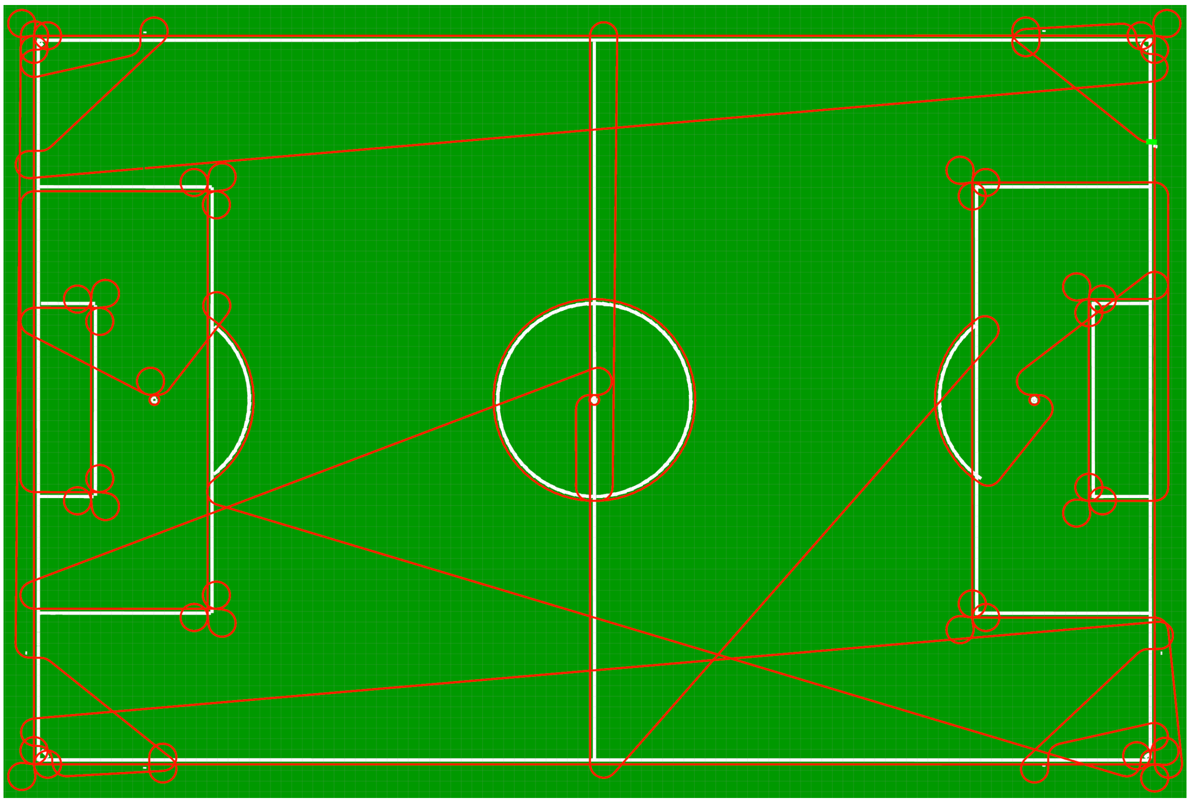
Continuous Curvature Path Planning for Differential Drive Robots

7th Semester Project Worksheet

Project Worksheet

ROB752



Aalborg University
The Technical Faculty for IT and Design
Department of Electronic Systems

AALBORG UNIVERSITY
STUDENT REPORT

The Technical Faculty for IT and Design
Niels Jernes Vej 10, 9220 Aalborg Øst
http://www.aau.dk

Title:
Continuous Curvature Path Planning for Differential Drive Robots

Theme:
Advanced Mobile Robots

Project Period:
ROB 7th Semester

Project Group:
752

Participant(s):

Anye Den Warren Igwacho	20231017
Asbjørn Lauritsen	20202917
Frederik Saldern Nielsen	20205137
Mads Frier	20204277
Victor Hoffmann Risager	20203377

Supervisor:
Kirsten Mølgaard Nielsen

Copies: 1

Page Numbers: 54

Date of Completion:
January 18, 2024

Abstract:

This paper presents a path planning algorithm for Turf Tank's robots enabling the robots to follow trajectories more precisely for line marking robot tasks. The planning algorithm improves the geometric continuity of the path from G^1 to G^2 by using clothoids as transition curves between the segments of the Dubins path. Testing of the performance of the proposed algorithm in terms of continuity, completeness, optimality, and wheel acceleration were conducted in a developed simulated environment.

The results show that the algorithm generates continuous curvature paths increasing the accuracy of the line following robot, but with a cost of longer trajectories. The parameters for the proposed algorithm considers the kinematic and dynamic constraints of the Turf Tank Two robot. Which is done to estimate parameters for the minimum turning radius and curvature rate, which results in a better transient response. Demonstrating the applicability of the algorithm for various line marking templates, has been done to compare it with the non-modified Dubins path planner. Evaluating benefits and drawbacks with the developed algorithm.

Contents

Acronyms

CCC	Curve Curve Curve
CCD	Continuous Curvature Dubins
CORS	Continuously Operating Reference Station
CSC	Curve Straight Curve
DC	Direct Current
DDR	Differential Drive Robot
DoF	Degrees of Freedom
DPP	Dubins Path Planner
EKF	Extended Kalman Filter
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
LRL	Left Right Left
LSL	Left Straight Left
LSR	Left Straight Right
MPC	Model Predictive Controller
PID	Proportional Integral Derivative
RLR	Right Left Right
ROS	Robot Operating System
RSL	Right Straight Left
RSR	Right Straight Right
RTK	Real Time Kinematic
UHF	Ultra High Frequency
UI	User Interface
UX	User Experience

1 Introduction

This project is a collaboration between the line marking robot company, Turf Tank and the project group. The goal of this collaboration is to alter the current trajectory planning to lower the high wheel acceleration. Since different Turf Tank robot models use the same software, this project will focus on the newest model, the Turf Tank Two platform.

1.1 Case Description

For companies such as Turf Tank accurate path following is crucial for their line-marking, as it ensures consistent and precise lines for the sporting fields their robot has to paint. The high wheel acceleration originates from the Turf Tank team's existing path planning algorithm, which connects two non colinear points with heading using a Dubins path. The Dubins path is the shortest path between two points, for a minimum turning radius constrained vehicle. However, the issue with employing the Dubins path is the lack of continuity in the curvature profile. This discontinuity results in infinite wheel acceleration, during transitions between the two segments in the Dubins path. The unobtainable acceleration subsequently results errors in the traced robot path. Turf Tank is aware of the issue and currently accomodates it by adding pre- and post padding to the transition paths, so the robot has time to reach a steady state, before it reaches the start of the paint segment. The pre- and post padding creates a suboptimal path, therefore the task is to insert a transition curve that reduces the high wheel acceleration.

1.2 Initial Problem Statement

The initial problem statement can now be formed on behalf of the case description.

How can an algorithm make the planned trajectory that allows for lower wheel accelerations, for the Turf Tank robotic platform?

---

2 Problem Analysis

In this chapter, there will be looked into the issues faced when plotting trajectories for differential drive robots. There will be a deep dive into the value Turf Tank brings to stakeholders, as a line marking company. The focus will shift to Turf Tank's hardware and software solution, aiming to understand key factors when developing software for their robot. Lastly different transition curves, will be analysed weighing their pros and cons for a better grasp of their applications.

2.1 Stakeholder Analysis

This section will delve into the landscape of stakeholders who play pivotal roles in shaping the trajectory of Turf Tank. Understanding the influence, interests, and levels of authority these stakeholders hold within the company is essential for successful project management and strategic decision-making. To provide a visual representation of this stakeholder landscape, a Power-Interest Matrix has been constructed, which can be found in Figure 2.1. This matrix will serve as a valuable tool in assessing the relative power and level of interest each stakeholder holds in Turf Tank's endeavours.

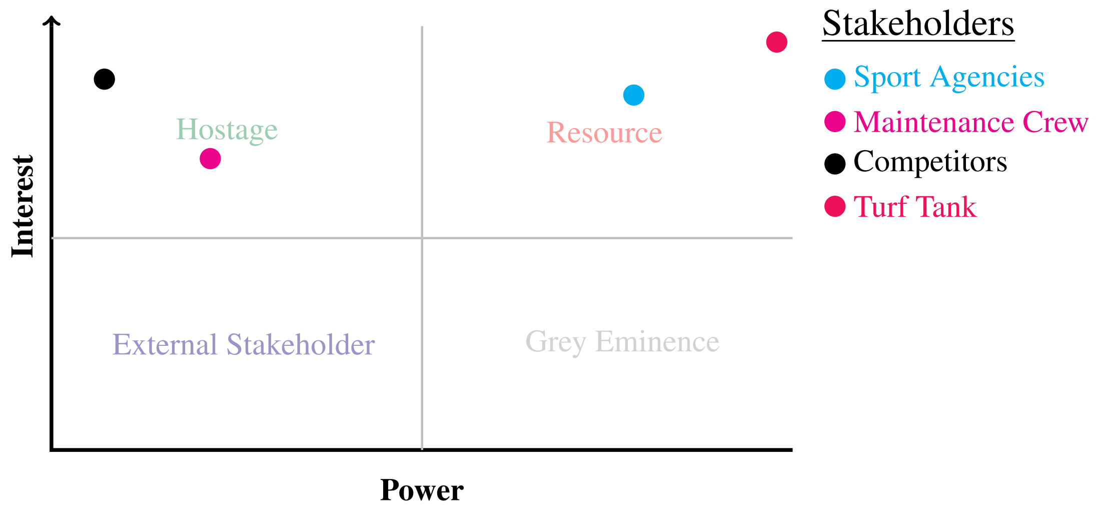


Figure 2.1: Power interest with the different listed stakeholders categorised.

2.1.1 Turf Tank

Turf Tank is a Danish-based company that specialises in the production of robots capable of projecting two-dimensional drawings onto the ground using paint. These robots have garnered significant popularity within the domain of sporting arenas, where they are employed for tasks such as drawing football, rugby fields, etc. On Figure 2.2, a Turf Tank robot can be seen in use on a larger football arena.

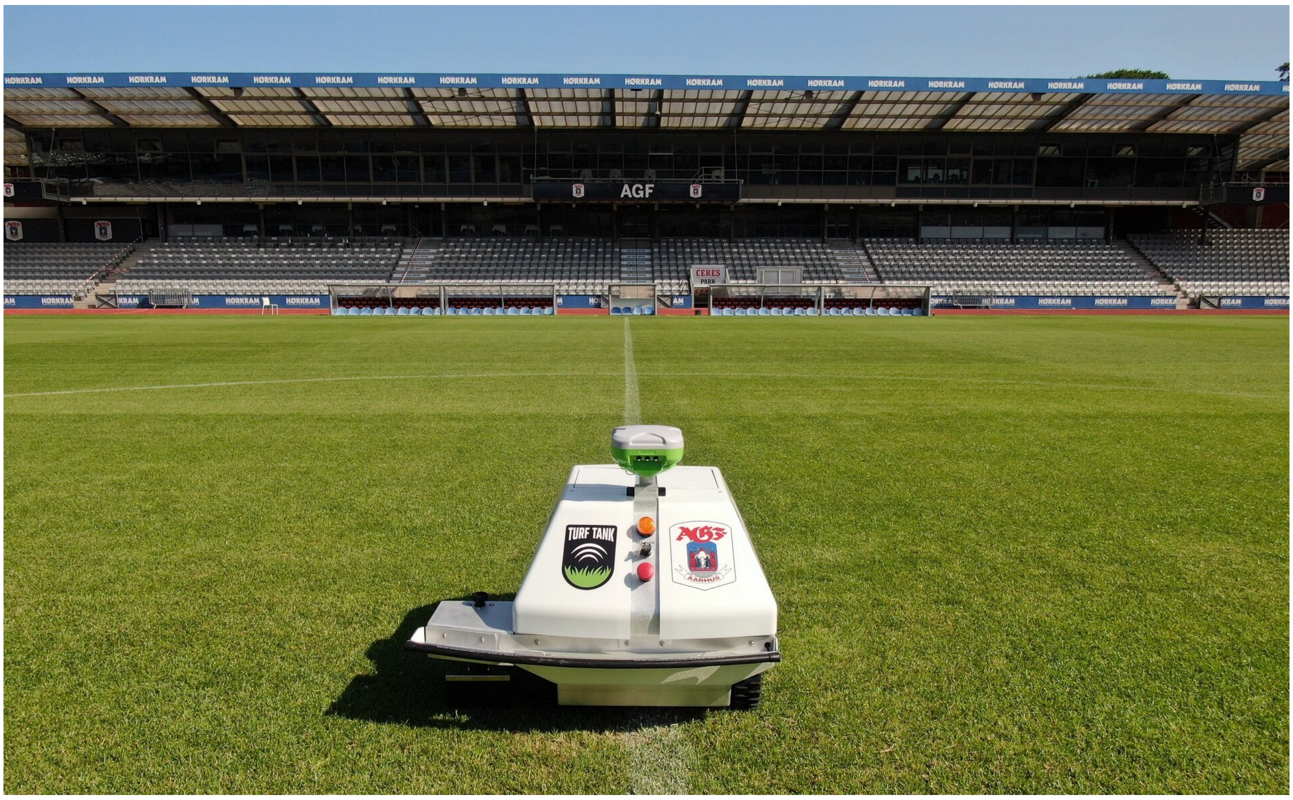


Figure 2.2: Turf Tank drawing a football field [1].

Through the process of automating the line-marking of sporting fields, the potential for human errors is substantially mitigated, resulting in enhanced line precision and a reduction in the time required for field delineation [2]. It was reported in the market watch news paper [3] that a Turf Tank robot can paint an entire American football field in 2 to 3 hours, where it would have taken 8 to 10 hours to do it manually. Whereas, for a European football field it takes below 30 minutes for the Turf Tank to draw the entire field versus 2 to 3 hours manually. Therefore, the Turf Tank robots bring down the manual labour hours used to maintain these sporting arenas and cuts down cost.

2.1.2 Sport agencies

The sports agencies are the people who have outdoor sports fields (football, rugby, baseball etc.), that has to be maintained, these sporting fields could be placed in the parks around cities, many schools have multiple fields for recess, and professional sports teams have both their main arena but also training grounds in the neighbourhood. Their interest in the project, is to cut down the costs of their sports field maintenance. One of the more labour-intensive tasks is the line-marking of these fields, where in a manual sense a human, can go around and try to accurately draw the straight lines that these fields require. The sport agencies have a large power over the project as well since they are the people who decides, if their fields will be marked by Turf Tank's robots.

2.1.3 Maintenance Crew

The maintenance crew are comprised by individuals responsible for up-keeping the sports arena, training grounds, and other related facilities. Their keen interest in a product such as Turf Tank Two lies in its capacity to minimise errors, reduce workload, and save time, when it comes to line-marking larger sports fields. This efficiency not only allows them to be more meticulous in their other work but also empowers them to accomplish a greater number of tasks throughout the day. The maintenance crew's power lies in that they are the people that will use the product, therefore if the product becomes tedious to use or brings any danger of mental or physical harm to them, they would choose a competitors line marking unit.

2.1.4 Competitors

Turf Tank competes with companies like TinyMobileRobots, another Danish-based line marking robot, and others in the same field. Turf Tank is highly interested in keeping a close eye on its competitors to ensure it can stay ahead of the curve by offering superior technology in the market.

On the flip side, competitors have limited influence over Turf Tank. Their ability to affect Turf Tank's decisions is minimal because Turf Tank's actions are primarily guided by the forces of the free market, where customer choices and market dynamics reign supreme.

Table 2.1 presents the stakeholder analysis table, a method that lists the most important stakeholders and describes their influence on the project. All stakeholders except for competitors were chosen because they do not directly contribute to the project, as explained earlier.

Stakeholder analysis table			
	Turf Tank	**Maintenance crew**	**Sports agencies**
Stakeholders' interest in the project	Growing business	Easier workflow	More optimized version of maintenance
Stakeholders requirements	Financial viable solution	Does not bring any mental or physical harm	The solution will save time for there employees
Stakeholders' contributions	Implements the robot into the real world	Deliver feedback from how the robot works	Adoption of the turf tank
Influence on the project	Sets the limits for how much, money they want to spend on the project	They can provide feedback for how they are using the project	They provide money for the people that provide the money to the project
Stakeholder motivation	Wants to expand and grow their market reach	Less physical work	Reduce money used on labour and more precise lines on the field
Possible conflicts		The product will be annoying to use	The product will not live up to their expectation
Participation	Knowledge and capital	UX and UI testing	A place for testing

Table 2.1: Analysis of the three most important stakeholders for this project

Two requirements are apparent among all stakeholders: the need to optimise the time required for marking lines for sporting events, such as a football field, and the importance of accurately placing these lines. Turf Tank has developed a robot capable of significantly reducing the time required for marking, while ensuring precise placement.

Turf Tank's primary customers are the sports agencies that oversee sports arenas and training grounds requiring regular maintenance. Therefore, the sports agencies hold significant importance as stakeholders and should be closely managed throughout the project.

Even though the maintenance crew hold little power over the project, they still influence the sports agencies in their decision to buy the product, in the sense that they are the end user that is in charge of the day to day use of the Turf Tank.

As the project progresses, effective communication and collaboration with these stakeholders will be crucial to address their needs and potential conflicts. By aligning the project goals with the interests of these stakeholders, Turf Tank can continue to grow its business, reduce manual labour, and provide a reliable solution for maintaining sports fields.

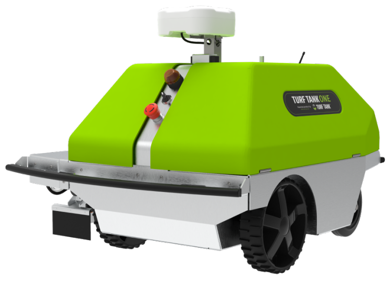
2.2 Turf Tank

This section focuses on the Turf Tank Robot, the existing line marking models will be presented and a general software framework overview will be provided.

2.2.1 Hardware

Turf Tank currently has 3 different line marking robots, where the Turf Tank One is the oldest design. The second generation robots, Turf Tank Lite and Turf Tank Two are general upgrades of Turf Tank One, such as maintenance improvements for changing battery and paint, more precise sensors, and easily configurable spray module. But other than design changes and better resolution sensors, the Turf Tank One, Two and Lite are physically similar to the extent where the same software can run on all Turf Tank models. Therefore from this point on, the phrase Turf Tank robot refers to either of the robots. It is however assumed that testing will be done on a Turf Tank Two robot, but the developed solution should work on all models. The three Turf Tank models can be seen on Figure 2.3.

[4] **(a)** Turf Tank Lite.


[5] **(b)** Turf Tank One.

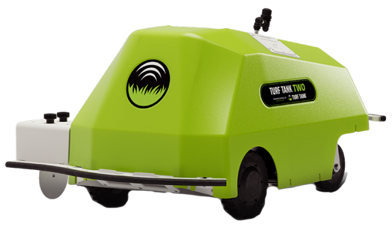

[6] **(c)** Turf Tank Two.

