EMBRY-RIDDLE
Aeronautical University
SCHOLARLY COMMONS

Doctoral Dissertations and Master's Theses

Winter 12-7-2023

Comparison of Methods for Estimating Instantaneous Turn Radius of Ackermann Steering Vehicles

Kenneth Gennaro Stutts
Embry-Riddle Aeronautical University, stuttsk@my.erau.edu

Follow this and additional works at: https://commons.erau.edu/edt

 Part of the Navigation, Guidance, Control, and Dynamics Commons, and the Other Mechanical Engineering Commons

Scholarly Commons Citation

Stutts, Kenneth Gennaro, "Comparison of Methods for Estimating Instantaneous Turn Radius of Ackermann Steering Vehicles" (2023). *Doctoral Dissertations and Master's Theses*. 781.
https://commons.erau.edu/edt/781

This Thesis - Open Access is brought to you for free and open access by Scholarly Commons. It has been accepted for inclusion in Doctoral Dissertations and Master's Theses by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu.

Comparison of Methods for Estimating Instantaneous Turn Radius of Ackermann Steering Vehicles

by

Kenneth Stutts

A Thesis Submitted to the College of Engineering Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mechanical Engineering

Embry-Riddle Aeronautical University
Daytona Beach, Florida
December 2023

Comparison of Methods for Estimating Instantaneous Turn Radius of Ackermann Steering Vehicles

by

Kenneth Stutts

This thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dr. Marc Compere, Associate Professor, Daytona Beach Campus, and Thesis Committee Members Dr. Shuzhen Luo, Assistant Professor, Daytona Beach Campus, and Dr. Christopher Hockley, Assistant Professor, Daytona Beach Campus, and has been approved by the Thesis Committee. It was submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Thesis Review Committee:

_____
Marc Compere, Ph.D.
Committee Chair

_____
Shuzhen Luo, Ph.D.
Committee Member

_____
Christopher Hockley, Ph.D.
Committee Member

_____
Jean-Michel Dhainaut, Ph.D.
Graduate Program Coordinator,
Mechanical Engineering

_____
Patrick Currier, Ph.D.
Department Chair,
Mechanical Engineering

_____
Jim Gregory, Ph.D.
Dean, College of Engineering

_____
Date

Acknowledgements

I would like to thank my advisor, Dr. Compere, for keeping me motivated through my thesis and my entire master's degree.

I would also like to thank my friends and family for pushing me to be the best I could be and being there for me when times got tough. This journey and degree would not have been possible without your help and encouragement.

Abstract

Researcher: Kenneth Stutts

Title: Comparison of Methods for Estimating Instantaneous Turn Radius of Ackermann Steering Vehicles

Institution: Embry-Riddle Aeronautical University

Degree: Master of Science in Mechanical Engineering

Year: 2023

The instantaneous turn radius of an Ackermann steering vehicle is the distance to a point in space about which the vehicle will travel in an arc during a turn. There are at least six ways to estimate instantaneous turn radius and each method uses different inputs and has distinct advantages and disadvantages. In this thesis, six different methods will be used to estimate the instantaneous turn radius of a vehicle traveling on a closed circuit. This testing will clarify similarities and differences between the methods. This thesis clarifies which method will be the most appropriate for a given set of available inputs. The testing will be conducted with commonly available sensors. The research in this thesis will allow developers to choose the best method for their sensor suite.

Keywords: Instantaneous Turn Radius, Ackermann Steering, GPS, Instrumentation, IMU, Encoder

Table of Contents

List of Figures

Page

Figure

List of Tables

Page

Table

Chapter I: Introduction

Instantaneous turn radius, R, is used extensively in vehicle dynamics literature. Turn radius is a critical part of Gillespie's graphical lateral acceleration derivation (Gillespie, 1992) and is a single parameter that characterizes the degree to which a vehicle is turning at any time during driving. Estimating lateral acceleration from this classical derivation, $a_{lat} = v^2/R$, is a common vehicle dynamics question. Also, steering system design specifically uses turn radius as a design criterion while designing linkage and the turning mechanisms.

1.1 Thesis Statement

The purpose of this study is to determine which of the methods of estimating ITR is the most applicable depending on available sensors. The thesis statement is as follows. At least 6 different methods exist for estimating turn radius for Ackermann Steering vehicles depending on available sensors. This thesis statement can be divided into three sections: (a) description of an Ackermann steered vehicle, (b) the six methods for estimating turn radius, and (c) the required sensors for each of the estimation methods. The research in this thesis will allow developers to choose the best method for their specific sensor suite and vehicle setup. The testing will be conducted with commonly available sensors in order to highlight the accessibility of the methods to all types of researchers. While this paper is focused on Ackermann steering vehicles exclusively, there are methods that would be applicable to other types of vehicles.

1.2 Turn Radius and Curvature

For straight-line driving the turn radius is infinite, so instead of R, the path curvature is preferred, $\rho = 1/R$. Sometimes Greek letter kappa is also frequently used to express curvature, $\kappa = 1/R$. The lower-case Greek letter rho, ρ, will be used in this thesis. Using curvature rather than turn radius, Figure 1, illustrates why curvature is used in simulation using an example of a typical left-right turning sequence. Note that both R and ρ are signed quantities to indicate left or right hand turning, assuming a right-handed SAE coordinate frame.

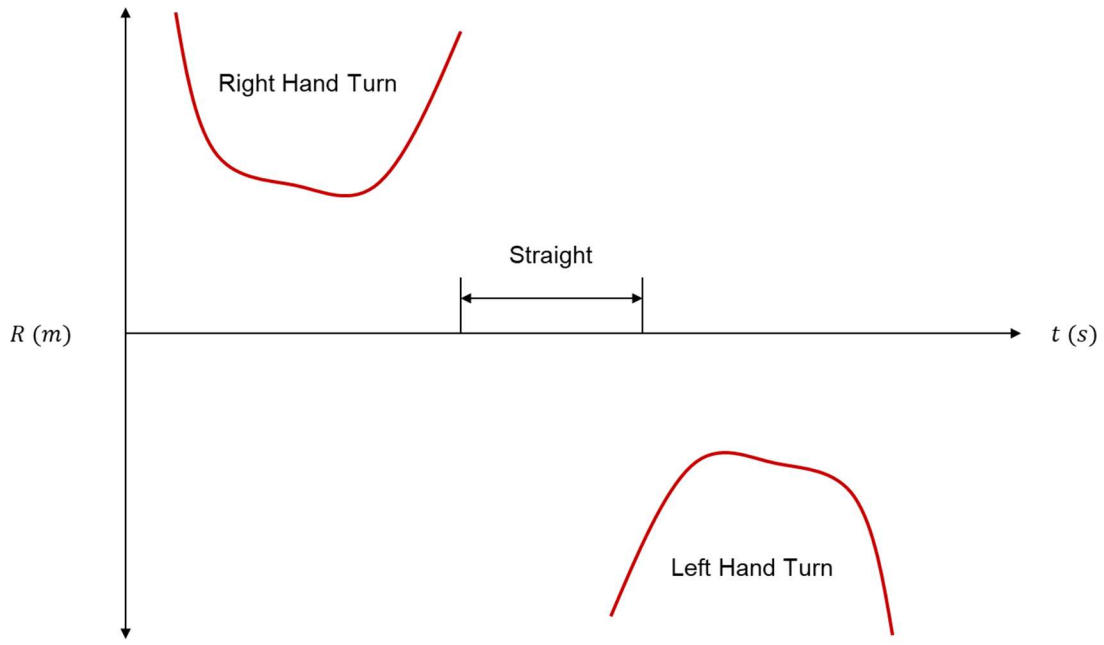

Figure 1.8: Turn radius, R, for a RH-straight-LH turn sequence

Notice the turn radius becomes infinite while driving straight. For this reason, sometimes curvature is used to express turn radius because during straight driving curvature is zero, as shown in Figure 1.9.

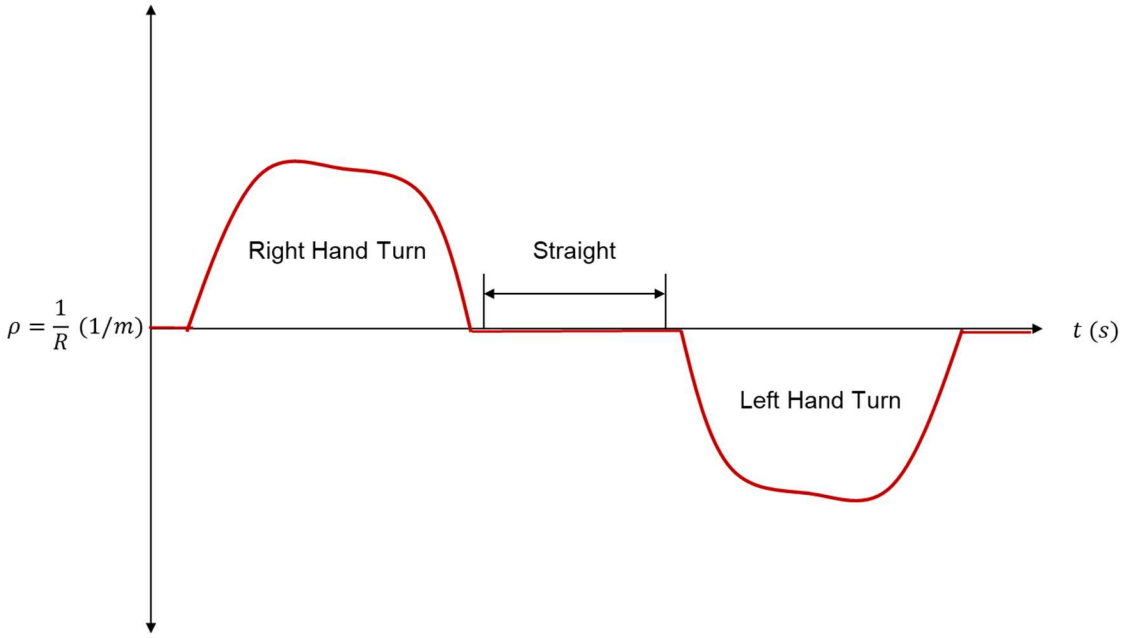

Figure 1.9: Curvature, ρ, for a RH-straight-LH turn sequence

Since R is seen so often in vehicle dynamics literature, this research aims to understand different ways of estimating turn radius and the various pros and cons of each method.

1.3 Importance of Estimating the Real-Time Turn Radius

Estimating the real-time turn radius of a vehicle is a key pillar of vehicle dynamics. The turn radius of a vehicle is important in fields like motorsports, autonomous vehicles, and vehicle maneuverability just to name a few examples. In motorsports generally the fastest and shortest path is the path with the minimum curvature possible. By comparing the real-time turn radius during different laps of a circuit an optimal path can be found. For autonomous vehicles, knowing the real-time turn radius of the vehicle will allow for better path planning and efficient traversal through complex environments. The maneuverability

of a vehicle is explicitly linked to the minimum turn radius of a vehicle and the majority of roadways are designed with a required minimum turn radius.

Alternatively knowing the real time turn radius of a vehicle also allows for the determination of many other factors of the vehicle. One of the most common ways to determine difficult to obtain factors of vehicles like the understeer gradient, is to drive a fixed radius turn with a set speed and steer angle. With these parameters the method used in this paper can then be reconfigured to find these difficult to obtain factors. Likewise, if you know the turn radius and forward velocity of the vehicle you can also find the lateral acceleration and yaw rate of the vehicle.

1.4 Ackermann Steering Vehicles

Ackermann Steering is a method of steering a vehicle that turns the front tires of the vehicle at different rates in order to keep the tires from slipping during truing (Mitchell 2006), see Figure 1.3. The inside tire turns at a greater angle than the outside tire so that all tires are perpendicular to the turn center. Figure 1.4 shows the difference the increased angle Ackermann steering geometry provides.

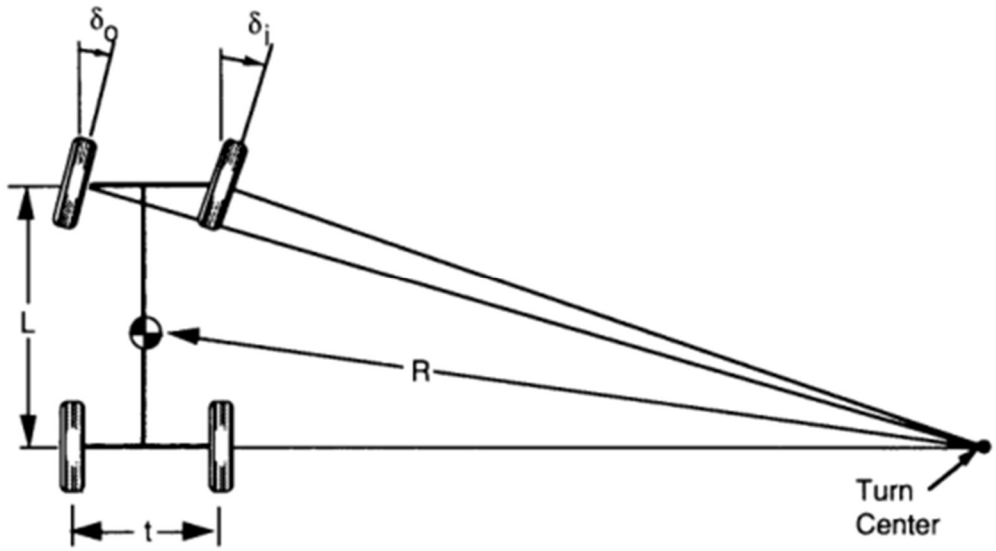

Figure 1.10 Geometry of an Ackermann Steering Vehicle during Turning, Gillespie (1992)

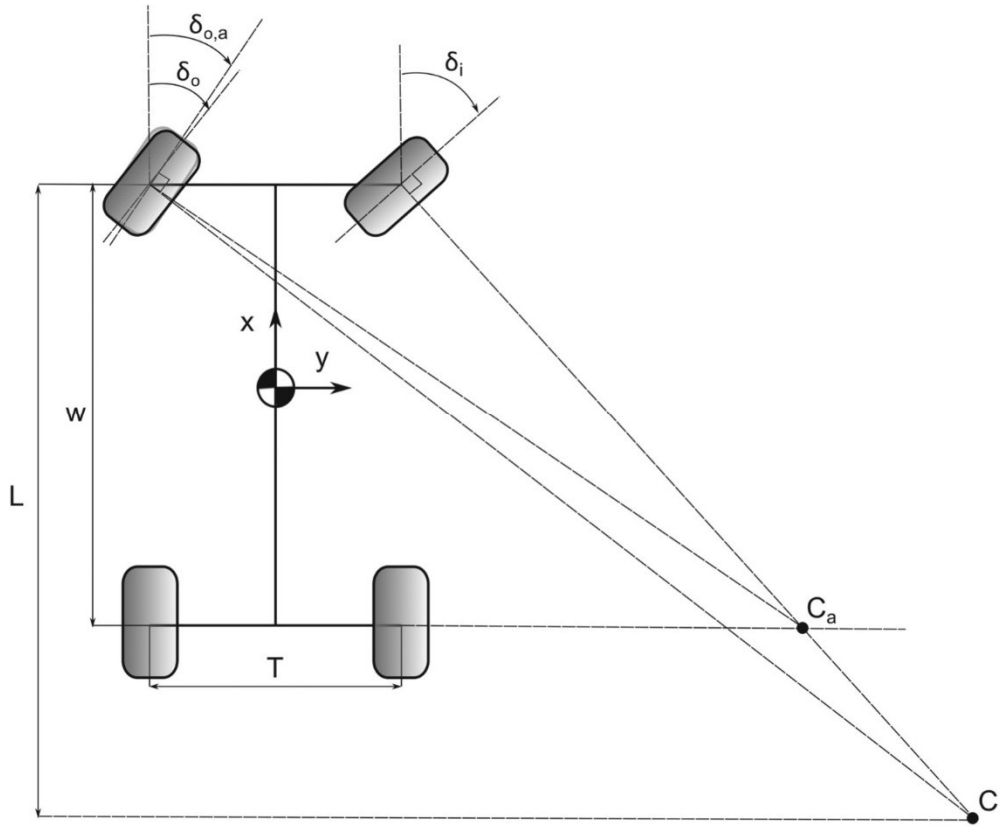

Figure 1.11: Actual steering vs. Ackermann steering, Veneri (2021)

1.5 Methods for Estimating Turn Radius

The 6 methods for estimating turn radius can be divided into 3 categories based upon the sensor required for their operation. These categories are the IMU Methods, Steer-Angle Methods, and the GPS Methods. The IMU Methods include the Yaw Rate Method and the Lateral Acceleration Method. These methods rely on data collected from an Inertial Measurement Unit (IMU) to operate. They specifically require the yaw rate and lateral acceleration of the vehicle. The Steer-Angle Methods are the Low-Speed Ackermann Steer Method and High-Speed Ackermann Steer Method. Both methods rely on the steer angle of the vehicle to operate. The GPS Methods include The Circle Method and the Cubic Spline Interpolation Method. These methods require GPS location points in order to operate.

