist when $k_1 = 0$ or $m = 0$ or or $\theta_f = 0°$, etc.), which means randomly choosing a value of the free variables is not an effective way for the program development.

To improve the smartness of this program, the first task was to enable the program to identify where to begin to vary k_1 or m or θ_f. As for special cases type I, a biarc will be formed by two arcs of equal length, when the initial value of m is set as 0.5 which can be directly observed from formula (3.2.1). As for special cases type II, there will be always valid biarcs that can be constructed, as long as θ_f is not equal to the absolute value of 0° or 360°, and hence the initial value of θ_f could be set as 90°.

However, since an initial value of k_1 cannot be directly identified based on formula (3.1.7), which will add difficulty for the program to automatically initialise a k_1 if

following (3.1.7) only. Similarly, to avoid random initialisation for k_1, the initial value of k_1 can be simply determined by incorporating one particular condition (i.e., $k_1 = k_2$) into formula (3.1.7). Therefore, the initial value of k_1 can be derived as shown in (3.3.1) (see full derivation in Appendix A (A.0.11)).

$$k_{1int} = k_2 = \frac{-(\vec{G}-\vec{S})(\vec{t_1}+\vec{t_2}) \pm \sqrt{[(\vec{G}-\vec{S})(\vec{t_1}+\vec{t_2})]^2 + 2(1-\vec{t_1}\vec{t_2})(\vec{G}-\vec{S})(\vec{G}-\vec{S})}}{2(1-\vec{t_1}\vec{t_2})} \tag{3.3.1}$$

The initial value of k_1 can either be the negative version or the plus version based on (3.3.1), but it cannot be the both. Therefore, in this project, a positive version is assigned to the initial value of k_1 with no specific reasons. Accordingly, given by the initial value of k_1, or m or θ_f, the initial biarc candidate would be generated, once the certain condition corresponding to one of the free variables was passed in the program. Then, several conditions to distinguish different biarc cases is described in Section 3.3.2.

3.3.2 Separation of different biarc cases

Since the program was designed as whole to be capable of generating different cases of biarc candidates, it would need to analyse the user-defined information ($\vec{S}$, $\vec{G}$, θ_{t1}, θ_{t2}) as given and then make a decision on whether a group of biarc candidates can be constructed by a standard approach or special approaches, or no biarcs would exist (biarc edge case). Any type I special cases will occur when both k_1 and k_2 tends to infinity as mentioned in Section 3.2.1. Based on formula (3.3.1), both k_1 and k_2 will tend to infinity, when $\vec{t_1}$ is equal to $\vec{t_2}$, and additionally are perpendicular to the vector ($\vec{G}$-$\vec{S}$), which hence can be used as a condition to separate the type I special cases.

As for any type II special cases, after testing, it was found that they can only occur when both $\vec{t_1}$ and $\vec{t_2}$ are parallel to the vector ($\vec{G}$-$\vec{S}$), and additionally are facing either away from each other or towards each other. However, as shown in Figure 3.9, no biarcs will be constructed with reference to each pair of mutually tangent circles, when both $\vec{t_1}$ and $\vec{t_2}$ are parallel to ($\vec{G}$-$\vec{S}$), and additionally they have the same direction, which hence can be considered as the condition for use in the edge case separation.

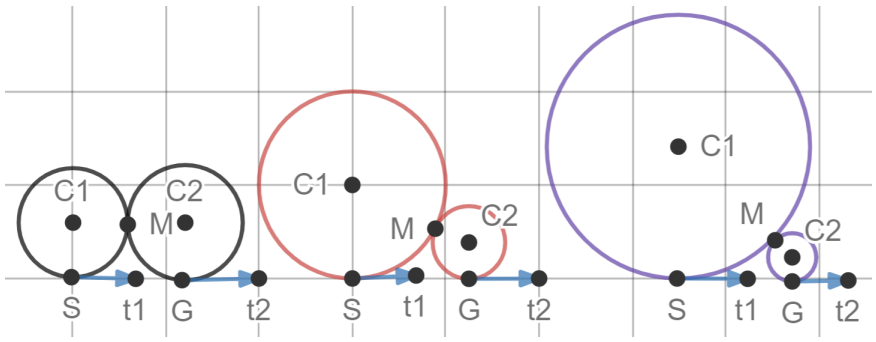


Figure 3.9: The demonstration of biarc edge cases

To maintain each pair of circles are mutually tangent given by a fix S and a fix G, C_2 will vertically move downwards as C_1 moves upwards vertically, which means the radius of the circle of C_2 will decrease, as the radius of the circle of C_1 increases. Therefore, arc SM will approach to the line segment SG in this process from the left side to the right side as shown in Figure 3.9. Accordingly, it can be deduced that there will be a straight line formed between S and G merely. when the circle of C_1 tends to infinity, whilst the circle of C_2 shrinks to a point. Finally, any valid biarc candidates will be grouped as a standard case by the program, as long as the user-defined information does not satisfy the conditions of either any biarc special cases or any biarc edge cases.

3.3.3 Program Integration

Generic cases calculator
1: k_2 and M calc (GE)
2: Radii and centres calc (GE)
3: Rotating vectors calc (GE)
4: Angles of rotation calc
5: Joint angles calc
6: Arc tangent angles interpolation
7: Arc points calc

Special cases (type I) calculator
1: k_1= k_2=infinity
2: M, radii and centres calc (SP_I)
3: Rotating vectors calc (SP_I)
4: Angles of rotation calc
5: Joint angles calc
6: Arc tangent angles interpolation
7: Arc points calc

Special cases (type II) calculator
1: $k_1$and k_2 calc (SP_II)
2: M calc (SP_II)
3. Radii and centres calc (GE)
4: Rotating vectors calc (GE)
5: Angles of rotation calc
6: Joint angles calc
7: Arc tangent angles interpolation
8: Arc points calc

Figure 3.10: Three self-defined biarc calculators with the main calculating procedures for use in the process of biarcs construction with different critical conditions as illustrated in Figure 3.11. It is noted that 'GE', 'SP_I' and SP_II' indicate 'standard', 'special type I' and 'special type II' respectively.

As shown in Figure 3.10, each calculator is one entire self-defined function composed of a series of minor self-defined functions responsible for the relevant calculations at different steps. As for each calculator, it will take a value of the corresponding free variable as the input argument, and then output a series of Cartesian vectors to represent the arc points as required to construct a biarc candidate. Some minor functions can be shared between each calculator, as the approach of the related calculations is identical as mentioned earlier. It is particular to note that any angles associated with the rotating panel (Figure 3.2) will always be automatically adjusted within the interval between -360° and 360° in each calculator.

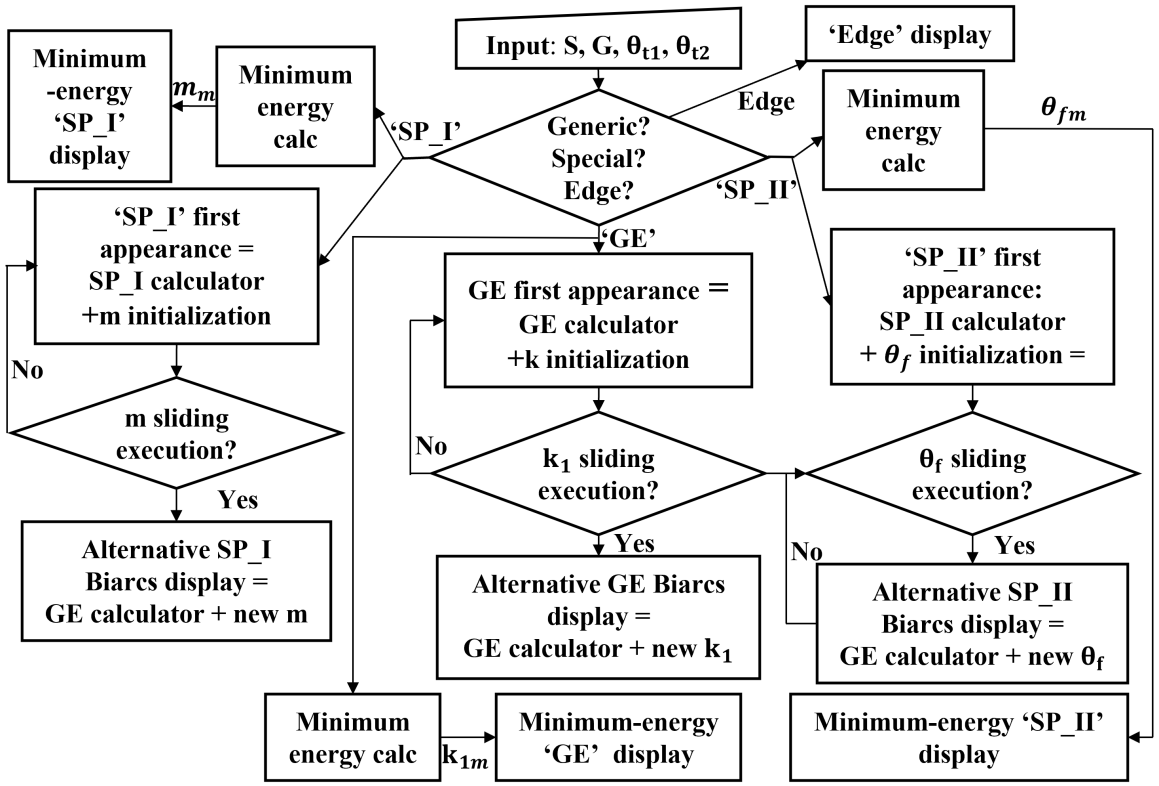


Figure 3.11: The program architecture of biarcs construction defined in Python.

As shown in Figure 3.11, the program will firstly analyse the input and then decide the approach to be used for biarcs construction. If the input information after processed, indicates the condition of a biarc edge case, then program will directly output a straight

line between S and G, or the warn to indicate that no biarcs will exist. When a critical condition indicates other cases ('GE', 'SP_I', or 'SP_II'), the program will perform the initialization of the relevant free control variables, and then incorporate the initialised values (the input arguments) to a corresponding calculator, and hence display the initial biarc candidate as the output for the corresponding case.

Moreover, as for any special type II cases, the range of the values of θ_f can be defined with the two values (-360° and 360°) without any specific calculations as mentioned in Section 3.2.2. However, as for any standard or special type I cases, given by the initialisation of the related free variables, the program will need to compute their lower and upper limits by investigating the variations in the energy of a biarc at different values of free variables.

Specifically, the program will compute an incremental step for the related free variable according to the predefined S and G. Then, the program will vary the free variable step by step, and calculate the energy of the biarc candidate at each step until almost no variations can be found in the energy calculations, where the critical values of the free variable will be considered as the lower and upper bounds for a certain group of biarc candidate. At the same time, the program will find the critical step (e.g., $k_1 m$, m_m or θ_{fm}) of the free variable corresponding to the lowest energy value, and use it as the input argument to the relevant biarc calculator to compute and display the biarc candidate of the lowest energy. The energy of a single biarc candidate can be calculated through (3.3.2).

$$E = \sum_{i=1}^{2} (c_i)^2 l_i = \sum_{i=1}^{2} \frac{1}{(r_i)^2} (r_i |\vec{\theta_{ri}}|) \tag{3.3.2}$$

Where c_i is the curvature of arc i, and it is equal to 1 divided by the square of the radius of arc i, while l_i is the arc length of arc i, and it is equal to the multiplication between the radius and the magnitude of the angle of rotation of arc i.

Furthermore, one slider widget corresponding to each critical condition ('GE', 'SP_I' or SP_II') was defined in order to vary the values of the related free variable within its range, hence displaying any alternative biarc candidates of different cases. The slider will work by manual control, and once the slider slides from the initial value of the related free variable to other values, the initial biarc candidate obtained from the free variable initialization will be removed, and then the program will read the new value of the free variable from the sliding action and use it as the input to the relevant biarc calculator for new biarc candidates construction.

