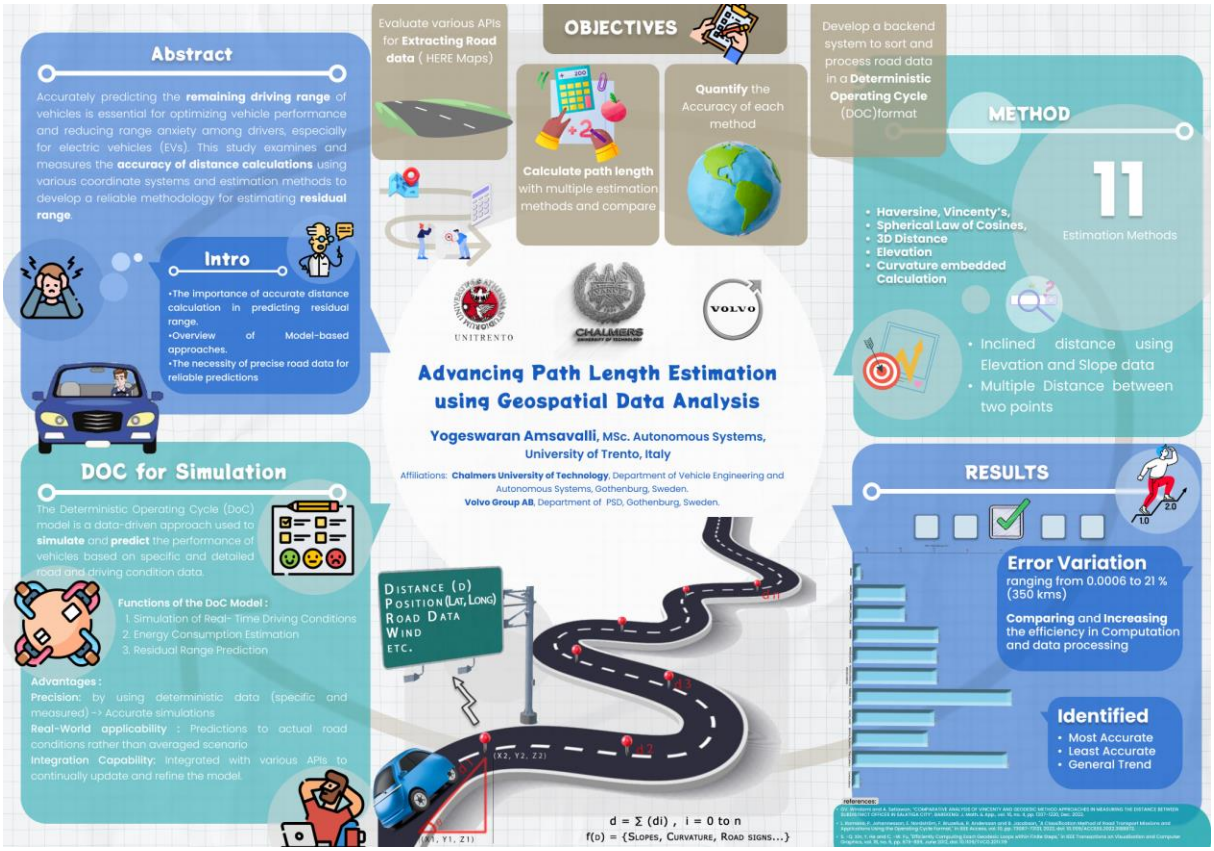
CHALMERS
UNIVERSITY OF TECHNOLOGY



Technical report: Estimation of Accurate Path Length with Geospatial Data Analysis

Department Of Mechanics and Maritime Sciences

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Technical report

Estimation of Accurate Path Length with Geospatial Data Analysis

Yogeswaran Amsavalli, Carl Emvin

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2024

Estimation of Accurate Path Length with Geospatial Data Analysis

Yogeswaran Amsavalli, Carl Emvin

Supervisor: Carl Emvin

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
The cover page was created by Yogeswaran Amsavalli and used during his poster presentation at Sveriges fordonstekniska förening (SVEA).

Estimation of Accurate Path Length with Geospatial Data Analysis

Technical report

Yogeswaran Amsavalli, Carl Emvin
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics
Chalmers University of Technology

Disclaimer

This technical report consists exclusively of an annex (Y. Amsavalli, 'Estimation of Accurate Path Length with Geospatial Data Analysis', 2024.). The annexed master's thesis was supervised by Carl Emvin (Chalmers University of Technology), but examined and published at the University of Trento and Budapest University of Technology and Economics. This technical report serves the purpose of publishing the work that has been conducted at Chalmers.

1 Annex

UNIVERSITÀ DI TRENTO
Dipartimento di Ingegneria Industriale

eit Digital MASTER SCHOOL

MŰEGYETEM 1782

Master's Degree in

Mechatronics Engineering

FINAL DISSERTATION

Estimation of Accurate Path Length with Geospatial Data Analysis

Studying Various Estimation Methods to Compare and Identify the Most Reliable Method

Supervisor UniTrento

Dr. Gastone Pietro Rosati Papini

Supervisor BME

Dr. Kiss Bálint

GASTONE PIETRO ROSATI PAPINI
Università degli Studi di Trento
11.06.2024 14:06:31 GMT+01:00

Student
Yogeswaran Amsavalli, 247000

ACADEMIC YEAR 2023/2024

Contents

Abstract **i**

List of Acronyms **ii**

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Objectives and Goals 6

1.3 Reasearch Questions 7

2 Literature Study **9**

2.1 Distance Measuring Methods in Different Domains 9

2.2 Existing Distance Calculation Methods 10

2.3 Haversine Formula 11

2.4 Vincenty Calculation 14

2.5 Spherical Law of Cosines 16

2.6 3D Distance Formula (Euclidean Distance) 18

2.7 Determinstic Operating Cycle 19

3 Methodology **23**

3.1 Data Collection 23

3.1.1 Leading Mapping APIs 24

3.2 HERE Maps API 25

3.2.1 HERE Maps API: Road Model and Topology 26

3.2.2 HERE Maps Attributes 28

3.3 Implementation 31

3.3.1 Data Preprocessing 31

3.3.2 Data Transformation from Raw to Usable Format 33

3.4 Methods for Precise Distance Calculation 34

3.4.1 Calculating Intermediate distance 34

3.4.2 Introducing Inclined Distance 36

3.5 Overview of Methods . . . 39
3.5.1 List of New Methods . . . 39
3.6 dOC Model Framework . . . 42
3.6.1 Distance as the Independent Variable . . . 42
3.6.2 Attributes of the dOC Model and Usage . . . 43

4 Results and Discussion **46**
4.1 Test Scenarios . . . 46
4.1.1 Selection of Track and Ground Truth Data Presentation . . . 47
4.1.2 Distance Calculations . . . 49
4.1.3 Accuracy and Efficiency Comparison . . . 49
4.1.4 Ideal Option: Balancing Accuracy and Efficiency . . . 51
4.2 Statistical Analysis of Distance Calculation Methods . . . 52
4.3 Experiments . . . 54
4.3.1 Experiment1: Adding Noise to the input . . . 55
4.3.2 Experiment2: Creating a dOC File for a Shorter Distance (Lund to Malmo) . . . 57
4.3.3 Experiment 3: Creating a dOC File for a Longer Distance (Oslo to Bergen) . . . 60

5 Conclusion **62**
5.1 Summary . . . 62
5.2 Limitations . . . 62
5.3 Future Works . . . 63

Acknowledgements **64**

Bibliography **65**

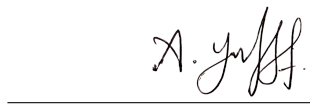
Appendix **68**
A.1 Poster . . . 68
A.2 Working Code . . . 69
A.2.1 Code . . . 70

Student Declaration

I, *Yogeswaran Amsavalli*, the undersigned, hereby declare that the present Master's Thesiswork has been prepared by myself and without any unauthorized help or assistance. Only the specified sources (references, tools, etc.) were used. All parts taken from other sources word by word, or after rephrasing but with identical meanings, were unambiguously identified with explicit reference to the sources utilized.

I authorize the Department of Industrial Engineering of the University of Trento to publish the principal data of the thesis work (author's name, title, abstracts in English and in a second language, year of preparation, supervisor's name, etc.) in a searchable, public, electronic and online database and to publish the full text of the thesis work on the internal network of the university (this may include access by authenticated outside users). I am also aware that in case of false declaration, I could incur in-law penalties and my admission to the final exam could be denied

Budapest, July 2, 2024



Yogeswaran Amsavalli
Student

Abstract

Determining the path length of road segments is crucial for various applications, especially with the advent of electric vehicles (EVs) and smart vehicles developed by companies like Volvo, Tesla, and other OEMs. Accurate path length estimation is essential for optimizing vehicle performance, reducing range anxiety among drivers, and ensuring reliable navigation. This study examines and measures the accuracy of distance calculations using various coordinate systems and estimation methods to develop a reliable methodology for estimating residual range.

The research begins with an evaluation of different methods for distance estimation, including Haversine, Geodesic, and elevation-embedded algorithms. These methods were applied to real-world road data to determine their accuracy and applicability in predicting residual range. Accurate road data, incorporating critical elements such as road slope, curvature, and speed limits, is essential for making reliable predictions.

The core objectives of this study included creating a backend system to process and sort road data, deploying multiple path length estimation methods, and quantifying their accuracy using standard measures. The performance of each method was showcased through a case study, highlighting their effectiveness in different scenarios. Additionally, the study developed a deterministic Operating Cycle (dOC) model to encapsulate road data and facilitate residual range prediction in simulations.

The findings of this study demonstrate the effectiveness of different distance calculation methods and the utility of the dOC model in improving residual range estimation accuracy. By integrating detailed road attributes into distance estimation methodologies, this study enhances the precision and reliability of residual range predictions for smart vehicles, filling a significant research gap and supporting the advancement of sustainable transportation systems.

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAS	Advanced Driver Assistance System
API	Application Programming Interfaces
BEV	Battery Electric Vehicles
ICEV	Internal Combustion Engine Vehicles
CO_2	Carbon - dioxide
CO_4	Methane
dOC	Deterministic Operating Cycle
EV	Electric Vehicles
GHG	Greenhouse Gas
GPS	Global Positioning System
HPX	High Precision Longitude
HPY	High Precision Latitude
HPZ	High Precision Height Coordinates
JSON	JavaScript Object Notation
N_2O	Nitrous Oxide
OC	Operating Cycle
WGS	World Geodetic System

Chapter 1

Introduction

The introduction section outlines the clear background and motivation for the research, leading to the formulation of the research questions. By exploring the challenges and limitations of traditional distance estimation methods, this study aims to enhance accuracy through advanced geospatial data techniques. This foundational understanding sets the stage for addressing critical questions that drive the research forward.

1.1 Background and Motivation

Global Challenges and Transportation's Role

Climate change is today considered one of the most pressing challenges to humankind and nature around the globe. Outcomes have varied from rising temperatures and changed weather patterns to extreme poverty and displacement of people. The major push behind climate change is the accumulation of greenhouse gases in Earth's atmosphere. The primary greenhouse gases (GHGs) include carbon dioxide, CO_2, CH_4, and N_2O, which entraps the heating force and causes global warming. The worldwide movement against the causes of climate change has concentrated on cutting GHG emissions, and international cooperation has constituted a significant framework settled in the Paris Agreement[32].

As was mentioned in the United Nations Climate Change, fossil fuels such as coal, oil, and gas are major contributors to GHGs, amounting to over 75% of global emissions of GHGs and almost 90% of all CO_2 released into the atmosphere. Global warming is accelerating, and every decade since 1980 has been warmer than the last, with the last one being the warmest on record.

Out of these sources, transportation is one primary source of GHG emissions. The transportation sector relies significantly on vehicles based on burning fossil fuels, including, but not limited to, cars, trucks, ships, and planes; this creates a highly abundant emission of CO_2. For an extended period, ICEVs have been among the major contributive factors to emissions arising from transportation. In addition, developments in the global level of transportation networks, the rapid pace of urbanization, and a fast-tracked increase in the demand for mobility have enhanced the sector's environmental impact[24].

Moving to sustainable transport is one of the most important agenda items about both climate change and environmental degradation. The shift should significantly reduce CO2 emissions, improve air quality in cities, and decrease dependence on fossil fuels, hence improving resistance to climate change impacts through a low-carbon and energy-efficient

transportation system. It's this transition toward sustainability that secures our future and helps avoid any of the backlash effects of having a climate-changed Earth.

Transportation's Contribution to CO_2 Emissions

The transport includes different modes of transport varying from road, rail, and air to maritime transport where each mode uniquely demands energy and has greater environmental impacts. According to the International Energy Agency (IEA), the transport sector accounts for one-fifth of global carbon dioxide emission which is 24% if the Carbon dioxide emission is considered from energy. The rapid increase of private automobiles and commercial vehicles has opened the gate widely for the emission of CO_2 from road transport particularly in urban areas where the traffic congestion and the emission collectively worsen the quality of air and the public health[25].

Similarly where aviation and maritime play a vital role in connectivity and global trading, have emerged as a great player in the charts of contribution to CO_2 emissions and other pollutants. The combustion of jet fuel in aeroplanes and marine diesel in ships emit CO_2 and other GHGs into the atmosphere, marking the sector's environmental footprint.

The contribution of the transport sector does not stop here, despite the direct emission from fossil fuel combustion they also generate direct emissions from the supply chain which includes vehicle manufacturing, fuel production and infrastructure development. The holistic approach can also extend to urban planning, consumer behaviour, and public policy.

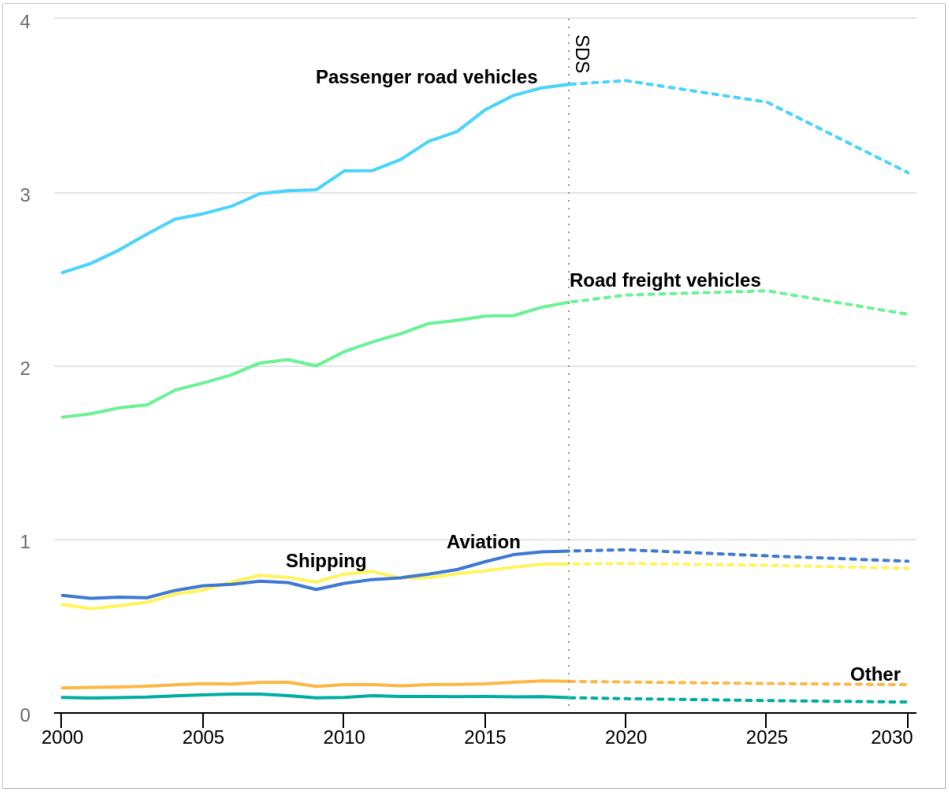


Figure 1.1: Emission of CO_2 in the transport sector by different modes in the Sustainable Development Scenario, 2000-2030 [25].

Sustainable Transportation Solutions

The question that prevailed in talks for a long time is how to reduce the CO_2 emissions from transportation reflecting the urgent need for sustainable mobility approaches that balance economic growth, social equity, and environmental control. The goal can be achieved by collaborative efforts across multiple fronts including:

Advancement in vehicle technology holds much importance in reducing CO2 emissions from the transportation sector. Electric and hybrid vehicles will effectively contribute toward reducing the carbon footprint. Advances in fuel-efficient vehicles integrated with renewable sources of energy will further help lower the consumption of fuel and less emissions. These technologies should be incredibly effective in reducing the transportation sector's impact on the environment and ultimately moving toward a more sustainable future. Vehicle operation through renewable energies would thus support broader efforts toward a low-carbon economy and reduce dependency on fossil fuels.

An immense role is also played by public transit and the development of a strong culture of active transportation. Strong development in public transit infrastructure, a culture of biking and walking, and improvement in shared transport services decrease the dependency on private autos, which helps reduce traffic congestion, hence reducing CO2 emissions, and contributing to better air quality and public health.

The other important issue is the dependence on renewable energy sources. This reliance is to include solar, wind, and hydropower in place of fossil fuels to help reduce carbon emissions. Through this shift, clean energy will be applied in charging electric vehicles and making alternative fuels, thus further reducing the environmental impacts of transportation[1].

The other involves integrating land use and transportation planning in compact, mixed-use, and transit-oriented development. Such an approach reduces the tendency to travel over longer distances. It makes cities more liveable and walkable as well, with reduced reliance on private cars, less traffic congestion, and lower air pollution levels, which in turn will enhance economic growth by fostering more sustainable urban environments.

This essentially means that the transport sector is responsible for improving the global carbon footprint, and the shift to greener solutions in transportation holds large hopes for climate change mitigation. Challenges to estimating the residual range of road vehicles are discussed in the following section, where also the development of a deterministic Operating Cycle (dOC) model is described to achieve better range prediction accuracy.

To conclude the transport sector constitutes a large proportion in terms of carbon footprint and the transition to sustainable transport solutions is needed to ensure the usual behaviour of climate change. In the subsequent section, we will delve in deep in discussing specific challenges in estimating the residual range for road vehicles and explore the development of a deterministic Operating Cycle(dOC) model as a convincing solution to intensify the range prediction accuracy.

Transition to Electric Vehicles

The Automotive Industry across the globe has become very keen on addressing the climate change problems and seeking to reduce carbon emissions by promoting the transition to electric vehicles and hybrid vehicles and producing the same, this is where the transition emerges as a pivotal strategy. EVs will replace traditional ICEVs and promise greater environmental benefits which are powered by rechargeable batteries or fuel cells. This gives

a significant solution being an alternative to fossil fuel powered vehicles thus eliminating the tailpipe emissions and making it independent of the finite resources. According to a study by the Union of Concerned Scientists [21], EVs produce lesser GHG during the whole life period when compared to ICEVs. As said before this leads to lower pollution leading to cleaner air quality and public health, particularly in the urban areas where pollution-related illness exists. The transition to EVs contributes to global support to mitigate climate change and constitutes a low-carbon economy, which will make a remarkable change in the transportation sector.

The trend towards electrification in the automotive industry has seen exponential growth in recent years, which has been driven by the combination of technological improvements, Government policies, and shifting consumer preferences. According to a report by BloombergNEF[2], global sales of EVs have been over 2 million units in 2019, making a major milestone in the adoption worldwide. The government has played an important role in supporting the adoption by implementing policies, incentives, and mandates to promote EV adoption. For example, Countries like Norway and Netherlands have implemented subsidies, tax benefits, and infrastructure investments like setting EV chargers available easily within the radar to encourage people to switch to EVs[22].

Figure 1.2: Image representing that BEV has grown equal to ICEV [11].

Range Estimation and the simulation problem

Residual range estimation is critical for optimizing the efficiency and performance of both battery-electric vehicles (BEVs) and conventional vehicles. It involves predicting the remaining distance a vehicle can travel with its current energy or fuel reserves, considering factors such as driving conditions, speed, load, and environmental conditions. For BEVs, accurate range estimation is essential in alleviating range anxiety—a common concern among consumers about running out of battery charge before reaching a destination. Reliable residual range predictions help drivers plan trips more effectively and make informed refuelling or recharging decisions, enhancing the overall user experience and confidence in electric mobility.

Accurate residual range estimation remains challenging due to the dynamic nature of driving conditions and the multiple variables influencing vehicle performance. Factors

such as road gradients, weather conditions, and terrain types significantly impact energy consumption. For instance, uphill travel requires more energy to counter gravitational forces, while downhill movements can leverage regenerative braking systems to recover energy. Vehicle characteristics, including aerodynamics, weight, and tire friction, also play a role in energy consumption. Advanced simulation and real-time prediction methods are essential for improving range estimation accuracy.

Figure 1.3: Imaginery Representation of Driving Anxiety[6].

To bridge the discussion from the challenges of accurate residual range estimation to the introduction of distance calculation methods, it is essential to highlight the importance of precise distance measurements in enhancing range predictions. The Haversine 2.3 and Vincenty 2.4 formulas are popular distance calculation methods as they provide an effective and simple way to find the shortest distance between any two points on the surface of a sphere. An implementation of the Haversine formula for determining the great-circle distance between two points with the latitude and longitude. This is a well-suited approximation for most applications. The Vincenty formula, based on more accurate ellipsoidal models of the Earth, provides better accuracy than the Haversine formula by, among other tricks, flattening the Earth where spherical one misses.

However, these traditional methods have significant limitations when applied to residual range estimation in automotive contexts. Both the Haversine and Vincenty formulas fail to account for elevation changes and the complex dynamics of real-world driving conditions. This omission can lead to inaccuracies in distance estimation, especially in varied terrains such as hilly or mountainous regions where elevation plays a critical role in energy consumption. For instance, the energy required to drive uphill is significantly higher than on flat terrain, while downhill driving can regenerate energy through braking systems in EVs.

Bridging the gap

Addressing the need for accurate distance estimation is essential for precise simulations and the advancement of sustainable transportation. My research aims to bridge the gaps in traditional distance calculation methods by integrating additional geospatial elements such as elevation and road curvature. By enhancing the Haversine and Vincenty formulas with these elements, my approach offers a more accurate estimation of the path length for

road vehicles. Additionally, my work improves the estimation of the total path length by providing intermediate distances between each point along the path, accounting for the dynamic nature of conditions operating between these points. This allows the distance to serve as an independent variable at each point, enabling the extraction of key information regarding road characteristics, such as variations in elevation, speed limits, slopes, and curvatures. The dOC model is based on this comprehensive information, reflecting the actual operating conditions of travel. The accurate depiction of these conditions directly impacts the range and performance of electric vehicles (EVs). Furthermore, this improved accuracy is critical for developing a dOC model, which is essential for simulating and predicting the residual range of EVs. By refining these estimations, my research contributes to more reliable simulation models, enhancing the practical deployment and user acceptance of EVs, and supporting the broader goal of reducing transportation-related emissions.

Moreover, combining accurate distance estimation with vehicle odometry can provide valuable insights into the wheel radius along the path. This integration allows for an even more detailed understanding of vehicle dynamics and road interactions, further enhancing the precision of simulations and predictions. The innovative methods developed in my study address immediate challenges in current distance estimation techniques, making them more applicable and valuable in the context of modern, sustainable transportation solutions.

1.2 Objectives and Goals

Objectives:

The primary objective of this research is to develop and evaluate a comprehensive model for accurately calculating the path length of vehicles using various distance measurement methods. This research aims to enhance sustainable transportation by providing precise distance estimations for both internal combustion engine vehicles (ICEVs) and battery electric vehicles (BEVs).

Goals:

1. Investigate Distance Measurement Methods

This study investigates both traditional and modern distance measuring techniques, including the Haversine and Vincenty formulas, by comparing their performance in terms of accuracy and computational efficiency to reveal the most effective methods.

2. Develop the DOC Model

A highly reliable operational model will be developed, and it is referred to as the deterministic Operational Cycle (dOC) model. This model will have many distance computation settings and will be flexible to any kind of vehicle so that it remains upgradeable to any of the future needs.

3. Evaluate Applicability:

To check the applicability of the model in real-life situations, the geospatial data obtained from the HERE Maps API will be used. This should be done by computing the total sum

of intermediate distances of all points received to check the total length of the path of vehicles and, hence, the model's efficiency.

4. Facilitate Sustainable Transportation:

The study will enable the sustainability of transportation in that the study results will offer the tools for accurate route planning, residual range estimation, and, as such, the design of transport networks that are efficient and environmentally friendly.

Additional Considerations:

The particular distance points of the links are added to the model to provide a more accurate prediction under different scenarios. These are the variations in vehicle load concerning energy and speed, safety for roundabouts or when a traffic light turns red, early or late starts from uphill/downhill affecting the management of energy, and gearshift strategies not depending on some positions for avoiding mismanagement.

Manual Implementation and Flexibility:

The different methods to calculate distance will be implemented using pure Python so that it's easily interpreted and flexible for further changes. In this way, it does not rely on the libraries of specific providers so it can be generally usable and easily adaptable to many systems.

This research will serve as a foundational step towards creating a cleaner, greener, and more sustainable transportation infrastructure. By addressing the pressing need for accurate distance measurement in route planning and vehicle range estimation, it aims to contribute significantly to the advancement of sustainable transportation solutions.