Method Number	Method Name	Sensors Required
1	Yaw Rate Method	IMU
2	Lateral Acceleration Method	IMU
3	Low Speed Ackermann Steering Method	Steer Angle Sensor
4	High Speed Ackermann Steering Method	Steer Angle Sensor and IMU
5	Circle Method	GPS
6	Cubic Spline Interpolation Method	GPS

Table 1.1: Sensors required for each method.

1.5.1 Yaw Rate Method Derivation

The Yaw Rate Method can be described as a general turn radius estimation method for a vehicle with nonzero velocity. The yaw rate method is derived by an overhead view of a simple 2D vehicle turning in the plane (Figure 1.5).

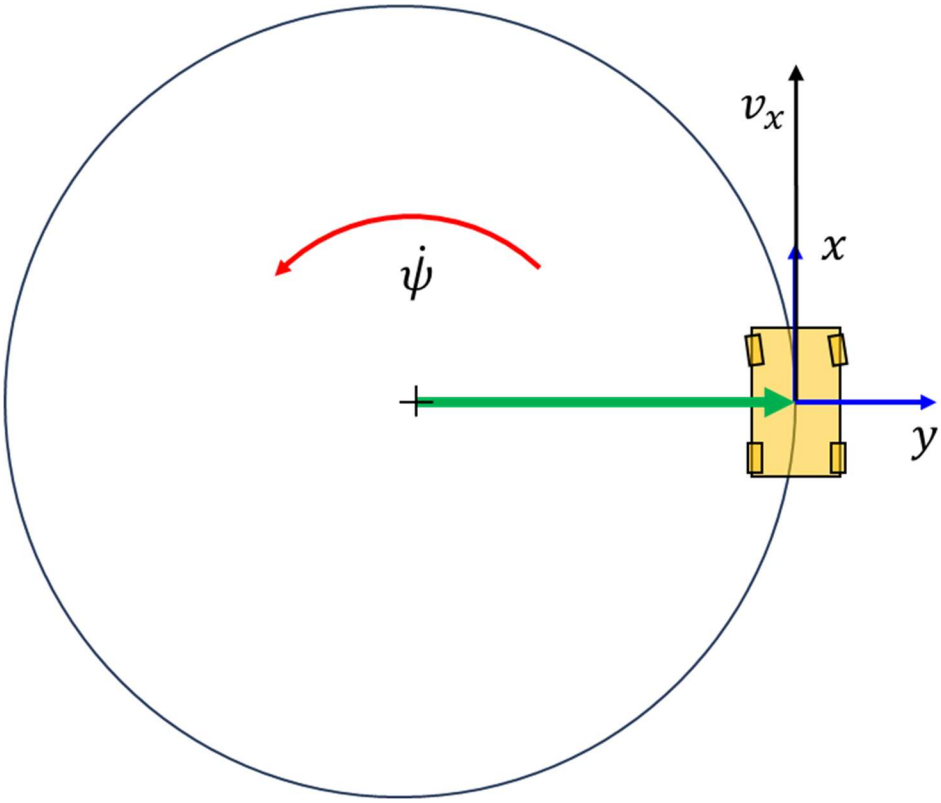

Figure 1.12: Top view of simple 2D vehicle turning.

By inspection, it is clear that the turning relationship is that the forward velocity is equal to the turn radius times the yaw rate or $v_x = R \cdot \dot{\psi}$. Solving for R, yields Equation 1.

$$R = \frac{v_x}{\dot{\psi}} \tag{1}$$

1.5.2 Lateral Acceleration Method Derivation

The Lateral Acceleration Method is described as a constant speed method used for estimating a changing radius turn at constant velocity Gillespie (1992). Lateral acceleration

is equal to forward velocity squared divided by the turn radius or $a_y = \frac{{v_x}^2}{R}$. By solving for R Equation 2 is formed.

$$R = \frac{{v_x}^2}{a_y} \tag{2}$$

1.5.3 Low-Speed Ackermann Steering Method Derivation

The Low-Speed Ackermann Method is described by Gillespie as low speed turning and he likens it to parking lot maneuvers. During these low speed turns the tires develop little to no lateral forces with the center of the turn being projected from the rear axle. Likewise, the perpendicular from each of the front wheels should pass through the same point (the center of turn) Gillespie (1992). We can see from Figure 1.6 that turn radius is a factor in the steer angle of a vehicle.

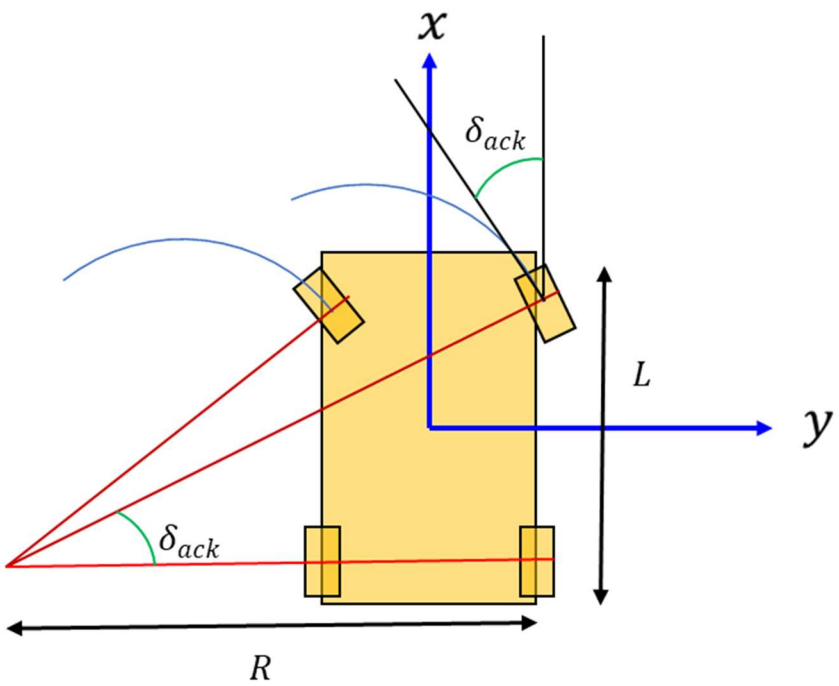

Figure 1.13: Depiction of Ackermann Steering

It is clear to see that $\tan(\delta_{ack}) = \frac{L}{R}$. Solving for R yields Equation 3 assuming small angles.

$$R = \frac{L}{\delta_{ack}} \tag{3}$$

1.5.4 High-Speed Ackermann Steering Method Derivation

The High-Speed Ackermann Method is a modified version of the Low-Speed Ackermann Method that takes the velocity of the vehicle and the understeer gradient into account which allows higher operating speeds than the Low-Speed Ackermann Method. Gillespie describes that the understeer gradient "consists of two terms, each of which is the ratio of the load on the axle (front or rear) to the cornering stiffness of the tires on the axle." Gillespie (1992). The Bicycle Model Figure 1.7 accurately shows all these parameters.

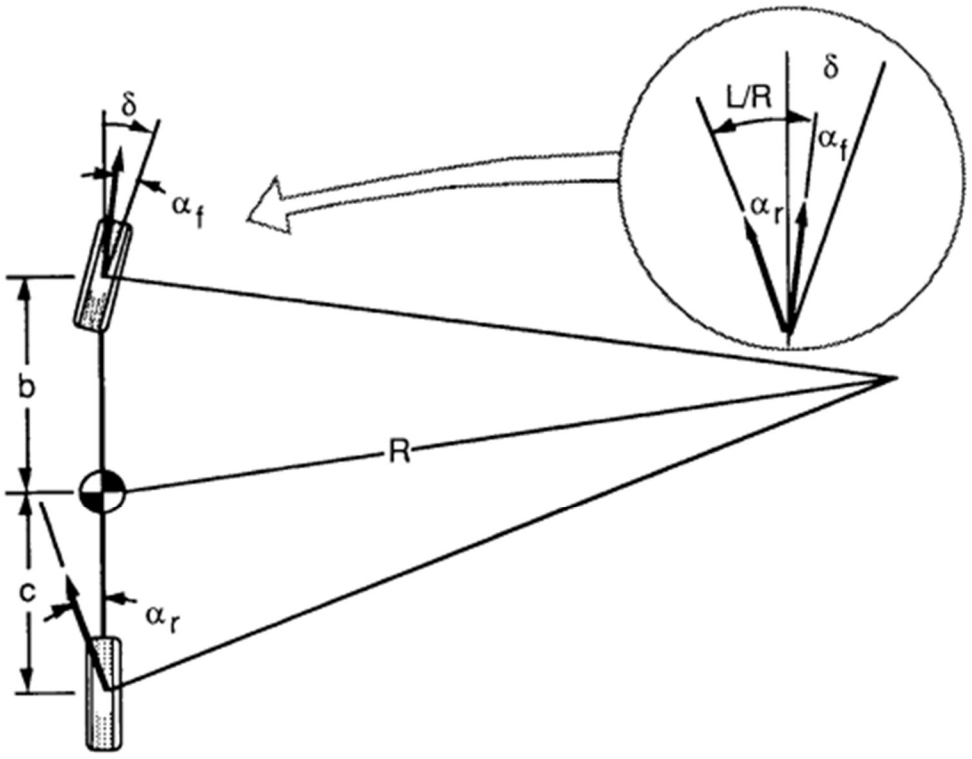

Figure 1.14: Bicycle Model

By examining Figure 1.6 we can see that tire angle δ_{ack} is explicitly connected to turn radius and from that we get $tan(\delta_{ack}) = \frac{b+c}{R} + a_f - a_r$. since the understeer gradient K_{US} is the ratio of the load on the axles (W_f and W_r) to the cornering stiffness of the tires (C_{af} and C_{ar}) on the axle (Gillespie, 1992) the following relationship can be formed $a_f - a_r = \frac{W_f {v_x}^2}{C_{af} \cdot g \cdot R} - \frac{W_r {v_x}^2}{C_{ar} \cdot g \cdot R} = \left(\frac{W_f}{C_{af}} - \frac{W_r}{C_{ar}}\right) \cdot \frac{{v_x}^2}{g \cdot R} = K_{US} \cdot a_y$ From this relationship we can see that the understeer gradient of the vehicle is implicitly related to the slip angle of the tires (a_f and a_r) and that the lateral acceleration of the vehicle multiplied by the understeer gradient of the vehicle is equal to the difference between the front and rear slip angles; $tan(\delta_{ack}) = \frac{b+c}{R} + K_{US} \cdot a_y$. Looking back at Equation 2 we can deduce that $tan(\delta_{ack}) = \frac{b+c}{R} + K_{us} \cdot \frac{{v_x}^2}{R}$. Finally, we solve for R, combine the lengths from the wheels to the center of gravity (b and c) into wheelbase, and assume small angles to get Equation 4.

$$R = \frac{L + K_{us} \cdot \boldsymbol{v_x^2}}{\delta_{ack}} \tag{4}$$

It can be observed that should the Understeer gradient of the test vehicle be equal to 0 then the High-Speed Ackermann and Low Speed Ackermann methods will be equal to each other. It can also be observed that during low-speed maneuvers the 2 methods will be roughly equivalent; regardless of the understeer gradient as shown by Figure 1.8.

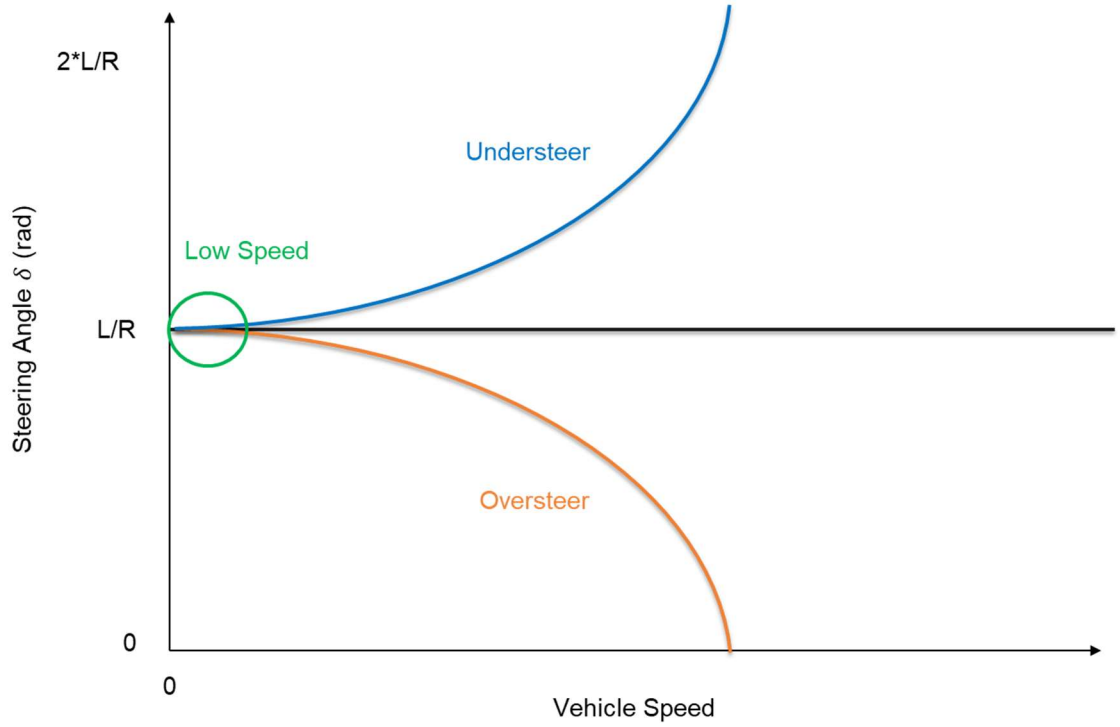

Figure 1.15: Steer Angle vs Vehicle Speed

1.5.5 Circle Method Derivation

The properties of a circle are defined by Euclid's Elements and shows us that from 3 or more non collinear points, a unique circle can be formed that passes through all those points. If the given points are locations that the vehicle has passed through, then the radius of the resulting circle is the turn radius of the vehicle as it traveled through those points (Figure 1.9).

