AN EFFICIENT ROAD REPRESENTATION FOR AUTONOMOUS VEHICLES USING ARC-SPLINES WITH APPLICATION TO TRAJECTORY PLANNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ATAKAN SALIH BOLAT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JULY 2024

Approval of the thesis:

AN EFFICIENT ROAD REPRESENTATION FOR AUTONOMOUS VEHICLES USING ARC-SPLINES WITH APPLICATION TO TRAJECTORY PLANNING

submitted by **ATAKAN SALIH BOLAT** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre ALTUN
Dean, Graduate School of **Natural and Applied Sciences** ______________

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** ______________

Prof. Dr. Klaus Werner Schmidt
Supervisor, **Electrical and Electronics Engineering, METU** ______________

Examining Committee Members:

Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering, METU ______________

Prof. Dr. Klaus Werner Schmidt
Electrical and Electronics Engineering, METU ______________

Assist. Prof. Dr. Ulaş Beldek
Mechatronics Engineering, Çankaya University ______________

Date: 14.06.2024

ABSTRACT

AN EFFICIENT ROAD REPRESENTATION FOR AUTONOMOUS VEHICLES USING ARC-SPLINES WITH APPLICATION TO TRAJECTORY PLANNING

Bolat, Atakan Salih
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Klaus Werner Schmidt

July 2024, 61 pages

Autonomous driving technology has rapidly become essential for the future of transportation, particularly on highways. These roads offer the perfect conditions for autonomous vehicles (AVs) due to steady traffic and minimal interruptions from pedestrians or cyclists. Implementing autonomous driving on highways enhances road safety, improves traffic flow, and reduces environmental impacts. It also positively affects long-distance travel and commercial logistics. As this technology advances, optimizing how AVs perceive and navigate these environments becomes increasingly critical.

The use of real-world data is essential for accurately simulating and analyzing the behavior of autonomous vehicles on highways. In addition, it is important to find a road representation with low memory requirements and a parametrization that is suitable for geometric computations such as trajectory planning. In this study, we make use of the fact that highways are commonly designed following clothoid curves. These curves support a linear change of the road curvature with the curve length and can be conveniently followed by road vehicles at high velocities. Since clothoids curves do not have an analytical representation, our work develops an algorithm for combining arc-splines and straight-line segments in order to tightly approximate clothoid curves with a low data requirement. Specifically, we show using real road data from OpenStreetMap and HERE maps that an analytical road representation using arc-splines and straight road segments can be obtained with an accuracy of $\tilde{3}$ centimeters. As an additional advantage of our method, it is the case that we only need to represent a single lane of a highway since our road model supports parallel-shifts by design. In order to demonstrate the usability of our road representation, we further perform trajectory planning for autonomous vehicles using Beziér curves and arc-splines under typical highway conditions. As a conclusion, this research demonstrates that it is feasible to represent extensive road networks with minimal yet powerful data that includes essential geometric and dynamic properties such as position, heading, and curvature at every point along a trajectory. This efficient representation not only reduces the computational burden but also enhances the precision and flexibility of autonomous vehicle navigation systems.

Keywords: Path planning, Tracking, Autonomous vehicles, Mobile robots, Bezier Curves, Arc splines, Highway

ÖZ

YOL TEMSİLİNDE VERİMLİ BİR YAKLAŞIM: YAY-EĞRİLERİ KULLANARAK OTONOM ARAÇLAR İÇİN YOL PLANLAMA UYGULAMASI

Bolat, Atakan Salih
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Klaus Werner Schmidt

Temmuz 2024 , 61 sayfa

Otonom sürüş teknolojisi, özellikle otoyollarda, ulaşımın geleceği için hızla önemli hale gelmiştir. Bu yollar, sabit trafik ve yayalar ya da bisikletçilerden kaynaklanan kesintilerin azlığı nedeniyle otonom araçlar (AV'ler) için mükemmel koşullar sunar. Otoyollarda otonom sürüşün uygulanması, yol güvenliğini artırır, trafik akışını iyileştirir ve çevresel etkileri azaltır. Ayrıca, uzun mesafeli seyahat ve ticari lojistiği olumlu etkiler. Bu teknoloji ilerledikçe, AV'lerin bu ortamları algılayıp gezme yöntemlerini optimize etmek giderek daha kritik hale gelmektedir.

Otoyollarda otonom araçların davranışını doğru bir şekilde simüle etmek ve analiz etmek için gerçek dünya verilerinin kullanımı esastır. Ayrıca, düşük bellek gereksinimlerine sahip ve yol planlama algoritmaları için uygun bir parametrelendirmeye sahip bir yol temsili bulmak da önemlidir. Bu çalışmada, otoyolların genellikle klotoid eğrileri takip edilerek tasarlandığı gerçeğinden yararlanıyoruz. Bu eğriler, yol eğriliğinin eğri uzunluğuyla birlikte doğrusal bir değişimini destekler ve yol araçları tarafından yüksek hızlarda rahatlıkla takip edilebilir. Klotoid eğrilerinin analitik bir temsili olmadığından, çalışmamız düşük veri gereksinimi ile klotoid eğrilerini sıkı bir şekilde yaklaşmak için yay-eğrisi ve düz hat segmentlerini birleştiren bir algoritma geliştirmektedir. Özellikle, OpenStreetMap ve HERE haritalarından alınan gerçek yol verilerini kullanarak, yay-spline ve düz yol segmentleri kullanarak analitik bir yol temsilinin $\tilde{3}$ santimetre doğrulukla elde edilebileceğini gösteriyoruz. Yöntemimizin ek bir avantajı olarak, yol modelimiz tasarımı gereği paralel kaymaları desteklediği için yalnızca bir otoyol şeridini temsil etmemiz yeterlidir. Yol temsilimizin kullanılabilirliğini göstermek amacıyla, tipik otoyol koşullarında Beziér eğrileri ve yay-eğrisi kullanarak otonom araçlar için yol planlaması yapıyoruz.

Sonuç olarak, bu araştırma, her bir yol boyunca konum, yön ve eğrilik gibi temel geometrik ve dinamik özellikleri içeren minimal ama güçlü verilerle geniş yol ağlarının temsil edilmesinin mümkün olduğunu göstermektedir. Bu verimli temsil, yalnızca hesaplama yükünü azaltmakla kalmaz, aynı zamanda otonom araçların navigasyon sistemlerinin doğruluğunu ve esnekliğini artırır.

Anahtar Kelimeler: Yol planlama, Yol takibi, Otonom araçlar, Mobil robotlar, Bezier eğrileri, Yay eğrileri, Otoyol

To my beloved ones...

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Klaus Werner Schmidt for his patience, support and assistance during this period.

I sincerely thank to my family and my friends for their support during the thesis work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CHAPTER 1

BACKGROUND INFORMATION

1.1 Introduction

The field of autonomous driving has been studied for many years [1] because of its potential to change transportation and make our lives easier. Autonomous vehicles are an important part of this research, offering benefits like improved road safety and efficiency. A key part of developing these vehicles is how we define and model roads. The way roads are represented affects the performance of trajectory planning algorithms, which are important for the safe and effective operation of autonomous cars. This study will look at different road modeling techniques and how various trajectory planning algorithms can be used with these models to help advance autonomous driving technology.

There has been study about road modelling, one of which is [2]. This paper discusses road modelling for autonomous vehicles in urban city environment with OpenStreetMap data. However, it does not implement the algorithm to highway environments. Highway characteristics are much different than city road networks. Moreover, OpenStreetMap data is not as reliable as HERE Maps data as it will be discussed in this research.

Another study that explores road modelling is [3]. This paper focuses on generating lane boundaries from OpenStreetMap data. The paper does not focus on modelling the centerline. For trajectory planning algorithms the reference is the road centerline and it should be modelled. Similar to [2], OpenStreetMap data is not as accurate as HERE Maps data.

Different algorithms were developed for trajectory planning in highway environments such as [4] which is a sampling based local trajectory planner. Generally, sampling based algorithms have more computation time compared to analytical solutions. Bézier curves and arc splines have much less computation time compared to this algorithm.

There are also trajectory planning algorithms utilizing machine learning such as [5]. The first issue that comes with machine learning algorithms is the collection of required training data. After training the learning model, the model has to be deployed in the vehicle. To run the model, the vehicle has to have enough computation power. Analytical solutions we have proposed can run on much lower requirements.

In this thesis, we propose a novel method for road modeling using arc splines. Assuming that highways are designed with clothoid curves, we leverage this characteristic to represent roads with arc splines. Our primary objective is to minimize the number of parameters required to accurately depict a highway. To achieve this, we implement predefined error metrics that balance the trade-off between the number of parameters and the accuracy of the model. Additionally, our approach concatenates multiple road segments into single segments wherever feasible. By utilizing arc spline definitions, our methodology also facilitates the generation of additional lanes from a single lane, significantly reducing the overall number of parameters required for road representation.

In addition to our road modeling method, we propose a trajectory planning algorithm based on arc splines, similar to our road modeling approach. This algorithm employs an analytical approach, ensuring low computational power and reduced computation time. By using arc splines for both the road model and the trajectory, the algorithm can compute a trajectory that achieves zero error at the destination point. For a comprehensive analysis, this study also implements Bézier curves, allowing for a comparison between Bézier curve-based trajectories and those based on arc splines. This comparative analysis aims to highlight the strengths and potential advantages of using arc splines in trajectory planning for autonomous vehicles.

The main contributions of our methodology are as follows:

1. **Arc Spline-Based Road Modeling**: We propose a novel method for modeling highways using arc splines, assuming highways are designed with clothoid curves.

2. **Parameter Minimization**: Our algorithm minimizes the number of parameters required to accurately represent a highway by implementing predefined error metrics that balance parameter count and model accuracy.

3. **Segment Concatenation**: The method concatenates multiple road segments into single segments wherever feasible, further reducing the number of parameters.

4. **Lane Generation**: By utilizing arc spline definitions, the methodology allows for the generation of additional lanes from a single lane, significantly decreasing the overall parameter count.

5. **Trajectory Planning Algorithm**: We introduce a trajectory planning algorithm based on arc splines, which is computationally efficient due to its analytical approach.

6. **Error-Free Destination Point**: The trajectory planning algorithm ensures zero error at the destination point by aligning both the road model and trajectory on arc splines.

7. **Comparative Analysis with Bézier Curves**: The study includes an implementation of Bézier curves for trajectory planning, providing a comparative analysis to highlight the advantages of arc spline-based trajectories.

1.2 Basic Concepts

This thesis focuses on the efficient representation and trajectory planning for autonomous vehicles on highways. First part of the thesis primarily involves representing roads as arc and line segments to capture their geometric properties effectively. By merging these segments where possible, the methodology aims to minimize the data needed while keeping important information like position, heading, and curvature. Another fundamental concept in this thesis is the use of Bézier curves for trajectory planning. These curves are particularly useful in generating smooth paths between specified points under various constraints, making them ideal for simulating realistic vehicle movements on the clothoid modeled roads. These concepts together form the backbone of developing a road representation solution together with an applied vehicle trajectory generation.

1.2.1 Road Modelling

1.2.1.1 Clothoids

In modern highway design, clothoids are widely used. Known as the Euler or transition curve, a clothoid keeps the same curvature rate along its length. This feature makes them very useful for road construction to ensure smooth transitions between straight and curved sections. Therefore clothoids become also important for setting a reliable path for autonomous vehicles on highways.

The mathematics of clothoids are based on Fresnel integrals, specifically the Fresnel Sine and Cosine integrals. These provide the equations for the curve's curvature and length, offering a detailed way to represent them.

The x and y coordinates of the clothoid are determined by the following integrals:

$$x(t) = \int_0^t \cos(\theta(s))\, ds + x_0$$

$$y(t) = \int_0^t \sin(\theta(s))\, ds + y_0$$

The curvature $\kappa(t)$ of a clothoid is defined as:

$$\kappa(t) = k_1 + \frac{(k_2 - k_1)t}{L} \qquad (1.1)$$

where k_1 is the initial curvature, k_2 is the final curvature while t being the arc length along the trajectory. L is the total length of the trajectory.

The integral of clothoid curve's curvature gives the heading change $\theta(t)$ over the trajectory of a clothoid curve. The heading at any point depends on curvature and

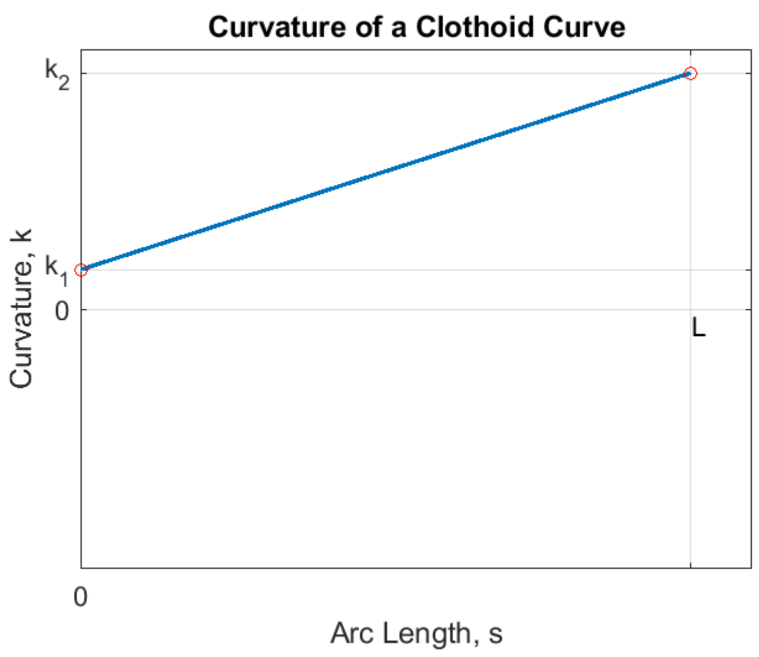

Figure 1.1: Curvature plot of a clothoid curve

path length:

$$\theta(t) = \theta_0 + \int_0^t \kappa(s)\, ds \tag{1.2}$$

where θ_0 is the initial heading and equation 1.1 gives $\kappa(s)$. This integrates to:

$$\theta(t) = \theta_0 + k_1 t + \frac{k_2 t^2}{2L} - \frac{k_1 t^2}{2L}$$

which shows how the heading changes as a function of the distance t along the curve.

Clothoids have a few different parametrizations. In this thesis $\mathcal{C} = [x_i, y_i, \theta_i, \kappa_i, \kappa_f, \mathcal{L}]$ parametrization and notation is used. x, y is position, θ_i is the initial heading of the trajectory, κ_i and κ_f is the initial and final curvature of the trajectory respectively while $\mathcal{L}$ is the length of the trajectory.

Figure 1.2 shows a plot of clothoid trajectory whose initial values are defined as zero.

A clothoid does not have to have increasing curvature along the trajectory. Curvature can also linearly decrease. Figure 1.3 gives an example for this type of clothoids.

Even though clothoids in figure 1.2 and figure 1.3 have similar parameter values the

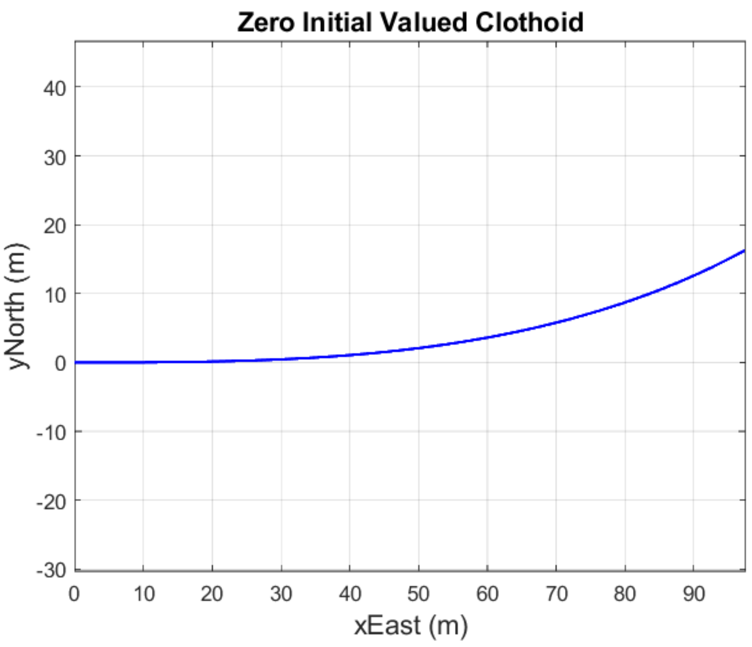

Figure 1.2: Plot of $\mathcal{C} = [0, 0, 0, 0, 0.01, 100]$ clothoid

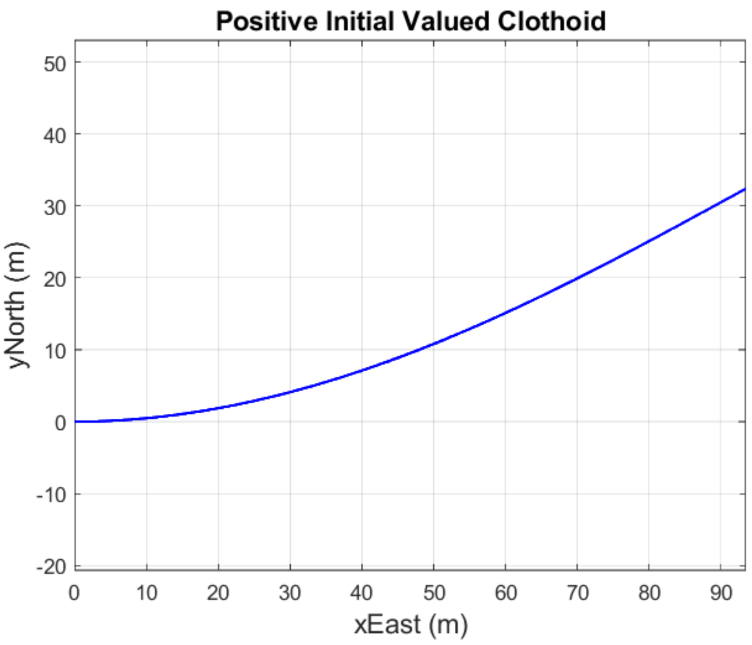

Figure 1.3: Plot of $\mathcal{C} = [0, 0, 0, 0.01, 0, 100]$ clothoid

trajectories are very different. This difference may be observed in figure 1.4.

