Vehicle Collision Avoidance System

Eivind Hope Sørbø

Master of Science in Engineering Cybernetics
Submission date: June 2013
Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name:	Eivind Sørbø
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Kollisjonsunngåelsesystem for fartøy
Thesis title (English):	Vehicle Collision Avoidance System

Thesis Description: The purpose of the thesis is to develop and simulate a collision avoidance system for a vehicle. Both static and dynamic obstacles should be considered.

The following items should be considered:

1. Literature study on collision avoidance methods.
2. Construct a vehicle simulator in Matlab for generation of vehicle trajectories. The vehicle simulator should be used in the case studies.
3. Choose methods for collision avoidance and implement and test the methods in Matlab using the vehicle simulator. Discuss stability and convergence properties of the collision avoidance methods.
4. Discuss methods for target tracking as well as estimation of bearing and velocity using position data.
5. Identify, motivate and present good cases for comparing the different collision avoidance methods. Cases for both static and dynamic obstacles should be included.
6. Conclude your results.

Start date:	2013-01-14
Due date:	2013-06-10

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU

Abstract

In this thesis a collision avoidance system for a small vehicle has been studied. A literature study on collision avoidance methods has been performed. Two of the methods, the Null-Space-Based behaviour (NSB) control and Dubins method for collision avoidance are investigated further and implemented in a collision avoidance system. The stability of the methods has been discussed and concluded that they are both locally stable.

A vehicle model have been constructed to test the methods on both static and dynamic circle shaped obstacles. Through four case studies, it has be shown that both methods manage to avoid all obstacles and reach a target, which is consistent with the stability analysis of the methods.

In the development of the collision avoidance methods it was assumed that the position and velocity for the obstacles where known. At the end of this thesis, methods for target tracking using the position measurement from a simulated radar has been discussed. A simulation has shown that it is possible to get good estimation of the velocity and bearing for a tracked object, using the measured position data from the simulated radar.

A digital attachment of the work done in this thesis is added, for possible rerun of the simulations. Also the data of the results has been stored in the attachment.

Sammendrag

I denne avhandlingen er et antikollisjonssystem for et lite fartøy blitt studert. Et litteraturstudie om antikollisjons metoder har blitt utført. To av metodene, Null-Space-Based behaviour (NSB) kontroll og Dubins metode for å unngå kollisjoner er det forsket videre og implementert i en kollisjonshindrende simulatior. Stabiliteten av metodene har vært diskutert, og konkluderte med at de er lokalt stabile.

En fartøys modell er blitt konstruert for å teste metodene på både statiske og dynamiske hindringer. Gjennom fire case-studier og det er vist at begge metodene klarer å unngå alle hindringer og nå et mål, noe som er konsistent med stabilitets-analyse av metodene.

I utviklingen av kollisjonshindrende metodene ble det antatt at posisjon og hastigheten til hindringer var kjent. På slutten av denne avhandlingen, har metoder for mål søking ved hjelp av posisjonen måling fra en simulert radar vært diskutert. En simulering har vist at det er mulig å få god estimering av hastighet og rettning for et objekt, ved hjelp av de målte posisjonsdataene fra den simulerte radar.

En digital vedlegg av arbeidet som er gjort i denne avhandlingen er lagt til, for mulig re-simuleringen. Også dataene av resultatene er lagret i vedlegget.

Preface

This thesis is the final work of my Master degree in cybernetics at the Department of Engineering Cybernetics of the Norwegian University of Science and Technology (NTNU). In the last year I have been working with collision avoidance system for a small vehicle though a project report and this Master thesis. The work have been challenges, but fun and it has taught me how to proceed in solving a larger technical problem and present the finding in a report.

I would like to thank my supervisor Thor I. Fossen at the Department of Engineering Cybernetics for his guidance in this thesis.

Eivind sørbø
Trondheim, June 2013

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Background and motivation

The main topics of this thesis are the development of collision avoidance system for a small vehicle and compare two collision avoidance methods. A collision avoidance method is a controller that is a part of the guidance system. In case of a collision it alters the preplanned path, so the vehicle does not collide. There are many different approaches to how the collision avoidance method interact with the vehicle. In a passenger jet, the system only tells the pilot, which way to safely turn if a collision is detected. Or a more autonomous approach, the collision avoidance system takes full control of the vehicle and guides it past any obstacles. In this thesis the focus will be on the autonomous approach, where the collision avoidance system re-plan the path generated by the guidance system in case of a possible collision.

Collision avoidance is a research field, which has become more relevant in recent years as the number of autonomous vehicles has increased. For a fully autonomous vehicle it can be the difference between a successfully executed mission or a total failure. If the vehicle collides with an obstacle it may have to abort the mission or in worst case be totally destroyed. The key features of a collision avoidance method are the stability properties and computational time. If the method is globally stable the method will converge to the target for any given cases while a locally stable method will only converge to the target for some cases. The computational time is most important if the method needs to re-plane the path very often, which is the case for dynamic obstacles with unknown trajectories. Thus it is interesting to study the stability properties of these methods and consider the computational time while implementing them.

1.2 Previous work

There has been done a lot of research within collision avoidance in the recent years and some of these methods are summarised in Chapter 2. The main work in this thesis is based on [Arrichiellos, 2006] the Null-Space-Based Behavioural control (NSB), the Dubins path [Dubins, 1979] and [Tsourdos et al., 2011]. This is a continuation of the project report, which focused on static obstacles avoidance with the NSB controller and compared the path length to a optimal Dubins path, which was calculated off-line without the vehicle model.

1.3 Scope of this thesis

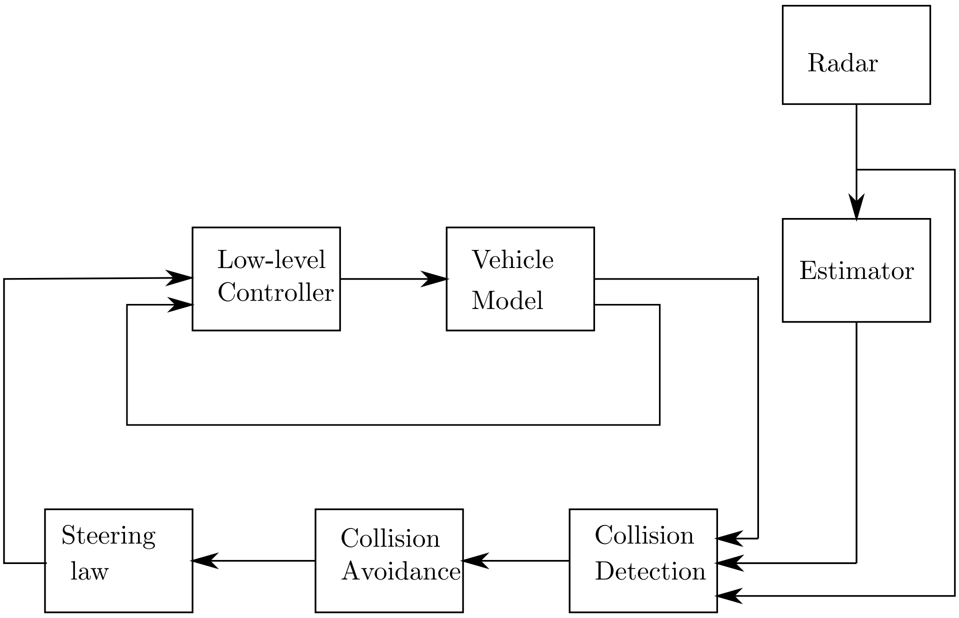


Figure 1.1: Model of the collision avoidance system

The objective of this thesis is to develop a collision avoidance system, which should handle both static and dynamic obstacles and generate good case studies to test the collision avoidance methods on a vehicle simulator. The task is solved by first performing a literature study on published collision avoidance methods and use the theory to develop two collision avoidance methods to handle both static and dynamic obstacles in a safe manner. The vehicle simulator and controller are developed in order to test the methods and create a more realistic case study for the simulations. In these case studies it will be assumed that the guidance system receive the position and velocity from the radar of all obstacles in range of the vehicle. Further it is studied methods for target tracking and estimation of the obstacles velocity and bearing using position data from a simulated radar and a Kalman filter.

Figure 1.1 shows the model of the collision avoidance system developed in this thesis and it can be seen that the guidance system is divided into three modules; collision detection, collision avoidance and the steering law. From the radar there are two arrows, one to the estimator and one directly into the collision detection block. The first one, illustrates the system when assuming the guidance system have all relevant data. The second one, illustrates when the system needs to filter and estimate the velocity and bearing.

1.4 Definitions and Nomenclature used in this thesis

Traversing obstacles

All the obstacles will be drawn in 2-D and Figure 1.2 shows the definitions on traversing the obstacles clockwise and anticlockwise.

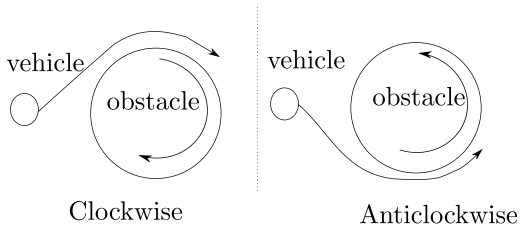


Figure 1.2: Definition of rotation direction around obstacles, where the definition of clockwise is shown to the left and anticlockwise to the right

Definition of a null space

The null space of matrix A is the set of all vectors x for, which Ax = 0.

$$\mathbf{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{0}\}$$

Reference frames

NED: The North-East-Down (NED) coordinate system $\{n\} = \{x_n,\ y_n\, z_n\}$, with origin o_n is defined relative to the Earth's reference ellipsoid. In this thesis only the x_n and y_n of the NED frame will be used.
BODY: The body-fixed reference frame $\{b\} = \{x_b,\ y_b,\ z_b\}$ with origin o_n is moving with the vehicle. In this thesis only the x_b and y_b of the BODY frame will be used.

Nomenclature

η:	$\in \mathbb{R}^{3\times 1}$ the vehicles x and y position and the heading ψ in NED frame
p_o:	$\in \mathbb{R}^{2\times 1}$ the x and y position for the center of obstacle in the NED frame
ν_v^b :	$\in \mathbb{R}^{3\times 1}$ the velocity in x and y of the vehicle and the turning rate r in the BODY frame
ν_o^b :	$\in \mathbb{R}^{2\times 1}$ the velocity of the obstacle in the BODY frame of the vehicle
ν_o^n:	$\in \mathbb{R}^{2\times 1}$ the velocity of the obstacle in the NED frame
β:	angle between the velocity vector of the of the vehicle and an obstacle
$\mathbf{R}$:	$\in SO(3) \cup \mathbb{R}^{3\times 3}$,Rotational matrix, where SO(3) is a subset of all orthogonal matrices of order 3, that is SO(3) $\cup$ O(3)
O(3)	:= defined as $\{\mathbf{R}\vert\mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{R}\mathbf{R}^{\mathbf{T}} = \mathbf{R}^{\mathbf{T}}\mathbf{R} = \mathbf{I}\}$
$\mathbf{N}_i$:	$\in \mathbb{R}^{2\times 2}$ the null space of tasks i = 1,2 and 3
$\mathbf{J}_i$:	$\in \mathbb{R}^{2\times 2}$ Jacokian matrix of task i = 1,2 and 3
$[a, b, -]$:	is vector where the third element is not relevant
$\mathbf{t}_i$:	$\in \mathbb{R}^{2\times 1}$ is the tangent waypoints calculated from the Dubins method
$\vert\vert\mathbf{x}\vert\vert$:	Euclidean norm $\vert\vert\mathbf{x}\vert\vert := \sqrt{x_1^2 + \cdots + x_n^2}$

1.5 Outline of this these

This thesis is organized as follow:

- *Chapter 2*, a literature study of different methods on collision avoidance.
- *Chapter 3*, the Null-Space-Based behavioural control (NSB) will be presented together with a simple test system for testing the algorithm.
- *Chapter 4*, the Dubins path will be presented and how to use the Dubins path in a collision avoidance approach.
- *Chapter 5*, implementation of the NSB controller and the Dubins method for collision avoidance. Then a discussion of stability of the methods and collision avoidance strategies for static and dynamic obstacles.
- *Chapter 6*, implementation of the vehicle model and the surge and heading controllers.
- *Chapter 7*, simulations from the case studies of the collision avoidance system
- *Chapter 8*, target tracking and estimation of bearing and velocity using position data
- *Chapter 9*, present the conclusion of this thesis and recommend further works.

Chapter 2

Literature study

This chapter will present the literature study performed in this thesis on collision avoidance. Two of the methods, which have been studied are not summarized here, Null-Space-based Behavioural control (NSB) and Dubins path. They will be presented separately in Chapter 3 and 4, respectively.

2.1 Collision avoidance methods

The collision avoidance methods can be roughly divided in two categories, local and global. The global approach requires knowledge about the entire environment and all obstacles must be known and is more a path planning problem than a collision avoidance problem. These methods can, as the name suggests give a global solution of the problem. The local methods only requires knowledge of a local area and make new decisions as the vehicle moves along the path towards the target.

2.1.1 Optimization problems

Optimization is a common way to solve an path planing or collision avoidance problem. An optimization problem can be expressed as

$$\min_{x} \mathbf{F}(\mathbf{x}) \tag{2.1}$$

$$subject\ to:$$

$$g_i(x) = 0, i = 1, ...n \tag{2.2}$$
$$h_j(x) \leq 0, j = 1, ...n \tag{2.3}$$

Where $\mathbf{F}(x)$ is the object function, which is minimized and $g_i(x)$ and $h_j(x)$ are the constraints. The object function can be optimized with respect to path length, fuel consumption, weather condition, time or speed, depending on the application of the vehicle. There are several methods for solving the optimization problem depending on the problem. Some of the methods are integer linear programming, SQR and QP. For more detail on these methods, see [Nocedal and Wright, 2006]. One challenge when solving a optimization problem is that it requires a lot of computation time. This is normally not a problem for off-line path planning before the mission have started. However in a collision avoidance approach with unknown obstacles, the use of optimization can be very challenging. Each time the vehicle discover a new obstacle it needs to redo the optimization problem, which takes time. The optimization approach can be used in both global and local methods and several methods use optimization to solve smaller sub-problems to find a collision free path.

2.1.2 Dynamic Window

The Dynamic Window approach was developed by Dieter Fox, Wolfram Burgard, and Sebastian Thrun and published in 1997,[Foxy et al., 1997]. The method was designed for mobile robot that operated with high speed. This was done by incorporate the dynamic of the robot as constraints in the collision avoidance system and solving the problem as a optimization problem.
The Dynamic Window approach suggested in [Foxy et al., 1997] is divided into two steps:

- Create a velocity search space with the dynamic of the vehicle.
- Solving a optimization problem with respect to heading, speed and distance between the vehicle and the obstacle using the search space as constraints.

Search space

The search space suggested in [Foxy et al., 1997] consists only of velocity vectors that are safe and reachable for the vehicle within the next iteration. If a velocity vector collided with an obstacle it is not considered. The search space suggested in [Foxy et al., 1997]

$$\mathbf{V}_a = \{(v, \omega) | v \leq \sqrt{2dist(v, \omega)\dot{v}_b} \wedge v \leq \sqrt{2dist(v, \omega)\dot{\omega}_b}\} \tag{2.4}$$

$$\mathbf{V}_d = \{(v, \omega) | v \in [v_a - \dot{v}t, v_a + \dot{v}t] \wedge \omega \in [\omega_a - \dot{\omega}t, \omega_a + \dot{\omega}t]\} \tag{2.5}$$

$$\mathbf{V}_s = v \in v_{possible} \wedge \omega \in \omega_{possible} \tag{2.6}$$

Where $\mathbf{V}_a$ are all admissible velocities, which allow the vehicle to make a full stop without colliding with obstacles. $dist(v, \omega)$ is the shortest distance from the vehicle to the obstacle.
$\mathbf{V}_d$ are all velocities that the object can reach at next time interval, which are limited by the dynamic of the vehicle. $\mathbf{V}_s$ are the space of possible velocities, then the resulting search space can be defined as

$$\mathbf{V}_r := \mathbf{V}_s \cap \mathbf{V}_a \cap \mathbf{V}_d \tag{2.7}$$

Maximizing the objective function

The object function suggested in [Foxy et al., 1997] is

$$\mathbf{G}(\mathbf{v}, \omega) = \sigma(\alpha\mathbf{H}(\mathbf{v}, \omega) + \beta\mathbf{D}(\mathbf{v}, \omega) + \gamma\mathbf{V}(\mathbf{v}, \omega) \tag{2.8}$$

which is maximized over $\mathbf{V}_r$. Where σ is a function that smoothness the sum of the three components, which results in a more side-clearance from the obstacles. All the three components , $\mathbf{H}$, $\mathbf{D}$ and $\mathbf{V}$ are normalized from [0, 1] with α, β and γ. $\mathbf{H}(\mathbf{v}, \omega)$ are the measured heading, which can be expressed as

$$heading(v, \omega) = 180 - \theta \tag{2.9}$$

Where θ are the angle of the target point relative to the objects heading. $heading(v, \omega)$ should be max if the vehicle is moving directly towards the target. $\mathbf{D}(\mathbf{v}, \omega)$ is the distance to the nearest obstacle on the path. If there is no obstacle, this values should be a large constant. $\mathbf{V}(\mathbf{v}, \omega)$ is the velocity of the vehicle and supports fast movements. When maximizing $\mathbf{G}$, it will result in the best heading towards the target with the highest speed and minimum clearings to the obstacle in theory.

Further works within the Dynamic Window approach

Since the Dynamic Window approach is susceptible to local minimum, most of the subsequent work within the Dynamic Window approach has been on developing it into a global method, without any local minimums.

In [Brock and Oussama.Khatib, 1999] they study the use of the Dynamic Window approach for holonomic robots to overcome the problem of local minimum. The advantages of holonomic robots are the instantaneous acceleration in all direction, which increase the manoeuvrability. In this paper they claim that the use of a holonomic dynamic widow approach would result in a global Dynamic Window approach. However as stated in [Ögren and Leonard, 2005] they never formally showed that the method was globally stable. In facts [Ögren and Leonard, 2005] constructed an exampled where the robots entered a limit-cycle and never reached the target, disproving that the holonomic Dynamic Window approach develop in [Brock and Oussama.Khatib, 1999] was global stable as stated. After disproving that the holonomic dynamic widow approach they develop an algorithm scheme to the Dynamic Window and proved convergence by Lyapunov.

2.1.3 Astar ($A^\star$)

In 1968 Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research first described the A star algorithm. It is an extension of the Dijkstra algorithm with a heuristics function, which increases the performers of the algorithm. A star is a discrete path finding algorithm, which can be used in collision avoidance to find a collision free path from a initial node to a goal node. The algorithm uses a distance-plus cost heuristic function $f(x)$

$$f(x) = h(x) + g(x) \tag{2.10}$$

to determine the order in, which it searches the nodes in the tree. Where the x is the vertex, which gives the lowest $f(x)$ values. $g(x)$ represents the cost function, which is the cost of moving from a start node to any node x, and $h(x)$ represent the heuristic estimated cost from node x to the goal. The heuristics function keeps a sorted priority queue of alternate path segment along the way and makes the algorithm convert faster to the target then a best-first search that need to explore the whole graph. For further reading on A star, see [Liu and Narayanan, 2011] and [Loong et al., 2011]

The problems with A star is that it requires a lot of computing time and if the problem becomes large with many nodes the method becomes very computationally intensive. Also it is discrete, which means that it finds an optimal path in a discrete grid, and this may differ from a optimal path in a continuous grid.

2.1.4 Potential Field

Khatib in 1985 [Khatib, 1985] suggested an idea of an imaginary force that acted on a vehicle, which could be used for off-line path planning. In the paper it was suggested that the obstacles should send out a repulsive force and the target should send out an attractive force acting on the vehicle. The sum of these forces should then determined the Potential Field, which could be used as a reference path. The Potential Field method became very popular and is a commonly used method for path planning and obstacle avoidance, because of it's mathematical elegance and simplicity. In 1986 Krog and Thorpe [Krogh and Thorpe, 1986] suggested a generalized Potential Field, which combined a global and local path planning.

One of the most recent published Potential Field method is [Yongjie and Yan, 2009], which suggested an artificial Potential Field where the repulsive force from the obstacle could be expressed as

$$\mathbf{U}_{rep} = \begin{cases} \frac{1}{2}k_r(\frac{1}{d(\mathbf{x},\mathbf{x}_{obs})} - \frac{1}{\rho})^2, & \text{if } d(\mathbf{x},\mathbf{x}_{obs}) \leq \rho_0 \\ 0 & \text{if } d(\mathbf{x},\mathbf{x}_{obs}) > \rho_0 \end{cases} \tag{2.11}$$

$$\mathbf{F}_{rep} = \begin{cases} k_r(\frac{1}{d(\mathbf{x},\mathbf{x}_{obs})} - \frac{1}{\rho})^2 \frac{1}{d(\mathbf{x},\mathbf{x}_{obs})^2} \frac{\partial d(\mathbf{x},\mathbf{x}_{obs})}{\partial \mathbf{x}}, & \text{if } d(\mathbf{x},\mathbf{x}_{obs}) \leq \rho_0 \\ 0 & \text{if } d(\mathbf{x},\mathbf{x}_{obs}) > \rho_0 \end{cases} \tag{2.12}$$

Where ρ represents the limit distance of the Potential Field influence and $d(\mathbf{x}, \mathbf{x}_{obs})$ is the shortest distance to the obstacle, and $\frac{\partial d(\mathbf{x},\mathbf{x}_{obs})}{\partial \mathbf{x}}$ is the the partial derivatives of $d(\mathbf{x}, \mathbf{x}_{obs})$. For the attractive force they suggested:

$$\mathbf{U}_{att}(\mathbf{x}) = \frac{1}{2}k_a|\mathbf{x} - \mathbf{x}_d|^2 \tag{2.13}$$

$$\mathbf{F}_{att} = -\nabla\mathbf{U}_{att} = -k_a|\mathbf{x} - \mathbf{x}_\mathbf{d}|^2 \tag{2.14}$$

where $k_a > 0$ is a constant and $\mathbf{x}$ the positions of the vehicle and $\mathbf{x}_d$ the target. The sum of $\mathbf{F}_{rep}$ and $\mathbf{F}_{att}$ should give the total Potential Field.

Other varieties of the Potential Field are:

- Vector field histogram, which crates a two dimensional Cartesian grid, were each cell holds a certainty value $c_{i,j}$. The certainty value represents if there is an obstacle in the cell, it should be large if there is an obstacle and low otherwise. Simultaneously, the Potential Field concept is applied to the Cartesian grid, see [Borenstein and Koren, 1989] for more detail.

- Harmonic Potential Field Path, use a harmonic function that satisfies the Laplace equation, such as the velocity potential function over a North, East plane (N,E)

$$\bigtriangledown\phi = \frac{\partial^2\phi}{\partial E^2} + \frac{\partial^2\phi}{\partial N^2} = 0 \tag{2.15}$$

Other harmonic function could also be used. Such as the steam function, which could represent the contours of a streamlines and represent the trajectories particles following a velocity vector field, See [Daily and Bevly, 2008]

- Personalized Potential Field, which includes a risk feeling variable that determines the contours of the Potential Field, see [NoriyasuNoto et al., 2011].

The Potential Field method have some problems that the reader should be aware of, such as:

1. Trap situations due to local minima, for instance if there is a u-shaped obstacle.
2. No passage between closely spaced obstacles. Since all obstacle sends out a repulsive force, which force the vehicle to go around all obstacles, even if the vehicle could go through. For the same reason, if there is an obstacle close to the target, the vehicle would not reach the target.
3. Oscillations in the presence of obstacles and narrow passages, due to fluctuation in the Potential Field.

Several papers on how to modify the Potential Field method and avoid some of these problems have been published. In [Yongjie and Yan, 2009] they suggested a method for avoiding trap situations, which is perhaps the most serious situation.

2.1.5 Limit cycle

Yongwoo Lee and Youdan Kim presented a way of solving the obstacle avoidance problem using a limit cycle around the obstacle, [Lee and Kim, 2011]. The limit cycle is generated around the obstacle and the corresponding vector field gives the steering command around the obstacle. The benefits with this approach are that it's very simple and intuitive, which resemble a human way of dealing with collision avoidance problem. The calculation load for calculating a limit cycle is not very high, which make this method very suitable if processing capacity is a problem. [Lee and Kim, 2011] suggested constructing a limit cycle as a two dimensional vector field with radius r. Let x and y be the position around the the circle,

$$\dot{x} = sign(u)x + \mu y(r^2 - x^2 - y^2) \tag{2.16}$$

$$\dot{y} = -sign(u)y + \mu x(r^2 - x^2 - y^2) \tag{2.17}$$

$$\psi_d = \arctan(\frac{\dot{y}}{\dot{x}}) \tag{2.18}$$

where u is the direction of the limit cycle and ψ_d is the desired steering angle around the obstacle on the limit cycle. This method can be combined with an other steering law and when the vehicle discover an obstacle it analysis if the obstacle is a threat or not. If it is a threat than it switching from the other steering law to the limit cycle.

2.1.6 Collision Cone Approach

In the collision cone approach, a collision cone is defined for each obstacle and if the obstacles relative velocity with respect to the vehicle velocity is within a collision cone, it is considered to be critical. If the vehicle avoid the collision cone to the obstacles, it will also avoid a collisions. In [Watanabe et al., 2007] they present a collision cone approach for static obstacle in a 3-Dimension environment. In this paper they calculated two vectors from the vehicle position to the safety circle around the obstacle. And then testing if the velocity vector for the vehicles would collide with the obstacle circle before the vehicle has reaching the target. If a collision is detected, a minimum effort guidance law was applied for waypoint following with obstacle avoidance. This method only considers changing the heading to the vehicle and not the speed. [Chakravarthy and Ghose, 1998] presented a collision cone approach for dynamic obstacles, but only for a 2-dimensional problems. They first consider the collision as a two pointed objects with the kinematic equation

$$V_r = V_b \cos(\beta - \theta) - V_a \cos(\alpha - \theta) \tag{2.19}$$

$$V_\theta = V_b \sin(\beta - \theta) - V_a \sin(\alpha - \theta) \tag{2.20}$$

where V_r and V_θ are the relative velocity's components with respect to the vehicle. V_a , α and V_b , β are the velocity and heading components to the vehicle and the obstacle respectively. θ is the angle from the xy-plane to the vector between the vehicle and the obstacle. Then they calculated a collision cone

$$\alpha_{collisionMin} \leq \alpha < \alpha_{collisionMax} \tag{2.21}$$

For collision avoidance they suggested using the collision cone (2.21) to make a collision cone with respect to the velocity such that the vehicle could regulate the speed such that the vehicle with its current heading would be outside of the collision cone.

The collision cone suggested in [Chakravarthy and Ghose, 1998] involved a lot of if else statements divided into different cases depending on the angle and speed of the vehicle and the obstacle. In total there where was 14 if statements divided in four cases only to find the collision cone in (2.21) and 38 if statements divided in seven cases to find the collision cone with respect to the velocity.

2.2 Summary

In this chapter it has been studied optimization, Dynamic Window, A star, Potential Field, limit cycle and collision cone approaches for collision avoidance. The optimization method presented first in Section 2.1.1 is normally not considered a collision avoidance method on its own, but rather a tool that some of the other methods use to solve the collision avoidance problem, such as the Potential Field and the Dynamic Window methods. The main difference between this two methods is the way they set up and solve the optimization problem. The weakness of both of these methods is that they are subject to local minimization, which can make the methods fail. Also depending on how the methods set up the optimization problem, the problem can be very computationally intensive. This is a problem especially considering dynamic obstacles or an environment where there are several unknown obstacles, which causes the methods to be resolved very often.

The collision cone approach from [Chakravarthy and Ghose, 1998] is the only method studied that uses speed regulation instead of heading control in order to avoid a collision. Also in contrast to the other methods, the collision cone can be solved using only kinematic equation and not optimization. However as seen in the articles [Watanabe et al., 2007] and [Chakravarthy and Ghose, 1998] it results in a lot of if else statements, which make this method very complex and error detection can be a problem. The collision cone from [Watanabe et al., 2007] shows, it is possible to combine the collision cone as a detection algorithm and use optimization to find a collision free heading. The A star is the only one of these methods, which is discrete and also gives a optimal solution, but in a discrete grid. The limit circles are very similar to the vector field, but instead of making a vector field of the entire map it only uses the limit circle to make a vector field around the obstacle.

Chapter 3

Null-space-based method for collision avoidance

3.1 Null-Space-Based behavioural Control

The basic idea behind Null-Space-Based behavioural (NSB) control originated according to [Cellini et al., 2007] in robotics research, with the control of redundant manipulators as objective. The null space based method subdivide all the different tasks and solve them as if they worked alone. Then combining the output from these tasks to obtain a motion command. In a general approach with n tasks, the task manager could look like this:

- Task 1: First priority task, with the highest priority
- Task 2: Second priority task, which gives a velocity contributions in the null space of task 1
- Task n: Nth priority task, which gives a velocity contributions in the null space of all the higher priority tasks

Each of the tasks generate a velocity vector defined as

$$\boldsymbol{\nu}_i := \{\dot{x}_i^n,\ \dot{y}_i^n\} \in \mathbb{R}^{2\times 1}$$

where i is the number of the task. Before adding the velocity vectors together, the lower-priority velocity is projected into the null-space of the higher priority tasks. One should note that if there are too many tasks, the lower-priority tasks might not run, since the lower-priority tasks is projected into the null space of all the higher tasks.

3.1.1 Mathematical modelling

The mathematical modelling of the NSB, which have been used in this thesis have been found in [Arrichiellos, 2006]. A short description is given here. Let $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ be the position and velocity vectors for the vehicle. $\boldsymbol{\sigma}$ is the task variable, which is the system should minimize.

$$\boldsymbol{\eta} = \begin{bmatrix} x, & y, & - \end{bmatrix}^T \tag{3.1}$$

$$\boldsymbol{\nu} = \begin{bmatrix} \dot{x}, & \dot{y}, & - \end{bmatrix}^T \tag{3.2}$$

$$\boldsymbol{\sigma} = f(\boldsymbol{\eta}) \tag{3.3}$$

$$\dot{\boldsymbol{\sigma}} = \frac{\partial f(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} \boldsymbol{\nu} := \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{3.4}$$

where $\mathbf{J}(\boldsymbol{\nu})$ is the configuration-dependent task Jacobian matrix. Solving (3.4) with respect to the minimum norm velocity, using least squares

$$\boldsymbol{\nu}_d = \mathbf{J}^\dagger \dot{\boldsymbol{\sigma}}_d \tag{3.5}$$

where $\mathbf{J}^\dagger$ is the pseudo inverse, for all $\mathbf{J} \neq 0$

$$\mathbf{J}^\dagger := \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1}$$

[Arrichiellos, 2006] suggested the following close loop control for the desired velocity

$$\boldsymbol{\nu}_i = \mathbf{J}_i^\dagger (\dot{\boldsymbol{\sigma}}_{i,d} + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}_{\boldsymbol{i}}) \tag{3.6}$$

Where $\tilde{\boldsymbol{\sigma}}_i = \boldsymbol{\sigma}_{i,d} - \boldsymbol{\sigma}_i$ and $\boldsymbol{\Lambda} > 0$ is the gain matrix. Projecting the lower priority task into the higher priority task gives the desired velocity $\boldsymbol{\nu}_d$ and for a n tasks system the desired velocity becomes

$$\boldsymbol{\nu}_d = \boldsymbol{\nu}_1 + \sum_{i=2}^{n} \mathbf{N}_i(\mathbf{J})\boldsymbol{\nu}_i \tag{3.7}$$

where $\mathbf{N}_i(\mathbf{J}_i)$ is the null-space projection, which can be expressed as

$$\mathbf{N}_i(\mathbf{J}) = \prod_{j=1}^{i-1} (\mathbf{I} - \mathbf{J}_j^\dagger \mathbf{J}_j) \tag{3.8}$$

For $n = 3$ task system the minimum norm velocity becomes

$$\boldsymbol{\nu}_d = \boldsymbol{\nu}_1 + (\mathbf{I} - \mathbf{J}_1^\dagger \mathbf{J}_1)(\boldsymbol{\nu}_2 + (\mathbf{I} - \mathbf{J}_2^\dagger \mathbf{J}_2)\boldsymbol{\nu}_3) \tag{3.9}$$

3.2 Null-Space-based Behavioral control for collision avoidance

When considering Null-Space-based Behavioral (NSB) control for collision avoidance it is important that the controller is able to reach a target while avoiding any obstacles. Also it should have the ability to traverse the obstacles both clock and anticlockwise in order to generate a safer path for the vehicle. This is solved by a three task controller and since obstacle avoidance is most critical, thus it is set to be the primary task. This section will present the mathematical theory for the three tasks of the NSB controller

- Task 1: Obstacle avoidance
- Task 2.1: Traverse obstacle
- Task 2.2: Go to target

The implementation of the controller is described in Chapter 5. It can be seen that tasks *Traverse obstacle* and *go to target* are named task 2.1 and 2.2. This is because they are mathematically identical and they both goes to a point, but considering that they have two different behaviours and is never active at the same time they have been divided up. This will be descried more in chapter 5 while presenting the implementation of the controller.