1.3 Reasearch Questions

In the context of ongoing efforts to address environmental issues and climate change, this thesis aims to contribute to the field of sustainable transportation by achieving several key objectives. First, it explores various distance measurement methods and implements conversions among different units of length measurements. Second, it develops and validates a deterministic Operating Cycle (dOC) model to enhance the accuracy of distance estimations. Third, it analyzes the suitability of the dOC model for both Internal Combustion Engine Vehicles (ICEVs) and Battery Electric Vehicles (BEVs). Through these efforts, the study seeks to provide practical solutions to improve the accuracy of distance estimation methods and support the development of more efficient and environmentally friendly transportation systems.

In this new emerging era for transport, this research is a roadmap to lead us down a greener and more sustainable path for future generations. This thesis has been undertaken to provide insights into the evolution of safe, sustainable, and greener transport systems that will help mitigate the negative impacts of climate change.

Towards These objectives, Three Research questions provide the main guidance for this study:

1. How is the path length of a road accurately estimated?
2. How do different methods for path length estimation perform for short and long paths?
3. What are the common factors that lead to large errors in the various estimation methods?

The first base question is the main that guiding the remaining questions in this study. It focuses on investigating the various methods for distance measures and the routes these methods can be used for path length estimation.

Based on the above questions, the research investigates the various methods for the estimation of large path lengths by providing the correct perception and the best adaption of these methods for the estimation of long paths. Besides, the study proposes the leading factors that lead to large errors in the various estimation methods to provide the most accurate assessment for these methods by identifying the boundaries of each method.

Chapter 2

Literature Study

2.1 Distance Measuring Methods in Different Domains

Different methods of distance measurement have been invented and used for transportation by water, air, and road. Every method has some challenges that need to be overcome to ensure smooth and efficient transportation.

When it comes to calculating distances in water transportation, unique challenges arise due to the absence of fixed markers and the dynamic nature of the maritime environment. As we sail over water, we face different problems for distance measurement. Firstly, there are no fixed markers on which we can rely to measure distance and secondly, while the land moves along its own axis, in the maritime environment, things move. To address these challenges, maritime navigation systems rely on a diverse strategy of technological measurement, encompassing three specific technologies: Global Navigation Satellite Systems (GNSS), radar and sonar.GNSS which is now widely used assigns the position of vessels through satellites. Their most common implementation, the Global Positioning System or GPS, navigates using distance and direction calculations. Receivers on the vessel calculate the latitude, longitude, and altitude triangulating from several satellites in orbit[16]. The Haversine formula, accounting for Earth's curvature, calculates distance for an interval of two points on its surface.

In air transportation, measurement of distance is needed for navigation, air traffic management and flight planning. Distance measurement is mainly conducted by radar and Global Positioning System(GPS).In Aviation Radar Systems, aviation radar emits radio waves that return to the radar after bouncing off the aircraft. Air traffic controllers use these signals to determine how far away aircraft are, even when the aircraft are invisible to the eye. This facilitates the separation of aircraft in controlled airspace. The aircraft Global Positioning System (GPS) system is based on the principle of receiving signals from satellites and provides the aircraft's exact coordinates (latitude, longitude, altitude, and groundspeed), which is an accurate way of navigation for a distance measurement purpose in the sector of air transport[15]. The distance of two points in terms of air transport is normally calculated based on the spherical law of cosines or Vincenty's formulae due to the curve of the Earth.

Accurate distance measurement plays an important role in operation in application in transportation by road. An accurate distance measurement is largely needed in the reliability of road transportation like route planning, navigation, and other vehicle operations on roads. A commonly used method to measure distance in road transportation is the Global Positioning System known as GPS. The vehicle's position, speed and direction of

travel along the journey are calculated from signals received by an onboard GPS (global positioning system) receiver from GPS satellites in orbit[20]. The distance travelled is calculated by integrating the average vehicle speed over time. The vehicle's location is updated at frequent and short intervals by the GPS receiver. Distance calculations between two GPS-determined locations on the surface of the Earth are usually calculated using the Haversine formula or Vincenty's formulae.

2.2 Existing Distance Calculation Methods

There are different kinds of distance calculations that have historically been used in development. This includes methods that use simple, two-dimensional formulae like the Haversine formula, as well as more advanced methods that leverage multi-dimensional and 3D calculations using matrices, Euler angles and elevation data. The goal is to compare the strengths, weaknesses, and decimal accuracy of these approaches so that we can determine the most correct and most efficient methods for use in geospatial applications.

These can be further categorized in to methods based on a spherical model of the Earth and those based on the viewing of the Earth as an ellipsoid. We'll look at the purpose, pros and cons of each of these models, and also look at the existing methods within each category. We can see the formulae used in this research further in this section[19].

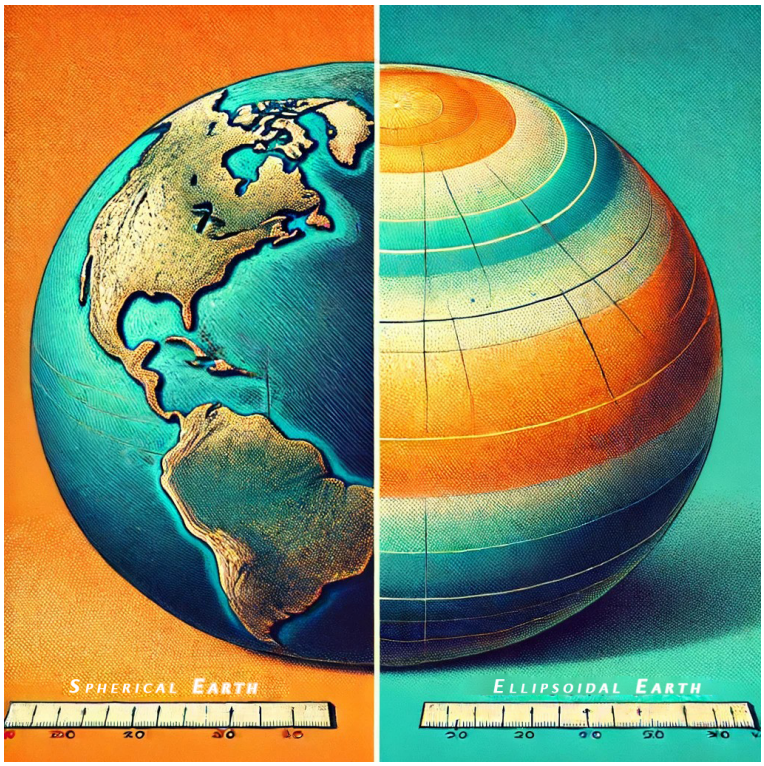


Figure 2.1: Imaginery Representation of Spherical vs Ellipsoidal Earth [8].

Spherical Model - Formulae and Purpose

The spherical model assumes a perfect sphere to represent the Earth. All the distance calculations are simple and very easy to implement, and they use formulae like the Haver-

sine formula or the Spherical Law of Cosines. The key benefit of the spherical model is that it is straightforward. The formulae involved are elementary, so they require less computation; calculations are done relatively quickly and are, therefore, acceptable for applications in need of fast estimations of distances. It is precisely for these reasons that the spherical model finds wide use and acceptance for small-area applications, with requirements for precision not of an extremely high order, such as simple navigation and a few other analyses involving applications in geospatial technology but not requiring exact results.

However, the spherical model also has its limitations. The main restriction occurs at large distances or near the poles. This happens by the inaccurate representation that the Earth is an oblate spheroid and not a sphere since an oblate spheroid is slightly flattened at the poles and bulging at the equator. Thus, using the spherical model to represent this will no doubt lead to errors when calculating the distances. The model is also rudimentary in that it does not factor in the various elevations on the surface of the Earth. These limitations make the spherical model inappropriate for demanding applications requiring high accuracy and precision, like advanced geospatial analysis or aviation and maritime navigation[15]. Nonetheless, the spherical model is still a good tool for more accessible, less demanding applications because it is pretty intuitive and computationally effective.

Ellipsoidal Model - Formula and Purpose

The ellipsoidal model considers the flattening at the poles and bulging at the equator, which gives more exact distance measurements than the spherical model. Vincenty's formula and the Geodesic formula are the main formulae of this model. The significant advantage of the ellipsoidal model is accuracy. It has been proven to have increased precision over long distances and over different latitudes, hence its use in tasks that need very high precision[35]. This model finds its particular importance in geospatial applications, aviation, and maritime navigation, where the distance between two points is highly affected by the actual shape of the Earth. The ellipsoidal model is significantly superior in terms of precision because it considers the shape and flattening of the Earth. This would be core to the precision capabilities in high-precision geospatial applications because the ellipsoidal model captures the variations in the curvature of the Earth, which the spherical model ignores. Such considerations make sure that when calculating the distance over vast distances and within differing geographical locations, the results for distance calculation are as accurate as can be.

However, there are some disadvantages to using the ellipsoidal model. The formulae applied are, in this case, Vincenty's and the Geodesic formulae, more complicated than those under the spherical model; hence, more computation-intensive. This generally becomes time-consuming and may sometimes be a disadvantage in some applications where speed is of the essence[29]. However, some of these limitations are usually outweighed by the added advantages attached to the increased accuracy and precision of the ellipsoidal model in scenarios where these measurements are critical.

2.3 Haversine Formula

The Haversine formula is a mathematical function that uses square roots, cosines, and inverse cosines to find the shortest distance between two points on the surface of a sphere – the great circle distance – based on the spherical law of cosines. It is numerically stable,

and computationally efficient, especially for small distances, while preserving the correct angle involved. 'Haversine' stems from the usual name of the function's value, 'haversin' which is short for half of the versed sine ($\frac{1-\cos(\theta)}{2}$).

Explanation of Great-Circle Distance

The great circle distance between any two points on the surface of a sphere is the shortest distance between them. This sort of path around the sphere, which lies on a circle of points that bisects the sphere into two equal halves, is called a 'great circle'. This concept is important for navigation, including geospatial analysis, because it gives the most efficient path from one place to another on the Earth, which can be approximated as a sphere.

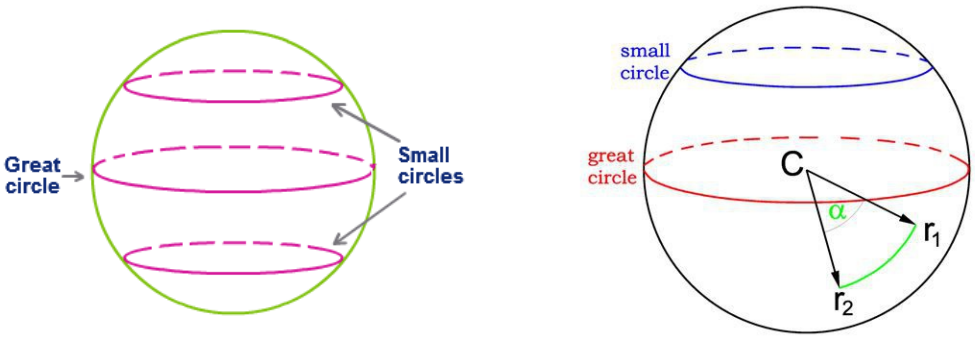


Figure 2.2: Illustration of Great-Circle Distance.

To understand how the Haversine formula calculates this distance, let's start with the concept of spherical trigonometry. The great-circle distance can be computed using the central angle between the two points. Given two points on a sphere, the central angle θ between them can be found using the spherical law of cosines 2.15. However, spherical trigonometry can be complex and prone to errors, especially over small distances. To simplify the calculations and minimize errors, the Haversine formula is used. The Haversine formula leverages trigonometric identities to compute the distance between two points on the surface of a sphere. It is particularly effective because it avoids the inaccuracies that can arise from using spherical trigonometric formulas over small distances.[33]. The Haversine formula is derived from the spherical law of cosines. By applying the Haversine function to the spherical law of cosines, we can simplify the calculation and reduce errors, especially for small distances.

The spherical law of cosines is given by 2.15:

$$\cos(c) = \cos(a)\cos(b) + \sin(a)\sin(b)\cos(C),$$

for two points on a sphere with latitude and longitude coordinates (ϕ_1, λ_1) and (ϕ_2, λ_2), the formula can be adapted as:

$$\cos(d) = \sin(\phi_1)\sin(\phi_2) + \cos(\phi_1)\cos(\phi_2)\cos(\Delta\lambda),$$

However, this formula can be problematic for small distances due to floating-point precision errors[33]. By applying the Haversine function, we transform it into a more stable form:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right),$$

The Haversine formula then becomes:

$$d = 2r \cdot \arcsin\left(\sqrt{a}\right),$$

The whole formula is expressed as,

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\left(\varphi_1\right)\cos\left(\varphi_2\right)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right), \tag{2.1}$$

where:

- d is the distance between two points
- r is the Earth's radius (mean radius = 6371 km)
- $\Delta\varphi$ is the difference in latitude between the two points (lat2 - lat1)
- $\Delta\lambda$ is the difference in longitude between the two points (lon2 - lon1)
- φ_1 and φ_2 are the latitudes of the two points in radians

This formula 2.1 calculates the distance by first converting the latitude and longitude of the two points from degrees to radians. It then computes the haversine of the central angle between the points and finally applies the arcsine function to find the central angle itself, which is then multiplied by the Earth's radius to get the distance[27].

Applications and Limitations of Haversine's Formula

Haversine's formula is highly applied across many transportation sectors because it can result in correct measurements of the distances between two points on the Earth's surface. In the case of aviation, this formula is applied to enable the planning of the path of a flight such that routes are the shortest and most efficient; hence, it is essential for fuel efficiency and time management on board an airplane. The Haversine formula is applied to the description of the track over the open sea when making a route chart for ships to be able to follow the shortest course between ports. This is quite important for optimum time and fuel considerations while traveling. Road transportation is used by navigation systems for approximate road distance estimation and route planning; it provides a fast and practical way of computing distances between locations for logistical planning or personal travel.

Despite the extensive use of the Haversine formula, there are some practical limitations to it in specific applications. First and foremost is the assumption of sphericity. The formula assumes that the Earth is a perfect sphere, while it is an oblate spheroid. This means that minor errors in distance calculation are also added for great distances and distances very close to the poles. Neglecting elevation is another significant drawback. Since elevation may take a significantly different value, especially in hilly areas, without considering such changes, this inaccuracy is bound to take place with the measurement of distance. Lastly, the curves and junctions of a simplified road network are not taken care of in this formula.

In practice, the roads do not go straight, and their bends are numerous, so the use of this formula does not obtain such distances. These limitations imply that even though the Haversine formula is useful for introductory approximations and general calculations, more accurate methods might have to be sought after for applications demanding high precision[17].

2.4 Vincenty Calculation

Vincenty's formulae are more recent than the Haversine and much more accurate. They were devised by Thaddeus Vincenty of the US Naval Oceanographic Office in 1975, considerably after the advent of geographical information systems (GIS) that could represent the Earth with any reasonable approximation. Where the Haversine's calculations assume a perfectly round sphere, Vincenty's formulae assume that the Earth more closely resembles an ellipsoid – accurate and research-tested to 10cm of accuracy within 20 degrees of the magnetic poles. Given a longer distance, Vincenty's formulae might be required. For this part of the world, Google Maps interprets Ella's route in terms of its precise meridian and parallels – running along a gradient of latitude and longitude.

Vincenty's formulae include the direct and inverse methods for geodesic calculations on the ellipsoid. The direct method determines the endpoint given a starting point (latitude φ_1, longitude λ_1), initial bearing (α_1) and distance (s). It involves converting coordinates to radians, computing auxiliary values like the reduced latitude (U), iteratively solving for the endpoint's latitude and longitude, and finally calculating the endpoint's bearing (α_2). This method is particularly useful in navigation and surveying.

The inverse method, used in our application, finds the distance and bearings between two known points (latitudes φ_1, φ_2 and longitudes λ_1, λ_2). It begins by calculating the reduced latitudes (U1, U2) and initializing the longitude difference (λ).Through iterative calculations, it determines the spherical distance(σ) and azimuthal angles(α_1, α_2), finally providing the ellipsoidal distance(s). This method is ideal for precise distance and direction calculations, crucial for geospatial analysis in vehicle path length computation. The steps for the calculation in the thesis context can be well understood with the figure 2.3

The Vincenty Formula is given by,
Notation

- φ_1, φ_2: Latitude of point 1 and point 2(in radians)
- L: Difference in longitude between point 1 and point 2(in radians)
- a: Length of semi-major axis of the ellipsoid(radius at the equator)
- f: Flattening of the ellipsoid
- b: $(1 - f) \times a$ length of semi-minor axis of the ellipsoid (radius at the poles)

The latitudes φ_1 and φ_2 are reduced to:

$$U_1 = \arctan((1 - f) \times \tan(\varphi_1)), U_2 = \arctan((1 - f) \times \tan(\varphi_2)), \tag{2.2}$$

which are the transformed latitude values that accounts for Earth's flattening. Calculate U_1, U_2, and L, and set the initial value of $\lambda = L$. Then evaluate the equation iteratively until the λ converges. The iterative evaluation begins,

$$\sin\sigma = \sqrt{(\cos U_2 \sin\lambda)^2 + (\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos\lambda)^2}, \tag{2.3}$$

$$\cos\sigma = \sin U_1 \sin U_2 + \cos U_1 \cos U_2 \cos\lambda, \tag{2.4}$$

$$\sigma = \arcsin(\sin\sigma), \tag{2.5}$$

σ is not evaluated directly from $\sin(\sigma)$ or $\cos(\sigma)$ to preserve the numerical accuracy near the poles and the equator,

$$\sin\alpha = \frac{\cos U_1 \cos U_2 \sin\lambda}{\sin\sigma}, \tag{2.6}$$

$$\cos^2\alpha = 1 - \sin^2\alpha, \tag{2.7}$$

$$\cos(2\sigma_m) = \cos\sigma - \frac{2\sin U_1 \sin U_2}{\cos^2\alpha} = \cos\sigma - \frac{2\sin U_1 \sin U_2}{1 - \sin^2\alpha}, \tag{2.8}$$

$$C = \frac{f}{16}\cos^2\alpha[4 + f(4 - 3\cos^2\alpha)], \tag{2.9}$$

$$\lambda = L + (1 - C)f \sin\alpha\{\sigma + C\sin\sigma[\cos(2\sigma_m) + C\cos\sigma(-1 + 2\cos^2(2\sigma_m))]\}, \tag{2.10}$$

when in 2.10 the λ converged to the desired degree of accuracy(10^{-12} corresponds approximately 0.6mm), now calculate the geodeic distance with the following,

$$u^2 = \cos^2(\alpha)\frac{a^2 - b^2}{b^2}, \tag{2.11}$$

$$A = 1 + \frac{u^2}{16384}\left(4096 + u^2\left(-768 + u^2\left(320 - 175u^2\right)\right)\right), \tag{2.12}$$

$$B = \frac{u^2}{1024}\left(256 + u^2\left(-128 + u^2\left(74 - 47u^2\right)\right)\right), \tag{2.13}$$

$$\begin{aligned}\Delta\sigma = B\sin(\sigma)\Bigg(&\cos(2\sigma_m) \\ &+ \frac{1}{4}B\Bigg(\cos(\sigma)\left(-1 + 2\cos^2(2\sigma_m)\right) \\ &- \frac{B}{6}\cos(2\sigma_m)\left(-3 + 4\sin^2(\sigma)\right)\left(-3 + 4\cos^2(2\sigma_m)\right)\Bigg)\Bigg),\end{aligned}$$

$$s = bA\left(\sigma - \Delta\sigma\right). \tag{2.14}$$

The value of 2.14 represents the ellipsoidal distance between two points on the earth's surface, accounting for the earth's flattening. This method is particularly advantageous for applications requiring high precision over long distances.

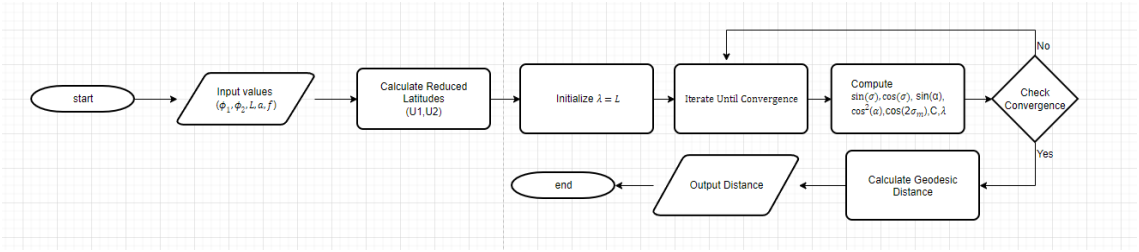


Figure 2.3: Flowchart explaining Vincenty Formula.

Applications and Limitations of Vincenty's Formula

The Vincenty formula can be used in a wide range of applications because of its precision in calculating distances on an ellipsoidal model of the Earth. In navigation and surveying, it is essential to ascertain the exact distance and bearing between two points on the Earth's surface, which forms a basic need for correct navigation and land survey projects. The Vincenty formula is generally used in geographic information systems for spatial data analysis within geospatial analysis, given that it can be applied to get accurate measurements within the computation of vehicle path length and other geospatial applications. Further, this formula is used in mapping and charting to provide the required accuracy for detailed mapping and charting, especially for large-scale and long-distance uses that require precision.

Despite its advantages, the Vincenty formula suffers a few limitations and limits its practicality for certain applications. Some of the serious ones are that the formula is computationally expensive since Vincenty's is an iterative formula that requires much more computational power than simpler formulas like the Haversine formula in less-than-global applications. Additionally, Vincenty's formula fails to converge or give accurate results near the poles, as well as when the points are nearly antipodal. It poses those challenges in those exceptional cases. In addition, the formula assumes that local deviations from a smooth ellipsoid may be neglected, which could introduce approximations that would cause errors for highly localized geospatial analyses[34].

2.5 Spherical Law of Cosines

This mathematical formula allows you to calculate the distance between two points on the surface of a sphere. With the Spherical Law of Cosines, one has to remember both your trigonometry and your spherical geometry to understand how to use the angular distances between points: the Spherical Law of Cosines uses properties of spherical geometry and is defined when the sides of the triangle are arcs of great circles.

Spherical Law of Cosines Formula in Detail

Consider a unit sphere 2.4 with points (u, v, and w), angles (A, B, and C), and distances (a, b, and c).

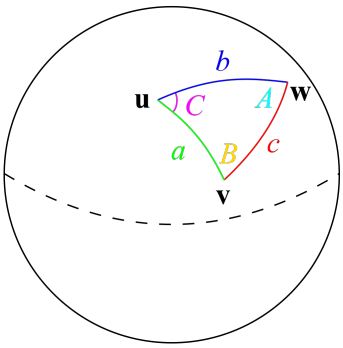


Figure 2.4: Spherical triangle solved by the law of cosines. [5].

The Spherical Law of Cosines then states that the great-circle distance c,

$$\cos c = \cos a \cos b + \sin a \sin b \cos C, \tag{2.15}$$

is a function of the latitude coordinates (φ_1 and φ_2) in the points u and w, the longitude coordinate difference $\Delta\lambda$ between the points u and w, and the central angle $\Delta\sigma$. The distance on a non-unit sphere can easily be shown to be proportional to the distance of the unit sphere and is thus on the earth:

$$\mathrm{d} = \mathrm{R}\Delta\sigma, \tag{2.16}$$

where R is the earth's radius and d is the distance between two points.

Applications and Limitations of Spherical Law of Cosines

The Spherical Law of Cosines is a fundamental formula; it has been used in many branches for computing distances and angles on the sphere. In geodesy, it is employed to carry out accurate distance computations between points on the surface of the Earth—a significant part of mapping, land surveying, and geographic information systems (GIS). This formula is also extensively applied in astronomy to calculate angular distances between celestial bodies that help determine the positions and movements of stars, planets, and other astronomical objects. Its versatility and simplicity make it a valuable tool in terrestrial and astronomical applications.

The Spherical Law of Cosines has good features for distance calculation, but a few drawbacks also. A significant con is associated with accuracy for minimal distances; numerical precision problems may arise because of using the arccosine function, resulting in errors in the computed distances. Besides, the formula assumes that Earth is a perfect sphere, which is incorrect because Earth is an oblate spheroid. The assumption slightly introduces inaccuracies of calculated distances, especially over long distances, where the curvature of the Earth and deviations in shape become pronounced. However, in any way, with these limitations considered, the Spherical Law of Cosines is widely used in applications where the preliminary approximations of distances need to be taken into account, and high precision is not so critically required[10].

2.6 3D Distance Formula (Euclidean Distance)

The 3D distance formula, also called Euclidean distance in three dimensions, measures the straight-line distance between two points in latitude, longitude and elevation (altitude). It includes the vertical dimension of the Earth's surface as part of the distance calculation, which yields a more accurate measure of the actual distance between two points on the Earth's surface.

3D Distance Formula in Detail

The 3D distance d between two points in three-dimensional space can be determined using the Euclidean distance formula. If we denote the two points as (lat_1, lon_1, h_1) and (lat_2, lon_2, h_2), the formula is given by:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (h_2 + h_1)^2}, [18] \tag{2.17}$$

where x and y are the cartesian coordinates corresponding to the latitude and longitude, and h represents the elevation. The latitude and longitude of both points from degrees are converted to radians, and then we can convert the spherical coordinates to Cartesian coordinates using,

$$x1 = R \times \cos(\varphi_1).\cos(\lambda_1), y1 = R \times \cos\varphi_1.\sin\lambda_1,$$

$$x2 = R \times \cos(\varphi_2).\cos(\lambda_2), y2 = R \times \cos\varphi_2.\sin\lambda_2,$$

the elevation h1 and h2 are already in the correct form and do not require conversion. Now, we can apply the 3D distance formula 2.17.