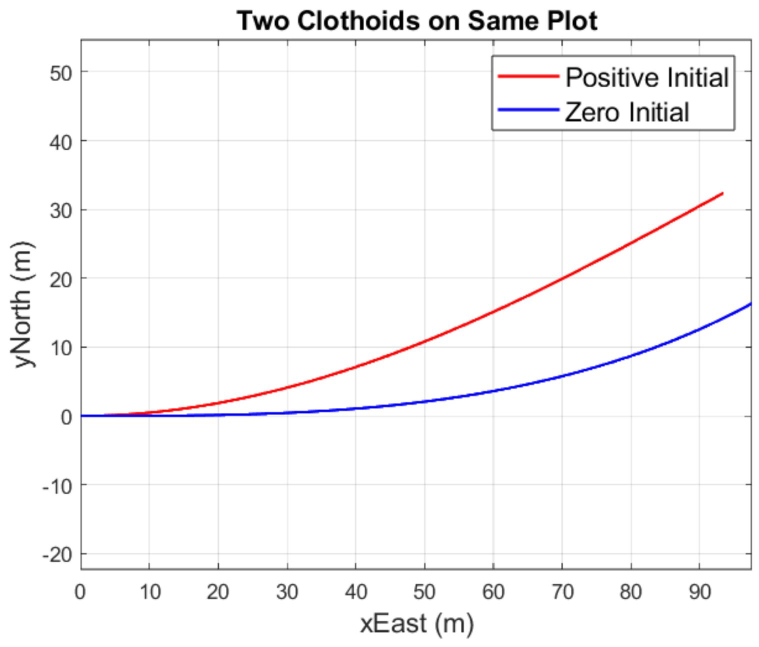

Figure 1.4: Plot of both $\mathcal{C} = [0, 0, 0, 0.01, 0, 100]$ and $\mathcal{C} = [0, 0, 0, 0, 0.01, 100]$

Clothoids are particularly useful in road construction due to their property of having a curvature that changes linearly with arc length, facilitating smooth transitions between straight sections and curves. The ability of clothoids to seamlessly connect different roadway segments minimizes abrupt changes in steering requirements, which is important for high-speed travel.

Despite these advantages for road design, the computational aspect of using clothoids has challenges. Clothoids do not have an analytical solution for their arc length, which means that calculating their coordinates from given parameters requires numerical integration. This process can be computationally intensive and less efficient for real-time applications, such as autonomous vehicle navigation systems that demand rapid calculations. The necessity for integration makes the implementation of clothoid-based paths more complex compared to simpler curve forms that offer analytical solutions.

1.2.1.2 Clothoid Approximation with Arc Splines

We use a method to obtain an analytical representation of clothoid segments. Our main motivation for this approximation is to simplify and enhance the usability of these curves in practical applications. Representing a segment with arc splines offers several advantages: it facilitates the creation of additional lanes from a single lane by simply adjusting the turning radius of the arcs by the width of a lane. This method also allows for straightforward computation of coordinates along the trajectory, eliminating the need for complex integration processes.

In the approximation of a clothoid curve $\mathcal{C} = [x_i, y_i, \theta_i, \kappa_i, \kappa_f, \mathcal{L}]$, we assume that the initial curvature k_i and the final curvature k_f share the same sign. The approximation involves creating an arc spline of order n, which consists of $n + 1$ arc segments. The key parameters defining an arc spline are as follows:

- **Curvature increment** h: Defined as $\frac{k_f - k_s}{n}$.
- **Curvature of arc** j, where $j = 0, \ldots, n$: Given by $k_j = k_s + jh = k_s + j\frac{k_f - k_s}{n}$.
- **Length of the first and last arcs** $j = 0, n$: $S_0 = S_n = \frac{S}{2n}$.
- **Length of the remaining arcs** $j = 1, \ldots, n - 1$: $S_j = \frac{S}{n}$.

Using these parameters, the arc spline representing the clothoid curve is parameterized accordingly.

For further observation on curvatures 1.5 may be observed. Over the length of the original clothoid a series of arcs are generated with curvature values at certain points. These points are related to the order of the approximation. In order to maintain second order derivative continuity consecutive arcs start with the heading of preceding arc's heading.

Figure 1.6 shows an arc spline with order 5. When the order is 5 there are 6 arcs to approximate the curvature. Each arc's center may also be observed. This clothoid's curvature increase therefore the turning radius decreases towards the end of the spline.

The order of the approximation depends on the error between approximation and

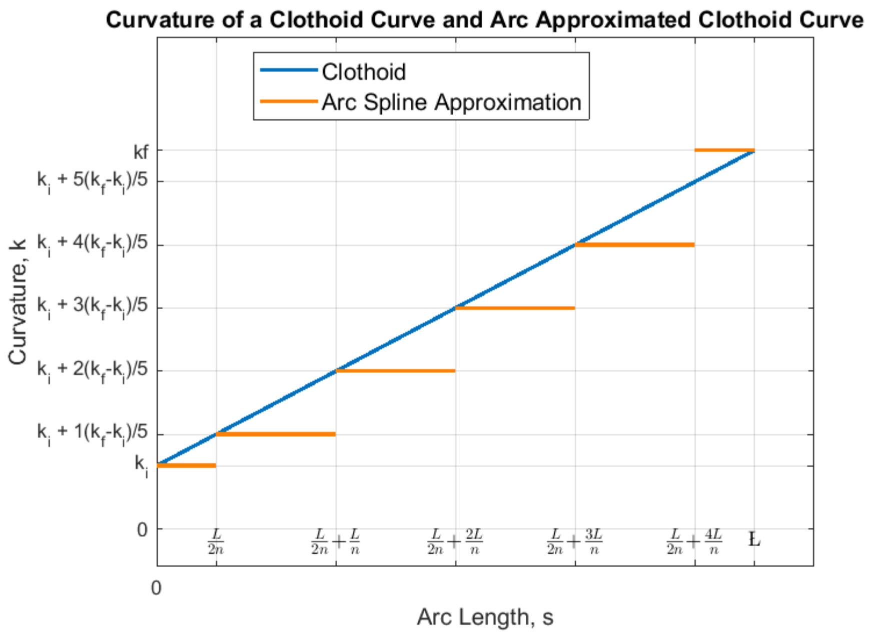

Figure 1.5: Curvature plot of a clothoid curve and curvatures of arcs

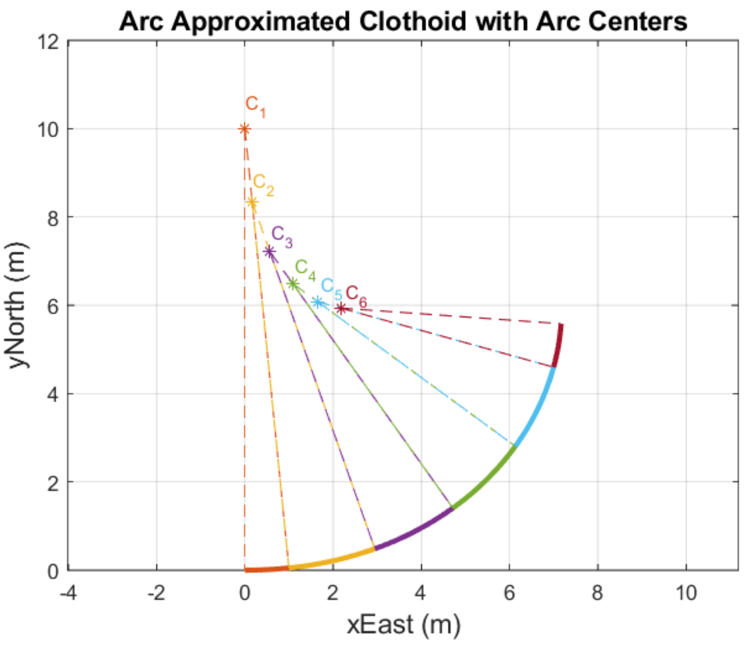

Figure 1.6: Approximation of $\mathcal{C} = [0, 0, 0, 0.1, 0.2, 10]$ with arc-spline

ground truth. If the error is higher than desired level the order is increased and if the error is below than the desired level the order is decreased. In order to be memory efficient, order is decreased as much as possible. The error metrics are RMS Euclidian distance and maximum Euclidian distance between approximation and ground truth over a whole road segment. The minimum distance between each point of approximation and ground truth are computed, then RMS and maximum is computed. Based on the accuracy objective, the order of approximation is either increased, decreased, or kept the same.The figure 1.7 shows an example of error plot. The error oscillates since the road is clothoid and makes a smooth turn whereas the approximation is a series of arcs.

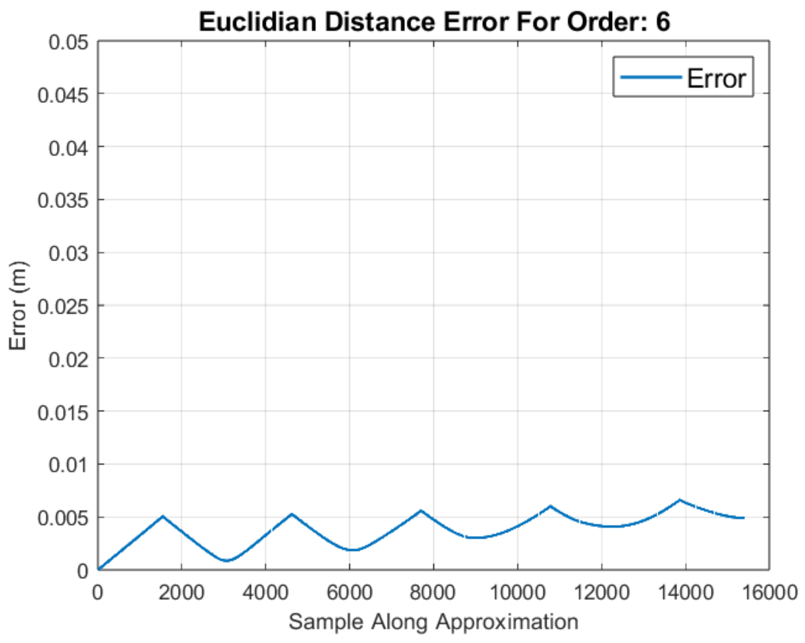

Figure 1.7: Error for a 154 meter long road segment with an order 5 approximation

Figure 1.8 shows a ground truth and approximated curve. The figure is zoomed in to better observe the error.

1.2.2 Waypoint Data

The input to our algorithm is a series of waypoints, where each waypoint includes position, heading, and curvature information. The position gives the exact location

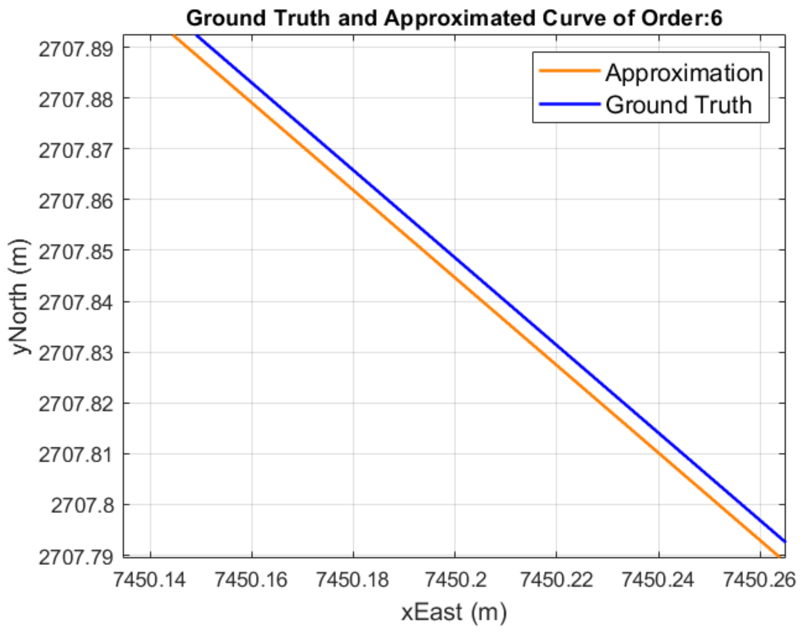

Figure 1.8: A closeup view of an approximation and real road data.

of the waypoint on the map, the heading tells us the direction the vehicle is facing at that point, and the curvature shows how sharply the road is turning at that location. While these three pieces of information provide some geometry information of the road, they are not enough to represent the entire road. To address this, we use the assumption that every road segment is basically a clothoid. With this assumption, we can approximate segments with an arc spline as discussed in section **??**, allowing for a practical representation of the road. Moreover, to generate the intermediate ground truth road between two waypoints a G1 clothoid fitting algorithm [6] is used.

1.2.2.1 OpenStreetMap Database

The OpenStreetMap database consists of latitude and longitude pairs along the requested highway. For each direction in a road a separate latitude and longitude pair exist in the data. Lane information does not exist in this dataset.

Figure 1.9 shows an example OpenStreetMap data. Waypoints are connected to each other with lines.

For some part of the road the waypoints are at the center of the lane however for some

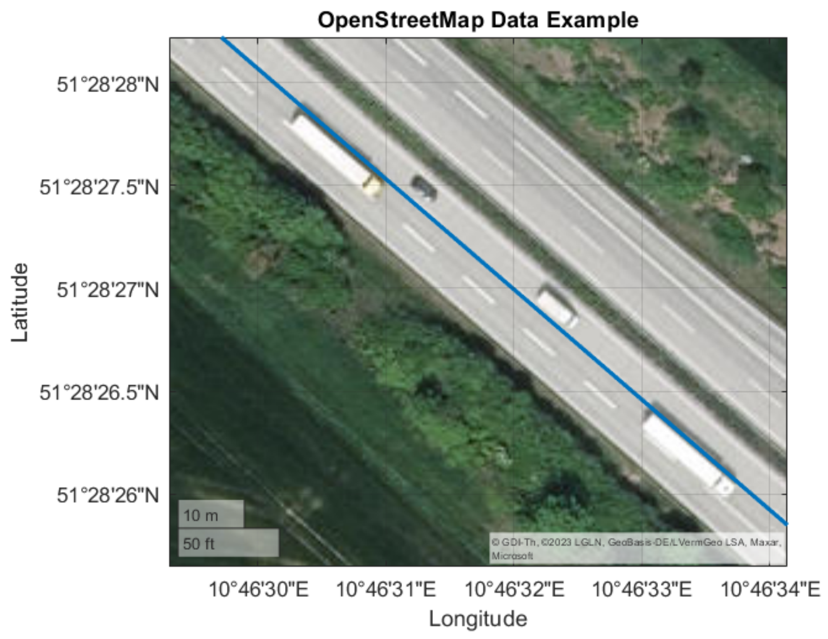

Figure 1.9: OpenStreetMap data example of Autobahn 38 in Germany

part of the road, the waypoints are located at lane borders. This type of data would create issues while modeling the road.

1.2.2.2 HERE Maps Database

The HERE Maps database consists of latitude and longitude pairs along the requested highway. For each lane a separate latitude and longitude pair exist in the data. This property allows for the computation of error in the parallel shifting procedure, which will be discussed in the upcoming sections. Figure 1.10 displays an example road segment where each color represents a different lane. Waypoints are connected to each other with a simple line.

The HERE Maps database is notably more accurate than the OpenStreetMap database. It offers a higher frequency of waypoints and provides position information for each individual lane, unlike OpenStreetMap. These features make HERE Maps more useful for this research.

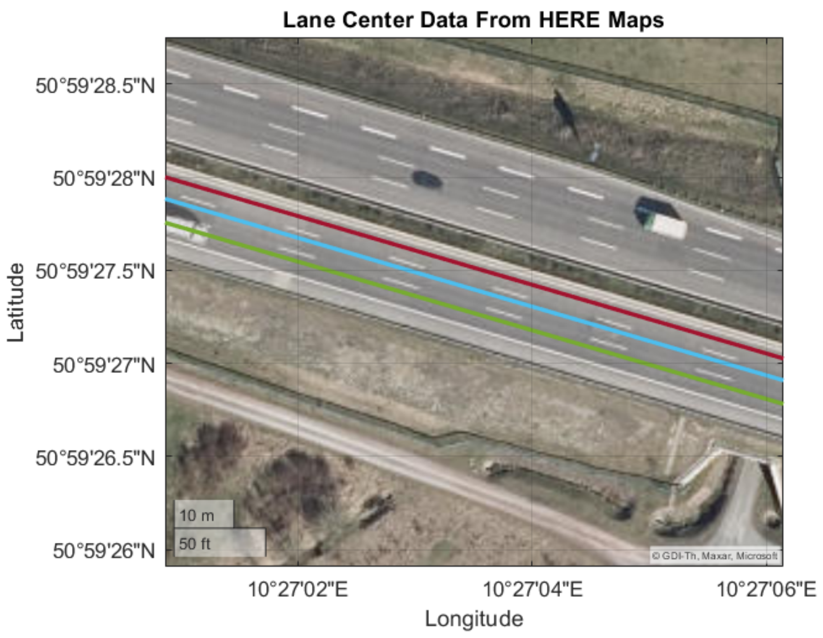

Figure 1.10: HERE Map example of Autobahn 4 in Germany

1.2.3 Bézier Curve Based Trajectory Planning

Bézier curves are handy tools used in computer graphics and vehicle design. They work by defining a curve with a few control points. These parametric curves are great for creating smooth shapes and paths in graphics and for designing smooth car bodies [7]. They give designers precise control over the shape of the curve, making them really useful in various applications where smooth, controlled curves are needed.

