FOI
Defence Research Agency
Systems Technology
SE-164 90 Stockholm

Phone: +46 8 555 030 00
Fax: +46 8 555 031 00

www.foi.se

Control and dynamic modeling of an Autonomous Ground Vehicle

HELENE MÖRTBERG

FOI-R--1911--SE
ISSN 1650-1942

Technical report
January 2006

Systems Technology

Helene Mörtberg

Control and dynamic modeling of an Autonomous Ground Vehicle

FOI-R--1911--SE Technical report **Systems Technology**
ISSN-1650-1942 January 2006

Issuing organisation FOI – Swedish Defence Research Agency Systems Technology SE-164 90 STOCKHOLM	**Report number, ISRN** FOI-R--1911--SE	**Report type** Technical report
	Research area code Mobility and Space Technology, incl Materials	
	Month year January 2006	**Project no.** E6041
	Sub area code Unmanned Vehicles	
	Sub area code 2	
Author/s (editor/s) Helene Mörtberg	**Project manager** Martin Hagstrom	
	Approved by Nils Olsson	
	Sponsoring agency Armed Forces	
	Scientifically and technically responsible Xiaoming Hu	

Report title
Control and dynamic modeling of
an Autonomous Ground Vehicle

Abstract
The aim of the thesis is to model and simulate a dynamic model and a motion control regulator for an autonomous ground vehicle. The vehicle model used is the so-called Bicycle model and the control regulator uses the Pure Pursuit algorithm. Both are modeled in Modelica. Another vehicle control law, Vector Pursuit, is studied but not modeled.
The report describes the theory behind the vehicle model and how it is modeled in Modelica. Furthermore, it illustrates two different motion control laws which are both based on geometric principles. One of them is modeled in Modelica. The vehicle model and the control law are then connected and simulated. These simulations are performed with different paths and with different look ahead distances.

Keywords
UGV, autonomous system, vehicle control

Further bibliographic information	**Language** English
ISSN ISSN-1650-1942	**Pages** 34 p.
	Price acc. to pricelist

Utgivare FOI – Totalförsvarets forskningsinstitut Systemteknik 164 90 STOCKHOLM	**Rapportnummer, ISRN** FOI-R--1911--SE	**Klassificering** Teknisk rapport
	Forskningsområde Farkost- och rymdteknik, inkl material	
	Månad år Januari 2006	**Projektnummer** E6041
	Delområde Obemannade farkoster	
	Delområde 2	
Författare/redaktör Helene Mörtberg	**Projektledare** Martin Hagstrom	
	Godkänd av Nils Olsson	
	Uppdragsgivare/kundbeteckning FM	
	Tekniskt och/eller vetenskapligt ansvarig Xiaoming Hu	
Rapportens titel Reglering och dynamisk modellering av ett autonomt markfordon		

Sammanfattning

Syftet med denna rapport är att modellera och simulera en dynamisk modell och en styrlag för ett autonomt markfordon. Den så kallade Cykelmodellen används som bilmodell och styrlagen modelleras efter Pure Pursuit-algoritmen. Båda är modellerade i Modelica. En annan styrlag, Vector Pursuit, studeras men modelleras inte.
Rapporten beskriver teorin bakom bilmodellen och hur den modelleras i Modelica. Vidare så illustreras två olika styrlagar som båda är baserade på geometriska principer. En utav dem modelleras sedan i Modelica. Bilmodellen och styrlagen sätts sedan ihop och simuleras. Dessa simuleringar utförs med olika banor och med olika framförhållningspunkter.

Nyckelord
UGV, autonoma system, fordonstyrning

Övriga bibliografiska uppgifter	**Språk** Engelska
ISSN ISSN-1650-1942	**Antal sidor:** 34 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Contents

Acknowledgments

I would like to express my appreciation to my supervisor at the Royal Institute of Technology (KTH), Professor Xiaoming Hu, for introducing me to the Swedish Defense Research Agency (FOI) and for his guidance and support through the project.

Special thanks to my supervisor at FOI, Martin Hagström, who gave me the opportunity to work on this instructive and exciting autonomous vehicle project.

Furthermore, I would like to thank the many individuals at FOI who have contributed with their technical expertise and that have made my experience at FOI enjoyable. Specifically, many thanks to Ulrik Nilsson for his continual support and for spending countless of hours by the computer with me and to Hugo Heden for all the laughs we shared during office hours or during Fridays after work in the bar.

I also need to express my gratitude to Johan Andreasson, Division of Vehicle Dynamics, KTH, and Hans Olsson, Dynasim, for their help with Modelica.

Finally, I would like to thank my parents who laid the foundation for everything I have achieved and become today and to my friends for their everlasting love and for seeing the possible when I saw the impossible.

This work, though, is dedicated to my best friend, my inspiration and my life love. Magnus, you are the reason I strive for excellence. I would be lost without you.

1 Introduction

The work presented in this Master of Science thesis is a part of an autonomous vehicle project at the Swedish Defense Research Agency (FOI), Division of Autonomous Systems. Supervisor at FOI has been Martin Hagström and his counterpart at the Royal Institute of Technology (KTH) has been Xiaoming Hu, Professor at the Division of Optimization and Systems Theory.

The work has been performed during one semester, i.e. 20 weeks, at FOI and is the final examination before earning a Master of Science degree in Engineering Physics at KTH. The work will be presented both in writing in this report and in oral during a lecture.

1.1 Thesis Background

One of the research areas of the Division of Autonomous Systems at FOI is to look into autonomous vehicles and the possible co-operation between them. The goal is to group the vehicles and let them perform certain missions. Depending on the type of the mission, the vehicles don't have to be of the same kind; it can for example be an autonomous ground vehicle interacting with an autonomous aerial vehicle.

The vehicles would all be equipped to communicate with each other and a base station. The payload, for example digital cameras and other sensors, may differ depending on the different missions. To obtain these interchanges, the payload is planned to be easily exchanged. Furthermore, the vehicles in a group don't need the same payload. They will be able to co-operate and exchange information between them to succeed with the mission. Most vehicles will carry a GPS receiver though, since tracking and positioning is important.

Problems arising with autonomous vehicles are the ability to navigate to a predefined target and avoid obstacles on the way. Groups of autonomous vehicles have an additional difficulty of in the need to travel in a formation well suited to the circumstances. Avoiding collisions between members of the group is also important.

1.2 This thesis

For the group of autonomous vehicles to be able to move to the predefined target, each of the vehicles has to have a control system modeled for each set of dynamic and kinematic constraints. This thesis handles the motion control system of an autonomous ground vehicle (AGV).

The aim is to create a control law that makes the AGV to move along a predefined path, avoiding obstacles, on as short time as possible. Hence, the path has to be defined, the vehicle localized and made to follow the path as good as possible and the velocity calculated and adjusted to the path. Important parameters are the maximal accepted deviation from the path and the velocity of the vehicle.

The AGV is, in this thesis, modeled using a simple vehicle model, also known as the bicycle model. This is one of the most simple models one can design, but it gives valuable information about the dynamics of the vehicle and is suitable for these kinds of simulations.

The autonomous ground vehicle is supposed to be used for a military purpose. It can also perform a mission in extremely dangerous areas where risks are high or in radioactive or chemically polluted environments.

1.3 The Autonomous Ground Vehicle

The autonomous ground vehicle will be based on a radio-controlled car and consists of the car base, four wheels and its suspensions, a computer super structure, two accumulators for the car and three for the computer.