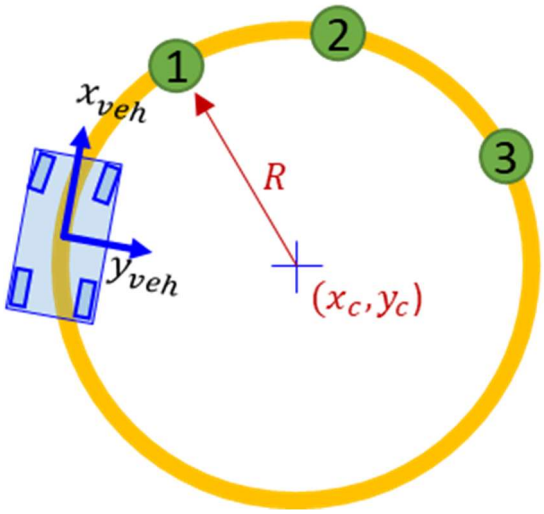

Figure 1.16: Circle Method

In order to find these circles, the squares of the x and y coordinates of first and second points must be subtracted from each other than that sum added together as shown in Equation 5. The same must be done with the first and third points as seen in Equation 6. A matrix is then formed from twice the difference between the first and third and first and second coordinates like in Equation 7. Equation 8 provides the center of the circle using the transposed matrix form Equation 7 multiplied by a matrix of the results of Equations 5 and 6. Finally, Equation 9 results in the radius of the circle.

$$k_{12} = x_2^2 - x_1^2 + y_2^2 - y_1^2 \tag{5}$$

$$k_{13} = x_3^2 - x_1^2 + y_3^2 - y_1^2 \tag{6}$$

$$A = \begin{bmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = A^{-1} \begin{bmatrix} k_{12} \\ k_{13} \end{bmatrix} \tag{8}$$

$$R = \sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2} \tag{9}$$

1.5.6 Cubic Spline Interpolation Method Derivation

The Cubic Spline Interpolation Method is similar to the Circle Method in that it uses GPS data points to find the turn radius of the vehicle, but instead of using a circle, cubic spline interpolation is used to connect the points and find the curvature of the arcs created. “The fundamental idea behind cubic spline interpolation is based on the engineer’s tool used to draw smooth curves through a number of points.” McKinley (1998).

For the Cubic Spline Interpolation Method there are 4 different methods for interpolating between the GPS data points. Those 4 methods are Piecewise Cubic Hermite Interpolating Polynomial, Cubic spline data interpolation, Modified Akima piecewise cubic Hermite interpolation, and Cubic Smoothing Spline. Piecewise Cubic Hermite Interpolating Polynomial is the most direct method resulting in nearly point to point interpolation by imposing local monotonicity in each interval as it computes the derivatives Fritsch (1985). Cubic spline data interpolation provides a very smooth interpolation by imposing the constraint of continuous second derivatives while computing the derivatives. Modified Akima piecewise cubic Hermite interpolation is an update made in 2017 to Akimas original formula published in his 1970 paper “A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures”. Makima specifically updates Akima's formula to improve the edge case of equal side slopes and to eliminate overshoot from consistent data between more than 2 consecutive nodes. The result is a middle ground between the Piecewise Cubic Hermite Interpolating Polynomial and Cubic spline data interpolation functions. (Moler, 2019) Figure 1.10 depicts the differences between Piecewise Cubic Hermite Interpolating Polynomial, Cubic spline data interpolation and Modernized Akima.

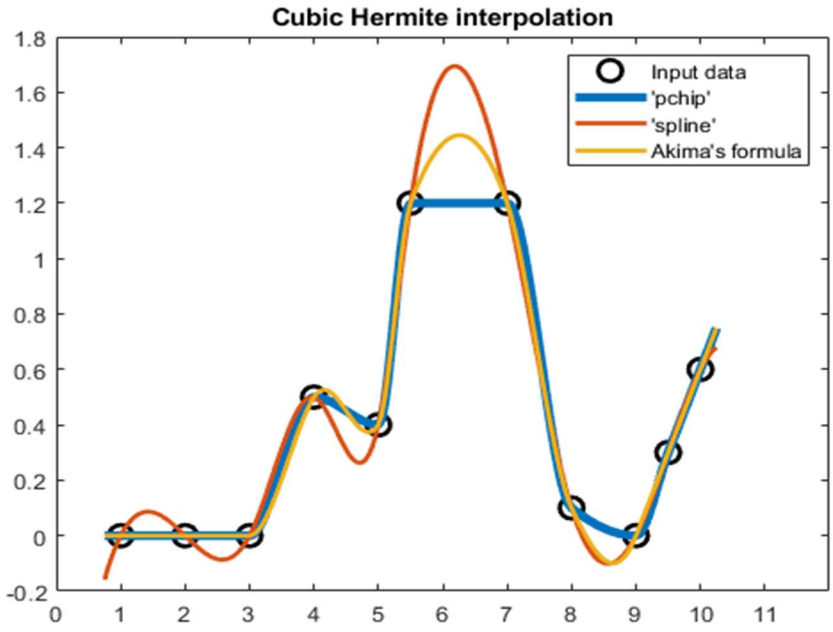

Figure 1.17: Differences between Spline Methods (Moler, 2019)

The Cubic Smoothing Spline function weights each data point with an error measure that is balanced with a roughness measure that is a cost function that minimizes curvature. Finding the right balance between smoothness of the curve and its closeness to the data points Pollock (1993). Equation 10 is the formula devised by Carl De Boor in the textbook A Practical Guide to Splines.

$$\underbrace{p\sum_{j=1}^{n} w_j|y_j - f(x_j)|^2}_{\text{error measure}} \quad + \quad \underbrace{(1-p)\int \lambda(t)|D^2 f(t)|^2 dt}_{\text{roughness measure}} \qquad (10)$$

The Cubic Smoothing Spline function has a fixed smoothing value p that is between 1 and 0. The higher this smoothing value is the closer the spline will fit to the data set as seen in Figure 1.11. The fit of a smoothing spline is not an exact fit to all data points

unless the smoothing value is equal to 1 because it is constrained by the roughness measure to smooth the spline. A Cubic Smoothing Spline can be seen as a sort of filter placed upon a dataset Peters (1981).

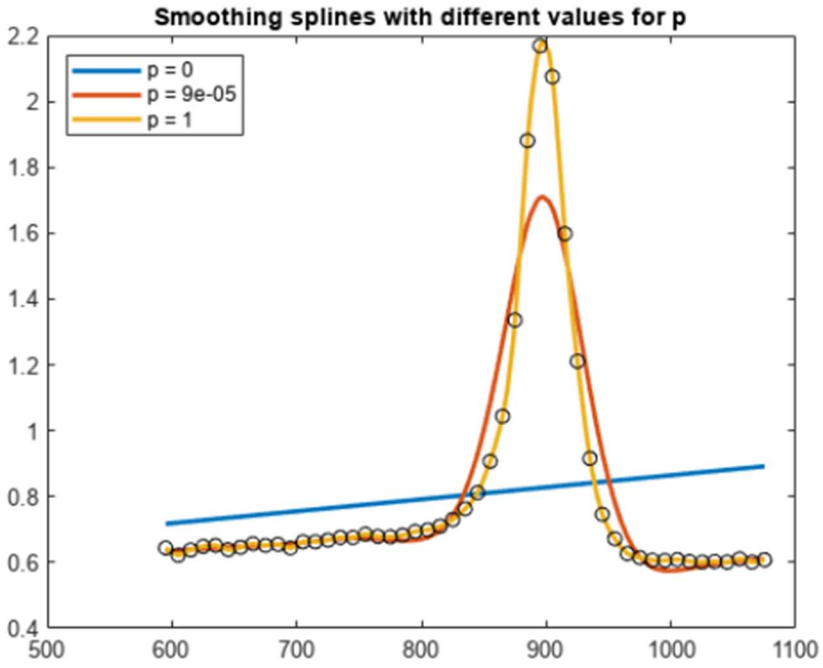

Figure 1.11: Smoothing Splines with different values of p

1.6 Usage of Methods in Literature

In this section, the thesis will explore research and studies relevant to the methods used in the paper. The following sections will cover each of the 3 method groups.

1.6.1 IMU Methods

The IMU Methods are by far the simplest of the methods with only 2 inputs for each method. This makes them prime candidates for use where computational complexity is at a premium and as such are the easiest to use in modeling and simulation. These methods rely on the velocity of the vehicle being nonzero.

A paper by VBOX Automotive and Racelogic (2014) emphasizes the importance of being able to measure the turn radius of the vehicles at multiple points on the vehicle. The example given for this scenario is with a heavy vehicle in order to show that tail swing is not largely different compared to the front of the vehicle. Jagelčák (2022) uses the Lateral Acceleration Method to determine the turn radius on cargo vehicles in order to study long average accelerations (minimum 1 s duration time) acting on a vehicle and cargo when cornering. Wang (2013) used the accelerometers built into smartphones to sense the forward velocity and lateral acceleration regardless of the smartphone's orientation. From this the turn radius can be determined as well as the position of the smartphone in the vehicle. These papers show other researchers using the IMU Methods to find the turn radius of vehicles as well as displaying the many applications of turn radius.

In Park (2018) the Yaw Rate Method is used to estimate yaw rate from a targeted ITR and constant velocity for use in a pure pursuit tracking algorithm. In Xu (2022) the Lateral Acceleration Method is used to estimate lateral acceleration from radius of curvature and velocity to help predict vehicle rollover. Both of these papers are applicable in this thesis because they show that these formulae are used in the literature with turn radius or curvature prescribed as a constant in order to calculate other variables, particularly yaw rate and lateral acceleration.

1.6.2 Steer-Angle Methods

The Steer-Angle methods are the only methods that take the steering angle of the vehicle into account when estimating ITR. This allows these methods to estimate the ITR of stationary vehicles. In Gitay (2018) the Low-Speed Ackermann Method is used to estimate turn radius from the steering geometry as part of the optimization of the steering

system of their Formula Student car. In Pompakdee (2016) the High-Speed Ackermann Method is used to determine understeer gradient in a steady state turn of fixed radius. In Broggi (2012) the inverse of the High-Speed Ackermann Method is used to estimate the relationship between curvature and vehicle speed. These papers show how the Steer-Angle Methods can be used to find values that are typically hard to estimate like understeer gradient.

1.6.3 GPS Methods

The biggest flaw in the GPS Methods is the fact that they can only estimate the turn radius of the vehicle a few time steps in the past. This is due to the fact that they require multiple points to accurately estimate ITR. Figure 1.12 shows that in order to find the curvature of position 1 you must have both positions 2 and 3 which are in the future compared to position 1.

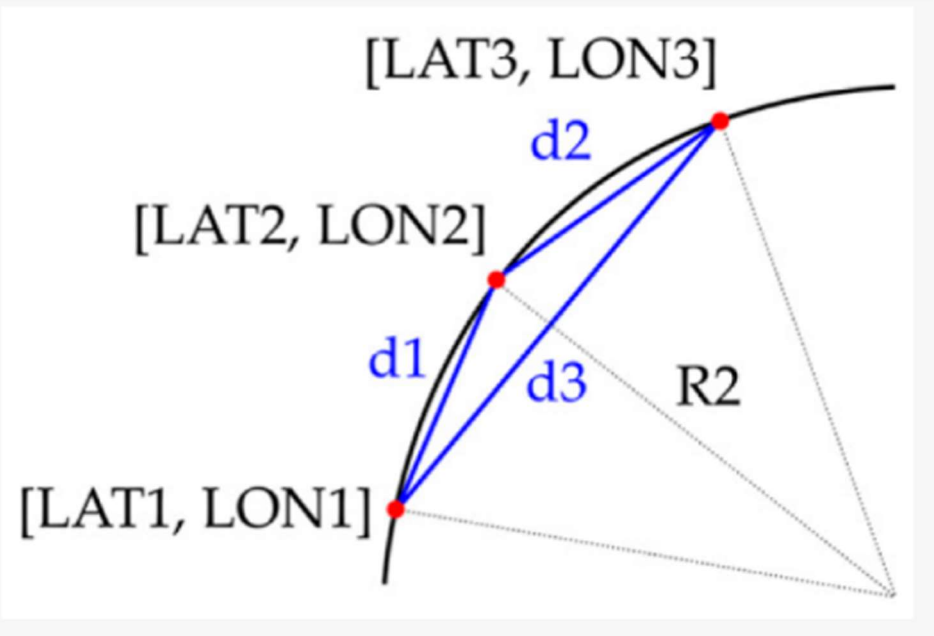

Figure 1.12: Turning radius R2 from circumscribed circle of a triangle defined by three points on curve. Jagelčák (2022)

This is shown in Jagelčák (2022) where a 2 second interval is used with the Circle Method. The 2 second interval covers 400 data points with the 3 points needed to for the circle being the 1st data point, the 200th data point, and the 399th data point. This makes every result 2 seconds delayed to the Lateral Acceleration Method also used in their paper.

Compere and Tucker (2022) logged the GPS data points of a 2016 Porsche Cayman GT4 around the MidOhio racetrack see Figure 1.13. These datapoints were then connected using cubic spline interpolation in order to find the ITR around the track. This is relevant as a proof of concept for the Cubic Spline Interpolation Method.

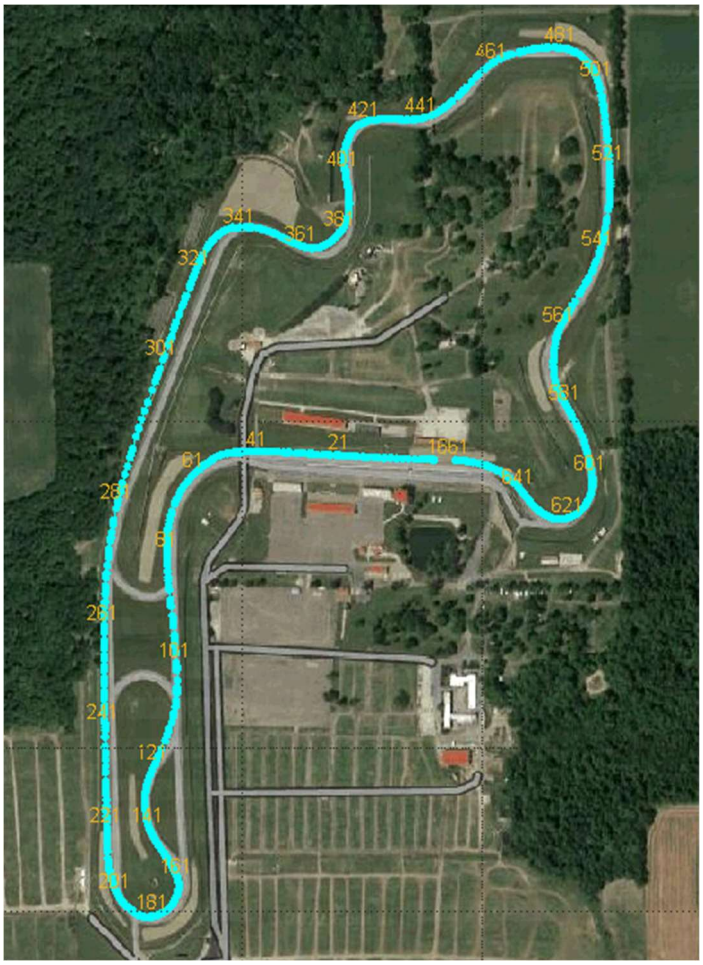

Figure 1.13 Implementation of the Cubic Spline Interpolation Method. (Compere, Tucker, 2022).

