Delft University of Technology

Internship DoBots
ME51015

Internship Report

Author:
Tijmen van Enckevort - 4552660

Due: November 12 2021

Contents

1 Introduction

A major problem when designing and programming autonomous robots is that it is very difficult for the engineer to know what the robot is 'thinking' [1] [2] [3]. This 'thinking' of the robot can best be defined as the way the robot is sensing its environment, and the way it is planning to act upon that environment. Take for example a robotic cart that uses a lidar sensor to scan its environment, and then has to plan a path to reach a certain goal destination. Without any tools it is very difficult for the engineer to know what is going on inside of the robot, as it is an object that is not able to convey its intent. The robot is only able to show its actions. This is a problem because then the engineer can only see that things are going wrong, but it is not able to know why things are going wrong. When he does know why the robot has taken a wrong decision it will be much easier to fix the underlying problem. In the robotic cart example, when the robot cannot detect a certain obstacle it might try to drive through it. This can have several causes, and only when the right cause is solved will the robot be able to work properly. Even more so, the engineer can also think something else than the cause is at fault, and try to fix it while it works correctly, making it even worse.

One possible way of solving this issue is by visualizing all the incoming sensor data and the decisions the robot is making. This will create a very intuitive way for the engineer to see what the robot is 'thinking', and will make it easier for the engineer to pinpoint which parts of the sensing and decision making of the robot are causing the adverse actions. Being able to quickly and accurately discern the problem at fault will make the time required to design, build, and debug the robot a lot lower [4] [5]. This is important as it can decrease the cost of robotic development and accelerate advancements in the robotic field.

The visualization of the robotic sensor data will be the topic of this internship project. Translating the raw sensor data into intuitive 2D or 3D visualizations can be done in multiple approaches of which many are already widely used and available. These tools and techniques will be researched and their relevance and usability will be checked and recommendations will be made on which tools and visualizations to use in which application. This internship project will take place at DoBots.

DoBots is a small scale company in Rotterdam, and part of the bigger investment company Almende. DoBots focuses on web-based robot simulations, which allow engineers to run the simulations of their robots without the inconvenience of installing a version of Linux and ROS on their computer, and with the added benefits of easy collaboration with multiple engineers or teams. Furthermore, DoBots is also occupied in two European research projects, Comp4Drones and Adacorsa. Comp4Drones is a project about drones in clustered environments, where DoBots is responsible for the simulation. DoBots is also responsible for the simulation in Adacorsa, which is a project about large swarms of drones in busy cities. My role at DoBots will be to research possible solutions to visualize the robot sensor data, and how to incorporate that in the simulation environments. The goal of this is to create a more extensive and intuitive simulation environment, which should make it more straightforward for engineers to design and program their robots.

On a more personal note, there are also some more learning goals that I hope to obtain. One major learning goal will be to get to know the Robot Operating System, also known as ROS [6]. It is a widely used software in robotic programming and will be very useful to know for my future studies and professional career. ROS works with both Python and C++ and although my level is sufficient for both, it will be useful to get a deeper understanding of them, and also use them outside of educational assignments but instead in more real-life use cases. Besides learning ROS it will also be informative to get to work with a real physical robot instead of simulations, because as of yet I have not done so in my studies, and I personally learn a lot from the practical implementations of robotics. Next to this, I would also like to get a better understanding of the full set of algorithms needed to run a robot. In courses there was usually only a focus on a very specific part of the See-Think-Act Cycle [7] and I think I will learn a lot from getting to know the more high-level overview and get a more thorough understanding as to how the systems and algorithms are working together.

Lastly, I would like to obtain a better understanding in the way companies operate. I would like to get to know what they think is and is not important, how cooperating with colleagues is different from working together in student projects, and how I can be the most valuable to a company. It will also be valuable to get to know the company's structure, how it gains its revenue, and how it plans ahead for future endeavours and possible products for market gaps. The last point is especially relevant for DoBots, because at the moment it is still a startup company that is planning to launch its new web-based simulation environment Asimovo during my internship, and it will be very interesting to get a deeper insight on this aspect. At the moment this is my first and only internship, and thus I have not got a lot of experience in the professional world.

This internship report is structured in the following way. Several approaches were done to visualize the robot sensor data, and these will be explained in detail in Chapter 2. Visualizations in Rviz will be explained in Section 2.1, and visualizations in the simulation environment Gazebo will be explained in Section 2.2. Section 2.3 will describe the use of the Iviz application and the functionalities it has with Augmented Reality. The gained insights about Virtual Reality can be found in Section 2.4. Lastly, photogrammetry is explained in Section 2.5. Each section can be read stand-alone, and is structured as follows. Each section starts with an introduction on the topic, followed by a subsection on the methodology of how to use the technique. Hereafter the results are presented followed by a discussion on these results. After this, the approaches as well as the personal learning goals will be discussed in Chapter 3, which is followed by a conclusion in 4.

This report was written for the internship course (ME51015) in the specialization BioRobotics of the track BioMechanical Design of the Mechanical Engineering Master's program at Delft University of Technology.

2 Methodology & Results

To visualize the incoming robot sensor data it is necessary to get a better understanding as to how the robot is working, and how it is programmed. The Robot Operating System is a widely used software for robotics and is also used by both DoBots and some courses at the Delft University of Technology. The simulation environment that is used by DoBots is the open source simulator Gazebo [8]. In order to get a better insight into the way a robot is operated it will be necessary to get a better understanding of both ROS and Gazebo.

To get this insight, a course on ROS and Gazebo that is made available by DoBots is done. The course starts with basic prerequisites, for example installing Ubuntu and ROS on a computer and the Linux terminal. It continues with the general ROS structure, with tutorials on ROS nodes, topics, messages, publishers, subscribers, and services. It also contains tutorials on Unified Robot Description Format (URDF), on simulation environments in the Gazebo simulator, and SSH connection to a real robot. Next to this, sensors like the camera, laserscanner, and Inertial Measurement Unit (IMU) are covered. Lastly it has tutorials on perception, Simultaneous Localization and Mapping (SLAM), and navigation algorithms that are used for the autonomous operation of the robot. For this it used standard ROS packages like Hector SLAM, gmapping, and the ROS Navigation Stack. After completion of the course a good understanding of the methods used by a robot is obtained which can be used further to visualize the incoming sensor data.