Chapter 4

Triarc Construction

This chapter describes how a valid triarc candidate with different cases can be constructed (i.e., triarc standard cases (Section 4.1), triarc special cases (Section 4.2), and triarc edge cases (Section 4.3)). This chapter then describes the process of developing a complete program that combines various calculators for different cases computation, the way of generating alternative triarc candidates by using the related biarc construction methods, and the approach of selecting a triarc candidate of the lowest energy in triarc cases (Section 4.4).

4.1 Standard Cases

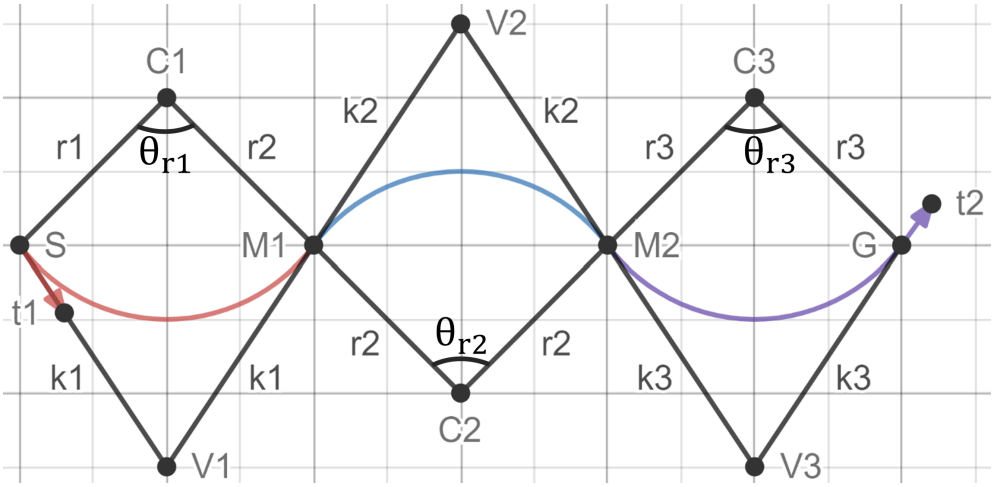


Figure 4.1: One standard instance of a triarc candidate defined with the related parameters. The triarc starts from S along the tangent $\vec{t_1}$, through two joint points M1 and M_2, and ends at G along the tangent $\vec{t_2}$. The following derivations are based on this Figure.

To connect the user-defined information with all the three free variables k_1, k_2 and

k_3 as shown in Figure 4.1, the first step is to convert the predefined information into V_1 and V_3 as expressed in (4.1.1).

$$\vec{V_1} = \vec{S} + k_1\vec{t_1}; \quad \vec{V_3} = \vec{G} \pm k_3\vec{t_2} \tag{4.1.1}$$

Where $\vec{V_3}$ can be obtained in two ways, which will be used for later derivations. Similar to (3.1.5), the relation between k_1 and k_2 can be established by calculating the distance between V_1 and V_2, while the relation between k_3 and k_2 can be obtained by calculating the distance between V_3 and V_2, as expressed in (4.1.2)

$$(\vec{V_2} - \vec{V_1})^2 = (k_1 + k_2)^2; \quad (\vec{V_2} - \vec{V_3})^2 = (k_2 + k_3)^2 \tag{4.1.2}$$

By expanding the two expressions in (4.1.2), the new expressions are shown in (4.1.3) and (4.1.4) respectively.

$$\vec{V_2}\vec{V_2} - 2(\vec{S} + k_1\vec{t_1})\vec{V_2} + \vec{S}\vec{S} + 2k_1\vec{S}\vec{t_1} = 2k_1k_2 + k_2^2 \tag{4.1.3}$$

$$\vec{V_2}\vec{V_2} - 2(\vec{G} - k_3\vec{t_2})\vec{V_2} + \vec{G}\vec{G} - 2k_3\vec{G}\vec{t_2} = 2k_2k_3 + k_2^2 \tag{4.1.4}$$

Where $\vec{V_3}$ = $\vec{G}$ - $k_3\vec{t_3}$ for use in deriving (4.1.3) and (4.1.4). Finally, by subtracting (4.1.3) by (4.1.4), the relation between the three free variables and the predefined parameters can be expressed in (4.1.5).

$$2(\vec{G} - \vec{S} - k_1\vec{t_1} - k_3\vec{t_2})\vec{V_2} = 2k_2(k_1 - k_2) - (\vec{S}\vec{S} - \vec{G}\vec{G}) - 2(k_1\vec{S}\vec{t_1} + k_3\vec{G}\vec{t_2}) \tag{4.1.5}$$

Although k_1, k_2 and k_3 were all known, $\vec{V_2}$ cannot be directly solved using (4.1.5) only, as $\vec{V_2}$ is a 2D vector that contains two Cartesian components. Randomly choosing $\vec{V_2}$ could be possible, however, it is not clear whether this action would always lead to valid triarc candidates to be constructed. Therefore, there must be a range where $\vec{V_2}$ could be varied, or $\vec{V_2}$ needs to have a valid initialisation. Accordingly, $\vec{V_2}$ can be analytically solved by a binary linear equation pair that can be established between (4.1.5) and one additional equation. By using $\vec{V_3}$ = $\vec{G}$ + $k_3\vec{t_3}$ and (4.1.2) as well as further simplifications similar to (4.1.3) and (4.1.4), this additional linear equation can be obtained and expressed in (4.1.6).

$$2(\vec{G} - \vec{S} - k_1\vec{t_1} + k_3\vec{t_2})\vec{V_2} = 2k_2(k_1 - k_2) - (\vec{S}\vec{S} - \vec{G}\vec{G}) + 2(k_1\vec{S}\vec{t_1} - k_3\vec{G}\vec{t_2}) \tag{4.1.6}$$

By incorporating the condition of $k_1 = k_3 = k_2$, the binary linear equation pair can be further simplified and shown in (4.1.7).

$$\begin{aligned} [\text{-}2\vec{V_2}(\vec{t_2} + \vec{t_1}) + 2(\vec{S}\vec{t_1} + \vec{G}\vec{t_2})]k_2 &= \vec{G}\vec{G} - \vec{S}\vec{S} - 2\vec{V_2}(\vec{G} - \vec{S}) \\ [2\vec{V_2}(\vec{t_2} - \vec{t_1}) + 2(\vec{S}\vec{t_1} - \vec{G}\vec{t_2})]k_2 &= \vec{G}\vec{G} - \vec{S}\vec{S} - 2\vec{V_2}(\vec{G} - \vec{S}) \end{aligned} \tag{4.1.7}$$

Both the two sides of each linear equation are equal to zero, arising from the subtracting action between the two expressions in (4.1.2) under this particular condition ($k_1 = k_3 = k_2$). Therefore, three new equations are expressed in (4.1.8).

$$\begin{aligned} Eqn1 &: \vec{G}\vec{G} - \vec{S}\vec{S} - 2\vec{V_2}(\vec{G} - \vec{S}) = 0 \Rightarrow V_{2x}F_x + V_{2y}F_y = \frac{1}{2}(\vec{G}\vec{G} - \vec{S}\vec{S}) \\ Eqn2 &: 2\vec{V_2}(\vec{t_2} - \vec{t_1}) + 2(\vec{S}\vec{t_1} - \vec{G}\vec{t_2}) = 0 \Rightarrow V_{2x}t_{x1} + V_{2y}t_{y1} = \vec{G}\vec{t_2} - \vec{S}\vec{t_2} \\ Eqn3 &: \text{-}2\vec{V_2}(\vec{t_2} + \vec{t_1}) + 2(\vec{S}\vec{t_1} + \vec{G}\vec{t_2}) = 0 \Rightarrow V_{2x}t_{x2} + V_{2y}t_{y2} = \vec{S}\vec{t_1} + \vec{G}\vec{t_2} \end{aligned} \tag{4.1.8}$$

Where V_{2x} and V_{2y} are the x and y components of $\vec{V_2}$, and F_x and F_y are the x and y components of $(\vec{G} - \vec{S})$. t_{x1} and t_{y1} are the x and y components of $(\vec{t_2} - \vec{t_1})$, while t_{x2} and t_{y2} are the Cartesian components of $(\vec{t_1} + \vec{t_2})$. In particular, t_{y1} and t_{y2} cannot be equal to zero simultaneously, or t_{x1} and t_{x2} cannot be equal to zero at the same time so that $\vec{V_2}$ is solvable. Accordingly, V_{2x} and V_{2y} can be solved using one of the two equations in different conditions as shown in Appendix B (B.0.1).

By substituting the obtained $\vec{V_2}$ into either of the two expressions in (4.1.2), k_2 can then be determined as shown in (4.1.9) (see the full derivation in Appendix B (B.0.2).

$$k_2 = \frac{\text{-}2(\vec{V_2}\vec{t_1} - \vec{S}\vec{t_1}) + \sqrt{4(\vec{V_2}\vec{t_1} - \vec{S}\vec{t_1})^2 - 12(2\vec{V_2}\vec{S} - (\vec{S})^2 - \vec{V_2}^2)}}{6} \tag{4.1.9}$$

Where the solution of the plus version is assigned to the initial value of k_2. and hence both k_1 and k_3 have the same initialisation equal to k_2. Accordingly, the positions of M_1 and M_2 can then be determined and expressed in (4.1.10).

$$\vec{M_1} = \frac{k_2}{k_1 + k_2}(\vec{S} + k_1\vec{t_1}) + \frac{k_1}{k_1 + k_2}\vec{V_2}; \quad \vec{M_2} = \frac{k_2}{k_2 + k_3}(\vec{G} + k_3\vec{t_2}) + \frac{k_2}{k_2 + k_3}\vec{V_2} \tag{4.1.10}$$

Similar to Section 3.1.2, r_1, $\vec{C_1}$, r_2, $\vec{C_2}$, and r_3 and $\vec{C_3}$ can be determine and expressed in (4.1.11).

$$\begin{aligned} r_1 &= \left| \frac{(\vec{M_1} - \vec{S})(\vec{M_1} - \vec{S})}{2\vec{n_1}(\vec{M_1} - \vec{S})} \right|; \; r_2 = \left| \frac{(\vec{M_2} - \vec{M_1})(\vec{M_2} - \vec{M_1})}{2\vec{n_m}(\vec{M_2} - \vec{M_2})} \right|; \; r_3 = \left| \frac{(\vec{M_2} - \vec{G})(\vec{M_2} - \vec{G})}{2\vec{n_2}(\vec{M_2} - \vec{G})} \right| \\ \vec{C_1} &= \vec{S} + |r_1|\vec{n_1}; \qquad \vec{C_2} = \vec{M_2} + |r_2|\vec{n_m}; \qquad \vec{C_3} = \vec{G} + |r_2|\vec{n_2} \end{aligned} \tag{4.1.11}$$

Where $\vec{n_1}$ and $\vec{n_2}$ are the unit normal vectors to $\vec{t_1}$ and $\vec{t_2}$ respectively, while $\vec{n_m}$ is defined as an unit normal vector to the line segment V_1V_2. The methods used for the subsequent calculations are similar to those as mentioned from Section 3.1.3 to Section 3.1.5 so that the triarc example as shown in Figure 4.1 with the initial critical condition

of (k_1=k_2=k_3) can be constructed. Once any arbitrary user-defined information is given, varying either $\vec{V_2}$, k_1, k_2 or k_3 can change the shape of a triarc candidate. Based on (4.1.6), one of the four variables can be obtained, once the other three are known. To generate alternative triarc candidates, $\vec{V_2}$, k_1 and k_2 are considered as the free control variables. Specifically, after one triarc candidate with an identical value of k is initially constructed, the initial triarc candidate (e.g., the one as shown in Figure 4.1) will then need to be considered to be formed by two biarcs: SM_2 (controlled by k_1) and M_1G (controlled by another k_1) in biarc construction. When a new $\vec{V_2}$ is given, k_1 and k_2 in triarc construction will need to be recalculated through (4.1.10) and they are still equal to each other, while k_3 will then need to be recalculated through the expression at the right side in (4.1.2). The new value of k_3 can be therefore updated through (4.1.12).

$$(\vec{V_2}-\vec{V_3})^2=(k_2+k_3)^2 \Rightarrow\ k_3=\frac{(k_2)^2+2\vec{V_2}\vec{G}-\vec{G}\vec{G}-\vec{V_2}^2}{2(\vec{V_2}\vec{t_2}-\vec{G}\vec{t_2}-k_2)} \tag{4.1.12}$$

Where $\vec{V_2}$ is actually constrained in an interval that was manipulated by the triarc construction program which is described in Section 4.4.