Applications and Limitations of 3D Distance Formula

The 3D Distance Formula, or Euclidean Distance, has a wide range of applications across various fields due to its ability to measure straight-line distances in three-dimensional space. In geospatial analysis, it is particularly valuable for accurate distance measurements in terrains with significant elevation changes, providing crucial data for mapping and geographic information systems (GIS). In aviation and aerospace, the formula is used to determine the direct path between two points, including altitude variations, which is essential for flight path planning and navigation. Engineering and construction projects also benefit from this formula, especially when planning and building structures in hilly or mountainous areas where elevation changes must be accurately accounted for. Additionally, in telecommunications, the 3D Distance Formula helps in assessing line-of-sight distances and optimizing signal paths that involve elevation differences, which is critical for ensuring effective communication links[3].

Despite its usefulness, the 3D Distance Formula has several limitations that can affect its accuracy and applicability in certain scenarios. One major limitation is its simplicity and the assumption that the Earth is a perfect sphere. This can introduce minor inaccuracies since the Earth is actually an oblate spheroid. More accurate models that account for the Earth's ellipsoidal shape can provide better precision for certain applications. Another significant limitation is the accuracy of the elevation data used in the calculations. The precision of the distance measurement heavily depends on the quality of this data; inaccu-

rate or low-resolution elevation data can lead to significant errors, particularly in regions with complex terrain. Furthermore, the 3D Distance Formula calculates the straight-line (Euclidean) distance between two points, which may not represent the actual path taken, especially in areas with physical obstacles like buildings, trees, or mountains. This can limit its practical application in environments where a clear line of sight is not available.

Despite its usefulness, the 3D Distance Formula has several limitations that can affect its accuracy and applicability in certain scenarios. A significant limitation is the accuracy of the elevation data used in the calculations. The precision of the distance measurement heavily depends on the quality of this data; inaccurate or low-resolution elevation data can lead to significant errors, particularly in regions with complex terrain. Furthermore, the 3D Distance Formula calculates the straight-line (Euclidean) distance between two points, which may not represent the actual path taken, especially in areas with physical obstacles like buildings, trees, or mountains. This can limit its practical application in environments where a clear line of sight is not available.

2.7 Determinstic Operating Cycle

Before going into detail, it is needed to grasp the basics of the deterministic operating cycle (dOC) representation. Often with current techniques to depict a driving cycle, strong limitations on potential applications exist when comparing different vehicles, resolving decelerations to match a reference trajectory, and modelling external factors' interference in driving behaviour.

Another problem with 'classical' descriptions of driving cycles is that they are directed towards the reference vehicle, causing them to be 'pathological'. In other words, the speed profile and the reference vehicle are 'causally' dependent on each other and, more significantly, this makes the speed profile itself lose any real meaning. If we want to record the driving cycle, we are recording not only the speed profile of the vehicle but also the traffic and the wind conditions. Thus, all those factors are implicitly integrated but difficult to disentangle and examine.

The OC representation is a promising way to respond to these challenges, based on the format invented and used by Pettersson. Unlike driving cycles, the OC format doesn't require a hypothetical speed profile but completely avoids the assumption of a predefined speed profile. Instead of hard-coding this information into the transport scenario beforehand, the OC model separates the characteristics of the transport mission and the external world. There are three distinct levels of representation in the OC format, each with a slightly different purpose and level of abstraction. Before looking at these levels of representation, we will define the transport application, transport operation and transport mission. The transport application is the high-level purpose of the vehicle from a life-cycle perspective, defining the context for the specifications. The transport operation is a sequence of quantifiable tasks over a travelled path, and the transport mission brings operational requirements together with context. These definitions highlight the sequential nature of transport operations and the corresponding need for a contextual, integrated representation.

To handle the already unresolved classification problem of transport missions as well as the inherent variations in the OC format, the category of representations is supplemented by three levels: the level of bird's-eye view that summarises changes in an OC at the phase level, and two operational levels – stochastic operating cycle (sOC) and deterministic operating cycle (dOC) – that sum up an OC in terms of well-defined metrics for

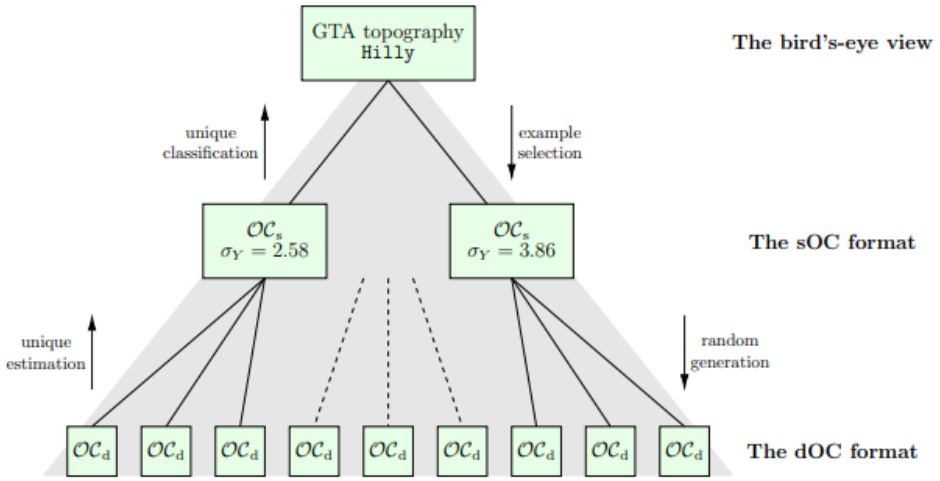


Figure 2.5: Schematic representation of the pyramidal structure of an OC. [26].

classification of transportation missions into respective categories. This way, most importantly, the complexity of a transportation mission that is too complex for a human eye to grasp or analyse becomes treatable and comparable. The OC paves the way for making complex problems more manageable and comparable, especially those involving extensive service and demand resources. However, even though the OC format represents significant potential, it can still be improved – in particular, by incorporating models of weather, traffic, and mission parameters in probabilistic terms. Moreover, testing synergies among the three levels of representation is important for demonstrating that the OC can be effectively adapted to the optimal vehicle selection processes and vehicle design procedures. This state of the art can be taken as ample motivation for the thrust of the present thesis, namely bridging the gaps identified above[26].

The figure 2.5 schematically represents the pyramidal structure of an OC, using the topography parameter as an example. All the missions of the same type, which is a GTA class, are part of the same transport, application (the bird's-eye view), The individual statistical properties may however differ within the transport application (sOC). Finally, transport missions can be statistically equivalent but significantly different in practice. This is captured by the dOC representation.

dOC - an Overview

Specifically in connection with this research, the dOC model represents the most granular, detailed and deterministic level of abstraction within the OC framework. The dOC model is made to represent transport missions in the most accurate, deterministic manner possible, as deterministic as one can go without actually simulating every single product feature associated with a given transport mission that is of interest.

The dOC model can be used to provide fine-grained analysis and simulations of transport missions. Unlike stochastic models, the dOC model is built upon deterministic principles and, as such, every aspect of the transport mission is represented in detail. This permits the modelling of the behaviour of the vehicle in a particular set of circumstances with

complete precision, enabling the perception of the interdependencies among the variables that influence a transport mission.

Moreover, this theory seeks to leverage the specificity of the dOC model with respect to the other levels of representation (the bird's-eye view and the sOC) of OC: the bird's-eye view can classify and articulate transport applications at their most general level, while the sOC captures variability through elementary statistical tools. Integrating the three levels thus allows us to transform the weaknesses of one representation into the strengths of another.

Crucially, the dOC model has also proved to be useful for designing and making more efficient and environmentally friendly vehicles. The ability of the dOC model to provide quantitative and deterministic information about any ground transport mission can help inform decisions about vehicle type and specification, the amount of energy it should require and the emissions it is likely to produce.

The dOC modelling is a central point of this thesis, which is developed to optimise and formalise this Operating Cycle (OC) framework however needed to improve the understanding and optimisation of transport operations. This modelling and analysis has the potential to make the OC representation much more useful in transport applications, e.g. to develop novel concepts in transport engineering and vehicle design and to help in the optimisation of the transport system[26].

Terrain Classification According to Volvo GTA System

In the context of this thesis study, the classification of terrain into categories such as flat or hilly follows the criteria set by the Volvo GTA system. This system provides a standardized approach for categorizing the terrain based on the grade of slopes encountered during driving. Understanding and using these classifications is crucial for accurately modelling the dOC (deterministic Operating Cycle) and for making reliable predictions about vehicle performance and residual range.

The Volvo GTA system classifies terrain into four distinct categories based on the slope grades encountered over the driving distance:

1. **FLAT (Level I)**: This category is defined when slopes with a grade of less than 3% occur during more than 98% of the driving distance. This classification indicates very mild terrain variations, suitable for simulations and analyses where minimal elevation change is assumed.

2. **P-FLAT (Level II)**: This category applies when slopes with a grade of less than 6% are present during more than 98% of the driving distance. P-FLAT represents moderately flat terrain with some gentle slopes, making it relevant for scenarios where slight elevation changes may impact vehicle performance.

3. **HILLY (Level III)**: In this classification, slopes with a grade of less than 9% occur during more than 98% of the driving distance. HILLY terrain involves more significant elevation changes, which are critical for understanding vehicle dynamics in varied landscapes, such as those found in suburban or rural areas.

4. **V-HILLY (Level IV)**: This category is used when none of the above criteria are fulfilled. V-HILLY terrain includes the most varied and challenging landscapes with slopes exceeding 9% over significant portions of the driving distance. This classifica-

tion is essential for modelling vehicle behaviour in highly undulating or mountainous regions.

The Volvo GTA system's detailed classification allows for a nuanced understanding of how different terrains impact vehicle performance and energy consumption. By applying these categories in the dOC model, the study ensures that simulations are more representative of real-world driving conditions, leading to more accurate predictions of residual range and overall vehicle efficiency[26].

Incorporating these classifications into the dOC model helps to refine the analysis and provides a clearer framework for interpreting the results of the distance calculation methods. It underscores the importance of considering terrain variability in simulations and highlights the adaptability of the dOC model to different driving environments.

Chapter 3

Methodology

This section provides a comprehensive overview of the methodology employed in this thesis. The methodology encompasses various steps and techniques used to gather, analyze, and interpret geospatial data. By outlining each stage of the process, we aim to provide a clear understanding of how the research was conducted and the rationale behind the chosen methods.

3.1 Data Collection

Accessing data through APIs has become one of the most essential supports for modern research and development across multiple fields. It is the researchers, developers, and businesses of today who need to plug in and derive relevant information from many sources that will serve to enrich their projects and applications accordingly. In analyzing geospatial information and the computation of distances for road vehicles, an application programming interface is essential in deriving several reasons.

Value and Importance of Use: APIs in Geospatial Analysis

APIs play a key role in geospatial analysis, making the work significantly easier. By providing access to geospatial data, APIs eliminate the need for manual data collection, thereby saving a considerable amount of effort and time. This is especially beneficial when obtaining geospatial data is just one part of a larger project, allowing professionals to focus on other critical aspects of their work. Having access to wide coverage and constantly updated information on maps that includes road networks, points of interest and even real-time updates on traffic. Continuously updating data helps to ensure that users are working with the most current information, which is crucial for accurate distance measurements and optimal route planning. API guarantees that new changes, inclusions of road layouts, closures, and new constructions can be factored in because maintaining accuracy and relevance is crucial for geospatial analysis.

Besides primary map data, APIs also have other functionalities like geocoding, reverse geocoding, and elevation data. Geocoding translates addresses to geographic coordinates—that is, it is a process for translating place names into a latitudinal/longitudinal view that can be plotted on a map. Reverse geocoding, on the other hand, translates degrees of latitude and longitude back into human-readable locations or places, giving details about locations. Elevation data enhances understanding of the topography of an area

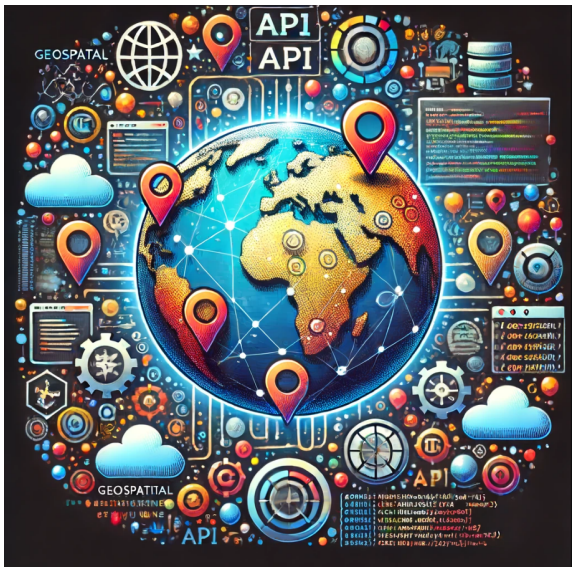


Figure 3.1: Geospatial API cartoon representation[7].

crucial for applications where such changes might affect travel time and fuel consumption, such as logistics and transportation. These extra capabilities further increase the analytic power that can aid the researcher or decision-maker in navigating through and interpreting complex spatial data landscapes with accuracy. APIs from well-respected providers assure that the data used are with integrity, consistency, and compliance with industry standards. This assurance reduces the chances of errors and inaccuracies in distance calculations, increasing the accuracy and general reliability of geospatial analyses.

The data from reliable APIs is already standardized, and as such, it becomes easy to have similar or comparable results for different users and applications. This forms a critical input when conducting studies over a wide scale and when running collaborative projects. Different people applying the same data could churn out standard and comparable results.

Nowadays, APIs have become versatile and more straightforward to approach relevant tools in geospatial research and mapping applications. APIs cater to a wide range of needs, from simple distance calculations to complex route optimization and spatial analysis. User-friendly interfaces further simplify the process of integrating geospatial data into applications, potentially enabling people with limited technical background knowledge to unleash valuable, sophisticated mapping tools. Making advanced geospatial tools widely accessible encourages innovation and expands the range of possible applications.

3.1.1 Leading Mapping APIs

There are several leading mapping APIs with robust features, vast coverage, and ease of use, among which are:

The Google Maps API is known as the most adequately covered and strategized detailed map data. It includes a wide range of services that include everything from geocoding to distance matrix calculation and also real-time traffic information. Developers and researchers worldwide prefer this API because it has a lot of documentation and community support.

The HERE Maps API offers top-quality mapping data and services primarily serving the automotive and transport industries. It comes packed with advanced features, for instance,

turn-by-turn navigation, current traffic information, and road network data that is rich in detail. HERE Maps is also appreciated for its professionalism and accuracy.

Bing Maps API features a rich set of mapping tools with aerial imagery, road views, and traffic data. It is known for its highly interactive user interface and capability to be integrated with other Microsoft services, so any business working inside the Microsoft ecosystem will definitely have it.

People prefer the customization capabilities and high-performance mapping services of the Mapbox API. This allows interactive mapping, providing even the most detailed geographic information, such as terrain and satellite imagery. In general, the applications for which Mapbox is utilized chiefly require very custom maps made visually attractive.

These APIs are famous for their powerful functionalities, comprehensive coverage, and accessible user interfaces. These functionalities appeal to professions engaging in geospatial research and mapping applications.

3.2 HERE Maps API

One of the most critical aspects that contributed to HERE Maps API's choice for this study is that it focuses totally on road data. Although other mapping APIs from Google and Bing and even from Mapbox are usually accompanied by a whole variety of elements of infrastructure, such as buildings, landmarks, or points of interest, in most cases, their road data can be less accurate. With the diversity in these APIs, there could be objects about highways, bicycle paths, pathways for pedestrians, railway tracks, or a side emphasis on the network of roads. Conversely, the HERE Maps API is specialized in a focused area of road data to have more prosperous, more detailed sets of road-related information compared to its counterparts.

The HERE Maps API emphasizes the provision, maintenance, and enrichment of detailed road data, ensuring that a comprehensive and up-to-date dataset is received regarding roads, connectivity, and routes. This is most helpful for geospatial analytic applications in need of accurate measurements of distance and routes, which correlate with my research on modeling the traveled distances of road vehicles. The road data from the HERE Maps API comes with access to extensive information that details features such as road geometries, traffic conditions, speed limits, road hierarchy, and turn restrictions[12].

Reason for choosing HERE Maps API

Among all the available APIs, HERE Maps was chosen for this study. The following section elaborates on the reasons for the Quality and coverage of road data and the much-improved Accuracy and detail behind this choice.

The HERE Maps API does not pass road data as a mere derivative feature but, instead, its core feature. Such a focus on the road ensures that high-quality, fine-granular, up-to-date road information is available from the API. The detailed road network data includes highways, bridges, local streets, rural roads, and many other categories of roads. This pertains to covering intricate details of the road network for distance and routing calculation; even small roads play an important role.

The HERE Maps API is an essential system for increasing the quality, detail, and accuracy of road data. The dedicated attention to road data allows HERE to produce maps with the highest level of detail and accuracy. In this thesis study, detailed information is

crucial due to the importance of precise distance calculations and measurement accuracy for the research. The capability of HERE Maps to offer real-time traffic information, road closures, and construction updates further enhances its utility by enabling dynamic and precise distance modelling.

Also, it provides with advanced functionalities and here are some advanced functionalities with the HERE Maps API that cannot be done without geospatial analysis. These include turn-by-turn directions, real-time traffic information, and advanced routing algorithms that consider factors such as traffic conditions, types of roads, and prohibition restrictions on certain roads. With these features, robust and precise models of geospatial studies can be done, which further help in estimating the exact path lengths of road vehicles. Industrially, HERE Technologies has a well-known delivery of top-notch data integrity and data consistency while adhering to industry norms. This reputation assures that the data supplied through the HERE Maps API is reliable and high-quality. One would like to use it in a research project where precision and dependability are of concern. That guarantees assurance over data accuracy and consistency. Moreover, HERE Maps API is chosen because it is a trusted supplier and adds confidence not only in its data but also in the results derived from that data.The user-friendly UI of the HERE Maps API and its elaborate documentation should make it easy to integrate with various applications. So, ease of use is essential in developing custom tools and models for geospatial analysis. Flexible to work in integration with other software and systems of the same research. This helps smoothly run information processing and analysis of data.

All the above reasons convinced me to use HERE maps API for my thesis research. This is because it is unbeatable to concentrate on-road data, thus providing adequate, accurate, and updated information vital in precise distance estimation and route modelling. Another critical consideration that supported the purpose of the research in integrating the API is that it contains advanced functionalities, high data quality, and can be integrated easily. Using HERE Maps API, this research can bring more accuracy and reliability while modelling distance travelled by road vehicles, which adds value within the field of geospatial analysis.

3.2.1 HERE Maps API: Road Model and Topology

Let's delve into the road topology method that HERE Maps follows to gain a better understanding of its intricacies and benefits.

Logical Topology of Roads

The domain of a digital road map is effectively a topology—a description of the network of roads and intersections. HERE's Road Model uses a logical topology in which roads (referred to as 'links') are represented as linear segments between intersections (called 'nodes'), with nodes in 2D space. Links take off and land and intersect at nodes and are treated as if they are straight. This idealization reduces the actual curvature roads take in the real world to a concept representation; it scribes the network but does not capture the exact geometry of roads.

Representing Curved Roads with Shape Points

To capture the alignment of roads, the Road Model uses 'shape points' placed along links to create connected line segments forming polylines. These shape points are an essential

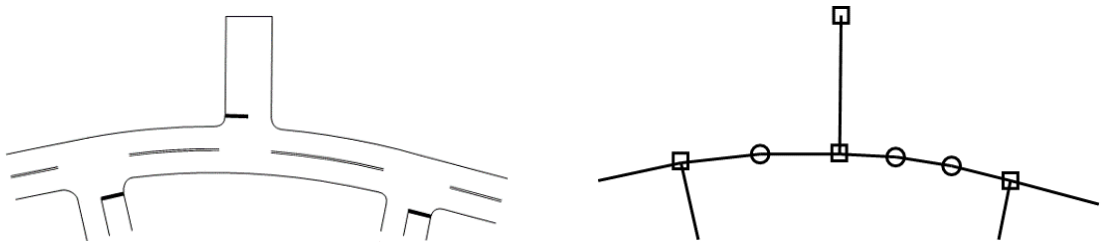


Figure 3.2: Road in reality vs. Logical topology and geometry[12].

element, for they interconnect the geometry of the lines to define the road's curved shape. Topologically, the road is still a single link because the 'road type' has not been altered even when shape points are added. This way, it becomes possible to digitize logically the representation of roads and, at the same time, their natural dimension of curved paths expressed as 3.2 clearly shows the ideology.

Enriching Road Data with More Attributes

The Road Model can adapt to the addition of multiple attributes, such as speed limits and travel directions, to enrich the content of the map. For instance, adding geolocated content, such as road signs, further makes the granularity of the map more useful. If the attributes need to be varied along a link, say a difference in speed limit, then 'bivalent nodes' are placed. These segments lie within the interior of the links under consideration and directly connect two, so attributes can vary without creating another junction point.

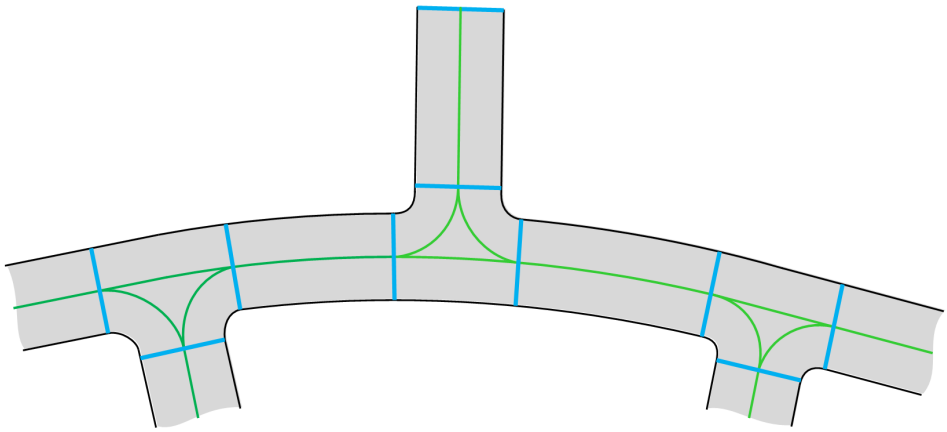


Figure 3.3: Advanced Topology using NURBS[12].

Enhanced Models: Topology Segments and Lane Modeling

Enhanced models for the road model are so advanced to introduce ' topology segments' and add detail to the individual lanes. There exist segments that require no bivalent nodes, as parameters can change along a subsection of any road. The Lane Model then enhances the geometric representation using NURBS (non-uniform rational B-splines), which are mathematical splines in 3D space. NURBS represent 3D curvature and elevation changes within the road topology precisely.

Rich Model Set

1. **Road Model**: The Road Model is topologically foundational and defines the basic topology and road attributes by which the detailed and accurate representation of roads is ensured, thereby laying down the ability to calculate distances precisely and to plan routes.

2. **Lane Model**: The Lane Model is more granular in that it models individual lanes and 3D curves and elevations using NURBS. This fine-tuning is necessary for applications requiring exacting data at the lane level as seen in 3.3

3. **Localization Model**: The Localization Model augments the map with objects necessary to realize the level of localization required, including signs and other roadside features, and is very important for advanced navigation and autonomous driving applications[12].

3.2.2 HERE Maps Attributes

As HERE Maps offers a comprehensive set of attributes, this section will delve into the main attributes used for distance calculation as well as the additional attributes that enhance the overall functionality of the platform.

Advanced Driver Assistance System(ADAS) Attributes

Several ADAS attributes are embodied in the HERE Maps API that enables more accurate distance calculations and drives geospatial analyses down to higher degrees of precision. Reporting complete and real-time information on road features and conditions is made possible through ADAS attributes, thus advancing the sensory capability of road vehicles to make the surroundings and environment observable for driving and navigation. Using the HERE Maps API to call out for ADAS attributes can undoubtedly help in distance estimation, as complete transportation information can be availed. Being each of the values returned by the HERE Maps API relative to previous values, only the first value will be absolute. This approach is highly effective because it stores less data to represent road geometry, thus saving memory and improving performance in return. It stores only the differences between consecutive points instead of full coordinates, so the amount of data stays at a minimum while no details are lost. This relative positioning especially helps in large data sets related to geospatial, where memory efficiency and processing speed are the two most critical factors. Below explains the various ADAS attributes and below is why these attributes are essential .

The ADAS attributes provide a detailed description of road geometry, which is highly important for the accuracy of distance estimations. The ADAS attributes offer high-precision coordinates, road slopes, headings, and curvatures, among others, that, when collected, will give the effect of much better road modeling[14].

Coordinate Attributes

The **Link ID** attribute is a permanent and unique identifier for each road segment, ensuring consistent identification across different releases.