Given the detailed comparison between lower-order curves and quintic Bézier curves in [8], the choice is made to utilize quintic Bézier curves for generating the final path. One of the primary advantages of fifth-order Bézier curves over cubic ones is the enhanced control they offer in shaping the curve, as well as the ability to impose curvature at both ends of the curve.

Equation 1.3 is the definition of quintic Beziér curves.

$$B(t) = \sum_{i=0}^{5} \binom{5}{i} (1-t)^{5-i} t^i P_i \tag{1.3}$$

Quintic Bézier curve requires 6 control points by definition. In order to define control points of a Bézier curve, we utilize the initial and final pose of the vehicle. P_0 and P_5 are directly the position of the vehicle and the destination position respectively. P_1 and P_4 are related to heading of the vehicle and the heading value at the destination respectively. Equation 1.4 is the derivative of Bézier curve. Similarly, P_2 and P_3 are related to curvature of the vehicle and the heading value at the destination respectively. Equation 1.5 is the second derivative of Bézier curve.

$$B'(t) = \sum_{i=0}^{4} 5\binom{4}{i}(1-t)^{4-i}t^i(P_{i+1} - P_i) \tag{1.4}$$

$$B''(t) = \sum_{i=0}^{3} 20\binom{3}{i}(1-t)^{3-i}t^i(P_{i+2} - 2P_{i+1} + P_i) \tag{1.5}$$

It is possible to impose initial heading and curvature values to a Bézier curve by substituting $t = 0$ to equations 1.4 and 1.5 respectively. Similarly, it is possible to impose destination heading and curvature values to a Bézier curve by substituting $t = 1$ to equations 1.4 and 1.5 respectively. The methodology for computing the Bézier curves will be discussed in the upcoming sections.

In this research, Bézier curves are defined by:

$$B = [(x_i, y_i), (x_f, y_f), \theta_i, \theta_f, \kappa_i, \kappa_f] \tag{1.6}$$

where (x_i, y_i) and (x_f, y_f) denote the initial and final positions, θ_i and θ_f represent the initial and final heading values, and κ_i and κ_f indicate the initial and final curvature values of the Bézier curve.

1.2.4 Arc Spline Based Trajectory Planning

In section 1.2.1.2 we have mentioned that roads can be modelled with arc splines. It is also be possible to generate a arc spline based trajectory to follow clothoid base roads. Arc spline based trajectories have a very similar definition to Bézier curves since the problem at hand is very similar.

The input to arc spline trajectory planning algorithm is the ego vehicle's attitude and the road definition. The output is a set of waypoints which may be connected with

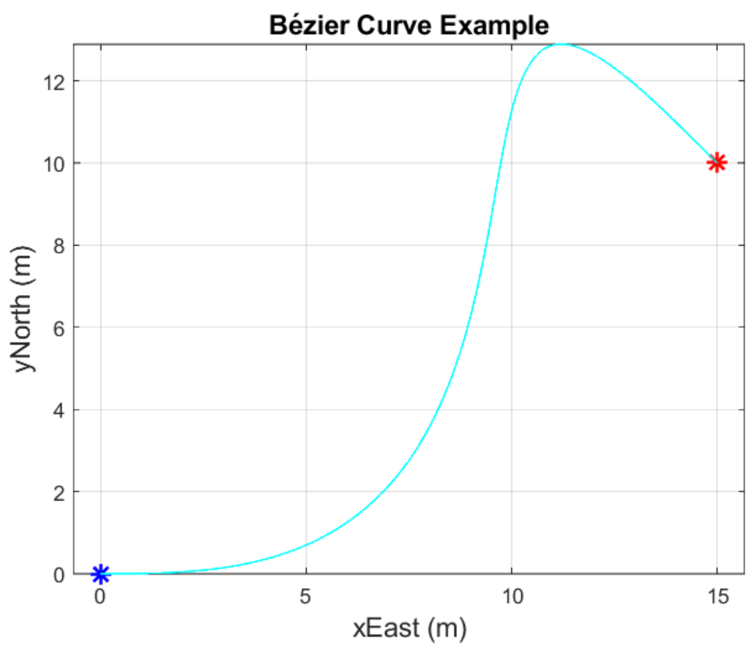

Figure 1.11: Bézier curve example for $B = [(0,0),(15,10),0,45,0,0.01]$

arc splines. The order of arc splines may be determined based on the length between each waypoint.

1.3 Contributions and Novelties

Our main contributions are as follows:

Arc Spline-Based Road Modeling: We propose a novel method for modeling highways using arc splines, assuming highways are designed with clothoid curves.

Parameter Minimization: Our algorithm minimizes the number of parameters required to accurately represent a highway by implementing predefined error metrics that balance parameter count and model accuracy.

Segment Concatenation: The method concatenates multiple road segments into single segments wherever feasible, further reducing the number of parameters.

Lane Generation: By utilizing arc spline definitions, the methodology allows for the generation of additional lanes from a single lane, significantly decreasing the overall parameter count.

Trajectory Planning Algorithm: We introduce a trajectory planning algorithm based on arc splines, which is computationally efficient due to its analytical approach.

Error-Free Destination Point: The trajectory planning algorithm ensures zero error at the destination point by aligning both the road model and trajectory on arc splines.

Comparative Analysis with Bézier Curves: The study includes an implementation of Bézier curves for trajectory planning, providing a comparative analysis to highlight the advantages of arc spline-based trajectories.

1.4 The Outline of the Thesis

The remainder of the thesis is as follows.In **??**, motivation is given. Then, basis studies of the proposed method will be examined from previous research. In chapter 2 The Road Representation method which is the the algorithm that generates the road model will be described. In chapter 3 Bézier trajectories and arc spline trajectories will be described. The mathematical background of arc splines trajectories will be detailed. Experimental results are shown in both chapter 2 and chapter 3 for both road modeling and trajectory generation respectively.

CHAPTER 2

ROAD SEGMENT REPRESENTATION METHODOLOGY

In trajectory planning for autonomous vehicles on highways, having a clear and accurate road representation is very important. Roads are usually represented with waypoints, each containing information about its position, heading, and curvature. However, these waypoints are often spread out and do not provide a detailed picture of the road between them. To achieve an accurate road representation, waypoints must be stored closely together, which requires a significant amount of memory. Additionally, waypoints need to be stored for each lane separately which would geometrically increase the amount of memory needed. A road segment is defined as the road between two waypoints.

To improve this, we suggest using arc spline approximation. This method helps to create a smoother and more continuous road representation. By fitting arc splines to the waypoints, we can accurately estimate the intermediate points between them. This approach reduces the amount of data needed to represent the road and provides an analytical representation with fewer parameters to store. Additionally, it facilitates working with curvature and tangent information along the road.

In the following sections, we will explain how we use a clothoid fitting algorithm [6] to get the ground truth for the road centerline, and then apply arc spline approximation to represent the road segments. This method aims to provide an accurate and efficient road representation, which will improve the performance of trajectory planning for autonomous vehicles. Algorithm 1 summarizes the general flow of the methodology.

Algorithm 1 Road Segment Approximation Methodology

```
1: Input: $leftUpperLatLon$, $rightLowerLatLon$, $errorCfg$, $laneWidth$, $numberOfLanes$
2: $waypoints$ = downloadDataFromHereMaps ($leftUpperLatLon$, $rightLowerLatLon$)
3: $clothoids$ = G1Fitting($waypoints$)
4: $approximatedSegments$ = approximateSegments ($clothoids$,$errorCfg$)
5: $mergedSegments$ = mergeSegments ($approximatedSegments$,$errorCfg$)
6: $additionalLanes$ = generateAdditionalLanes ($approximatedSegments$, $laneWidth$, $numberOfLanes$)
```

2.1 Ground Truth Generation

In order to generate the ground truth a fast G1 clothoid fitting algorithm is used [6]. Initially, HERE maps database provides only the positions of waypoints, not the heading and curvature. After clothoid fitting, every waypoint earns heading and curvature property.

Figure 2.1 shows a small example of fitting procedure. 5 waypoints are provided as input and each color on the lane center represents a different road segment.

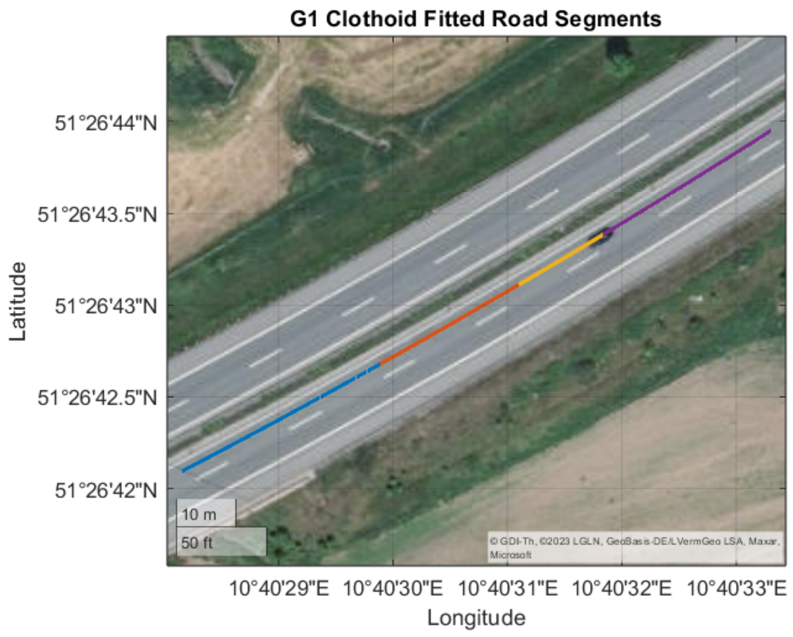

Figure 2.1: G1 clothoid fitting applied on 5 waypoints

2.2 Approximation

The input to the approximation algorithm is a series of road segment clothoids. The output of the approximation is a series of road segments where each segment is either a line segment or an arc spline segment. Therefore, the type of each road segment must be decided to make the approximation more efficient. In this context, efficiency refers to both the number of parameters required to represent the road and the accuracy of the representation. Line segments are more efficient than arc spline segments as they require fewer parameters to store and are simpler to compute. Consequently, line segments will be used whenever possible to minimize data storage and computational effort. As mentioned in **??**, it is possible to approximate clothoids with a series of arcs, balancing the trade-off between simplicity and accuracy.

To determine whether a road segment can be approximated by a line segment, we first assess the heading change between the initial and destination points. If the heading difference between the two waypoints is sufficiently small, a straight line connecting the initial point to the final point is fitted. Subsequently, the Euclidean distance error between the approximation and the ground truth is calculated.

If the maximum error exceeds the predefined allowable error or if the Root Mean Square (RMS) error exceeds the maximum allowable RMS error, the segment cannot be accurately represented by a straight line. In such cases, the segment must be fitted with an arc spline instead. Conversely, if the error remains within acceptable limits, the segment is considered successfully fitted with a straight line.

To approximate a road segment with an arc spline, predefined error metrics must be established. Similar to line segment approximation, the error metrics for arc spline approximation include the Root Mean Square (RMS) error and the maximum error. The arc spline approximation process is iterative, beginning with an initial order of 5 for the spline.

First, the segment is approximated using this initial order, and the Euclidean distance error is computed. If the error falls within the acceptable range, the order is reduced to decrease the number of parameters, thereby simplifying the representation. Conversely, if the error exceeds the acceptable range, the order is increased to achieve a more accurate representation of the road segment. This iterative approach ensures that each road segment is represented with optimal accuracy while minimizing the complexity of the spline.

Figure 2.2 shows the error along a trajectory for low and high order approximations. It is clear that higher order approximations are much accurate. However, the memory needed increases linearly with the order level.

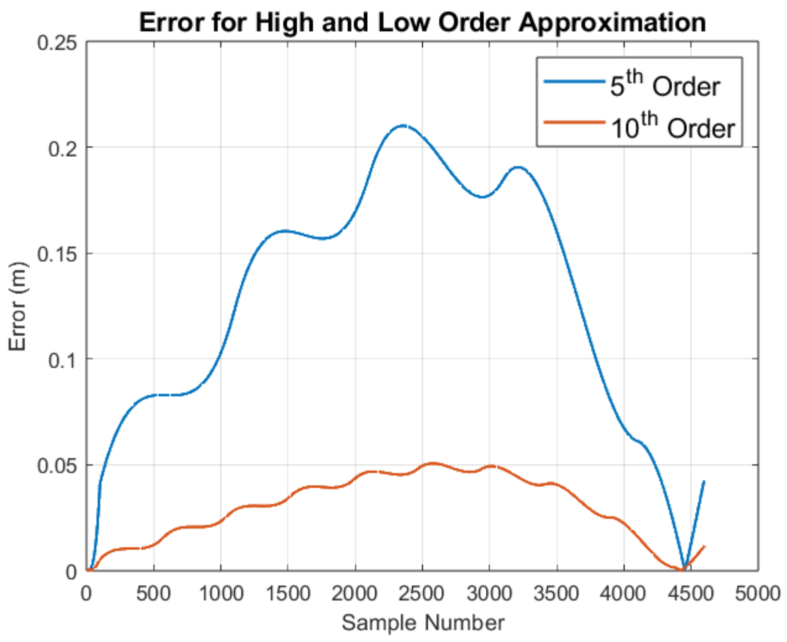

Figure 2.2: Error comparison of low and high order approximations for the clothoid $\mathcal{C} = [0, 0, 0, 0, 0.1, 50]$

Table 2.1 is given for better understanding of predefined error metrics. Endpoint degree deviation is the deviation from ground truth at the end waypoint of the segment. Arc splines do not have an endpoint degree deviation error by their nature.

	Endpoint Degree Deviation	RMS Error (m)	Maximum Error (m)
Line Segment	0.2°	0.10	0.15
Arc Spline	-	0.10	0.15

Table 2.1: Error Table

These processes ensures that each road segment is represented with minimal error, maintaining the accuracy and increasing the efficiency.

Figure 2.3 shows the error of an example approximation of a real road. The RMS and maximum error is computed for every 10 meter long subsegment to better observe the evolve of error values.

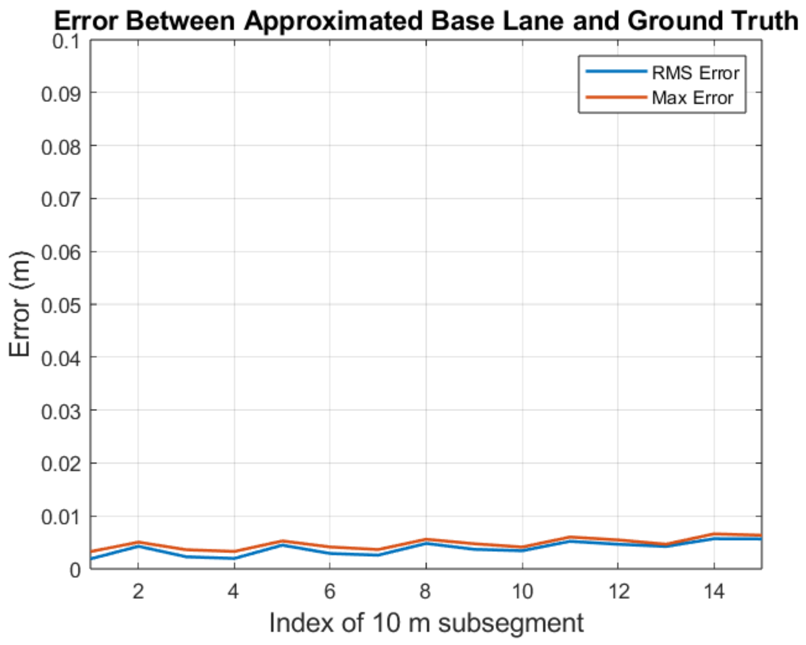

Figure 2.3: Euclidian error between approximated reference lane and ground truth

Pseuodocode 2 gives the algorithm for road segment approximation algorithm.

2.3 Combination

A method to further decrease the number of parameters in road representation is the combination process, which involves merging consecutive road segments. In theory, if a road is entirely straight and has multiple waypoints, it should be possible to represent the entire road with a single line segment. Similarly a group of arc splines may be represented by a single arc spline.

If consecutive arc spline segments have similar curvature rates, it may be possible to concatenate a group of arc splines to a single arc spline. It is known that arc splines have either increasing or decreasing curvature. Therefore if we want to represent

Algorithm 2 Road Segment Approximation

```
1:  for each road segment $r$ do
2:      $\theta \leftarrow$ headingChange($r$)
3:      if $\theta <$ lineDegreeDeviation then
4:          $error \leftarrow$ fitLine($r$)
5:          if $error <$ predefinedMetrics then
6:              save(lineApproximation)
7:              continue
8:          end if
9:      else
10:         $order \leftarrow 5$
11:         $error \leftarrow$ fitArcSpline($r, order$)
12:         $fitFlag \leftarrow False$
13:         while $error <$ predefinedMetrics do
14:             $order \leftarrow order - 1$
15:             $error \leftarrow$ fitArcSpline($r, order$)
16:             if $error >$ predefinedMetrics then
17:                 save($order + 1$)
18:                 $fitFlag \leftarrow True$
19:                 break
20:             end if
21:         end while
22:         while $error >$ predefinedMetrics do
23:             $order \leftarrow order + 1$
24:             $error \leftarrow$ fitArcSpline($r, order$)
25:             if $error <$ predefinedMetrics then
26:                 save($order$)
27:                 break
28:             end if
29:         end while
30:     end if
31: end for
```

a few road segments with a single arc spline, the road segments must have either linearly increasing or decreasing curvatures.

Similarly for consecutive line segments, if the initial and final heading values of initial and final waypoints are close to each other, it may be possible to merge these line segments into a single line segment.