The course uses simulations based on the TurtleBot3 Waffle, but as DoBots did not have this available but instead uses a Husarion Rosbot robot it was decided to use a simulation with the Husarion Rosbot robot instead [9] [10]. It should also be noted that the course is not fully up to date anymore, and uses some deprecated and discontinued packages and commands that are not supported anymore in the latest versions of ROS and Ubuntu, which are ROS Noetic and Ubuntu 20.04 at the time of writing. It is advised to search for any errors on the internet or ask the available supervisor about it, as it will usually yield a quick solution.

There are multiple possible approaches to visualizing sensor data, of which several will be explained in this Chapter. First, Rviz will be explained in 2.1 [11]. The Rviz package is a 3D visualization tool for ROS, and it is widely used by many engineers. As such, it is a good starting point in visualizing sensor data. Next visualizations inside of the Gazebo simulator are explained in Section 2.2. They are useful as they will be similar to augmented reality, as the simulation environment is visible as well. In Section 2.3 the Iviz application will be explained, which is a Unity based application that can be run on mobile phones and on computers, and can display incoming sensor data similar to Rviz [12]. It also has Augmented Reality (AR) functionalities that can use the phone's camera to drive a robot model around on for example a desk or on the floor. An approach to visualizing sensor data in Virtual Reality (VR) is described in Section 2.4. Unfortunately the only available VR glasses at DoBots are the Oculus Quest glasses, which are not very suitable for developing apps onto and connections to ROS or Gazebo [13]. In order to create a more realistic simulation environment a technique called photogrammetry is explained in Section 2.5 [14]. Photogrammetry maps the 2D capture of photographs back into the original 3D model, and can thus be used to create 3D models of real objects, which can improve the perception of the simulation environment.

The Github repo of the internship project can be found online [15].

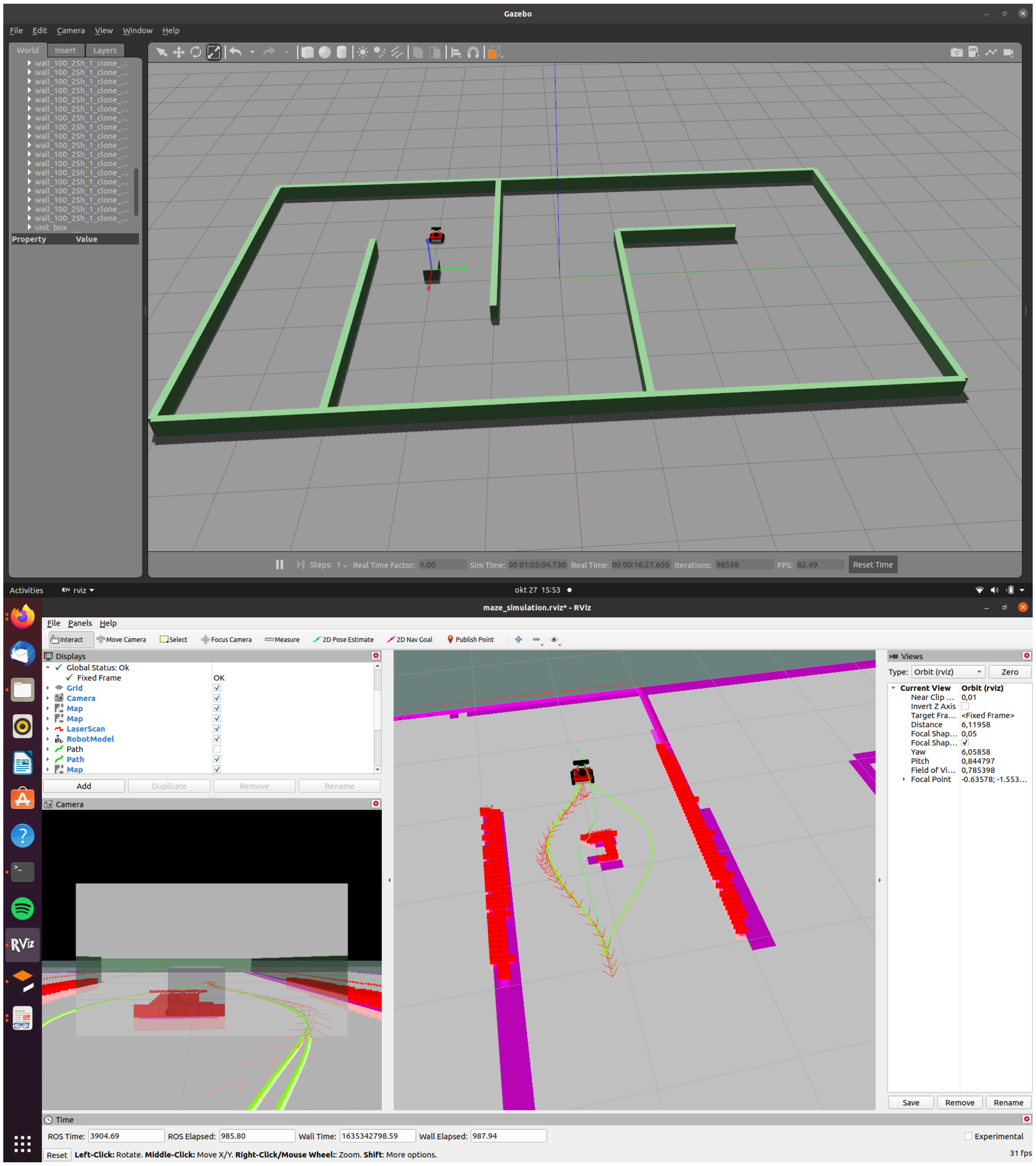
2.1 Rviz

The ROS package Rviz is a 3D visualisation tool. It is widely used and can be easily used to visualize the incoming sensor data. The Rviz tool can subscribe to a multitude of topics published by the robot. These topics can range from raw sensor data, like the IMU, camera, or laserscan topics, but also include topics that are published by the SLAM or navigation algorithms. These include topics like the local costmap of the SLAM algorithm or the global path that is planned by the robot to obtain its goal position.

As Rviz can only visualize topics that are published by the robot or the simulation, it is important to know the setup that is used. The Rosbot robot and robot model is equipped with 4 types of sensors of which 2 can easily be used to visualize in Rviz, namely, the camera topic published by the Kinect camera, and the laserscan topic published by the lidar. The robot then uses the gmapping SLAM algorithm to localize itself in its environment [16]. The gmapping algorithm uses the laserscan sensor information. The Time Elastic Band (TEB) local planner from the movebase package is used for the local path planning of the robot [17]. The reason for this is that it is a time optimal planner, and also allows for backward driving of the Rosbot. Furthermore, it is very convenient as it publishes a lot of topics that are able to be visualized in Rviz. An overview of all the published topics by the robot and its relevance and interpretation can be found in Table 1. To overlay the camera display with a visualization of other sensor data topics it must be made sure to check the <visualize> tag in Rviz.