The attributes **HPX, HPY, HPZ** refer to High Precision Longitude (HPX), Latitude (HPY), and Height Coordinates (HPZ) along the link. These coordinates are offsets from

the previous point, providing very high precision in the spatial positioning of the road segment, which is essential for accurate distance measurements.

Slope and Heading

The **Slopes** provide the vertical road direction at coordinate points, measured relative to the reference node. This slope information is crucial for calculating the actual travel distance, especially through undulating terrain.

The **Headings** provide the horizontal road heading at coordinate points, measured from the reference node. This heading information is useful in determining the travel direction and the overall geometric course of the road.

Curvature and Road Geometry

The **Curvature** at coordinate points relative to the reference node. Curvatures help estimate reasonably how much the road is bending, which is essential in correctly estimating path length, mainly for winding or curved roads.
The **Vertical Flags** at coordinate point positions along the link, indicating straight road sections based on changes in height. This helps to know the straight length and which sections are slightly curved or inclined.

Other Road Attributes

The **Reference and Non-reference Curveheads** refers to the Curvature at reference and nonreference nodes at the link. These attributes describe the curvature in the link's start and end location. So that the actual road geometry is well-accommodated using this information.
The **Topology ID** is a Unique ID for topology segments presenting the intersection-to-intersection connectivity of the network. This helps in understanding the overall connectivity of the road network, which is important for route planning.
The **Start and End Offset** indicate the relative start and end positions of the link in the topology, which helps in mapping the exact segments of the road used in the distance computation.

	Attribute	Description
0	LINK_ID	Permanent link ID, uniquely identifying roads across map releases.
1	HPX	High precision longitude coordinates, relative to previous points.
2	HPY	High precision latitude coordinates, relative to previous points.
3	HPZ	High precision height coordinates, with special handling for unknown values.
4	SLOPES	Vertical road direction at coordinate points, relative to the reference node.
5	HEADINGS	Horizontal road heading at coordinate points, relative to the reference node.
6	CURVATURES	Road curvature at coordinate points, relative to the reference node.
7	VERTICAL_FLAGS	Indicates presence on straight road sections based on height changes.
8	REFNODE_LINKCURVHEADS	Curvatures at the reference node of the link.
9	NREFNODE_LINKCURVHEADS	Curvatures at the non-reference node of the link.
10	BUA_ROAD	Identifies links inside built-up areas.
11	BUA_ROAD_VERIFIED	Verification status of built-up area roads.
12	LINK_ACCURACY	Indicates ADAS compliance of the link's geometry.
13	TOPOLOGY_ID	Unique ID for topology segments.
14	START_OFFSET	Relative start position of the link in the topology.
15	END_OFFSET	Relative end position of the link in the topology.

Figure 3.4: ADAS Attributes Table.

Integration of Additional Attributes

While the Advanced Driver Assistance System (ADAS) layer attributes supplied by the HERE Maps API are crucial for correct distance computations, we must add some other relevant attributes to complete the geospatial analysis. Attributes such as speed limits, weather conditions, and traffic conditions are all sourced from different layers in the HERE Maps API and add a lot to the estimates of the distance in terms of accuracy and reliability. In any case, the attribute layer for ADAS is very vital for distance calculation in that it provides information at a detailed coordinate level.

Speed Limits

Speed limits aid us in determining the allowable driving speeds in the various road segments. This data is retrieved from the suitable API layers to assist in the modeling of actual travel times, and route planning is optimized. Equally, the addition of speed limits integrated into the distance estimated process will go a long way in helping the model reflect average speed variations on road types and regions, arriving at practical and accurate simulation.

Traffic Conditions

Real-time traffic data is another crucial element for accurate distance estimation and simulation. Traffic conditions, including Traffic signals, stop signs, and pedestrian crossings, can cause significant deviations from expected travel times. By integrating real-time traffic information from the HERE Maps API, geospatial analyses can dynamically adjust routes and distance calculations based on current traffic conditions, ensuring that the most efficient and realistic paths are considered.

Weather Conditions

Weather conditions may influence the driving conditions and travel time, given precipitation, wind velocity, and direction. Access to live weather information from the HERE weather API layer facilitates appropriate distance estimation model changes to represent adverse weather's influence on travel times. It will be helpful considering, for example, the presence of heavy rain or snow, road traffic may go down speed, resulting in an increased travel time that necessitates re-calculation of estimated distances and travel time to retain such estimates correct with different weather scenarios.

Importance of ADAS Attributes in Distance Calculation

Although the other extra properties also apply, the ADAS attribute layer forms the most critical data layer while calculating distances mainly because it contains the primary coordinate data. ADAS attributes constitute high-precision data of longitudes, latitudes, and heights used to define road geometry districts. These attributes are relative to previous values, of which the first value is absolute and, hence, allows for efficient data representation and memory use—essential for extensive geospatial data. To facilitate any distance calculation model, such as those based on ADAS attributes that have captured detailed road geometries—including slopes, headings, and curvatures—backbones are formed. Combining speed limits, weather conditions, and traffic data with such precision could not be

Figure 3.5: Imaginery Representation of a car containing ADAS Layer data[9].

enough to calculate accurate distances without accurate coordinate information. Other layers, therefore, enrich the analysis and provide a majorly rich context within which distance can be calculated, with the ADAS attributes acting as a skeletal structure for attaining the correct spatial context, whose geospatial process maintains top high fidelity.

3.3 Implementation

Here, we describe how we have implemented the HERE Maps API to accurately estimate path length. It contains, in more or less detail, the detailed procedures of making the API call, the nature of information retrieved, and how this information is translated and used in the application to get the length of the path. The implementation phase of this thesis involves several critical steps, from accessing the HERE Maps API to transforming the retrieved data into a usable format for distance estimation. This section provides an overview of these steps, laying the groundwork for more detailed explanations to follow.

3.3.1 Data Preprocessing

Data preprocessing is a critical step before performing any operations, as unprocessed data can lead to inaccurate results. This section elaborates on the methods used for data collection and the preprocessing techniques applied in this research to ensure the data is clean, accurate, and ready for analysis.

API Call

For the purpose of this thesis, we utilize the HERE Maps API to gather essential data for accurate distance estimation and the creation of a dOC model. The specific API endpoint

we employ is designed to match routes and retrieve a comprehensive set of attributes necessary for our calculations. The API call used is as follows:

```
https://routematching.hereapi.com/v8/match/routelinks?apikey=API_KEY
&mode=fastest;truck;traffic:disabled&routeMatch=1
&attributes=ADAS_ATTRIB_FCn(*)&attributes=APPLICABLE_SPEED_LIMIT(*)
&attributes=TRAFFIC_SIGN_FCn(*)&attributes=TRAFFIC_PATTERN_FCn(*)
&attributes=ARCHIVED_WEATHER(*)
```

Listing 3.1: An example API call with placeholder API key

You need an API key to use the HERE Maps API. This key is tied to your account, and it's what provides you with access to HERE's full range of mapping and geospatial services. You can obtain an API key by signing up for a HERE developer account[13]. The POST method of making the API call allows you to submit a GPS trace and return route data that matches. The endpoint will receive a response, and process that by structuring the data and returning the data according to the structure.

Parameters

- **apikey**: This is your API key that you got from HERE.
- **mode**: If you request the fastest truck-optimal route, this should be "fastest; truck; traffic, and you would like to switch off the traffic-related information. According to the application, the traffic function can be on or off.
- **routeMatch**: Set to 1 to ensure that the API matches the GPS trace given as input in the body of the API call. This parameter helps in filtering out unreliable GPS coordinates, such as those outside the road or mistakenly placed in buildings.
- **attributes**: This parameter specifies the various attribute layers to be included in the response. For our thesis, we include the attributes which we already mentioned 3.2.2, 3.2.2.

The API call 3.1 is structured to ensure that all necessary data is retrieved efficiently.

```
https://routematching.hereapi.com/v8/match/routelinks
```

Listing 3.2: URL Endpoint for matching GPS

The 3.2 is responsible for matching the provided GPS trace to the road network and returns the related data. The output of the API call is often in JSON format which is structured and easy to parse. This request will introduce the following features: ADAS data, speed limit, traffic signs, traffic patterns, and weather information. The JSON response has been saved for post-processing. This could be using any number of tools and approaches, from direct calls to Python code to API-testing tools such as Postman. In this thesis, we used Python's ' **requests** ' library to make the API call and handle the response.

Handling Missing and Inconsistent data

Although we all want to work with perfect data, the case is not so in practice. While HERE Maps supplies good quality data, there might be some missing values in the road links; most of the time, coordinates are available consistently, but elevation, slopes, and curvature values may be missing occasionally. When such gaps exist, data-cleaning processes are initiated to rectify the inconsistencies.

For missing elevation, slope, or curvature values, we have proximity-based imputation, referring to the data from the links nearby. After this process, there shall be logical and spatial consistency in filling the missing data points. For instance, if a location does not have an actual speed value, we can refer to the nearest link and use this value. Missing fields are then accurately filled up based on such neighbour values; this guarantees the completeness of the dataset. Second, it is also necessary, therefore, that all fields in the dataset be uniformly fulfilled in length, such that now that we have created a CSV for it, we want to analyze them. This allows these further data to be incorporated and subsequently processed in the latter stages of the analytical pipeline; doing so avoids a lack of uniformity that might make further data integration and processing impossible.

Data Integration from Different layers

To obtain a complete dataset, derived datasets from all the required layers have to be combined. In the context of HERE Maps, for example, this can be several attributes related to the same road link but from different layers or the inclusion of ADAS attributes, speed limits, traffic patterns, and weather data. We will merge these layers based on linkID. Such a task may involve creating a dictionary where all keys are associated with a link whose values are lists corresponding to attributes and defining that particular link as explained in 3.2.1.

For other cases, we would envisage additional or changed attributes with values that may be constant along the length of a link. Others will be variable; this can be illustrated using examples like a speed limit and a collection of coordinates. We will duplicate constant values to the count of varying lists to standardize the length across all the attribute lists for each link. For example, where the speed limit is constant along a link, we would repeat that value to match the number of coordinate points within any given link. Similarly, attributes such as the presence of traffic signals, which might be the case only once within a link, would have all instances marked with True at that position and False otherwise. All the attributes would thus have the same length, which significantly simplifies subsequent processing and analysis of the data.

This outcome of the data integration approach ensures that the dataset will be well structured in advance for easier reference and utilization of link values during the creation of the dOC file. It also means that with such, it will be much easier to check that all the attributes are equal in length—which is an integral part of data integrity.

3.3.2 Data Transformation from Raw to Usable Format

Data in its raw format from the HERE Maps API generally comes in JSON. A transformation series will be needed to have this in a format that is viable actually to conduct the distance analysis and computation. These should thus involve various steps to ensure the data is adequately prepared for an exact geospatial computation.

Relative to Absolute Data Values

Most of the data values, such as latitude, longitude, elevation, slopes, and curvatures, are expressed in the changes between a value and the previous point, unlike absolute values. Every first value in each link is absolute; the subsequent ones are relative to the last point. The given relative value figure has to be brought to an absolute-type figure

through automated calculation. For this, this step will include the relative changes because it will add and return actual measurements of each point along the link.

The coordinate measurements of latitude, longitude, and other relative parameters, if any, such as elevation, slopes, and curvature, require calculating their cumulative sum to add each relative value to the sum of all previous values. This somewhat converts the series of incremental changes into a complete set of absolute values and retains total accuracy at the same time when representing paths geographically.

Scaling for Calculation

The retrieved data are expressed in units that may not be directly useful to the calculations. For example, latitudes and longitudes are in 10^{-7} degrees WGS84, height in centimetres above WGS84, slope in 10^{-3} degrees, heading in 10^{-3} degrees, and curvatures in $10^{-6}m^{-1}$. These values must be scaled appropriately to make calculations more efficient and the results easier to read. For example, scaling these measurements into more easily handled units, latitude and longitude into degrees, elevation in meters, and perhaps the slopes and curvatures into intuitive units.

Data Normalization

To ensure consistency and to allow reasonable calculations, all data from the HERE Maps API is either converted or normalized to units in the International System of Units. Latitudes and longitudes are converted to degrees. Elevations that are provided in centimetres above WGS84 are converted to meters. Additionally, the speed limits provided in the data are converted from kilometres per hour to meters per second and other attributes to their respective SI units. These values are normalized to SI units so that they are in universally acceptable formats, and geospatial analysis, along with distance calculations, can be as accurate and reliable as possible in this research.

3.4 Methods for Precise Distance Calculation

This section will explain the various methods for precise distance calculation, detailing how each method operates and the scenarios in which they are most effective.

3.4.1 Calculating Intermediate distance

With most conventional mapping systems, the input coordinates are likely to return the total length of the path. While this is useful, inherently, the process lacks a degree of precision for more complex or fine-level detail geospatial analysis. Most advanced systems such as HERE Maps, does one step better, giving the total length and link lengths, but there is still some inaccuracy within links. However, it differs from this study because it further extends its accuracy by estimating the intermediate link point-to-point distances. By so doing, it ensures that the distance measurements have a much higher resolution and, therefore, significantly increases the accuracy of the entire path length measurement.

The measurement of intermediate distances within each of the links involves dividing each of the links and measuring the distance between every pair of consecutive points. This method takes the actual path travelled into much more detail for road geometric features, elevation differences, and curvature changes than the general calculation can account for.

Benefits of Calculating Intermediate Distances

Calculating intermediate distances between points within links offers several key benefits, especially in the context of residual range estimation and vehicle simulation. One of the primary advantages is the enhancement of accuracy in residual range predictions. By calculating distances between intermediate points within links, the representation of the actual path taken by a vehicle is captured with greater precision. This ensures that small geometrical features, such as minor bends, curves, and elevation changes, are considered. As a result, the sum-path length estimate is much more accurate than the standard link-based or overall path length, leading to more reliable residual range predictions.

Accurate intermediate distances are crucial for simulation models used to predict a vehicle's residual range. By incorporating detailed distance measurements, these models can better account for the varying road conditions and driving patterns that affect energy consumption. This level of detail allows for more precise monitoring of vehicle components and more accurate predictions of when maintenance is needed based on the actual distance traveled under various conditions. Consequently, vehicle performance simulations become more reliable, enhancing the practical deployment and user acceptance of electric vehicles (EVs).

Furthermore, calculating accurate intermediate distances enables the determination of the wheel radius along the path. This is particularly beneficial for electric vehicles, where precise knowledge of wheel radius variations can improve the accuracy of energy consumption models. By accounting for changes in wheel radius, simulations can more accurately reflect the real-world performance of the vehicle, leading to better predictions of residual range.

In simulations, using accurate intermediate distances as the basis for the dOC model ensures that the behavior of the vehicle is modeled with greater fidelity. This is crucial for testing new vehicle technologies, safety systems, and performance enhancements in a controlled environment. Accurate distance measurements enable more realistic simulations, which are essential for evaluating the impact of different driving conditions on the vehicle's residual range.

Calculation of Methods

The strategy has been implemented using the HERE Maps API, which is called to return all the points or data along the vehicle's path. This calculates the distances between these intermediate points that give a better measurement of the actual path length. The steps are as follows,

1. **Fetch data**: Detailed coordinates and attributes were fetched from the HERE Maps API.
2. **Compute Distance**: We compute the distances of each link between subsequent points as explained in 3.4.2
3. **Sum up the distances**: Sum these intermediate distances to get the total path length.

This is a much more accurate intermediate distance calculation than the old way. In some cases, the very hilly routes winding routes straight lines or link-based approach will return results in the wrong.

The intermediate distance calculation is indeed a considerable enhancement for analysis in geography and distance measurement purposes. Focusing on the details across links and turning points between links further improves the quality of path length assessment and, thus, reliability. It increases the precision and quality of the route for both planning and real-time navigation, as well as the quality of the routes being analyzed. Hence, this is one of the essential contributions of this thesis that emanates a final robust solution toward attaining higher levels of accuracy in distance estimation with geospatial data.

3.4.2 Introducing Inclined Distance

When we check how such distances are calculated, most methods use the straight-line distance between points. For example, HERE Maps states in the documentation that the length of a link is calculated as straight lines between consecutive shape points, regardless of whether it goes over several tiles[12]. The method does not include splines, other smoothing techniques, or geodesic computations. Errors in the estimated path length, therefore, could potentially be because, in practice, the distance covered by the vehicle is very much more complex than that defined by the measurement between two points in a map, especially if significant changes in altitude take place along the path. For this, we introduce the concept of 'inclined distance', which includes elevation data to give a more realistic measure of the vehicle's drive. Further, if the curvature data is available, we will use this to find the arc length of the path for extra accuracy.

The methodology for calculating inclined distances involves a combination of traditional surface distance calculations and the incorporation of elevation and curvature data. By using established methods such as the Haversine, Vincenty, spherical law of cosines, and 3D distance formulas, we ensure robust and accurate surface distance measurements. These measurements are then enhanced by embedding elevation and curvature data, providing a true representation of the actual path length travelled by a vehicle. The detailed procedures and calculations for determining inclined distances are discussed in the sections that follow.

Inclined Distance Concept

This section outlines the methodology for calculating inclined distances, as detailed in Section 3.4.2.

Inclined Distance with Elevation Data:

Theorem 1 (Pythagoras theorem). In a right-angled triangle, the square of the length of the hypotenuse (c) is equal to the sum of the squares of the lengths of the other two sides (a and b)[30]. Mathematically, this is expressed as:

$$c^2 = a^2 + b^2$$

.

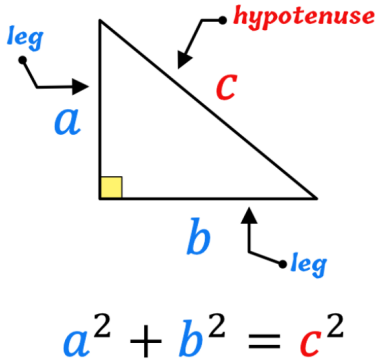


Figure 3.6: Visual Representation of Pythagoras Theorem data[4].

From the figure 3.6 we can see that:

- b is the triangle's base (straight line distance between two points on the Earth's surface)
- a is the triangle's height (i.e., the difference in elevation of the two points)
- c is the triangle's hypotenuse (i.e., the inclined distance)

Inclined distance is a measure of the actual distance traveled by a vehicle considering the varying elevations in the journey. This involves inbuilt elevation and the application of the rule of trigonometry to get the actual path length rather than just the straight line distance. Simply put, the straight-line distance, according to a standard, on a map between two points is the base of a triangle, and the difference in elevation between those two points is the height of that triangle. Therefore, the actual route through which the vehicle has been driven stands for the hypotenuse of that right-angled triangle.

By calculating the hypotenuse of the triangle: Applying simple trigonometry, we can accurately calculate the base (straight line distance) and the height (elevation difference) to create a right-angle triangle.

Inclined Distance with Slope Data:

In addition to using elevation data, inclined distance can also be calculated using slope data. The slope of a road provides another way to determine the inclined distance by applying trigonometric functions. When slope data is available, we can use the cosine function from trigonometry to find the actual path length.

The relationship can be expressed using the trigonometric identity:

$$\cos(\theta) = \frac{\text{adjacent}}{\text{hypotenuse}},$$

and rearranging to solve for the hypotenuse, we get:

$$\text{hypotenuse} = \frac{\text{adjacent}}{\cos(\theta)}.$$

In this context, the adjacent side represents the straight-line distance or the surface distance (d_{surface}) between two points on the Earth's surface and θ is the angle of the slope

Using the slope information, the inclined distance (d_{inclined}) can be calculated as:

$$d_i = \frac{d_s}{\cos(\theta)},$$

where d_i represents the inclined distance and d_s represents the surface distance.

This method is particularly useful in scenarios where precise slope data is available, allowing for an accurate determination of the actual distance travelled considering the road's gradient.

Curvature Information Considerations

If we have curvature information, we could compute the arc length along the path for a more precise estimate of the distance the vehicle has travelled. We can take that curvature information and estimate the central angle and the arc length. Here is the process:

Find the Central Angle (θ):

$$\theta = \text{curvature} \times \text{arc length}$$

Curvature is given as $\frac{1}{\text{radius}}$. The central angle θ should be in radian measure.

Find the Arc Length:

$$\text{Arc Length} = \text{Radius} \times \theta$$

Radius r is the reciprocal of the curvature $\left(r = \frac{1}{\text{curvature}}\right)$.

Thus, when we have curvature data, we can calculate the actual arc length along the path, which provides a more accurate measure of the distance travelled than the straight-line distance.

Mathematical Steps to Calculate Inclined Distance

Given two points with coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2), where x and y represent the latitude and longitude (or horizontal coordinates) and z represents the elevation, the following are the steps of the calculation: First, the straight-line distance (Base) is calculated,

$$\text{Base} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2},$$

then follows the calculation of the elevation difference (Height),

$$\text{Height} = z_2 - z_1,$$

Now, the inclined distance (Hypotenuse) from the Pythagorean theorem is calculated:

$$\text{Inclined Distance} = \sqrt{(\text{Base})^2 + (\text{Height})^2},$$

After the calculation, we check if curvature data is available. If yes, then the central angle is calculated, and the arc length is determined accordingly.

$$\theta = \text{curvature} \times \text{Base},$$

$$\text{Arc Length} = \frac{1}{\text{curvature}} \times \theta.$$

So, for arcs when curvature data are available, the additional arc length would sum to the inclined distance, making it closer to reality.

3.5 Overview of Methods

In this section, we will explain the various methods for precise distance calculation, detailing how each method operates and the scenarios in which they are most effective.

3.5.1 List of New Methods

In this section, we explore the methodology for calculating inclined distances as discussed in 3.4.2, which is an essential aspect of achieving accurate path length estimations in geospatial analysis. Inclined distance incorporates elevation data to provide a more realistic measure of the actual distance travelled by a vehicle. This method goes beyond the simple 2D straight-line distance by considering the three-dimensional aspects of the terrain.

To compute the inclined distance, the first step is to determine the surface distance, which serves as the base of the right-angled triangle in our distance calculations 3.4.2. The surface distance can be calculated using several traditional methods, each offering different levels of precision and computational complexity.

In this section, we present a detailed overview of the new methods that have been employed and tested in the experiments. These methods incorporate elevation data into traditional distance calculation techniques to enhance accuracy as detailed in 2. Each method has been adapted to account for elevation changes, either using elevation data directly or through slope data.

Haversine Method

It is one of the most common procedures in calculating the surface distance between two points on the surface of the Earth. It takes into consideration the fact that the Earth is approximately spherical in shape. The conventional procedure determines the great-circle distance between two points from longitudes and latitudes as detailed in Section 2.3. In this work, we have extended the Haversine method to calculate the inclined distance in two ways. First, with elevation data, we use the Pythagorean theorem to calculate the inclined distance given the elevation difference between two points, as explained clearly in Section 3.4.2. This approach accounts for the vertical component of the distance, providing a more realistic representation of the actual distance travelled. Second, with slope data, we calculate the distance on slopes by using the angle of slope, θ, between the two points. The distance along the surface is corrected using trigonometric functions, as referred to

in Section 3.4.2. This method ensures that the inclined distance reflects the true path length, including the effects of elevation changes and slopes.

Geodesic Method

It is also known as Vincenty's formula, which finds the shortest distance between two points on an ellipsoid, as explained in Section 2.4. This method has the best accuracy in measuring distances. In this research, we have extended Vincenty's formula to include elevation and slope data. With elevation data, the inclined distance is calculated similarly to the Haversine method, incorporating height differences using the Pythagorean theorem 1. With slope data, the horizontal distance is adjusted using the cosine function to account for the sloping angle, as referred to in 3.4.2. These adjustments ensure that the calculated distance reflects the true path length, including the effects of elevation changes and slopes, thereby enhancing the accuracy of the distance measurements.

Spherical Law of Cosines

The Spherical Law of Cosines is another way of calculating horizontal distance on a sphere [2.15]. This method, using trig functions, determines the central angle to provide. We increased the method's scope by incorporating elevation data to improve the study's accuracy. The inclined distance is calculated by combining the surface distance obtained from the Spherical Law of Cosines with the elevation difference using the Pythagorean theorem.

Vincenty Variant Method

This method is a combination of both Vincenty 2.4 and the Spherical law of Cosines 2.15 and is a precise algorithm used to calculate the distance between two points on the Earth's surface, taking into account the Earth's ellipsoidal shape. This method incorporates three key steps to ensure both accuracy and computational efficiency.

The method begins by computing the reduced latitude for each point. This reduced latitude is an adjusted value that accounts for the Earth's flattening due to its ellipsoidal shape. By transforming the geographical latitude into this auxiliary value, the calculations can more accurately reflect the Earth's shape, enhancing the precision of the subsequent distance computations. The use of the reduced latitude is crucial as it corrects for the slight bulge at the equator and the flattening at the poles, which are not addressed in simpler spherical models. Then, The central angle between two points is calculated using the Spherical Law of Cosines. This step simplifies the computational process by initially determining the angle on a sphere, providing a straightforward and efficient way to measure the separation between points. The Spherical Law of Cosines uses trigonometric functions to find the central angle based on the reduced latitudes and the difference in longitudes. This simplification to a spherical model allows for a faster initial calculation, which is then refined to consider the Earth's ellipsoidal shape. Finally, the central angle obtained from the Spherical Law of Cosines is used in the Vincenty formula to refine the distance calculation. The Vincenty formula adjusts the initial spherical calculation to account for the ellipsoidal shape of the Earth, incorporating factors such as the Earth's semi-major axis and flattening. This approach leverages the simplified spherical calculation for speed while using the Vincenty adjustments to enhance accuracy. Additionally, the Vincenty formula includes corrections for the Earth's curvature, providing a more precise measurement than methods that assume a perfect sphere. By combining these steps, the Vincenty Variant

method ensures that the calculated distance is both accurate and computationally efficient, making it suitable for the thesis in estimating accurate path length.

