CHALMERS
UNIVERSITY OF TECHNOLOGY

GPS-based path-following control used for local distributions

Master's thesis in Electric Power Engineering

HUAN ZHANG

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

GPS-based path-following control of i-dolly used for local distributions

HUAN ZHANG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Electric Power Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

GPS-based path-following control of i-dolly used for local distributions

HUAN ZHANG

Examiner: Professor Jonas Fredriksson, Vice Head Of Department for doctoral programme at Electrical Engineering
Supervisor: Maliheh Sadeghi Kati, Postdoctoral Researcher at Mechatronics/Systems and Control/Electrical Engineering
Supervisor: Jose Miguel Vilca Ventura, Senior Research Engineer at Volvo Group Trucks Technology

Master's Thesis 2020:NN
Department of Electrical Engineering
Division of Electric Power Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Owned by Chalmers University, picture taken by Anna-Lena Lundqvist
Chalmers University of Technology
Gothenburg, Sweden 2020

Abstract

Automation of local distribution using automated dolly technology is a viable and productive alternative for future state of art green terminals and local distribution of container trailers. This technology offers unmanned transports. It reduces driver operator cost and removes diesel fuel cost. The semi-trailer unit is towed by an intelligent and electrically propelled converter dolly (i-dolly) with automatic coupling and parking of semi-trailers. In this solution, autonomous dolly-semitrailer combination is considered to be used to transport the containers between the dry port and company terminals.

Within this thesis, a GPS-based path-following control of i-dolly for the local distribution is designed and implemented. The thesis starts with the study and derivation of a kinematic model of single unit vehicle. Then the control algorithm for a single unit vehicle (i-dolly) is designed and implemented based on the predefined path with Global Position System*(GPS). Then, an articulation angle between the dolly and semitrailer is considered in the kinematic model and control algorithms. The kinematic model and control algorithms are upgraded for the double units vehicle (i-dolly and semitrailer). At last, the kinematic model is replaced by the advanced vehicle model VTM and the control algorithms are tested on VTM.

The controller is designed to be low-speed independent and works well to make the single unit vehicle track the given trajectory closely. It makes the autonomous vehicle drive up to 9 m/s forward and -3 m/s backward to follow the given path within a high precision. But the linear controller does not work well to control the combination vehicle to follow the given path stably and precisely.

Keywords: Path generation, Forward & reverse motion control, Articulated vehicle, LQR controller.

Acknowledgements

This thesis is carried out at Volvo 3P and Chalmers University of Technology at the Department of Electrical Engineering in the division of Electric Power Engineering. It is one part of an i-dolly project at Volvo 3P involved the cooperation with Chalmers Revere Lab and Chalmers Vehicle Dynamic Group.

I spend a great time during my thesis project with lots of kind help. I want to begin with thanks to Berntsson Lars-Olof from Volvo Group and my two supervisors Jose Miguel Vilca Ventura from Volvo Group, Maliheh Sadeghi Kati from Chalmers. They give me this great opportunity to get involved in the i-dolly project and do my thesis work at Volvo Group. Great thanks for their constant guidance, feedback, encouragement and trust.

Also thanks to Sachin Janardhanan from Volvo Group. The nice discussion with him gives me insights and idea and also thanks for his great effort in VTM delivery which is used in my simulation. Special thanks to my manager Karlsson Daniel at Volvo Group, who is appreciative of all work. I am appreciated for all people from the EE department at Volvo Group for the good Friday afternoon Fika activity with good cakes and rest.

Great thanks to my examiner Prof.Jonas Fredriksson for feedback and patience. Special thanks to Stefan Lundbergn from Chalmers for his generous help and valuable time to guide me at the beginning of this thesis. Prof.Bengt J H Jacobson always kindly organizes the weekly x-dolly Chalmers meeting and makes the whole project process cleared. Fredrik Von Corswant regularly organizes the i-dolly technical coordination meeting. Great appreciation to Berntsson Lars-Olof, Jose Miguel Vilca Ventura, Maliheh Sadeghi Kati, Jonas Fredriksson, Arpit Karsolia, Janardhanan Sachin, Per A Larsson, Toheed Ghandriz for joining the meeting and sharing great idea and discussion.

I also want to take this opportunity to thank my friends and family members. Special thanks to my friends Katumba Brian, Mengyue Li, Abhineet Singh Tomar, Cai Yao, Duo XU, who always encourage me and take care of me and give me good suggestions both in life and study. Big thanks to my parents who always give me enough freedom and support and encourage me think positively towards any problem, which directly and indirectly motivates me to apply for and finish this thesis finally.

HUAN ZHANG, Gothenburg, August 2020

Contents

List of Abbreviations

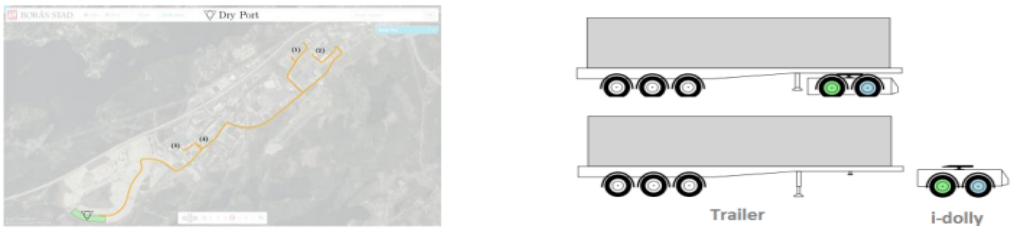
Readers should be familiar with the terms below related to vehicle dynamics and controlling to follow this document. They are summarized with short description here before next sections are presented.

GPS	Global Positioning System
VTM	Virtual Truck Model
i-dolly	Intelligent Dolly
COG	Center of Gravity
IRC	Instant Rotating Center
LQR	Linear Quadratic Controller
MPC	Model Predictive Controller
LTV	Linear Time Variant

1
Introduction

1.1 Background

Nowadays, Automation of local distribution is one alternative of achieving future state of art green terminals and local distribution of container trailers. It provides a unmanned transport between the company terminals and dry ports. The semi-trailer is towed by an intelligent converter dolly (i-dolly) which is propelled by electricity completely. This technology reduces driver operator costs as well as the fuel cost during the whole transportation. The dry port (green color) and the local distribution network (yellow color) of container transport are shown in the Figure.1.1 (a). Thus, it is envisioned that the i-dolly could serve as a tractor for the semitrailer units to reduce the use of diesel engine for smaller routes. The Figure.1.1 (b) depicts the schematic diagram of the proposed vehicle combination with automatic coupling devices and the i-dolly in which the electric propulsion is set on the rear axle and the steering is set on the front axle.



(a) Dry port distribution ***(b)*** Trailer and i-dolly

Figure 1.1: Steering behaviour

1.2 Aim

The main purpose of this thesis project is to design a GPS-based path following controller in which the converter i-dolly follows prescribed path with and without semitrailer coupled to perform forward and reverse motion autonomously.

1.3 Objects

In this thesis, a GPS-based path following controller is studied and designed to achieve forward and reverse motion autonomously. The thesis work consists of the

following parts:

- Study the kinematic model of the single unit and combination vehicle.
- Derive single- or/and two-track kinematic vehicle models.
- Synthesize a GPS-based path-following controller at low speed including
  - Lateral vehicle motion control via steering forward motion at low speed
  - Lateral vehicle motion control via steering reverse motion at low speed
- Build and simulate the system and the controller in Simulink.
- Integrate the LQR controller with VTM for single track as well as two tracks model.

1.4 Limitations

The following problems are not considered and solved in this thesis.

- The autonomous i-dolly technology investigated in the thesis is proposed for the dedicated path between the dry port and company terminals and obstacles avoidance is not considered.
- Navigation strategy when the i-dolly navigates into the GPS-dropout areas is not proposed. The i-dolly technology is based on unstable GPS signal.
- Sensor fusion and filters are not considered. All measurements from sensors are considered to be reliable and noise free.

1.5 Method

The thesis work starts with the literature study of vehicle's kinematic modelling. Then the path following algorithms are studied and investigated for delivering the desired states to the controller. The controller is formulated based on the kinematic model and the error dynamic model of states. The controller is implemented with the single unit kinematic model and then extended to the double units model. Then it is integrated and tested with VTM. VTM is the complicated truck model developed by Volvo 3P. Sensor data delivered by VTM is the feedback to the path follower and controller module. Algorithms are implemented and tested on VTM.

1.6 Report Outline

The report starts with introduction including background, aim, objectives, limitation and method. Chapter 2 describes the mathematical modelling of kinematic model and also VTM. Chapter 3 introduces the path generation algorithm design and implementation. Controller strategies for path following are described in Chapter 4. In Chapter 5, simulation and results are presented and discussed. Conclusion and future work are described in Chapter 6.

2
Vehicle Modelling

2.1 Introduction

Vehicle modelling is required and necessary for engineering projects. In this thesis firstly kinematic modelling is derived. Compared with dynamic modelling, the kinematic one is regardless of force or torque that causes its motion and it is able to calculate the position and orientation of the vehicle. Then the algorithm is also tested on an advanced vehicle model which is called Virtual Truck Model (VTM). More details about VTM can be found at section 2.3.

2.2 Kinematic bicycle model

2.2.1 Single unit vehicle modelling

For low speed lateral motion control, a rigid body and non-slippage of tire are considered. Under the basic assumption of planar motion, a four-wheeled bicycle vehicle model is considered here shown as the Figure.2.1 [1] to describe the single unit tractor/dolly moving behavior.

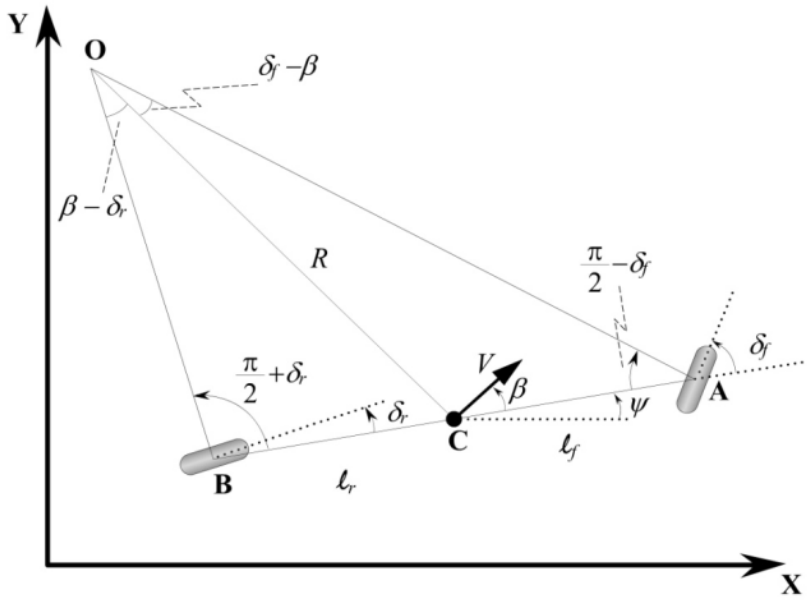


Figure 2.1: Kinematic bicycle model [1]

In the bicycle model, the left and right steering angles in front and also in rear wheels are approximately equal, then the two front and rear wheels are both represented to be one single wheel at point A and B respectively. The Global coordinates X-Y is fixed to represent the vehicle motion. (x,y) is the tracking point which is located at the rear wheel. Vehicle velocity v is usually defined at point C and C is the center of gravity (COG). The vehicle is rear-wheel-driven, thus, the point C is regarded as the same point as B and the velocity is also defined at point B as the same direction of heading. Side-slip angle β is zero without skidding considered under low speed maneuvers. δ_f is the front steering angle and δ_r is the rear steering angle. Since the vehicle is front-wheel-steering, the rear steering angle δ_r is set as zero. O is the Instant Rotating Center (IRC) based on steering angles and the wheelbase L. The wheelbase L is formulated as follow:

$$L = l_f + l_r \tag{2.1}$$

l_f and l_r are the distance between A and B point to the center of gravity C point respectively. ψ is the vehicle heading angle. R is turning radius. Under the low speed moving, the radius of the vehicle wheel changes slowly and the rate of the change of the vehicle orientation is equal to the angular velocity of the vehicle.

Referring to the Figure.2.1, the geometric relationships of vehicle motion are formulated as follow:

$$\dot{x} = vcos(\psi) \tag{2.2}$$

$$\dot{y} = vsin(\psi) \tag{2.3}$$

$$\dot{\psi} = \frac{v}{L}tan(\delta_f) \tag{2.4}$$

In reverse motion control of the single unit tractor, the global coordinates system is fixed and the vehicle velocity direction in the vehicle body coordinates is opposite to the vehicle heading direction. Equations for reverse motion are formulated by setting v to $-v$ from the Equation 2.1 to Equation 2.4. Coordinates are shown as the Figure.2.2 and the Figure.2.3. The heading angle is set and initialised in a different body frame. In the forward motion body frame, the heading angle is initialised as zero radian and in the reverse body frame, the initial heading angle is set as π radians.

In this thesis, the reference point for forward and backward motion is set at the center of the rear wheel of the vehicle. For the forward motion, tracking point is put either at the front wheel or at the rear wheel. For the reverse motion, if the reference point is at the front of the vehicle, then the vehicle dynamics constitute a non-minimum phase system. The non-minimum phase system means the transfer function as well as the inverse transfer function of the system are not stable and casual. When the number of poles and the number of zeros are not equal and also all poles and all zeros are not within the unit circle for discrete system (or not lie on the left hand side of the s-plant for continuous system), then it is a non-minimum phase system. In conclusion, it is more stable to set the tracking point at the rear wheel of the vehicle.