The results of the Rviz visualization can be seen in Figure 1. In the upper part of the figure the simulation environment in Gazebo can be seen with a maze world and an obstacle. The Rosbot is instructed to go to a goal position that is behind the obstacle. It can be seen that the Rosbot perceives all the walls and obstacles correctly and is able to plan local paths around the obstacle. It can also be seen that the Rosbot is using its TEB local planner to plan the most time optimal path, and discard any suboptimal paths. This is also clearly illustrated in Figure 2 and in this video.

The Rviz tool is straightforward and simple to use, which makes it one of the go-to applications whenever the sensor data of the robot has to be visualized. It can display a wide range of incoming sensor data, making it suitable for most robots. As can be seen from Figure 1 and 2 it clearly displays the processes going on in the robot, and it is able to effortlessly translate what the robot is 'thinking' to the engineer. Whenever a robot is performing unexpected behaviour or whenever a clear presentation of the robot is needed it is recommended to use the Rviz tool.



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 01:05:04.730 Real Time: 00 00:16:27.650 Iterations: 98598 FPS: 62.49 Reset Time

Type: Orbit (rviz) Zero

Current View	Orbit (rviz)
Near Clip ...	0,01
Invert Z Axis	☐
Target Fra...	<Fixed Frame>
Distance	6,11958
Focal Shap...	0,05
Focal Shap...	✓
Yaw	6,05858
Pitch	0,844797
Field of Vi...	0,785398
Focal Point	-0.63578; -1.553...

ROS Time: 3904.69 ROS Elapsed: 985.80 Wall Time: 1635342798.59 Wall Elapsed: 987.94

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options.

Figure 1: Overview of the Rosbot Robot in the simulator Gazebo (upper) and the sensor data visualization in Rviz (lower).

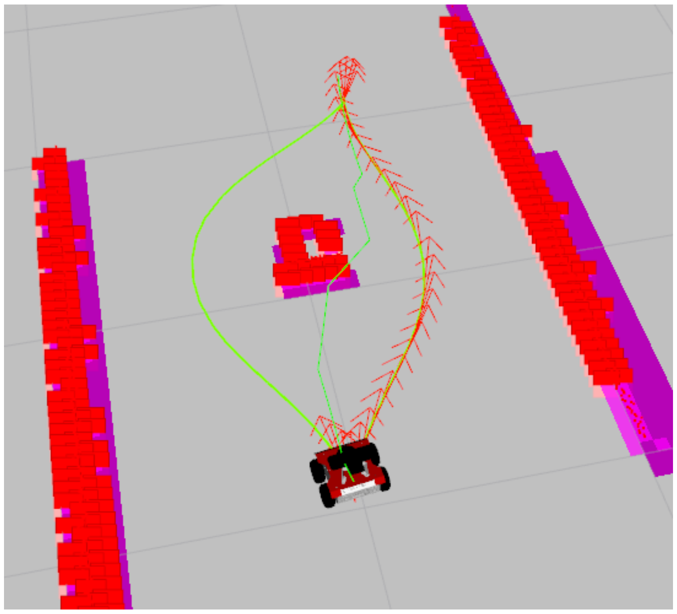


Figure 2: Visualization of two local paths around an obstacle.

Name and topic	Description	Relevance
Camera /camera/rgb/image_raw	An image stream of the camera in the left lower corner	The camera is able to be see the objects like a human does with its eyes and gives a good overview of the environment the robot is driving in. This is particularly useful when driving around on a real robot, where it might not be easy to get a good overview of the environment which is more easy to do in a simulator like Gazebo.
Map /map	A map of the explored environment. Free space is gray and obstacles are black.	This map is the outcome of the SLAM algorithm and is relevant because it pictures the areas the robot has not yet explored yet. It also displays difference in free space and occupied space and can thus display if the robot has perceived an obstacle.
Local Costmap /move_base/local_cost map/costmap	A local map that where all the local occupied space is displayed in a purple color	It is relevant because it shows the obstacles the robot is able to perceive and is taking into account in its local path planning. It can explain why or why not the robot is colliding with its environment
Laserscan /scan	Red dots that are the endpoints of the lidar scanner	This is a visualization of the raw sensor data of the lidar and displays which objects the robot is able to perceive. It also uses this in its gmapping SLAM algorithm
Robotmodel URDF files	A 3D colored model of the Rosbot robot	This can be used to get an idea of the size and orientation of the robot
Global path /move_base/TebLocal PlannerROS/global_plan	A blue line that depicts the global path the robot is planning to take	The global path is the path the robot is planning to take using the explored map. It shows how the robot is planning to move around all of the known obstacles
Global costmap /move_base/global_cost map/costmap	A global map that where all the local occupied space is displayed in a purple color	It is relevant because it shows the obstacles the robot is able to perceive and is taking into account in its global path planning. It can show more high level navigation decisions the robot is taking as it does or does not take certain obstacles into account.
Marker /move_base/TebLocal PlannerROS/teb_markers	These green line markers display the possible local paths the robot is taking into account when optimizing for the time optimal path	These markers show the possible options the robot is taking into consideration when computing the time optimal path. They can show why or why not the robot is taking a certain path, for example if it did not take it into consideration, or if it computes that another path is more time optimal.
PoseArray /move_base/TebLocal PlannerROS/teb_poses	A pose array with red arrows that display the chosen local path and in which pose the robot is planning to follow the path.	This displays the local path that is most time optimal, and is the actual path the robot is trying to follow. It can be used to see if the robot is navigating around obstacles correctly, and why it is preferring certain paths around obstacles over other paths

Table 1: The possible visualizations of the topics published by the Rosbot robot or simulation

2.2 Gazebo

Gazebo is an open source simulator with a high-performance and robust physics engine, making it especially useful for robotic simulations that are used to develop and build real-life robots. As such, it is also used by DoBots and many other organizations occupied with robots. In Gazebo it is possible for a simulated robot to operate in a simulated 3D environment. An example of this can be seen in the upper part of Figure 1, where a model of the Rosbot robot is driving around in a maze-like environment. The visual rendering of the Gazebo simulation is similar to how a robot would look like in a real situation. This makes it that any visualizations of incoming sensor data in Gazebo are parallel to how augmented reality would look like in a real situation. It is different to how visualizations of sensor data looks like in Rviz, as in Rviz it is not possible to see the simulation environment directly. This can make it valuable for the engineer to visualize the incoming sensor data in Gazebo directly.

