Design of a 360 degree obstacle detection system with haptic feedback

F. (Fabian) van Hummel

BSc Report

Committee:
Prof.dr.ir. G.J.M. Krijnen
Dr.ir. D. Dresscher
Dr. A.H. Mader

May 2018

008RAM2018
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE.

MIRA CTIT
BIOMEDICAL TECHNOLOGY AND TECHNICAL MEDICINE

Summary

The i-Botics center is an initiative of the University of Twente and TNO. The main goal of the i-Botics group is to design a remote controlled robot with telepresence technologies. In this, an operator should be able to operate the robot in a remote environment, while having the sensation of being present at the robot's location. This can be useful in situations which require human expertise but where safety of humans is not guaranteed. To be able to experience telepresence, feedback must be given to the operator. Multiple forms of feedback exist such as audio, visual and haptic feedback.

This thesis is a subpart within the larger i-Botics project and contains the design of a haptic feedback system for the detection of obstacles in the surrounding area of the robot. Such a system can be useful to prevent collisions between the robot and the environment, making sure that damage to both is prevented. The research done is design oriented and puts the focus on the design of the obstacle detection system. The software structure for the haptic feedback system is implemented but the implementation and testing of the haptic feedback algorithm is left for future work.

For the obstacle detection system, it is necessary to have sensors on the robot. Based on previous work, an ultrasonic sensor is chosen to fulfill this task. In the analysis, the important scanning area around the robot is determined. Based on this, a sensor configuration is chosen. An Arduino is used to interface the sensors' data measurements.

A software system is implemented in the production phase. The sensor data measurements are translated to an obstacle list in a base frame. Filtering of obstacles is done next as it is decided to only give haptic feedback on obstacles in the path of motion. For obstacles in the path of motion a path length is calculated. The path length is seen as the distance from the robot to an obstacle. Haptic feedback will be based on this path length and on the velocity of the robot. The velocity is received from the IMU sensor.

Experiments were conducted to test the performance of the system. It appears that the system responds as expected but that includes limitations. The main issue is the calculation for the path length to an obstacle. To keep track of time, simplifications are made. These result in large, but predictable, path length errors. There are 3 parts which contribute to the path length errors. The first introduced path length error is caused by the assumed path starting point, resulting in a maximum error of 0.647 m. The second introduced path length error is caused by the calculations for obstacles in the swerving area, resulting in a maximum error of 14 m. The final introduced path length error is caused by the assumed angle of an obstacle. It turns out that in some cases the theoretical assumed angle is so far off from the actual angle, that the system ignores obstacles which lie in the path of motion of the robot. This results in collisions between the robot and obstacles.

The conclusions is drawn that a proper structural basis is given in which all the elements needed to translate sensor data to haptic feedback is included. However, some improvements are needed to further reduce the path length errors. Therefore, multiple recommendations for future work are given. These include the need for a different angle approximation method such as a worst case assumed angle. Next, a simple check must be implemented that determines whether an obstacle is in the swerving area. Finally, The proper collision point on the platform must be determined. This can be done by using the calculated radii and angles.

Samenvatting

Het i-Botics centrum is een initiatief van de Universiteit Twente en TNO. Het hoofd doel van de i-Botics groep is om een op afstand bestuurbare robot met telepresentie technologieën te ontwerpen. Hierin moet de bestuurder de robot op afstand kunnen besturen en tegelijkertijd het gevoel krijgen aanwezig te zijn op de locatie van de robot. Dit kan nuttig zijn in situaties die menselijke expertise vereisen maar waar veiligheid van mensen niet gegarandeerd kan worden. Om telepresentie te ervaren moet feedback worden gegeven aan de bestuurder. Verschillende vormen van feedback zoals audio, visuele en haptische feedback bestaan.

Deze scriptie is een subonderdeel binnen het grotere i-Botics project en gaat over het ontwerp van een haptisch feedback systeem voor het detecteren van obstakels in het nabijgelegen gebied van de robot. Dit systeem kan nuttig zijn om botsingen tussen de robot en de omgeving te voorkomen en zorgt ervoor dat schade aan beide voorkomen wordt. Het onderzoek is ontwerpgericht en de focus ligt bij het ontwerp van het obstakeldetectiesysteem. De software structuur voor de haptische feedback is geïmplementeerd maar de implementatie en het testen van het haptische feedback algoritme zullen in de toekomst moeten worden gedaan.

Voor het obstakeldetectiesysteem zijn er sensoren nodig op de robot. Er is gekozen voor een ultrasonische sensor en deze keuze is gebaseerd op eerder werk. In de analyse is het belangrijke scangebied rondom de robot gedefinieerd. Gebaseerd op het scangebied is een sensorconfiguratie gekozen. Een Arduino zal dienen als interface voor de data metingen van de sensoren.

In de productiefase is het softwareonderdeel geïmplementeerd. De datametingen van de sensor zijn omgezet naar een obstakellijst in een basis coördinatenstelsel. Hierna worden obstakels gefilterd omdat de keuze is gemaakt om alleen feedback te geven op obstakels in het pad van beweging. Voor obstakels in het pad van beweging worden de padlengtes berekend. De padlengte kan gezien worden als de afstand van de robot tot aan het obstakel. De haptische feedback wordt uiteindelijke gebaseerd op 2 componenten: de afstand tot een obstakel en de snelheid van de robot. De padlengte wordt gebruikt voor het afstands component. De snelheid van de robot wordt doorgegeven door de IMU sensor. Het combineren van beide componenten geeft de input voor de haptische feedback.

Experimenten zijn uitgevoerd om de functionering van het systeem te testen. Het blijkt dat het systeem zich gedraagt zoals verwacht, echter kwamen hiermee ook de bijbehorende beperkingen naar voren. Het voornaamste probleem is de berekening van de padlengte naar een obstakel. Om de tijd in de gaten te houden, zijn er simplificaties uitgevoerd. Hierdoor zijn er significante, maar voorspelbare, padlengtefouten geïntroduceerd. Er zijn 3 onderdelen die bijdragen aan de padlengtefouten. De eerste geïntroduceerde padlengtefout wordt veroorzaakt door het aangenomen padstartpunt, dit resulteert in een maximale fout van 0.647 m. De tweede geïntroduceerde padlengtefout wordt veroorzaakt door de berekening voor obstakels in het zwenkgebied, dit resulteert in een fout van maximaal 14 m. De laatste geïntroduceerde padlengtefout wordt veroorzaakt door de veronderstelde hoek van een obstakel. Het blijkt, dat in sommige gevallen, de theoretische veronderstelde hoek zo ver van de werkelijke hoek af ligt, dat het systeem obstakels negeert die in het pad van beweging liggen. Dit resulteert in botsingen tussen de robot en obstakels.

Concluderend, er is een degelijke structurele basis gegeven waarin alle benodige elementen aanwezig zijn die sensordata omzetten naar haptische feedback. Echter, een paar verbeteringen zijn nodig om de padlengtefouten te verminderen. Daarom zijn er meerdere aanbevelingen voor toekomstig werk. Een andere hoekbenaderingsmethode is nodig, dit kan bijvoor-

beeld een worst case veronderstelde hoek zijn. Daarnaast moet er een simpele check worden gedaan of er een obstakel in het zwenkgebied zit. Als laatste, moet het juiste botsingspunt op het platform worden bepaald. Dit is mogelijk door gebruik te maken van de berekende radii en hoeken.

Contents

1 Introduction

1.1 Context

The I-botics center is an initiative of TNO and the University of Twente. The aim of the I-botics center is to design a robot with telepresence technologies such that the operator can operate the robot from a distant place while still having the sensation of being in a remote environment. The focus in this research is on the design of a subsystem in this larger project. To be able to experience telepresence, feedback must be given to the operator of the robot. There are many different feedback implementations such as audio, visual and haptic feedback. This research focuses on the design of a haptic feedback system for the detection of obstacles.

1.2 Project goal

When driving in a remote environment, it is likely that the platform encounters obstacles. Detecting and avoiding these obstacles is essential as colliding with the obstacle could result in damage to the platform or environment. An obstacle detection system can support the operator in this by detecting obstacles and making their presence known to the operator. The goal of this particular project is therefore to extend the existing 1 degree of freedom obstacle detection sensory system to a full 360° obstacle detection system. With this detection system, haptic feedback is provided to notify the operator where obstacles are located in the surrounding area without having any visual information. The system aims on supporting the user when trying to avoid obstacles rather than driving autonomously.

1.3 Approach

The first steps on designing this obstacle detection system have already been done and all the information about the existing system can be found in [6]. This system uses one ultrasonic sensor which is mounted at the front of the robot's platform. This allows detection of obstacles at the front of the platform. The existing system needs to be extended such that 360° obstacle detection is realized. Therefore multiple sensors need to be added to the sides and back of the platform. Consequently, research needs to be done on the feasibility of the use of multiple ultrasonic sensors. Also, the currently existing haptic feedback feature is designed for the use of one ultrasonic sensor. It must be extended for the use of multiple sensors and multiple obstacles. In addition to this, it is important to evaluate and redesign the algorithm used in the haptic feedback feature as multiple users stated a drop in the interpretation of the force especially at the closer regions to the obstacle. During this project, software is written in C++ and the middleware ROS is used as communication framework.

1.4 Research questions

The main goal of this design is to achieve 360° obstacle detection with haptic feedback. The system is divided in two subsystems: the obstacle detection system that will detect obstacles and the haptic feedback feature to notify the operator of the presence of obstacles. To be able to achieve the main goal, research questions are formulated for each subsystem. These research questions are:

Main research question:

- How to design a properly functioning 360° obstacle detection system with an intuitive feeling haptic feedback implementation?

Obstacle detection system:

- What area around the platform is of interest when trying to detect obstacles?
- How many sensors with what individual position and orientation are needed to be able to properly scan this area of interest?
- How are interference and dead zones influencing the performance of the sensor setup? How can this be tested?

Haptic feedback:

- What sensor data processing must be done such that it can be used to generate haptic feedback based on the location of one or more obstacles?
- On which obstacles does the system need to give haptic feedback?

1.5 Report outline

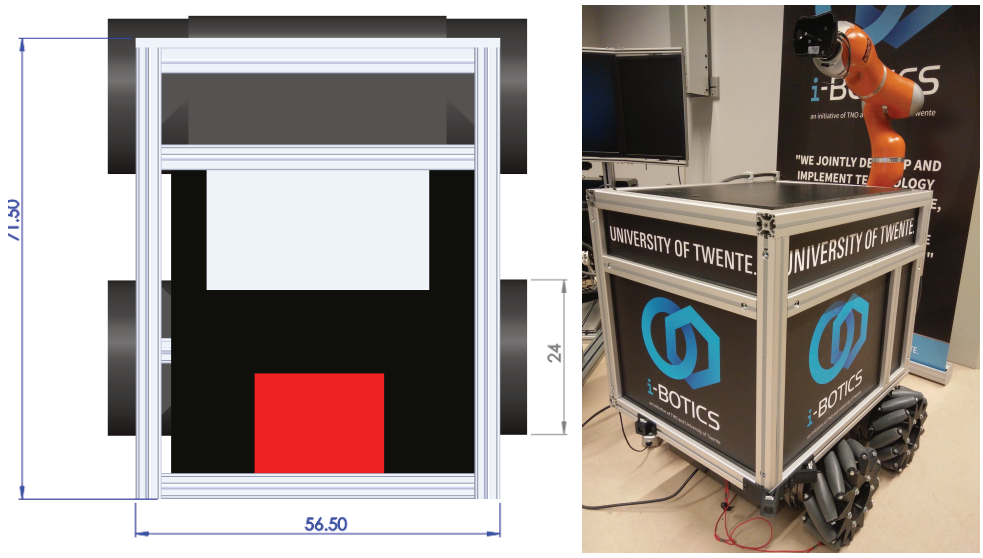
In Chapter 2, the analysis part for the obstacle detection system is done. In this, the requirements will be formulated and the general approach is given. In Chapter 3, the steps towards the design and implementation of the obstacle system is outlined. Next, experiments are conducted and these will be presented in Chapter 4, together with the results. In that same chapter, a discussion about the results is given. Based on the results, conclusions can be drawn and recommendations can be given, this is done in Chapter 5.

2 Analysis

In this chapter, the steps towards the conceptual design of the obstacle detection system are presented. First, the requirements on the obstacle detection system are discussed and afterwards the used sensor is characterized. Based on this, a sensor configuration is chosen. Finally, a decision is made on how to interpret the measurements done by the sensor. This all gives a basis for the design phase of the system. The haptic feedback feature will not be implemented due to time constraints and only some initial ideas are presented.

2.1 Robotic setup

The robotic setup is shown in Figure 2.1. In Figure 2.1a a drawing for the top view of the platform is shown. The dimensions of the platform are 71,50 cm by 56,50 cm and via its 4 mecanum wheels it is able to move around. In Figure 2.1b, the physical system is shown. The horizontal bottom aluminum bars can be used to mount ultrasonic sensors on. Haptic feedback will be given on the input device that actuates the platform. This could either be an omega device or pedals.



(a) Top view platform

(b) Platform used

Figure 2.1: Robotic setup

2.2 Existing system

As mentioned in the introduction part, the existing system exists of a single ultrasonic sensor mounted at the front of the platform. With this, obstacles can be detected at the front of the platform. When an obstacle is detected, the ultrasonic sensor will measure a distance to the obstacle. The distance measurement from the sensor is presented as an analog voltage signal and is send to an Arduino Uno. The Arduino Uno translates the analog voltage to a distance and afterwards it sends the information via ROS to the PC. The control system of the platform runs on the PC. This control system makes sure that the user input is read to actuate the platform and haptic feedback is given to the user. The haptic feedback implementation consists of a linear force feedback algorithm. The force algorithm exists of the sum of 2 individual components. One component is based on the distance to an obstacle and the other component is based on the velocity of the platform. Both these components have a linear relation with the

force. The equation used for haptic feedback is shown in Equation 2.1:

$$F = c \cdot \frac{dx}{dt} + k \cdot x \tag{2.1}$$

. The velocity is included because the operator has different response times at different velocities. An overview of the haptic feedback system is shown in Figure 2.2. For additional information on the design of the existing system, see [6].

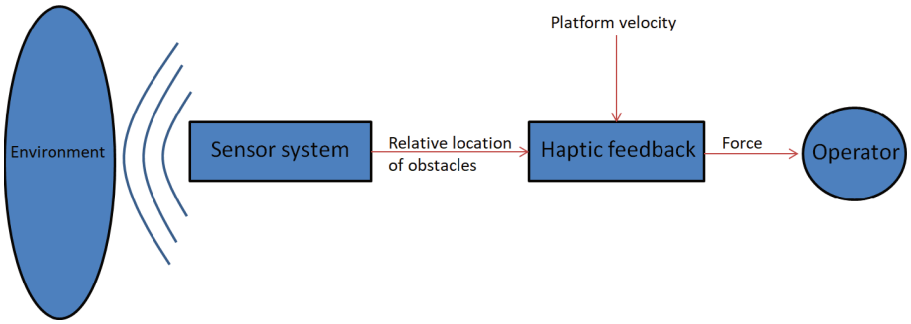


Figure 2.2: Haptic feedback system overview

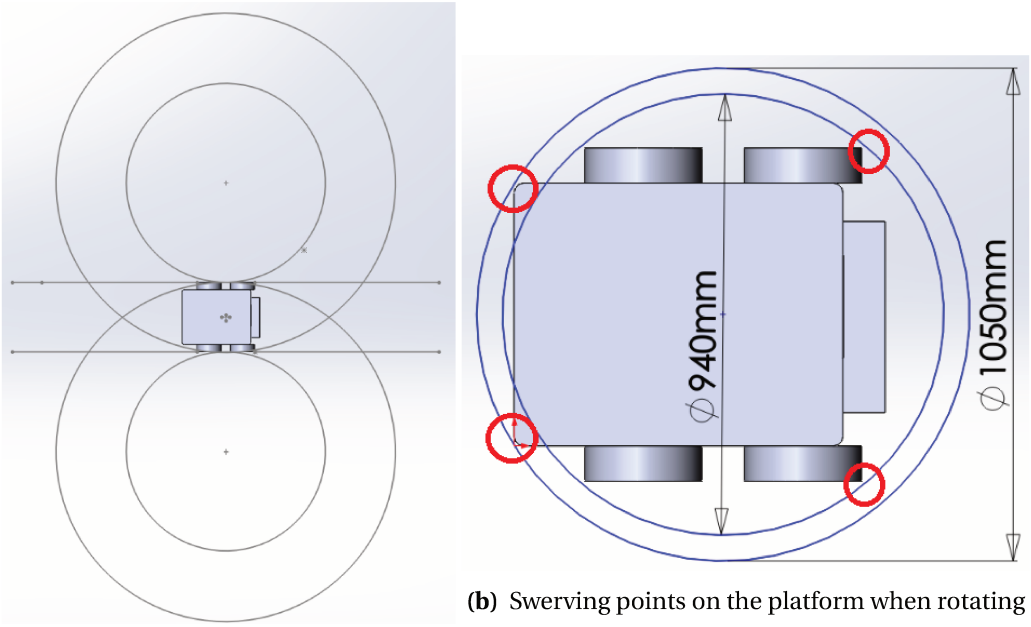
2.3 Obstacle detection system

In this section the requirements on the sensor system are discussed. Based on this a sensor configuration is chosen.