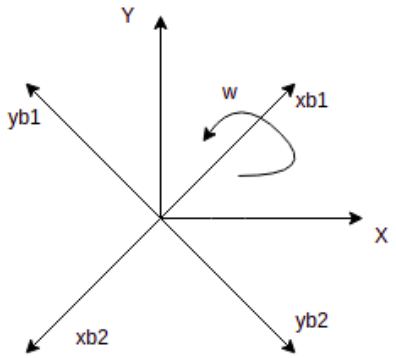


Figure 2.2: X, Y frame represents the fixed global inertial frame; x_{b1}, y_{b1} frame represents the vehicle body frame in forward motion; x_{b2}, y_{b2} frame represents the vehicle body frame in reverse motion

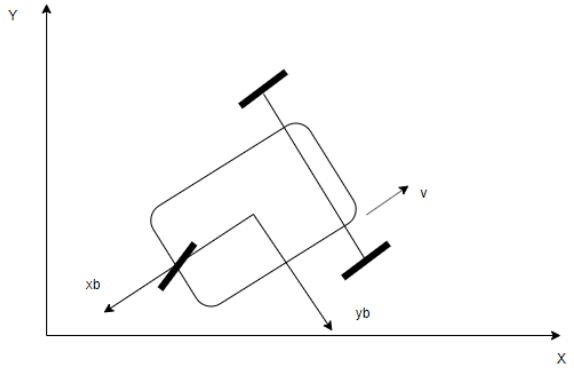


Figure 2.3: Vehicle reverse motion frame

2.2.2 Articulated vehicle modelling

For the articulated vehicle tractor/dolly with a semi-trailer, In this thesis the tracking point is set at the rear wheel of tractor/dolly both in the forward motion and the reverse motion. An additional articulation angle connecting the tractor and semi-trailer is also considered to design the model and controller. The geometry is shown as the Figure.2.4. (x_1,y_1) is the global position of rear wheel of the tractor/dolly and (x_2,y_2) is the global position of rear wheel of the semi-trailer. ϕ is the articulation angle of the tractor/dolly and the semitrailer. There is a joint point located on the tractor. c is the distance between the center point of the tractor rear wheel and the joint point. When c is not zero, the joint hitching is not on the rear wheel of the tractor. It is called "off axle" hitching. When c is zero, it is defined as "on axle" hitching [3]. On axle hitching is considered in this thesis. v_{u1} and v_{u2} are driven speed at reference points (x_1,y_1) and (x_2,y_2) respectively. θ_1 is the heading angle of

the tractor/dolly while θ_2 is the heading angle of the trailer. l_2 is the wheelbase of the trailer and l_1 is the wheelbase of the tractor/dolly.

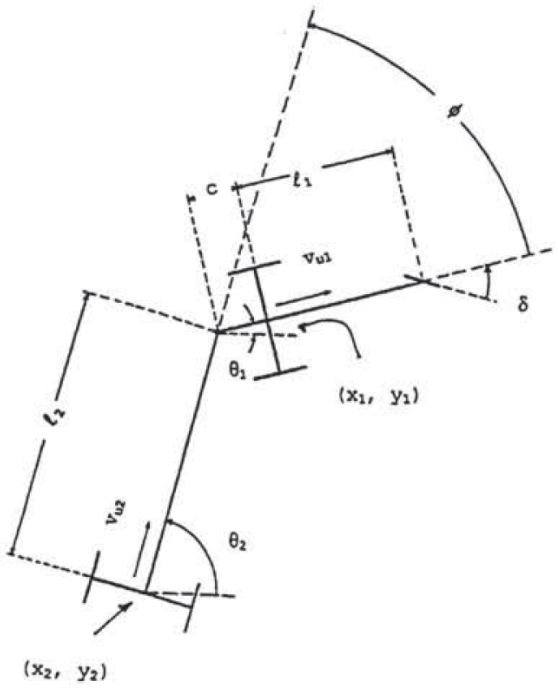


Figure 2.4: Tractor with semi-trailer geometry [2]

Table 2.1: Definition of variables

Name	Description
(x_1,y_1)	rear wheel position of tractor/dolly
(x_2,y_2)	rear wheel position of semi-trailer
l_1	wheel base of tractor/dolly
l_2	wheel base of semi-trailer
ϕ	articulation angle
v_{u1}	speed at (x_1,y_1)
v_{u2}	speed at (x_2,y_2)
θ_1	heading angle of the tractor
$\theta 2$	heading angle of the trailer
δ	steering angle of the front wheel of tractor/dolly

When the vehicle is moving forward and backward, the Cartesian model under the assumption of planar and slippage free is formulated. When it is moving forward, the tracking point is located at the rear wheel of tractor:

$$\dot{x_1} = v_1 cos\theta_1 \tag{2.5}$$

$$\dot{y_1} = v_1 sin\theta_1 \tag{2.6}$$

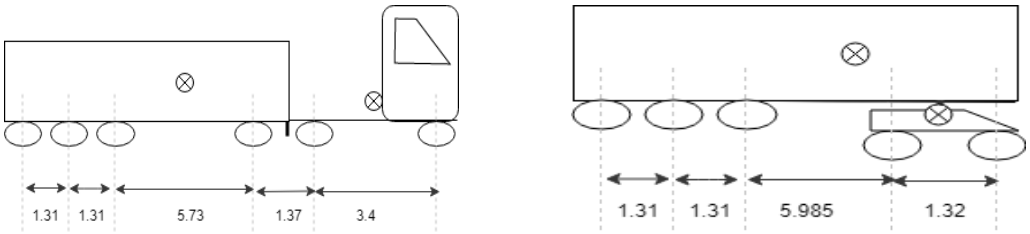
$$\dot{\theta_1} = \frac{v_1}{l_1} tan\delta \tag{2.7}$$

$$\dot{\phi} = \dot{\theta_2} - \dot{\theta_1} = \frac{-v_1}{l_2} sin\phi - \frac{v_1}{l_1} tan\delta \tag{2.8}$$

Similarly, when it comes to backward motion, in this thesis the tracking point is also located at the rear wheel of the tractor. The speed v_1 is negative and the initiated heading angle θ is set as $-\pi$ or zero in case of reversing the given path coordinates.

2.3 Virtual Truck Model

Virtual Truck Model(VTM) is the advanced vehicle model developed at Volvo 3P and it contains details of vehicle dynamics and is implemented in Matlab Simulink and SimMechanics. The model is proved to be accurate for real test data and here it is used for testing the controller which is designed based on the kinematic model. The structure and parameters are represented as the Figure.2.5. The simulation parameters details are also described in the Chapter 5. The dolly model is structured based on the truck/tractor model.



(a) Truck with Semitrailer Configuration ***(b)*** Trailer and i-dolly

Figure 2.5: VTM structure and configuration

3

Path Generation

3.1 Introduction

In the path following application of autonomous vehicle, there are lots of work related to path processing for vehicle tracking. In this part, several methods for path generation are described and studied. Path generation is to represent the given trajectory and provide way points along the trajectory. It delivers a set of way points as the input of the control algorithm. In this thesis, spline fitting is chosen as the way to represent the given trajectory since it provides a compact and smoother path for vehicle tracking. The spline algorithms are described in detail in the following sections. [4]

3.2 Literature Review

The literature review in this part introduces several trajectory representation methods to show how to generate way points based on a given trajectory.

3.2.1 Straight Segments [5]

Straight Segments is one of linear interpolation methods. It connects every two global coordinates points along the path with straight line and then it uses the current vehicle position which is got from sensor data to calculate the closest intersection point for every piece-wise straight line to get the way point. One of drawbacks of this method is for each interval the calculated vehicle heading angle is constant and it leads to a non-smooth steering behavior which is shown in the simulation part in Chapter 5.

3.2.2 Spline Interpolation [6]

Compared with straight segments, spline curve provides a smoother and tighter curve from one point to another. Each individual segment is a smooth arc. In definition, a spline is a set of polynomial curve usually with degree 3 or more. It is smoothly connected at certain points. At each connecting point, a piece-wise polynomial has a continuous first-degree derivative (tangent vector). The definition also requires that if the spline has degree in k, then all the derivatives up to$(k-1)^{st}$ are equal at the connecting point.

3.2.2.1 Piece-wise polynomial curve

Piece-wise polynomial curve is the curve proceeding from one point to another point as a polynomial function. The general form of the polynomial is as:

$$y = a + bx + cx^2 + dx^3 + ... \tag{3.1}$$

Here, the case of polynomial function with degree 3 is explained as an example to show how does the smooth cubic spline fit data points. If the degree of the polynomial function is more than three, the polynomial function cannot give a smooth shape because it tends to be quite sensitive to the position of control points. All data points that are supposed to be passed through are also called control points. For example, if there are four points and they are fitted with cubic spline in each segment, the solution steps are shown as follow:

- Enforce the C^0 continuity, which means the curves meet each other where they join. This requires each segment curve passes through all control points in this segment.
- Enforce the C^1 continuity, which means the slopes match where the curves join. This requires $f_i^{'}(x_i) = f_{i+1}^{'}(x_{i+1})$ at joining point x_{i+1}. f_i and f_{i+1} are curve functions for two joining segments respectively.
- Enforce the C^2 continuity, which means the curvatures match where the curves join. This requires $f_i^{''}(x_i) = f_{i+1}^{''}(x_{i+1})$ at joining point x_{i+1}.

Thus, the solution comes to solve a set of linear equations to get the unknown coefficients. It comes to solve the matrix equation since all linear equations are rewritten to a matrix form. The example of four control points has 12 coefficients to be solved and 12 linear equations are formulated as a matrix form from three segments f_0, f_1, f_2. The Figure.3.1 shows how does the cubic spline look like and The Table.3.1 shows the three segments equations formulation.

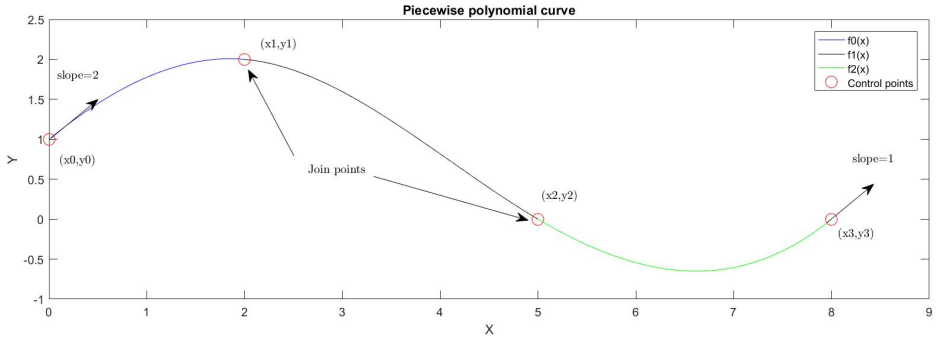


Figure 3.1: Piece-wise polynomial curve with four points

Table 3.1: Linear equations formulation

Segment $f_0(x)$	Segment $f_1(x)$	Segment $f_2(x)$
Slope at (x_0, y_0)	Curve through (x_1, y_1)	Curve through (x_2, y_2)
Curve through (x_0, y_0)	Curve through (x_2, y_2)	Curve through (x_3, y_3)
Curve through (x_1, y_1)	Slope match at (x_2, y_2) with f_2	Slope at (x_3, y_3)
Slope match at (x_1, y_1) with f_1	Curvature match at (x_2, y_2) with f_2	
Curvature match at (x_1, y_1) with f_1		
Sum:5 equations	Sum:4 equations	Sum:3 equations

The 12 coefficients are solved by the equation below:

$$Me = y \tag{3.2}$$

The 12×12 matrix M encodes the C^0, C^1, C^2 conditions. The e is the coefficients variables scalar and the y is the matrix derived from the linear equations which contains the value from the right side of linear equations.

3.2.2.2 Curve parameterization

Curve parameterization introduces other parameters to represent the given function variables. For example, for the given function with variables x and y there is another parameter t. x is represented as $x = g(t)$ and y is represented as $y = f(t)$. Then the function is reshaped with the parameter t and becomes as $(g(t), f(t))$.

Parametric curve is used to shape a more complex plane as well as a space curve. Parametric curve includes the uniform and nonuniform parametric one. The uniform parametric curve produces a good result when the points are roughly spaced equally. Otherwise, the nonuniform parametric curve is better to be used. Curve parameterization introduced in this part is applied to the Hermite cubic splines as well as the Catmull-Rom splines.

3.2.2.3 Hermite cubic splines & Catmull-Rom splines

The method of computing a natural cubic spline using the C^0,C^1 ,C^2 continuity constraints has been introduced and explained in the section 3.2.2.1. It is good to understand how to formulate a piece-wise cubic spline and also a higher degree polynomial curve. One drawback of the piece-wise polynomial curve is that it is fully defined by a set of linear equations and it is easy to wiggle if there is one outlier among the control points. When there are a set of points, a local shape control is more useful. The local shape control means a local control over the certain points to shape the fitting curve without consideration of the change in these points.

The Hermite spline and the Catmull-Rom spline are good local shape control methods [6]. The theories of these two methods are explained in this section and the implementation with the given path is shown in the section 3.3.