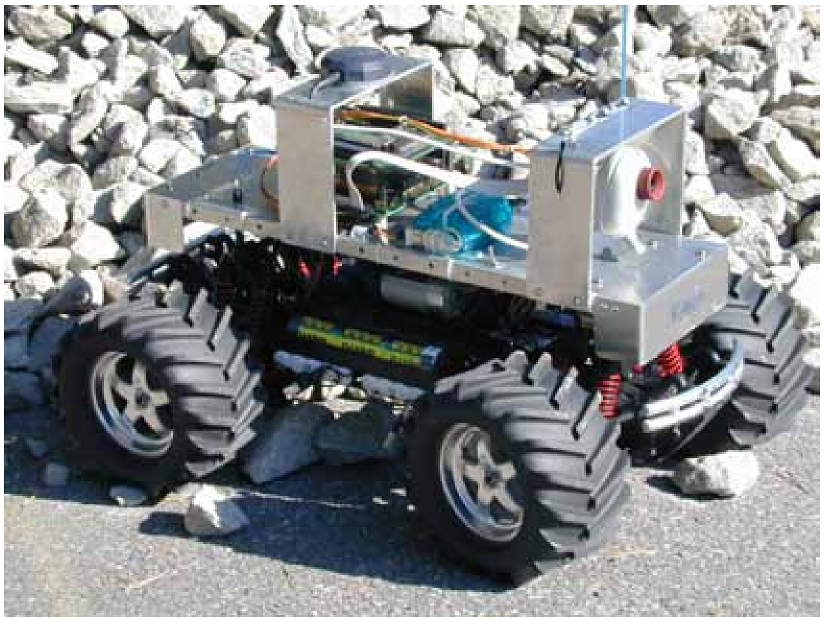


Figure 1.1: Autonomous Ground Vehicle

The measurements of relevance in this thesis are:

- Vehicle mass, m
  The vehicle mass is approximated to be the car base mass which is 1.935 kg.
- Length of the wheel base, L
  The length is measured from the front wheel hub to the rear wheel hub and is 0.310 m.
- Vehicle center of gravity, λ
  Due to the vehicle construction, the lateral symmetry of the vehicle can with a good approximation be assumed to be mirror symmetric in the xz-plane. The vehicle center of gravity is therefore assumed to be centered in the middle of the left and the right side of the car, which gives a λ-value of 0.50.
- Moment of inertia based on the z-axis, I
  The moment of inertia of the car is approximated to be the moment of inertia of a rectangular prism of height h, width w, length l and mass m. As the car is modeled to move in the plane, the moment of inertia is calculated about the z-axis

$$I = I_h = \frac{1}{12} m(w^2 + l^2) \tag{1.1}$$

and equals 0.036 $\text{kg} \cdot \text{m}^2$. The inertia can also be calculated more accurately by using the concept of free pendulum and is left for future work.

Several of these measurements have been performed at an earlier stage of the project at FOI, [1].

1.4 Model overview

The model consists of two main boxes, one handling the vehicle control law and one handling the vehicle model, which are joint together. The third box has, as its only purpose, to extract relevant parameters from the vehicle model.

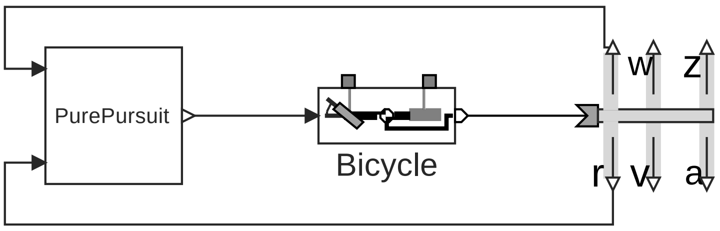


Figure 1.2: Model overview in Modelica

The vehicle control law is, in simple words, a box where the current position and the current heading of the vehicle enter as inputs and the steering angle exits as output. This box is then connected to the vehicle model box which makes the vehicle move according to the wanted steering angle. The outcome of this box is the, at this new time, current position and heading which is fed back and used as input into the first box. These iterations are proceeded until the vehicle reaches its goal point.

2 Vehicle Model

The vehicle model should handle yaw, pitch and roll motions for the simulations to be as accurate as possible. These models can be pretty advanced and are left for future work.

The model used in this thesis is an improved version of a simple vehicle model, also known as the bicycle model. The improvement concerns the linearity in the tires that the simple model employs and takes combination effects when accelerating, braking and steering into consideration.

2.1 Simple Vehicle Model

The AGV is modeled using the theory of a simple vehicle model, [2] and [3], and gives valuable information about the dynamics of the vehicle and is suitable for these kinds of simulations.

The vehicle consists of one rear wheel and one steerable front wheel and it can move in the x-y plane. This means that the usual four wheels of a vehicle are replaced by two and put in the middle of the axes in front and in the rear.

The input signal is the steering angle, δ, from the control model. The output signals are the position, r, and the orientation angle, ϕ. These are then used as inputs into the control model.

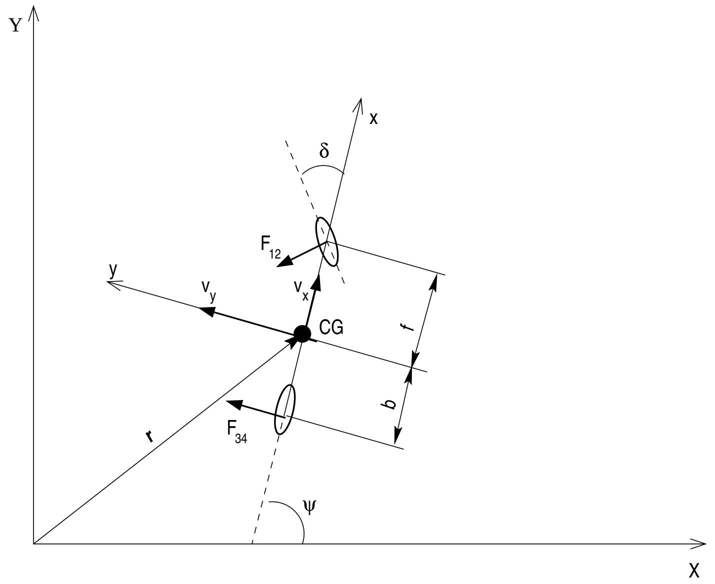


Figure 2.1: The bicycle model

There are two coordinate systems associated with the model; the one based on the vehicle (CG, x, y, z) and the one based on the world (O, X, Y, Z). The velocity vector of the center of gravity CG is

$$\dot{\bar{r}} = v_x\hat{x} + v_y\hat{y} \tag{2.1}$$

and the vehicle yaw rate is

$$\bar{\omega} = \dot{\psi}\hat{z} \tag{2.2}$$

where

$$\dot{\psi} = \frac{d\psi}{dt} \tag{2.3}$$

The velocities for the middle point of the front and rear axles can now be derived by using common derivation rules.

$$\dot{\bar{r}}_f = \dot{\bar{r}} + \bar{\omega}x(f,0,0) = v_x\hat{x} + (v_y + \dot{\psi}\cdot f)\hat{y} \tag{2.4}$$

$$\dot{\bar{r}}_b = \dot{\bar{r}} + \bar{\omega}x(-b,0,0) = v_x\hat{x} + (v_y + \dot{\psi}\cdot b)\hat{y} \tag{2.5}$$

and the tire side slip angles for the front and rear wheels can be derived as

$$\alpha_{12} = \arctan(\frac{v_y + \dot{\psi}\cdot f}{v_x}) - \delta \tag{2.6}$$

$$\alpha_{34} = \arctan(\frac{v_y - \dot{\psi}\cdot b}{v_x}) \tag{2.7}$$

The side slip angle occurs when the wheel doesn't move in the direction of its longitudinal center line, because of the effect of the reaction force $F_{y'}$, but instead moves sidewards.

The acceleration of the center of gravity is achieved by derivation of the velocity vector, stated in the vehicle coordinate system.

$$\ddot{\bar{r}} = \dot{v}_x\hat{x} + \dot{v}_y\hat{y} + \bar{\omega}x(v_x, v_y, 0) = (\dot{v}_x - \dot{\psi}v_y)\hat{x} + (\dot{v}_y + \dot{\psi}v_x)\hat{y} \tag{2.8}$$

It is assumed that the only external powers affecting the motion of the vehicle are the wheel forces. These lateral forces are, in this simple model, approximated to be linear functions of the tire side slip angles.

$$F_{12} = -C_{12}\alpha_{12} \tag{2.9}$$

$$F_{34} = -C_{34}\alpha_{34} \tag{2.10}$$

where C_{12} and C_{34} are the lateral force coefficient of the axles.