Figure 2.3: Three Turf Tank platforms.

Turf Tank Two is the 2nd generation autonomous line marker robot released by Turf Tank. According to Turf Tank, the robot weighs 56kg, the fully loaded robot with paint weighs around 100kg, it has a battery life of 5+ hours (680Wh), which Turf Tank claims is enough to mark 11 football fields without recharge. The robot spray paints through a single interchangeable nozzle with a solenoid valve, and a 20L paint container enough for painting one American football field [6]. The spray module can easily be customised by adjusting spray height and width, and the nozzle can be switched for other types [7]. The Turf Tank Two has a operating speed of 1m/s and transit speed of 1.4m/s when moving between fields. The Turf Tank Two can finish a standard American football field in 3 hours and 29 minutes [6]. The Turf Tank Two robot has an accuracy of ± 1 cm. [7]

The robot has Real Time Kinematic (RTK) - Global Navigation Satellite System (GNSS) positioning which with a carefully and repeatably positioned Base Station receiver, can accurately localise the robot, as seen on Figure 2.4. The GNSS operates on a 10Hz frequency and has a range of about 1km. It is possible for the robot to utilise the Continuously Operating Reference Station (CORS) network instead of a local Base Station, but sufficient global coverage is not guaranteed. [8]

Figure 2.4: Turf Tank Base Station for RTK - GNSS positioning [8].

According to Turf Tank, the robot looses significant accuracy when operating under trees or enclosed stadiums, because the signal strength is greatly reduced. The robot receives RTK pose from line of sight Ultra High Frequency (UHF) signal, from the Base Station¨which itself needs a GNSS connection. [7]

The Turf Tank two has in addition to the GNSS localisation, a set of encoders with for the differential drive dc-motors. Coupled with a 9 Degrees of Freedom (DoF) Inertial Measurement Unit (IMU) consisting of an accelerometer, a gyroscope, and a magnetometer, the robot is able to precisely localise itself. The plastic cover has Bumper sensors stopping the robot in case of collision.

User Interface (UI) is handled by a Samsung tablet, connected the robot over an internal Wi-Fi receiver. The onboard controller runs on a Raspberry Pi 4 (CM4).

According to Turf Tank the Turf Tank Two is lent to customers on a subscription basis of $16.000 US annually, for typical US costumers depending on level of subscription. 85% of all sales are by subscription basis, with currently around 2000 Turf Tank One, Lite, and Two robots operating around the world. [7]

It is evident that the Turf Tank robots are advanced, and incorporate several sensors and actuators. In order to fuse these devices into an intelligent robot which solves problems for customer, a great software stack is used.

2.2.2 Software

The robots made by Turf Tank, are based on the open-source framework, Robot Operating System 2 (ROS2). ROS2 provides a set of software libraries and tools for building robot applications. The ROS2 environment can collect all sensor data in an accessible way for processes to send and receive data, making it convenient to develop a sophisticated robotic system. Turf Tank use a custom ROS2 Galactic distribution.

The general system is split up into an online control part, and an offline application part. Localisation and robot control is handled online by an Extended Kalman Filter (EKF) with 12 states, controlling the current delivered to the two DC motors. The EKF is encompassed by a Model Predictive Controller MPC running at 100Hz, following a trajectory segment.

Turf Tank has provided a soccer field template in the same format that they use for their actual robots.

The template is represented in JSON, which consists of a sequence of lines and arcs. Lines are represented with start and end coordinates, while arcs are represented with a start point, center of rotation, and a sweep angle that determines how far along the arc the path segment is. For each segment an actuation parameter is assigned, that determines the level of activation of the sprayer unit on the robot. This template only represents the paint segments which should be painted on the ground, and not any transitions between these paint segments, nor is there any velocity assigned to these paint segments.

The trajectory generator generates the combined trajectory, resulting in the full path is generated offline from a template either placed or drawn by the user on the tablet. The trajectory is generated with an offset to the template such the painting nozzle will be centred to the template path. The full trajectory is separated into trajectory segments, containing all information needed for the robot to peform the trajectory segment. A service called `/trajectory_controller/start_traversal` with a service interface including `Trajectory` as input and `total_time` and `success` as output is called to get the robot to follow the sent trajectories.

Below, the structure of the nested ROS2 messages can be seen. The first message; `Trajectory` is the message the service gets as input. The trajectory message contains all the message data in the tree.

```
tt_trajectory_msgs/Trajectory trajectory
├── Path path
│   └── Segment[] segments
│       ├── string id
│       ├── string label
│       ├── tt_geometry_msgs/Curve curve
│       │   ├── CurveType type
│       │   │   └── uint8 value
│       │   ├── Line[<=1] line
│       │   │   ├── geometry_msgs/Point start
│       │   │   └── geometry_msgs/Point end
│       │   └── Arc[<=1] arc
│       │       ├── geometry_msgs/Point start
│       │       ├── geometry_msgs/Point center
│       │       └── float64 sweep_angle
│       └── tt_geometry_msgs/LinearDirection direction
│           └── int8 value
├── ActuationProfile actuation_profile
│   └── PathActuationParameter[] parameters
│       ├── float64 arc_length
│       └── Actuation actuation
│           └── uint8 value
└── VelocityProfile velocity_profile
    └── PathVelocityParameter[] parameters
        ├── float64 arc_length
        └── Velocity velocity
            ├── VelocityType type
            │   └── uint8 value
            └── float64 value
```

The offline planner inputs all the information for each trajectory segment and saves all segments in an array with actuation- and velocity profiles. This structure will be mimicked for the project in order to directly apply the solution on the Turf Tank platform. The full structure, including constants, can be seen in Appendix B.

As the generated trajectory have some substantial problems with current generation of path and related velocity profile, it is important to understand how the path planner works currently, to understand how the message structure should be altered for the solution to be implemented. [7]

2.3 Path Planning

Turf Tank uses an offline path planner to plan the entire path of the robot prior to the execution. The Turf Tank robot have an offset paint nozzle, on the right side of the robot, along the common wheel axis. This entails that turning during painting is asymmetrical and a right corner is different from a left corner. As the paint does not dry immediately, it is undesired for the robot to cross previously painted lines. Therefore due to these constraints the preferred line segment order, is to paint clockwise around e.g. a football field.
During operation, the robot is painting homogeneous lines and curves with a constant width. These lines and curves constitute the final image, however due to the path having sharp corners or possibly being discontinuous, a complete path between two positions and heading directions, must be generated. Turf Tank uses Dubins paths to solve this problem. [7]

2.3.1 Dubins Curves

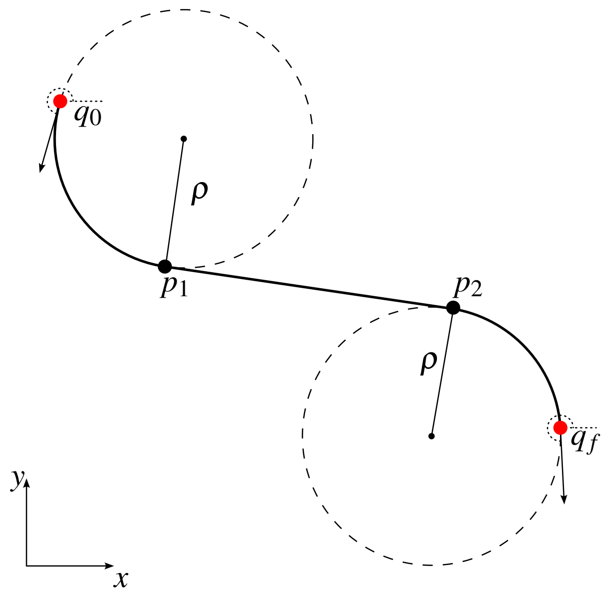


Figure 2.5: LSR Dubins path between two configurations q_0 and q_f where ρ is the minimum turning radius.

A complete and continuous path is generated by connecting the line segments with Dubins Path. The Dubins path considers physical- as well as minimum turning radius constraints. The objective is to find the shortest intermediate path between the two configurations:

$$q_0 = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix}, \quad q_f = \begin{bmatrix} x_f \\ y_f \\ \theta_f \end{bmatrix} \tag{2.1}$$

Where the path curvature is limited by $\rho = \frac{1}{\kappa}$ where ρ is the minimum turning radius[1] of the robot and κ is the cuvarture. [9] Different combinations of heading angles exist, which evidently alters the optimal path and turning direction for the robot to take. The optimal path can be categorised into 6 different configurations, RSR, RSL, LSR, LSL, RLR, and LRL, where R is short for 'right turn', S for 'straight' and L for 'left turn', the unit turns R, L, and S are refereed as controls. An example of a LSR Dubins path can be seen in Figure 2.5. The types LSR and RSL are only valid if $|q_f - q_0| > 2\lambda$. Note that the types LRL and RLR does not contain a straight segment, but rather three arcs. These are only valid if $|q_f - q_0| < 4\lambda$. Thus different paths are valid based on the configuration combination, where one path in the set of valid paths, will be the shortest. Even though Dubins paths are the shortest path between the two configurations, it does not come without problems. One problem with Dubins paths curvature is the fact that it is not differentiable at changeover points p_{ci}.

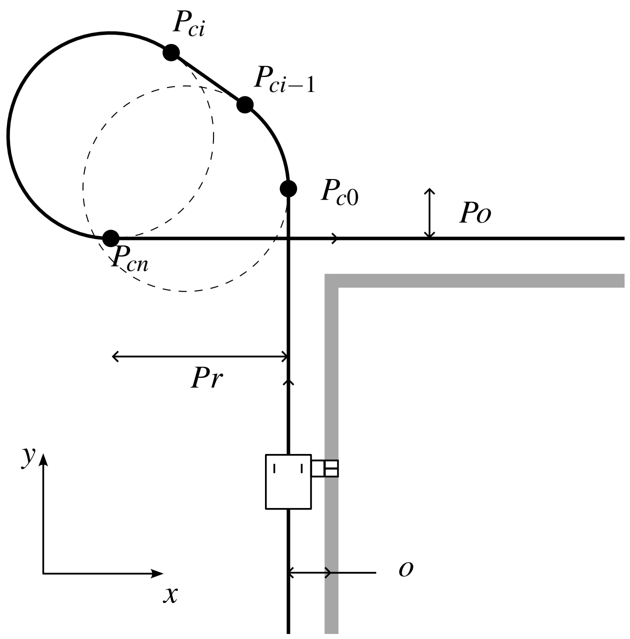


Figure 2.6: Dubins path in a right corner of a painted segment with offset o from the the path traced by the centre of the robot. The segments have prepended p_r and appended p_o padding. Additionally there are 4 different changeover points.

2.3.2 Parametric curvatures

Since Dubins path often consist of circular arcs, it is obvious that standard curvature analysis must be considered. Consider a curve ℓ specified parametrically by $\mathbf{c}(t)$ in the plane which at point $\mathbf{c}(t)$ has a tangential unit velocity vector $\hat{\mathbf{T}}(t)$.

$$\hat{\mathbf{T}}(t) = \frac{\frac{dc}{dt}}{\left|\frac{dc}{dt}\right|} \tag{2.2}$$

[1] Note that for a differential drive robot, ρ is constrained to be in the interval $[0;\infty)$ where $\rho = 0$ makes the robot rotate on the spot, and $\rho \to \infty$ is approaching a straight line.

Linearly interpolating the curve in terms of the arc length, $\mathbf{c} = \mathbf{c}(s)$,where $s = s(t)$, gives the unit tangent velocity vector as:

$$\hat{\mathbf{T}}(s) = \frac{dc}{ds} \tag{2.3}$$

Where the curvature of ℓ at the point $\mathbf{c}(s)$ is the length of $\frac{d\hat{\mathbf{T}}}{ds}$ and is denoted by κ thus:

$$\kappa(s) = \left| \frac{d\hat{\mathbf{T}}}{ds} \right| \tag{2.4}$$

And the radius of the curvature is denoted as κ is the reciprocal of the curvature[2]:

$$\rho(s) = \frac{1}{\kappa(s)} \tag{2.5}$$

The curvature can be thought of as the rate of turning of the unit tangent vector $\hat{\mathbf{T}}$ denoted as $\Delta\phi$ which is the angle between $\hat{\mathbf{T}}(s+\Delta s)$ and $\hat{\mathbf{T}}$ on an interval along the curve containing s. [10] Thus we can approximate κ by:

$$\kappa(s) = \lim_{\Delta s \to 0} \left| \frac{\Delta\phi}{\Delta s} \right| \tag{2.6}$$

Which is applicable in discrete time systems. However for analytical reasons, the curvature properties are researched using the non-discrete domain. This leads to a deeper dive into the continuity of curves.

2.3.3 Continuity

By exploring the meaning behind the word ”continuous”, which can be either be applied directly to the curve itself, but also more generally to the curve itself and adjacent curves. In this section continuity is analysed with respect to the latter. When looking at the connection between curves, there is different kinds of continuity. Firstly, recall the linear interpolation of a curve in terms of the arc length $\mathbf{c} = \mathbf{c}(s)$ where $s \in [0,1]$, results in a curve between the start point p_i and p_{i+1}. Sequentially connecting m curves and linear segments[3]$c_j \in S = \{\mathbf{c}_0, \dots, \mathbf{c}_m\}$ such that the point pairs $p_{i+1} = p_{2(j+1)} \forall i \in [0,n]$, results in a sequence of curves, which for context is referred to as a path, is said to be C^0 continuous. This can be seen in Figure 2.7. This type of continuity is called parametric continuity. Therefore a path of curves and lines can be C^0 continuous if both curves intersect at the same point. However, just because a sequence of curves can be C^0 continuous, it does not entail that its derivative, which is the velocity, is continuous. This is called first-order parametric continuity C^1, and a necessary condition of C^1 at point p_i is that it has to be C^0 continuous. It is C^1 parametric continuous if $\frac{d\mathbf{c}}{ds}$ is continuous, namely if the angle between $\mathbf{T}_j$ and $\mathbf{T}_{j+1}$ is 0 and they have the same length i.e. $|\mathbf{T}_j| = |\mathbf{T}_{j+1}|$ such that the velocity does not jump at that point. [11]

In relation to the Dubins path at the points p_{ci} the path is C^1 continuous. Note that here it is important to consider not only the position, but also the heading. Looking at the position only, it is evident that the Dubins path is continuous, however determining continuity of the heading of the robot is non-trivial. To analyse this, another term called Geometric continuity G^k must be introduced. Consider the scenario where the curves c_j and c_{j+1} intersect tangentially, but where $|\mathbf{T}_2| \neq |\mathbf{T}_3|$. This results in the curvature changing abruptly based on the equation (2.4). Note that this is not in the position domain, but in the velocity domain, so it is therefore denoted as G^1 continuity which can be seen in

[2] Since the robot is modelled in the plane, torsion of a curve is neglected.

[3] A linear segment in this case can be considered a curve with curvature $\kappa(s) = 0$.

Figure 2.7. Note that when having G^1 continuity, it is possible to have the same curvature, but coming from two different directions which is only C^0 continuity, therefore a connection can be both G^1 and C^0 continuous. Then G^2 continuity is achieved by connecting curvature such $\frac{d\kappa}{ds}$ becomes feasible. [11]

Revisiting the continuity of the Dubins' path, can now be defined as C^1 and G^1 continuous, and essentially here C^1 continuity means continuity in the tangent vector and the G^1 continuity means that the curvature is not continuous but the velocity vectors are. Thus the goal of path planner for Turf Tank should incorporate G^2 continuity. Note that some ambiguities reside in the geometric continuity space, where the radii ρ_1 and ρ_2 of two circular arcs of the osculating circle can be the same, but be on two different sides of the path. This entails that even though $\rho_1 = \rho_2$ which results in $\kappa_1 = \kappa_2$, and are therefore G^2 continuous, there is still a jump in the curvature direction. Therefore because the problem of focus considers a differential drive robot moving in the plane, it is important to consider the direction of curvature, namely which side of the line the osculating circle resides. From this point forward we consider a positive curvature if the osculating circle is on the right side of the robot, and negative curvature if the circle is on the left relative to the robot heading. This eliminates the continuity ambiguities. As a result, Turf Tanks problem is primarily based on the order of geometric continuity being too low.

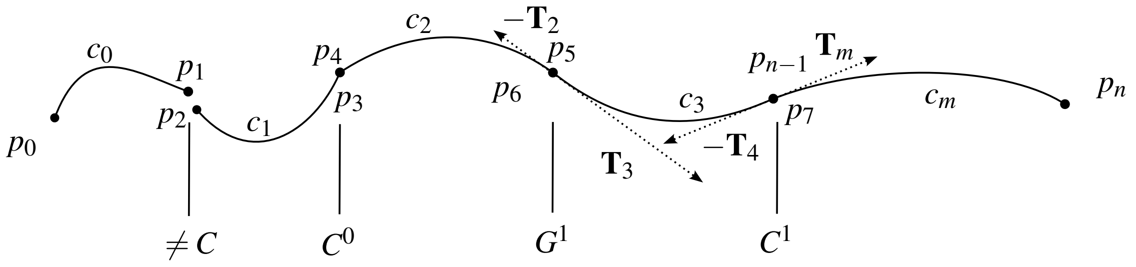


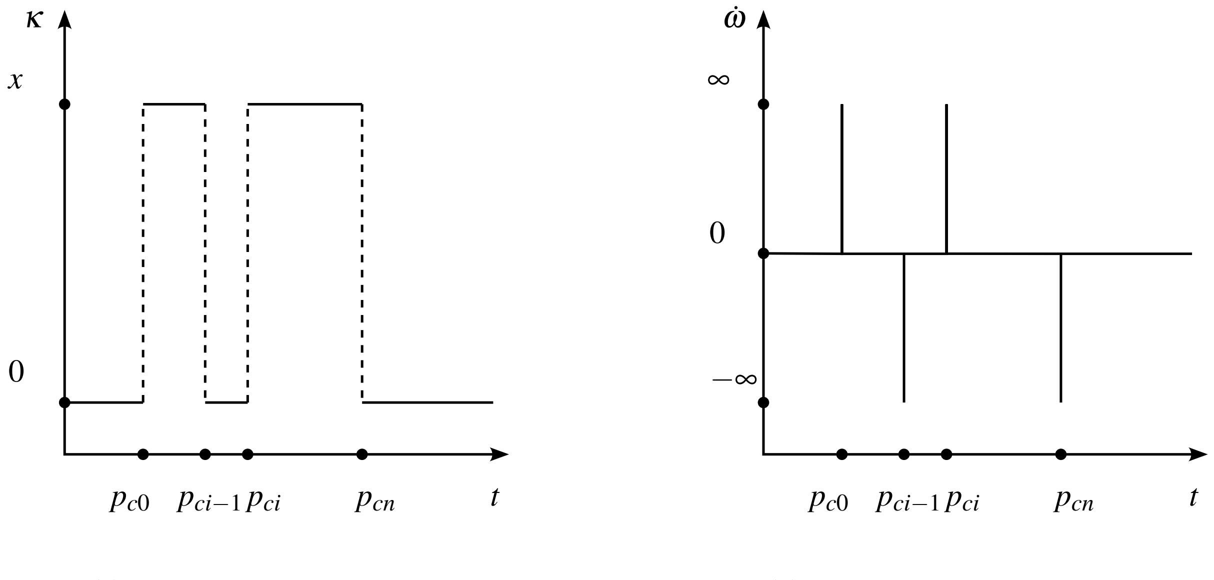
Figure 2.7: Curve sequence continuity where c_j, $j = 0 \dots m$ is a linear interpolation between point p_i and p_{i+1} $\forall i = 2j$. The C^k denotes the parametric or geometric continuity of the curves at point pairs $(p_{i+1}, p_{2(j+1)})$. $\mathbf{T}_2$ and $\mathbf{T}_4$ are illustrated in their negative direction for graphical purposes.