2.3.1 Area of interest

Before going into detail about the sensor configuration, it must be clear what particular areas around the platform are important to scan when trying to prevent collisions. This area of interest is dependent on the motion capabilities of the platform. By knowing the motion capabilities of the platform, it can be determined which obstacles form a threat to the platform. Therefore, it is necessary to determine the motion capabilities first. This will be done now.

The movement of the platform is assumed to be differential. This means that the platform is able to move forward and backward but also rotate around its axis. Any combination of these is also possible resulting in a circular motion with a radius. Now the movement has been defined it is possible to address the important areas which need scanning. In Figure 2.3b a top view of the system is shown. The red circles show the points which are most important when rotating as they define the amount of swerving. These are: the back corners and front wheels. The blue circles represent the space that the platform needs when rotating for each mentioned point. The sensor system should thus scan the area within the outer blue circle. The next movement is translation which can be seen in Figure 2.3a, where the straight lines represent the forward and backward movement. A combination of translation and rotation is shown with the circles in Figure 2.3a. The motion capabilities of the platform can thus be seen as the set of different circular movements. These motion capabilities are of importance when defining an area of interest.



(a) Translation with rotation

(b) Swerving points on the platform when rotating

Figure 2.3: Movements of the platform [3]

There are many combinations possible of translation and rotation, which each result in a different path of motion. The total of all these different paths of motion form an area around the platform which needs to be scanned. The shape and size of this area must be determined before concluding which sensor configuration is needed.

The maximum velocity of a movement (combination of translation and rotation) is determined by the maximum wheel velocity. The maximum translational velocity is thus not the same for each radius r. This is important as it affects the area of interest. Now an equation will be derived relating the radius r and the corresponding maximum velocity via the maximum wheel velocities. This will give important information to determine the path length l.

The first step is to define a relation between the translational velocity v and the angular velocity ω. Any movement of the platform can be seen as a circular movement with a certain radius r. When the platform rotates around its axis thus having only angular velocity the radius approaches 0. When the platform moves forward, having only translational velocity its radius approaches infinity. To get to a relation between v and ω the path length l needs to be known. The path length l is defined according to Equation 2.2. The path length l represents the part of the circumference covered of the circle with radius r and the shape of the path covered given the translational velocity v, for a travel time t.

$$l = v \cdot t \tag{2.2}$$

This part of the circumference covered can also be defined given the angular velocity ω as in Equation 2.3.

$$l = \omega \cdot r \cdot t \tag{2.3}$$

The path length l calculations shown in Equation 2.2 and Equation 2.3 can be combined into Equation 2.4.

$$r = \frac{v}{\omega} \tag{2.4}$$

At this point the translational and angular velocity are related but individually still unknown. The next step is to determine how these velocities are defined on the platform. To be able to

do this a reference is made to previous work. [7] describes how to relate the wheel velocities to platform velocities and these can be used for further calculations. In Equation 2.5, it is shown what the relations are between the wheel velocities and the planar velocities. Figure 2.4 shows a visualization of the platform and the corresponding parameters. With this information it is now possible to determine the velocities of interest.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} -1 & 1 & -(l_1+l_2) \\ 1 & 1 & l_1+l_2 \\ 1 & 1 & -(l_1+l_2) \\ -1 & 1 & l_1+l_2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{2.5}$$

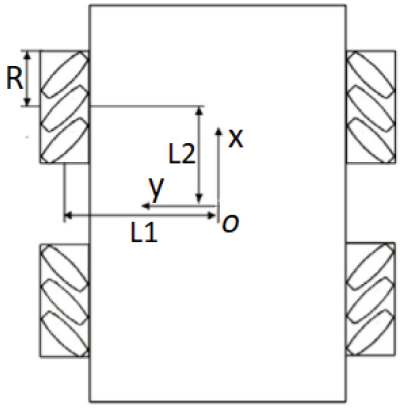


Figure 2.4: platform dimensions [7]

According to Equation 2.5 we have the following relation:

$$\omega_2 = \frac{1}{R} \cdot (v_x + v_y + (l_1 + l_2) \cdot \omega_z) \tag{2.6}$$

With wheel radius $R = 0.125\ m$, $l1 = 0.3\ m$ and $l2 = 0.139\ m$. ω_2 is the velocity of wheel 2. Each wheel can rotate at a maximum velocity w_{max}, which then restricts the sum of v_x, v_y and w_z. In case of a pure translational velocity this would result in: $\omega_{max} = v_{max}/R = 0,83/0,125 = 6,64\ rad/s$. ω_z is the angular velocity around the vertical z axis. v_x is the forward translational motion and v_y the sideways translational motion. Sideways motion is not used and thus v_y is 0. Rewriting Equation 2.6 and replacing ω_2 with ω_{max}, gives the maximum rotational and maximum translational velocity for maximum wheel velocity.

$$\omega_z = \frac{\omega_{max} \cdot R - v_x}{(l_1 + l_2)} \ or \ v_x = \omega_{max} \cdot R - (l_1 + l_2) \cdot w_z \tag{2.7}$$

Now it is known how the relations are defined with respect to the platform, Equation 2.7 can be substituted into Equation 2.4. Giving the following result:

$$r = \frac{\omega_{max} \cdot R - (l_1 + l_2) \cdot \omega_z}{\omega_z} = \frac{v_x(l_1 + l_2)}{\omega_{max} \cdot R - v_x} \tag{2.8}$$

rewriting Equation 2.8 gives the following result:

$$r(\omega_{max} \cdot R - v_x) = v_x(l_1 + l_2) \tag{2.9}$$

$$r \cdot \omega_{max} \cdot R - r \cdot v_x = v_x(l_1 + l_2) \tag{2.10}$$

$$r \cdot \omega_{max} \cdot R = v_x(r + l_1 + l_2) \tag{2.11}$$

$$v_x = \frac{r \cdot \omega_{max} \cdot R}{r + l1 + l2} \; with \; \omega_{max} \cdot R = v_{max} \tag{2.12}$$

With this final equation the maximum translational velocity of the platform is known for each radius r. This gives a basis to define the area of interest. To further investigate this, the minimum travel time t must be defined. This is defined as the minimum time frame in which the operator should experience haptic feedback a priori a collision. The assumption is made to have a minimum travel time of 5 seconds.

With the minimum travel time, the minimum path length l is calculated. This is done via Equation 2.13, with v_x calculated in Equation 2.12 and minimum travel time t.

$$l = v_x \cdot t \tag{2.13}$$

Now varying the radius r of the circle at which the platform is moving on, paths for different platform motions can be determined. To get to the total area of interest, each radius varying from 0 to infinity is evaluated and drawn in a plot. The result is shown in Figure 2.5. The entire yellow area indicates where the platform is able to move within the minimum travel time of 5 seconds. The area of interest for a minimum travel time of 3 seconds is also shown which is the area within the inner black contour. Drawing the 3 second minimum travel time gives a proper visualization on how the minimum travel time will influence the area of interest around the platform.

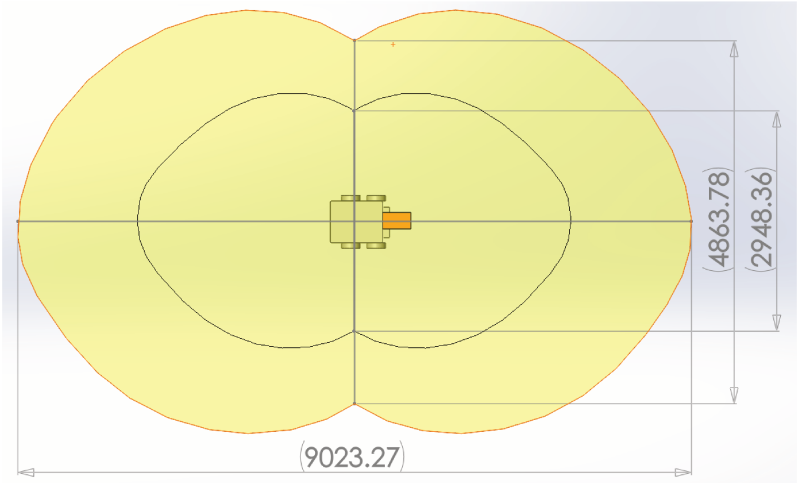


Figure 2.5: Area of interest in perspective, dimensions given in mm [3]

Figure 2.5 shows a representation of the area that must be adequately covered by the sensor system. Based on this and the characterization of the sensors, a decision can be made on where to place sensors and what their individual orientation is. The characterization of the sensors is done in the next Section and the decision for the configuration and orientation of the sensors will be presented in Section 2.3.3.

2.3.2 Sensor characterization and interface

In this section, the characterization and interface of the ultrasonic sensor is discussed. The specific sensor used right now is the EZ1 which is also called the MB1010. This sensor and series is fabricated by MaxBotix. A switch is made towards the MB1000 in the same series as this sensor has a wider beam pattern when comparing it to the MB1010. This is done to decrease the number of sensors needed.

In Figure 2.6 is shown how the beam pattern (detection area) of the sensor is shaped for certain objects. In each scenario A, B and C, the sensor is positioned at the bottom (middle) of the grid. Pattern A is measured with a dowel (cylindrical rod) of 6.1mm as object while pattern B is measured with a dowel of 1 cm as object. It is mentioned by MaxBotix that the beam pattern size and shape for human detection lies between pattern A and B. The assumption is now made that pattern A is an "at least" requirement. This means that the area around the platform must be at least covered when having radiation pattern A. In other words each sensor must exactly touch or have overlap with its neighbor sensors when applying beam pattern A. With this statement, it should thus be possible to detect at least humans and dowels of 6.1mm diameter. As different materials and shapes will influence the radiation pattern it could be possible that some obstacles have an even smaller detection area or are not detected at all.

When implementing the sensor beam patterns into the drawing of Figure 2.5 a few things became evident. The first point is that using the MB1000, the system needs at least 10 sensors to cover the full 360° range, assuming the at least case scenario with beam pattern A. So when proposing configurations (which will be done in the next section), the least amount of used sensors is 10. Another remark is about the area close to the platform. It can be seen in Figure 2.6 that the sensor has a small beam width in the first few inches. The data sheet also states that any distance closer than 6 inch is represented as 6 inch. It will thus be difficult to detect obstacles in this area. This should be kept in mind when proposing configurations.

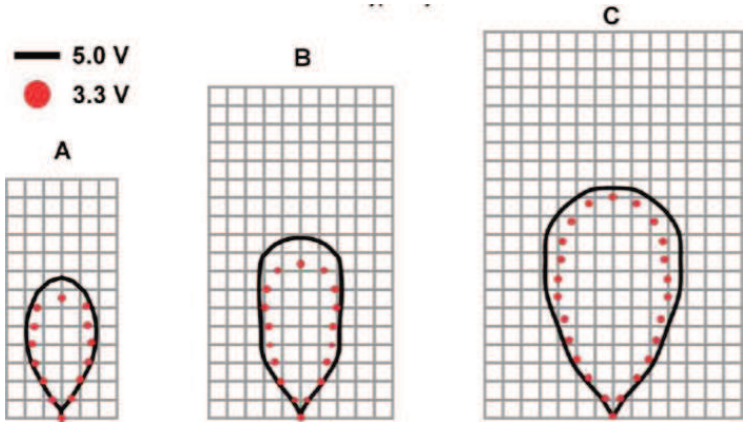


Figure 2.6: Radiation patterns MB1000 for obstacles A: dowel 6.1mm, B: dowel 1 cm, C: dowel 8.89cm on a 30 cm grid [1]

For the interface of the sensors there are multiple data representations and these are listed below:

- Analog voltage
- Pulse width
- RS232 Serial

To visualize each of these data representations a scheme is given in Figure 2.7.

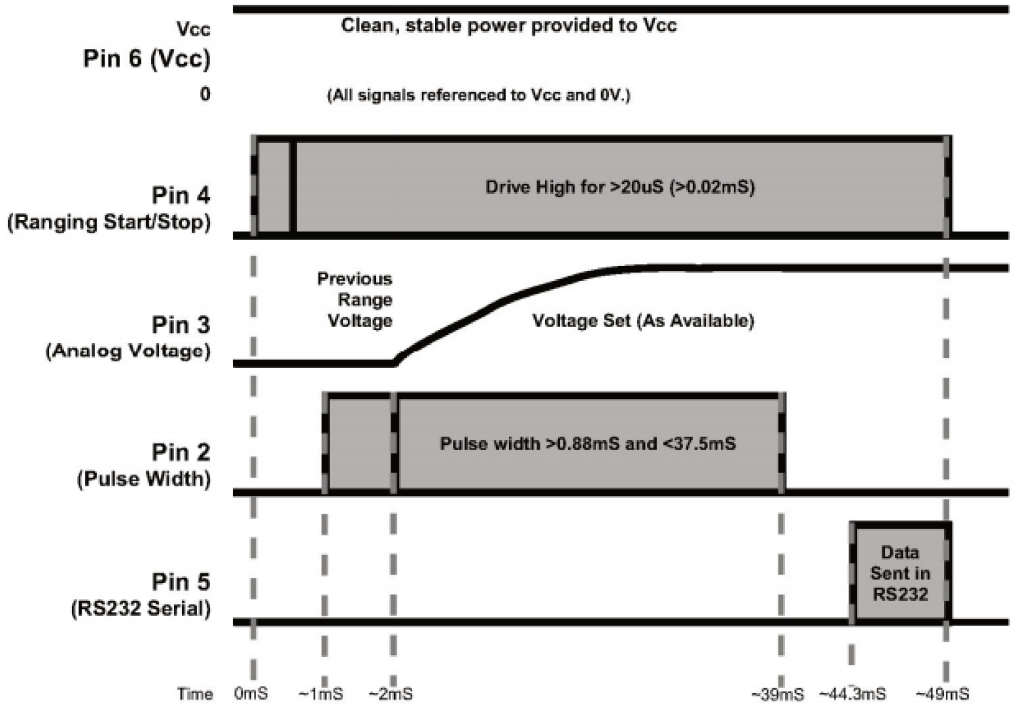


Figure 2.7: Output data representations with their respective time instances [1], (S=seconds)

The first data representation is analog voltage, where the distance to an obstacle is represented by a linearly scaled voltage between 0 and V_{cc}.The scaling factor is $V_{cc}/512$ per inch. With a power supply of 5V this results in 9.8 mV/inch. The problem with this method is that when the difference in distance to an obstacle between two measurements is large, the analog voltage needs time to reach the set point value. The possible consequence could be that incorrect data is being collected.

The data representation Pulse Width (PW) uses the width of a pulse to represent the distance to an obstacle. After 0.88 ms the PW pin is set high. If no obstacle has been detected it will keep the PW pin high for 37.5 ms. When an obstacle has been detected it will pull the pin low. The distance to an obstacle is represented as the time the PW pin is pulled low. The scaling factor is equal to 147uS per inch. One advantage of this method is that it doesn't have timing issues which the analog voltage implementation has.

RS232 Serial is a communication protocol which uses serial communication for transmission of data. the TX output pin delivers asynchronous serial with an RS232 format, with an exception for the voltage 0 - Vcc. The TX pin will output an ASCII capital "R" which is followed by three ASCII characters ranging to a maximum value of 512. These ASCII characters represent the distance to an obstacle in inches. Also in the work previously done an Arduino Uno is used and this Arduino does not have a RS232 interface. An RS232 interface should be added for each individual sensor. This would add complexity to the system.