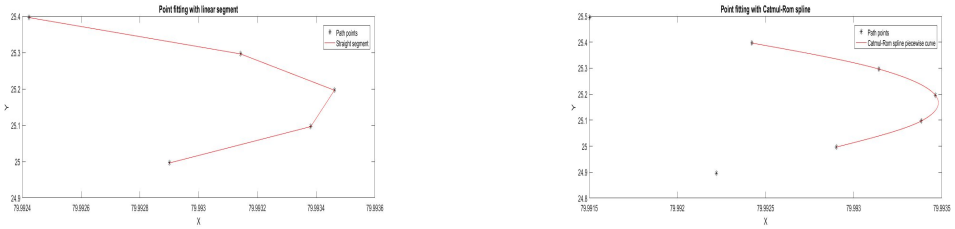
- Hermite spline
  The Hermite spline is applied to every two control points p_i and p_{i+1} with the given point coordinates and slope at each control point. The slopes are provided in the form of tangent vector. It is transferred to the parameterization form by introducing u as the additional parameter varying form 0 at p_i to 1 at p_{i+1}. Thus, the variables x and y are represented as x_u and y_u functions. C^0 and C^1 continuity constraints are used to formulate a matrix equation in the parametric form and the slopes are transferred to be the parametric one from the given tangent vector as $\begin{bmatrix}\Delta x & \Delta y\end{bmatrix}^T$.

- Catmull-Rom spline
  The Catmull-Rom spline is also applied to every two points. Without the given slopes at every two control points, It uses additional two points to calculate the slopes. Thus, basically the fitting algorithm is applied to every four points while the spline is only drawn between the middle two control points. This means it uses the point p_{i-1} before the point p_i and the subsequent point p_{i+2} after the point p_{i+1} to determine the slopes in the tangent vector form at the points p_i and p_{i+1}. The vector between p_{i-1} and p_{i+1} gives the tangent vector at p_i. The vector between p_i and p_{i+2} gives the tangent vector at p_{i+1}. Overall, two imaginary points are created before the first point and after the last point of the given path in order to get slopes at the start and the end point respectively. Then the other work is the same as the Hermite spline to formulate the parametric matrix form and to solve the coefficients of x_u, y_u function.

3.3 Algorithm Implementation

3.3.1 Linear Segment & spline interpolation

The Figure.3.2 compares the performance of the linear segment and the spline interpolation methods for data points fitting.



(a) Linear segment ***(b)*** Catmull-Rom spline curve

Figure 3.2: Data points fitting

The Figure.3.2 shows the Catmull-Rom spline curve fitting is smoother than the linear segment fitting. Because the Catmull-Rom spline curve has the smoothing heading angles while the linear segment holds the same slope in each piece-wise, namely, it holds the heading angles in one segment. For the controlling point, the smoothing heading angles are important for producing better controlling results. Because the steering angle is also related to the heading angle and if the headings are kept in a constant value for each segment then the steering becomes more fluctuating and unstable.

3.3.2 Piece-wise cubic spline

3.3.2.1 Algorithm

In order to avoid a bad result from a not good scaling raw data, the curve parameterization is combined with the piece-wise polynomial curve to get a better performance. Since each piece-wise is a cubic curve and the polynomial degree is 3. Thus, it formulates the piece-wise cubic spline going through the given path points. The math form of the piece-wise cubic spline is shown as follow:

$$f(t_i) = a_i + b_i t_i + c_i t_i^2 + d_i t_i^3 \tag{3.3}$$

And

$$t_i = \frac{(x - x_i)}{x_{i+1} - x_i} \tag{3.4}$$

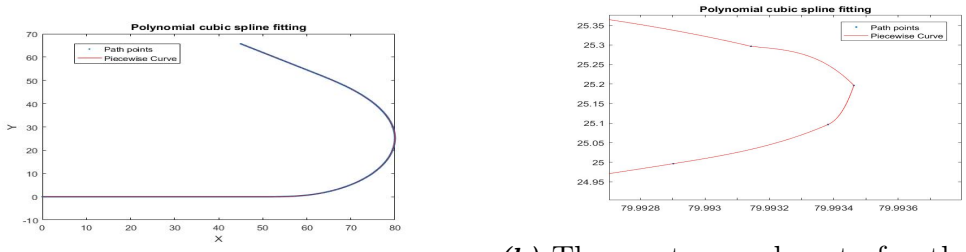
Table 3.2: Piece-wise Cubic Spline Algorithm

Piece-wise Cubic Spline Algorithm	
Iteration i from 1 to N	
Segment i:	
Slope entering (x_i,y_i)	$b_i = (x_{i+1} - x_i)\tan\theta_i$
C^0 continuity at (x_i,y_i)	$a_i = y_i$
C^0 continuity at (x_{i+1},y_{i+1})	$a_i + b_i + c_i + d_i = y_{i+1}$
C^1 continuity at (x_{i+1},y_{i+1})	$b_i + 2c_i + 3d_i - \frac{x_{i+1}-x_i}{x_{i+2}-x_{i+1}} b_{i+1} = 0$
C^2 continuity at (x_{i+1},y_{i+1})	$2c_i + 6d_i - 2\frac{(x_{i+1}-x_i)^2}{(x_{i+2}-x_{i+1})^2} c_{i+1} = 0$
Segment i+1:	
C^0 continuity at (x_{i+1},y_{i+1})	$a_{i+1} = y_{i+1}$
C^0 continuity at (x_{i+2},y_{i+2})	$a_{i+1} + b_{i+1} + c_{i+1} + d_{i+1} = y_{i+2}$
Slope leaving (x_{i+2},y_{i+2})	$b_{i+1} + 2c_{i+1} + 3d_{i+1} = (x_{i+2} - x_{i+1})\tan\theta_{i+2}$

θ_i is the vehicle heading angle at point (x_i, y_i). N is the total path point numbers. Each time every three points are selected to calculate the two segments cubic spline functions and the matrix form is shown as below. The matrix Co is the coefficients values that is expected to be figured out.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & -\frac{x_{i+1}-x_i}{x_{i+2}-x_{i+1}} & 0 & 0 \\ 0 & 0 & 2 & 6 & 0 & 0 & -2\frac{(x_{i+1}-x_i)^2}{(x_{i+2}-x_{i+1})^2} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} Co \end{bmatrix} = \begin{bmatrix} \theta_i(x_{i+1} - x_i) \\ y_i \\ y_{i+1} \\ 0 \\ 0 \\ y_{i+1} \\ y_{i+2} \\ \theta_{i+2}(x_{i+2} - x_{i+1}) \end{bmatrix}$$

3.3.2.2 Simulation Results



(a) Pre-defined path 1

(b) The most curved part of path 1 after zooming in

Figure 3.3: Piece-wise polynomial cubic spline

The Figure.3.3 (a) shows the overall piece-wise cubic spline fitting of the given path points. The Figure.3.3 (b) is the points fitting performance of the most curved path segments after zooming in. It shows that when the tangent is close to 90°, the method fails to enforce the C^1 and C^2 continuity. When the path is more complicated, this method has more unexpected fitting problems. This problem has been solved by introducing the Catmull-Rom cubic spline fitting method which is described in the next part.

3.3.3 Catmull-Rom Spline

3.3.3.1 Algorithm

The Catmull-Rom spline introduces another parameter u to represents x and y respectively as follow:

$$x = a_{xi} + b_{xi}u_i + c_{xi}u_i^2 + d_{xi}u_i^3 \tag{3.5}$$

$$y = a_{yi} + b_{yi}u_i + c_{yi}u_i^2 + d_{yi}u_i^3 \tag{3.6}$$

Table 3.3: Catmull-Rom Algorithm

Catmull-Rom Algorithm	
Add two imaginary points	$(x_0, y_0), (x_{N+1}, y_{N+1})$
Iteration i from 1 to (N-3)	
Each loop picks four points	from (x_i, y_i) to (x_{i+3}, y_{i+3})
Calculate the i_{th} piecewise entering slope	$s_e = \frac{1}{2}(p_{i+2} - p_i)$
Calculate the i_{th} piecewise leaving slope	$s_l = \frac{1}{2}(p_{i+3} - p_{i+1})$
Curve passes through p_{i+1} at $u_i = 0$	$a_x = p_{i+1}(1)$
	$a_y = p_{i+1}(2)$
Curve passes through p_{i+2} at $u_i = 1$	$a_x + b_x + c_x + d_x = p_{i+2}(1)$
	$a_y + b_y + c_y + d_y = p_{i+2}(2)$
Slope at p_{i+1}	$b_x = s_e(1)$
	$b_y = s_e(2)$
Slope at p_{i+2}	$b_x + 2c_x + 3d_x = s_l(1)$
	$b_y + 2c_y + 3d_y = s_l(2)$

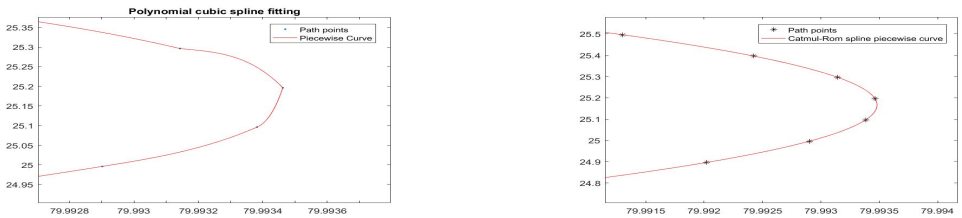
The Table.3.3 above depicts the Catmull-Rom algorithm. N is the total number of the path points after adding two imaginary points. p_i is the point coordinates in the matrix form $\begin{bmatrix} x & y \end{bmatrix}^T$. s_e and s_l are the slopes matrix having the same form as p_i.

Matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} = \begin{bmatrix} p_{i+1}(1) & p_{i+1}(2) \\ p_{i+2}(1) & p_{i+2}(2) \\ s_e(1) & s_e(2) \\ s_l(1) & s_l(2) \end{bmatrix} \tag{3.7}$$

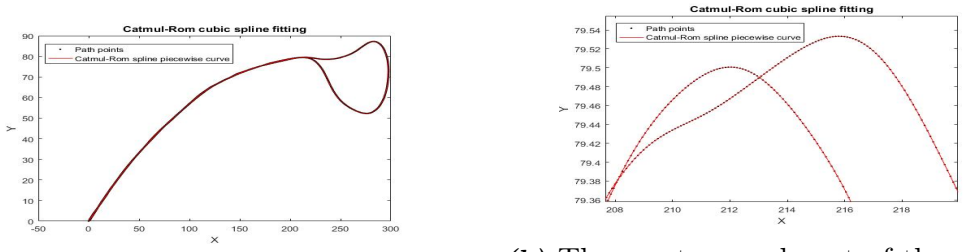
The x and y functions are derived by solving the coefficients for each piece-wise.

3.3.3.2 Simulation Results



(a) Piece-wise polynomial cubic spline ***(b)*** Catmull-Rom Spline

Figure 3.4: Zoom in the most curved part of path 1



(a) Pre-defined path 2

(b) The most curved part of the path 2 after zooming in

Figure 3.5: Catmull-Rom Spline

The Figure.3.4 compares the fitting performance of the polynomial cubic spline and the Catmull-Rom Cubic spline by applying them to the same pre-defined path. The Catmull-Rom Cubic spline fitting algorithm represents a better fitting result. The Figure.3.5 (a) shows the overall Catmull-Rom spline fitting result on the pre-defined path 2. The Figure.3.5 (b) shows how does catmul-Rom spline work in the most curved part of the path 2. In conclusion, the Catmul-Rom cubic spline has a better fitting performance for a more complicated and curved path compared with the piece-wise polynomial spline.

4

Control Strategy

4.1 Introduction

The controller designed in this thesis is for making the model follow the given path with the feedback of the sensor data. The reverse motion control algorithms for the tractor/dolly and the tractor/dolly with semi-trailer are the most tricky parts. VTM is nonlinear and there are also lots of constraints that affect the stability and controllability of the reverse motion controller.

4.2 Literature review

In this section, literature review is done for several control strategies and model linearization methods. The applied control theory is described in detail in the section 4.3 and the section 4.5.

- Lyapunov Controller
  The Lyapunov function [7] is a function for a system with the control inputs. It is applied to design a feedback control law and make the system model behave in a way of asymptotically stable and follow the reference. The lyapunov controller is not implemented in this thesis because it is quite difficult to formulate the Lyapunov function for a complex system and VTM is a quite advanced and complex system.

- Model predictive controller
  The Model predictive control(MPC) [8] is a more advanced controller for controlling a process to satisfy a sets of constraints and it requires a heavy computational resource.

- LQR controller
  The LQR controller [9] is described in detail in the section4.3 since it is implemented to solve the control problem in this thesis.

- Input–output feedback linearization & Jacobian linearization
  The object of the Input-output feedback linearization [10] is to linearize the map between the inputs x and the outputs y. Then a linear controller is designed for the linearized model. The controller ensures the lateral position as well as the vehicle heading to track the desired values. A preview control law

is considered to meet the performance both on a straight road and a curve road [10].

The idea of Jacobian linearization is to linearize the system around a path. It is possible to achieve the local asymptotic stability and it is also easy to be implemented with a linear controller.

Comparing these two linearization methods, the Jacobian linearization is an exact representation of a nonlinear model only at the current operating point and the controller based on Jacobian linearization model may not yield unexpected performance at the other operating points. The Feedback linearization is an exact representation of a nonlinear model over a large set of operating points.

In this thesis, the Jacobian linearization and the LQR controller are investigated and implemented to solve the control problem.