The kinematic equations can now be stated as follows:

$$\uparrow m(\dot{v}_x - \dot{\psi}v_y) = -F_{12}sin\delta \tag{2.11}$$

$$\rightarrow m(\dot{v}_y - \dot{\psi}v_x) = F_{34} + F_{12}cos\delta \tag{2.12}$$

$$J_z\ddot{\psi} = fF_{12}cos\delta - bF_{34} \tag{2.13}$$

where

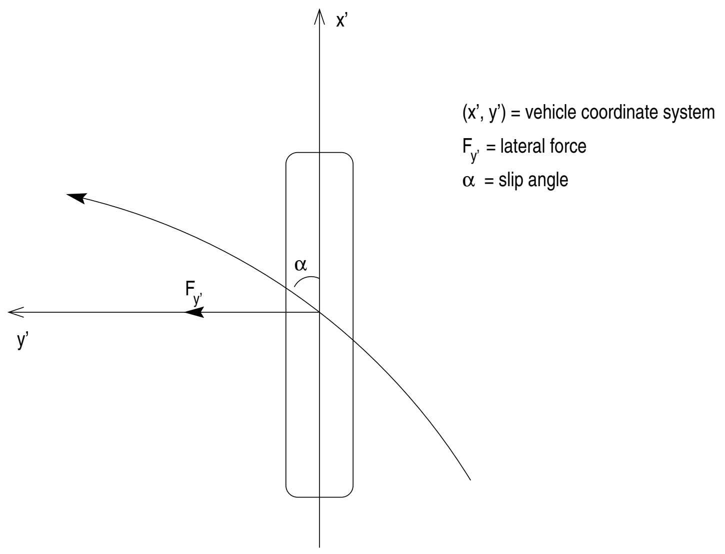


Figure 2.2: Tire side slip angle and lateral force

$\cdot$	is the time derivative
m	is the mass of the vehicle
v_x, v_y	are the velocities in the direction of x and y respectively
ψ	is the orientation angle of the vehicle
J_z	is the moment of inertia around the z axis
f, b	are the distances between the center of gravity and the front and rear wheel respectively
F_{12}, F_{34}	are the lateral forces of the front and rear wheels respectively

3 Vehicle Control Law

An autonomous vehicle needs to seek answers to a couple of questions before it can perform its mission. These questions are regulated and handled by the control and sensing and can be formulated as follows:

- Where am I?
- Where do I want to go?
- How do I get there and not get hurt?

The question 'Where am I?' handles the issue of positioning. The AGV used in this thesis has a GPS receiver which helps it to locate itself in the world coordinate system. It also has pulse sensors on two wheels which measure the speed of the vehicle. With this information the control law knows the actual position of the vehicle and the speed at that position.

The question 'Where do I want to go?' is in this thesis a pretty simple question as the target and the path, on which the vehicle should travel, is predefined. The path is modeled in different ways in order to apprehend changes in the performance of the control law.

The question of 'How do I get there and not get hurt?' is answered by a so called path tracking method. There are several ones already developed with different levels of accuracy and difficulties. The method used and further developed in this thesis is the 'Pure Pursuit' method. A more complex technique is the the 'Vector Pursuit' method which is explored, but not modeled, in this thesis. These methods are so-called geometric techniques, which means that they use a look-ahead point which is on the path at a distance L ahead of the orthogonal projection of the vehicle's position onto the path. Determining L usually becomes the hardest part when modeling the methods, due to tradeoff. The larger L becomes, the larger is the damping effect does the system have which makes it more stable and with less oscillation. On the other hand, increasing L makes the vehicle to cut corners of the path. Therefore, it is desirable to have a small look-ahead distance so that the navigation error becomes small, but a large value is typically used to achieve a stable system with little oscillation.

These methods needs also to handle the complexity of the car in terms of its capability of rolling and pitching. Therefore it is important that there is a velocity regulator included which makes sure that the vehicle doesn't travel at such a high speed that it might roll or pitch.

It is further discussed in [4].

3.1 Pure Pursuit

Pure pursuit is a method where the vehicle is supposed to follow a moving target point, a so-called carrot point, a specified distance ahead on the predefined path, [5]. The vehicle steers towards this point which continuously moves at the same speed as the vehicle. In order for the vehicle not to cut corners in a turn, it follows a circular arc that is fitted between the vehicle and the carrot point. The circle is uniquely defined by adding the condition that the heading of the vehicle should be along the tangent line of the turning circle. The carrot point, turn radius and steering angle are repeatedly updated as the vehicle moves.

This method enables a smooth steering control and improves the vehicles ability to handle curved paths with not that many short cuts. An analogy is the driver of a car who looks on a point further ahead on the road and then tends to steer smoothly toward that point and thereby drives in circular arcs on a curved road.

The Pure Pursuit Algorithm is as follows,[6] :

1. Determine the current location of the vehicle in world coordinates
2. Find the point on the path closest to the vehicle, see figure 3.2
3. Find the carrot point
4. Transform the carrot point and the vehicle location to the vehicles coordinates
5. Calculate the curvature, γ, of the circular arc
6. Determine the steering angle, δ

3.1.1 Algorithm

The algorithm is coded in Modelica for simulation purposes. The path and vehicle parameters are predefined. The current position is recorded in each iteration and is defined as the input. The path is stated as a matrix, where each row is the coordinates of a point on the path. Such a point is referred to as a way point. The difference between one point and the next one closest to it, is called a segment. The path can be seen as a series of way points with segments joining them.

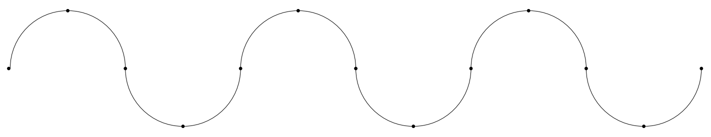


Figure 3.1: Path divided into a series of way points

The algorithm loops over each segment of the path matrix to find which the vehicle is the closest to. When the right segment is found, the point which corresponds to the orthogonal projection of the vehicle's current position onto the planned path and the distance between that point and the vehicle center point is calculated. The algorithm from [7] is used.

```
$distance(Point\ P, Segment\ P_0 : P_1)$

$v = P_1 - P_0$
$w = P - P_0$

$if(w \bullet v < 0)$ //the point is to the left of $P_0$
$return\ d(P, P_0)$

$if(v \bullet v < w \bullet v)$ //the point is to the right of $P_1$
$return\ d(P, P_1)$

$else$
$b = \frac{w \bullet v}{v \bullet v}$ //the point is on the segment
```

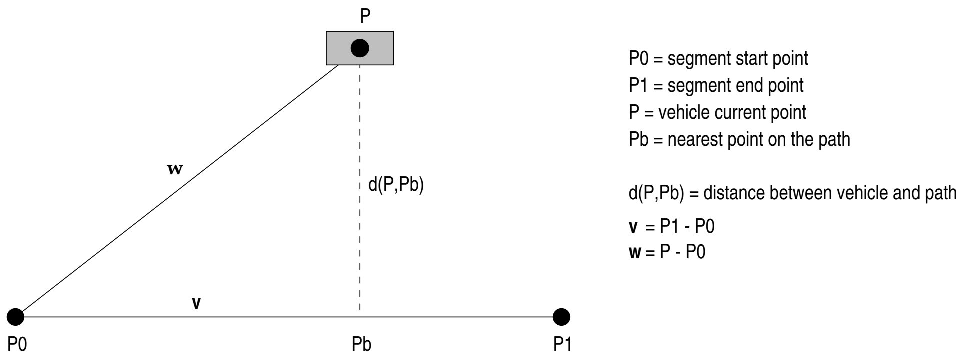

P
w
d(P,Pb)
v
P0
Pb
P1
P0 = segment start point
P1 = segment end point
P = vehicle current point
Pb = nearest point on the path
d(P,Pb) = distance between vehicle and path
v = P1 - P0
w = P - P0

Figure 3.2: Nearest point on the path

```
$P_b = P_0 + bv$
$return\ d(P, P_b)$
```

where

P	is the center point of the vehicle
P_0, P_1	are the endpoints of the current path segment
P_b	is the point on the path closest to the vehicle
$d(P_1, P_2)$	is the distance between two points

The carrot point is then calculated a set distance ahead of this point on the segment.

