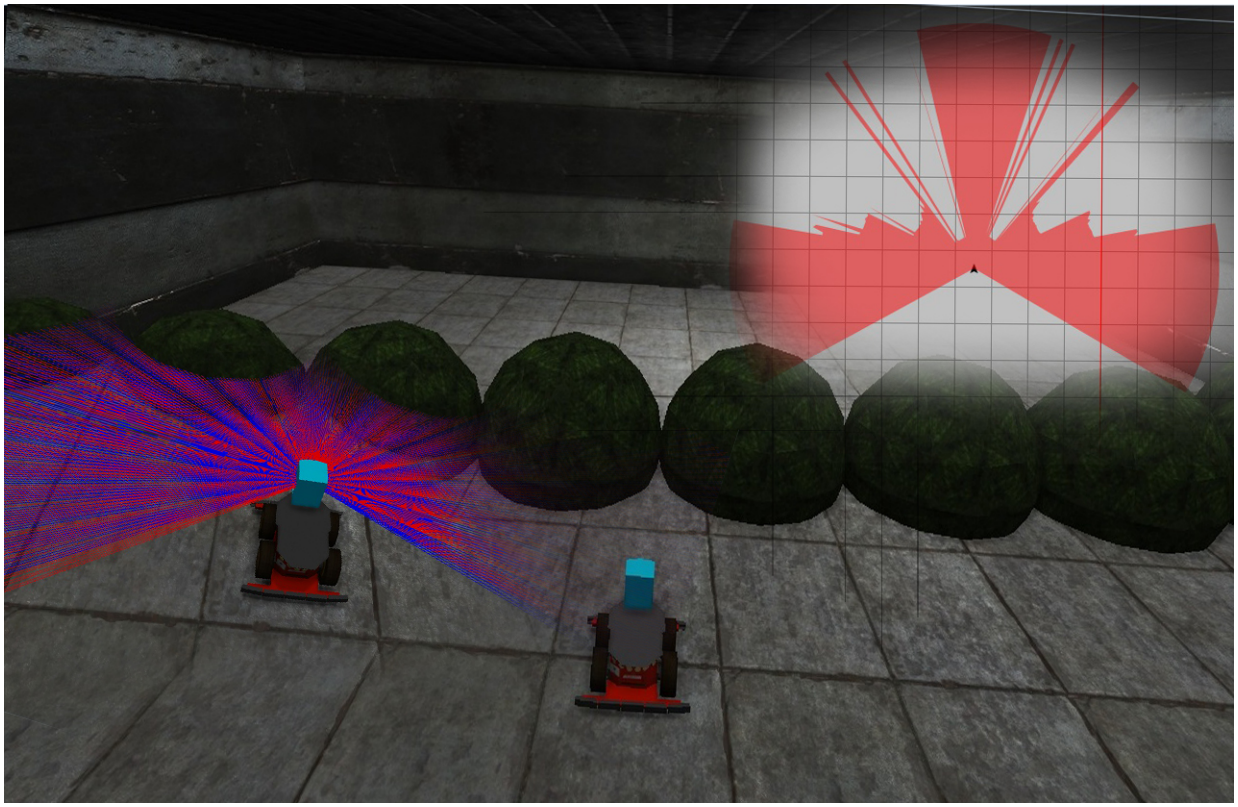
NAVIGATION USING A RADAR SENSOR IN USARSIM



Navigation using a radar sensor in USARSim

Richard Rozeboom
6173292

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dr. A.Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 26th, 2012

Abstract

This thesis describes how, in the USARSim environment ray tracing can be used to implement a virtual radar sensor and how such a sensor can be used for navigation. What makes the radar more desirable than other sensors is that a radar is able to capture information through materials such as light foliage and smoke, and is able to capture the reflection of multiple objects at once. Previous work has analyzed large multi sensor datasets in order to demonstrate the behavior of a radar system in a real world environment.

Contents

1 Introduction

Information gathering about the environment has always been one of the most important aspects of robotics. It is crucial that a robot or agent has the information it requires to localize itself and its surroundings in an environment that may be difficult to navigate in. To acquire information about the environment senors are used in a wide variety of applications. Proximity sensors can be used in car safety applications and military applications, and motion sensors are used for security purposes, to name a few.

One sensor that can gather information of the surroundings is the radar, which has recently become lightweight and affordable, and has several advantages compared to other sensors, one of which is being able to capture information through certain densities of foliage and smoke whereas other sensors like cameras and laser scanners would perceive an obstacle. Another advantage is that a radar can perceive multiple objects from one reflection at once.

Another important aspect of robotics is simulation, for it is necessary to first test the behavior of the robot or agent without endangering any equipment. A robot might not act as expected and damage itself in the process, making testing of the robot an expensive procedure. Secondly a simulator allows for customization of the environment and circumstances, which in turn allows for reproducible experiments. Furthermore, a simulator grants access to visual aides and visualizations such as the ground truth. which are not always available in a real world environment. One example of such a simulator is USARSim (Unified System for Automation and Robot Simulation), which is a detailed simulator made for robots and their environments, and is used by many robotics projects such as the Urban Search And Rescue project, and the annual RoboCup.