Even though the Spherical Law of Cosines assumes a perfect sphere, the calculated central angle (σ) can still be useful as an initial approximation in more complex models. Once σ is obtained, it can be refined using adjustments that account for the Earth's ellipsoidal shape, as shown in the Vincenty formula 2.4.

Here's the detailed process:

The initial central angle is being calculated by the spherical law of cosines 2.15:

$$\cos(\sigma) = \sin(\beta_1)\sin(\beta_2) + \cos(\beta_1)\cos(\beta_2)\cos(\Delta\lambda).$$

This step quickly provides an approximate central angle and it gets refinement Using Vincenty:

$$\sigma = \arccos(\cos(\sigma))$$

$$P = \frac{\beta_1 + \beta_2}{2}$$

$$Q = \frac{\beta_2 - \beta_1}{2}$$

$$X = (\sigma - \sin(\sigma))\sin^2(P)\cos^2(Q)$$

$$Y = (\sigma + \sin(\sigma))\cos^2(P)\sin^2(Q)$$

$$s = a\left(\sigma - \frac{f}{2}(X + Y)\right)$$

By integrating the spherical approximation within the ellipsoidal refinement, this hybrid method ensures computational efficiency without compromising on accuracy. This combination allows the initial spherical model to guide the more precise ellipsoidal calculations, making the overall process faster and more efficient than iterating from scratch.

These methods have been experimented and the results are formulated in the experiment chapter of this thesis. By using these established techniques, we ensure that the surface distance calculations are grounded in robust mathematical principles. Once the surface distance is determined using one of the methods mentioned above, the next step is to calculate the inclined distance by incorporating elevation data.

Practical Relevance

The addition of inclined distance and curvature data gives this system high accuracy. This also represents the driving path of the vehicle and is critical for applications such as navigation and routing. Better calculation of distance results in improved route planning, especially in hilly or mountainous regions with frequent elevation and curvature changes. Geospatial analysis benefits by providing valuable insights into travel patterns and road usage, considering actual driven distances rather than approximate straight-line distances. Addressing height variations and curvature possibilities corrects deficiencies in traditional distance measurements. Using the Pythagorean theorem and arc length calculations provides a closer, more realistic estimate of the distance travelled, reflecting the true terrain. This approach enhances geospatial analysis accuracy and is practically relevant for accurate simulation and navigation.

3.6 dOC Model Framework

The Deterministic Operating Cycle (dOC) model is a robust framework designed to enhance the precision and reliability of vehicle distance estimations and simulations. By making distance the independent variable, the dOC model facilitates a more accurate and functional representation of various vehicle parameters and attributes over the course of a journey. This approach not only improves the accuracy of distance calculations but also provides a comprehensive basis for various applications in vehicle simulation, energy estimation, and safety systems.

3.6.1 Distance as the Independent Variable

In the dOC model, distance is treated as the independent variable. This means that after calculating the accurate distance between each subsequent point, we use this distance as the foundation for the model. The attributes or parameters of the model are then expressed as functions of this independent variable. This setup allows for a structured and precise mapping of vehicle data along the travelled path.

The attributes of the model, which reflect the operating condition data, are expressed as functions of this independent variable. Specifically, the distance is a function of these attributes, allowing for a structured and precise mapping of vehicle data along the travelled path as expressed in the figure 3.7. Mathematically, this relationship can be depicted as:

$$d = \sum_{i=0}^{n} (d_i),$$

$$f(d) = \{\text{Slopes, Curvature, Road signs...}\}.$$

This setup ensures that distance is accurately mapped as a function of various operating conditions, providing a comprehensive and detailed framework for analyzing and simulating vehicle performance over a journey.

The process begins with the precise calculation of the distance between each subsequent point using the methods discussed previously, such as Haversine, Vincenty, spherical law of cosines, and the 3D distance formula 3.4.2. Once the accurate distance is determined, it serves as the independent variable, and at each point along the path, a set of information is recorded where distance is the key reference point.

Impact on Vehicle Simulation and Safety Systems

Accurate distance data enhances vehicle simulation models, making them more reliable and realistic. This is essential for testing and developing new vehicle technologies and systems in a controlled environment before real-world implementation. Reliable distance measurements are vital for accurate energy estimation, particularly in electric and hybrid vehicles. Understanding how energy consumption varies with distance under different conditions helps in optimizing energy use and improving the efficiency of these vehicles. Safety systems rely on precise distance calculations to function correctly. For example, adaptive cruise control, collision avoidance systems, and autonomous driving technologies all depend on accurate distance data to make real-time decisions that ensure the safety of the vehicle and its occupants.

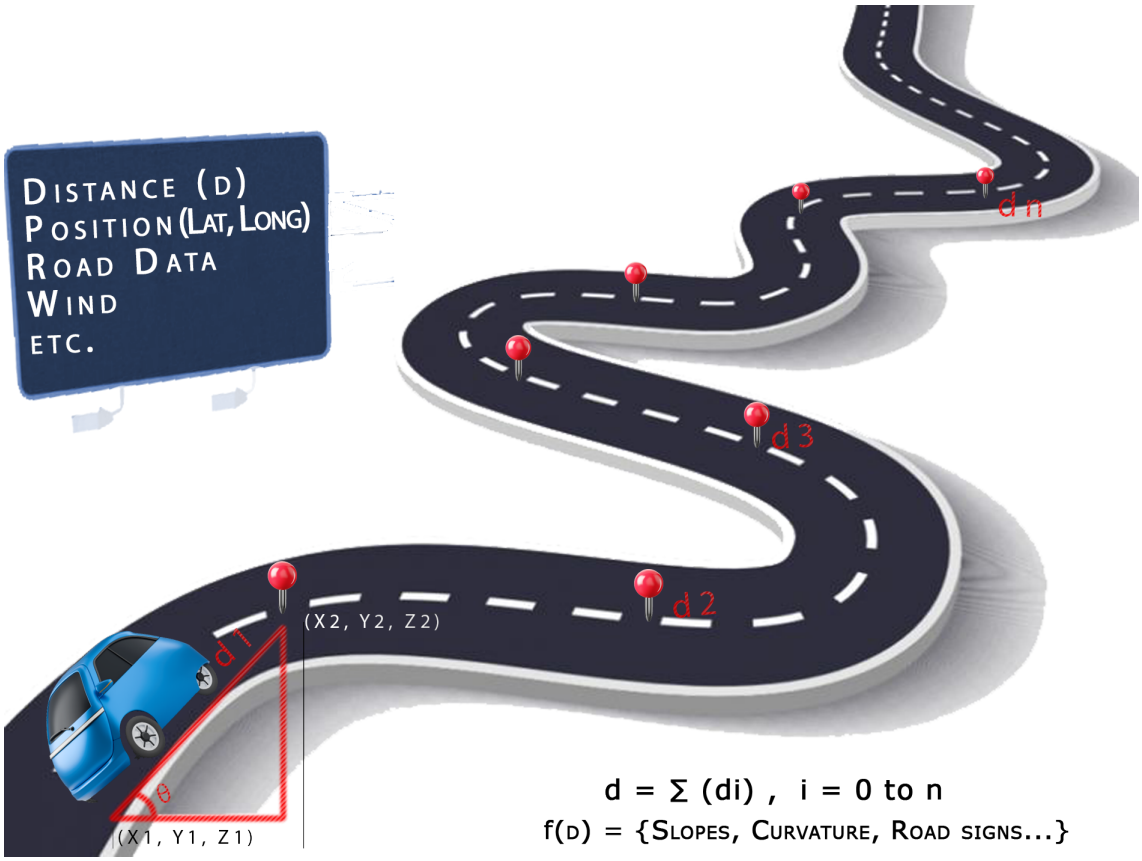


Figure 3.7: Imaginary Representation of dOC file Concept.

Benefits of Using Distance as the Independent Variable

Using distance as the independent variable, attributes can be identified and analyzed more precisely. This is particularly useful in simulations where the exact geographic location may be less relevant than the distance travelled. This approach simplifies the analysis and visualization of how various factors change over the course of a journey.

Distance-based data allows for precise monitoring of vehicle components. Predictive maintenance schedules can be developed based on the actual distance travelled under various conditions. This ensures that maintenance is performed only when necessary, optimizing the lifespan and performance of vehicle components.

In vehicle simulations, using distance as the independent variable allows for more accurate and realistic modelling of vehicle behaviour. This is crucial for testing new vehicle technologies, safety systems, and performance enhancements in a controlled environment.

3.6.2 Attributes of the dOC Model and Usage

In the dOC model, all attributes are treated as functions of distance, ensuring that precise data is available at each and every coordinate point along the vehicle's path. This approach allows for a detailed and accurate representation of various operating conditions, enhancing the model's reliability and usability. Each attribute is recorded in SI units, as previously discussed 3.3.2, to maintain consistency and facilitate straightforward analysis and calculations. The table 3.1 contains the key attributes included in the dOC model file.

Attribute	Description
Distance (m)	Accurate Distance by Calculation, Independent
Latitudes (degree)	High precision latitude coordinates [10^{-7}degree WGS84]
Longitudes (degree)	High precision longitude coordinates [10^{-7}degree WGS84]
Elevation (m)	High precision coordinate heights [m above WGS84 ellipsoid]
Gradient (degree)	Vertical road direction [10^{-3} degree] at coordinate points along the link. No gradient or missing values will be represented as 0.
Headings (degree)	Horizontal road heading [10^{-3} degree] at coordinate points along the link. Missing values will be represented as 0.
Curvatures (m)	Curvature (= 1 / radius) [10^{-6} 1/meter] at coordinate points along the link. Missing values or no curvature will be represented as 0.
Speed Limits (m/s)	Speed Limits when driving on that link. Missing speed limit values are adjusted by comparing the adjacent speed limit values from links and the free-flow speed of that link.
Free Flow Speeds (m/s)	A static average travel speed value for the link in m/s.
Traffic Signal, Stop Sign, Yield Sign (give way), Pedestrian Crossing Sign (Binary)	Binary values represent whether the sign is present in the corresponding link.
Wind Direction (degree)	Direction in degrees.
Wind Velocity (m/s)	Wind velocity in m/s. Nullable.

Table 3.1: dOC Model Attributes

Usage of dOC Model File

The dOC model is encapsulated in a CSV file format, which includes the various attributes recorded as functions of distance 3.1. Each row in the CSV file corresponds to a specific coordinate point along the vehicle's path, with columns representing different attributes such as speed, elevation, slope, curvature, weather conditions, traffic data, and vehicle data. By organizing the data in this structured format, the dOC model file provides a comprehensive and easily accessible dataset for various analyses and applications.

The structured and detailed data within the dOC model CSV file is invaluable for a wide range of applications. In particular, it significantly enhances the accuracy and reliability of residual range prediction. By analyzing the detailed distance-based data, the model can accurately predict the remaining range of the vehicle, considering various factors such as speed, elevation, and operating conditions data. This enables more reliable planning and decision-making for vehicle operations, particularly for electric and hybrid vehicles where precise range estimation is crucial. Additionally, the comprehensive dataset can be utilized for predictive maintenance, route optimization, safety analysis, and improving energy efficiency, making the dOC model file a useful tool in advanced vehicle management and analysis.

Organizing the data in a structured format allows for easier integration and analysis across various platforms and tools. This standardization is essential for ensuring that different stakeholders, such as fleet managers, engineers, and researchers, can access and utilize the data efficiently. The CSV format, being widely supported, ensures compatibility with numerous data processing and analysis software, further enhancing its utility and making it easier to visualise the data.

The diverse range of applications includes optimizing gearshift strategy and adjusting it based on distance to enhance fuel efficiency and vehicle performance.

Overall, the dOC model CSV file serves as a powerful tool for advanced vehicle management and analysis, providing a wealth of detailed and structured data that can be utilized for a variety of critical applications. Its role in enhancing the accuracy of residual range prediction, facilitating predictive maintenance, optimizing routes, improving safety, and boosting energy efficiency highlights its significance in modern vehicle management and research.

Chapter 4

Results and Discussion

In this section, we will take the test scenarios and experiments conducted during this research and discuss the findings. By analyzing the performance of various distance calculation methods in different geographical conditions, we aim to evaluate their accuracy, efficiency, and suitability for practical applications. The discussion will highlight key observations, compare results, and provide insights into the strengths and limitations of each method. Additionally, we will present the experiments involving the creation of dOC files for both short and long distances.

4.1 Test Scenarios

In this section, we aim to evaluate the accuracy of various distance calculation methods in both flat and hilly regions as classified here 2.7. For the flat surface, we choose the Stockholm Marathon Course, known for its relatively even terrain. For the hilly region, we select the track of the Tour de France from Nice to Col de la Couillole, characterized by significant elevation changes. By comparing the accuracy of different methods in these distinct environments, we can assess their performance and suitability for diverse geographical conditions.

The setup of the environment and the implementation have been explained in detail in the Implementation section 3.3. In brief, we pass the input coordinates to the HERE Maps API, retrieve the relevant data, perform the distance calculations using various methods with the Python code, and then analyze the results.

Before proceeding with the analysis of the results, it is important to note that even though the truth value is measured correctly, there is a chance that the measurements were obtained using methods such as running the tangents, which is a common practice. In running, "running the tangents" refers to running the shortest distance possible in a race, certified by the USATF as the Shortest Possible Route (SPR). This method involves cutting corners and taking the most direct path along a course[23]. However, on a curvy course with many turns, it is unlikely for a vehicle to perfectly follow the SPR. While the race course might be USATF certified and your GPS accurate, failing to trace the Shortest Possible Route will likely result in running a greater distance than expected. Despite this potential discrepancy, for the purpose of this test, we rely on the measurements provided as the ground truth value.

4.1.1 Selection of Track and Ground Truth Data Presentation

To compare the results of the experiments, we need a distance that is estimated with high precision and we need different geographical terrains to check the robustness and compare against them to come up with a solution.

- **Flat Surface**: The Stockholm Marathon Course, which is 42,195 meters long and control-measured according to the rules established by the Swedish Athletics Federation and the International Association of Athletics Federations (IAAF). This course provides a reliable ground truth value for comparison.
- **Hilly Region**: The Tour de France track which is 206 km long from Nice to Col de la Couillole, is known for its varied and challenging hilly terrain, which includes significant elevation changes.

Flat Surface

The Stockholm Marathon is an internationally recognized event, and the accuracy of its course measurement is crucial for official records and athlete performance assessments. The measurement process involves multiple steps to ensure precision, including the use of calibrated measuring equipment and validation by certified officials. This high standard of measurement makes the Stockholm Marathon Course an ideal benchmark for evaluating the accuracy of various distance calculation methods.

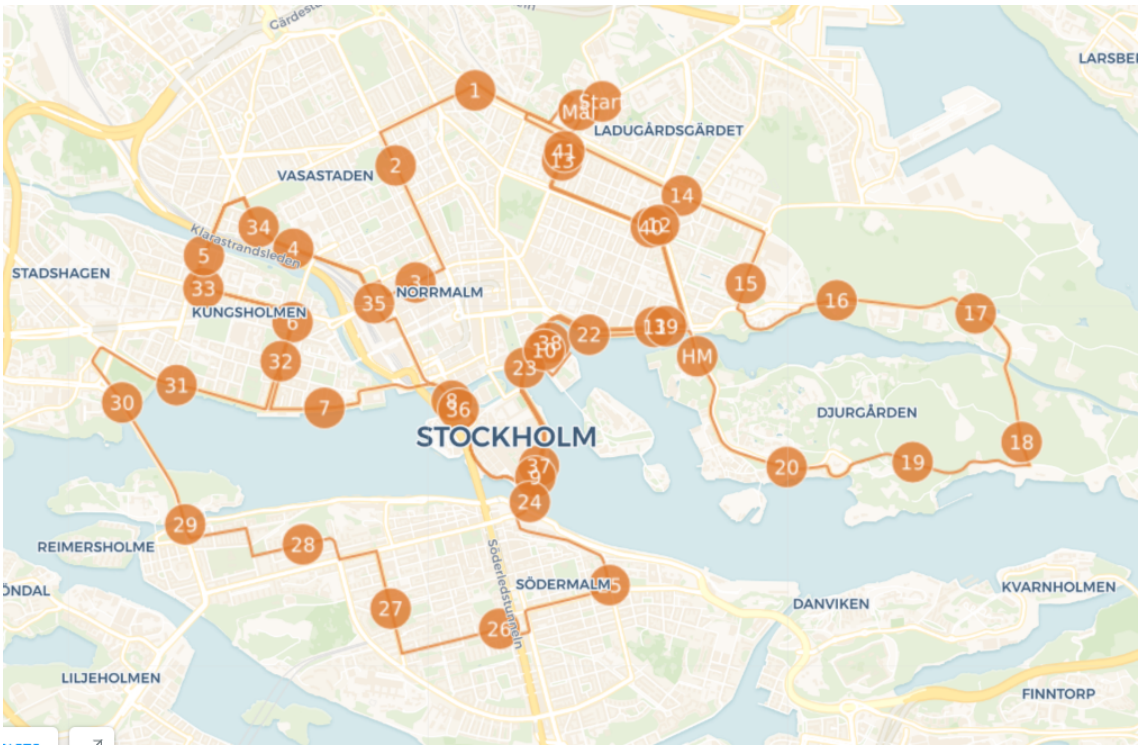


Figure 4.1: Track Marked on Map with Kilometers[28].

By comparing the methods against this well-documented ground truth value, we can determine which method provides results closest to the actual measured distance for the flat terrain. This comparison is critical for assessing the reliability of the methods in real-world applications, such as vehicle navigation, geospatial analysis, and residual range prediction.

The use of such a reliable reference ensures that our findings are robust and applicable to other precise distance estimation needs[28].

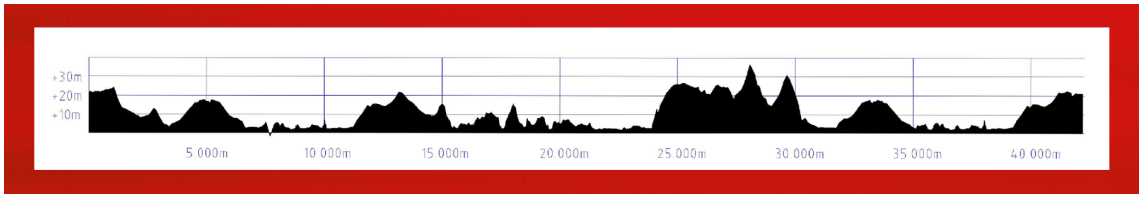


Figure 4.2: Elevation Profile of the Track[28].

Hilly Region

For the hilly region, we selected one of the iconic tracks from the Tour de France, specifically the route from Nice to Col de la Couillole. This track is approximately 132.8 kilometres long and is known for its significant elevation changes, providing a rigorous test for any distance measurement method. By examining the accuracy of different measurement techniques on this challenging course, we can gain valuable insights into their performance and suitability for diverse geographical conditions. The varied terrain of this route, with its steep climbs and descents, offers a comprehensive assessment of how well these methods handle the complexities of real-world environments.

Figure 4.3: Track of the Path on Map[31].

By analyzing the results, we can determine which methods provide the most reliable measurements under such demanding conditions. To illustrate this analysis, the track path 4.3 and elevation profile 4.4 images, offer a visual representation of the terrain and the

specific challenges it presents. This comparative study aims to enhance our understanding of distance measurement techniques in hilly regions, ultimately contributing to more accurate and effective applications in various fields.

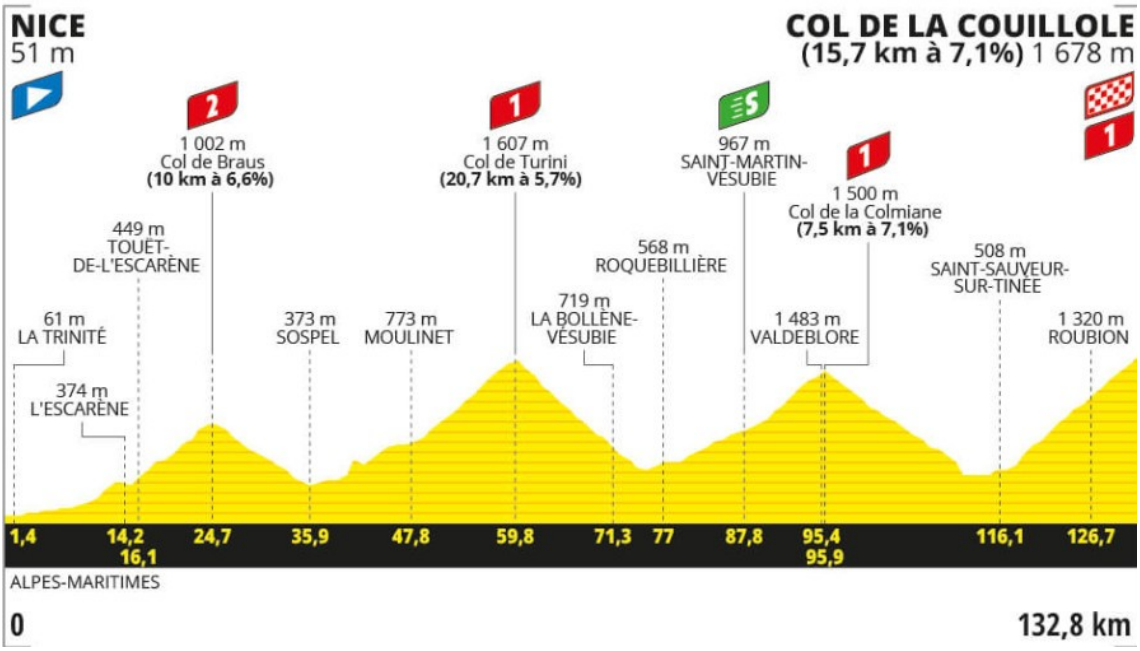


Figure 4.4: Elevation profile of the track[31].

4.1.2 Distance Calculations

Using the GPS coordinates of the marathon track as inputs, we calculated the distances with each method for both flat surface and hilly regions, incorporating both elevation and slope data where applicable. This allows us to determine which method provides results that are closest to the ground truth value. The results are presented in the tables below 4.5 for flat and 4.6 hilly regions respectively, showing the calculated distances and the error percentages relative to the ground truth value. This comparison highlights the most reliable methods for accurate distance estimation, which are crucial for applications such as the dOC model and residual range prediction.

4.1.3 Accuracy and Efficiency Comparison

In this section, we compare the accuracy and efficiency of various distance calculation methods for flat and hilly regions. The flat surface data is taken from the Stockholm Marathon Course, and the hilly region data is from the Tour de France track from Nice to Col de la Couillole. The methods include Haversine, Geodesic, Spherical Law of Cosines, Vincenty Variant, and 3D Distance Formula. The analysis aims to identify the most reliable and computationally efficient method for precise distance estimation in the dOC model.

The tables 4.5 and 4.6 show the calculated distances, their differences from the ground truth, error percentages, and computation times for each method. Here's a detailed discussion of the results:

FLAT SURFACE

Methods	Distance	Difference	Error percentage	Time (s)
Haversine	42064.80018	-130.1998216	0.308566943	0.005748034
Haversine(elevation)	42064.91904	-130.0809575	0.308285241	0.005748034
Geodesic	42166.1446	-28.85540389	0.068385837	0.398048878
Geodesic(elevation)	42166.26317	-28.73682551	0.068104812	0.398048878
Spherical Law of Cosines	42064.80103	-130.1989672	0.308564918	0.005023956
Spherical Law of Cosines(elevation)	42064.9199	-130.0801032	0.308283216	0.005023956
Vincenty Variant	42191.54552	-3.454476449	0.008186933	0.009519339
3D Distance Formula	42065.05736	-129.9426352	0.307957424	0.006843805
Ground Truth	42195	0	0	0

Figure 4.5: Comparison Table for flat surface.

HILLY REGION

Methods	Distance	Difference	Error percentage	Time (s)
Haversine	131968.4957	-831.5043015	0.626132757	0.021517992
Haversine(elevation)	132400.2206	-399.7794173	0.301038718	0.021517992
Geodesic	132084.3495	-715.6504542	0.538893414	1.202219486
Geodesic(elevation)	132515.6255	-284.3744642	0.214137398	1.202219486
Spherical Law of Cosines	131994.9129	-805.087063	0.606240258	0.019957304
Spherical Law of Cosines(elevation)	132426.6343	-373.365651	0.281148834	0.019957304
Vincenty Variant	132664.176	-135.8239862	0.102277098	0.031770706
3D Distance Formula	132418.1581	-381.8419036	0.287531554	0.033400297
Ground Truth	132800	0	0	0

Figure 4.6: Comparison Table for hilly region.