4.3 LQR controller

4.3.1 Motivation

The controller strategy implemented in this thesis is called Linear quadratic regulator controller. There are several reasons why it is chosen here. Firstly, compared with the Lyapunov controller, it is easy to formulate the cost function and perform the parameters tuning. It solves the MIMO control issues easily and it is also easy to prioritize between the states error and the control inputs to evaluate the cost. Secondly, it is computationally fast for the case in the thesis and the calculation of the feedback gain is easy to be derived in the Matlab.

4.3.2 Basic theory

The LQR controller is one type of the optimal control method and it concerns with operating a linear dynamic system at a minimum cost. This regulator law drives its input states to be zero or to be stable at zero closely.

For a linear time variant(LTV) system:

$$\dot{x}(t) = A(t)x(t) + B(t)x(t) \tag{4.1}$$

The LQR works with the linear system to make the cost function reach its minimum value and the cost function is usually represented as a quadratic criterion:

$$\int_0^\infty x^T(t)Qx(t) + u^T(t)Ru(t)dt \tag{4.2}$$

The Q and R variables prioritize between the states error and the control input by weighing the states error and the control signal differently.

The linear feedback control law for minimizing the cost function is represented as:

$$K(t) = R^{-1}B^{T}P(t) \tag{4.3}$$

$$u(t) = -K(t)x(t) \tag{4.4}$$

Where $P(t)$ is derived by solving the Riccati equation:

$$A(t)^{T}P(t) + P(t)A(t) - P(t)B(t)R^{-1}B^{T}P(t) + Q(t) = 0 \tag{4.5}$$

4.4 Path-tracking offsets model

The controller is designed for making the vehicle model track the desired path. The LQR drives the inputs states to be zero with a minimum cost. Thus, a dynamic error model is formulated for the controller design. This error model is also called path-tracking offsets model. This part derives the offsets model for a single unit vehicle as well as for an articulated vehicle.

4.4.1 Single unit vehicle forward & backward offsets model

4.4.1.1 Offsets model geometry

The offsets model geometry for the single unit vehicle is shown as Figure.4.1. The reference point is set at the rear wheel of the vehicle regardless of the moving direction. In the reverse motion, the vehicle body is reversed with the initiated states $(0, 0, \pi)$ compared with the initiated state $(0, 0, 0)$ in forward. The reversing speed is negative refer to the body frame.

In the Figure.4.1, l_{os} is the lateral position error between the desired path position(x_d, y_d) and the vehicle current position (x, y). θ_{os} is the heading error between the desired path heading θ_d and the current vehicle heading θ. R_d is the radius of the path followed by the reference point at the tractor. v is the actual vehicle speed at the reference point and v_d is the expected speed at the current path point.

The geometrical equations for the two error states l_{os} and θ_{os} are represented as:

$$l_{os} = -(x - x_d)sin(\theta_d) + (y - y_d)cos(\theta_d) \tag{4.6}$$

$$\theta_{os} = \theta - \theta_d \tag{4.7}$$

$$\frac{vcos(\theta_{os})}{l_{os} + R_d} = \frac{v_d}{R_d} \tag{4.8}$$

When the vehicle is reversing, the lateral offset l_{os} has the opposite sign compared with the one in forward motion. Because its sign is determined by the vehicle position and the desired heading angle.

The desired curvature k_d is derived from the given path:

$$\frac{\partial\theta_d}{\partial s_d} = k_d \tag{4.9}$$

s_d is the desired distance travelled along the path.

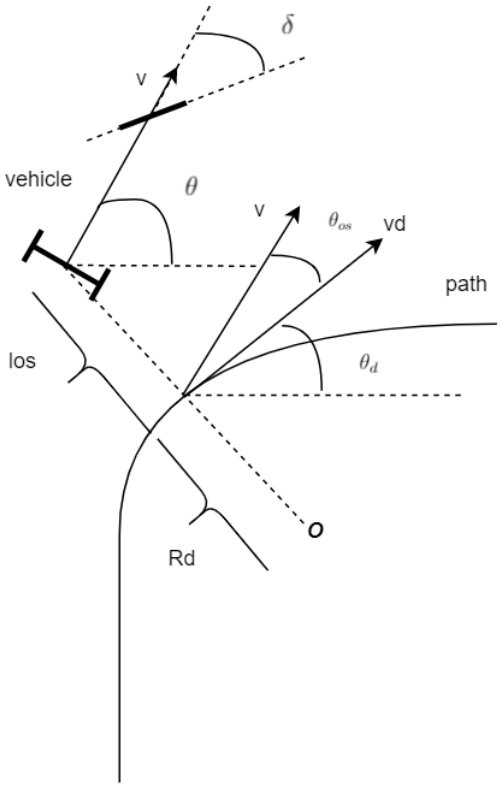


Figure 4.1: Path-tracking offsets model geometry for a single unit vehicle

4.4.1.2 Forward path-tracking offsets model

The path-tracking offsets model is represented as a function of different variables. v is the speed at the reference point.

- Offsets dynamics as a function of the time t

$$\dot{l}_{os} = vsin(\theta_{os}) \tag{4.10}$$

$$\dot{\theta}_{os} = \frac{v}{l}tan\delta - \frac{vcos(\theta_{os})}{l_{os} + R_d} \tag{4.11}$$

- Offsets dynamics as a function of the vehicle driving distance s

$$\frac{\partial l_{os}}{\partial s} = sin(\theta_{os}) \tag{4.12}$$

$$\frac{\partial \theta_{os}}{\partial s} = \frac{tan(\delta)}{l} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.13}$$

- Offsets dynamics as a function of the desired distance travelled along the path s_d

$$\frac{\partial l_{os}}{\partial s_d} = \frac{R_d + l_{os}}{R_d}tan(\theta_{os}) \tag{4.14}$$

$$\frac{\partial \theta_{os}}{\partial s_d} = \frac{R_d + l_{os}}{R_d cos(\theta_{os}) l} tan(\delta) - \frac{1}{R_d} \tag{4.15}$$

The offsets dynamics models as a function of the vehicle driving distance s and as a function of the desired distance travelled along the path s_d are relatively better to be selected for the controller design because they are speed independent which simplifies the model in case of without speed controller. In this thesis, the s dependent offsets model is implemented.

4.4.1.3 Backward path-tracking offsets model

The path-tracking offsets model for reverse motion is similar with the forward one. Since the vehicle kinematic model is same and the only difference is the speed direction and the initiating states of the reference point.

It also has three types functions.

- Offsets dynamic as a function of the time t

$$\frac{\partial l_{os}}{\partial t} = -v sin(\theta_{os}) \tag{4.16}$$

$$\frac{\partial \theta_{os}}{\partial t} = \frac{v tan(\delta)}{l} + \frac{v cos(\theta_{os})}{l_{os} + R_d} \tag{4.17}$$

- Offsets dynamic as a function of the vehicle driving distance s

$$\frac{\partial l_{os}}{\partial s} = sin(\theta_{os}) \tag{4.18}$$

$$\frac{\partial \theta_{os}}{\partial s} = -\frac{tan(\delta)}{l} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.19}$$

- Offsets dynamic function as a function of the desired driving distance s_d

$$\frac{\partial l_{os}}{\partial s_d} = \frac{R_d + l_{os}}{R_d} tan(\theta_{os}) \tag{4.20}$$

$$\frac{\partial \theta_{os}}{\partial s_d} = -\frac{R_d + l_{os}}{R_d cos(\theta_{os}) l} tan(\delta) - \frac{1}{R_d} \tag{4.21}$$

In this thesis, the s dependent offsets model is implemented.

4.4.2 Articulated Vehicle forward & backward offsets model

4.4.2.1 Offsets model geometry

The offsets model geometry is shown as the Figure.4.2. The Cartesian models for articulated vehicle have been described in Chapter 2. The reference point is set at the rear wheel of the tractor/dolly in the forward motion as well as in the reverse motion. From the geometry, the Equation 4.6 - Equation 4.9 are also applicable here. Another relationship between the speed v_2 and v_1 is added. v_1 and v_2 are the vehicle speed at the rear wheel point of the tractor and the semitrailer respectively.

When the vehicle is reversing, v_1 has the negative sign. ϕ is the articulation angle of the tractor and the semitrailer

$$v_2 = v_1 cos(\phi) \tag{4.22}$$

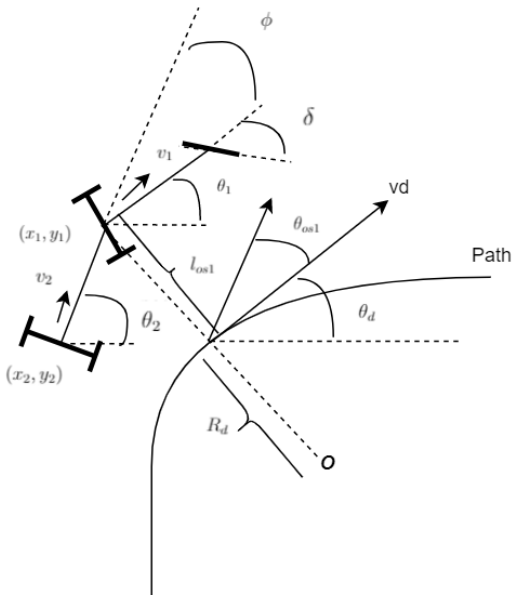


Figure 4.2: Offsets model geometry for the articulated vehicle: the reference point is located at the rear wheel of tractor/dolly

4.4.2.2 Forward path-tracking offsets model

Three types path-tracking offsets model functions are introduced here. The distance s dependent model is selected to be implemented for the controller design.

- Offsets dynamics as a function of the time t

$$\dot{l}_{os} = v_1 sin(\theta_{os}) \tag{4.23}$$

$$\dot{\theta}_{os} = \frac{v_1 \tan(\delta)}{l_1} - \frac{v_1 cos(\theta_{os})}{l_{os} + R_d} \tag{4.24}$$

$$\dot{\phi}_{os} = -v_1(\frac{\sin(\phi)}{l_2} + \frac{\tan(\delta)}{l_1}) \tag{4.25}$$

- Offsets dynamics as a function of the vehicle driving distance s

$$\frac{\partial l_{os}}{\partial s} = sin(\theta_{os}) \tag{4.26}$$

$$\frac{\partial \theta_{os}}{\partial s} = \frac{tan(\delta)}{l_1} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.27}$$

$$\frac{\partial \phi_{os}}{\partial s} = -\frac{\sin(\phi)}{l_2} - \frac{\tan(\delta)}{l_1} \tag{4.28}$$

- Offsets dynamics as a function of the desired distance travelled along the path s_d

$$\frac{\partial l_{os}}{\partial s_d} = \frac{l_{os} + R_d}{R_d} tan(\theta_{os}) \tag{4.29}$$

$$\frac{\partial \theta_{os}}{\partial s_d} = \frac{tan(\delta)(l_{os} + R_d)}{R_d l_1 cos(\theta_{os})} - \frac{1}{R_d} \tag{4.30}$$

$$\frac{\partial \phi_{os}}{\partial s_d} = -\frac{(R_d + l_{os})\sin(\phi)}{l_2 R_d cos(\theta_{os})} - \frac{(R_d + l_{os})tan(\delta)}{l_1 R_d cos(\theta_{os})} \tag{4.31}$$

4.4.2.3 Backward path-tracking offsets model

From the Cartesian model of the reverse model in Chapter 2 and the geometry Equations 4.6 - Equation 4.9, the error dynamic model for the reverse motion is derived and it also has three types functions. The speed v_1 here is negative refer to the vehicle body frame.

- Offsets dynamics as a function of the time t

$$\dot{l}_{os} = -v_1 sin(\theta_{os}) \tag{4.32}$$

$$\dot{\theta}_{os} = \frac{v_1 \tan(\delta)}{l_1} + \frac{v_1 cos(\theta_{os})}{l_{os} + R_d} \tag{4.33}$$

$$\dot{\phi}_{os} = -v_1 \frac{\sin(\phi)}{l_2} - v_1 \frac{\tan(\delta)}{l_1} \tag{4.34}$$

- Offsets dynamics as a function of the vehicle driving distance s

$$\frac{\partial l_{os}}{\partial s} = sin(\theta_{os}) \tag{4.35}$$

$$\frac{\partial \theta_{os}}{\partial s} = -\frac{tan(\delta)}{l_1} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.36}$$

$$\frac{\partial \phi_{os}}{\partial s} = \frac{\sin(\phi)}{l_2} + \frac{\tan(\delta)}{l_1} \tag{4.37}$$

- Offsets dynamics as a function of the desired distance travelled along the path s_d

$$\frac{\partial l_{os}}{\partial s_d} = \frac{l_{os} + R_d}{R_d} tan(\theta_{os}) \tag{4.38}$$

$$\frac{\partial \theta_{os}}{\partial s_d} = -\frac{tan(\delta)(l_{os} + R_d)}{R_d l_1 cos(\theta_{os})} - \frac{1}{R_d} \tag{4.39}$$

$$\frac{\partial \phi_{os}}{\partial s_d} = \frac{(R_d + l_{os})\sin(\phi)}{l_2 R_d cos(\theta_{os})} + \frac{(R_d + l_{os})tan(\delta)}{l_1 R_d cos(\theta_{os})} \tag{4.40}$$

With:

$$\phi = \phi_d + \phi_{os} \tag{4.41}$$

4.5 Model Linearization

The linearization is required to linearize the offset model for designing the linear controller.

4.5.1 Jacobian linearization

The Jacobian linearization is linearizing a nonlinear model around the equilibrium points and it computes the Jacobian matrix J, which is done by computing all its first-order partial derivatives with all its variables.