2.3.4 Padding elimination

As a result of the raising and lowering of the paint assembly, the robot needs to extend its path in both directions, to accommodate the lifting or lowering of the paint discs, and to toggle the nozzle. Currently this is accounted for by adding pre- and post-padding which can be seen in Figure 2.6. In Turf Tanks current solution, the pre-padding $p_r = 0.5[m]$ and the post-padding $p_o = 0.1[m]$. The main reason for the $p_r >> p_o$ is that the robot must be aligned completely and in a steady state, before initiating painting. The source of this is the discontinuities of the curvature at change over points p_{ci}, the path following is hit with a step in the curvature. The transient response of this step response before it reaches steady state, does take some time, which is the reason of the longer pre-padding p_r to line segments. As Turf Tank desires their robot path to be time optimised, this pre-padding should be minimised. As a result of the step response, the robot essentially experiences infinitely high wheel acceleration, where the mechanical upper bound for acceleration is reached.

2.3.5 High Wheel Acceleration Issue

High wheel acceleration occurs because of the step in the path curvature at change over points as depicted in Figure 2.8a. This presents a problem regarding the acceleration when the path transitions from a straight line to a curve, as illustrated in Figure 2.8b. At these changeover points, a request of infinite wheel acceleration of the system is encountered, known as the Dirac delta function, however, since this is not physically possible due to various reasons, where the main one is inertia, a transient response which have a transient response time of some value greater than zero. This relates back to the padding problem in section 2.3.4, where the padding is utilised to alleviate the transient response created by the infinite acceleration.



(a) A curvature graph of the trajectory. **(b)** An acceleration graph of the trajectory.

Figure 2.8: Two figures showing the curvature and acceleration over time of Figure 2.6.

As these issues effects the total time of which the Turf Tank robot takes to paint some template, which evidently is desirable to reduce for the customers. In the pursuit of reducing the total time and increasing the order of continuity, it is important to be familiar with existing solutions of the topic of continuously transitioning from curves to line segments and vice versa.

2.4 Curvature Continuous Spirals

This section will cover the most relevant transition curves developed for the infinite acceleration issue and showcase what change to the trajectory the solutions have. Spirals are commonly used for solving the issue of connecting two line segments, although many spirals exist. The most relevant once will be covered below.

2.4.1 Clothoid

A ”Clothoid” also known as ”Euler spiral” or ”Cornu spiral” is a special curve of which the curvature is linearly related to the arc length. The properties of the clothoid spiral is very convenient when designing turns in different transportation and engineering applications. This linear relationship between curvature and arc length allows for smoother and more gradual transitions in curves, making clothoids an ideal choice for designing corners in transportation and engineering, where precise and gradual changes in direction are essential for safety and efficiency. [12].

The clothoid is calculated by the parametric computation of the Fresnel Integral [12] which can be seen in equations 2.7 and 2.8.

$$S(x) = \int_0^x \sin(t^2)\,dt \tag{2.7}$$

$$C(x) = \int_0^x \cos(t^2)\,dt \tag{2.8}$$

A clothoid can be inserted as a curve segment between two points, e.g. a straight line and a circle, and create a linear curvature transition. Below in Figure 2.9, a clothoid is inserted between the two points s_1 and s_2 to connect the straight line with the circle. [13]

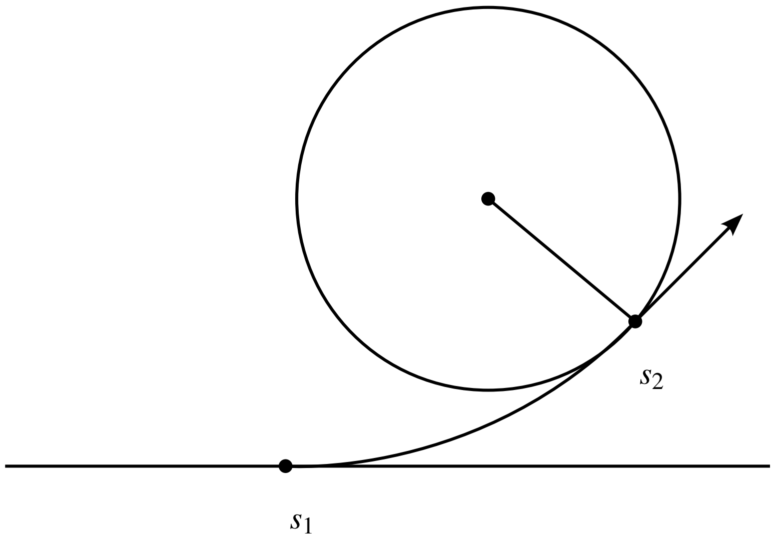
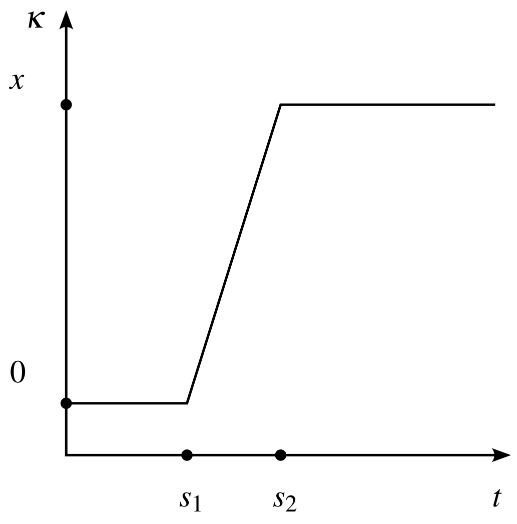


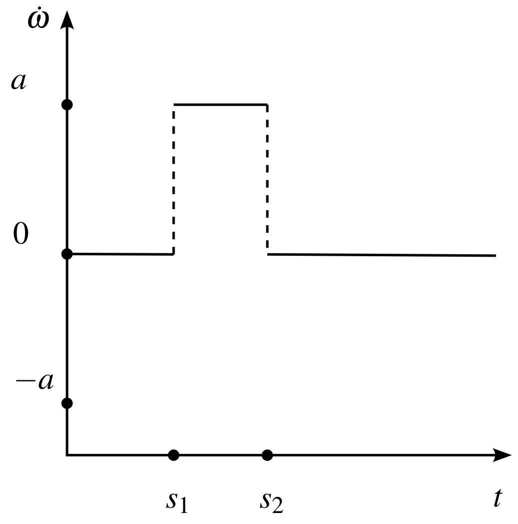
Figure 2.9: A single clothoid from a straight line to a circle.

The curvature in Figure 2.10a shows the curvature for the clothoid, and suggests that the change in curvature is linear, resulting in a smoother transition from one curvature magnitude to another. The acceleration graph, seen in Figure 2.10b shows that the acceleration is not an infinite direct delta function, but a step function with a constant acceleration, a. [13]

Since the clothoid does not have a closed form solution, meaning the clothoid has to be approximated. The better the approximation is, the more computationally heavy the approximation will be. [14]



(a) A Curvature graph of the trajectory.



(b) An acceleration graph of the trajectory.

Figure 2.10: Two figures showing the curvature and acceleration over time of figure 2.9.

It is important to note that the clothoid can be implemented for different segment configurations such as line to circle, circle to circle, line to line, etc.

Additionally, the change in curvature or σ, makes the spiral easier to model based acceleration constraints.

It is assumed the main reason to use clothoids, is because it is possible to model σ from maximum wheel accelerations. Since the rate of curvature can be estimated based on physical constraints, it is assumed the steepest possible spiral satisfying the dynamical constraints.

2.4.2 Fermat's Spiral

The Fermat spiral is a case of an Archimedean spiral, and it has a closed form solution, and limits the computation time when comparing it to the clothoid that has to compute the Fresnel's integral. The equation for the the Fermat's spiral is given by Equation 2.9 [15].

$$r = k \cdot \sqrt{\theta} \tag{2.9}$$

Where r is the radial distance, k is the scaling constant, and θ represents the polar coordinates. The transition curve resembles the clothoid method, in the transition curve shown in Figure 2.9, the properties of the Fermat spirals are as following:

- Without any constraints the Fermat spiral like the clothoid is G^2 continuous [15].
- The Fermat spiral has a zero curvature at its origin, which allows it to be connected with a straight line smoothly, without any curvature discontinuity [15].
- Since the Fermat spiral is closed loop, the computational complexity of the Fermat spiral is lowered [15].
- If you imagine that the way-points are connected in a piece-wise linear function, the Fermat spiral will deviate less from these points than the clothoid will [15].

2.4.3 Bezier curve

Another widely used path planning algorithm in robotics is the Bezier curve, which is valued for its ability to smoothly represent paths and trajectories, making it suitable for applications such as robotic motion planning and computer-aided design (CAD) software. The Bezier Curve is described by equations 2.10 and 2.11.

$$p(t) = \sum_{i=0}^{n} b_i^n(t) \cdot p_i, t \in [0,1] \tag{2.10}$$

Where P_i is the control points of the system, n is the degree of the Bezier curve and B is the Bernstein polynomial:

$$b_i^n = \binom{n}{i} (1-t)^{n-i} \cdot t^i, i \in 0,1,...,n \tag{2.11}$$

Figure 2.11 shows how a quadratic Bezier curve can be used as a smooth transitioning curve onto a Dubins path.

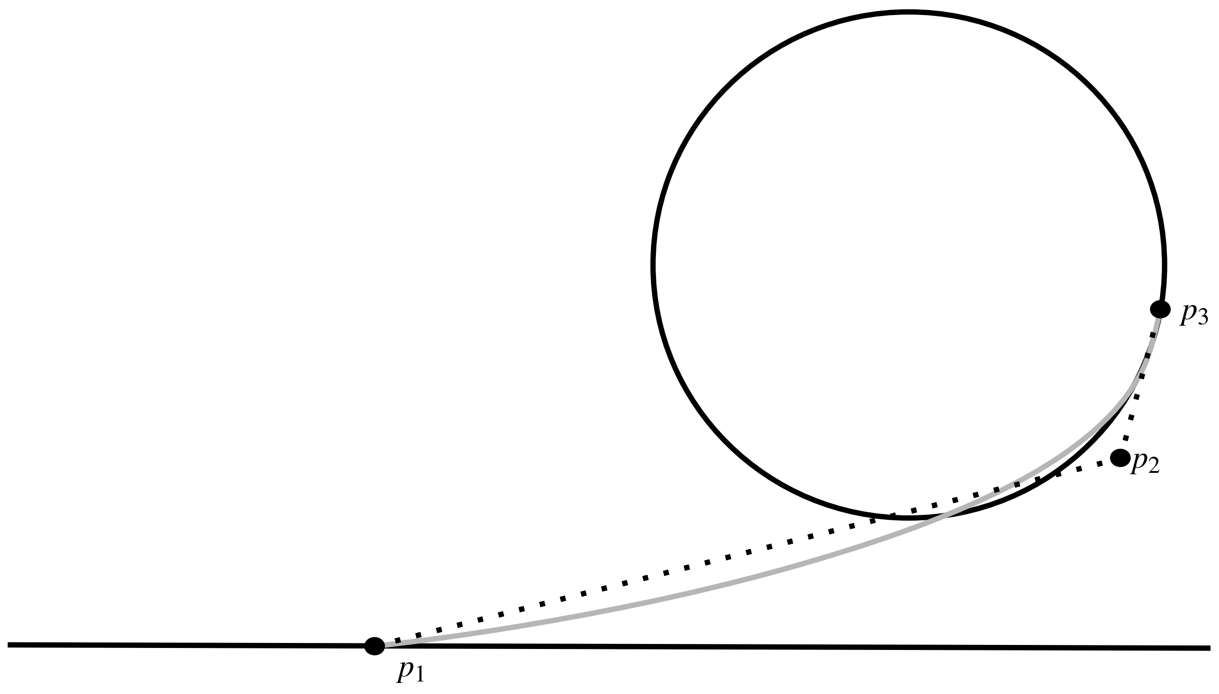


Figure 2.11: A quadratic Bezier curve employed as a smooth transition onto a Dubins path.

Bézier curves are commonly used in robotics due to their valuable properties. These properties include:

- Passing through Control Points: A Bézier curve always starts from and passes through its control points. This characteristic ensures that the curve precisely follows the specified path defined by these control points. This also means that the curve starts and finishes at the first and last control points [16].
- Tangent Lines: The lines connecting two adjacent control points on a Bézier curve are always tangent to the curve at those points. This tangency property facilitates smooth transitions and predictable motion, making Bézier curves suitable for robotic trajectory planning and motion control [16].
- Lower Computational Complexity: Bézier curves require less computational effort compared to clothoids since they do not necessitate the calculation of Fresnel integrals to generate a curve [17].
- Can be constrained: The Bézier curve can be constrained to follow the different parametric and geometrical continuities for example C^1 or G^2 [17].

To summarise, Table 2.2 shows the different characteristics for the curvatures.

Curve	Characteristics		
	Equation	Continuity G^N	Continuity C^N
Clothoid	$x(t) = \int_0^t \sin(t^2)\,dt$ $y(t) = \int_0^t \cos(t^2)\,dt$	2	1
Fermats spiral	$r = k \cdot sqrt(\theta)$	2	1
Bezier	$p(t) = \sum_{i=0}^{n} b_i^n(t) \cdot p_i, t \in [0,1]$	constrainable to 2	constrainable to 1

Table 2.2: Table showing the Curvature Characteristics from the curves shown in this section.

Comparing the three different solutions from Table 2.2. Even though the Fermat spiral has a closed-form solution, it is considered hard to model for a physical system compared to a clothoid, because of its non-linear rise in curvature. Bezier curves however are hard to constrain to G^2 continuity, and if done, it is much more constrained in completeness than the clothoid and Fermat spiral. This leaves clothoids, which have the desirable property of linear rise in curvature, and are considered the ideal spiral despite their open-form solution.

2.5 Conclusion

In the problem analysis, the stakeholders have been examined in order to understand where the stakeholders' interest lies. Next, the available robot from Turf Tank, the Turf Tank Two have been analysed to further understand how the solution will be implemented onto the robot. Furthermore, the issue of infinite acceleration between line segments have been analysed and examined, as well as existing solutions for the issue where clothoid has been chosen the ideal spiral. When modelling and implementing the solution, the knowledge gathered in this section will be crucial in order to make valid decisions, taking into consideration the requirements and how the knowledge gathered adhere to those requirements.

2.5.1 Final Problem Formulation

Based on the problem analysis, a final problem formulation for the system can be formed to describe what the final solution must solve:

How can an offline planner create a C^1 and G^2 continuous trajectory in order to paint an arbitrary template using the Turf Tank Two platform?

---

3 Requirements

This chapter outlines a series of requirements the project should fulfil in order to be deemed successful. These requirements are based on findings from the problem analysis.

3.1 Fundamental

Fundamental requirements are requirements which are of highest importance, and are thus crucial in order for the solution to be considered a success.

1. The solution must be applicable on any 2D template consisting of arcs and lines.

3.1.1 Trajectory Planner

2. The trajectory must be C^1 continuous.
3. The trajectory must be G^2 continuous.
4. The solution must be applicable for all transitions between line or arc segments.
5. The solution must output a full trajectory for a given template.
6. The maximum curvature: κ of the trajectory should be based on the dynamics of the robot.
7. The maximum curvature rate: κ' of the trajectory should be based on the dynamics of the robot.
8. The solution must find a trajectory that is complete.

3.1.2 Software

9. The solution must be applicable directly on the current Turf Tank robot, without the need for additional hardware.
10. The solution must be applicable to the current Turf Tank ROS2 framework.

3.2 Performance

Performance requirements are based on measurable metrics which determines the standard which the solution must uphold.

3.2.1 Trajectory Planner

11. The total trajectory time T_t should be minimised with respect to the existing Dubins path based trajectory T_e.
    (a) **Marginal:** T_t should at least be the same as T_e.
    (b) **Ideal:** T_t should be lower than T_e.
12. The total path length P_t excluding padding should be minimised with respect to the existing Dubins path P_e including padding.
    (a) **Marginal:** P_t must be less than 10% longer than P_e.
    (b) **Ideal:** P_t must be less than P_e.
13. The change in curvature at any point during the trajectory should be minimised.

(a) **Marginal:** The change in curvature: κ' must not surpass 15.75. This is based on the maximum angular acceleration of the robot which is computed in equation 4.15.

(b) **Ideal:** The change in curvature κ' should ideally not exceed 3.00. This is based on the undesired effect of acceleration on the line quality.

14. The max wheel acceleration $\dot{\omega}_{max}$ must be reduced.
    (a) **Marginal:** The maximum wheel velocity must be less than the current accelerations experienced during execution of a Dubins trajectory.
    (b) **Ideal:** The maximum wheel velocity must be less than half the current wheel accelerations experienced during execution of a Dubins trajectory.

3.3 Accept Tests

In this section it will be discussed, how the different requirements will be tested, to see if the solution is viable. In the following enumeration, an overview of the different tests are provided.

1. Continuity
2. Completeness
3. Path length optimality
4. Time optimality
5. Wheel acceleration

3.3.1 Test 1: Continuity

Requirement: 2, 3
The objective of this test is to determine if the solution's trajectory is both G^2 and C^1 continuous during the transitions between two lines. The test involves inspecting the full trajectory and its first time-derivative to assess C^1 continuity.

Subsequently, to ascertain G^2 continuity, the curvature of a single transition will be inspected. A successful test result indicates both C^1 and G^2 continuity, observed in throughout this test.

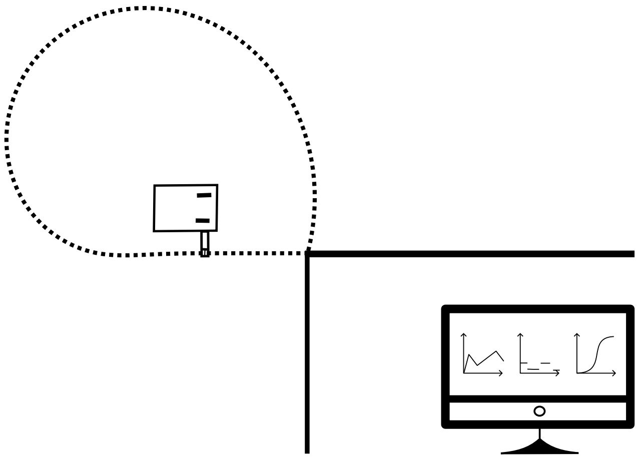


Figure 3.1: Illustration of the continuity test setup.