However, the decision for the combination process cannot rely solely on these criteria. The Euclidean distance error metric must also be considered. This metric ensures that the combined representation maintains an acceptable level of accuracy, balancing the reduction in parameters with the need for an accurate road representation.

Algorithms 3 and 4 describe the combination algorithms for arc splines and line segments respectively. The algorithms are applied according the type of the segment at hand.

2.4 Generation of Additional Lanes from a Single Lane

It is possible to generate adjacent lanes if the road width and lane information are known for a particular road segment. Lane width is usually standardized in every country. Lane information refers to the position of the base lane, such as whether it is the leftmost lane, the rightmost lane, or another lane in between. With this information, it should be possible to derive the representation of the other lanes. This process would further improve the efficiency of the algorithm, making it possible to represent all lanes using only a single lane's information.

The procedure differs for arc spline segments and line segments. For arc spline segments it is possible to achieve other lanes by changing the turning radius of each arc. From 1.2.1.2 section it is known that an arc spline consists of several consecutive arcs.

Figure 2.4 shows a parallel shifting process where the base arc has a turning radius of 10 meters. The base arc's turning radius is increased by the lane width which is 3.7 meters. In the end, shifted arc has a turning radius of 13.7 meters. In this example it is convenient to think the "Original Lane" as the leftmost lane and we are trying to

Algorithm 3 Arc Spline Combination Algorithm

```
1: Input: $approximatedSegments$, $groundTruth$, $errorCfg$
2: Output: $mergedArcSplineSegments$
3: $max\_concat \leftarrow 5$
4: $i \leftarrow 1$
5: for each arc spline $s$ in $approximatedSegments$ do
6:     $kappa \leftarrow s$.curvatureDerivative
7:     for $j \leftarrow 1$ to $max\_concat$ do
8:         if segment $i + j$ is arcSpline then
9:             $ratio \leftarrow kappa/$next segment's curvature
10:            if $ratio < errorCfg.curvatureDerivativeTolerance$ then
11:                Append index to $arcSplineConcatenationList$
12:            else
13:                break
14:            end if
15:        else
16:            break
17:        end if
18:    end for
19:    $i \leftarrow i + j + 1$
20: end for
21: for each concatenation group $g$ in $arcSplineConcatenationList$ do
22:     $temporarySpline \leftarrow fitArcSpline(g)$
23:     $error \leftarrow computeError(temporarySpline, groundTruth(g))$
24:     if $error < errorCfg.concatenationError$ then
25:         $mergedArcSplineSegments \leftarrow g$
26:     end if
27: end for
```

Algorithm 4 Line Segment Combination Algorithm

```
Input: $approximatedSegments$, $groundTruth$, $errorCfg$
Output: $mergedLineSegments$
$max\_concat \leftarrow 5$
$i \leftarrow 1$
for each line segment $l$ in $approximatedSegments$ do
    $theta \leftarrow l.startHeading$
    for $j \leftarrow 1$ to $max\_concat$ do
        if segment $i + j$ is line segment then
            $headingError \leftarrow$ next segment's final heading
            if $headingError < errorCfg.headingDeviationTolerance$ then
                Append index to $lineSegmentConcatenationList$
            else
                break
            end if
        else
            break
        end if
    end for
    $i \leftarrow i + j + 1$
end for
for each concatenation group $g$ in $lineSegmentConcatenationList$ do
    $temporaryLine \leftarrow fitLine(g)$
    $error \leftarrow computeError(temporaryLine, groundTruth(g))$
    if $error < errorCfg.concatenationError$ then
        $mergedLineSegments \leftarrow g$

```

get the lane center coordinates of the right lane which corresponds to "Shifted Lane" in Figure 2.4. This procedure can be applied to every arc of an arc spline.

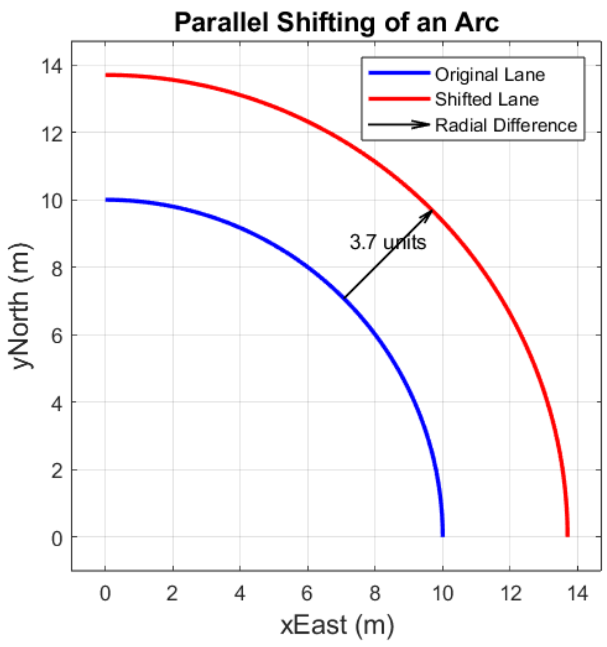

Figure 2.4: Parallel shifting of a single basic arc

Figure 2.5 shows the error between shifted lane and ground truth. The Euclidian distance error is computed by dividing the road segment to subsegments where each subsegment is 10 meters long. For each subsegment, RMS and maximum value of the error is computed.

For line segments, parallel shifting procedure is fairly simpler. The line segment's initial and final position is shifted in the direction perpendicular to the heading angle of line segment. The exact direction depends on the relative position of the base lane and desired lane. Figure 2.6 demonstrates a basic example for line parallel shifting. Similar procedure can be applied to line segments. Figure 2.7 shows the error between shifted lane and ground truth. The Euclidian distance error is computed by dividing the road segment to subsegments where each subsegment is 10 meters long. For each subsegment, RMS and maximum value of the error is computed.

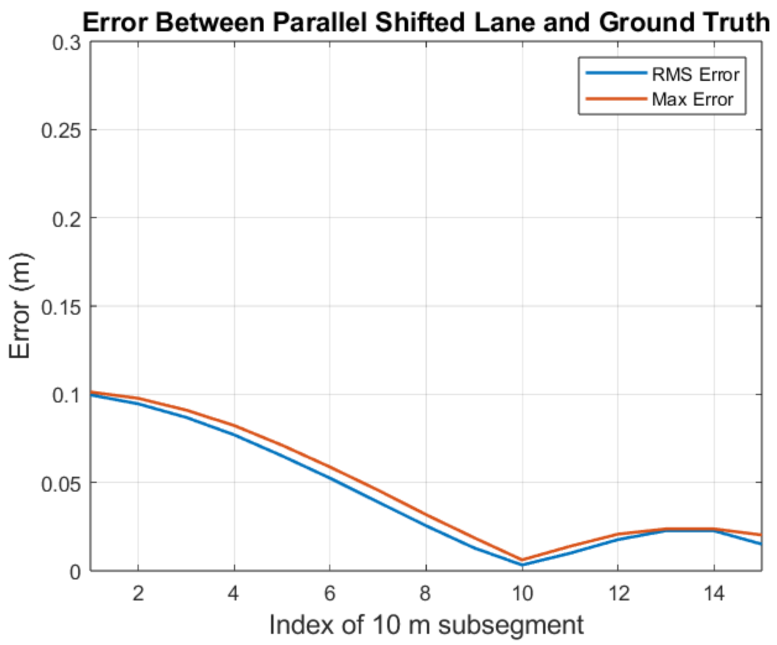

Figure 2.5: Parallel shifting error of an arc spline representing a real road segment

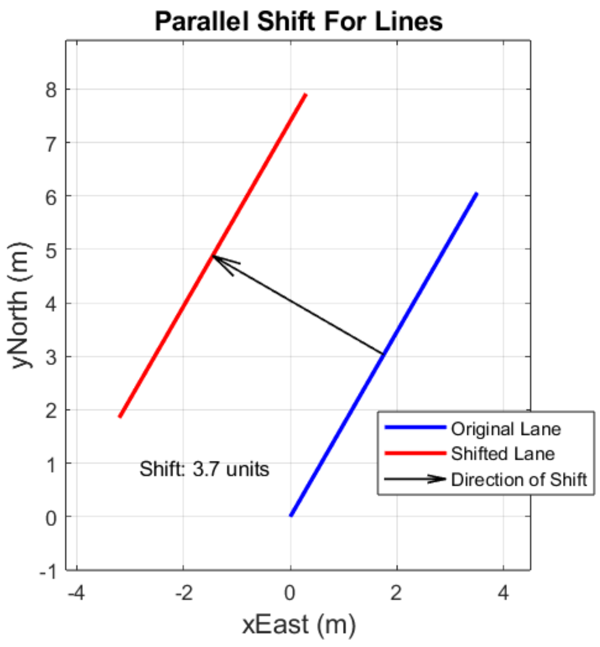

Figure 2.6: Parallel shifting of a single basic arc

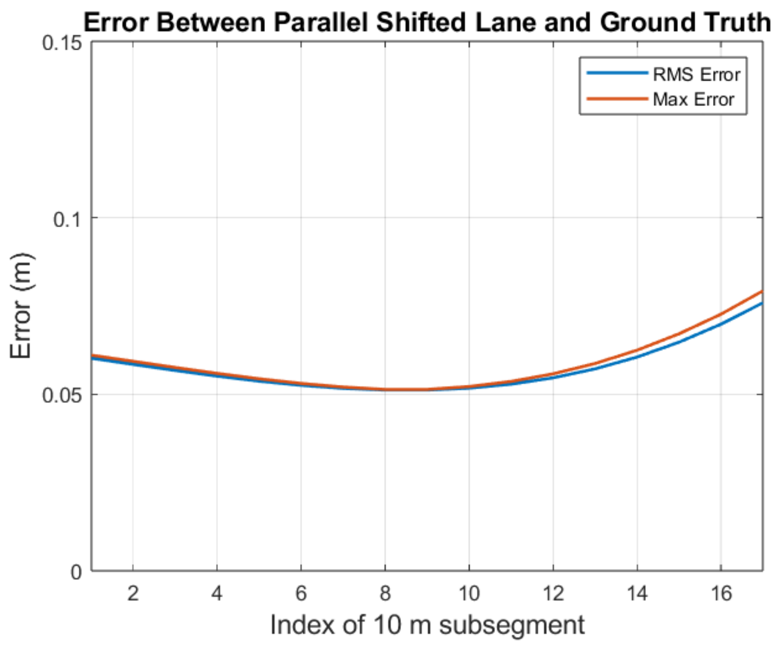

Figure 2.7: Parallel shifting error of a line segment representing a real road segment

2.5 Evaluation Metrics and Results

For experiment, Autobahn 38 in Germany is selected. Data is downloaded from HERE maps and clothoid is fitted between waypoints. Satellite imagery together with clothoids is given in Figure 2.8.

2.5.1 RMSE Spatial Coordinates

To evaluate the accuracy of the approximations, Root Mean Squared Error (RMSE) is computed. RMSE is useful since it is possible to see the overall performance of the approximation. Road segments have lengths varying from 20 meters to 400 meters therefore it would not be feasible to directly compute the RMSE for whole segment. We might need to inspect some parts of segments to observe the approximation in more detail also RMSE for long sequential data may hide some error peaks since overall error would be low. Therefore, RMSE is computed for 10 meter long subsegments i.e each segment is divided to 10 meter long subsegments and RMSE is computed for these subsegments.

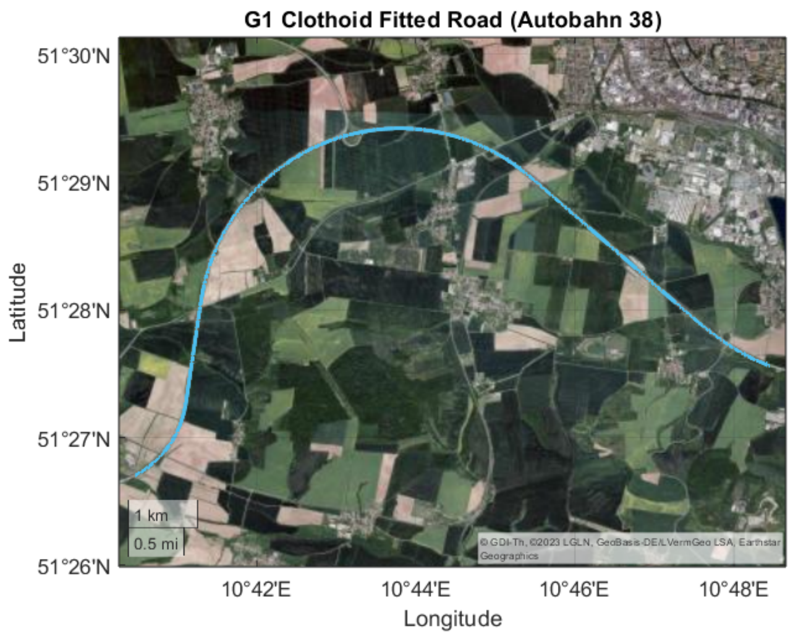
G1 Clothoid Fitted Road (Autobahn 38)
51°30'N
51°29'N
51°28'N
51°27'N
51°26'N
Latitude
10°42'E
10°44'E
10°46'E
10°48'E
Longitude
1 km
0.5 mi
©GDI-Th, ©2023 LGLN, GeoBasis-DE/LVermGeo LSA, Earthstar Geographics

Figure 2.8: Satellite image overlaid with G1 fitted clothoid road segments (plotted in light blue)

Figure 2.9 shows the RMS error plot over a ~370 meters long segment. For every 10 meter long segment the RMSE is computed. There is a data point for each centimeter for line segments. The RMS error reaches 9 centimeters at most which would be accurate enough for autonomous driving.

Figure 2.10 shows the RMS error plot over a ~150 meters long arc spline segment. There is a data point for each centimeter for arc spline segments. The order of arc spline approximation for this segment is 6 which means that this segment's curvature rate is relatively high for a highway.

Figure 2.11 shows the RMS error plot over a ~12500 meters long road. For each segment, the RMSE is computed. As mentioned before, the length of each road segment can vary from 20 meters to 400 meters. In this plot, segments are classified as either arc spline or line segment. The RMS error over an entire segment never exceeds 0.1 meters. This result is promising given that there are 323 segments, and this road includes every type of section a highway would have, such as curves and straight parts.

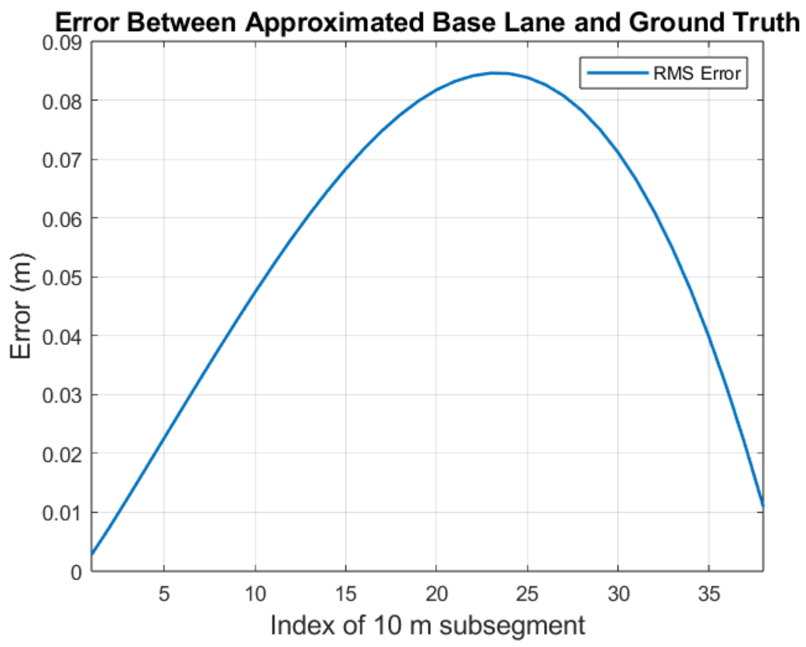

Figure 2.9: RMSE of a line approximated segment

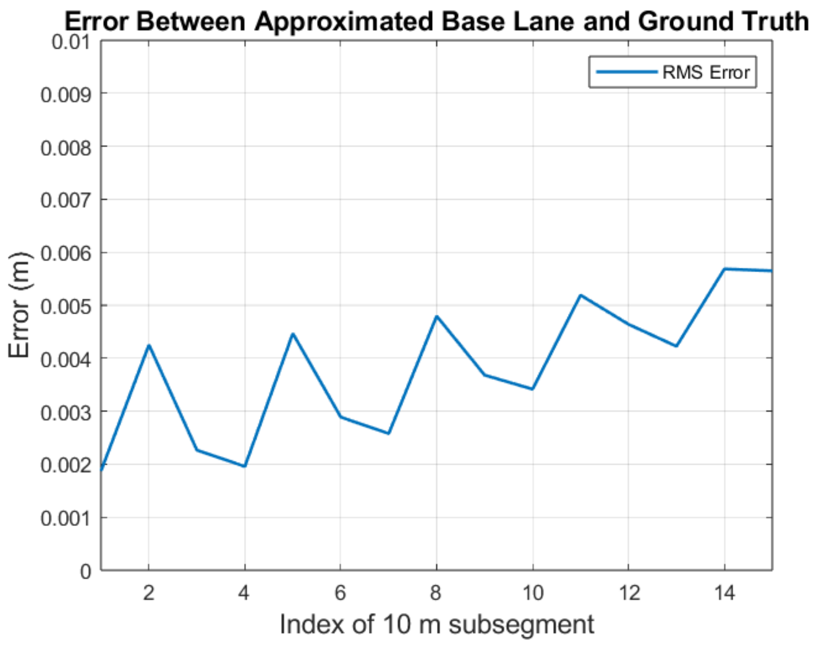

Figure 2.10: RMSE of a arc spline approximated segment

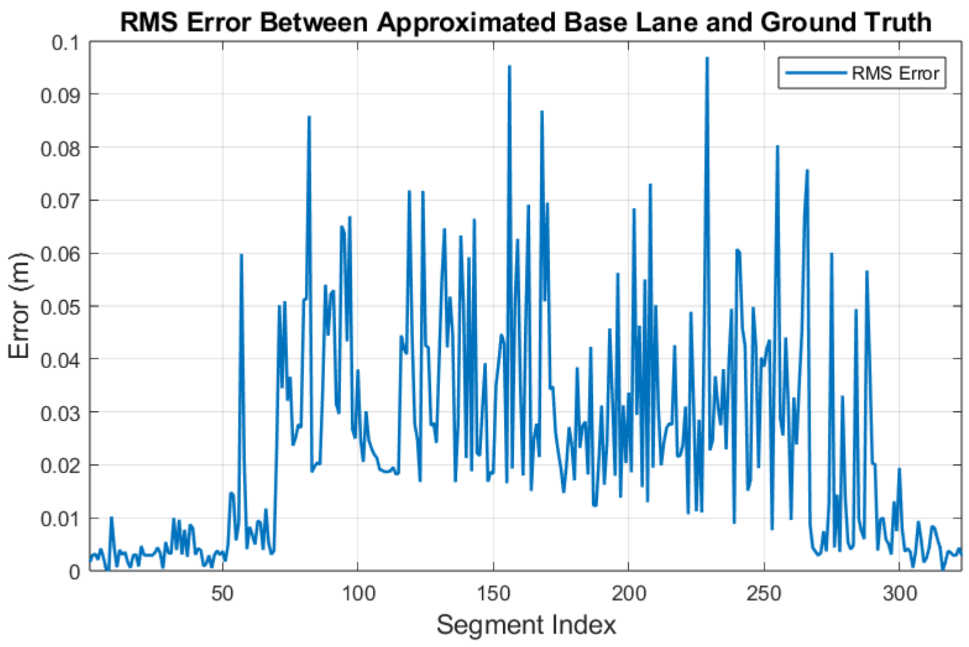

Figure 2.11: RMSE for a 323 segment road (overall length around 12.5 km)

Figure 2.12 shows the RMS error and average curvature on the same plot. As mentioned in 1.2.2 every waypoint has the curvature data stored in it and road segments are the structures that connect waypoints. To define a curvature for a road segment, the average of curvature values at both ends of a segment is computed. High curvature parts of the road seem to have low RMS error. This could be because algorithm uses higher order arc spline approximation for the curved sections of the road.

