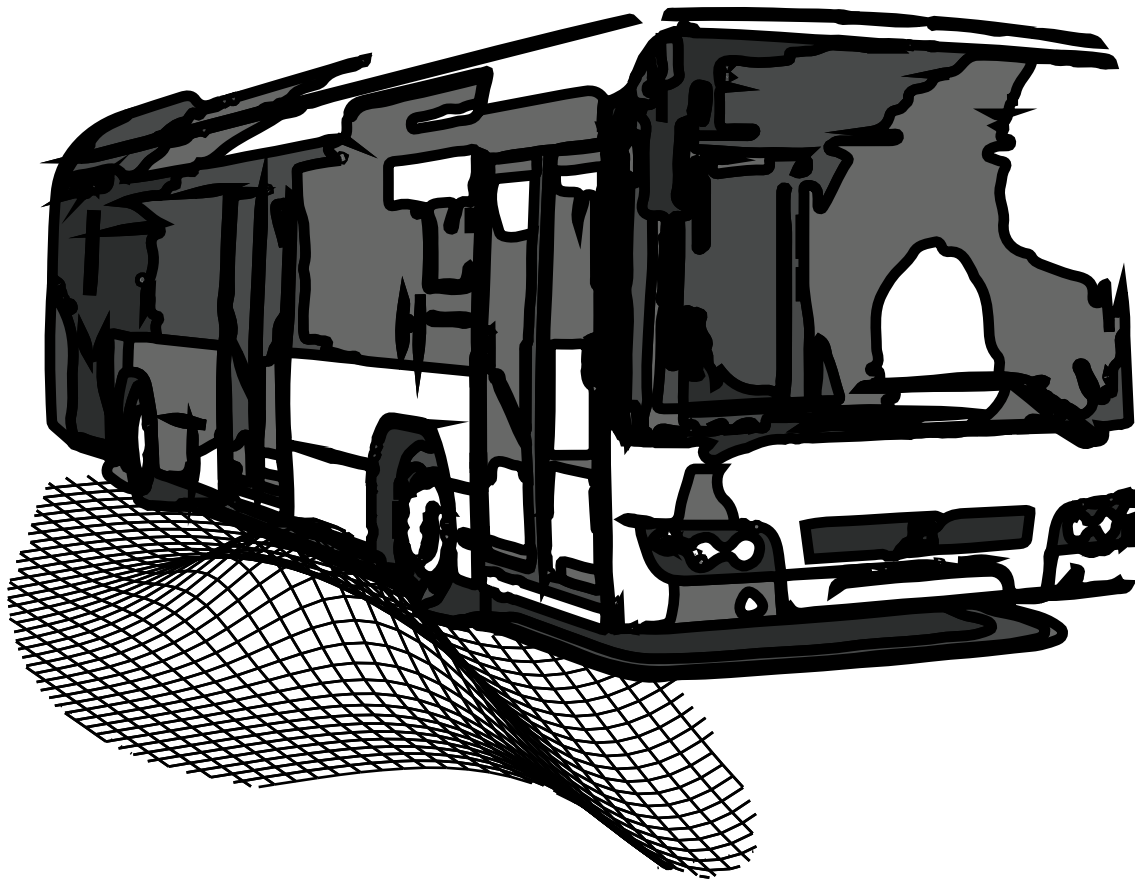


CHALMERS



Assisted Docking of an Electrically Steered Bus

Master's Thesis in Systems, Control & Mechatronics

ROBERT KULL

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2013
Master's Thesis EX048/2013

Assisted Docking of an electrical steered bus
Master's thesis at Systems, Control and Mechatronics
ROBERT KULL
Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
Chalmers University of Technology

Abstract

Public transportation has increased a lot in the last couple of years and in Sweden, buses have made the largest gain. People need to safely get on and off the buses and therefore, it is important that the drivers stop close to the curbs at the bus stops. This causes the drivers to sometimes hit the curb with the front right tyre which damages the bus and become an expense for the bus companies. Especially unexperienced drivers need help to complete the docking without damaging the bus.

This report presents a solution to the problem using an electric motor which gives the driver feedback in the steering wheel by a torque in the direction he/she should steer. Path planning was done using a gradient based minimization in a generated potential field with moving goal and obstacle points. To be able to use the path planning approach, the system needed information about the distance and approach angle to the curb.

Sensors that could give this information were examined. Ultrasonic sensors and computer vision using a single camera was tested practically before finally choosing the ultrasonic sensor as the sensor to use. To be able to detect the curb and not get interference from the asphalt, the mounting height and angle of the ultrasonic sensors were of great importance.

The system was simulated using the mathematical non-slipping single track model. Signal processing were made to filter the distance measurements to the curb. The result of this project is the report which describes what to implement and how to implement it to prevent further damaging of buses at bus stops. The path planning strategy has been validated indoors and outdoors to see that the correct steering wheel angle was generated and the ultrasonic sensors have been tested outdoors against a real curb.

The future development will consist of putting the parts together and test the system on a real bus with the electrical steering.

Keywords: Bus docking, electrical steering, autonomous driving, path planning, potential field, curb detection

Preface

This report has been written as a final result of my M.Sc. degree at Chalmers University of Technology and the project has been carried out at CPAC Systems AB in Gothenburg, Sweden.

CPAC has done a lot of work with the electric motor used in the Volvo FH truck series to control steering but now wants to investigate how this motor could be of use in buses to help the bus drivers.

Acknowledgements

I would like to express my gratitude to my supervisor Knut Åkesson for the useful comments, remarks and engagement through the learning process of this master thesis. I also want to thank my co-workers at CPAC Systems and especially Martin Ryd, Erik Larsson and Magnus De Canesie who were supporting me throughout the project.

Gothenburg September 25, 2013
Robert Kull

Contents

1	Introduction	9
1.1	Purpose of the project	9
1.2	Project restrictions	9
1.3	Project deliveries	10
1.4	Project outline and overview	11
2	Mathematical modeling of a four wheeled vehicle	13
2.1	The full four wheeled vehicle model	13
2.2	The single track (bicycle) model	13
2.3	The final reduced model used for simulations	15
3	Path planning using potential fields	17
3.1	What is a potential field?	17
3.2	The specific potential field equation used for the project	19
3.3	The effect of changing the parameters α , β and γ	20
3.4	Minimization problem to reach the goal	22
3.5	Generating the reference trajectory from the potential field	23
3.6	The role of the potential field in this project	23
4	Simulations of the closed loop behavior	25
4.1	Phase 1	25
4.2	Phase 2	27
4.3	Phase 3	31
4.4	Running the simulation	31
5	Validation	35
5.1	Hardware and software used	35
5.2	The setup rigs used for the validation	35
5.3	Results	37
6	Sensors for robust curb detection	41
6.1	Ultrasonic sensors	41
6.1.1	Theory	41
6.1.2	How many sensors will be needed if one is failing?	43
6.1.3	Testing of the ultrasonic sensor	44
6.1.4	Results of the tests	45
6.2	Monocular vision	47
6.2.1	Theory	47
6.2.2	Testing of monocular vision	48
6.2.3	Results of the tests	50
6.3	Stereo Vision	51
6.3.1	Theory	51
6.3.2	How to implement stereo vision	52
6.4	Laser range finders (LRF's)	53
6.4.1	Theory	53
6.4.2	How a LRF sensor could be implemented	53

6.5	Sensor discussion	54
6.5.1	Which sensor should be used?	55
7	Signal processing and filter design	57
7.1	The median and mean value (averaging) filters	57
7.2	First order low pass filter	57
7.3	The Kalman filter	58
7.3.1	Basic theory of the Kalman filter	58
7.3.2	Implementing the Kalman filter to process the signal	59
7.4	Which filter should be used for the bus docking application?	62
8	Project status	63
8.1	Future work	63
	References	64
A	The MATLAB interface explained	66

List of Figures

1	Bus stopping maneuver split into two parts. The first part is to find the bus stop and steer into it while the other is to end up close to the curb. The driver will only be assisted with the second part of the maneuver.	10
2	Flow chart of how the final system works. This loop is done in every time step to give the driver continuous feedback in the steering wheel.	11
3	The electrical steering system used in the Volvo FH series. The electrical motor can be used to e.g. remove vibrations, make steering easier or automatically steer the vehicle.	12
4	The full single track model for the dynamic motion of a vehicle. Slip angles, lateral tire forces and air resistance are handled by this mathematical model.	14
5	The final model used for computer simulations when simplifications have been made. This is the non-slipping single track model. Because of the low speed, the movement of the vehicle can be expressed by the kinematic expressions in Equations 6-9.	15
6	A potential field created by a function where goal position is the minima and obstacle position is the maxima. A 3D view of a field can be seen to the left and a top view of the same field can be seen to the right. In this field, the goal position has been placed in the lower left corner and an obstacle in the middle. Red color indicates high point and blue indicates low point.	18
7	The path taken to end up at the goal position without colliding with the obstacle. The controlled object follows the negative gradient of the potential field function to end up in the correct position.	19
8	The difference between a higher γ_a (blue line) and a lower (red line). A wider field gives a straighter path to the goal position than a thinner field. The thinner field however makes the bus approach the curb in an earlier stage of the docking.	21
9	The different fields generated by $\gamma_a = 10$ (to the left) and $\gamma_a = 4$ (to the right).	21
10	Simulations made to test two different minimization methods to reach the goal. Newton's method (to the left) gives an unwanted path to the goal because it will always make the bus steer straight against the goal. The bus will then reach the goal faster but with an inappropriate path. The steepest descent method (to the right) makes the bus steer against the curb in the beginning of the docking and then align against the curb. This is a wanted motion and steepest descent should thereby be used.	23
11	The sensor readings and vehicle parameters generates the goal and obstacle position of the potential field. The steering angle at this time step can then be calculated by following the negative gradient of the field.	24
12	Flow chart of the algorithm used for the simulation. The simulations aimed to test if the path planning method made the bus dock in a correct way.	25
13	Location and direction of the simulated sensors. To get the calculation needed to generate the steering angle, two sensors were needed.	26
14	Simulated sensor (to the left) only having one ray pointing in a -90° offset of the heading of the vehicle. A real sensor (to the right) having a beam with an opening angle.	26

15	The approach angle to the curb is calculated by the sensor readings, the positioning of the sensors and geometry of the bus.	27
16	The turning radius of the vehicle is calculated by Ackermann geometry of the steering.	29
17	Simulating a docking situation with a constant speed at 20 <i>km/h</i> and varying γ_a . The plot shows the lateral acceleration for the different fields generated by the different γ_a . The goal is to choose γ_a so that the lateral acceleration never gets outside the comfort zone, marked in the plot as dashed lines. . .	30
18	Initiation of the bus docking simulation. The bus has been initiated with some approach angle to the curb and will continue moving straight forward until both sensors get a reading.	32
19	The activated sensors and the contours of the potential field (to the left). 3D plot of the potential field (to the right). Both sensors have now locked to the curb and the bus is steering according to the negative gradient of the potential field.	33
20	The bus simulation in its final pose. The bus has reached close to the curb and the simulation terminates (upper). In reality, the bus driver will stop when the bus is in position. The wheel angle during the docking can be seen to the lower left. The front sensor measurement to the curb can be seen to the lower right.	34
21	Indoor setups without a curb (left) and with an artificial curb (right). Testing to see that the correct steering angle is calculated in reality.	35
22	The built MATLAB interface used for the test with the car. The left plot of the interface shows the wanted steering direction while the right plot is showing an overview of the vehicle and the angle it has to the curb.	36
23	The sensors mounted on the car for doing more real-life situation tests. . . .	37
24	The pose of the rig (left) and the corresponding MATLAB GUI (right). . .	37
25	The car in Pose one and the corresponding MATLAB GUI.	38
26	The car in Pose two and the corresponding MATLAB GUI.	38
27	Distance calculations because of sensor displacement.	39
28	Ultrasonic sensor form Pepperl Fuchs.	41
29	An example of raw data from an arbitrary ultrasonic sensor. The peaks at t_1 and t_2 are objects detected at two different distances from the sensor. . .	42
30	Usage of deflector with an ultrasonic sensor. A sensor without a deflector emitting sound and its dead zone (to the left). A sensor with a deflector and its dead zone (to the right). The deflector compress the sound and can “remove” the dead zone. The sound will be echoed back the same way as it is emitted.	43
31	The effect of one failing sensor (Sensor 1) when having a system consisting of two, three or four sensors. The lines that can be drawn between each sensor represents the measured curb position. Read below for further explanation. . .	43
32	The rig used for the tests with the ultrasonic sensor. The sensor could be mounted at different heights and different angles for testing the detection range. A fake curb made of wood was built to have a curb of the correct height.	44

33	Screenshot of the graphical user interface of the ultrasonic sensor. Through the interface, thresholds like sonic beam width could be set to see how the result changed.	45
34	Test of the ultrasonic sensor as if vehicle is moving. The fake curb was moved irregularly and sensor data was logged and plotted. The sensor shows very stable results.	46
35	Webcam from Logitech used as camera sensor for the monocular vision test.	47
36	Mounting position of a camera on the bus. By this mounting, the approach angle and distance to the curb could be calculated using a pinhole camera model an geometry.	47
37	Pinhole camera model.	48
38	Original image (top left), grayscale image (top right), filtered image (lower left), Canny image (lower right).	50
39	Lines found in an image using the Hough transformation and Canny edge detector. Too many lines of the same type are found and no significant color can be used to pick out the correct one (the white area does not exist at every bus stop).	51
40	Stereo camera sensor from Point Grey. Two parallel cameras mounted at a distance apart from each other.	52
41	Stereo vision triangulation (reconstruction).	52
42	2D Laser Range Finder from Pepperl Fuchs. LRF sensors are widely used by autonomous prototype vehicles.	53
43	Implementation of a horizontal LRF sensor. The top mounted sensor could detect two reflectors mounted at the bus stop. By triangulation, the position and pose of the bus can be calculated. With this setup, the whole bus stopping maneuver could be done by the system since the bus stop can be detected at a further distance. In combination with a GPS and data base for the outlay of each bus stop, precise docking can be made. A top mounted LRF can also be used in other autonomous bus applications and active safety solutions.	54
44	Implementation of a vertical LRF sensor. A scan is made in the vertical plane at the side of the bus. The curb can then be detected as a L-shaped figure in the raw data from the sensor.	54
45	Tests of first order low pass filter (or moving average filter) with varying parameter δ	58
46	Kalman filter with constant R and varying Q . High Q (to the left) and low Q (to the right).	60
47	Kalman filter with Q held constant and varying R . High R (to the left) and low R (to the right).	60
48	The movement of the sensors depending on the steering wheel angle. Since the Ackermann steering geometry is assumed, every point of the bus will rotate around the same point and the sensor movement can thereby be calculated using geometry.	61
49	Simulations of the Kalman filter using information about the movement of the sensors.	62
50	The MATLAB interface at start up.	66

51	MATLAB interface initializing.	67
52	MATLAB interface initialized and ready.	67
53	MATLAB interface when no curb has yet been found.	68
54	MATLAB interface when the curb has been found.	68
55	MATLAB interface, closer to the curb.	69
56	MATLAB interface when the wanted position and pose has been reached. .	69

1 Introduction

Public transportation is used more today than ever as more people are aware of the environmental damage caused by personal vehicles. In Sweden, transportation by bus has the biggest increase in the last couple of years [1].

Local buses driven in cities passes a lot of bus stops to leave and pick up passengers. The driver needs to stop the bus as close as possible to the curb for everyone to be able to get on and to minimize the risk of injury. The width of the vehicle and the positioning of the driver makes it hard to end up close. To do this maneuver, the driver sometimes lets the front right tire hit the curb and roll against it the last meters to ensure that the bus is in the right position. This damages mostly the tire, but also the suspension, and reduces the time it can be used. When the tire is worn out, the bus needs to be taken out of traffic and repaired. This is a great expense for transport companies like e.g. Veolia¹, that should be removed if possible.

Although this expense is of great importance, it is even more important that everyone is able to use the public transportation in a safe way. For older people and people with certain handicaps, getting on and off the bus is a greater deal and the risk of injury increases when the bus is further away [2].

Systems that, more or less, automatically parks cars have been available for some years and the first commercial system was released by Toyota² in 2003. Since then, more companies have also introduced self parking systems in their cars. These systems often helps the driver find a parking spot big enough for the car and then, from a command by the driver, steers the car into the spot with the driver being in control of gas and brakes only.

There also exists assistance systems for parking cars. These systems are less complicated and do not control any actuators of the car. Two examples are the parking sensors and obstacle warning sensors [3]. The assistance systems alerts the driver when being too close to another vehicle or obstacle with sensors mounted externally on the car. Although the market has already introduced these active safety features for cars, buses have not yet been equipped with such systems.

Buses need active safety solutions to make driving both simpler and more safe for the passengers. An assisting parking system would be such a solution. The application would help the driver come as close to the curb as possible without hitting it and save a lot of tires and reparation time for the bus companies.

1.1 Purpose of the project

Examine if a technical solution, that will eliminate the wear on the tires at bus stops, exists and can be commercially built. Also to identify potential challenges in the design of such a product.

1.2 Project restrictions

To be able to carry out this project within the limited time, some restrictions were made for the project:

¹Veolia is one of the transport companies responsible for and operating buses in Gothenburg, see www.veolia-transport.se for more information.

²Toyota introduces self parking car, <http://edition.cnn.com/2003/TECH/ptech/09/01/toyota.prius.reut/index.html>.

1. The bus stop maneuver was split into two parts (shown in Figure 1). The first part was finding the bus stop and steer the vehicle into that space. The second part was to steer the bus close and align it along the curb. The second part was handled by the system, i.e. the first part of the maneuver was left completely to the driver. When the bus is steered into the space, the system will be assisting the driver to complete the docking. This restriction simplified the finding of the curb since the vehicle will be close to the curb at all times when the system is active.

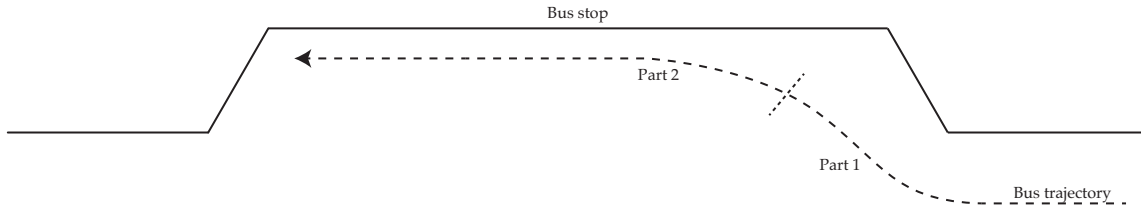


Figure 1: Bus stopping maneuver split into two parts. The first part is to find the bus stop and steer into it while the other is to end up close to the curb. The driver will only be assisted with the second part of the maneuver.

2. The driver controls gas and brakes completely, he/she will only be helped with the steering. If the system would be in control of gas and brakes as well, a lot more safety issues must be considered. The driver must still be aware of the situation and be able to brake if, for example, a person would step out in the trajectory of the bus.
3. The system must be designed in a way that meet the requirements while not being too expensive. When creating such a system, design difficulties appear and compromises needs to be made to complete the project. The system only needs to be “good enough” to meet the expected performance while being as cheap as possible.