3.3.2 Test 2: Completeness

Requirement: 8, 4, 5
The objective of this test is to verify completeness of the path. This will be done using Monte Carlo trials, where 100 paths are generated between arbitrary poses as seen in equation 3.1, in the plane within finite bounds. Each path will be manually inspected to check if completeness is achieved.

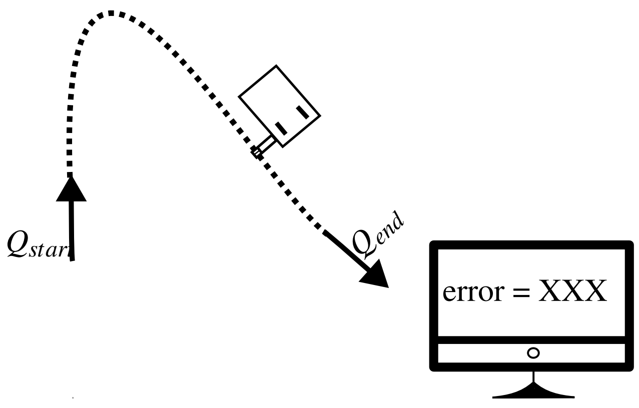


Figure 3.2: Illustration of the completeness determination.

$$q_{start} = \begin{bmatrix} x_{start} \\ y_{start} \\ \theta_{start} \end{bmatrix}, q_{end} = \begin{bmatrix} x_{end} \\ y_{end} \\ \theta_{end} \end{bmatrix} \tag{3.1}$$

The system should then be able to compute a path Π that connect these two points. The requirements are then fulfilled if it finds a feasible path which upholds the constraints given in requirement 6 and 13.

3.3.3 Test 3: Path Length Optimality

Requirement: 12 The objective of this test is to verify the length of the trajectory, and compare it to the Dubins planner. This is done to provide a good metric on the possible improvement of the planner in regards to length. This test is deemed successfull if it achieves a reduction in overall path length or a partial success if if achieves less than 10% increase in path length.

3.3.4 Test 4: Time Optimality

Requirement: 11
The objective of this test is to see if the the system is faster or slower than the trajectory generated by Turf Tank today. The test will be conducted by comparing the simulated execution time of the current Turf Tank planner, and the provided planner. If the test shows a positive result above 5% the test will be deemed successful.

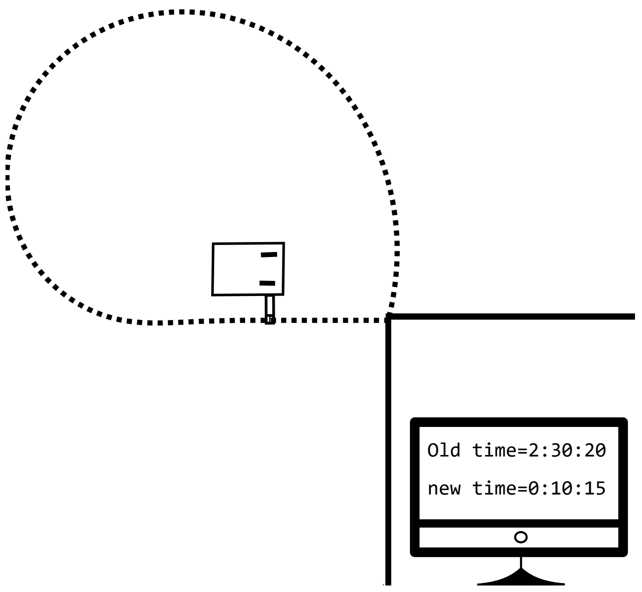


Figure 3.3: Illustration of the test setup, where the time of the total trajectory of the old and new planner is recorded and compared.

3.3.5 Test 5: Wheel Acceleration

Requirement: 14
This test is conducted to compare the wheel accelerations of the Dubins planner with the provided solution. The max wheel accelerations during a trajectory will be measured and compared. The test is deemed a partial success if a wheel acceleration reduction of $0-50\%$ is achieved, and a full success if $> 50\%$ is achieved.

4 Concept Design

4.1 Turf Tank Two Kinematics

The Turf Tank robot is a Differential Drive Robot (DDR). Thus it has two independently actuated wheels on a common axis, and two passive caster wheels for support. This has certain mechanical properties which affect how the robot can move. Modelling of the kinematics is essential in the control of differential drive robots. Initially the robot will be modelled by neglecting any dynamics and forces, the kinematics of the robot are based on the configuration space of the DDR.

The relationship between individual linear wheel velocities along the ground and the angular velocity ω of the robot is:

$$v_r = \omega(\rho + r) \tag{4.1}$$
$$v_l = \omega(\rho - r) \tag{4.2}$$

Where ρ is the signed radius of from the centre point of the robot to the centre of rotation, formally defined as the Instantaneous Centre of Curvature (ICC) or the centre of the osculating circle. Recall that the curvature of rotation is defined as $\kappa = {}^1\!/_\rho$. r is the radius of the robot. The distance between the wheels of the robot is also referred to as the wheel base, and in this case r is half of the wheel base. [18]

It is imperative to define the relationship between angular- and linear velocity of a body:

$$v_b = r_b \cdot \omega_b \leftrightarrow \omega_b = \frac{v_b}{r_b} \tag{4.3}$$

Where v_b and ω_b is the linear- and angular velocity of the body, respectively, and r_b is the radius of the body. Evidently, this is useful when the velocities of the wheels are defined in angular velocities, and when computing the linear velocity from the angular velocity of the robot. From (4.1), it is possible to compute the turning radius ρ and the angular velocity of the robot. [18]

$$\rho = r\frac{v_l + v_r}{v_l - v_r} \tag{4.4}$$
$$\omega = \frac{v_r - v_l}{2r} \tag{4.5}$$

The linear velocity of the robot, can be computed with (4.3) using the angular velocity and radius r of the robot as shown in eqaution 4.6. [18]

$$v = \frac{v_l + v_r}{2} \tag{4.6}$$

A visual representation of the kinematic model are shown in figure 4.1.

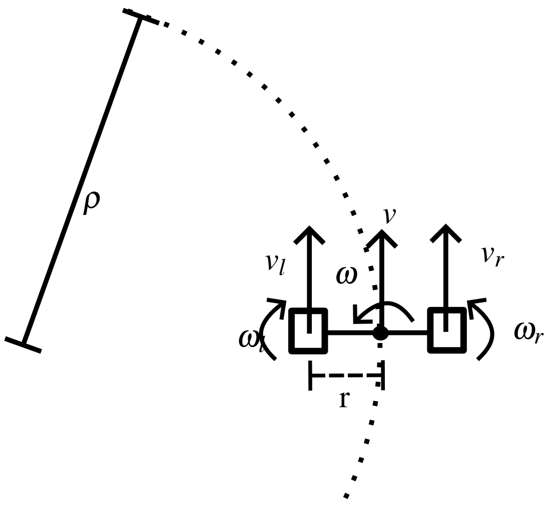


Figure 4.1: Illlustration of the kinematic model of the Robot.

Let $q(t) = \langle x(t), y(t), \theta(t) \rangle$ be the configuration of the robot at time t. (x,y) is the position of the centre point of the axis between the wheels and θ is the angle from the x-axis. Note that for notation simplicity, from this point forward, t will be disregarded. By manipulating the wheel velocities v_r and v_l, it is possible for the robot to move to different positions and orientations in the plane. The point (ICC) can be computed by using (4.4) and by knowing the velocities v_r,v_l.

$$ICC = \begin{bmatrix} x - \rho sin(\theta) & y + \rho cos(\theta) \end{bmatrix} \tag{4.7}$$

With a foundation of linear and angular velocities, the forward kinematics of the DDR robot can be defined. Consider the robot pose at every time instance t, and at the next time instance $t + \delta t$, the robot pose will be:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix} \tag{4.8}$$

Where $\dot{q}$ is defined as the next state of the robot given the time difference δ. As a result of the kinematics, it is now possible to explore the constraints applicaple specifically to the Turf Tank Two robot.

4.2 Dynamic Constants

Since the robot is a DDR, with it's kinematics derived in Section 4.1, there is some dynamical constraints of the robot which must be considered, during trajectory generation. The constraints constitute the maximum angular velocity ω_w^{max} and angular acceleration $\dot{\omega}_w^{max}$ of the wheel motors. Note that other dynamics can be modelled, like moment of inertia, noise, etc, but these are assumed to be included in ω and $\dot{\omega}$ throughout this section. Turf Tank provided the dynamic constraints for the individual wheels:

- $\omega_w^{max} = 20\ ^{\text{rad}}/_{\text{s}}$

- $\dot{\omega}_w^{max} = 40 \, \mathrm{rad}/\mathrm{s}^2$

And relevant kinematic parameters of the robot:

- $r_w = 0.1$ m
- $r = \frac{0.5079}{2} = 0.25395$ m

The maximum linear velocity of a wheel can then be defined:

$$v_w^{max} = r_w \omega_w^{max} \tag{4.9}$$

Based on this value, it is evident that if $v_r - v_l = 0$, the maximum linear velocity of the robot is equal to v_w^{max}. By saturating the angular velocities in (4.3), the maximum velocities of the robot are:

$$\omega_r^{max} = \frac{v_w^{max} - (-v_w^{max})}{2r} \tag{4.10}$$

The relation to maximum and minimum curvature is essentially not applicable as the robot is a DDR, for which the maximum and minimum turning radius is ∞ and 0 respectively. This is however based on the assumption that both the angular and linear velocities can change. However by setting a desired linear velocity of the robot, and thus fixating the velocity, it is applicable to find the maximum curvature for that linear velocity while the robot is at max speed.

with v_l, v_r capped to be below v_w^{max}, under this constraint, one of the wheels should run at maximum velocity. This entails if one is fixed at max velocity, the other should be computed like so:

$$v_l = v - (v_r - v) \text{ where } v_r = v_w^{max} \tag{4.11}$$

$$v_r = v - (v_l - v) \text{ where } v_l = v_w^{max} \tag{4.12}$$

Now that the velocities are calculated the minimum turning radius and the curvature can be calculated by equation 4.13 and 4.14.

$$\rho = r\frac{v_r + v_l}{v_r - v_l} \tag{4.13}$$

$$\kappa^{max} = \frac{1}{\rho} = 1.125 \tag{4.14}$$

Which is based on the value of ω_w^{max} and may change depending on the desired linear velocity.

4.2.1 Curvature Rate

Recall that curvature can be thought of as the rate of turning of the tangent vector of the path. Following this perspective, it can also be framed as the change of heading along the path. The change of curvature is thus based on change of heading, i.e. angular acceleration. This logically leads to the the interpretation of maximum curvature rate as the maximum angular acceleration of the robot. The angular acceleration of the robot is given by:

$$\dot{\omega}^{max} = \sigma_{max} = \frac{r_w}{2r}(\dot{\omega}_l^{max} - \dot{\omega}_r^{max}) = 15.75\frac{\mathrm{rad}}{\mathrm{s}^2} \tag{4.15}$$

where $\dot{\omega}_r^{max} = -\dot{\omega}_l^{max}$ is set to the maximum angular acceleration of the wheels. Note that this is based on the assumption that the maximum wheel acceleration is measured directly on the robot, and does therefore account for any dynamic properties of the robot. It is also assumed that there is no additional inertia during angular acceleration and linear acceleration.

4.3 Continuous Dubins Path

In order to ensure continuous curvature during transitions, a general algorithm based on the Dubins Path Planner will be made. Since, the Dubins path is optimal, the geometric way it finds finds optimal paths will be used. Between all segments in the Dubins configuration, a clothoid spiral will be inserted. This should make the clothoid modified Dubins path as optimal as possible while still satisfying C^1 and G^2 continuity.

This section will delve into the mathematical model used to construct the continuous curvature Dubins curve. The fundamental elements of how a normal Dubins path can be geometrically constructed, will be covered. The clothoid intended for ensuring G^2 continuity will be designed, and then be combined into a modified continuous Dubins path.

4.3.1 Geometric Description of Dubins Path

Between two poses, a Dubins path can be generated using statements that choose the optimal class and geometry to determine tangent points between different controls for the optimal configuration.

For CSC paths, the tangent line between the Ω_s and Ω_e circles connected to the start and end pose respectively needs to be found. For an arbitrary start and end pose, four tangent lines can be spanned, as seen in Figure 4.2. The all circles Ω have a radius of $1/\kappa$ and is tangent to the pose it is connected to.

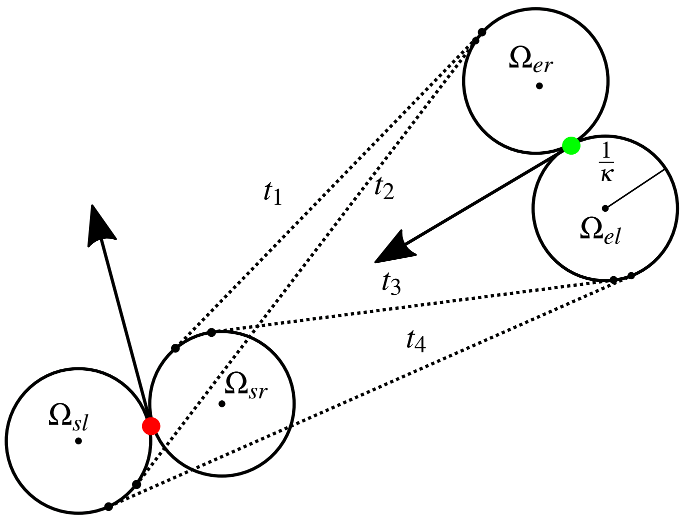


Figure 4.2: Illustrations of all four tangents for CSC paths. The start pose is the red dot, and the end pose is the green dot. s and e stands for start and end, and l and r stands for left and right. t_1 to t_4 is the tangent lines.

For CCC paths, a tangent line cannot be spanned as Ω_s and Ω_e is connected with a third circle Ω_t for tangent, meaning tangent points between Ω_s, Ω_r and Ω_t has to be calculated instead. This can be seen in Figure 4.3.

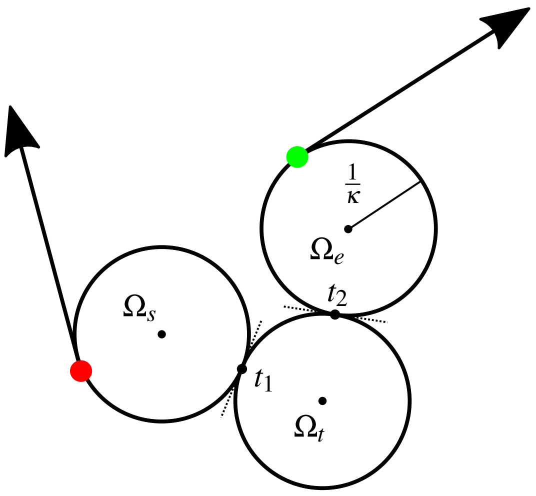


Figure 4.3: Illustrations of a RLR path where Ω_t is the tangent circle connecting Ω_s and Ω_e. t_1 and t_2 are the tangent points. The start pose is the red dot, and the end pose is the green dot.

From a given start and end pose, all six Dubins configurations will be solved to find the parametric distance of all of them, which for arcs are the angle between the entry and exit point, and for lines are the length of said line, from the start to the end pose. It is stated in section 2.3.1 that a CCC path can only be valid if $|q_f - q_0| > 2\lambda$, so if the start and end pose is further away than 2λ, then CCC paths should be disregarded. From all valid parametric lengths, the shortest will be selected to be the optimal configuration. The six configurations can be seen below in Figure 4.4.

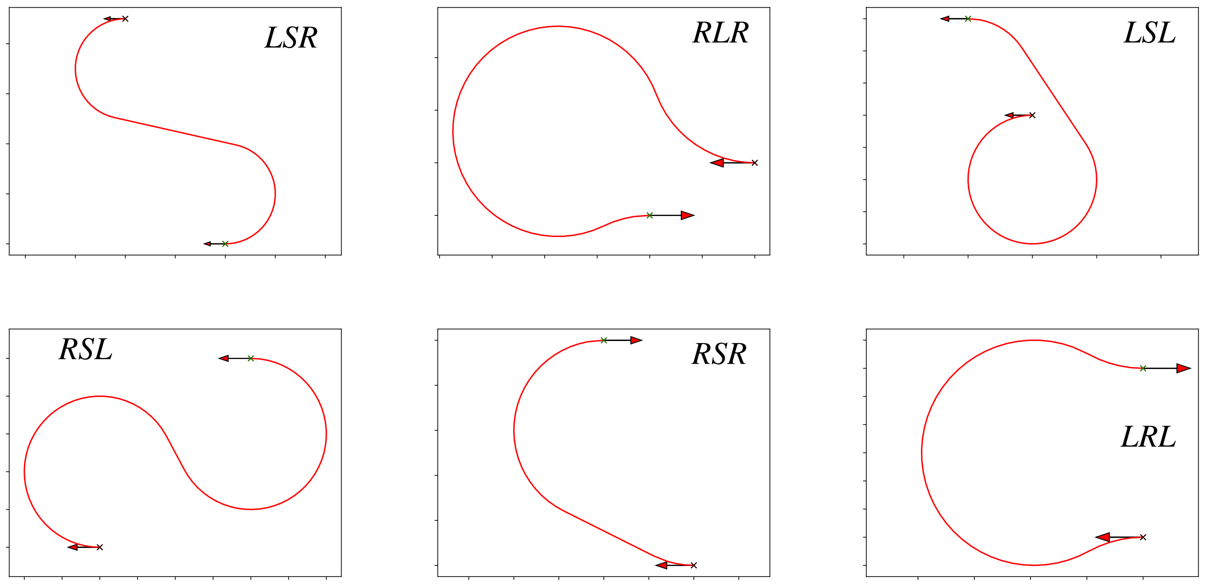


Figure 4.4: All six Dubins paths.

4.3.2 Continuous Curvature Control

Since the unmodified Dubins path are discontinuous in acceleration, a Dubins curve can be modified with clothoids to ensure $G2$ continuity. Continuity can be ensured with clothoids inserted between all controls in the Dubins configuration, where R and L controls have a clothoid entering the control arc, and a second clothoid leaving the control. An example of a continuous curvature modified Dubins control can be seen on Figure 4.5.

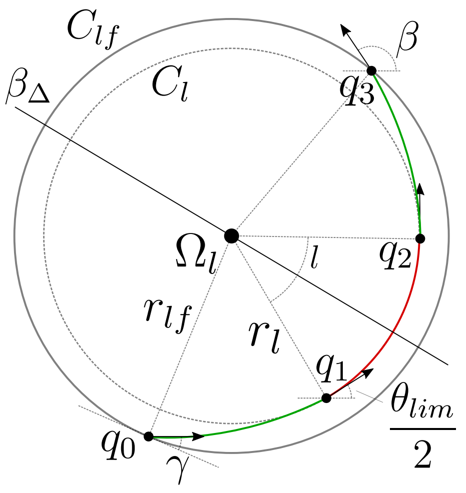


Figure 4.5: Left Dubins control with continuous curvature clothoids.