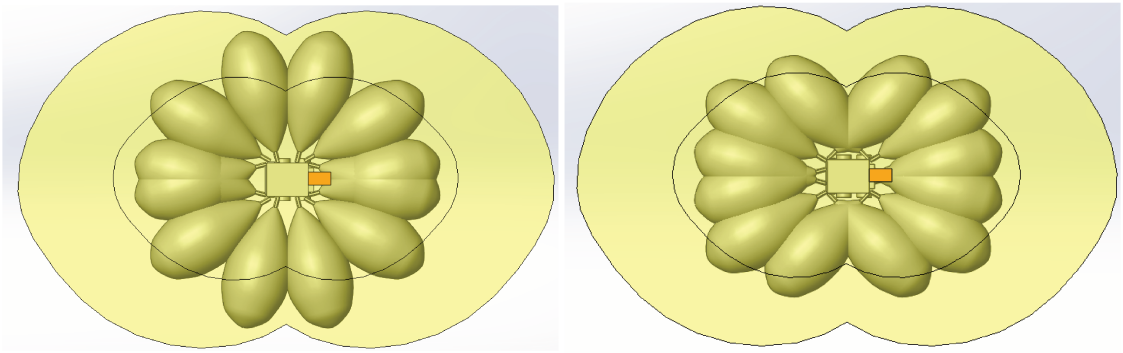
The decision is made to go for the pulse width method. This method ensures proper data which the analog input method does not. The analog input needs time to reach its set point and it is thus possible that the measured value has not (yet) reached the actual value and it is more inherent to noise. The RS232 method is also not chosen as it introduces complexity to the system which is unnecessary. The pulse width method is much more simpler than the RS232 method and it does the job.

2.3.3 Sensor configuration

In the previous sections, the sensor characterization and the area of interest is discussed. In this section different sensor configurations will be discussed to visualize the area coverage at which obstacles can be detected. Based on this, a sensor configuration is chosen. There are three parameters when designing such a configuration, namely: the amount of sensors, position of sensors and their individual orientation. This design space is too large to cover systematically and therefore we will be searching for a configuration with minimum sensors that covers the area of interest shown in Figure 2.5. In general, there seems to be a trade-off between overlap of sensors and optimizing the coverage of the area of interest. If the area is not completely covered, there is a chance of obstacles being undetected. On the other hand interference issues may occur when sensors have overlap in their scanning area.

In Figure 2.8 the comparison is made between a configuration with crossing side sensors and a configuration with non-crossing side sensors, both using 12 sensors. Between those configurations, there is a trade off between covering more at close distance (< 20 cm) or far distance. The ultrasonic sensor has a small beam pattern when measuring at close distance. Any distance measured under 15 centimeter is represented as 15 centimeter, this can also be seen in Figure 2.8. Figure 2.8a shows that it can measure further on the sides of the platform. On the other hand Figure 2.8b has better area coverage at close distance. Given that the area of interest is ellipse shaped the configuration in 2.8b has better overall coverage.

The other difference which can be seen when comparing Figure 2.8a and Figure 2.8b is the spacing between the front and back sensors. In 2.8a the sensors are positioned at the corner points of the platform while in 2.8b they are positioned near the center. In 2.8a there is more overlap between the sensors. The consequence is that in 2.8a a larger area is uncovered. The best solution to the spacing decision can be defined as: all sensors have overlap with their neighbor sensors and there is minimum uncovered area in the far and close area.

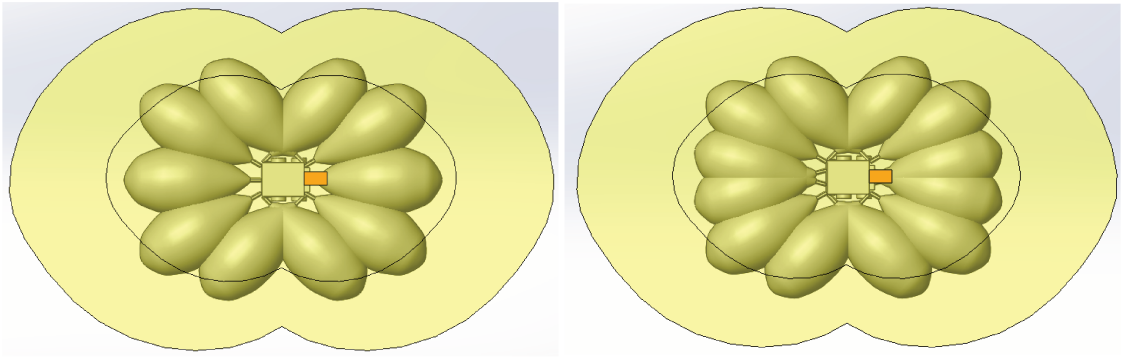


(a) 12 sensors with straight side sensors

(b) 12 sensors with crossed side sensors

Figure 2.8: Comparing straight and crossed side sensors with beam pattern A [3]

The next comparison is between 10 and 12 sensors, based on the configuration shown in Figure 2.8b. Figure 2.9a visualizes the configuration with 10 sensors. Because the sides of the platform are longer than the front and back it became evident that at least 2 sensors needed to be placed on the sides. For this reason the only option to cover the entire area with 10 sensors is to place 1 sensor at the front and back. It can be seen that there is overlap with each neighbor sensor which is the requirement for a valid configuration. When comparing this to Figure 2.9b (copy of Figure 2.8b), it can be seen that the area at front and back is covered less. Especially the dead zones emerging in the far distance are quite significant. There is thus a trade off between reducing the amount of sensors and dead zones emerging.

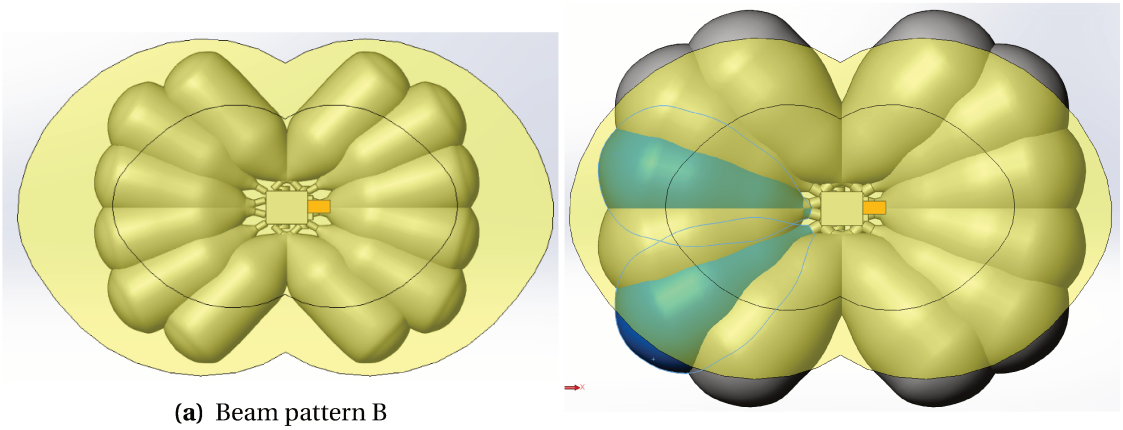


(a) 10 sensor configuration **(b)** 12 sensor configuration

Figure 2.9: Comparison between 10 and 12 sensors with beam pattern A [3]

Keeping the above in mind the configuration in Figure 2.9b is chosen. The reason for 12 sensors instead of 10 is because of the introduced dead zones at both far and close distance. Because an operator will typically have a forward motion, the front and back areas are quite important. So preventing the dead zones at these areas is necessary. For the same reason the front and back sensors will be positioned next to each other. Due to the ellipse shaped area of interest, the side sensors will be crossed, enabling more area coverage at close distance. The downside of choosing 12 sensors is, that the costs rise and that data processing time might increase. For now the consequences of this are not considered.

The last important element to mention is interference. It is hard to predict how much interference will be of an issue. When looking back at Figure 2.9b it can be seen that each sensor has overlap with its neighbor but not with their neighbor's neighbor. This is favorable as neighbor sensors need overlap to prevent dead zones but don't interfere with any other sensors. In the next sections methods will be mentioned to handle this neighbor overlap in software. The problem is that Figure 2.8 and Figure 2.9 only show beam pattern A. Beam pattern A is assumed as the smallest beam at which the system needs to detect obstacles. Any other beam is assumed to have a bigger area coverage and thus the possibility of having overlap with the neighbor's neighbor increases as well. To visualize this, beam patterns B and C are shown in Figure 2.10.



(a) Beam pattern B **(b)** Beam pattern C

Figure 2.10: Chosen sensor configuration with beam pattern B and C [3]

The blue encirclements in pattern C show the boundaries of each sensor beam. It can be seen that the overlap at the front and back between sensors becomes quite significant. It is also

unknown how the beam pattern is shaped when detecting other obstacles than the mentioned dowels. So the case of a beam pattern which is bigger than beam pattern C is to be considered. On the other hand the manufacturer quotes the following: *"The MaxSonar® sensors, because of continuously variable gain, will typically ignore adjacent sensors when running simultaneously"* [4]. This suggests that the sensors would ignore each others signal bursts and thus interference would typically not be a problem. Extensive testing needs to be done to be able to measure how much problems interference will give. In this research no extensive testing will be done and is left for future work. In the next section some possible software based solutions will be given in case interference causes problems.

2.3.4 Multi-operation use of sensors

In the previous section a sensor configuration is chosen and this showed that there will be overlap in the radiation area of neighbor sensors and in some cases also other sensors. This could cause interference when the sensors are operating simultaneously. The manufacturer claims that due to continuously variable gain it will typically ignore adjacent sensors and thus prevents interference problems. Because the manufacturer does not guarantee it, it could still be a problem and thus a backup plan is made. The manufacturer has multiple ways on how chaining of sensors could work without interference. These are:

- Free run operation
- Simultaneous Operation
- Commanded Sequential Reading
- Constant Looping Operation

The free run operation is commonly used when there is a single sensor operating in the area. If there are multiple sensors operating next to each other they are not synchronized because of frequency drifts. When a sensor has just started its cycle it will send out a burst. When this burst has been send out it will go into listening mode. While in listening mode it could be possible that another burst coming from a different sensor is being reflected back by an obstacle and received by the listening sensor.

Simultaneous operation is a method which is used when chaining of sensors is desired. This method enables synchronization of the sensors. All RX pins of each of the sensors are connected. This means that all sensors will start their burst at the exact same time. Their cycles are thus synchronized but still it could be that a reflected signal from a sensor is received by another nearby sensor. This could result in collecting incorrect data.

Commanded sequential reading is also a method which can be used when chaining sensors. The chain is put in a sequential configuration meaning that when the first sensor is triggered it will first finish its cycle before triggering the next sensor. The chain is triggered by an external device and after one iteration of the entire chain it will wait for the external device to trigger it again. This is a safe method because there won't be any chance on having interference problems. This is because there is always just one sensor operating. The downside for this method is that the overall update frequency will decrease with the amount of sensors used.

The last method is the constant looping iteration which uses the same principle as the commanded sequential reading but this one does not have to be triggered by the external device after each iteration. It will trigger the loop once and then the chain is continuously operating.

Another method could be to combine the Commanded sequential reading and the simultaneous operation. For example, 2 sequential pairs could be formed where the individual sensors in a pair operate simultaneous. This could be seen as an odd and even pair. Where sensor 1 is

in simultaneous operation with number 3,5 and so on. Sensor 2 is then paired with 4,6 and so on. This offers a trade-off between chances of interference and update frequency.

To determine the refresh time of the system for a given multi-operation, Figure 2.7 can be used. The figure shows that an entire cycle of the sensor will take 49 ms, resulting in a maximum update frequency of approximately 20 Hz. This is when using just the simultaneous operation. When using sequential chaining the frequency will drop. For example when using the two pair system mentioned in the previous paragraph, the frequency will drop to 10 Hz. This is because two pairs are sequentially operating and will thus wait until the other is finished before starting its measurement. Depending on the maximum velocity of the platform a minimum frequency is desired. The maximum velocity of the platform is 0.83 m/s. That means when having a measurement every 50 ms (20Hz) and the platform has maximum velocity the platform is able to move $83/20 = 4,15$ cm before the next measurement can be started. This calculation can also be done for the chaining configurations.

It is hard to say before actually testing the system which is the best solution as it depends on how troubling the interference issue will be and what the minimum refresh time is. But for now it is possible to go to a certain direction and if necessary a decision change can be made after testing the system. The best solution can be specified as the solution which has no interference issues but with the highest possible refresh rate. Keeping the above in mind it is chosen to implement the chaining method where 2 sequential pairs with 6 synchronous sensors is used. As this prevents neighbor sensors to operate at the same time. Any overlap between neighbor sensors is thus not important any more. The frequency of this system is half of the maximum frequency of the sensors. This corresponds to measuring every 8,3 cm when the platform is moving at maximum translational velocity. This is reasonable compared to the maximum of frequency with 4,15 cm per measurement. If there is still interference it would be possible to scale to 3 sequential pairs where 4 sensors operate synchronously. The method where all 12 sensors are operating synchronously will not be tested due to time constraints. In future work an experiment should be conducted to determine if the synchronous method does or does not cause interference problems.

2.3.5 Detection resolution

The last part to discuss about the obstacle detection system is the fact that the ultrasonic sensors only measure distance and not an angle. Figure 2.11 shows 12 vectors, which represent sensors. These sensors have fixed angles with respect to the platform. With this, it is possible to approximate an angle to an obstacle.

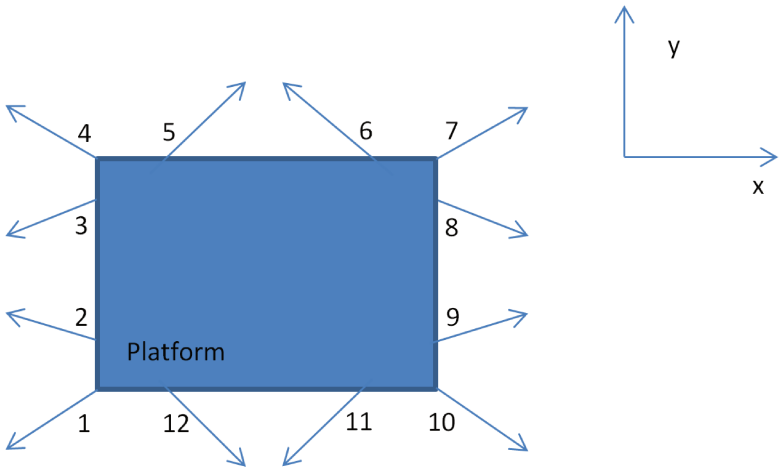
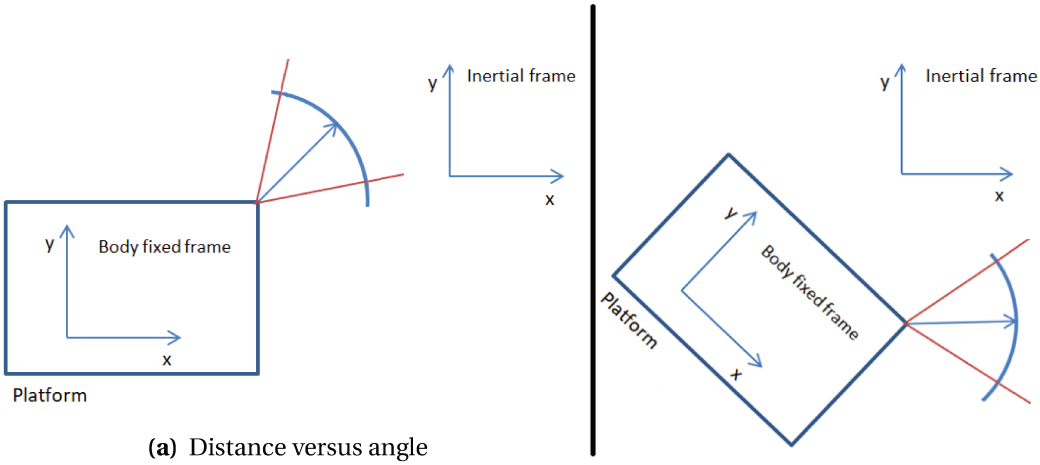


Figure 2.11: Directions of sensors

Figure 2.12a shows a single sensor where the red lines represent the beam width. When this sensor detects an obstacle, given a certain distance, the entire blue arch represents the possible position of the obstacle. At this point two scenario's can be distinguished which are:

- Obstacles measured by one sensor
- Obstacles measured by multiple sensors

In the first case an obstacle is measured by a single sensor. The area where the obstacle could be located is within the beam pattern of a single sensor. For example this could be the area within the red lines of Figure 2.12a.



(a) Distance versus angle

(b) Body fixed frame platform

Figure 2.12: Reference frame versus body fixed frame

Depending on the width of the beam of the sensor there is an according angle range. This can be used as an approximation of the angle of the obstacle. There is a downside to using this method as the beam width will vary much with different obstacles. In some cases, this method would have a large error margin. To estimate the error margin, probability theory could be applied. It is unknown which beam pattern is assigned to what obstacle but it is known for certain dowels, humans and an 11 inch wide board [1]. This could give a certain likelihood for each beam pattern if anything about the location is known. Based on this, the beam pattern width at any distance could be estimated, with which the error margin can be estimated.