Figure 2.13 shows the RMS error and segment length on the same plot. It is observed that the error is not correlated with the segment length. Long segments do not have significantly high error values. Similarly short segments do not have low error values.

2.5.2 Maximum Error In Spatial Coordinates

To further evaluate the practicality of the approximations, maximum error is computed. maximum error is an important metric since it means that if a vehicle follows the approximated road, it will be guaranteed that it will not leave the lane. Given that road segments can range in length from 20 meters to 400 meters, directly calculating the maximum error for an entire segment is impractical. It may be necessary to ex-

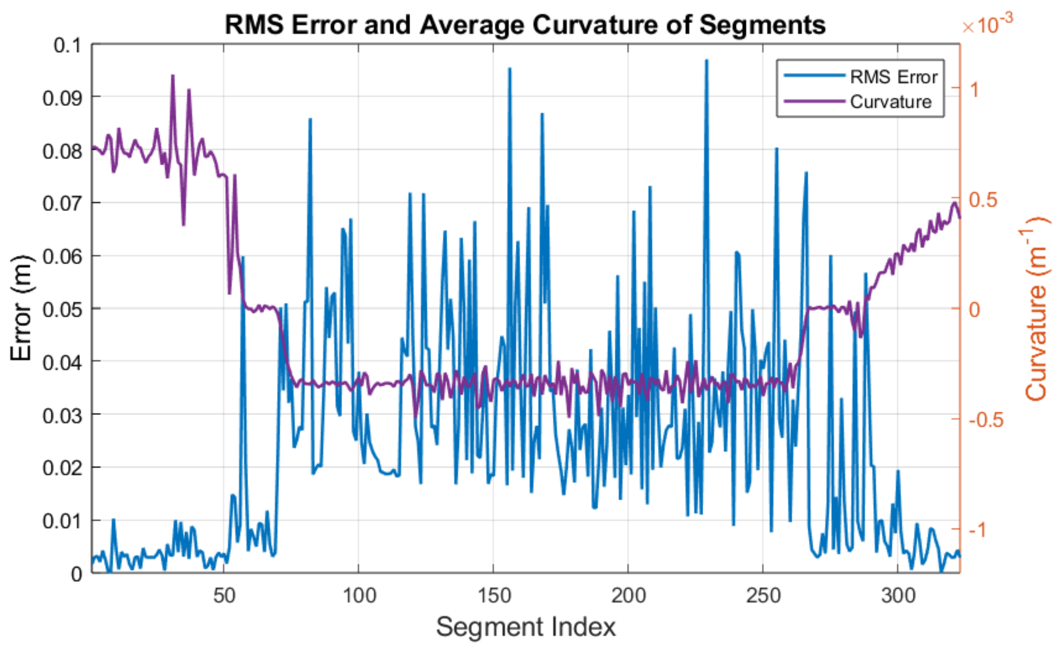

Figure 2.12: RMSE and average curvature for a 323 segment road (overall length around 12.5 km)

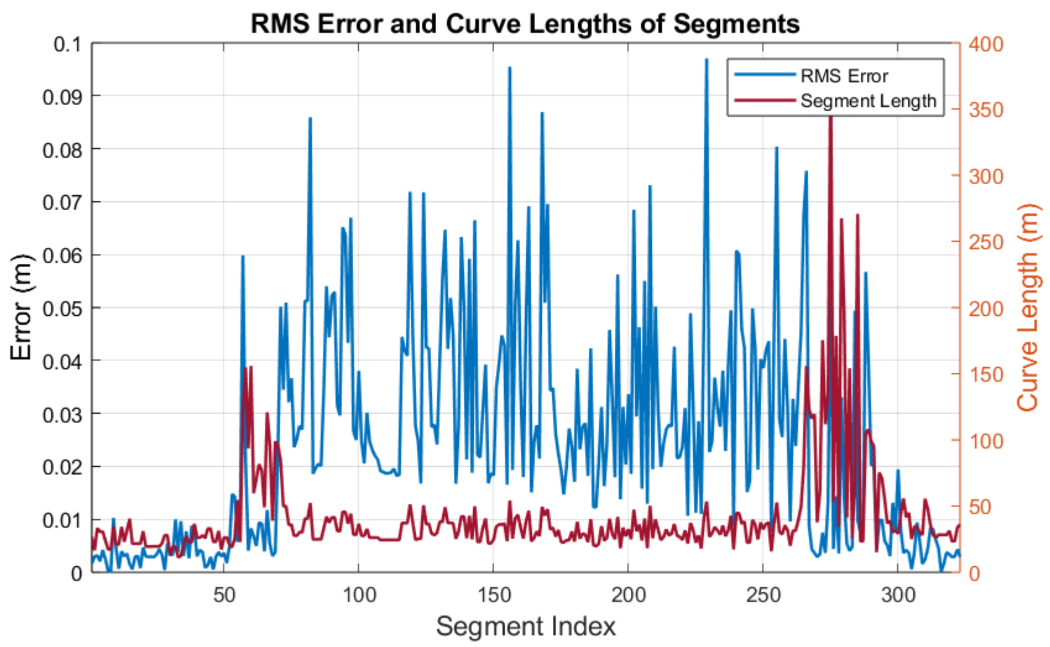

Figure 2.13: RMSE and segment length for a 323 segment road (overall length around 12.5 km)

amine specific parts of segments to closely observe the approximation. Additionally, calculating the maximum error for long sequences of data might obscure other error peaks, as it only highlights the highest value. Therefore, maximum error is computed for 10 meter long subsegments i.e each segment is divided to 10 meter long subsegments and maximum error is computed for these subsegments.

Figure 2.14 shows an maximum error error plot over a ~370 meters long segment. For every 10 meter long segment the maximum error is computed.

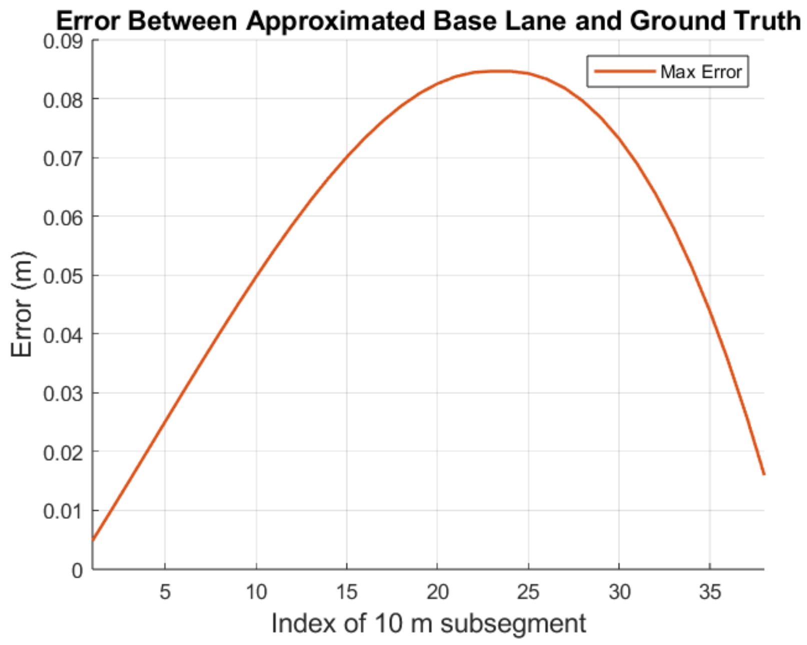

Figure 2.14: Maximum error of a line approximated segment

Figure 2.15 shows an maximum error plot over a ~150 meters long segment. For every 10 meter long segment the maximum error is computed.

Figure 2.16 shows an maximum error plot over a ~12500 meters long road. For each segment maximum error is computed. As mentioned before, length of each road may vary from 20 meters to 400 meters. The segments may be classified as arc spline or line segment in this plot.

Figure 2.17 shows an maximum error and average curvature on the same plot. As mentioned in 1.2.2 every waypoint has the curvature data stored in it and road segments are the structures that connect waypoints. To define a curvature for a road

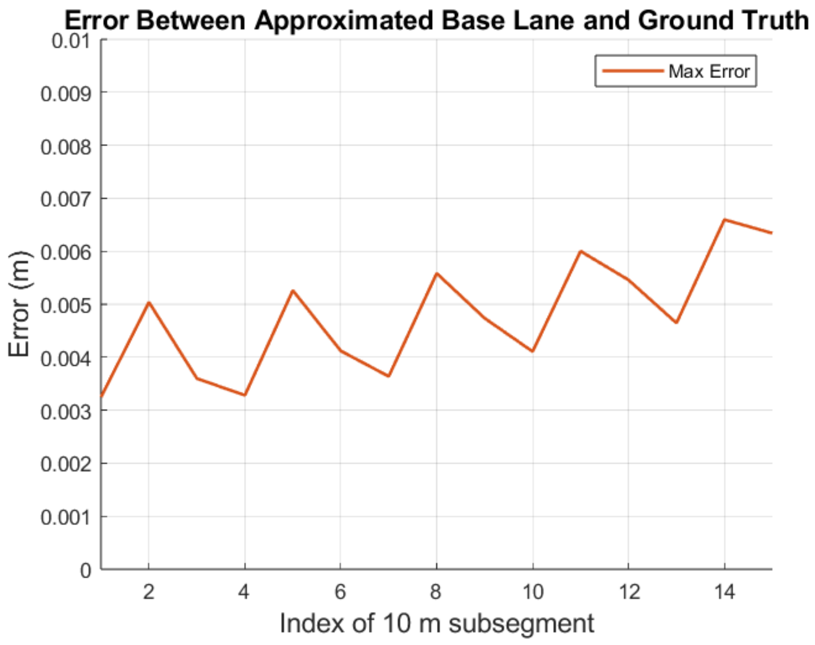

Figure 2.15: Maximum error of a arc spline approximated segment

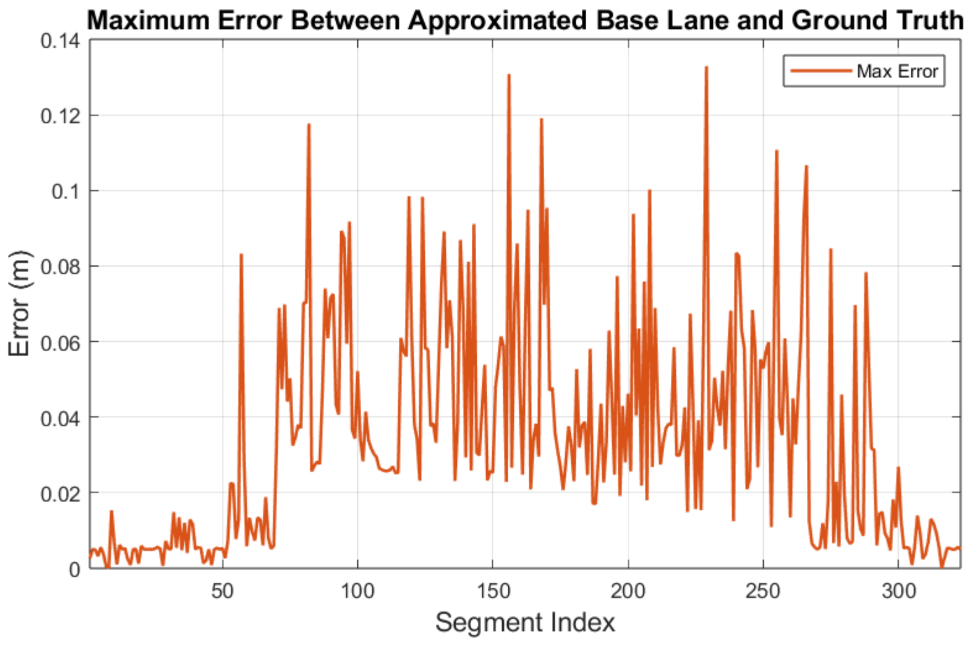

Figure 2.16: Maximum error values for a 323 segment road (overall length around 12.5 km)

segment, the average of curvature values at both ends of a segment is computed.

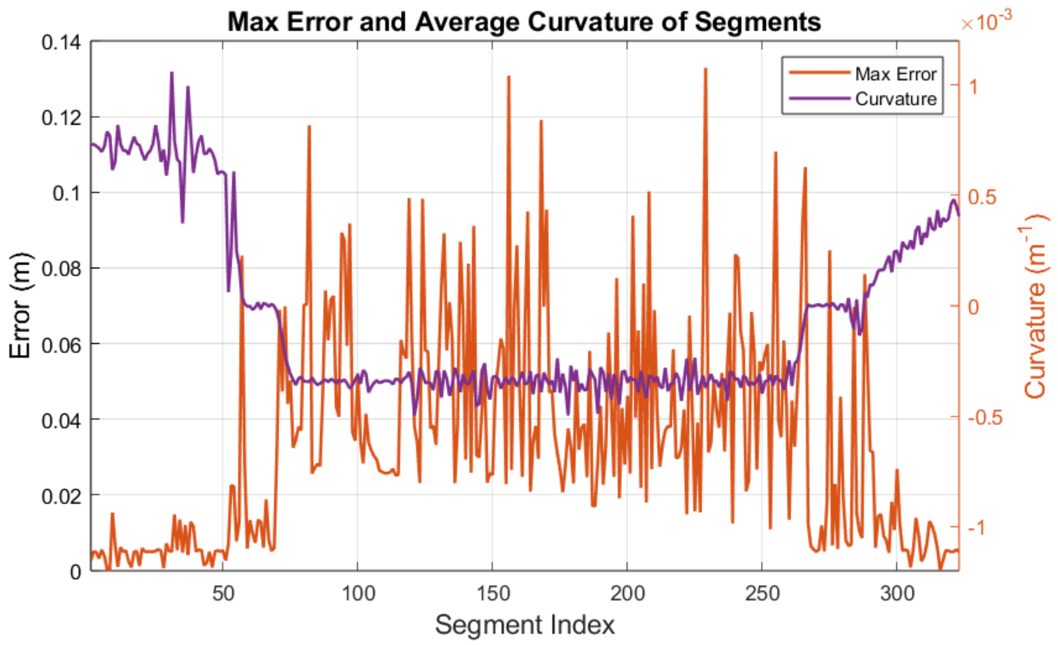

Figure 2.17: Maximum error and average curvature for a 323 segment road (overall length around 12.5 km)

Figure 2.18 shows the maximum error and segment length on the same plot. It is observed that the error is not correlated with the segment length. Long segments do not have significantly high error values. Similarly short segments do not have low error values.

2.5.3 Number of Segments

As mentioned in 2.3 segments may be merged on some error configurations. This would result in reduced number of segments and parameters which would decrease the memory usage of the offline map.

Table 2.2 shows the number of segments before and after combination operation. !TODO TABLE SHOWING THE NUMBER OF SEGMENTS BEFORE AND AFTER MERGE!