There are some extreme cases that have to be taken into account. If the carrot point is not on the same segment as the point on the path, the algorithm has to loop over each segment between the point on the path and the carrot point and add the distance recursively. If the point is on the last segment or on the path so that the carrot point is calculated to be outside the path, then the carrot point is set to the end point, i.e. the goal point.

When the carrot point is found, the curvature of the circular arc and the steering angle can be derived.

The vehicle coordinate system is defined so that the y-axes points in the direction of the moving vehicle, the x-axes to the right, perpendicular to the tangent line through the vehicle, and the z-axes completes the right-hand coordinate system.

From fig 3.3, three equations can be derived. The first one describes the radius, D, of the circle with VCP as the center point. It is derived from the geometry of the left triangle in the picture. The radius, D, is the locus of possible carrot points for the vehicle. The second one is similarly derived from the right triangle in the picture.

$$x_c^2 + y_c^2 = D^2 \tag{3.1}$$

$$d^2 + y_c^2 = r^2 \tag{3.2}$$

The third equation states that the radius of the arc, r, and the x coordinate of the carrot point in the vehicle coordinate system, x_c, are independent and differ by d. It is derived from the summing of line segments on the x axis in the vehicle coordinate system.

$$x_c + d = r \tag{3.3}$$

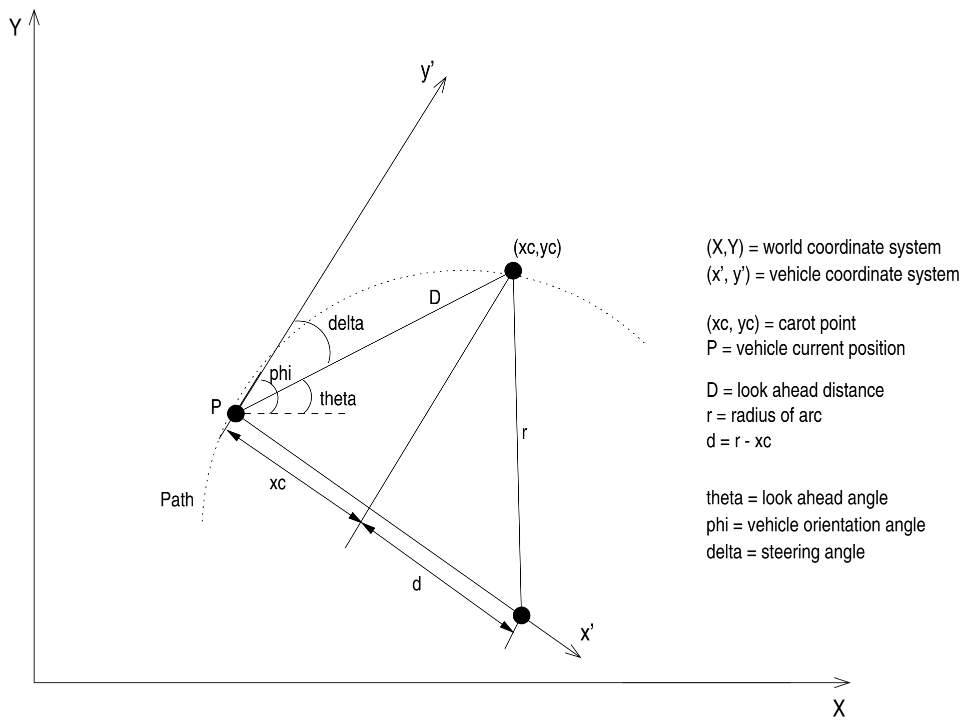


Figure 3.3: Angles and distances

The curvature, γ, and the radius, r, of the arc are now derived by combining the three equations. Notice that the the curvature is the inverse of the radius of the arc.

$$d = r - x_c \tag{3.4}$$

$$(r - x_c)^2 + y_c^2 = r^2 \tag{3.5}$$

$$r^2 - 2r x_c + x_c^2 + y_c^2 = r^2 \tag{3.6}$$

$$2r x_c = D^2 \tag{3.7}$$

$$r = \frac{D^2}{2x_c} \tag{3.8}$$

$$\gamma = \frac{1}{r} = \frac{2x_c}{D^2} \tag{3.9}$$

where

(x_c, y_c)	is the coordinates of the carrot point in the vehicle coordinate system
D	is the distance between the VCP and the carrot point
r	is the radius of the arc that joins the VCP and the carrot point
γ	is the curvature of the arc

The steering angle, δ, is now derived by

$$\delta = \theta - \phi \tag{3.10}$$

where

δ	is the steering angle
θ	is the look ahead angle
ϕ	is the orientation angle (fed as an input from the vehicle model)

The steering angle varies with surface conditions, vehicle speed, tire stiffness etc, [8]. As the vehicle is modeled by a simple vehicle model, the vehicle speed is the only condition that affects the steering angle. A more advanced dynamic model where different surface conditions and tire stiffnesses can be modeled is interesting and is left for future work.

3.2 Vector Pursuit

Vector Pursuit is a more advanced technique based on the theory of screws, [9]. The theory of screws is a very old theory which describes the instantaneous motion of a rigid body relative to a given coordinate system. The rigid body is in this case the autonomous ground vehicle. Therefore, this theory can be used to represent the motion of a vehicle from its current position and orientation to a desired position and orientation that is on a given path, i.e. the carrot point. Different from the Pure Pursuit method is that it considers not only the position of the carrot point but also the possible orientation of the vehicle at the carrot point regarding the curvature of the path there. This means that it ensures that the vehicle arrives to the carrot point with the correct orientation and steering angle.

3.2.1 Algorithm

Before describing the Vector Pursuit path tracking method it is essential to mention the basics of screw theory. A screw consists of a centerline that is defined in a given coordinate system. An instantaneous screw is the one describing the instantaneous motion of a rigid body, in this case the AGV. Two specific screws are used in developing the path tracking algorithms, translation and rotation screws. The motion about a screw with an infinite pitch, h, models pure translation of a rigid body at a velocity v along the direction $\mathbf{S}$.

The velocity of a rigid body can be quantified by

$$\omega\$ = (\omega\mathbf{S}; \omega\mathbf{S}_{0h})$$

where

$$\mathbf{S}_{0h} = \mathbf{S}_0 + h\mathbf{S} = \mathbf{r} \times \mathbf{S} + h\mathbf{S}$$

and $\mathbf{r}$ is any vector from the origin to the centerline of the screw.

The translation screw is now defined as the screw with the centerline at infinity:

$$v\$ = (0; v\mathbf{S})$$

while the rotation screw is defined as the screw with a pitch equal to zero:

$$\omega\$ = (\omega\mathbf{S}; \omega\mathbf{S}_0)$$

where

$	signifies the screw
$(\mathbf{S}, \mathbf{S}_0)$	is the centerline of the screw
h	is the pitch
v	is the velocity of the rigid body
ω	is the angular velocity of the rigid body

Another essential background information is the definition of coordinate systems. The world coordinate system is defined where the x-axis points north, the z-axis points down and the y-axis points east to form a right hand coordinate system. In addition to the world coordinate system, both a moving and a vehicle coordinate system are defined. The moving coordinate system has its origin in a point on the planned path which is a given distance in front of the orthogonal projection of the vehicle's position onto the planned path. In similarity to the pure pursuit method, the point is called the look-ahead point, the distance the look-ahead distance D and the path is modeled as a series of way points. The x-axis of the moving coordinate system is oriented in the direction of the planned path at that point, the z-axis is down and the y-axis is defined to form a right hand coordinate system. This moving coordinate system is referred to as the look-ahead coordinate system. Finally, the vehicle coordinate system is defined where the x-axis is in the forward direction of the vehicle, the z-axis is down and the y-axis forms a right hand coordinate system. The