Gazebo does not provide a lot of straightforward and ready-to-use options for visualizations and that is why it is chosen to visualize the two most important sensor data for the Rosbot setup. The gmapping algorithm that is used by the Rosbot is constructing its localization and mapping algorithm on the incoming lidar sensor data, which make the laserscan a valuable visualization to display. The second visualization that is valuable is the local path the robot is planning to take, as this is the path that can portray the process of the TEB Local planner and can show why or why not the robot is driving around obstacles.

The laserscanner data from the lidar sensor can be displayed relatively straightforward. In the URDF files of the robot there is a `rosbot.gazebo` file. In this file there are several Gazebo plugins, including the RpLidar A2 plugin named `head_rplidar_sensor`. This plugin has a <visualize> tag, which, if set to true, will make Gazebo visualize the laserscanner sensor data. The result of this can be seen in Figure 3.

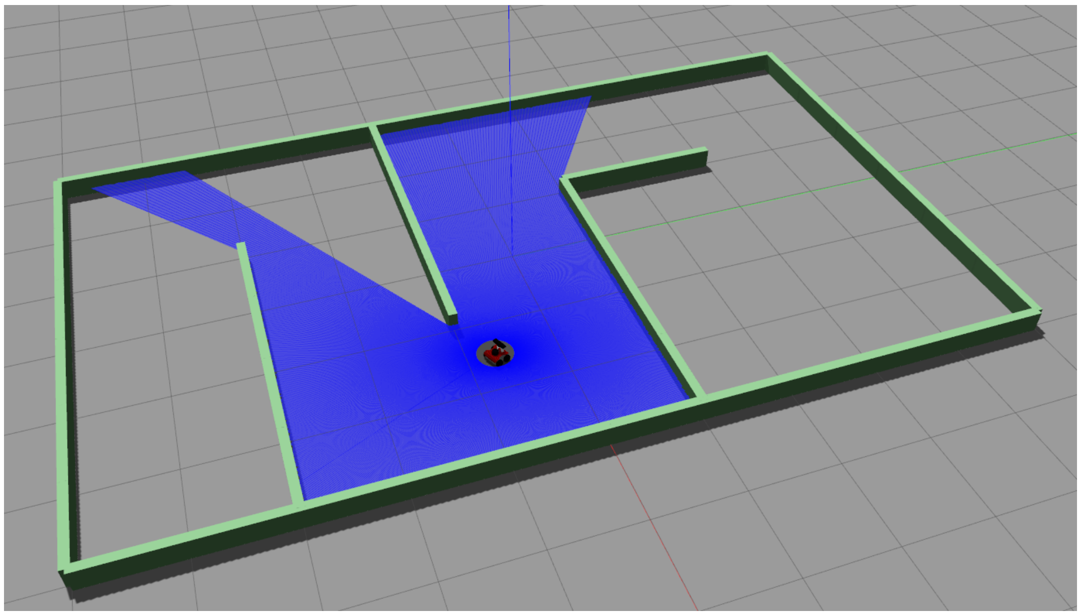


Figure 3: Visualization of the lidar sensor data in Gazebo

The visualization of the path of the robot is less straightforward and cannot directly be visualized by Gazebo. It is however possible to create marker messages that display a marker in Gazebo. One of these possible marker messages is the triangle list, and this is the type of message that is used in the `gazebo_visualize_path` node. This node subscribes to the PoseArray messages that are published by the TEB local planner. For each of the poses in the message it then creates a triangle in the form of an arrow along the pose and adds it to the triangle list. This triangle list is published as an Ignition message, and this makes that Gazebo displays a line of arrows along the local path of the robot. The result of this can be seen in Figure 4. The code for this visualization can be found in the Github repo [15]. To run in first run the simulation with `roslaunch rosbot_gazebo husarion_simulation.launch` and then run a separate node with `rosrun gazebo_visualize_path`.

It should be noted that the use of this node can be computationally expensive. The size of the array in the PoseArray message is varying over time and therefor it is needed to delete and create a new triangle list on every update in Gazebo. This is not preferred, as it is less computationally expensive to initialize the triangle list once, and then modify the locations of the triangle list to the updated locations of the poses of the local path.

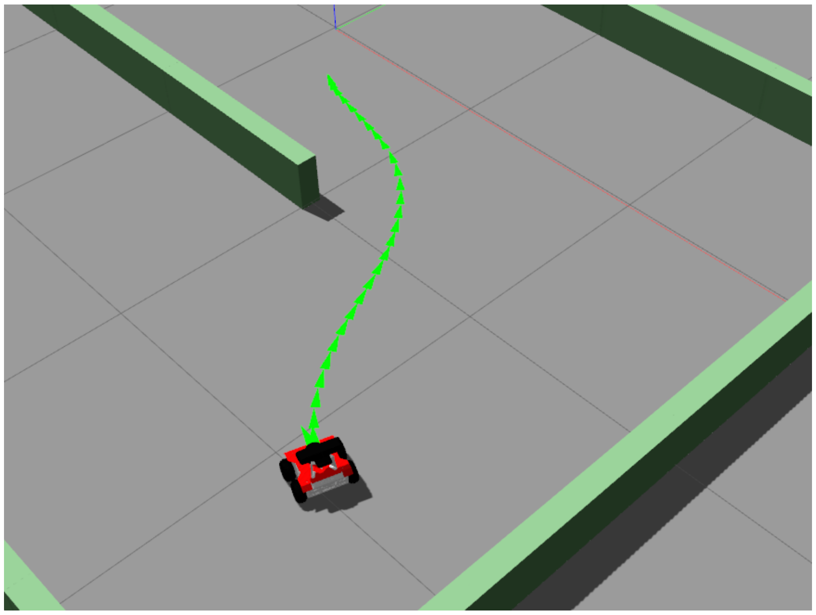


Figure 4: Visualization of the local path in Gazebo

Both the laserscan and local path visualizations in Gazebo are able to visualize the incoming sensor data of the robot, and can bridge the gap between what the robot is 'thinking' and the engineer. The major differences between these visualizations and those done in Rviz are that visualizing in Gazebo directly allows for the engineer to see the simulated environment instead of just the perception of the environment by the robot in Rviz. This makes it analogous to an augmented reality setting with a real robot, and can give extra information about the simulated environment that would otherwise not be possible. As explained above, it should be noted that the visualizations in Gazebo can be computationally expensive, which is not preferable as it can slow down simulation time. This is especially the case in simulations where multiple robots are used, or when there is insufficient computational power.