For a given nonlinear function $f(x)$, this function takes a point $x \in \mathrm{R}^n$ to map the result $f(x) \in \mathrm{R}^m$. x is a scalar from x_1 to x_n. Then the Jacobian matrix of function f is denoted by J with the $(i,j)_{th}$ entry $J_{ij} = \frac{\partial f_i}{\partial x_j}$. More details are shown in the explicit application below for linearizing the path-tracking offsets model.

4.5.2 Offsets model linearization for the single unit vehicle

4.5.2.1 Forward motion offsets model linearization

From the Equation 4.12 - Equation 4.13, a nonlinear system is formulated as:

$$f(l_{os}, \theta_{os}) = sin(\theta_{os}) \tag{4.42}$$

$$g(l_{os}, \theta_{os}) = \frac{tan(\delta)}{l} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.43}$$

δ is the control input. l_{os} and θ_{os} are the two error states. A linear system $\dot{X} = Ax + Bu$ is intended to be solved. x is the states error and u is the input steering angle. For the two error states and one input system, B matrix is easy to be derived as $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ with $u = \frac{tan(\delta)}{l_1}$. The A matrix is derived by computing the Jacobian matrix around zero value of the error states.

$$A = \begin{bmatrix} \frac{\partial f}{\partial l_{os}} & \frac{\partial f}{\partial \theta_{os}} \\ \frac{\partial g}{\partial l_{os}} & \frac{\partial g}{\partial \theta_{os}} \end{bmatrix} \tag{4.44}$$

$$B = \begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial g}{\partial u} \end{bmatrix} \tag{4.45}$$

With $l_{os} = 0$ and $\theta_{os} = 0$. Then:

$$A = \begin{bmatrix} 0 & 1 \\ \frac{1}{R_d^2} & 0 \end{bmatrix} \tag{4.46}$$

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.47}$$

4.5.2.2 Reverse motion offsets model linearization

Similarly, the nonlinear system is formulated as:

$$f(l_{os}, \theta_{os}) = sin(\theta_{os}) \tag{4.48}$$

$$g(l_{os}, \theta_{os}) = -\frac{tan\delta}{l} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.49}$$

with A and B matrix as follow:

$$A = \begin{bmatrix} 0 & 1 \\ \frac{1}{R_d^2} & 0 \end{bmatrix} \tag{4.50}$$

$$B = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{4.51}$$

$$\frac{1}{R_d} = k_d = \frac{\partial \theta_d}{\partial s_d} \tag{4.52}$$

k_d is the desired path curvature, which is calculated along the given path points.

4.5.3 Articulated vehicle model linearization

4.5.3.1 Forward motion offsets model linearization

From the Equation 4.26 - Equation 4.28, the nonlinear system is formulated as:

$$f(l_{os}, \theta_{os}, \phi_{os}) = sin(\theta_{os}) \tag{4.53}$$

$$g(l_{os}, \theta_{os}, \phi_{os}) = \frac{tan(\delta)}{l_1} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.54}$$

$$h(l_{os}, \theta_{os}, \phi_{os}) = -\frac{sin(\phi_{os} + \phi_d)}{l_2} - \frac{tan(\delta)}{l_1} \tag{4.55}$$

Apply Jacobian linearization for the model and get A and B matrix:

$$A = \begin{bmatrix} \frac{\partial f}{\partial l_{os}} & \frac{\partial f}{\partial \theta_{os}} & \frac{\partial f}{\partial \phi_{os}} \\ \frac{\partial g}{\partial l_{os}} & \frac{\partial g}{\partial \theta_{os}} & \frac{\partial g}{\partial \phi_{os}} \\ \frac{\partial h}{\partial l_{os}} & \frac{\partial h}{\partial \theta_{os}} & \frac{\partial h}{\partial \phi_{os}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{R_d^2} & 0 & 0 \\ 0 & 0 & \frac{-cos(\phi_d)}{l_2} \end{bmatrix} \tag{4.56}$$

$$B = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \tag{4.57}$$

With $u = \frac{tan(\delta)}{l_1}$

4.5.3.2 Reverse motion offsets model linearization

From the Equation 4.35 - Equation 4.37, the nonlinear system is formulated as:

$$f(l_{os}, \theta_{os}, \phi_{os}) = sin(\theta_{os}) \tag{4.58}$$

$$g(l_{os}, \theta_{os}, \phi_{os}) = -\frac{tan(\delta)}{l_1} - \frac{cos(\theta_{os})}{l_{os} + R_d} \tag{4.59}$$

$$h(l_{os}, \theta_{os}, \phi_{os}) = \frac{sin(\phi_{os} + \phi_d)}{l_2} + \frac{tan(\delta)}{l_1} \tag{4.60}$$

A and B matrix are calculated around the equilibrium points as:

$$A = \begin{bmatrix} \frac{\partial f}{\partial l_{os}} & \frac{\partial f}{\partial \theta_{os}} & \frac{\partial f}{\partial \phi_{os}} \\ \frac{\partial g}{\partial l_{os}} & \frac{\partial g}{\partial \theta_{os}} & \frac{\partial g}{\partial \phi_{os}} \\ \frac{\partial h}{\partial l_{os}} & \frac{\partial h}{\partial \theta_{os}} & \frac{\partial h}{\partial \phi_{os}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{R_d^2} & 0 & 0 \\ 0 & 0 & \frac{cos(\phi_d)}{l_2} \end{bmatrix} \tag{4.61}$$

$$B = \begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial g}{\partial u} \\ \frac{\partial h}{\partial u} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \tag{4.62}$$

R_d is calculated as the same way of the Equation 4.52. ϕ_d is consumed to be constant for each point and is delivered from the path geometry.

4.6 Control Algorithm overview

The LQR controller is implemented in this thesis to drive all error states to be zero by controlling the input variable u. With the control law $u = -Kx$, the gain K is computed directly in Matlab using the lqr command or calculated by solving the Ricatti equation. In the single unit vehicle motion control, the Ricatti equation is solved mathematically to get this gain as well as the control inputs since the single unit model is not complex and the gain K is possible to be calculated for each simulation step. However, for the articulated vehicle control, the Ricatti equation is not solved easily in Matlab. The gain K is not calculated in each simulation step and it is calculated for all path points beforehand. If the path segments which are designed in the Chapter 3 consists of more interpolation points, then the calculated gain K is more close to the real time calculation result. Tuning Q and R parameters also has a big influence on the control results. More details about tuning Q and R parameters are shown and explained in the next simulation part. The parameter Q is defined as $\begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$ and R is a matrix of R_1x1. q_1, q_2 and R_1 are constant value.

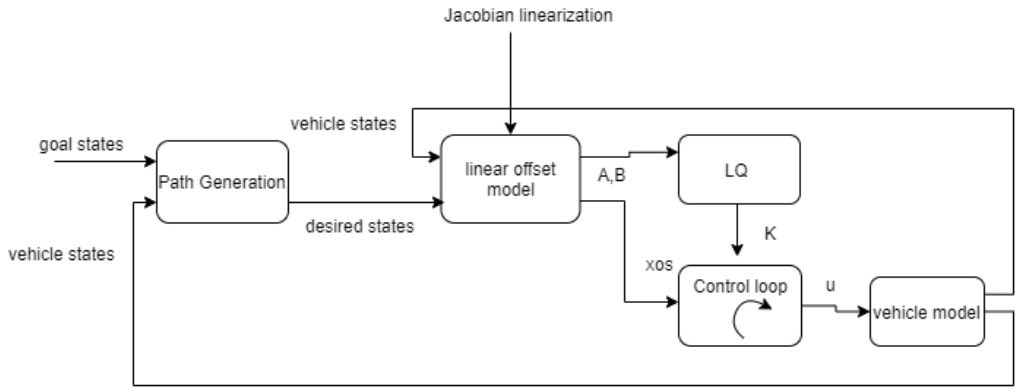


Figure 4.3: Complete control algorithm overview

5

Simulation and Results

5.1 Introduction

5.2 VTM Parameters for simulation

The Table.5.1 and The Table.5.2 below show parameters used in the Matlab simulation for VTM. These parameters are also shown in the Fig.2.5.

Name	Value	Description
l_1	4.085	wheelbase of tractor
l_2	7.725	wheelbase of semi-trailer
T	0.001s	VTM sampling time

Table 5.1: Tractor with trailer simulation parameters

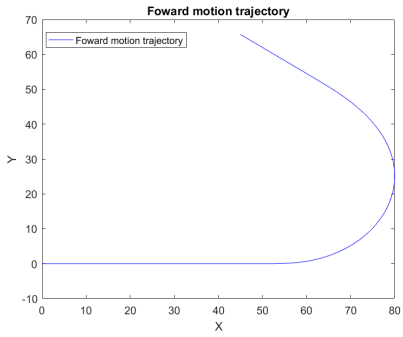
Name	Value	Description
l_1	1.32	wheelbase of Dolly
l_2	7.295	wheelbase of semi-trailer
T	0.001s	VTM sampling time

Table 5.2: Dolly with trailer simulation parameters

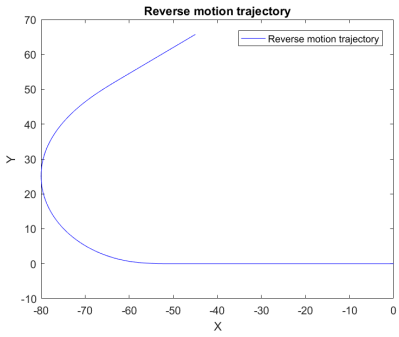
5.3 Single unit vehicle testing

5.3.1 Forward and reverse trajectory

The control algorithm for the signal unit vehicle is performed on the VTM (Virtual Truck Model). The algorithm is designed based on the kinematic model and then tested on VTM. The testing includes both the forward motion and reverse motion path following, which shows how the controller and algorithms work to achieve the autonomous forward and backward motion. The Fig.5.1(a) and the Fig.5.1(b) depict the pre-defined trajectory for forward and backward motion respectively. For the backward moving, the initial states and the body frame of the vehicle are kept the same as the forward motion, While the trajectory is reformulated to be adapted to the reverse motion frame. The Q and R parameters of LQR controller are tuned for the formulated offsets dynamics model which is described in the Chapter 4.



(a) Forward motion trajectory



(b) Reverse motion trajectory

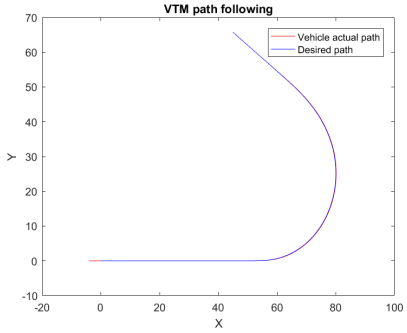
Figure 5.1: Single unit vehicle motion trajectory

5.3.2 Forward Motion Testing

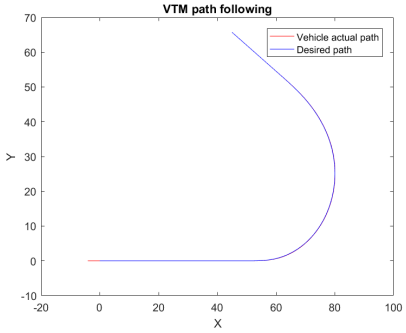
5.3.2.1 Simulation results

The simulation is performed on the tractor model in this thesis. The dolly model is almost the same as the tractor model only with the change of the wheelbase. The controller works for a low speed ranging from 3 m/s to 9 m/s with the same control parameter, since the controller is designed as low-speed independent. The simulation results are shown and compared in the different scenarios with the speed at 3 m/s and 9 m/s.