Accuracy Comparison:

1. **Flat Surface (Stockholm Marathon Course)**:
   - The Vincenty Variant method shows the highest accuracy with an error percentage of 0.008186933%.
   - Geodesic methods also perform well with error percentages around 0.068385837%.
   - The Haversine and Spherical Law of Cosines methods show similar moderate accuracy with error percentages around 0.308%.
   - The 3D Distance Formula provides moderate accuracy of 0.30795%.
2. **Hilly Region (Tour de France Nice to Col de la Couillole)**:
   - The Vincenty Variant method again demonstrates the highest accuracy with an error percentage of 0.102277098%.
   - Geodesic (elevation) shows a relatively low error percentage of 0.214137398%.
   - Haversine (elevation) and Spherical Law of Cosines (elevation) also perform well with error percentages around 0.301% and 0.281% respectively.
   - The basic Haversine and Spherical Law of Cosines methods have higher error percentages around 0.626% and 0.606% respectively.

- 3D Distance formula has a difference of -381.84 metres with an absolute error percentage of 0.2875%.

Efficiency Comparison:

1. **Flat Surface (Stockholm Marathon Course)**:
   - The Haversine and Spherical Law of Cosines methods are the most efficient with computation times around 0.005 seconds.
   - The Vincenty Variant method, while highly accurate, takes slightly longer at approximately 0.009 seconds.
   - The Geodesic methods are the slowest, with computation times around 0.398 seconds.
   - The computation time for the 3D Distance Formula on the flat surface is 0.006843805 seconds, indicating it is reasonably efficient.
2. **Hilly Region (Tour de France Nice to Col de la Couillole)**:
   - The Haversine and Spherical Law of Cosines methods remain the most efficient with computation times around 0.021 and 0.019 seconds respectively.
   - The Vincenty Variant method takes approximately 0.031 seconds.
   - The Geodesic methods have the highest computation times, taking about 1.202 seconds.
   - In the hilly region, the computation time is 0.033400297 seconds, which, while slightly longer than some other methods, still represents a good balance between speed and accuracy.

4.1.4 Ideal Option: Balancing Accuracy and Efficiency

As we have compared the accuracy and efficiency of various distance calculation methods, it becomes evident that finding a balance between these two factors is crucial for selecting a reliable method.

Vincenty Variant Method

- **Accuracy**: The Vincenty Variant method consistently provides the highest accuracy for both flat and hilly regions, with minimal error percentages of 0.008186933% and 0.102277098% respectively.
- **Efficiency**: While it is not the fastest method, its computation times of 0.009519339 seconds for flat regions and 0.031770706 seconds for hilly regions are reasonable given its superior accuracy.
- **Analysis**: This method is ideal for applications where precision is critical, such as residual range prediction and high-precision geospatial analyses.

Geodesic Method

- **Accuracy**: This method also demonstrates high accuracy, particularly in hilly regions with an error percentage of 0.214137398%.
- **Efficiency**: It is slower, with computation times around 0.398048878 seconds for flat regions and 1.202219486 seconds for hilly regions.
- **Analysis**: Suitable for applications where high accuracy is required and there are sufficient computational resources to handle the longer processing times.

Haversine and Spherical Law of Cosines

- **Accuracy**: Both methods exhibit moderate accuracy, with error percentages around 0.308% for flat regions and 0.626% for hilly regions.
- **Efficiency**: These methods are highly efficient, with computation times around 0.005 to 0.021 seconds.
- **Analysis**: These methods are best for applications that necessitate quick computations, where moderate accuracy is acceptable.

Conclusion

The analysis highlights that more complex methods like the Vincenty Variant and Geodesic (elevation) offer superior accuracy, though they come at the cost of increased computation time. For applications requiring high precision, the Vincenty Variant method is recommended. However, for real-time applications where speed is critical, the Haversine or Spherical Law of Cosines methods may be more appropriate despite their lower accuracy. This comprehensive evaluation provides a clear understanding of the trade-offs involved, guiding the selection of the most suitable method based on specific application requirements and geographical conditions. Whether the application involves predominantly flat or hilly terrains, the choice of method can be tailored to balance accuracy and efficiency accordingly.

4.2 Statistical Analysis of Distance Calculation Methods

To assess the accuracy and reliability of various distance calculation methods, three tests were conducted using routes from the Tour de France: Test 1 (Piacenza to Turin, flat terrain), Test 2 (Semur to Colombey-les-Deux-Églises, hilly terrain), and Test 3 (Monaco to Nice, short hilly terrain).

The table 4.7 shows the difference between the truth value and the error percentage for each test. Using this information, we will focus on Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Standard Deviation (SD) to compare the performance of each method. This analysis aims to identify the most accurate and reliable distance calculation method based on these statistical measures.

Methods	TEST 1 - Flat		TEST 2- Hilly		TEST 3- Hilly	
	Difference	Error percentage	Difference	Error percentage	Difference	Error percentage
Haversine	544.7110526	0.239960816	-480.2749091	0.271341757	-151.28	0.429894308
Haversine(elevation)	698.2137165	0.307583135	-354.8422603	0.200475853	-90.4624	0.257068396
Geodesic	894.9577762	0.394254527	-247.1825034	0.139651132	-108.131	0.307278147
Geodesic(elevation)	1048.315443	0.46181297	-121.9549839	0.068901121	-47.3921	0.134674898
Spherical Law of Cosines	546.6692733	0.240823468	-429.0493205	0.242400746	-123.95	0.352231586
3D Distance Formula	1192.871701	0.525494141	-343.0148668	0.19379371	-88.9492	0.252768536
Vincenty Variant	703.8767556	0.310077866	39.6206936	0.022384573	-7.19686	0.020451432

Figure 4.7: Test Results Showing Differences and Error Percentages.

The Table 4.8 shows the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Standard Deviation (SD) for each distance calculation method across the three test routes from the Tour de France. Using these statistical measures, the accuracy and reliability of each method are analysed.

Methods	MAE (%)	RMSE (%)	SD (%)
Haversine	0.3137	0.3246	0.0799
Haversine (elevation)	0.255	0.2587	0.0471
Geodesic	0.2804	0.2808	0.0204
Geodesic (elevation)	0.2218	0.227	0.044
Spherical Law of Cosines	0.2785	0.2845	0.0545
Vincenty Variant	0.1176	0.1241	0.0866
3D Distance Formula	0.1183	0.1292	0.1166

Figure 4.8: MAE, RMSE, and Standard Deviation for Various Distance Calculation Methods Across Different Terrains

Haversine Method: The Haversine method shows moderate accuracy across all tests, with an MAE of 0.3137% and an RMSE of 0.3246%. The method has relatively higher error in hilly terrains, indicating its sensitivity to elevation changes.

Haversine with Elevation: Incorporating elevation improves the accuracy of the Haversine method, reducing the MAE to 0.2550% and RMSE to 0.2587%. This indicates that considering elevation data significantly enhances distance measurement accuracy.

Geodesic Method: The Geodesic method demonstrates better accuracy compared to the standard Haversine method, with an MAE of 0.2804% and RMSE of 0.2808%. It performs well in both flat and hilly terrains but shows some sensitivity to elevation.

Geodesic with Elevation: This method yields the lowest errors among all tested methods, with an MAE of 0.2218% and RMSE of 0.2270%. The inclusion of elevation data greatly enhances its performance, making it the most accurate method in this study.

Spherical Law of Cosines: Similar to the Haversine method, the Spherical Law of Cosines shows moderate accuracy with an MAE of 0.2785% and RMSE of 0.2845%. Its performance improves slightly with the inclusion of elevation data.

Vincenty Variant: The Vincenty method has the lowest error rates, with an MAE of 0.1176% and RMSE of 0.1241%. This suggests that it is the most accurate method in this study, making it suitable for routes with significant elevation changes.

3D Distance Formula: This method shows low errors, with an MAE of 0.1183% and RMSE of 0.1292%. The 3D Distance Formula's ability to directly incorporate elevation changes makes it highly accurate, especially in hilly terrains.

The analysis reveals that methods incorporating elevation data, such as Geodesic with Elevation and the Vincenty Variant, provide the most accurate distance measurements. The Vincenty Variant method shows the best overall performance with the lowest MAE and RMSE, making it ideal for applications requiring high precision. The Geodesic with Elevation method also performs exceptionally well, particularly in hilly terrains. Conversely, the Haversine method, while accurate in some scenarios, shows higher errors in this study.

4.3 Experiments

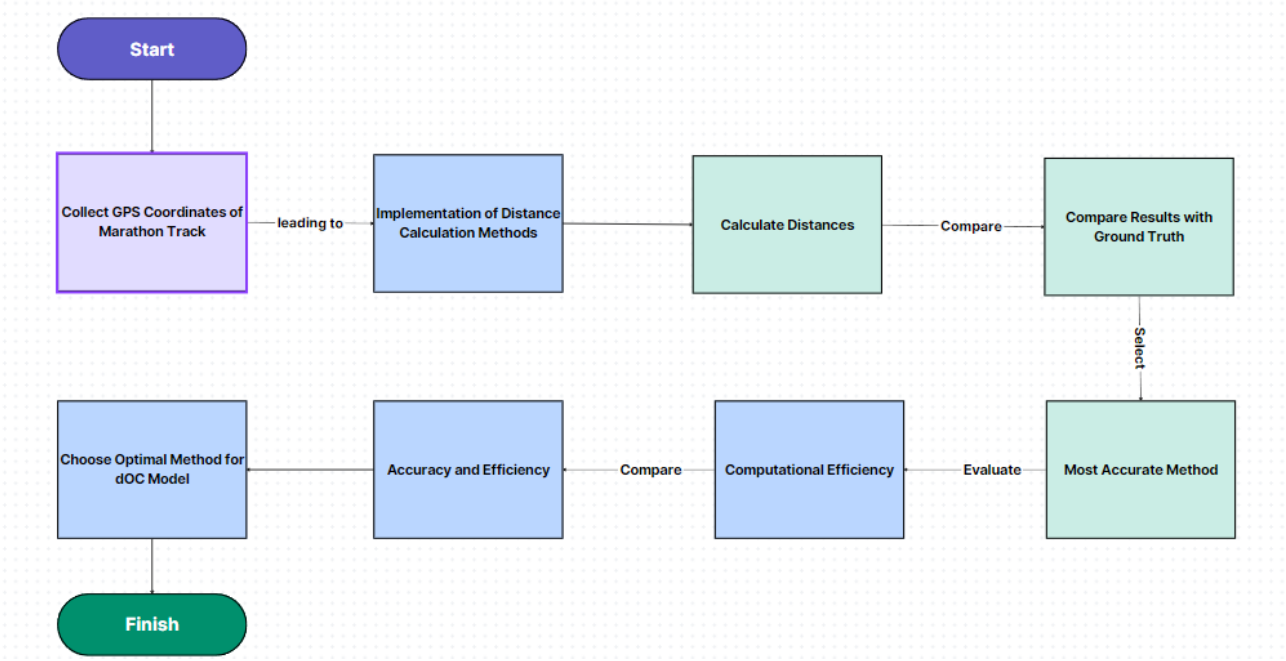


Figure 4.9: Workflow of the process.

Therefore, before diving into the creation of the dOC files, the understanding of the experiment within the context of the flowchart has to be made clear. Each step listed in the 4.9 provides a rigorous measure upon proper evaluation of the various distance calculation methods in terms of accuracy and computational efficiency. It is only then that the process of choice will be ideal in the establishment of dOC files, which have to be efficient and reliable in residual range prediction, among other uses.

The basis of dOC creation is organizing the data about the distance calculated and all applicable attributes varying from speed, elevation, slope, curvature, weather, and traffic data in CSV format. Such structured data is created as an operational model of the vehicle over its journey; it, therefore, details the exact path of travel. The optimal method that has been selected according to the flowchart procedure ensures that the dOC files shall be accurate but also very computationally efficient. Hence, they, in both ways, can serve the purpose of several geospatial analyses or vehicle simulations.

In this experiment, we utilize the Geodesic Inclined distance method as the independent variable to create Deterministic Operating Cycle (dOC) files using coordinates from real road scenarios. This approach allows us to evaluate the method's effectiveness in practical, real-world applications, assess the number of intermediate points generated, and evaluate the overall efficiency of the process.

Furthermore, along with the creation of dOC files, we introduce normally distributed noise to the input coordinates to assess how the results change. This additional step aims to evaluate the robustness of the dOC model and the distance calculation methods in the presence of noise. By comparing the original and noisy data, we can identify which methods are most prone to noise and understand the impact of data quality on the accuracy of the dOC files. This comprehensive analysis ensures that the chosen methods and the resulting dOC files are robust, reliable, and applicable to various real-world scenarios.

4.3.1 Experiment1: Adding Noise to the input

Before generating the dOC files, we conducted an experiment in which normally distributed errors were introduced into the input coordinates. Specifically, we added an error with a standard deviation of 0.0001 to the latitude and longitude values sent to the HERE Maps API. This test aimed to identify the robustness of the HERE Maps data and the distances calculated after data manipulation in the presence of such noise.

In this context, noise was added to mimic the real-world inaccuracies that could arise from GPS errors or data entry inconsistencies. Introducing normally distributed noise allows us to assess the sensitivity of the HERE Maps API and the distance calculation methods to such perturbations. By doing so, we can evaluate how resilient the HERE Maps data is and how the resulting distance calculations vary due to the presence of noise. To further understand the impact of noise, we also directly employed the noisy latitude and longitude values in our distance calculations without passing them through the HERE Maps API again. This was done to see how the distance calculations change when noise is directly applied to the input coordinates versus when the noise is processed by HERE Maps.

The noisy inputs were then sent to the HERE Maps API to evaluate how well the API could handle such perturbations. Interestingly, despite the fact that many of the noisy input points were located off-road, the HERE Maps API demonstrated significant robustness. The HERE maps effectively mapped these noisy points to the nearest road, filtering out the noise and providing clean output coordinates that were accurately aligned with the road. This behaviour was evident in the figure 4.10 where black points represented the noisy input traces, and green points showed the corresponding matched points on the road. The matched path indicated that HERE Maps successfully corrected the noisy inputs, resulting in latitude and longitude values that were consistent with those obtained without noise.

Additionally, as depicted in the figure 4.12, the noisy coordinates sometimes marked a path that was not exactly on the actual route. Despite this, the HERE Maps API, by considering the direction and alignment of the surrounding points, successfully matched the expected route. This demonstrates the HERE maps sophisticated ability to interpret and correct noisy input data, ensuring that the overall route remains consistent with the intended path. This capability is crucial for maintaining the accuracy and reliability of geospatial analyses, especially when dealing with imperfect input data.

However, to further understand the impact of noise, An additional experiment was conducted to understand the impact of noise on distance calculation methods by using noisy

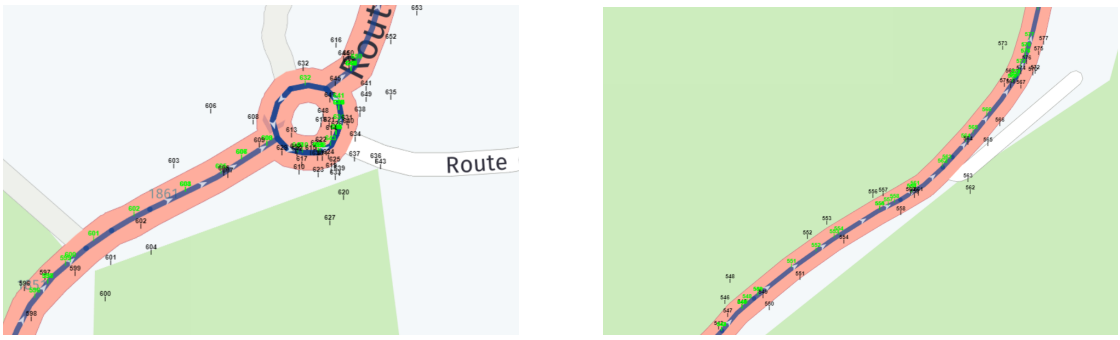


Figure 4.10: Images where black points represent noisy input trace points and green points represent matched points and route

Methods with noise	Distance	Difference	Error percentage
Haversine	255516.24404253913	74474.81958781517	41.136894394266285
Haversine(elevation)	255598.32387450957	74413.72421690417	41.07066735115894
Geodesic	255803.57191563002	74474.08571	41.07113921165204
Geodesic(elevation)	255885.55884173987	74413.10053283401	41.00517578605047
Spherical Law of Cosines	255516.7177541151	74472.32643477159	41.13484316860727
Spherical Law of Cosines(elevation)	255598.79115657913	74411.31571024537	41.06868619
Vincenty Variant	256090.8331660371	74511.91443877804	41.035553554924945
3D Distance Formula	255602.4936770258	74414.89746544018	41.070635640278944
Ground Truth	181100	0	0

Figure 4.11: Results of Distance Calculations with Directly Input Noisy Data

latitude and longitude values directly, along with the actual elevation data, without processing them through the HERE Maps API. The results, as shown in the table 4.11, revealed significant errors in the distance calculations across all methods, including Geodesic Inclined, Haversine, Spherical Law of Cosines, and Vincenty. The introduced noise led to large discrepancies in the computed distances, with error percentages exceeding 41%. This indicates a critical limitation: while the HERE Maps API effectively handles noisy inputs by mapping coordinates to the nearest road, the distance calculation methods themselves are highly susceptible to errors when directly fed noisy data.

The results of this experiment revealed significant errors in the distance calculations. Since the methods used for calculating distances—such as Geodesic Inclined, Haversine, Spherical Law of Cosines, and Vincenty—were not inherently resistant to noise, the introduced perturbations led to large discrepancies in the computed distances. This highlighted a crucial limitation: while the HERE Maps API is robust in handling noisy inputs, the distance calculation methods themselves are vulnerable to errors when directly fed noisy data. The magnitude of the error matters greatly, as even small inaccuracies can significantly impact the effectiveness and reliability of these applications.

In conclusion, this case study demonstrates the importance of using robust APIs like HERE Maps to preprocess noisy GPS data before performing distance calculations. The API's ability to filter out noise and provide accurate road-matched coordinates ensures the reliability of the distance measurements. Conversely, directly using noisy data without such preprocessing can lead to substantial errors, emphasizing the need for robust preprocessing steps in any geospatial analysis workflow.

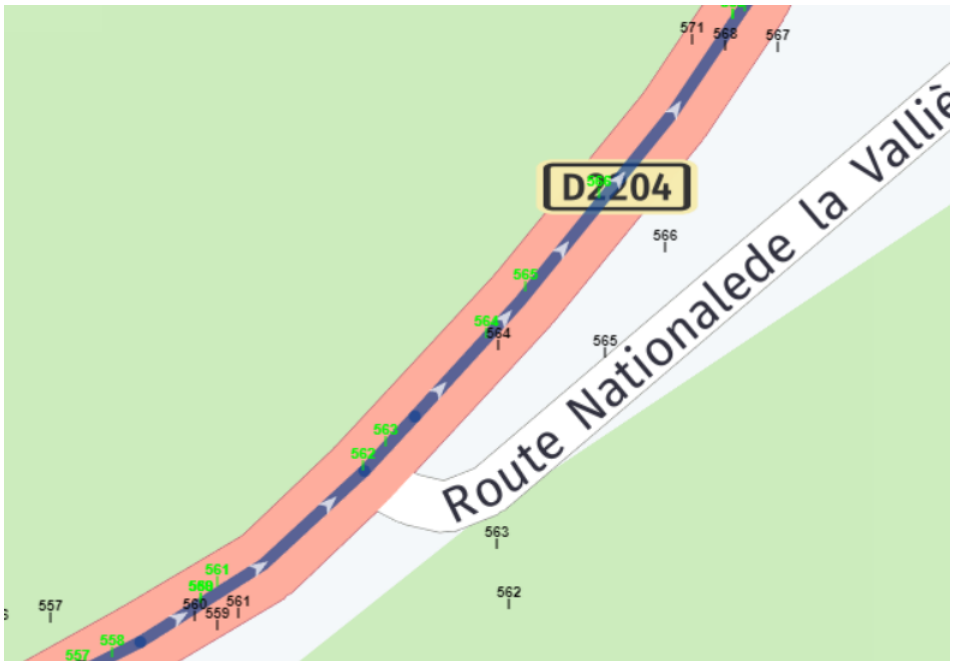


Figure 4.12: Image showing path matched by here maps despite of noisy input

4.3.2 Experiment2: Creating a dOC File for a Shorter Distance (Lund to Malmo)

In this section, we discuss how the dOC file can be generated for a lesser length, say from Lund to Malmo, which is approximately 20 km long. By employing the Geodesic Inclined distance method as the independent variable for the dOC file, we will attempt to capture all of the operation conditions data available at various locations along the path.

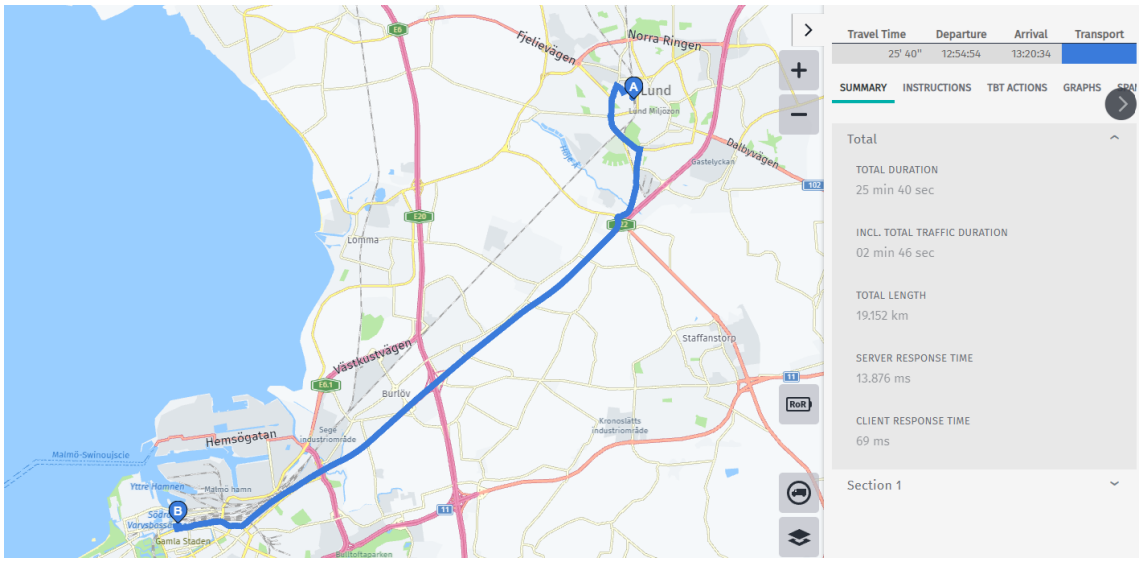


Figure 4.13: Path from Lund to Malmo on HERE Maps Router

Steps:

1. **Selection of Route**: The route is between Lund and Malmo, covering around 20 km, given the input as GPS coordinates
2. **Data Collection**: The Coordinates, elevation and the data along the route were collected using HERE Maps API
3. **Distance Calculation**: The Geodesic Inclined distance method is used to get the distances between subsequent points so that in a way the distance includes the elevation along with the distance.
4. **dOC File**: dOC file is created taking distance as an independent variable from the above step. We have a resultant number of points as 527 along the path. This allows for detailed operating conditions data at 527 distinct points over the 20-kilometer route.

Data Visualization and Analysis

Once the dOC file is generated, it can be utilized in several data visualization and analysis applications. The following are a few use-case scenarios:

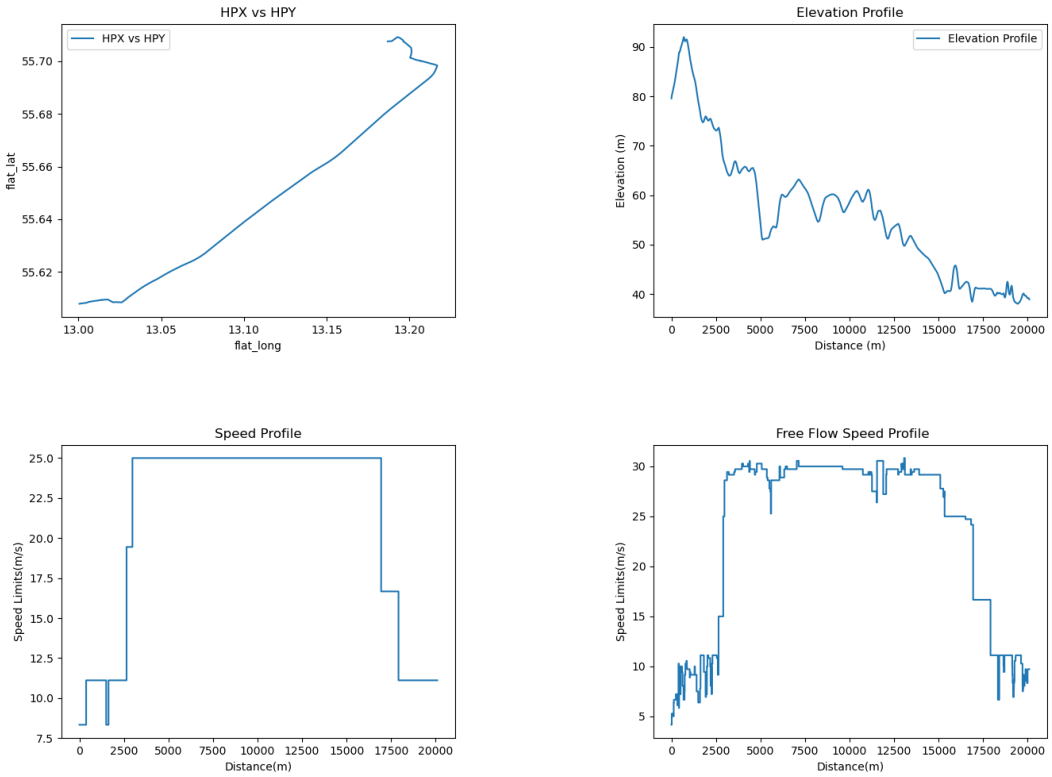


Figure 4.14: Visualisation of data from dOC file.