The situation is slightly different when an obstacle is measured by multiple sensors . This occurs when an obstacle lies in the detection area of two sensors where their beam patterns overlap. Let's have a look at the example of Figure 2.11 where an obstacle lies between sensor 1 and 2 and in the detection area of both sensors. It is thus known that the angle will probably lie somewhere between the angle of sensor 1 and sensor 2. The easiest way of approximating the angle would be to take the average of the 2 angles. This is pretty much the same method as in the first case where the pre defined angle is taken as obstacle angle. Another approximating method would be to calculate the angle from the distance measurements of both sensors. In this case there would be two exact points in the plane at which the obstacles could be located. This can be visualized as drawing circles around the sensors with the diameter equal to the distance measured. The individual circles will intersect at two points at which the angle can be calculated. One point would lie on the platform and can be excluded.

The last method is using probability theory. If there is a likelihood that a beam pattern is occurring in the measurement, the overlap area between two sensors determine the possible angle range of the measurement. With this, the probability of an angle can be calculated. The angle with the highest probability could be the assumed angle. Again it is important to mention that

it is difficult to determine whether both sensors are detecting the same obstacle or a different obstacle. To keep track of time the decision is made to treat each measurement of sensors as a separate obstacle even when it is the same obstacle. So this scenario would not occur but in future work research could be done to handle this. To again keep track of time it is decided to assign the angle of the sensor to the obstacle.

2.4 System structure

In section 2.3 the analysis for the sensor system is done and the conclusion is drawn that it is now possible to locate obstacles in the area of interest. This gives the basis to start on the processing part of the system. In this section, the control structure of the entire system is presented. This to give an overview on the steps that are needed to translate sensor measurements to data, which can be presented to the haptic feedback component.

2.4.1 Control system

An overview of the control system that is used is given in Figure 2.13. This control system is also used in the existing system and therefore it is also be used in this design.

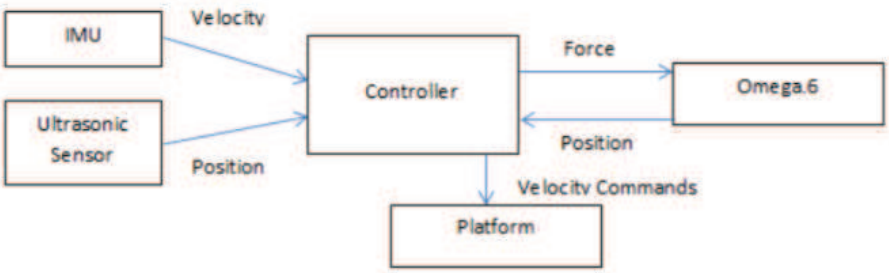


Figure 2.13: Extended system overview [6]

The first part of the system is to make sure the platform is able to move. This is done via the Omega haptic device. The user moves the Omega around and gives the controller a position input. The controller translates this position to a velocity set point and sends velocity commands to the platform. A detailed discussion of this system is out of scope for this assignment.

The other part of the system generates haptic feedback based on measurements of the ultrasonic sensors and the velocity received from the IMU. The controller calculates the feedback force and the Omega is set accordingly. How the controller determines the according feedback force is discussed in the next section.

2.4.2 Model for calculating force feedback

The model used to calculate the force feedback is visualized in Figure 2.14. Here the vertical fixed world represents an obstacle. When this obstacle is detected by the ultrasonic sensor a virtual spring and damper can be used to model the force feedback. When approaching the obstacle the spring is compressed and its counterforce depends on the displacement x of the spring. The damper is used to generate a force based on the velocity of the platform. In the linear case a formula as shown in Equation 2.14 represents the force feedback calculation. The damping factor c and stiffness factor k are constant in this case.

$$F = c \cdot \frac{dx}{dt} + k \cdot x \tag{2.14}$$

.

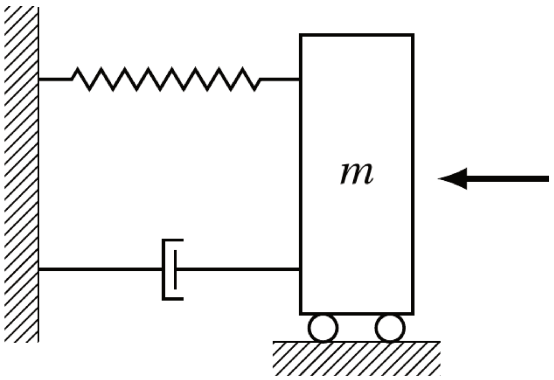


Figure 2.14: IPM platform [5]

2.4.3 Application of non-linear spring and damper

A possible solution to increase the force sensitivity and improve the distance estimation of the user an application of a non-linear spring and damper could be used. This will not be worked out further but an initial idea for this is given in appendix A

2.4.4 Selective haptic feedback

In a remote environment it could be possible that there are multiple obstacles in the surrounding area. The main question is whether the operator should receive force feedback when the obstacle is not in the path of motion. It could be possible to ignore these obstacles. Not ignoring these obstacles means that the haptic feedback feature will try to push you away from the obstacle while driving in a different direction. It is chosen to only give feedback on obstacles is in the direction of motion. Because obstacles which are not in the direction of motion are not a threat to the platform. The aim of this system is to prevent obstacle collisions by alerting the operator and therefore feedback is only necessary for obstacles in the direction of motion.

2.5 Conclusion

In this chapter the area of interest is defined and the used sensor is characterized. Based on this, a sensor configuration is chosen. An interface for the system has been proposed to keep track of all the sensor measurements while maintaining a proper refresh rate. After interfacing the sensors, methods have been proposed on how to process the distance measurements. Having processed the distance measurements, the control system is used to make sure that the information is available for haptic feedback. The haptic feedback component will not be implemented due to time constraints. Based on the analysis, a conceptual design towards the obstacle detection system is given. In Chapter 3 the actual design and implementation of the obstacle detection system is done.

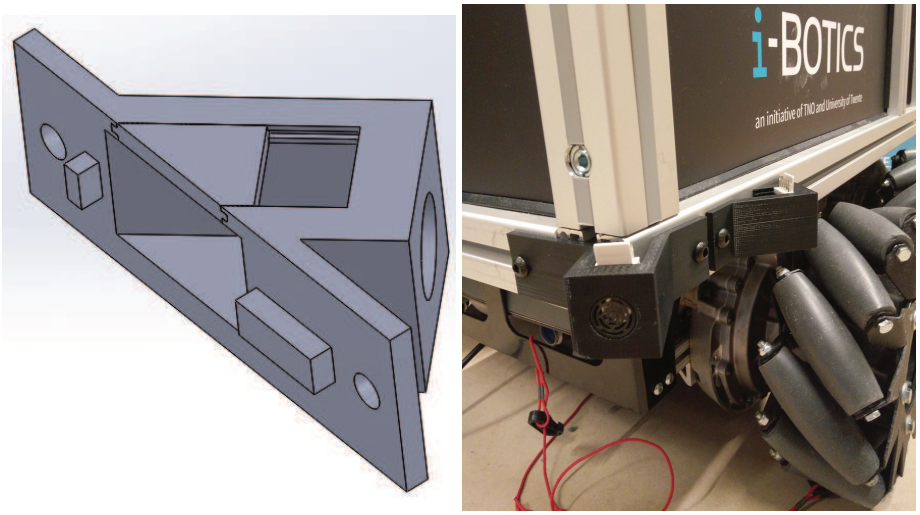
3 Design and implementation

In this section the design and implementation towards a functioning obstacle detection system is presented. First, the sensor system is implemented to pass on its distance measurements to the rest of the system. After that, an overview is given for the software implementation, which will further process the distance measurements. Each of the components in the software implementation will be discussed. After this chapter, the system should be able to deliver both the distance and velocity component for haptic feedback.

3.1 Sensor system

3.1.1 Sensor casings

To be able to properly mount the sensors with their respective angles on the platform some casings are designed. The angles of each sensor in the to be defined base frame are determined in Section 3.2.3. The design for the cases are done in Solidworks [3]. An example of such a drawing in solidworks is given in Figure 3.1a.



(a) Sensor casing drawing in SolidWorks

(b) Sensors mounted on platform

Figure 3.1: Sensor setup

The result of mounting these cases on the platform is shown in Figure 3.1b. With the sensors being properly positioned on the platform it is possible to perform the next step, which is to interface the sensor data with the PC. How this is done will be explained in the next subsection.

3.1.2 Arduino hardware

To be able to process the sensor data an interface is needed that connects the sensor system and the PC. The distance measurements done by the sensors are processed by an Arduino Mega. The Arduino Mega uses a microchip which provides 54 digital pins. So with 12 sensors the Arduino Mega is able to receive all the data with the used pulse width data type. Two more digital pins are needed to trigger the 2 sequential pairs. When the data is processed by the Arduino it will present the data via the ROS middleware to the PC. The PC further processes this data to eventually determine the distance to obstacles. How this is done will be discussed in the next section.

3.2 Calculation of distance to obstacle

In this section all the steps performed to translate the distance measurements from the ultrasonic sensors to the inputs for haptic feedback are discussed.

3.2.1 Software structure overview

Before going into detail about how the data processing is done an overview is given on how the system works. The structure of the system is shown in Figure 3.2. In this, each block represents a ROS node and the arrows represent the information exchange between certain ROS nodes. It can be seen that the Arduino will output its distance measurements. How this is done is explained in Section 3.2.2. These distance measurements are received by the component coordinate transformations. This component translates the distance measurements to a cartesian coordinates obstacle list in the to be defined base frame. This will be explained in Section 3.2.3. The coordinate transformations component sends its data to the obstacle processing component.

The obstacle processing component is the part at which all the input variables are calculated for haptic feedback and is further explained in Section 3.2.4. The obstacle processing component needs some more information such as the velocity of the platform. The velocity of the platform is received via the IMU sensor. The IMU sensor outputs an acceleration and therefore an integration component is implemented. The other part that the data processing component needs is the rotation of the platform which is directly provided by the IMU. With the above information the obstacle processing component outputs all the information needed to calculate the haptic feedback. This information contains the distance to an obstacle, the platform velocity and the sensor number that measured the obstacle. The haptic feedback component calculates the according force feedback and translates this to a input device specific force feedback. This is not implemented and is thus not explained any further.

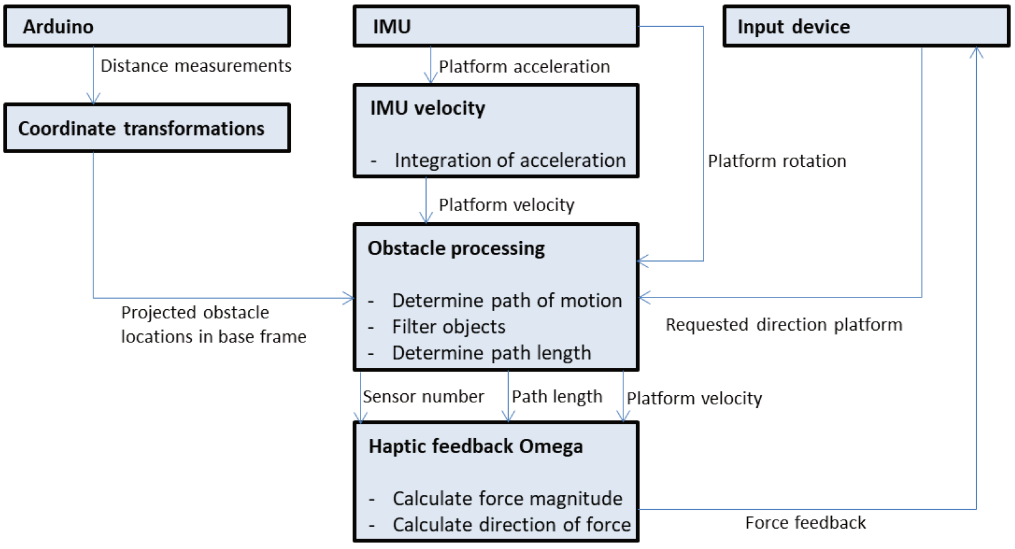


Figure 3.2: software structure overview

Now the software structure is known, a more in depth explanation of each of the components is given in the next sections.

3.2.2 Arduino software implementation

The first component mentioned in the previous section was the Arduino. In this, the data coming from the sensors is received, processed and eventually presented to the coordinate transformations component. The first step in the design for this is the measurement it self. As mentioned in the analysis, the sensor system will have 2 separate chains in which each chain has 6 sensors measuring synchronously. This is visualized in Figure 3.3, where 2 triggers individually trigger 6 sensors at the same time.

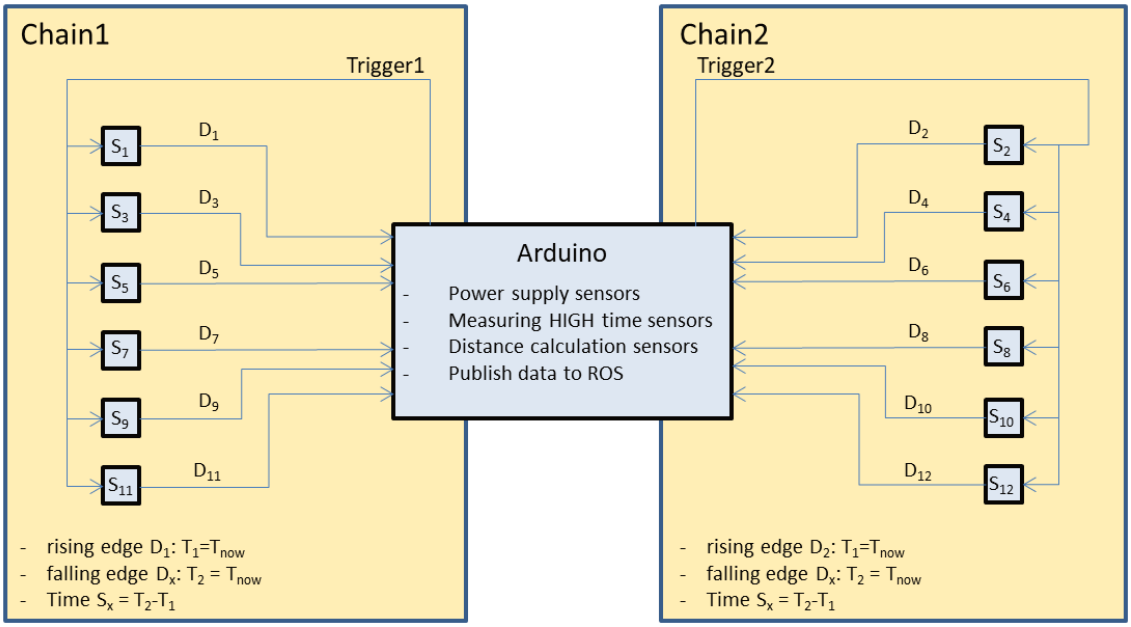


Figure 3.3: Arduino software structure

In this, it is important to mention that any neighbor sensor is in a different chain for interference prevention. Because 6 sensors are operating at the same time, a method needs to be implemented that makes sure that the sensors data pins can be read out real-time. To facilitate this timestamps are used. The Arduino $PulseIn()$ function is used to delay the system until the rising edge of the data pin of the first sensor in the chain. This marks the time when the sensor starts its measurement and thus the first time stamp is set here. It is assumed that the data pins are synchronized because of the connected RX pins, this has not been verified. The $PulseIn()$ function needs to measure a falling edge first before it will finish its measurement on the rising edge. The sensor data pins will only provide a falling edge after a measurement is done. Therefore, the refresh rate of the system is half of what it could be. A solution should be found to use timestamps without the $PulseIn()$ function.

Having set the first time stamp, a while loop will continuously check each data pin to see if it has been pulled low. When this occurs, the second time stamp for a specific sensor is set. Subtracting the second time stamp from the first time stamp gives the range reading. This time reading can be translated to a distance as the data sheet mentions that 147 micro seconds equals 2,54 cm. This will continue until all sensor data pins are pulled low. Afterwards the second sensor chain will be triggered in the same way and so on. This range reading list will be send to the coordinate transformations component via ROS which is discussed in the next section.

3.2.3 Coordinate transformations

Having received the range reading list from the sensor system it is now possible to project the obstacles in a base frame. This step is done because the sensors measure a distance with respect to their own location. Generalizing the list of obstacles to a representation in a uniform frame will make it simpler to process the obstacle list further on. This base frame is defined as shown in Figure 3.4. The reason that this base frame orientation is chosen, is because the IMU sensor also uses this reference frame. To keep the system simple the base frame will have the same orientation. To be able to present obstacles in the defined base frame, coordinate transformations have to be done for each obstacle measured. In the following calculations, the theory behind these coordinate transformations is explained.