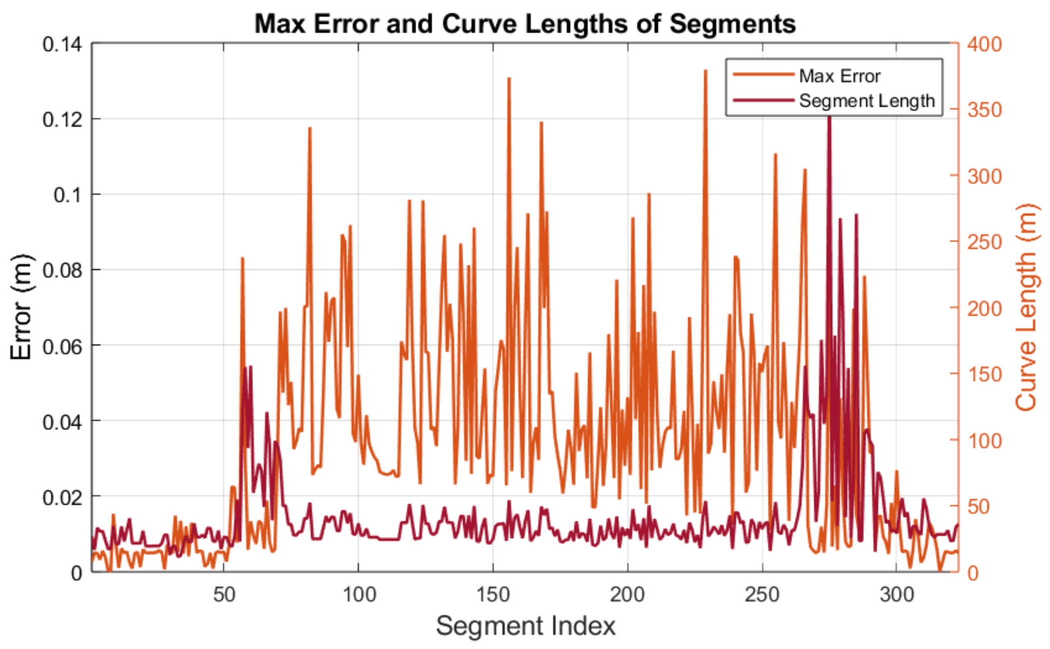

Figure 2.18: Maximum error and segment length for a 323 segment road (overall length around 12.5 km)

	Endpoint Degree Deviation	RMS Error (m)	Maximum Error (m)
Line Segment	0.2°	0.10	0.15
Arc Spline	-	0.10	0.15

Table 2.2: Error Table

CHAPTER 3

TRAJECTORY GENERATION METHODOLOGY

3.1 Bézier Trajectory Generation

Bézier curves are highly useful for the trajectory planning problem addressed in this thesis. Bézier curves require initial pose of the ego vehicle and a final pose to generate a trajectory. For our case initial pose is already given. We can give any point along a road segment as final pose since we have already represented the whole road with either arc splines or line segments. Thanks to this road representation method we can compute the required pose for a vehicle at any point on the road segment.

It is important to generate many candidate Bézier curves for trajectory planning. Collision checking algorithms need a variety of Bézier curves because a single curve does not guarantee a collision-free path. By having multiple candidates, we can find a path that avoids obstacles. Additionally, generating many candidates allows us to choose a path with the desired curvature characteristics, which helps in keeping the vehicle stable and comfortable. This method balances safety and computational efficiency, allowing us to maintain real-time performance while exploring different trajectory options. Thus, having a variety of Bézier curve candidates is essential for reliable trajectory planning in autonomous vehicles.

3.1.1 Increasing the Number of Candidate Curves

As mentioned in 1.2.3 quintic bezier curves have 6 control points. It is not possible to change the position of first and last control point since they directly affect the initial and final point of the curve. However, it is possible to slightly vary P_1, P_2, P_3 and P_4

points while maintaining the initial and final pose [8].

Control point computations are given is computed with the equation 3.1, 3.2, 3.3 and 3.4. In these equations t_0 and t_f correspond to initial and final velocity vectors respectively. Similarly a_0 and a_f vectors correspond to initial and final acceleration vectors respectively. In order to have a variety of Bézier curves we need to have a variety of control points. To achieve this we can change the magnitude of velocity and acceleration vectors. It should be noted that, the directions of velocity and acceleration vectors are directly related to pose hence the directions of these vectors cannot be altered.

$$P_1 = P_0 + \frac{1}{5}\vec{t_0} \tag{3.1}$$

$$P_2 = \frac{1}{20}\vec{a}_0 + 2P_1 - P_0 \tag{3.2}$$

$$P_3 = \frac{1}{20}\vec{a}_f + 2P_4 - P_0 \tag{3.3}$$

$$P_4 = P_5 - \frac{1}{5}\vec{t_f} - P_5 \tag{3.4}$$

Equation 3.5 gives the formulation for $\vec{t_0}$ candidate computation. d is the distance between starting point and destination point, m_t is the tangential multiplier and $\vec{t_0^u}$ is the unit vector in the direction of initial heading.

m_t is a design parameter for Bézier curve candidates. Possible values of m_t are given in 3.6. $m_t^{\min}$ and $m_t^{\max}$ are design parameters that affect the shape of Bézier curves, N_t is the number of candidates for each tangential vector.

$$\vec{t_0} = d \cdot m_{tn} \cdot \vec{t_0^u} \tag{3.5}$$

$$\forall m_{tn} \in \left[m_t^{\min}, m_t^{\max}\right] \quad n = 1, \ldots, N_t \tag{3.6}$$

Similar to $\vec{t_0}$, equation 3.7 gives the formulation for $\vec{t_f}$ candidate computation. $\vec{t_f^u}$ is the unit vector in the direction of destination heading. For both $\vec{t_0}$ and $\vec{t_f}$ same m_{tn} candidates are used.

$$\vec{t_f} = d \cdot m_{tn} \cdot \vec{t_f^u} \tag{3.7}$$

Equation 3.8 gives the formulation for $\vec{a}_0$ candidate computation. m_κ is the acceleration multiplier and $\vec{t_0^n}$ is the unit vector in the direction of initial curvature. Initial

Parameter	Value
N_t	5
N_k	3
$m_t^{\min}$	0.3
$m_t^{\max}$	1.7
$m_k^{\min}$	0
$m_k^{\max}$	5

Table 3.1: Configuration Parameters for Candidate Bézier Curve Generation

curvature direction is the vector connecting the vehicle's current point to turning center. m_κ is a design parameter for Bézier curve candidates. Possible values are given in 3.9. $m_k^{\min}$ and $m_k^{\max}$ are design parameters that affect the shape of Bézier curves, N_k is the number of candidates for each acceleration vector.

$$\vec{a}_0 = \frac{d}{5} \cdot m_\kappa \cdot \vec{t_0^u} + d^2 \cdot \kappa_i \cdot \vec{t_0^n} \tag{3.8}$$

$$\forall m_\kappa \in \left[m_k^{\min}, m_k^{\max}\right] \quad n = 1, \ldots, N_k \tag{3.9}$$

Similar to $\vec{a}_0$, equation 3.10 gives the formulation for $\vec{a}_f$ candidate computation. $\vec{t_f^u}$ is the unit vector in the direction of destination curvature. For both $\vec{a}_0$ and $\vec{a}_f$ same m_κ candidates are used.

$$\vec{a}_f = \frac{d}{5} \cdot m_\kappa \cdot \vec{t_f^u} + d^2 \cdot \kappa_f \cdot \vec{t_f^n} \tag{3.10}$$

Table 3.1 gives the configuration parameters used in this thesis. The values of N_t and N_k define the number of curve candidates which is 225 for these values. The limits of tangential and acceleration multipliers are picked to be low since this research involves highway trajectory planning and trajectories on highways cannot tolerate high curvatures.

For better understanding of these parameters, a simulation is given with $B = [(0,0)$ $, (15,10), 0, -45°, 0, 0.01]$ and some control points are varied while keeping the others same.

Figure 3.1 demonstrates the effect of varying P1. As the magnitude of t_0 increases, the trajectory moves in the direction of initial tangent more. This parameter can be associated with initial velocity of the ego vehicle.

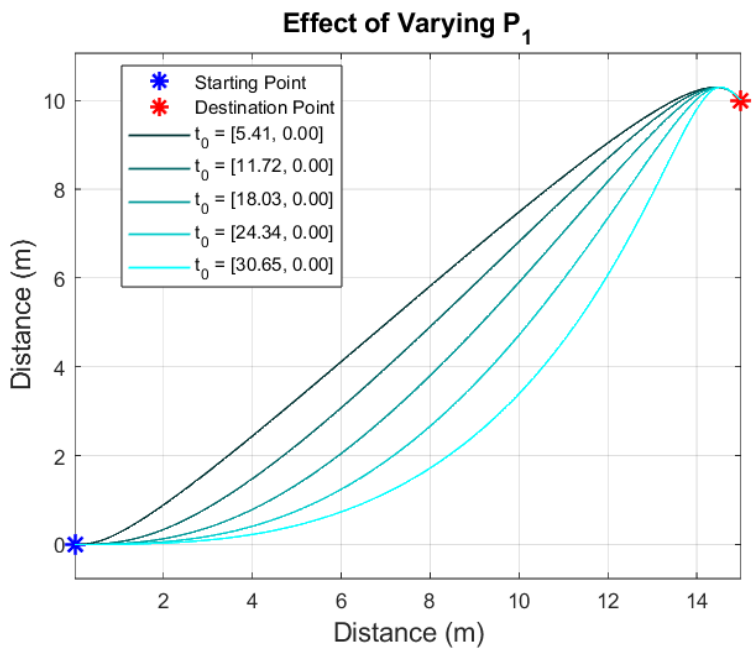

Figure 3.1: Effect of varying P1 while keeping the other parameters same

Figure 3.2 demonstrates the effect of varying P2. As the magnitude of a_0 increases, the trajectory slightly changes at the beginning of the trajectory. The effect is minor since the 3.8 has a damping factor for $\vec{a}_0$. This parameter can be associated with initial curvature of the ego vehicle.

Figure 3.3 demonstrates the effect of varying P3. As the magnitude of a_f increases, the trajectory slightly changes towards the end of the trajectory. The effect is minor since the 3.10 has a damping factor for $\vec{a}_f$.

Figure 3.4 demonstrates the effect of varying P4. As the magnitude of t_f increases, the trajectory moves in the direction of destination tangent more. This parameter can be associated with destination velocity.

3.2 Arc Spline Trajectory Generation

Arc splines are quite useful for trajectory planning since these splines consist of consecutive arcs, it is very easy to get the steering input required to follow the trajectory. Arc spline trajectories require a initial pose and a clothoid road as input. The output

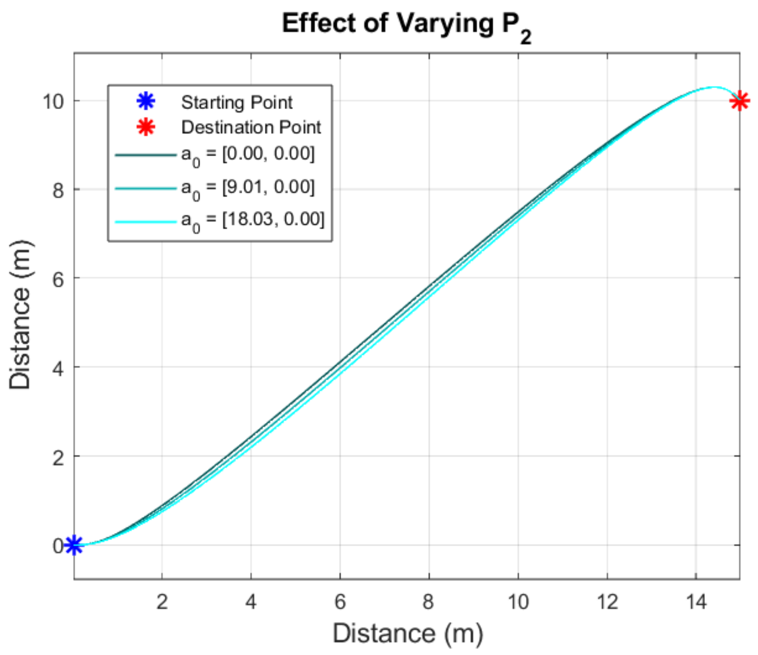

Figure 3.2: Effect of varying P2 while keeping the other parameters same

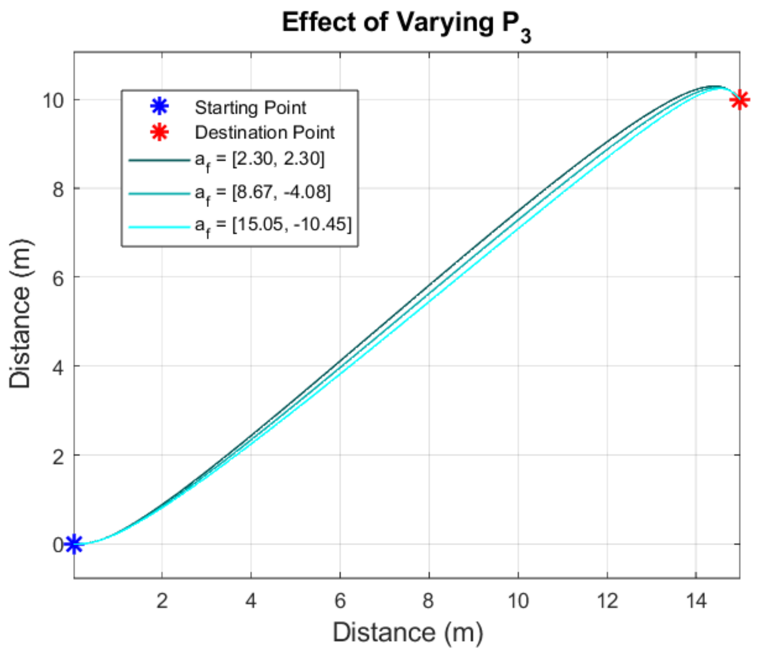

Figure 3.3: Effect of varying P3 while keeping the other parameters same

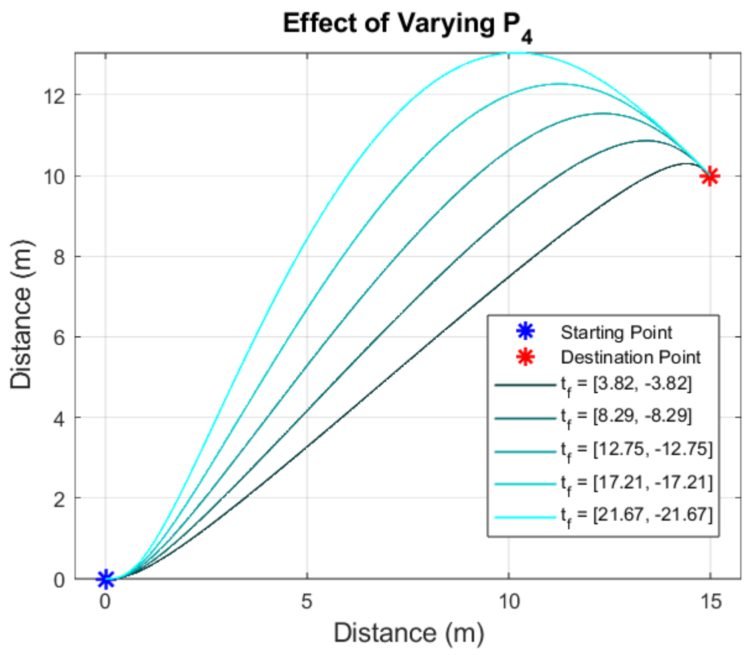

Figure 3.4: Effect of varying P4 while keeping the other parameters same

is a series of waypoints which may be connected with arc splines.

To compute an arc spline for a vehicle that is outside the road centerline, it is necessary to correct the heading, curvature, and position. This method is divided into two parts: the first part describes the heading and curvature correction algorithm, while the second part details the position correction algorithm.

3.2.1 Heading and Curvature Correction

In this part heading and curvature correction maneuvers will be described. Let's assume that a vehicle is exactly on the road centerline however it has heading and curvature error. The initial heading and curvature error can be compensated by cleverly computing required curvature rate along some length. There are 4 cases for correction. The first case is where the initial heading error ($theta_{error}$) is positive and initial curvature error (κ_{error}) is negative. We want to make the curvature and heading error zero with a clothoid. The required curvature plot is given in Figure 3.5.

While correcting the curvature it is necessary to correct for heading as well since they are closely related to each other. It is known that heading change along a curve is

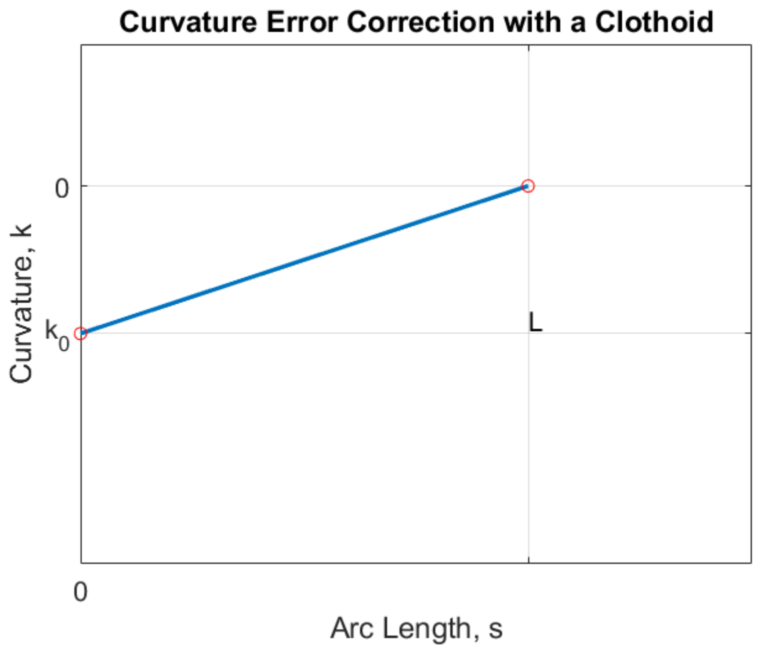

Figure 3.5: Curvature error correction with a clothoid

the integral of curvature. With this knowledge equation 3.11 is constructed. σ and L stand for curvature rate and curve length respectively.

$$\theta_{error} + \kappa_{error} \cdot L + \frac{\sigma \cdot L^2}{2} = 0 \tag{3.11}$$

Since Figure 3.5 is a clothoid, it has a constant curvature rate depending on κ_{error} and L. The relation is given in equation 3.12.

$$\sigma = \frac{\kappa_{error}}{L} \tag{3.12}$$

Substituting 3.12 into 3.11 and solving for L and σ yields the equations 3.13 and 3.14 respectively.

$$L = -\frac{2 \cdot \theta_{error}}{\kappa_{error}} \tag{3.13}$$

$$\sigma = \frac{\kappa_{error}^2}{2 \cdot \theta_{error}} \tag{3.14}$$

With these equations, we have the required curve length and curvature rate for heading and curvature correcting maneuver. With the assumption of a completely straight road, it is possible to generate the next waypoint that has zero curvature and heading error at the next endpoint.