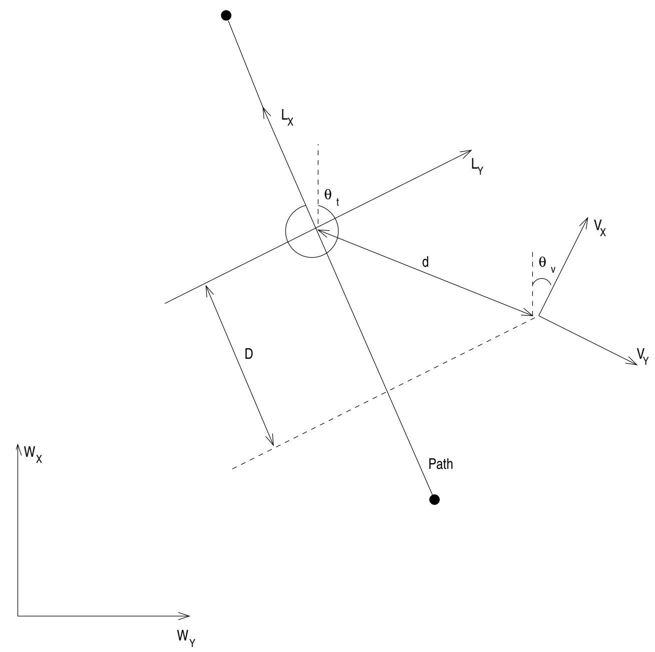


Figure 3.4: Defined coordinate systems

Vector Pursuit path tracking method can now be summarized in four steps. The first step calculates two instantaneous screws. The first screw, $\$_t$, accounts for the translation from the current vehicle position for the translation from the current vehicle position to the location of the look-ahead point while the second screw, $\$_r$, accounts for the rotation from the current vehicle orientation to the desired orientation at the look-ahead point. The desired orientation at the look-ahead point is defined as the direction tangent to the path at the look-ahead point.

$$^{W}\$_t = k_t\left(0,0,1; {}^{W}y_V + \frac{d^2}{2^{V}y_L}cos(\theta_V), -{}^{W}x_V + \frac{d^2}{2^{V}y_L}sin(\theta_V), 0\right)$$

$$^{W}\$_r = k_r\left(0,0,1; {}^{W}y_V, {}^{W}x_V, 0\right)$$

The second step uses the additive property of instantaneous screws to calculate $\$_d$, the sum of $\$_t$ and $\$_r$, which defines the desired instantaneous motion of the vehicle.

$$\begin{gathered} ^{W}\$_d = {}^{W}\$_t + {}^{W}\$_r = \\ = \left(0,0,k_t + k_r;\right. \\ \left. k_r{}^{W}y_V + k_t\left({}^{W}y_V + \tfrac{d^2}{2^{V}y_L}cos(\theta_V)\right), -k_r{}^{W}x_V + k_t\left(\tfrac{d^2}{2^{V}y_L}sin(\theta_V)\right), 0\right) \end{gathered}$$

The third step uses $^{W}\$_d$ to determine the coordinates of the point in the XY plane on the centerline of the desired screw in the world coordinate system.

$$^{W}x_{\$_d} = {}^{W}x_V - \frac{k_t}{k_t + k_r}\left(\frac{d^2}{2^{V}y_L}sin(\theta_V)\right)$$

$$^{W}y_{\$_d} = {}^{W}x_V + \frac{k_t}{k_t + k_r}\left(\frac{d^2}{2^{V}y_L}cos(\theta_V)\right)$$

The fourth and final step determines the coordinates of the point in the XY plane on the centerline of the desired instantaneous screw in the vehicle coordinate system. These coordinates help to determine the desired motion of the vehicle.

$$^{V}x_{\$_d} = {}^{W}x_V cos(\theta_V) + {}^{W}y_V sin(\theta_V) - \left({}^{W}x_{\$_d}cos(\theta_V) + {}^{W}y_{\$_d}sin(\theta_V)\right)$$

$$^{V}y_{\$_d} = {}^{W}x_V sin(\theta_V) + {}^{W}y_V cos(\theta_V) - \left(-{}^{W}x_{\$_d}sin(\theta_V) + {}^{W}y_{\$_d}cos(\theta_V)\right)$$

where

$^{V}x_L, {}^{V}y_L$	are the coordinates of the origin of the look-ahead coordinate system in the vehicle coordinate system
$^{W}x_V, {}^{W}y_V$	are the coordinates of the vehicle position in the world coordinate system
k_t, k_r	are weighting factors that are used to control how much the desired instantaneous screw is influenced by $\$_t$ and $\$_d$, respectively
d	is the distance from the origin of the vehicle coordinate system to the origin of the look-ahead coordinate system
θ_V	is the angle from the x-axis of the world coordinate system to the x-axis of the vehicle coordinate system

3.3 Look-Ahead distances

Geometric techniques, such as Pure Pursuit and Vector Pursuit, use a look-ahead point which is on the path at a distance L ahead of the orthogonal projection of the vehicle's position onto the path, [6] and [9]. It is used to determine the desired motion of the vehicle. As mentioned earlier, the look-ahead distance is an important variable when using path tracking methods.

Two main factors when choosing the look-ahead distance are the vehicle speed and the anticipated position and heading errors.

As the vehicle speed increases, the look-ahead distance typically needs to be increased, too. Having a look-ahead distance greater than zero allows the vehicle to start turning before it actually reaches a curve in the path. Starting the turn early is desirable because of the fact that a certain amount of time is required for the vehicle to execute a commanded turning rate. The faster the vehicle is going, the sooner the vehicle needs to start its turn. Ideally then, a geometric path-tracking technique would allow small look-ahead distances to accurately track the given path and not be sensitive to small changes in vehicle speed.

When determining the distance L, there is a tradeoff between the stability of the system and position and heading errors. Increasing L tends to dampen the system, leading to a stable system with less oscillation. On the other hand, increasing L also tends to cause the vehicle to cut corners of the path. Therefore, it is desirable to have a small look-ahead distance in order to accurately navigate the path, but out of necessity, a large value typically is used to achieve a stable system with little oscillation.

4 Modelica

The purpose of a modeling language is to describe the behavior of small pieces of a larger system. It should also encourage reuse of previous work and help manage the complexity of systems as they become larger. It should be possible, once a reusable set of components has been created, to work at an increasingly higher level, i.e. getting away from writing equations at the component level and working more on the assembly of a complex system. Ultimately, this leads to the ability to build systems using 'top-down' approach rather than a 'bottom-up' approach.

Modelica is such a modeling language which can be used to simulate both continuous and discrete behavior. It was created to fulfill a lack of a language that could express the behavior of models from a wide range of engineering domains without limiting those models to a particular commercial tool. In other words, Modelica is both a modeling language and a model exchange specification.

Reasons why Modelica has been world-wide accepted are based on its ease of use, visual design of models with combination of Lego-like predefined model building blocks, its ability to define model libraries with reusable components, its support for modeling and simulation of complex applications involving parts from several application domains and many more useful facilities. Much of the Modelica syntax is hidden from the end-user because, in most cases, a graphical user interface is used to build models by selecting icons for model components, using dialogue boxes for parameter entry and connecting components graphically.

Basic Modelica programming can be studied in [10] and [11].

4.1 Modelica versus Simulink

The main difference between Modelica and Simulink is that Simulink is a so-called block oriented programming language, whilst Modelica is an object oriented language.

When programming in Simulink, the first step is to sort the equations by hand and then draw them by using the Graphical User Interface. This means that the user draws the equations and not the actual system.

In Modelica, on the other hand, the modeler encapsulates the data and behavior in individual components and thereby minimizes global data. The public interface is in this case parameters, ports and data and the local variables, discrete events and equations remain private.

4.2 Vehicle Model in Modelica

The vehicle model that was used in simulation is part of the Planar Multibody library coded in Modelica, [12]. The model is a simplification of a car chassis, where the roll, pitch and bounce motions are neglected and no load transfer between the wheels can occur.