- $v = 3\ m/s$

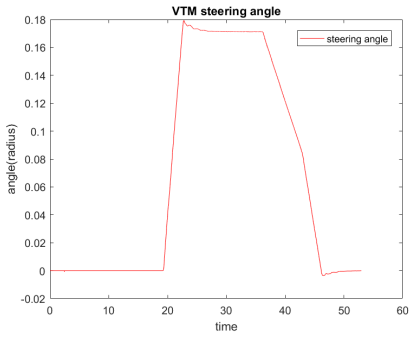


(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$

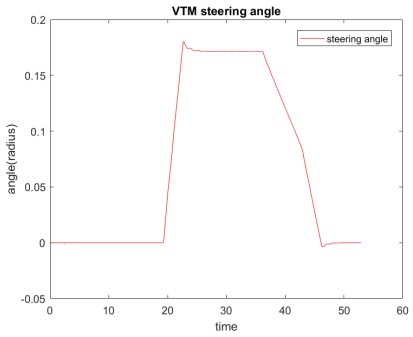


(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.2: Path following

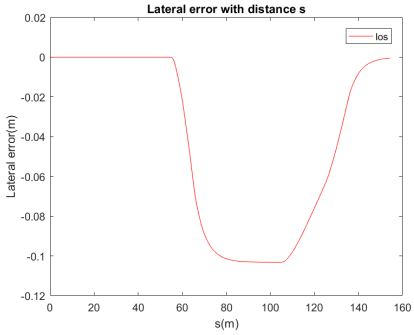


(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$

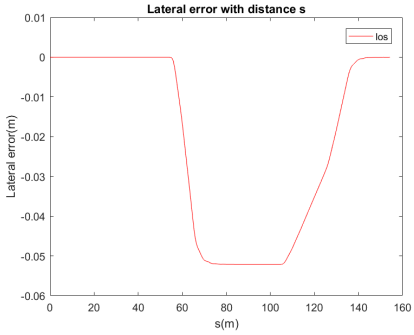


(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.3: Steering behaviour

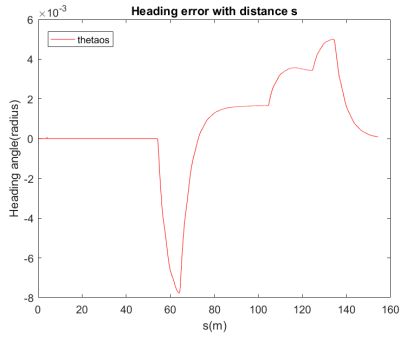


(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$

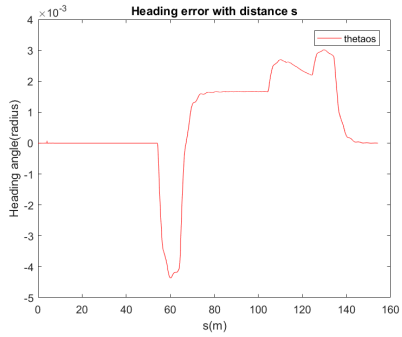


(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.4: Lateral error with distance



(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$



(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.5: Heading angle error with distance

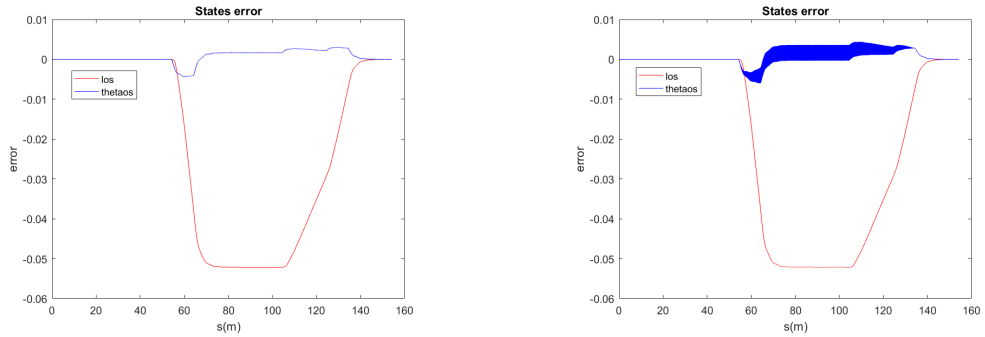


Figure 5.6: Smoothing and Non-smoothing heading angle simulation with $R = 1.0/q_1 = 0.8/q_2 = 5.0$

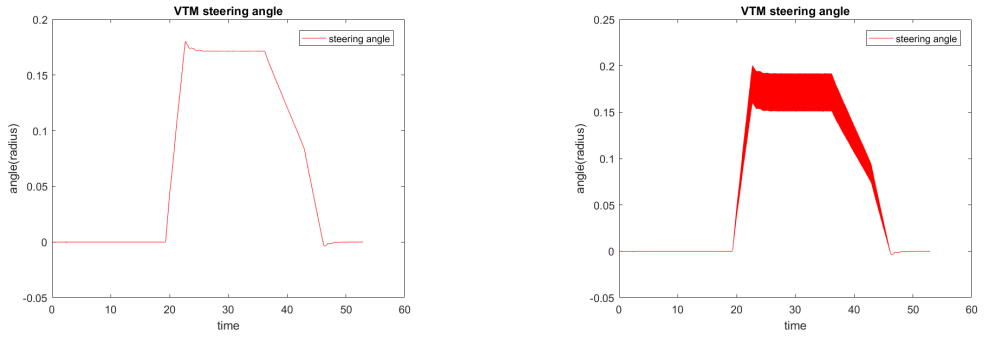
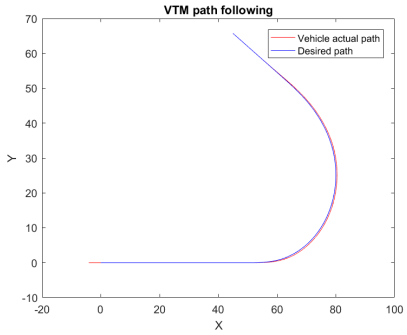


Figure 5.7: Smoothing and Non-smoothing heading angle simulation with $R = 1.0/q_1 = 0.8/q_2 = 5.0$

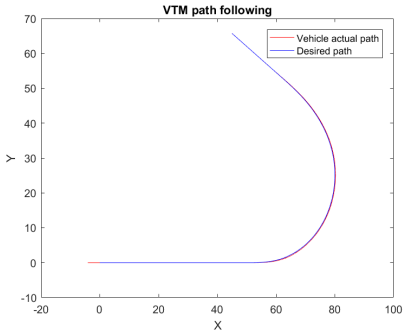
The Fig.5.2 - Fig.5.5 show the comparison of VTM simulation results with two different chosen Q and R parameters. The control parameter q_1 penalizes the lateral error and q_2 penalizes the heading error and R is the weight of the steering angle. The states error is narrowed by increasing q_1 from 0.2 to 0.8 and keeping the value of the parameter R and parameter q_2. The follower is more precise while it makes the steering angle becomes slighter fluctuating. Since it is in a low speed $v = 3\ m/s$, the fluctuation is quite slight. At the beginning of the path, the lateral and heading error is zero and when it goes to the first curve, the states offsets are increased then decreased to zero again. The steering angle is under control and within the constraints $\pm 20°$.

The Fig.5.6 - Fig.5.7 represent the simulation results compared between the trajectory with the non-smoothing heading angle and with the smoothing heading angle. Without the smooth heading angle, the steering fluctuates dramatically. The smoothing algorithm is introduced and implemented in the Chapter 3 in detail.

- $v = 9\ m/s$

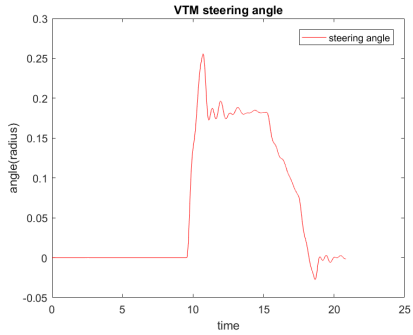


(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$

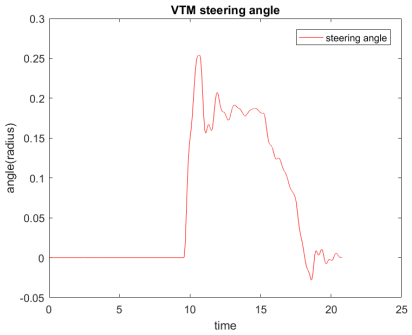


(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.8: Path following

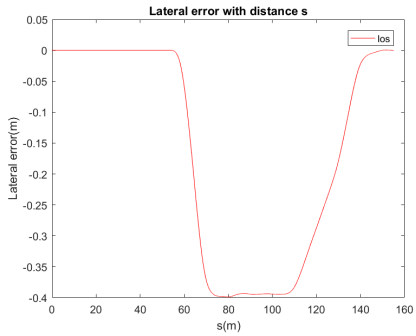


(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$

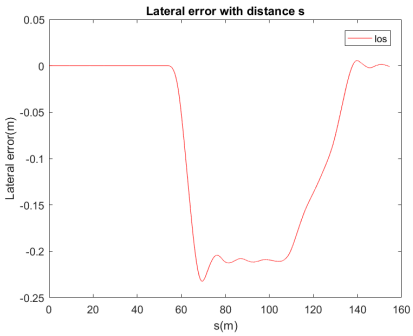


(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.9: Steering behaviour

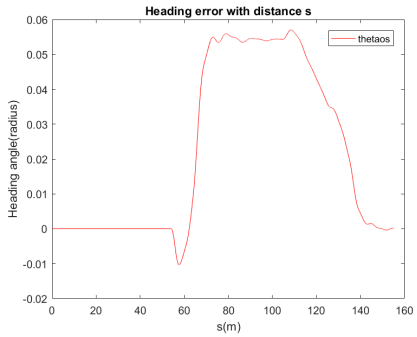


(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$

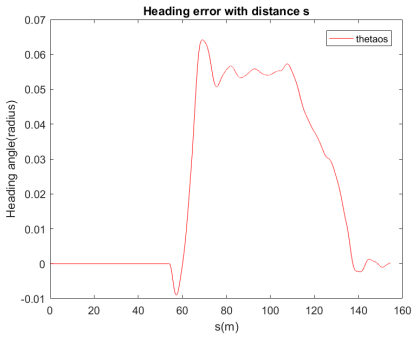


(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.10: Lateral error with distance

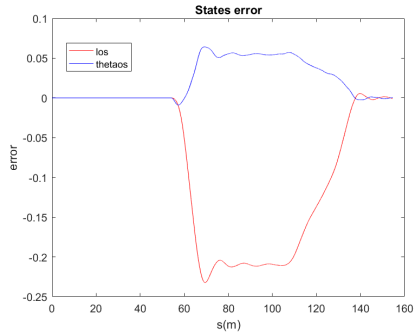


(a) $R = 1.0/q_1 = 0.2/q_2 = 5.0$



(b) $R = 1.0/q_1 = 0.8/q_2 = 5.0$

Figure 5.11: Heading angle error with distance



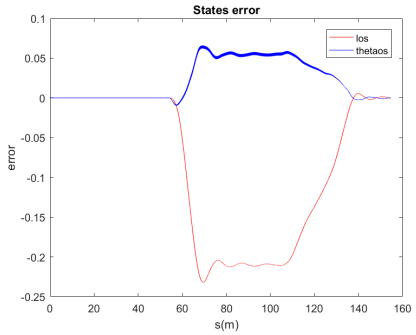
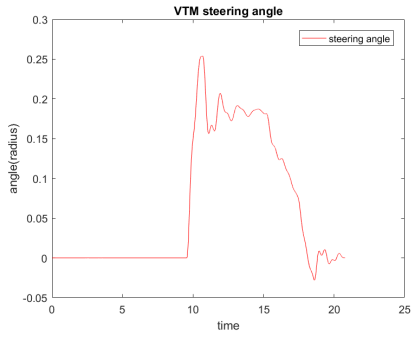


Figure 5.12: Smoothing and Non-smoothing heading angle simulation with $R = 1.0/q_1 = 0.8/q_2 = 5.0$



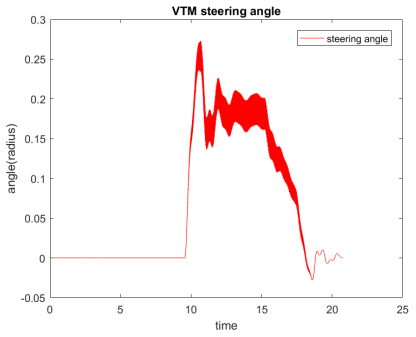


Figure 5.13: Smoothing and Non-smoothing heading angle simulation with $R = 1.0/q_1 = 0.8/q_2 = 5.0$

The Fig.5.8 - Fig.5.11 show the simulation results when $v = 9\ m/s$ with the same control parameters which are used in the case of $v = 3\ m/s$. The controller still works with the speed increased to 9 m/s. Compared with the lower speed at $v = 3\ m/s$, the lateral and heading errors increase a little bit and the steering is more fluctuating. By increasing the parameter q_1 from 0.2 to 0.8, the states errors become smaller while this also leads to less customer comfort.

The Fig.5.12 - Fig.5.13 represent the comparison of the follower precision as well as

the steering behaviour between the trajectory with the smooth heading angle and the non-smoothing heading angle. It shows that it is important to make the heading angle smoothed to achieve a relatively stable steering behaviour especially with a simple linear controller.

The Table.5.3 - Table.5.4 below show the tuning results of Q and R parameters.

- $v = 3\ m/s$

evaluations / $R/q_1/q_2$	Max(l_{os})	Max(θ_{os})	Aver(l_{os})	Aver(θ_{os})	Max(δ)
1.0/0.2/5.0	0.1032	0.0078	0.0411	0.0016	0.1793
1.0/0.8/5.0	0.0522	0.0044	0.0207	0.0012	0.1803
1.0/1.5/5.0	0.0383	0.0034	0.0152	0.0010	0.1804

Table 5.3: Evaluations of the control parameter when $v = 3\ m/s$

- $v = 9\ m/s$

evaluations / $R/q_1/q_2$	Max(l_{os})	Max(θ_{os})	Aver(l_{os})	Aver(θ_{os})	Max(δ)
1.0/0.2/5.0	0.3989	0.0570	0.1335	0.0188	0.2557
1.0/0.8/5.0	0.2322	0.0642	0.0709	0.0188	0.2539
1.0/1.5/5.0	0.1799	0.0637	0.0537	0.0188	0.2519

Table 5.4: Evaluations of the control parameters when $v = 9\ m/s$

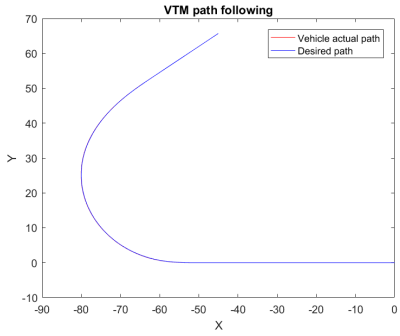
In the tables above, the steering angle δ is limited between -(π/9) and (π/9). Max(l_{os}) and Aver(l_{os}) are the maximum and average absolute value of the lateral errors respectively. Max(θ_{os}) and Aver(θ_{os}) are the maximum and average absolute value of the heading error respectively. Max(δ) represents the maximum absolute value of steering angle in radians. According to the LQR control strategy, the R parameter penalizes the steering input. The parameter q_1 and q_2 penalize the lateral error and the heading error respectively.