3.2.1 Task manager

Figure 3.1 illustrates the task manager with three tasks and the supervisor. The supervisor decide, which tasks that should be active or not and it can be seen that all tasks generate one velocity vector each and the lower priority tasks are projected into the null space of task one. Normally task 2.2 should be projected into the null space of task 2.1 also, but since task 2.1 and 2.2 are never active at the same time it is not necessary. Also it should be noted that if task 1 is not active, task 2.1 or 2.2 do not need to be projected into the null space of task one.

Task 1: Obstacle avoidance

Obstacle avoidance is the primary task of the system and since the obstacle have been defined as circle, the calculation of $\boldsymbol{\nu}_1$ becomes straight forward. Let the task variable $\boldsymbol{\sigma}_1$ be the length from the vehicle to the obstacle center and $\sigma_{1,d}$ be the

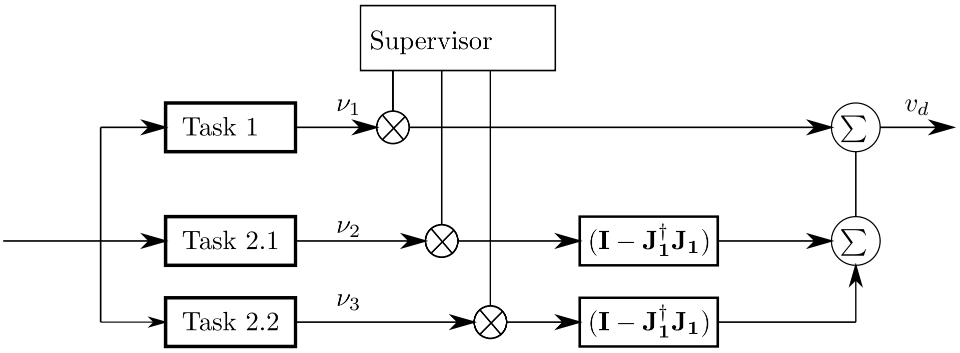


Figure 3.1: A task manager with three tasks and the supervisor

safety distance d, which the vehicle should stay outside of.

$$\boldsymbol{\sigma}_1 = \|\boldsymbol{\eta} - \boldsymbol{p_o}\| \tag{3.10}$$

$$\boldsymbol{\sigma}_{1,d} = d \tag{3.11}$$

$$\tag{3.12}$$

with the corresponding Jacobi matrix $\mathbf{J}_1 \in \mathbb{R}^{2\times 1}$ according to (3.4)

$$\begin{aligned}\mathbf{J_1} &:= \frac{\partial \boldsymbol{\sigma_1}}{\partial \boldsymbol{\eta}} \\ \mathbf{J_1} &= \frac{\boldsymbol{\eta} - \boldsymbol{p}_o}{\|\boldsymbol{\eta} - \boldsymbol{p_o}\|} = \hat{\mathbf{r}}^T\end{aligned} \tag{3.13}$$

where $\boldsymbol{p}_o$ is the position of the center of the obstacle and $\hat{\boldsymbol{r}}$ is the unit vector aligned with the obstacle-to-vehicle direction. $\hat{\boldsymbol{r}}^T$ exist if and only if

$$\boldsymbol{\eta} \neq \boldsymbol{p}_o \tag{3.14}$$

since this would cause a singularity in $\mathbf{J}_1$ Then according to (3.6) $\boldsymbol{\nu}_1$ becomes

$$\boldsymbol{\nu}_1 = \mathbf{J}_1^\dagger \lambda_1 (d - \|\boldsymbol{\eta} - \boldsymbol{p_o}\|) \tag{3.15}$$

with the null space $\mathbf{N}(\mathbf{J_1}) \in \mathbb{R}^{2\times 2}$

$$\mathbf{N}(\mathbf{J_1}) = \mathbf{I} - \mathbf{J_1^\dagger J_1} = \mathbf{I} - \hat{\mathbf{r}}\hat{\mathbf{r}}^\mathbf{T} \tag{3.16}$$

Task 2.1: Traverse obstacle

The second priority task is introduced such that the controller can guide the vehicle both clock and anticlockwise around obstacles. The task variables $\boldsymbol{\sigma}_2$ and $\boldsymbol{\sigma}_{2,d}$

becomes

$$\boldsymbol{\sigma}_2 = \boldsymbol{\eta} \tag{3.17}$$
$$\boldsymbol{\sigma}_{2,d} = \mathbf{t}_o \tag{3.18}$$
$$\tag{3.19}$$

where $\mathbf{t}_o$ is a point on the obstacle, which is calculated form (4.7) or (4.9) depending on the direction the vehicle will take around the obstacle.. And since it is only a straight line the Jacobi matrix becomes

$$\mathbf{J_2} = \mathbf{I} \tag{3.20}$$

where $\mathbf{I} \in \mathbb{R}^{2\times2}$. Then according to (3.6) the second task velocity becomes

$$\boldsymbol{\nu}_2 = \boldsymbol{\Lambda}_2(\mathbf{t}_o - \boldsymbol{\eta}) \tag{3.21}$$

Then the null space $\mathbf{N_2}$ becomes zero according to (3.8) and any velocity projected into this null space becomes zero. However task 2.2 will never be active if task 2.1 is active, thus this null space will never be used.

Task 2.2: Go to target

The third task in the controller is go to target. A way to solve this is to consider a straight line between the vehicle and the target. Then the task variables σ_3 and $\sigma_{3,d}$ becomes

$$\boldsymbol{\sigma}_3 = \boldsymbol{\eta} \tag{3.22}$$
$$\boldsymbol{\sigma}_{3,d} = \mathbf{p}_{goal} \tag{3.23}$$
$$\tag{3.24}$$

And since it is only a straight line the Jacobi matrix becomes

$$\mathbf{J_3} = \mathbf{I} \tag{3.25}$$

where $\mathbf{I} \in \mathbb{R}^{2\times2}$. Then according to (3.6) the second task velocity becomes

$$\boldsymbol{\nu}_3 = \boldsymbol{\Lambda}_2(\mathbf{p}_{goal} - \boldsymbol{\eta}) \tag{3.26}$$

The null space of task 2.2 also becomes zero, but considering that there are no lower priority tasks this null space will not be used.

3.2.2 The resulting velocity vector

According to (3.7) the desired velocity vectors becomes

$$\boldsymbol{\nu}_d = \boldsymbol{\nu}_1 + (\mathbf{I} - \mathbf{J}_1^\dagger \mathbf{J}_1)\boldsymbol{\nu}_2 + (\mathbf{I} - \mathbf{J}_1^\dagger \mathbf{J}_1)\boldsymbol{\nu}_3 \tag{3.27}$$

When both $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ is active, it will result in a sliding motion around the obstacle and toward the target and when $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ is active, it will result in a sliding motion towards the point t_o on the obstacle circle. The resulting desired velocity vectors, $\boldsymbol{\nu}_d$ will not be used directly into the controller, but a heading controller will follow the angle to this vector. This will be descried more in Chapter 5.1

3.3 Stability of NSB

In order to analysis the convergence of the global task to see if the NSB can complete all tasks, one must evaluate the convergence of each task variable separately. Start with (3.27) and multiplying it with $\mathbf{J_1}$ and observing that $\mathbf{J}_1(\mathbf{I} - \mathbf{J}_1^\dagger \mathbf{J}_1) = \mathbf{0}$.

$$\mathbf{J}_1\boldsymbol{\nu}_d = \mathbf{J}_1\boldsymbol{\nu}_1 \tag{3.28}$$

Using (3.5) and (3.6) for $i = 1$ and inserting for $\boldsymbol{\nu}_d$ and $\boldsymbol{\nu}_1$, gives the error dynamic of the first task.

$$\dot{\boldsymbol{\sigma}}_1 = \dot{\boldsymbol{\sigma}}_{1,d} + \boldsymbol{\Lambda}(\boldsymbol{\sigma}_{1,d} - \boldsymbol{\sigma}_1) \tag{3.29}$$

$$\dot{\tilde{\boldsymbol{\sigma}}} = -\Lambda\tilde{\boldsymbol{\sigma}} \tag{3.30}$$

where $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_{1,d} - \boldsymbol{\sigma}_1$ and $\dot{\tilde{\boldsymbol{\sigma}}}_1 = \dot{\boldsymbol{\sigma}}_{1,d} - \dot{\boldsymbol{\sigma}}_1$ The proof that $\boldsymbol{\sigma}_1$ converge towards $\boldsymbol{\sigma}_{1,d}$ is straightforward, by using the Lyapunov function and Theorem 4.1 in [Khalil, 2001]

$$V = \frac{1}{2}\tilde{\boldsymbol{\sigma}_1}\mathbf{P}\tilde{\boldsymbol{\sigma}_1}^T \tag{3.31}$$

where

$$\mathbf{V}(0) = 0 \; and \; \mathbf{V}(\tilde{\boldsymbol{\sigma}}) > 0 \; in \; D - [0] \tag{3.32}$$

$$\begin{aligned} \dot{\mathbf{V}} &= \dot{\tilde{\boldsymbol{\sigma}}}_1\mathbf{P}\tilde{\boldsymbol{\sigma}_1}^T \\ \dot{\mathbf{V}} &= -\Lambda\tilde{\boldsymbol{\sigma}_1}\mathbf{P}\tilde{\boldsymbol{\sigma}_1}^T \\ \dot{\mathbf{V}} &< \mathbf{0} \; in \; D - [0] \end{aligned} \tag{3.33}$$

Where D is the domain containing $\tilde{\boldsymbol{\sigma}} = \mathbf{0}$ and since $\boldsymbol{\Lambda}$ and $\mathbf{P}$ are positive definite. Then $\tilde{\boldsymbol{\sigma}} = \mathbf{0}$ is asymptomatic stable, which means that $\tilde{\boldsymbol{\sigma}}$ converges to zero and $\boldsymbol{\sigma}_1$ converges towards $\boldsymbol{\sigma}_{1,d}$. Thus, the primary task is always fulfilled.

The second priority task 2.1 is only fulfilled if it is not in conflict with the higher priority task. By multiplying (3.27) with $\mathbf{J_2}$ and observing that $\mathbf{N}_2 = \mathbf{0}$

$$\boldsymbol{\nu}_d = \mathbf{J}_2\boldsymbol{\nu}_1 + \mathbf{J}_2(\mathbf{I} - \mathbf{J_1^\dagger J_1})\boldsymbol{\nu}_2 \tag{3.34}$$

Using (3.5) and (3.6) for $i = 1, 2$ and inserting for $\boldsymbol{\nu}_d$, $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$, gives

$$\dot{\boldsymbol{\sigma}}_2 = \mathbf{J}_2\mathbf{J}_1^\dagger(\dot{\boldsymbol{\sigma}}_{1,d} + \boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1) + \mathbf{J}_1(\mathbf{I} - \mathbf{J_1^\dagger J_1})\mathbf{J}_2^\dagger(\dot{\boldsymbol{\sigma}}_{2,d} + \boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2) \tag{3.35}$$

Assuming that there are no conflict:

$$\mathbf{J}_2\mathbf{J}_1^\dagger = \mathbf{0}$$

Then the resulting error dynamic for task 2 becomes

$$\dot{\tilde{\boldsymbol{\sigma}}}_2 = -\boldsymbol{\Lambda}_2(\boldsymbol{\sigma}_{2,d} - \boldsymbol{\sigma}_2) \tag{3.36}$$

Using the Lyapunov function and Theorem 4.1 in [Khalil, 2001]

$$V_2 = \frac{1}{2}\tilde{\boldsymbol{\sigma}}_2\mathbf{P}\tilde{\boldsymbol{\sigma}}_2^T \tag{3.37}$$

where

$$\mathbf{V}_2(0) = 0 \textit{ and } \mathbf{V}_2(\tilde{\boldsymbol{\sigma}}_2) > 0 \textit{ in } D - [0] \tag{3.38}$$

$$\begin{aligned}\dot{\mathbf{V}}_2 &= \dot{\tilde{\boldsymbol{\sigma}}}_2\mathbf{P}\tilde{\boldsymbol{\sigma}}_2^T \\ \dot{\mathbf{V}}_2 &= -\Lambda_2\tilde{\boldsymbol{\sigma}}_2\mathbf{P}\tilde{\boldsymbol{\sigma}}_2^T \textit{ in } D - [0]\end{aligned} \tag{3.39}$$

which is negative definite and $\tilde{\boldsymbol{\sigma}}_2$ is asymptotic stability stable and will converge to zero, which mean that $\boldsymbol{\sigma}_2$ converge towards $\boldsymbol{\sigma}_{2,d}$. The assumption that $\mathbf{J}_2\mathbf{J}_1^\dagger = \mathbf{0}$ can be verified since the point $\boldsymbol{\sigma}_{2,d}$ is on the obstacle circle, thus the NSB controller can achieve both task one and two.
The stability analysis of task 2.2 becomes very alike the stability analysis of task 2.1. The difference is that the assumption $\mathbf{J}_2\mathbf{J}_1^\dagger = \mathbf{0}$ is not true as long as the target is no the obstacle circle and it will not converge as long the null space of task two is zero. However as a separate task it will converge to the target.

3.4 Limitations and improvement

Before modelling the NSB controller into the vehicle simulator a test system have been implmented to test the algorithm, consisting of the NSB controller and an integrator for transforming the velocity vector to position measurement as shown in Figure 3.2. This system was only used to test the NSB algorithm for static obstacles, thus only task 1 and task 2.2 was implemented. From the stability analysis and testing the algorithm it was discovered four problems:

1. After avoiding the obstacle the vehicle got stuck on the obstacle circle and did not continue to the target. This is shown Figure 3.3 and is because of the desired velocity vector enters a equilibrium point. If tasks 1 and 2.2 are active, this equilibrium point is located in the intersect point between the obstacle circle and the line from the center of obstacle to the target.

2. If the vehicle start position, obstacle center and the target are placed on a straight line, then the vehicle will stop on the obstacle circle and not reach the target. This is shown Figure 3.4. This is a very unlikely situation, since the center of the obstacle has to be exactly on the line from the vehicle to the obstacle. A small change in heading or noise in the system would have solved this problem.

3. The vehicle is pulled toward the obstacle even if the vehicle is not on collision courses with the obstacle. This is shown in Figure 3.5 and the solid line is the path for the vehicle and the doted line is a straight line from start to target. It is desired that the vehicle should follow this line instead of approaching the obstacle. The reason that the vehicle is pulled towards the obstacle is very simple. $\boldsymbol{\nu}_1$ gives a velocity contribution toward the obstacle, even if the vehicle is not on a collision course

4. The method can fail if two obstacles overlapped each other. This is illustrated in Figure 3.6, and it can be seen that the vehicle travel around the first obstacle and then into the safety circle to the second obstacle. Also in this case, the vehicle gets stuck between the two obstacles.

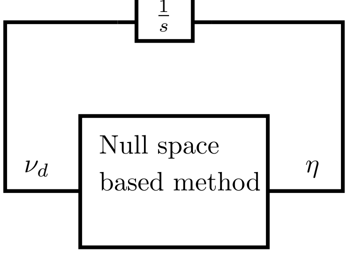


Figure 3.2: Simple test system used to test the NSB algorithm

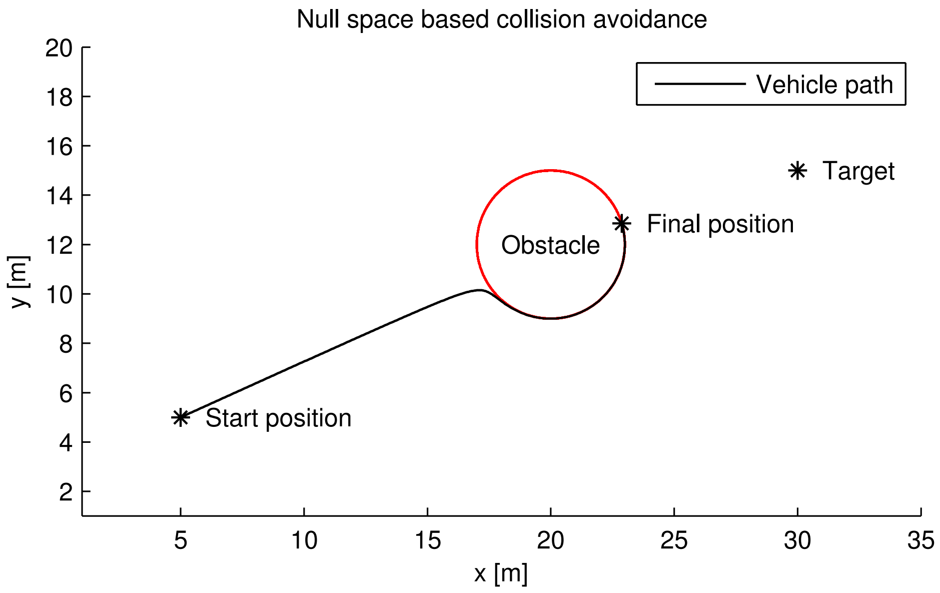


Figure 3.3: The vehicle gets stuck when trying to pass the obstacle

3.5 Summery

This chapter has presented a mathematical description of a three task NSB controller for collision avoidance, performed a stability analysis and constructed a test system for the NSB controller. From the stability analysis and the test system it was discovered some limitations with this approach, which one must take into consideration when designing the final controller.

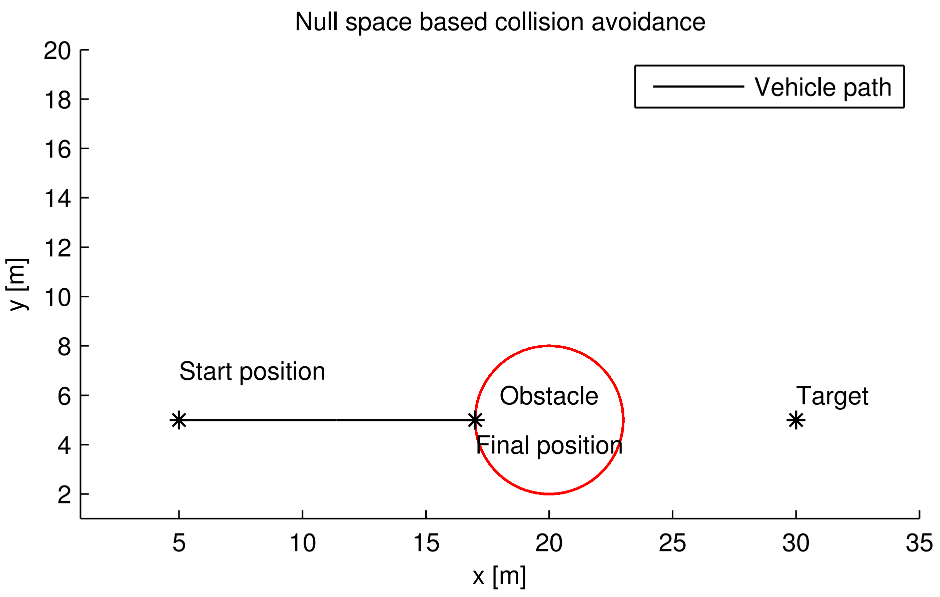


Figure 3.4: vehicle approach an obstacle head on, it gets stuck

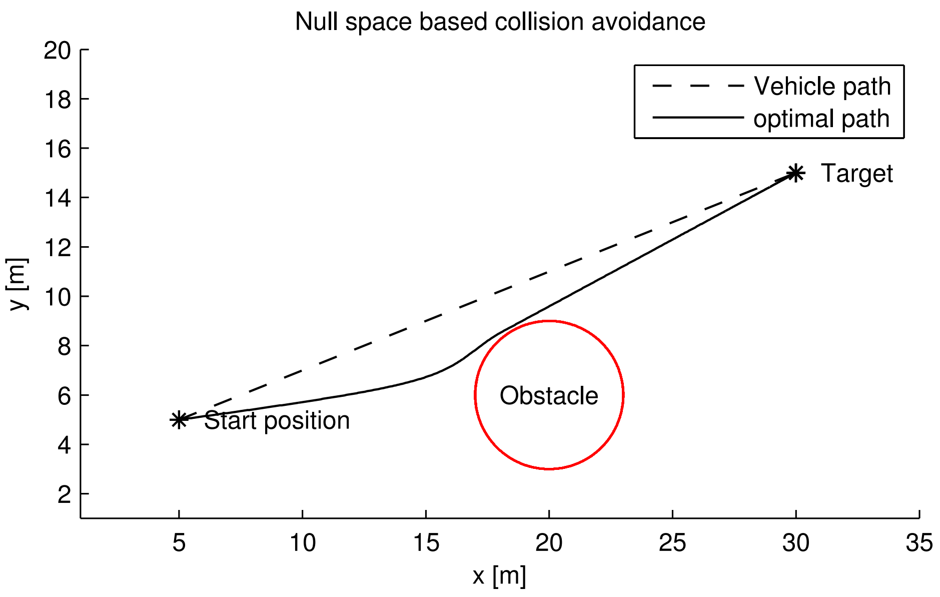


Figure 3.5: Unnecessary obstacle avoidance

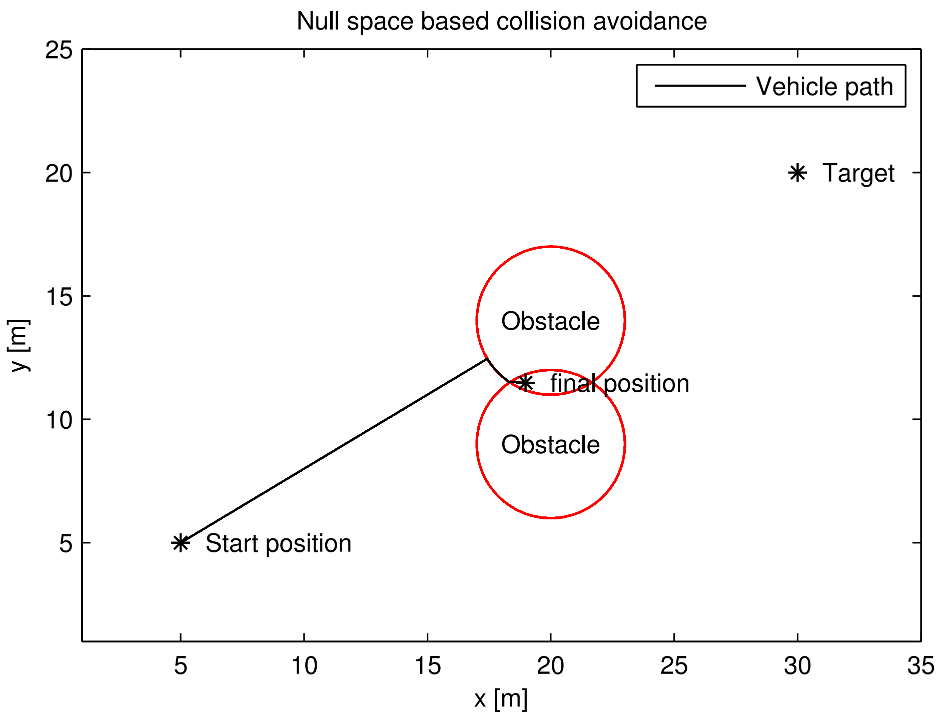


Figure 3.6: Overlapping obstacles

Chapter 4

Using Dubins path for collision avoidance

The Dubins path has been proved by Lester Eli Dubins to be the shortest path between two points and is therefore considered an optimal method with respect to the length, [Dubins, 1979]. The method was originally intended for path planning, but in thesis the method will be implemented as a waypoint collision avoidance method to avoid both static and dynamic obstacles.

The Dubins method has been chosen to be implemented as a collision avoidance method in order to compare the NSB controller against an optimal method and to study the difference between the NSB controller and the Dubins method considering the computational time.

This chapter is organized as following; first a general description of a Dubins path, then how to calculate the path from a starting position, around one single obstacle and multiple obstacles to a target. At the end of the chapter, it will be a discussion on how to implement the Dubins method for collision avoidance, which will be presented in Chapter 5.

4.1 Dubins path

Lester Eli Dubins showed in 1957 a mathematical proof that the shortest path between to points consist of straight lines and arc circles. For more detail on this proof see [Dubins, 1979].[Tsourdos et al., 2011] has the following definition of a Dubins path; The shortest possible path that meets the maximum curvature bound between two points with specific orientations in a plane is either a CLC or a CCC path, or a subset of them, where C represents circular arc and L represents straight-line tangent to C. A Dubins path is illustrated in Figure 4.1 where the black line is the path, P_s is the start position and P_f is final position. It can be seen that the direction on the line from P_x to P_n is equal the tangents to the two circles in the points P_x and P_n. P_x is the end of the first arc circle and P_n is the start of the second arc circle in Figure 4.1. The procedure of making a Dubins path in 3-dimensional can be found in [Tsourdos et al., 2011]. This chapter will only focus on the 2-dimensional path.

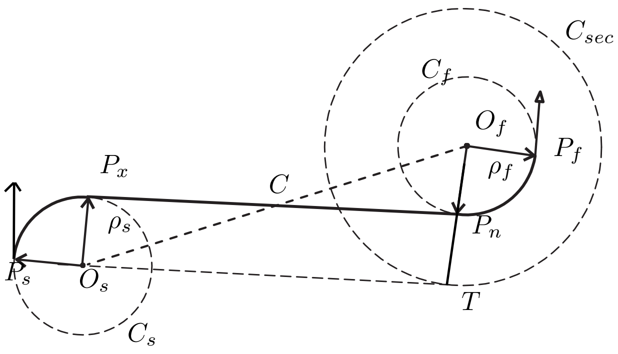


Figure 4.1: Dubins path with internal tangent, Reprinted from Cooperative Path Planning of Unmanned Aerial Vehicles [Tsourdos et al., 2011]

4.2 Dubins methods for collision avoidance

This section will show how to find a Dubins path around one or several obstacles and to a target by calculating waypoints to the start of the arc circles and the end of the arc circles, which represent the obstacles. The problem has been divided in two sub-problems

- First, find a Dubins path from a start position and around one single obstacle to a target.
- Second, find a Dubins path between two obstacles.

4.2.1 Dubins path around one obstacle to a target

Finding a Dubins path from a start position and around one obstacle to a target has been done by using vector geometry and the result is shown in Figure 4.2. The figure shows two valid Dubins paths from a start position $\boldsymbol{p}_{start}$ and around one single obstacle with center $\boldsymbol{p}_o$ whit radius r_o to a target $\boldsymbol{p}_{target}$. The path from $\boldsymbol{p}_{start} \rightarrow \boldsymbol{t2} \rightarrow \boldsymbol{t4} \rightarrow \boldsymbol{p}_{target}$ is define as the clockwise path and $\boldsymbol{p}_{start} \rightarrow \boldsymbol{t1} \rightarrow \boldsymbol{t3} \rightarrow \boldsymbol{p}_{target}$ anticlockwise. The goal is to find the unknown points $\boldsymbol{t1}, \boldsymbol{t2}, \boldsymbol{t3}, \boldsymbol{t4}$ and arch segments on the obstacle circle from $\boldsymbol{t1}$ to $\boldsymbol{t3}$ and $\boldsymbol{t2}$ to $\boldsymbol{t4}$.

The first step is to define the vectors $\boldsymbol{SO}$ from $\boldsymbol{p}_{start}$ to $\boldsymbol{p}_o$ and $\boldsymbol{ST_2}$ from $\boldsymbol{p}_{start}$ to $\boldsymbol{t2}$ with lengths d_{so} and d_{st2}.

$$\boldsymbol{SO} := \boldsymbol{p}_o - \boldsymbol{p}_{start}$$
$$\boldsymbol{ST_2} := \boldsymbol{t2} - \boldsymbol{p}_{start}$$

Then calculate the lengths d_{st2} and d_{so}. The length d_{st2} can be found by Pythagoras, since $\boldsymbol{ST_2}$ is unknown and d_{so} using the euclidean norm between $\boldsymbol{p}_o$ and $\boldsymbol{p}_{start}$

$$d_{so} = ||\boldsymbol{p}_o - \boldsymbol{p}_{start}||_2 \tag{4.1}$$
$$d_{st_2} = \sqrt{d_{so}^2 - r_o^2} \tag{4.2}$$

The second step is to find the angle θ_{t2} between the xy-plane and the vector $\boldsymbol{ST_2}$

$$\theta_{t2} = \theta_{t2,1} + \theta_{t2,2} \tag{4.3}$$

Where $\theta_{t2,1}$ is the angle between $\boldsymbol{SO}$ and $\boldsymbol{ST2}$ and $\theta_{t2,2}$ is the angle from $\boldsymbol{OS}$ to the xy-plane. $\boldsymbol{ST2}$ is unknown, but the length d_{st2} is not, thus the lengths r_o and d_{so} are used to find $\theta_{t2,1}$. The two angles then becomes

$$\theta_{t2,1} = asin(\frac{r_o}{d_{so}}) \tag{4.4}$$
$$\theta_{t2,2} = atan2(\boldsymbol{SO}_y, \boldsymbol{SO}_x) \tag{4.5}$$

The last step is to use the two dimensional rotation matrix

$$\mathbf{R}(\theta) := \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{4.6}$$

and the point $\boldsymbol{t2}$

$$t2 = \mathbf{R}(\theta_{t2}) \begin{bmatrix} 1 \\ 0 \end{bmatrix} d_t + \boldsymbol{p}_{start} \tag{4.7}$$

Finding $\boldsymbol{t1}$ is very similar and the same angles $\theta_{t2,2}$ and $\theta_{t2,1}$ can be used, the only difference is the rotation direction.

$$\theta_{t1} = \theta_{t2,2} - \theta_{t2,1} \tag{4.8}$$

and $\boldsymbol{t1}$ becomes

$$\boldsymbol{t1} = \mathbf{R}(\theta_{t1}) \begin{bmatrix} 1 \\ 0 \end{bmatrix} d_t + \boldsymbol{p}_{start} \tag{4.9}$$

The two line from $\boldsymbol{t3}$ and $\boldsymbol{t4}$ to the target can be found using the same approach that was used to $\boldsymbol{t1}$ and $\boldsymbol{t2}$, only switching the $\boldsymbol{p}_{start}$ with the $\boldsymbol{p}_{target}$.