The vehicle is connected to an inertial system (1) via a planar joint (2) allowing x-y translation around z. The vehicle consists of a body (3) with mass and inertia, translations (4,5) from the center of gravity to the front and rear wheels and a revolute

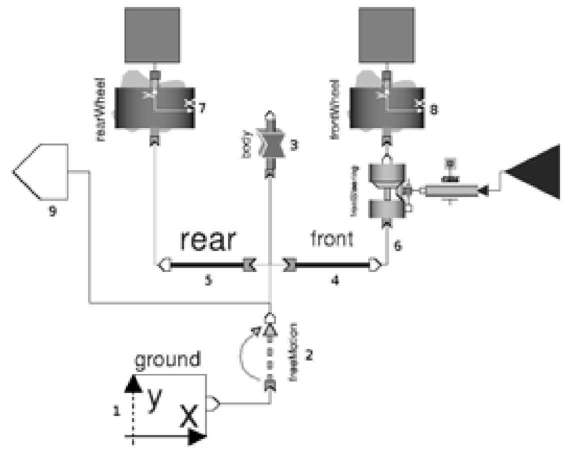


Figure 4.1: Vehicle model in Modelica

joint for the steering (6). The wheels (7,8) could be of any kind as long as they use a certain Base Wheel interface in Modelica. Additionally, a connector COG (9) makes it possible to attach further components to the model.

5 Simulation

The simulation is carried out in Modelica and with different paths so that an accurate performance of the model can be assured. The modeled paths are a straight line, a curved line and a sine-curve. Finally, the dependence of the look ahead distance is tested by simulating with different values of it.

The plots presented for each simulation are the position of the vehicle versus the carrot point, the relevant angles (δ, ϕ and θ) and the vehicle velocity. Note that the positions and the velocities are plotted as functions of time. The initial velocity of each simulation has been 5 m/s and the look ahead distance 0.5 times the vehicle length, i.e. 0.155 m. In the last simulation, different look ahead distances are used. They are 0.1 and 1 times the vehicle length.

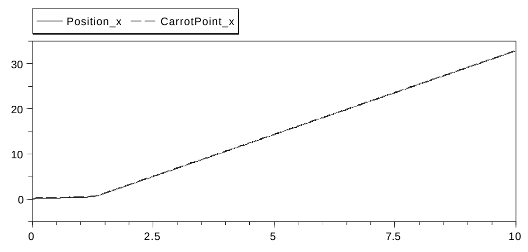
5.1 Straight path

A simple straight path was used in the beginning of the simulation to make sure that the method works accurately. The path has a constant y-value of 5, as can be seen in figure 5.1.

The vehicle moves fast up to the straight path and then stabilizes on the path, which takes about 1.5 seconds. Figure 5.1 shows that the vehicle doesn't move in the x direction in the beginning, i.e. x=0, but then has a constant increase in x. On the other hand, the second plot shows that the vehicle has a fast increase in the y direction, but when it has found the straight path it only moves in the x direction. The carrot point, which the vehicle follows, is constantly on the straight path, as it should be.

The initial velocity is 5 m/s and in the beginning it is only in the y direction, shown in figure 5.2. The small disturbance in velocity at time 1.4 seconds is due to the stabilization phase of the vehicle. It drops in velocity so that it can smoothly join the path.

From figure 5.3 it can be seen that the vehicle orientation angle, ϕ, starts at 0 radians, i.e. the y-axis of the vehicle is in the same direction as the global X-axis. When the simulation starts, ϕ increases to $\frac{\pi}{2}$ radians and the vehicle steers straight up towards the path which is above the vehicle. In order for the vehicle to make a smooth turn and slide up onto the path, the orientation angle slowly decreases to 0 radians where it stabilizes when the vehicle has found its path. The steering angle, δ, starts at $\frac{\pi}{2}$, which indicates that the carrot point is above the vehicle. Then, while the vehicle moves up to the carrot point, i.e. turns left, the steering angle decreases. At zero radians, the vehicle moves straight up towards the path. At time 1.4 seconds the steering angle has a small drop which means that the vehicle turns left and then right, before it finds the path, stabilizes and has a constant steering angle at zero radians.



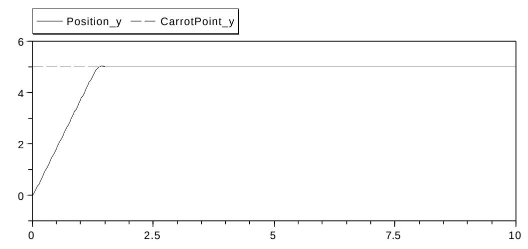


Figure 5.1: The position of the vehicle on the path, in the direction of x and y respectively

5.2 Curved path

The next step in the simulation process is to advance the path to see if the vehicle still follows it. The path used in this simulation is similar to the straight path at y = 5, except that it has a decrease to y = -5 at x = 30, an increase to y = 15 at x = 85, a decrease to y = -5 at x = 170, etc.

Similar to the first case, the vehicle finds the path after approximately 1.5 seconds, seen in figure 5.4. X-wise, it has a constant increase, while it increases and decreases in the y direction. As can be seen, the vehicle has no problem finding the path and following it.

The vehicle follows the carrot point and the path at a speed of approximately 3.5 m/s (figure 5.5). The decreasing and increasing velocity in the y direction can be explained by the fact that the vehicle moves up and down along the path. This means that when the vehicle follows the straight part of the path, the velocity is zero, when it goes down, the velocity decreases below zero and when it goes up, the velocity increases above zero. It should not be confused with the case when the vehicle turns 180° and moves in the opposite direction.

As mentioned in earlier chapters, $\delta = \theta - \phi$. The easiest way to explain figure 5.6 is to describe a decreasing steering angle with a left turn and an increasing steering angle with a right turn. When comparing these plots to the position plots, it can be seen that this is true. As in the previous case, the steering angle starts at $\delta = \frac{\pi}{2}$, which indicates that the vehicle is turned in the same direction as the global X-axis.

5.3 Sine-curved path

The next path advancement is to smooth the edges of the previous path and transform it to a sine-curve.

When the path is sine-curved, figure 5.7, and therefore oscillates a lot, it takes a while for the vehicle to respond to the changes in the path. Thus, there is always a small distinction between the vehicle and the path of the carrot point.

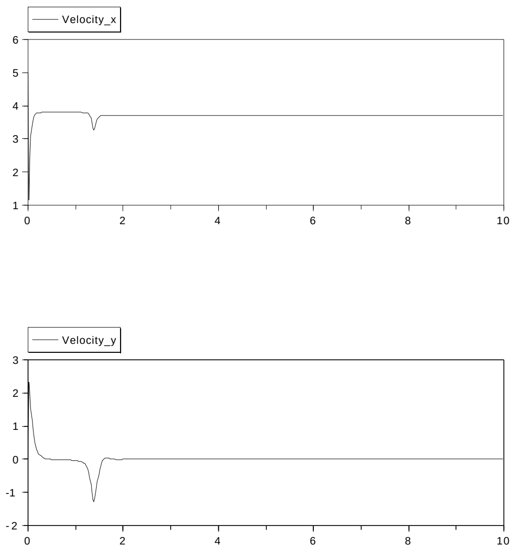


Figure 5.2: The velocity of the vehicle, in the direction of x and y respectively

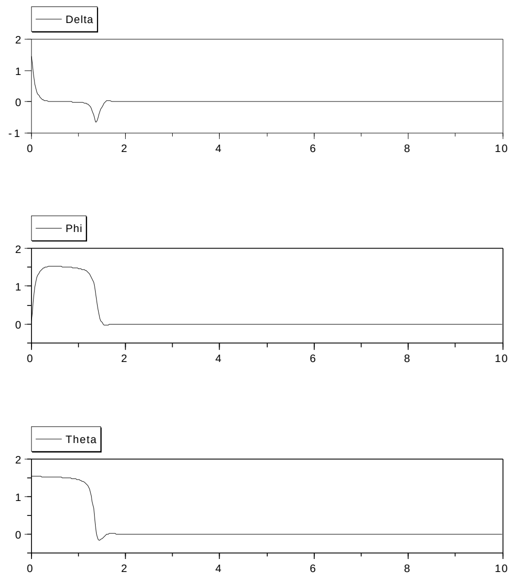
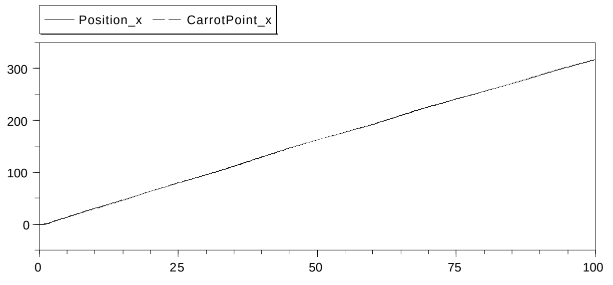


Figure 5.3: The angles δ, ϕ and θ

As can be seen in figure 5.8 the velocity in the x-direction is constantly declining, which might indicate that the position errors might depend on the velocity and the vehicle wants to adjust it by slowing down. The velocity in the y-direction increases and decreases due to the ups and downs in the position. The small oscillations that occur are due to the vehicle turning on the top and bottom of the curves.