1.7 Dubins Path

Dubins Path developed by L.E. Dubins is the shortest arc connecting 2 points on a 2D plane. Dubins Path uses the minimum radius in order to achieve the shortest path between points. A Dubins Vehicle is described by Chen (2020) as "A typical nonholonomic vehicle that moves only forward at a constant speed with a minimum turning radius." This method of pathing is exceedingly used by autonomous vehicles due to the simplicity of the resulting paths. An example of Dubins Paths can be seen in Figure 1.14 where the goal is to start at P1 move through P2 and P3 to reach P4. A second path is shown where the goal is to start at P1 and move through Q1 and Q2 to reach P4. The minimum turn radius of a vehicle almost entirely relies on the velocity of the vehicle. During low-speed maneuvers the easiest way to determine the minimum turn radius would be with the Low-Speed Ackermann Method. By dividing the wheelbase of the vehicle by the maximum left and right Ackermann steering angle the minimum turn radius for both directions can be found during low-speed maneuvers. As speed increases the High-Speed Ackerman Steering Method would replace the Low-Speed Ackermann Method. With the High-Speed Ackermann Method turn radius is directly associated with the forward velocity of the vehicle and thus the minimum turn radius is likewise linked to forward velocity.

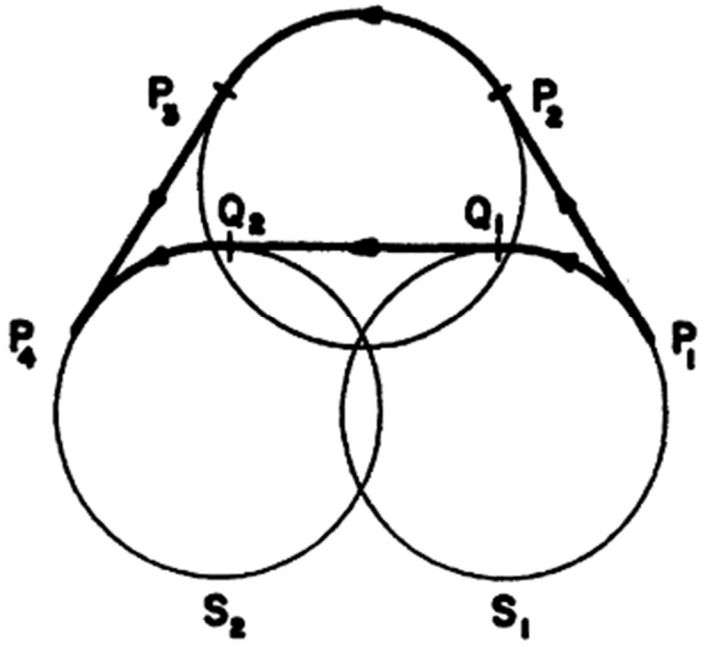

Figure 1.14: Dubins Path Example (Dubins 1970)

1.8 Potential Future Applications

1.8.1 Torque Vectoring

Fuse (2022) describes torque vectoring as a system of power delivery "...that freely generates a braking/driving torque difference between left and right wheels to direct control yaw moment applied to the vehicle, thus improving driving performance." By changing the amount of power applied to each wheel the turn radius can be altered. The main application of torque vectoring is vehicle stabilization De Novellis (2012).

1.8.2 4-Wheel Steering

Bari (2014) describes 4-wheel steering as "…a method developed in automobile industry for the effective turning of the vehicle and to increase the maneuverability." 4-

wheel uses all 4 wheels for steering. Arvind (2013) splits 4-wheel steering into 2 different sections, positive 4-wheel steering where the rear wheels turn in the same direction as the front wheels and negative 4-wheel steering where the rear wheels turn in the opposite direction as the front wheels. Singh (2014) explains that positive 4-wheel steering is used during higher speed maneuvers and that negative 4-wheel steering is used during low seed maneuvers. All of the methods covered by this paper would be applicable to 4-wheel steering vehicles apart from the Ackermann Steering methods which would need slight adjustments to accommodate the different steering geometry.

1.8.3 Aircraft

Aircraft are the target of many papers attempting to understand key performance variables like turn radius. These methods can be used to estimate the ITR of aircraft more accurately both fixed and rotary wing.

Bender (2019) uses both the Circle Method and Dubins Path to estimate the radius of turn and flight path of an aircraft. The Lateral Acceleration method could also be applied to give critical redundancy to the system. In Sun (2019) aircraft turn maneuvers are reconstructed using the Automatic Dependent Surveillance-Broadcast (ADS-B). The methods used in this thesis could greatly reduce the complexity of the formulas Sun (2019) used to estimate turn radius.

Chapter II

Methodology

The Experiment will be conducted with commonly available sensors so as to show the ease in which any perspective team could uses these methods in their own experiments. While the results of the experiment would almost certainly be improved by using a laboratory grade IMU and a dual antenna GPS receiver, the integrated IMU and GPS receiver in a modern smartphone are sensors that are exceedingly common and are proven capable of being used for similar activities (Kwapisz 2011), (Ofstad 2008). The encoder used is affordable for the vast majority of teams who would wish to replicate the experiment of this paper.

The drive cycle to gather the data that will be used for each method will be conducted simultaneously using an instrumented vehicle. The experiment will be conducted on a closed course. It is assumed that any change in road surface has little to no effect on ITR. It is assumed that any change in road banking or angle has little to no effect on ITR. The vehicle used is a John Deere Gator 855D S4 see Figure 2.1 and 2.2. The vehicle has a rear mounted engine and was operated in rear wheel drive. This layout tends towards oversteer. It is important to acknowledge this tendency especially since the understeer gradient of the vehicle is assumed to be 1 which implies perfect neutral steer.

Figure 2.1: John Deere Gator 855D S4

Figure 2.2: Mounting position for phone used as IMU.

Figure 2.3: CUI AMT-203 Encoder

The IMU data is taken from the LSM6DSO of a Samsung Galaxy S10 Smartphone running the Hyper IMU App. The GPS data is also taken from the same Samsung Galaxy S10. The steering input is taken from a CUI AMT-203 Absolute Encoder attached to the steering column as depicted in Figure 2.3. These sensors were chosen due to their availability.

2.1 The Course

The experiment uses the Embry-Riddle Aeronautical University campus roads for the course. The course is easily broken up into 4 sectors as shown in Figure 2.4. The first sector represents driving the vehicle to the start of the course and is comprised of a few low-speed 90 degree turns. The second sector is comprised of high-speed large radius turns. The third sector consists of medium and low speed chicanes. The 4th and final sector consists of medium speed 90 degree turns.

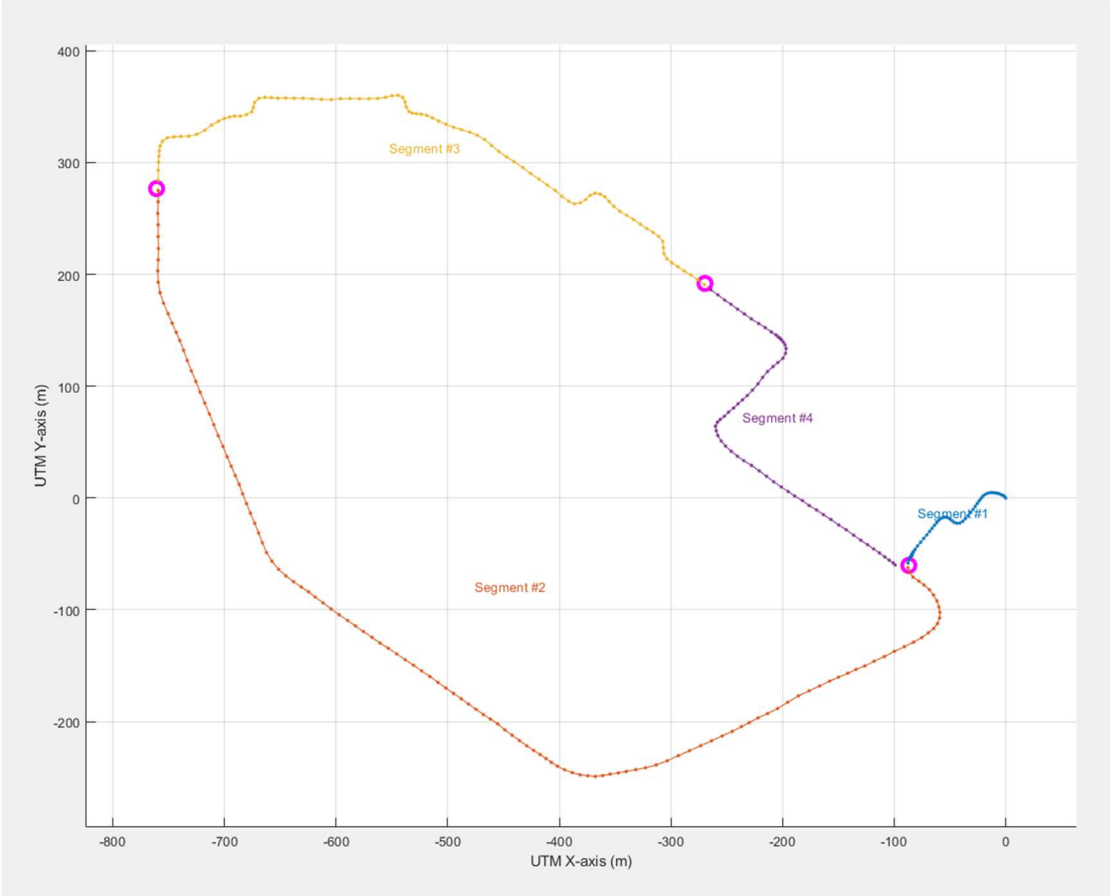

Figure 2.4: Course with 4 Segments

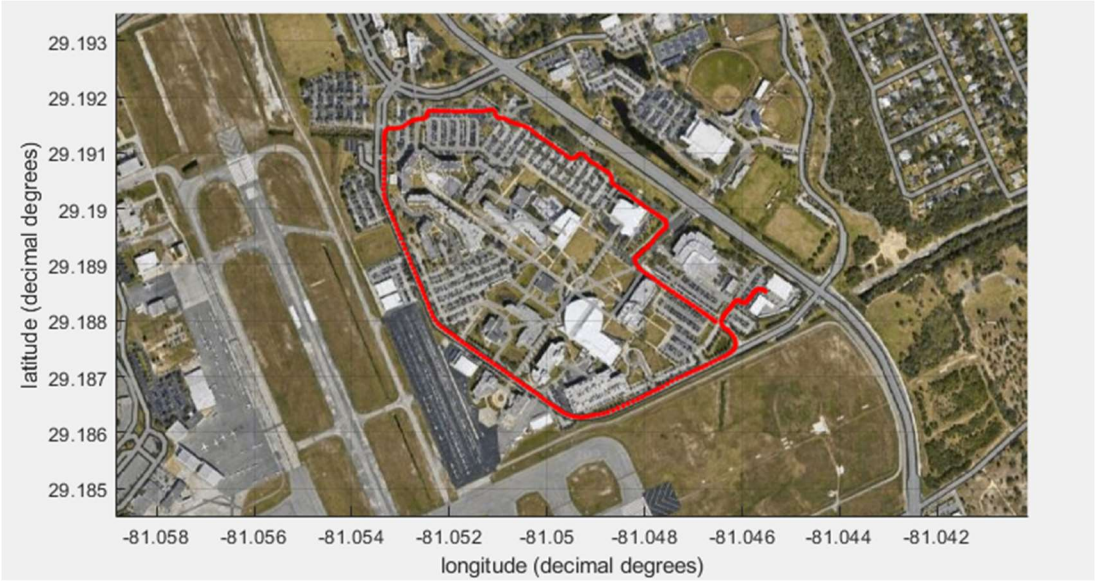

Figure 2.5: Satellite Image of Course

2.2 Sources of Data

Sensors on the vehicle will collect vehicle velocity, yaw rate, lateral acceleration, Ackermann steering angle, wheelbase length, and GPS position of the vehicle. In order to capture yaw rate and lateral acceleration an IMU will capture the acceleration of the vehicle in the Society of Automotive Engineers (SAE) standard axis system as shown in Figure 2.6. From the IMU data acceleration in the Y axis is considered lateral acceleration. The IMU can also detect the change in heading of the vehicle and from that determine the yaw rate.

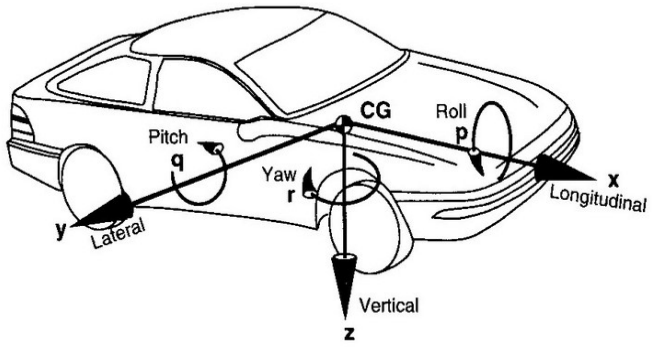

Figure 2.6: SAE Standard Axis System

The Ackermann Steering angle will be calculated based on the value of an encoder placed on the steering column of the vehicle. The encoder in degrees multiplied by the Ackermann steering ratio results in the Ackermann steering angle. Wheelbase length will be measured for the vehicle while it is stationary before the data collection phase.

2.3 Treatment of the Data

The coding language used to perform each of the methods is MATLAB. Each method will use the same data collected during a single run of the course. Velocity is found from the NMEA (National Marine Electronics Association) GPS data.

The sampling rate of the NMEA is lower than the sampling rate of the other required variables. The velocity found from the NMEA data is interpolated to the sampling rate of the other variables using the Modernized Akima formula. All repeated GPS points are removed so that every GPS point is unique for the GPS methods to work properly. A filter is placed on the steering input data that removes consistently erroneous datapoints as shown in Figure 2.7. An FIR moving average filter with a 1st order time constant of 1.2. this filters out a most of the noise experienced through not having a perfectly stable attachment point to the vehicle see Figure 2.8.