To target

$$\boldsymbol{t3} = \mathbf{R}(\theta_{t3}) \begin{bmatrix} 1 \\ 0 \end{bmatrix} d_{obstaclTarget} + \boldsymbol{p}_{target} \tag{4.10}$$

$$\boldsymbol{t4} = \mathbf{R}(\theta_{t4}) \begin{bmatrix} 1 \\ 0 \end{bmatrix} d_{obstaclTarget} + \boldsymbol{p}_{target} \tag{4.11}$$

Where θ_3 , θ_4 and $d_{obstacleTarget}$ can be found in Appendix C.

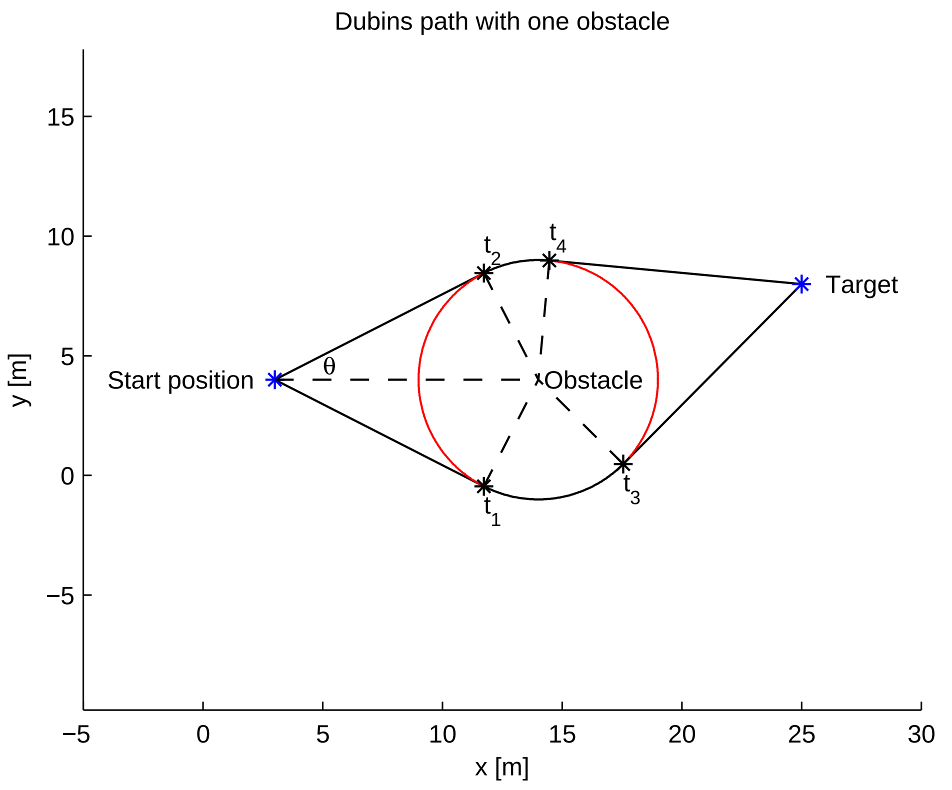


Figure 4.2: Dubins path with one obstacle

4.2.2 Dubins path between obstacles

How to generate a Dubins path between two obstacles is shown in Figure 4.3. The figure shows all Dubins paths between two obstacles. There is in total four possible paths and eight unknown points, which must be calculated, $\boldsymbol{t1}, \boldsymbol{t2}, \boldsymbol{t3}, \boldsymbol{t4}, \boldsymbol{t5}, \boldsymbol{t6}, \boldsymbol{t7}$ and $\boldsymbol{t8}$. This is done by dividing the problem into two parts. First finding the paths with opposite rotational direction, $\boldsymbol{t1} \rightarrow \boldsymbol{t2}$ and $\boldsymbol{t5} \rightarrow \boldsymbol{t6}$. Then the paths with same rotational direction, $\boldsymbol{t3} \rightarrow \boldsymbol{t4}$ and $\boldsymbol{t7} \rightarrow \boldsymbol{t8}$. The obstacle to the left in Figure 4.3 is defined as obstacle one and the obstacle to the right as obstacle two. Finding the Dubins path between two obstacles is slightly different than finding the paths around only one single obstacle. However, one can use the same strategy for finding these paths.

1. Find the length from a known point to the unknown.

2. Find the angle from a known point to a unknown point.

3. Rotating with the known angle and length to the unknown point.

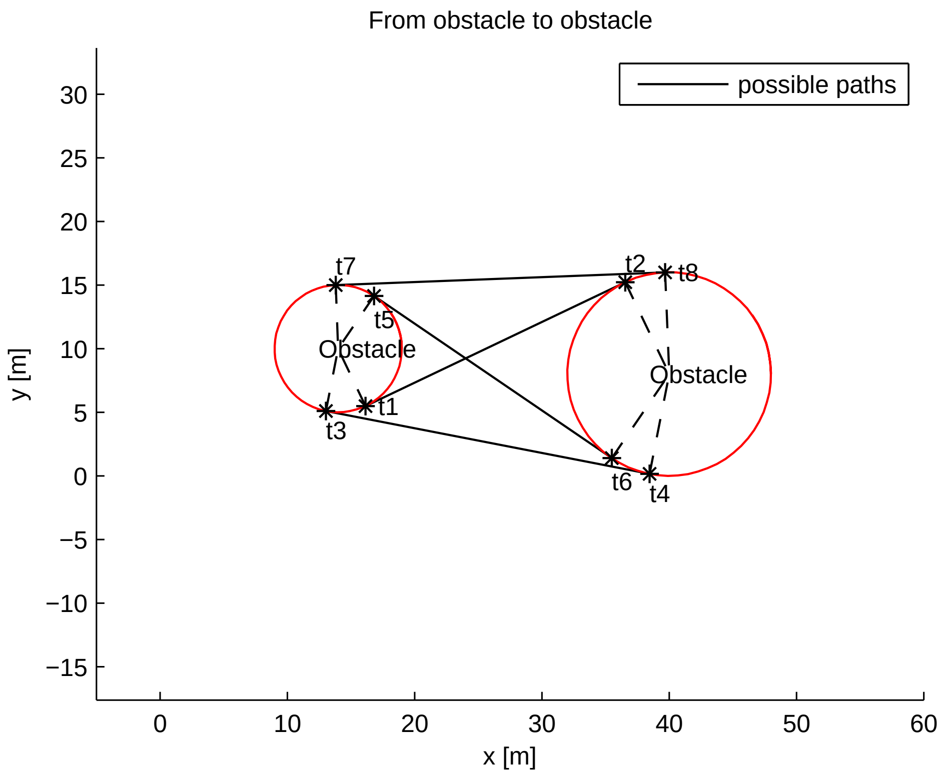


Figure 4.3: Dubins paths from one obstacle to another obstacle

Finding the Dublin path between two obstacles with opposite rotation direction

Figure 4.4(a) shows one Dubins path from one obstacle to another with opposite rotational direction. Both of the tangent point $\boldsymbol{t1}$ and $\boldsymbol{t2}$ are unknown and the only information, which is known is the positions and the radius of the obstacles, $\boldsymbol{p}_{o1}$,$\boldsymbol{p}_{o2}$, r_o and r_{o2}. The first step is to find the length between the two obstacles d_{oo}, which is given by

$$d_{oo} = ||\boldsymbol{p}_{o1} - \boldsymbol{p}_{02}|| \tag{4.12}$$

Then using Pythagoras to find the length, d_{tt} between the two tangent point ,$\boldsymbol{t1}$ and $\boldsymbol{t2}$

$$d_{tt} = \sqrt{d_{oo}^2 - (r_{o2} + r_{o1})^2} \tag{4.13}$$

Then define two vectors from the center of the first obstacle to $\boldsymbol{t2}$ as $\boldsymbol{V}_{o1t2}$ with length d_{o1t2} and from the second obstacle center to $\boldsymbol{t1}$ as $\boldsymbol{V}_{O2t1}$ with length d_{O2t1}.

$$\boldsymbol{V}_{o1t2} := \boldsymbol{t2} - \boldsymbol{p}_{o1}$$
$$\boldsymbol{V}_{o2t1} := \boldsymbol{t1} - \boldsymbol{p}_{o2}$$

These vectors are unknown, but the length can be found by Pythagoras

$$d_{O1T2} = \sqrt{d_{oo}^2 + r_o^2} \tag{4.14}$$

$$d_{O2T1} = \sqrt{d_{oo}^2 + r_{o2}^2} \tag{4.15}$$

The last step is to find the angles from $\boldsymbol{p}_{O1}$ and $\boldsymbol{p}_{O2}$ to $\boldsymbol{t1}$ and $\boldsymbol{t2}$ such that we can rotate and find the tangent point $\boldsymbol{t2}$ and $\boldsymbol{t1}$. The cosine rule gives the angle between $\boldsymbol{V}_{O1T2}$ and $\boldsymbol{V}_{O1T1}$ and the angle between $\boldsymbol{V}_{O2T1}$ and $\boldsymbol{V}_{O2T2}$

$$\theta_1 = acos(\frac{d_{oo}^2 + d_{O2T1}^2 - r_{o2}^2}{2 * d_{oo} + d_{O2T1}}) \tag{4.16}$$

$$\theta_2 = acos(\frac{d_{oo}^2 + d_{O1T2}^2 - r_o^2}{2 * d_{oo} + d_{O1T2}}) \tag{4.17}$$

and the angle to vector $\boldsymbol{V}_{o1o2}$ between the two obstacle in the xy-plane

$$\phi = atan2(\boldsymbol{V}_{o1o2,y}, \boldsymbol{V}_{o1o2,x}) \tag{4.18}$$

Then the tangent point $\boldsymbol{t1}$ and $\boldsymbol{t2}$ becomes

$$\boldsymbol{t1} = \mathbf{R}(\phi + \theta_2 + \pi)) \begin{bmatrix} 1 \\ 0 \end{bmatrix} d_{O2T1} + \boldsymbol{p}_{O1} \tag{4.19}$$

$$\boldsymbol{t2} = \mathbf{R}(\phi + \theta_1) \begin{bmatrix} 1 \\ 0 \end{bmatrix} d_{O1T2} + \boldsymbol{p}_{O2} \tag{4.20}$$

Finding the path from $\boldsymbol{t5}$ and $\boldsymbol{t6}$ is very similar since the two obstacle are symmetric the same geometry is applied.

Finding the Dublin path between two obstacles with same rotation direction

Figure 4.4(b) shows a Dubins path between two obstacles with the same rotation direction. Finding this path is slightly easier than finding a Dubins path between two obstacle with opposite rotation direction. From Figure 4.4(b) one can identify that the rectangle given by the corners $\boldsymbol{p}_{O1}$, $\boldsymbol{p}_{O2}$, $\boldsymbol{t4}$ and $\boldsymbol{t3}$ can be divided into one right angled triangle and one rectangle. The first step is to find the length d_{t3t4} from $\boldsymbol{t3}$ to $\boldsymbol{t4}$. Since $\boldsymbol{p}_{01}$, $\boldsymbol{p}_{O2}$ and $\boldsymbol{t5}$ makes right angled triangle and define the vectors $\boldsymbol{V}_{O1T5}$ from $\boldsymbol{p}_{O1}$ to $\boldsymbol{t5}$, with length d_{O1T5}, which is parallel to vector $\boldsymbol{V}_{t3t4}$ from $\boldsymbol{t3}$ to $\boldsymbol{t4}$ with length d_{t3t4}.

$$\boldsymbol{V}_{O1T5} := \boldsymbol{t5} - \boldsymbol{p}_{O1}$$
$$\boldsymbol{V}_{t3t4} := \boldsymbol{t4} - \boldsymbol{t3}$$

Considering that the two vectors $\boldsymbol{V}_{O1T5}$ and $\boldsymbol{V}_{t3t4}$ are parallel and part of the same rectangle, which means that the length, d_{O1T5} is equal to d_{t3t4} and d_{t3t4} can be found by Pythagoras

$$d_{t3t4} = \sqrt{d_{oo}^2 - (r_{o2} - r_{o1})^2} \tag{4.21}$$

The second step is to find the angles, θ and ϕ. θ is the angle between the vector $\boldsymbol{V}_{o1o2}$ and $\boldsymbol{V}_{o1t5}$. ϕ is the angle from the xy-plane to the vector $\boldsymbol{V}_{o1o2}$, which is the vector between the two obstacles.

$$\theta = acos(\frac{d_{t3t4}}{d_{oo}}) \tag{4.22}$$

$$\phi = atan2(\boldsymbol{V}_{o1o2,y}, \boldsymbol{V}_{o1o2,x}) \tag{4.23}$$

Then the point $\boldsymbol{t3}$ and $\boldsymbol{t4}$ are given by

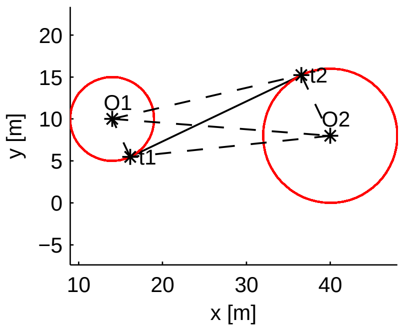
$$\boldsymbol{t3} = \mathbf{R}(\phi - \theta - \frac{\pi}{2}) \begin{bmatrix} 1 \\ 0 \end{bmatrix} r_o + \boldsymbol{p}_{O1} \tag{4.24}$$

$$\boldsymbol{t4} = \mathbf{R}(\frac{3\pi}{2} - \theta + \phi) \begin{bmatrix} 1 \\ 0 \end{bmatrix} r_{o2} + \boldsymbol{p}_{O2} \tag{4.25}$$

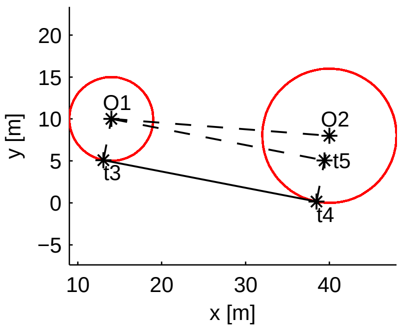
Finding $\boldsymbol{t7}$ and $\boldsymbol{t8}$ in Figure 4.3 is very similar sine the two obstacle are symmetric the same geometry is applied.

4.3 The length of a Dubins path

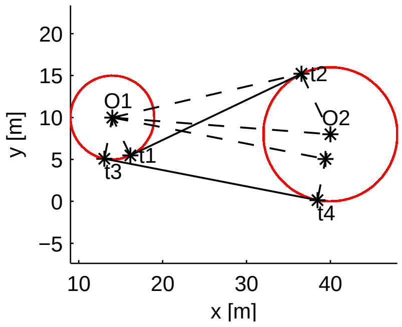
The length of the Dubins path consist only of lines and arc lengths, thus the sum of these length will give us the total length of the Dubins path. The arc length is



(a) Dubins path between two obstacle with opposite rotation direction



(b) Dubins path between two obstacle with the same rotation direction



(c) Dubins path between two obstacle with the same and opposite rotation direction

Figure 4.4: Dubins paths between two obstacles

given by

$$L_b = r_o \Delta\theta$$

where $\Delta\theta$ are the angles between the first and last waypoint on the obstacle circle. The total length is given by the

$$L = \sum_{i_1=0}^{n_1} L_{li_1} + \sum_{i_2=0}^{n_2} L_{bi_2} \tag{4.26}$$

where L_{li_1} is the length of the line and L_{bi_2} are the arc length. The shortest path from start to target can be found by iteration all the possible Dubins path using (4.26) to find the shortest.

4.4 Summary and discussion

In this chapter it has been shown how to generate a Dubins path from a start position, around one or two obstacles and to the target. The tangent points $\boldsymbol{t}_i$ can be used as waypoints in the development of the Dubins method for collision avoidance with a steering law that follows the waypoints. However as seen in Figure 4.5, which shows all four possible paths from a start position, around two obstacles and to a target the problem becomes what path to choose. If the method chooses to consider all obstacles in range such as in Figure 4.5, equation (4.26) can be used on all paths to find the shortest. This will give n^2 possible paths, which the method will have to check in order to find the shortest, where n is the number of obstacles in range of the vehicle. Also considering that the method does not know about all obstacles, which is in the path between the vehicle and the target and that the method can't foresee where the obstacles will be in the future. It might not be smartest to use the computational time to calculate all possible paths. Thus it is decided in this thesis that the Dubins method will only consider one obstacle at a time, which will result in less computational time and choose the passing direction on either the safest path or the shortest considering only one obstacle at a time. The implementation of the Dubins method for collision avoidance is described in Chapter 5.

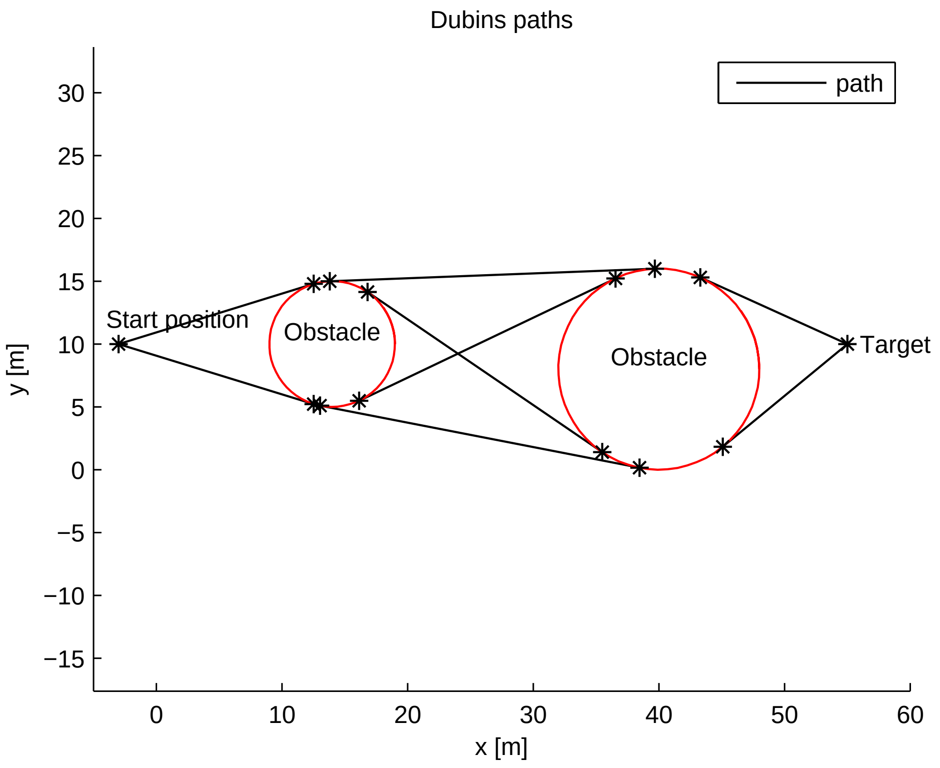


Figure 4.5: All Dubins path from start to target with tow obstacles

Chapter 5

Collision avoidance strategy and implementation of the guidance system

The theory about the two collision avoidance methods, which was chosen in this thesis, has been presented in Chapter 3 and 4. This chapter will present the implementation of the collision avoidance methods and the strategies to avoid collision, which is both part of the guidance system. When considering only static obstacles the collision avoidance problem is quite simple as opposed to dynamic obstacles. In a case with circle shaped static obstacles it is safe to both pass them clockwise and anticlockwise. However when considering dynamic obstacles with unknown future trajectory the problem becomes more challenging since the methods have to take in consideration and estimate where the obstacles will be in the future.

The guidance part of the model for the collision avoidance system is shown in Figure 5.1. This chapter is organized as following; first the collision avoidance strategies for static and dynamic obstacles, then the rest of the guidance system, which consists of the collision detection, collision avoidance and the steering law as shown in the Figure 5.1.

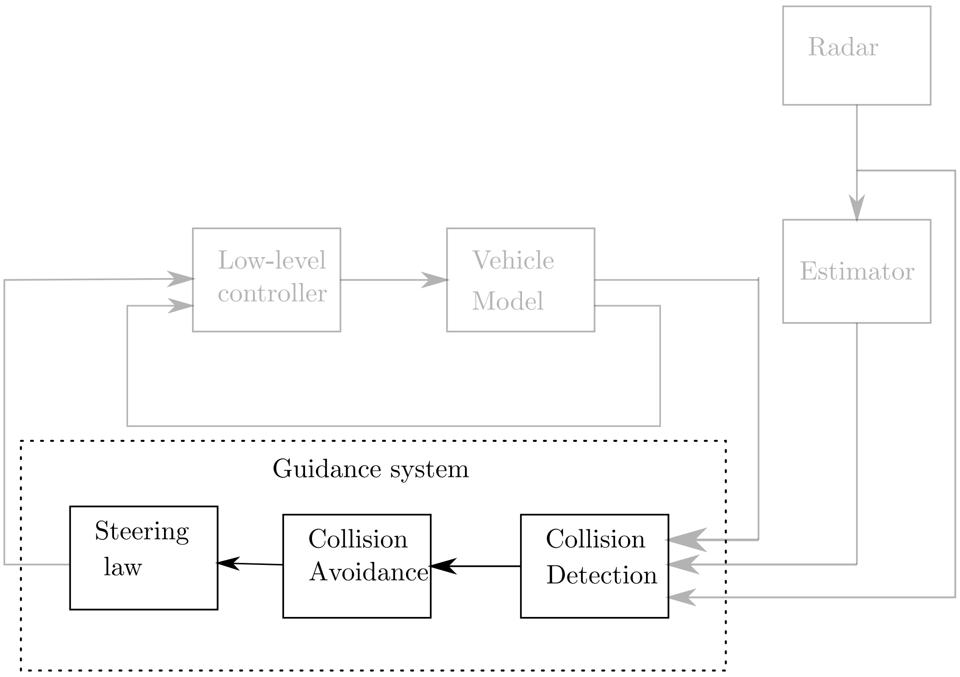


Figure 5.1: Model of the collision avoidance system, where the focus is on the guidance system

5.1 Collision avoidance strategies

The collision avoidance strategies will be based on the assumption that there are no communication between the vehicle and the obstacles, thus the vehicle has to consider that the obstacles will not do anything to prevent the collision. When detecting an obstacle on collision course with the vehicle, there are two main strategies to avoid a collision.

- Altering the speed, slow down or speed up, which allow the vehicle to maintain the preplanned path towards the target.
- Altering the vehicles heading and pass the obstacle clockwise or anticlockwise, which allow the vehicle to maintain the current speed.

The vehicle could also use a combination of the two and both slow down and alter the heading, which can be used in very dangerous situation. The decision if the vehicle should pass static obstacles clockwise or anticlockwise will be based on the shortest path around the obstacle and towards the target. The NSB controller will do this automatically, since (3.7) the two tasks *obstacle avoidance* and *go to target* active results in the shortest path to the target considering only one obstacle. The Dubins method has to test the angles between two vectors from the vehicle to the first waypoints on the obstacle circle and from the vehicle to the target, which can be found by the cosine rule. The smallest angle will give the shortest path around the obstacle. Considering dynamic obstacles the problem becomes more challenging and finding the shortest path around obstacles will no longer be considered, but finding the safest path around. This section describes three main collision situations, which can occur between the vehicle and one single obstacle. The main goal is to find out how the vehicle should pass dynamic obstacles, without colliding and minimizing the risk of collision in the future.

5.1.1 Obstacle approach; Heads-on

Figure 5.2 illustrates a heads-on situation, where the vehicle and an obstacle has opposite direction and is heading directly towards each other. This will lead to an extra dangerous situation since the relative velocity between them will become larger, thus the safety circle around the obstacle will be increased with a factor two. Altering the speed will not be an option, since this will not avoid the collision. Therefore only altering the heading will be considered and passing the obstacle clock or anticlockwise will be equally safe. However in this thesis the vehicle should pass anticlockwise in this situation as shown in the figure.

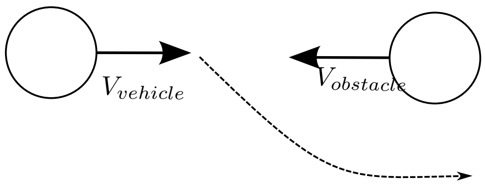


Figure 5.2: Heads-on situation the vehicle and the obstacle has opposites headings

5.1.2 Obstacle approach; from left or right

Figure 5.3 illustrates two cases with one obstacle on collision course with the vehicle where the passing direction should be anticlockwise. This can be seen from the two velocity vectors, if the vehicle had passed the obstacle clockwise the risk of the two velocity vectors could collide, thus passing anticlockwise would be safest. Figure 5.4 illustrates the opposite situation where the safest passing direction is clockwise to avoid the collision.

One could argue that the vehicle could alter the speed in these two cases to avoid an collision. In a narrow environment where it would be difficult to alter the preplanned path this could be a good alternative. This will not be considered in this thesis since it is assumed that the environment is not narrow and the NSB or the Dubins methods are not designed to altering the speed to avoid collision.

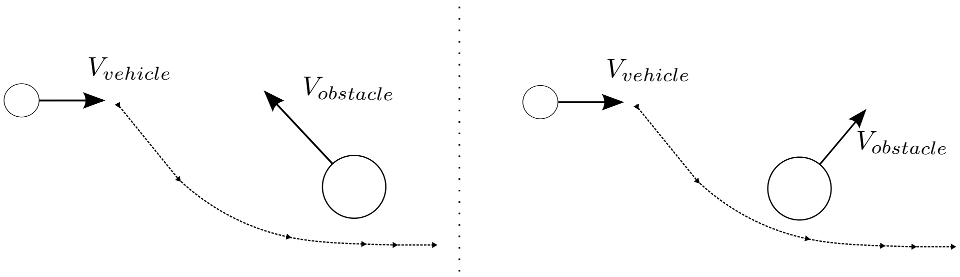


Figure 5.3: When the vehicle detect a collision and the passing direction is anticlockwise

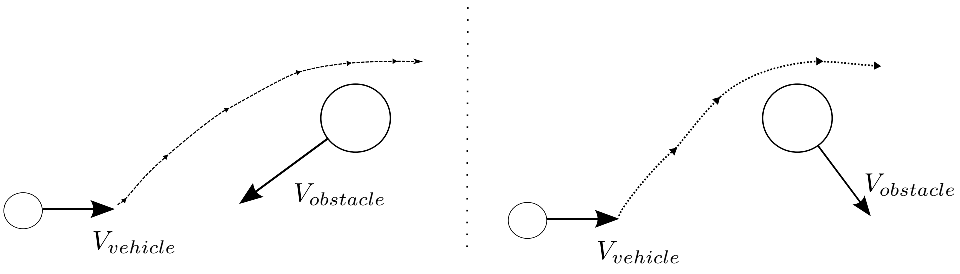


Figure 5.4: When the vehicle detect a collision and the passing direction is clockwise

5.1.3 Collision from behind

Figure 5.5 illustrates two cases with collision from behind, one where the vehicle has greater speed than the obstacle and approaching the obstacle from behind, and the other where the obstacle has greater speed than the vehicle and approaching the vehicle from behind. In the case where the vehicle is approaching the obstacle from behind, it is decided to pass the obstacle anticlockwise as seen in top of the figure. If the obstacle approaching the vehicle from behind the passing direction should be clockwise. Considering that the two methods are not very suited for handling an obstacle that approaches the vehicle from behind. It is placed a virtual obstacle in front of the vehicle with the same distance as the distance to the obstacle behind.

Such that the vehicle will actually pass the virtual obstacle. However, avoiding the virtual obstacle will also result in avoiding the real obstacle as seen in the bottom of the figure.

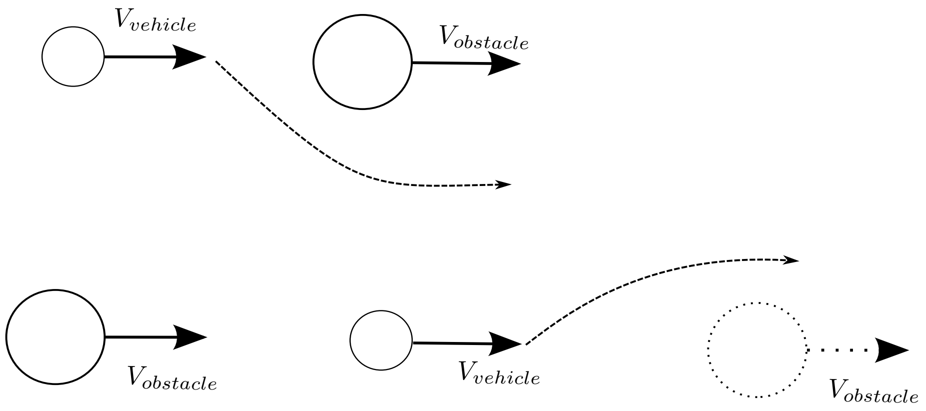


Figure 5.5: When the vehicle detect a collision and the vehicle is approaching the obstacle from behind and the other where the obstacle approaching the vehicle from behind.

5.1.4 Multiple obstacles

If there is more than one obstacle on collision course with the vehicle, only the obstacle, which will collide with the vehicle first will be processed by the collision avoidance methods.

5.1.5 Choose direction mathematically

In order to achieve the strategy described above the function *ChooseDirection* is implemented in the collision detection block. It uses the assumption that the vehicle knows the velocity vector and the bearing to the obstacle to decide if the vehicle should pass the obstacle clockwise or anticlockwise. Figure 5.6 illustrates how the function works in a case where the vehicle and an obstacle is on collision course. The function rotates the velocity vector to the obstacle $\boldsymbol{\nu}_o^n$, which is given in the NED frame to the BODY frame of the vehicle $\boldsymbol{\nu}_o^b$

$$\boldsymbol{\nu}_o^b = \mathbf{R}(\psi)\boldsymbol{\nu}_o^n \tag{5.1}$$

Where ψ is the vehicles heading and $\mathbf{R}(\psi) \in SO(2) \subset \mathbb{R}^{2\times 2}$ is the two dimensional rotation matrix

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{5.2}$$

which will give the velocity vector the obstacle in the NED frame of the vehicle. The direction to this vector β is given by

$$\beta = atan2(\boldsymbol{\nu}^b_{o,y}, \boldsymbol{\nu}^b_{o,x}) \tag{5.3}$$

Considering that atan2 only gives a four quadrant solution (from $-\pi < \theta < \pi$) and it is desired with a four quadrant solution (from $0 < \theta < 2\pi$), thus the function must add 2π if β becomes less than zero. Since β is the angle to the obstacle in the BODY frame of the vehicle, this will also be the angle in a four quadrant solution between the vehicle and the obstacle. Thus by testing which quadrant β lies in, the function determines the direction the vehicle should pass the obstacle on. This is illustrated in the right part of Figure 5.6, which shows the velocity vector to the obstacle rotated in the BODY frame of the vehicle and it can be seen that β is in the third quadrant in this case. Considering the collision avoidance strategy above, it is checked if β lays in the first or second quadrant. Then vehicle should pass clockwise and if the β lays in the third or fourth quadrant, the vehicle should pass anticlockwise. This will cause the two velocity vectors never to cross and the vehicle would always pass behind the obstacle, thus making the path safer.