1.3 Project deliveries

The project resulted in the following main deliveries:

- A detailed report containing the mathematical model, a path planning strategy, an algorithm for the implementation and a signal processing chapter containing information how the signal from the sensors can be filtered. The report can be seen as a guideline for how the system can be implemented and how the problem can be solved.
- A deeper investigation of different sensor types, amount of sensors and sensor placements. This were mainly a theoretical investigation due to the limited amount of time.
- An investigation and identification of important design difficulties of a commercially applicable system. This delivery were of great interest since the result should be commercially available in the future.

1.4 Project outline and overview

The project was split up in some main steps that were followed to reach the result. This part will roughly explain what had to be done to meet the deliveries stated in the previous part.

A mathematical model of the vehicle motion were created to carry out computer simulations of the system. When the mathematical model was created, stepwise simplifications according to reasonable assumptions were made which led to simplified calculations [4].

Path planning was made by using a potential field approach where the information about the bus position and pose were used to calculate the steering angle of the wheels. The main idea was to generate a steering angle in every time step depending on where the vehicle should end up (the goal position) and where it should not (the obstacle position). The potential field approach were used because of its simplicity and flexibility to change when the scenario changes.

When all the equations of the mathematical model and the path planning were at hand, the bus stoping maneuver was simulated using MATLAB [5]. At this stage, the sensors measuring the distance to the curb was ideal (i.e. noise free) and of unknown type. A rough overview of what the algorithm looked like is presented as a flow chart in Figure 2.

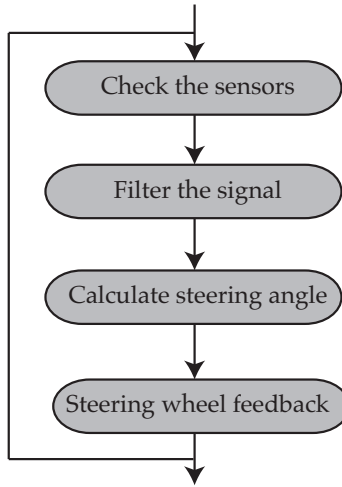


Figure 2: Flow chart of how the final system works. This loop is done in every time step to give the driver continuous feedback in the steering wheel.

The practical implementations were done in several steps, starting with system validation to see that the path planning method gave the correct steering angles. These validation tests were done indoors and with a car provided by CPAC. A computer interface was created using MATLAB to give the driver instructions of how he/she should steer to dock the vehicle. For the validation, cheaper ultrasonic sensors were used.

After the validation, sensors that could give the correct information were chosen. A theoretical study was made of the available sensors that could solve the problem and real tests were made with ultrasonic sensors and a single camera vision sensor.

Simulated noisy sensors were processed to assure that the correct signal could be sent to the electric motor. One of the filters that was tried out was the Kalman filter, where the model of the system was used in combination with the noisy simulated sensors to give

a more stable measurement to the curb.

The electrical steering system consists of an electric motor that is creating a torque in the rod attached to the steering wheel (see Figure 3). It is the same technology used in the Volvo FH³.



Figure 3: The electrical steering system used in the Volvo FH series. The electrical motor can be used to e.g. remove vibrations, make steering easier or automatically steer the vehicle.

The motor gives the driver feedback in the steering wheel by generating a torque in the direction he should move the wheels. This can be compared to the lane cruising systems used in some cars. But instead of following a lane, the bus will guide the driver to follow the trajectory to make the docking as wanted.

³See <http://www.volvotrucks.com/trucks/global/en-gb/newsmedia/pressreleases/Pages/pressreleases.aspx?pubId=14852> and video at <http://www.youtube.com/watch?v=pn6dwyUqvA8>.

2 Mathematical modeling of a four wheeled vehicle

To make simulations for the path planning and to try out different filtering techniques for the sensors, a mathematical model of the vehicle motion was needed.

The assumptions that can be made to go from a full four wheeled vehicle model to a simplified mathematical model that were used for this project are presented in the following parts. Although a mathematical model never tell the whole truth about the system they do save a lot of time and money since different theories can be tested in a computer before an implementation is made in real-life.

2.1 The full four wheeled vehicle model

A complete four wheeled vehicle model with three degrees of freedom as presented by Pacejka [6] is very detailed. The model contains pitch, roll and yaw but the only motion of interest for the system to dock is yaw. For more information about the four wheeled model, please read [6].

2.2 The single track (bicycle) model

By assuming that the vehicle is symmetric and that no pitch or roll but only yaw motion is present, along with the Ackermann steering geometry [7], the four wheeled model can be simplified to a single track model [8]. The assumption holds if the vehicle is driven at low speeds which was the case for this project. The vehicle will always be in a docking (parking) situation when the system is active and low speeds ($v(t) \leq 5 \text{ m/s}$) will thereby be present. In Gothenburg, bus drivers should never drive faster then 15 km/h ($= 4.17 \text{ m/s}$) around bus stops [9]. This assumption led to further simplifications later on.

This mathematical model is now a bicycle without pitch and roll motion. When the single track model is applied to a four wheeled vehicle, the front wheels and the rear wheels have been lumped together. Figure 4, from [10], shows the full single track model with all its variables.

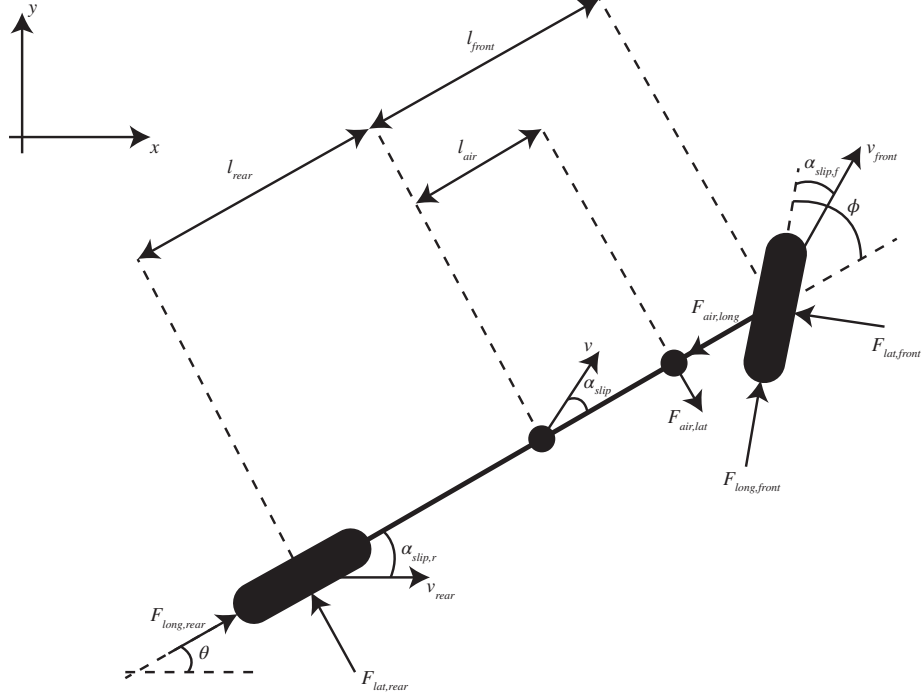


Figure 4: The full single track model for the dynamic motion of a vehicle. Slip angles, lateral tire forces and air resistance are handled by this mathematical model.

All the equations and explanation of Figure 4 can be found in [10] and [11]. The equations of motion for this model is;

$$\dot{x}(t) = v(t) \cos(\theta(t) - \alpha_{slip}(t)) \quad (1)$$

$$\dot{y}(t) = v(t) \sin(\theta(t) - \alpha_{slip}(t)) \quad (2)$$

$$\begin{aligned} \dot{v}(t) = \frac{1}{m} & ((F_{long,rear} - F_{air,long}) \cos \alpha_{slip}(t) + F_{long,front} \cos(\phi(t) + \alpha_{slip}(t)) - \\ & - (F_{lat,rear} - F_{air,lat}) \sin \alpha_{slip}(t) - F_{lat,front} \sin(\phi(t) + \alpha_{slip}(t))) \end{aligned} \quad (3)$$

$$\begin{aligned} \dot{\alpha}_{slip}(t) = \dot{\theta}(t) - \frac{1}{mv(t)} & ((F_{long,rear} - F_{air,long}) \sin \alpha_{slip}(t) + F_{long,front} \sin(\phi(t) + \alpha_{slip}(t)) + \\ & + (F_{lat,rear} - F_{air,lat}) \cos \alpha_{slip}(t) + F_{lat,front} \cos(\phi(t) + \alpha_{slip}(t))) \end{aligned} \quad (4)$$

$$\ddot{\theta}(t) = \frac{1}{I_{zz}} (F_{lat,front} l_{front} \cos \phi(t) - F_{lat,rear} l_{rear} - F_{air,lat} l_{air} + F_{long,front} \sin \phi(t)) \quad (5)$$

The model has slip angles and lateral tyre forces which come from the fact that the wheels do not roll without slipping against the surface. It also handles air resistance. From now on the time dependent variables will not be expressed by the (t) for simplicity.

Further simplifications will be made in the next part.

2.3 The final reduced model used for simulations

Since the assumption of low speeds were made in the previous part, it will still hold. This, along with an assumption of good road conditions, i.e. high friction between the wheels and the road, will make the wheels roll without slip angles [12], [8].

The slip angles α_{slip} , $\alpha_{slip,f}$ and $\alpha_{slip,r}$ will all be equal to zero. This will make Equations 1 and 2 result in Equations 6 and 7. Also, the lateral tyre forces will disappear because of the front and rear slip angles being zero [10]. The air resistance will also be very low and therefore assumed to be 0.

The single track model can now be simplified even further. Figure 5 introduces the notation to the model.

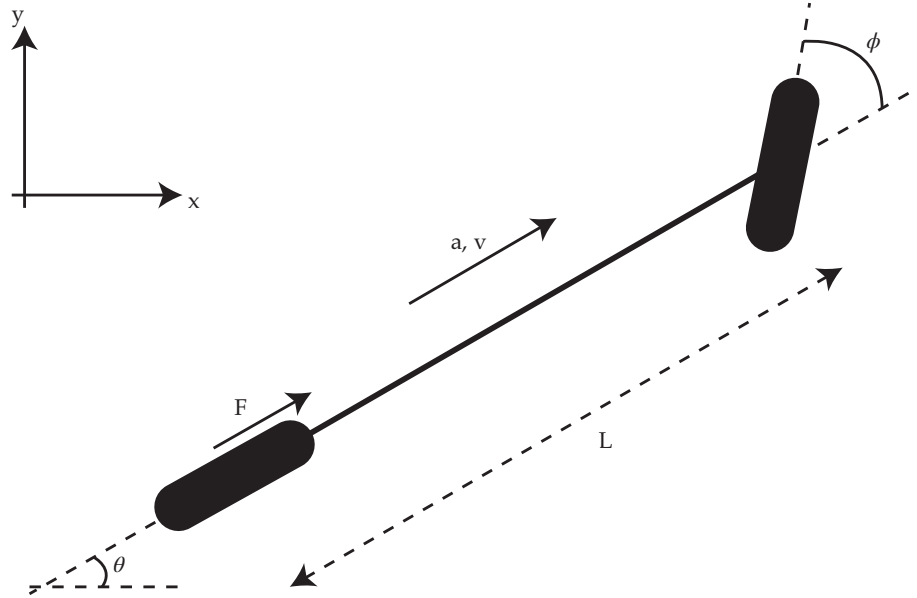


Figure 5: The final model used for computer simulations when simplifications have been made. This is the non-slipping single track model. Because of the low speed, the movement of the vehicle can be expressed by the kinematic expressions in Equations 6-9.

The dynamics of the non-slipping single-track vehicle model is described by the differential Equations 6-9;

$$\dot{x} = v \cos \theta \quad (6)$$

$$\dot{y} = v \sin \theta \quad (7)$$

$$\dot{\theta} = v \frac{\tan \phi}{L} \quad (8)$$

$$a = \dot{v} = \frac{F}{m} \quad (9)$$

where x and y is the position of the rear wheel, v is the absolute speed, θ is the heading and ϕ is the steering angle of the vehicle at time t . The constant L is the length between

the front and rear axis. Equation 9 describes how the vehicle acceleration is dependent on the force created by the torque from the driving wheel.

Note that this model might be very simple, but it is very applicable for a lot of vehicles since so few parameters are needed [13]. The input signal is the steering wheel angle ϕ and the output of the system is the heading θ and the position $[x, y]$ of the vehicle. The force F is controlled by the driver and thereby seen as a disturbance. As seen in Equation 9, the acceleration \dot{v} is directly dependent on this disturbance F . Since $a(t) = \dot{v}(t) = \frac{d}{dt}v(t)$, also the absolute speed of the vehicle is dependent of the disturbance created by the driver and can not be controlled, it can however be measured. Equations 6-8 describes how the heading angular velocity $\dot{\theta}$ is dependent of the absolute speed v and the control signal ϕ . The velocities $[V_x, V_y] = [\dot{x}, \dot{y}]$, is in turn dependent on the heading of the vehicle.

How to choose the control signal ϕ is the problem. The vehicle should follow a specific trajectory to be aligned along the curb in its final pose. This is a navigation problem and a quite effective approach to do this is to generate a potential (artificial) field, explained in the next chapter. Another design constraint is that the lateral acceleration of the vehicle can not be too big. If it exceeds some comfortable constant c_{latacc} the ride will not be comfortable for the passengers. This limitation will lead to constraints on the steering wheel angle (the control signal) depending on the speed of the vehicle.

3 Path planning using potential fields

Path planning is the method of deciding what way to go from point A to B. From the positioning and speed of the vehicle, a reference steering direction needs to be constructed in such a way that the vehicle could follow a certain trajectory. The navigation must also be made in such a way that the bus avoids hitting the curb.

A bus is non-holonomic⁴ and the whole path from point A to point B must thereby be planned for the bus to end up in the correct position and pose. There exists a variety of path planning algorithms that could be used and some are more complicated than others. In the field of autonomous robots, grid-based methods like “Best-first search”, “Dijkstra’s” or the “A*” algorithms are quite popular. These methods require that the controlled object generates an artificial grid in its environment and then navigates through the nodes in the grid. A non-grid-based method that is also popular is the potential field method (see below). Model Predictive Control (MPC) could also be used to generate the path that should be followed by the vehicle. MPC is an optimal control strategy used when constraints are present. For more path planning strategies, see [15], [16].

For this project, the potential field method were used because it is simple and flexible to changes in the scenario. It has also already been studied for lane keeping by cars, see [17].

3.1 What is a potential field?

The easiest way to describe the potential field method is to imagine a landscape with peaks and dips where if you drop a ball, it will roll down to the lowest point. An artificial field is set up in the “brain” of the robot which looks pretty much like the landscape that was just described. Obstacles will become peaks and goals will become dips. Artificial forces will act on the object controlled pushing it to reach to a lower point in its surrounding [18]. Figure 6 shows a potential field graphically. Note that if you would place a ball in the upper right corner it will roll down to the lower left corner without colliding with the obstacle placed in the middle.

⁴A non-holonomic system has constrained degrees of freedom. The path the vehicle can take from a given position and pose depends on the previous path taken and in this case, e.g. the maximum steering angle [14].

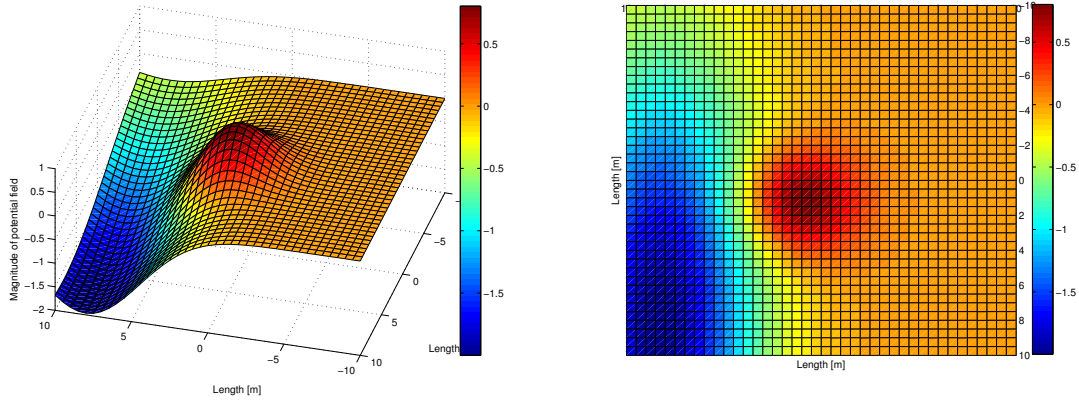


Figure 6: A potential field created by a function where goal position is the minima and obstacle position is the maxima. A 3D view of a field can be seen to the left and a top view of the same field can be seen to the right. In this field, the goal position has been placed in the lower left corner and an obstacle in the middle. Red color indicates high point and blue indicates low point.

There are many ways to setup a potential field and many equations can be used to build the peaks and dips. One thing to be aware of when setting up the field is the existence of local minimas where the controlled object could get stuck, thinking that it is in the goal position [18]. This were however not an issue for this project since there will only exist one minima which the bus will try to reach.

Back to the potential field in figure 6, a ball is dropped and its path is shown in Figure 7.

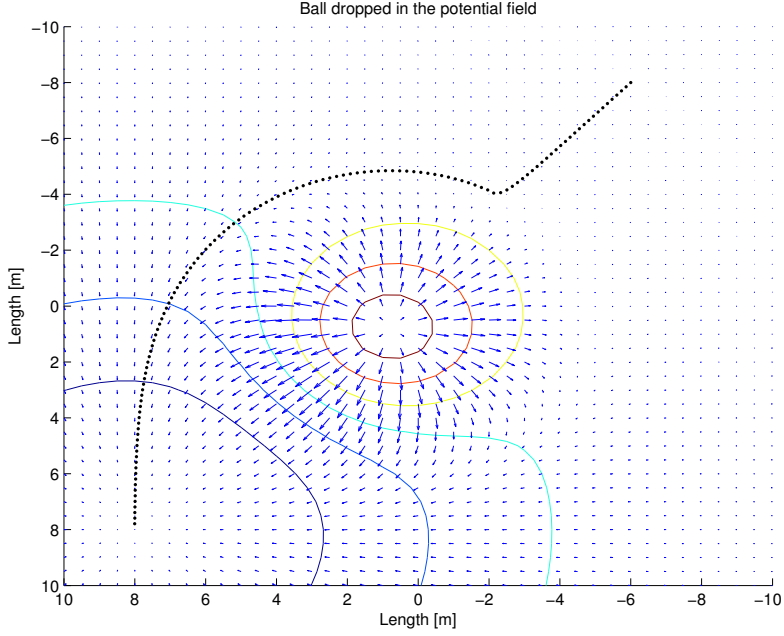


Figure 7: The path taken to end up at the goal position without colliding with the obstacle. The controlled object follows the negative gradient of the potential field function to end up in the correct position.

The contours of the field have been plotted along with the negative gradients of the field. These negative gradients are the main idea of how the potential field works. If the controlled object always follow the negative gradient of the field, it will eventually end up in the goal position (if not in a local minima).

The goal position is given one attractive field while the obstacles are given repulsive fields. One potential can be described by the function;

$$\psi_i(x, y, x_p, y_p, \dots) \quad (10)$$

where (x, y) is the current position of the object being controlled, (x_p, y_p) is the position of the goal/obstacle i . The “...” indicates that more variables could be needed to set up the potential field depending on the equation used.