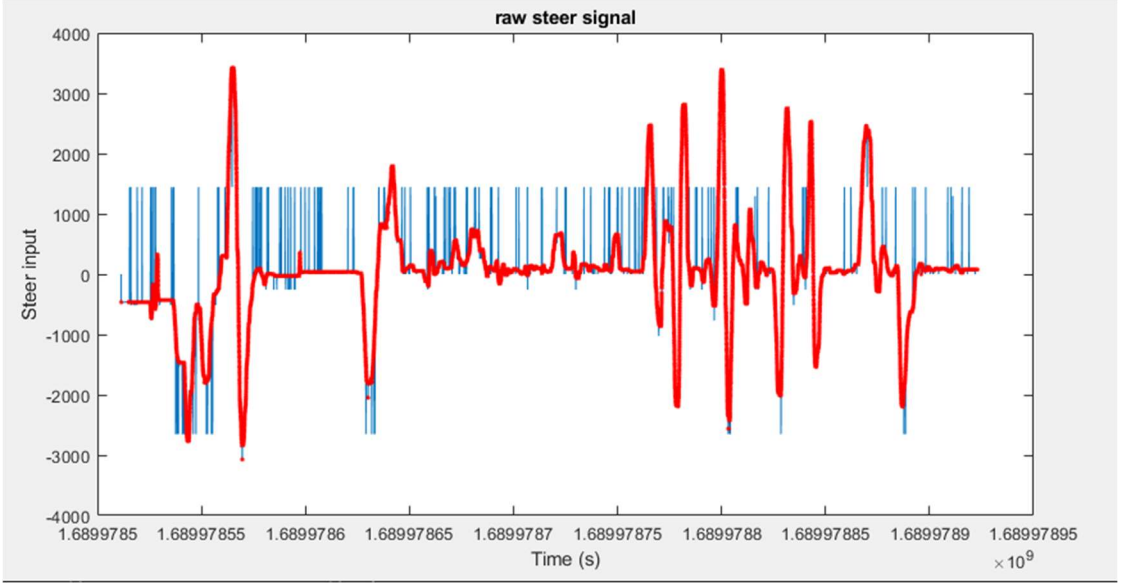

Figure 2.7: Filtered Steering Input

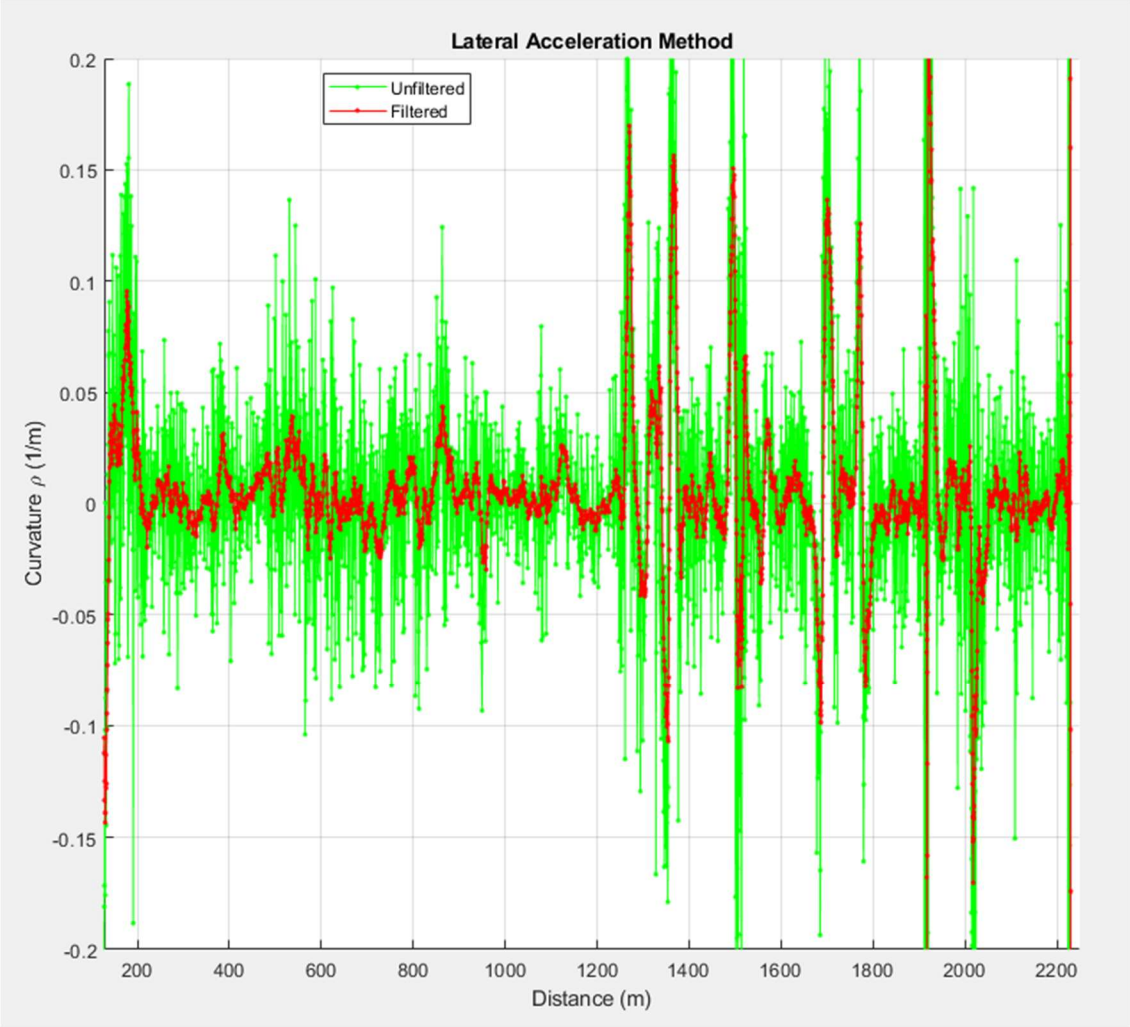

Figure 2.8: Filtered vs Unfiltered Lateral Acceleration Results

Chapter III

Results, Discussion, and Conclusions

3.1 Results

The results of each Method are largely similar; however, each method has its own distinct difference. The results are expressed in terms of curvature vs distance traveled. Curvature is used to prevent discontinuities that occur with turn radius on straight sections of the course. Distance is used to more accurately gauge where on the course the results are generated. The Yaw Rate Method and Lateral Acceleration Method generally have

higher noise than the other methods. The Circle method is reliant entirely on the amount of unique GPS data points and as such its sampling rate is lower leading to a lower resolution. The Cubic Spline Interpolation Method remedies this problem by interpolating between the unique GPS points. The Cubic Spline Interpolation Method uses Cubic Smoothing Splines as the interpolation method and due to the smoothing nature of Cubic Smoothing Splines this data sets' peaks and troughs are smoothed over compared to the other data sets.

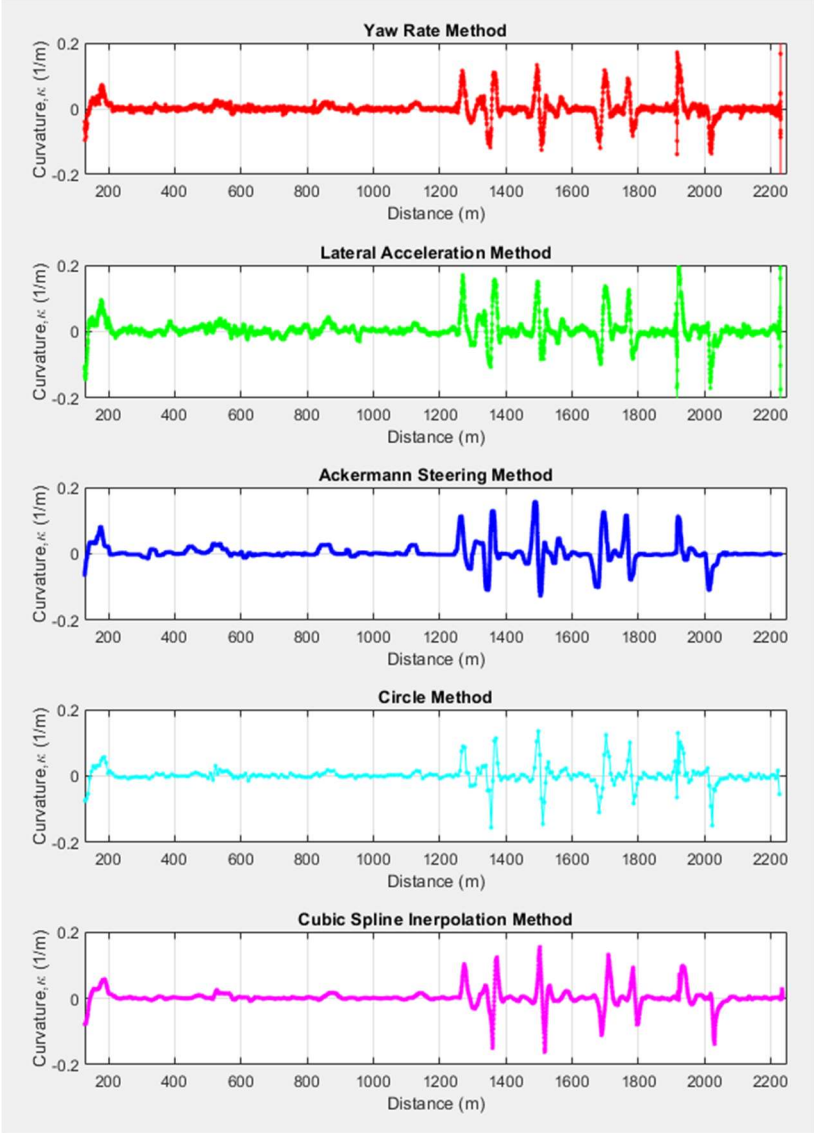

Figure 3.1 Graphs of the Results of Each Method (Curvature vs Distance)

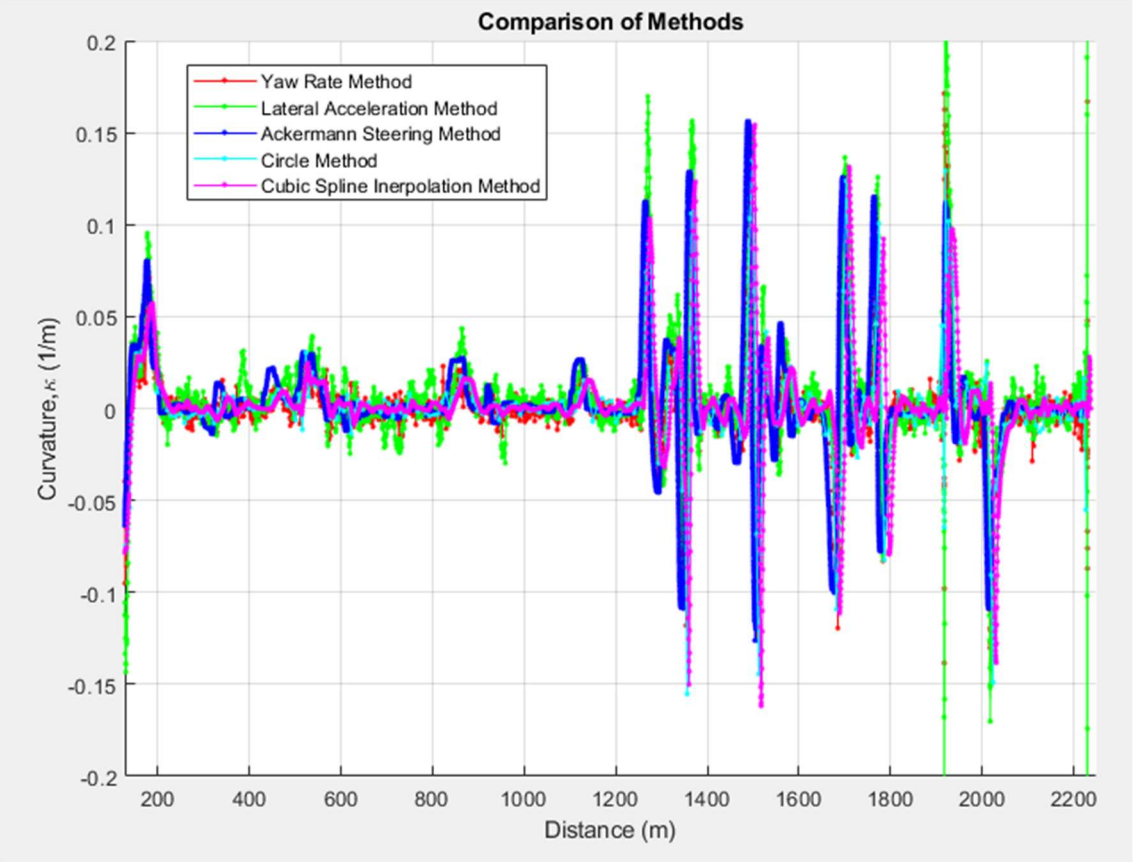

Figure 3.2 All methods on same graph (Curvature vs Distance)

3.2 Discussion

Both GPS Method data sets are translated to the right slightly implying that the events take place later on the course than the other methods. The Ackermann Steering Method gives results that are translated slightly to the left implying that the events take place earlier on the course than the other methods. This discrepancy in the position of the turns in the data sets is due to the nature of each of the methods. The Ackerman Steering Methods direct link to the steering mechanism results in the curvature dictated by the driver. The IMU methods depict the curvature in relation to the forces acting on the vehicle. Lastly the GPS Methods show the curvature of the path that the vehicle has already taken.