The last things the function checks are; if the vehicle and the obstacle is on a heads-on collision course, if the vehicle is approaching an obstacle from behind or if the obstacle is approaching the vehicle from behind. The heads-on problem is solved by testing if

$$|\beta - \pi| < HeadsOnConstant$$

is true or not. The $HeadsOnThreshold$ is a constant, which is used to define the heads-on situation. In this thesis it is considered to be a heads-on situation if the vehicle and the obstacle has opposite direction ± 20 degrees, thus the heads-on constant is set as ±0.35 rad (± 20 degrees). In the same way, the approach from behind situations is defined. By testing if

$$\beta < approachFromBehindConstant\ Or$$
$$2\pi - \beta < approachFromBehindConstant$$

is true or not. If it is true a second test is performed to see if the vehicle is approaching the obstacle or the obstacle is approaching the vehicle. By testing if the vehicle is closer to the target than the obstacle. The $approachFromBehindConstant$ is set to 0.2 rad in this thesis (11.5 degrees)

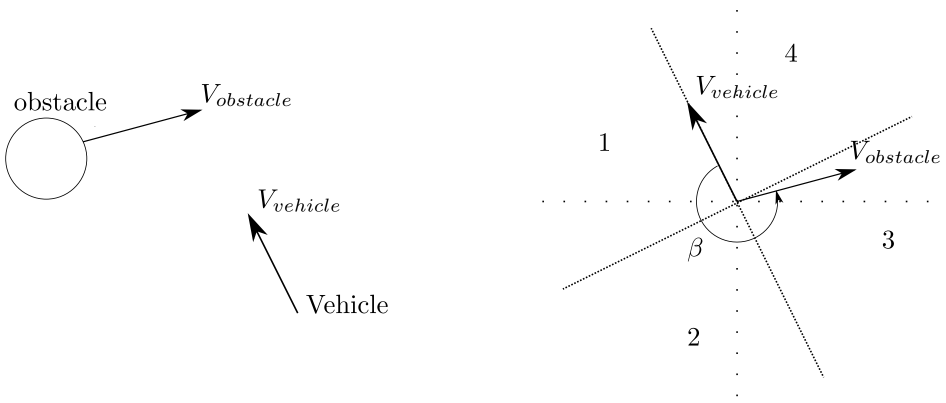


Figure 5.6: The figure shows the vehicle and the obstacle on a collision course and the velocity vector to the obstacle is rotated into the BODY frame to the vehicle. The 1, 2, 3 and 4 illustrate the first, second, third and fourth quadrant.

5.2 Collision detection

The collision detection method developed in this thesis should determine if any obstacles in range of the vehicle is on collision course with the vehicle and if there are more than one, find the one that is the biggest threat. The method has defined a constant value called threat, which is set to 1 if there is a threat and zero otherwise. It is assumed that the vehicle knows the current velocity of the obstacles, but not the future velocity. Therefore the method assumes that the vehicle and the obstacle will continue with the same velocity to calculate the future trajectories. Thus the method needs to redo and retest if the obstacles and the vehicle is on collision course in every iteration. Figure 5.7 illustrate how the method works where the vehicle is placed in three different places and the estimated trajectories are plotted as the dotted lines. First the vehicle discovers an obstacle and based on the two trajectories the collision detection activates the collision avoidance system. In the second place, the method retest if the obstacle and the vehicle are still on collision cures. By using the estimated trajectory from the obstacle and the trajectory the vehicle would have if the vehicle had moved directly to the target. Finally the path to the target is clear and the collision detection should deactivate the collision avoidance system by setting the threat value to zero. The collision detection system returns the position and safety radius for the obstacle to the collision avoidance system.

The trajectory of the vehicle $\boldsymbol{N}(t)$ and the obstacle $\boldsymbol{O}(t)$ are estimated using the

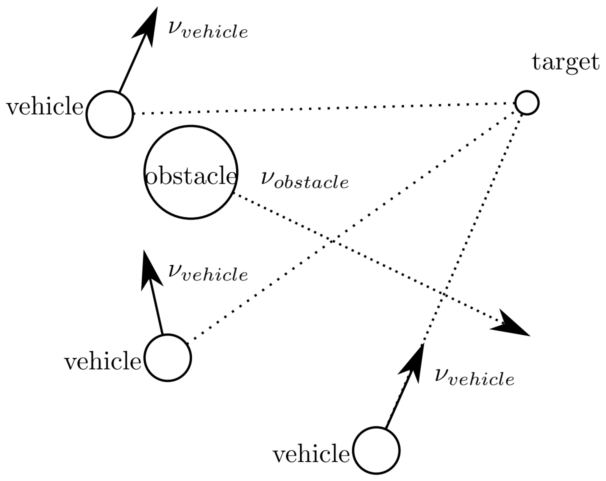


Figure 5.7: Collision detection method checks if the trajectory for the vehicle to the target collide with the trajectory of the obstacle

equation of motion.

$$\boldsymbol{O}(t) = \begin{bmatrix} x_o(t) \\ y_o(t) \end{bmatrix} = \begin{bmatrix} V_o t\cos(\theta_o) + p_{o,x} \\ V_o t\sin(\theta_o) + p_{o,y} \end{bmatrix} \tag{5.4}$$

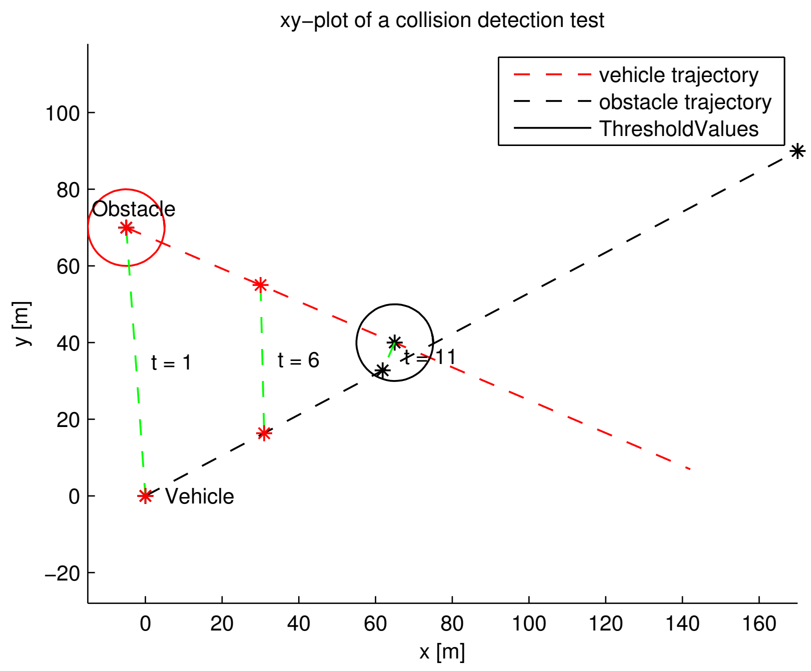
$$\boldsymbol{N}(t) = \begin{bmatrix} x_v(t) \\ y_v(t) \end{bmatrix} = \begin{bmatrix} V_v t\cos(\theta_{vehicleTarget}) + p_{v,x} \\ V_v t\sin(\theta_{vehicleTarget}) + p_{v,y} \end{bmatrix} \tag{5.5}$$

Where θ_o and $\theta_{vehicleTarget}$ are the headings to the obstacle and the direction to the vector between the vehicle and the target. V_o and V_v are the current speeds of the obstacle and the vehicle. t is the time horizon and have been set from 0 to 50 second in this thesis. By finding the distance $f(t)$ between these two trajectories

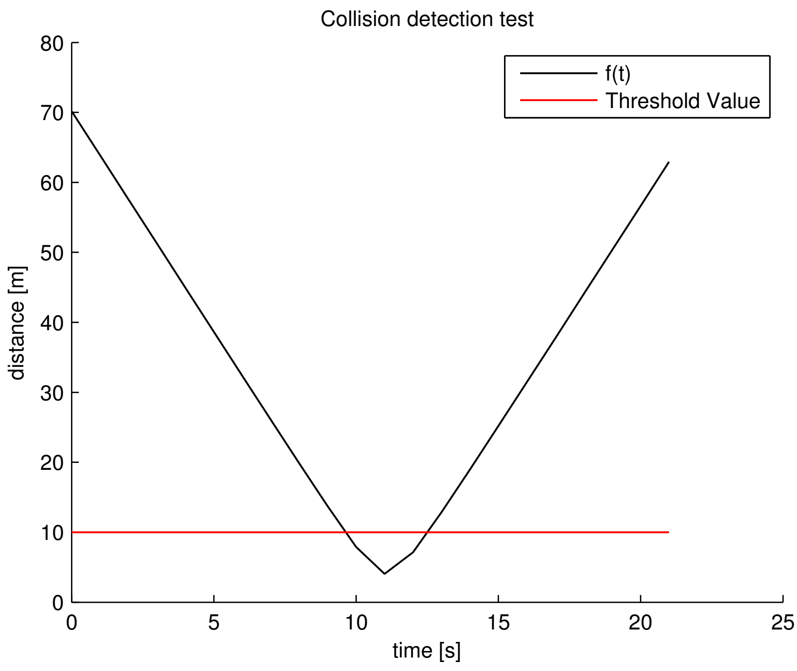
$$f(t) = ||\boldsymbol{N}(t) - \boldsymbol{O}(t)||_2 \tag{5.6}$$

where $||\bullet||$ is euclidean norm and testing if $f(t)$ is less than a threshold value (less than the safety radius for the obstacle). A possible collision is detected and the collision avoidance system is activated. There can be a possibility of more than one obstacle that can be a threat. Therefore the method calculate the length C from the vehicle position to $\boldsymbol{O}(t_{FirstCollision})$, where $t_{FirstCollision}$ is the first t in $f(t)$ that is under the threshold value. The obstacle, which is a threat and has lowest C value is considered the biggest threat.

Figure 5.8(a) shows an algorithm test of the collision detection system. The estimated trajectories are plotted in the xy-plane and the green line between the two



(a) xy plot of the estimated trajectory of the vehicle and the obstacle. Where the distance between the two trajectory at time t is shown in the green dotted line for t = 1, 6 and 11. The black circle is the threshold value



(b) The distance function between the obstacle and the vehicle, if the distance function becomes less than the threshold value the collision avoidance system is activated

Figure 5.8: Collision detection test, the two figures shows the estimated trajectories and the distance between then

trajectories are $f(t)$ at t= 1, 6 and 11, where t=11 is the first value under the threshold value. Thus the collision avoidance system should be activated and the C value should be calculated from this point. Figure 5.8(b) shows the entire $f(t)$ function to Figure 5.8(a) with the threshold value.

5.3 Implementation of the Null-Space-Based behaviour control

The NSB controller developed in this thesis should be able to handle both static and dynamic obstacles in a safe manner. The controller should also be able to follow the desired direction around the obstacles (clockwise or anticlockwise), according to the collision avoidance strategies presented in Section 5.1. As a result of this, the three tasks NSB controller described in Chapter 3 have been implemented

- Task 1: Obstacle avoidance
- Task 2.1: Traverse obstacle
- Task 2.2: Go to target

Task 1 is the obstacle avoidance task, which make sure that the vehicle does not enter the safety radius of the obstacles. This task is only active inside activation circle one. The activation circles one and two is shown in Figure 5.10. Task 2.1 makes the vehicle converge to a point on the obstacle circle depending on the passing direction such that the vehicle can choose direction around the obstacles and is activated inside of activation circle one. Figure 5.9 shows the two tangent $t_{1,2}$ points on the obstacle circle, which task 2.1 converges towards. When the vehicle has passed the dotted line from the target and through the obstacle, task 2.1 can be turned off. Since the vehicle will continue passing the obstacle with the same direction if task 2.2 is activated. Task 2.2 will makes the vehicle converge to the target. Table 5.1 shows the tuning parameter for the tasks in the NSB controller and the size of the activation circles, the d is the size of the safety circle to the obstacle. The main reason for the two activation circles in the NSB controller is that the vehicle can converge to the tangent points without having to project the velocity vector in the null-space of task one, which would have resulted in curved motion towards the point t_i instead of a straight line.

The pseudo code of the NSB controller implemented is shown bellow and the threat value in the code is received from the collision detection method, see Section 5.2. It can be seen that task 2.1 can be turned off and on by setting $\nu_{2,1}$ to zero and the null-space equal the identity matrix, thus task 2.1 will not give any contribution to the desired velocity, making it a two task controller with only obstacle avoidance and go to target. Or setting $\nu_{2,1}$ according to (3.21) and the null-space equal zero,

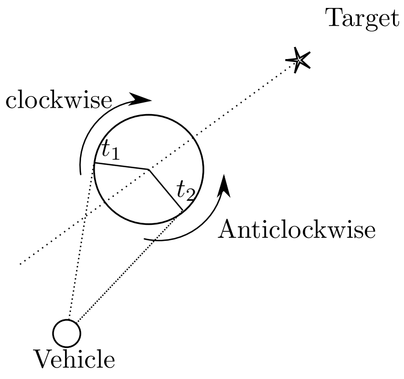


Figure 5.9: Shows the two tangent points on the obstacle circle and the dotted line from the target illustrates, which side the vehicle has to be on in order to pass the obstacle clockwise or anticlockwise

thus turning of task 2.2 of. Assuming task 2.1 is turned of and the vehicle is inside of activation circle two the desired velocity becomes ν_1 plus $\nu_{2,2}$ projected into the null-space of task one, thus choosing the shortest paths around the obstacle will avoiding it. When the obstacle is avoided the obstacle detection system sets the threat value false and the vehicle continues toward the target with the desired velocity equal $\nu_{2,2}$.

For dynamic obstacles or static obstacles where the vehicle is heading directly towards the obstacle center $\nu_{2,1}$ gives a velocity contribution toward a point t_i depending on the passing direction according to (4.9) or (4.7) on the obstacle circle. When the vehicle is inside of activation circle one the desired velocity becomes only $\nu_{2,1}$, thus making the vehicle choose the passing direction around the obstacle. Inside of the activation circle two the desired velocity becomes ν_1 plus $\nu_{2,1}$ projected into the null-space of task one until the vehicle has reached t. Then $\nu_{2,1}$ is set to zero and the null-space becomes an identity matrix. Thus making the desired velocity ν_1 plus $\nu_{2,2}$ projected into the null-space of task one until the threat is avoided. The HasReachObstacle function is the function that tests if the vehicle is close or has passed the point t_i on the obstacle circle.

Table 5.1: Tuning parameter for NSB in case studies in Chapter 7

Case number	λ_1	λ_2	λ_3	Activation circle one	Activation circle two
One	5	5	3	-	$3d$
Two	5	5	10	$10d$	$1.6d$
Three	5	5	10	$10d$	$1.6d$
Four	5	5	10	$10d$	$1.6d$

NSB controller: pseudo code

```
$\nu_1$ according to (3.15)
$\nu_3$ according to (3.26)
$N_1$ according to (3.16)
**if** HasReachObstacle == true Or ActivateTaskTwo == false **then**
    $\nu_{2,1} = [0,0]^T$
    $N_{2,1} = I$
**else**
    $\nu_{2,1}$ according to (3.21)
    $N_{2,1} = 0$
**end if**
**if** Activation circle one == TRUE and Threat == true **then**
    $V_d = \nu_{2,1} + N_{2,1}\nu_{2,2}$
**else if** Activation circle two == true and Threat == true **then**
    $V_d = \nu_1 + N_1\nu_{2,1} + N_1 N_{2,1}\nu_{2,2}$
    **if** Inside the safety circle **then**
        $V_d = \nu_1$
    **end if**
**else**
    $V_d = \nu_{2,2}$
**end if**
```

5.4 Implementation of the Dubins method for collision avoidance

The Dubins method for collision avoidance developed in this thesis should, as the NSB controller be able to handle static and dynamic obstacles. The Dubins method for collision avoidance developed using the theory in Chapter 4 to generate way-points, which generates a path that fulfills the criteria of a Dubins path. However, considering that the method only consider one obstacle at a time and the vehicle can not turn infinitely fast, the path might differ from an optimal Dubins path

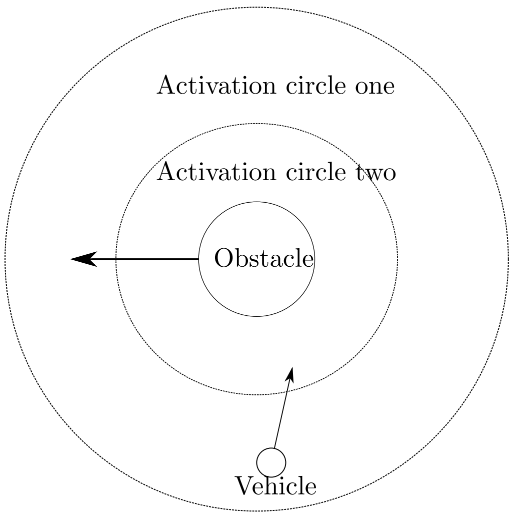


Figure 5.10: Activation circles for the NSB controller. The figure shows the two activation circles implemented in the NSB controller where the vehicle is inside of activation circle one

calculated off line where all obstacles are known. The waypoints generated by the method are sent to a steering law such that the vehicle can follow the generated path. Figure 5.11 shows how the Dubins method generate waypoints for avoiding an obstacle when the vehicle is outside of the activation circle. The only difference between inside or outside the circle is inside the vehicle does not need to run the go to obstacle function while outside it needs to find the first waypoint on the obstacle circle. The pseudo code for the method is shown below. It can be seen that the method is divided into three main cases.

- If there is no threat
- If there is a threat and the obstacle is dynamic
- If there is a threat and the obstacle is static

The first one, if no obstacles are in range or are considered to be a threat, go directly to the target. The difference between the last two is that, for static obstacles the code is only run once, while for dynamic the code is repeated and regenerate a new set of waypoints as the obstacles are moving. It can be seen that the method for all three cases first generate one waypoint according to (4.7) or (4.9) (depending on the passing direction), then the last waypoint before the target by (4.10) or (4.11) and the waypoint named circle waypoints, such that the vehicle can follow the circle trajectory smoother. However if the vehicle had been inside of the activation circle it would not needed to go to the obstacle. Thus the method only generates the circle waypoints and the last waypoint before the target using the vehicles position as the first waypoint.

Dubins method: pseudo code

```
WP is a matrix containing all the waypoints generated by the Dubins method
if Threat == false Or NoNewObstacleInRage == true then
    WP = [vehicle position , target]
else if Threat == true And DynamicObstacle == true then
    if  Inside activationCircle == true  then
        $wp_{k+1}$ = FindWaypointFromObstacleToTarget
        $wp_{circle}$ = FindWaypointOnObstacleCircle
        WP= [vehicle position , $wp_{circle}$ , $wp_{k+1}$ , target]
    else
        $wp_{k+1}$ = FindWaypointFromVehicleToObstacle
        $wp_{k+2}$ = FindWaypointFromObstacleToTarget
        $wp_{circle}$ = FindWaypointOnObstacleCircle
        WP= [vehicle position ,$wp_{k+1}$ , $wp_{circle}$ , $wp_{k+2}$ , target]
    end if
else if Threat == true And DynamicObstacle == false then
    if NotLastObstacle == true then
        if  Inside activationCircle == true  then
            $wp_{k+1}$ = FindWaypointFromObstacleToTarget
            $wp_{circle}$ = FindWaypointOnObstacleCircle
            WP= [vehicle position , $wp_{circle}$ , $wp_{k+1}$ , target]
        else
            $wp_{k+1}$ = FindWaypointFromVehicleToObstacle
            $wp_{k+2}$ = FindWaypointFromObstacleToTarget
            $wp_{circle}$ = FindWaypointOnObstacleCircle
            WP= [vehicle position ,$wp_{k+1}$ , $wp_{circle}$ , $wp_{k+2}$ , target]
        end if
    end if
else
    WP = Last WP
end if
```

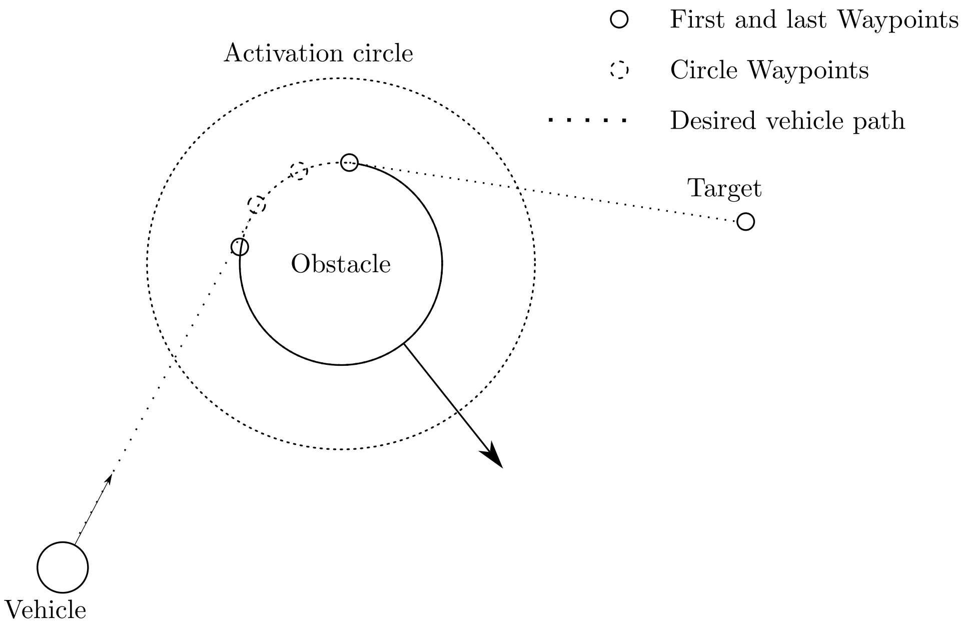


Figure 5.11: Dubins method: The figure shows how the Dubins method generating waypoints when the vehicle consider an obstacle to be a threat and is outside of the activation circle

5.5 Steering law

The steering law is the second part of the guidance system. This will use the output from the collision avoidance methods to generate a desired heading that is sent to the controller.

5.5.1 Steering law for the NSB controller

The NSB controller gives a desired velocity vector as output, so the steering law will have to find the heading $\chi_{d,NSB}$ of this vector

$$\chi_{d,NSB} = atan2(\boldsymbol{\nu}_{d,y}, \boldsymbol{\nu}_{d,x}) \tag{5.7}$$

where $\boldsymbol{\nu}_{d,x}, \boldsymbol{\nu}_{d,y}$ is the x and y component in the NED frame of the desired velocity vector $\boldsymbol{\nu}_d$.

5.5.2 Steering law for the Dubins method

The Dubins method for collision avoidance generates waypoints, thus the steering law should manage to follow the path generated by these waypoints. Therefore the LOS (Line-of-Sight) steering law presented in [Fossen, 2011] is chosen.

$$\chi_{d,Dubins}(e) = \chi_p + \chi_r(e) \tag{5.8}$$

where

$$\chi_p := a_k \tag{5.9}$$
$$a_k = atan2(WP_{k+1,y} - WP_{k,y}, WP_{k+1,x} - WP_{k,x}) \tag{5.10}$$

is the path-tangential angle and

$$\chi_r := atan(\frac{-e}{\Delta} - K_i \int_0^t e \; d\tau) \tag{5.11}$$

where K_i is the integral and e is the cross track error

$$e = -(\eta_x - WP_{k,x}) \sin(a_k) + (\eta_y - WP_{k,y}) \cos(a_k) \tag{5.12}$$

and from Figure 5.12 it can be seen that Pythagoras can be applied

$$e(t)^2 + \Delta(t)^2 = R^2 \tag{5.13}$$

where $\Delta > 0$ is the lookahead distance ahead of the direct projection of $\eta(t)$ on to the path. R is a tuning parameter, which needs to be tuned. By using the control law

$$K_p = \frac{1}{\Delta} \tag{5.14}$$

where

$$\Delta = \sqrt{R^2 - e^2} \tag{5.15}$$

then χ_r becomes

$$\chi_r = atan(-eK_p - K_i \int_0^t e \; d\tau) \tag{5.16}$$

In this thesis the tuning parameter K_i and R are set as

$$K_i = 0.00001 \; , \; R = 40 \tag{5.17}$$

There have also been set a saturation limit on the integral effect, such that if $\int e \; d\tau$ becomes more that 200 it goes to zero in order to avoid integral wind-up.

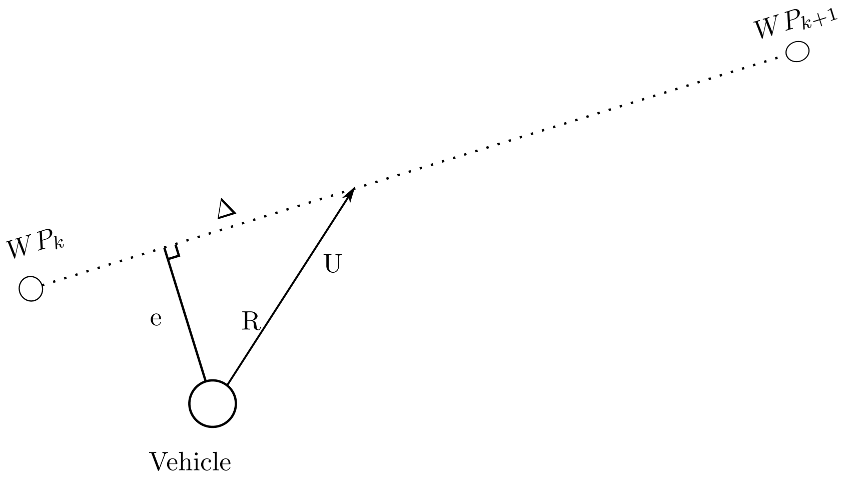


Figure 5.12: LOS steering law: The figure shows the vehicle with the LOS steering law when the vehicle steers on the line between the two waypoints.

5.6 Stability and convergence of the methods

The stability of the methods is very important to consider when designing a collision avoidance system. Considering stability in a collision avoidance approach means that the methods will converge to the target without colliding with any obstacles. The distinction between a locally stable method and a globally stable method, is that the local cannot guarantee to converge to the target for every case, only if some given assumption valid. The global methods will converge for any given cases.

5.6.1 The NSB controller

The stability properties for the NSB controller are studied in Chapter 3, Section 3.4. From this stability analysis and the discussion from the test system, which was used for testing the NSB algorithm it was found several equilibrium points, which caused the method to fail. The problems was divided up into four sub problem, which is illustrated in Figure 3.3 - 3.6;

The first one where the vehicle got stuck on the obstacle circle is solved by the collision detection system. Considering that the equilibrium point on the obstacle circle lies on the line from the obstacle center to the target, such that the obstacle will not be a threat in this point. Thus the collision detection system will turn off the collision avoidance and the vehicle will continue towards the target before reaching the equilibrium point for all obstacles.

The second problem occurs only if the vehicle is heading directly towards the obstacle center and if the vehicle position, center obstacle and the target are on a straight line. Then the NSB controller fails, but this is very unlikely to occur since the obstacle center, vehicle and the target need to be perfectly aligned. For dynamic obstacles the *traverse obstacle* task will solve this problem by considering that the vehicle converge to a point $\boldsymbol{t}_i$ on the obstacle circle for then heading to the target. Thus the vehicle will never have a heading, which is directly towards the obstacle center. However for static obstacles this situation can occur. Two different methods have been considered for solving this problem for static obstacles:

- first, to add a $\epsilon = [0.1,\ 0.1]$ to the targets position if the situation occurred. This would have solved the problem since the obstacle center, vehicle and the target would not by perfectly aligned, but this would have resulted in a velocity vector that would have headed very much towards the obstacle and not around it.
- The second method for solving this problem and also added to the NSB controller was to detect if the angle to the vector between the vehicle and the target is equal the angle to the vector between the obstacle center and the target. If true, the heading is directly towards the obstacle center and then use the *traverse obstacle* task with a default rotation direction as clockwise and converge towards the point $\mathbf{t}_i$ on the obstacle circle.

The third problem, where the vehicle was dragged unnecessary towards a non-threatening obstacle does not have a direct impact on the stability of the system and the collision detection system has solved this problem for all obstacles. Considering that the collision avoidance task will not be active unless there is an actual collision threat.

The last problem that was discussed in Section 3.4 when the NSB controller steered the vehicle into another overlapping obstacle and stopped is an example on what happen if the vehicle takes a path around an obstacle, which does not lead to the target. This problem was discussed in my project report, fall 2013 and solved by merging all overlapping obstacle together and making a large obstacle around all of them. In this thesis it will be assumed that all paths around all obstacles leads to the target (no dead ends) such that merging obstacles will not be considered.

The conclusion form this analysis is that the method is not globally stable since there are cases, which make the NSB controller not converge to the target. Also (3.13) include a singularity when $\eta = p_o$, which makes it not globally stable. However under the following assumption

- All path around the obstacles leads to the target. No dead ends.
- The collision detection system prevents the vehicle to reach the equilibrium points on the obstacle circle
- $\eta \neq p_o$, which prevents the singularity in (3.13)

the method is locally stable.

5.6.2 Dubins method

The Dubins method for collision avoidance presented in Section 5.4 is a waypoint generating algorithm, thus the stability analysis becomes a two step process.

- Step one: Stability of the waypoints generated. Does the method manage to generate feasible waypoints to the target for all cases ?
- Step two: Stability of the path following algorithm. Does the method mange to follow the path generated ?

Considering that the vehicle does not know where all obstacles are before the vehicle is in range of the obstacles, an argument can be made that if the method decides to take a path around one obstacle, which leads to a dead end the method does not have the ability to turn around and tray another route. Also if the target is inside the obstacle circle the method would fail since the vehicle would go to the target at the end of the waypoint list, thus the method is not globally stable. However under the assumption, that there are no dead ends and the target is outside of any obstacles, the method will generate a path to the target, thus the method is locally stable.

The LOS steering law is stable and will converge to the path if the cross track error e (5.12) goes to zero.

$$\lim_{t \to \infty} e(t) = 0 \tag{5.18}$$

In [Breivik and Fossen, 2004] it is proven that the cross track error converge to zero if the speed is non-zero for the LOS steering law by a Lyapunov analysis. Thus the steering law will converge to the path generated by the Dubins method.

5.7 Summary

In this chapter it has been shown how the guidances system for the vehicle simulator was implemented. The guidance system consists of the collision detection, collision avoidance and the steering law. The collision detection method detects if obstacles are on collision course with the vehicle, by estimating the trajectory of the vehicle towards the target and the trajectory of the obstacle and assuming that they will continue with the constant speed, and testing if the distance between these tow trajectories are under a threshold value. If a threat is detected and the obstacle is dynamic the system determines the passing direction, by rotating the velocity vector to the obstacle into the BODY frame of the vehicle and checking, which quadrant it is in. For static obstacles the shortest path around the obstacles is chosen. Both the collision avoidance methods presented use the same collision detection method. Considering that the NSB controller generates a desired velocity vector and the Dubins method generates waypoints, two steering laws where implemented: One for following the heading to the desired velocity vector and one LOS steering law for waypoint following. The stability and convergence of the two collision avoidance methods has been discussed and concluded that they are not globally asymptotically stable, only locally stable.

Chapter 6

Vehicle simulator and control design

Figure 6.1 shows a model of the vehicle simulator and this chapter will present the vehicle model and the controller block as seen in this figure. The vehicle simulator is developed under the following assumptions:

1. There are no environmental forces that act on the vehicle, so the environmental force acting on the vehicle is not considered.
2. All signals to the vehicle and controller are assumed to be perfect.

This chapter is organized as following: first the vehicle model will be presented with a analysis of the model, then the surge and heading controllers.