Figure 4.5 is made from the already chosen L control, with respect to the entry pose to the left control. Note that all formulas are mode from null-configuration meaning that the starting point starts at origin, the entire modified control can then later be transformed. The starting point q_0 is represented by:

$$q_0 = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \\ \kappa_0 \end{bmatrix} = \mathbf{0} \tag{4.16}$$

From origin to q_1 the first clothoid is generated by Fresnel integrals introduced in Section 2.4. The position of q_1 is spanned from the null-configured clothoid end, by Fresnel integrals C, and S:

$$C(x) = \int_0^x \cos(\frac{\pi}{2}t^2)\,dt \qquad S(x) = \int_0^x \sin(\frac{\pi}{2}t^2)\,dt \tag{4.17}$$

Where $C(x)$ and $S(x)$ are the x- and y-coordinates, respectively, for the clothoid end point. x is the integral range input, specifying the length of the clothoid. t is the integrated variable. The end point of the clothoid called q_1 found by the states:

$$q_1 = \begin{bmatrix} x_1 = C\left(\sqrt{\frac{\theta_{lim}}{\pi}}\right)\sqrt{\frac{\pi}{\sigma_{max}}} \\ y_1 = S\left(\sqrt{\frac{\theta_l im}{\pi}}\right)\sqrt{\frac{\pi}{\sigma_{max}}} \\ \theta_1 = \theta_{lim}/2 \\ \kappa_1 = \kappa_{max} \end{bmatrix} \tag{4.18}$$

The curvature rate of the Fresnel is directly dependant on the specified σ_{max}, derived from the robot dynamics.

From Equation 4.18, The Fresnel input $\sqrt{\theta_{lim}/\pi}$ determining the parametric length of the clothoid, containing θ_{lim}. θ_{lim} is the double difference in heading from the start of the clothoid and the end clothoid heading, as seen on Figure 4.5.

θ_{lim} can be estimated based on the dynamically constrained constant σ_{max} and the maximum curvature κ_{max}, seen in Equation 4.19.

$$\theta_{lim} = \frac{\kappa_{max}^2}{\sigma_{max}} \tag{4.19}$$

In Equation 4.18, angle of q_1 is the heading difference between one clothoid, thus $\theta_1 = {}^{\theta_{lim}}/_{2}$. The curvature of q_1 is κ_{max}, since the clothoid always end in maximum curvature.

From q_1 to q_2 the arc of radius ${}^{1}/_{\kappa_{max}}$ and a centre point Ω, seen in Equation 4.20.

$$\Omega = \begin{bmatrix} \Omega_x = x_1 - \frac{sin\theta_1}{\kappa_{max}} \\ \Omega_y = y_1 + \frac{cos\theta_1}{\kappa_{max}} \end{bmatrix} \tag{4.20}$$

The outer circle where the clothoid points q_0 and q_3 lie, is called R_T, with centre point in Ω, and radius found by: $R_T = \sqrt{x_\Omega^2 + y_\Omega^2}$.

The maximum curvature circle spans with an arc length of: $l = \beta - \theta_{lim}$, as seen on Figure 4.5.

The modified continuous controls are completely symmetrical around ${}^{\beta}/_{2}$, indicated by the symmetry-line β_Δ on Figure 4.5. From q_2, the last clothoid pose in the modified control can be found with a clothoid flipped on the x axis. The clothoid start at $\kappa = \kappa_{max}$ and ends with $\kappa = 0$ with linearly decreasing curvature rate, as seen on Equation 4.21.

$$q_3 = \begin{bmatrix} x_3 = -C\left(\sqrt{\frac{\theta_{lim}}{\pi}}\right)\sqrt{\frac{\pi}{-\sigma_{max}}} + x_2 \\ y_3 = S\left(\sqrt{\frac{\theta_{lim}}{\pi}}\right)\sqrt{\frac{\pi}{-\sigma_{max}}} + y_2 \\ \theta_3 = \beta \\ \kappa_3 = 0 \end{bmatrix} \tag{4.21}$$

Note the clothoid made in Equation 4.21, is made from null-configuration, inverted with $-\sigma_{max}$ and should be transformed with end point in q_2 and the 0 curvature point is the q_3 clothoid point pose.

By changing the deflection angle β the continuous heading difference between the critical clothoid points: q_0 and q_3 can be manipulated.

This method can effectively function as a normal Dubins control, with two constraints: The θ limit constraint, and the γ rotation constraint.

The θ limit constraint is broken when the two clothoids on the control are self intersecting, meaning the dashed red clothoid is moves to p_1, thereby crossing the red clothoid, as seen in Figure 4.6. This happens when the entering and exiting clothoids pass each other before reaching the inner circle, which results in the path having to wrap around the inner circle.

The two clothiods on the control are self intersecting when $\beta < \theta_{lim} = 2\theta$, resulting in the clothoids overshooting and the continuous control is no longer optimal, as seen on Figure 4.6.

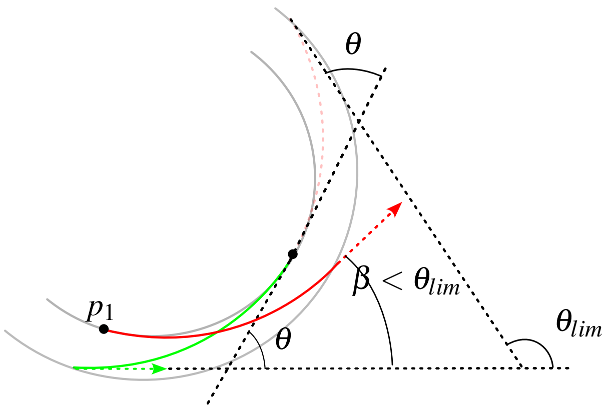


Figure 4.6: Self intersecting clothoids, resulting in suboptimal continuous control.

The γ rotation constraint is the angle between the headings of either q_0 or q_3 and the tangent to the R_T circle, seen on Figure 4.5. γ is always the same for both clothoid points on the control, and can be calculated by the Equation 4.22.

$$\gamma = -tan^{-1}\left(\frac{x_\Omega}{y_\Omega}\right) \tag{4.22}$$

4.3.3 Combining Clothoid Control with Dubins

To incorporate the curvature continuity into the Dubins path, modification to the existing Dubins planner must be done. First, CSC paths are considered. In Figure 4.7, the grey path shows the original Dubins path, where the red path is the modified path using γ. The modification is necessary to do as the clothoid must start with a heading difference of γ from the tangent line of the circle. It is important to note that the radius of the Dubins curves must not be $\frac{1}{\kappa}$ but R_T from Figure 4.8. This makes space for the clothoid to curve from C_{ef} to C_e to reach κ for instance.

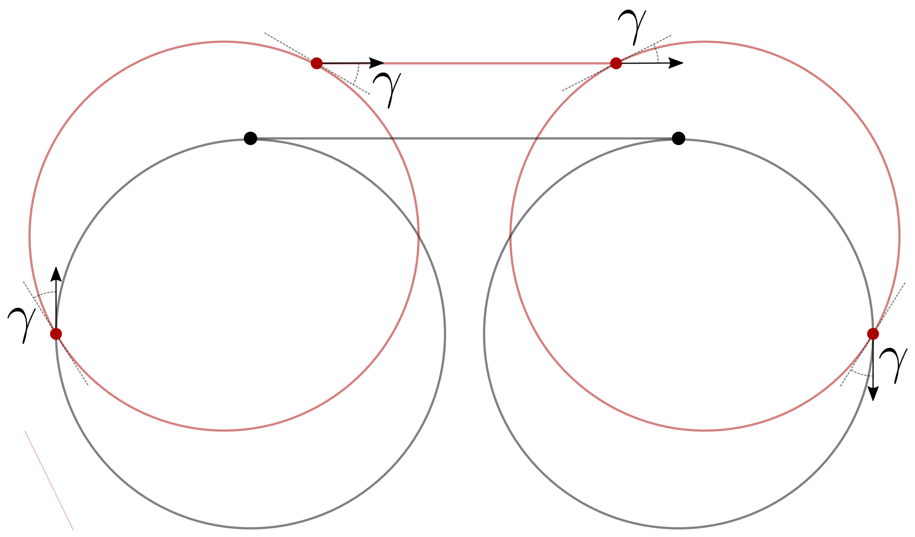


Figure 4.7: Dubins RSR configuration as the gray path and the modified Dubins RSR configuration as the red path.

Considering Figure 4.8, the angle difference between the heading of q_{s0} and the tangent line to C_{sf} in q_{s0} is γ, and the Dubins path always has its heading tangent to the adjacent circle, a temporary heading modification of γ must be made to generate the Dubins path with the properties for a clothoid to be generated. This insures the correct offset when generating the new Dubins path with clothoids. The same procedure is done for q_e3. Finding the exit pose on C_{sf} and entry pose on C_{ef} is done by adding γ to the original Dubins exit angle, which will end up at q_{s3} for instance. Connecting C_{sf} and C_{ef} with the control S is done by offsetting the exit point for C_{sf} and entry point for C_{ef} with γ. As doing so will lower the distance of S, S has to be recalculated, using the formula $b_2 = R_T \cdot sin(|\gamma|) \cdot \kappa$ meaning that $S_{clothoid} = S - 2b_2$. b_2 is calculated using the sinus relations on the triangle spanned between Ω_e, R_T and the angle γ. The four critical poses referred to as clothoid poses, being q_{s0}, q_{s3}, q_{e0}, and q_{e3}, can be used to generate the full path. The last step is to find the poses on C_s and C_e and calculate the arc spanning between q_{s1} and q_{s2} as well as q_{e1} and q_{e2}.

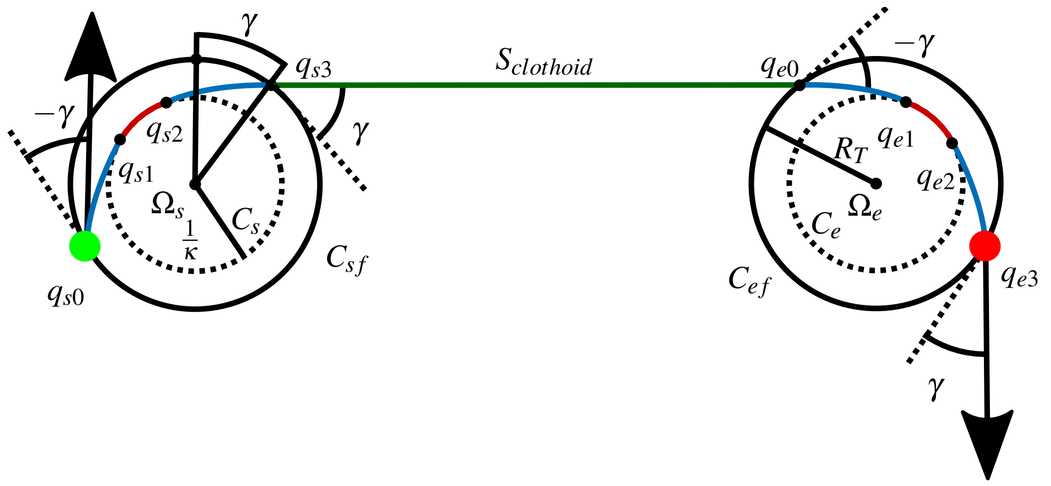


Figure 4.8: Curvature continuous RSR configuration.

It is important to notice the importance of γ as it modifies the entry and exit poses for Ω_s and Ω_e to allow for continuous curvature. With this modification, the correct conditions discussed in section 4.3.2 is fulfilled with the correct offsets between circles and headings of γ.

For CCC paths, there is no tangent line, as the control S is invalid. Considering Figure 4.9, the connecting poses from Ω_t must have an offset of γ with respect to the tangent line to Ω_t.

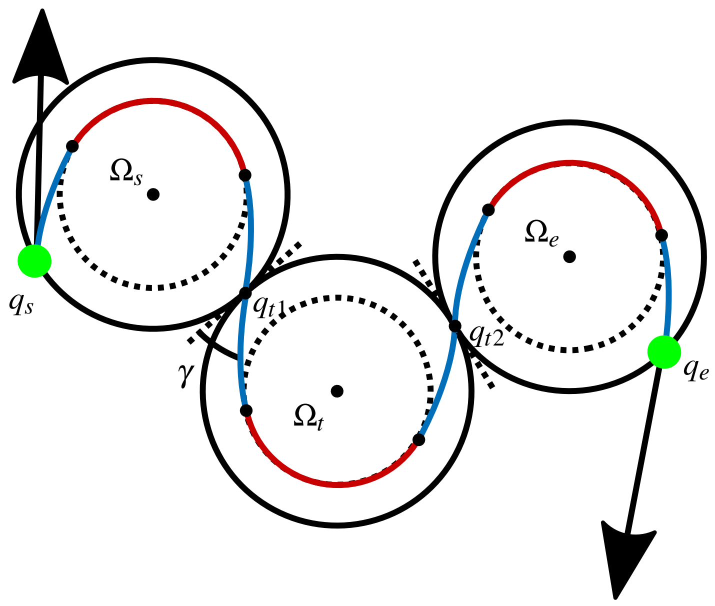


Figure 4.9: Curvature continuous RLR configuration.

4.4 ROS Architecture

Turf Tank currently utilizes a ROS framework, as detailed in Section 2.2.2. In order to meet the requirement for C^1 and G^2 continuous paths for Turf Tank robots, a new node will be added to the Turf Tank ROS environment, as described in Section 4.3. This node's purpose is to take a path, such as a football field, in its template form without transits, as input. The output will be a path with added transits achieved by incorporating transition curves between what was previously only Dubins' paths.

The ROS2 environment will be integrated into the offline planner developed by Turf Tank, responsible for generating the entire trajectory. The new node will take this trajectory as input and refine it into a smooth trajectory for the robot to follow.

It is crucial that the developed node conforms to the existing Turf Tank ROS2 message structure, as outlined in Appendix B, to ensure seamless integration with the Turf Tank robot's existing capabilities. Moreover, the developed node must exhibit flexibility, allowing the Turf Tank team to create templates for various sports applications within their current catalogue without the need for designing a new path node. Further details on the setup and functionality of these nodes will be explored in the Implementation section.

5 Implementation

The node tree in 5.1 shows a simplified version of the developed ROS architecture that will be added to the Turf Tank robot. Figure 5.1 shows that instead of the controller sending information to the robot it is sent to a simulation environment used in this project to test the robots response to the trajectory given. This section covers how the different ROS nodes are managed and how they work.

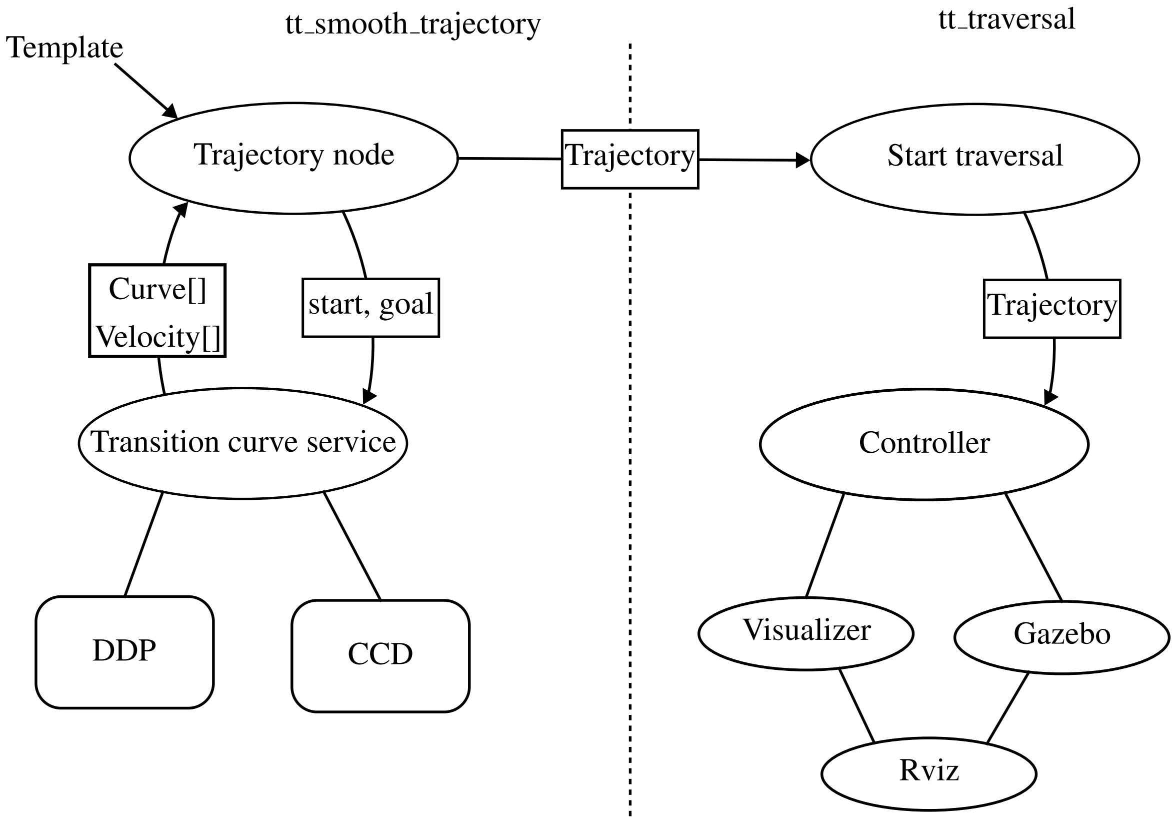


Figure 5.1: ROS Package architecture.

5.1 Offline Planner

The purpose of the offline planner is to generate a trajectory for a given template. This trajectory will be executed by the controller. The Offline Planner is supposed reflect how Turf Tank's ROS stack executes pr-planned trajectories.

5.1.1 Trajectory Node

A template in JSON format is loaded in and parsed to a dictionary format to make path modifications easier. Each line or arc in the template is referred to as segments. A plot of a 1×1 m square can be seen below in Figure 5.2.

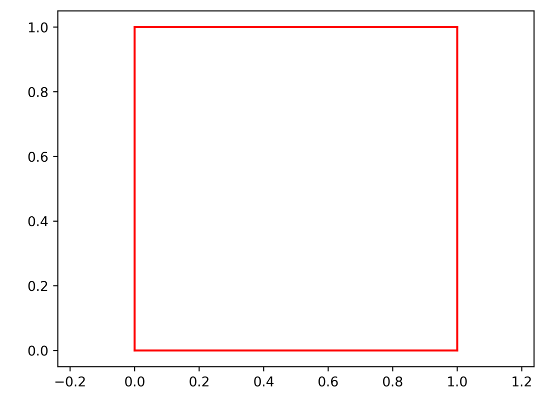


Figure 5.2: 1×1 m square template path.