It is also possible in Gazebo to create a projector to display an image directly on the simulated environment. This can be seen in Figure 5. This projector was deemed very useful to display any decisions or path the robot is taking, but as it is not able to display images that are changing at runtime, or videos, it was discarded. It is possible to display static images however, so if it is needed to display for example the DoBots logo in the Figure 5 this is a possibility. The projector is built using the `libgazebo_ros_projector.so` plugin and can be found in the URDF file `rosbot_gazebo`. The image to be displayed should be added to the */usr/share/gazebo-11/media/materials/textures* folder.

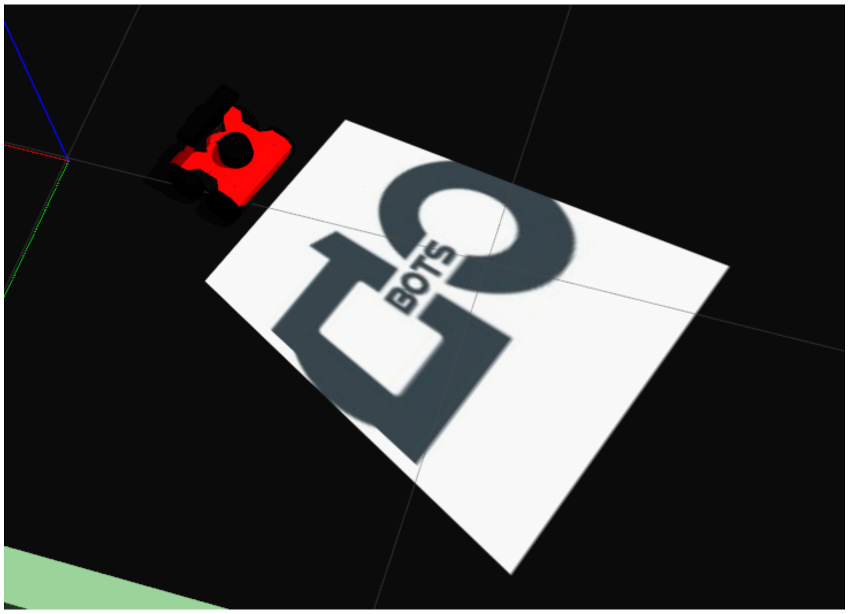


Figure 5: Visualization of the projector in Gazebo

2.3 Iviz

Another approach to visualizing incoming sensor data is Iviz. Iviz a mobile 3D visualization application that is based on the Unity Engine [18]. It can be run both on a computer in the Unity3D platform, and as a standalone application for mobile devices like mobile phones. This makes it a very powerful tool for quick visualizations, especially whenever it is cumbersome to carry a computer around. Another major asset of the Iviz application is its ability to build the robot in an augmented reality setting. This is very useful to get a good understanding of the robot as one is able to walk around it in a real environment. If the simulation of the robot is physically correct and adheres to the laws of physics, it is similar to operating a real robot and can thus replace one in certain situations.

To run the Iviz application one can download the .apk file on the Github page [19], which can then be installed on Android devices. On a computer one has to run the Iviz scene on the Unity3D platform. The Iviz application can connect to the ROS master by connecting to the URL of the ROS master via a Wifi connection. Once connected, it is possible to subscribe to multiple topics, which will then be displayed. This layout is very similar to Rviz, and has mostly the same functionalities. The visualization of the sensor data is straightforward in Iviz by subscribing to the published topics, but the robot model visualization is not as convenient. The URDF files and the accompanying meshes of the robot have to be uploaded to the mobile device. This can be done via the *Modelservice* as explained on the Iviz Github page. A point of attention are the meshes. The standard meshes of the Rosbot robot are `.stl` files, which are not supported by Unity. Instead one has to convert the `.stl` meshes into `.dae` meshes. This can be done in Blender, by importing the `.stl` and converting it to a `.dae` file [20].

As said before Iviz is also able to display the robot in an Augmented Reality. This can be selected via the Augmented Reality module. Once active, it will use the available camera to find a ground plane where the robot can drive onto. On here, it is able to visualize the sensor data from the simulation onto the real environment. An example of this can be seen in Figure 6 and Figure 7. This creates an immersive visualization and is able to display the sensor data from all angles in a real environment. It should be noted that in Figures 6 and 7 the tape on the floor is meant to resemble the walls in the simulation environment found in the upper part of Figure 1, but the alignment is not perfect, which creates the gaps between the visualization of the laserscanner and the tape.

The basic Iviz application without an active augmented reality module is an useful tool for quick visualizations in situations where the use of a laptop is cumbersome. Take for example a situation where it is desired to walk alongside the robot, or in an outdoor testing setting. This makes it especially useful in testing and developing on real robots, and although it is still suitable for testing in simulation environments, it does not outperform tools like Rviz regarding visualization purposes.

The augmented reality module is a very useful asset of Iviz. It does not allow for quick prototyping, as it needs to calibrate the ground plane and tune the settings upon each use. It does however create an immmersive visualization, and can also bridge the gap between simulation environments and real environments by visualizing simulated sensor data onto the real world, like in Figure 6 and 7. It suits well in situations where the simulation is working correctly and an immersive visualization is desirable. From [12] it is also an useful tool to control real robots from mobile devices, but this has not been tested.