4.2 Special Cases

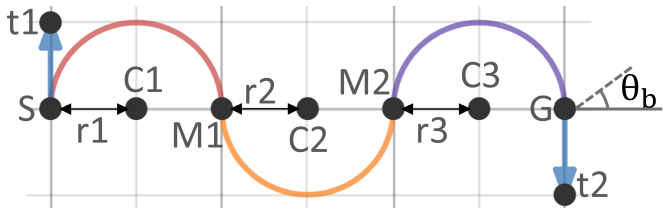


Figure 4.2: One example of a triarc candidate composed of three semicircles where k_1, k_2 and k_3 tend to infinity

Based on the first expression in (4.1.7), k_2 (equal to k_1 and k_3) will tend to infinity when its left factor of multiplication is equal to zero, and hence the critical conditions to determine any triarc special cases can be derived and shown in (4.2.1).

$$\begin{aligned} &-2\vec{V_2}(\vec{t_2}+\vec{t_1})+2(\vec{S}\vec{t_1}+\vec{G}\vec{t_2})=0 \Rightarrow\ \vec{St_1}+\vec{Gt_2}=(\vec{t_1}+\vec{t_2})\vec{V_2} \\ \Rightarrow\ &\vec{V_2}\to\infty \Rightarrow\ (1):\ \vec{t_1}=-\vec{t_2} \Rightarrow\ (2):\ (\vec{G}-\vec{S})\vec{t_2}=0 \Rightarrow\ (\vec{G}-\vec{S})\perp\vec{t_2} \end{aligned} \tag{4.2.1}$$

According to (4.2.1), it was found that $\vec{t_1}$ is opposite to $\vec{t_2}$ and both the two vectors are perpendicular to the line segment GS when k_1, k_2 and k_3 tend to infinity, which can be used as the critical conditions for the program to distinguish any triarc special cases.

Accordingly, $\vec{V_2}$ is not finite and it cannot be considered as one free control variable for any triarc special cases. Taking the triarc candidate as shown in Figure 4.2 for an instance, it can be considered to be made of two biarcs of type I special cases: SM_2 and M_1G. To generate alternative triarc candidates of special cases, there are only two free control variables that need to be defined: m_1 controlling biarc SM_2, and m_2 controlling biarc M_1G, and the control mechanism for each of them follows (3.2.1). In order to have a valid initialisation, both the values of m_1 and m_2 would be set as 0.5 so that a triarc candidate can be formed as three semicircles of equal radius connecting of each other (e.g., the one illustrated in Figure 4.2). Accordingly, the radius of each semicircle and C_2 can be calculated as expressed in (4.2.2).

$$r_1 = r_2 = r_3 = \frac{|\vec{G} - \vec{S}|}{6}; \qquad \vec{C_2} = \frac{\vec{S} + \vec{G}}{2} \tag{4.2.2}$$

Except for C_2, the calculations of C_1, M_1, M_2 and C_3 depend on the angle (θ_b) between the line segment SG and the horizontal. Prior to the calculations, a vector ($\vec{Z}$) related to θ_b would need to be defined as shown in (4.2.3). Therefore, C_1, C_3, M_1 and M_2 can be obtained as shown in (4.2.4).

$$\vec{Z} = [r_1 \cos(\theta_b),\ \ r_1 \sin(\theta_b)] \tag{4.2.3}$$

$$\vec{C_1} = \vec{S} + \vec{Z}; \qquad \vec{C_3} = \vec{S} + 5\vec{Z}; \qquad \vec{M_1} = \vec{S} + 2\vec{Z}; \qquad \vec{M_2} = \vec{S} + 4\vec{Z} \tag{4.2.4}$$

When calculating the angle of rotation, both $\vec{\theta_{r1}}$ and $\vec{\theta_{r3}}$ can be obtained based on Appendix A (A.0.5), while ($\vec{\theta_{r2}}$) can be determined using Appendix A (A.0.6). Finally, the subsequent computations (tangent angles calculations and Cartesian points calculations) will still follow the methods as mentioned from Section 3.1.4 to Section 3.1.5, as the triarc special cases are similar to the biarc type I special cases as described in Section 3.2.1.

However, the critical conditions for separating the biarc type II special cases does not independently exist in triarc construction. As shown in Figure 3.8, if the first arc SM is divided into two equal or unequal arc segments in length, V_1, V_2 and V_3 can exist at different Cartesian positions. Therefore, when both $\vec{t_1}$ and $\vec{t_2}$ are parallel to the vector ($\vec{G}$-$\vec{S}$), and additionally are facing either away from each other or towards each other, any triarc candidates under this critical condition can be constructed by the triarc standard approach as described in Section 4.1.

4.3 Edge Cases

When both the two unit vectors $\vec{t_1}$ and $\vec{t_2}$ have a vertical direction (i.e., [0, 1]) or a horizontal direction (i.e., [1, 0]), both t_{x1} and t_{x2} will be equal to zero so that V_{2x} cannot be solved according to Appendix B (B.0.1) under the two scenarios, which are considered as the critical conditions for any triarc edge cases (e.g., the one as shown in Figure 4.3).

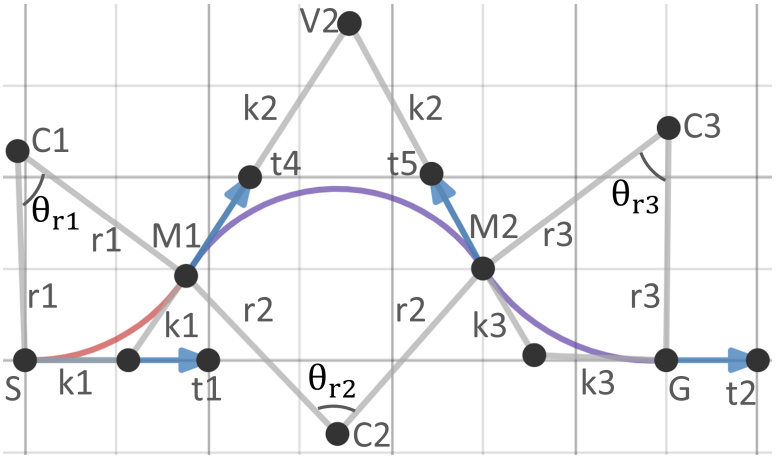


Figure 4.3: One example of a triarc edge case

Instead, to construct the $\vec{V_2}$ as shown in Figure 4.3, $\vec{\theta}_{r1}$ and r_1 would need to be chosen randomly (e.g., $|\vec{\theta}_{r1}|$=30° and r_1=0.3($|\vec{G}-\vec{S}|$), as long as the following biarc to be constructed is valid) so that C_1 can be determined with respect to the S. By using the tangent angles interpolation and the Cartesian points interpolation, arc SM_1 can be initially constructed with an end point M_1 at the tangent $\vec{t_4}$. Given by M_1 and G, a biarc with the joint point M_2 at the tangent -$\vec{t_5}$ will then need to be initially constructed using the biarc construction program so that the initial triarc candidate starting from S and ending at G can be formed. The $\vec{V_2}$ can be determined by finding the intersection point between the line extending from M_1 at $\vec{t_4}$ and the line extending from M_2 at $\vec{t_5}$.

To generate an alternative triarc candidate, given by a new $\vec{V_2}$ with respect to the initial $\vec{V_2}$, the related parameters can be calculated following from formula (4.1.9) to formula (4.1.12). Once the new triarc candidate is formed, it will then need to be considered to be formed by two biarcs: SM_2 controlled by k_1 and M_1G controlled by another k_1 for further variations of triarc alternative candidates. As for any biarc edge cases, when both the related $\vec{t_1}$ and $\vec{t_2}$ are horizontal or vertical, the problem of biarc edge cases can be addressed by using the approach as mentioned here, otherwise, the problem can be directly resolved by using the triarc standard approach.

4.4 Program Integration

The triarc construction program will firstly determine the method to be adopted for generating triarc candidates by analysing the user-defined information as given. As for any triarc standard cases, the program will initialise a valid triarc candidate by calculating the $\vec{V_2}$ when k_1, k_2 and k_3 have identical values. Then, the program can vary $\vec{V_2}$ to generate alternative triarc candidates, but the $\vec{V_2}$ must be assigned with a range, hence reducing the computational efforts of finding an optimal triarc candidate. Since the program will consider the current triarc candidate to be formed by two biarcs that share the same middle arc at each $\vec{V_2}$, and then execute the biarc construction method to further construct other alternative triarc candidates, the variations of each biarc are independent so that the initial arc of a triarc candidate could intersect with its end arc at a certain $\vec{V_2}$. With respect to the initial $\vec{V_2}$, it will be set to vary in four orthogonal directions in the 2D Cartesian space, until the limit of $\vec{V_2}$ for each direction is found when the biarc intersection issue occurs.

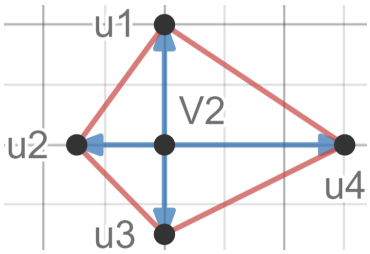


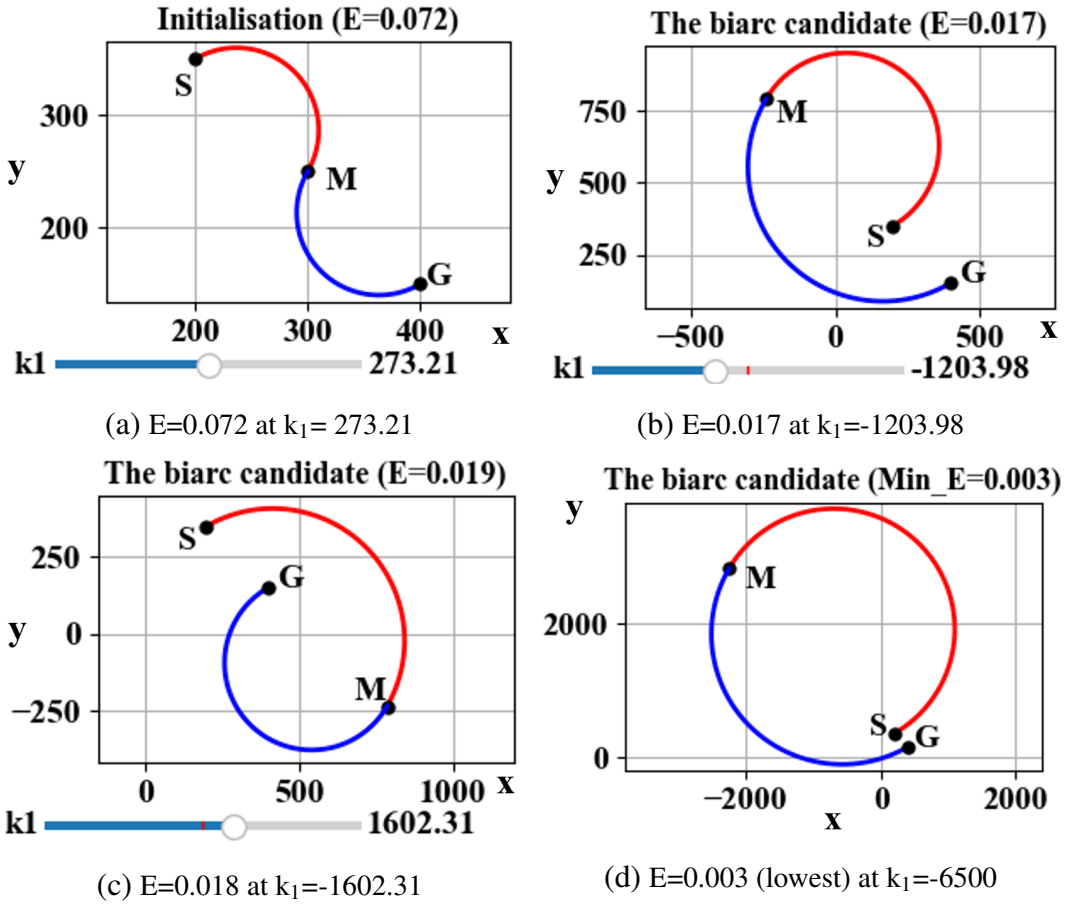
Figure 4.4: One example of a triarc edge case