The template needs to be transformed with respect to the robot model to align the paint nozzle with the template. As the offset $k = -0.41124$ for the y-axis in the local robot frame (x is pointing straight forwards), the transformation for all segments is done using:

$$\begin{aligned}
x_{offset\_start} &= x_{start} + \cos\left(\theta + \frac{\pi}{2} \cdot |k|\right) \\
y_{offset\_start} &= y_{start} + \sin\left(\theta + \frac{\pi}{2} \cdot |k|\right) \\
x_{offset\_end} &= x_{end} + \cos\left(\theta + \frac{\pi}{2} \cdot |k|\right) \\
y_{offset\_end} &= y_{end} + \sin\left(\theta + \frac{\pi}{2} \cdot |k|\right)
\end{aligned}$$

Where the current segment S is a line with start position x_{start}, y_{start} and end position x_{end}, y_{end} and $\theta = atan2(y_{end} - y_{start}, x_{end} - x_{start})$.
The offset template can be seen below in figure 5.3.

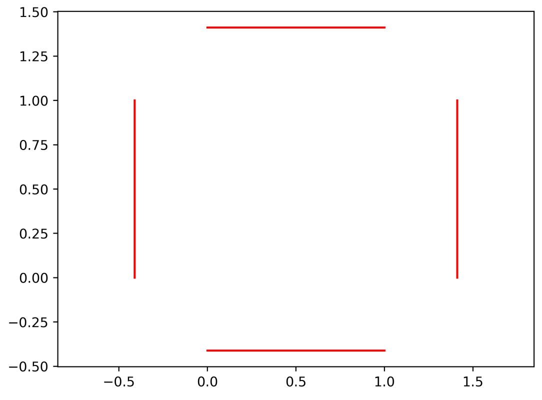


Figure 5.3: 1×1 m square offset by k.

For every segment, a pre- and post padding must be connected to said segment. For S, the pre padding is spanned using:

$$
\begin{aligned}
x_{prepad\_start} &= x_{start} - \cos(\theta) \cdot l_{prepad} \\
y_{prepad\_start} &= y_{start} - \sin(\theta) \cdot l_{prepad} \\
x_{prepad\_end} &= x_{start} \\
y_{prepad\_end} &= y_{start}
\end{aligned}
$$

And finding the post padding is done similarly using:

$$
\begin{aligned}
x_{postpad\_start} &= x_{end} \\
y_{postpad\_start} &= y_{end} \\
x_{postpad\_end} &= x_{end} + \cos(\theta) \cdot l_{postpad} \\
y_{postpad\_end} &= y_{end} + \sin(\theta) \cdot l_{postpad}
\end{aligned}
$$

Where $\theta = atan2(y_{end} - y_{start}, x_{end} - x_{start})$ and l_{prepad}, $l_{postpad}$ is the length of pre-padding and post-padding respectively in meters. This will be done for all segments in a template and will result in figure 5.4 below.

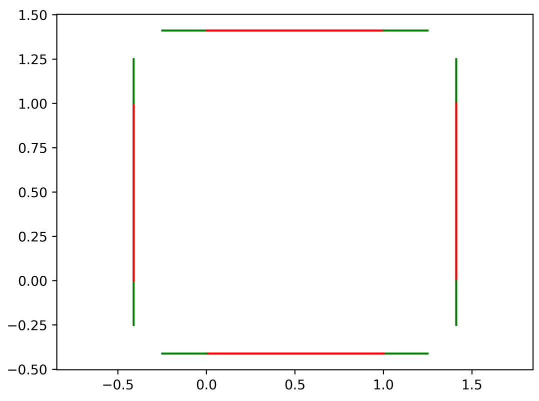


Figure 5.4: 1×1 m square with padding.

Now, each pre-padding needs to be connected to each post-padding, meaning a transition path curve must be calculated with start pose from the end of post-padding, and goal pose from the start of pre-padding.

5.1.2 Transition Curve Service

After receiving a start and goal pose, a transition path between those two poses must be generated. For the final solution, a modified Dubins Path planner is used to generate a curvature continuous Dubins Path, and because Turf Tank currently runs a Dubins Path planner, a Dubins planner made by Atsushi Sakai has been implemented as well for testing purposes.[1]

The Dubins Path Planner (DPP) plans all configurations except for RSL and LSR as they were not implemented for the Continuous Curvature Dubins (CCD) planner, between a start and end pose, and

[1] https://github.com/AtsushiSakai/PythonRobotics/blob/a141cf41807c508e58691c3c42919387e5ff9ded/PathPlanning/DubinsPath/dubins_path_planner.py

checks if the configurations was valid. For all valid configurations, parametric lengths for arcs and lines respectively are calculated. The parametric lengths describe the point at which the a segment ends. For arcs, the unit is radians, and for lines, the unit is euclidean distance. For each configuration, three parametric lengths are computed and used to calculate the cost for each configuration. The absolute value for the three parametric lengths are added together for each configuration, and the configuration with the lowest cost is chosen as the best configuration. The method of running all configurations and testing for the best one is called brute forcing, and it works well for low iterations, which it is in this case with only 6 iterations at maximum. When the best path is generated, it will be descretized as lines.

To describe the algorithm for the curvature continuous Dubins planner (CCD), consider the block diagram in figure 5.5

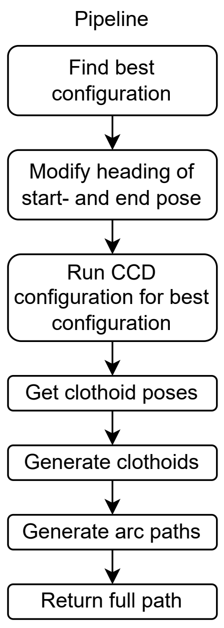


Figure 5.5: Block diagram for the CCD algorithm.

As a disclaimer, is should be mentioned that the ['R', 'S', 'L'] and ['L', 'S', 'R'] configurations were not implemented due to complications with using a geometric approach. This will only affect the optimally of the planner and not the completeness, as the Dubins planner must have at least one CSC configuration such as ['L', 'S', 'L'] to be considered complete.

The CCD assumes that the optimal configuration is the same as for the DPP, meaning the best configuration is found via the unmodified Dubins planner. From this information, a geometric approach is used to generate the path. Because the unmodified Dubins planner can be modified to enhance the properties for clothoids to be generated, the start and goal headings are modified with γ. The modification is either a addition or subtraction to the heading, depending om the configuration, as seen in the code snippet below.

```
if best_config == ['L', 'S', 'L']:
    qs_mod = [sx, sy, syaw + gamma] # start pose
    qg_mod = [gx, gy, gyaw - gamma] # goal pose
elif best_config == ['R', 'S', 'R']:
    qs_mod = [sx, sy, syaw - gamma]  # start pose
    qg_mod = [gx, gy, gyaw + gamma] # goal pose
elif best_config == ['L', 'R', 'L']:
    qs_mod = [sx, sy, syaw + gamma]  # start pose
    qg_mod = [gx, gy, gyaw - gamma] # goal pose
elif best_config == ['R', 'L', 'R']:
    qs_mod = [sx, sy, syaw - gamma]  # start pose
    qg_mod = [gx, gy, gyaw + gamma] # goal pose
```

After the modifications, the local goal is calculated. The local goal is the goal pose where the start pose is in origin. This is due to the implementation of clothoids and Dubins paths are all done from origin, and transformed to global coordinates after the path has been generated. Now the CCD configuration that match the best configuration chosen by the unmodified Dubins planner, is called. As an examples, consider the best configuration was ['L', 'S', 'L'] as both ['L', 'S', 'L'] and ['R', 'S', 'R'] are modified, where ['L', 'R', 'L'] and ['R', 'L', 'R'] did not need any modifications.

```
def _CCD_LSL(alpha, beta, d, gamma, R_T, kappa_max):
    b2 = R_T * np.sin(abs(gamma))*kappa_max
    sin_a, sin_b, cos_a, cos_b, cos_ab = _calc_trig_funcs(alpha, beta)
    p_squared = 2 + d ** 2 - (2 * cos_ab) + (2 * d * (sin_a - sin_b))
    if p_squared < 0:  # invalid configuration
        return None, None, None
    tmp = atan2((cos_b - cos_a), d + sin_a - sin_b)
    d1 = _mod2pi(-alpha + tmp - gamma)
    d2 = sqrt(p_squared) - b2*2
    d3 = _mod2pi(beta - tmp - gamma)
```

The three modifications made are at $d1$, $d2$ and $d3$.
$d1$ and $d3$ are in this case the sweep angles of the two circles generated by the Dubins planner, meaning the angle from the start and goal pose respectively of which the arc should end. This arc is extended by subtracting γ, to take the offset of γ into account. By doing so, the length of $d2$, which is the line connecting the two circles must be shorter, which is why $2 \cdot b2$ is subtracted from $d2$.
$b2 = R_T \cdot sin(\gamma) \cdot \kappa_{max}$ is the short cathetus of the illustrated triangle in Figure 5.6.

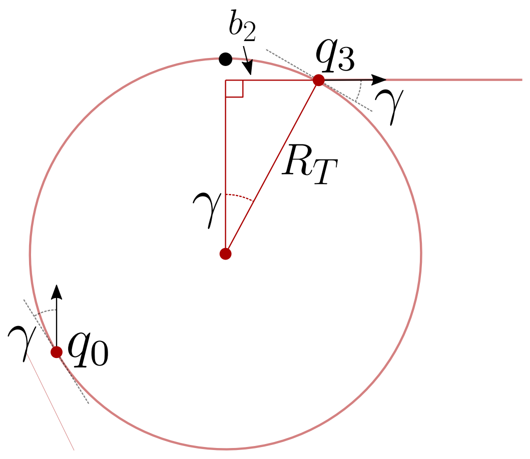


Figure 5.6: Illustration of how b_2 is calculated.

Given d1, d2, and d3, the poses of where the two circles and the line-start and -end, can be found, which is necessary to generate the clothoids. For each clothoid pose, the x, y, and heading has to be modified with γ, since change in the start and goal pose does not carry through to all poses. The clothoid poses will have correct positions, but wrong headings, so the heading will be reverted for all clothoid poses, to account for the offset of γ. Now that all the clothoid points are correct, they have to be transformed to global poses. This is done using a transformation matrix. Generating the full path is done with functions that generate the clothoids and arcs for the clothoid controls.

```
if best_config == ['L', 'S', 'L']:
    cx, cy = _transform(x_omega, y_omega, pt[0])

    _plan_left(
        pt[0], pt[1], C_path, S_path, cx, cy, transition_dict)

    _plan_straight(pt[1], pt[2], transition_dict)

    cx, cy = _transform(x_omega, y_omega, pt[2])

    _plan_left(
        pt[2], pt[3], C_path, S_path, cx, cy, transition_dict)
```

For each turn, Ω is transformed according to the relevant clothoid pose, as the position of Ω determines where the clothoids are positioned. The clothoids are descretized as lines, since the message system Turf Tank is using cannot handle clothoids. The arc between two clothoids is found, and the two turns are connected with a straight line. The full path for a ['L', 'S', 'L'] configuration is now generated. An example of a ['L', 'S', 'L'] configuration can be seen below in figure 5.7.

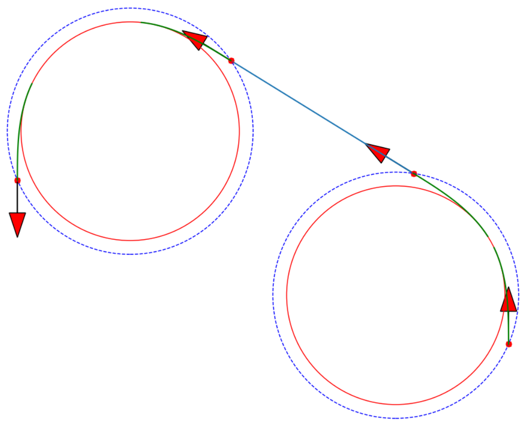


Figure 5.7: A ['L', 'S', 'L'] configuration generated using the described algorithm. The green paths are the clothoids, the red dots with arrows are the clothoid poses, and the blue line is the straight line connecting the controls.

The generated path for the 1×1 meter square can also be seen below in Figure 5.8.

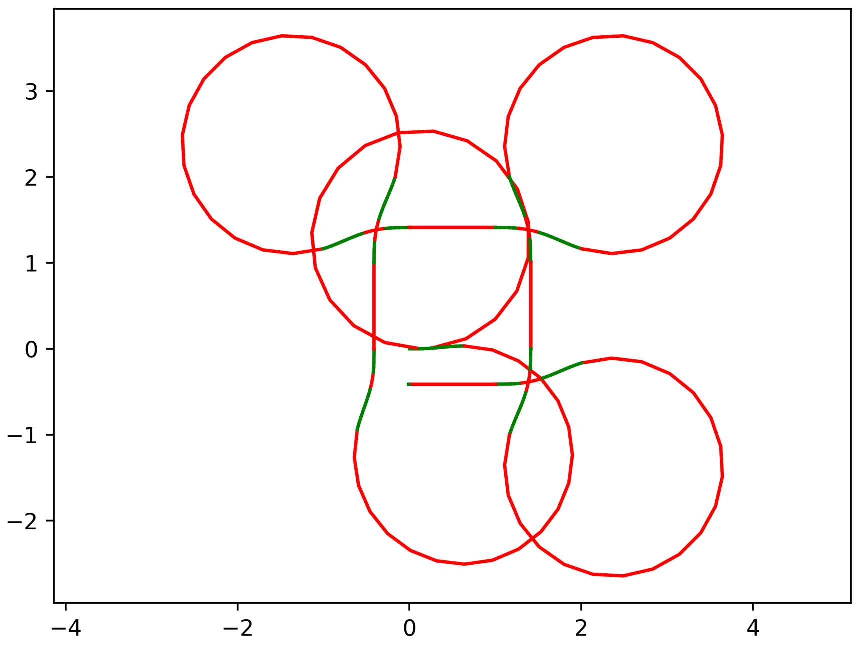


Figure 5.8: 1×1 m square with transitions between padding.

To make the path a trajectory, a velocity profile has to be attached to the path, which is simple to incorporate as the velocity is constant for all segments of the path. The trajectory will be parsed to Turf Tanks message system before the message is sent to the controller to be executed.

5.2 Online Path Follower

When the offline planner finishes creating a path, the online planner, which operates within the robot, is initiated. This planner is responsible for taking immediate actions while the robot is in motion. Therefore, once the offline planner completes the trajectory, it requests a service called 'start traversal', and passes the trajectory to it. This service node, in turn, forwards the trajectory to the robot's controller, allowing the robot to execute the specified movements. The later part is build from scratch and simulated, which is described further in Section 5.2.2.

5.2.1 Controller

To control the robot in the simulation a pure pursuit algorithm with PID control. The pure pursuit algorithm is designed to find a θ value that describes the heading difference between the desired heading and the robots heading, it finds this difference from Equation 5.1 and 5.2.

$$d = \begin{bmatrix} x_g \\ y_g \end{bmatrix} - \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{5.1}$$

$$\theta = \tan^{-1}\left(\frac{d_y}{d_x}\right) - \theta_r \tag{5.2}$$

Where $(\cdot)_{g,r}$ is goal and robot respectively. θ is angular difference.

To make sure the robot drives in a semi constant velocity, the pure pursuit algorithm is also used to find the difference between the desired velocity and the robots real velocity. This equation is described in Equation 5.3.

$$v_{error} = v_g - v_r \tag{5.3}$$

where v is the linear velocity. An illustration of the pure pursuit PID controller and its values can be seen in Figure 5.9.

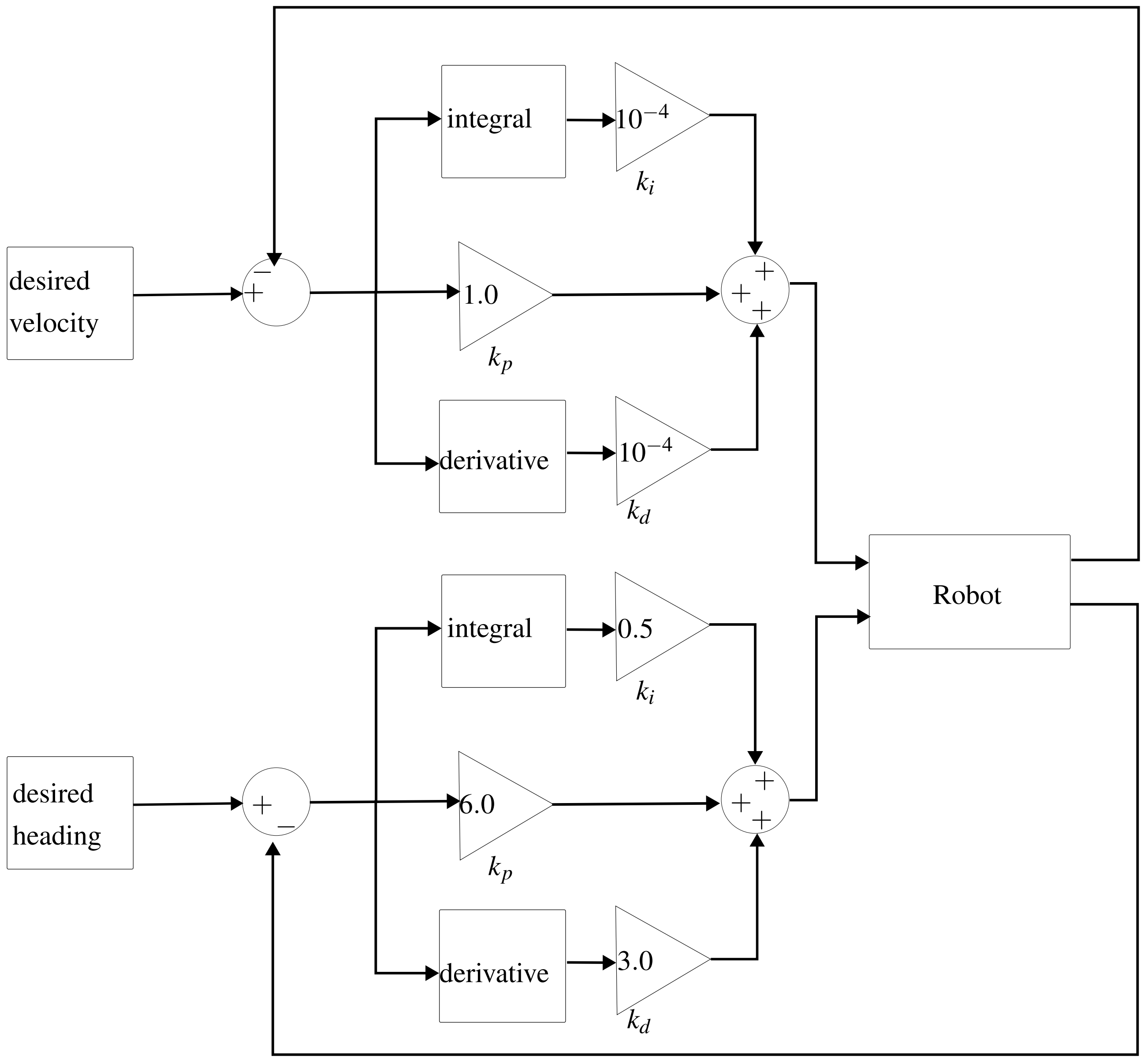


Figure 5.9: Pure pursuit PID controller.