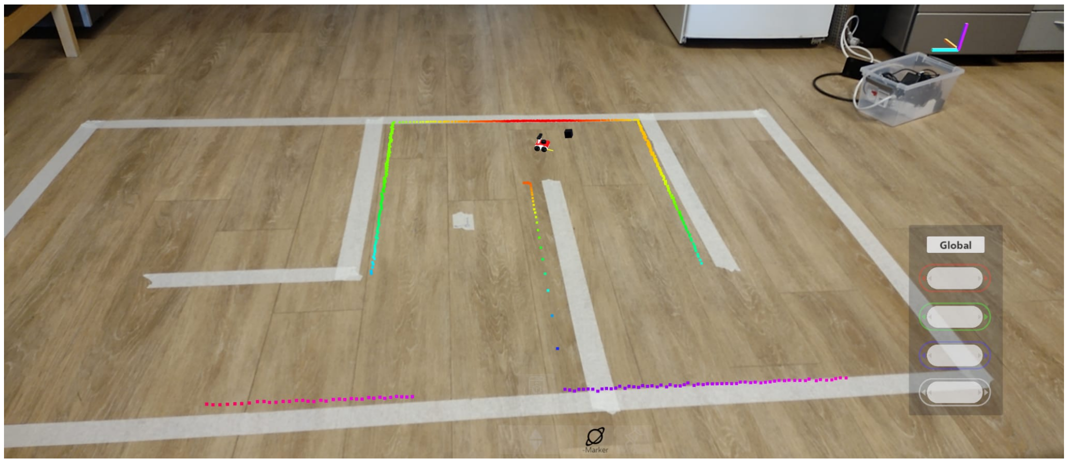


Figure 6: An overview picture of the Augmented Reality in Iviz

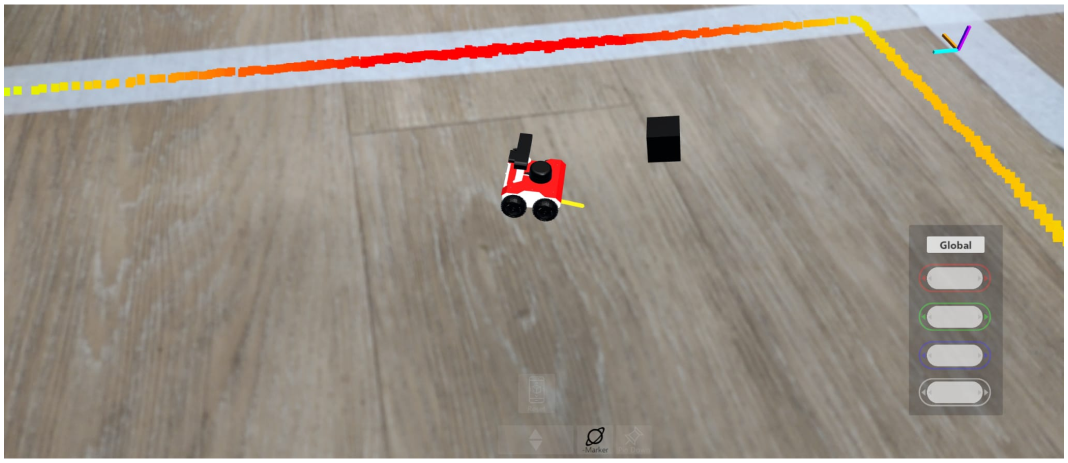


Figure 7: A close up picture of the Augmented Reality in Iviz

2.4 Virtual Reality

An approach to visualize the incoming sensor data in a Virtual Reality setting has also been attempted. This creates, just like the Augmented Reality, an immersive visualization that would be effective at conveying all the functionalities of a robot without the need of the real robot. This would make a good fit in situations where robots are developed in simulations, or where the real robot is not yet available. However, all the attempts to obtain this visualization have not succeeded.

The available VR glasses at DoBots are the Oculus Quest glasses. These glasses are not open source and developing for them is not straightforward and not recommended. Via a Sideloading application it is possible to push .apk files to the glasses. This is only possible when the Oculus Quest is in developer mode. Halfway through the internship there was a bug in the Oculus software which would not allow users to set their Oculus Quest to developer mode. This was not solvable and only after contacting the Oculus helpdesk it was found that it was needed to use the Oculus Developer Hub to connect to the Oculus Quest, which was not the case before.

The most promising route to a Virtual Reality visualization was thought to be via the Iviz application. As this application is able to build on Android phones using the .apk file, this should also be possible on the Oculus Quest. The .apk file that can be found on the Iviz Github page does sideload to the Oculus Quest, but it only works as a 2D screen inside the virtual environment and thus does provide more functionalities than using it on a mobile device. When attempting to start the augmented reality module the application crashes as the Google Play Services do not support it. The Iviz tool in Unity3D has not only got an AR scene, but also a VR scene. An attempt was done to build this scene into an Android application, but unfortunately all of these builds crashed on the Oculus Quest. It could be possible that this feature is implemented in the future for the Iviz application so future readers should check the Iviz Github page for any updates.

In order to overcome this problem it is recommended to use VR glasses that can function together with Gazebo or are open-source. An example of VR glasses that can connect to Gazebo are the Oculus Rift glasses, and although they are discontinued it is recommended to buy a pair of these via unofficial sources [21]. It is also possible to use HTC Vive glasses, which have more capabilities to work with Linux and ROS packages directly [22].

2.5 Photogrammetry

It is desirable to create more realistic simulation environments or simulation environments that are similar to the environments the robot will be used in as a product. These realistic environments can enhance recognition of the engineer and potential unexpected behaviour can be recognised at an earlier stage [23]. It can also lower tuning time when the robot is deployed into its final working environment [24]. These realistic environments can be build by developing realistic 3D models of the obstacles and the environment, but these can take a lot of time to build. Therefor a method to quickly build these 3D objects is needed.

A fast and straightforward way of building the 3D objects is with the use of photogrammetry. Where a photograph is mapping a 3D environment onto a 2D image, photogrammetry is a technique that can do the reverse. It uses multiple images of the object, from different angles and locations. These images can be uploaded to a photogrammetry software. In this report the open source software Meshroom was used [25]. This runs the images through a pipeline, a series of different computations and the outcome is a textured 3D mesh of the original object. An example of this can be seen in Figure 8 and 10.