First it must be mentioned that the needed coordinate transformations are a static transformation as the base frame and the sensor frames will be located on the same body, namely the platform. Because of this the needed coordinate transformations can be performed by using the transformation matrix H given in Equation 3.1.

$$H(x,y,\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & x \\ -sin(\theta) & cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

In Equation 3.1, θ represents the angle offset between the original frame and the new frame. A counter clockwise rotation means a positive θ. x and y represent the coordinate of the origin of the original frame expressed in the new frame. This transformation matrix can be used to describe a point in a base frame which is measured in a sensor frame. To transform the point p measured in a sensor frame (s) into a point p described in a base frame (b), Equation 3.2 holds:

$$^{b}P = H_{s}^{b}\,{}^{s}P \tag{3.2}$$

In Equation 3.2 ${}^{s}P$ describes the point p in a sensor frame, ${}^{b}P$ describes the point p in the base frame. When writing this out further this will result in Equation 3.3:

$$\begin{bmatrix} {}^{b}x \\ {}^{b}y \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) & x \\ -sin(\theta) & cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{s}x \\ {}^{s}y \\ 1 \end{bmatrix} \tag{3.3}$$

The frame of the sensors are defined in a way such that the y-axis of the sensor frames are in line with the sensors line of sight. This is visualized in Figure 3.4. The assumption was made that the measured obstacle is assigned the same angle as the sensor angle. Because of this the cartesian coordinates of the obstacle which is described in the sensor frame will always have an x coordinate equal to 0 and a y coordinate value equal to the distance d, measured by the sensor. Thus ${}^{s}P$ becomes as is shown in Equation 3.4:

$${}^{s}P = \begin{bmatrix} 0 \\ {}^{s}d \\ 1 \end{bmatrix} \tag{3.4}$$

Thus now the only thing left to do is to actually determine the angle offset θ from the sensor frame to the base frame and to define the origin of each sensor frame expressed in the base frame. As mentioned earlier all the frames are fixed with respect to the same body and thus the parameters x,y and θ are fixed as well. Figure 3.4 shows the assigned frame number for each sensor. The θ of interest for each sensor is the relative angle difference between a sensor

frame and the base frame given that counter clockwise rotation results in a positive angle. The defined angles for $n_{1..12}$ are shown in Equation 3.5:

$$\theta_n = \begin{bmatrix} -120 & -105 & -75 & -60 & -45 & 45 & 60 & 75 & 105 & 120 & 135 & -135 \end{bmatrix} \tag{3.5}$$

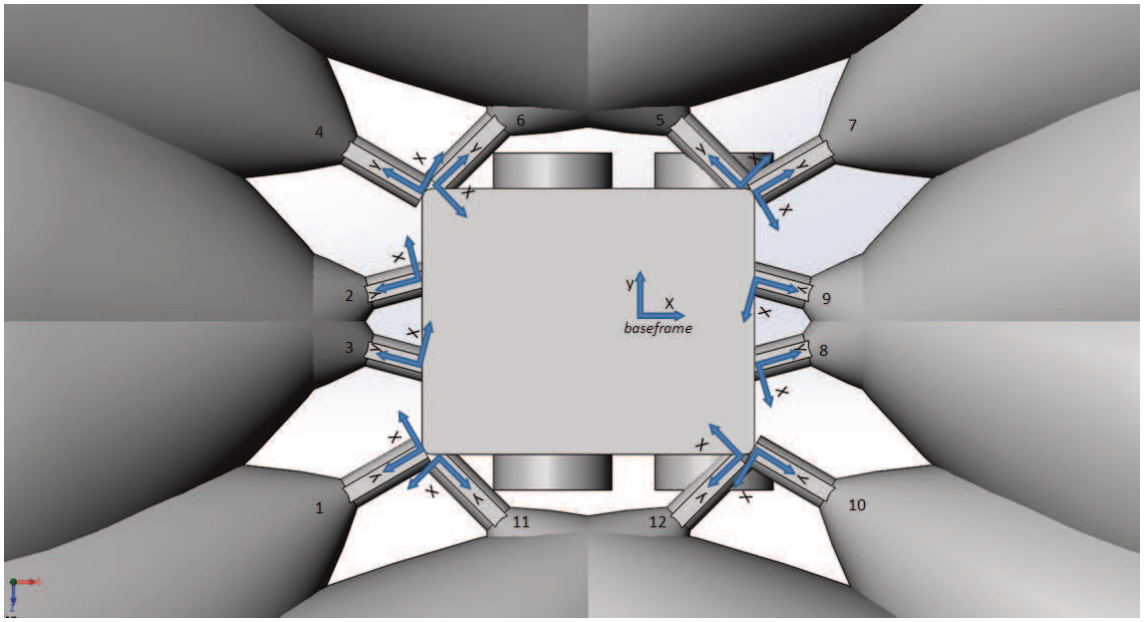


Figure 3.4: Orientation and numbering of the sensor frames

Figure 3.5 shows the position and distance offset for the sensors 1,3,8,10,11 and 12 from the center of rotation. On the other side the other 6 sensors are positioned and these are the mirrored versions of the shown ones. The base frame will have its origin in the center of rotation (same as the IMU) and thus these parameters can be used to express the origin of each sensor frame defined in the base frame. The Cartesian coordinates (x.y) in meters for $n_{1..12}$ are shown in Equation 3.6:

$$\begin{bmatrix} {}^b x_n \\ {}^b y_n \end{bmatrix} = \begin{bmatrix} -0.42 & -0.42 & -0.42 & -0.42 & 0.24 & -0.37 & 0.29 & 0.29 & 0.29 & 0.29 & -0.37 & 0.24 \\ -0.28 & 0.13 & -0.13 & 0.28 & 0.28 & 0.28 & 0.28 & -0.13 & 0.13 & -0.28 & -0.28 & -0.28 \end{bmatrix} \tag{3.6}$$

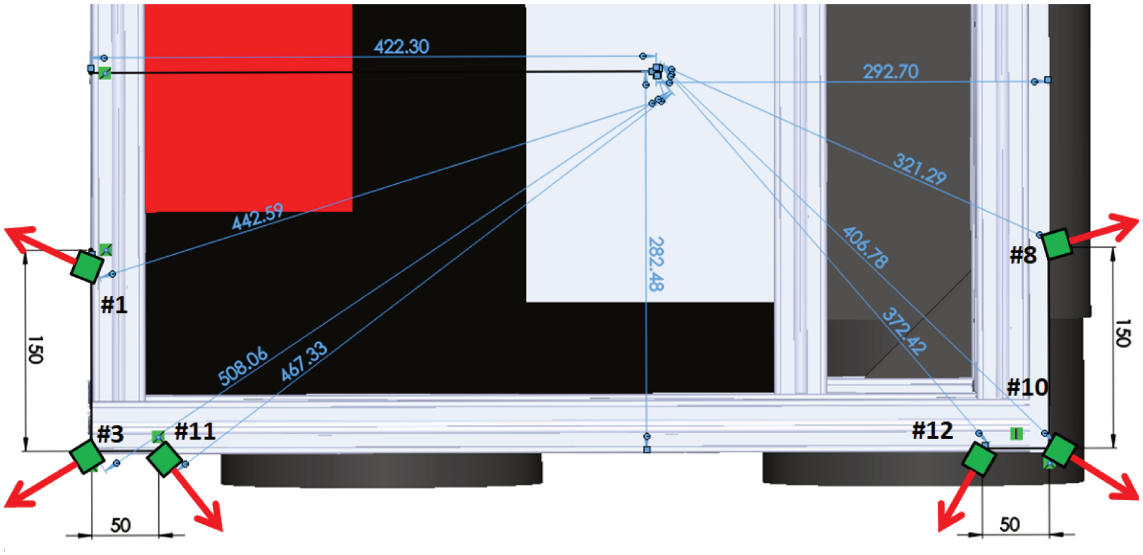


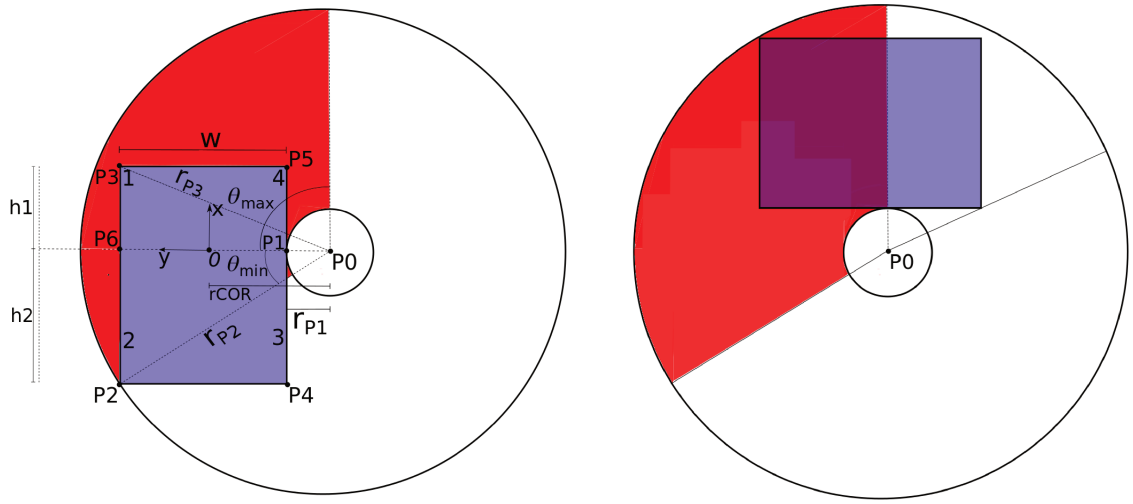
Figure 3.5: The position of sensors on the platform, dimensions in mm [8]

At this point all the needed parameters and variables are known to perform the coordinate transformations, this to present a cartesian coordinates obstacle list measured in sensor frames to a representation in the base frame. This obstacle list is send to the next component in the system, which is the obstacle processing. In the next section is discussed what steps are done in the obstacle processing.

3.2.4 Obstacle processing

Having received the obstacle list from the coordinate transformations component, the only missing information left is the velocity and rotation of the platform. The velocity and rotation of the platform is received from the IMU and can directly be used. The main goal of the obstacle processing component is to calculate the path length to the closest obstacle in the path of motion. The path length is defined as the length of an arc. Ideally, the arc has a radius based on the path of motion and ranges from the colliding point of the platform to the obstacle. To keep track of time the colliding point is defined with the same radius but located on the y-axis. The step to determine what point on the platform is the colliding point is not implemented and is left for future work. To be able to get to this result some steps and calculations need to be done first. These will now be explained.

The first step in this process is to derive a collision area. Because haptic feedback must only be given in the path of motion it is necessary to calculate the corresponding collision area. The collision area is the area in which the platform will move given the minimum travel time. It was mentioned in the analysis section that any movement of the platform can be seen as a movement over an arc with a certain radius. In case of a forward motion this results in a radius of infinity and a motion around its axis results in a radius of zero. Using this, the collision area will be defined with a radius and angle range. The angle range is used to represent the minimum travel time, defined in the analysis as 5 seconds. According to a minimum travel time, the platform is able to move a certain circumference part of the circular path. This can be visualized as the red area in Figure 3.6a. It can be seen in Figure 3.6b that the maximum angle boundary is calculated for the center of rotation line. The consequence is that the travel time will be less than 5 seconds. A future option would be to increase the maximum angle boundary to cover all parts of the platformas indicated in Figure 3.6b.



(a) Collision area visualization **(b)** Position of the platform at maximum angle

Figure 3.6: Circular motion of the platform in quadrant 4

Before continuing with finding the boundaries for the collision area, a quadrant system is defined. It will depend on the direction of motion of the platform what the conditions on the boundaries will be. In the example of Figure 3.6a there are 4 numbers 1,2,3 and 4. These quadrants are defined with respect to the platform, according to the velocity v_x and rotational velocity r_z as defined in Equation 3.7. A counter clock wise rotation around the z-axis is defined as positive rotation. As example, the platform in Figure 3.6a is moving in the 4th quadrant.

$$quadrantofmotion = \begin{cases} 1 & for\ v_x > 0\ and\ r_z > 0 \\ 2 & for\ v_x < 0\ and\ r_z < 0 \\ 3 & for\ v_x < 0\ and\ r_z > 0 \\ 4 & for\ v_x > 0\ and\ r_z < 0 \end{cases} \tag{3.7}$$

To be able to define the radius and angle boundaries of the collision area, Equation 2.8 can be used. It can be seen that Equation 2.8 is based on velocity v_x. The velocity v_x is received from the IMU, and is located at the center of rotation and also at the origin of the base frame. This calculated radius is thus defined as the length from the base frame origin *0* to the point P0. These points are shown in Figure 3.6a. This radius, which is named r_{cor}, gives the basis to calculate the radius and angle boundaries for the collision area.

Now based on r_{cor} the radius boundaries are defined. Figure 3.6 is taken as example. In here, the platform moves in quadrant 4. Point P1 and P2 are taken as outer points on the platform. The reason that P2 is taken is because the center of rotation is not in the middle of the platform. This causes swerving of the platform and thus the radius to P2 will be taken as maximum radius. The other point P1 is taken as this is the point that will have the minimum radius. So the radius for P1 will be the distance from point P0 to point P1. The radius for P2 will be the distance from point P0 to point P2. How to determine the radius for both the points is shown in Equations 3.8 and 3.9:

$$r_{P1} = r_{min} = r_{cor} - \frac{w}{2} \tag{3.8}$$

$$r_{P2} = r_{max} = tan^{-1}\left(\frac{h_2}{r_{cor} + \frac{w}{2}}\right) \tag{3.9}$$

In Equations 3.8 and 3.9 the following parameters are valued: h_1 = 0.293m , h_2 = 0.422m and w = 0.566m.

This calculation of the minimum and maximum radius also holds when the platform is moving in quadrant 1 and 2. P0 is in that case located on the left side of the platform. In that case P6 would be taken to calculate the minimum radius and P4 would be taken to calculate the maximum radius. The platform is symmetrical around the x-axis and therefore the length from the points P0 to P6 and P0 to P4 are equal to the lengths defined for motions in quadrant 3 and 4 .

Having determined the radius boundaries, the next step is to calculate the angle boundaries. With the angle boundaries the collision area is fully defined. The angle boundaries are formulated as shown in Equations 3.10 and 3.11

$$quadrant\ 1\ and\ 4 \begin{cases} \theta_{min} = 2\pi - sin^{-1}(\frac{h2}{r_{max}}) \\ \theta_{max} = \frac{|v_x| \cdot t}{r_{cor}} \end{cases} \tag{3.10}$$

$$quadrant\ 2\ and\ 3 \begin{cases} \theta_{min} = -2\pi + sin^{-1}(\frac{h1}{r_{p3}}) \\ \theta_{max} = -\frac{|v_x| \cdot t}{r_{cor}} \end{cases} \tag{3.11}$$

The only unknown variable r_{p3} (shown in Figure 3.6) is calculated according to Equation 3.12:

$$r_{p3} = \sqrt{(r_{cor} + \frac{w}{2})^2 + h_1^2} \tag{3.12}$$

As mentioned earlier moving in quadrant 1 and 4 (thus having a forward motion) will result in a positive angle. Moving in equadrant 2 and 3 (thus having a backwards motion) will result in a negative angle. This is also used for the boundaries. Figure 3.6a shows a visualization for θ_{min} and θ_{max} in case of a forward motion. It can be seen that θ_{min} in Equation 3.10 is always positive and θ_{max} in Equation 3.11 is always negative. This may seem strange at first but it has to do with path length calculations for obstacles in the collision area, as will be explained later on in this section.

Having defined the radius and angle boundaries, the collision area is defined. If the radius and angle to an obstacle is known a check can be done whether these lie within the collision area. Therefore the calculation of the radius and angle of an obstacle is the next step in the process.