The variations of $\vec{V_2}$ will be confined within the shape formed by connecting the four limits (u_1, u_2, u_3, and u_4 as shown in Figure 4.4) to each other. As there will be two free variables (i.e., k_1 and k_2) controlling two biarcs within one triarc candidate, the range of each variable can be found using the same method as mentioned in Section 3.3.3. By iterating the available steps of $\vec{V_2}$, k_1, and k_2, the triarc candidate with the minimum energy can be determined. As for any triarc special cases, once the triarc program have finished the initialisation procedure, the initial triarc candidate will only need to be controlled by two biarc control variables (i.e., m_1 and m_2) for further variations. After k_1 and k_2 are assigned with a range, the optimal triarc candidate will be found by iterating their available steps that can generate valid triarc candidates. As for any triarc edge cases, the program will construct a $\vec{V_2}$ from initialisation, and use the same method (as used in triarc standard cases) to determine the optimal triarc candidate among alternative triarc candidates.

Chapter 5

Results and Discussions

5.1 Biarc Standard Cases



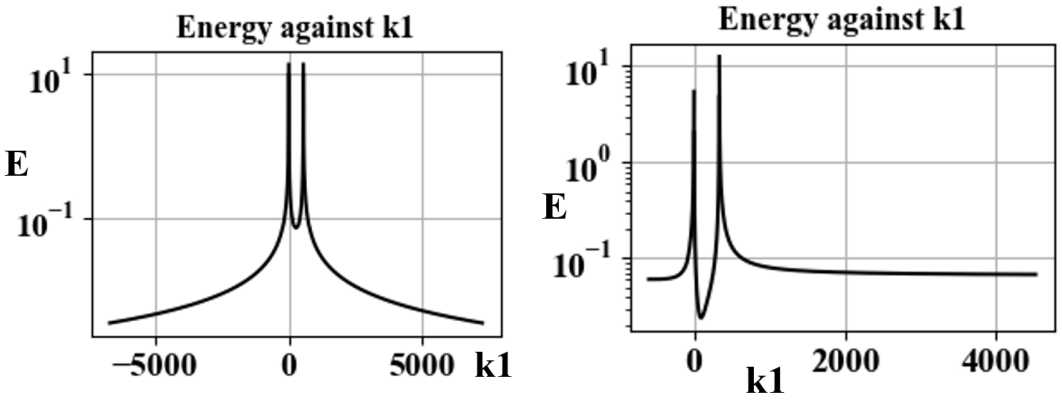
(a) E=0.072 at k_1= 273.21

(b) E=0.017 at k_1=-1203.98

(c) E=0.018 at k_1=-1602.31

(d) E=0.003 (lowest) at k_1=-6500

Figure 5.1: The illustration of several examples of biarc candidates of different energy constructed at different values of k_1 under the configurations (i.e., $\vec{S}$=[200, 350], $\vec{G}$=[400, 150], θ_1=30°, θ_2=30°)



(a) when θ_1= 30° and θ_2= 30°

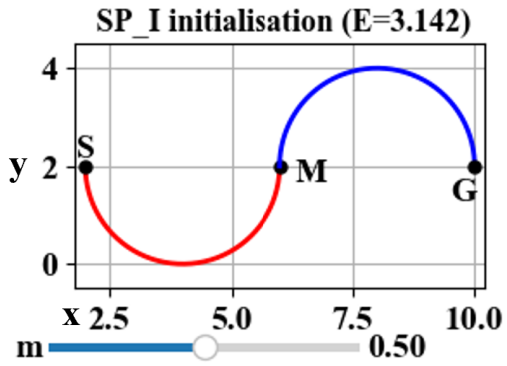
(b) when θ_1= 20° and θ_2= 150°

Figure 5.2: Two different energy distributions against the whole range of k_1 at the same $\vec{S}$ ([200, 350]) and $\vec{G}$ ([400, 150]), but different pairs of tangent angles

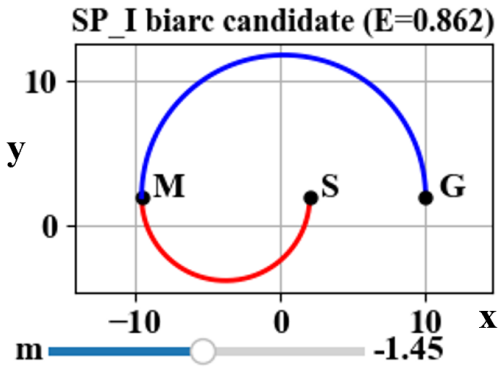
When either k_1 or k_2 is equal to zero, the energy will tend to infinity. As in either of the cases. the radius of the related arc will become zero, which makes its curvature approaching to infinity based on (3.3.2). Therefore, as shown in Figure 5.2 (a), there are two spikes occurring near k_1=0 and k_1= 546.41 (i.e., k_2=0) respectively. When k_1 is further decreased from the left spike or increased from the right spike, both the radius and the arc-length of each arc will increase (e.g., as illustrated in Figure 5.1 (b) and (c)). However, the biarc energy will increase as shown in Figure 5.2 (a), since the squared function in (3.3.2) makes a decrease in curvature being more dominant than an increase in arc-length for energy variations.

Since the energy distribution against k_1 in Figure 5.2 (a) is symmetric, there is another biarc candidate that also has the lowest energy (E=0.003) at k_1=6500, besides the one as shown in Figure 5.1 (d). As for any symmetric energy distributions that exist, the value of the minimum energy depends on the precision of the limits of k_1. For any biarc standard cases, when the four significant figures behind the difference between the energy at the current k_1 and the energy at the previous k_1 can be rounded to zero, the current values of k_1 will then be set as the limits. If the precision was set as less than 4 significant figures, the value of the minimum energy would shift toward the centre of the distribution. Furthermore, the minimum energy of a biarc configuration might not always occur at the lower or the upper limit of k_1. Instead, it can occur around the middle position along the range of k_1 (e.g., Figure 5.2 (b)). Accordingly, it is reasonable to iterate through the entire interval of k_1 to find the biarc candidate that has the minimum energy among other alternative ones.

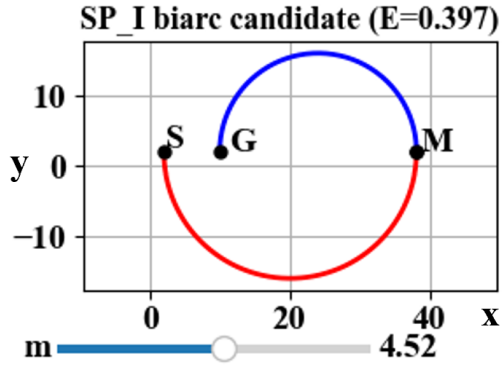
5.2 Biarc Special Cases



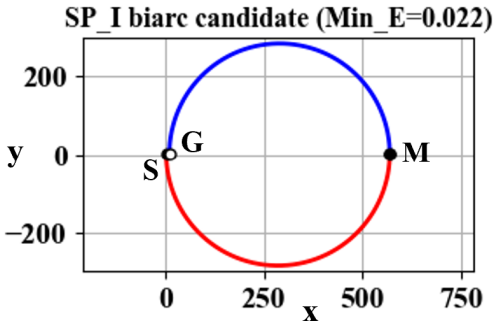
(a) E=3.142 at m= 0.5



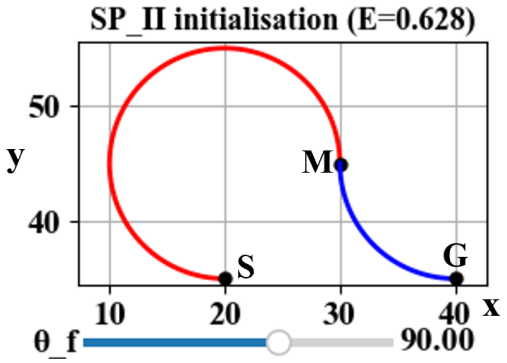
(b) E=0.862 at m=-1.45



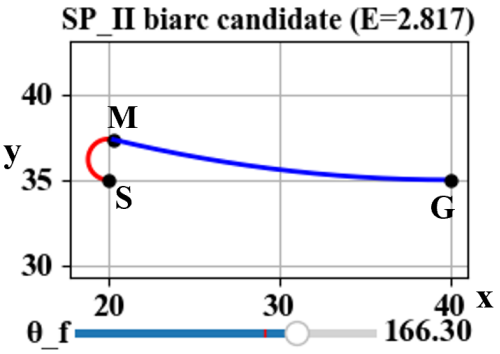
(c) E=0.397 at m= 4.52



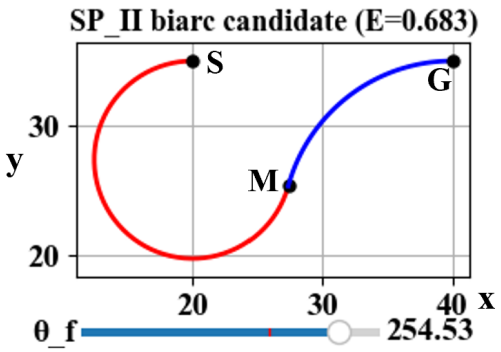
(d) E=0.022 (lowest) at m= 71



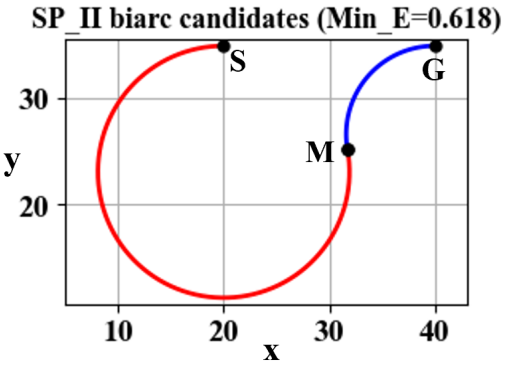
(e) E=0.628 at θ_f = 90°



(f) E=2.817 at θ_f = 166.30°

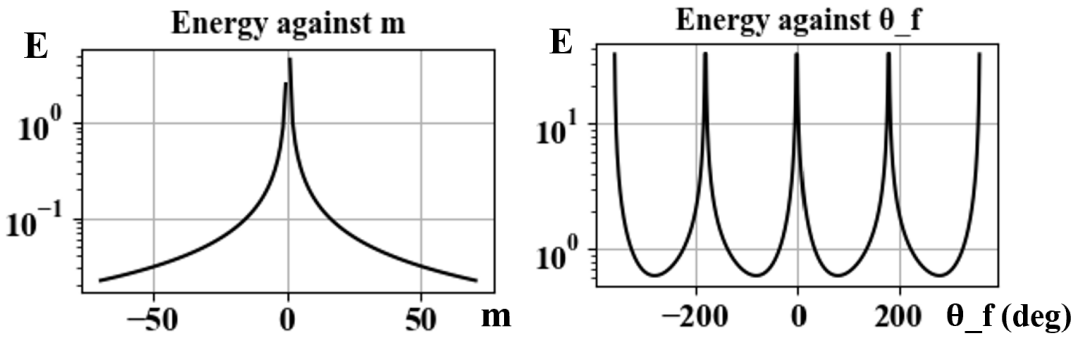


(g) E=0.683 at θ_f = 254.53°



(h) E=0.618 (lowest) at θ_f = -80°

Figure 5.3: SP_I (type I special cases) biarc candidates from (a) to (d) under the condition ($\vec{S}$=[2, 2], $\vec{G}$=[10, 2], θ_1=θ_2=-90°), and SP_II (type II special cases) biarc candidates from (e) to (h) under the condition ($\vec{S}$=[20, 35], $\vec{G}$=[40, 35], θ_1= 180°, θ_2= 0°)