6.1 Vehicle model

As shown in [Fossen, 2011], the 3 DOF (Degree of Freedom) non-linear equation of motion for a vehicle can be written as

$$\dot{\eta} = \mathbf{R}(\eta)\nu \tag{6.1}$$

$$\mathbf{M}\dot{\nu} + \mathbf{C}(\nu)\nu + \mathbf{D}(\nu)\nu + g(\eta) = \tau \tag{6.2}$$

Where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix from BODY to NED frame and $\eta = [x, y, \psi]^T$ and $\nu = [u, v, r]^T$ are the position and velocity vectors, which is given in NED and BODY frame respectively. $\mathbf{M}$ is the mass matrix, which consists of the mass of the vehicle and the added mass. $\mathbf{C}(\nu)$ is the Coriolis and added Coriolis

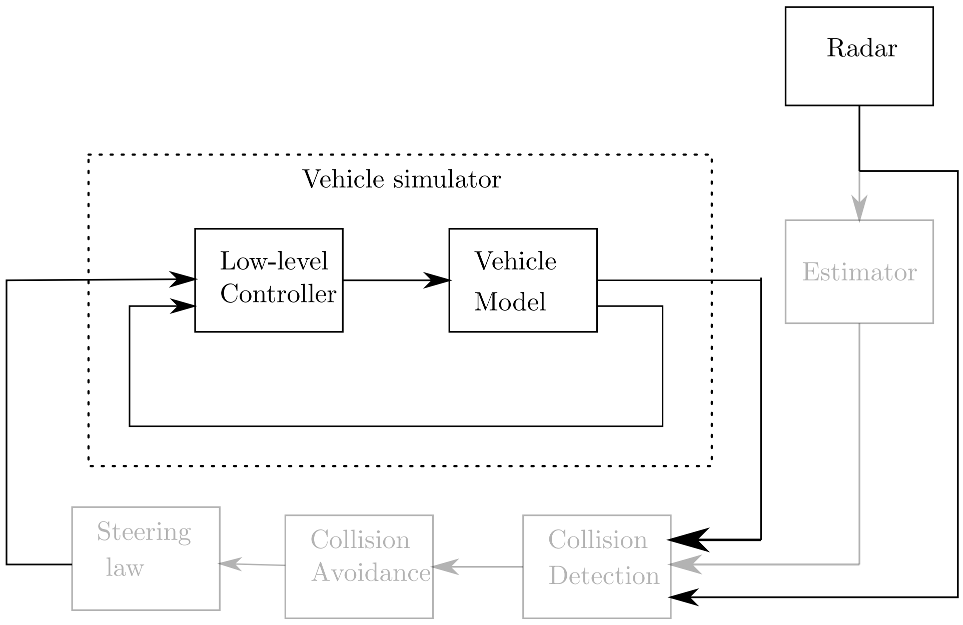


Figure 6.1: The model of the collision avoidance system, where the focus is on the vehicle model and the low-level surge and heading controller

matrix. $\mathbf{D}(\nu)$ is the non-linear dampening term and $g(\eta)$ is the linear dampening term. τ is all the external forces that acts on the vehicle. Since the main focus in this thesis is on collision avoidance, the vehicle model used is very simplified with the following assumptions:

1. Only consider a 3 DOF model (surge, sway and yaw).
2. The vehicle is moving with constant or at least slowly varying speed, such that $U = \sqrt{u^2 + v^2} = u$, since $u \gg v$
3. There are no coupling effects in surge, sway and yaw.
4. Considering only low velocity, such that all the linear terms dominate the non-linear terms and the vehicle behaves as a first order system.

Given these assumptions one can transform the no-linear system (6.1) and (6.2) into a decoupled first order system with 3 DOF. The velocity model in sway is disregarded, since it is assumed $u \gg v$. The velocity in surge can be expressed as

$$T_{surge}\dot{u} + u = \tau_{surge} \tag{6.3}$$

where u is the surge velocity and T_{surge} is a time-constant in surge for the vehicle. τ_{surge} is the force that acts on the surge motion.

The first order Nomoto model from [Fossen, 2011, eq. (7.49)] is often used in linear autopilot for heading control. In this thesis it has been used to model the turning rate in yaw.

$$T_{yaw}\dot{r} + r = b\delta \tag{6.4}$$

Where T_{yaw} is a time-constant in yaw and r is the turning rate in yaw, $b\delta$ is the force that acts on the yaw motion. In this thesis it is considered a single rudder and under the assumption that the vehicle is moving with constant speed, b becomes constant, δ is the rudder angle. In reality $b\delta$ is speed depending and no-linear, but this is not considered in this thesis. There should also be included some limitation on the rudder and in this thesis the limitation has been set as $\delta = \pm 35\,\text{deg}$.

Using the rotational matrix $\mathbf{R}$ for 3 DOF.

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.5}$$

and by remembering that the sway velocity is neglected $\nu = [u, 0, r]$, then from (6.1) the position velocity in NED becomes

$$\begin{aligned} \dot{x} &= ucos(\psi) \\ \dot{y} &= usin(\psi) \\ \dot{\psi} &= r \end{aligned} \tag{6.6}$$

where u is the vehicle speed in BODY frame.

6.1.1 Vehicle parameters used in this thesis

Based on the systems (6.3) and (6.4) it is chosen to test the collision avoidance methods using the model parameter according to Table 6.1. The time-constant on a real vehicle can vary from a couple of seconds to a couple of minutes, depending on the vehicle. It can be seen in the table that the parameters used in this thesis gives a relatively fast responding vehicle.

Table 6.1: Parameter for the vehicle model used in the simulations in chapter 7

Parameters	Value
T_{surge}	5
T_{yaw}	4
b_{yaw}	1
k_{surge}	3
$\delta_{max}, \delta_{min}$	± 35 deg
τ_{max}, τ_{min}	± 10

6.1.2 Max turning radius

When designing a collision avoidance system or performing a path planning, it is essential to know how fast the vehicle can turn. It makes no sense to avoid obstacles, which the dynamic of the vehicle can not achieve. Thus it is better to increase the size of the safety radius around the obstacles, such that the path will be feasible. In [Fossen, 2011] one can find a test to determine the steady turning radius for a vehicle. This will give an indication on how fast the vehicle can turn. The test is performed by turning the vehicle over at maximum speed with maximum rudder angle. The definition on turning radius R can be found in [Fossen, 2011, eq. (12.47)]

$$R := \frac{U}{r} \tag{6.7}$$

where U is the vehicles speed and r is the turning rate, which can be found form the Nomoto model and have the explicit solution

$$r(t) = exp(-\frac{t}{T_{yaw}})r(0) + (1 - exp(-\frac{t}{T_{yaw}})b\delta_{max}) \tag{6.8}$$

The steady turning radius is when $t \to \infty$ and r becomes $b\delta$. From (6.8) it can be seen that r converge faster for smaller T_{yaw}, but the final value of r becomes the same. The steady turning radius becomes ($t = \infty$)

$$R(\infty) = \frac{U_{max}}{b\delta} \tag{6.9}$$

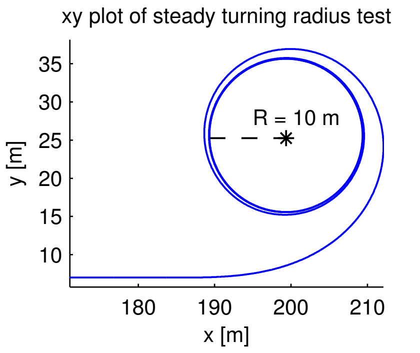
This test has been performed for the vehicle model in (6.3) and (6.4). The results are shown in Figure 6.2 for the vehicle with the model parameters in Table 6.1. The figure shows the path of the vehicle when it achieves the steady turning radius, which is 10 meters. It can be seen in Figure 6.2(b) that the vehicle starts turning after 30 seconds and have achieved the steady turning radius after 40 seconds when the turning rate has reached its maximum value. Table 6.2 summarizes the most important results from the test, the steady turning radius, transfer, advance and tactical diameter. The transfer and advance is the length from when the vehicle starts turning in x and y direction respectively, to the vehicles position (x and y) when the vehicle have reached 90 degree in heading the first time. When considering obstacle avoidance and deciding a minimum safety circle for the obstacle it is unlikely that the vehicle will achieve a steady turning radius, considering that it require more than 360 degree turn. Also the vehicle will have to react much faster. However a $0-90$ degrees turn is more likely and using the maximum of Transfer at 90 degrees heading or Advance at 90 degrees heading from Table 6.2 as minimums safety radius around the obstacles will ensure that the vehicle manage all the turns. In this thesis the Advance at 90 degrees heading is 27 meter and the Transfer at 90 degrees heading 17, thus the minimum safety circle around the obstacle will be 27.

Table 6.2: Results from the turning radius test

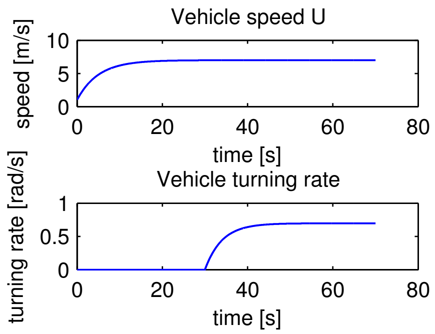
Steady turning radius	10
Transfer at 90 degrees heading	17.24 meter
Advance at 90 degrees heading	27.102 meter
tactical Diameter at 180 degrees	29.93 meter

6.2 Surge and heading controllers

In this thesis it is desired that the vehicle should keep a constant speed and be steered by a heading controller. To achieve this, two decoupled controllers, a surge and a heading controller have been implemented. The surge controller should keep



(a) xy plot of the vehicles position in the steady turning radius test



(b) The speed U and the turning rate r to the vehicle in the steady turning radius test

Figure 6.2: Results from the steady turning radius test performed on the vehicle

the vehicle at a contestant speed, therefore a PI controller has been chosen.

$$\tau = K_{p,surge} u_d - \int_0^t K_{i,surge} \tilde{u} d\tau \tag{6.10}$$

Where $\tilde{u} := u - u_d$ and u_d is the desired velocity, $K_{p,surge}$ and $K_{i,surge}$ are the P and I-gains to the controller. Considering the second order mass damper system

$$m\ddot{x} + d\dot{x} + kx = \tau \tag{6.11}$$

then comparing the mass damper equation with (6.3)

$$m = 0, \ d = T_u, \ k = 1$$

Then in order to find the P-gain and I-gain [Fossen, 2011, Table 12.4] has been used.

$$\omega_{n,1} = \frac{1}{\sqrt{1 - 2\xi_{surge}^4 + \sqrt{4\xi_{surge}^4 - 4\xi_{surge}^2 + 2}}} \omega_{b,surge} \tag{6.12}$$

$$K_{p,surge} = m\omega_{n,1}^2 \tag{6.13}$$

$$K_{i,surge} = \frac{\omega_{n,1}}{10} K_{p,surge} \tag{6.14}$$

Where $\omega_{b,surge}$ is the desired bandwidth and ξ_{surge} is the relative damping ratio $\xi_{surge} > 0$. Both parameters are tuning parameters and the values used are listed in Table 6.3.

The heading controller should both be able to keep a constant heading and manage to follow a fast change in heading. Therefore a PID controller has been chosen.

$$\delta = K_{d,yaw}\dot{\psi} - K_{p,yaw}\tilde{\psi} - \int_0^t K_{i,yaw}\tilde{\psi} d\delta \tag{6.15}$$

Where $K_{p,yaw}$, $K_{d,yaw}$ and $K_{i,yaw}$ are the P, D, I-gains to the heading controller. $\tilde{\psi} := \psi - \psi_d$, where ψ_d is the desired heading. Considering the Nomto model for yaw motion, (6.4) and inserting $\dot{\psi} = r$.

$$T_{yaw}\ddot{\psi} + \dot{\psi} = b\delta \tag{6.16}$$

and compare it with the mass damper system (6.11)

$$m = \frac{T_{yaw}}{b},\ d = \frac{1}{b},\ k = 0$$

and finding the P-gain I-gain D-gain as in [Fossen, 2011, Table 12.4]

$$\begin{aligned}
\omega_{n,2} &= \frac{1}{\sqrt{1 - 2\xi_{yaw}^4 + \sqrt{4\xi_{yaw}^4 - 4\xi_{yaw}^2 + 2}}}\omega_{b,yaw} && (6.17)\\
K_{p,yaw} &= m\omega_{n,2}^2 && (6.18)\\
K_{d,yaw} &= 2\xi_{yaw}\omega_{n,2}m - d && (6.19)\\
K_{i,yaw} &= \frac{\omega_{n,2}}{10}K_{p,yaw} && (6.20)
\end{aligned}$$

6.2.1 Tuning parameters

The tuning parameters used for the controllers are given in Table 6.3, where each controller have been tuned to give a fast response.

Table 6.3: Tuning parameter used in simulation studies for a vehicle with fast dynamic, according Table 6.1

Controller	Parameter	Value
Surge controller	$\omega_{b,surge}$	0.8
Surge controller	ξ_{surge}	1.1
Heading controller	$\omega_{b,yaw}$	0.7
Heading controller	ξ_{yaw}	0.9

6.3 static and dynamic obstacles

Obstacles can be divided into two main categories, static or dynamic obstacle depending if they are moving or not. The simulations in Chapter 7 all obstacles are simulated as particles with a safety radius that the vehicle should stay out of.

The obstacles are tracked in the radar as shown in Figure 6.1 and it is assumed that the collision avoidance system receive the position and velocity from the radar perfectly filtered, thus the signal is sent direly to the collision detection block (bypass the estimation block) as illustrated in Figure 6.1.

$$\mathbf{O} = \begin{bmatrix} x_{o1}^n & y_{o1}^n & d_1 & bool_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{oi}^n & y_{oi}^n & d_i & bool_i \end{bmatrix}, \ \mathbf{V} = \begin{bmatrix} \dot{x}_{o1}^n & \dot{y}_{o1}^n \\ \vdots & \vdots \\ \dot{x}_{oi}^n & \dot{y}_{oi}^n \end{bmatrix} \tag{6.21}$$

$\mathbf{O} \in \mathbb{R}^{i\times4}$ and $\mathbf{V} \in \mathbb{R}^{i\times2}$ are the position and velocity matrix for the obstacles, where the x_{oi}^n and y_{oi}^n are the x and y position of the obstacles in the NED frame and d_i is the safety radius, which the vehicle should pass outside of. The last bool value in $\mathbf{O}$ is a dummy value, that is true if the vehicle is close enough to see the obstacles and false otherwise. The $\dot{x}_{oi}^n$ and $\dot{x}_{oi}^n$ are the velocity to the obstacles in the NED frame.

Chapter 7

Simulation: Collision avoidance using the vehicle model

This chapter will present four case studies of the NSB controller and the Dubins method for collision avoidance using the vehicle simulator described in Chapter 6. The following cases will be studied:

- **Case one**: Multiple static obstacles.
- **Case two**: Dynamic obstacles approaching from the left and right.
- **Case three**: Dynamic obstacle approaching heads-on.
- **Case four**: Dynamic obstacle approaching from behind.

The first case will test the collision avoidance methods on static obstacles. The next three cases will consider dynamic obstacles approaching the vehicle from different directions creating all the situations that was discussed in the collision avoidance strategies Section 5.1. This will test the collision avoidance strategies and if the collision avoidance methods manage to follow the correct direction around the obstacles. The two collision avoidance methods will be compared with respect to the path of the vehicle and the average computational time of the methods during the simulations. This chapter is organized as following, first each case will be presented with the simulation results, than a discussion and the conclusion of the results.

7.1 Case one: Multiple static obstacles

The first case will be in an environment with four static obstacles where two of them are placed in the direct path of the vehicle, such that the vehicle will have to perform obstacle avoidance and ignore the two others. The simulation will be run with the vehicle simulator described in Chapter 6, with the NSB controller and the Dubins method. The path of the vehicle is plotted in the xy-plane with obstacles position and radius according to Table 7.1. The start and target positions are $\boldsymbol{\eta}_{start} = [1, 1, 0]$ and $\boldsymbol{\eta}_{target} = [600, 80, -]$. The goals of these simulations are to test:

- If the vehicle manage to avoid all obstacles and safely reach the target.
- Compare the path of the vehicle with the NSB controller against the Dubins method for collision avoidance.
- Compare the average computational time of the NBS controller against the Dubins method.

Table 7.2 shows the average runtime for both methods, the steering law and the simulation time before the vehicle reached the target. The internal dynamic for the vehicle in these simulations are given Figure A.1(NSB controller) and Figure A.2 (Dubins) in appendix A.

Table 7.1: Obstacle list from case one

x-position	y-position	radius
140	30	45
280	-18	35
420	72	90
200	150	40

Table 7.2: Runtime results from case one: Multiple static obstacles

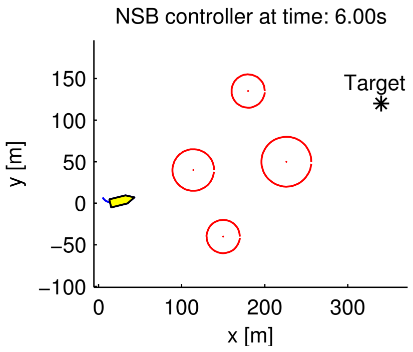
Methods	Collision avoidance	Steering law	Simulation time
NSB controller	$0.724 \times 10^{-5}\ s$	$0.132 \times 10^{-5}\ s$	59.2 s
Dubins	$0.298 \times 10^{-5}\ s$	$0.298 \times 10^{-5}\ s$	56.7 s

Simulation results with the NSB controller

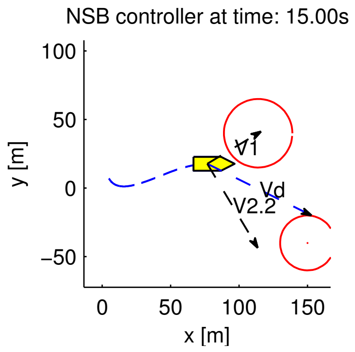
Figure 7.1 shows the path of the vehicle in the xy-plane with the NSB controller at 6, 15, $30.2, 37$ and 59.3 seconds in simulation time for static obstacles. The first figure(a) shows all the obstacles between the vehicles start position and the target. Figure 7.1(b) show the vehicle passing the first obstacle anticlockwise. The arrows in Figure 7.1(b)-(c) $V1, V2.2$ and V_d represent the velocities from task 1, 2.2 and the sum of $V1$ and $V2.2$ projected into the null space of task 1. In (d) it can be seen that the vehicle has passed the second obstacle clockwise and the threat is avoided, thus V_d is pointing towards the target. In Figure 7.1(e) the vehicle has reached the target without colliding with any of the obstacles and it shows the complete path of the vehicle.

Simulation results with Dubins method for collision avoidance

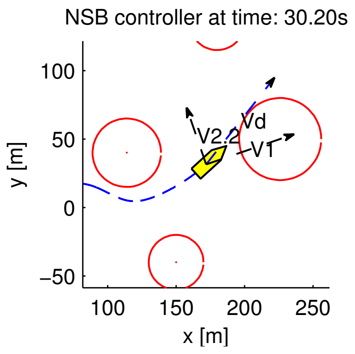
Figure 7.2 shows the simulation of case one multiple static obstacles with the Dubins method for collision avoidance at 5.9, 14.0, 20, 29, 35 and 56.7 seconds in simulation time. The first figure(a) shows all the obstacles between the vehicles start position and the target. Figure(b)-(c) shows the vehicle passing the first obstacle anticlockwise and in (d)-(e) the vehicle pass the second obstacle clockwise. The black stars in Figure 7.2(b) and (d) are the next waypoints that the vehicle is aiming towards. The complete path can be seen in Figure 7.2(f) when the vehicle has reached the target.



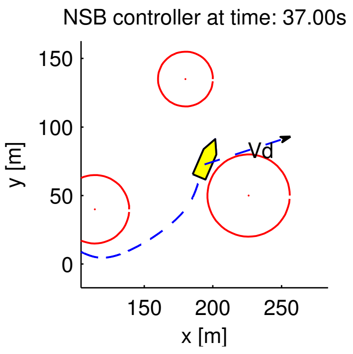
(a) The simulation at time 6 seconds, the vehicle has not detected the obstacle and is heading towards the target



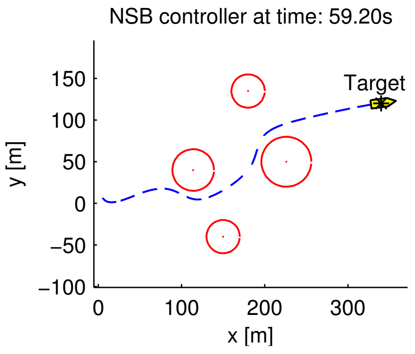
(b) The simulation at time 15 seconds, the NSB controller has activated tasks 1 and 2.2 and is passing the obstacle anticlockwise



(c) The simulation at time 30.2 seconds, the NSB controller has activated tasks 1 and 2.2 and is passing the obstacle clockwise

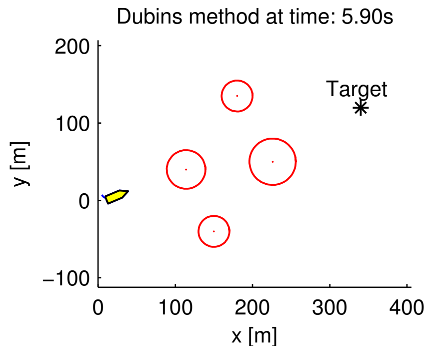


(d) The simulation at time 37 seconds, the collision detection method has deactivated the collision avoidance and the vehicle is heading towards the target

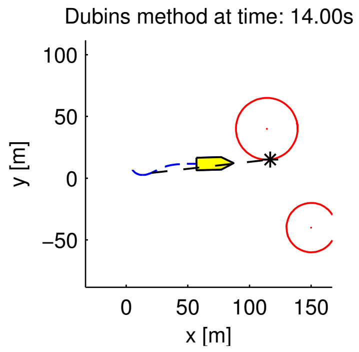


(e) The simulation at time 59.2 seconds, the vehicle has reached the target

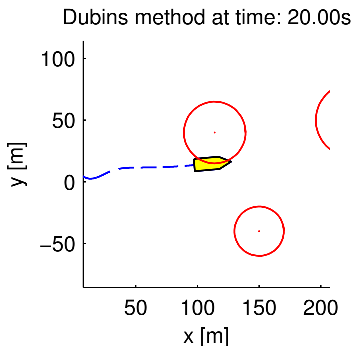
Figure 7.1: Simulation case one: static obstacle with the NSB controller, plotted in the xy-plane



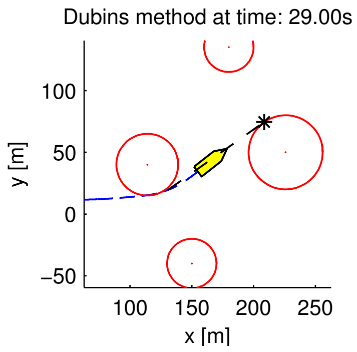
(a) The simulation at time 5.9 seconds, the vehicle has not detected the obstacle and is heading towards the target



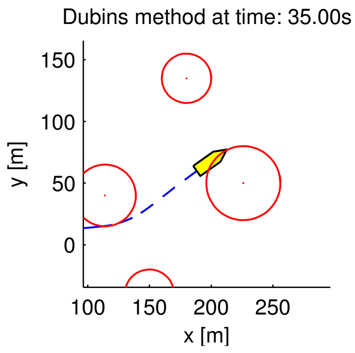
(b) The simulation at time 14 seconds, the vehicle has detected the obstacle and is passing the obstacle anticlockwise



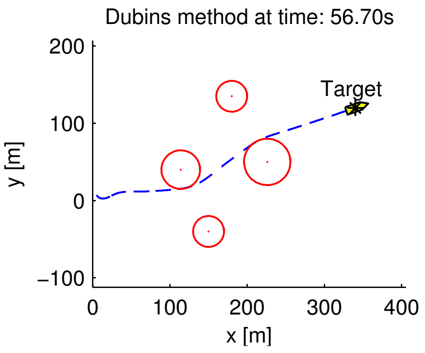
(c) The simulation at time 20 seconds, the vehicle is passing the first obstacle



(d) The simulation at time 29 seconds, the vehicle is heading towards the seconds obstacle



(e) The simulation at time 35 seconds, the vehicle has reached the first waypoint on the obstacle circle



(f) The simulation at time 56.7 seconds, the vehicle has reached the target

Figure 7.2: Simulation case one: static obstacle with the Dubins method for collision avoidance, plotted in the xy-plane

7.2 Case two: Dynamic obstacles approaching from left and right

Case two: In these simulations there will be one obstacle approaching the vehicle from the left and one obstacle approaching the vehicle from the right. Both of them have been constructed such that the vehicle will have to perform collision avoidance in order to safely reach the target. The starting position, safety radius, heading and speed for the obstacles are given in Table 7.3. Table 7.4 shows the average runtime for both methods, the steering law and the simulation time before the vehicle reach the target. The start and target positions are $\boldsymbol{\eta}_{start} = [5, 7, 1]$ and $\boldsymbol{\eta}_{target} = [180, 60, -]$. The goals of these simulations are to

- Test the collision avoidance methods with dynamic obstacles approaching the vehicle from the left and right.
- Test if the collision avoidance systems manage to follow the correct path around the obstacles.
- Compare the average runtime of the collision avoidance methods and the internal dynamic of the vehicle with the two collision avoidance methods.

Table 7.3: Obstacle list from case two

Starting position [x , y]	radius	speed	heading
[200 , 200]	30	2.55	−168.7 degrees
[−30 , 140]	20	5	126 degrees

Table 7.4: Runtime results form case two

Methods	Collision avoidance	Steering law	Simulation time
NSB controller	0.575×10^{-5} s	0.24×10^{-5} s	52 s
Dubins	0.468×10^{-5} s	0.26×10^{-5} s	49.9 s

Simulation results with the NSB controller

Figure 7.3 shows the path of the vehicle in the xy-plane for the NSB controller at 8, 15, 27.4 and 52 seconds in simulation time. It can be seen that the vehicle avoided the first obstacle clockwise and the second obstacle anticlockwise. Figure 7.3(b) includes the desired velocity vector V_d and it can be seen that it points towards a point on the obstacle circle such that the vehicle will avoid the obstacle clockwise. Figure 7.3(d) shows the vehicle passing the first obstacle inside of activation circle two with the velocity vectors $V1$ and $V2.1$ for the tasks 1 and 2.1.

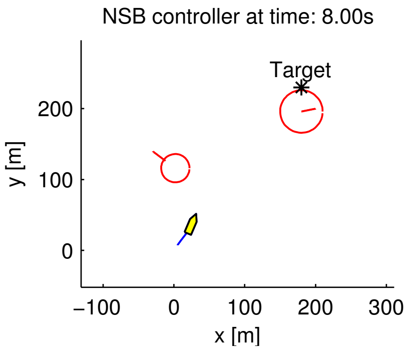
The complete path of the vehicle is shown in Figure 7.3(f) where the blue dotted line is the path of the vehicle and the red dotted lines are the paths of the obstacles.

Figure 7.4 shows the internal dynamic (vehicle speed, turning rate steering command, vehicle heading and the rudder angle) for the vehicle.

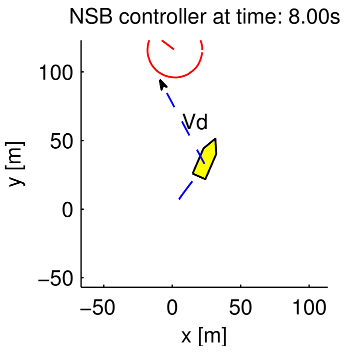
Simulation results with the Dubins method for collision avoidance

Figure 7.5 shows the path of the vehicle in the xy-plane with the Dubins method for collision avoidance at 8, 12, 18, 28, 39 and 49.9 seconds in simulation time. It can be seen that the vehicle avoid the first obstacle clockwise and the second obstacle anticlockwise before reaching the target. The black stars in Figure 7.5 are the next waypoints, which the vehicle aims towards and the target. The blue dotted line is the path of the vehicle and the red dotted lines are the path for the obstacles.

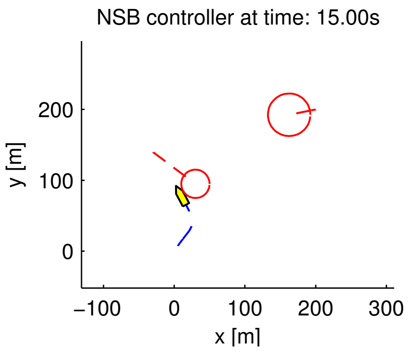
Figure 7.6 shows the internal dynamic (vehicle speed, turning rate steering command, vehicle heading and the rudder angle) for the vehicle.



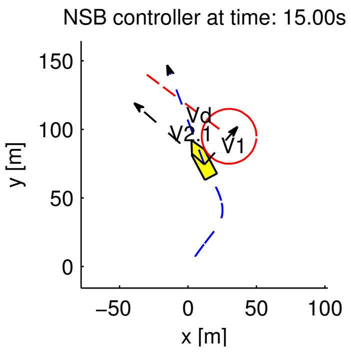
(a) The simulation at time 8 seconds. The vehicle has detected a threat and activated the collision avoidance



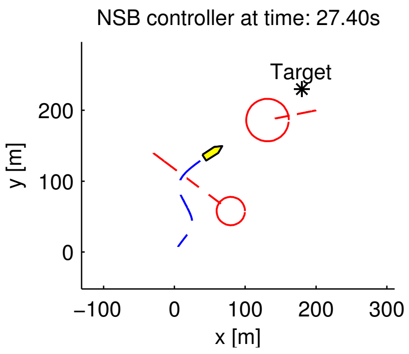
(b) The Figure 7.3(a) zoomed in. The desired velocity ν_d points at a point on the obstacle circle such the vehicle will pass clockwise



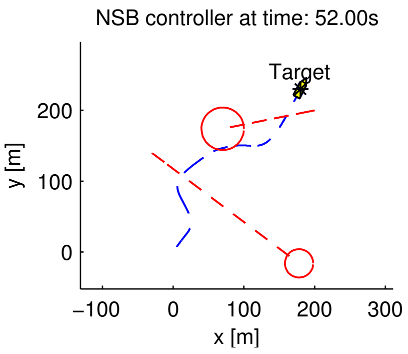
(c) The simulation at time 15 seconds. The vehicle is passing the obstacle clockwise



(d) The simulation at time 15 seconds, both tasks 1 and 2.1 is active and the two velocity vectors from the tasks are shown as V1 and V2.1

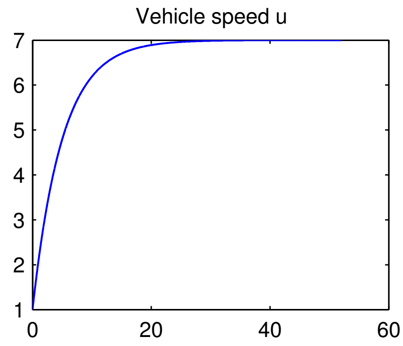


(e) The simulation at time 27.4 seconds. The vehicle has passed the first obstacle and detected the next obstacle

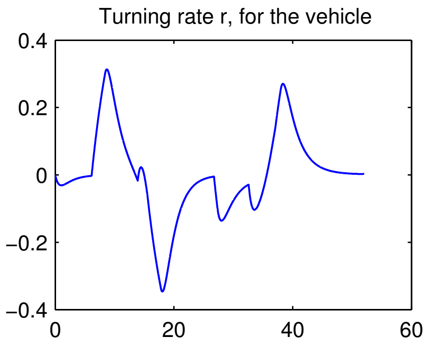


(f) The simulation at time 52 seconds. The vehicle has passed the second anticlockwise and reached the target

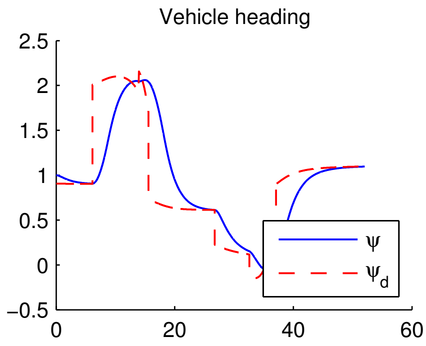
Figure 7.3: Simulation case two with the NSB controller: The vehicle is avoiding two obstacles approaching from the left and right by passing the obstacles clockwise and anticlockwise



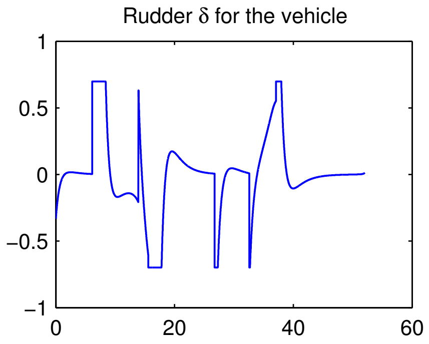
(a) The vehicle speed for the simulation in Figure 7.3



(b) The turning rate for the simulation in Figure 7.3

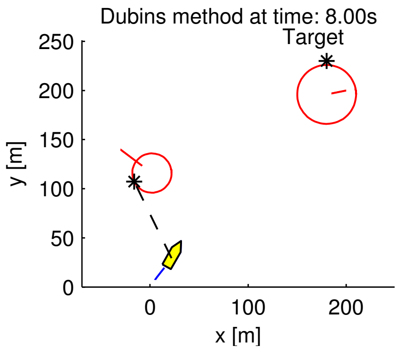


(c) The vehicles heading and the steering command from the NSB controller for the simulation in Figure 7.3

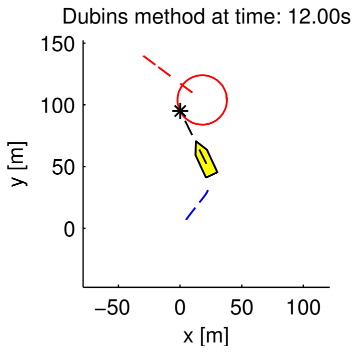


(d) The rudder angle for the vehicle for the simulation in Figure 7.3

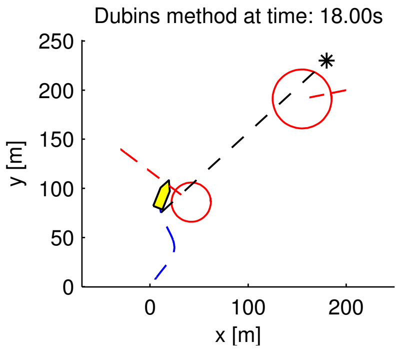
Figure 7.4: The internal dynamic for the vehicle in the simulation shown in Figure 7.3



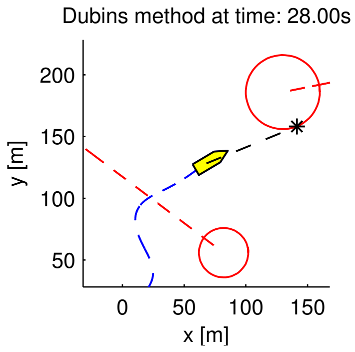
(a) The simulation at time 8 seconds. The vehicle has detected an obstacle and has calculated the tangent line to the obstacle and starts to turn towards it in order to avoid the obstacle.