For determining the radius and angle of an obstacle, again a quadrant system is used. Any position of an obstacle is known in cartesian coordinates with respect to the center of rotation of the platform, as explained in the coordinate transformations component. These cartesian coordinates are used to define the obstacle's quadrant. The numbering of the quadrants is the same as defined in the quadrant of motion system and is shown in Equation 3.13.

$$Obstacle\ quadrant = \begin{cases} 1 & for\ x > 0\ and\ y > 0 \\ 2 & for\ x < 0\ and\ y > 0 \\ 3 & for\ x < 0\ and\ y < 0 \\ 4 & for\ x > 0\ and\ y < 0 \end{cases} \tag{3.13}$$

With the information about the quadrant of motion and obstacle quadrant it is time to determine the radius and angle of an obstacle. A switch structure is used to do this. The switch

statement determines the quadrant of motion first and afterwards the quadrant of the obstacle. In this there are 48 different sets of equations. The calculations are variations of the equations for the radius and angle shown in Equation 3.14 and Equation 3.15.

$$Radius = \begin{cases} \sqrt{x^2 + (r_{cor} - |y|)^2} = a \\ \sqrt{x^2 + (|y| - r_{cor})^2} = b \\ \sqrt{x^2 + (|y| + r_{cor})^2} = c \end{cases} \tag{3.14}$$

$$Angle = \begin{cases} tan^{-1}(\frac{|x|}{r_{cor} - |y|}) = d \\ tan^{-1}(\frac{|x|}{|y| - r_{cor}}) = e \\ tan^{-1}(\frac{|x|}{|y| + r_{cor}}) = f \end{cases} \tag{3.15}$$

The variations for the radius are given in table 3.1 and for the angle in table 3.2.

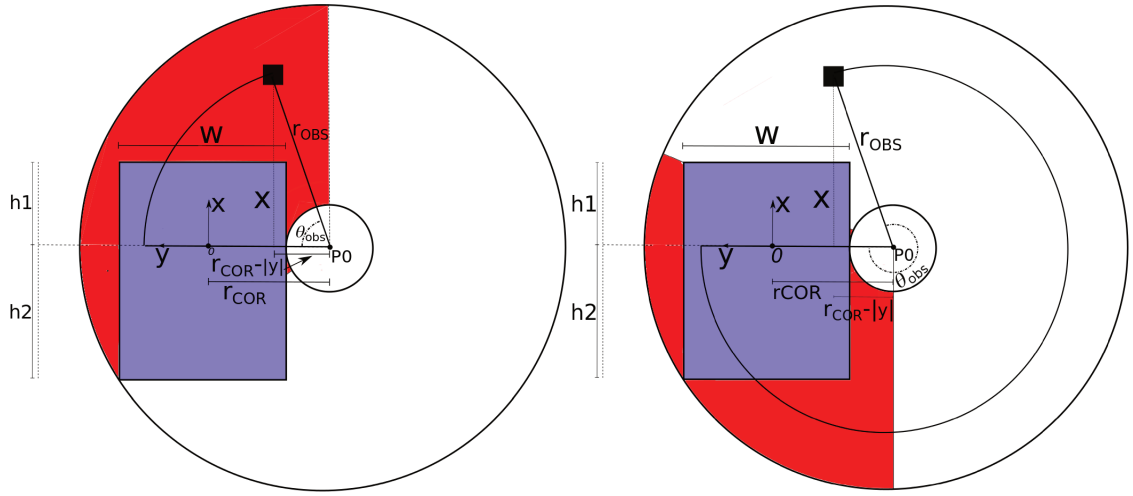
	rcor>|y|				rcor<|y|			
obstaclequadrant	1	2	3	4	1	2	3	4
quadrant of motion 1	a	a	c	c	b	b	c	c
quadrant of motion 2	a	a	c	c	b	b	c	c
quadrant of motion 3	c	c	a	a	c	c	b	b
quadrant of motion 4	c	c	a	a	c	c	b	b

Table 3.1: Radius calculations used in the quadrant system

	rcor>|y|				rcor <|y|			
obstaclequadrant	1	2	3	4	1	2	3	4
quadrant of motion 1	d	2π-d	2π-f	f	π-e	π+e	2π-f	f
quadrant of motion 2	-2π+d	-d	-f	-2π+f	-π-e	-π+e	-f	-2π+f
quadrant of motion 3	-2π+f	-f	-d	-2π+d	-2π+f	-f	-π+e	-π-e
quadrant of motion 4	f	2π-f	2π-d	d	f	2π-f	π+e	π-e

Table 3.2: Angle calculations used in the quadrant system

With the conditions defined for the switch statement an example is given. In the following example an obstacle is positioned in quadrant 4. The platform will either move in quadrant of motion 3 or 4. A visualization of this example is shown in Figure 3.7.



(a) Motion in quadrant 4, with $r_{cor} > |y|$ and an obstacle located in quadrant 4

(b) Motion in quadrant 3, with $r_{cor} > |y|$ and an obstacle located in quadrant 4

Figure 3.7: Path length representation for different quadrants of motion

In here the black square represents an obstacle and the radius r_{obs} defines the radius of the obstacle. In both Figure 3.7a and 3.7b the radius is defined as shown in Equation 3.16, with x and y equal the obstacle's coordinate.

$$r_{obs} = \sqrt{x^2 + (r_{cor} - |y|)^2} \tag{3.16}$$

Figure 3.7a and 3.7b show a different definition of θ_{obs}. This is caused by the direction of collision of the platform with the obstacle. The calculation of the angle for the specific example of Figure 3.7 is shown in Equation 3.17

$$\theta_{obs} = \begin{cases} tan^{-1}(\frac{|x|}{r_{cor}-|y|}) \; in \; Figure \; 3.7a \\ -2\pi + tan^{-1}(\frac{|x|}{r_{cor}-|y|}) \; in \; Figure \; 3.7b \end{cases} \tag{3.17}$$

It can be seen in Equation 3.17 that in case of a forward motion the angle is always positive and in case of a backwards motion always negative. This is forced by taking the absolute values of x and y and holds for every case in the switch statement. This is done to check whether the platform will have a collision driving forwards or backwards and that may influence the path length to the obstacle.

There is a specific case when the angle calculation will go wrong, this is when an obstacle is in the swerving area. For example when an obstacle is positioned in the swerving part of the collision area of quadrant 2 in Figure 3.7a. The system will determine that the angle of this obstacle is $2\pi - \theta_{obs}$ but in reality this will be θ_{obs}. A solution has not been implemented yet to fix this problem but should be fixed in future work.

Now having determined the radius and angle for an obstacle it is possible to see if it is in the collision area or not. The specific boundaries for $r_{min}, r_{max}, \theta_{min}$ *and* θ_{max} were determined in Equations 3.8, 3.9, 3.10 and 3.11. The algorithm that determines whether the obstacles are in the collision area is shown in algorithm 1.

Based on algorithm 1 each obstacle that is in the collision area will be stored and each obstacle which is not in the collision area is left out. The path length to every obstacle in the collision

Algorithm 1 Check for obstacle in collision area

```
if quadrant is 1 or 4 then
    if r_obs > r_min AND r_obs < r_max then
        if θ_obs > θ_min AND θ_obs < θ_max then
            Obstacle in collision area
        end if
    end if
end if
if quadrant is 2 or 3 then
    if r_obs > r_min AND r_obs < r_max then
        if θ_obs < θ_min AND θ_obs > θ_max then
            Obstacle in collision area
        end if
    end if
end if
```

area will be calculated next. Due to the definitions of the radius and angle for the obstacles Equation 3.18 can be used:

$$l = |\theta_{obs}| \cdot r_{obs} \tag{3.18}$$

With this, it is known what the path length to each obstacle is in the collision area. To keep track of time it is decided to only pass on information about the obstacle with the shortest path length to the haptic feedback component (which will not be implemented). Once the obstacle with the shortest path length is determined the obstacle processing component will output information about the path length and velocity of the platform. These are the parameters needed to calculate the haptic feedback components.

4 Testing and Results

At this point the obstacle detection system has been designed and implemented. In this section experiments are executed to test the system. The experiments in section 4.1, 4.2, 4.3, and 4.4 determine if the structure of the system works as intended. In these experiments it is determined respectively, if the quadrant system works, if the collision area is properly calculated, if filtering of obstacles is done properly and if the closest obstacle is chosen for haptic feedback. If all of these results comply with the theory, the obstacle processing is done as intended.

After this, the experiments performed in section 4.5, 4.6 and 4.7 should present the known limitations of the system defined in Chapter 3. As there are multiple assumptions made, false calculations occur. These false calculations are known and these experiments will present them. The experiments respectively determine the severity of the path length error due to the path starting point, show the false angle calculation for an obstacle in the swerving area and the path length error introduced due to the assumed angle of an obstacle.

Now each experiment will be performed, but first, an overview of all the experiments executed and their goals is given in table 4.1:

Experiment	Description	Goals
1	Validate the working of the quadrant system for the collision area	-Determine if the quadrants are determined properly - Show the similarities in boundary calculations between quadrants - Give a basis for upcoming experiments such that if an experiment is done in a quadrant, it holds for every quadrant
2	Validate the working of the collision area	- Determine if the collision area boundaries are calculated properly - Find the limits of the system in terms of collision area boundaries - Give a basis for upcoming experiments such that the collision area is always properly calculated
3	Test the filtering of obstacles near the collision area	- Determine whether the radius and angle for an obstacle are calculated properly - Ensure that filtering of obstsacles outside the collision area is done properly
4	Determine if the closest obstacle is chosen for haptic feedback	- Determine whether the path length to an obstacle is calculated properly - Ensure that the closest obstacle is chosen for haptic feedback
5	Determining the severity of the path length error due to the path starting point	- Determine the severity of the path length error due to the starting point of the path calculation
6	Show the false angle calculation for an obstacle in the swerving area	- Show the possibility of false angle calculations for an obstacle in the swerving area
7	Determine the severity of the introduced path length error due to the assumed angle of an obstacle	- Determine the severity of the possible path length error due the assumed angle of an obstacle

Table 4.1: Overview of the experiments executed and their goals

4.1 Experiment 1: Validate the quadrant system working for the collision area

This experiment has two main goals. The first goal is to determine whether the quadrants are properly determined using the path of motion and to show that there are similarities in the boundary calculations between quadrants. The second goal is to give a basis for the upcoming experiments, such that if an experiment is done in a quadrant of motion, it holds for all quadrants of motion.

The steps executed in this experiments are as follows. First a path of motion in each of the 4 quadrants of motion (with equal radius) is defined. Then the radius and angle range for each of the 4 quadrants of motion is calculated. Finally, the theoretical calculations are compared to the system calculations.

The hypothesis is, that the radius boundaries for each quadrant of motion should give the same results. Also, the angle boundaries for the quadrants of motion 1, 4 and quadrants of motion 2, 3 should give the same results.

The magnitude of the values chosen for v_x is 0.4 m/s and ω_z is 0.4 rad/s. The theoretical values for the radius and angle boundaries are shown in table 4.2.

$Direction\,of\,motion$	$Quadrant$	$r_{min}\,[m]$	$r_{max}\,[m]$	$\theta_{min}\,[rad]$	$\theta_{max}\,[rad]$
$v_x = \ w_z = 0.4$	1	0.1260	0.8105	5.7356	4.8904
$-v_x = -w_z = 0.4$	2	0.1260	0.8105	-5.8826	-4.8904
$-v_x = \ w_z = 0.4$	3	0.1260	0.8105	-5.8826	-4.8904
$v_x = -w_z = 0.4$	4	0.1260	0.8105	5.7356	4.8904

Table 4.2: Theoretical calculations for quadrants of motion

The system calculated values for the radius and angle boundaries are shown in table 4.3

$Direction\,of\,motion$	$Quadrant$	$r_{min}\,[m]$	$r_{max}\,[m]$	$\theta_{min}\,[rad]$	$\theta_{max}\,[rad]$
$v_x = \ w_z = 0.4$	1	0.1265	0.8102	5.7349	4.8904
$-v_x = -w_z = 0.4$	2	0.1265	0.8102	-5.8827	-4.8904
$-v_x = \ w_z = 0.4$	3	0.1265	0.8102	-5.8827	-4.8904
$v_x = -w_z = 0.4$	4	0.1265	0.8102	5.7349	4.8904

Table 4.3: System calculations for quadrants of motion

It can be seen that the theoretical- and system calculations are close to identical and comply with the expected result. Thus, determining the quadrant of motion is done properly for al quadrants. Based on experiment 1, the assumption is made that the results for a quadrant of motion holds for all quadrants of motion. The following experiments will use this assumption.

4.2 Experiment 2: Validate the working of the boundary conditions of the collision area

This experiment has multiple goals. The main goal is to verify that the calculation of the collision area boundaries are done according to the theory. Another goal is to show that the angle boundaries are limited to a full circle in case of small radii and that large radii may introduce difficulties in angle boundary calculations. This experiment aims thus on finding the limits of the system in terms of boundary conditions for the collision area. The final goal is to give a basis for the following experiments, such that it can be assumed that the collision area is properly determined in each quadrant of motion.

The steps executed in this experiment are as follows. First some radii of interest for the path of motion are defined. The collision areas for the radii of interest are calculated next. finally, these results will be compared to the system calculated collision area.

Interesting path of motions are: rotation only ($r = 0.01m$), motion with radius equal to half the width of the platform ($r = 0.3m$), random movements ($r = 1,10m$) and translation only ($r = 100m$) Thus, the defined radii of interest are: $r_{cor} = [0.01, 0.3, 1, 10, 100]m$. In table 4.4 the theoretical calculations for the collision area are shown.

$r_{cor}\ [m]$	$r_{min}\ [m]$	$r_{max}\ [m]$	$\theta_{min_{1,4}}\ [rad]$	$\theta_{max_{1,4}}\ [rad]$	$\theta_{min_{2,3}}\ [rad]$	$\theta_{max_{2,3}}\ [rad]$
0.01	0	0.514	5.319	9.230 (5.309*)	-5.498	-9.233 (-5.488*)
0.3	0.017	0.720	5.657	5.611	-5.817	-5.611
1	0.717	1.350	5.965	2.882	-6.059	-2.883
10	9.717	10.291	6.242	0.398	-6.255	-0.398
100	99.717	100.284	6.279	0.041	-6.280	-0.041

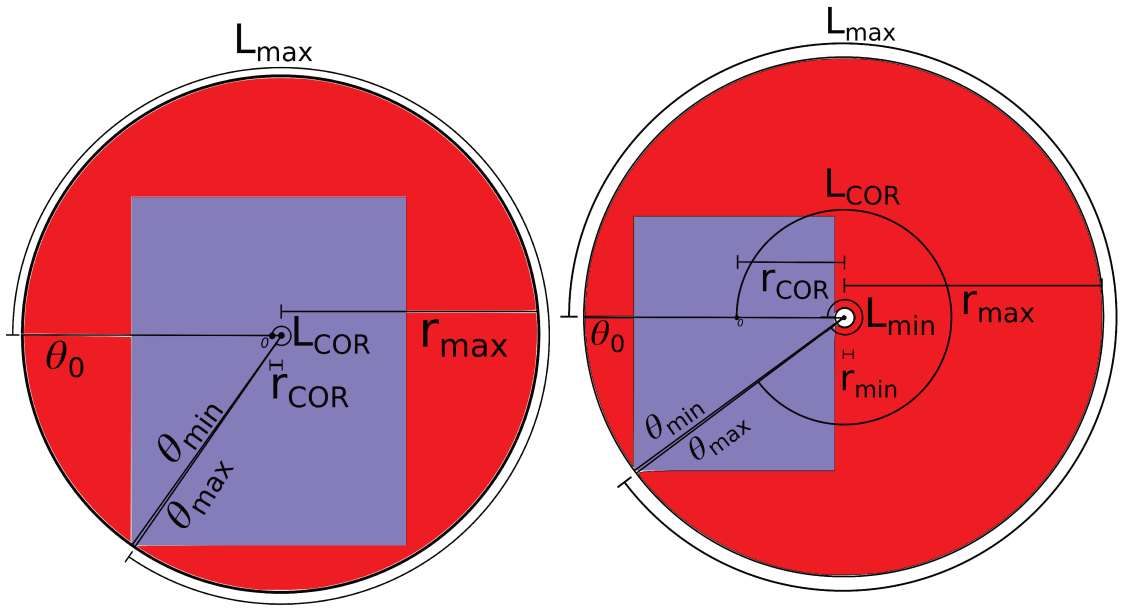
Table 4.4: Theoretical collision area
*System should limit these values to ensure proper boundaries

The system calculations for the collision area are shown in table 4.5.