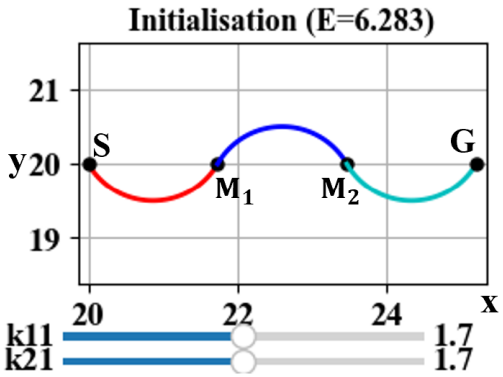
(a) As for the type I special case (b) As for the type II special case

Figure 5.4: Two energy distributions corresponding to the two conditions in Figure 5.3

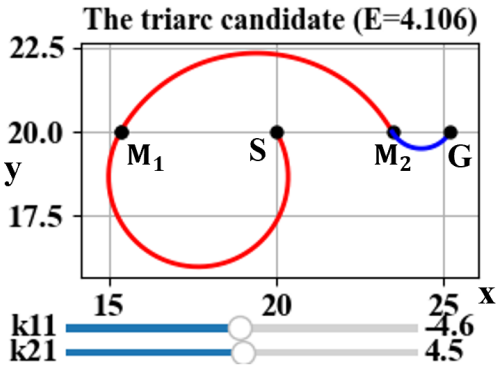
As for the type I special case, when m is slid to 0, arc SM will shrink to one point at which M will coincide with S, or arc MG will shrink to one point where M will coincide with G when m is slid to 1. In either case, the energy will tend to infinity so that there are two spikes taking place near m=0 and m=1 respectively as shown in Figure 5.3 (a). The SP_I biarc candidate with the lowest energy can be found by either decreasing or increasing the value of m, as the related energy distribution is almost symmetric about m=0. Therefore, besides the one as shown in Figure 5.3 (d), there is another biarc candidate of E = 0.022 that can be constructed at m = -71, but in a configuration similar to the biarc candidate (e.g., Figure 5.3 (b)). However, it may not be necessary to iterate the values of m in both negative and positive directions for path selection in real-world applications (e.g., robot's obstacles avoidance). If the obstacles existed at the left side of the starting position (S), it would be more effective to only consider the alternative biarc candidates that are generated by increasing the value of m positively.

As for the type II special case, when θ_f is slid to $\pm 180°$, $0°$, or $\pm 360°$, one of the arcs will shrink to one point, causing the energy approaching to infinity. Therefore, there are five spikes that can occur near these angles as shown in Figure 5.3 (b), as θ_f ranges between -360° and 360°. Since the energy distribution is symmetric about θ_f=0°, θ_f has another three values that can create one biarc candidate having the same lowest energy of 0.618, apart from the one generated at θ_f = 80° as shown in Figure 5.3 (h). Due to the property of symmetry, seeking one optimal SP_II biarc candidate can be further simplified by iterating the values of θ_f merely in one-quarter of the whole range (e.g, (180, 360°]) where there must be one optimal SP_ biarc candidate taking place.

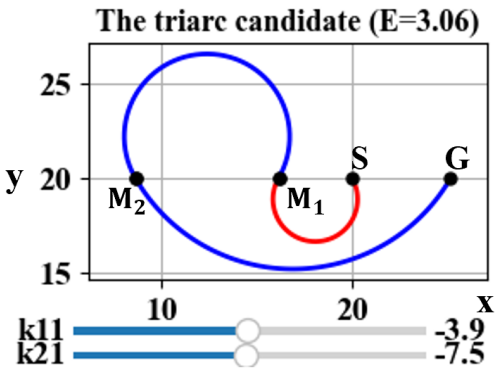
5.3 Triarc Standard and Special Cases



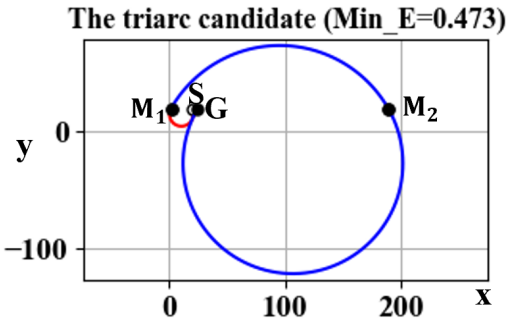
(a) E=6.283 at k_{11}=k_{21}= 1.7



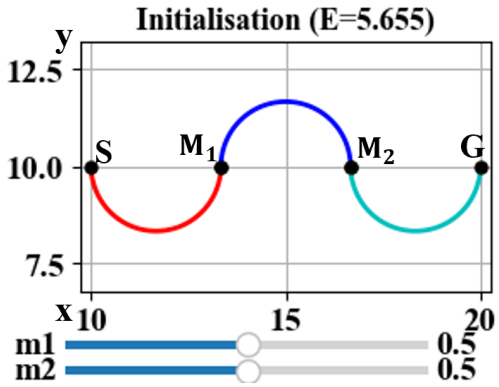
(b) E=4.106 at k_{11}=-4.6 and k_{21}= 4.5



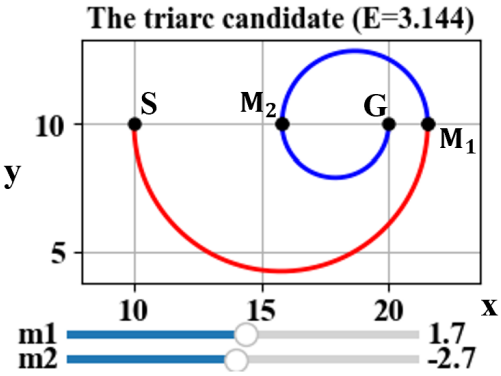
(c) E=3.06 at k_{11}=-3.9 and k_{21}=-7.5



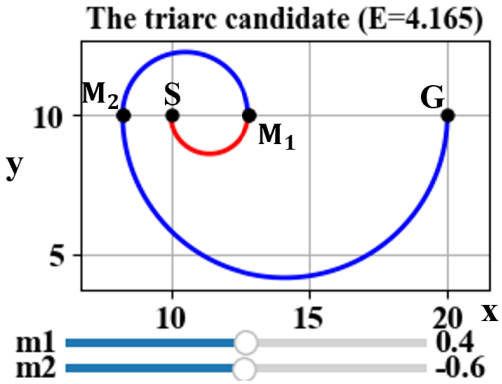
(d) Min E=0.473 at k_{11}=-17.7 and k_{21}= 186.6



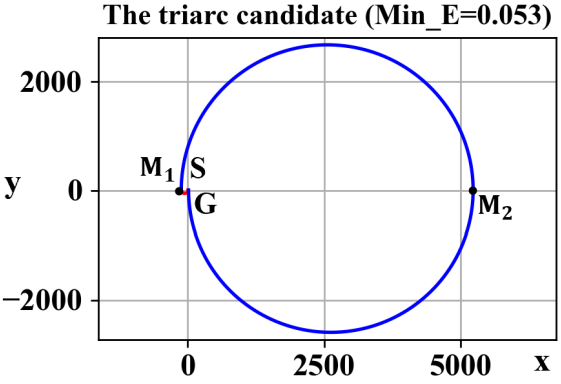
(e) E=5.655 at m_1= m_2= 0.5



(f) E=3.144 at m_1=1.7 and m_2=-2.7

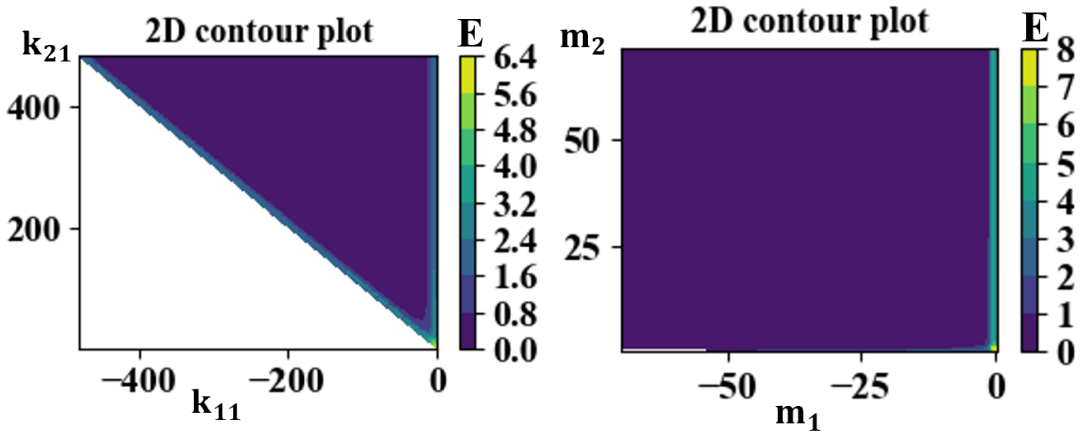


(g) E=4.165 at m_1= 0.4 and m_2=-0.6



(h) Min E=0.053 at m_1=-70 and m_2= 71

Figure 5.5: Examples of standard triarc candidates from (a) to (d) under the condition ($\vec{S}$=[20, 20], $\vec{G}$=[25.2, 20], θ_1=300°, θ_2=60°), and triarc candidates of special cases from (e) to (f) under the condition ($\vec{S}$=[10, 10], $\vec{G}$=[20, 10], θ_1=-90°, θ_2=90°). It is noted that both k_{11} and m_1 are used for biarc SM_2 variations in their own cases, while both k_{21} and m_2 are used for biarc M_1G variations in their own cases.



(a) As for the triarc generic case

(b) As for the triarc special case

Figure 5.6: Two energy distributions corresponding to the two conditions in Figure 5.5

After iterating all the available $\vec{V}_2$, the triarc group with the lowest energy occurred at the $\vec{V}_2$ ([22.6, 21.5]) where there are several alternative triarc candidates that can be constructed as shown from Figure 5.5 (a) to (d). Based on the discussion of biarc cases, the biarc energy could decrease in most of the cases as the absolute value of the related control variables increase. As for the standard condition, the triarc candidate with the lowest energy (e.g., Figure 5.5 (d)) could be found when k_{11} increases negatively, whilst k_{21} increases positively, and vice versa.

However, the variations of biarc SM_2 and biarc M_1G are independent of each other so that arc SM_1 can intersect with arc M_2G in some pairs of k_{11} and k_{21}, and hence any triarc candidates with this issue are not considered as valid ones. Accordingly, as shown in Figure 5.6 (a), the 2D energy contour contains an empty space with no energy values where any pairs of k_{11} and k_{21} can cause an invalid triarc candidate, and hence these combinations were removed by the triarc construction program before any energy computations.