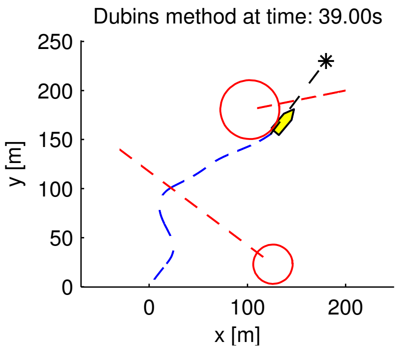
(b) The simulation at time 12 seconds, the vehicle is approaching the obstacle.



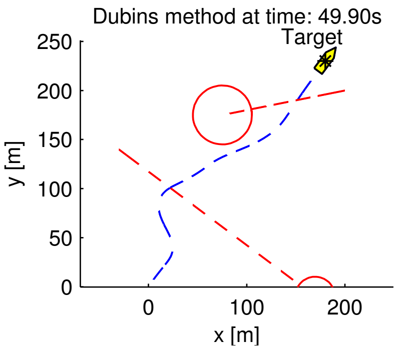
(c) The simulation at time 18 seconds. The vehicle has passed the first obstacle and has not discovered the seconds obstacle.



(d) The simulation at time 28 seconds the vehicle has discovered the second obstacle and is heading for the first waypoint on the obstacle circle.

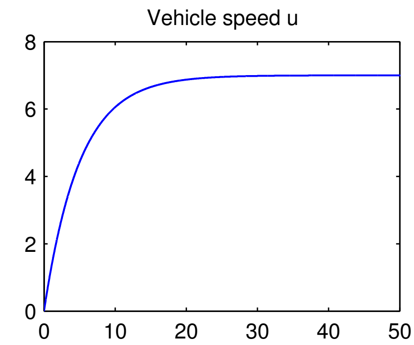


(e) The simulation at time 39 seconds. The vehicle has passed the second obstacle.

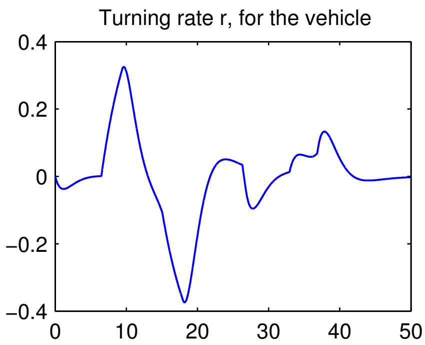


(f) The simulation at time 49.9 seconds. The vehicle has reached the target and the simulation is finished.

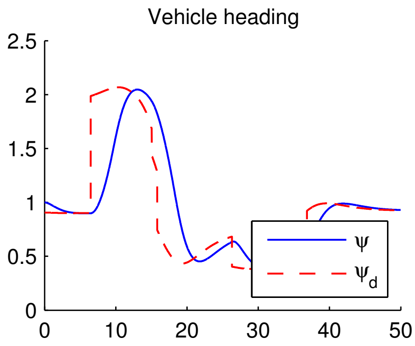
Figure 7.5: Simulation case two with the Dubins method: the vehicle are avoiding the first obstacle clockwise and the second anticlockwise.



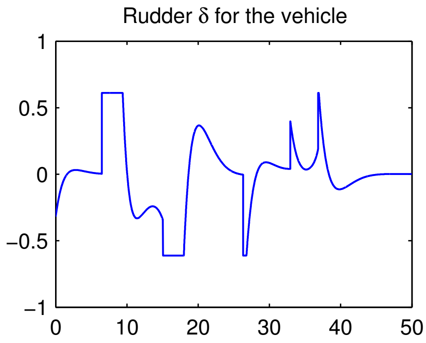
(a) The vehicle speed for the simulation in Figure 7.5



(b) The turning rate for the simulation in Figure 7.5



(c) The vehicles heading and the steering command from the NSB controller for the simulation in Figure 7.3



(d) The rudder angle for the vehicle in simulation Figure 7.5

Figure 7.6: The internal dynamic for the vehicle in the simulation shown in Figure 7.5

7.3 Case three: Dynamic obstacle avoidance in a heads-on situation

Case three will simulate a heads-on situation, where the vehicle and the obstacle move towards each other heads-on. The vehicle will move towards the target and the obstacle will start at the target and move towards the vehicles start position. The start position, safety radius, heading and speed for the obstacle are given in table 7.5. Table 7.6 shows the average runtime for both methods, the steering law and the simulation time before the vehicle reached the target. The start and target positions are $\boldsymbol{\eta}_{start} = [5, 7, 1]$ and $\boldsymbol{\eta}_{target} = [150, 200, -]$. The goals of these simulations are:

- Test the collision avoidance methods in a heads-on situation and to test if the vehicle chooses the right direction around the obstacle.
- Compare the average runtime of the collision avoidance methods and the path of the vehicle.

The internal dynamic for the vehicle in this simulations are given Figure A.3(NSB controller) and Figure A.4 (Dubins) in appendix A.

Table 7.5: Obstacle list from case three

Starting position [x , y]	radius	speed	heading
[150 , 200]	20	7.8	-130

Table 7.6: Runtime results form case three

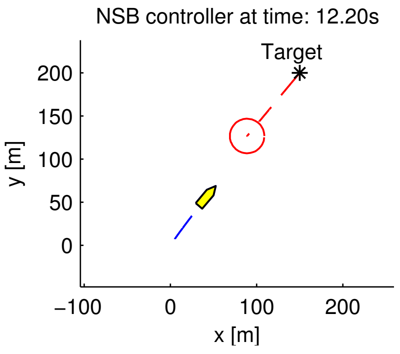
Methods	Collision avoidance	Steering law	Simulations time
NSB controller	$0.526 \times 10^{-5}\ s$	0.263×10^{-5}	39.5 s
Dubins	$0.520 \times 10^{-5}\ s$	0.292×10^{-5}	40 s

Simulation results with the NSB controller

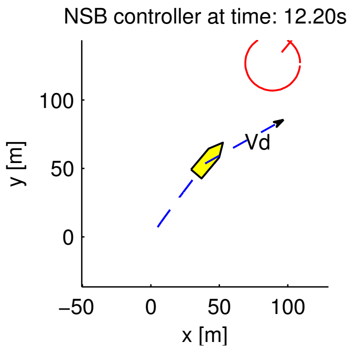
Figure 7.7 shows the path of the vehicle in the xy-plane for the NSB controller at 12.2, 14, 18, 21.7 and 39.6 seconds in simulation time. In (b) the collision avoidance system is activated and it can be seen that v_d points away from the obstacle circle. In (c) both task 1 and 2.1 are active and the three arrows $V1$, $V2.1$ and V_d are the velocity vectors from task one, task 2.1 projected into the null space of task 1 and the desired velocity, which is the sum of $V1$ and $V2.1$ projected into the null-space of task 1. (d)-(e) shows when the threat is avoided and the vehicle starts heading towards the target again. The complete path can be seen in (f) where the blue line is the path of the vehicle and the red is the path of the obstacle with the red safety radius.

Simulation results with the Dubins method for collision avoidance

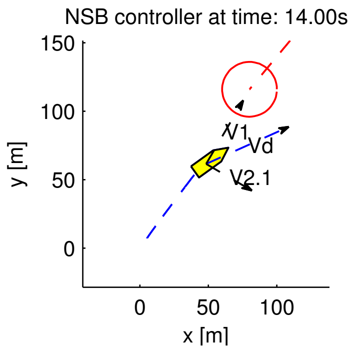
Figure 7.8 shows the path of the vehicle in the xy-plane for the Dubins method at 14, 16, 18.2, 25 and 40.1 seconds in simulation time. The black starts with the black line from the vehicle is the next waypoint, which the vehicle is aiming towards. In (a)-(b) the vehicle aims at the first waypoint outside of the stander obstacle circle and in (c) the vehicle is heading towards the last waypoint outside of the obstacle circle. The threat is avoided in (d) and the vehicle is heading towards the target in (e), the complete path of the vehicle is shown in (f).



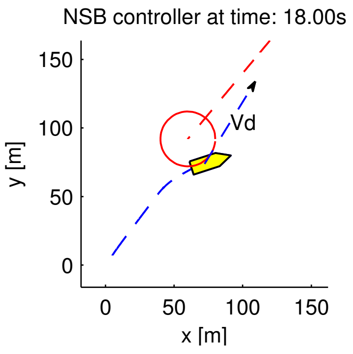
(a) The vehicle position inside of activation circle one at time 12.2 seconds. The collision detection method has detected a collision threat and activates the NSB controller to go anticlockwise with increased safety circle



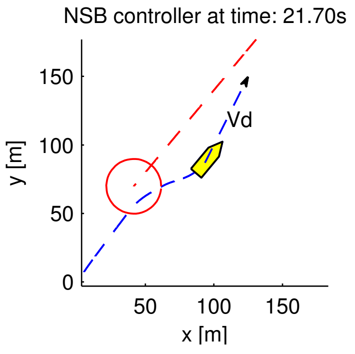
(b) The vehicle position at time 12.2 seconds zoomed in, where v_d points away from the obstacle



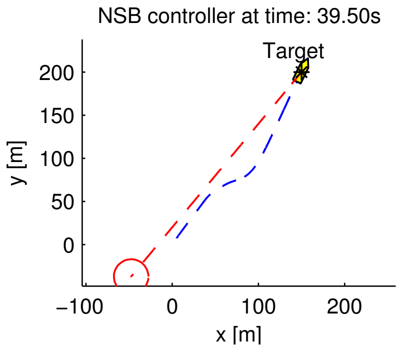
(c) The vehicle position inside of activation circle two at time 14 seconds, It can be seen from the velocity vectors that both task 1 and 2.1 are active



(d) The vehicle position at time 18 seconds zoomed in and the threat is avoided

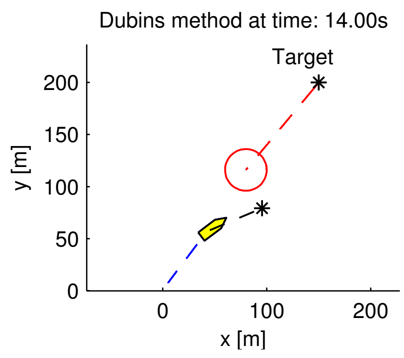


(e) The vehicle position at time 21.7 and the vehicle is heading towards the target

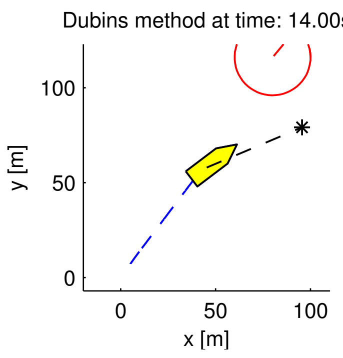


(f) The vehicle position at time 39.5 seconds. The simulation is finished and the vehicle has reached the target

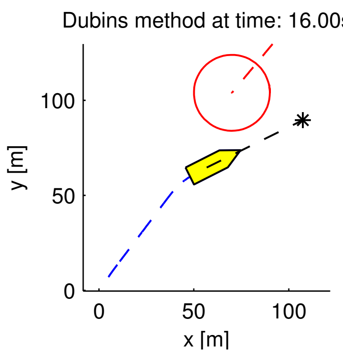
Figure 7.7: Simulation case three with the NSB controller: the vehicle approaching an obstacle heads-on and avoiding it anticlockwise



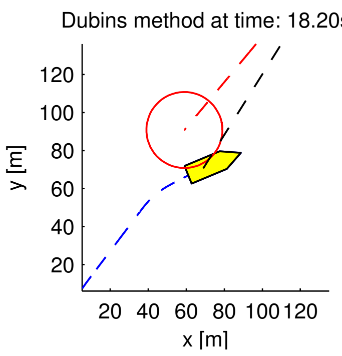
(a) The vehicle position at time 14 seconds. The collision detection has detected a collision threat and activate the collision avoidance method to go anticlockwise with increased safety circle



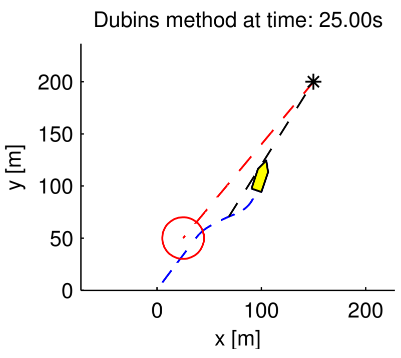
(b) The vehicle position at time 14 seconds zoomed in



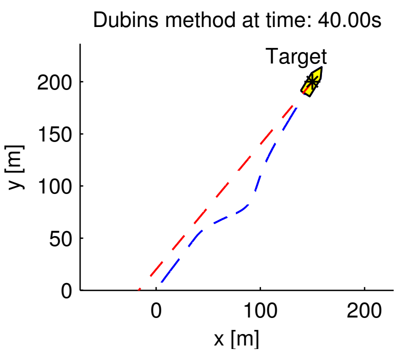
(c) The vehicle position at time 16 seconds. The vehicle heading towards the last waypoint on the obstacle before the target



(d) The vehicle position at time 18.2 seconds. The obstacle in no longer a threat and the vehicle makes a waypoint at its current position and the next at the target



(e) The vehicle position at time 25 seconds. The vehicle is on the path to the target



(f) The vehicle position at time 40 seconds. The simulation is finished and the vehicle has managed to reached the target without colliding with the obstacle

Figure 7.8: Simulation case three with the Dubins method: the vehicle approaching an obstacle heads-on and avoiding it anticlockwise

7.4 Case four: Dynamic obstacle avoidance form behind

Case four will simulate the vehicle approaching an obstacle from behind and pass it with the collision avoidance methods. After 40 seconds into the simulations the obstacle will increase the speed by 4.5 times its initial speed and approach the vehicle from behind, forcing the vehicle to perform obstacle avoidance again before reaching the target. The starting position, safety radius, heading and speed for the obstacles are given in table 7.5. The average runtime for both methods, the steering law and the simulation time before the vehicle reached the target are given in Table 7.8. The start and target positions are $\boldsymbol{\eta}_{start} = [5, 7, 0]$ and $\boldsymbol{\eta}_{target} = [400, 500, -]$. The goals of these simulations are to

- Test the collision avoidance methods when the vehicle approaches the obstacle from behind and when the obstacle approaches the vehicle from behind.
- Compare the average runtime of the collision avoidance methods and the path of the vehicle.

The internal dynamics for the vehicle in these simulations are given Figure A.5(NSB controller) and Figure A.6 (Dubins) in appendix A.

Table 7.7: Obstacle list from case four

Starting position [x , y]	radius	speed	heading
[50 , 62]	30	2	51.3 degrees

Table 7.8: Runtime results form case four

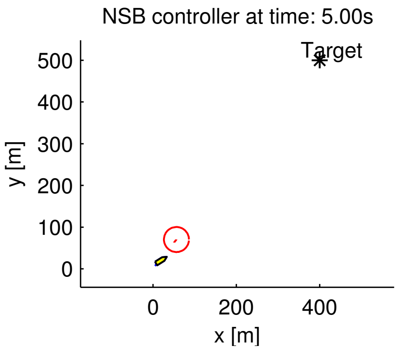
Methods	Collision avoidance	Steering law	Simulation time
NSB controller	0.637×10^{-5} s	0.377×10^{-5} s	103.6 s
Dubins	0.338×10^{-5} s	0.27×10^{-5} s	99.9 s

Simulation results with the NSB controller

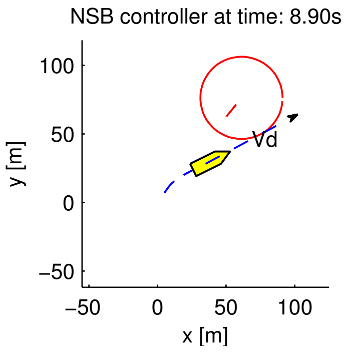
Figure 7.9 shows the path of the vehicle in the xy-plane with the NSB controller at 5, 8.9, 12, 47, 55 and 103.8 seconds in simulation time. The first subfigure (a) shows the vehicle right after start when the vehicle approaches the obstacle from behind and (b)-(c) shows the vehicle passing the obstacle the first time anticlockwise. Subfigure (d)-(e) shows the obstacle increasing the speed and approaching the vehicle, such that the vehicle has to perform collision avoidance again in order to let the obstacle pass clockwise. The complete path for the vehicle and the obstacle are given in Figure 7.9(f), where the path for the vehicle is plotted as the blue dotted line and the path for the obstacle is plotted as the red dotted line with the red safety circle.

Simulation results with the Dubins method for collision avoidance

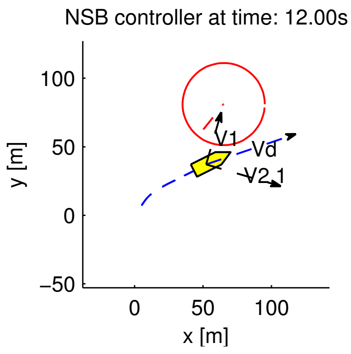
Figure 7.10 shows the path of the vehicle in the xy-plane with the Dubins method for collision avoidance at 5, 8, 45, 60, 65 and 99.9 seconds in simulation time. In Figure 7.9(a)-(b) it can be seen the vehicle approaching the obstacle from behind and passing it anticlockwise and in (c)-(d) the obstacle is approaching the vehicle and the vehicle avoid the obstacle by moving out to the left. After 65 seconds in Figure 7.9(e) the threat is avoided and the vehicle starts moving on the line from the point where the threat was avoided first to the target. The vehicle has reached the target after 99.9 seconds and the complete path of the vehicle is shown in Figure 7.9(f).



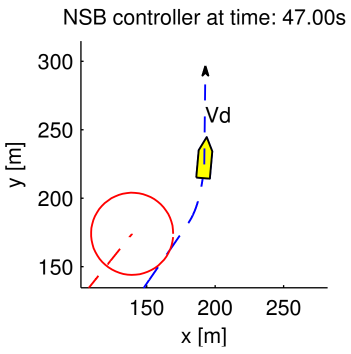
(a) The simulation at time 5 seconds, the vehicle has detected the obstacle



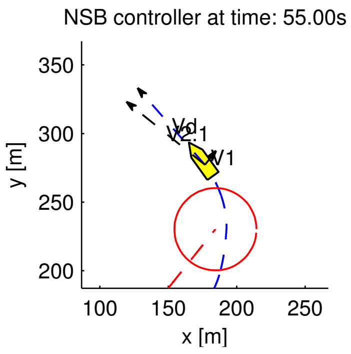
(b) The simulation at time 8.9 seconds, the vehicle is inside of activation circle one and is passing the obstacle anticlockwise



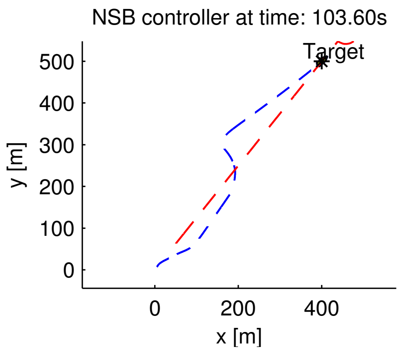
(c) The simulation at time 12 seconds, the vehicle is inside of activation circle two and the tasks *obstacle avoidance* and *traverse obstacle* are active



(d) The simulation at time 47 seconds, the obstacle has speeded up and approaching the vehicle from the behind and the vehicle has started obstacle avoidance

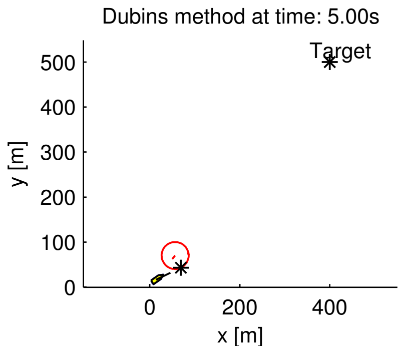


(e) The simulation at time 55 seconds, the vehicle is inside of activate circle two and the tasks *obstacle avoidance* and *traverse obstacle* are active

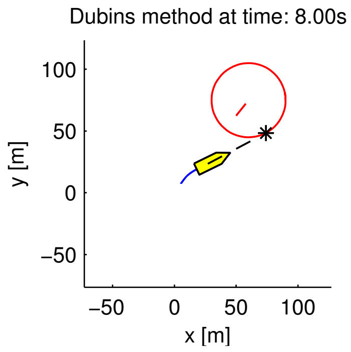


(f) The simulation at time 103.6, the vehicle has reached the target

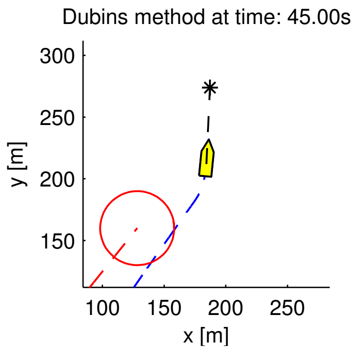
Figure 7.9: Simulation of the NSB controller when vehicle approaches the obstacle from behind and when the obstacle speeds up and the obstacle approaches the vehicle from behind.



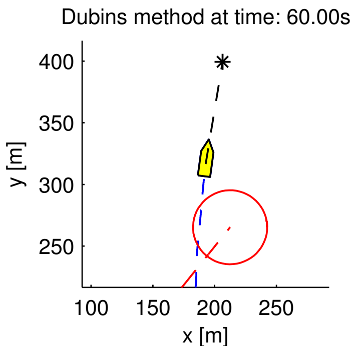
(a) The vehicle position at time 5 seconds approaching the obstacle from behind



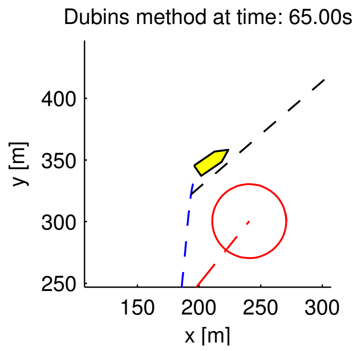
(b) The simulation at time 8 seconds, the vehicle is heading to the first waypoint on the obstacle circle in order to pass the obstacle anticlockwise



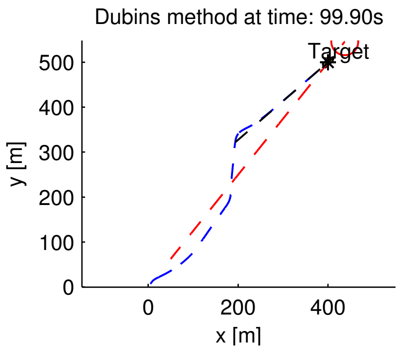
(c) The simulation at time 45 seconds the vehicle has passed the obstacle, but the obstacle is speeding up and the vehicle has to perform obstacle avoidance again



(d) The simulation at time 55 the vehicle heading to the first waypoint on the virtule obstacle circle in order to move out of the path of the obstacle



(e) The simulation at time 65, the vehicle has managed to get out of the obstacles path and is heading to the target



(f) The simulation at time 99.9, the vehicle has reached the target

Figure 7.10: Simulation of Dubins method for collision avoidance when the vehicle approaches the obstacle from behind and then the obstacle speeds up and the obstacle approaches the vehicle from behind.

7.5 Discussion

In this chapter four simulation cases have been presented with the NSB controller and the Dubins method for collision avoidance: One case with static obstacles and three cases with dynamic obstacles. In this section there will be a discussion about the results from these cases.

From the simulation cases it was found the average runtime for the methods and the steering laws, with the performance tool in Simulink. It was discovered that using this performance tool to find the average times was very unreliable since running the same simulation twice could give two difference results. Also for the steering law, which performs exactly the same calculations for all the simulations vary from $(0.26 - 0.377)10^{-5}$ seconds for the LOS and $(0.13 - 0.26)10^{-5}$ seconds for the steering command from the NSB controller. However in all the simulations the Dubins method for collision avoidance was faster than the NSB controller with some margin, but the LOS steering law was slower. Thus it can be concluded that the Dubins method is less computational intensive (excluding the steering law) than the NSB controller even considering the unreliable performance tool in Simulink. If the steering law is included the methods becomes more or less equal.

7.5.1 Case one: Static obstacles

The first case study where the vehicle had to avoid several static obstacles is shown in Figure 7.1 for the NSB controller and in Figure 7.2 for the Dubins method. It can be seen from both simulations that the vehicle avoids the first obstacle by passing it anticlockwise and the second clockwise, while ignoring the two other obstacles, which was not a threat. In Figure 7.1(b)-(c) it is shown the vehicle avoiding the first obstacle anticlockwise and the second clockwise, which is plotted with the velocity vectors V1 and V2.2 from the two tasks *obstacle avoidance* and *go to target*. It can be seen that task 2.1 *Traverse obstacle* is not activated since the vehicle is only avoiding static obstacle. The sum of the two tasks generates a desired velocity V_d, which points away from the obstacles, thus avoiding the collision. It should be noted that the velocity vector from task 2.2 does not point at the target when it is projected into the null-space of task 1.

The path of the vehicle for both simulations are plotted in Figure 7.11 with the four static obstacles. It can be seen that the vehicle with the Dubins method moves much closer to the obstacles, while the vehicle with the NSB controller moves further away from the obstacles in a s-shaped curve. Considering that the vehicle moves with constant speed, which is equal in both simulations and the vehicle with the Dubins used less time to reach the target, thus the path with the Dubins method is shortest.

From Table 7.4 it can be seen that the Dubins method for collision avoidance had an average runtime 41% faster than the NSB controller. It should be noted that the Dubins method does not generate new waypoints so often for static obstacles, while the NSB controller run continuously, which can account for most of the difference.

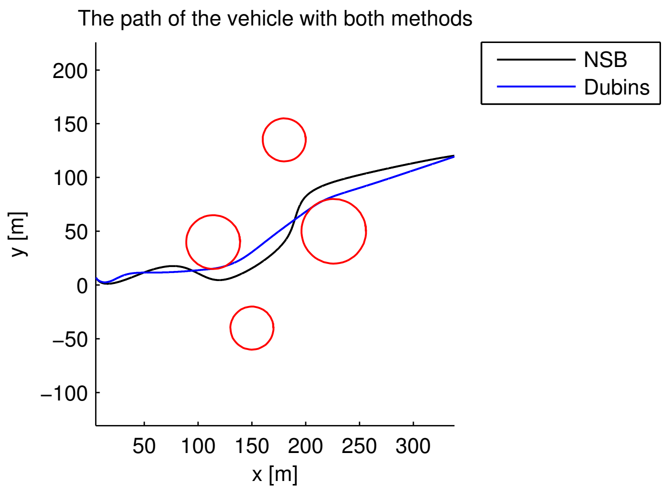


Figure 7.11: Case one with static obstacles with the path of the vehicle with both collision avoidance methods plotted as the blue and black lines and the obstacles plotted as the red circles

7.5.2 Obstacles from left and right

The second case study, which considers dynamic obstacles approaching the vehicle from the left and right is shown in Figure 7.3 for the NSB controller and Figure 7.5 for the Dubins method. Both of the methods passed the first obstacle clockwise and the second anticlockwise. The velocities for the obstacles rotated into the BODY frame of the vehicle are in the fourth quadrant for the first obstacle and the second quadrant for the second obstacle. Thus both of the methods have behaved according to the collision avoidance strategies described in Chapter 5 and successfully reached the target without colliding with any obstacles. It can be seen in the simulations that passing the obstacles from behind was safest. Considering that, an increase or decrease in speed for the obstacles would not have caused any problems for the vehicle.

The path of the vehicle for both simulations are shown in Figure 7.12. It can be seen that the vehicle with Dubins method makes a harder turn after the first obstacle, while the vehicle with the NSB controller moves in a slower circler path around. From Table 7.4 it can be seen that the vehicle with the NSB controller uses more time to reach the target than with the Dubinds method. The larger turning circle for the NSB controller around the first obstacle could account for this. It should be noted that the NSB controller can be tuned differently and achieve different behaviour and turn faster. However ν_d depend on both the tuning parameter for the tasks and the safety radius for the obstacle. Thus one set of parameter could work good for on set of obstacles, but bad for another set. The tuning parameter used in this thesis have been found to fit obstacles with a safety radius from 20-50 meter.

The internal dynamic in the simulations are plotted in Figure 7.4 for the vehicle with the NSB controller and Figure 7.6 for the vehicle with the Dubins method. It can be seen that the speed for both simulations are alike, this is because the vehicle in both simulations have used a constant speed value (7 m/s) and only the heading have been used to avoid collisions. The steering command from the collision avoidance methods and the vehicles heading are plotted in the figurers (c). It can be seen that the steering commands for both methods haves a lot of jumps. This is a consequence of the methods switching between different states, the NSB controller have the activations circles and the Dubins methods have the waypoints that it switching between, but the vehicle manages to follow the steering command relatively well. Even considering the saturations in the rudder, which is shown in the figures (d).