The complete field is then generated by adding the attractive and the repulsive components according to Equation 11.

$$\Psi(x, y) = \sum_n \psi_n \quad (11)$$

3.2 The specific potential field equation used for the project

The potential field will figure as a generator of the reference steering angle. The equation to set up the potentials is taken from [19];

$$\psi(x, y, x_p, y_p, \alpha, \beta, \gamma) = \alpha e^{-\left(\frac{x-x_p}{\beta}\right)^2 - \left(\frac{y-y_p}{\gamma}\right)^2} \quad (12)$$

where (x, y) and (x_p, y_p) is the same as stated above. The parameter α describes the height of the potential, therefore a positive α is repulsive while a negative α is attractive. The magnitude of α describes how low or high the potential will be.

The last two variables, β and γ , describes the shape of the potential. Different shapes might need to be used for different potentials and by varying β and γ this is possible. β controls the width of the field along the x-axis and γ controls the width along the y-axis.

As mentioned above, other equations can be used to describe potential fields. Equation 12 has been chosen because of the flexibility in creating the potentials. The shapes can be controlled by changing the variables α , β and γ . From now on, the parameters will be divided into attractive and repulsive, e.g. α_a is the α used for the attractive potential and α_r is the α used for the repulsive potential.

3.3 The effect of changing the parameters α , β and γ

By changing the variables α , β and γ the structure of the different potentials can be changed. This is of great importance when the vehicle is driven in different speeds and has altering approach angle to the curb. By changing the variables, the path taken by the vehicle can be controlled in a more efficient way.

If, for example, the vehicle would approach the curb at a higher speed, low steering angles must be present for the passengers to feel comfortable. The vehicle should then not steer too much straight into the curb in the beginning of the path, which would lead to a much tighter turn later on. Instead, the γ_a parameter can be increased giving a wider field and a straighter path to reach the goal. When the speed decreases, the γ_a parameter can be decreased because larger steering angles will be possible. Figure 8 shows this graphically, although at constant speed but the difference between a higher and lower γ_a is clearly visible.

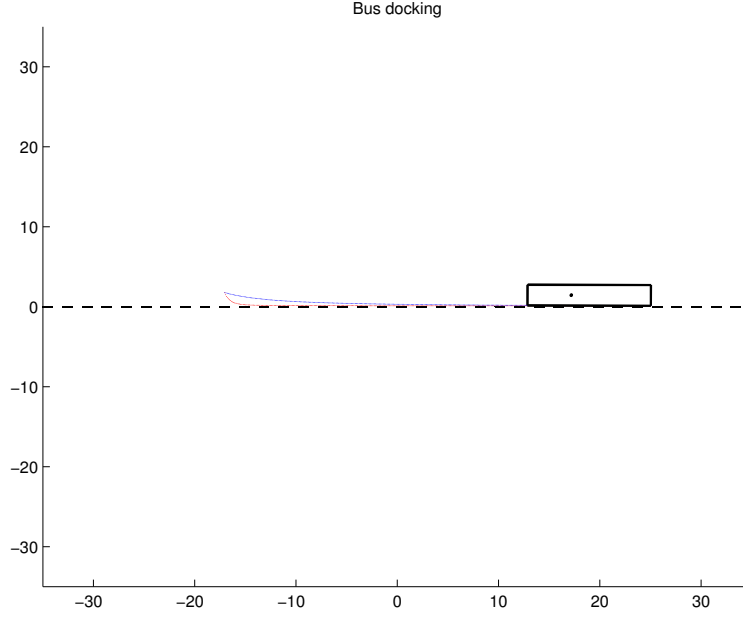


Figure 8: The difference between a higher γ_a (blue line) and a lower (red line). A wider field gives a straighter path to the goal position than a thinner field. The thinner field however makes the bus approach the curb in an earlier stage of the docking.

As can be seen in Figure 8, the lower γ_a makes the bus come closer to the curb in an earlier stage of the docking. This is a wanted motion as the bus should align with the curb as fast as possible but it is only possible at lower speeds because of the higher lateral acceleration. Figure 8 was the result of two simulations, one with $\gamma_a = 10$ and the other with $\gamma_a = 4$. The other four variables α_a , α_r , β_a and β_r were kept the same.

The motion of the vehicle is determined by the characteristics of the potential field. In Figure 9, a snapshot of the two different fields have been taken right in the initiation phase of the docking.

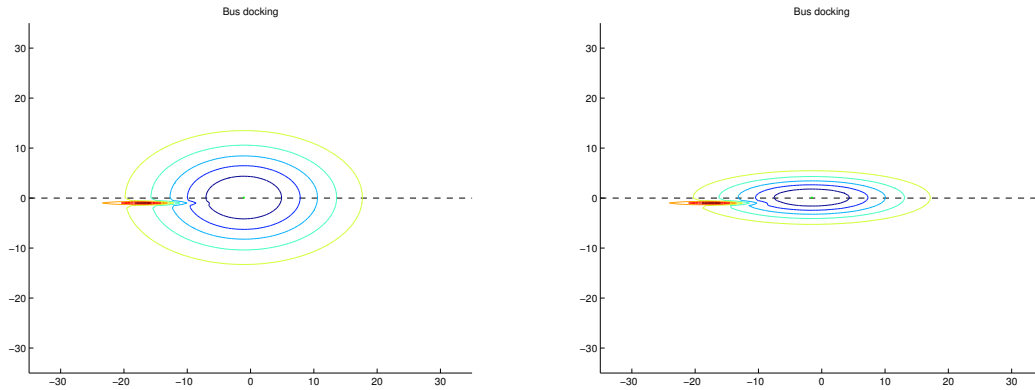


Figure 9: The different fields generated by $\gamma_a = 10$ (to the left) and $\gamma_a = 4$ (to the right).

Note how the left plot in Figure 9 corresponds to the straighter path in Figure 8 and how the shape of the field changes the path of the vehicle.

Through a relationship between the steering angle ϕ and the vehicle speed the lateral acceleration of the vehicle can be computed. From this, the parameter γ_a can be computed in such a way that the lateral acceleration of the vehicle never exceeds the comfort constant c_{latacc} . More about this in the next chapter.

If instead β_a would be changed, the fields would change in the direction of the x-axis. This variable does not control the motion of the vehicle as much as γ_a and will not be varied depending on the speed. Thereby it will be seen as a constant.

As earlier mentioned, α controls the “height” of the field and should be chosen in such a way that the goal will be reached and the obstacle will not be hit. This variable will not be changed either. A good relationship between α_a and α_r could be found through simulations.

3.4 Minimization problem to reach the goal

When the potential field is set up, the direction of descent will be the direction of motion for the vehicle. This means that the vehicle will move away from obstacles while approaching the goal if the field variables are chosen in a correct way. The usage of potential fields is basically a minimization problem, where different methods can be used to reach the minimum. Two well known minimization methods are:

- Steepest descent method [20], which iterates towards the minimum by using the negative gradient of the function;

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \nabla \Psi \quad (13)$$

where $\mathbf{x}_n = [x_n, y_n]$ is the position at iteration n , $\nabla \Psi = (\frac{\partial \Psi}{\partial x_n}, \frac{\partial \Psi}{\partial y_n})$ is the gradient of the function $\Psi(x_n, y_n)$.

- Newton’s method [20], which also uses the inverse Hessian of the function as;

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\mathbf{H}]^{-1} \nabla \Psi \quad (14)$$

In order for the vehicle to be able to steer properly towards the curb, the steepest descent method will be used for this problem. Newton’s method will always point towards the minimum and the vehicle will find the minimum faster but (for this project) in an undesired motion [21].

Also, by simulating this, using steepest descent method and Newton’s method, the difference is clear. Newton’s method finds the direction towards the minimum which is not a wanted solution for the problem with the vehicle and the curb. Figure 10 below shows the problem with using Newton’s method. Newton’s method will also increase the computational load by the inversion of the Hessian. For the response to be as fast as possible, inverting a matrix should whenever possible be avoided.

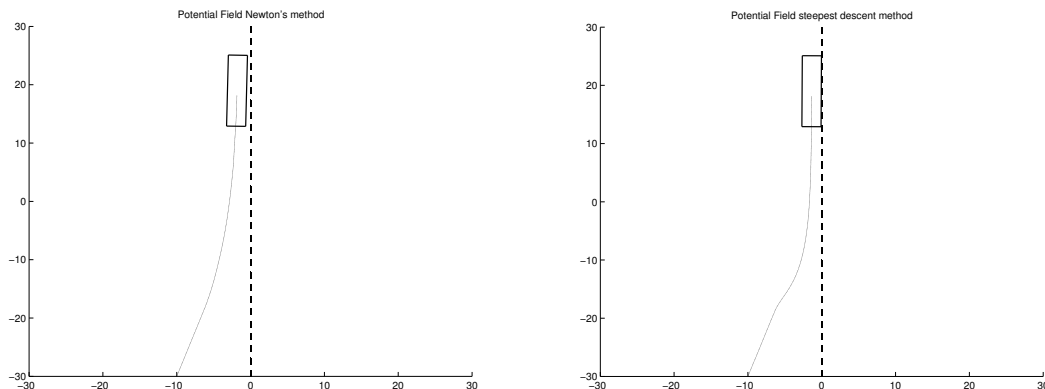


Figure 10: Simulations made to test two different minimization methods to reach the goal. Newton's method (to the left) gives an unwanted path to the goal because it will always make the bus steer straight against the goal. The bus will then reach the goal faster but with an inappropriate path. The steepest descent method (to the right) makes the bus steer against the curb in the beginning of the docking and then align against the curb. This is a wanted motion and steepest descent should thereby be used.

In Figure 10, the dashed black line symbolizes the curb, the blue squares are the vehicle and the blue lines following are the positions of center of rotation of the vehicles. As can be seen, the steepest descent method firstly makes the vehicle approach the curb and then turn, giving it a nice final pose. Newton's method goes straight for the final position which makes the vehicle reach the goal faster but is inappropriate if people wants to enter or get off in the rear part. The usage of Newton's method would also be dangerous in traffic where the rear will be sticking out into the lane.

3.5 Generating the reference trajectory from the potential field

When the minimization method have been chosen to be steepest descent, the direction of motion of the steering angle is generated by the normalized negative gradient of the field as [19];

$$\mathbf{f} = -\frac{\nabla \Psi}{|\nabla \Psi|} \quad (15)$$

where the vector $\mathbf{f} = [f_x, f_y]$ can be seen as a force where the direction will be;

$$\phi' = \tan\left(\frac{f_y}{f_x}\right) \quad (16)$$

3.6 The role of the potential field in this project

The information from the sensors about distance from the curb and the approach angle, along with the current speed of the vehicle will generate the new steering angle for the vehicle, see Figure 11.

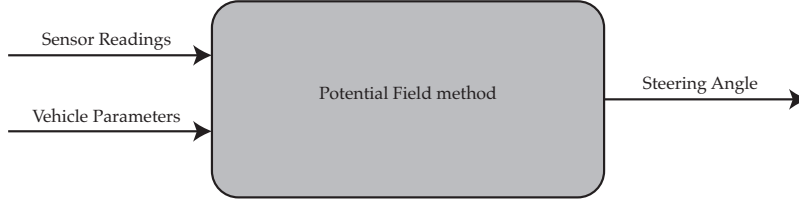


Figure 11: The sensor readings and vehicle parameters generates the goal and obstacle position of the potential field. The steering angle at this time step can then be calculated by following the negative gradient of the field.

In every time step, a potential field will be set up, i.e. a desired path for the wheel by different steering angles. Sensor readings will determine the distance and the approach angle (vehicle heading) to the curb. From these known data, the position of the goal point and the obstacle is calculated and used to set up the field and the reference steering angle for this time step can be set.

4 Simulations of the closed loop behavior

For the computer implementation, a simulator was setup using MATLAB. The simulator consists of different parts which was tested one by one before the complete simulator was set up. Starting with the motion of a vehicle, to the implementation of potential field and sensors.

The algorithm used are presented as a flow chart with different phases in Figure 12 and an explanation of each phase can be found below. The simulations were made with ideal sensor readings, i.e. no noise were added at this stage.

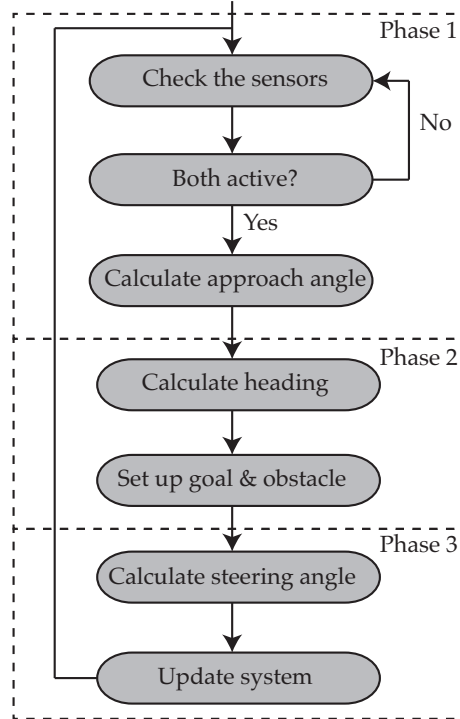


Figure 12: Flow chart of the algorithm used for the simulation. The simulations aimed to test if the path planning method made the bus dock in a correct way.

This algorithm is used until the difference between the sensor readings are below some break criterion specified by the user.

4.1 Phase 1

In the simulations, the vehicle uses two sensors located at the right side shown in Figure 13. This setup is used to be able to get an approach angle from the sensor readings. In real-life, the sensors might not use this technique but the simulated system will do so. The global position and heading of the sensors can be calculated using simple geometry if the position and heading of the vehicle is known.

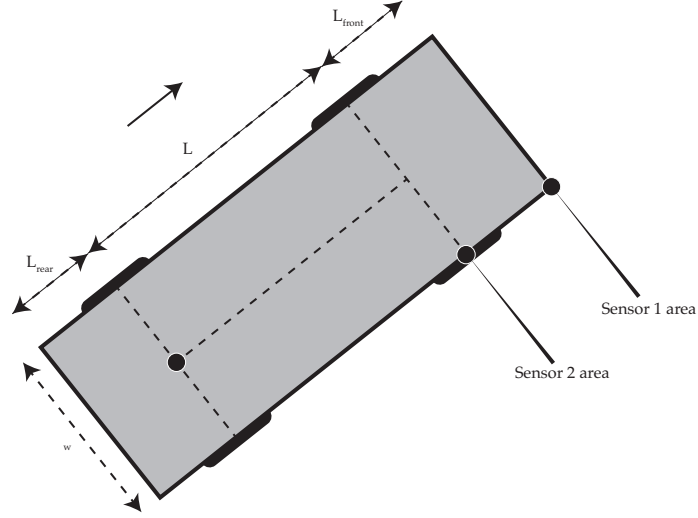


Figure 13: Location and direction of the simulated sensors. To get the calculation needed to generate the steering angle, two sensors were needed.

The sensors are simulated using the same technique as in [19] but with only one ray for each sensor pointing in a offset of 90° related to the heading of the vehicle. This is an approximation of real-life since a real sensor would have some opening angle σ (visualized in Figure 14).

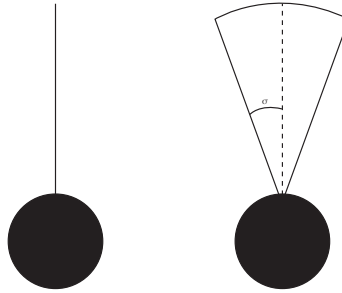


Figure 14: Simulated sensor (to the left) only having one ray pointing in a -90° offset of the heading of the vehicle. A real sensor (to the right) having a beam with an opening angle.

When both sensors gives a reading, the system activates by a command from the driver. Instead of simulating a driver command, the system takes control of the steering as soon as both sensors are active.

The approach angle is calculated using the measured data from the sensors. As the sensors are simulated like one ray pointing in a 90° offset from the heading of the bus some geometry needs to be done. Figure 15 shows how the approach angle is calculated.

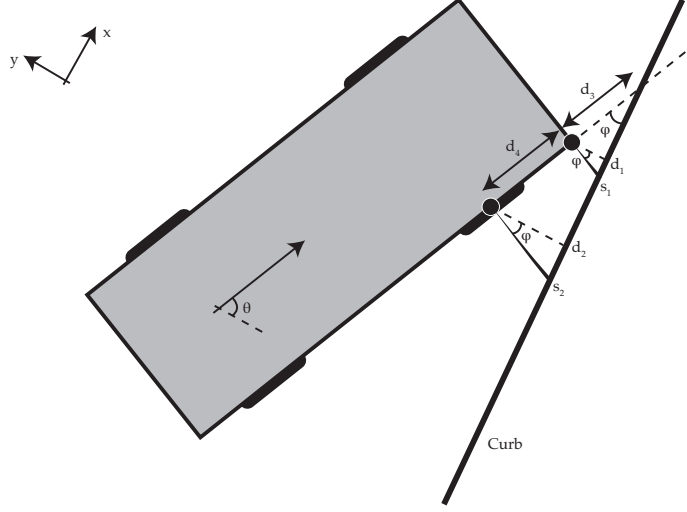


Figure 15: The approach angle to the curb is calculated by the sensor readings, the positioning of the sensors and geometry of the bus.

In Figure 15, φ is the approach angle, s_1 and s_2 are the sensor readings from sensor 1 and 2, d_1 and d_2 are the closest distance to the curb from sensor 1 and 2. The length d_3 is the distance from sensor 1 to the point of the curb where the bus would hit if it, from this pose, kept on moving straight forward. d_4 is the length between the sensors.

The approach angle φ is calculated as;

$$\varphi = \arctan\left(\frac{s_2}{d_3}\right) \quad (17)$$

where

$$d_3 = \frac{d_4}{1 - \frac{s_1}{s_2}} \quad (18)$$

4.2 Phase 2

The the global heading is then;

$$\theta = -\varphi. \quad (19)$$

The goal and obstacle positions are set up using the knowledge gained in the previous step. These positions are calculated with respect to the front right corner of the vehicle which position is known in the global coordinate system. In the simulation, this position is the same as the position of sensor 1.

Firstly, the actual distance d_1 is calculated as;

$$d_1 = \cos(\varphi)s_1 \quad (20)$$

Note again that this is only done in the simulations of the real-life situation. The position of the obstacle is then calculated as;

$$[x_{obstacle}, y_{obstacle}] = [x_{sensor1}, y_{sensor1} - d_1 - \Delta_{obstacle}] \quad (21)$$

where $\Delta_{obstacle}$ is a margin related to γ_r of the obstacle field. A tuning parameter that in the simulations have been tuned as $\Delta_{obstacle} = 1$.

The goal position is calculated in the same way as;

$$[x_{goal}, y_{goal}] = [x_{sensor1} + \Delta_{goal}, y_{sensor1} - d_1 + \xi] \quad (22)$$

where ξ is how far from the curb the vehicle should end up and Δ_{goal} is how far ahead the goal should be. Δ_{goal} is related to β_a and these two parameters also needs to be tuned to get a neat behavior of the vehicle. In the simulations, this parameter has been set to $\Delta_{goal} = 15$.