5.2.2 Simulation

The controller is part of a simulation environment which emulates the Turf Tank navigation stack and visualises it, based on the sparse information provided from Turf Tank. The need for developing and implementing an entire simulation environment arose from the issues associated with on-site testing, as no physical specimen was provided. To accommodate this, part of the project was spend on developing the simulation environment to test the solution on a robot like Turf Tank's. Had Turf Tank provided a simulator, more time could have been used on the trajectory generator. The simulation is build as a separate ROS package called ”tt_traversal”. The simulation environment utilises the 3D physics simulator called Gazebo, and a visualiser called RVIZ. These tools together with their respective plugins and associated nodes will be explained throughout this section.

The robot model is developed in a format called URDF (Universal Robot Description Format), which is build upon xml format. This format is used to describe the dimensions, joints and links of the robot. For a differential drive robot it includes the body, the wheels and the rear caster wheels. These are build from basic shapes, like boxes, cylinders and spheres. They are described in relation to each other using joints. Whenever the simulator is launched, a robot state publisher is launched. This robot state publisher publishes all of the transforms and joint states, like the angle of the continuous rotating wheels. In addition the inertial properties, bounding box, and the colours of each body is described. The inertial properties and bounding box are used by Gazebo in the physics engine.

Gazebo uses a differential drive plugin which subscribes to the robot state publisher and takes in the command velocities. These command velocities come from the controller, and are converted into the individual wheel velocities. The wheel velocities are also used to get the odometric position and orientation of the robot, which is published to be used in the controller feedback as well as in the visualiser node. Gazebo initially renders an entire empty 3D environment with only the robot in. As the robot is also visualised in RVIZ and no obstacles are present, then Gazebo can be run headless in case of performance issues.

The colours are used both by Gazebo and RVIZ to visualise the robot. In RVIZ however, the addition of the robot path, the painted lines as well as the grass pane are visualised. This is all handled by the visualiser node which uses the odometric position of the robot. The visualiser node adds paint segments which are small white tiles with close to zero height, to a message array, as well as the robots position to the path, which is then published to RVIZ. The grass is done close to the same way as the paint tiles, but rather just as one large rectangle in green.

The controller essentially binds these components together by interacting with both Gazebo, RVIZ and the visualiser node. As the controller uses a pure pursuit algorithm, it follows points, which is why everything must be discretisied.

5.2.3 Discretisation

Computers in general are not able to work in continuous time, which is a necessity for a transcendental function such as the clothoid. The clothoids therefore have to be to approximated into a new function set. The Turf Tank robot today can operate with two sets of functions, parameterised arcs and lines. A Linear approximation of the clothoid types is shown in 5.10.

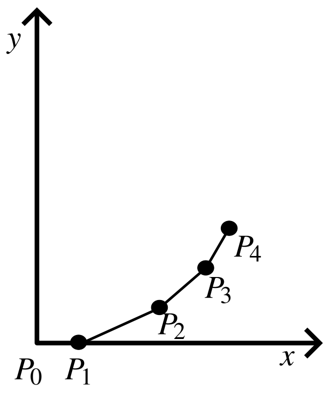


Figure 5.10: Linear approximation of a clothoid.

The linear approximation is the simplest version of the approximation, where a list of evenly spaced points are being calculated via the Fresnel integral. The output of that functions brings out a series of evenly spaced coordinates, which together gives the shape of a clothoid. The approximation is then determined by vectors between two points as shown in equation 5.4.

$$\Pi_j = \sum_{i=0}^{n} p^i \tag{5.4}$$

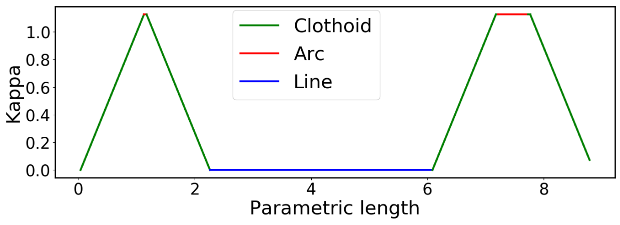
Where n is the number of coordinates p and Π_j is path segment j. The distance between the actual clothoid and the approximated clothoid is decreased, when the number of points increases.

6 Results

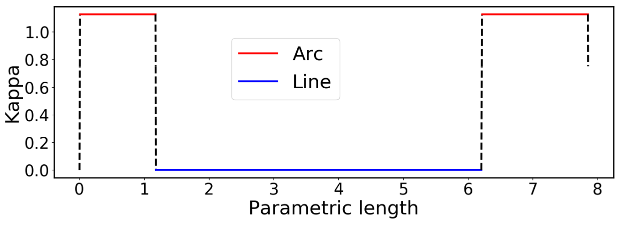
This Chapter will delve into the performance of the new improved path planner for the Turf Tank robot. All testing of the algorithms have been conducted by simulation in Gazebo, with the ROS and the visualisation program RVIZ. In which a simple pure-pursuit velocity PID controlled robot, follows the generated trajectories. Note that this is not the control algorithm running on the real robot, and is not as sophisticated and precise. When testing the proposed CCD algorithm, it will be tested against the original DPP algorithm.

6.1 Continuity Test

The objective of this test is to verify C^1 and G^2 continuity of the path generated. To ensure this, 100 paths generated with CCD and 100 with DPP between arbitrary start and goal poses. These poses are randomly scattered in a 10×10 m area. For each path a curvature plot is generated and is inspected manually to verify G^2 continuity. An example of the curvature plot of a CCD- and DPP path, can be seen in Figure 6.1a, and 6.1b, respectively.



(a) Curvature of a RSR CCD path.



(b) Curvature of the RSR DPP path.

Figure 6.1: Example of curvature plots of the same start and end pose.

Testing for C^1 continuity was done by generating a full trajectory for a soccer field template and check if each segment is connected. Therefore, if the goal pose for segment: i, is equal to start pose for segment $i+1 \pm$ a threshold of 0.001 m, that connection is considered adequate. The results for the continuity tests can be seen in Table 6.1.

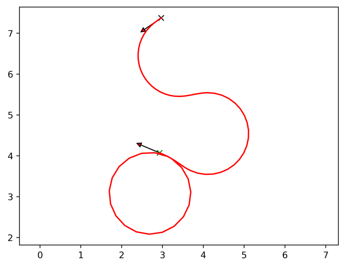
Continuity test		
	CCD	Dubins
C^1	100%	100%
G^2	100%	0%

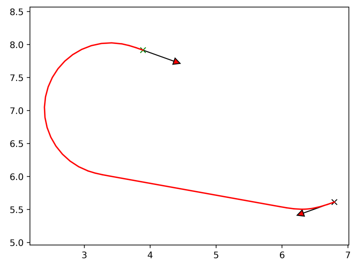
Table 6.1: Results from the continuity tests.

As seen in 6.1, 100% continuity is achieved across both planners while 100% G^2 continuity is only achieved in CCD in transitions.

6.2 Completeness Test

This test checks whether the CCD is able to go from any start- to any goal pose. The test was conducted with 100 different trials, where the path planner had to calculate a path from a start to a goal pose, where checking if the path was generated successfully has been done manually. The poses are uniformly randomly scattered in a 10×10 m area. Examples of the generated paths can be seen in Figure 6.2.


(a) LRL configuration.


(b) RSR configuration.

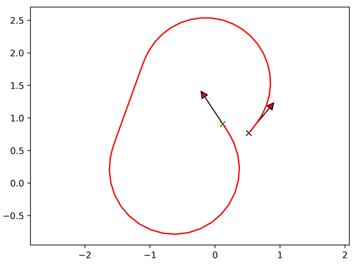

(c) LSL configuration.

Figure 6.2: Examples of the generated paths.

In Figure 6.2, examples of the generated paths can be seen. In Table 6.2, the results after 100 trials can be seen where the CCD planner achieved 100% completeness.

Completeness Test	
Complete	100%
Incomplete	0%

Table 6.2: Results from the completenes test.

6.3 Robot Path Optimality Test

With the CCD, it is necessary to see if it improved the path distance with respect to the current DPP planner. Table 6.3 highlights the measured path length at selected σ based on the plot seen in Figure 7.2. As a result, the path length depends on σ, where a lower σ increases path length and vice versa. Note that for this test, the DPP planner does not include padding, as this better compares the path length of the planners. In addition only 4 out of the 6 Dubins configurations are used in both planners, because RSL and LSR was not developed for CCD.

Optimality Test				
	σ_{max}	Path length m	Difference m	Increase %
DPP	-	1653.55	-	-
CCD	1.0	1915.15	+261.6	$\approx$ 14
CCD	2.1	1781.59	+128.04	$\approx$ 8

Table 6.3: Results from the optimality test.

In Table 6.3 the path length comparison can be seen between the CCD and DPP planner. Depending on σ, an increase in path length can be reduced to as low as 8% relative to the DPP planner.

6.4 Time Optimality Test

The objective of this test is to see if the CCD is faster than the existing DPP. The test will be made by comparing the total time it takes for both trajectory planners to complete a soccer field template, as seen in Figure 6.3.

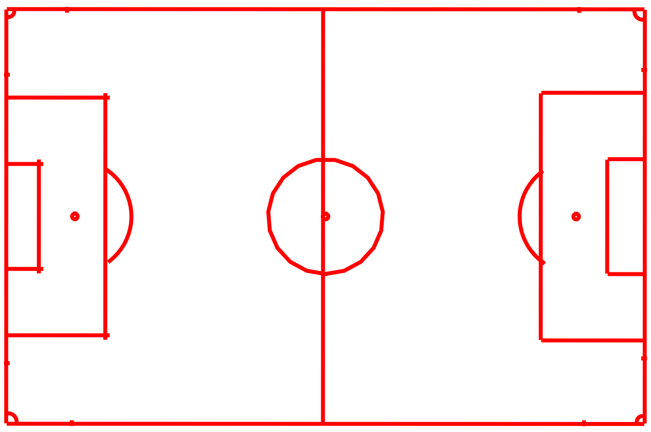


Figure 6.3: Soccer field template with offset transformations.

If the test shows a 5% improvement for the time traversal of the CCD trajectory, the test will be deemed successful. The entire trajectory with painted segments performed by the robot, can be seen in Figure 6.4.

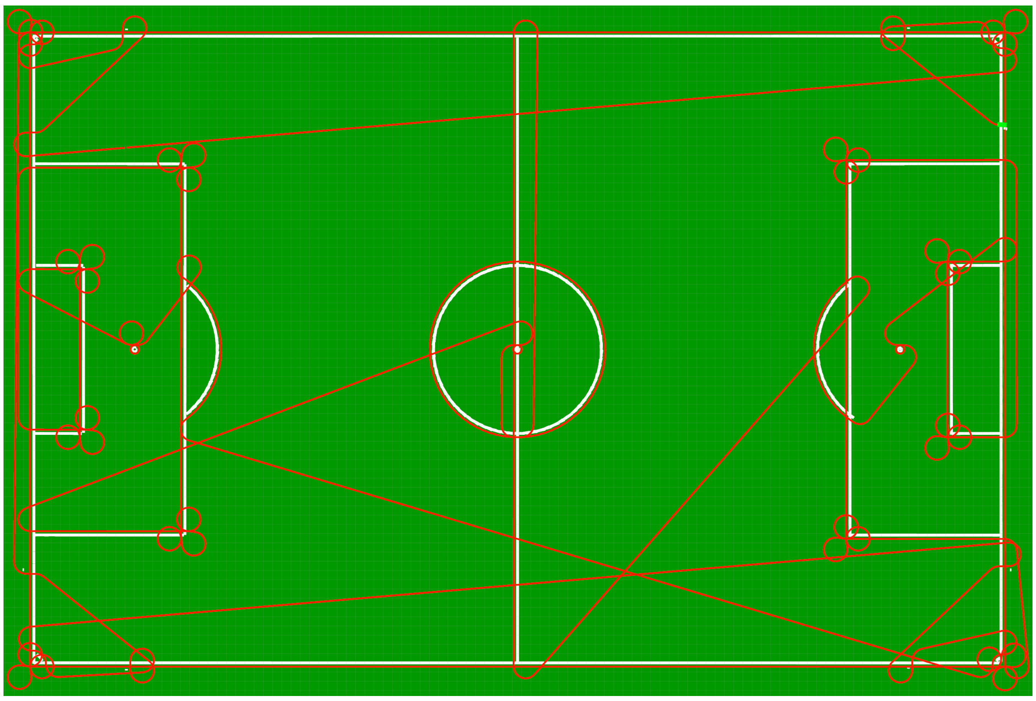


Figure 6.4: Soccer field trajectory where red is the robot path and white is the robot paint.

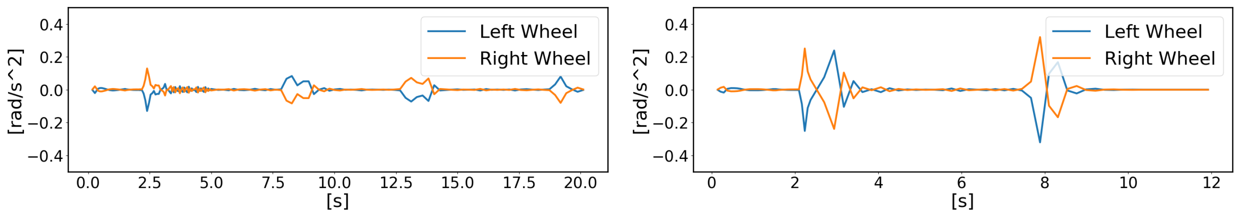
Table 6.4 shows the difference in traversal time of the CCD- and DPP trajectory. The trajectory is generated with $\sigma_{max} = 1.0$. In addition, the DPP planner is executed with the provided padding measurements of $0.5, 0, 1$ pre- and postpadding, while the CCD path does not include padding. This increases the length of the Dubins path, to provide greater knowledge of the tradeoffs with the CCD algorithm relative to Turf Tanks current trajectory generator. All throughout the trajectory, the padding-, transition, and paint velocity are all set to the specified 1.5 $^{m}/_{s}$, as the exact velocities that Turf Tank uses are unknown. Likewise with the path length optimality test, only 4 configurations will be used. In this test however the full pre- and post padding of 0.5 and 0.1 m will be used for the DPP planner. This will however not be used on the CCD planner as the test hypothesis states that the padding can be eliminated with the use of the CCD planner.

Time Optimality Test				
	CCD min:s	DPP min:s	Difference min:s	Increase %
Results	21:37	18:41	2:56	12

Table 6.4: Results from the time optimality test.

6.5 Wheel Acceleration Test

This test is conducted by executing a CCD- and a DPP trajectory, along a single transition on a right corner transition. The main objective is to register the wheel accelerations experienced by each wheel along the trajectory. The trajectory is executed completely while recording wheel acceleration experienced by each wheel. The trajectory is generated with $\sigma_{max} = 1.0$ and the trajectory spans for 3 meters in the entry- and exit direction to ensure a steady velocity. The measured wheel accelerations can be seen in Figure 6.5.



(a) CCD **(b)** DPP

Figure 6.5: Wheel accelerations of a right corner transition.

Looking at Figure 6.5a, it is obvious that there are 4 spikes in acceleration compared to the 2 spikes in Figure 6.5b. This is due to additional wrap around in control 1 and 3 in the path, which happens when the control is self intersecting, because the θ_{lim} constraint is violated. This is also a great contributor to the additional execution time seen along the horizontal axis. During the acceleration test, the maximum magnitude of the acceleration was recorded over a span of 10 runs, and the average is denoted for each planner in Table 6.5.

Max recorded acceleration				
	CCD $\mathrm{rad}/\mathrm{s}^2$	DPP $\mathrm{rad}/\mathrm{s}^2$	Difference $\mathrm{rad}/\mathrm{s}^2$	Improvement %
Results	0.198	0.480	0.282	≈ 59

Table 6.5: Max recorded accelerations and comparison between CCD and DPP planner.

7 Discussion

7.1 Test Discussions

This sections main focal point is to evaluate and discuss the test results in relation to the final solution. During all tests, a normal Dubins planner approximating Turf Tanks current planner, is tested against the continuous curvature Dubins planner.

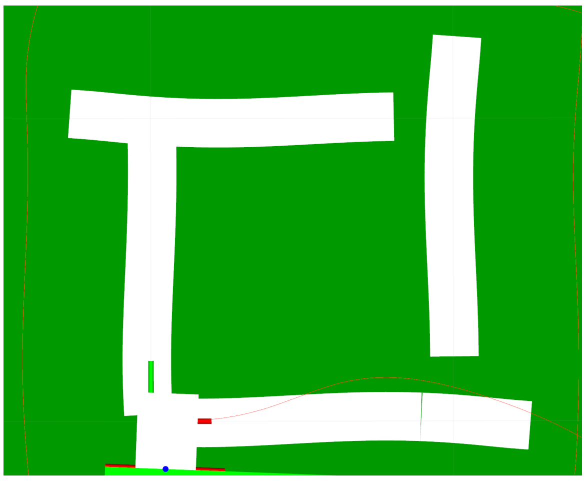
Continuity Test: The tests demonstrated that the CCD planner successfully transformed the transition points between lines, which were initially only C^1 and G^1 continuous, into a C^1 and G^2 continuous state. It is important to note, however, that the CCD planner is specifically designed to ensure continuity in transitions between two positions. This implies that while the robot can maintain continuous curvature κ from a line to padding, it may not achieve such continuity when drawing arcs and circles, such as those present in the centre field of a soccer field.

Note that G^2 continuity is technically lost during transitions between template line to template arc segments. In addition, padding is only implemented as lines, and therefore it is also lost at every arc segment with padding. This leads to a problem with the implementation, that transitions must start and end with $\kappa = 0$. Therefore it can only successfully achieve G^2 continuity when transitioning between line segments. This issue evidently raises questions about the design of the continuity, which only verifies geometric continuity of isolated transitions but not on greater trajectories with arcs.

A crucial disclaimer is warranted regarding the discretization process, which leads to a κ that increases linearly in predefined step sizes rather than following a truly linear function, as the latter is not physically feasible. Nevertheless, with an increased number of points in the discretization that approximates a clothoid, the system approaches true G^2 continuity.

Completeness Test: By doing a Monte Carlo trials by placing 100 random goal and starting poses sampled from a uniform distribution. The CCD planner finds a path between these two points. It is outside the scope of this project to mathematically prove completeness of the CCD algorithm. This is why random start and goal poses have been generated within the approximated sizes and turning constraints applied on the real robot. The test indicate completeness, although it should be noted, earlier iterations of the test had some incomplete situations occurring when the start and goal points are closer than 0.1 m. This problem is not expected to happen in the size of transitions done by the real robot, but it still important to keep in mind.