Figure 3.6 shows the change of curvature and heading along an arc spline.

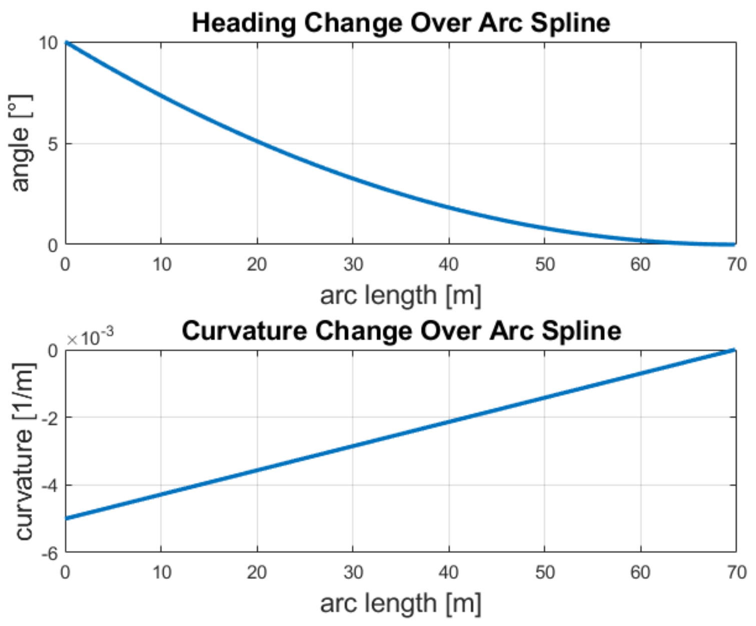

Figure 3.6: Curvature and heading error correction with a clothoid

The second case is where the initial heading error (θ_{error}) is positive and initial curvature error (κ_{error}) is also positive. We want to make the curvature and heading error zero with clothoids. The curvature plot is given in Figure 3.5.

In this case it is not possible to correct the heading and curvature with a single clothoid. Two clothoids with opposite curvature rate are needed which is called a counter maneuver. Figure 3.7 shows the required heading and curvature change along an arc spline. In this plot, two clothoids have curvature rates of equal magnitude but opposite sign. This constraint can be expressed together with curvature correction by the equation 3.15.

$$k_0 - \sigma \cdot l_1 + \sigma \cdot l_2 = 0 \tag{3.15}$$

Similar to case 1, $theta_{error}$ must be zero at the end of the arc spline. The equation for this constraint is given in equation 3.16 where l_1 and l_2 are the length of first and second clothoids respectively.

$$\theta_{error} + \kappa_{error} \cdot l_1 - \frac{\sigma}{l_1^2} + (\kappa_{error} - \sigma \cdot l_1) \cdot l_2 + \frac{\sigma \cdot l_2^2}{2} = 0 \tag{3.16}$$

Equation 3.16 must be solved for l_1, l_2 and σ in order to define the subsequent waypoints. However, this equation has more freedom than case 1 therefore it makes sense to fix a variable to reduce the degree of freedom. It does not make sense to fix l_1

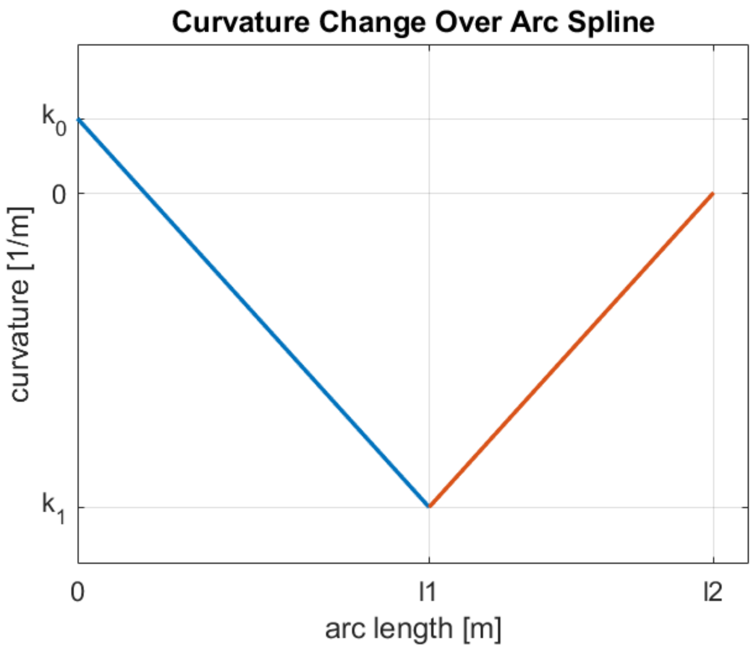

Figure 3.7: Curvature error correction with two clothoids

and l_2 since constraining trajectory length is risky. The best choice is to fix σ since it depends on vehicle dynamics. In this implementation the value of σ is chosen to be 0.0025. Substituting the equation 3.15 into 3.16 yields the solution for l_1 and l_2 given in 3.17 and 3.18 respectively.

$$l_1 = \frac{\kappa_{error}}{\sigma} + \sqrt{\frac{\kappa_{error}^2}{2 \cdot \sigma^2} + \frac{\theta_{error}}{\sigma}} \tag{3.17}$$

$$l_2 = l_1 - \frac{\kappa_{error}}{\sigma} \tag{3.18}$$

When these clothoids are imposed on the initial pose, heading and curvature change over the ego vehicle can be observed in Figure 3.8.

Case 3 ($theta_{error} < 0, \kappa_{error} < 0$), has the same solution as case 2.

Similarly case 4 ($theta_{error} < 0, \kappa_{error} > 0$), has the same solution as case 1.

By applying these trajectories it is possible to correct heading and curvature along a completely straight road. However, this method introduces position error which will be mentioned in 3.2.2 section.

The algorithm 5 describes the position heading and curvature correction maneuver. $correctionManeuver$ is the heading and curvature correction clothoid maneuver.

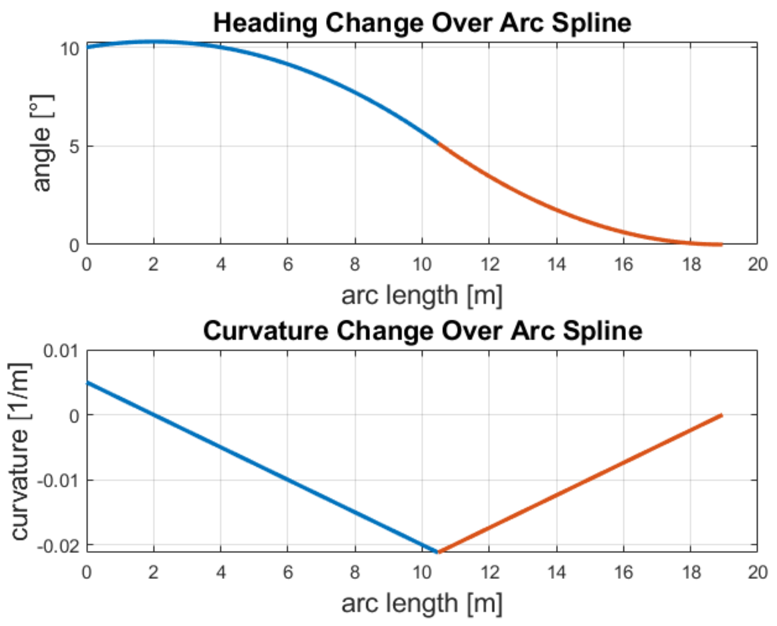

Figure 3.8: Curvature and heading error correction with two clothoids

The algorithm is written for case 1 and 2 only. Case 3 and 4 are asymmetrical to case 1 and case 2. "Hcc" stands for heading and curvature correction.

3.2.2 Position Correction

Position correction with arc splines require a maneuver called bi-elementary paths. Bi-elementary paths take position error and trajectory length as input as described in section1.2.4. In this implementation λ and γ values are chosen to be 0.5 for a more harmonic trajectory.

Even if the position error of a vehicle's initial pose is known, the position error is introduced by heading and curvature correction maneuver as mentioned in 3.2.1.

In order to compensate for the initial error and the error introduced by heading and curvature correction maneuver, the position error after heading and curvature error correction maneuver is given input to bi-elementary path parameter computation.

The length of the bi-elementary path is chosen to be the same as heading and curvature correction maneuver. With this design decision, it is possible to make both maneuvers

Algorithm 5 Heading and Curvature Error Correction

```
1: procedure COMPUTEHCCMANEUVER($x_0$, $y_0$, $\theta_0$, $\kappa_0$, $clothoidRoadSegments$)
2:     $\theta_{error} \leftarrow theta_0 - roadInitialHeading$
3:     $\kappa_{error} \leftarrow theta_0 - roadInitialHeading$
4:     if $\theta_{error} > 0$ and $\kappa_{error} < 0$ then
5:         $case \leftarrow 1$
6:         $\sigma \leftarrow \frac{\kappa_{error}^2}{2 \cdot \theta_{error}}$
7:         $hccLength = \frac{-2 \cdot \theta_{error}}{\kappa_{error}}$
8:     else if $\theta_{error} > 0$ and $\kappa_{error} > 0$ then
9:         $case \leftarrow 2$
10:        $\sigma \leftarrow 0.0025$
11:        $l_1 \leftarrow \frac{\kappa_{error}}{\sigma} + \sqrt{\frac{\kappa_{error}^2}{2 \cdot \sigma^2} + \frac{\theta_{error}}{\sigma}}$
12:        $l_2 \leftarrow l_1 - \frac{\kappa_{error}}{\sigma}$
13:    end if
14:    if $case = 1$ then
15:        $\kappa_1 \leftarrow \kappa_0 + \sigma \cdot hccLength + roadCurvature \cdot hccLength$
16:        $arcSpline \leftarrow \mathcal{C} = [x_0, y_0, \theta_0, \kappa_0, \kappa_1, hccLength]$
17:    else if $case = 2$ then
18:        $\kappa_1 \leftarrow \kappa_0 - \sigma \cdot l_1 + roadCurvature \cdot l_1$
19:        $arcSpline(1) \leftarrow \mathcal{C} = [x_0, y_0, \theta_0, \kappa_0, \kappa_1, l_1]$
20:        $x_1, y_1 \leftarrow endPoint(arcSpline(1))$
21:        $\theta_1 \leftarrow endHeading(arcSpline(1))$
22:        $\kappa_2 \leftarrow \kappa_1 + \sigma \cdot l_2 + roadCurvature \cdot l_2$
23:        $arcSpline(2) \leftarrow \mathcal{C} = [x_1, y_1, \theta_1, \kappa_1, \kappa_2, l_2]$
24:    end if
25:    return $HCCarcSpline, positionError$
26: end procedure=0
```

at the same time.

To make the both maneuvers, maneuvers should be superposed as described section in 3.2.4.

3.2.3 Road Curvature Change Correction

It is known that the given road is a clothoid with a constant curvature rate along the road segment. Therefore, it is possible to compensate for curvature error along the road by simply integrating the curvature rate of the road.

The road's curvature rate may change while making the attitude correction maneuvers. In this case the change of the curvature rate should also be taken into account.

3.2.4 Superposition of Maneuvers

This is the section where the arc spline trajectory is generated by superposing the maneuvers described in sections 3.2.1, 3.2.3 and 3.2.3.

Generating an arc spline requires well defined consecutive $waypoints$. Each $waypoint$ consists of position, heading, curvature and length information. Length information is the arc spline length connecting the previous $waypoint$ to current $waypoint$. Length is determined by heading and curvature correction maneuver.

At the very beginning of the algorithm, the case at hand is determined as described in section 3.2.1. When the case is determined, automatically σ and length of arc spline is also determined.

Then, the required parameters for position correction maneuver is also computed with the output of heading and curvature correction maneuver.

Depending on the positions of waypoints, the road's curvature is taken into account as well.

After computing the required waypoints for heading and curvature correction maneuver, position correction maneuver and road curvature change correction maneuver it is possible to superpose these maneuvers. The superposition requires ordering of waypoints with respect to their positions along a road.

The algorithm 6 describes the arc spline trajectory computation process. First the number of different road segments is determined. Every time a road segment is changed, the curvature rate coming from the nature of the road segment changes. The curvature rate of the road is defined as $roadRate$.

Number of waypoints are the number of heading and curvature correction waypoints, number of bi-elementary path waypoints and number of number of road segments passed until the end of the heading and curvature correction maneuver length which is named $HCClength$.

Each waypoint consists of position, heading, curvature information. Additionally, for computational ease, waypoints store the required arc spline length from the previous waypoint.

$lengthSoFar$ is the current arc spline's length. It increases in for loop until it reaches $HCClength$. This variable is like the arc length pointer on Figure 3.9.

$findClosestWp$ function gets the closest waypoint in all the waypoints given the $lengthSoFar$. Then the required arc length until the next waypoint l_i is stored.

Rates are pulled from waypoint sets. At this point we know the position of our length pointer and we just pull the curvature rate for each maneuver along that segment.

Next curvature rate is computed with the rates and the length between the current waypoint and next waypoint by summation. This is the part where superposition is applied.

A clothoid (it is approximated with an arc spline) is constructed with the known parameters.

Finally, arc spline is added to the list and waypoint list is also updated.

Algorithm 6 Fit Arc Spline

```
1: procedure FitArcSpline($waypoints$)
2:     Input: $x_0$, $y_0$, $\theta_0$, $\kappa_0$, $HccWps$, $biElemWps$, $clothoidRoad$
3:     Output: $waypoints$, $arcSplines$
4:     $num(roadSegments) \leftarrow numberOfRoadSegments$ until $HCClength$
5:     $numWaypoints \leftarrow num(HccWps) + num(biElemWps) + num(roadSegments) + 1$
6:     $waypoints(0) = [x_0, y_0, \theta_0, \kappa_0, 0]$
7:     $lengthSoFar \leftarrow 0$
8:     for $i \leftarrow 1$ to $numWaypoints$ do
9:         $closestWp \leftarrow findClosestWp(lengthSoFar, HccWps, biElemWps, roadSegments)$
10:        $l_i \leftarrow length(closestWp)$
11:        $hccRate \leftarrow getHccCurvatureRate(lengthSoFar + l_i)$
12:        $biElemRate \leftarrow getBiElemCurvatureRate(lengthSoFar + l_i)$
13:        $roadRate \leftarrow getRoadCurvatureRate(lengthSoFar + l_i)$
14:        $lengthSoFar = lengthSoFar + l_i$
15:        $\kappa_i \leftarrow k_{i-1} + hccRate \cdot l_i + biElemRate \cdot l_i + roadRate \cdot l_i$
16:        $tempClothoid \leftarrow \mathcal{C} = [x_{i-1}, y_{i-1}, \theta_{i-1}, \kappa_{i-1}, \kappa_i, l_1]$
17:        $arcSplines(i) \leftarrow tempClothoid$
18:        $x_i, y_i \leftarrow tempClothoid.lastPosition$
19:        $waypoints(i) = [x_i, y_i, \theta_i, \kappa_i, l_i]$
20:    end for
21:    return $arcSpline$
22: end procedure
```

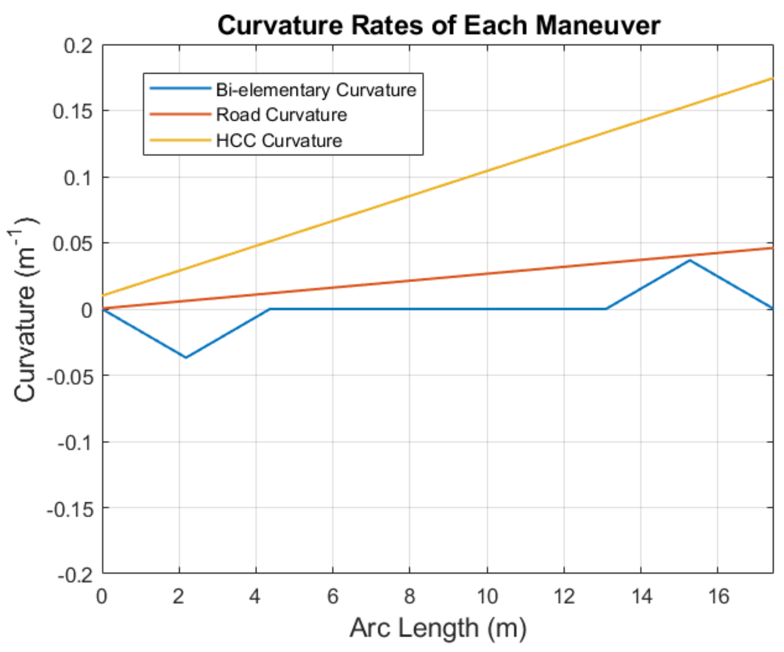

Figure 3.9: Curvature changes over $HCClength$ as an example. This data is simulated for visualization

3.3 Evaluation Metrics and Results

Evaluation metrics for trajectory algorithms include position, heading, and curvature errors. Position error is defined as the Euclidean distance between the planned trajectory and the road centerline along the path. Heading error represents the difference in orientation between the reference road and the computed trajectory. Curvature error measures the discrepancy in curvature between the road centerline and the computed trajectory.