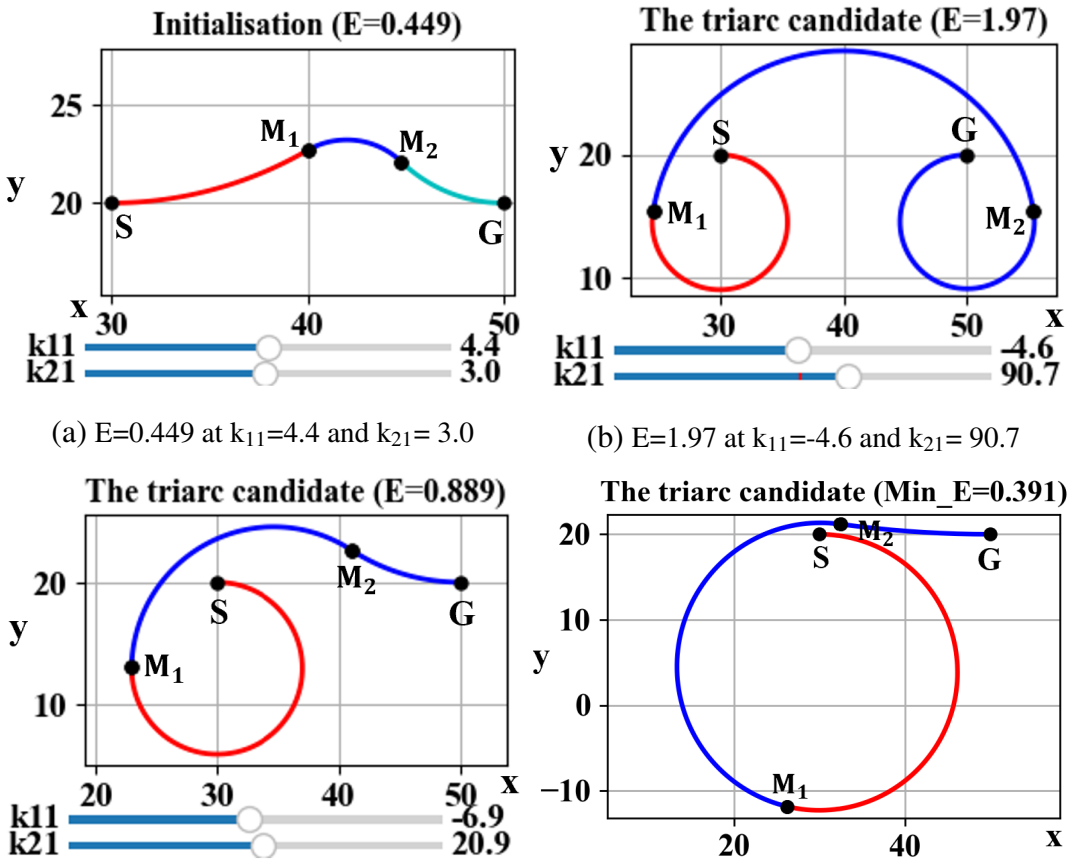
Moreover, as mentioned earlier, the range of a certain $\vec{V}_2$ can be confined by connecting several critical points where the intersection issue occurred due to independent variations of k_{11} and k_{21}. In fact, when this issue occurred, it could be addressed by tuning k_{11} or k_{21} again so that some of the points of limits might not exist, which means

the approach of finding the range of $\vec{V_2}$ in this project may not be unique. The lowest energy (0.053) of the standard triarc group is near zero so that the range of the $\vec{V_2}$ could be acceptable. However, due to the uncertainties of finding the range of $\vec{V_2}$, it could be hard to decide whether an optimal triarc candidate obtained from other standard cases has a local minimum or a global minimum. Meanwhile, as the range of a $\vec{V_2}$ becomes larger, the triarc program will require more computational efforts to find an optimal triarc candidate for any standard cases, which could be less feasible for any cases that result in a large range of $\vec{V_2}$.

As for the triarc special case, the triarc candidate with the lowest energy (as shown in Figure 5.5 (h)) can be found by decreasing m_1, whilst increasing m_2, and vice versa. If the triarc candidate with the global minimum energy was infeasible in a real application (e.g., robot's obstacles avoidance as mentioned earlier), the robot could pick a feasible one with similar low energy (e.g., as shown in Figure 5.6 (b) where almost all of the pairs of m_1 and m_2 can construct a triarc candidate with the values of energy less than 1. Moreover, finding an optimal triarc candidate of any triarc special cases will only involve tuning m_1 and m_2 without any further computations for $\vec{V_2}$, so the optimal triarc candidate that is obtained will have a global minimum value of energy.

5.4 Triarc Edge Cases

Under the triarc edge case, the triarc group with the lowest energy occurred at the $\vec{V_2}$ ([42.5, 24.0]) where there are several alternative triarc candidates that can be constructed as shown from Figure 5.7 (a) to (d). Similarly, the optimal triarc candidate (Figure 5.7 (d)) can be found by decreasing k_{11}, whilst increasing k_{21}. Similar to Figure 5.6 (a), there is also an empty space in the energy distribution as shown in Figure 5.8, as not all the pairs of k_{11} and k_{21} can produce valid triarc candidates due to the intersection issue. Since determining an optimal triarc candidate for any triarc edge cases will still involve in finding a range of the related $\vec{V_2}$, it could be difficult to judge whether the optimal triarc candidate among the related triarc candidate group has a local minimum value or a global minimum value.



(a) E=0.449 at k_{11}=4.4 and k_{21}= 3.0

(b) E=1.97 at k_{11}=-4.6 and k_{21}= 90.7

(c) E=0.391 at k_{11}=-6.9 and k_{21}= 20.9

(d) E=0.473 at k_{11}=-208.7, k_{21}= 335.8

Figure 5.7: Examples of triarc candidates of one edge case from (a) to (d) under the condition ($\vec{S}$=[30, 20], $\vec{G}$=[50, 20], θ_1=θ_2=0°)

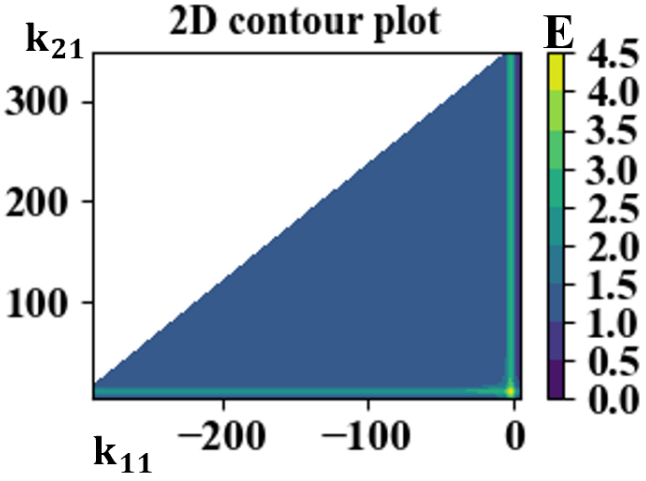


Figure 5.8: The energy distribution of the triarc edge case

Chapter 6

Conclusion

Due to the advantages of arc-based paths (conceptually simpler and less computational efforts) compared to other complex curves, this study focused on the arc-based path construction (i.e., biarcs and triarcs) within the context of path planning in robot navigation, which has been presented in this thesis. Both biarc and triarc construction methods are geometric-oriented, due to the advantages (less computational efforts and greater density of valid path candidates) compared to other numerical methods. In either construction method, a valid arc-based path can be obtained with a series of parameters calculations (i.e., arcs' joint points, arcs' centres, arcs' radii, arcs' angles of rotation, arcs' tangent angles and arcs' Cartesian points).

According to various user-defined information, the biarc construction program will consider three categorisations (standard cases, special cases (type I and type II), and edge cases). Special type I cases will occur when both $\vec{t_1}$ and $\vec{t_2}$ are perpendicular to the line segment composed of S and G, and additionally are equal to each other. Any biarc candidates of these cases will always consist of two semicircles. Special type II cases will occur when both $\vec{t_1}$ and $\vec{t_2}$ are parallel to the line segment (SG), and additionally are either towards or opposite each other. However, any biarc edge cases will occur when the two tangent vectors have the same direction, and additionally are parallel to SG, which cannot be handled by the biarc construction program. All the other conditions will be identified as biarc standard cases by the biarc construction program.

As for biarc standard cases, some of the energy distributions could be symmetrical about a certain point so that there will be two optimal candidates having the same lowest energy that occurs at the lower and upper bounds of k_1 respectively, while there could be only one optimal candidate that occurs around the middle position of k_1. However, as for biarc special type I cases, the energy distributions under any arbitrary conditions

will always be symmetric about 0 so that these cases will always have two optimal biarc candidates occurring at the two bounds of m, which means the range of m could be cut in half for use in practice. As for biarc special type II cases, there will always be four optimal biarc candidates of the same lowest energy so that one quarter range of θ_f could be enough for use in practice without any further constraints (e.g., obstacles). Since only one free control variable is enough for tuning the variations of biarc candidates, any valid optimal biarc candidates to be found by the biarc construction program will have a global minimum value of energy.

By extending the biarc program to the triarc construction program, the triarc program will also need to classify different user-defined information into three categorisations (triarc standard cases, triarc special cases, and triarc edge cases). Any triarc special cases will occur when both $\vec{t_1}$ and $\vec{t_2}$ are perpendicular to SG, but the two tangent vectors are opposite of each other. Any triarc edge cases will occur when both $\vec{t_1}$ and $\vec{t_2}$ have a vertical direction or they have a horizontal direction where any biarc edge cases can be solved here. As for any triarc standard cases or triarc edge cases, displaying alternative triarc candidates can be achieved by varying $\vec{V_2}$, k_{11} and k_{21}. At a certain $\vec{V_2}$, the triarc candidate will be considered as two biarcs controlled by k_{11} and k_{21} respectively.

The method of finding the range of $\vec{V_2}$ is based on the intersection issue where arc one and arc three of a triarc candidate could intersect at some pairs of k_{11} and k_{21} due to their independent variations. However, the intersection issue can be handled by re-tuning k_{11} and k_{21} at where it occurs. As a result, the range of $\vec{V_2}$ under a certain configuration could be not unique so that it may be difficult to judge whether an optimal triarc candidate obtained for a certain triarc standard case or a certain triarc edge case will have a global minimum value of energy. Additionally, more computational efforts could arise for the triarc program as the range of $\vec{V_2}$ becomes greater. Accordingly, further investigations could be performed on how to determine a certain range of $\vec{V_2}$, or deriving another free control variable to replace $\vec{V_2}$.

As for any triarc special cases, only two free control variables (i.e., m_1 and m_2) are needed for controlling the variations of triarc candidates, as $\vec{V_2}$ does not exist in these cases. Therefore, an optimal triarc candidate obtained for a certain triarc special case will have a global minimum value of energy. Furthermore, the two arc-based construction algorithms were developed in a 2D space in this project, it could be difficult to directly apply them onto a differential robotic car for further practical testing. Therefore, future work can also be focused on extending the 2D algorithms in a 3D space and then testing the path-planning performance with the modified 3D algorithms.