$r_{cor}\ [m]$	$r_{min}\ [m]$	$r_{max}\ [m]$	$\theta_{min_{1,4}}\ [rad]$	$\theta_{max_{1,4}}\ [rad]$	$\theta_{min_{2,3}}\ [rad]$	$\theta_{max_{2,3}}\ [rad]$
0.01	0	0.514	5.318	5.308	-5.497	-5.487
0.3	0.017	0.719	5.656	5.611	-5.818	-5.611
1	0.718	1.350	5.965	2.883	-6.059	-2.882
10	9.718	10.291	6.242	0.398	-6.255	-0.398
100	99.718	100.283	6.279	0.041	-6.280	-0.041

Table 4.5: System collision area

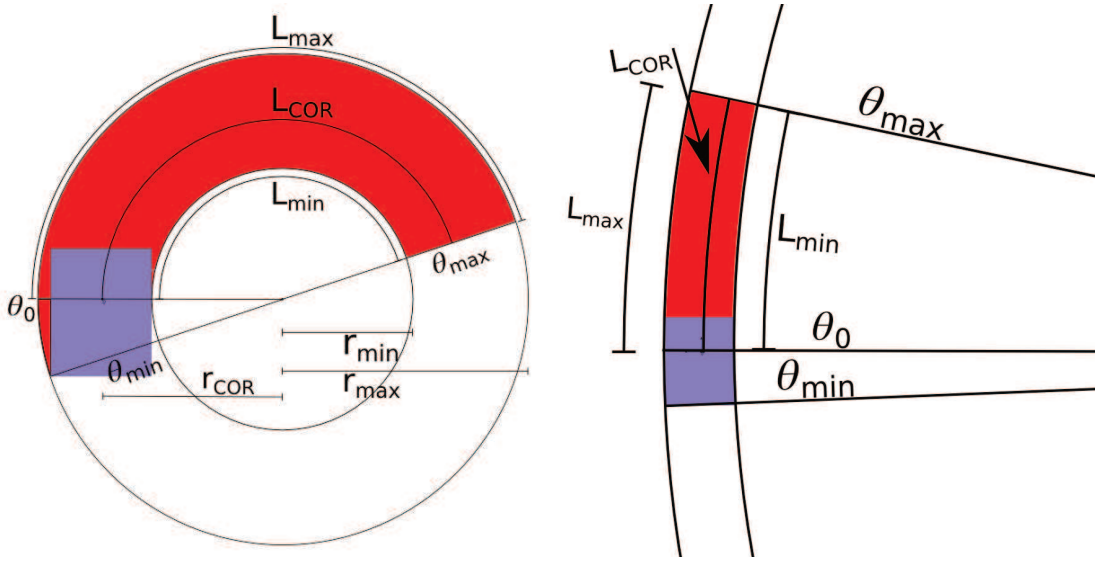
A visualization for the collision area for the radii of interest of $r_{cor} = [0.01, 0.3, 1, 10]m$ are given respectively in Figures 4.1a, 4.1b, 4.2a and 4.2b.



(a) $R_{cor} = 0.01m, L_{cor} = 0.05m, L_{max} = 2.73m$

(b) $R_{cor} = 0.3m, L_{min} = 0.08m,$ $L_{cor} = 1.68m, L_{max} = 4.04m$

Figure 4.1: Visualization of the collision area for given path of motion



(a) $R_{cor} = 1m, L_{min} = 2.07m, L_{cor} = 2.88m$, $L_{max} = 3.89m$

(b) $R_{cor} = 10m, L_{min} = 3.86m, L_{cor} = 3.98m$, $L_{max} = 4.09m$

Figure 4.2: Visualization of the collision area for given path of motion

In general, it can be stated that the system calculates the collision area as expected. The boundaries are limited to a full circle as is shown in the calculation for the radius of interest equal to 0.01. Also, for a large radius of motion the angle range become quite small. In this case, the resolution of the system becomes vital. Based on this experiment it is assumed that the collision area is properly calculated, this will be used in the following experiments.

4.3 Experiment 3: Test the filtering of obstacles near the collision area

The main goal of this experiment is to make sure that the radius and angle calculation for an obstacle is done properly. The other goal is to ensure that filtering of obstacles outside the collision area is done.

The steps executed in this experiment are as follows. First a radius of motion is defined and the collision area is calculated. The corner points of the collision area are then determined in polar coordinates and translated to cartesian coordinates. Obstacles will be positioned around these corner points and their cartesian coordinates are calculated. Translating the obstacles cartesian coordinates back to polar coordinates will show which obstacles lie within the boundaries of the collision area. Finally, a comparison is made between theory and practice.

An example radius of motion is chosen as $r_{cor} = 2m$. An approximation visualization of the collision area is shown in Figure 4.3.

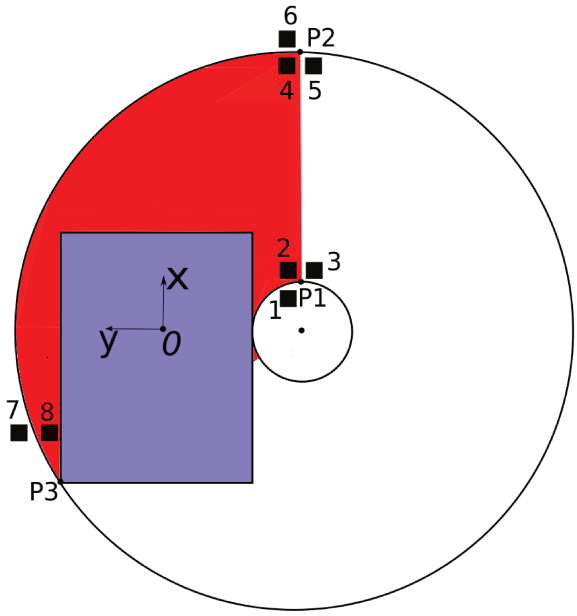


Figure 4.3: Defined obstacles near collision area boundaries

The radius and angle boundaries for the collision area are shown in table 4.6

r_{min} $[m]$	r_{max} $[m]$	θ_{min} $[rad]$	θ_{max} $[rad]$
1.718	2.321	6.100	1.701

Table 4.6: Collision area parameters

Points P1, P2 and P3 are defined as the corner points for the collision area. Table 4.6 is used to define their polar coordinates, these are: $P1 = (r_{min}, \theta_{max})$, $P2 = (r_{max}, \theta_{max})$ and $P1 = (r_{max}, \theta_{min})$. These polar coordinates are translated to cartesian coordinates and are shown in table 4.7.

Point	x-coordinate $[m]$	y-coordinate $[m]$
P1	1.702	-2.220
P2	2.302	-2.300
P3	-0.422	0.282

Table 4.7: Coordinates of corner points P1, P2 and P3

Based on the cartesian coordinates of point P1, P2 and P3, obstacles are defined around these points. The defined obstacles are shown in Figure 4.3 and it can be seen that obstacle 2,4 and 8 lie inside the collision area and the other obstacles not. The cartesian coordinates of all the obstacles are shown in table 4.8.

Obstacle	x-coordinate $[m]$	y-coordinate $[m]$
1	1.68	-2.20
2	1.72	-2.20
3	1.72	-2.24
4	2.28	-2.28
5	2.28	-2.32
6	2.32	-2.28
7	-0.44	0.30
8	-0.40	0.26

Table 4.8: Cartesian coordinates of obstacles defined near the points P1, P2 and P3

The cartesian coordinates of the obstacles are translated back to polar coordinates and are shown in table 4.9. Based on table 4.9 it can be seen that obstacle 2,4 and 8 lie in the colli-

obstacle	Radius [m]	Angle [rad]
1	1.692	1.689
2	1.731	1.687
3	1.737	1.709
4	2.297	1.692
5	2.302	1.710
6	2.337	1.690
7	2.341	6.094
8	2.295	6.108

Table 4.9: Polar coordinates of obstacles

sion area as their radius and angle is within the boundaries defined in table 4.6. The other obstacles lie outside the collision area. This corresponds with the theory.

What has not been tested is whether the resolution of the system for large radii of motion suffices. It might be possible that for large radii of motion an obstacle is falsely positioned inside or outside the collision area due to the resolution of the system. Because for large radii of motion the radius calculation for both the collision area and obstacle approaches infinity and the corresponding angle calculations approach 0. This should be examined in future work.

4.4 Experiment 4: Determine if the closest obstacle is chosen for haptic feedback

The main goal of this experiment is to determine whether the path length calculation is done properly. The other goal is to see if the closest obstacle is taken for haptic feedback.

The steps executed in this experiments are as follows. First a radius of motion is defined. Then 3 obstacles with different radii in the collision area will be defined. The path length to each of these obstacles is calculated next. Finally, the system should determine which obstacle is the closest one.

The same radius of motion of experiment 4.3 is chosen and is defined as $r = 2m$. The corresponding collision area boundaries are given in table 4.6. The obstacles that are defined will all have an angle equal to, $\theta_{max} - 0.01$, and radii as shown in table 4.10a.

$Obstacle$	$Radius\,[m]$	$path\,length\,[m]$
1	$r_{min} + 0.01$	2.918
2	$\frac{r_{min}+r_{max}}{2}$	3.415
3	$r_{max} - 0.01$	3.909

(a) Theoretical path length calculations

$Obstacle$	$Radius\,[m]$	$path\,length\,[m]$
1	$r_{min} + 0.01$	2.921
2	$\frac{r_{min}+r_{max}}{2}$	3.415
3	$r_{max} - 0.01$	3.908

(b) System calculated path length calculations

Table 4.10: Path length calculations

A visualization of the positioning of each obstacle is shown in Figure 4.4 and it can be seen that obstacle 1 is the closest obstacle to the platform.

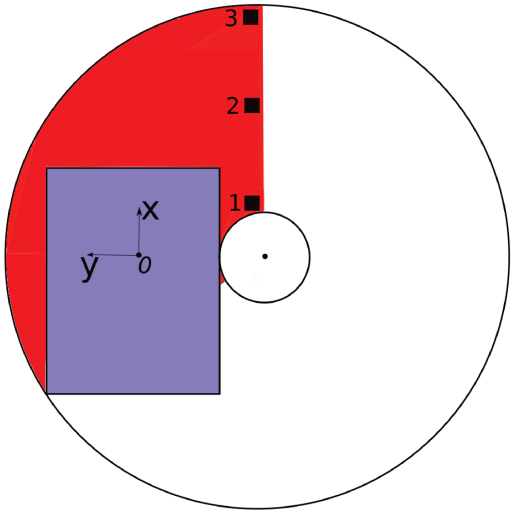


Figure 4.4: Defined obstacles for determining the closest obstacle

The path lengths calculated by the system are shown in Figure 4.10b and obstacle 1 is chosen as closest obstacle. The theoretical path length calculations match the system calculations and the proper obstacle is taken as closest obstacle.

4.5 Experiment 5: Determine the severity of the path length error due to the path starting point

The goal of this experiment is to determine the severity of the path length error due to the starting point of the path calculation.

To keep track of time, a simplification was made. In this it is defined that the path length to an obstacle, is the arc from the center of rotation line, to the obstacle. In practice, the path length will be the arc from the colliding point of the platform to the obstacle. This experiment will show what the consequence is of this simplification.

The steps executed in this experiment are as follows. First a radius of motion is defined. The next step is to define the cartesian coordinates of obstacles specifically near the edges of the platform. The path length to each obstacle is determined and it is assumed that these path lengths can be seen as the path length error.

The radius of motion is defined as $r = 0.3m$. The obstacles' cartesian coordinates are given in table 4.11 and a visualization is shown in Figure 4.5.

Obstacle	x-coordinate [m]	y-coordinate [m]
1	0.3	0.28
2	-0.43	0.27
3	0.28	-0.29
4	-0.402	0.29
5	-0.43	-0.27

Table 4.11: Cartesian coordinates of the obstacles near the edges of the platform

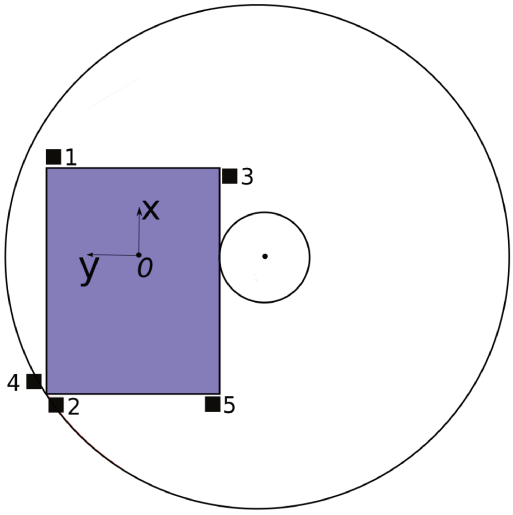


Figure 4.5: Defined obstacles to determine the maximum path length error

The cartesian coordinates are translated to polar coordinates and the according path lengths are calculated. The theoretical path lengths are shown in table 4.12a

$Obstacle$	$path\ length\ error\ [m]$
1	0.312
2	0.462
3	0.427
4	0.430
5	0.647

(a) Theoretical path length error

$Obstacle$	$path\ length\ error\ [m]$
1	0.311
2	0.461
3	0.427
4	0.430
5	0.647

(b) System calculated path length error

Table 4.12: Path length errors

The system calculated path lengths are shown in table 4.12b. It can be seen that the path lengths vary from 0.311m to 0.647m. As the obstacles are defined near the edges of the platform it is assumed that these are the path length errors to these points on the platform.

4.6 Experiment 6: Show the false angle calculation for an obstacle in the swerving area

The goal of this experiment is to show the possibility of false angle calculations for an obstacle in the swerving area.

The steps executed for this experiment are as follows. A radius of motion will be defined and an obstacle will be positioned in the swerving area. After that, the radius, angle and path length will be calculated, a conclusion can be drawn afterwards.

The radius of motion and obstacle 8 of experiment 4.3 are used. This obstacle lies in the swerving area of the collision area. The system calculated radius is $2.296m$ and the angle is $6.1715rad$. The corresponding path length is $14.019m$. The visualization for the path length calculation is shown in Figure 4.6.

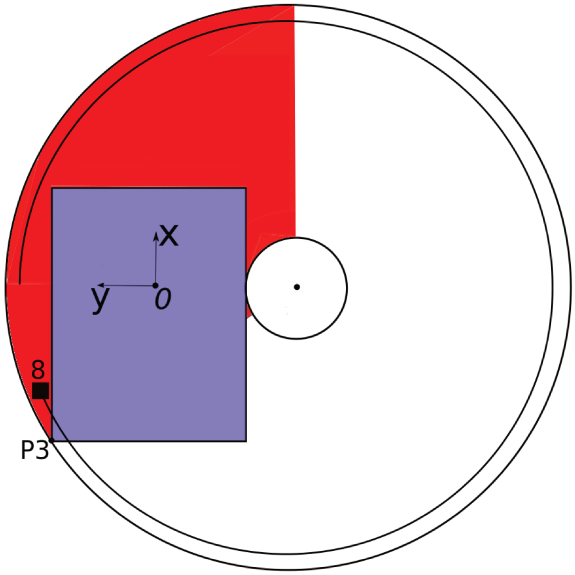


Figure 4.6: Path length representation for an obstacle in the swerving area

It can be seen that the path length calculation is not correct. This is caused by the fact that the system assumes that the front of platform will collide with the obstacle.

4.7 Experiment 7: Determine the severity of the introduced path length error due to the assumed angle of an obstacle

The assumption was made that the angle to an obstacle is equal to the angle of the sensor in the base frame. This assumption may introduce path length errors. Therefore, the goal of this experiment is to determine the severity of this possible path length error.

The steps executed in this experiment are as follows. First an obstacle is defined in the widest line of beam pattern C with an angle equal to the to be chosen sensor. This obstacle represents a projected obstacle. Another obstacle is positioned at the side in the widest line of beam pattern C. This obstacle represents an actual obstacle. This example is seen as a worst case scenario as the error in angle is maximum. Based on the position of both obstacles, a radius of motion will be chosen and for this, both obstacles must be in the collision area. The path length to each obstacle will be calculated and a conclusion can be drawn on the path length error.

The widest line in beam pattern C is 2,4m and is located at a distance of 2,1m from the sensor. It is chosen to conduct this experiment for a front and a back sensor. A visualization for the setup of this experiment is shown in Figure 4.7 and the cartesian coordinates are shown in table 4.13.

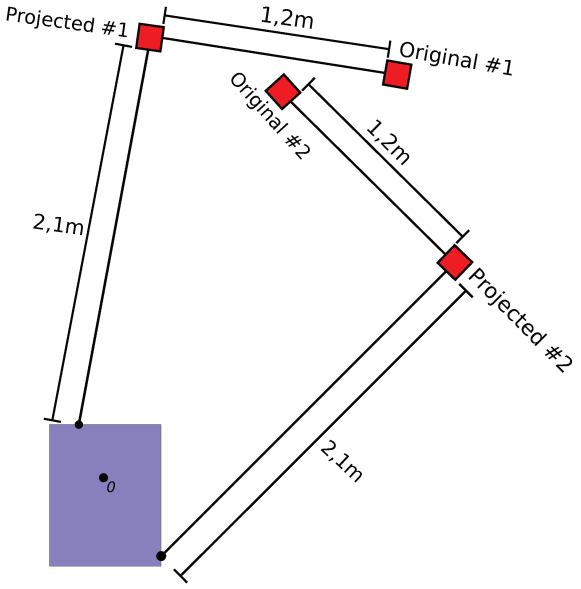


Figure 4.7: Visualization of the possible positioning error due to assumed angle

obstacle number	x-coordinate	y-coordinate
Projected 1	2.32	-0.3
Original 1	2	-1.57
Projected 2	1.11	-1.77
Original 2	1.95	-0.92

Table 4.13: Cartesian coordinates obstacles

The next step would be to define a radius of motion at which both obstacles lie in the collision area. It appeared that there was no radius of motion that could contain both the projected and the original obstacle. This is caused because the obstacles are located too far apart. There is thus a possibility that the system projects an obstacle at a location outside its collision area while in reality it is inside the collision area.