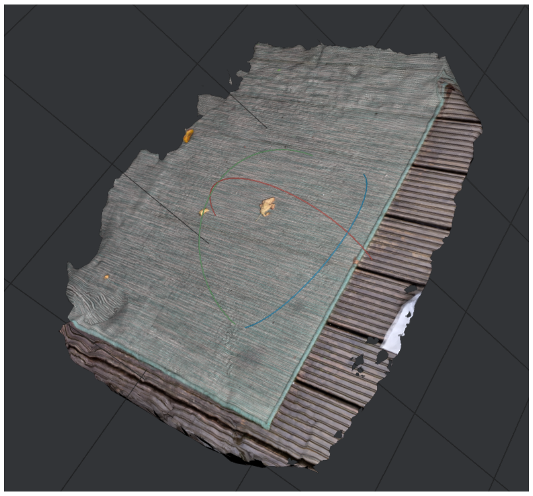


Figure 8: A 3D model of roof terrace

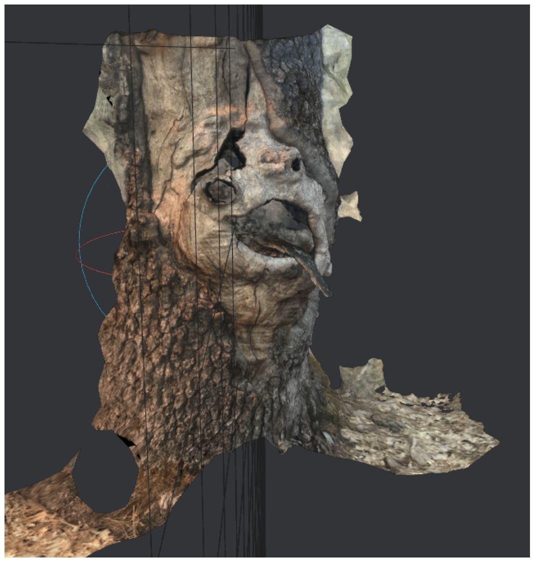


Figure 9: A 3D model of a tree

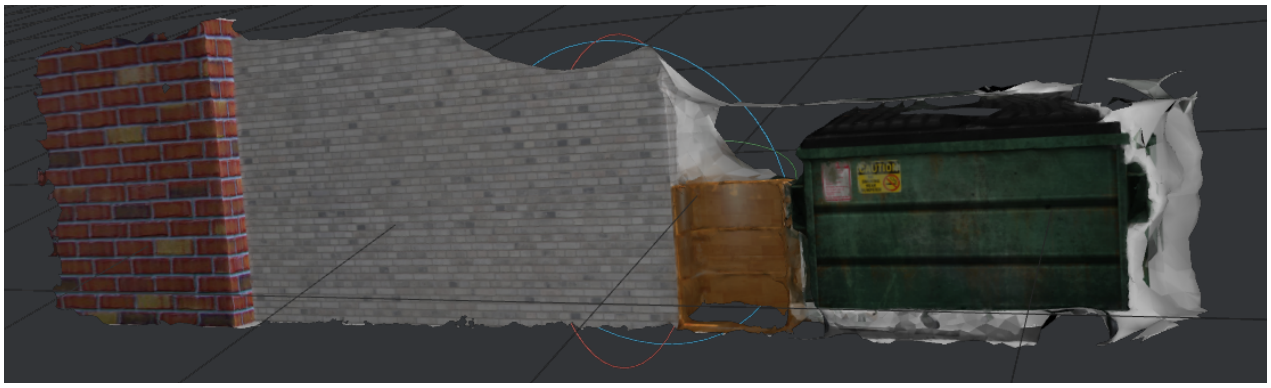


Figure 10: A 3D model of objects in Gazebo

As can be seen from the Figures 8 and 10 photogrammetry is far from a perfect recreation of the original 3D object. It is especially vulnerable to locations on the 3D object that are a not photographed from different locations and angles, like the corners and edges of the objects. On a tested and verified dataset however this effect is not as prevalent, as can be seen in Figure 9. It is a drawback to the technique however, and it is recommended to research the optimal set of images that can recreate the original 3D model without a lot of imperfections.

Another disadvantage to the technique can be the computational time required to build the 3D models. The models of Figure 8 and Figure 9 can take upwards of 1 hour, which is substantial considering the low amount of images, 24 and 41 respectively. The 3D model of Figure 10 also takes roughly 1 hour, but has more images (87). This can be explained by the fact that these images were of a lower resolution, which is also something to take into account when making the image dataset. These computations do not require any attention from the engineer while running the computations, which still make it a convenient alternative to creating 3D models conventional ways.

3 Discussion

Several approaches have been attempted to visualize incoming sensor data in robots. These visualizations can be used by an engineer to get a better idea of what the robot is 'thinking'. With this, the engineer is able to pinpoint bugs and wrong decisions that the robot is making more quickly, which will allow the engineer to develop and build the robot in less time, reducing costs. The advantages and disadvantages and findings of the approaches can be found in detail in their respective Sections in Chapter 2.

In general it is recommended to use Rviz for visualization purposes. It is fast, easy to use and can visualize almost all of the sensor data topics published by the robot in an intuitive way. Whenever it is required to visualize the sensor data of the robot directly in the simulation environment so that the obstacles of the environment are visible as well this can be done in Gazebo directly. It is more computationally expensive and less straightforward to implement the visualizations. The report shows two options of visualizing the laserscan data and the local path, which resemble the most important perception and navigation choices by the robot in the used setup. In a setting where a visualization on a mobile device is desired, for example outdoor tests or tests where one walks alongside the robot it is recommended to use the Iviz application. It allows for visualizations similar to Rviz, but on a mobile device. Next to this it also has Augmented Reality functionalities which can be useful to display the operating of the robot in an immersive way. It is not recommended to use the Augmented Reality when developing or debugging the robot, as it needs calibration and tuning upon startup. The Virtual Reality approach was not tested as it was not possible to run a virtual environment of the robot on the available Oculus Quest glasses.

To create a more realistic simulation environment, it is possible to use the photogrammetry software Meshroom to create 3D models by meshing images of the object. Special note should be taken to the computational time of the software, and it is recommended to do more research on the optimal angles and locations of the set of images, as the photogrammetry can be vulnerable to underrepresented parts of the object.

3.1 Personal Learning Experiences

During the internship I have obtained new insights in robotics, deepened my knowledge on programming and took my first step into the professional world. These experiences are outlined below.

First and foremost this internship has been my first contact with a real physical robot during my studies of BioRobotics. This can be partly explained by the online education due to Covid, and on the courses that are mostly based on more theoretical background than the physical robot. For me personally it gives a drive to see the algorithms that you have built in a simulation being performed on a real robot, and I think it is a very valuable experience for an engineer to see the effects and consequences of its decisions in practice.

I have also gained a deeper knowledge on ROS, and especially the full cycle of algorithms that are needed to autonomously operate a robot. During my courses I have touched upon ROS but the tutorials that were given to me by DoBots have really put the algorithms more in perspective, and also made clear how certain frameworks in ROS are the way they are. As the ROS platform is widely used in academia and in robotic engineering companies it is very useful to have a more thorough understanding of ROS.