As for the parameters of the potential field, the variable γ_a controls the width of the field around the goal position in the direction of the y-axis. A wider field gives a smoother path while a thinner field will give harder turns but also a faster docking of the vehicle. It is the comfort of the ride, i.e. the lateral acceleration of the vehicle, that controls γ_a . By [22], humans judged that the lateral acceleration should be in the range $(0.06 - 0.22)g$ where $g = 9.82 \text{ m/s}^2$ to be comfortable. The mean value of this will be used as a “comfort constant” c_{latacc} , which should not be compromised during the docking.

$$c_{latacc} = \frac{(0.06 + 0.22)g}{2} = 1.3748 \text{ m/s}^2 \quad (23)$$

The total lateral force from the tires of the vehicle is then given by Equation 24 as; [12]

$$F_{lat} = \frac{mv^2}{r} \quad (24)$$

where r is the turning radius of the vehicle which relation to the steering angle ϕ is given by Equation 25;

$$r = L \cot(\phi) \quad (25)$$

where w is the width of the vehicle. Off course, the turning radius will be different in different parts of the vehicle but this is the turning radius of the point located at the center of the rear wheel axis, seen in Figure 16. This can be seen as a mean value of the lowest turning radius.

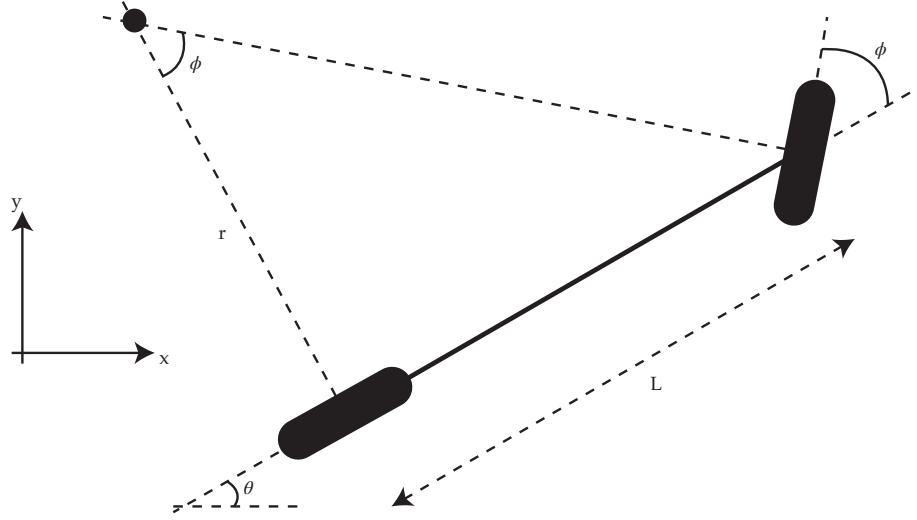


Figure 16: The turning radius of the vehicle is calculated by Ackermann geometry of the steering.

By using Newton's second law and combining Equation 24 and 25 the lateral acceleration a_{lat} is given by;

$$F_{lat} = ma_{lat} = \frac{mv^2}{L \cot(\phi)} \Rightarrow a_{lat} = \frac{v^2}{L \cot(\phi)} \quad (26)$$

This lateral acceleration is depending on the speed of the vehicle and the steering angle ϕ which, in turn, is depending on the potential field parameter γ_a .

By simulating the docking at a constant maximum speed of $v = 20 \text{ km/h}$ and varying γ_a , the lateral acceleration can be calculated using Equation 26 and plotted along with the comfort constraint as in Figure 17.

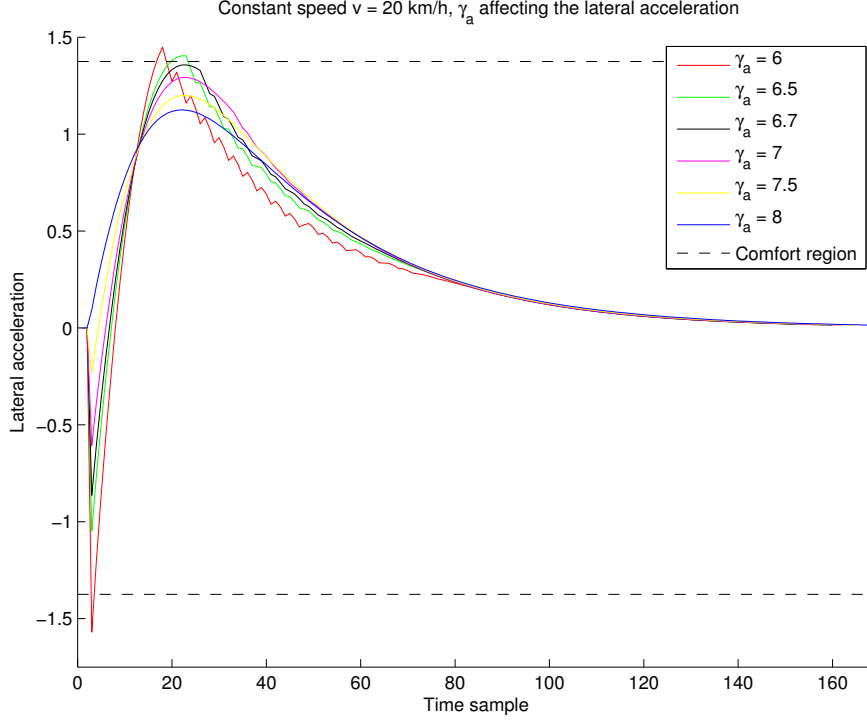


Figure 17: Simulating a docking situation with a constant speed at 20 km/h and varying γ_a . The plot shows the lateral acceleration for the different fields generated by the different γ_a . The goal is to choose γ_a so that the lateral acceleration never gets outside the comfort zone, marked in the plot as dashed lines.

As can be seen in Figure 17, the lateral acceleration constraint is violated for $\gamma_a < 6.7$. By this, γ_a has been chosen to be 7 for a speed of 5.556 m/s (20 km/h). With this tuning, the system will not violate the lateral acceleration constraint and there will be a safety margin if the vehicle would be driving faster than 20 km/h .

In Figure 17, one can also see that there is discontinuities in the lateral acceleration, which would lead to an infinite jerk (time derivative of the acceleration). This is because the simulated system is very simplified and lack dynamics. But since the purpose of the docking system only is to guide the driver of how he/she should turn, this will not be a problem.

For the lowest value of γ_a , simulations was also made but the lateral acceleration could not be checked since low speeds will lead to very low lateral acceleration because of the squared speed in Equation 26. The γ_a parameter was set to 2 for speeds at $v \leq 1 \text{ m/s}$ which gave good performance and not too high steering angles. When γ_a was set lower it affected the system in a negative way and the goal point could not be reached.

The relationship between the vehicle speed v and the parameter γ_a could then be set up as;

$$\gamma_a = v(t)K_\gamma + \gamma_{min} \quad (27)$$

Where the proportional constant $K_\gamma = \frac{\gamma_{max} - \gamma_{min}}{v_{max} - v_{min}} = \frac{5}{4.5556} = 1.1$ and $\gamma_{min} = 2$.

Through simulations, also the other parameters were set up and tuned. The values of all parameters can be seen in Table 1.

Table 1: The parameters of the potential field. All parameters except γ_a has been set through simulations and tests of the bus behavior during docking. γ_a is dependent on the speed to have a lateral acceleration within the comfort zone.

Parameter	Value
α_a	-3
α_r	2
β_a	14
β_r	5
γ_a	$v(t)K_\gamma + \gamma_{min}$
γ_r	0.3

4.3 Phase 3

Since the angle ϕ' is a global angle and the steering angle ϕ is related to the heading θ of the vehicle, the steering angle ϕ is calculated as;

$$\phi = \phi' - \theta \quad (28)$$

and the heading and position of the vehicle is then updated using Equations 6-8 and the forward Euler difference approximation;

$$\dot{\theta}(t) \approx \frac{\theta(t + dT) - \theta(t)}{dT} \approx v(t) \frac{\tan \phi(t)}{L} \implies \theta(t + dT) \approx \theta(t) + v(t) \frac{\tan \phi(t)}{L} dT \quad (29)$$

$$\dot{x}(t) \approx \frac{x(t + dT) - x(t)}{dT} \approx v(t) \cos \theta(t) \implies x(t + dT) \approx x(t) + v(t) \cos \theta(t) dT \quad (30)$$

$$\dot{y}(t) \approx \frac{y(t + dT) - y(t)}{dT} \approx v(t) \sin \theta(t) \implies y(t + dT) \approx y(t) + v(t) \sin \theta(t) dT \quad (31)$$

The loop then repeats itself until some break criterion is fulfilled. Off course, in real-life, the driver is the one controlling when the vehicle should stop.

4.4 Running the simulation

With the algorithm set up as described in 4.1-4.3, the simulations are initialized by setting the heading $\theta_0 = -\pi/8 \text{ rad}$ and the absolute speed v_0 of the vehicle. The speed is then held constant through the simulation, $v(t) = v_0 = 2.5 \text{ m/s}$.

The initial phase of the simulation can be seen in Figure 18. Once again the dashed line represents the curb to which the bus should come close to but not hit.

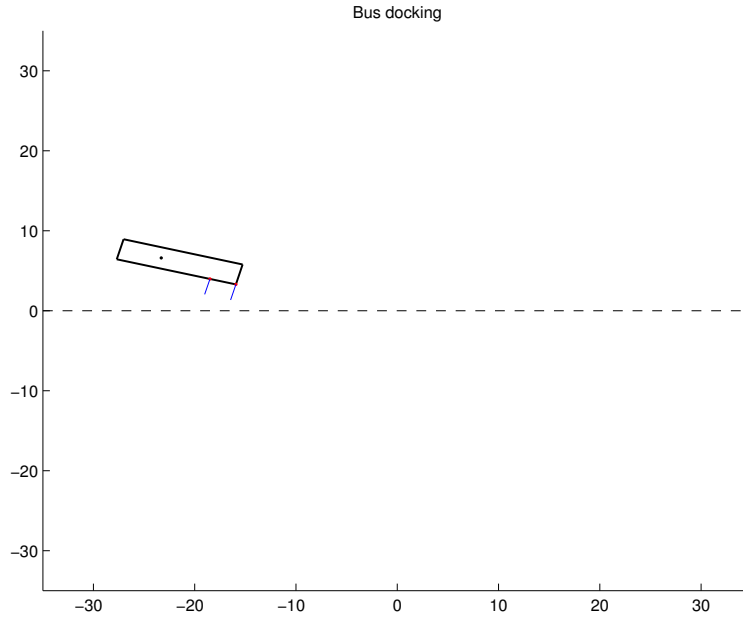


Figure 18: Initiation of the bus docking simulation. The bus has been initiated with some approach angle to the curb and will continue moving straight forward until both sensors get a reading.

The two red dots at the side of the bus in Figure 18 represents the positions of the sensors and the lines connected to these dots are the simulated sensor rays. The range of the sensors has been set to 2 *meter*.

When both sensors become active, they will change color to green. When activated, the goal and obstacle positions will appear along with the contours of the potential field. This can be seen to the left in Figure 19 and a 3D plot of the potential field can be seen to the right.

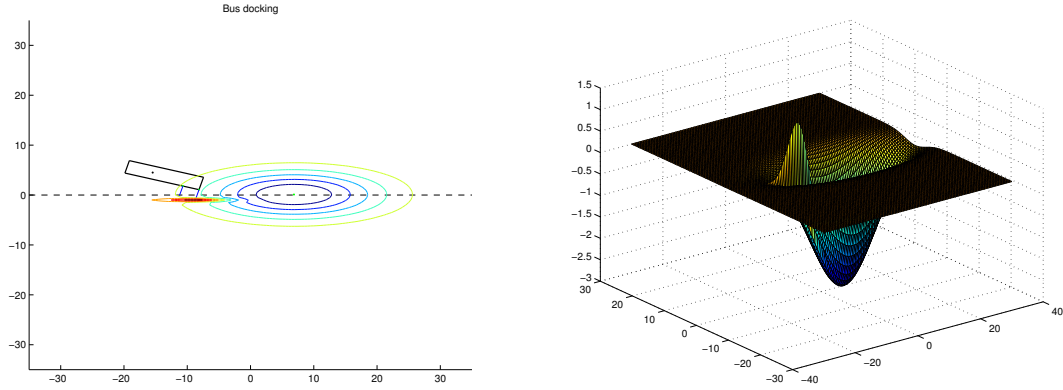


Figure 19: The activated sensors and the contours of the potential field (to the left). 3D plot of the potential field (to the right). Both sensors have now locked to the curb and the bus is steering according to the negative gradient of the potential field.

To the right in Figure 19, the goal and obstacle positions are clearly visible as the lowest and the highest points of the graph. This field will then continuously move along the y-axis as the bus drives forward.

The bus is now turning according to the steering angle that is generated by the potential field. The goal position has been set to be 5 *cm* away from the curb. In real life, the bus driver will decide when to stop but the simulation uses a break criterion saying that the front sensor should be less than 7 *cm* from the curb while the absolute difference between sensor measurements should be less than 3 *cm*. The final pose of the bus can be seen to the left in Figure 20.

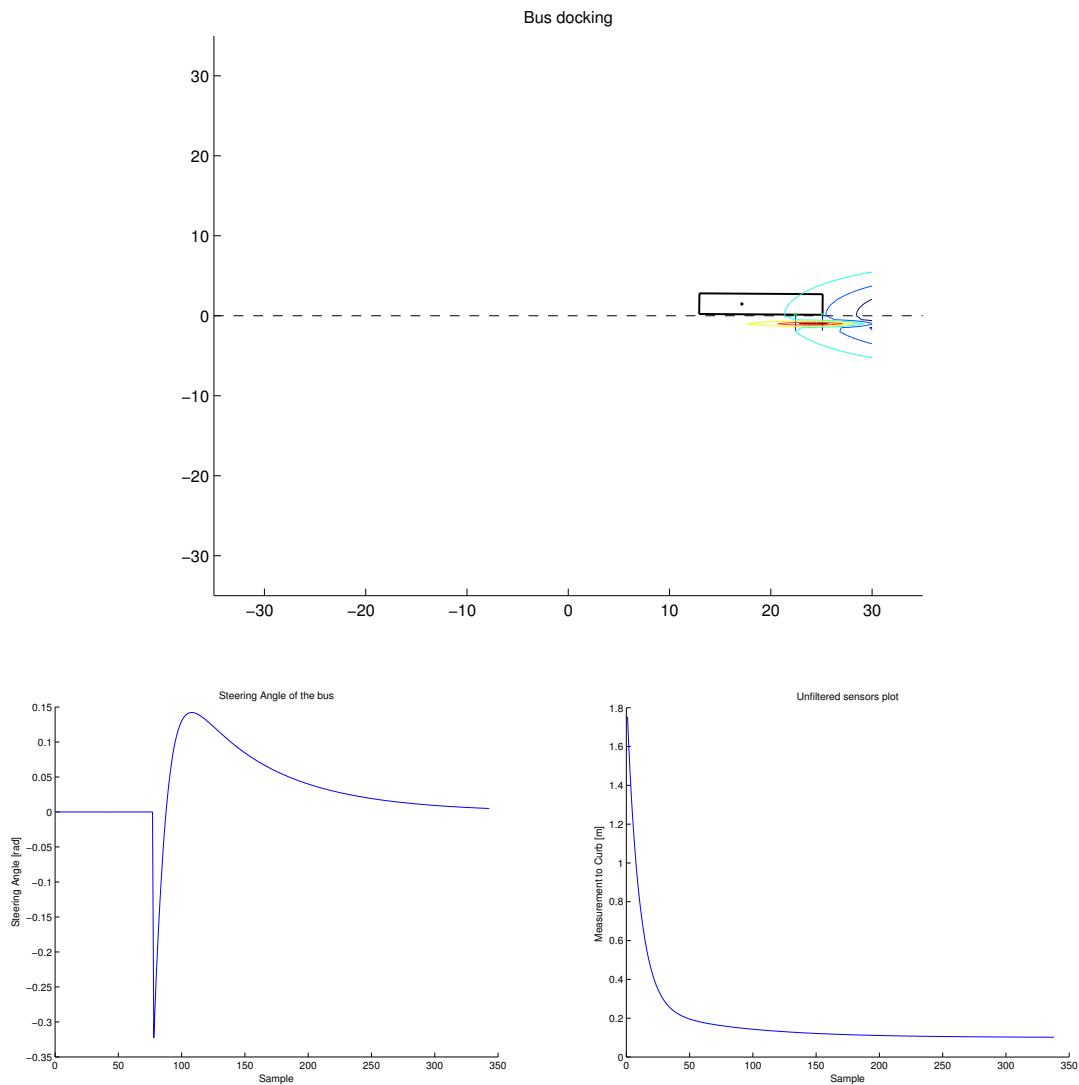


Figure 20: The bus simulation in its final pose. The bus has reached close to the curb and the simulation terminates (upper). In reality, the bus driver will stop when the bus is in position. The wheel angle during the docking can be seen to the lower left. The front sensor measurement to the curb can be seen to the lower right.

The right plot in Figure 20 shows how the steering angle is varying along the simulation. From this plot, it can be seen that the absolute maximum steering angle is approximately $0.32 \text{ rad} \approx 18^\circ$.

5 Validation

The system needs to be validated through testing to see if it really works since mathematical models never tell the whole truth about a system [4]. The system was validated using two regular ultrasonic sensors against a wall, both indoors with a rig and in a garage with a car. Some tests were also made with an artificial curb made of boxes indoors.

5.1 Hardware and software used

The different main hardware parts that were used for the tests along with descriptions is found below;

- Arduino⁵ MEGA 2560; an “open source electronics prototyping platform”. This board is used to connect to the sensors from MATLAB with a user package (ArduinoIO⁶).
- Two Maxbotix XL-MaxSonar WRMA1 MB7092⁷.

The software used for the validation was MATLAB combined with ArduinoIO.

5.2 The setup rigs used for the validation

The setup of the indoor validation can be seen in Figure 21. It consists of a rolling table where the computer on top is connected through MATLAB to the Arduino below which in turn is connected to the sensors (one in the front of the table and one in the back).



Figure 21: Indoor setups without a curb (left) and with an artificial curb (right). Testing to see that the correct steering angle is calculated in reality.

The Arduino MEGA 2560 was used so that MATLAB could connect to the ultrasonic sensors. A GUI was built in MATLAB to interact with the user and to see how the steering

⁵Information about Arduino can be found at www.arduino.cc.

⁶ArduinoIO is a user package for MATLAB that can be downloaded from [23]. With this package installed, MATLAB will be able to connect to the Arduino and control the Arduino pins. ArduinoIO makes rapid prototyping very simple since the algorithm used in the computer simulations can be used for the practical implementation by reading from the sensors connected to the Arduino instead of the simulated sensors in MATLAB.

⁷Detailed information about the sensors can be found at www.maxbotix.com.

wheel angle changed in different poses of the table. A screenshot of the interface can be seen in Figure 22 and for a more detailed explanation of the interface, please read Appendix A.