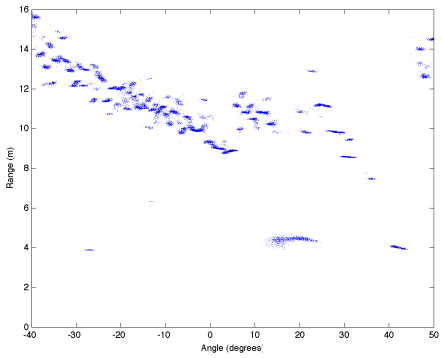
This thesis presents a way to implement a radar sensor in USARsim using ray tracing to capture information about the surroundings in the simulated environment. Furthermore, a way to navigate in an environment where several obstacles like foliage are present will also be discussed. Using the radar sensor to map environments in conditions that prove difficult to other sensors will make it clear that the radar has advantages compared to other sensors in such situations.

2 Related Work

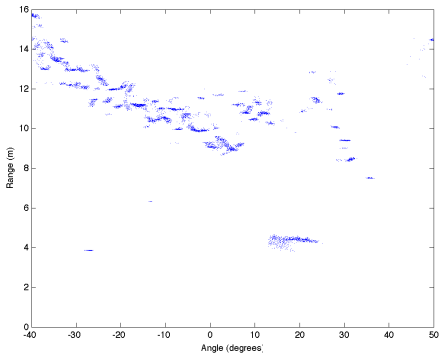
The Marulan datasets are large accurately calibrated and time-synchronized datasets which hold the readings of multi-sensor vehicles that have been scanning a real world environment, and are described in the work *Multi-Sensor Perception for Unmanned Ground Vehicles*[6]. These datasets enables research and validation towards several sensors under not only normal circumstances such as an open field on a clear day, but also extreme circumstances with obscuration such as heavy rain and smoke. The scenes that the vehicles have scanned are made of ordinary objects such as plants, poles and boxes. One of the scenes that has been used in the making of the Marulan datasets is shown in figure 1.

Figure 1: Static trial scene, figure from Peynot and Scheding [6, fig 5]

The vehicle is equipped with many sensors, one of which is the radar, and in *Multi-Sensor Perception for Unmanned Ground Vehicles* some of the advantages of the radar sensor become apparent when plotting the data from the sensors. For example, the readings from a radar in clear conditions and in heavy dust is illustrated in figures 2 and 3 respectively, and the readings of a laser range scanner in the same conditions are illustrated in figures 4 and 5. The images make it clear that the radar has little to none interference in the dusty conditions, whereas the laser scanner is influenced significantly. Tests that have taken place under different conditions like heavy rain show similar results.

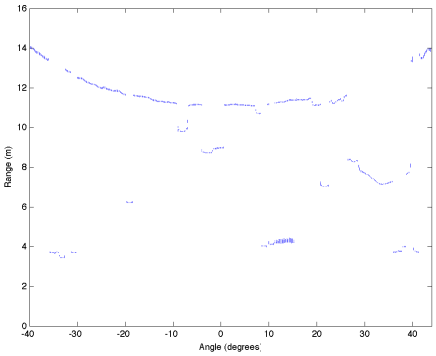


(a) Figure 2: Radar readings, clear conditions

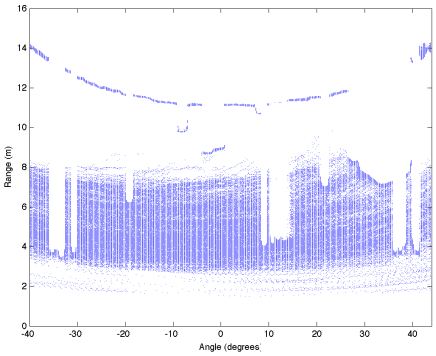


(b) Figure 3: Radar readings, dusty conditions

The data from the radar in the datasets has been used for research to measure the capabilities concerning radar simulation, among other goals. More specifically, the thesis *Virtual radar sensor for USARSim*[5], in which the Marulan dataset has been thoroughly analyzed to demonstrate the behavior of a radar in a real world environment, and to answer the question 'how realistic a new sensor based on radar technology can be added to the USARSim



(c) Figure 4: Laser readings, clear conditions



(d) Figure 5: Laser readings, dusty conditions

Radar and laser readings in clear and heavy dust conditions, figure from Peynot and Scheding [6, fig 26-28]

world'. The thesis offers a method to simulate a radar sensor, and concludes that simulating a radar sensor in the USARsim environment is possible. In a way *Navigation using a radar sensor in USARsim* is a continuation of the research on a virtual radar in USARSim, as this thesis provides an implementation of such a radar sensor that is presented in *Virtual radar sensor for USARSim*.

As stated before, simulation is a valuable asset in the field of robotics. USARsim (Unified System for Automation and Robot Simulation) is a simulator for robots and their environment based on UDK (Unreal Development Kit), and is used for research in the field of robotics in numerous occasions. *USARSim: a robot simulator for research and education*[2] is the work that has presented the field of robotics with USARSim, and provides its general architecture, describe examples of utilization, and provide a comprehensive overview for those interested in robot simulations for education, research and competitions.

In the Metsäteho Report, "Sensing Through Foliage"[4] the different technologies capable of seeing through foliage are examined to achieve better measurement and perception of trees, ground and obstacles that harvesters might encounter in the wood production. The report concludes that the most sensor technologies that are able to see through foliage are FMCW radars operating on gigahertz or terahertz frequencies, and that they can form accurate 3D images of the environment. Furthermore the report states that the only drawback is that the most accurate radar systems are very expensive measuring devices, and not easily obtained.