Figure 3.3: Turn 3

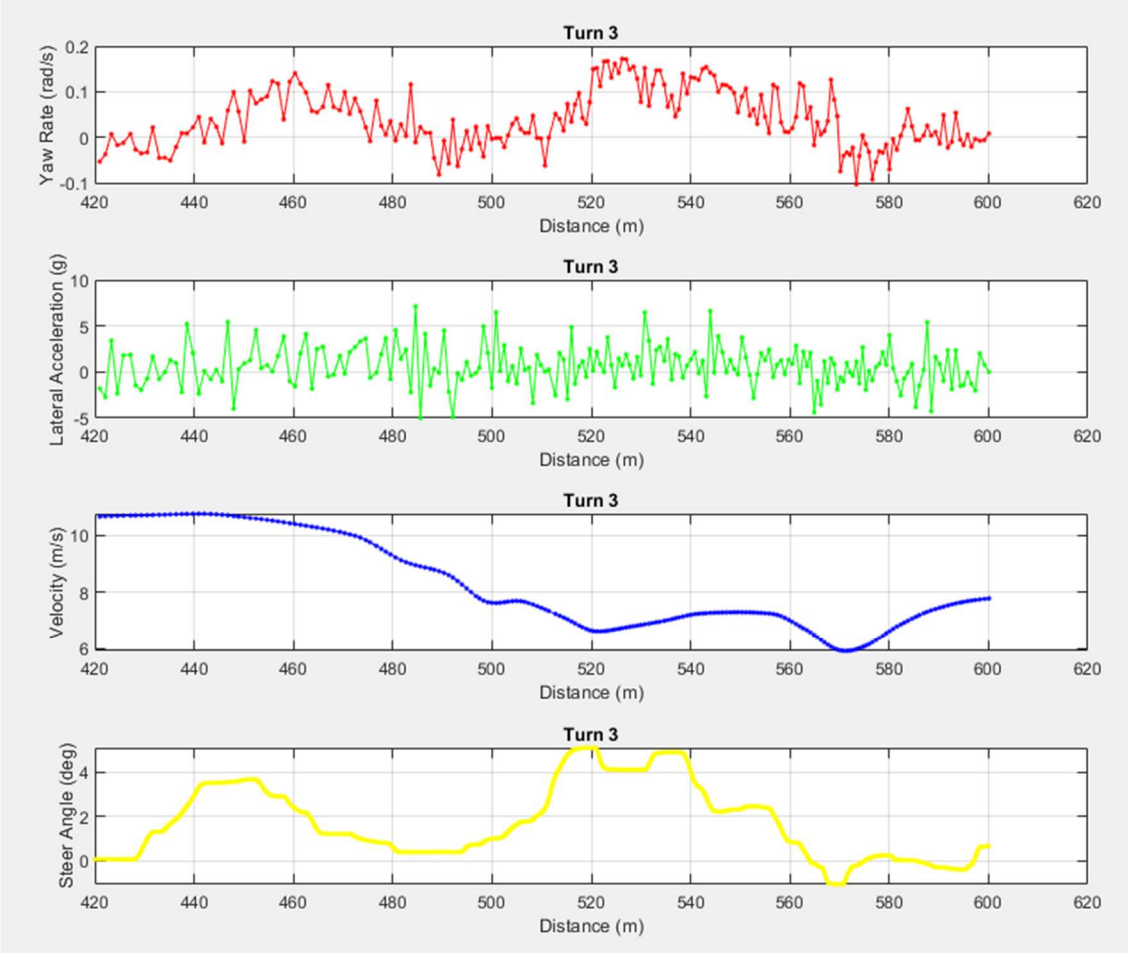

Figure 3.4: Data Through Turn 3

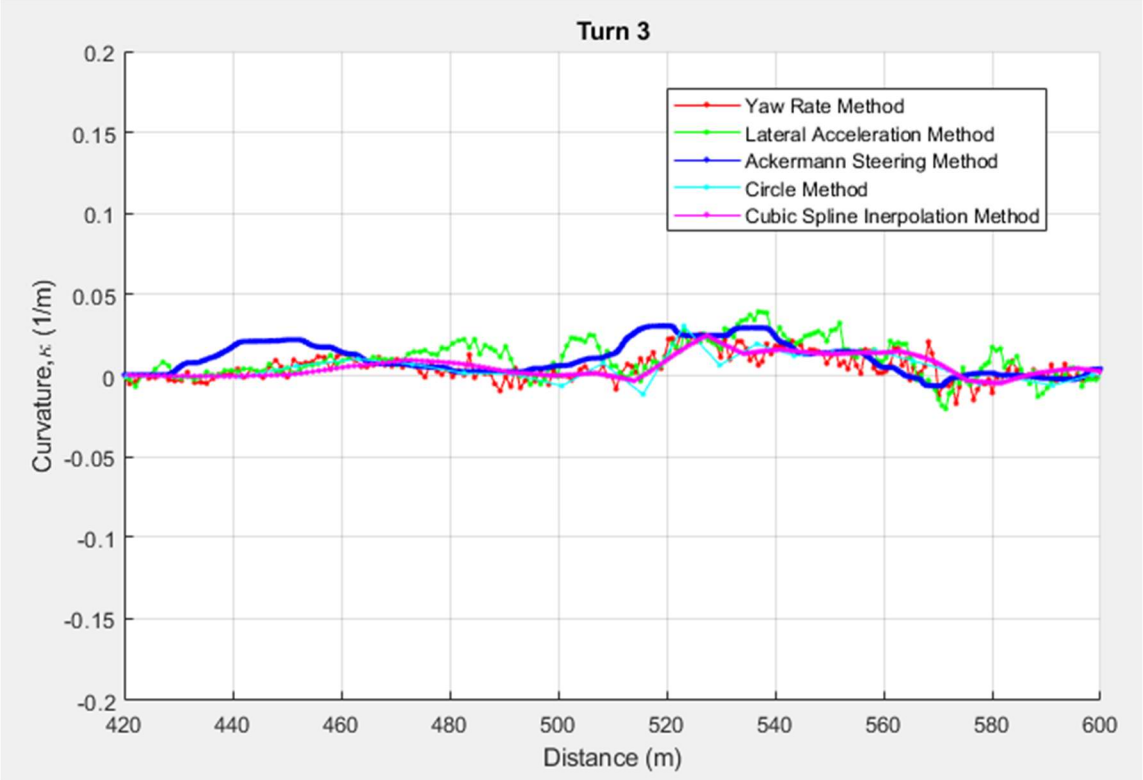

Figure 3.5: Curvature of Turn 3

Figure 3.6: Turn 19

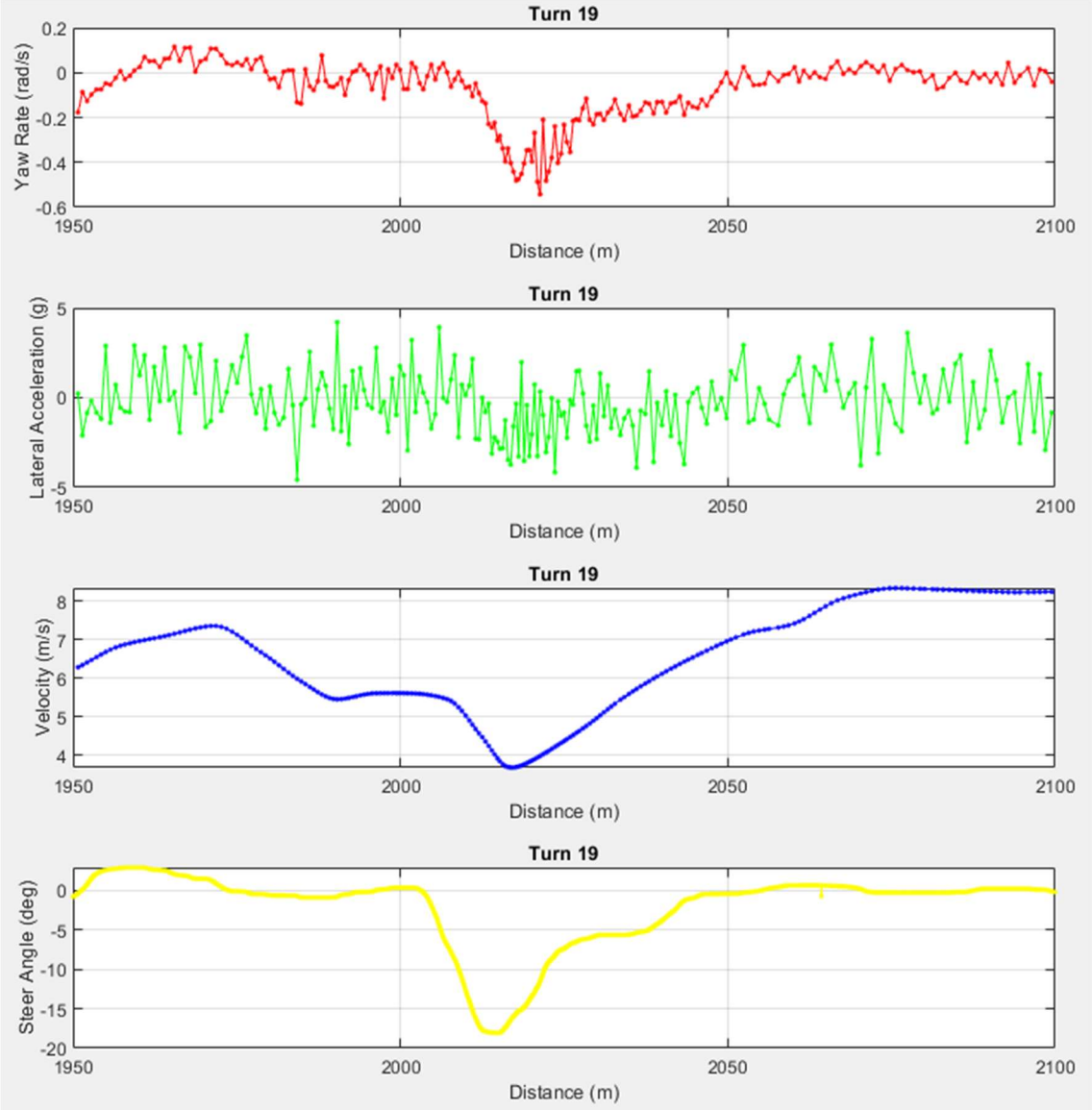

Figure 3.7: Data Through Turn 19

Figure 3.8: Curvature of Turn 19

In comparing the 3 groups we see that the Ackermann Steering Methods predict the forces that act on the vehicle due to the time difference between when the driver turns the steering wheel and when the yaw rate and lateral acceleration begin affecting the vehicle. Likewise, since the GPS methods use the current and one previous step to calculate the curvature 2 time steps in the past the result will be delayed in comparison to the other methods. This phenomenon is showcased during Turn 19 in Figure 3.8. It is important to

note that the offset of the different methods increases with distance traveled as seen when comparing Turns 3 and 19 in Figures 3.5 and 3.8.

Method Category	Input	Distance Offset
IMU Methods	Vehicle Kinematics	Early
Ackermann Steering Methods	Driver Steering	Neutral
GPS Methods	True Path	Delayed

Table 3.1 Distance Offset of the Methods

3.2.1 Alternatively Steered Vehicles

The methods covered in this paper do not only work for conventional Ackermann steering vehicles or in zero slip situations. The GPS Methos should be applicable to any vehicle as they only require the path taken by the vehicle. A small addition would need to be added to the Circle method in order to account for vehicles that move in 3 dimensions such as aircraft and submersibles. The Lateral Acceleration Method would apply to any forward moving vehicle so long as the forward velocity and lateral acceleration inputs are oriented correctly. Similarly, The Yaw Rate Method would still be applicable so long as the yaw rate input is oriented correctly. The Ackerman steering Methods would not be applicable to these vehicles.

Likewise, in conditions where an Ackermann steering vehicle is moving with side slip for example during drifting, the same methods would be applicable given that the coordinate system take into account the different orientation of the vehicle. The only method that would not be applicable to drifting cars and aircraft would be the Yaw Rate Method. During drifting the vehicle yaws at different rates through a corner than the same vehicle would without drifting.

3.3 Conclusions

From the results of the test, it can be seen that each of the method groups has a niche that it fulfills. The IMU Methods have good sensitivity and low complexity, however even after the noise is filtered out the clarity of the data is not ideal. The GPS Methods result in very good clarity especially the Cubic Spline Interpolation Method with a Cubic Smoothing Spline. The downside to the GPS methods is their low sensitivity and high complexity. The Ackermann Steering Methods direct link to the steering column allows for increased sensitivity over the other methods. The clarity of the Ackermann Steering Method is also excellent and while there is some noise it is easily filtered out resulting in middling complexity compared to the other methods. The Ackerman Steering Methods are the only methods that can detect the change in turn radius of a stationary vehicle due to its direct link to the steering system. Due to all these reasons, it can be concluded that the Ackermann Steering Method is the best method to use for its excellent clarity and sensitivity and its acceptable noise and complexity in comparison to the other methods, provided the required sensor is available.

Method	Noise	Clarity	Sensitivity	Complexity
Yaw Rate Method	High	Low	Middling	Lowest
Lateral Acceleration Method	Highest	Lowest	High	Low
Low Speed Ackermann Steering Method	Middling	High	Highest	Middling
Circle Method	Low	Middling	Low	High
Cubic Spline Interpolation Method	Lowest	Highest	Lowest	Highest

Table 3.2: Conclusions

3.4 Future Work

Foremost amongst the necessary future work is recreating the test to corroborate and validate the results. Secondly it is important to recreate the test with higher sampling rates on all data collection sensors. It is not unreasonable to believe that higher sampling rate on the data collection devices could give a fuller picture of the methods at work.

Tests should also be conducted with vehicles that have understeer gradients other than 0. Higher speeds like those achieved during travel on highways would also be a pertinent variable to test. These changes would allow a proper comparison of the High-Speed Ackermann Steering Method and the Low-Speed Ackermann Steering Method.

Another aspect that should be explored is using these methods for vehicles that do not use Ackermann steering. Specifically, both fixed wing and rotary wing aerial drones would be able to use one of the methods effectively as discussed in chapter 3.2.1.

It would also be pertinent to redo the test with vehicles that have access to 4-wheel steering as well as vehicles with torque vectoring to see how these technologies affect the methods.

Appendix A

Bibliography

Akima, H. (1970). A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. Journal of the ACM, 17(4), 589–602. https://doi.org/10.1145/321607.321609

Arvind, V. "Optimizing the turning radius of a vehicle using symmetric four wheel steering system." International Journal of Scientific & Engineering Research 4.12 (2013): 2177-2184

Bari, Dishank, et al. "Design and manufacturing of a system to measure the turning radius of vehicle." Int J Eng Sci Adv Technol 4.6 (2014): 536-540.

Benders, Sebastian, and Simon Koch. “Radius of Turn and Flight Path Angle Estimation from Unmanned Aircraft Flight Trajectories.” 2019 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2019, pp. 1336–43, https://doi.org/10.1109/ICUAS.2019.8798170.

Broggi, A., Medici, P., Zani, P., Coati, A., & Panciroli, M. (2012). Autonomous vehicles control in the VisLab Intercontinental Autonomous Challenge. Annual Reviews in Control, 36(1), 161–171. https://doi.org/10.1016/j.arcontrol.2012.03.012

Chen, Zheng. “On Dubins Paths to a Circle.” Automatica (Oxford), vol. 117, 2020, pp. 108996-, https://doi.org/10.1016/j.automatica.2020.108996.

Compere, M. (2021). Will The Real Turn Radius Please Stand? PowerPoint presentation Embry-Riddle Aeronautical University ME620 Advanced Vehicle Dynamics.

Compere, M. Tucker, C. (2022). Race Line Data Post-Processing and Curvature Estimation PowerPoint presentation Embry-Riddle Aeronautical University.

De Boor, Carl. A practical guide to splines. Vol. 27. New York: springer-verlag, 1978.

De Novellis, Leonardo, et al. "Torque vectoring for electric vehicles with individually controlled motors: State-of-the-art and future developments." World Electric Vehicle Journal 5.2 (2012): 617-628

Dubins, Lester E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics 79.3: 497-516.

Euclid Author, Of Alexandria Contributor Hypsicles, and Giovanni Mocenigo. Euclid's "Elements". Venice: Erhard Ratdolt, -05-25, 1482. Pdf. Retrieved from the Library of Congress, <www.loc.gov/item/2021667076/>.

Fritsch, F. N. "PCHIP, Piecewise Cubic Hermite Data Interpolation." (1985)

Fuse, Hiroyuki, et al. “Application of Independent-left-and-right-wheel-driving Force Controller to Torque Vectoring Differential with Two-input-two-output Motor Drive System for Electrified Vehicles.” Electrical Engineering in Japan, vol. 215, no. 4, 2022, https://doi.org/10.1002/eej.23400.

Gillespie, T. D. (1992). Fundamentals of vehicle dynamics. Society of Automotive Engineers. https://doi.org/10.4271/r-114

Gitay, N. N., Joshi, S. A., Dumbre, A. A., & Juvekar, D. C. (2018). Design & Manufacturing of an Effective Steering System for a Formula Student Car.

Gutiérrez, J., Apostolopoulos, D., & Gordillo, J. L. (2007). Numerical comparison of steering geometries for robotic vehicles by modeling positioning error. Autonomous Robots, 23(2), 147–159. https://doi.org/10.1007/s10514-007-9037-8

Jagelčák, J., Gnap, J., Kuba, O., Frnda, J., & Kostrzewski, M. (2022). Determination of Turning Radius and Lateral Acceleration of Vehicle by GNSS/INS Sensor. Sensors (Basel, Switzerland), 22(6), 2298-. https://doi.org/10.3390/s22062298

Kwapisz, Jennifer, et al. “Activity Recognition Using Cell Phone Accelerometers.” SIGKDD Explorations, vol. 12, no. 2, 2011, pp. 74–82, https://doi.org/10.1145/1964897.1964918.

Mitchell, Wm. C., et al. Analysis of Ackermann Steering Geometry. 2006, https://doi.org/10.4271/2006-01-3638.

Moler, C. (2019, April 29). Makima Piecewise Cubic Interpolation [web log]. Retrieved November 2, 2022, from https://blogs.mathworks.com/cleve/2019/04/29/makima-piecewise-cubic-interpolation/

Ofstad, Andrew, et al. “AAMPL: Accelerometer Augmented Mobile Phone Localization.” Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in Gps-Less Environments, ACM, 2008, pp. 13–18, https://doi.org/10.1145/1410012.1410016.

Park, H.-G., Ahn, K.-K., Park, M.-K., & Lee, S.-H. (2018). Study on Robust Lateral Controller for Differential GPS-Based Autonomous Vehicles. International Journal of Precision Engineering and Manufacturing, 19(3), 367–376. https://doi.org/10.1007/s12541-018-0044-9

Peters, Kenneth E., and Edward R. Cook. "The cubic smoothing spline as a digital filter." (1981).