During my bachelor mechanical engineering and minor computer science I had already learned the basics of Python and during the course Robot Software Practicals I had brief introduction to C++. During the internship this knowledge has deepened, and I also learned that ROS is able to work both with Python and C++, and that they are able to communicate with each other without problems, which was new to me. The knowledge of programming languages is crucial for not only robotics engineers but for any type of engineering as the calculations done by them are shifting more towards numerical calculations and analysis. I am content with the fact that I am much more comfortable with both languages now, because they will be important in robotic programming which I hope to do in the future as an engineer as well. Doing the programming at an internship has also thought me that the problems to solve tend to be similar, but the way to solve them can be different. Where in educational assignments there was usually a setup which would require you to program the algorithm in a much more structured way, going step by step and always being able to consult the given books or sources or asking peers this was not directly the case at the internship. Here the structure was not defined for me and had to be thought of by myself, and the sources of information were not as obvious as in assignments. I think this is a valuable new insight and lesson as this will probably not be the case for the rest of my career.

The simulations of the robot that had to be made were also a valuable lesson. During my bachelor I have done projects were the mechanical design could be tested in CAD software, but these usually did not consist of more than seeing of the parts fit together, and doing some finite element analysis on the parts to see if they are able to withstand the forces applied. The robotic simulation was a new experience as it can truly simulate the whole robot in a chosen environment, and the perception, planning and navigation algorithms can be tested on the robot. These simulations have also proved me the point of the need for them, as they allow for much faster prototyping and testing, and can easily show the operation of the robot without the need of turning on the robot and placing it in a real testing environment.

I have also reflected back on the See-Think-Act cycle that was thought to me in courses, and I think I am now more able to understand a robot from a more high level overview. This helps me put certain design decisions into place and hopefully make me a more rounded robotics engineer. Doing this, I have also found that this is maybe the topic that is more interesting to me. Instead of going very deep into an algorithm to try and improve it with a novel finding, which is usually the basis of the courses, my interest has shifted towards a higher level view of the robot where every algorithm has to robustly cooperate in order for the robot to function properly. As of yet, I do not fully know what kind of role I can play as an engineer if I wish to pursue this, but I hope that my upcoming thesis project is able to give me more knowledge on this aspect.

Insights have also been gained on the difference between communication and planning with students and with colleagues. I have found there is a noticeable difference in planning with students, who usually prefer to do the bulk of the workload when the deadline is nearing, and then do a lot of work in the evenings and on weekends. I am also very guilty of doing this. In the professional setting this can be very different as people tend to do their work spread out over the project, and might not always be available in their free time. This is of course very logical but still it was a valuable lesson for me.

Being at the office of DoBots has also gained me more insights into the way companies are run, and how a start up company is planning to launch its new products. During my internship DoBots has been developing a new web-based simulation environment, and it was very interesting to see how they tackled the problem of launching this, and in which ways they were hoping to gain a competitive advantage.

4 Conclusion

Whenever developing and building robots that are able to sense their environment and act upon it is hard for engineers to know what is going on inside of the robot. This translation gap between the engineer and the robot can lead to longer developing and building processes as the engineer will not be able to pinpoint exactly where the robot is at fault, and which algorithms and design choices need further investigation to solve the issue. This internship report aims to close this gap by means of several visualization approaches of the incoming sensor data and navigation methods of the robot. The three recommended approaches to use are Rviz, visualization in Gazebo, and Iviz. Rviz is useful for fast testing and prototyping. Visualizations in Gazebo are useful whenever a visualization on top of the simulated environment is required, and Iviz is useful when the simulation of the robot is working properly, but an immersive Augmented Reality visualisation is desired. These approaches can visualize the incoming sensor data and navigation methods, and can be used by engineers to have a better understanding as to what is going on inside of the robot, which can speed up developing and building processes of robots.

References

[1] M. C. Thorstensen, "Visualization of robotic sensor data with augmented reality," Master's thesis, 2017.

[2] D. Brutzman, "Virtual world visualization for an autonomous underwater vehicle," in *'Challenges of Our Changing Global Environment'. Conference Proceedings. OCEANS'95 MTS/IEEE*, vol. 3. IEEE, 1995, pp. 1592–1600.

[3] V. Gupta, R. G. Chittawadigi, and S. K. Saha, "Roboanalyzer: robot visualization software for robot technicians," in *Proceedings of the Advances in Robotics*, 2017, pp. 1–5.

[4] A. Dietrich, M. Schulze, S. Zug, and J. Kaiser, "Visualization of robot's awareness and perception," in *Proceedings of the First International Workshop on Digital Engineering*, 2010, pp. 38–44.

[5] M. Campusano and J. Fabry, "From robots to humans: Visualizations for robot sensor data," in *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*. IEEE, 2015, pp. 135–139.

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[7] M. Wahde, "Introduction to autonomous robots," *Lecture Notes from the course Autonomous Agents, Chalmers university of technology*, 2012.

[8] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[9] Husarion. Autonomous robots made simple. [Online]. Available: https://husarion.com/

[10] Robotis. Turtlebot3. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

[11] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: a toolkit for real domain data visualization," *Telecommunication Systems*, vol. 60, no. 2, pp. 337–345, 2015.

[12] A. Zea and U. D. Hanebeck, "iviz: A ros visualization app for mobile devices," *Software Impacts*, vol. 8, p. 100057, 2021.

[13] Oculus. Explore oculus quest. [Online]. Available: https://www.oculus.com/quest/features/

[14] T. Schenk, "Introduction to photogrammetry," *The Ohio State University, Columbus*, vol. 106, 2005.

[15] T. van Enckevort. Dobots internship. [Online]. Available: https://github.com/dobots/tijmen_internship

[16] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2432–2437.

[17] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[18] Unity. Unity. [Online]. Available: https://unity.com/

[19] A. Zea and U. D. Hanebeck. Iviz. [Online]. Available: https://github.com/KIT-ISAS/iviz

[20] Blender. The freedom to create. [Online]. Available: https://www.blender.org/

[21] gazebosim. Oculus rift. [Online]. Available: https://gazebosim.org/tutorials?tut=oculus&cat=rendering

[22] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, "Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[23] J. Faust, C. Simon, and W. D. Smart, "A video game-based mobile robot simulation environment," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3749–3754.

[24] C. Paulo, G. José, L. José, and M. Paulo, "Simtwo realistic simulator: A tool for the development and validation of robot software," *Theory and Applications of Mathematics & Computer Science*, vol. 1, no. 1, pp. 17–33, 2011.

[25] C. Griwodz, S. Gasparini, L. Calvet, P. Gurdjos, F. Castan, B. Maujean, G. De Lillo, and Y. Lanthony, "Alice-vision meshroom: An open-source 3d reconstruction pipeline," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 241–247.