In the paper *Mapping of the environment with a high resolution ground-based radar imager*[7] the potential is presented of microwave radar functioning as a high resolution ground-based imager, in order to build radar maps in environmental applications. A new radar sensor names K2Pi is described, which makes use of FMCW (frequency-modulated continuous wave). Using an algorithm that is based on SLAM (simultaneous localization and mapping) global radar maps have been constructed using a data merging process.

In the book *Principles of Robot Motion: Theory Algorithms, and Implementation*[3] the advances that have taken place in the the past are reflected, which includes sensor-based planning, probabalistic planning, localization and mapping, and motion planning. Its presentation makes the mathematical side of robot motion accessible to students of computer science and engineering. This book features several useful algorithms that can be used to navigate using the radar sensor.

3 Radar sensor

In order to perceive objects, a radar sensor has a transmitter that sends out radio waves of a certain frequency making the radar waves collide with the surrounding environment. These reflected waves are then scattered from the point that the wave has hit an object, and some of the reflected waves will return and be measured by the radar receiver. The time between sending a receiving a pulse gives a measurement of the range of the object the radio waves reflected off of. The process of sending and receiving radio waves is illustrated in figure 5.

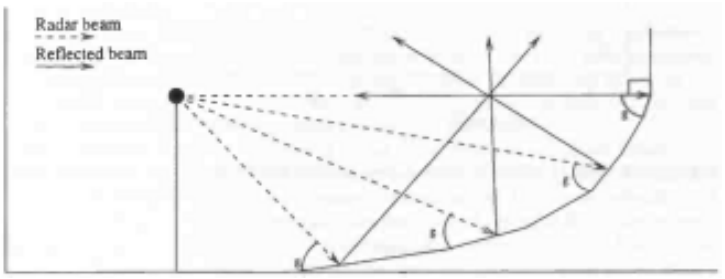


Figure 5: Radar beam scatter, image from Bosman [1, fig 4.3]

A radar system transmits and receives the radio waves by making use of one or more antennas. Separate antennas can be used to send and receive signals however due to interference it is more common to use one antenna that is able to switch between sending and receiving. The direction in which a radar transmitter can send its signal is 360° in one plane, which it goes through in steps. Small steps will ensure high accuracy, but will require more radio signals to be sent for the full range to be covered.

In the field of robotics measurements are needed of 100 meters and below, which requires making use of high-frequency electromagnetic waves. Loss of signal intensity is to be kept at a minimum to keep the transmission energy as low as possible. The loss of signal intensity is at its lowest when the frequency of the signal is 94GHz which is reserved for military and experimental applications such as robotics. Frequencies between 40GHz and 300GHz toughly corresponds to wavelengths between 1 and 10mm[8].

Conventional radar systems are not suitable for measurements in the field of robotics because such a radar system would be required to be of high precision and speed, making it very expensive equipment. An alternative to counter this problem is a radar system that makes use of FMCW, Frequency Modulation Continuous Wave technology. Using this the frequency shifts in

a way that makes it easier to calculate the time it took for a signal to return, and thus to determine the range. The largest advantage to robotics of FMCW is the lower transmission power that makes it safer and more efficient to use.

Using radar has several advantages compared to other common visual based sensors such as laser scanners and video, the greatest advantage being that radar readings are less effected by visual conditions such as weather, smoke or light foliage (An example is shown in figures 2-5). Microwave radar has been used to build maps of the environment using a radar that scans in 360° at 24GHz[7].

3.1 Radar noise and interference

The radar can be adjusted to scan with a very high resolution, resulting in highly accurate readings. In practice a radar module is often set to a lower resolution, which has the benefit of reducing computational load considerably. This comes at a cost however in the form of noise and interference. It is also possible for a signal to only partially hit an object, which means that the rest of the signal will fly past the object, potentially hitting another object. This would cause the radar to perceive two objects in the same direction, and on two different ranges[5]. This can occur due to the beam divergence of the signal, which holds that the beam diameter increases with distance from the origin. In some radar systems only the information is kept of the object that returned the highest reflectivity, and this can cause fluctuations in readings coming from one direction.

4 Radar sensor in USARsim

USARsim (Unified System for Automation and Robot Simulation) is a simulator for robots based on UDK (Unreal Development Kit) with several tools and sensors that make a realistic simulation possible. The radar sensor however, is not yet one of the implemented sensors that are available in USARsim, and with the use of a tool called ray tracing it is possible to implement a sensor that would have the same behavior as a radar. Previous research, *Virtual radar sensor for USARSim*[5] suggests that it is indeed possible to model such a sensor in USARsim to some extent, by having analyzed the data of a radar from large, accurately calibrated and time-synchronized datasets, The Marulan Datasets[6].

Ray tracing in USARsim allows for a beam to be sent from the sensor through the scene/environment, and doing so will gather information from the virtual objects that it goes through, making it possible to retrieve information such as object type, density, and transparency of objects that the sensor detects. Using this information it is possible to determine whether a radar beam would go through material, and if so, how far would it go through. This is essentially all that is needed to implement a sensor that behaves like a radar in the simulated environment.

Once it is made clear how far the radar module should be able to see through materials such as foliage and smoke, it is needed to build an internal representation that can be used to localize the agent and the surroundings. Some entities in the simulator, for example smoke, are safe for an agent to

navigate through, trying to move through a shrub however might render the agent unable to move, even though the radar can gather information through both of these objects. Therefore it is necessary to make a clear distinction between such virtual objects in the internal representation of the surroundings.