The steering angle, δ, in figure 5.9 follows the same pattern as the velocity in the y direction. This is also due to the heavy turns that the vehicle has to perform often.



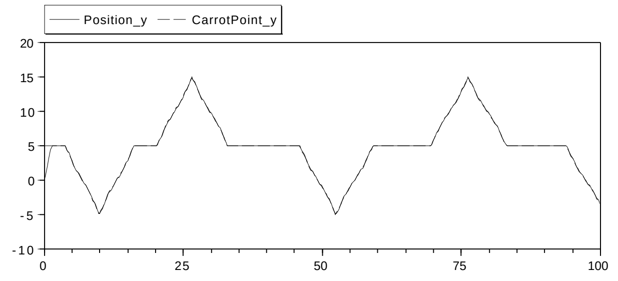


Figure 5.4: The position of the vehicle on the path, in the direction of x and y respectively

5.4 Test of different Look Ahead distances

The problem dependence of the size of the look ahead distance can be studied by altering it in the simulator. To show this, tests with two different look ahead distances have been performed; 0.1 and 1 times the vehicle length respectively. Recall that the vehicle is 0.310 m long from the front wheel hub to the rear.

Figure 5.10 shows the path the vehicle travels, with the two different look ahead distances, and the path the carrot point travels. It is clear how sensitive the vehicle is to different look ahead distances. When the vehicle has a small look ahead distance it accurately follows the path without any heading or position errors. The larger the distance gets the more willing the vehicle is to cut corners and not track the path, which can be seen in the vehicle's position in the y-direction. This is due to the fact that the vehicle always steers towards the carrot point, which in this case is far ahead of the vehicle. Thus, the vehicle doesn't take any notice in the actual path that it has to follow, only the path on which the carrot point lies.

When studying the velocities in the x-direction in figure 5.11, it can be seen that the vehicle holds a much greater velocity when the look ahead distance is larger and it doesn't have to slow down in corners to track the path. The velocities in the y-direction are different only because the tops and bottoms of the path the vehicle travels occur at different stages, which can be seen in the plot of the positions.

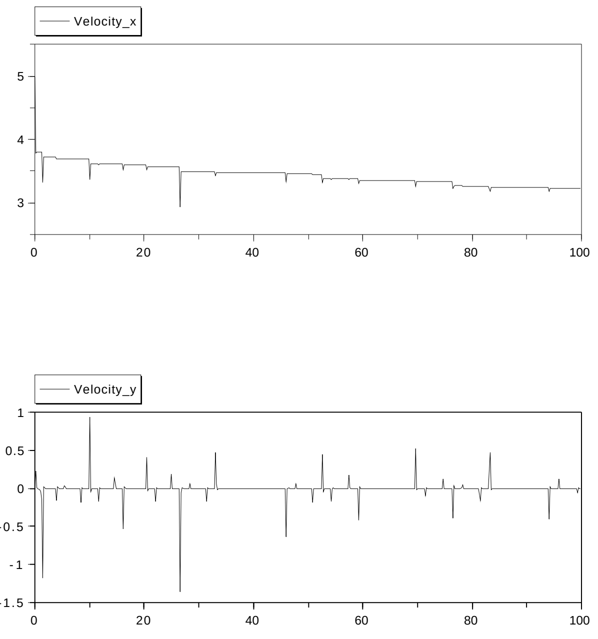


Figure 5.5: The velocity of the vehicle, in the direction of x and y respectively

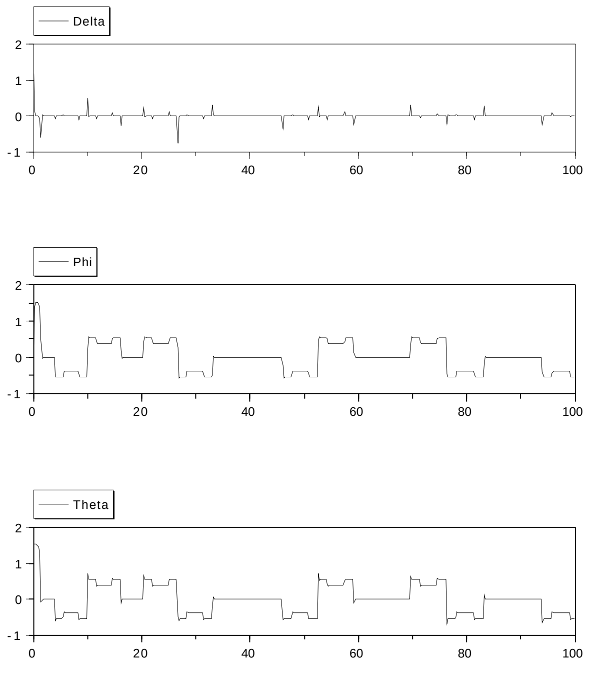


Figure 5.6: The angles δ, ϕ and θ

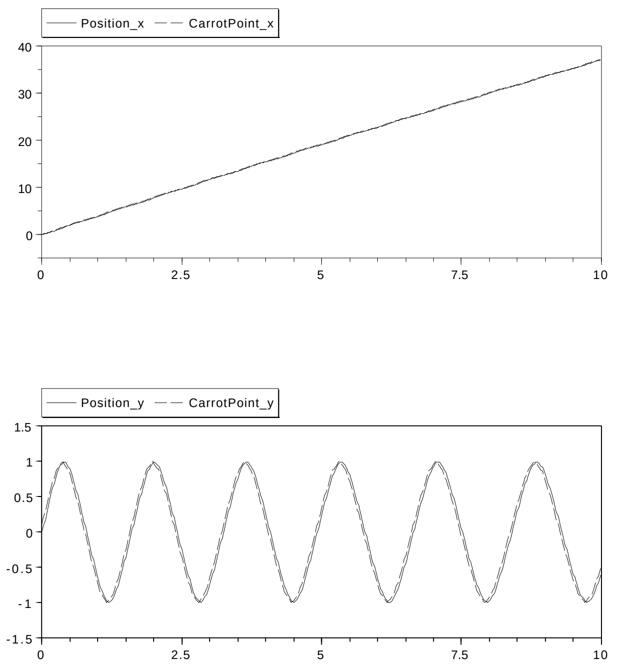


Figure 5.7: The position of the vehicle on the path, in the direction of x and y respectively

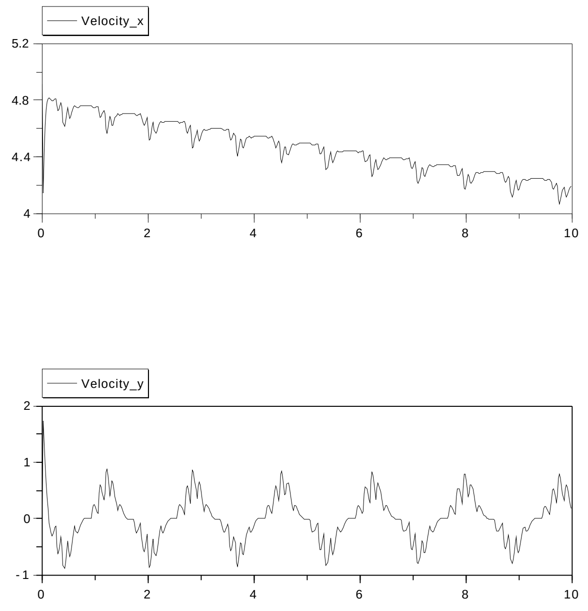


Figure 5.8: The velocity of the vehicle, in the direction of x and y respectively