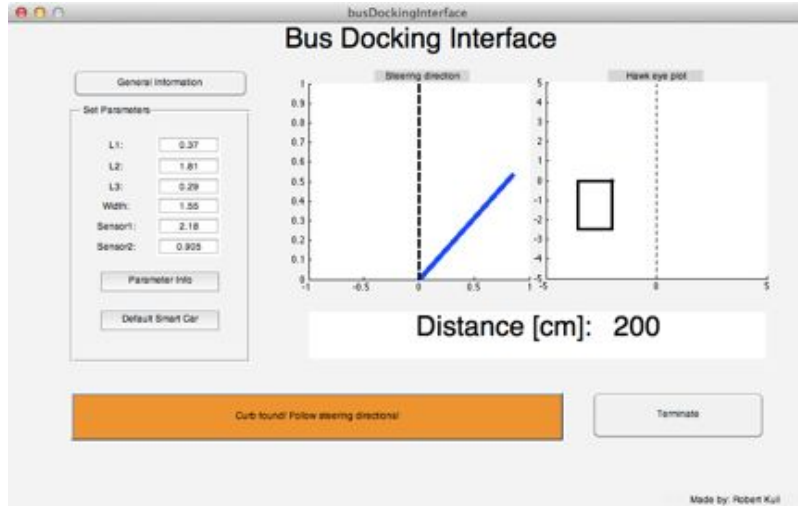


Figure 22: The built MATLAB interface used for the test with the car. The left plot of the interface shows the wanted steering direction while the right plot is showing an overview of the vehicle and the angle it has to the curb.

The tests with a car were set up according to Figure 23. Two metal brackets were screwed into a wooden stick at a proper distance from each other. The sensors were then mounted to the brackets and the wires were taped along the stick. The stick was then mounted underneath the car with the help of three straps. The Arduino board was once again connected to the sensors and the computer where the same MATLAB GUI was used to communicate with the driver.



Figure 23: The sensors mounted on the car for doing more real-life situation tests.

5.3 Results

When the tests were made, no effort was put in optimizing the rigs. Discoveries were made that could be avoided for the upcoming outdoor tests with better ultrasonic sensor (see below).

The MATLAB GUI gave the user directions of how to turn to be in the right direction at the current distance and approach angle to the curb. Some scenarios are shown in Figures 24-26 where the rigs are seen to the left and the screenshots of the GUI to the right.

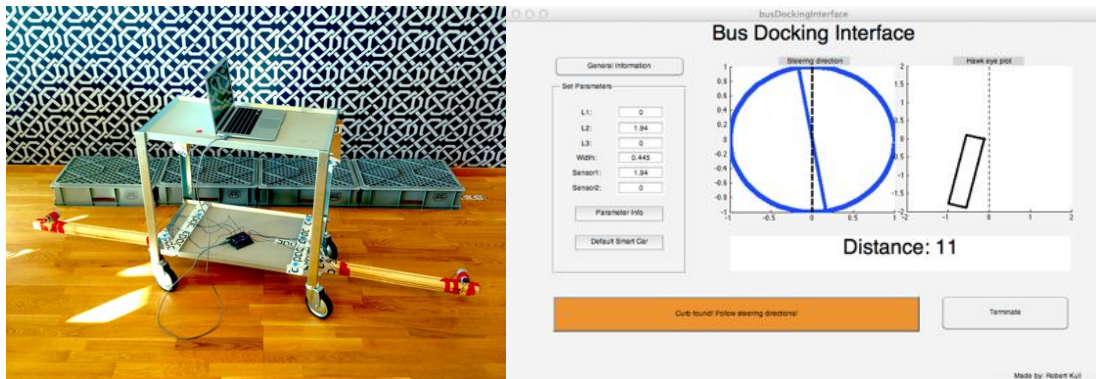


Figure 24: The pose of the rig (left) and the corresponding MATLAB GUI (right).

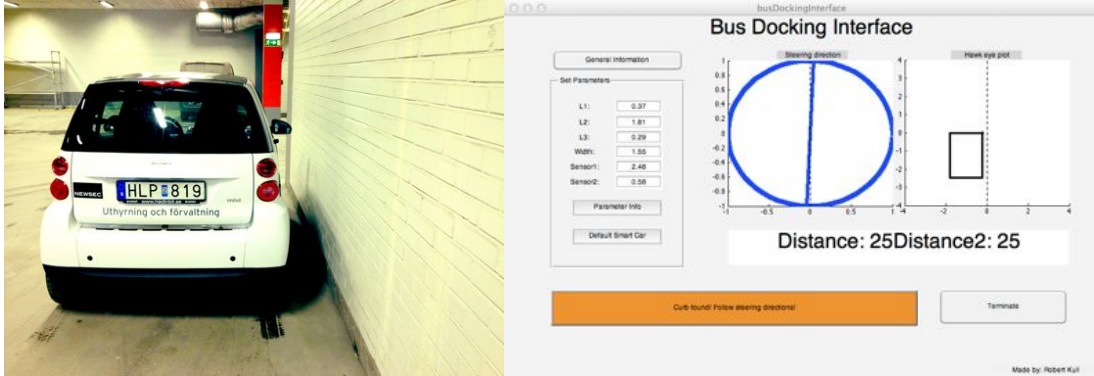


Figure 25: The car in Pose one and the corresponding MATLAB GUI.

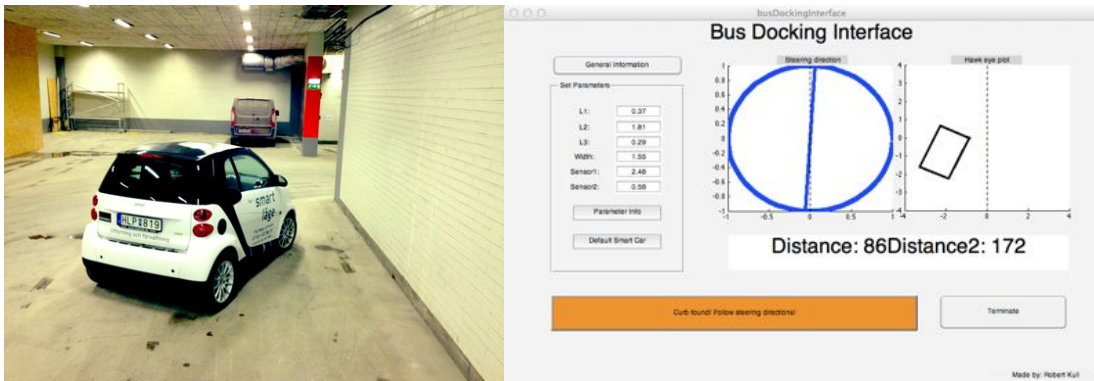


Figure 26: The car in Pose two and the corresponding MATLAB GUI.

The tests showed that the concept of potential fields are working for generating the steering direction. The sensors were averaged over five samples to give a more robust measurement although keeping the response of the system fast enough.

Some important discoveries and notes about the tests:

- Sensor interference; the sensors interfered with each other when not spaced that far away from each other (like to the left in Figure 21). When the table was turned in such a way that e.g sensor one measured 70 *cm* and sensor two measured 90 *cm*, sensor two could rapidly change to 190 *cm*. This ought to be because of sensor one emitting sound that echoes into sensor two. The same holds for sensor two measuring the smaller distance.

Toggling between the sensors was tried but all the interference did not disappear and the response time of the system became slower. When the sensors were spaced further apart (like to the right in Figure 21) the interference were much less and the response time was still fast. This is also a more real-life setup of the system since the sensors would be spaced further apart on a vehicle. Also, as more information about e.g. vehicle speed and current steering angle will be available through CAN-bus, a filter could be set up using the model of the system.

- Sensor range; From 25 *cm* up to around 6 *m*, the sensors were accurate, but in the close region (0 – 25 *cm*) the sensors measured 25 *cm*. This was the reason for the

mounting of the sensors. E.g. to the right in Figure 21, the sensors have been mounted as if underneath the vehicle and the distance is calculated as;

$$distance = reading - \epsilon$$

where

$$\epsilon = \eta \cos \varphi(t)$$

according to Figure 27. In the ideal case, the sensors would have been mounted at the edge of the vehicle but these calculations makes the error less.

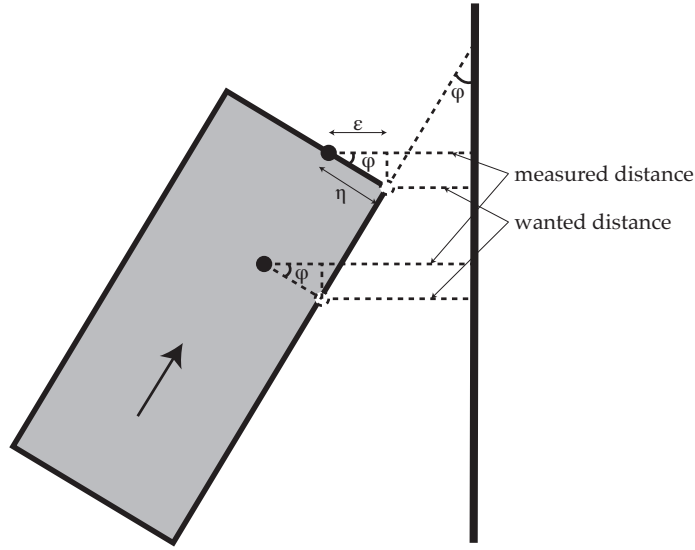


Figure 27: Distance calculations because of sensor displacement.

- Indoor rig; the indoor rig was not ideal for the tests although good results were retrieved in the end. Starting with the wheels, it was possible to turn not just the front wheels but also the rear wheels which gave a non ideal movement of the table (the table is holonomic while a car is non-holonomic). It became harder to simulate a docking process as the rear wheels turned.

Another problem with the indoor rig was the acoustic echo created by the “legs” of the rolling table. The sound emitted by the sensors bounced into the wall and then at the legs of the table and back to the sensors, giving unreliable measurements. This was however only a problem when the sensors where placed higher above the ground (like to the left in Figure 21). The sound could also be reflected in such a way that it did not echo back to the sensor.

- Car design; the Smart car is really low and it was difficult to mount something underneath it. The ground needs to be very flat to ensure that the sensors are not damaged in the tests. This problem would not exist on a regular bus since they are

much higher, still the sensors can not be mounted too low.

Also, the length of the area in front of the front wheel axis is very short on a Smart car. The ideal would be to mount the front sensor in the front edge of the vehicle but if that would have been done, the sensor would have “seen” the right front tyre all the time. The solution became to let the stick point out at a safe distance from the tyre (see Figure 23) and then making the system believe that this was the actual length of the car.

Overall, the tests gave good understanding about the problems that could arise if the system was designed in a problematic way by e.g. not spacing the sensors too far apart or mounting them close to parts of the vehicle that could reflect the echo in an unwanted way.

The tests showed a good behavior and the concept of the potential field approach is working really well to generate steering directions.

6 Sensors for robust curb detection

In today's vehicles, different sensors are used for parking assistance systems. Some parking assistance systems (passive systems) give the driver a warning when coming within a certain distance from an obstacle (like another vehicle). Other, more advanced systems (active systems), also control actuators of the car to fulfill the parking [3].

The sensors that will be tested in this project are ultrasonic sensors and monocular vision (single camera sensor). Another sensor that could be used is a LRF (Laser Range Finder) which is used by autonomous vehicles. The reason why it will not be tested practically is because of the price of such a sensor. Stereo Vision could also be used for curb detection [24] but it requires too much time and will not be tested here.

In the following parts, the basic theory behind each sensor option will be explained along with its implementation and the following results⁸. For the LRF sensor and stereo vision, there will only be a theoretical part and an explanation of how it could be implemented.

6.1 Ultrasonic sensors

Ultrasonic sensors (like the one from Pepperl Fuchs⁹ in Figure 28) are widely used in the industry because of their relatively cheap price and accurate measurements in a variety of environments.



Figure 28: Ultrasonic sensor from Pepperl Fuchs.

6.1.1 Theory

Ultrasonic sensors use high frequency sound ($> 20\text{ kHz}$) and the time-of-flight technique [25]. Sound pulses are emitted from the sensor through the surrounding medium, when an object is in the range the sound will bounce on its surface and echo back to the sensor which now is “listening” for the echoes. By the knowledge of the speed of sound and the time it takes for the sound to travel from the sensor and echo back, the distance can be calculated [26]. Some ultrasonic sensors consist of one transmitter and one receiver¹⁰ while other models (like in Figure 28) have one part that switches between being transmitting and receiving. An example of raw data from an ultrasonic sensor can be found in Figure 29.

⁸For the ultrasonic sensor and monocular vision.

⁹Information about Pepperl Fuchs can be found at www.pepperl-fuchs.com.

¹⁰Like the Ping))) sensor from Parallax (see <http://www.parallax.com/tabid/768/productid/92/default.aspx>)

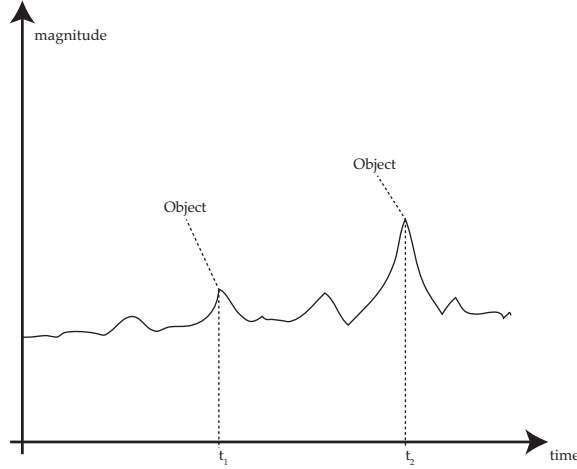


Figure 29: An example of raw data from an arbitrary ultrasonic sensor. The peaks at t_1 and t_2 are objects detected at two different distances from the sensor.

The sensors are widely used in the automotive industry because of their accurate measurements and their ability to be easily integrated in the vehicle design [27]. As an example, some cars have ultrasonic sensors mounted in the bumpers of the car to alert the driver when being close to obstacles that could damage the car. They are also used in parallel parking aid systems to scan for parking spaces (between other parked cars) that are big enough.

Although some ultrasonic sensor models are made to fit in an outdoor environment it is much harder to use them outdoors because of the higher rate of noise being present. Filters need to be applied to cope with this problem.

All ultrasonic sensors have a dead zone at close range where they cannot detect objects. This is because the reflected sound of these close objects returns to the receiver before it has been turned on. Usually, if the sensor has a long detection range, the dead zone is also longer (for the sensor used for validation, the dead zone was in the interval $0 - 25\text{ cm}$). To handle the dead zone problem, the mounting of the sensors is of great importance. The sound emitted from the ultrasonic sensors can also be reflected by deflectors to both change the direction and beam pattern¹¹ of the sound. This can make the dead zone from the sensor “disappear” by being before the detector, see Figure 30. By using a deflector, the sensor can also be mounted in a more covered way to prevent damage of the sensor.

¹¹A beam pattern of an ultrasonic sensor show how the sound spreads from the sensor to objects. The beam pattern for the sensor used in the validation part can be found in [28].

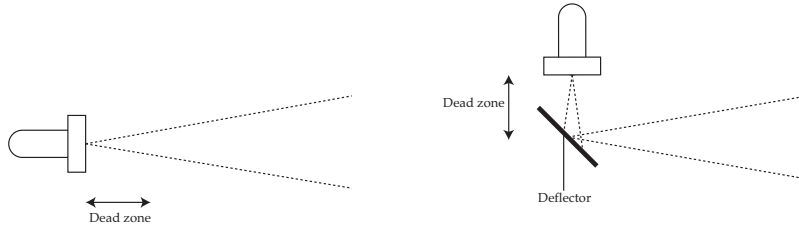


Figure 30: Usage of deflector with an ultrasonic sensor. A sensor without a deflector emitting sound and its dead zone (to the left). A sensor with a deflector and its dead zone (to the right). The deflector compress the sound and can “remove” the dead zone. The sound will be echoed back the same way as it is emitted.

Since the sensors are well used in commercial parking aid systems, although not for curb detection, they need to be tested if they can detect the high curb at the bus stop.

6.1.2 How many sensors will be needed if one is failing?

The information that the system need from the sensors to be able to calculate the steering angle is the distance from the curb and the approach angle to the curb. If only ultrasonic sensors should be used, a minimum of two sensors must be mounted (like in Figure 15).

The failure of one sensor can lead to steering angles being incorrect. If two sensors are used, the failure of one will lead to an incorrect approach angle and maybe also incorrect distance to the curb if the front sensor is the failing one. Figure 31 shows one way of how to view this and also how an increasing amount of sensors can be used to handle the problem with one failing sensor measurement.

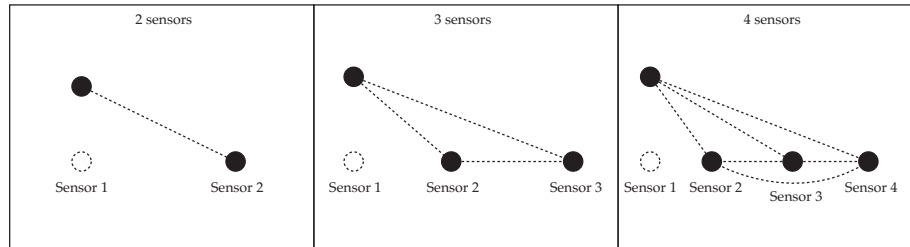


Figure 31: The effect of one failing sensor (Sensor 1) when having a system consisting of two, three or four sensors. The lines that can be drawn between each sensor represents the measured curb position. Read below for further explanation.

Sensor 1 is the failing sensor in all of these scenarios. Figure 31 is explained further;

- Two sensors are mounted; if one of the two sensors fail this will lead to an immediate failure of the system. The sensor that collects the wrong measurement will cause the system to calculate the incorrect approach angle to the curb and steering angle. The system can not detect a failing sensor from these measurements.
- Three sensors; by having this setup, a failing sensor an be detected but the system cannot know which sensor is the failing one. There will exist 3 different approach angles and the system will not know which one is the correct one.

- Four sensors; now the system can know that one sensor is failing and it can determine that Sensor 1 is the failing sensor. As can be seen in Figure 31, a total of 6 approach angles can be calculated from the sensor data and 3 of them will (approximately) be the same. The approach angles calculated between sensors 2 – 3, 2 – 4 and 3 – 4 will be the same and the system can determine that Sensor 1 is the failing sensor during this time sample.

Adding another sensor cannot give the system additional information and will not be needed.

If another sensor would fail making 2 sensors measure the wrong distance, a total of 5 sensors will be needed for redundancy.

This is however only if one sensor is failing. If tests of the sensors would show stable results, only two will be needed to get the information necessary for the system to calculate the steering angle.

6.1.3 Testing of the ultrasonic sensor

The specific ultrasonic sensor that will be tested for curb detection is the UC2000-30GM-IUR2-V15 from Pepperl Fuchs. Specifications and information about the sensor can be found in [29]. The sensor is tested using the rig in Figure 32. With this rig, different mounting heights and angles could be tested at different distances from the curb. The sensor could be mounted at intervals of 5 *cm*. A fake bus stop curb was built using wood since a bus stop without any traffic could not be found. The wooden curb has the same height as bus stop curbs (17 *cm*) [30].