1. **Path Tracing**:
   - **Latitude and Longitude Plotting**: Plotting the latitude and longitude coordinates can result in a view of the complete route from Lund to Malmo on

a map. This helps to understand precisely which route is followed and if any deviations or something is interesting en route.

- **Tools**: From the data these can be plotted using GIS or Python libraries with Matplotlib and Folium for the visualization

2. **Plot Elevation Profile**:
    - One can plot the elevation to show how it varies across a route. This is particularly useful in determining the elevation and the amount of any significant climbing and descending.
    - **Tools**: Python can be used with libraries like Matplotlib for plotting an elevation plot, which produces a careful breakdown of the ground.
3. **Speed Analysis**:
    - **Plotting Speed**: If speed data is in hand, it can be plotted against distance to see how speed changes over the route. This can help detect dropping or gaining speeds in segments, which might be due to several effects like road conditions and traffic.
    - **Free Flow Speed Plotting**: Additionally, the free flow speed along the path can be plotted to compare the actual speed with the ideal conditions, providing insights into the impact of traffic and other constraints.
    - **Tools**: Speed data could be visualized in an application for plotting libraries to make speed profiles in detail.
4. **Other Plots**: Besides the analyses described above, the values present in any of the dOC files can be easily mapped on user demand for multiple requirements. For example, traffic signal locations over the path can be located to analyze how the density of the signals affects travel times. The same is true for other elements: speed limits, road types, and points of interest. This enables an all-inclusive analysis of the various factors that influence the journey, thus allowing more informed decisions regarding route planning, traffic management, and vehicle navigation. The dOC model becomes a powerful tool for a wide range of applications related to geospatial and transportation by tailoring the visualizations according to specific requirements.

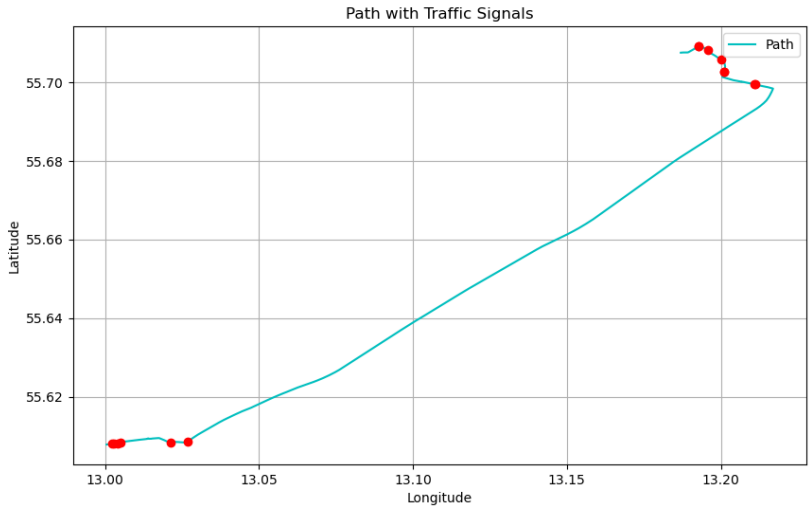


Figure 4.15: Ploting of Traffic Signals along the path.

4.3.3 Experiment 3: Creating a dOC File for a Longer Distance (Oslo to Bergen)

In this experiment, we aim to create a Deterministic Operating Cycle (dOC) file for a longer distance, specifically the route from Oslo to Bergen, which spans approximately 463.34 kilometres. By having distance as the independent variable, we can capture detailed operating conditions data at each point along the route. For this extensive distance, we have 15,360 points, ensuring comprehensive coverage of the route's operating conditions.

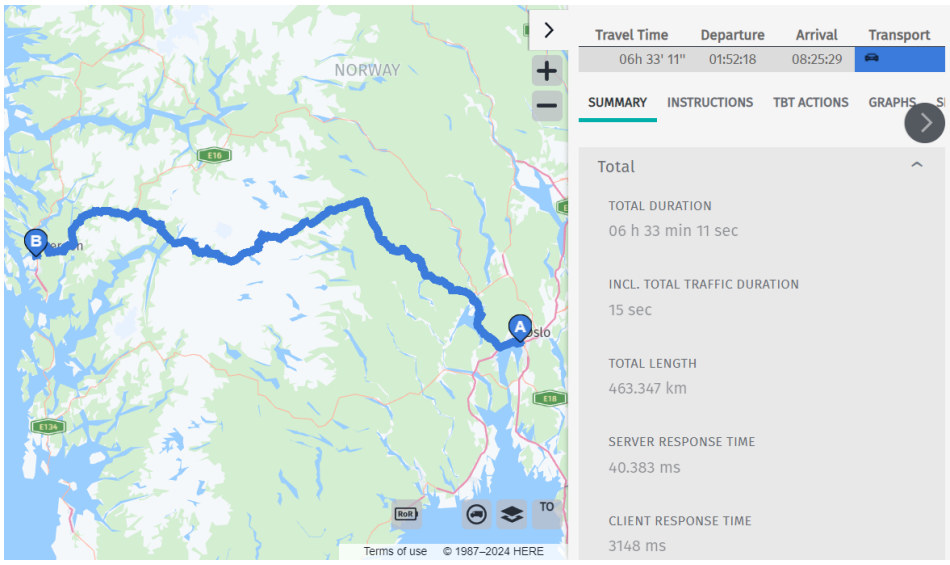


Figure 4.16: Path from Oslo to Bergen on HERE Maps Router.

The process of creating the dOC file involves collecting GPS coordinates and elevation data using the HERE Maps API, similar to the experiment conducted for shorter distances 4.3.2. The Geodesic Inclined distance method is applied to calculate the distances between subsequent points, incorporating elevation data to enhance accuracy. Each point along the 463-kilometer route is then annotated with various attributes, such as speed, elevation, slope, curvature, and other relevant operating conditions. This approach ensures that the model is robust and provides a granular view of the route's characteristics.

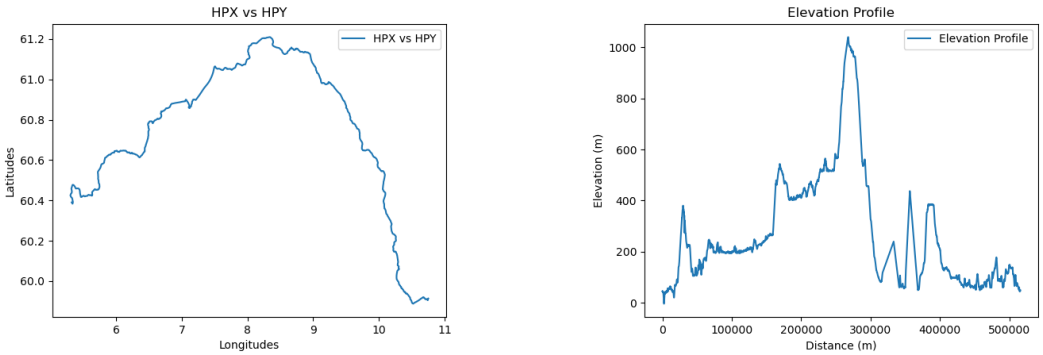


Figure 4.17: Visualisation of data from dOC file.

The creation of such a detailed dOC file, with distance as the independent variable, offers significant advantages for numerous applications. By capturing operating conditions at each of the 15,360 points along the route, the model becomes highly reliable for tasks such as residual range prediction, vehicle simulation, route optimization, and real-time navigation assistance. The ability to visualize and analyze data for longer distances, regardless of the number of points, enhances the model's applicability and usability in real-world scenarios 4.17. This comprehensive dataset allows for in-depth analysis and decision-making, making the dOC model a valuable tool for a wide range of geospatial and transportation-related applications.

Chapter 5

Conclusion

5.1 Summary

his thesis has extensively analyzed and compared multiple distance calculation methods to arrive at an optimal solution for the crucial need for accurate distance measurements in predicting the residual range. The methodologies evaluated include the Haversine method, Geodesic method, and elevation-embedded algorithms, all of which are essential for calculating precise path lengths. These accurate distance measurements are fundamental to the deterministic Operating Cycle (dOC) model, which encapsulates critical road data and serves as a crucial element in simulations for predicting the residual range of vehicles.

The study has demonstrated that the choice of distance estimation method significantly impacts the accuracy of residual range predictions. The findings enable the selection of the most suitable method based on specific application requirements, thereby enhancing the reliability of vehicle performance simulations. This research not only addresses the posed research questions but also contributes to the advancement of sustainable transportation systems. By providing insights into effective distance calculation techniques and their integration into the dOC model, this study supports the broader goal of promoting sustainable, safe, and efficient transport solutions, ultimately aiding in the mitigation of climate change impacts.

5.2 Limitations

Despite its contributions, this study has several limitations. The model's reliance solely on data from HERE Maps means that using a different geospatial API would require adjusting the dOC model attributes. Additionally, importing new attributes from other platforms may necessitate further modifications. The distance calculations are more computationally intensive than traditional methods, potentially affecting evaluation efficiency and run-time. The accuracy of the model is heavily dependent on the quality of input data, such as GPS accuracy and road information. Furthermore, while the model has been tested with real-time data and can be customized for different vehicle types, integrating data from multiple sources presents interoperability challenges, affecting consistency and accuracy.

5.3 Future Works

Future work should focus on seamlessly integrating attributes from various geospatial platforms, taking into account the distance traveled by the vehicle. The dOC model will be developed with a more flexible and adaptive framework to accommodate heterogeneous data sources, enhancing its applicability and robustness. This integration will ensure the model remains adaptable to new data formats and attributes, maintaining compatibility with different geospatial APIs and platforms.

Efforts should also be made to eliminate small discrepancies in distance calculations and attribute integrations. By refining algorithms and methodologies, the dOC model can be transformed into a comprehensive road data tool for diverse applications. This tool can provide accurate and detailed road data tailored to specific use cases, such as urban planning, autonomous vehicle navigation, and transportation logistics.

Further improvements should aim at enhancing the computational efficiency of distance calculation methods, making them suitable for real-time applications. Optimization of computational efficiency will reduce processing times without losing information, through improved algorithms and advanced computing techniques.

In conclusion, the development of the dOC model in geospatial analysis and vehicle simulation holds significant potential. With the integration of attributes from different platforms, accuracy enhancements, and improved computational efficiency, the model will become a valuable tool for researchers, developers, and industry experts, contributing to the evolution of sustainable and intelligent transportation systems.

Acknowledgements

I would like to express my deepest gratitude to Carl Emvin, my thesis supervisor from Chalmers, for always guiding me in the right direction whenever I veered off course. Your selfless dedication and the considerable amount of time you spent mentoring me opened many doors of opportunities. Thank you for your unwavering support and commitment.

I also wish to thank Dr. Fredrik Bruzelius from Chalmers for helping me understand the basics and giving me a much-needed push during the early stages of my research. More than a supervisor, you were a friend, and your support was invaluable.

Additionally, I am grateful to Dr. Bengt J H Jacobson from Chalmers, who took the time to guide me amidst his busy schedule. Giving me the opportunity to present my thesis at SVEA was considered a significant milestone, made possible through your encouragement and inspiration, particularly your exemplary time management skills.

I extend my gratitude to the VEAS department at Chalmers for creating a workspace that felt like home and providing me with the freedom to work independently. Your support made a significant difference in my research journey.

I am also thankful to Andersson Rickard from the Volvo Group for accepting me for this thesis and for providing timely help whenever I needed clarification. Your assistance was crucial for the progress of my work.

I would like to acknowledge my universities and supervisors: Dr Kiss Balint from Budapest University of Technology, Dr Gastone Pietro Rosati Papini, and Dr Davide Brunelli from the University of Trento. Your trust and the freedom you granted me in my thesis work were pivotal to my success.

A heartfelt thank you to my brother Raghul for his financial support, ensuring that I could focus entirely on my thesis work without any distractions. Additionally, I am deeply grateful to my mother and Maria for their constant motivation throughout this journey.

Finally, I would like to thank all my friends in Gothenburg who supported me during this thesis work. Your encouragement and companionship were greatly appreciated.

Thank you all for your contributions and support. This research was only possible with you.

Bibliography

[1] International Energy Agency. Advancements in vehicle technology and their impact on co2 emissions. *IEA Transport Sector Report*, 29(3):12–34, 2021. URL https://www.iea.org/reports/advancements-in-vehicle-technology-2021. Accessed: 2024-06-09.

[2] BloombergNEF. Electric vehicle outlook 2020, 2020. URL https://about.bnef.com/electric-vehicle-outlook/.

[3] Lambert Bovens. *Applied Mathematics for Geospatial Analysis: Theory and Practice*. Springer, New York, NY, 2003. ISBN 978-0-387-95548-6.

[4] ChiliMath. Pythagorean theorem. https://www.chilimath.com/lessons/geometry-lessons/pythagorean-theorem/. Accessed: 2024-06-12.

[5] Wikipedia contributors. Spherical law of cosines. https://en.wikipedia.org/wiki/Spherical_law_of_cosines, 2024. Accessed: 2024-06-03.

[6] Microsoft Copilot. A man driving a car, sweating on forehead, bit nervous with driving anxiety thinking about the battery percentage like on the top right of his head. Generated by Microsoft Copilot, 2024.

[7] OpenAI DALL-E. Digital artwork representing geospatial api. https://your_image_url_here, . Accessed: 2024-06-12.

[8] OpenAI DALL-E. Digital artwork of spherical and ellipsoidal earths. https://your_image_url_here, . Accessed: 2024-06-12.

[9] OpenAI DALL · E. A car on a road with here maps api information, 2024. Generated using DALL · E from OpenAI. The image includes latitude, longitude, elevation, and slope information displayed around the car on a scenic road.

[10] Roger Deakin. *A Guide to Coordinate Systems in Great Britain*. Ordnance Survey, Southampton, UK, 2006. URL https://www.ordnancesurvey.co.uk/documents/resources/guide-coordinate-systems-great-britain.pdf.

[11] Dubizzle. The future of petrol cars: What you need to know. https://www.dubizzle.com/blog/cars/petrol-cars-future/. Accessed: 2024-06-12.

[12] HERE. Introduction to Mapping Concepts User Guide, n.d. URL https://www.here.com/docs/bundle/introduction-to-mapping-concepts-user-guide/page/topics/topology.html.

[13] HERE Technologies. Here developer portal. https://www.here.com/developer?cid=Developer_Here-Google-YT-0-Dev-EMEA-SE-matchtype=e&matchtype=e&gad_source=1. Accessed: 2024-06-09.

[14] HERE Technologies. Maps api for javascript developer guide, Year of Access. URL https://www.here.com/docs/bundle/maps-api-for-javascript-developer-guide/page/README.html.

[15] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS—Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[16] Elliott D Kaplan and Christopher J Hegarty. Understanding gps principles and applications. *Artech house*, 2005.

[17] Elliott D. Kaplan and Christopher J. Hegarty. Understanding gps: Principles and applications. *Artech House Mobile Communications Series*, 2006.

[18] David C. Lay. *Linear Algebra and Its Applications*. Addison-Wesley, 3rd edition, 2003. ISBN 978-0-201-70970-5.

[19] Hagar Mahmoud and Nadine Akkari. Shortest path calculation: A comparative study for location-based recommender system. In *2016 World Symposium on Computer Applications Research (WSCAR)*, pages 1–5, 2016. DOI: 10.1109/WSCAR.2016.16.

[20] Pratap Misra and Per Enge. Global positioning system: signals, measurements and performance. *Ganga-Jamuna Press*, 2011.

[21] Union of Concerned Scientists. Cleaner cars from cradle to grave, 2015. URL https://www.ucsusa.org/sites/default/files/attach/2015/11/Cleaner-Cars-from-Cradle-to-Grave-full-report.pdf.

[22] International Council on Clean Transportation. Incentives for electric vehicles in european countries, 2019. URL https://theicct.org/publications/incentives-electric-vehicles-european-countries.

[23] Running on Happy. Gps accuracy and race course distances, 2017. URL https://runningonhappy.com/2017/06/gps-accuracy-race-course-distances/. Accessed: 2024-06-23.

[24] United Nations Environment Programme. Emissions gap report 2023: Broken record – temperatures hit new highs, yet world fails to cut emissions (again), 2023. https://wedocs.unep.org/20.500.11822/43922.

[25] Hannah Ritchie. Cars, planes, trains: where do co_2 emissions from transport come from? *Our World in Data*, 2020. https://ourworldindata.org/co2-emissions-from-transport.

[26] Luigi Romano, Pär Johannesson, Erik Nordström, Fredrik Bruzelius, Rickard Andersson, and Bengt Jacobson. A classification method of road transport missions and applications using the operating cycle format. *IEEE Access*, 10:73087–73121, 2022. DOI: 10.1109/ACCESS.2022.3188872.

[27] Roger W Sinnott. Virtues of the haversine. *Sky and Telescope*, 68(2):158–159, 1984.

[28] Stockholm Marathon. Bansträckning stockholm marathon. https://www.stockholmmarathon.se/start/banan/. Accessed: 2024-06-11.

[29] Gilbert Strang. *Calculus*. Wellesley-Cambridge Press, 1991.

[30] George B. Thomas and Ross L. Finney. *Calculus and Analytic Geometry*. Addison-Wesley, 9 edition, 1996.

[31] Tour de France. Stage 20 - lacapelle-marival > rocamadour. https://www.letour.fr/en/stage-20. Accessed: 2024-06-12.

[32] United Nations Framework Convention on Climate Change (UNFCCC). Paris agreement. 2015. URL https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement. Accessed: 2024-06-16.

[33] G. Van Brummelen. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press, 2013. ISBN 9780691148922. URL https://books.google.se/books?id=0BCCz8Sx5wkC.

[34] Thaddeus Vincenty. *Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations*. National Geodetic Survey, NOAA, Rockville, MD, 1975. URL https://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf.

[35] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93, 1975.

Appendix

A.1 Poster

In the appendix, I have included the poster summarizing my thesis work, which was presented at the SVEA (Swedish Vehicular Engineering Association) conference. This poster provides a concise overview of my research findings and methodologies, highlighting the key aspects and contributions of my study.

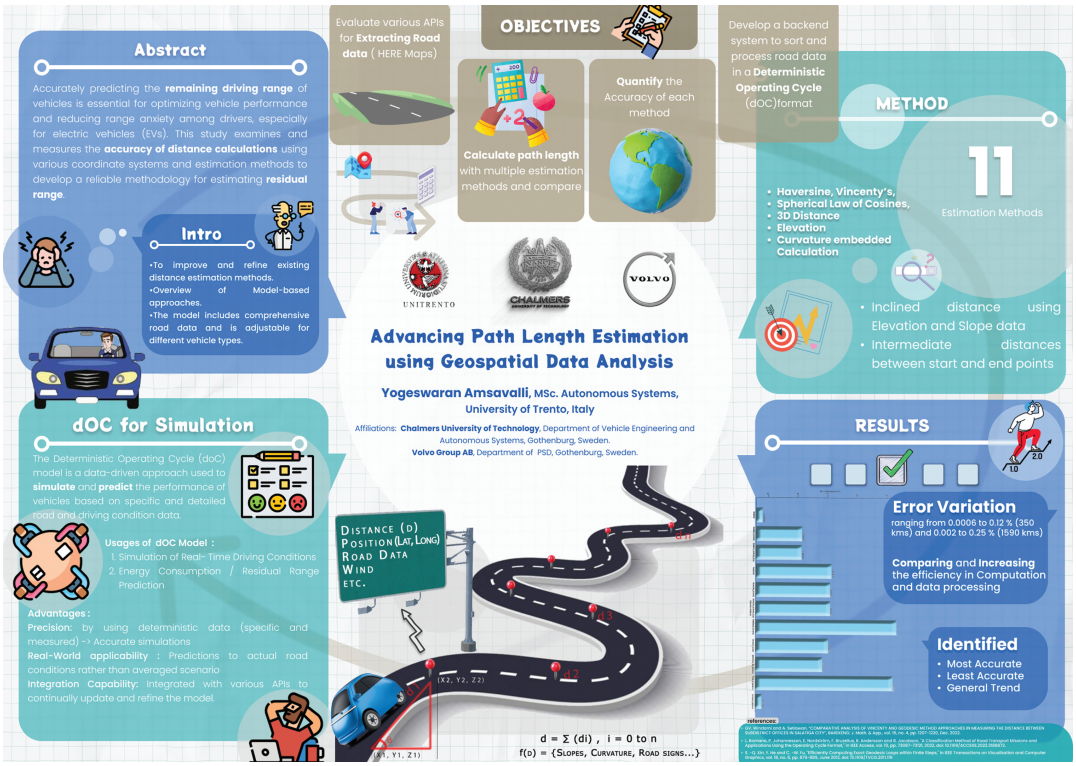


Figure A.1.1: Summary of Thesis Work Presented at the SVEA Conference: Advanced Distance Estimation Methods and Usage of the dOC Model

A.2 Working Code

In this section, I have included the working code used in my thesis. This code is essential for replicating the study and understanding the detailed computational processes involved in the research. Below is an overview of the functionalities provided by the code. I have also added my GitHub below where you can refer and clone the repository for the usage.The primary goal of this code is to extract detailed road data from GPS traces and create a deterministic operating cycle (DOC) model. This model is used for various applications, including residual range estimation for vehicles.

Features

- **Extract Road Data**: The code extracts road data by processing latitude and longitude GPS traces from a vehicle.
- **Generate DOC Model**: It returns a deterministic operating cycle (DOC) model in CSV format, which includes attributes such as elevation, slope, speed limit, and more.
- **Distance Calculation**: Implements accurate distance calculation using Haversine and Vincenty formulas.
- **Energy Consumption Analysis**: Facilitates analysis of energy consumed by the vehicle and estimation of remaining travel distance based on available battery or fuel.

Installation and Usage

To get started, clone the repository of mine to your local machine

```
    git clone https://github.com/Yogiii1762/doc_model_road.git
```

Requirements

Ensure you have the following dependencies installed:

```
jsonpath-ng==1.5.3
numpy==1.21.2
pandas==1.3.3
pyproj==3.2.1
haversine==2.5.1
matplotlib==3.4.3
requests==2.26.0
```

Usage

Extracting Road Data

To extract road data and generate the DOC model, you will need a HERE Maps API key. Run the script given in the github

- **API key**: Your HERE Maps API key

- **Input**: Path to the input CSV file containing latitude and longitude GPS traces.
- **Output**: Path to the output CSV file where the DOC model will be saved. (Optional: by default the input directory)

Input and Output

The input should be a text file and should contain gps traces like below :

```
Latitude, Longitude
x1, y1
x2,y2
xn,yn
```

The output is a dOC file as explained before 3.6.2.