5.3.3 Reverse Motion testing

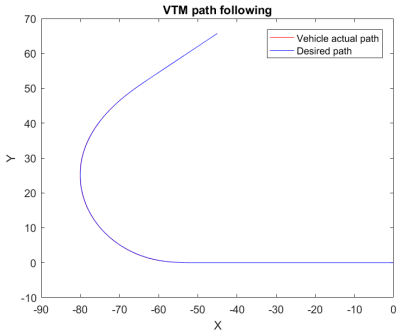
5.3.3.1 Simulation results

The simulation results are shown and compared from different scenarios, which represent how the controller works for the reverse motion at two certain speeds with the same control parameter. It also represents the control performance at one speed with a different control parameter. The simulations are performed only with the smoothing head angle.

- $v = -1\ m/s$ and $v = -3\ m/s$ with $R = 0.07$, $q_1 = 0.3$, $q_2 = 1.0$

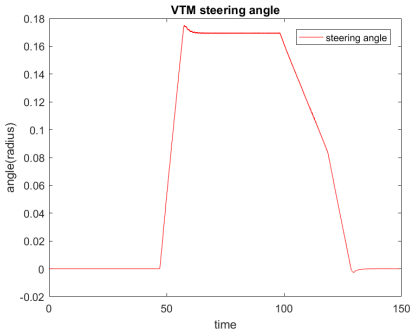


(a) $v = -1\ m/s$

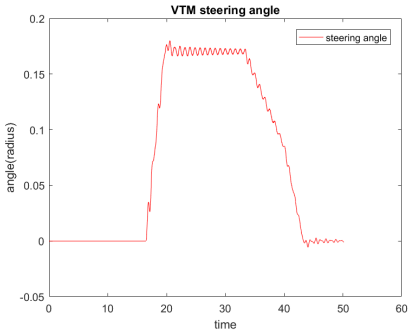


(b) $v = -3\ m/s$

Figure 5.14: Path following

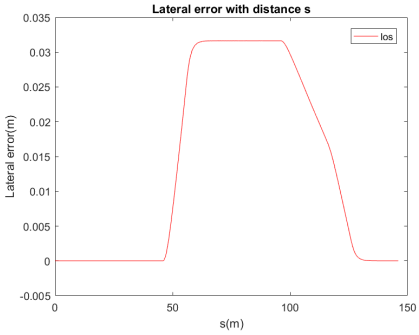


(a) $v = -1\ m/s$

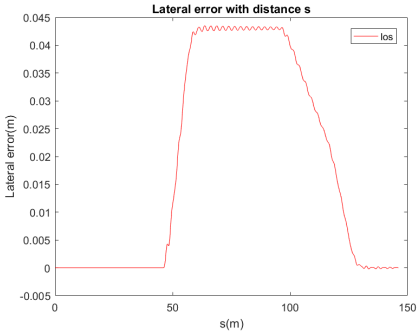


(b) $v = -3\ m/s$

Figure 5.15: Steering behaviour

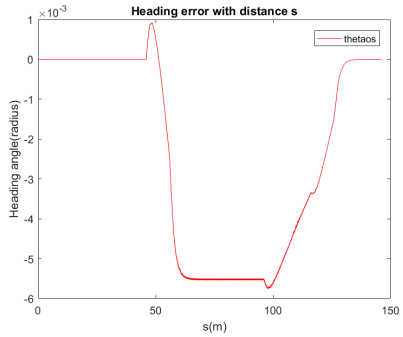


(a) $v = -1\ m/s$

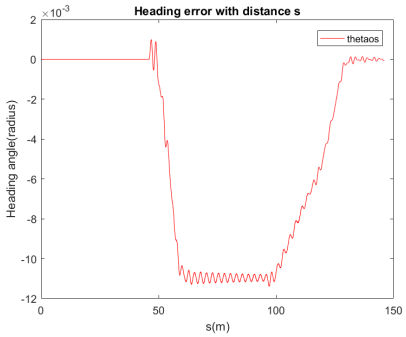


(b) $v = -3\ m/s$

Figure 5.16: Lateral error with distance



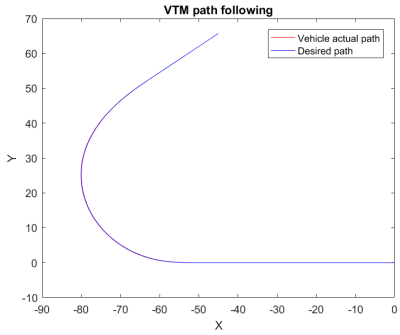
(a) $v = -1\ m/s$



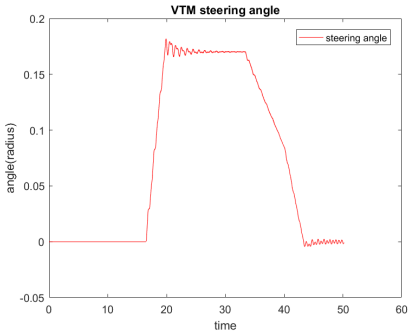
(b) $v = -3\ m/s$

Figure 5.17: Heading angle error with distance

- $v = -3\ m/s$ with $R = 0.07$, $q_1 = 0.1$, $q_2 = 1.0$

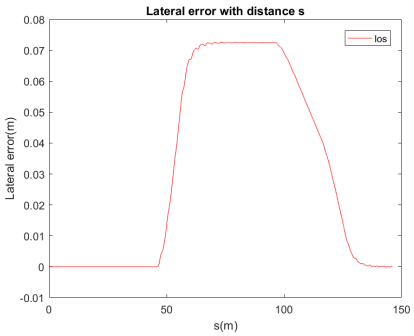


(a) Steering behaviour

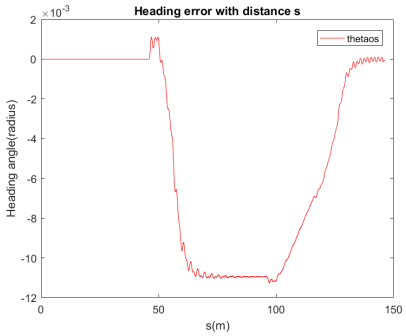


(b) Lateral and heading error

Figure 5.18: Heading angle error with distance



(a) Steering behaviour



(b) Lateral and heading error

Figure 5.19: Heading angle error with distance

The Fig.5.16 - Fig.5.17 show the results comparison between $v = -1\ m/s$ and $v = -3\ m/s$ with the same Q and R parameter values. It represents that increasing speed in reverse moving direction, the steering becomes a little bit oscillating within a small range $\pm 0.0008\ radians$. This is because when the vehicle is moving backward, the controller is more sensitive to the speed increment. In addition, there is no speed controller implemented in this thesis which also affects the performance. The path following precision is still high. The maximum lateral errors at $v = -1\ m/s$ and $v = -3\ m/s$ are both less than 4.4 cm and the maximum heading errors at these two speeds are both less than 0.012 $radians$.

The Fig.5.18 - Fig.5.19 show that at the certain speed $v = -3\ m/s$ when the parameter q_1 is decreased to 0.1 from 0.3, the steering is more stable with less oscillation while the lateral error is increased a little bit. Thus, Penalizing state error less relatively puts more wight on steering input and it is good to the customer comfort.

The Table.5.5 - Table.5.6 below show the evaluations of the control parameters under $v = -3\ m/s$ and $v = -1\ m/s$ in VTM.

- $v = -1\ m/s$

evaluations / $R/q_1/q_2$	Max(l_{os})	Max(θ_{os})	Aver(l_{os})	Aver(θ_{os})	Max(δ)
0.07/0.06/1.0	0.0690	0.0060	0.0295	0.0025	0.1750
0.07/0.1/1.0	0.0538	0.0059	0.0230	0.0025	0.1751
0.07/0.3/1.0	0.0317	0.0058	0.0135	0.0024	0.1751

Table 5.5: Evaluations of the control parameters when $v = -1\ m/s$

- $v = -3\ m/s$

evaluations / $R/q_1/q_2$	Max(l_{os})	Max(θ_{os})	Aver(l_{os})	Aver(θ_{os})	Max(δ)
0.07/0.06/1.0	0.0927	0.0113	0.0388	0.0046	0.1804
0.07/0.1/1.0	0.0727	0.0113	0.0303	0.0046	0.1819
0.07/0.3/1.0	0.0435	0.0114	0.0180	0.0046	0.1801

Table 5.6: Evaluations of the control parameters when $v = -3\ m/s$

In the tables, Max(l_{os}) and Max(θ_{os}) are the maximum absolute value of the states error. Aver(l_{os}) and Aver(θ_{os}) represent the average value of the absolute states error. Max(δ) is the maximum steering angle. The steering angle changes within the constraints $\pm 20°$ (from -0.3491 $radians$ to 0.3491 $radians$).

The tables show that under the given speed when the parameters are $R = 0.2$ and $q_1 = 0.1$, $q_2 = 1.0$, the maximum and average states errors are minimised and also the maximum steering angle is smallest. But compared with the forward motion control, the errors are slightly bigger. Since when the vehicle is moving backward, it is more tricky to control its motion with a linear speed-independent controller.

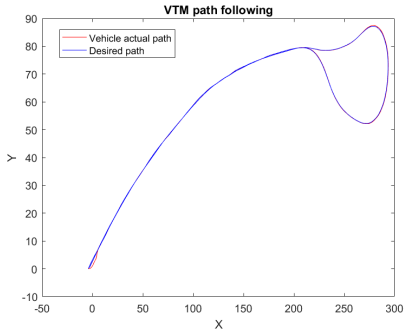
5.4 Articulated vehicle testing

This part is testing the controller on the VTM of the tractor-semitrailer. Since the controller is a simple linear one and also the offsets controller gain is calculated offline along the trajectory. Thus, the control performance is not good.

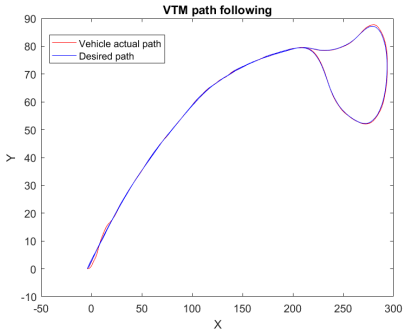
5.4.1 Forward Motion Testing

The simulation is performed with the VTM when the vehicle is running forward at the speed of 3 m/s and 7 m/s with the same control parameter. The controller is designed as low-speed independent, which is expected to be able to handle the different speed scenarios.

5.4.1.1 Simulation results

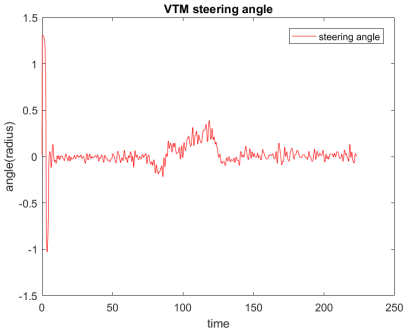


(a) $v = 3\ m/s$

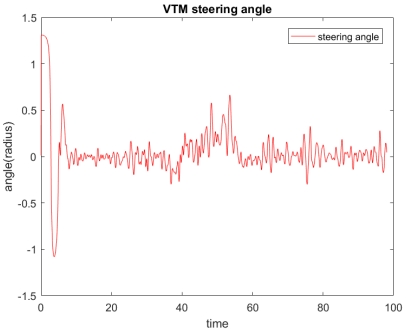


(b) $v = 7\ m/s$

Figure 5.20: Path following

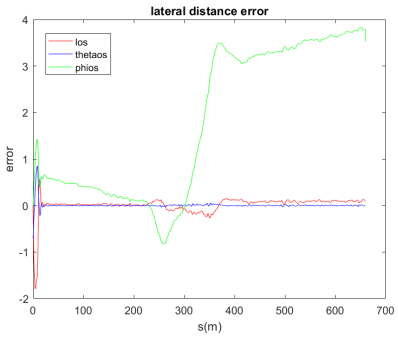


(a) $v = 3\ m/s$

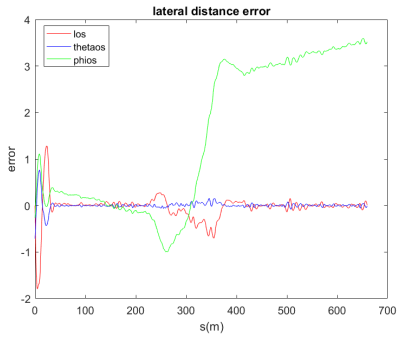


(b) $v = 7\ m/s$

Figure 5.21: Steering behaviour



(a) $v = 3\ m/s$



(b) $v = 7\ m/s$

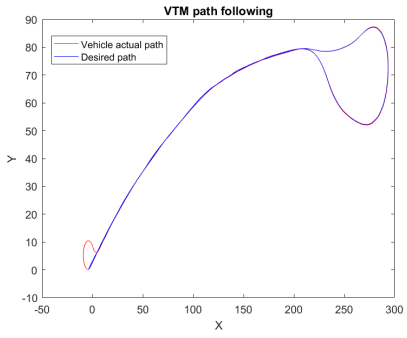
Figure 5.22: States error

The Fig.5.20- Fig.5.22 above show the comparison of the controller performance between the speed at 3 m/s and the speed at 7 m/s on VTM. The tracking point is located at the center of rear wheel of the tractor and the initial state of the reference point is $(0, 0, 0, 0)$, which means the vehicle starts at the point $(0, 0)$ and has zero heading angle and zero articulation angle. The simulation shows it has a relative big error and also big steering angle to drive the vehicle to follow the steering path at the beginning. Compare the control performance at these two speeds with the same control parameters, the steering angle is less stable and following accuracy is also lower when the speed is relatively higher. It represents that the control performance with the simple linear LQR controller and offline control gain calculation is not good for a complicate double units vehicle to move autonomously. Thus, a more complex and dynamic controller is required for to improve the controller performance for the double unit vehicle autonomous motion .