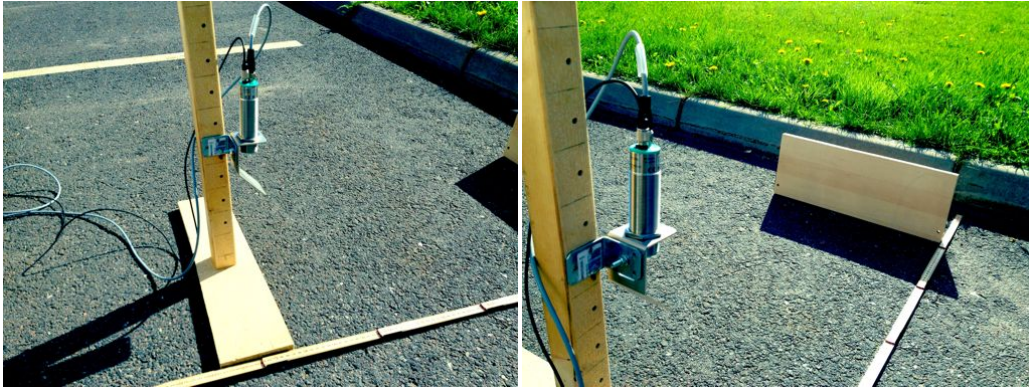


Figure 32: The rig used for the tests with the ultrasonic sensor. The sensor could be mounted at different heights and different angles for testing the detection range. A fake curb made of wood was built to have a curb of the correct height.

A software to tune some of the sensor parameters was also available through Pepperl Fuchs. Thresholds like sonic beam width, sensitivity and range could be set to see how the sensor readings changed. A screenshot of the software can be seen in Figure 33 and more information about possibilities of using the software can be found in [31].

The sensor should be mounted as high up as possible to prevent it from getting damaged. The problem with mounting the sensor high is not detecting the curb when being close to it. Different mounting angles and heights were tried along with different thresholds to find

a good combination. The sensor can also be mounted further underneath the bus to get the curb within the sensors range at all times.

The distance between a Volvo city bus floor and the ground is approximately 20 cm [32] while speed bumps are dimensioned to be approximately 10 cm high [33]. This will give the sensor a minimum space of 10 cm between the bus floor and the ground. Because of heavy loading of the buses, the sensor should however still be mounted as high as possible.

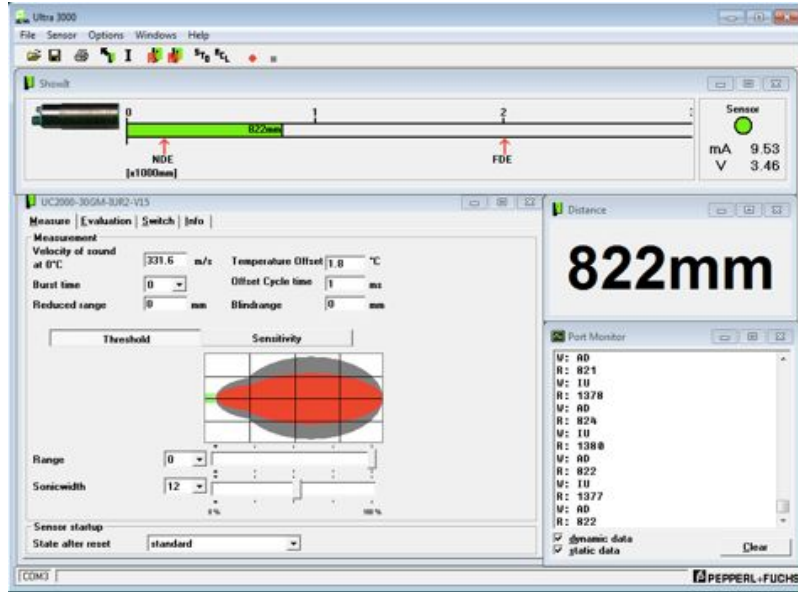


Figure 33: Screenshot of the graphical user interface of the ultrasonic sensor. Through the interface, thresholds like sonic beam width could be set to see how the result changed.

A variety of tests were made at different mounting heights, both with and without the deflector. The goal of each test was to be able to detect the curb at a range where $Maximum\ Distance - Minimum\ Distance \geq 1\ m$, where the *Minimum Distance* was the limiting factor.

Example: If *Minimum Distance* = 20 cm (the closest distance where the sensor detects the curb) the *Maximum Distance* must be $\geq 120\ cm$. The sensor could then be moved 20 cm underneath the bus and still measure the distance from the side of the bus to the curb in the range [0, 1] [m].

6.1.4 Results of the tests

The tests when using the deflector was poor compared to the tests with the sensor alone. Thus, the results are based on the tests without the deflector. Also, when the sensor could only handle small mounting angles ($\leq 5^\circ$) to work properly. If the angle was higher or if the beam width was too big, the measurement to the ground interfered with the measurement to the curb.

The “best” results for the different mounting heights can be seen in Table 2 (the “best” being the one having the lowest *Minimum Distance*). Different mounting heights was tested because other buses might have other ground clearance heights than the Volvo bus in [32].

Table 2: Results of the ultrasonic sensor for different mounting heights.

<i>Mounting Height</i>	<i>Angle</i>	<i>Minimum Distance</i>	<i>Maximum Distance</i>	<i>Threshold</i>
$\leq 15\text{ cm}$	$\sim 0^\circ$	<i>sensor deadzone</i>	$\geq 150\text{ cm}$	≥ 20
20 cm	$\sim 0^\circ$	$\sim 10\text{ cm}$	$\geq 150\text{ cm}$	10
25 cm	$\sim 0^\circ$	$\sim 25\text{ cm}$	$\geq 150\text{ cm}$	7
30 cm	$\sim 0^\circ$	$\sim 40\text{ cm}$	$\geq 140\text{ cm}$	0

As can be seen in Table 2, the sensor perform very well when the mounting height is low. The threshold value determines the width of the sonic beam where a high value means a thinner beam (max 25, min 0). A low mounting height is however not wanted and by moving the sensor up, it needs to be moved further in underneath the bus to be able to have a detection range of $\geq 1\text{ m}$.

It would be harder to mount the sensor any higher than 30 cm because of the *Minimum Distance* and the *Threshold* which are at the widest possible beam.

If information about the distance from the sensor to the ground was available, it would be possible to tune the threshold of the sonic beam width with a lookup table. This would be a good option if real-life tests show that the bus floor height is varying when driving.

Data was logged from the sensor at *Mounting Height* = 20 cm and *Threshold* = 10 to see how the performance of the sensor as if the vehicle was moving. The result of the test is shown in Figure 34.

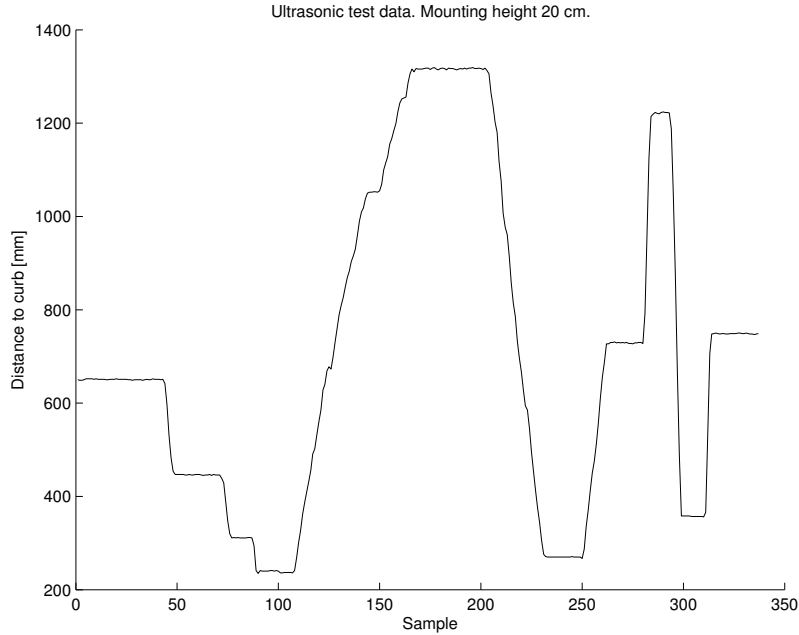


Figure 34: Test of the ultrasonic sensor as if vehicle is moving. The fake curb was moved irregularly and sensor data was logged and plotted. The sensor shows very stable results.

As can be seen in Figure 34, the sensor is very stable and does not fluctuate that much. It shows a very good performance overall.

6.2 Monocular vision

Cameras are also used as sensors in today's vehicles. Some of the applications where a camera is used are e.g. parking aid systems (rear cameras for obstacle avoidance), lane keeping and road sign recognition. Cameras are relatively cheap but they require a lot of computational power for the calculations. The camera that will be used for the tests are a simple webcam from Logitech, shown in Figure 35.



Figure 35: Webcam from Logitech used as camera sensor for the monocular vision test.

6.2.1 Theory

By image processing the camera sensors can give more information than just a distance measurement. Unfortunately, they do not perform as well in e.g. dark or misty environments when it is hard to get a clear image of the scene. Because of the image processing, they also require more computational power than e.g. the ultrasonic sensor.

A camera can be implemented by mounting it on the bus at a known distance to the ground, facing the ground (see Figure 36). Image filters can then be applied to try to extract the curb from the rest of the scene. The problem will consist of finding lines in the image and also to find the line corresponding to the edge of the curb.

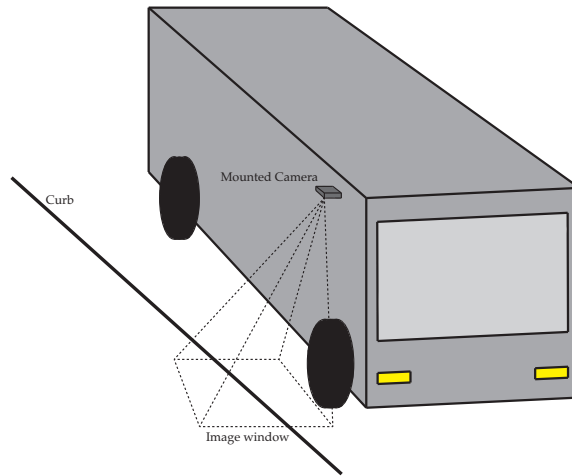


Figure 36: Mounting position of a camera on the bus. By this mounting, the approach angle and distance to the curb could be calculated using a pinhole camera model and geometry.

The distance to the curb can then be calculated by assuming a pinhole camera model, knowing the resolution of the images and using geometry. The pinhole camera model is presented in Figure 37 and the equations used for the calculations are;

$$X_{world} = \frac{x_{image}Z_{world}}{f}$$

$$Y_{world} = \frac{y_{image}Z_{world}}{f}$$

where f is the focal length of the camera, Z_{world} is the distance to the object, i.e. the mounting height of the camera, $[x_{image}, y_{image}]$ is the coordinate of the pixel in the image while $[X_{world}, Y_{world}]$ is the 2D coordinate of the pixel in the world. Since the coordinate $[X_{world}, Y_{world}]$ is given in *[pixels]* the resolution must be known to convert the coordinate to *[m]*. But as the distance to the object Z_{world} is known, a converting constant can be calculated by measuring the real world size of the image.

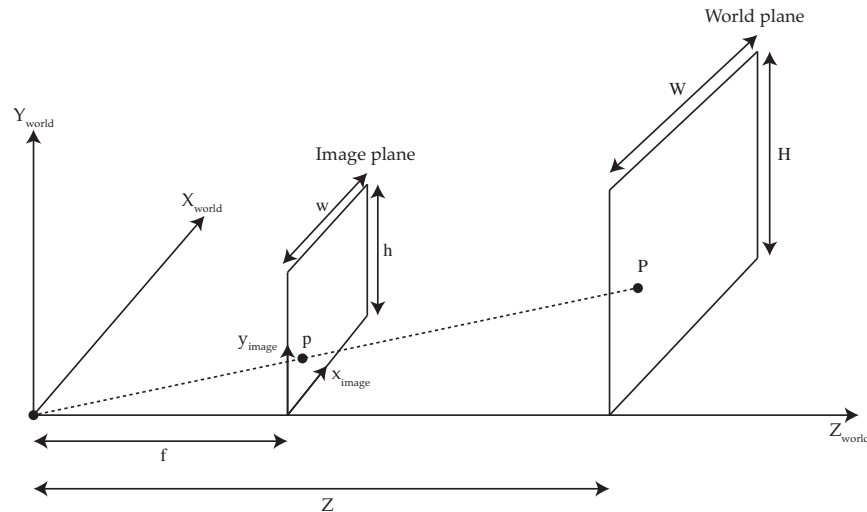


Figure 37: Pinhole camera model.

The point p represents the coordinate in the image of the real world point P . The image resolution is $w \times h$ *[pixels]* while the real world window has the area $W \times H$ *[m]*. From these relationships, the real distance in *[m]* between two points in the image can be calculated.

6.2.2 Testing of monocular vision

The implementation was made by using the webcam in Figure 35 along with MATLAB. Images and videos were acquired from the camera as test data to be processed. The acquired data was not taken when the camera was actually mounted on a vehicle but just held in the approximately same pose as if it were. This had no effect on the result of the tests.

When the image processing was made, different filters were applied to find the line belonging to the edge of the curb. The steps used for the processing were;

1. Load the image (or frame); the resolution of the original image was 1024×768 and 8-bit grayscale. This corresponds to a 1024×768 matrix where the elements is in the interval $[0, 255]$ depending on the color (0 being black and 255 being white).
2. Resize the image; to make the calculations faster, the new image matrix has the dimension 400×300 .
3. Rotate the image 90° ; the reason for this step will be clear later on.
4. Blur the image; a median filter¹² is applied to the image to reduce the noise and smoothen the image.
5. Edge detection; a Canny¹³ edge detection filter was used to detect the contours in the image. The output from this filter is a binary image where the contours are white ($= 1$) and the rest of the image is black ($= 0$).
6. Remove small objects; objects less than 50 *pixels* were removed to prevent that unnecessary lines were found.
7. Find the lines of the image; the Hough-transform¹⁴ was used to detect the straight lines in the image. The rotation of the image makes it possible to constrain the angle interval of the lines that were supposed to be found to $[-45, 45]^\circ$. This is the same as saying that the bus approach angle will never be greater than 45° or less than -45° .

The result of some of the steps are shown in Figure 38.

¹²A median filter works through the image in pixel segments by 3×3 and changes the color to the median value of the pixels in the segment. The segment size could be varied but the effect to the image was the same. A larger segment gave slower calculations. Please see <http://www.mathworks.se/help/images/ref/medfilt2.html> for a complete description of the MATLAB function.

¹³There exists a couple of edge detection algorithms like Sobel, Roberts etc. that could be used. However the Canny filter gave the best result for this particular case. The MATLAB function is described at <http://www.mathworks.se/help/images/ref/edge.html>.

¹⁴The Hough transform can be used to find features in an image, in this case it is straight lines. The theory behind how the transform works is beyond the scope for this project. The MATLAB functions used are described at <http://www.mathworks.se/help/images/ref/hough.html>, <http://www.mathworks.se/help/images/ref/houghpeaks.html> and <http://www.mathworks.se/help/images/ref/houghlines.html>.

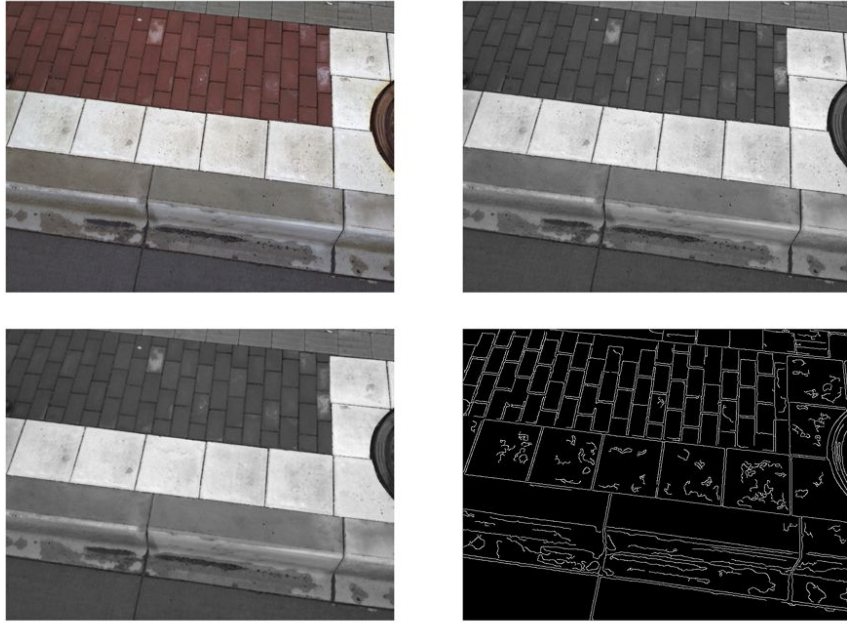


Figure 38: Original image (top left), grayscale image (top right), filtered image (lower left), Canny image (lower right).

6.2.3 Results of the tests

Results showed that the monocular vision sensor was not well suited for finding the correct measurement to the edge belonging to the curb, see Figure 39. Only one of the processed images will be showed here since the results were the same for all the images and the video.

Too many lines were found and it was hard for the algorithm to detect only the correct one. However, the approach angle of the bus can be estimated by implementing a single camera sensor.



Figure 39: Lines found in an image using the Hough transformation and Canny edge detector. Too many lines of the same type are found and no significant color can be used to pick out the correct one (the white area does not exist at every bus stop).

As can be seen in Figure 39, a total of 7 lines are found that belongs to 5 edges in the image. The approach angle of the bus can be found quite easily, note that all the lines in Figure 39 are parallel. But the distance to the correct line cannot be found in the image.

6.3 Stereo Vision

The concept of stereo vision is basically how our eyes work. From two views of the same scene, we are able to determine the depth of the scene or approximate the distance to certain objects in the field of view. By adding another camera to the single camera setup, the same technique can be used to find the curb [24], [34].

6.3.1 Theory

Stereo vision requires two cameras that are separated by a distance (called baseline) and then by detecting the same object (point in the world) in both images, triangulation can be used to determine the distance to the object. This is shown in Figure 41. A stereo camera sensor from Point Grey¹⁵ is shown in Figure 40.

¹⁵More information at www.ptgrey.com.



Figure 40: Stereo camera sensor from Point Grey. Two parallel cameras mounted at a distance apart from each other.

The process of finding features in the first image and then find the same feature in the second image can be very time consuming. The process is much faster if it is known where to search. This is made by epipolar geometry¹⁶. The complete theory of stereo vision will not be covered here but from Figure 41, the depth to a point can be calculated from the points p_l and p_r as;

$$Z = \frac{bf}{d} \quad (32)$$

where b is the baseline, f is the focal length and d is the disparity, i.e. $x_l - x_r$ (difference between the images). The disparity will be higher for closer objects than for objects being further away.

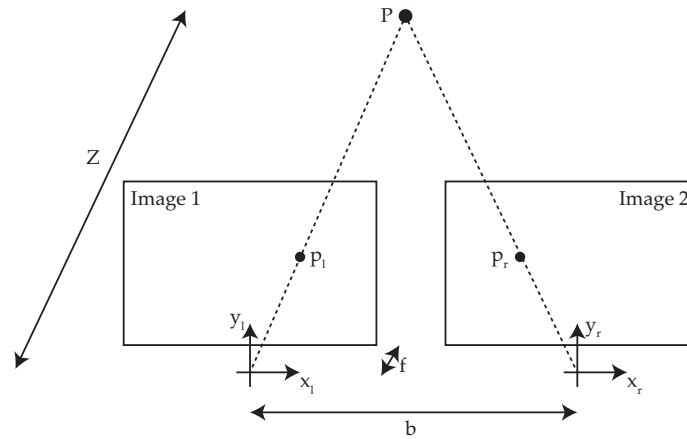


Figure 41: Stereo vision triangulation (reconstruction).