For the radar sensor to be able to move through a simulated environment it will need a way of transportation. The USARsim environment has several bots/vehicles available, which can be adjusted to be equipped with any sensor. One of which is the P3AT which has been chosen to be equipped with the new radar sensor in order to move throughout the environment, and is shown in figure 6. The P3AT has been chosen because it resembles the vehicle used in the Marulan data gathering[6] the most, and is also easily controlled.

4.1 Test environment

In order to test new implementations of radar sensors, a map is needed that demonstrates the advantage of the radar. To make an environment that cause difficulties to other sensors, a map has been made for USARSim in which other sensors would see a solid 'wall' of objects, whereas the radar sensor would be able to sense gaps in the wall. The basic foliage that is used is made of spherical shapes and has been given a property for the sensor to recognize as foliage that is thin enough to be able to penetrate. Alternatively other objects can be chosen for other scenarios. Spherical shapes have been chosen in order to create thin sections in the wall, and the round shape ensures that the readings are diverse enough to analyze. The foliage that is available in the UDK environment have cubic boxes for collision models, which would make the readings insufficiently diverse. The test map is shown in figure 6.

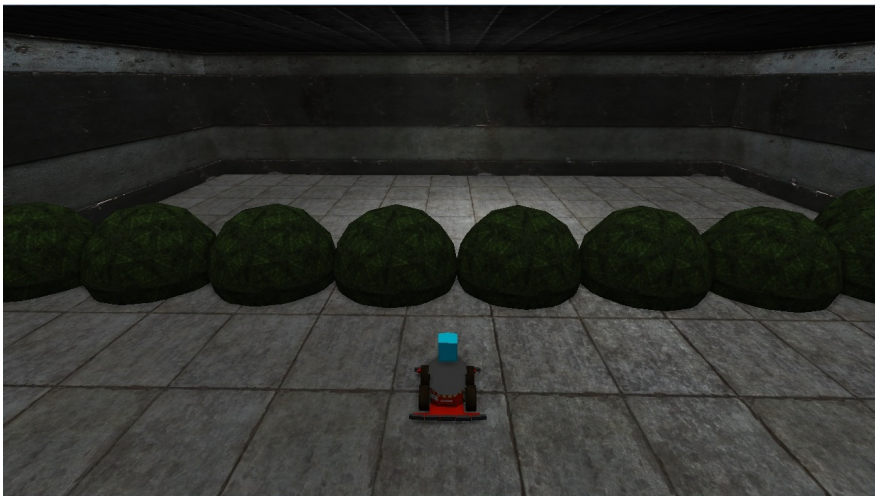


Figure 6: Test environment of the radar.

The map has been designed so that when passing the foliage the radar sensor is able to sense what is behind the obscured part of the map, whereas an other visual sensor will not be able sense in that area of the map from any position on the left of the foliage. An other visual sensor that is available in USARsim, the Hokuyo sensor has been chosen to demonstrate this behavior in the

test environment. The Hokuyo sensor is not able to sense past the obstacles, in this case light foliage, which prevents exploration in the part of the room that is obstructed. The readings of the Hokuyo sensor made in the test environment are shown in figure 7.

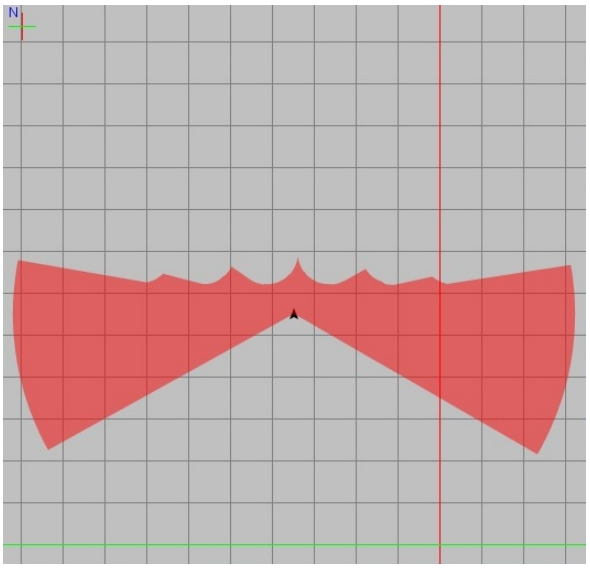


Figure 7: Sensor reading of the Hokuyo sensor module.

4.2 Radar Module

The radar implementation is a modification of the range-scanner module in USARsim. Using ray tracing it is possible to cast a ray from the sensor through the environment as a radar system would cast radio waves. The raytracing method not only returns the range of the detected object, but also the material and other information, and doing so will enable the sensor to determine how far the beams should penetrate through the material.

The radar sensor is utilizing an algorithm that finds the range recursively by casting rays through the scene. If a normal object is hit by the rays, there is no need to calculate the range any further, and that range is returned. However if an object is hit that is made from material that a radar wave can penetrate, the algorithm casts a ray from the point of collision through the object to a certain extent, depending on the penetration power of the radar, and the distance the ray has gone through objects before, and the density of those objects. When a ray goes through an object, the penetration power is decreased until it runs out, at which point the range is returned. This means if an object is too large to sense through, the range that is acquired lies somewhere between the 'entrance' point of the ray, and the back side of the object.