Prompakdee, E., Boonporm, P., & Rooppakhun, S. (2016). The influence of weight distribution on the handling characteristics of intercity bus under steady state vehicle cornering condition. In MATEC Web of Conferences (Vol. 81, p. 04014). EDP Sciences.

Singh, Arun, et al. "Study of 4 wheel steering systems to reduce turning radius and increase stability." International Conference of Advance Research and Innovation (ICARI-2014). 2014

Sun, J., et al. Reconstructing Aircraft Turn Manoeuvres for Trajectory Analyses Using ADS-B Data. 2019.

VBOX Automotive, Racelogic. (2014). Calculating radius of turn from Yaw Rate - VBOX Automotive. Retrieved from https://www.vboxautomotive.co.uk/downloads/Calculating%20Radius%20of%20Turn%20from%20Yaw.pdf

Veneri, M., and M. Massaro. “The Effect of Ackermann Steering on the Performance of Race Cars.” Vehicle System Dynamics, vol. 59, no. 6, 2021, pp. 907–27, https://doi.org/10.1080/00423114.2020.1730917.

Wang, Y., Yang, J., Liu, H., Chen, Y., Gruteser, M., & Martin, R. (2013). Sensing vehicle dynamics for determining driver phone use. Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, 41–54. https://doi.org/10.1145/2462456.2464447

Xu, J., Xin, T., Gao, C., & Sun, Z. (2022). Study on the Maximum Safe Instantaneous Input of the Steering Wheel against Rollover for Trucks on Horizontal Curves. International Journal of Environmental Research and Public Health, 19(4), 2025–. https://doi.org/10.3390/ijerph19042025