Stereo vision is not that simple and requires a lot of work with calibration to get a good result from the images. The complete theory of stereo vision is beyond the scope of this project but more information can be found in [36], [35], [37].

6.3.2 How to implement stereo vision

A stereo vision sensor could be implemented in the same way as the single camera, by facing down towards the curb. A correctly implemented camera would see the curb as a distinct

¹⁶For more information about epipolar geometry, see [35]

change in the depth. The correct line within this area could then be found in the same way as with monocular vision and the distance could be measured.

6.4 Laser range finders (LRF's)

Different laser range finders are often used for autonomous vehicles and prototype vehicles¹⁷ and depending on the sensor, it can scan in either 2D or 3D. These sensors are very expensive but they give a very accurate measurement of the distance to an object.

6.4.1 Theory

The LRF sensor rotates and scans its surrounding. The sensor in Figure 42 can scan 360° and is continuously measuring. Like the ultrasonic sensor, the LRF uses the time-of-flight measurement to determine the distance to an object with laser light instead of ultrasonic sound. This gives the laser sensor much faster and accurate measurements and it does not depend on color or shape of the objects around it.

The raw data from a 2D laser sensor is given as a distance and angle to the object that gets hit by the laser beam.



Figure 42: 2D Laser Range Finder from Pepperl Fuchs. LRF sensors are widely used by autonomous prototype vehicles.

6.4.2 How a LRF sensor could be implemented

This sensor could be mounted either on top or at the side of the bus. If mounting the LRF on top of the bus, a scan could be made in the horizontal plane to search for some reflectors at the bus stop and then by triangulation, the position and pose can be calculated (see Figure 43). By mounting the LRF on top of the bus, more autonomous applications could be implemented in the future to assist the driver.

¹⁷Like the Google driverless car.

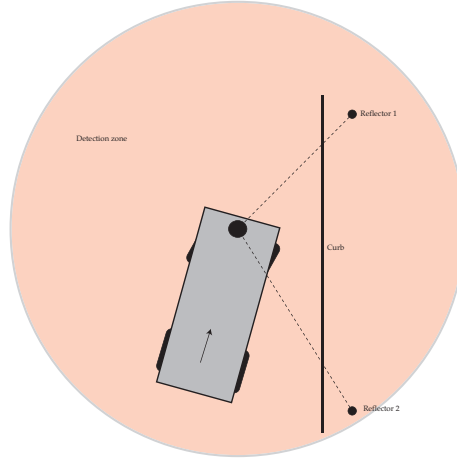


Figure 43: Implementation of a horizontal LRF sensor. The top mounted sensor could detect two reflectors mounted at the bus stop. By triangulation, the position and pose of the bus can be calculated. With this setup, the whole bus stopping maneuver could be done by the system since the bus stop can be detected at a further distance. In combination with a GPS and data base for the outlay of each bus stop, precise docking can be made. A top mounted LRF can also be used in other autonomous bus applications and active safety solutions.

The other way of mounting the sensor would result in a vertical scan at the side of the bus. The scan would then result in raw data where a L-shaped pattern would represent the curb. Two LRF's would then be needed or one combined with e.g. computer vision to get the approach angle to the curb. The side mounting of the sensor can be seen in Figure 44.

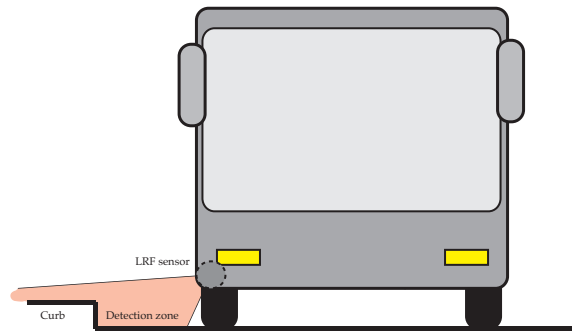


Figure 44: Implementation of a vertical LRF sensor. A scan is made in the vertical plane at the side of the bus. The curb can then be detected as a L-shaped figure in the raw data from the sensor.

6.5 Sensor discussion

For the system to be of any use for the drivers, it relies on correct sensor measurements. For autonomous driving, laser seems to be the “standard” option to use although these autonomous driving vehicles are still prototypes (like the Google Car).

The problem with implementing these kind of systems on buses is mostly that the price cannot be too high. One of the reasons for this is the smaller manufacturing volumes of

buses compared to cars. Another is the customer, where the passenger of the bus will not be able to enjoy the consequences of a system like the one described in this report because he/she is not driving the vehicle. In a car, people are more willing to pay for active safety solutions and comfort (like self parking applications).

Laser sensors are robust in the sense of precise and very fast measurements. Because of the time-of-flight measurements they need to respond very fast because of the high speed of the light. If the budget for the project had been greater, these sensors would have been the obvious choice. Also because of their ability to be used in other applications than just at the bus stops. A top mounted LRF (like in Figure 43), with either a 2D- or 3D-scanning technique could be used to implement more active safety solutions along with this application.

The ultrasonic sensors require that the vehicle is closer to the curb before they activate the steering assist. This is not a bad thing but it requires more from the driver in the sense of coming into the bus stop in a better way than if the system would give assistance in an earlier stage of the maneuver. At this distance, they give an accurate enough measurement to the curb.

The price basically controls what should be possible for the system and how much help the driver should get. For this project where the bus stopping maneuver was divided into two parts (as in Figure 1), the ultrasonic sensors will work and less tires will be worn out because of the driver hitting the curb.

Tests for the ultrasonic sensor have only been made for curbs that are of Swedish standard height (17 cm). All new and rebuilt bus stops have this height because it makes it easier for old people and people with certain handicaps to get on and off. The sensors need to be tested for the lower curbs as well to see if they can handle such a situation. A problem that could occur would be different sensor parameter tuning for different curb heights. Since the system is unable to measure the height of the curb it will not be possible to tune the parameters depending on the curb height. This would require a more advanced sensor.

The ultrasonic sensors are very good in the sense that they do work nicely in darkness and does not depend on the color of the measured object (the curb). A LRF sensor would also have managed to measure even in complete darkness. For the camera, if a good algorithm would have been found that could extract the correct edge from the image it would still have had problem at night when it is dark. Night vision is an alternative but would most likely make the sensor more expensive.

Ultrasonic sensors would also do good in a dusty environment where dust might end up on the sensor or the deflector. The sound will still be able to travel and hit the correct object for measuring.

The hardest part to conquer is a snowy environment. The ultrasonic sensor will be able to get a good reading of the side of the curb as long as the snow does not cover too much of it. Snow has a sound absorbing performance which can make the measurement less accurate. The camera would not be able to see any edges at all if the ground was covered with the least layer of snow.

6.5.1 Which sensor should be used?

Table 3 explains the strengths and weaknesses of each sensor alternative covered in this chapter.

Table 3: Pros and cons for the different sensor alternatives.

Sensor type	Pros	Cons
Ultrasonic Sensor	Precise measurement, cheap	Not knowing if measuring distance to curb
Monocular Vision	Approach angle could be found	Could not find correct line
Stereo Vision	Curb can be found from the depth	Computationally heavy, sensitive for vibrations
Laser Range Finder	Very precise	Expensive

Since the goal with the implementation is to be able to build a system that is as cheap as possible. For this, the ultrasonic sensor is a good enough option. The system will only be active when the bus is close to the curb and within this range, the ultrasonic sensors are reliable enough to detect and measure the distance to the curb.

7 Signal processing and filter design

The result from the ultrasonic sensor test shown in Figure 34 is stable and does not contain that much noise. If the sensor were to perform like that when mounted on a real bus, any filtering would not be necessary. However, mounting the sensor on a real bus would introduce some error sources like e.g. sound noise and altering mounting height which can make the sensors measure the wrong distance to the curb.

This chapter aims to explain what filtering techniques that can be used if there would be a noisy signal from the ultrasonic sensor. Since no real noisy sensor data were available, the simulator described in chapter 4 was used and noise were added to the simulated sensor signals. Different filters were then applied for removing the noise and to see how the simulated bus behaved.

The filters that were applied were the simpler median and mean value (averaging filters), the first order low pass filter and the more advanced Kalman filter. The Kalman filter was firstly implemented as a static type and then by using information about the system to see the difference in performance.

7.1 The median and mean value (averaging) filters

These two averaging filters smoothens the signal so that a more solid measurement is given by the sensors. They filter out unwanted small and rapid changes in the measurements by collecting n measurements at time step k , take the mean or median value of the n measurements and use that value as the sensor signal at the current time step.

The discrete mean value filter can be used when the signal is known to be noisy without any abrupt changes (like if the sensor would loose a value).

As can be predicted, the performance of the filter gets better as n gets higher. The drawback of this filter is the time it takes for an ultrasonic sensor to update the measurements. The speed of the sensor controls how large n can be to get a wanted sample frequency of the system. Also, if the sensor would drop a reading (which would make the sensor give its maximum value), the mean value will also change. This makes the filter unuseful for a sensor where readings could be lost.

A median value filter uses exactly the same technique as the mean value filter but takes the median of the collected measurements instead of the mean. By doing this, the filter will have the ability to filter out a lost reading.

Because these filters require many readings to smoothen the signal, they are inappropriate to implement in the bus. The slow filters could make the sample time drop and information might be lost which can make the bus hit the curb.

7.2 First order low pass filter

A first order low pass filter can be implemented to get rid of the noise in the sensor signal [38]. The filter is implemented as;

$$\bar{z}_k = \delta \bar{z}_{k-1} + (1 - \delta) z_k \quad (33)$$

where \bar{z} is the filtered signal and z is the measurement. The constant δ depends on the sample time and time constant of the system. How to calculate δ is described in [38]. The results of tests with different δ can be seen in Figure 45. The tests were made with a

variance of 0.1. Note that Figure 45 is the same as the lower right of Figure 20 but with the added noise.

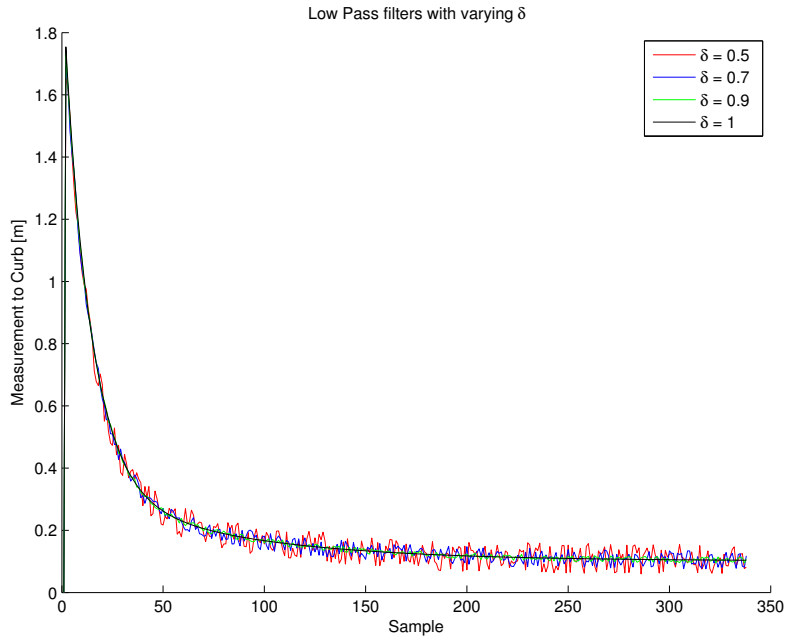


Figure 45: Tests of first order low pass filter (or moving average filter) with varying parameter δ .

A higher value of δ gives a more stable result but requires a slower process, i.e. a higher time constant ([38]). The lower the δ gets the more noise comes through the filter and will affect the system.

This is a simple filter to implement and only requires tuning one parameter to get a pleasing result.

7.3 The Kalman filter

When the behavior of the system is known, i.e. the next step of the process can be estimated mathematically, the Kalman filter is good to use to filter the sensor signal. The Kalman filter is an optimal estimator [39] which are used to filter signals and to fuse different signals together (sensor fusion) [40].

7.3.1 Basic theory of the Kalman filter

The process to be estimated by the Kalman filter is $x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$ with the measurement $z_k = Hx_k + v_k$. These are linear equations with w_k and v_k being the process and measurement noise [39]. The steps of the general implementation¹⁸ of the discrete Kalman filter is then to;

¹⁸These are the same steps as presented in [39]. For a more detailed description of the Kalman filter and how it is implemented, see [39].

1. Predict the next value of the state from the linear model. Calculate the predicted state $\hat{x}_k^- = Ax_{k-1} + Bu_{k-1}$.
2. Estimate the process error $P_k^- = AP_{k-1}A^T + Q$. Where Q is the process noise covariance.
3. Calculate the Kalman gain as $K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$. R is the sensor noise covariance.
4. Correct the predicted state with the measured state as $\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$.
5. Correct the estimated process error as $P_k = (I - K_k H)P_k^-$.

7.3.2 Implementing the Kalman filter to process the signal

A scalar Kalman filter can be implemented for the two sensors of the simulated bus. Noise is added to the sensors and the unfiltered signal is sent through the filter.

If nothing would be known about the system, i.e. the scalars/matrices A and B are not known the prediction of the next value of the state would simply be $\hat{x}_k^- = x_{k-1}$, ($A = 1$, $B = 0$). This is a stationary form of the Kalman filter (stationary Kalman filter) which is used in [41] and can also be implemented to the system with the bus. By following the steps in 7.2.1, the scalar equations used by the filter are;

$$\hat{x}_k^- = x_{k-1} \quad (34)$$

$$P_k^- = P_{k-1} + Q \quad (35)$$

$$K_k = \frac{P_k^-}{P_k^- + R} \quad (36)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-) \quad (37)$$

$$P_k = (1 - K_k)P_k^- \quad (38)$$

The equations 34-38 are then used for both sensors.

The Kalman filter needs initial guesses for the error P and the noise covariances Q and R . Figure 46 and 47 illustrates how the covariance variables affect the performance of the filter. When these simulations were made, the bus was initialized 2 m from the curb with a heading $\theta_0 = 0$ and the goal was set 0.05 m from the curb.

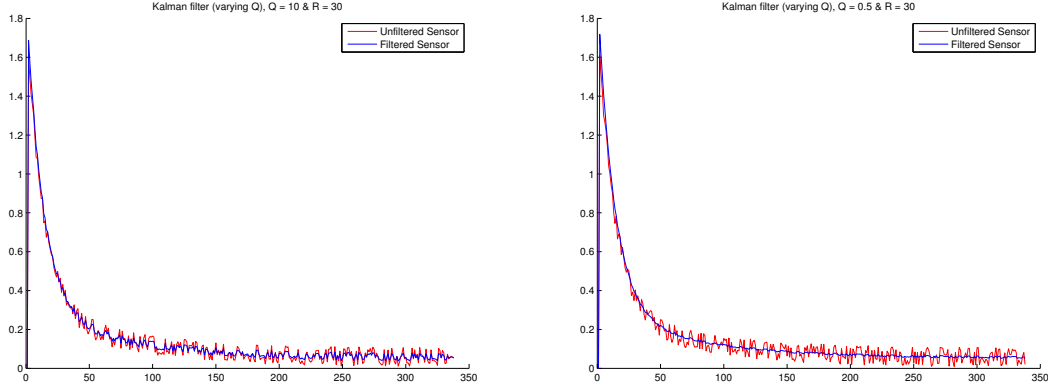


Figure 46: Kalman filter with constant R and varying Q . High Q (to the left) and low Q (to the right).

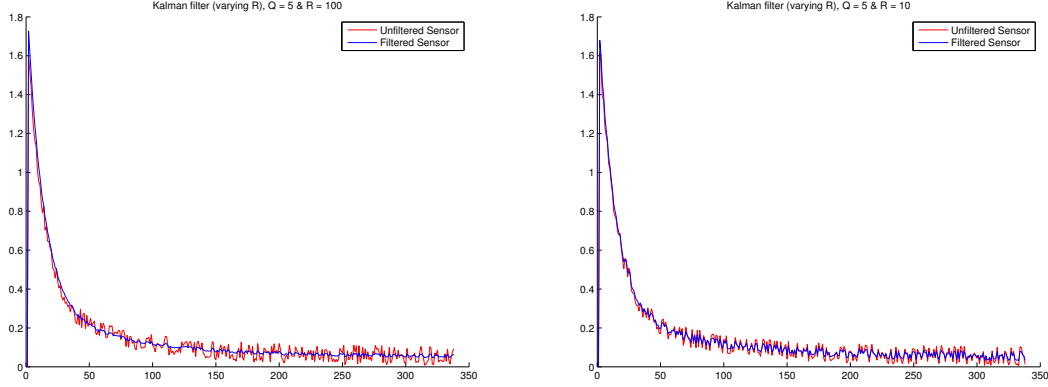


Figure 47: Kalman filter with Q held constant and varying R . High R (to the left) and low R (to the right).

From Figure 46 and 47 it can be seen that the low Q and high R gives the best performance. The Kalman filter “trust” more in the prediction of the sensors (\hat{x}_k^-) since the covariance of the sensor noise is much higher than the process noise. This was an expected result since noise has only been added to the sensors and the fact that the process is quite static from around sample 75 and forward. By the static behavior of the process, $\hat{x}_k^- = x_{k-1}$ becomes a very good prediction of the next value and thereby the good result.

As the system model is known, i.e. the movement of the sensor can be estimated using knowledge from the mathematical relationships defining the movement of the vehicle. Since the Ackerman steering geometry is assumed to hold, the movement of any point of the vehicle can be predicted since they are rotated around the same point [7]. This is shown in Figure 48.

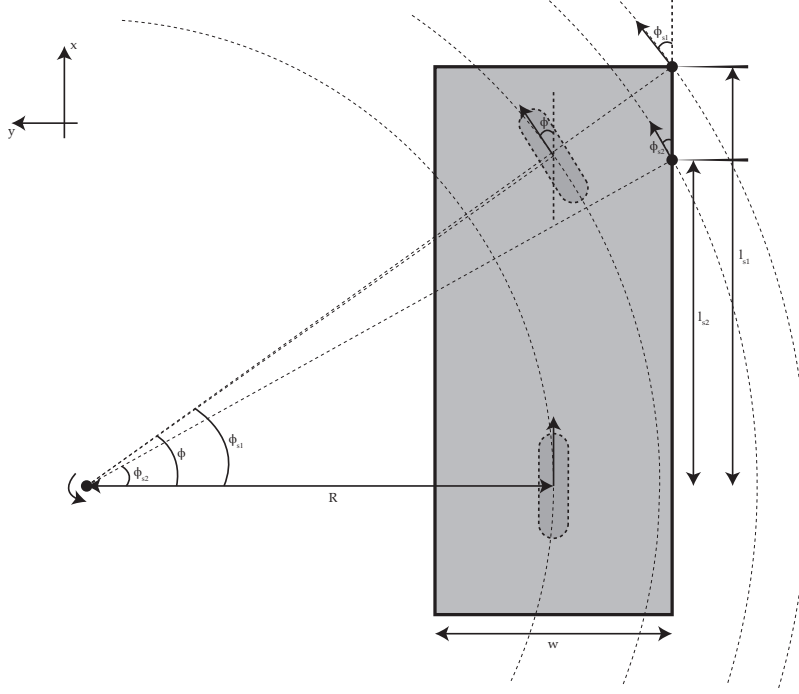


Figure 48: The movement of the sensors depending on the steering wheel angle. Since the Ackermann steering geometry is assumed, every point of the bus will rotate around the same point and the sensor movement can thereby be calculated using geometry.