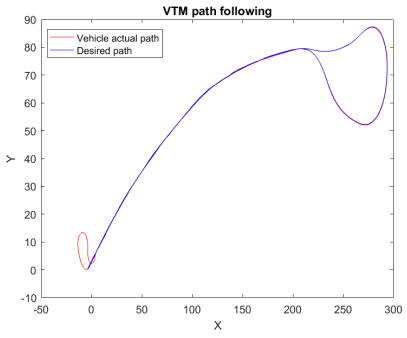
5.4.2 Reverse Motion Testing

When it is moving backward, the low speed scenarios of -1 m/s and -3 m/s are tested on the VTM. The heading angle of the trajectory is reformulated refer to the reverse body frame.

5.4.2.1 Simulation results

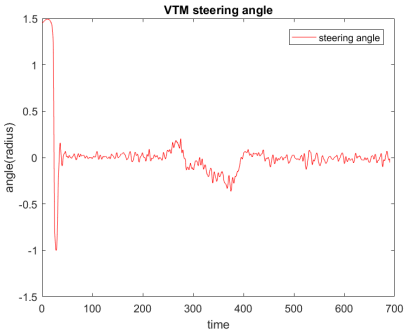


(a) $v = -1\ m/s$

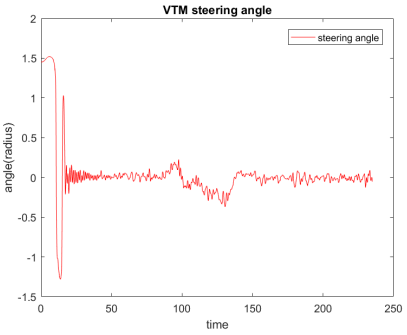


(b) $v = -3\ m/s$

Figure 5.23: Path following

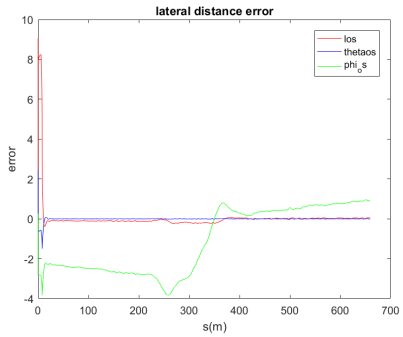


(a) $v = -1\ m/s$

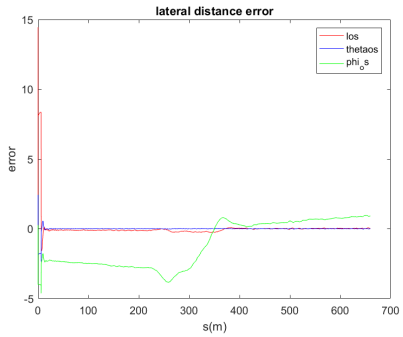


(b) $v = -3\ m/s$

Figure 5.24: Steering behaviour



(a) $v = -1\ m/s$



(b) $v = -3\ m/s$

Figure 5.25: States error

The Fig.5.23- Fig.5.25 show the comparison of the controller performance between the speed at -1 m/s and the speed at -3 m/s on VTM with the same LQR controller parameter. The reference point is put at the center of the real wheel of the tractor. Compare with the forward motion, the reverse motion has a bigger error of following and also the more oscillating steering angle. Compare the performance at two speeds, the performance is relatively worse with the higher reversing speed. The speed controller is not implemented and the implemented controller is linear and simple. It is quite difficult for the simple linear controller with the off-line calculated control gain to control the double units vehicle to move backward autonomously.

6

Conclusion and Future work

6.1 Conclusions

Based on the formulated mathematical model of a tractor and a tractor with a semitrailer, a forward and reverse motion controller have been developed and implemented.

Based on the given predefined trajectory, a smoothing algorithm for the heading angle has been investigated and implemented to make the simple linear controller perform better along the trajectory.

The Controller works well and tracks the path closely in the single unit (tractor) autonomous motion. The controller is designed to be low-speed independent and it works good with the speed up to 9 m/s in the forward motion and to -3 m/s in the reverse motion. A precision of lateral error less than 4 cm and less than 4.5 cm are achieved with the speed at 3 m/s in forward and with the speed at -3 m/s in reverse respectively.

For the control of the combination vehicle autonomous motion, the linear LQR controller does not work well. A real time solution of the Ricatti Equation is not solved and implemented. Then the offline LQR calculation is applied in this thesis. This means the LQR parameters are calculated for the given trajectory beforehand instead of being calculated for each simulation step. This makes the controller gain not been updated with the real time curvature and leads to an inaccurate and unstable controller. Another reason is the controller implemented in this thesis is a simple linear controller without considering dynamics and constraints. In addition, the controller is designed as a speed independent one which also affects the steering behaviour and the tracking precision.

6.2 Future work

The algorithm and controller developed in this thesis could maneuver the single unit vehicle as well as the double units vehicle to follow a predefined path in forward and backward. However, in order to achieve a more precise and stable autonomous vehicle motion, there are some steps left to be investigated and finished in the future.

- The next step to continue this thesis work would be investigating a better dynamic controller for the tractor with a semi-trailer to track the given path in a more stable and smooth way.
- All sensor data in this thesis is considered as noise free and stable. While in reality the noise cannot be ignored. Thus, the sensor fusion algorithm should be developed and implemented with the accuracy consideration. Then the physical testing should be considered.
- To achieve a complete autonomous motion, a further step of including obstacle avoidance should be made.

Bibliography

[1] R. Rajamani, Vehicle Dynamics and Control, Mechanical Engineering Series, DOI 10.1007/978-1-4614-1433-9_2, ©Rajesh Rajamani 2012.

[2] DeSantis, R.M., 1998, “Path-tracking for articulated vehicles via exact and Jacobian linearization,” Proc.of IAV’ 98, Madrid,Spain.

[3] DeSantis, R.M. and Bourgeot, Jean-Matthieu and Todeschi,J.N. and Hurteau, Richard,2002,02, “Path-tracking for tractor-trailers with hitching of both the on-axle and the off-axle kind" ISBN 0-7803-7620-X, DOI 10.1109/ISIC.2002.1157763, pp.206-211.

[4] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera, “Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles,” Journal of Advanced Transportation, vol. 2018, Article ID 6392697, 10 pages, 2018. https://doi.org/10.1155/2018/6392697

[5] Computer Vision, Graphics, and Image Processing Volume 36, Issue 1, October 1986, Pages 10-30, Son Pham

[6] D.Salomon, The Computer Graphics Mannual, Texts in Computer Science, DOI 10.1007/978-0-85792-866-7_12,©Springer-Verlag London Limited 2011.

[7] Alcalá, E., Puig, V., Quevedo, J., Escobet, T., Comasolivas, R. (2018). Autonomous vehicle control using a kinematic Lyapunov-based technique with LQR-LMI tuning.

[8] David Q. Mayne. 2014. Model predictive control. Automatica 50, 12 (December 2014), 2967-2986. DOI: http://dx.doi.org/10.1016/j.automatica.2014.10.128

[9] Hoel, C.; Falcone, P. (2013) "Low speed maneuvering assistance for long vehicle combinations". IEEE Intelligent Vehicles Symposium, Proceedings pp. 598-604.

[10] Rajamani, R Zhu, C Alexander, L. (2003). Lateral control of a backward driven font-steering vehicle. Control Engineering Practice. 11. 531-540. 10.1016/S0967-0661(02)00143-0.

A

Appendix

A.1 Spline Curve Fitting Algorithm

```
% Catmull-Rom Spline curve fitting (MATLAB)

s=0.1;
x_ps=x_p(1)+s*cos(theta_p(1));
y_ps=y_p(1)+s*sin(theta_p(1));
x_pl=x_p(length(x_p))-s*cos(theta_p(length(x_p)));
y_pl=y_p(length(x_p))-s*sin(theta_p(length(x_p)));
x_p=[x_ps;x_p;x_pl];
y_p=[y_ps;y_p;y_pl];
xx_=[];
yy_=[];
co_=[];

for i=1:(length(x_p)-3)
    p1=[x_p(i);y_p(i)];
    p2=[x_p(i+1);y_p(i+1)];
    p3=[x_p(i+2);y_p(i+2)];
    p4=[x_p(i+3);y_p(i+3)];
    t1=(1/2)*(p3-p1);
    t2=(1/2)*(p4-p2);
   M=[1  0  0  0;
       1  1  1  1;
       0  1  0  0;
       0  1  2  3];
    yy=[p2(1)  p2(2);p3(1)  p3(2);
    dpi(1)  dpi(2);dpi_1(1)  dpi_1(2)];
    co=inv(M)*yy;
  u=linspace(0,1, 100);
  x_=co(1,1)+co(2,1)*u+co(3,1)*u.^2
     +co(4,1)*u.^3;
  y_=co(1,2)+co(2,2)*u+co(3,2)*u.^2
     +co(4,2)*u.^3;
  xx_(i,:)=x_;
  yy_(i,:)=y_;
```

```
    co_(1:4,i)=co(:,1);
    co_(5:8,i)=co(:,2);
end

figure()
plot(x_p,y_p,'k.');
hold on;
for i=1:(length(x_p)-3)
    plot(xx_(i,:),yy_(i,:),'r');
    hold on;
end
legend('Path Points','Catmul-Rom spline piecewise curve')
```

A.2 Path Generation Algorithm

```
% Path generation for tractor forward motion (MATLAB)

function[xd,yd,thetad,cd,done]=pathplanner(x,y,x_p,y_p,
        theta_p,co_)
    done=false;
    persistent i
    if(isempty(i))
        i=1;
    end
    if x<x_p(1)
        cd=0;
        xd=(((x-x_p(1))^2)*x-((y_p(1)-y)*(x_p(1)*y-x*y_p(1)))
            -((y_p(1)-y))*((x-x_p(1)))*y)/((y_p(1)-y)^2
            +(x-x_p(1))^2);
        yd=(((y_p(1)-y)^2)*y-((x-x_p(1))*(x_p(1)*y-x*y_p(1)))
            -((y_p(1)-y))*((x-x_p(1)))*x)/((y_p(1)-y)^2
            +(x-x_p(1))^2);
        thetad=atan2(y_p(1)-y,x_p(1)-x);
    else
        if (i== length(x_p))
            done=true;
            thetad=theta_p(i);
            xd=x_p(i);
            yd=y_p(i);
            ds=sqrt((y_p(i)-y_p(i-1))^2+(x_p(i)-x_p(i-1))^2);
            dtheta=theta_p(i)-theta_p(i-1);
            cd=dtheta/ds;
        else
            thetad=theta_p(i);
            x_n=cos(thetad)*x_p(i+1)+sin(thetad)*y_p(i+1);
            x_c=cos(thetad)*x+sin(thetad)*y;
            if (x_c>=x_n)
                i=i+1;
            else
            end
            if (i== length(x_p))
                done=true;
                thetad=theta_p(i);
                xd=x_p(i);
                yd=y_p(i);
                ds=sqrt((y_p(i)-y_p(i-1))^2
                    +(x_p(i)-x_p(i-1))^2);
                dtheta=theta_p(i)-theta_p(i-1);
                cd=dtheta/ds;
```

```
            else
              n=1000;
              u=linspace(0,1,n);
              x_i=co_(1,i)+co_(2,i)*u+co_(3,i)*u.^2
                   +co_(4,i)*u.^3;
              y_i=co_(5,i)+co_(6,i)*u+co_(7,i)*u.^2
                   +co_(8,i)*u.^3;
              dx=co_(2,i)+2*co_(3,i)*u +3*co_(4,i)*u.^2;
              dy=co_(6,i)+2*co_(7,i)*u +3*co_(8,i)*u.^2;
              s=dy./dx;
             distt=zeros(length(u),1);
             for j=1:length(u)
                  ex=x_i(j)-x;
                  ey=y_i(j)-y;
                  dist=sqrt(ex^2+ey^2);
                  distt(j)=dist;
             end
             [M,N]=min(distt(1:length(u)));
             xd=x_i(N);
             yd=y_i(N);
            end
            if N<n
                 if atan(s(N))<0 && y_i(N+1)>y_i(N)
                    && x_i(N+1)<x_i(N)
                   thetad=atan(s(N))+pi;
                 elseif atan(s(N))>0 && y_i(N+1)<y_i(N)
                        &&x_i(N+1)<x_i(N)
                   thetad=atan(s(N))-pi;
                 else
                   thetad=atan(s(N));
                 end
                 ds=sqrt((y_i(N+1)-y_i(N))^2
                    +(x_i(N+1)-x_i(N))^2);
                 thetapp=linspace(theta_p(i),theta_p(i+1),n);
                 dtheta=thetapp(N+1)-thetapp(N);
                 cd=dtheta/ds;
            else
                 thetad=theta_p(i+1);
                 ds=sqrt((y_i(N)-y_i(N-1))^2
                    +(x_i(N)-x_i(N-1))^2);
                 thetapp=linspace(theta_p(i),theta_p(i+1),n);
                 dtheta=thetapp(N)-thetapp(N-1);
                 cd=dtheta/ds;
            end
        end
    end
```

end