Appendix B

Code

```
% post process HyperIMU log for GPS NMEA strings
%
% Ken Stutts, stuttsk@my.erau.edu
% Marc Compere, comperem@erau.edu
% created : 21 Jul 2023
% modified: 13 Nov 2023


clear ; clear all ; clear functions

addTextAtEachPoint          = 0; % (0/1) add waypoint numbers?
csv_file_steer_enc = '2023_07_21__18_28_31_data.xlsx';         % 2nd Gator
drive around campus with HyperIMU and steering wheel data
csvFname = 'HIMU-2023-07-21_18-28-13_clean.csv'; % gator drive around campus

data = readtable(csvFname);
dataTable_steer_enc = readtable(csv_file_steer_enc); % this skips text on the
top line
% data(1:5,:)
%
%    timestamp    MAC_address    Battery_level    Battery_status    Battery_ac
lsm6dso_accelerometer_x    lsm6dso_accelerometer_y    lsm6dso_accelerometer_z
lsm6dso_gyroscope_x    lsm6dso_gyroscope_y    lsm6dso_gyroscope_z    GPS_lat
GPS_long    GPS_alt                  NMEA_a                       NMEA_b
NMEA_c
%    _________    __________    _____________    _____________    _________
_____________________    ______________________    ______________________
_____________________    ____________________    ____________________    _______
________    _______    ____________________________________    _________
_________
%
%    1.69e+12         NaN             91                0               0
-0.17228                    -1.5347                    10.12
0                    0                       0         29.189     -81.046
3.6795     {'$PSMSG;magcal=FAIL;bluesky=0*00'}    {0×0 char}    {0×0 char}
%    1.69e+12         NaN             91                0               0
1.573                      -1.2185                    10.168                  -
0.084221                  0.03184                0.026266         29.189     -
81.046     3.6795     {0×0 char                          }    {0×0 char}
{0×0 char}
%    1.69e+12         NaN             91                0               0
0.73968                    -0.14835                   9.4017
-0.094605                 0.041003                0.026877         29.189     -
81.046     3.6795     {0×0 char                          }    {0×0 char}
{0×0 char}


t_unix   = data.timestamp/1000;          % (s) seconds since the epoch, 01 Jan
1970
t        = t_unix - t_unix(1);           % (s) elapsed time
```

```
Ax       = data.lsm6dso_accelerometer_x; % (m/s^2) accel x
ay       = data.lsm6dso_accelerometer_y; % (m/s^2) accel y
az       = data.lsm6dso_accelerometer_z; % (m/s^2) accel z
gx       = data.lsm6dso_gyroscope_x;     % (rad/s) gyro x
gy       = data.lsm6dso_gyroscope_y;     % (rad/s) gyro y
gz       = data.lsm6dso_gyroscope_z;     % (rad/s) gyro z
lat      = data.GPS_lat;                 % (dec deg) latitude
lon      = data.GPS_long;                % (dec deg) longitude
elev     = data.GPS_alt;                 % (m?)(ft?) altitude
Z_origin = min(elev); % (m) GPS reports MSL, so remove this offset to show
vehicle heights from ground
elev     = elev - Z_origin;

nmea_a = data.NMEA_a;
nmea_b = data.NMEA_b;
nmea_c = data.NMEA_c;


 %% extract STEER data
    tDT =
datetime(dataTable_steer_enc.time,'ConvertFrom','posixtime','TimeZone','Americ
a/New_York');
    t_steer      = dataTable_steer_enc.time; % (s) GPS time, seconds since 01
Jan 1970
    t0_steer     = t_steer(1);
    tACC_Steer   = t_steer-t0_steer; % (s) elapsed time, seconds since logging
start
    steer_raw    = dataTable_steer_enc.wrappedPos;

%% combine timelines
t_start=max(min(t_unix),min(t_steer));
t_end=min(max(t_unix),max(t_steer));
t_trim=t_unix;
while t_trim(end) > t_end
t_trim(end)=[];
 t_unix(end)=[];
 Ax(end)=[];
 ay(end)=[];
az(end)=[];
gx(end)=[];
gy(end)=[];
gz(end)=[];
lat(end)=[];
lon(end)=[];
elev(end)=[];
nmea_a(end)=[];
nmea_b(end)=[];
nmea_c(end)=[];
end



while t_trim(1) < t_start
t_trim(1)=[];
t_unix(1)=[];
```

```
Ax(1)=[];
ay(1)=[];
az(1)=[];
gx(1)=[];
gy(1)=[];
gz(1)=[];
lat(1)=[];
lon(1)=[];
elev(1)=[];
nmea_a(1)=[];
nmea_b(1)=[];
nmea_c(1)=[];
end
t=t_unix - t_unix(1);  % (s) elapsed time
% t=t_trim;
%% pull out unique vehicle information
reduce_to_unique_GPS_points=1; % (0/1) 0-->do nothing; 1-->downsample to
remove repeated GPS points
if reduce_to_unique_GPS_points==1
    method=4; % use distance-based method to reduce to unique [X,Y,Z] points
    [lat_sm,lon_sm,elev_sm,idx_keep] =
reduce_to_unique_spatial_GPS_datapoints(lat,lon,elev,method);
    t_sm       = t(idx_keep);
    gx_sm       = gx(idx_keep);
    gy_sm       = gy(idx_keep);
    gz_sm       = gz(idx_keep);
    ax_sm       = Ax(idx_keep);
    ay_sm       = ay(idx_keep);
    az_sm       = az(idx_keep);
    str=sprintf('reduced %i redundant GPS points to a unique set of %i
points',length(lat),length(lat_sm)); disp(str)
else
    lat_sm              = lat;
    lon_sm              = lon;
    t_gps_sm            = t_gps;
    t_sm                = t;
end


figure(10),clf
plot(t,ay,'r.-')
xlabel('HIGH FREQ time (s)')
ylabel('accel (m/s^2)')
grid on

figure(11),clf
plot(t(1:(end-1)),diff(t),'b.')
xlabel('HIGH FREQ time (s)')
ylabel('dt (s)')
grid on
```

```
% ----------------- show trace on google map -----------------------
figure(20),clf
plot3(lon_sm,lat_sm,elev_sm,'r.')
xlabel('longitude (decimal degrees)')
ylabel('latitude (decimal degrees)')
zlabel('elevation (m)')
grid on


plot_google_map('APIKey','AIzaSyCstEH_eMsiqIk5JemUjVvjG8THE4EYuno') %
Compere's Google Maps API key
%plot_google_map('MapType','satellite') % 'roadmap' 'hybrid' 'terrain'
plot_google_map('MapType','hybrid','ShowLabels',0,'Scale',2,'Resize',2,'Refres
h',1,'FigureResizeUpdate',1) % 'roadmap' 'hybrid' 'terrain' 'satellite'

plot_waypoint_labels=0; % (0/1) put waypoint numbers on every N'th point?
skipNth=10; % only put point markers every N'th to avoid cluttered plot

if plot_waypoint_labels==1
    for i=1:skipNth:length(t_sm)
        %myStr=num2str(i);
        myStr=sprintf('t=%0.1f(s)',t_sm(i)+t_unix(1));

text(lon_sm(i),lat_sm(i),elev_sm(i),myStr,'VerticalAlignment','bottom','Horizo
ntalAlignment','center');
    end
end

% if plot_waypoint_labels==1
%     for i=1:skipNth:length(t_sm)
%
text(lon_sm(i),lat_sm(i),elev_sm(i),num2str(i),'FontSize',10,'VerticalAlignmen
t','bottom','HorizontalAlignment','center') %,'BackgroundColor','White');
%     end
% end
view(0,90) % top-down view



%% convert from (lat,lon) to cartesian (X,Y) coordinates in meters and plot
figure(21),clf
[X_raw,Y_raw,utmzone,utmhemi] = wgs2utm(lat_sm,lon_sm); % convert from
(lat,lon) to (X,Y) in meters
X_origin=X_raw(1); % (m)
Y_origin=Y_raw(1); % (m)
X=X_raw-X_origin; % (m)
Y=Y_raw-Y_origin; % (m)
Z=elev_sm; %-Z_origin;
plot3(X,Y,Z,'b.')
view(0,90)
xlabel('X-axis (m)')
ylabel('Y-axis (m)')
zlabel('Z-axis (m)')
```

```
grid on
axis equal
%view(-25,30)

%plot_waypoint_labels=0; % (0/1) put waypoint numbers on every N'th point?
if plot_waypoint_labels==1
    for i=1:skipNth:length(t_sm)
        %myStr=num2str(i);
        myStr=sprintf('t=%0.1f(s)',t_sm(i)+t_unix(1));

text(X(i),Y(i),Z(i),myStr,'VerticalAlignment','bottom','HorizontalAlignment','
center');
    end
end



nema=1;
do_segment_path=1; % (0/1) segment path or not?
if do_segment_path>0
    segment_path
end




% walk through each GPS nmea string searching for GPRMC
% http://aprs.gids.nl/nmea/#rmc
%
myStr='$GNRMC;223523.00;A;2911.278028;N;08102.791208;W;0.0;;210723;4.0;W;A;V*7
B'
gps_t_unix=[];
gps_vel=[];
for i=1:length(nmea_a)
    for j=1:3
        if j==1, myStr = nmea_a{i}; end
        if j==2, myStr = nmea_b{i}; end
        if j==3, myStr = nmea_c{i}; end
        if ~isempty( strfind(myStr,'RMC') )
            myDisp=sprintf('found RMC in i=%d, j=%d, str=[ %s ]',i,j,myStr);
disp(myDisp)
            gpsField = strsplit(myStr,';');
            gpsTimeStr = gpsField{2}; % '223523.00' --> hhmmss.ss    from:
http://aprs.gids.nl/nmea/#rmc
            dt = datetime(gpsTimeStr, 'InputFormat', 'HHmmss.SS'); % 21-Jul-
2023 22:28:13
            gps_t_unix = [ gps_t_unix   ; posixtime(dt)          ]; % (s)
unixtime
            gps_vel    = [ gps_vel ; str2double(gpsField{8})*0.514444 ]; %
(m/s) converted from knots, 1knot = 0.514444 m/s
        end
    end % for j=1:3
end % for i=1:length(nmea_a)
```

```
gps_t = gps_t_unix - gps_t_unix(1); % (s) elapsed time, from zero

figure(100),clf
plot(gps_t,gps_vel,'ko-','MarkerSize',4)
xlabel('elapsed GPS time (s)')
ylabel('GPS vel (m/s)')
grid on
%xlim([128 450])

% create a velocity vector at the higher frequency time
% vq = interp1(x,v,xq)
gps_vel_hi_pchip = interp1(gps_t,gps_vel,t,'pchip');
gps_vel_hi_spline = interp1(gps_t,gps_vel,t,'spline');
gps_vel_hi_makima = interp1(gps_t,gps_vel,t,'makima');

hold on
plot(t,gps_vel_hi_pchip,'r.')
plot(t,gps_vel_hi_spline,'g.')
plot(t,gps_vel_hi_makima,'b.')
legend('1Hz GPS velocity','velocity interpolated: pchip','velocity
interpolated: spline','velocity interpolated: makima','Location','best')



gps_vel_sm_pchip = interp1(gps_t,gps_vel,t_sm,'pchip');
gps_vel_sm_spline = interp1(gps_t,gps_vel,t_sm,'spline');
gps_vel_sm_makima = interp1(gps_t,gps_vel,t_sm,'makima');







dist = zeros(size(t));
dist(2:end) = (t(2:end) - t(1:end-1)) .* (gps_vel_hi_makima(1:end-1) +
gps_vel_hi_makima(2:end))/2;
dist = cumsum(dist);

dist_sm = zeros(size(t_sm));
dist_sm(2:end) = (t_sm(2:end) - t_sm(1:end-1)) .* (gps_vel_sm_makima(1:end-1)
+ gps_vel_sm_makima(2:end))/2;
dist_sm = cumsum(dist_sm);
distIC=dist_sm(1:end-2,:);




%% Yaw rate method


R1=(gps_vel_hi_makima./-gz); % (m/(rad/s)) turn radius
rho1=1./R1; % (1/m) curvature
figure(1), clf
plot(dist,rho1,'.-');
```

```
title('Yaw rate Method')
ylabel('Curvature \rho (1/m)') %Turn Radius (m)')
xlabel('Distance (m)')
xlim([128 2250])
ylim([-0.2 +0.2])
grid on

%% Lateral Acceleration Method

R2XP=((gps_vel_hi_makima.^2)./(Ax));
rho2XP=1./R2XP; % (1/m) curvature


deltat=sum(diff(t))/(length(t)-1); % estimated sample time

tau=1.2; % (s) 1st order filter's time constant

windowLen = ceil((tau)/deltat);



RHOP = fir(t,rho2XP,windowLen); % FIR moving average filter

%Rho2F = iir(t,rho2XP,tau);       % IIR first order filter
figure(2), clf


hold on
plot(dist,rho2XP,'g.-');
plot(dist,RHOP,'r.-');


title('Lateral Acceleration Method')
ylabel('Curvature \rho (1/m)') %Turn Radius (m)')
xlabel('Distance (m)')
xlim([128 2250])
ylim([-0.2 +0.2])
grid on
legend('Unfiltered','Filtered','Location','best')



%% Ackermann Steering method
figure(201), clf
    plot(tACC_Steer+t_steer(1),steer_raw);
    title('raw steer signal')
    ylabel('Steer input')
    xlabel('Time (s)')

    steer = steer_raw; % duplicate the wonky data set
    t_steer_unwonky = tACC_Steer; % duplicate the associated time array
    wonky_numbers_to_remove = [-256,1445,0,-2651,-512,1296,512,256,-768,-
1024,-1536];
    for k=1:length(wonky_numbers_to_remove)
```

```
        idx_wonky = find(steer== wonky_numbers_to_remove(k) ); % find indices
where steer_raw is clearly incorrect
        myStr=sprintf('removing %i numbers',length(idx_wonky)); disp(myStr)
        steer(idx_wonky)=[]; % delete these weird points
        t_steer_unwonky(idx_wonky)=[]; % delete corresponding times for wonky
points
    end
    %1445
    hold on
    plot(t_steer_unwonky+t_steer(1),steer,'r.');


 L=2.896; %wheel base in m
    steer_offset=100;
    TireDeg=0.007875*(steer-steer_offset);
    TireRad=TireDeg.*(pi/180);
    R3=L./TireRad;
    rho3=1./R3;

figure(202),clf
plot(t,dist,'ko-','MarkerSize',4)
xlabel('elapsed GPS time (s)')
ylabel('distance covered (m)')
grid on
%xlim([128 450])

% create a velocity vector at the higher frequency time
% vq = interp1(x,v,xq)
dist_hi_pchip = interp1(t,dist,t_steer_unwonky,'pchip');
dist_hi_spline = interp1(t,dist,t_steer_unwonky,'spline');
dist_hi_makima = interp1(t,dist,t_steer_unwonky,'makima');

hold on
plot(t_steer_unwonky,dist_hi_pchip,'r.')
plot(t_steer_unwonky,dist_hi_spline,'g.')
plot(t_steer_unwonky,dist_hi_makima,'b.')
legend('IMU Distance','Distance interpolated: pchip','Distance interpolated:
spline','Distance interpolated: makima','Location','best')

 figure(3), clf
    plot(dist_hi_makima,rho3,'.-');
    title('Ackermann Steering Method')
    ylabel('Curvature \rho (1/m)')
    xlabel('Distance (m)')
    grid on
    xlim([128 2250])
    ylim([-0.2 +0.2])
figure(203)
    plot(t_steer_unwonky,TireDeg);
 title('Tire Angle')
ylabel('tire on the ground (deg)')
xlabel('Time (s)')
% xlim([0 3600])
 %ylim([-0.2 +0.2])
grid on
```

```
%% Circle Method

N=length(X);
A=zeros(2);
k12=zeros(N-2,1);
k13=zeros(N-2,1);
x12=zeros(N-2,1);
y12=zeros(N-2,1);
x13=zeros(N-2,1);
y13=zeros(N-2,1);
R5=zeros(N-2,1);
for i=1:N-2
k12(i)=(X(i)^2)-(X(i+1)^2)+(Y(i)^2)-(Y(i+1)^2);
k13(i)=(X(i)^2)-(X(i+2)^2)+(Y(i)^2)-(Y(i+2)^2);

x12(i)=2*(X(i+1)-X(i));
x13(i)=2*(X(i+2)-X(i));

y12(i)=2*(Y(i+1)-Y(i));
y13(i)=2*(Y(i+2)-Y(i));

A=[x12(i) y12(i); x13(i) y13(i)];

C=[-k12(i);-k13(i)];
D=inv(A)*C;
R5(i)=sqrt((X(i)-D(1,1))^2+(Y(i)-D(2,1))^2);
end
rho5=(1./R5).*(-sign(gz_sm(1:end-2,1))); % (1/m) curvature
figure(5), clf
plot(distIC,rho5,'.-');
title('Circle Method')
ylabel('Curvature \rho (1/m)')
xlabel('Distance (m)')
xlim([128 2250])
ylim([-0.2 +0.2])
grid on
```

```
%% Cubic Spline Method

 nav_S(:,1)=X; % nav_S is the variable used for spline interpolation and
curvature estimation
 nav_S(:,2)=Y;


nav_Ns = length(nav_S);
```

```
% make splines for the entire path
spline_select=4; % 1:spline(), 2:pchip(), 3:makima()
% for N waypoints, spline generates (N-1) polynomials of 4 coefficients each
% pp = spline(x,y)
if spline_select==1
    p_X = spline( [1:nav_Ns] , nav_S(:,1) ); % for N waypoints, spline
generates
    p_Y = spline( [1:nav_Ns] , nav_S(:,2) ); % (N-1) polynomials of 4
coefficients each
    spline_str='spline()';
elseif spline_select==2
    p_X = pchip( [1:nav_Ns] , nav_S(:,1) ); % for N waypoints, spline
generates
    p_Y = pchip( [1:nav_Ns] , nav_S(:,2) ); % (N-1) polynomials of 4
coefficients each
    spline_str='pchip()';
elseif spline_select==3
    % modified Akima's formula for smaller wigglies
    p_X = makima( [1:nav_Ns] , nav_S(:,1) ); % for N waypoints, spline
generates
    p_Y = makima( [1:nav_Ns] , nav_S(:,2) ); % (N-1) polynomials of 4
coefficients each
    spline_str='makima()';
elseif spline_select==4
    % pp = csaps(x,y) *smoothing* spline, fundamentally different than
spline/pchip/makima
    % values = csaps(x,y,p,xx), smoothing parameter, p=1 is closest to data,
p=0 is flat-line average
    %p=0.9;
    %p=0.5;
    p=0.85; % this works surprisingly well by changing (lat,lon) locations
towards greater smoothness
    p_X = csaps(1:nav_Ns,X,p); % generate smoothing cubic spline in pp form
    p_Y = csaps(1:nav_Ns,Y,p);
    spline_str='csaps()';
end

% plot splines in parameter space
figure(101),clf
ds=0.1; % (%) sub-interval discretization length as a fraction on [0 1]
%ds=0.5;
X_spline = ppval(p_X,1:ds:(nav_Ns+1));
Y_spline = ppval(p_Y,1:ds:(nav_Ns+1));
subplot(2,1,1)
    plot(1:(nav_Ns+0),X,'r.','MarkerSize',8)
    hold on
    plot(1:ds:(nav_Ns+1),X_spline,'b.-')
    xlabel('waypoint number')
    ylabel('X-value (m)')
    grid on
    legend('unique XY points (m)','spline interpolant
(m)','location','southeast')
subplot(2,1,2)
    plot(1:(nav_Ns+0),Y,'r.','MarkerSize',8)
```

```
    hold on
    plot(1:ds:(nav_Ns+1),Y_spline,'b.-')
    xlabel('waypoint number')
    ylabel('Y-value (m)')
    grid on
    legend('unique XY points (m)','spline interpolant
(m)','location','southeast')
% compare unique XY points and interpolated points
figure(102),clf
hold on
plot(nav_S(:,1),nav_S(:,2),'r.','MarkerSize',12) % (m) unique XY points from
GPS data
plot(X_spline,Y_spline,'b.')     % (m) interpolated XY points
grid on
if addTextAtEachPoint==1
    for i=1:skipNth:nav_Ns

text(nav_S(i,1),nav_S(i,2)+15,num2str(i),'HorizontalAlignment','Center','Verti
calAlignment','Middle','FontSize',14);
    end
end
xlabel('X-axis (m)')
ylabel('Y-axis (m)')
title('Spline vs Waypoints')
axis equal



% plot cubic splines between every waypoint with different colors
figure(103),clf
hold on
%cmap=colormap(jet(nav_Ns));
cmap=colormap(lines(nav_Ns));

kappa_full  = [];
domain_full = [];
length_full = []; L_base=0;
for i=1:(nav_Ns-1) % step through each spline just created

    domain=[i:ds:(i+1)];
    xx = ppval(p_X,domain); % evaluation along the i'th polynomial
    yy = ppval(p_Y,domain);
    plot(xx,yy,'.-','Color',cmap(mod(i,length(cmap)),:));

    [kappa_out,domain_out , p_X_deriv,p_Y_deriv] = computeCurvature(i, p_X,
p_Y, ds );
    N=length(domain)-1; % estimate arc length with curvature discretization
    [L,L_cumu] = computePolyLength( p_X , p_Y , [domain(1) domain(end)], N );

    kappa_out = -kappa_out; % sign change: RH-turn is (+) curvature
    domain_full = [domain_full, domain_out(2:end)    ];
    kappa_full  = [kappa_full , kappa_out(2:end)     ];
    length_full = [length_full, L_base+L_cumu(2:end) ];
    L_base = L_base + L;
```

```
end

%plot(nav_S(:,1),nav_S(:,2),'bo')
grid on
if addTextAtEachPoint==1
    for i=1:skipNth:nav_Ns

text(nav_S(i,1),nav_S(i,2)+15,num2str(i),'HorizontalAlignment','Center','Verti
calAlignment','Middle','FontSize',14);
    end
end
xlabel('X-axis (m)')
ylabel('Y-axis (m)')
title('Plot of Each Spline')
axis equal

% plot curvature as a function of time
% figure(6),clf
% plot(t_sm,kappa_full)
% grid on
% xlabel('Waypoint number')
% ylabel('Curvature,\kappa (1/m)')
% ylim([-0.04 +0.04])

% plot curvature as a function of waypoint number
figure(104),clf
plot(domain_full,kappa_full,'.-')
grid on
xlabel('Waypoint number')
ylabel('Curvature,\kappa (1/m)')
%hax=gca; % get current axes
%hax.XTick=[1:1:nav_Ns]; % improve tick marks
 ylim([-0.2 +0.2])



% plot curvature as a function of distance along the track
figure(6),clf
plot(length_full,kappa_full,'.-')
grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
%hax=gca; % get current axes
%hax.XTick=[1:1:nav_Ns]; % improve tick marks
title('Cubic Spline Method')
xlim([128 2250])
ylim([-0.2 +0.2])


% % compute spline polynomial length
% N=100;
% [L,L_cumu] = computePolyLength( p_X , p_Y , [1 nav_Ns], N );
%
% fprintf('\npath length %0.2f(km)\n',L/1000);
```

```
figure(7), clf
hold on
plot(dist,rho1,'r.-');
plot(dist,RHOP,'g.-');
plot(dist_hi_makima,rho3,'b.-');
plot(distIC,rho5,'c.-');
plot(length_full,kappa_full,'m.-')


grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Comparison of Methods')
xlim([128 2250])
ylim([-0.2 +0.2])
legend('Yaw Rate Method','Lateral Acceleration Method','Ackermann Steering
Method','Circle Method','Cubic Spline Inerpolation Method','Location','best')


figure(8), clf
tiledlayout(5,1)
nexttile
plot(dist,rho1,'r.-');
grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Yaw Rate Method')
xlim([128 2250])
ylim([-0.2 +0.2])
nexttile
plot(dist,RHOP,'g.-');
grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Lateral Acceleration Method')
xlim([128 2250])
ylim([-0.2 +0.2])
nexttile
plot(dist_hi_makima,rho3,'b.-');
grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Ackermann Steering Method')
xlim([128 2250])
ylim([-0.2 +0.2])
nexttile
plot(distIC,rho5,'c.-');
grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Circle Method')
```

```
xlim([128 2250])
ylim([-0.2 +0.2])
nexttile
plot(length_full,kappa_full,'m.-')
grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Cubic Spline Inerpolation Method')
xlim([128 2250])
ylim([-0.2 +0.2])




figure(155),clf
plot3(lon_sm,lat_sm,elev_sm,'r.')
xlabel('longitude (decimal degrees)')
ylabel('latitude (decimal degrees)')
zlabel('elevation (m)')
grid on


plot_google_map('APIKey','AIzaSyCstEH_eMsiqIk5JemUjVvjG8THE4EYuno') %
Compere's Google Maps API key
%plot_google_map('MapType','satellite') % 'roadmap' 'hybrid' 'terrain'
plot_google_map('MapType','hybrid','ShowLabels',0,'Scale',2,'Resize',2,'Refres
h',1,'FigureResizeUpdate',1) % 'roadmap' 'hybrid' 'terrain' 'satellite'

plot_waypoint_labels=1; % (0/1) put waypoint numbers on every N'th point?
skipNth=2; % only put point markers every N'th to avoid cluttered plot

if plot_waypoint_labels==1
    for i=1:skipNth:length(dist_sm)
        %myStr=num2str(i);
        myStr=sprintf('Dist=%0.1f(m)',dist_sm(i));

text(lon_sm(i),lat_sm(i),elev_sm(i),myStr,'VerticalAlignment','bottom','Horizo
ntalAlignment','center','FontSize',10,'Color','y');
    end
end

% if plot_waypoint_labels==1
%     for i=1:skipNth:length(t_sm)
%
text(lon_sm(i),lat_sm(i),elev_sm(i),num2str(i),'FontSize',10,'VerticalAlignmen
t','bottom','HorizontalAlignment','center') %,'BackgroundColor','White');
%     end
% end
view(0,90) % top-down view


%T3 = 420m to 600m
%T19 = 1950m to 2100;
```

```
T3_entrence = 1426;
T3_exit = 1631;
T19_entrence = 3307;
T19_exit = 3525;

T3_entrence_ack = 15199;
T3_exit_ack = 17401;
T19_entrence_ack = 35597;
T19_exit_ack = 37964;

figure(156),clf
tiledlayout(4,1)
nexttile

plot(dist(T3_entrence:T3_exit),-gz(T3_entrence:T3_exit),'r.-')
grid on
xlabel('Distance (m)')
ylabel('Yaw Rate (rad/s)')
title('Turn 3')
nexttile
plot(dist(T3_entrence:T3_exit),Ax(T3_entrence:T3_exit),'g.-')
grid on
xlabel('Distance (m)')
ylabel('Lateral Acceleration (g)')
title('Turn 3')
nexttile
plot(dist(T3_entrence:T3_exit),gps_vel_hi_makima(T3_entrence:T3_exit),'b.-')
grid on
xlabel('Distance (m)')
ylabel('Velocity (m/s)')
title('Turn 3')
nexttile
plot(dist_hi_makima(T3_entrence_ack:T3_exit_ack),TireDeg(T3_entrence_ack:T3_ex
it_ack),'y.-')
grid on
xlabel('Distance (m)')
ylabel('Titre Angle (deg)')
title('Turn 3')



figure(157),clf
tiledlayout(4,1)

nexttile

plot(dist(T19_entrence:T19_exit),-gz(T19_entrence:T19_exit),'r.-')
grid on
xlabel('Distance (m)')
ylabel('Yaw Rate (rad/s)')
title('Turn 19')
nexttile
plot(dist(T19_entrence:T19_exit),Ax(T19_entrence:T19_exit),'g.-')
grid on
```

```
xlabel('Distance (m)')
ylabel('Lateral Acceleration (g)')
title('Turn 19')
nexttile
plot(dist(T19_entrence:T19_exit),gps_vel_hi_makima(T19_entrence:T19_exit),'b.-
')
grid on
xlabel('Distance (m)')
ylabel('Velocity (m/s)')
title('Turn 19')
nexttile
plot(dist_hi_makima(T19_entrence_ack:T19_exit_ack),TireDeg(T19_entrence_ack:T1
9_exit_ack),'y.-')
grid on
xlabel('Distance (m)')
ylabel('Tire Angle (deg)')
title('Turn 19')

figure(158), clf
hold on
plot(dist,rho1,'r.-');
plot(dist,RHOP,'g.-');
plot(dist_hi_makima,rho3,'b.-');
plot(distIC,rho5,'c.-');
plot(length_full,kappa_full,'m.-')


grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Turn 3')
xlim([420 600])
ylim([-0.2 +0.2])
legend('Yaw Rate Method','Lateral Acceleration Method','Ackermann Steering
Method','Circle Method','Cubic Spline Inerpolation Method','Location','best')


figure(159), clf
hold on
plot(dist,rho1,'r.-');
plot(dist,RHOP,'g.-');
plot(dist_hi_makima,rho3,'b.-');
plot(distIC,rho5,'c.-');
plot(length_full,kappa_full,'m.-')


grid on
xlabel('Distance (m)')
ylabel('Curvature,\kappa (1/m)')
title('Turn 19')
xlim([1950 2100])
ylim([-0.2 +0.2])
legend('Yaw Rate Method','Lateral Acceleration Method','Ackermann Steering
Method','Circle Method','Cubic Spline Inerpolation Method','Location','best')
```