A.2.1 Code

```
   # file: Doc_Model.py
#Description : This program reads an input file to extract latitude and longitude data, which it
     sends to the HERE API to retrieve route information. It then processes the JSON response to
     obtain necessary attributes, calculates the distance between consecutive points, and saves the
     data to a CSV file. Additionally, the program generates a report and a graph of the vehicle's
     path.It cleans the data automatically.
# author: Yogeswaran Amsavalli
# version: 3.2.2
# date: 2024-05-09

#Import the necessary libraries.
import json
from jsonpath_ng import jsonpath, parse
import numpy as np
import pandas as pd
from math import radians, sin, cos, sqrt, atan2, asin
from pyproj import Geod
from haversine import haversine
import matplotlib.pyplot as plt
import os
import requests


def process_file(filepath, output_directory):

#importing Json file from API_call (Replace with your own API key)
    url = "https://routematching.hereapi.com/v8/match/routelinks"
    api_key = "Replcae with your HERE Maps API Key"


    #To access single file from the computer: Add the filepath here


    with open(filepath, "r") as file:
        lat_lon_data = file.read().strip()

    headers = {
        "Content-Type": "application/json"
    }

    params = {
        "apikey": api_key,
        "mode": "fastest;truck;traffic:disabled",
        "routeMatch": "1",
        "attributes": ["ADAS_ATTRIB_FCn(*)", "APPLICABLE_SPEED_LIMIT(*)", "TRAFFIC_SIGN_FCn(*)", "
     TRAFFIC_PATTERN_FCn(*)", "ARCHIVED_WEATHER(*)"]
    }

    response = requests.post(url, headers=headers, params=params, data=lat_lon_data)
```

```
if response.status_code == 200:
    data1= response.json()
    print("Getting Response from API Successful")
else:
    print("Error:", response.status_code, response.text)


#This Functiuon reads the coordinates from the file and returns the start and end coordinates for
  calculation

def read_coordinates(filepath):
    with open(filepath, "r") as file:
        lines = [line.strip() for line in file.readlines() if line.strip()]  # Skip empty lines

    # Skip the first line (header)
    lines = lines[1:]

    # Initialize start and end coordinates
    start_lat, start_lon, end_lat, end_lon = None, None, None, None

    # Iterate over the lines
    for line in lines:
        # Replace ", " with "," and split the line into latitude and longitude
        try:
            lat, lon = map(float, line.replace(", ", ",", ).split(","))
            # If this is the first valid line, set the start coordinates
            if start_lat is None and start_lon is None:
                start_lat, start_lon = lat, lon
            # Update the end coordinates
            end_lat, end_lon = lat, lon
        except ValueError:
            # If the line could not be split into two floats, skip it
            continue

    return start_lat, start_lon, end_lat, end_lon

start_lat, start_lon, end_lat, end_lon = read_coordinates(filepath)


#Uncomment the below code to read the JSON file from the computer and specify the path of the
 file in the filePath1 variable:

# filePath1 = "/home/yogi1762/MScThesis_dOC-generator_2024/MScThesis_dOC-generator_2024/
 StartupLibrary/Test_Responses/Oslo.json"
# data1= data1
#Extracting the data from the JSON file:
# with open(filePath1, 'r') as file:
#     data1 = json.load(file)
# start_lat,start_lon = add start coordinates here
# end_lat,end_lon = add end coordinates here

# Exrtracting data from json and appending to lists for calculations
Latitude = []
Longitude = []
Elevation = []
Slopes = []
Curvatures = []
Headings = []
Ref_Nodes = []
NonRef_Nodes =[]

jsonpath_expr = parse("$.response.route[0].leg[0].link[*].linkId")
LinkID = [int(match.value) for match in jsonpath_expr.find(data1)]

#Link data from HERE API
link_length = parse("$.response.route[0].leg[0].link[*].length")
link_length_Match = [float(match.value) for match in link_length.find(data1)]
Distance_HERE= data1['response']['route'][0]['summary']['distance']
```

```
attributes = ['HPY', 'HPZ', 'HPX', 'SLOPES', 'CURVATURES','HEADINGS','REFNODE_LINKCURVHEADS','
 NREFNODE_LINKCURVHEADS']

attr_matches = {}

# Iterate over attributes and find matches
for attr in attributes:
    jsonpath_expr = parse(f"$.response.route[0].leg[0].link[*].attributes.ADAS_ATTRIB_FCN[0].{
 attr}")
    matches = [json.loads(match.value) for match in jsonpath_expr.find(data1)]
    attr_matches[attr] = matches

# Extend lists maintaining order
for i in range(len(attr_matches[attributes[0]])):
    for attr in attributes:
        matches = attr_matches[attr]
        if i < len(matches):
            if attr == 'HPY':
                Latitude.append(matches[i])
            elif attr == 'HPZ':
                Elevation.append(matches[i])
            elif attr == 'HPX':
                Longitude.append(matches[i])
            elif attr == 'SLOPES':
                Slopes.append(matches[i])
            elif attr == 'CURVATURES':
                Curvatures.append(matches[i])
            elif attr == 'HEADINGS':
                Headings.append(matches[i])
            elif attr == 'REFNODE_LINKCURVHEADS':
                Ref_Nodes.append(matches[i])
            elif attr == 'NREFNODE_LINKCURVHEADS':
                NonRef_Nodes.append(matches[i])
        else:
            # If no match found for a particular index, extend with [0, 0, 0]
            if attr == 'REFNODE_LINKCURVHEADS':
                Ref_Nodes.append([0, 0, 0])
            elif attr == 'NREFNODE_LINKCURVHEADS':
                NonRef_Nodes.append([0, 0, 0])
            else:
                # For other attributes, extend with None or any other default value if necessary
                pass

#Creating the Lists for the data and Replacing the missing data with 0

for i in range(len(Curvatures)):
    for j in range(len(Curvatures[i])):
        if Curvatures[i][j]== 1000000000 :
            Curvatures[i][j]=0

for i in range(len(Curvatures)):
    if not Curvatures[i]:
        if LinkID[i] > 0:
            Curvatures[i] = [Ref_Nodes[i][1], NonRef_Nodes[i][1]]
            Headings[i] = [Ref_Nodes[i][2], NonRef_Nodes[i][2]]
        elif LinkID[i] < 0:
            Curvatures[i] = [NonRef_Nodes[i][1], Ref_Nodes[i][1]]
            Headings[i] = [NonRef_Nodes[i][2], Ref_Nodes[i][2]]

    else :
        if LinkID[i] > 0:
            Curvatures[i].insert(0,Ref_Nodes[i][1])
            Curvatures[i].append(NonRef_Nodes[i][1])
            Headings[i].insert(0,Ref_Nodes[i][2])
            Headings[i].append(NonRef_Nodes[i][2])
        elif LinkID[i] < 0:
            Curvatures[i].insert(0,NonRef_Nodes[i][1])
            Curvatures[i].append(Ref_Nodes[i][1])
            Headings[i].insert(0,NonRef_Nodes[i][2])
            Headings[i].append(Ref_Nodes[i][2])
```

```
#Creation of Lists Ends here
#Scaling of the data for calculation:

flip_data = [Latitude, Longitude, Elevation, Slopes, Curvatures, Headings]
scales =     [1e7,          1e7,      1e2,      1e3,     1e6,       1e3]

for i in range(len(LinkID)):
    for data_list, scale in zip(flip_data, scales):
        data_list[i] = list(np.cumsum([float(x) for x in data_list[i]]))
        data_list[i] = [x / scale for x in data_list[i]]
        if int(LinkID[i]) < 0:
            data_list[i] = np.flip(data_list[i]).tolist()

#Slicing Start and Final:

def find_nearest_point(input_points, latitudes, longitudes):
    min_distance = float('inf')
    nearest_point = None
    nearest_index = None
    for i, (lat, lon) in enumerate(zip(latitudes, longitudes)):
        point = (lat, lon)
        distance = haversine(input_points, point)
        if distance < min_distance:
            min_distance = distance
            nearest_point = point
            nearest_index = i
    return nearest_point, nearest_index
if start_lat is not None and start_lon is not None and end_lat is not None and end_lon is not
 None:
    input_points_start = (start_lat, start_lon)
    input_points_end = (end_lat, end_lon)
    start_point, start_index = find_nearest_point(input_points_start, Latitude[0], Longitude[0])
    end_point, end_index = find_nearest_point(input_points_end, Latitude[-1], Longitude[-1])

    #Start point and end point variables can be accessed if needed

#Slicing the data based on the start and end points and slicing operation has been written long
 for understanding: can be made shorter//


    if start_index == len(Latitude[0]) - 1:
        Latitude[0] = Latitude[0][start_index-1:]
        Longitude[0] = Longitude[0][start_index-1:]
        Latitude[0][0] = start_lat
        Longitude[0][0] = start_lon
        Elevation[0] = Elevation[0][start_index-1:]
        Slopes[0] = Slopes[0][start_index-1:]
        Curvatures[0] = Curvatures[0][start_index-1:]
        Headings[0] = Headings[0][start_index-1:]

    elif start_index == 0:
        Latitude[0]=Latitude[0][start_index:]
        Longitude[0]=Longitude[0][start_index:]
        Elevation[0]=Elevation[0][start_index:]
        Slopes[0]=Slopes[0][start_index:]
        Curvatures[0]=Curvatures[0][start_index:]
        Headings[0]=Headings[0][start_index:]

    else:
        Latitude[0]=Latitude[0][start_index:]
        Longitude[0]=Longitude[0][start_index:]
        Elevation[0]=Elevation[0][start_index:]
        Slopes[0]=Slopes[0][start_index:]
        Curvatures[0]=Curvatures[0][start_index:]
        Headings[0]=Headings[0][start_index:]


    if end_index == 0:
        Latitude[-1]= Latitude[-1][:2]
        Longitude[-1]= Longitude[-1][:2]
```

```
            Latitude[-1][-1]= end_lat
            Longitude[-1][-1]= end_lon
            Elevation[-1]= Elevation[-1][:2]
            Slopes[-1]= Slopes[-1][:2]
            Curvatures[-1]= Curvatures[-1][:2]
            Headings[-1]= Headings[-1][:2]

        elif end_index == len(Latitude[-1]) - 1:
            Latitude[-1]= Latitude[-1][:]
            Longitude[-1]= Longitude[-1][:]
            Elevation[-1]= Elevation[-1][:]
            Slopes[-1]= Slopes[-1][:]
            Curvatures[-1]= Curvatures[-1][:]
            Headings[-1]= Headings[-1][:]
        else:
            Latitude[-1]= Latitude[-1][:end_index+1]
            Longitude[-1]= Longitude[-1][:end_index+1]
            Latitude[-1][-1]= end_lat
            Longitude[-1][-1]= end_lon
            Elevation[-1]= Elevation[-1][:end_index+1]
            Slopes[-1]= Slopes[-1][:end_index+1]
            Curvatures[-1]= Curvatures[-1][:end_index+1]
            Headings[-1]= Headings[-1][:end_index+1]

    else:
        print("Start and end points not provided. Skipping slicing operation.")

    #Distance Calculation Functions :

    def haver(point1, point2):

        earth_radius = 6371e3
        lat1, lon1, elev1 = radians(point1[0]), radians(point1[1]), point1[2]
        lat2, lon2, elev2 = radians(point2[0]), radians(point2[1]), point2[2]
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = sin(dlat / 2) * sin(dlat / 2) + cos(lat1) * cos(lat2) * sin(dlon / 2) * sin(dlon / 2)
        c = 2 * asin(sqrt(a))
        horizontal_distance = earth_radius * c
        vertical_distance = abs(elev2 - elev1)
        inclined_distance = sqrt(horizontal_distance ** 2 + vertical_distance ** 2)
        return horizontal_distance, inclined_distance

    def Geo_distance(point1, point2):

        geod = Geod(ellps='WGS84')
        ele1 = point1[2]
        ele2 = point2[2]
        _, _, distance = geod.inv(point1[1], point1[0], point2[1], point2[0])
        inclined_distance = sqrt(distance ** 2 + (ele2 - ele1) ** 2)
        return distance, inclined_distance

    #calculation Execution

    distance_list_hav = []
    distance_list_hav_inclined = []
    distance_list_geo = []
    distance_list_geo_inclined = []
    for i in range(len(Latitude)):
        link_hav = []
        link_hav_inlcined=[]
        link_geo = []

        link_geo_inclined = []
        distance_list_hav.append(link_hav)
        distance_list_geo.append(link_geo)
        distance_list_geo_inclined.append(link_geo_inclined)
        distance_list_hav_inclined.append(link_hav_inlcined)

        for j in range(len(Latitude[i])-1):

            point1 = (Latitude[i][j], Longitude[i][j], Elevation[i][j])
```

```
            point2 = (Latitude[i][j+1], Longitude[i][j+1], Elevation[i][j+1])
            hav,hav_inclined = haver(point1, point2)
            link_hav.append(hav)
            link_hav_inlcined.append(hav_inclined)
            geodesic, geodesic_inclined= Geo_distance(point1, point2)
            link_geo.append(geodesic)
            link_geo_inclined.append(geodesic_inclined)

    #Link Length Calculation below :
    Haversine_distance =[]
    Haversine_distance_inclined = []
    Geodesic_distance = []
    Geodesic_distance_inclined = []

    for i in range(len(Latitude)):
        Haversine_distance.append(sum(distance_list_hav[i]))
        Haversine_distance_inclined.append(sum(distance_list_hav_inclined[i]))
        Geodesic_distance.append(sum(distance_list_geo[i]))
        Geodesic_distance_inclined.append(sum(distance_list_geo_inclined[i]))


    Total_haversine_distance = sum(Haversine_distance)
    Total_haversine_distance_inclined = sum(Haversine_distance_inclined)
    Total_geodesic_distance = sum(Geodesic_distance)
    Total_geodesic_distance_inclined = sum(Geodesic_distance_inclined)

    #Model Attributes

    #Speed Limit

    links_parse = parse("$.response.route[0].leg[0].link[*]")
    speed_limit_parse = parse("$.attributes.APPLICABLE_SPEED_LIMIT[0].APPLICABLE_SPEED_LIMIT")
    speed_limits = []

    for link in links_parse.find(data1):
        speed_limit = speed_limit_parse.find(link.value)
        speed_limits.append(int(speed_limit[0].value) if speed_limit else None)


    speed_limit_updated = speed_limits.copy()

    for i in range(len(speed_limits)):
        # If the current speed limit is None
        if speed_limits[i] is None:
            # If it's the first element, find the next available speed limit
            if i == 0:
                j = 1
                while speed_limits[j] is None and j < len(speed_limits) - 1:
                    j += 1
                speed_limit_updated[i] = speed_limits[j]
            else:
                # Otherwise, use the previous speed limit
                speed_limit_updated[i] = speed_limits[i - 1]

    #Get from the LINKS* Tree:

    links_parse = parse("$.response.route[0].leg[0].link[*]")

    Traffic_Condition = []
    Traffic_Sign = []

    Free_Flow_Speed = []

    Wind_Direction = []
    Wind_Velocity = []

    for i,link in enumerate(links_parse.find(data1)):
        condition_parse = parse("$.attributes.TRAFFIC_SIGN_FCN[0].CONDITION_TYPE")
        condition = condition_parse.find(link.value)
        Traffic_Condition.append(int(condition[0].value) if condition else None)

        sign_parse = parse("$.attributes.TRAFFIC_SIGN_FCN[0].TRAFFIC_SIGN_TYPE")
```

```
        sign = sign_parse.find(link.value)
        Traffic_Sign.append(int(sign[0].value) if sign else None)

        free_flow_speed_parse = parse("$.attributes.TRAFFIC_PATTERN_FCN[0].FREE_FLOW_SPEED")
        free_flow_speed = free_flow_speed_parse.find(link.value)
        Free_Flow_Speed.append(int(free_flow_speed[0].value) if free_flow_speed else
    speed_limit_updated[i])

        wind_direction_parse = parse("$.attributes.ARCHIVED_WEATHER[0].WIND_DIRECTION")
        wind_direction = wind_direction_parse.find(link.value)
        Wind_Direction.append(float(wind_direction[0].value) if wind_direction else None)

        wind_velocity_parse = parse("$.attributes.ARCHIVED_WEATHER[0].WIND_VELOCITY")
        wind_velocity = wind_velocity_parse.find(link.value)
        Wind_Velocity.append(float(wind_velocity[0].value) if wind_velocity else 0)


    #Creating a list of lists as Latitudes, Longitudes, Elevation etc. to match the length


    Speed_parse = dict(zip(LinkID, speed_limit_updated))
    speed_limits_list = [[Speed_parse[link_id]] * len(lat) for link_id, lat in zip(LinkID, Latitude)]
    speed_limits_list_float = [[float(value) / 3.6 for value in sublist] for sublist in
     speed_limits_list]

    traffic_signal_parse = dict(zip(LinkID, Traffic_Condition))
    traffic_signals = [[traffic_signal_parse[link_id]] + [0]*(len(lat)-1) for link_id, lat in zip(
     LinkID, Latitude)]

    Traffic_sign_parse = dict(zip(LinkID, Traffic_Sign))
    traffic_signs = [[Traffic_sign_parse[link_id]] + [0]*(len(lat)-1) for link_id, lat in zip(LinkID,
      Latitude)]

    free_flow_speed_parse = dict(zip(LinkID, Free_Flow_Speed))
    Free_Flow_Speed = [[free_flow_speed_parse[link_id]] * len(lat) for link_id, lat in zip(LinkID,
     Latitude)]
    Free_Flow_Speed_float = [[float(value) / 3.6 for value in sublist] for sublist in Free_Flow_Speed
     ]

    wind_direction_parse = dict(zip(LinkID, Wind_Direction))
    Wind_Direction = [[wind_direction_parse[link_id]] + [0]*(len(lat)-1) for link_id, lat in zip(
     LinkID, Latitude)]

    wind_velocity_parse = dict(zip(LinkID, Wind_Velocity))
    Wind_Velocity = [[wind_velocity_parse[link_id]] + [0]*(len(lat)-1) for link_id, lat in zip(LinkID
     , Latitude)]
    wind_velocity_float = [[float(value) / 3.6 for value in sublist] for sublist in Wind_Velocity]

    # Flattening of Lists for DOC File

    flat_latitudes = [item for i, sublist in enumerate(Latitude) for item in sublist[:-1] if i != len
     (Latitude) - 1] + Latitude[-1]
    flat_longitudes = [item for i, sublist in enumerate(Longitude) for item in sublist[:-1] if i !=
     len(Longitude) - 1] + Longitude[-1]
    flat_elevations = [item for i, sublist in enumerate(Elevation) for item in sublist[:-1] if i !=
     len(Elevation) - 1] + Elevation[-1]
    flat_slopes= [item for i, sublist in enumerate(Slopes) for item in sublist[:-1] if i != len(
     Slopes) - 1] + Slopes[-1]
    flat_curvatures = [item for i, sublist in enumerate(Curvatures) for item in sublist[:-1] if i !=
     len(Curvatures) - 1] + Curvatures[-1]
    flat_headings = [item for i, sublist in enumerate(Headings) for item in sublist[:-1] if i != len(
     Headings) - 1] + Headings[-1]
    flat_speed_limits = [item for i, sublist in enumerate(speed_limits_list_float) for item in
     sublist[:-1] if i != len(speed_limits_list_float) - 1] + speed_limits_list_float[-1]
    flat_free_flow_speed = [item for i, sublist in enumerate(Free_Flow_Speed_float) for item in
     sublist[:-1] if i != len(Free_Flow_Speed_float) - 1] + Free_Flow_Speed_float[-1]
    flat_traffic_signals = [item for i, sublist in enumerate(traffic_signals) for item in sublist
     [:-1] if i != len(traffic_signals) - 1] + traffic_signals[-1]
    flat_traffic_signs = [item for i, sublist in enumerate(traffic_signs) for item in sublist[:-1] if
      i != len(traffic_signs) - 1] + traffic_signs[-1]
    flat_wind_direction = [item for i, sublist in enumerate(Wind_Direction) for item in sublist[:-1]
     if i != len(Wind_Direction) - 1] + Wind_Direction[-1]
```

```
flat_wind_velocity = [item for i, sublist in enumerate(wind_velocity_float) for item in sublist
 [:-1] if i != len(wind_velocity_float) - 1] + wind_velocity_float[-1]

#Sgnals and signs filtering

Traffic_signal =[flat_traffic_signals[i] == 16 for i in range(len(flat_traffic_signals))]
Stop_sign=[flat_traffic_signs[i] == 20 for i in range(len(flat_traffic_signs))]
Yield_sign=[flat_traffic_signs[i] == 42 for i in range(len(flat_traffic_signs))]
pedestrian_crossing=[flat_traffic_signs[i] == 41 for i in range(len(flat_traffic_signs))]

print("Extracting the attribues from the JSON file is successful")



#Distance between Consecutive Points after flattening the lists:
def calculate_distances(points):
    wgs84_geod = Geod(ellps='WGS84')
    distances = []
    inclined_distances = []

    for i in range(len(points)-1):
        lon1, lat1, ele1 = points[i]
        lon2, lat2, ele2 = points[i+1]

        # Calculate horizontal distance
        az12, az21, dist = wgs84_geod.inv(lon1, lat1, lon2, lat2)
        distances.append(dist)

        # Calculate inclined distance
        ele_diff = abs(ele2 - ele1)
        inclined_dist = sqrt(dist**2 + ele_diff**2)
        inclined_distances.append(inclined_dist)

    return distances, inclined_distances

print("Distance Calculation Successful")

#CSV FILE Creation Section :


flat_distance, flat_inclined_distance = calculate_distances(list(zip(flat_longitudes,
 flat_latitudes, flat_elevations)))
flat_distance.insert(0, 0)
flat_inclined_distance.insert(0, 0)
distance_model = np.cumsum(flat_inclined_distance)

check_distance=  np.round(flat_distance,2)
check_distance = sum(check_distance)
Total_flat_distance = sum(flat_distance)
Total_flat_inclined_distance = sum(flat_inclined_distance)


#CSV FILE and Folder Creation Section
csv_files_directory = os.path.join(output_directory, 'CSV_Files')
os.makedirs(csv_files_directory, exist_ok=True)
base_name = os.path.splitext(os.path.basename(filepath))[0]
output_file_path = os.path.join(csv_files_directory, base_name + '.csv')

Test = pd.DataFrame({'Linklenghth': link_length_Match,   'Geodesic Distance': Geodesic_distance,
 'Geodesic Distance Inclined': Geodesic_distance_inclined})
Test.to_csv('Test.csv', index=False)


Model = pd.DataFrame({'Latitude': flat_latitudes, 'Longitude': flat_longitudes, 'Elevation':
 flat_elevations, 'Slopes': flat_slopes, 'Curvatures': flat_curvatures, 'Headings': flat_headings
 , 'Distance': distance_model,
                    'Speed Limits': flat_speed_limits, 'free_flow_speed': flat_free_flow_speed,'
 Traffic Signals': Traffic_signal, 'Stop Sign': Stop_sign, 'Yield Sign': Yield_sign, 'Pedestrian
 Crossing': pedestrian_crossing, 'Wind Direction': flat_wind_direction, 'Wind Velocity':
 flat_wind_velocity})
Model.to_csv(output_file_path, index=False)
print(f"Model File Saved")
```

```
    #Printing Function/ creating Report :
    reports_directory = os.path.join(output_directory, 'Reports')
    os.makedirs(reports_directory, exist_ok=True)

    with open(os.path.join(reports_directory, base_name+ 'Distance Report.txt'), 'w') as f:
        # print("Distance from HERE MAPS:", Distance_HERE)
        print(f"\n HAVERSINE:{Total_haversine_distance} Difference: {Total_haversine_distance -
     Distance_HERE}\n Error: {((Total_haversine_distance - Distance_HERE)/Distance_HERE)*100}%", file
     =f)
        print(f"\n HAVERSINE Inclined:{Total_haversine_distance_inclined} Difference: {
     Total_haversine_distance_inclined - Distance_HERE}\n Error: {((Total_haversine_distance_inclined
      - Distance_HERE)/Distance_HERE)*100}%", file=f)
        # # print(f"\n Geodesic Distance:{Total_geodesic_distance} Difference: {
     Total_geodesic_distance - Distance_HERE}\n Error: {((Total_geodesic_distance - Distance_HERE)/
     Distance_HERE)*100}%")
        # # print(f"\n Geodesic Distance Inclined:{Total_geodesic_distance_inclined} Difference: {
     Total_geodesic_distance_inclined - Distance_HERE}\n Error: {((Total_geodesic_distance_inclined -
      Distance_HERE)/Distance_HERE)*100}%")
        print(f"\n GEODESIC:{Total_flat_distance} Difference: {Total_flat_distance - Distance_HERE}\n
      Error: {((Total_flat_distance - Distance_HERE)/Distance_HERE)*100}%", file=f)
        print(f"\n GEODESIC Inclined:{Total_flat_inclined_distance} Difference: {
     Total_flat_inclined_distance - Distance_HERE}\n Error: {((Total_flat_inclined_distance -
     Distance_HERE)/Distance_HERE)*100}%", file=f)
        # print(f"\n Geodesic Distance:{total_geo_flat} Difference: {total_geo_flat - Distance_HERE}\
     n Error: {((total_geo_flat - Distance_HERE)/Distance_HERE)*100}%")
        # print(f"\n rounded distance:{check_distance} Difference: {check_distance - Distance_HERE}\n
      Error: {((check_distance - Distance_HERE)/Distance_HERE)*100}%")


    # #Plotting the Graphs

    # plt.plot(flat_longitudes, flat_latitudes)
    # # # plt.plot(np.cumsum(dist_vinc_int),interpol_HPZ,  'x-', label='interpolated HPZ')
    # # plt.axes('equal')
    # plt.xlabel('Longitudes')
    # plt.ylabel('Latitudes')
    # # plt.legend()
    # plt.title('Vehicle Path')
    # plt.savefig(os.path.join(reports_directory, base_name + 'Vehicle Path.png'))
    # # plt.show()
    pass

# The Program Execution Starts from here : Incase if you are using json file directly, Comment this
     function
def process_files_in_directory(directory, output_directory, extension=".txt"):
    for filename in os.listdir(directory):
        if filename.endswith(extension):
            file_path = os.path.join(directory, filename)
            try:
                print(f"Processing file: {filename}")
                process_file(file_path, output_directory)  # Call your function here
            except Exception as e:
                print(f"Failed to process file: {filename}. Error: {str(e)}")

input_directory = "C:/path/to/folder" # Specify the input txt files folder here
output_directory = input_directory # Specify the output folder by default it will be inside the input
      directory
process_files_in_directory(input_directory, output_directory)
```

Methodology and Application

The methodology and the application of the code as follows, **Methodology:**

- **Data Collection**: The script collects GPS traces (latitude, longitude, and optionally elevation) from a CSV file.

- **Distance Calculation**: It calculates the distance between consecutive GPS points using Haversine and Vincenty formulas.
- **Road Data Extraction**: It uses HERE Maps API to extract road attributes such as speed limit, slope, and curvature based on the GPS traces.
- **DOC Model Generation**: The extracted data is compiled into a deterministic operating cycle (DOC) model in CSV format.

Applications:

- **Residual Range Estimation**: The DOC model can be used to estimate the remaining travel distance with the available energy in the battery or fuel.
- **Energy Consumption Analysis**: Analyzes the energy consumed by the vehicle based on accurate distance estimation and road conditions.
- **Simulation and Testing**: Provides detailed road conditions and vehicle dynamics for simulation and testing purposes.