Using this implementation, the radar sensor is able to see through the foliage to a certain extent. The foliage in the test environment consist of spherical shapes, and the penetration power is insufficient to see through the center of the foliage, yet the edges of the foliage are thin enough to see through. This leads the sensor to see holes through the wall of foliage, and uncover

the section of the test environment that is obscured. This behavior can be seen in the ground-truth of the radar sensor, and is shown in shown in figure 8.

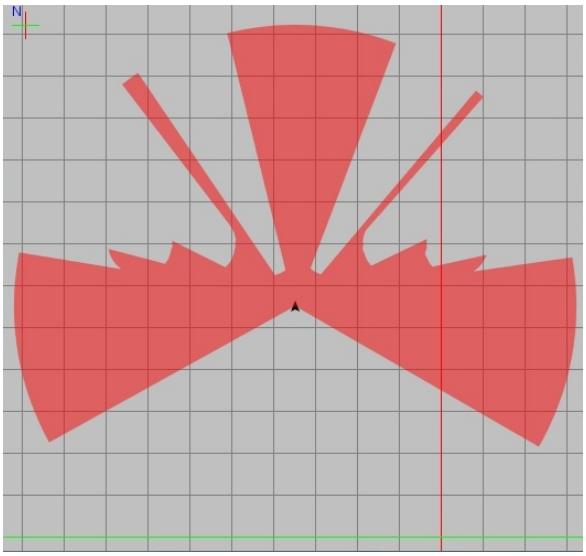
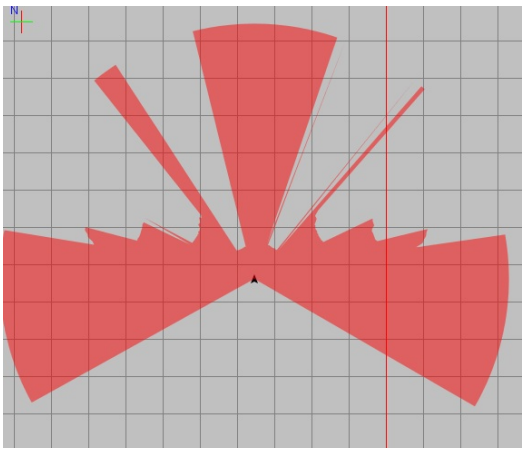


Figure 8: The ground-truth reading of the radar sensor module.

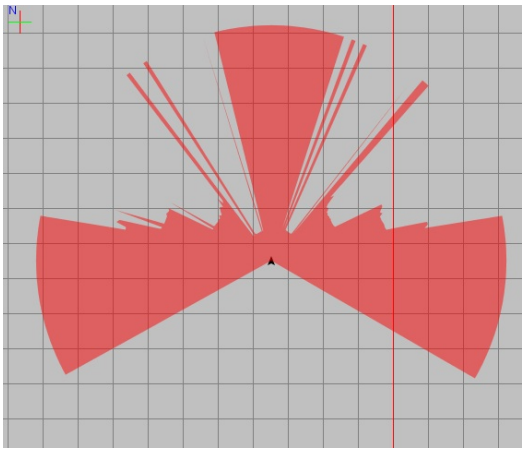
4.3 Simulating radar noise and interference

To recreate noise and interference for the virtual radar sensor, a model has been made to simulate beam divergence. The rays that are cast in the UDK environment are infinitely thin, therefore the model uses not only one beam, but also casts two 'side beams' for each original ray that the sensor would cast. Because the sensor is scanning in a horizontal plane, the extra beams that are cast are also in that plane. The are cast to the left and to the right of the original beam, relative from the position of the sensor. Each ray has a chance to be used to acquire the range. In this way, interference that occurs when a beam comes close to grazing an object is simulated. The offset of the side beams is measured in units in the UDK coordinate system, which will be referred to as N. The simulation of noise becomes visible in the ground truth, and two examples with N=50 and N=100 is shown in figure 9 and figure 10 respectively.

To illustrate how this takes place in the UDK environment, images 11 and 12 demonstrate the extra beams that are cast. The red lines represent the beams that are cast left of the original beam, the blue lines are the beams that have been cast on the right side. The space where the two kinds of beams overlap each other does not seem to be giving much information, however on the left and right side, it is clear that the beams exceed each other. The amount N=100 is shown for demonstration purposes only, as 100 units would be of little use to simulate usable or realistic noise. The recommended amounts for N are 50 and lower, however these amounts are insufficient for demonstration purposes. The sensor is mounted on a P3AT vehicle from the USARsim environment.