7.5.3 Heads-on

The third case, where the obstacle and the vehicle is on a heads-on collision course are shown in Figure 7.7 for the NSB controller and Figure 7.8 for the Dubins method. From Figure 7.7(b)-(c) it can be seen that the desired velocity vector from the NSB controller steer the vehicle away from the obstacle and in (d) the threat is avoided. In the simulation with the NSB controller the safety circle was increased by a factor of two inside of activation circle one, but not inside of activation circle two where the *obstacle avoidance* task was active. It was simulated and tested to have an increased safety circle inside of activation circle two also, but the vehicle entered the increased safety circle, which resulted in oscillation in v_d. This was also a problem with the Dubins method and in Figure 7.8(c) it can be seen that the vehicle is inside of the increased obstacle circle, since the black star, which is the next waypoint is on the wrong side of the vehicle. It should be noted that only increasing the safety circle around the obstacles in a heads-on situation might not be the best idea, since both of the methods failed to keeps the

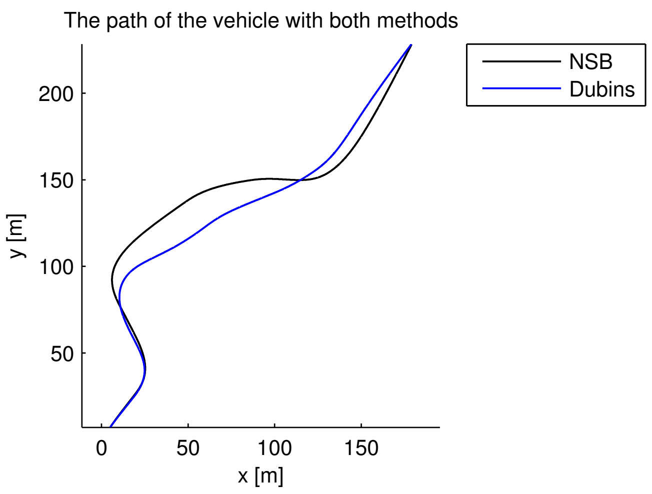


Figure 7.12: The path of the vehicle with both the collision avoidance methods in case two with dynamic obstacle approaching from the left and right

vehicle outside of the increased obstacle circle. If the speed to the obstacle had been increased, the methods might even fail to keep the vehicle outside of the original safety circle. A other approach could in these cases be to estimate where the obstacle and the vehicle would collide and set this point as an obstacle and used it as input to the collision avoidance methods. Or estimated the point between the collision point and the current location of the obstacle as center of a new obstacle with a radius equal the distance from the new center to the obstacle plus the old safety radius. This would have resulted in a new obstacle that overlapped the old obstacle and the collision point. However this shows the importance of considering the dynamic of the vehicle and also the dynamic of the obstacles when designing a collision avoidance system. The increased safety circle has saved us this time, but it cannot be guaranteed for all cases. In Figure 7.7(d) the threat is avoided for the NSB controller and the vehicle moves towards the target. The same happen in Figure 7.8(e) for the vehicle with the Dubins method.

The paths of the vehicle with the two methods are plotted in Figure 7.13. It can be seen that the paths are very alike before the obstacle, but around the obstacle the NSB takes the vehicle further away from the obstacle and after the obstacle the Dubins method as shown in Figure 7.8(e) moves on the line generated by waypoints from the vehicles position after the threat was avoided and the target while the NSB only aiming towards the target.

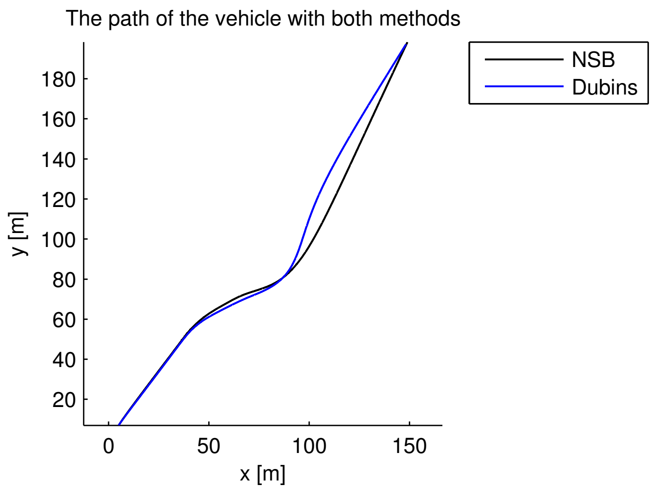


Figure 7.13: The path of the vehicle with both the collision avoidance methods in case three with dynamic obstacle approaching the vehicle heads-on

7.5.4 Approach form behind

Case four, where the vehicle approaches the obstacle from behind and passes it for then having to perform obstacle avoidance again when the obstacle increases the speed, is shown in Figure 7.9 for the NSB controller and Figure 7.10 for the Dubins method. It can be seen from Figure 7.9(a)-(c) and Figure 7.10(a)-(b) that the vehicle in both simulations successfully manage to avoid the obstacle anticlockwise the first time without any problems. After 40 seconds into the simulation the obstacle increase the speed and the vehicle has to perform obstacle avoidance again. This is shown in Figure 7.9(d)-(e) and Figure 7.10(c)-(e) for the NSB controller and the Dubins method respectively. For the NSB controller in Figure 7.9(d)-(e) it can be seen that the vehicle starts turning slowly to the left as it should according to the collision avoidance strategy described in Chapter 5. However when the obstacle gets closer to the vehicle, it starts turning very hard to the left and the heading is almost 90 degrees of with respect the target. Thus the vehicle has to turn more after the threat is avoided. The vehicle with the Dubins method for the same situation is slightly different, the turning direction is the same as with the NSB method, but the vehicle keeps more or less same course until the threat is avoided. This causes the vehicle to turn less when the vehicle gets back on the path to the target in comparison to the NSB controller.

The decision to avoid the obstacle clockwise if the obstacle approaches the vehicle from behind can be argued. It is seen from the simulations that the vehicle crossing in-front of the obstacle, thus making the velocity vector to the obstacle and the vehicle to cross. This can lead to a collision if the vehicle does not have time to cross over before the obstacle has reached the vehicle. It might be smarter to let the vehicle decide how to pass the obstacle based on, which direction the two velocity vector does not collide. In this case it would be anticlockwise and the vehicle would have moved out to the right, but if the obstacle also had a collision avoidance system and behaved in the same way that the vehicle when passing the obstacle the first time. It could have resulted in both the vehicle and the obstacle moving out to the right until they had collided into each other.

The Figure 7.14 shows the two methods plotted in the xy plane, the vehicle pass the obstacle the first time at position [80, 45] and the second time at [200, 200]. It can be seen that the two methods behave very alike around the obstacle the first time, but not the second time. Here the NSB controller makes a large s-turn where the Dubins method cross over and goes to the target. It can be seen from the simulations that the vehicle with the NSB controller used 103.6 second on the simulation and with the Dubins method 99.9 seconds and this is most likely because of the passing the seconds obstacle.

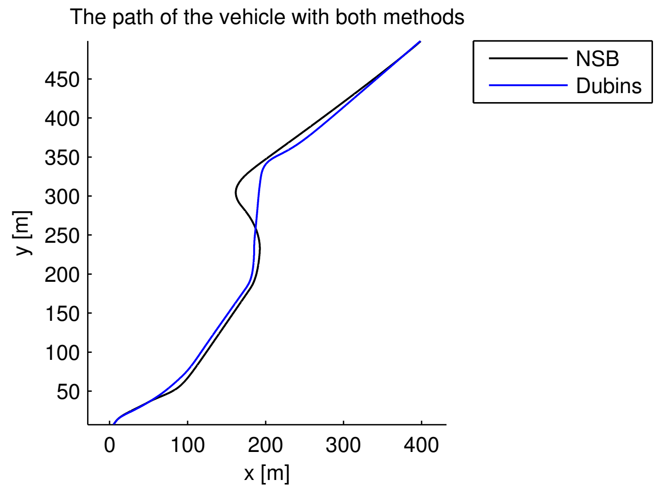


Figure 7.14: The path of the vehicle with both the collision avoidance methods in case four with dynamic obstacle approaching from behind

7.6 Conclusions from the simulations

A collision avoidance system using both the NSB controller and the Dubins method on a vehicle for collision avoidance have been studied with four simulations cases in this chapter. The methods worked very well for both static and dynamic obstacles and the vehicle has successfully managed to reached the target for all for cases. It has been seen that the Dubins method described in Chapter 5 was faster then the NSB method considering the average calculation time in all the simulations and the vehicle used less time to reach the target in all cases except in the heads-on simulations. Both of the methods resulted in jumps for the steering command, which resulted in saturation in the rudder for the vehicle when they jumped between different states, but the vehicle manage to follow the steering command relatively well.

Chapter 8

Object tracking and estimation of velocity and bearing to obstacles

In Chapter 3 to 6 of this thesis, it was assumed that the collision avoidance system received filtered and perfect information about the obstacles position, bearing and velocity. However this assumption have to be verified. In this chapter methods for object tracking and estimation of bearing and velocity using position data from a simulated radar will be discussed. Then there will be a simulation case to test the methods using a Kalman filter. Figure 8.1 illustrates the collision avoidance system where the focus is on the radar and estimation without the bypass.

8.1 Object-tracking equipment

One of the must common equipment for object-tracking is the radar. The radar will give a distance and an angle between the object and the vehicle as illustrated in Figure 8.2. The figure shows the vehicle and the object as two particles with a distance d between the vehicle and the object and the angle β to this vector.

Given a distance d and the angle β to the length from the vehicle and to the object it is possible to calculated the position function $f(\beta, d)$ to the object

$$f(\beta, d) = \begin{bmatrix} x_o^n \\ y_o^n \end{bmatrix} \tag{8.1}$$

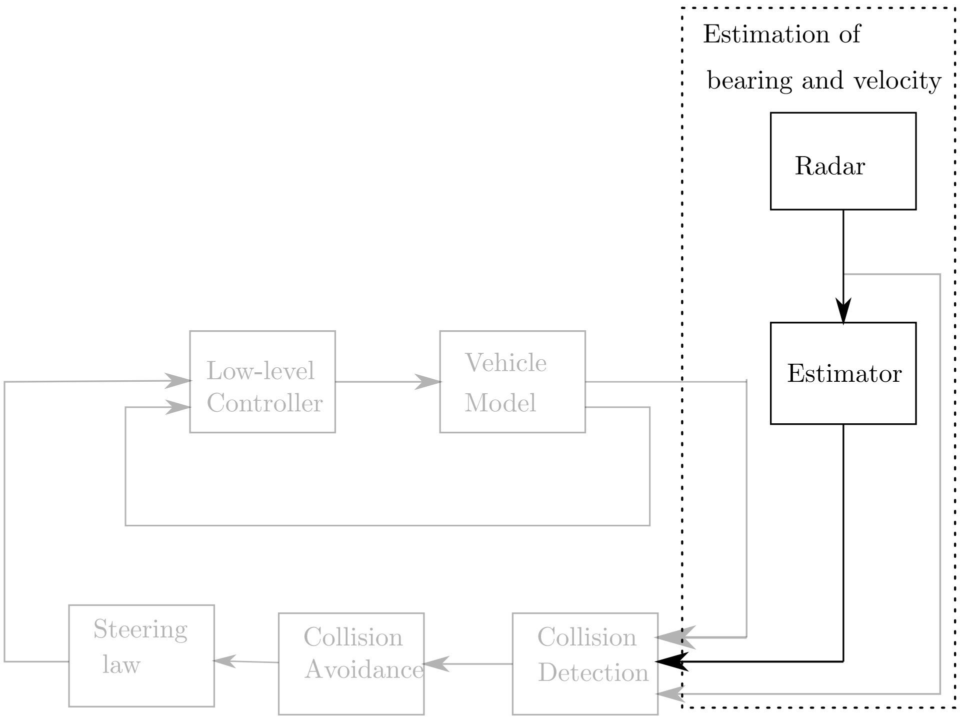


Figure 8.1: Model of the system, where the focus is on estimation of velocity and bearing using the position data from the radar

and use the position measured in a filter to estimate the velocity and the bearing to the object. From geometric equations we have

$$x^2 + y^2 = d^2 \tag{8.2}$$

$$tan(\beta) = \frac{y}{x} \tag{8.3}$$

where $x = x_o^n - x_v^n$ and $y = y_o^n - y_v^n$. Inserting (8.2) in (8.3) and solving for x_o^n and y_o^n

$$x_o^n = \pm\sqrt{\frac{d^2}{tan(\beta)^2 + 1}} + x_v^n \tag{8.4}$$

$$y_o^n = \pm\sqrt{\frac{tan(\beta)^2 d^2}{tan(\beta)^2 + 1}} + y_v^n \tag{8.5}$$

Which gives the x, y position to the obstacle in the xy-plane. The bearing θ_o to the object can be found by two or more position measurements

$$\theta_o = atan2(y_o^n(k) - y_o^n(k-1), x_o^n(k) - x_o^n(k-1)) \tag{8.6}$$

where $x_o^n(k)$ and $y_o^n(k)$ is the position of the object in the xy-plane at time k. Or by using the velocity in the NED frame

$$\theta_o = atan2(v_o^n, u_o^n) \tag{8.7}$$

where v_o^n is the y-component of the velocity and u_o^n is the x-component of the velocity in the NED frame. In the rest of this thesis it is assumed that $f(\beta, d)$ is measured and will be used directly into the filters with a sampling time of 10 Hz to estimate the velocity and bearing of the objects.

$$\mathbf{P}_o^n(k) := f(\beta, d) \quad at \quad 10Hz$$

8.2 Target tracking methods

For object-tracking of an unknown object it is unlikely to have a perfect model of the object since parameters are unknown. Two models have been studied for estimation of velocity and bearing, one using the NED frame and one using the BODY frame. In NED it is assumed that the Coriolis effect can be neglected and that the NED frame is equal the inertial frame (flat earth navigation), such that

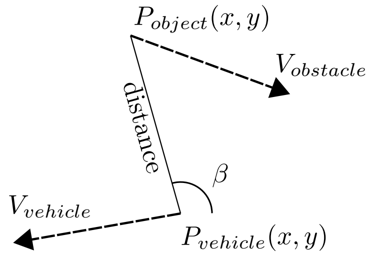


Figure 8.2: The figure show the distance and the bearing to a obstacle, which can be extracted from a radar measurement

Netwons laws can be applied. This assumption can be justified for small distance between the vehicle and the object and considering that the range of a radar is not that long, the assumption is reasonable.

Newtons second law $a^n M = \sum F$ have been used to estimate the position and velocity in the NED frame. Since both the mass M and the force $\sum F$ is unknown, the $\sum F/M$ term is replaced with a bias term B' and

$$\dot{B}'T + B' = 0 \tag{8.8}$$

where T is a tuning parameter. The model in the NED frame used to estimated the velocity becomes:

$$\dot{\boldsymbol{\eta}}_o^n = \boldsymbol{\nu}_o^n \tag{8.9}$$

$$\dot{\boldsymbol{\nu}}_o^n = B' \tag{8.10}$$

$$\mathbf{T}^n \dot{\boldsymbol{B}}' + \boldsymbol{B}' = 0 \tag{8.11}$$

where $\boldsymbol{\eta}_o^n = [x_o^n,\ y_o^n]^T$ is the x and y position in the NED frame and $\boldsymbol{\nu}_o^n = [\dot{x}_o^n,\ \dot{y}_o^n]^T$ is the velocity in the NED frame. $\mathbf{T}^n = diag([T1,\ T2])$ is tuning parameter

$$T1 = T2 = 10$$

and $\boldsymbol{B}' = [B1,\ B2]^T$ is the bias term, which replace the $\sum F/M$ in x and y direction. It has been tested with other tuning parameter (T1 and T2), but it has been found that these values work best for tracking the object presented in Section 8.4. The system can be written on the linear state space form

$$\dot{\mathbf{x}}_o^n = \mathbf{A}\mathbf{x}_o^n + \mathbf{B}u \tag{8.12}$$

$$\mathbf{y}_o^n = \mathbf{H}\mathbf{x}_o^n + \mathbf{v} \tag{8.13}$$

where

$$\mathbf{x}_o^n = [x_o^n,\ y_o^n,\ \dot{x}_o^n,\ \dot{y}_o^n,\ B1,\ B2]^T$$
$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & -\mathbf{T}^{n^{-1}} \end{bmatrix}, \quad \mathbf{B} = \mathbf{0}_{6\times6}$$
$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times4} \end{bmatrix}$$

and $\mathbf{v}$ is white noise from the radar signals. For the method in the BODY frame a vessel has been used

$$\dot{\boldsymbol{\eta}}_o^b = \boldsymbol{\nu}_o^b \tag{8.14}$$
$$\mathbf{M}\dot{\boldsymbol{\nu}}_o^b + \mathbf{D}\boldsymbol{\nu}_o^n = b\boldsymbol{\tau} \tag{8.15}$$

where $\boldsymbol{\eta}_o^b = [x_o^b,\ y_o^b]^T$ is the x and y position in the BODY frame and $\boldsymbol{\nu}_o^b = [u_o^b,\ v_o^b]^T$ is the velocity in the BODY frame. Since all the terms are assumed to be unknown the model can be simplified by defining

$$\mathbf{T^b} := \frac{\mathbf{M}}{\mathbf{D}} \tag{8.16}$$
$$\mathbf{B}^b := \frac{b\boldsymbol{\tau}}{\mathbf{M}} \tag{8.17}$$
$$\tag{8.18}$$

where $\mathbf{T}^b = diag(T1^b,\ T2^b)$ is tuning parameter and $\mathbf{B}^b = [b1^b,\ b2^b]^T$ is states with

$$\dot{\mathbf{B}}^b - \frac{1}{\boldsymbol{T}}^b \boldsymbol{B}^b = 0 \tag{8.19}$$

The system can be written on the linear state space form

$$\dot{\mathbf{x}}_o^b = \mathbf{A}\mathbf{x}_o^b + \mathbf{B}u \tag{8.20}$$

where

$$x_o^b = [x_o^b,\ y_o^b,\ u_o^b,\ v_o^b,\ b1^b,\ b2^b]^T$$
$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & -\mathbf{T}^{b^{-1}} \end{bmatrix}, \quad \mathbf{B} = \mathbf{0}_{6\times6}$$

8.2.1 Discussion

The output from the radar measurement simulated in this thesis will be the obstacles position $\mathbf{P}_o^n(k) = [x_o^n(k),\ y_o^n(k)]^T$ in the NED frame and will be sampled at $10Hz$ with white noise. The output from the estimator should be the filtered position in NED, the bearing of the obstacle and the velocity in the BODY frame. This will give the collision avoidance methods the data it needs to avoid collision with the obstacles. It can be seen from the models in NED and BODY that the bearing is not included, thus it needs to be calculated. There are two ways of calculating the bearing, one uses the position data from two or more samples:

$$\theta_o = atan2(y_o^n(k) - y_o^n(k-1), x_o^n(k) - x_o^n(k-1)) \tag{8.21}$$

where $x_o^n(k)$ and $y_o^n(k)$ is the position of the object in the xy-plane at time k. Or by using the velocity in the NED frame.

$$\theta_o = atan2(\dot{y}_o^n, \dot{x}_o^n) \tag{8.22}$$

where $\dot{x}_o^n$ is the x-component of the velocity and $\dot{y}_o^n$ is the y-component of the velocity in the NED frame. The method in the NED frame can use the input directly and calculate the bearing from the estimated velocity using (8.22). The second method in the BODY frame cannot use the the measurement input directly since it needs the bearing of the obstacle before it can rotate the positions from NED to BODY frame. It was tested using the position measurement and (8.21), but because of the noise introduced into the system it was difficult to get a good bearing θ_o on the object. This resulted in a worse position estimate then the measurement position with white noise and the velocity could not be estimated. The simulation results from the method in the BODY frame is included in the Appendix B and the only focus further will be on the method using the NED frame.

8.3 Kalman filter

Considering that the radar is simulated to run at 10 Hz and the filter will run at 100 Hz, the discrete Kalman filter is chosen to estimate the velocity and bearing for the object. Kalman filter was published in 1960 [Kalman, 1960] and is one of the most commonly used estimation method. As stated in [Vik, 2012] the linear discrete Kalman filter has the following properties:

1. The estimate is unbiased and of minimum variance
2. The Kalman filter is the optimal linear state estimator
3. The Kalman filter is asymptotically stable

if the following assumption are valid:

1. The process and measurement noise are white and Gaussian

2. The initial state is Gaussian
3. The system is linear
4. The system is observable

Table 8.1 summarize the discrete kalman filter. Using forward Euler

$$x(k+1) = x(k) + hf(x(t), u(t)) \tag{8.23}$$

to discretize the model (8.9) -(8.11) in NED on discrete from

$$x(k+1) = \mathbf{\Phi}_{NED}(k)\mathbf{x}(k) + \mathbf{\Delta}(\mathbf{k})u(k) + \mathbf{\Gamma}w(k) \tag{8.24}$$

$$y(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{8.25}$$

where

$$\mathbf{\Phi}_{NED} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{I}_{2\times2}h & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{I}_{2\times2} & \mathbf{I}_{2\times2}h \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{I}_{2\times2} - h\mathbf{T}^{-1} \end{bmatrix}, \quad \mathbf{\Delta} = \mathbf{0}_{6\times2} \tag{8.26}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times4} \end{bmatrix}, \quad \mathbf{\Gamma} = 1 \tag{8.27}$$

h in $\mathbf{\Phi}_{NED}$ is the step size in the euler equation and in this thesis it has been set as 0.01. It can be seen that the model does not include the bearing θ_o, thus (8.7) is used with the estimated velocity in the NED frame.

The pseudo code for the kalman filter implemented in Matlab is shown below. It can be seen that the equation from Table 8.1 is used, however there are only a correction in the filter for every 10 samples since the filter runs at 100 Hz and the input runs at 10 Hz.

Estimator: pseudo code

```
$y^n$ measured from radar
$\mathbf{R_d}(k) = \mathbf{R_d}(k)^T > 0$
$\mathbf{Q_d}(k) = \mathbf{Q_d}(k)^T > 0$
$\mathbf{H}(k)$ according to (8.27)
$\mathbf{\Gamma}(k)$ according to (8.27)

if counter == 10 then
    $\mathbf{K} = \mathbf{K}(k) = \bar{\mathbf{P}}(k)\mathbf{H}[\mathbf{H}(k)\bar{\mathbf{P}}(k)\mathbf{H}(k) + \mathbf{R}_d(k)]^{-1}$
    $counter = 0$
else
    $\mathbf{K} = 0$
end if

$\hat{x} = \hat{x} + K[y - H\bar{x}]$
$\hat{\mathbf{P}}(k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\bar{\mathbf{P}}(k)[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]^T + \mathbf{K}(k)\mathbf{R}_d(k)\mathbf{K}^T(k)$
$\bar{x} = \mathbf{\Phi}(k)\hat{\mathbf{x}}(k)$
$\bar{\mathbf{P}}(k+1) = \mathbf{\Phi}(k)\hat{\mathbf{P}}(k)\mathbf{\Phi}^T(k) + \mathbf{\Gamma}(k)\mathbf{Q}_d(k)\mathbf{\Gamma}^T(k)$
$counter = counter + 1$
Find $\hat{\theta}_{object}$ according to (??)
Find velocity in BODY according to (??)
```

Table 8.1: Discrete Kalman filter

Tuning matrix	$\mathbf{Q_d}(k) = \mathbf{Q_d}(k)^T > 0 \quad \mathbf{R_d}(k) = \mathbf{R_d}(k)^T > 0$
Initial conditions	$\hat{\mathbf{x}}(0) = \mathbf{x}_0$ $\bar{P}(0) = \mathbf{E}[\ (\mathbf{x}(0) - \hat{\mathbf{x}}(0))(\mathbf{x}(0) - \hat{\mathbf{x}}(0))^T]$
Kalman gain matrix, State estimate update and Error covariance update	$\mathbf{K}(k) = \bar{\mathbf{P}}(k)\mathbf{H}[\mathbf{H}(k)\bar{\mathbf{P}}(k)\mathbf{H}(k) + \mathbf{R}_d(k)]^{-1}$ $\hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k)\mathbf{K}(k)[\mathbf{y}(k) - \mathbf{H}(k)\bar{\mathbf{x}}(k)]$ $\hat{\mathbf{P}}(k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\bar{\mathbf{P}}(k)[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]^T$ $+\mathbf{K}(k)\mathbf{R}_d(k)\mathbf{K}^T(k)$
State estimation propagation Error covariance propagation	$\bar{\mathbf{x}}(k+1) = \mathbf{\Phi}(k)\hat{\mathbf{x}}(k) + \mathbf{\Delta u}(k)$ $\bar{\mathbf{P}}(k+1) = \mathbf{\Phi}(k)\hat{\mathbf{P}}(k)\mathbf{\Phi}^T(k) + \mathbf{\Gamma}(k)\mathbf{Q}_d(k)\mathbf{\Gamma}^T(k)$

8.3.1 Tuning parameters and the initial conditions for the Kalman filter

For the Kalman filter one needs to define the tuning matrix's $\mathbf{R}_d$, $\mathbf{Q}_d$, the initial condition for the state estimator $\hat{\mathbf{x}}_0$ and the initial covariance propagation matrix $\bar{\mathbf{P}}$. In this thesis the $\mathbf{R}_d$ have been set as the variance of the white noise (0.1^2)

multiplied with the step size h , $\mathbf{R}_d = diag([0.1^2,\ 0.1^2])h$. The $\mathbf{Q}_d$ as been set as $\mathbf{Q}_d = diag([0.01,\ 0.01,\ 0.01,\ 0.01,\ 0.01,\ 0.01]$. The initial for the state estimator has been set to the first measured value of x_o^n and y_o^n and zero for the four other terms. The initial covariance propagation matrix $\bar{\mathbf{P}}$ has been set to $\mathbf{0}_{6\times6}$.

8.4 Object simulator

In order to test the tracking method on an object, the vehicle model (6.3)-(6.6) in Chapter 6 has been used with the controller and the LOS steering law presented in Chapter 5 as the object. The parameters to the object model is shown in Table 8.2.

Table 8.2: Parameter for the object model used in the simulation for object-tracking

Parameters	Value
T_{surge}	5
T_{yaw}	7
b_{yaw}	1
k_{surge}	1
$\delta_{max}, \delta_{min}$	±35 deg
τ_{max}, τ_{min}	±10

8.5 Simulation: Object-tracking

In this section there will be one simulation of object-tracking using the Kalman filter and the model in the NED frame, where the object represent an obstacle. The initial conditions of the object is shown in Table 8.3 and the object will follow the path generated by the waypoints and follow the speed according to Table 8.3. After 100 seconds into the simulation the speed of the object will increase from 3 m/s to 7 m/s. The measured position data, which is the input to the Kalman filter is set to 10 Hz while the Kalman filter will run at 100 Hz and the noise on the input position data has a variance of 0.1. The goals of this simulation are to:

- Test the estimation of velocity and bearing using the position data.
- Test the Kalman filter ability to follow a change in both heading and speed.
- See if the results are good enough to be used in the collision avoidance system.

Table 8.3: Initial condition for the object

Conditions	values
Starting position $[x,\ y,\ \psi]$	$[1\ \ 1\ \ 0]$
speed from 0 to 100 seconds	3 m/s
speed from 100 to 200 seconds	7 m/s
waypoints	$\begin{matrix} -5 & 200 & 500 \\ 5 & 150 & 800 \end{matrix}$

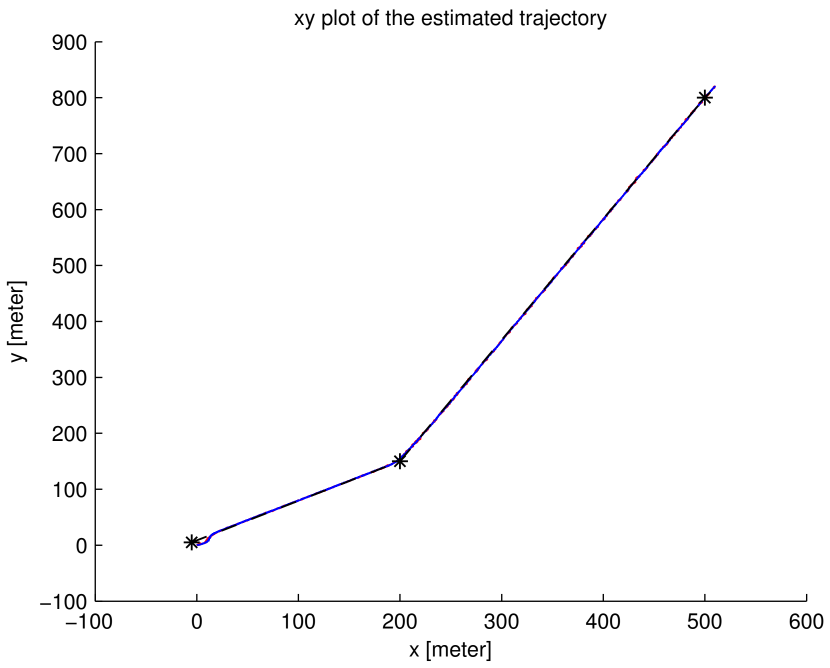
8.5.1 Results

In Figure 8.3(a) the path of the tracked object is plotted in the xy-plane, where the black stars are the waypoints and the blue line is the path of the object. Figure 8.3(b) shows the x, y positions in the NED frame and the bearing to the tracked object where the blue line is the estimated values and the red line is the true values. The velocity in the BODY frame (u_o^b, v_o^b) to the tracked object and the error between the estimated u_o^b and the true value is shown in Figure 8.4(a) where the red dotted line is the true value without noise and the blue line is the estimated values. The error between the estimated positions and the true value is shown in Figure 8.4(b) and the error in bearing for the time interval from 50 to 200 seconds.

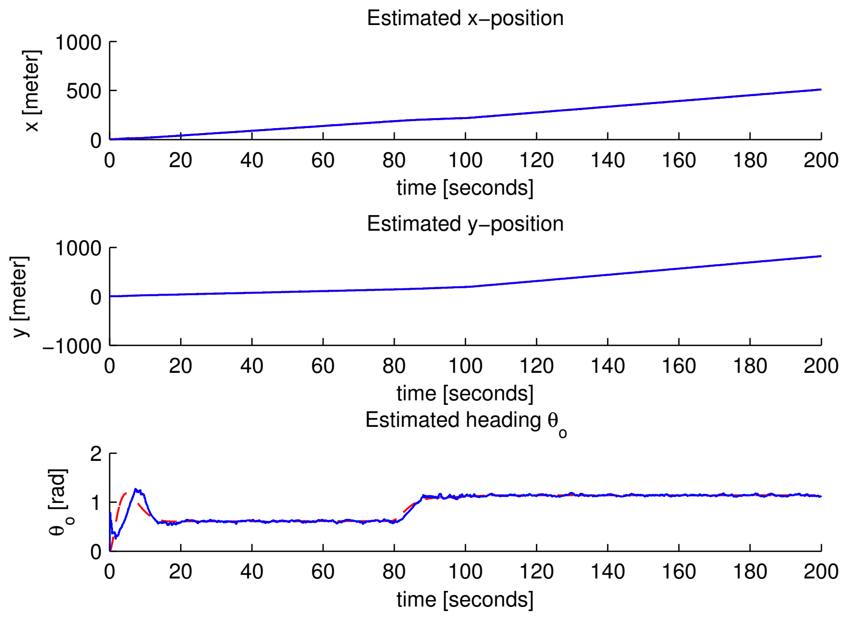
8.5.2 Discussion and conclusion

In the simulation in Section 8.5, shown in Figure 8.3 and Figure 8.4 it has been tested a discrete Kalman filter tracking an object. The path is plotted in the xy-plane in Figure 8.3(a) and the error between the estimated and the true values for the positions is plotted in Figure 8.4(b). It can be seen that the error in x and y is approximately 1 to -0.6, except at time 0-20 seconds and around time 100 seconds. The first jump in the error is expected since the filter need time to calibrate and the second around time 100 seconds is most likely because of the change in the speed. However the position estimation overall is relatively good.