4.8 Discussion

In this section the results of the experiments are discussed.

4.8.1 Collision area calculation

The experiments in Section 4.1, 4.2 and 4.3 were executed to determine whether the collision area and filtering of obstacles is done properly. In this, it appeared that the system worked as predicted, but that includes limitations. Difficulties were found for large radii of motion as the angle boundaries become small. It has not been tested at what point the resolution of the system becomes an issue.

4.8.2 Path length calculations

Multiple experiments have been executed to show all the possible path length errors due to the methods used. Each of these methods will now be discussed.

The experiment in Section 4.5 mentions about the path length error introduced due to the starting point of the path. The starting point is located in the center of rotation line and this introduced path length errors varying from 0.312m to 0.647m. The proper path length can be

found by taking the starting point of the path at the point on the platform that will collide with the obstacle. The polar coordinates of this point must be found in order to determine the path length to an obstacle.

The path length error for obstacles in the swerving area was analyzed in the experiment of Section 4.6. The experiment showed that for the given obstacle, the path length was equal to 14.018m. That was caused by the assumption that the platform would collide with the obstacle in a forward motion. A first step would be to define the angle as $2\pi - \theta_{obs}$ instead of θ_{obs} (forward motion) and $-2\pi + |\theta_{obs}|$ instead of θ_{obs} (backwards motion) for obstacles in the swerving area. The boundaries of the collision area can be used to implement this.

The experiment in Section 4.7 was executed to determine the path length error due to the assumed angle of an obstacle. A scenario was chosen in which the largest angle error occurs. It appeared that no radius of motion could be found such that both the projected and actual obstacle were in the collision area. There is a possibility that the projected obstacle lies outside the collision area while the actual obstacle lies inside the collision area. This could cause a collision between the platform and obstacle. No additional tests have been done to find the largest path length error when both obstacles lie in the collision area. This would give an idea on what the path length error could be.

5 Conclusion and Recommendations

The conclusions of this research are described in this chapter by reviewing the research questions defined in Section 1.4. Based on this, recommendations for future work are given.

5.1 Conclusion

The focus in this research is summarized by the main research question:

How to design a properly functioning 360° obstacle detection system with an intuitive feeling haptic feedback implementation?

The first research question, that followed from the main research question, defined the area of interest and gave the requirements for the sensor system:

What area around the platform is of interest when trying to detect obstacles?

For this, the requirement of a travel time of 5 seconds was set. Due to the omni-directional movement of the platform it appeared that the area of interest is ellipse shaped as was shown in Figure 2.5. A sensor configuration had to be found that could properly scan this area of interest, which led to the following research question:

How many sensors with what individual position and orientation are needed to be able to properly scan this area of interest?

The sensor configuration shown in Figure 2.9b was chosen. The area coverage in the area of interest for this configuration was optimal compared to others. It appeared that it was not possible to guarantee full coverage of the area of interest. Therefore, a travel time of 5 seconds is not ensured. The consequence is, that the user might have less response time for smaller obstacles, but the severity of this is unknown.

Using this sensor configuration, sensors will have overlap in their scanning area. This was done to prevent dead zones and consequently prevent collisions. There is a trade off between preventing dead zones and interference. Both of these could influence the performance of the system and therefore the following research question was derived:

How are interference and dead zones influencing the performance of the sensor setup? How can this be tested?

A sensor chaining method is implemented, in which 2 chains of 6 sensors are working sequentially. This ensures that neighbor sensors are not measuring at the same time. This could prevent interference. Experiments to test the performance of the sensor system in terms of dead zones and interference have not been performed. In future work experiments should be done to test whether dead zones and interference are problematic.

An Arduino Mega interfaces the sensor data and is responsible for the chaining of sensors. In this, the update frequency could be improved. The frequency is half the frequency of what is desired. This is caused by the PulseIn() function used for setting timestamps. Also, based on the interference experiment results, a decision could be made whether sequential measuring is actually necessary.

Having received obstacle data from the sensor system, processing the obstacle data was necessary to translate it to haptic feedback. First, it had to be determined on which obstacles haptic feedback should be given. This led to the following research question:

On which obstacles does the system need to give haptic feedback?

Because the main goal of this system is to prevent obstacle collisions it was determined that haptic feedback is only given on obstacles in the path of motion of the platform. Other obstacles are not a threat to the platform and can be ignored. Knowing this, it was possible to determine what data processing was required. This led to the following research question:

What sensor data processing must be done such that it can be used to generate haptic feedback based on the location of one or more obstacles?

Methods have been implemented to translate sensor data measurement to haptic feedback. These are: calculating the collision area, positioning of obstacles, filtering of obstacles and calculating the path length to an obstacle.

It appeared that the collision area was properly calculated but for large radii of motion the resolution of this system becomes significant. Errors in filtering of obstacles might occur. It has not been tested if the resolution becomes a problem and at what point. This should be done in future work.

The next method is to position obstacles in the surrounding area. For this, an assumption was made that the angle of an obstacle is equal to the sensor angle. It appeared that some obstacles are positioned outside the collision area, while in reality, they are in the collision area. This may cause collisions between platform and obstacles. A more accurate method to determine an obstacle angle should be implemented.

Knowing the collision area and position of obstacles, a method is implemented that determines whether the obstacle is inside the collision area. This appeared to work properly.

The final method implemented was the calculation of the path length. Simplifications were done for time's sake but a proper basis is given. Multiple experiments showed that there are path length errors. Causes for this are: using the false starting point of the path, false calculations for obstacles in the swerving area and assumed angle of an obstacle. It appeared that in some cases the deviations are large and an approximation of the real path length can not be guaranteed. Improvements on the calculations have to be done to reduce the path length errors before it is usable for haptic feedback.

Having reflected back on the research questions it can be concluded that the implementation as is contains all the necessary structural components from measurements to haptic feedback. This has given a good basis to further improve the system. This is needed as collisions with obstacles can not yet be fully prevented. Recommendations to improve the system are given in Section 5.2.

5.2 Recommendations

From the conclusions drawn in Section 5.1 it appeared that the performance of the system can be improved. These improvements will be discussed now.

For the sensor system there are 3 recommendations to improve the performance of the system. First, the 9 remaining sensors must be mounted on the platform. At this point, only 3 sensors are mounted due to time constraints. All the sensors and cases are already available, but wiring and mounting is yet to be done.

The second improvement might be to implement data filtering, which is not done right now. During the implementation phase it became clear that there are fluctuations in the data measurements in a static environment. The difficulty in this, is to determine whether fluctuating data is the result of detecting different obstacles or sensor characteristics.

The third improvement is to increase the update frequency. The PulseIn() function used to set the first timestamp must be replaced by a different function. Also, removing the sequential measuring increases the update frequency but this can only be done if interference is not an issue. This should be tested.

For the obstacle processing system there are also 2 recommendations which could improve the performance of the system. The first improvement is to change the angle approximation method. The method used now, assigning the sensor angle to the obstacle angle, does not guarantee a collision prevention. A different method is to take the worst case angle. This means that an obstacle is positioned at an angle which is the biggest threat to the platform. This can be defined as the angle at which an obstacle is in the collision area and has the shortest path length. It might still be possible that there is a significant path length error, but it ensures that at least all collisions are prevented. Another method is, to use probability theory for approximating an angle to an obstacle, this might decrease path length errors.

The second improvement is the calculation of the path length to an obstacle. In this, there are 2 parts to discuss. The first one is to decrease the path length error for obstacles in the swerving area. Now, the system assumes these collisions only occur at the front of the platform. An extra check must be done to verify that an obstacle is in the swerving area. Changing the angle calculation can be done as mentioned in Section 4.8. The second and most challenging improvement is the starting point of the path. Now, the starting point is assumed in the center of rotation line. A method must be implemented that determines the point of collision on the platform. The path length is then equal to the arc from the collision point to the obstacle.

Bibliography

[1] Datasheet Ultrasonic sensor, visited 25-10-17, https://www.maxbotix.com/documents/LV-MaxSonar-EZ_-Datasheet.pdf

[2] Matlab data plots, Software: Matlab R2016a education edition, MathWorks, https://nl.mathworks.com/company.html

[3] Beam pattern drawings, Software: SolidWorks 2016 education edition, Dassault Systèmes SOLIDWORKS Corp, Waltham, Massachusetts, USA http://www.solidworks.com/

[4] Chaining method Ultrasonic sensors, visited 30-10-17 https://www.maxbotix.com/tutorials1/031-using-multiple-ultrasonic-sensors.htm

[5] IPM platform drawing, visited 02-11-17, https://tex.stackexchange.com/questions/13933/drawing-mechanical-systems-in-latex

[6] Design and Implementation of a 3 Layered Obstacle Detection System with Haptic Feedback Notification, K. Amr, University of Twente, BSc Report 033RaM2017, 2017

[7] Design of an anti-slip control system of a Segway RMP 50 omni platform, R Lieftink, University of Twente, BSc Report RaM 2017, 2017

[8] On design and realisation of a telemanipulation demonstration setup, A. (Arnold) Hofstede, University of Twente, BSc Report RaM 2017, 2017

[9] Visuo-Haptic Interface for Teleoperation of Mobile Robot Exploration Tasks, Nikos C. Mitsou, Spyros V. Velanas and Costas S. Tzafestas, arnumber=4107802, 2006, visited 06-11-17 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4107802

[10] Predictive Haptic Feedback for Obstacle Avoidance Based on Model Predictive Control, Avinash Balachandran, Matthew Brown, Stephen M. Erlien, and J. Christian Gerdes, 2016, visited 06-11-17 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7332989

[11] Haptic obstacle detector for the blind, Master of Science Thesis, The Chinh Nguyen, Stockholm, Sweden 2014, visited 06-11-17, page 33 http://www.diva-portal.org/smash/get/diva2:818827/FULLTEXT01.pdf

[12] Force Feedback Implementation Based on Recognition of Obstacle for the Mobile Robot Using a Haptic Joystick, Dong-Hyuk, Lee Kyoung-Taik, Park Sun-Kyun Kang and Jangmyung Lee, visited 07-11-17, page 85 https://link.springer.com/content/pdf/10.10072F978-3-642-40852-6.pdf

A Initial idea for haptic feedback

In the previous work done the haptic feedback system is modeled as a linear spring damper system. The application has been tested and the conclusion was that there was a drop in force interpretation especially at closer regions to the obstacles [6]. This appendix will present an initial idea to possibly increase the force interpretation. The drop in force interpretation might be caused by the linear operation as it was hard to estimate the distance to an obstacle based on the force feedback. To increase the sensitivity at short distance a non-linear model of the spring damper system could be applied. This to increase the force interpretation and to make sure that distance estimation to an obstacle is improved. Keeping the above in mind the following applications of a non-linear spring damper system are proposed and will be discussed:

- Exponential function
- Increasing linear function
- Staircase function

The first algorithm is the exponential function. Due to its exponential behavior, the slope of the function and thus the force sensitivity also has an exponential behavior. The force equation shown in Equation 2.14 needs to be changed as shown in Equation A.1. A visualization of this equation for the distance component is shown in Figure A.1. The velocity component would have a reversed behavior as the maximum force feedback should be given at maximum velocity. This is opposed to the distance component where a maximum force should be given when the distance to an obstacle is minimal.

$$F = e^{c \cdot \frac{dx}{dt}} + e^{\frac{k}{x}} \tag{A.1}$$

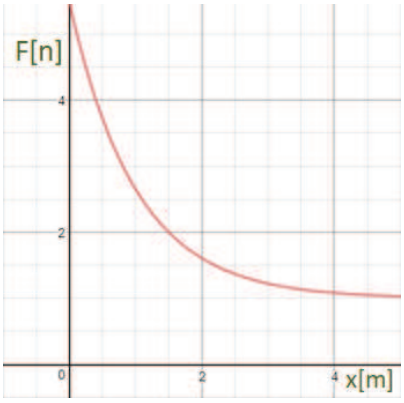


Figure A.1: exponential function

the constants P and Q will determine the sensitivity to a change in distance and velocity. In terms of having a changing force sensitivity when varying distance this method has the highest sensitivity. Especially at close distances to an obstacle the force sensitivity increases a lot which is definitely a positive aspect of this function. On the other hand when talking about distance interpretation the user might not able to match a certain force to a certain distance as there is no sudden force increase.

The second listed function is the increasing linear function. It uses a stepwise operation and is a linear function within an area but its slope increases when reaching the next area. The slope increase would be dependent on the velocity of the platform. In physical terms this means that the spring and damper constant is increased when reaching the next area. The same IPM shown in Figure 2.14 can thus be used to represent the system. Equation 2.14 is used but the constants C and K are varied as shown in Equation A.2. A visualization of this Equation is shown

in Figure A.2

$$F = \begin{cases} c_1 \cdot \frac{dx}{dt} + \frac{k_1}{x} & for\ x_1 < x \leq x_2 \\ c_2 \cdot \frac{dx}{dt} + \frac{k_2}{x} & for\ x_2 < x \leq x_3 \end{cases} \tag{A.2}$$

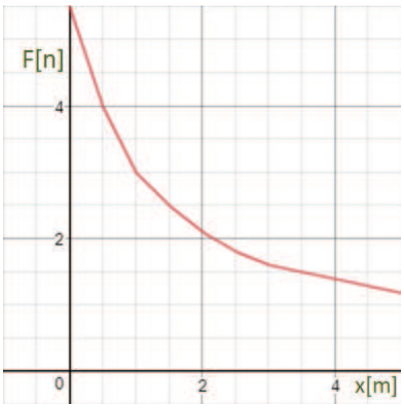


Figure A.2: increasing linear function

In this algorithm it is possible to experience distance change within an area which is an advantage. Also its slope is increasing when approaching an obstacle so the force sensitivity is changing at certain distances. A decision has to be made on the size of each area to reach the best qualitative result.

The last mentioned function is the staircase function. The staircase function could be used to give a sudden increase in force. It would divide the detection area in smaller areas where there is a constant force within one smaller area. When the distance to an obstacle decreases and a different area is reached the force is step wise increased. The size of the increment would then be dependent on the velocity of the platform. This is shown in Equation A.3. A and B are force constants and x represents the actual distance to an obstacle and thus not the displacement of the spring. A visualization of this equation is shown in Figure A.3

$$F = \begin{cases} A & for\ x_1 < x \leq x_2 \\ B & for\ x_2 < x \leq x_3 \end{cases} \tag{A.3}$$

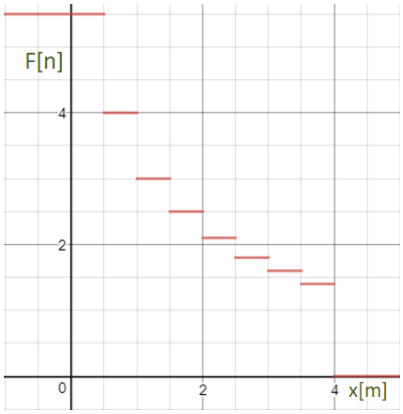


Figure A.3: staircase function

The advantage of this function might be that there is a sudden change of force which will alert the operator. Also the areas are predefined so an operator could know in which specific area

the obstacle is. This could increase the distance estimation of the user. This could also be a disadvantage because it might be disturbing for the operator to have non continuous force feedback. Another downside of this algorithm is that the user experiences a constant force within each area. Any distance change within the area goes unnoticed by the operator. It depends on the area size whether this is a significant problem. A solution could be to have small area sizes when the platform is close to the obstacle such that there are many steps at close range. This can be seen as the same principle a parking sensor uses. The difference is that the parking sensor uses audio feedback. When the car approaches an obstacle the sensor will increase its frequency to indicate how close the obstacle is. With the haptic feedback it will increase its step frequency to alert the driver.

To conclude the haptic feedback application part it is hard to predict which algorithm would fit the requirement of "intuitive feeling". It is a qualitative factor rather than a quantative factor. Because of this it is not possible to define the best method for this application before physically testing each application. An experiment must be designed in which each of the applications is tested and a conclusion must be drawn on which application fits the intuitive feeling best. After this a comparison should be made between the existing linear application and the newly chosen application. A conclusion must be drawn whether the force interpretation and distance estimation has increased.