(e) Figure 9: Radar readings in test environment with N=50



(f) Figure 10: Radar readings in test environment with N=100

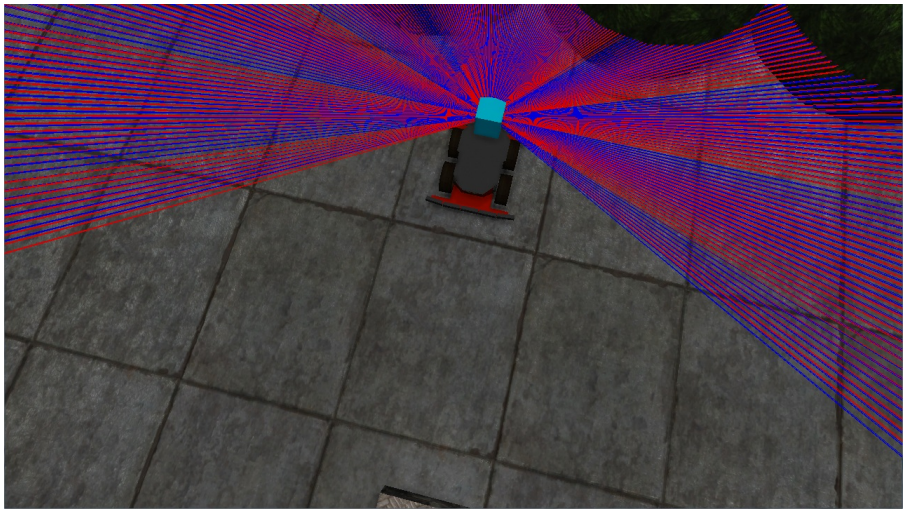


Figure 11: The radar module on the P3AT with visible offset-beams with N=50.

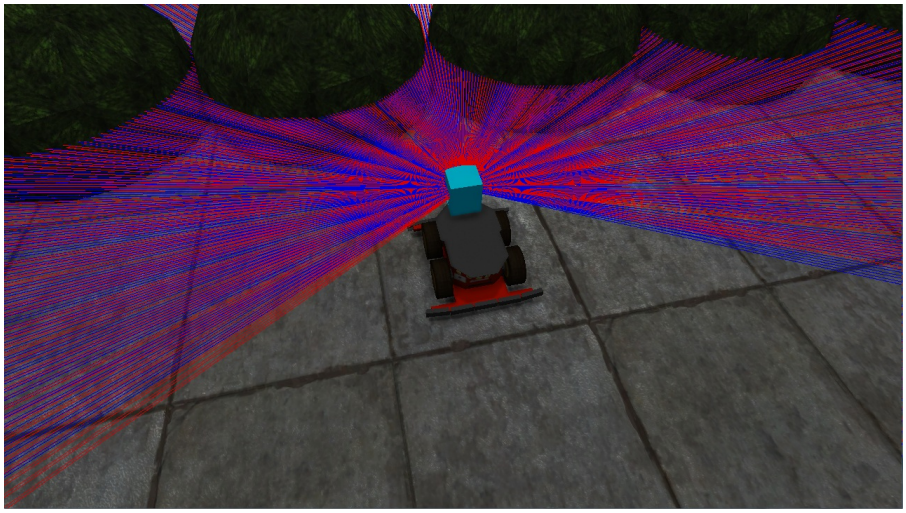


Figure 12: The radar module on the P3AT with visible offset-beams with N=100.

5 Navigation

In order for a agent to fully localize and map a space that is larger than its senor range or obstructed from the view of its sensors, the agent needs to move throughout the space. Navigation of agents equipped with a sensors such as a radar to create a map of the surroundings can be done in several ways, for example the agent could head in a different direction each time it would be close to a wall or other obstacle, much like how a basic vacuum bot would navigate through a house. Making a map in an efficient way however, can be done in a much more systematic way such as following a wall once it has been found to find the contours of the room, after which the agent needs to put the interior of the room on the map. This method proves useful for environments such as enclosed rooms, however it might be possible for the agent to be in a border-less environment, in which case a center-out approach would be of more use to map the surroundings and find objects. Ultimately the agent should navigate using the information that it acquires (or does not acquire), choose between exploration strategies, and act accordingly to efficiently map the area.

5.1 Navigation algorithm

To demonstrate that complex navigation is possible using a radar sensor, a basic algorithm has been chosen to illustrate how radar sensor data can be used for the purpose of navigation. The algorithm is simple and is similar to the vacuum bots' behavior mentioned earlier. The chosen algorithm is the Bug algorithm[3], which is best explained using the image shown in figure 13.

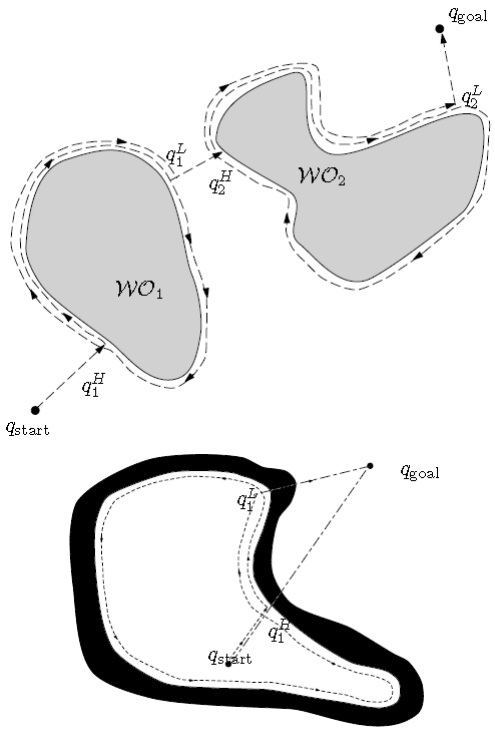


Figure 13: The Bug1 algorithm. (Top) The Bug1 algorithm finds a path to the goal. (Bottom)The Bug1 algorithm reports failure.