For the velocity, it can be seen that the Kalman filter manages to estimate the velocity relatively well, but in the start of the simulation (from 0 to 15 seconds) the estimation of velocity have a overshot to 5 m/s when the true value is 3 m/s before the filter converge to the true value. Considering that the filter starts with wrong initial values and the filter normally needs time to start up, it is not that surprising results. When locking at the change in velocity (at time 100 seconds) the filter manages to follow the change and converge to the new value. The sway velocity is estimated to be approximately 2 $\times 10^{-15} m/s$ to -2 $\times 10^{-15} m/s$, which is approximately zero and the true value is zero, thus the speed for the object will be equal the velocity in surge u. The heading, which is shown in the bottom of Figure 8.3(b) uses 15 seconds to converge and manages to follow the change in heading relative well. The error is around 0.02 to -0.04 [rad] if the start and the change in heading is not included. The jump in heading is not included since the error is so large, which can also be seen in Figure 8.3(b). It can be concluded that it is possible to use both the velocity and bearing from the filter in a collision avoidance system. However it can be smart to let the filter run for a couple of second before the collision avoidance system starts using the estimated values. The error in speed or bearing can be compensated in the collision avoidance system with increased safety circle around the obstacle.

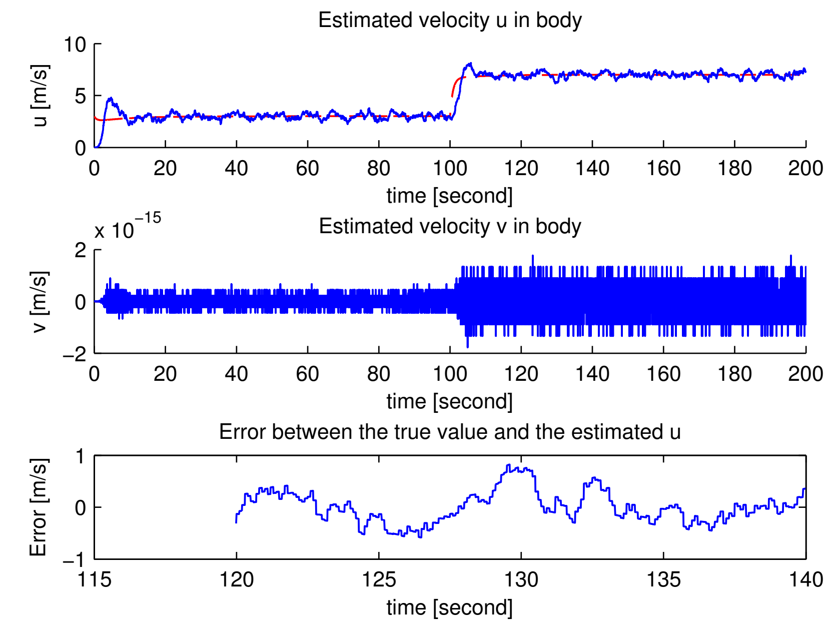


(a) The xy-plot of the estimated path as the blue line and the true path as the red line. The black stars are the waypoints

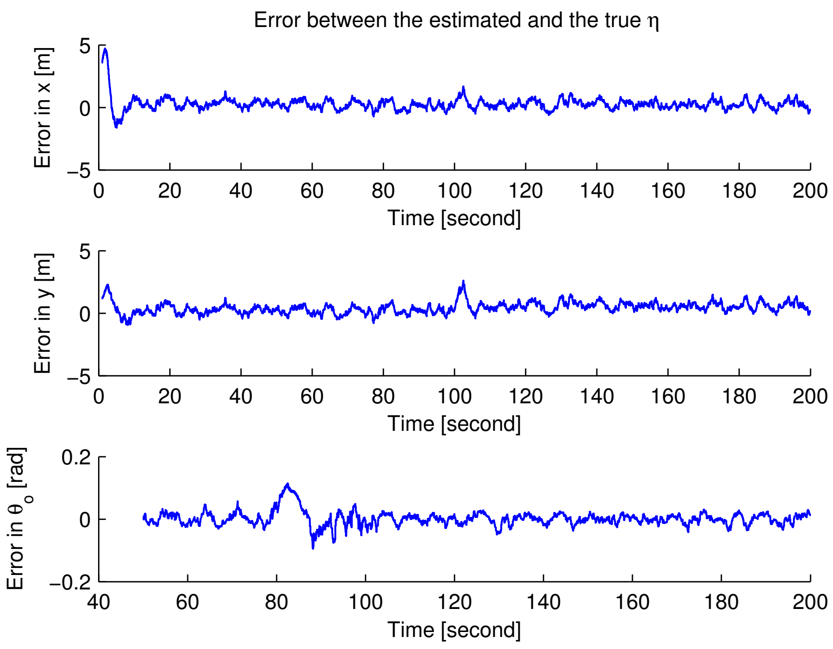


(b) Estimated x,y positions and the bearing as the blue line and the true values as the red lines

Figure 8.3: Object-tracking: The tracked object in the xy-plane, the estimated positions in x and y in the NED frame and the bearing to the tracked object



(a) The estimated velocity in the BODY frame and the error velocity in surge



(b) The error in the positions and the bearing

Figure 8.4: Object-tracking: The estimated velocity in the BODY frame and the error in surge, positions and the bearing

Chapter 9

Conclusion and further work

9.1 Conclusion

In this thesis a collision avoidance system has been constructed with two collision avoidance methods, the NSB controller and the Dubins method for collision avoidance. The vehicle model was implemented to simulate more realistic case studies, which is presented in Chapter 7. One case with static obstacles and the last three with dynamic obstacles. In all the simulations with the collision avoidance methods the vehicle manage to reach the target without colliding with any obstacles. This is consistent with the stability analysis, which concluded with locally stability for both methods.

The average calculation time to the collision avoidance methods simulated in Chapter 7 was logged and it was concluded that the Dubins method was faster than the NSB controller in all cases, but the LOS steering law, which the Dubins method used was slower than finding the steering command to the NSB controller. By considering both the steering laws and the methods, which resulted in a total average calculation time that was approximately equal for both methods. Thus the average calculation time could not be used to determine, which method that is best. By considering the complexity of the codes. The Dubins method included a lot of if else statements, which made it more complex in comparison to the NSB controller. Also the fact that the Dubins method would enter the obstacle circle if the target is inside of the obstacle circle, where the NSB method could at least guarantee not to enter the obstacle circle. Thus it is recommended to use the NSB controller for further work.

In Chapter 8 it was discussed methods for target tracking as well as estimation of bearing and velocity and it was concluded that with a model in NED and a Kalman filter it is possible to get good estimation of velocity and bearing using position data, which can be used in the collision avoidance system.

9.2 Further work

It is recommended for further work within collision avoidance system to continue with the NSB controller rather than the Dubins method. Mainly because of all the if else statement in the Dubins method, which made it more complex than the NSB controller. Also it should be investigated different approaches on the most dangerous situations like the heads-on situation shown in Section 7.3 and the approach from behind shown in Section 7.4.

Bibliography

[Arrichiellos, 2006] Arrichiellos, F. (2006). *Coordination Control of Multiple Mobile Robots*. PhD thesis.

[Borenstein and Koren, 1989] Borenstein, J. and Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots.

[Breivik and Fossen, 2004] Breivik, M. and Fossen, T. I. (2004). Path following for marine surface vessels. *OCEANS '04. MTTS/IEEE TECHNO-OCEAN '04.*

[Brock and Oussama.Khatib, 1999] Brock, O. and Oussama.Khatib (1999). High-speed navigation using the global dynamic window approach. *lEEE International Conference on Robotics and Automation Detroit. Michigan May 1999.*

[Cellini et al., 2007] Cellini, Mati, Pollini, and Innocenti (2007). Obstacle avoidance for autonomous ground vehicles in outdoor environments.

[Chakravarthy and Ghose, 1998] Chakravarthy, A. and Ghose, D. (1998). Obstacle avoidance in a dynamic environment: A collision cone approach. *AIAA Guidance, Navigation and Control Conference and Exhibit.*

[Daily and Bevly, 2008] Daily, R. and Bevly, D. M. (2008). Harmonic potential field path planning for high speed vehicles.

[Dubins, 1979] Dubins, L. E. (1979). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, pages 497–516.

[Fossen, 2011] Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. WILEY.

[Foxy et al., 1997] Foxy, D., Burgardy, W., and Thrunyz, S. (1997). The dynamic window approach to collision avoidance.

[Fu-guang et al., 2005] Fu-guang, D., Peng, J., Xin-qian, B., and Hong-jian, W. (2005). Auv local path planning based on virtual potential field.

[Gunnarsson et al., 2002] Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., and Nordlund, P.-J. (2002). Particle filters for positioning and navigation, and tracking.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

[Khalil, 2001] Khalil, H. K. (2001). *Nonlinear systems.* IE.

[Khatib, 1985] Khatib, O. (1985). Real time obstacle avoidance for manipulators and mobile robots. *Robotics and Automation. Proceedings. 1985 IEEE International Conference.*

[Koren and Borenstein, 1991] Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation.

[Krogh and Thorpe, 1986] Krogh, B. H. and Thorpe, C. E. (1986). Integrated path planning and dynamic steering control for autonomous vehicle. *Robotics and Automation. Proceedings. 1986 IEEE International Conference.*

[Lee and Kim, 2011] Lee, Y. and Kim, Y. (2011). Distributed unmanned aircraft collision avoidance using limit cycle.

[Liu and Narayanan, 2011] Liu, F. and Narayanan, A. (2011). Real time replanning based on a* for collision avoidance in multi-robot systems. *The 8th International Conference on Ubiquitous Robots and Ambient Intelligence URAI 20.*

[Loong et al., 2011] Loong, W. Y., Long, L. Z., and Hun, L. C. (2011). A star path following mobile robot. *International Conference on Mechatronics.*

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization.* Springer.

[NoriyasuNoto et al., 2011] NoriyasuNoto, HiroyukiOkuda, YuichiTazaki, ShinkichiInagaki, and TatsuyaSuzuki (2011). Obstacle avoidance assisting system basedon personalized potentialfield.

[Tsourdos et al., 2011] Tsourdos, A., White, B., and Shanmugavel, M. (2011). *Cooperative Path Planning of Unmanned Aerial Vehicles.* WILEY.

[Vik, 2012] Vik, B. (2012). *Integrated Satellite and Inertial Navigation Systems.* Department of engineering cybernetics.

[Watanabe et al., 2007] Watanabe, Y., Calisey, A. J., and Johnsonz, E. N. (2007). Vision-based obstacle avoidance for uav‘s. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS.*

[Yongjie and Yan, 2009] Yongjie, Y. and Yan, Z. (2009). Collision avoidance planning in multi-robot based on improved artificial potential field and rules.

[Ögren and Leonard, 2005] Ögren, P. and Leonard, N. E. (2005). A convergent dynamic window approach to obstacle avoidance. *IEEE TRANSACTIONS ON ROBOTICS, VOL. 21, NO. 2, APRIL 2005.*

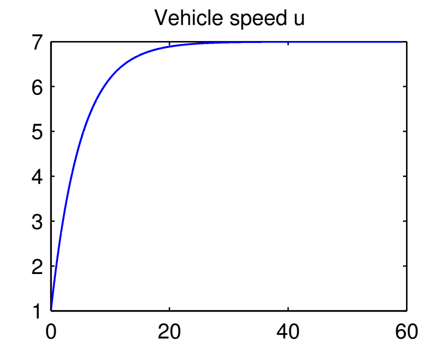
Appendices

Appendix A

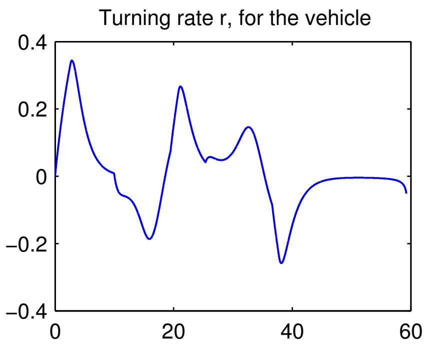
Simulations results from collision avoidance

The internal dynamic for the vehicle in the simulations from Section 7.1 , 7.3 and 7.4 with the NSB controller and the Dubins method for collision avoidance are shown in this appendix.

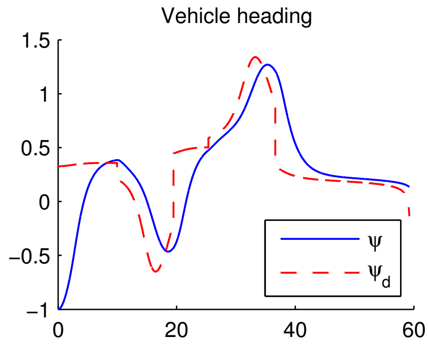
Case One static obstacles



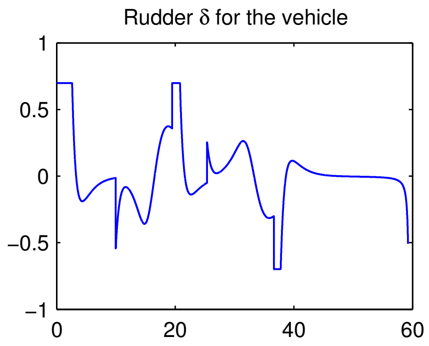
(a) The vehicle speed for the simulation in Figure 7.1



(b) The vehicle turning rate for the simulation in Figure 7.1

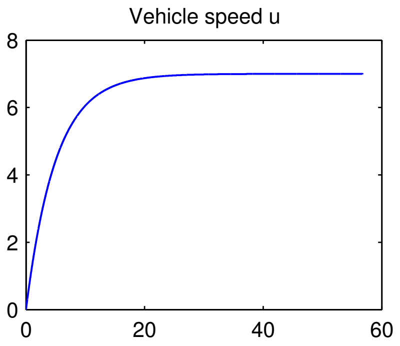


(c) The vehicles heading and the steering command from the NSB controller in simulation shown in Figure 7.1

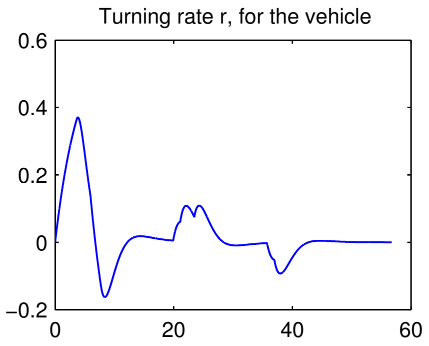


(d) The rudder angle for the vehicle in simulation Figure 7.1

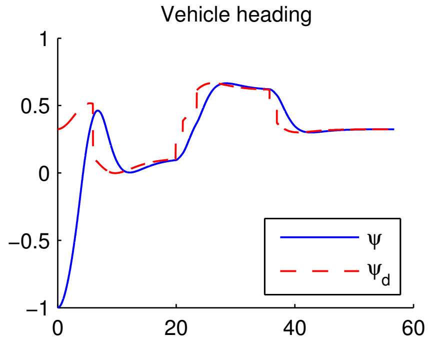
Figure A.1: The internal dynamic for the vehicle in the simulation shown in Figure 7.1 with the NSB controller



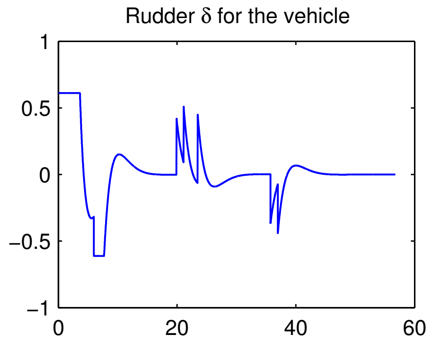
(a) The vehicle speed for the simulation in Figure 7.2



(b) The vehicle turning rate for the simulation in Figure 7.2



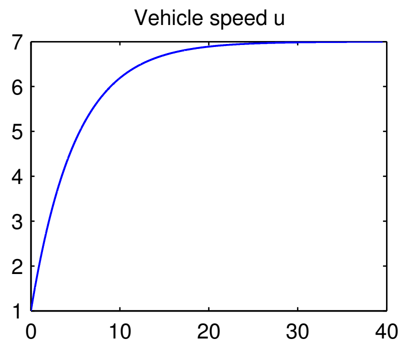
(c) The vehicles heading and the steering command from the LOS steering law in Figure 7.2



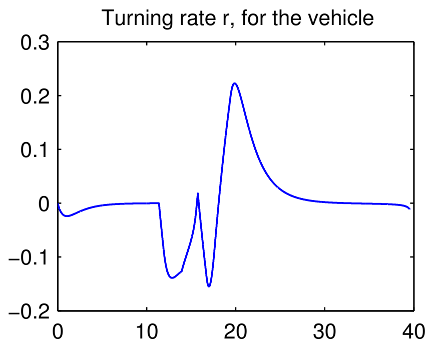
(d) The rudder angle for the vehicle in simulation Figure 7.2

Figure A.2: The internal dynamic for the vehicle in the simulation shown in Figure 7.2 for the Dubins method

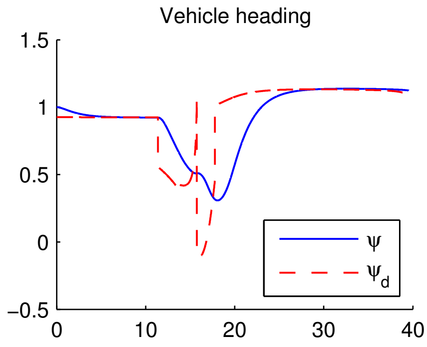
Case Three: Heads-on situation



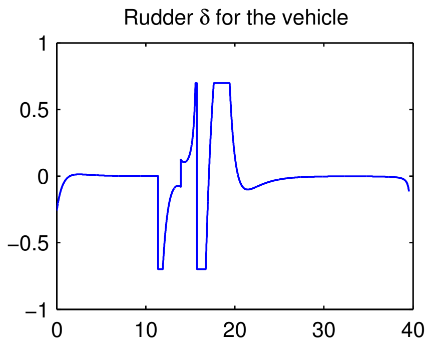
(a) The vehicle speed for the simulation in Figure 7.7



(b) The vehicle turning rate for the simulation in Figure 7.7

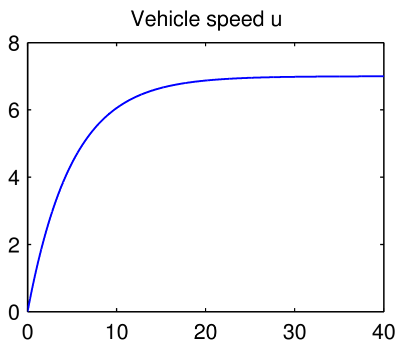


(c) The vehicles heading and the steering command from the NSB controller in simulation shown in Figure 7.7

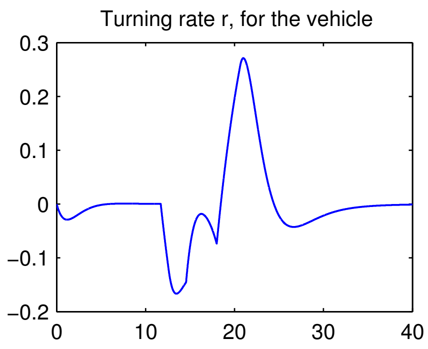


(d) The rudder angle for the vehicle in simulation Figure 7.7

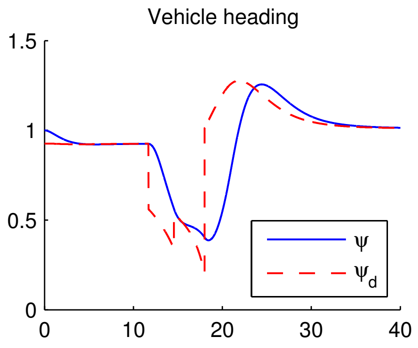
Figure A.3: The internal dynamic for the vehicle in the simulation shown in Figure 7.7



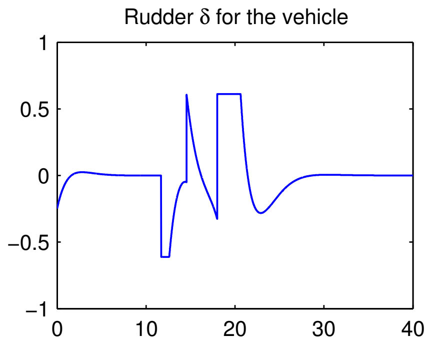
(a) The vehicle speed for the simulation in Figure 7.8



(b) The vehicle turning rate for the simulation in Figure 7.8



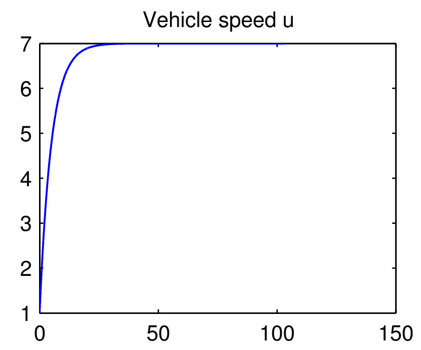
(c) The vehicles heading and the steering command from the LOS steering law in simulation shown in Figure 7.8



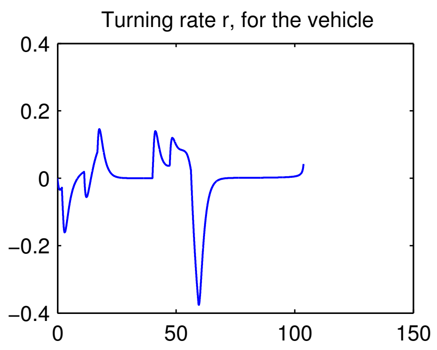
(d) The rudder angle for the vehicle in simulation Figure 7.8

Figure A.4: The internal dynamic for the vehicle in the simulation shown in Figure 7.8

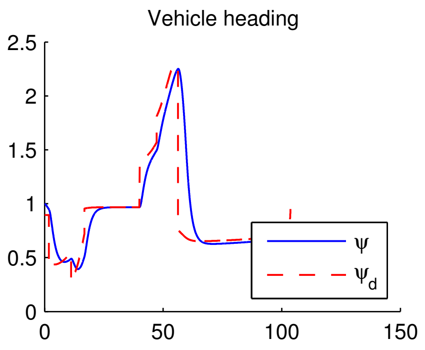
Case four: Approach form behind



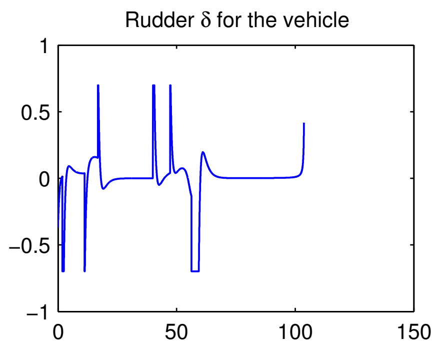
(a) The vehicle speed for the simulation in Figure 7.9



(b) The vehicle turning rate for the simulation in Figure 7.9

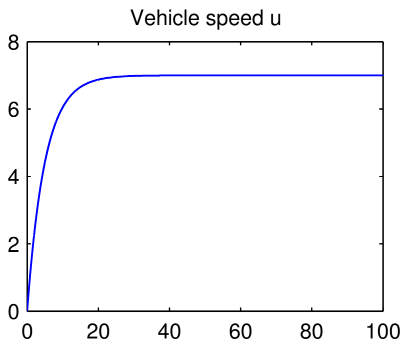


(c) The vehicles heading and the steering command from the NSB controller in simulation shown in Figure 7.9

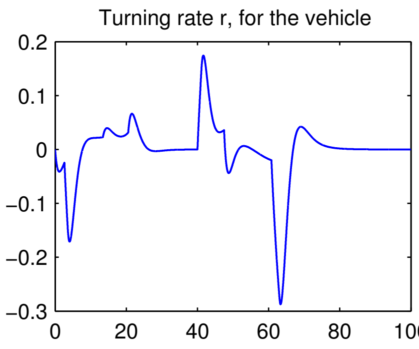


(d) The rudder angle for the vehicle in simulation Figure 7.9

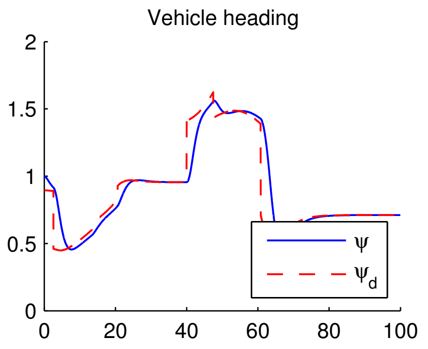
Figure A.5: The internal dynamic for the vehicle in the simulation shown in Figure 7.9



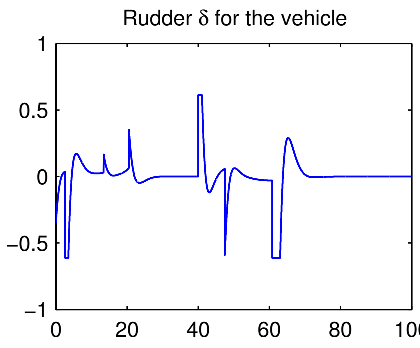
(a) The vehicle speed for the simulation in Figure 7.10



(b) The vehicle turning rate for the simulation in Figure 7.10



(c) The vehicles heading and the steering command from the LOS steering law in the simulation shown in Figure 7.10



(d) The rudder angle for the vehicle in simulation Figure 7.10

Figure A.6: The internal dynamic for the vehicle in the simulation shown in Figure 7.10

Appendix B

Simulations results from target tracking using the model in body

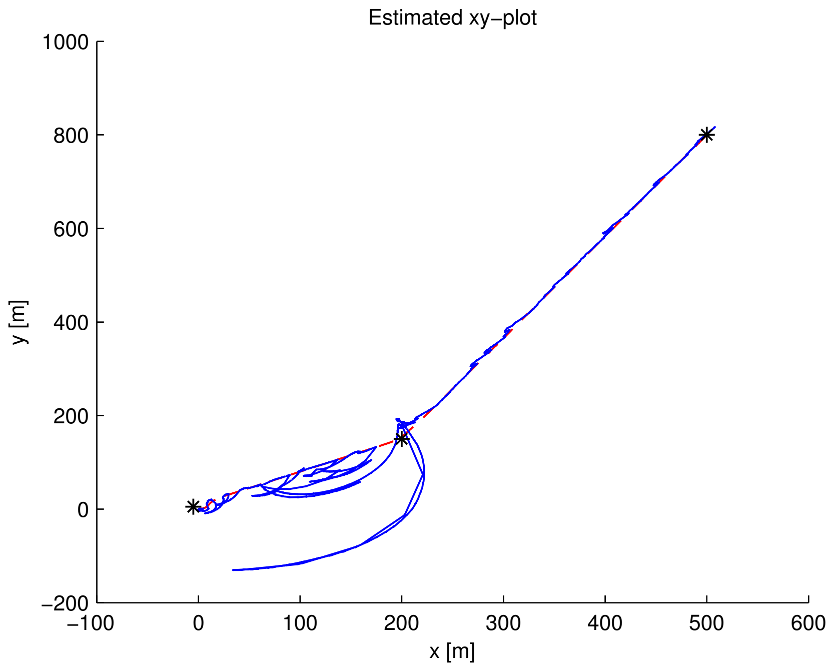
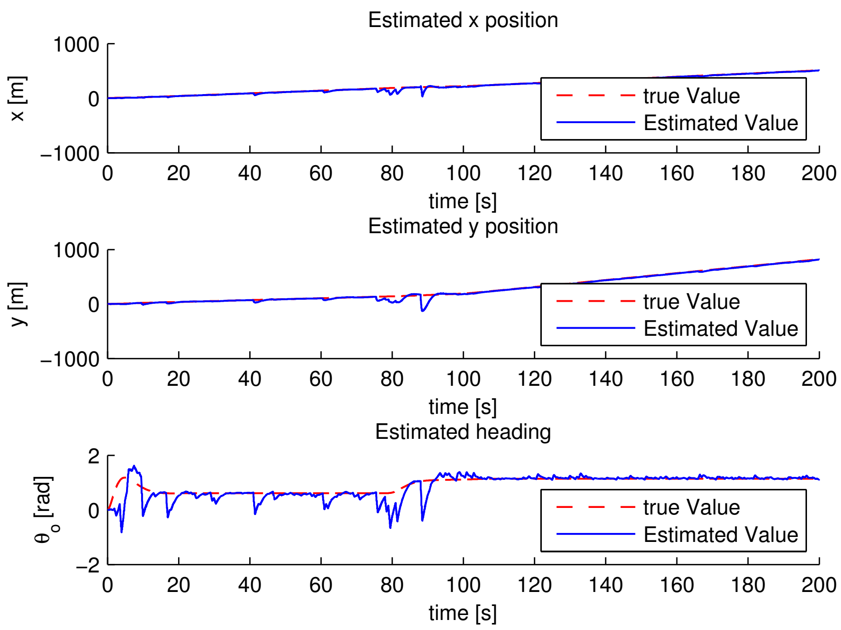
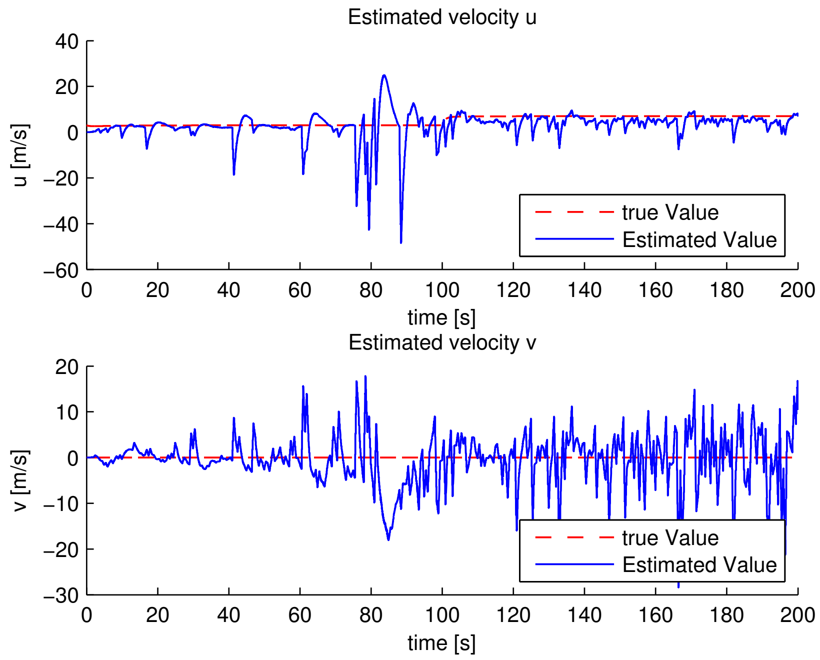


Figure B.1: The estimated position to the object using the model 8.20



(a) The estimated x, y position and the heading to the object using the model 8.20



(b) The estimated velocity to the object using the model 8.20

Figure B.2: The estimated x, y position, the heading and the velocities using the model 8.20

Appendix C

Dubins calculations

Calculation of $d_{obstacleTarget}$, θ_3 and θ_4

$$d_{obstacleTarget} = ||\boldsymbol{p}_o - \boldsymbol{p}_{target}||$$

where $\boldsymbol{p}_o$ and $\boldsymbol{p}_{target}$ are the positions for the obstacle and the target and the vector from the target position and the obstacle is defined as $\boldsymbol{TO}$

$$\boldsymbol{TO} := \boldsymbol{p}_o - \boldsymbol{p}_{target}$$

The length from the target to one of the tangent points is given by

$$d_{t3} = \sqrt{d^2_{obstacleTarget} - ro^2}$$

where ro is the radius for the obstacle and defining θ_4 and θ_4 as

$$\theta_4 := \theta_{toTarget,1} - \theta_{toTarget,2}$$
$$\theta_3 := \theta_{toTarget,1} + \theta_{toTarget,2}$$

where

$$\theta_{toTarget,1} = atan2(\boldsymbol{TO}_y, \boldsymbol{TO}_x)$$
$$\theta_{toTarget,2} = asin(\frac{ro}{d_{obstacleTarget}})$$

Which is the calculation needed to find 4.10 and 4.11.