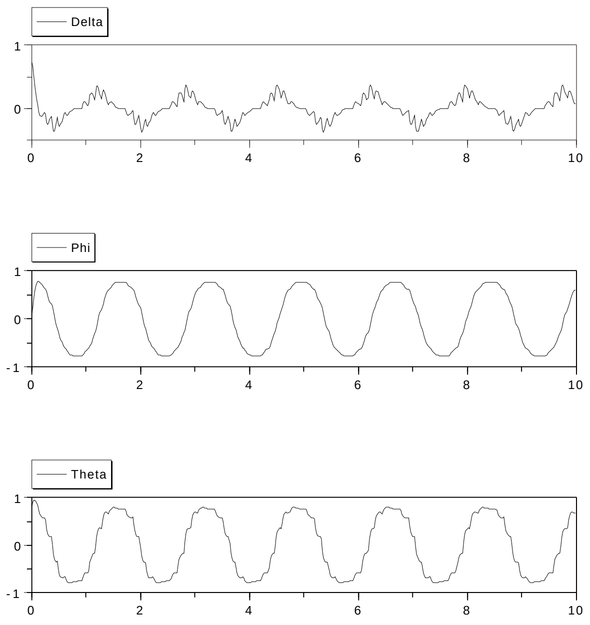


Figure 5.9: The angles δ, ϕ and θ

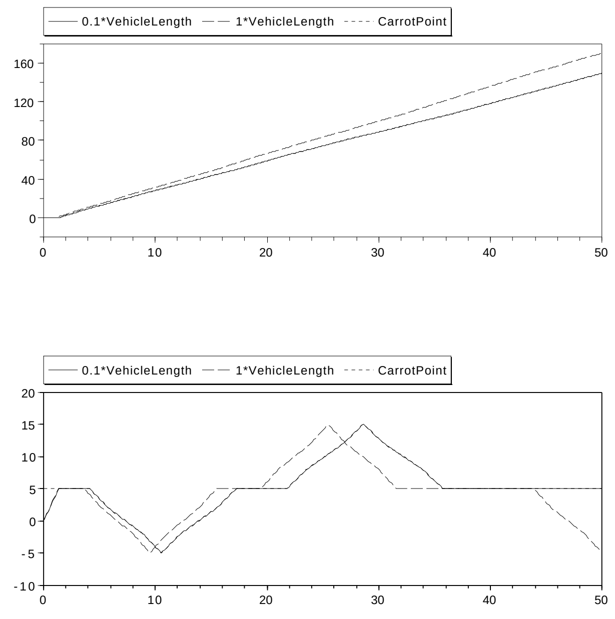
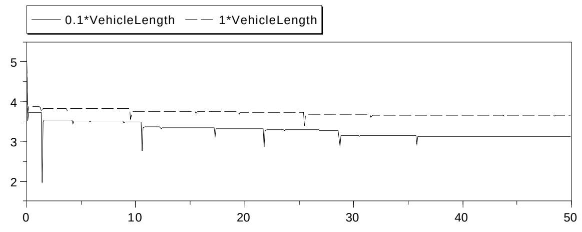


Figure 5.10: The position of the vehicle on the path, in the direction of x and y respectively



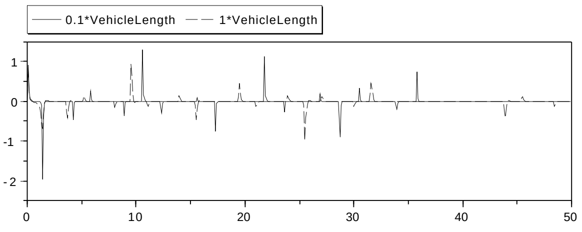


Figure 5.11: The velocity of the vehicle on the path, in the direction of x and y respectively

6 Conclusions

The intent of this thesis has been to create a control algorithm to keep a vehicle following a path. The algorithm has been coded and simulated in Modelica and will be, at a later stage, implemented on the computer controlled car.

The vehicle model used is a so-called bicycle model, where the usual four wheels of a car have been diminished to two wheels - one in the front and one in the back. It is a simple vehicle model, well suited for these kinds of simulations where the focus has been on the motion control algorithm. The model should eventually be updated to include vehicle yaw, pitch and roll motions to ensure an accurate simulation of the actual vehicle.

When modeling the motion of the vehicle, the Pure Pursuit algorithm has been used. It is a geometric method where the vehicle is supposed to follow a carrot point which lies on the predefined path. The carrot point is calculated to be a set distance ahead of the orthogonal projection of the vehicle center point onto the path. The mission has been to study how well the vehicle follows certain predefined paths and how the accuracy depends on different look ahead distances, i.e. the distance between the carrot point and the orthogonal projection point. The simulations indicate that a short look ahead distance results in less position errors but also a smaller velocity. It is therefore important to find the right balance between the position error and the velocity and the best look ahead distance is in this thesis found to be 0.5 times the vehicle length.

Another vehicle control law studied is the Vector Pursuit algorithm. As with Pure Pursuit, it follows a carrot point which lies on the predefined path, but it also takes into account the orientation of the vehicle in that carrot point. This results in fewer position errors, especially when the path is turning. This method has not been modeled though and is left for future work.

7 Future Work

- Update the vehicle model to be more dynamically advanced
  For the model to be as accurate in simulations as possible it is important for the vehicle model to include the dynamics of the AGV. Thus, important measurements are the yaw, pitch and roll motions.

- Update the control model to include the orientation of the carrot point
  This update equals the implementation of the Vector Pursuit path tracking method, see chapter 4.2 for the theory.

- Create a velocity controller
  A ordinary PID regulator can be used to control the velocity of the vehicle. The wanted velocity is used as input and the controller minimise the error in each iteration. The error is calculated as the difference between the wanted velocity and the actual velocity, measured by the pulse sensor attached to each wheel.

- Calculate the moment of inertia of the vehicle
  Methods that can be used are free pendulum in swings or in 'multi-file pendulums', pendulum on a torsion table or rotation on a table, [1].

Bibliography

[1] Daniel Svensson. Mätningar på radiostyrd bil - lägen, massor och fjädring. Technical report, Swedish Defense Research Agency, FOI, 2002.

[2] Erik Wennerström, Staffan Nordmark, and Boris Thorvald. Fordonsdynamik. Avd. Fordonsdynamik, Institutionen för Farkostteknik, KTH, 1999.

[3] Jürgen Ackermann. *Robust Control - Systems with uncertain physical parameters*. Springer-Verlag, 1993.

[4] Omead Amidi. Integrated mobile robot control. Technical Report CMU-RI-TR-90-17, The Robotics Institute, Carnegie Mellon University, May 1990.

[5] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Robotics institute, Carnegie Mellon University, January 1992.

[6] Ola Ringdahl. Path tracking and obstacle avoidance algorithms for autonomous forest machine. Master's thesis, Umeå University, 2003.

[7] SoftSurfer. About lines and distance to a line, 2002. http://geometryalgorithms.com/Archive/algorithm_0102/.

[8] Karl N. Murphy. Analysis of robotic vehicle steering and controller delay. Technical report, National Institute of Standards and Technology, 1994.

[9] Jeffery S. Wit. *Vector Pursuit path Tracking for Autonomous Ground Vehicles*. PhD thesis, University of Florida, 2000.

[10] Michael M. Tiller. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers, 2004.

[11] Peter Fritzon. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-Interscience, 2004.

[12] Johan Andreasson and Jonas Jarlmark. Modularised tyre modelling in modelica. In *Proceedings of 2nd International Modelica Conference*, www.modelica.org, 2002. Modelica Association.