The Bug1 algorithm holds the following: Go for the goal or way point in a straight line until an obstacle is encountered. The place location where the agent has detected the obstacle for the first time will be called the hitlocation. Follow the boundary of the obstacle (either left or right) until the goal is reached or until the hitlocation is reached again. If the hitlocation has been reached again, determine the closest point from the boundary of the obstacle to the goal, this point shall be called the leaving point. Traverse to the leaving point and repeat the process, starting from going to the goal in a straight line. If the line connecting the leaving point to the goal intersects the obstacle that was previously encountered, conclude there is no path to the goal.

There is also a variation of the Bug1 algorithm, simply called the Bug2 algorithm. This algorithm will cover less ground than the Bug1 algorithm, but still ensures finding a path to the goal if it is possible. The Bug2 algorithm is similar to the Bug1, and is as follows: Head for the goal in a straight line until an obstacle has been detected. Follow the boundary of the obstacle until the initial line going to the goal has been crossed. If at this point the agent is closer to the goal, repeat the process. If the agent is not closer to the goal once it crosses the line, conclude there is no path leading to the goal. The Bug2 algorithm is shown in figure 14.

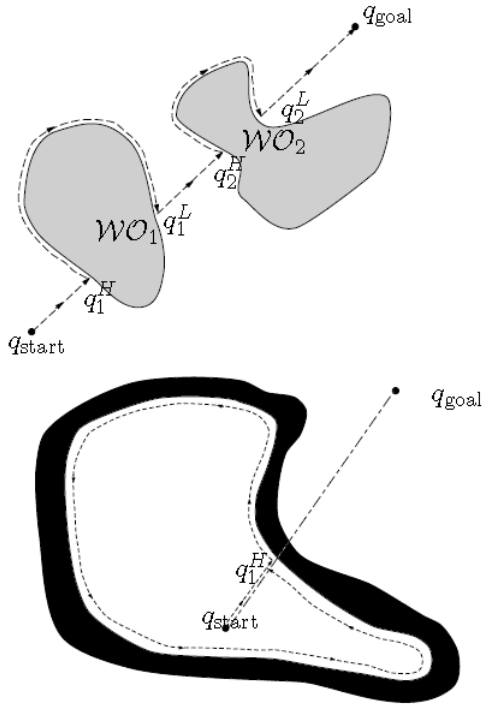


Figure 14: The Bug2 algorithm. (Top) The Bug2 algorithm finds a path to the goal. (Bottom)The Bug2 algorithm reports failure.

At a first glance it might seem like the Bug2 algorithm is more efficient since it will cover less ground. However it might be more desirable for the agent to explore the surrounding area before reaching the goal, in which case the Bug1 algorithm would be more appropriate. Besides the difference of how much ground coverage might be desirable, it is possible to think of an obstacle in which the Bug2 algorithm is less efficient than the Bug1 algorithm. Consider an obstacle boundary that intersects the line to the goal n times. The Bug1 algorithm will traverse at most one and a half of the boundary before leaving for the goal or other obstacle, whereas the Bug2 algorithm will traverse at most almost the entire boundary n times. The most suitable algorithm of the two therefore entirely depends on what obstacles are present in the environment. For simple object the Bug2 algorithm will perform best, whereas the Bug1 algorithm will outperform Bug2 when there are more complex obstacles encountered.

5.2 Using the simulated radar data for navigation

To use the radar data for navigation purposes, the range of each individual ray must be considered. Due to the large amount of rays that are cast, it would be best to make several groups that categorize parts of the beams, making a long range 'bumper'. Navigating using four to eight bumpers instead of 360 rays(for a full circle with a 1 degree step size) is considerably easier. The groups could be simply 'left, right, front and back' but more bumpers are also suitable, depending on the task at hand, and the objects the sensor

will encounter. The ranges to which a bumper must activate depend on the vehicle the sensor is mounted on, and the density/penetration of the objects that the radar can sense through. For this radar module mounted on a vehicle using default settings, it is recommended to stay away from obstacles for two to three times the length of the vehicle. This will ensure that there will be no collision with the vehicle and surroundings, and also takes into consideration the amount of the foliage the sensor senses through. Once the agent stays away from obstacles, it must be determined when a bumper is pressed safely. This can be done in the following way: If all rays in the left bumper are between the safety-distance and safety-distance + 50cm, the left bumper is pressed. This length of course depends again on the situation and obstacles that are encountered.

Once the groups of rays are configured to activate on the right times, a few simple rules must be created to be able to use the Bug algorithm. When an object is encountered by the bumpers, the agent must align itself with the obstacle boundary and must turn until only one of the side bumpers is being activated. This means in the four-sided example, that the three other bumpers two of which are front and back, are all not being activated. The agent can then move forward and backward, following the boundary of the obstacle until the state of one of the bumpers has changed. If the previously activated bumper is not activated anymore, lets say the left, the agent has gone too far from the boundary and must move towards the left until the bumper is activated again. If the front (or back) bumper is activated the agent must have found a corner or turn in the boundary. To follow the turn the agent must steer away from the already pressed side bumper, and move towards the right until the front bumper is no longer activated, but instead the back is now activated. The agent is now aligned again and can move forward, along the boundary of the object. Once these rules have been implemented to follow boundaries and to align to surfaces, the Bug algorithm can be implemented.

6 Results

The radar sensor module has been mounted to the P3AT vehicle, which has been driven through the test area using the interface USARCommander. The USARCommander can not only control the agents in the USARsim environment, but can also use the sensors used to make a map using SLAM, Simultaneous localization and tracking. The result of driving through the area and using SLAM to create a map is shown in figure 15. The blue area is where the sensor cannot sense, the white area is what the sensor can sense, a series of black dots represent obstacles such as walls, and gray areas are clear space to the sensor.