From knowing the steering wheel angle ϕ , the direction of movement of the front sensor (ϕ_{s1}) can be calculated as;

$$\phi_{s1}(t) = \arctan\left(\frac{l_{s1}}{R + \frac{w}{2}}\right) \quad (39)$$

where R is calculated according to Equation 16 and the movement along the y-axis (Equation 7) for the sensor will be;

$$\dot{y}_{s1} = v(t) \sin \phi_{s1}(t). \quad (40)$$

From the simulations it can be seen that the steering wheel angles are not that large. As shown by Figure 20 the maximum steering wheel angle was $\approx 0.32 \text{ rad}$ which is assumed to be low. The small angle approximation ($\sin \phi_{s1} \approx \phi_{s1}$) can then be used to linearize Equation 40, hence

$$\dot{y}_{s1} = v(t) \phi_{s1}(t) \quad (41)$$

and the maximum error caused by this approximation is only 1.7 %. The prediction of the next value of the state (step 1 in 7.2.1) can be calculated as;

$$\hat{y}_{s1,k}^- = y_{s1,k-1} + v_{k-1} dT \phi_{s1,k-1} \quad (42)$$

with $A = 1$, $B = v_{k-1} dT$ and $u_{k-1} = \phi_{s1}$. By following the implementation steps, the rest of the calculations will yield;

$$P_{s1,k}^- = P_{s1,k-1} + Q \quad (43)$$

$$K_{s1,k} = \frac{P_{s1,k}^-}{P_{s1,k}^- + R} \quad (44)$$

$$\hat{y}_{s1,k} = \hat{y}_{s1,k}^- + K_{s1,k}(z_{s1,k} - \hat{y}_{s1,k}^-) \quad (45)$$

$$P_{s1,k} = (1 - K_{s1,k})P_{s1,k}^- \quad (46)$$

The same technique is then used for the second sensor, with l_{s2} instead of l_{s1} in Equation 39. The result of this filter for noisy sensors with $var = 0.1$ can be seen in Figure 49. The plot shows the result of the front mounted sensor.

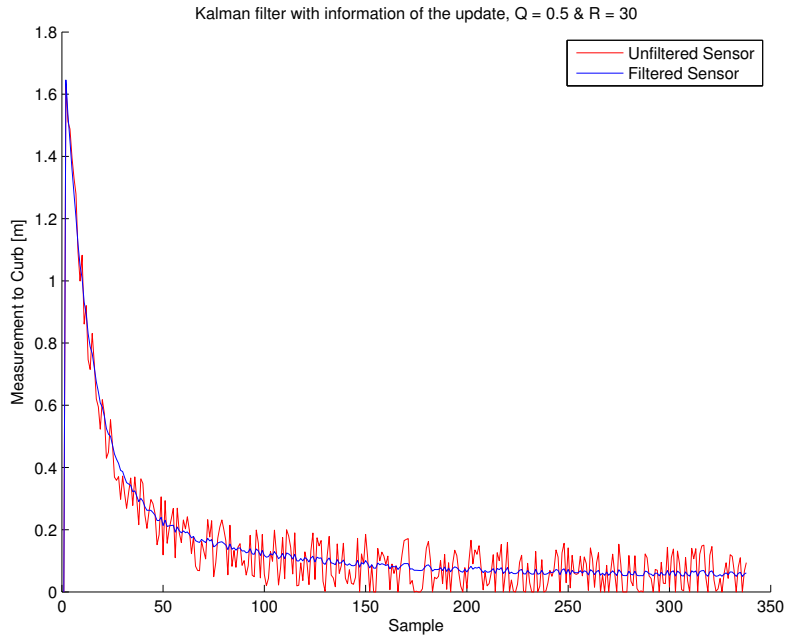


Figure 49: Simulations of the Kalman filter using information about the movement of the sensors.

7.4 Which filter should be used for the bus docking application?

The Kalman filter and the first order low pass filter shows very good results from the simulations. Real-life tests however needs to be made to see the true performance of the filters. The Kalman filter requires more calculations and the low pass filter should thereby be used if possible. Also, the sensor seem to give quite robust raw data so it might be enough to use the first order low pass filter.

8 Project status

The project has been examining the possibility to make an assisting system for docking buses. The simulated closed loop system shows good results and a sensor that could manage to detect the curb has been found. Filtering techniques to use if sensors would give noisy readings have also been considered. The project is now in turns of moving from a theoretic to a more practical approach with implementing the different parts described in this report.

8.1 Future work

Since the time for the master's thesis is restricted, the implementation of the system in the bus could not be made. The project will however continue with an implementation phase to get the prototype done. A bus with the electric steering is available and the system will be implemented according to this thesis.

One of the things that remains to be considered is the response of the electric motor, i.e. in what way should the driver receive the steering wheel feedback? It is possible to make the motor generate a turning torque with an magnitude depending on the error between the actual steering angle and the steering angle calculated from the potential field. The motor could also make the steering wheel vibrate when not being inside some interval from the reference steering angle. These tests should appropriately be made in association with bus drivers since they will be the ones using the system. A good functioning system can be seen as a learning tool for unexperienced drivers to help them get the feeling of where to position the bus at curbs.

The bus docking system is one of several applications that could be enabled by the installation of an electric motor. The buses are not even close of having the same active safety solutions that exists in cars but with the electrical steering, there are a lot of possibilities.

References

- [1] (2011). [Online]. Available: www.trafa.se
- [2] T. Falkmer, “Fordonsrelaterade skador vid av och påstigning i samband med kollektivtrafik och färdtjänstresor (al90 b 2006:9001),” Slutredovisning av projekt, 2006.
- [3] A. Eskandarian, Ed., *Handbook of intelligent vehicles*. Springer-Verlag, 2012.
- [4] L. Ljung and T. Glad, *Modeling of Dynamic Systems*. Prentice Hall, 1994.
- [5] The Mathworks, Inc. (2013). [Online]. Available: www.mathworks.se
- [6] H. B. Pacejka, *Tyre and Vehicle Dynamics*, 2nd ed. Elsevier Ltd., 2006.
- [7] R. N. Jazar, *Vehicle Dynamics, Theory and Application*. Springer, 2008.
- [8] R. G. Longoria, “Steering and turning vehicles,” Lecture Notes from the course Vehicle System Dynamics and Control, 2012.
- [9] Trafikkontoret, Göteborgs Stad, *TRI-Buss, Trafikinstruktioner för buss*, December 2006.
- [10] D. M. Gerdt, “The single track model,” Universität Bayreuth, Tech. Rep., 2003.
- [11] K. Popp and W. Schiehlen, *Ground Vehicle Dynamics*. Springer, 2010.
- [12] R. Rajamani, *Vehicle dynamics and control*. Springer Science, 2012.
- [13] M. B. Oetiker, G. P. Baker, and L. Guzzella, “A navigation-field based semi-autonomous non-holonomic vehicle-parking assistant,” 2009.
- [14] A. M. Bloch, J. E. Marsden, and D. V. Zenkov, “Nonholonomic dynamics,” 2005.
- [15] M. Juliá, A. Gil, and O. Reinoso, “A comparison of path planning strategies for autonomous exploration and mapping of unknown environments,” 2012.
- [16] V. J. Lumelsky and A. A. Stepanov, “Path-planning strategies for a point mobile automation moving amidst unknown obstacles of arbitrary shape,” 1987.
- [17] E. J. Rossetter, “A potential field framework for active vehicle lanekeeping assistance,” Ph.D. dissertation, Stanford University, 2003.
- [18] *Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation*, 1991.
- [19] M. Wahde, “Introduction to autonomous robots,” Lecture Notes from the course Autonomous Agents, Chalmers university of technology.
- [20] N. Andréasson, A. Evgrafov, and M. Patriksson, *An Introduction to Continuous Optimization*. Studentlitteratur AB, 2005.
- [21] J. C. Principe, N. R. Euliano, and W. C. Lefebvre, *Neural and Adaptive Systems: Fundamentals Through Simulations*. John Wiley and Sons, Inc., 2000.

- [22] J. W. Gebhard, “Acceleration and comfort in public ground transportation,” 1976.
- [23] (2012). [Online]. Available: <http://www.mathworks.se/academia/arduino-software/>
- [24] Z. Zeng and Y. Li, Eds., *Roadside curb detection based on fusing stereo vision and mono vision*, 2012.
- [25] *Ultrasonic Sensor*, Murata Manufacturing Co., Ltd., 2008.
- [26] J. Fonseca, J. S. Martins, and C. Couto, “An experimental model for sonar sensors,” 2001.
- [27] L. Alonso, V. Milanés, C. Torre-Ferrero, J. Godoy, J. P. Oria, and T. de Pedro, “Ultrasonic sensors in urban traffic driving-aid systems,” 2011.
- [28] *XL-MaxSonar-WRMA1 MB7092 Datasheet*, Maxbotix, 2012.
- [29] Pepperl Fuchs, *Ultrasonic Sensor, UC2000-30GM-IUR2-V15*, 2012.
- [30] Västtrafik, *Handbok för hållplatser utformning*, 2nd ed., 2006.
- [31] Pepperl Fuchs, *Instruction Manual, Service Program Ultra 2001*, 2004.
- [32] Volvo, *Volvo 7900 Hybrid Specifications*, 2013.
- [33] Västtrafik and Vägverket, *Handbok om farthinder*, 2003.
- [34] V. Pradeep, G. Medioni, and J. Weiland, “Piecewise planar modeling for step detection using stereo vision,” 2008.
- [35] R. Collins, “Introduction to computer vision,” Lecture notes from the course Computer Vision I, Penn State University.
- [36] G. Bradski and A. Kaehler, *Learning OpenCV, Computer Vision with the OpenCV Library*. O’Reilly Media, 2008.
- [37] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
- [38] M. T. Tham, “Dealing with measurement noise,” Technical information about noise filtering from Newcastle University, 2009.
- [39] G. Welch and G. Bishop, “An introduction to the kalman filter,” 2006.
- [40] T. Glad and L. Ljung, *Control Theory*. Taylor & Francis, 2000.
- [41] Interactive Matter Lab, “Filtering sensor data with a kalman filter,” 2009. [Online]. Available: <http://interactive-matter.eu/blog/2009/12/18/filtering-sensor-data-with-a-kalman-filter/>

Appendix

A The MATLAB interface explained

The interface used for the implementation in the car was built using MATLAB. MATLAB has a tool called MATLAB GUI for this which is simple to work with. This part will explain the layout and functions of the interface. In Figure 50, the interface has just been started up.

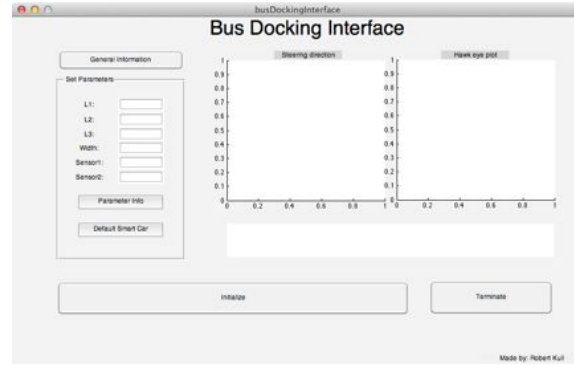


Figure 50: The MATLAB interface at start up.

In this view, we see the basic layout of the interface. Basically it consists of;

1. General information button; by pushing this button, an information box of how to use the interface will pop up.
2. “Set parameters” panel; in this panel, the user can input the specific parameters needed for the system to operate, these are:
 - (a) L1: The length between the front of the vehicle and the front wheel axis.
 - (b) L2: The length between the front and rear wheel axis.
 - (c) L3: The length between the rear of the vehicle and the rear wheel axis.
 - (d) Width: The width of the vehicle.
 - (e) Sensor1: The length between the front sensor position and the rear wheel axis.
 - (f) Sensor2: The length between the rear sensor position and the rear wheel axis.
- (g) “Parameter information” button; by pushing this button the same information as in (a)-(f) will pop up in a message box.
- (h) “Default Smart Car” button; by pushing this button, the interface will fill out the parameters according to the test with the Smart car done in 3.2.2.

Note: all of the parameters should be set in meters.

Example: Please go back and look at Figure 13. That vehicle would give the parameters; $L1 = L1$, $L2 = L$, $L3 = L2$, $Sensor1 = L + L1$ and $Sensor2 = L$.

3. “Steering direction” plot window; in this window, the direction that the user should move the steering wheel will appear as a line.
4. “Hawk eye” plot window; in this window, the vehicle will be plotted along with the curb as is seen from above.
5. The white area underneath the plot windows is an information field. In this field, the user will get information about the current process of the system.
6. “Initialize” push button; by pushing, the system will connect with the Arduino. This button will change its appearance during the docking, see forward.
7. “Terminate” button; by pushing, the session will be terminated.

If the user first push the “Default Smart Car” button and then hit “Initialize”, the interface will change its display and show the information seen in Figure 51.

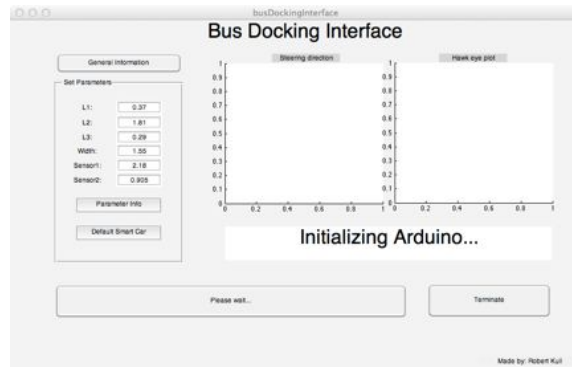


Figure 51: MATLAB interface initializing.

When the system has initialized and connected to the Arduino board the interface will be displayed as in Figure 52.

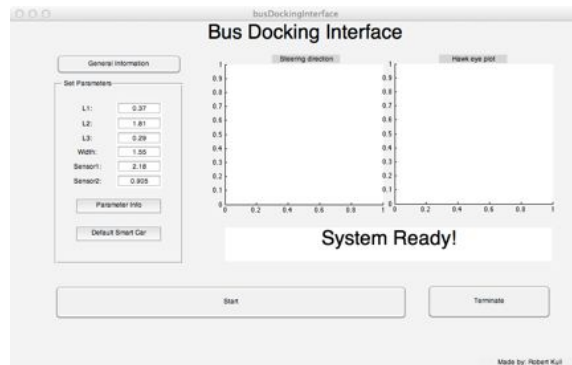


Figure 52: MATLAB interface initialized and ready.

The system is now ready to be used and by hitting the “Start” button, the sensors will start to search for the curb. The interface will now have three different looks depending on what stage of the docking the vehicle is in.

Figure 53 displays the first stage, when the sensors are searching but have not yet found the curb.

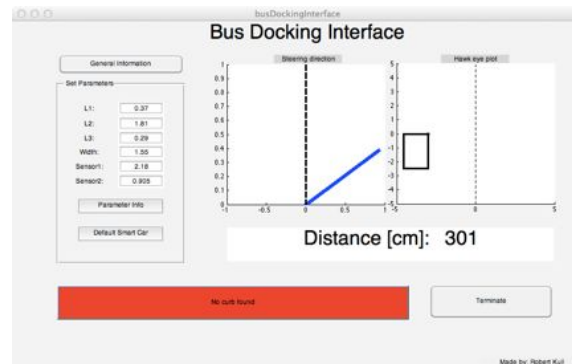


Figure 53: MATLAB interface when no curb has yet been found.

Note how the push button that was named "Start" in the earlier stage now reads "No curb found" and has changed color to red. Also, the information field now displays the distance from "Sensor1" to the curb (the dashed line in the "Hawk eye plot"). The "Steering direction plot" wants the driver to turn quite a lot to the right which is reasonable.

In the next stage, the curb has been found and the interface will be displayed as in Figure 54.

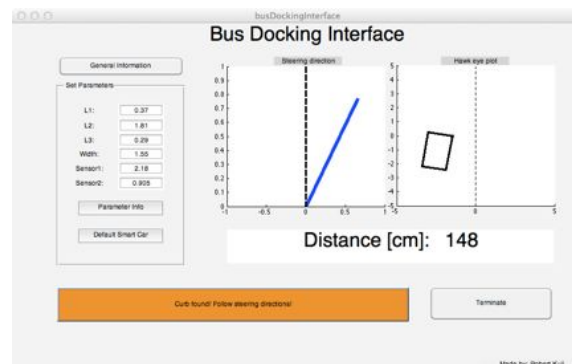


Figure 54: MATLAB interface when the curb has been found.

Note now how the push button once again has changed its color to orange and reads "Curb found! Follow steering directions!". Also note the changes in distance to the curb, steering direction and "Hawk eye plot". Moving forward a bit in this stage a scenario could become as in Figure 55.

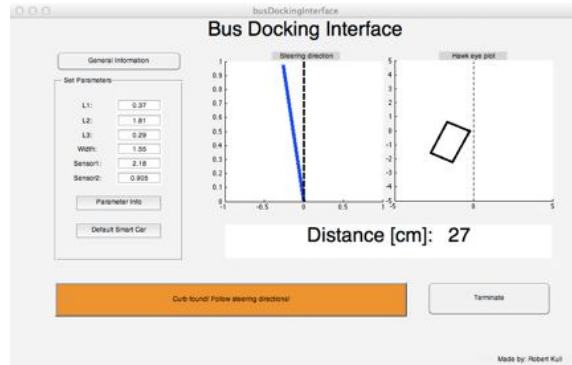


Figure 55: MATLAB interface, closer to the curb.

Now the vehicle is closer to the curb and the steering direction has changed. If the directions are followed correctly by the driver, the display of the interface will eventually change to the one seen in Figure 56.

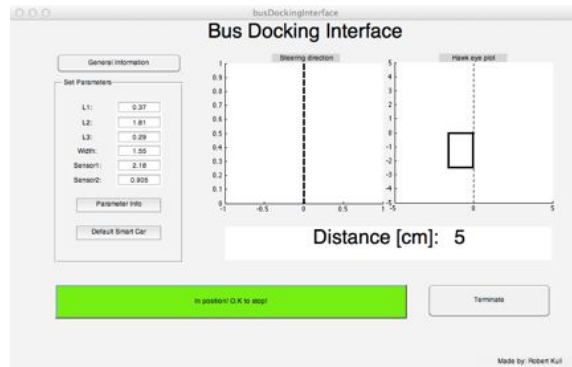


Figure 56: MATLAB interface when the wanted position and pose has been reached.

Once again the color of the push button has been changed, now to green, reading “In position! O.K. to stop!”. The driver then knows that he is in a good position and pose close to the curb. The current session can be terminated and the driver can leave his vehicle.