Acceleration Test: The acceleration test revealed a noticeable decrease in wheel acceleration spikes following the introduction of a linear curvature rate, with a precise reduction of $\approx 59\%$. The simulation clearly illustrated that this decline in wheel acceleration had a significant impact on the transient response of the robot, as depicted in Figure 7.1. This reduction is attributed to the inherent characteristic of the clothoid in the CCD, which dictates that when the robot reaches its goal position, the curvature should be zero and therefore is in the steady state. The robot's transient response is thereby naturally constrained. As a consequence of there being little to none transient response in the CCD planner, the padding can also be limited accordingly.


(a) Simulated DPP drawing a 1×1m square.

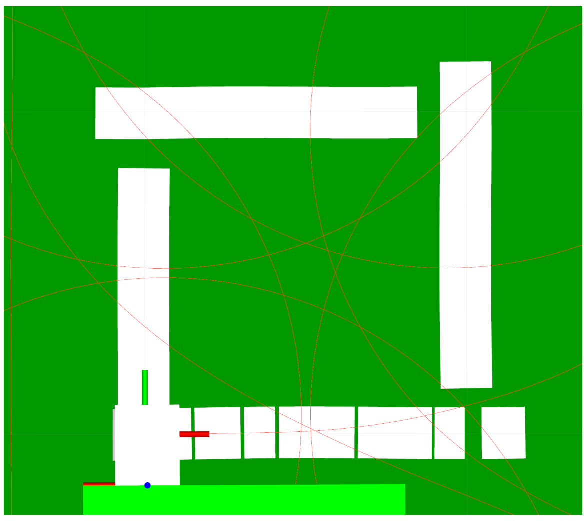

(b) Simulated CCD drawing a 1×1 square, with a sigma value of 0.5.

Figure 7.1: Stress test comparisons between the two planners, by trying to draw a square of 1×1m without padding.

Path Optimality: The optimality test reveals a discernible correlation between σ and the duration as well as the distance covered by the robot in painting a football field. Both the CCD and DPP planners were tested without extra padding during testing. An increase in σ corresponds to reduced time and distance, as the distance converges towards the optimal Dubins path when $\sigma \to \infty$. However, it comes at the cost of an amplified steady-state error, necessitating a larger padding. Hence, these two tests underscore a trade-off between the precision of the robot and the time/distance it must traverse.

Another interesting observation during this test, is how the CCD robot path changes in relation to the σ_{max}. As the curvature rate rises the CCD converges to a normal Dubins path, this relation can be seen on Figure 7.2.

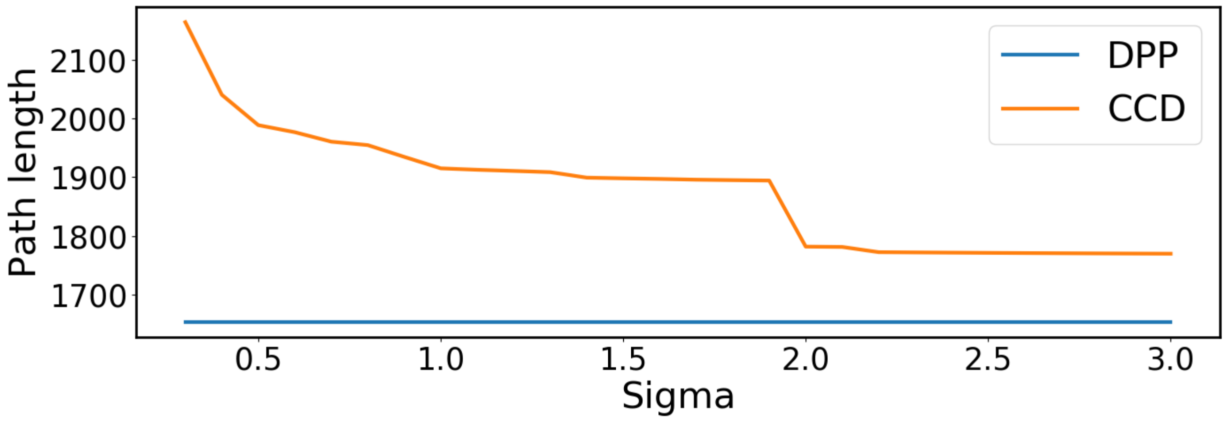


Figure 7.2: The parametric path length m of CCD & DPP in relation to σ_{max}, evaluated $\forall \sigma_{max} \in [0.3, 0.4, \ldots, 2.9, 3.0]$.

The length of the path changes significantly with $\sigma_{max} \to 0$, This is because of clothoids' parametric length greatly increases, to achieve the same curvature as seen in typical clothoid plots. Therefore multiple self intersections occur when σ is in the neighbourhood of 0, and it is believed that for $\sigma = 0.2 \vee 0.1$, the path length would follow the exponential trend, seen in the left side of Figure 7.2. When $\sigma_{max} > 2$ the curvature rate becomes high enough for CSC configurations to avoid two self intersecting controls in each of the corner transitions of the football field. As a result, a negligible effect of increasing σ is observed where $\sigma > 2$, since from this point on most self intersections are eliminated. However, there is still a gap in distance to the DPP path, which is suspected to be due to the γ offsets mentioned in Section 4.3.3.

Total Time Test:

The total time test is conducted to check if the CCD algorithm is quicker for a robot to execute than the original DPP algorithm. Even though the CCD is longer parametrically, the decreased wheel accelerations observed on Figure 7.1, makes it possible to remove the 0.5 m and 0.1 m pre- and post-padding for CCD. In this test the CCD planner unfortunately was slower than the DPP, even with padding removed on the CCD test. This is mainly due to the θ_{lim} constraint explained in Section 4.3.2, where some CCD controls have to do a full circle, a θ_{lim} constrained CCD turn can be seen on Figure 7.3.

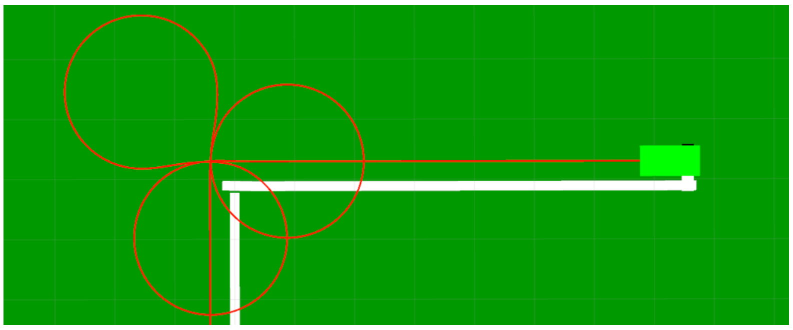


Figure 7.3: A junction where the θ_{lim} constraint is breached.

This problem in CCD seems relatively simple to fix, since it is already known which controls will act like this. Without the extra path length given by the θ_{lim} constrained controls, the CCD planner most likely would pass this test and be faster than the DPP.

7.2 Requirement Validation

Following the discussion, the requirements established in Chapter 3 are validated with respect to the provided solution.

Requirements		
No.	Requirement	State
1	Applicable on any 2D template	🟩 ✓
8	Complete	🟩 ✓
2	Parametric continuity of C^1	🟩 ✓
3	Geometric continuity of G^2	🟨 /
4	Applicable between lines and arcs	🟨 /
5	Output a full trajectory	🟩 ✓
6	Curvature based on dynamics	🟩 ✓
7	Curvature rate based on dynamics	🟨 /
9	No additional hardware	🟩 ✓
10	Applicable to current ROS framework	🟦 ?
11	Total trajectory time minimised	🟥 ✗
12	Total path length minimised	🟨 /
13	Minimized curvature	🟩 ✓
14	Max wheel accelerations reduced	🟩 ✓

🟩 = Fulfilled
🟨 = Partially Fulfilled
🟥 = Not Fulfilled
🟦 = Uncertain

Table 7.1: Indication of fulfilment of established requirements listed in Chapter 3.

The requirements are validated in in Table 7.1, based on the test results of the current solution. Out of the 14 total requirements, only 8 were ”fulfilled”, 4 ”partially fulfilled” and 1 ”not fulfilled” and 1 ”uncertain”. Some of the most important requirements, namely requirement 3 and 4 were partially fulfilled, because of the geometric discontinuities around arcs. If the template only consists of lines, then the requirement is fulfilled. In addition requirement 7 is partially fulfilled because the formula for σ_{max} is provided in equation 4.15, but the assumption mentioned in the same section is too naive. This results in a sigma being too high relative to the test results seen in the path length optimality test, where when σ grows the path length stops decreasing when $\sigma > 2.5$. Any additional increase in σ will only result in an increase in acceleration. Therefore this requirement is partially fulfilled, as σ most likely should be a tuning parameter that Turf Tank can tune to their robot. In addition requirement 12 is deemed partially fulfilled as the path length is depending on the value of σ, thus for high σ values, the requirement is partially fulfilled, but it does not achieve the ideal requirement. The uncertainty around requirement 10 is based on the infeasibility of a physical test on the Turf Tank Two robot. In addition Turf Tank did not provide complete access to their navigation stack, but rather examples of outputs and relevant measurements. Lastly requirement 11 is deemed not fulfilled as the CCD trajectory did not show an improvement in execution time over the DPP planner.

7.3 Shortcomings

The θ_{lim} constraint could easily be solved as the clothoid crossover points are already known. The problem could either be solved by reducing the κ_{max} of the configuration thus make it possible to exit and enter the larger maximum curvature circle with out crossover points. Another solution could be not to utilise Dubins configurations in transitions violating the θ_{lim} constraint, and instead make the entire transition purely out of clothiods, which would most likely decrease the total length of transition.

As explained in Section 7.1, the continuity test does only test G^2 continuity in transitions, as there are not any curved padding at arcs. This could relatively easily be implemented, and the entire path would be G^2 continuous if the clothoids in transition points could end with a specific curvature. If the ability for CCD to start or end with a non-zero curvature, the entire path would be C^1- and G^2-continuous.

With the current implementation of CCD, the LSR and RSL configurations are not implemented. Since the DPP configuration solver is calculated geometrically and is used in the CCD algorithm, the γ constraint for RSL and LSR has to be amended. The DPP without LSR and RSL is still complete but can only be optimal with all 6 configurations available.

Currently Dubins optimality for configurations are assumed optimal for CCD configurations too. The shortest path chosen by the CCD algorithm is currently based on the length of a Dubins path. This means optimality of CCD is not guaranteed and there might be situations with the current planner where the shortest configuration is not chosen.

This problem could be fixed by expressing the parametric length of the clothoid modified Dubins configurations mathematically. And then check for the shortest parametric length continuous curve Dubins path instead.

7.4 Future Work

The current CCD planner operates under the assumption that there are no obstacles present in the robot's workspace. This means that potential obstacles such as goal posts, sprinklers, fences, or dynamic elements like humans and tree branches, are not taken into account. In the presence of these obstacles, the robot may encounter errors as it is unable to perceive them. In future iterations, it is essential to integrate an obstacle avoidance algorithm into the CCD planner. This algorithm should be capable of detecting and responding to both static obstacles and dynamic elements, allowing the robot to navigate around them while still adhering to the continuity requirements.

The current implementation of the CCD utilises the standard Dubins path planner to determine optimal configurations and subsequently modifies the mathematical description accordingly. Simplifying the solution could be achieved by directly deriving the precise formulas for each configuration, encompassing arc sweep angles, line lengths, and clothoid points. This approach not only streamlines the solution but also facilitates the description of RSL and LSR manoeuvres. Furthermore, by calculating configurations without the imposition of the gamma constraint, the overall process becomes more feasible and efficient.

As it is right now, the solution has not undergone real-life testing, raising questions about its practical feasibility. Nevertheless, the established ROS environment, configured to the messaging system the robot utilises, suggests that, with assistance from the Turf Tank team, integration should be seamless, akin to a plug-and-play scenario. This would enable comparing the CCD algorithm with the Dubins planner Turf Tank uses, utilising accurate segment velocities and physical painting of templates.

8 Conclusion

The goal of the project was to assess the influence of continuous curvature on the Turf Tank trajectory planning algorithm. This has been done by presenting a Continuous Curvature path planner, that plans transitions between segments. The solution provides a full path and with an appropriate velocity profile to simulate painting of an entire football field, which uses the same message interface as Turf Tank in order to permit easy integration. The general hypothesis posited that by introducing continuous curvature, the wheel accelerations of the differential drive robot would decrease, resulting in the removal of padding without sacrificing quality of the lines. While the trajectory planner has a few short comings, like missing RSL and LSR configurations, and self intersecting paths. The planner showed to be complete, C^1, and piecewise G^2 continuous during testing. As a result of this, the tests showed a reduction of 59% in wheel accelerations during transitions. This is high enough to conclude that the padding can be greatly reduced if not removed, without sacrificing line marking quality. Furthermore, the tests revealed a longer trajectory compared to Turf Tanks current Dubins planner, which showed to be increased by at least 8% depending on the curvature rate σ, which must be tuned to the dynamics of the physical robot. Revisiting the problem formulation:

How can an offline planner create a C^1 and G^2 continuous trajectory in order to paint an arbitrary template using the Turf Tank Two platform?

It can be concluded that it is partially fulfilled, as G^2 continuity is not achieved along the entire trajectory. The planner does however work for arbitrary templates, as long as they adhere to the same format as Turf Tank uses. By using this Continuous Curvature Planner, Turf Tank can reduce their robots wheel accelerations, to conduct high quality line marking for their customers.

Bibliography

[1] Turf Tank. “Image from the turf tank website”. In: (). Link. Retrieved: 13-09-2023.

[2] Fact.Mr. “Automation in Line Marking Robots Accelerating Demand across Regions Surpassing a Market Value of US$ 273.95 Million by 2033”. In: (2023). Link. Retrieved: 13-09-2023.

[3] *Turf Tank Hits the Field*. eng. 2022.

[4] TurfTank. “Image of Turk Tank Lite”. In: (). Link. Retrieved: 04-10-2023.

[5] Milati. “Image of Turk Tank One”. In: (). Link. Retrieved: 04-10-2023.

[6] Turf Tank. “Turf Tank Two - GPS Line Marking Robot for all Sports”. In: (2023). Link. Retrieved: 22-09-2023.

[7] Anders Daniel Lassen. *Turf Tank Introductory meeting*. 2023. Retrieved: 21-09-2023.

[8] Turf Tank. “BENEFITS OF THE BASE STATION IN LINE MARKING”. In: (2023). Link. Retrieved: 25-09-2023.

[9] Ibrahim A. Hameed. “Path Planning for Line Marking Robots Using 2D Dubins’ Path”. eng. In: *Advances in Intelligent Systems and Computing*. Vol. 533. Cham: Springer International Publishing, 2017, pp. 900–910. ISBN: 3319483072.

[10] Roboert A Adams and Cristopher Essex. *Calulus Volume 2*. Pearson, 2020. ISBN: 9781839616310. Retrieved: 29-09-2023.

[11] Muhammad Ammad, Md Yushalify Misro, and Ahmad Ramli. “A novel generalized trigonometric Bézier curve: Properties, continuity conditions and applications to the curve modeling”. In: *Mathematics and computers in simulation* 194 (2022), pp. 744–763. ISSN: 0378-4754. Retrieved: 03-10-2023.

[12] Raph Levien. *The Euler spiral: a mathematical history*. Tech. rep. UCB/EECS-2008-111. Link. EECS Department, University of California, Berkeley, Sept. 2008.

[13] D.S. Meek and D.J. Walton. “A note on finding clothoids”. In: *Journal of Computational and Applied Mathematics* (2004). DOI, pp. 433–453. Retrieved: 29-09-2023.

[14] Anastasios Lekkas, Andreas Reason Dahl, Morten Breivik, and Thor Fossen. “Continuous-Curvature Path Generation Using Fermat’s Spiral”. In: *Modeling, Identification and Control (MIC)* 34 (Oct. 2013). DOI, pp. 183–198.

[15] Mauro Candeloro, Anastasios M. Lekkas, Asgeir J. Sørensen, and Thor I. Fossen. “Continuous Curvature Path Planning using Voronoi diagrams and Fermat’s spirals”. In: (2013). Link. Retrieved: 06-10-2023.

[16] Jiwung Choi, Renwick E. Curry, and Gabriel Hugh Elkaim. “A note on finding clothoids”. In: *IAENG International Journal of Applied Mathematics* (2010). Link. Retrieved: 29-09-2023.

[17] Kwangjin Yang and Salah Sukkarieh. “3D smooth path planning for a UAV in cluttered natural environments”. In: (2008). Link. Retrieved: 09-10-2023.

[18] Dudek and Jenkin. *Differential Drive Kinematics*. Link. Retrieved: 27-11-2023.

A Github Repository

Throughout the development of the project, the different versions and changes of the code have been tracked in a Github repository. The branch containing the release version can be found in ”main”. The repository can be visited through the following link: Github Repository.

B Turf Tank ROS2 Message Structure

The full ROS2 message tree, disregarding the official ROS2 message called `geometry_msgs/Point` as the documentation of the message structure can be found on the ROS2 wiki. Note that both constants and constraints is highlight in italic.

```
tt_trajectory_msgs/Trajectory trajectory
├── Path path
│   └── Segment[] segments
│       ├── string id
│       ├── string label
│       ├── tt_geometry_msgs/Curve curve
│       │   ├── CurveType type
│       │   │   ├── uint8 NONE = 0
│       │   │   ├── uint8 LINE = 1
│       │   │   ├── uint8 ARC = 2
│       │   │   └── uint8 value
│       │   ├── Line[<=1] line
│       │   │   ├── geometry_msgs/Point start
│       │   │   └── geometry_msgs/Point end
│       │   └── Arc[<=1] arc
│       │       ├── geometry_msgs/Point start
│       │       ├── geometry_msgs/Point center
│       │       └── float64 sweep_angle
│       └── tt_geometry_msgs/LinearDirection direction
│           ├── int8 BACKWARD = -1
│           ├── int8 NONE = 0
│           ├── int8 FORWARD = 1
│           └── int8 value
├── ActuationProfile actuation_profile
│   └── PathActuationParameter[] parameters
│       ├── float64 arc_length
│       └── Actuation actuation
│           ├── uint8 NONE = 0
│           ├── uint8 PUMP = 1
│           ├── uint8 VALVE = 2
│           ├── uint8 SPRAY = 3
│           ├── uint8 DISC = 4
│           ├── uint8 DISC_SPRAY = 7
│           ├── uint8 ALL = 7
│           └── uint8 value
└── VelocityProfile velocity_profile
    └── PathVelocityParameter[] parameters
        ├── float64 arc_length
        └── Velocity velocity
            ├── VelocityType type
            │   ├── uint8 NONE = 0
            │   ├── uint8 LINEAR = 1
            │   ├── uint8 ANGULAR = 2
            │   └── uint8 value
            └── float64 value
```