Appendix A

Relevant Formula used in Chapter 3

$$
\begin{aligned}
(\vec{V}_2-\vec{V}_1)^2=(k_1+k_2)^2 \;\Rightarrow\; (\vec{G}-k_2\vec{t_2}-\vec{S}-k_1\vec{t_1})^2 &= k_1^2+2k_1k_2+k_2^2\\
\Rightarrow\; (\vec{P}-k_2\vec{t_2}-k_1\vec{t_1})^2 &= k_1^2+2k_1k_2+k_2^2\\
\Rightarrow\; \vec{P}\vec{P}-2k_2\vec{P}\vec{t_2}-2k_1\vec{P}\vec{t_1}+k_2^2\vec{t_2}\vec{t_2}+2k_1k_2\vec{t_1}\vec{t_2}+k_1^2\vec{t_1}\vec{t_1} &= k_1^2+2k_1k_2+k_2^2\\
\Rightarrow\; \vec{P}\vec{P}-2k_2\vec{P}\vec{t_2}-2k_1\vec{P}\vec{t_1}+k_2^2+2k_1k_2\vec{t_1}\vec{t_2}+k_1^2 &= k_1^2+2k_1k_2+k_2^2\\
\Rightarrow\; \vec{P}\vec{P}-2k_2\vec{P}\vec{t_2}-2k_1\vec{P}\vec{t_1}+2k_1k_2\vec{t_1}\vec{t_2}-2k_1k_2 &= 0\\
\Rightarrow\; \vec{P}\vec{P}-2k_1\vec{P}\vec{t_1}-k_2[2\vec{P}\vec{t_2}-2k_1(\vec{t_1}\vec{t_2}-1)] &= 0
\end{aligned}
\tag{A.0.1}
$$

$$
\begin{aligned}
\frac{\vec{M}-\vec{V}_1}{\vec{V}_2-\vec{M}}=\frac{k_1}{k_2} \;\Rightarrow\; k_2(\vec{M}-\vec{V}_1)&=k_1(\vec{V}_2-\vec{M})\\
\Rightarrow\; k_2\vec{M}-k_2\vec{V}_1=k_1\vec{V}_2-k_1\vec{M} \;\Rightarrow\; k_2\vec{M}+k_1\vec{M}&=k_1\vec{V}_2+k_2\vec{V}_1\\
\Rightarrow\; (k_1+k_2)\vec{M}=k_2\vec{V}_1+k_1\vec{V}_2 \;\Rightarrow\; \vec{M}&=\frac{k_2}{k_1+k_2}\vec{V}_1+\frac{k_1}{k_1+k_2}\vec{V}_2\\
\Rightarrow\; \vec{M}=\frac{k_2}{k_1+k_2}(\vec{S}+k_1\vec{t_1})+\frac{k_1}{k_1+k_2}&(\vec{G}-k_2\vec{t_2})
\end{aligned}
\tag{A.0.2}
$$

$$
\begin{aligned}
\vec{M}=\vec{V}_1+b\vec{t_2} \;\Rightarrow\; \vec{M}&=\vec{V}_1+[(\vec{G}-\vec{V}_1)\vec{t_2}]\vec{t_2}\\
\Rightarrow\; \vec{M}&=(\vec{S}+k_1\vec{t_1})+[(\vec{G}-\vec{S}-k_1\vec{t_1})\vec{t_2}]\vec{t_2}\\
\Rightarrow\; \vec{M}&=(\vec{S}+k_1\vec{t_1})+(\vec{G}\vec{t_2}-\vec{S}\vec{t_2}-k_1\vec{t_1}\vec{t_2})\vec{t_2}
\end{aligned}
\tag{A.0.3}
$$

$$
\begin{aligned}
(\vec{S}-\vec{C_1})^2 = (\vec{M}-\vec{C_1})^2 \;\Rightarrow\; \vec{S}\vec{S}-2\vec{S}\vec{C_1}+\vec{C_1}\vec{C_1} &= \vec{M}\vec{M}-2\vec{M}\vec{C_1}+\vec{C_1}\vec{C_1} \\
2\vec{M}\vec{C_1}-2\vec{S}\vec{C_1} = \vec{M}\vec{M}-\vec{S}\vec{S} \;\Rightarrow\; 2\vec{C_1}(\vec{M}-\vec{S}) &= \vec{M}\vec{M}-\vec{S}\vec{S} \\
2(\vec{S}+r_1\vec{n_1})(\vec{M}-\vec{S}) = \vec{M}\vec{M}-\vec{S}\vec{S} \;\Rightarrow\; 2\vec{S}\vec{M}-2\vec{S}\vec{S}+2r_1\vec{n_1}\vec{M}-2r_1\vec{n_1}\vec{S} &= \vec{M}\vec{M}-\vec{S}\vec{S} \\
2r_1\vec{n_1}(\vec{M}-\vec{S}) = \vec{M}\vec{M}-2\vec{S}\vec{M}+\vec{S}\vec{S} \;\Rightarrow\; r_1 &= \left|\frac{\vec{M}\vec{M}-2\vec{S}\vec{M}+\vec{S}\vec{S}}{2\vec{n_1}(\vec{M}-\vec{S})}\right| \\
\Rightarrow\; r_1 &= \left|\frac{(\vec{M}-\vec{S})(\vec{M}-\vec{S})}{2\vec{n_1}(\vec{M}-\vec{S})}\right|
\end{aligned}
\tag{A.0.4}
$$

$$
\vec{\theta_{r1}} = \begin{cases}
-\arccos(\hat{C_2G}\cdot\hat{C_2M})+2\pi & k_1<0 \;\; and \;\; \hat{C_1S}\times\hat{C_1M}\le 0 \\
\arccos(\hat{C_2G}\cdot\hat{C_2M})-2\pi & k_1<0 \;\; and \;\; \hat{C_1S}\times\hat{C_1M}>0 \\
\arccos(\hat{C_2G}\cdot\hat{C_2M}) & k_1>0 \;\; and \;\; \hat{C_1S}\times\hat{C_1M}>0 \\
-\arccos(\hat{C_2G}\cdot\hat{C_2M}) & k_1>0 \;\; and \;\; \hat{C_1S}\times\hat{C_1M}\le 0
\end{cases}
\tag{A.0.5}
$$

$$
\vec{\theta_{r2}} = \begin{cases}
\arccos(\hat{C_2M}\cdot\hat{C_2G})-2\pi & k_2\le 0 \;\; and \;\; \hat{C_2M}\times\hat{C_2G}<0 \\
-\arccos(\hat{C_2M}\cdot\hat{C_2G})+2\pi & k_2<0 \;\; and \;\; \hat{C_2M}\times\hat{C_2G}>0 \\
-\arccos(\hat{C_2M}\cdot\hat{C_2G}) & k_2>0 \;\; and \;\; \hat{C_2M}\times\hat{C_2G}>0 \\
\arccos(\hat{C_2M}\cdot\hat{C_2G}) & k_2>0 \;\; and \;\; \hat{C_2M}\times\hat{C_2G}\le 0
\end{cases}
\tag{A.0.6}
$$

$$
\theta_{m2} = \begin{cases}
(\theta_{m1}-180^\circ)-180^\circ & (\theta_{m1}\in Q_1 \;\; or \;\; \theta_{m1}\in Q_4) \;\; and \;\; \theta_{m1}>0 \\
(\theta_{m1}+180^\circ)+180^\circ & (\theta_{m1}\in Q_1 \;\; or \;\; \theta_{m1}\in Q_4) \;\; and \;\; \theta_{m1}<0 \\
(\theta_{m1}-540^\circ)-180^\circ & (\theta_{m1}\in Q_2 \;\; or \;\; \theta_{m1}\in Q_3) \;\; and \;\; \theta_{m1}>0 \\
(\theta_{m1}+540^\circ)+180^\circ & (\theta_{m1}\in Q_2 \;\; or \;\; \theta_{m1}\in Q_3) \;\; and \;\; \theta_{m1}<0 \\
-\theta_{m1}-180^\circ & (|\theta_{m1}|=90^\circ \;\; or \;\; |\theta_{m1}|=270^\circ) \;\; and \;\; \theta_{m1}>0 \\
-\theta_{m1}+180^\circ & (|\theta_{m1}|=90^\circ \;\; or \;\; |\theta_{m1}|=270^\circ) \;\; and \;\; \theta_{m1}<0 \\
-\theta_{m1} & |\theta_{m1}|=180^\circ \;\; or \;\; |\theta_{m1}|=360^\circ \\
\theta_{m1}+360^\circ & |\theta_{m1}|=0^\circ \;\; and \;\; |\vec{\theta}_{r2}|>0 \\
\theta_{m1}-360^\circ & |\theta_{m1}|=0^\circ \;\; and \;\; |\vec{\theta}_{r2}|<0
\end{cases}
\tag{A.0.7}
$$

$$(x_i,\, y_i) = \begin{cases} x_o + r\sin(|\theta_t|),\; y_o + r\sin(|\theta_t|) & \theta_t \in [\text{-}90^\circ,\; 0^\circ];\;\; |\vec{\theta}_r| < 0 \\ x_o + r\sin(180^\circ\text{-}\theta_t),\;\; y_o + r\cos(180^\circ\text{-}\theta_t) & \theta_t \in [90^\circ,\; 180^\circ];\;\; |\vec{\theta}_r| > 0 \\ x_o\text{-}r\sin(360^\circ\text{-}|\theta_t|),\;\; y_o + r\cos(|360^\circ\text{-}\theta_t|) & \theta_t \in [\text{-}360^\circ,\; \text{-}270^\circ];\;\; |\vec{\theta}_r| < 0 \\ x_o\text{-}r\sin(\theta_t\text{-}180^\circ),\;\; y_o + r\cos(\theta_t\text{-}180^\circ) & \theta_t \in (180^\circ,\; 270^\circ];\;\; |\vec{\theta}_r| > 0 \\ x_o\text{-}r\sin(|\theta_t|\text{-}180^\circ),\;\; y_o\text{-}r\cos(|\theta_t|\text{-}180^\circ) & \theta_t \in (\text{-}270^\circ,\; \text{-}180^\circ];\;\; |\vec{\theta}_r| < 0 \\ x_o\text{-}r\sin(360^\circ\text{-}\theta_t),\;\; y_o\text{-}r\cos(360^\circ\theta_t) & \theta_t \in (270^\circ,\; 360^\circ];\;\; |\vec{\theta}_r| > 0 \\ x_o + r\sin(180^\circ\text{-}|\theta_t|),\;\; y_o\text{-}r\cos(180^\circ\text{-}|\theta_t|) & \theta_t \in (\text{-}180^\circ,\; \text{-}90^\circ);\;\; |\vec{\theta}_r| < 0 \\ x_o + r\sin(\theta_t),\;\; y_o\text{-}r\cos(\theta_t) & \theta_t \in [0^\circ,\; 90^\circ);\;\; |\vec{\theta}_r| > 0 \end{cases} \tag{A.0.8}$$

where $|\vec{\theta}_r|$ is the magnitude of the angle of rotation of an arc.

$$\vec{\theta_{r1}} = \begin{cases} -180^\circ & (\vec{G}-\vec{S})\times\vec{t_2} \geq 0 \\ 180^\circ & (\vec{G}-\vec{S})\times\vec{t_2} < 0 \end{cases} \tag{A.0.9}$$

$$\vec{\theta_{r2}} = \begin{cases} -180^\circ & (\vec{G}-\vec{S})\times\vec{t_2} \leq 0 \\ 180^\circ & (\vec{G}-\vec{S})\times\vec{t_2} > 0 \end{cases} \tag{A.0.10}$$

$$\begin{aligned} k_2 = \frac{(\vec{G}-\vec{S})(\vec{G}-\vec{S}) - 2(\vec{G}-\vec{S})k_1\vec{t_1}}{2(\vec{G}-\vec{S})\vec{t_2} - 2k_1(\vec{t_1}\vec{t_2}-1)} &\Rightarrow k_2 = \frac{(\vec{G}-\vec{S})(\vec{G}-\vec{S}) - 2(\vec{G}-\vec{S})k_2\vec{t_1}}{2(\vec{G}-\vec{S})\vec{t_2} - 2k_2(\vec{t_1}\vec{t_2}-1)} \\ \Rightarrow\; -2k_2^2(\vec{t_1}\vec{t_2}-1) + 2(\vec{G}-\vec{S})\vec{t_2}k_2 + 2(\vec{G}-\vec{S})\vec{t_1}k_2 - (\vec{G}-\vec{S})(\vec{G}-\vec{S}) &= 0 \\ \Rightarrow\; k_2^2(1-\vec{t_1}\vec{t_2}) + (\vec{G}-\vec{S})(\vec{t_1}+\vec{t_2})k_2 - \frac{1}{2}(\vec{G}-\vec{S})(\vec{G}-\vec{S}) &= 0 \\ \Rightarrow\; k_2 = \frac{-(\vec{G}-\vec{S})(\vec{t_1}+\vec{t_2}) \pm \sqrt{[(\vec{G}-\vec{S})(\vec{t_1}+\vec{t_2})]^2 + 2(1-\vec{t_1}\vec{t_2})(\vec{G}-\vec{S})(\vec{G}-\vec{S})}}{2(1-\vec{t_1}\vec{t_2})} & \\ \Rightarrow\; k_{1int} = k_2 = \frac{-(\vec{G}-\vec{S})(\vec{t_1}+\vec{t_2}) + \sqrt{[(\vec{G}-\vec{S})(\vec{t_1}+\vec{t_2})]^2 + 2(1-\vec{t_1}\vec{t_2})(\vec{G}-\vec{S})(\vec{G}-\vec{S})}}{2(1-\vec{t_1}\vec{t_2})} & \end{aligned} \tag{A.0.11}$$