The road centerline is a real road in Germany, specifically Autobahn 38. The error is artificially introduced by shifting the position of the road by desired position error. Similarly, heading and curvature error is also introduced by adding up the initial heading and curvature with heading error and curvature error respectively.

Table 3.2 shows the introduced errors for different cases. A positive position error indicates that the ego vehicle is to the left of the road centerline. A positive heading error signifies that the vehicle is oriented to the right. Similarly, a positive curvature error means that the vehicle is turning to the right more sharply than intended.

	Position Error	Heading Error	Curvature Error
Case 1	+0.55	+2°	0.01m^{-1}

Table 3.2: Error configurations for different cases

3.3.1 Arc Spline Trajectory

Figure 3.10 displays the error over an arc spline. The algorithm makes the position error zero at the end. However, while making the maneuver the position error increases initially, this is due to curvature error. Figure 3.12 shows the curvature error over arc spline. Since there is a negative curvature error initially the position error increases initially. Figure 3.16 displays the trajectory and road centerline.

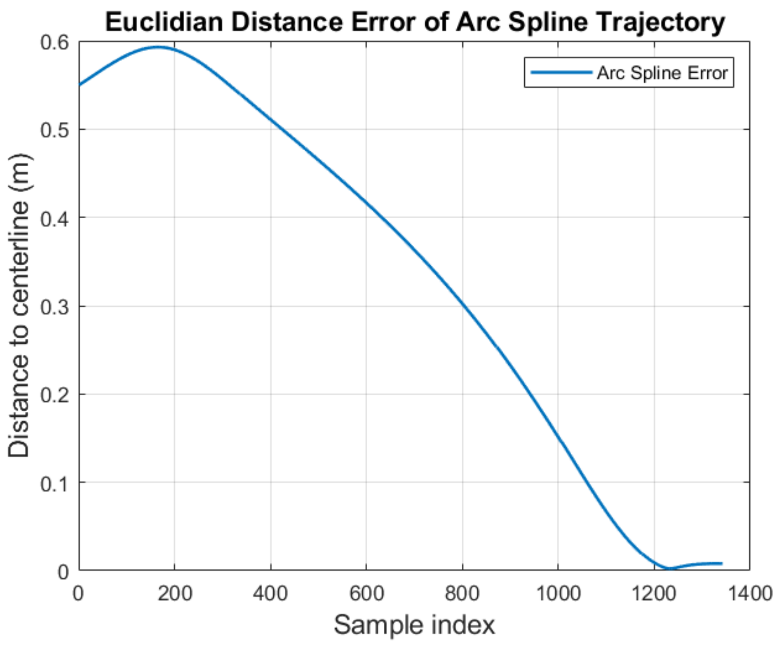

Figure 3.10: Position error over an arc spline trajectory for case 1

3.3.2 Bézier Trajectory

Figure 3.14 displays the error over a Bézier trajectory. The algorithm makes the position error zero at the end. However, while making the maneuver the position error significantly increases initially. Figure 3.15 shows the heading error over Bézier

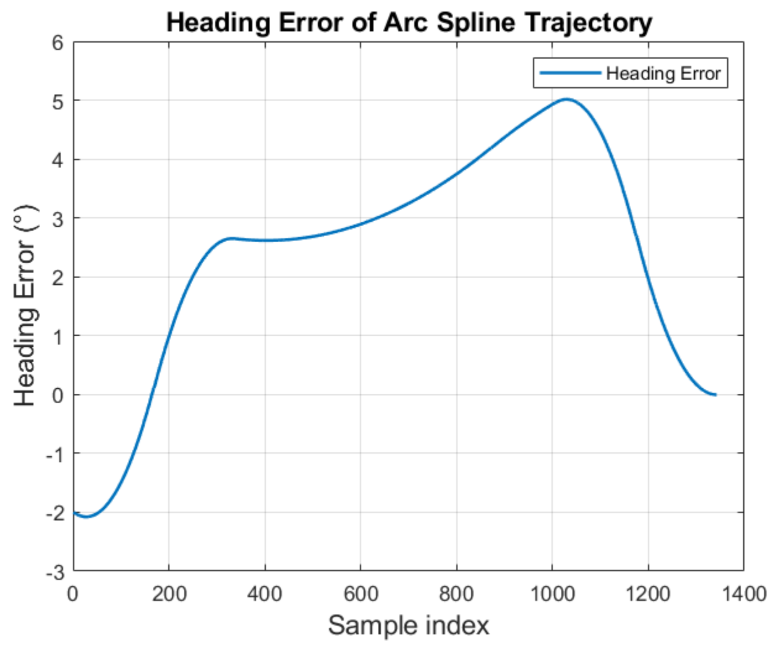

Figure 3.11: Heading error over an arc spline trajectory for case 1

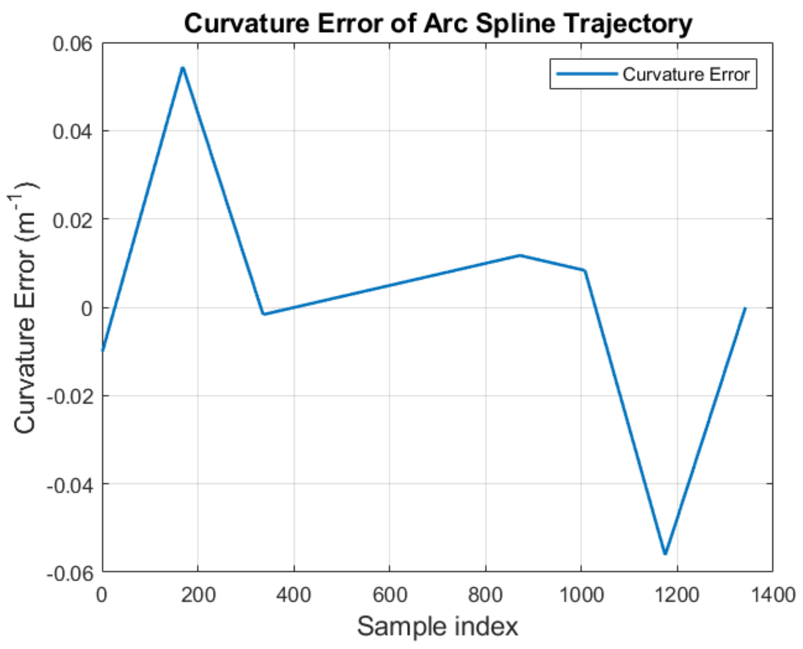

Figure 3.12: Curvature error over an arc spline trajectory for case 1

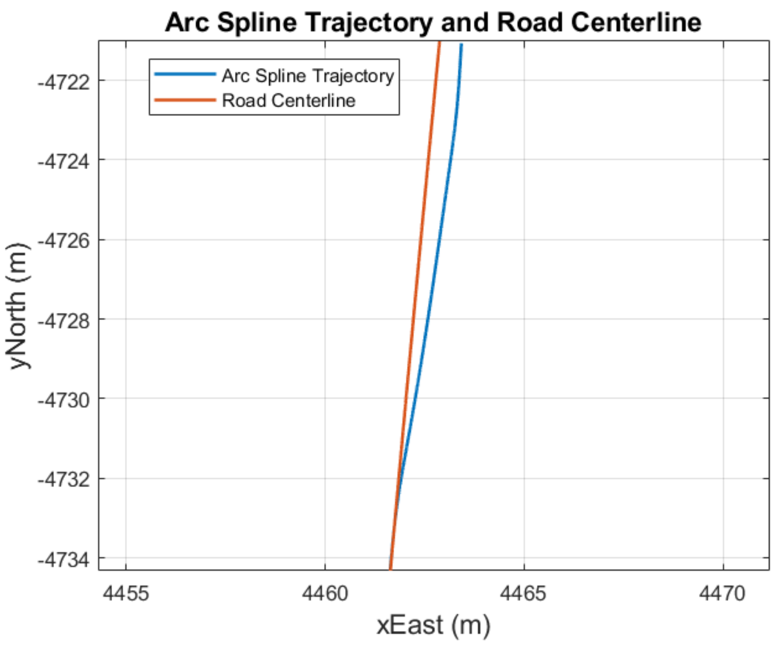

Figure 3.13: Arc spline trajectory on Autobahn 38 case 1

trajectory. The position error increases significantly and this is a problem for Bézier curves.

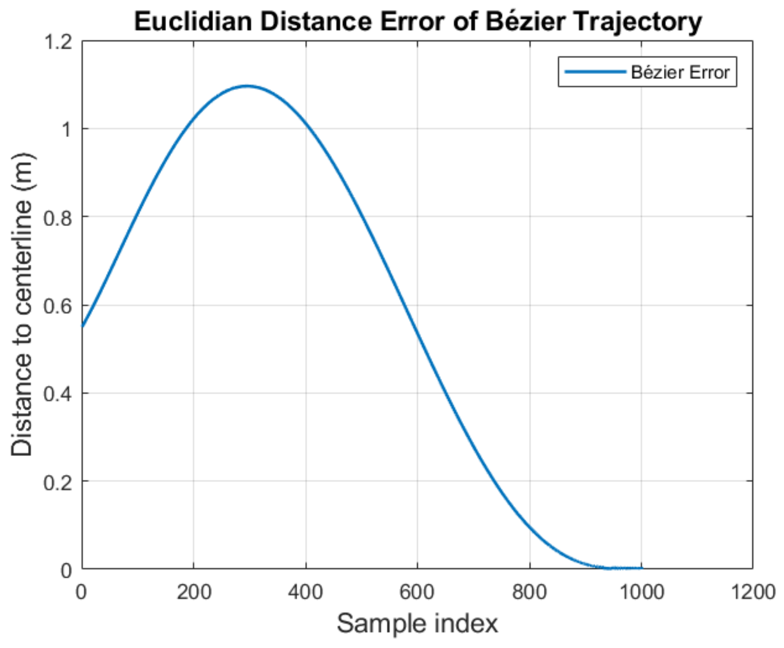

Figure 3.14: Position error over a Bézier trajectory for case 1

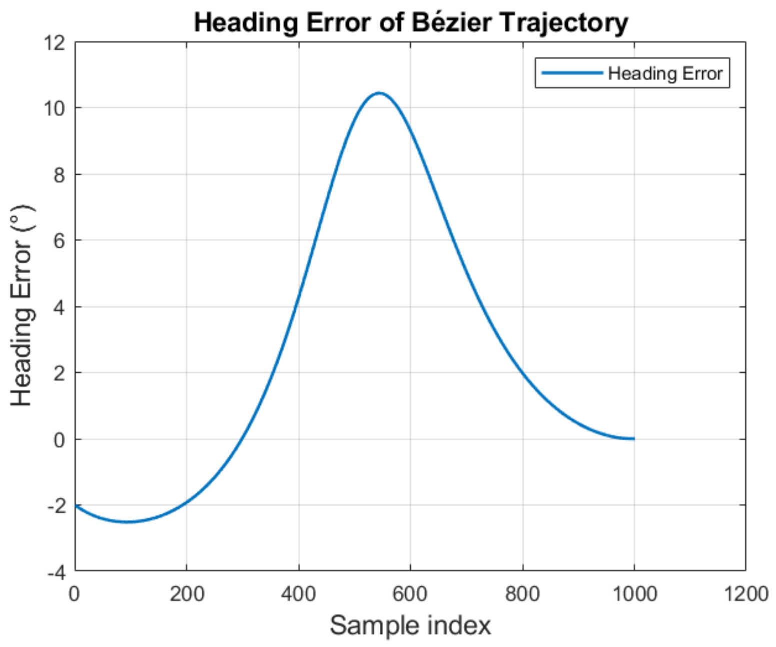

Figure 3.15: Heading error over a Bézier trajectory for case 1

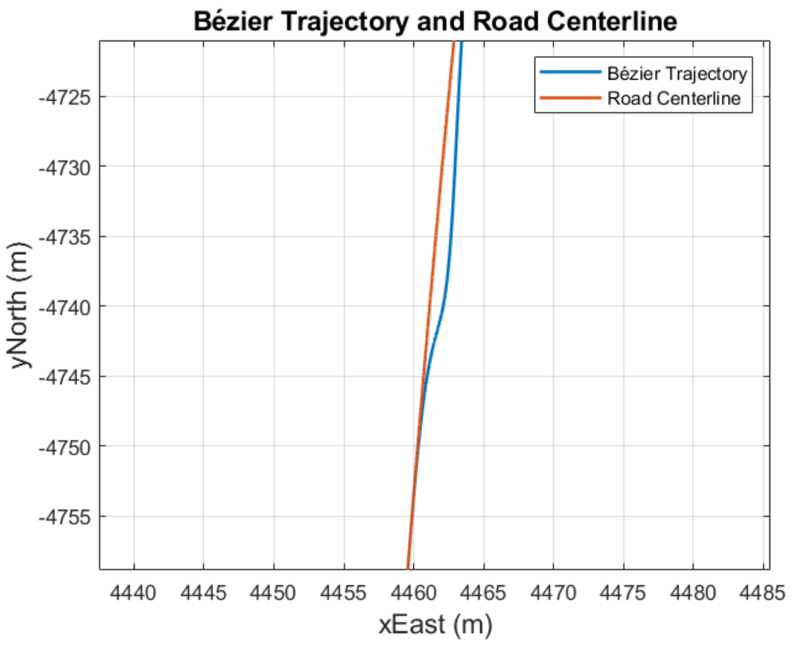

Figure 3.16: Bézier trajectory on Autobahn 38 case 1

CHAPTER 4

CONCLUSION

This thesis has made significant improvements in advancing road modeling and trajectory planning for autonomous vehicles, specifically designed for highway environments. Our research focused on two important aspects: firstly, how to represent the road itself (modeling), and secondly, how to plan the optimal path for the vehicle to follow (trajectory planning).

A novel approach for highway modeling using arc splines was introduced. Arc splines demonstrably align well with highways due to their inherent ability to represent clothoid curves, a fundamental component of highway design. This methodology facilitates the creation of accurate road representations with a minimal set of parameters. The focus on parameter minimization translates to reduced computational complexity, ensuring faster processing times for real-world applications. Additionally, predefined error metrics were implemented to achieve a balance between model accuracy and the number of required parameters. This ensures the generated road model faithfully reflects the actual highway while maintaining computational efficiency.

Beyond parameter minimization, our road modeling approach offers further benefits. The method facilitates the concatenation of multiple, smaller road segments into single, larger ones wherever feasible. This concatenation process reduces the overall number of parameters needed to represent the entire highway, streamlining the representation. Furthermore, our methodology leverages the inherent flexibility of arc splines to generate additional lanes from a single lane definition. This capability proves particularly valuable for scenarios where highways have multiple lanes in each direction. By effectively addressing these aspects, our approach offers a comprehensive and efficient solution for modeling highways in autonomous vehicle applications.

The trajectory planning algorithm we developed seamlessly complements our arc spline-based road modeling method. Similar to the road model, the trajectory planning algorithm utilizes the analytical nature of arc splines. This translates to low computational requirements and reduced execution time, making it well-suited for real-time implementation on autonomous vehicles. Moreover, by aligning both the road model and the trajectory on arc splines, the algorithms are easily integrated. This guarantees a smooth and precise arrival at the desired location, enhancing the overall safety and reliability of autonomous driving.

To provide a more comprehensive analysis and highlight the potential of arc splines, this study also implemented a Bézier curve-based trajectory planning algorithm. The comparative analysis between the two approaches served to illuminate the strengths of using arc splines. By demonstrating the ability to achieve zero error at the destination point and the inherent efficiency of the analytical approach, this thesis establishes arc splines as a strong contender for trajectory planning in autonomous vehicles, particularly for highway environments.

In conclusion, this thesis presents an advancement in the field of autonomous driving technology. The proposed arc spline-based road modeling and trajectory planning algorithms offer efficiency in terms of memory usage and computational power, accuracy, and guaranteed performance. These advancements pave the way for a future where autonomous vehicles can navigate highway environments with greater precision, safety, and reliability. Future research can further explore the integration of this methodology with real-time sensor data for dynamic adaptation to changing road conditions. Additionally, investigating the performance of arc splines in more complex highway scenarios with features like exits, merges, and intersections will be valuable for further refining the approach.

REFERENCES

[1] D. Parekh, N. Poddar, A. Rajpurkar, M. Chahal, N. Kumar, G. P. Joshi, and W. Cho, “A review on autonomous vehicles: Progress, methods and challenges,” vol. 11, no. 14, p. 2162. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[2] A. Jafari, A. Both, D. Singh, L. Gunn, and B. Giles-Corti, “Building the road network for city-scale active transport simulation models,” vol. 114, p. 102398.

[3] J. Godoy, A. Artuñedo, and J. Villagra, “Self-generated OSM-based driving corridors,” vol. PP, pp. 1–1.

[4] X. Li, Z. Sun, A. Kurt, and Q. Zhu, *A sampling-based local trajectory planner for autonomous driving along a reference path.* Journal Abbreviation: IEEE Intelligent Vehicles Symposium, Proceedings Pages: 381 Publication Title: IEEE Intelligent Vehicles Symposium, Proceedings.

[5] E. Zhang, R. Zhang, and N. Masoud, *Predictive Trajectory Planning for Autonomous Vehicles at Intersections Using Reinforcement Learning*.

[6] E. Bertolazzi and M. Frego, “Fast and accurate g1 fitting of clothoid curves,”

[7] M. E. Mortenson, *Mathematics for Computer Graphics Applications*. Industrial Press Inc. Google-Books-ID: YmQy799flPkC.

[8] A. Artuñedo, J. Villagra, and J. Godoy, “Real-time motion planning approach for automated driving in urban environments,” vol. 7, pp. 180039–180053.

Appendix A

PROPOSED ALGORITHMS RESULTS