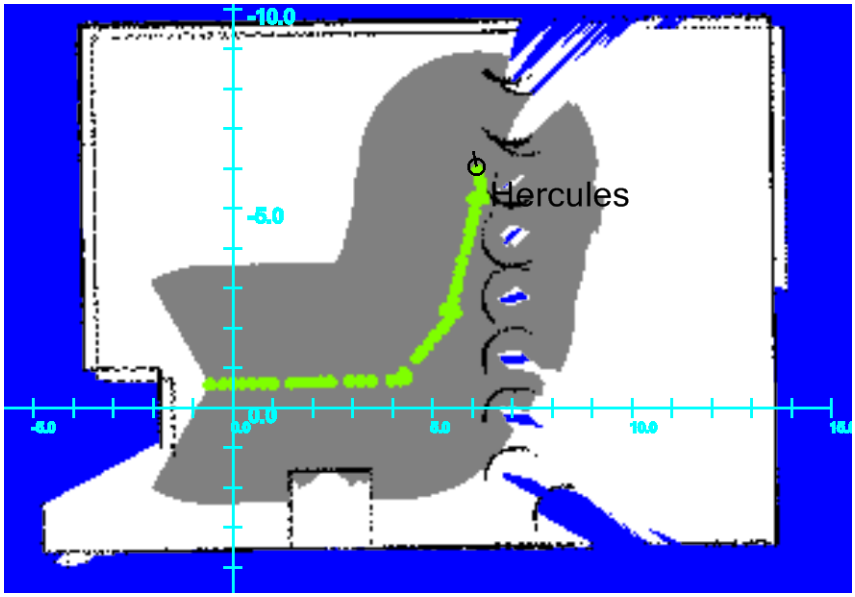


Figure 15: Top-down view of explored test environment. Image generated by UsarCommander.

As is shown the area to the right side of the foliage is mostly white, and accessible for the sensor. Small cores of blue reside in the center of the pieces of foliage, where it is too deep for the rays to penetrate through. The gray space that is also present in the foliage is clear to the sensor, however not clear for the agent himself, as indicated by the series of black dots around the foliage. Other sensors showed similar results as shown in figure 7, as they are not able to sense through the foliage they leave the right side of test room blue in the readings.

7 Conclusions, discussions and future work

A basic model of radar module has successfully been implemented in USARsim which can now be used to test radar sensors in a safe and simulated environment. The radar has a distinct advantage compared to other sensors, as it is able to sense through some visual obscuration, in this case foliage. Results show that the virtual radar sensor is capable of sensing through the simulated foliage, where other visual sensor are not. Furthermore it has been made clear that navigation using a radar sensor is possible using this implementation of the radar system.

The radar module is a model, and has not been fine tuned to a specific model, however it is possible to make adjustments to the module with little effort, to make it behave like a specific commercial model. Doing so would require to adjust the specifications such as resolution, step size, range and noise level to achieve similar results of the desired commercial model. Such modifications require minimal effort as the variables that account for those parameters are accessible and are easily adjusted.

The current implementation of noise (illustrated in figures 10-13) uses a chance model that gives all rays(one original, 2 offset rays) that are cast an

equal chance to be chosen to determine the range. Adjustments can be made for the chances to increase or decrease depending on the angle of the surface that was hit by each ray. Two rays are cast, however beams diverge in all directions, not only left and right. To approach a model that is more similar to beam divergence, more rays could be added pointing towards certain directions. The amount of rays have been kept low due to the computational load it requires to cast more rays. Each ray that successfully hits an object the radar can penetrate requires retrieving the range recursively. There could be improvement in optimizing the algorithm that calculated the range.

The navigation algorithm discussed earlier can also be implemented to demonstrate that a vehicle equipped with a radar sensor can navigate in an area without the use of other sensors. Other improvements are also possible for example adding new behavior to materials, such as radio wave absorbing substances and reflecting surfaces. New objects can also be added, however this will require building a collision model for the objects. Objects such as trees and other foliage exist in the UDK environment, however the collision models are not very detailed, giving readings that would be the same as detecting a cube or box. Adding collision models to these objects would allow for more realistic scenes and readings.

References

[1] D. Bosman. Radar image fusion and target tracking. 2002.

[2] Stefano Carpin, Michael Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. Usarsim: a robot simulator for research and education. In *ICRA*, pages 1400–1405. IEEE, 2007.

[3] Howie Choset, Kevin M Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Intelligent Robotics and Autonomous Agents. Mit Press, 2005.

[4] Heikki Hyyti. Sensing through foliage, April 2012.

[5] Niels Out. Virtual radar sensor for usarsim, June 2010. Bachelor Thesis, University Of Amsterdam.

[6] T. Peynot, S. Scheding, and S. Terho. The Marulan Data Sets: Multi-Sensor Perception in Natural Environment with Challenging Conditions. *International Journal of Robotics Research*, 29(13):1602–1607, November 2010.

[7] R. Rouveure, M.O. Monod, and P. Faure. High resolution mapping of the environment with a ground-based radar imager. In *Radar Conference - Surveillance for a Safer World, 2009. RADAR. International*, pages 1 –6, oct. 2009.

[8] M.I. Skolnik. Introduction to radar systems, 1980.