Appendix B

Relevant Formula used in Chapter 4

$$\begin{cases} (Eqn1,\ Eqn2) & (V_y = 0\ \ and\ \ t_{y1} \neq 0)\ \ or\ \ (V_x = 0\ \ and\ \ t_{x1} \neq 0) \\ (Eqn1,\ Eqn3) & (V_y = 0\ \ and\ \ t_{y1} = 0)\ \ or\ \ (V_x = 0\ \ and\ \ t_{x1} = 0) \\ (Eqn1,\ Eqn2) & (V_x \neq 0\ \ and\ \ V_y \neq 0)\ \ and\ \ (t_{x1} \neq 0\ \ and\ \ t_{y1} \neq 0) \\ (Eqn1,\ Eqn2) & (V_x \neq 0\ \ and\ \ V_y \neq 0)\ \ and\ \ (t_{x1} \neq 0\ \ and\ \ t_{y1} \neq 0) \\ (Eqn1,\ Eqn2) & (V_x \neq 0\ \ and\ \ V_y \neq 0)\ \ and\ \ (t_{x1} = 0\ \ and\ \ t_{y1} \neq 0) \\ (Eqn1,\ Eqn3) & (V_x \neq 0\ \ and\ \ V_y \neq 0)\ \ and\ \ (t_{x1} = 0\ \ and\ \ t_{y1} = 0) \end{cases} \tag{B.0.1}$$

$$\begin{aligned} (\vec{V_2} - \vec{V_1})^2 = (k_1 + k_2)^2 \Rightarrow\ & (\vec{V_2} - \vec{V_1})^2 = (2k_2)^2 \\ \Rightarrow\ & 3k_2^2 + 2(\vec{V_2}\vec{t_1} - \vec{S}\vec{t_1})k_2 + (2\vec{V_2}\vec{S} - (\vec{S})^2 - \vec{V_2}^2) = 0 \\ \Rightarrow\ k_2 = & \frac{\text{-}2(\vec{V_2}\vec{t_1} - \vec{S}\vec{t_1}) \pm \sqrt{4(\vec{V_2}\vec{t_1} - \vec{S}\vec{t_1})^2 - 12(2\vec{V_2}\vec{S} - (\vec{S})^2 - \vec{V_2}^2)}}{6} \\ \Rightarrow\ k_2 = & \frac{\text{-}2(\vec{V_2}\vec{t_1} - \vec{S}\vec{t_1}) + \sqrt{4(\vec{V_2}\vec{t_1} - \vec{S}\vec{t_1})^2 - 12(2\vec{V_2}\vec{S} - (\vec{S})^2 - \vec{V_2}^2)}}{6} \end{aligned} \tag{B.0.2}$$

Bibliography

[1] Estifanos Mihret. Robotics and artificial intelligence. *International Journal of Artificial Intelligence and Machine Learning*, 10, 10 2020.

[2] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors (Basel)*, 21(6):2140, March 2021.

[3] Srikanth Kavirayani, Aravind Papasani, sai vamsi Tataverthi, and Divya Sree. Robot for delivery of medicines to patients using artificial intelligence in health care. 10 2020.

[4] Rafael Garcia, Nuno Gracias, Tudor Nicosevici, Ricard Prados, Natalia Hurtos, Ricard Campos, Javier Escartin, Armagan Elibol, and Ramon Hegedus. *Exploring the Seafloor with Underwater Robots: Land, Sea and Air*, pages 75–99. 02 2017.

[5] Robin Murphy, Satoshi Tadokoro, Daniele Nardi, Adam Jacoff, Paolo Fiorini, Howie Choset, and Aydan Erkmen. *Search and Rescue Robotics*, pages 1151–1173. 01 2008.

[6] J. Ricardo Sánchez Ibáñez, Carlos Perez-del Pulgar, and Alfonso Garcia. Path planning for autonomous mobile robots: A review. *Sensors*, 21:7898, 11 2021.

[7] Márcia M. Costa and Manuel F. Silva. A survey on path planning algorithms for mobile robots. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–7, 2019.

[8] Aida A. Nasr, Nirmeen A. El-Bahnasawy, and Ayman El-Sayed. Straight-line: A new global path planning algorithm for mobile robot. In *2021 International Conference on Electronic Engineering (ICEEM)*, pages 1–5, 2021.

[9] Haikuo Shen, Liangwei Jiang, Qinjian Zhang, Yong Tao, and Yunan Cao. A new method for high speed and smooth transfer of robot motion trajectory. *Advances in Mechanical Engineering*, 8(3):1687814016638309, 2016.

[10] Dorian Cojocaru, Liviu Florin Manta, Cristina Floriana Pană, Andrei Dragomir, Alexandru Marin Mariniuc, and Ionel Cristian Vladu. The design of an intelligent robotic wheelchair supporting people with special needs, including for their visual system. *Healthcare (Basel)*, 10(1):13, December 2021.

[11] Asif Ahmed Neloy, Rafia Alif Bindu, Sazid Alam, Ridwanul Haque, Md. Saif Ahammod Khan, Nasim Mahmud Mishu, and Shahnewaz Siddique. Alpha-n: Shortest path finder automated delivery robot with obstacle detection and avoiding system. In Ngoc Thanh Nguyen, Kietikul Jearanaitanakij, Ali Selamat, Bogdan Trawiński, and Suphamit Chittayasothorn, editors, *Intelligent Information and Database Systems*, pages 202–213, Cham, 2020. Springer International Publishing.

[12] Abhijeet Ravankar, Ankit Ravankar, Yukinori Kobayashi, Yohei Hoshino, and Chao-Chung Peng. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors*, 18:3170, 09 2018.

[13] Mohamed Elhoseny, Alaa Tharwat, and Aboul Ella Hassanien. Bezier curve based path planning in a dynamic field using modified genetic algorithm. *J. Comput. Sci.*, 25:339–350, 2018.

[14] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*, PP:1–8, 08 2017.

[15] Rida T Farouki, Carlotta Giannelli, Duccio Mugnaini, and Alessandra Sestini. Path planning with pythagorean-hodograph curves for unmanned or autonomous vehicles. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 232(7):1361–1372, 2018.

[16] Joshué Pérez, Jorge Godoy, Jorge Villagrá, and Enrique Onieva. Trajectory generator for autonomous vehicles in urban environments. In *2013 IEEE International Conference on Robotics and Automation*, pages 409–414, 2013.

[17] L. Hilario, Marta Mora, Nicolas Montes, and Antonio Falco. *Path Planning Based on Parametric Curves*. 09 2018.

[18] Thomas F. Banchoff and Peter J. Giblin. On the geometry of piecewise circular curves. *American Mathematical Monthly*, 101:403–416, 1994.

[19] Xunnian Yang. Efficient circular arc interpolation based on active tolerance control. *Computer-Aided Design*, 34:1037–1046, 11 2002.

[20] Alexey Kurnosenko. Biarcs and bilens. *Computer Aided Geometric Design*, 30:310–330, 03 2013.

[21] A. I. Kurnosenko. On the length preserving approximation of plane curves by circular arcs. *Computational Mathematics and Mathematical Physics*, 57(4):590–606, apr 2017.

[22] K.M. Bolton. Biarc curves. *Computer-Aided Design*, 7(2):89–92, 1975.

[23] Taweechai Nuntawisuttiwong and Natasha Dejdumrong. An approximation of bézier curves by a sequence of circular arcs. *Information Technology and Control*, 50:213–223, 06 2021.

[24] Les A. Piegl and Wayne Tiller. Biarc approximation of nurbs curves. *Computer-Aided Design*, 34(11):807–814, 2002.

[25] G.-P Wang and J.-G Sun. Biarc approximation of planar nurbs curve and its offset. 11:1368–1374, 10 2000.

[26] Mohamed Essid, Bassem Gassara, Maher Baili, Moncef Hbaieb, Gilles Dessein, and Wassila Bouzid Saï. Analytical modeling of the CNC machine axis motion in high-speed milling with local smoothing. *International Journal of Advanced Manufacturing Technology*, 105(1-4):457–470, 2019.

[27] Xujing Yang and Zezhong C. Chen. A practicable approach to g1 biarc approximations for making accurate, smooth and non-gouged profile features in cnc contouring. *Computer-Aided Design*, 38(11):1205–1213, 2006.

[28] Shunsaku Arita and Pongsathorn RAKSINCHAROENSAK. Optimal path construction incorporating a biarc interpolation and smooth path following for automobiles. *SICE Journal of Control, Measurement, and System Integration*, 13:23–29, 03 2020.

[29] Rahul Vishen, Marius C. Silaghi, and Joerg Denzinger. Gps data interpolation: Bezier vs. biarcs for tracing vehicle trajectory. In Osvaldo Gervasi, Beniamino

Murgante, Sanjay Misra, Marina L. Gavrilova, Ana Maria Alves Coutinho Rocha, Carmelo Torre, David Taniar, and Bernady O. Apduhan, editors, *Computational Science and Its Applications – ICCSA 2015*, pages 197–208, Cham, 2015. Springer International Publishing.

[30] Zonggao Mu, Peng Wang, Zheng Li, and Wenfu Xu. A biarc method for kinematics and configuration planning of concentric wire-driven manipulators. *IEEE Access*, 7:151439–151448, 2019.

[31] Liang Meng, Wei-Hong Zhang, Ji-Hong Zhu, and Liang Xia. A biarc-based shape optimization approach to reduce stress concentration effects. *Acta Mech. Sin.*, 30(3):370–382, June 2014.

[32] Tirupathi Chandrupatla and Thomas Osler. Planar biarc curves – a geometric view. *The Mathematical Scientist*, 36, 01 2011.

[33] D.S. Meek and D.J. Walton. The family of biarcs that matches planar, two-point g1 hermite data. *Journal of Computational and Applied Mathematics*, 212(1):31–45, 2008.

[34] Chongyang Deng and Weiyin Ma. Matching admissible g2 hermite data by a biarc-based subdivision scheme. *Computer Aided Geometric Design*, 29(6):363–378, 2012.

[35] Bahattin Koç, Yuan-Shin Lee, and Yawei Ma. Max-fit biarc fitting to stl models for rapid prototyping processes. In *SMA'01*, 2001.

[36] Zbynek Sir, Robert Feichtinger, and Bert Juttler. Approximating curves and their offsets using biarcs and pythagorean hodograph quintics. *Computer-Aided Design*, 38(6):608–618, 2006.

[37] Enrico Bertolazzi and Marco Frego. A note on robust biarc computation. *Computer-Aided Design and Applications*, 16, 11 2017.

[38] Dongfang Hu and Jianwei Guo. Curve-fitting of cam profile and its improvement. 01 2017.

[39] D.S. Meek and D.J. Walton. Planar osculating arc splines. *Computer Aided Geometric Design*, 13(7):653–671, 1996.

[40] Mohammad Nahid Islam Razive. Evaluation of linear segment length and local curvature radius along airfoil leading and trailing edges. 2012.

[41] M.-Y. Yang, T.-Y. Shon, T.-M. Lee, and T.-M. Lee. Cam profile machining by triarc curve fitting. *International Journal of Production Research*, 36(7):1767–1778, 1998.

[42] Ao Li and Bahari Idrus. Integration of a* algorithm and brute force algorithm in solving a multi destination path planning. 2018.