Title of the Paper

Student Name and Number
Mechanical Engineering Department, University College London

ABSTRACT

Autonomous mobile robots are vital in areas such as logistics and healthcare, where they rely on complex navigation systems to effectively manoeuvre in dynamic indoor environments. Conventional 2D LiDAR systems, while favourable for obstacle detection in a single plane. However, they perform poorly in recognising low-lying and complex obstacles, limiting their operational efficiency and safety. This project proposes a novel sensor fusion method that combines the 2D breadth of LiDAR with the stereo vision data from a depth camera. By converting depth camera outputs into LiDAR messages, the robot-accessible environmental data is enriched, resulting in significantly enhanced obstacle detection in multiple dimensions. This fusion technique significantly closes the gap in 2D LiDAR obstacle detection capabilities. This sensor fusion algorithm's is implemented based on enhancements to existing algorithms in the ROS2 framework, simulated and evaluated in Gazebo and RViz2. The enhanced algorithm was combined with the fusion algorithm of this project to consider and improve the problems that can arise when multiple fusions are performed. The results show a significant improvement in the robot's ability to detect and bypass direct and low-lying obstacles.

Keywords: Mobile Robot, Depth Camera, Laser Scan, 2D Mapping, Localisation, Mapping, Navigation

NOMENCLATURE

α alpha

1 INTRODUCTION

The emergence and popularity of indoor mobile robots has dramatically altered various fields such as logistics, healthcare, and home services, allowing for the automation of navigation tasks in indoor environments. Among them, Autonomous mobile robots are used in a variety of fields to improve services and daily activities, including households, hospitals, businesses, institutions, agriculture, and industry [1].

Indoor autonomous robots rely heavily on advanced navigation systems that can safely navigate complex and changing indoor environments. One important feature of such systems is their ability to detect and avoid obstacles, which is critical for operational efficiency and safety. These robots navigate autonomously, determining their position in the environment and plotting a collision-free path to their destination. It must be essential for these robots to gather information from their surroundings and perceive objects or relative positions around them. Perception is an important aspect of mobile robotics research. Tasks like accurately estimating the position of an object may become difficult if the mobile robot is unable to observe its surroundings correctly and efficiently [1]. This process necessitates complex interactions among sensor inputs, data processing, and actuation mechanisms [11].

Among the sensors used for this purpose, LiDAR stands out for its ability to accurately measure distance and detect obstacles. LiDAR is an acronym for "light detection and ranging", also known as "Laser Scanning Radar". Despite these benefits, LiDAR has inherent limitations in its detection capabilities. These limitations include, in particular, the high cost of 3D LiDAR and its massive computational requirements, as well as 2D LiDAR's detection dimension limitations.

The 2D LiDAR system is often installed on top of the robot, allowing it to detect obstacles in the horizontal plane of its laser beams. However, it is difficult to detect obstacles that do not intersect this plane, particularly those near the ground (e.g., chair supports or lamp bases). This detection capability gap highlights the need for 2D LiDAR navigation systems to recognise obstacles in multiple dimensions. To address the challenge of detecting low-lying obstacles,

this project presents a LiDAR and camera system by converting depth camera data into a LiDAR message (LaserScan Message). This method combines the 3D visual information provided by the camera with the 360-degree measurements provided by the LiDAR, resulting in a synergistic effect and increased environmental awareness. The camera captures detailed images of the environment, including objects below the LiDAR detection plane, whereas the LiDAR aids spatial understanding by accurately mapping obstacles at greater distances and over a larger area. ROS2 includes the software package "Depthimage_to_Laserscan" [4]. However, the algorithm is configured to use the depth camera as a single sensor for obstacle avoidance information.

As a result, adapting the principal from this package to convert depth information from the camera into LiDAR information and then integrating it with LiDAR data via a series of algorithms, a comprehensive environment model can be created. This method significantly improves the robot's ability to detect and avoid direct and low-lying obstacles, increasing the safety and efficiency of indoor navigation. The LiDAR-camera fusion approach resolves a critical sensor selection conflict, closes the detection capability gap in 2D LiDAR systems. Also, it improves the autonomy and efficiency of indoor robots, advancing the field of autonomous robot navigation.

2 LITERATURE REVIEW

2.1 LiDAR

The sensing system for automated vehicle navigation is made up of active and passive sensors, such as cameras, radar, and lidar [13]. One such device LDAR, is an advanced active sensor system that uses lasers to illuminate and detect its surroundings. These devices measure distances to various objects precisely by emitting lasers and receiving them back from reflective surfaces. In a typical LiDAR system, one or more laser beams are scanned across its field of view. The generation and control of these laser beams is dependent on a sophisticated beam control system. Lasers are typically created using modulated laser diodes that emit light at near-infrared (NIR) wavelengths. The laser beams are then reflected back by objects in the environment, and the resulting signals are captured by photodetectors on the scanner. Fast electronics in the lidar system filter these returned signals and calculate the time difference between the emitted and received laser light. This time difference is the key to calculating the distance because it is proportional to it. This time difference enables the system to precisely estimate the distance between the laser and the object. Furthermore, advanced signal processing techniques can compensate for variations in reflected intensity caused by differences in surface materials or environmental conditions between the transmitter and receiver. LiDAR typically produces a point cloud of data that depicts the structure of the scanned environment and includes information on each point's location as well as the intensity of the reflected laser energy. This makes LiDAR extremely useful for environmental modelling and object recognition. LiDAR ranging is capable of working in two modes: direct detection and coherent detection. Direct detection laser rangefinders are devices that use a pulsed laser to directly measure distance by measuring the laser's time of flight (ToF). The other type of device, known as a coherent detection laser rangefinder, uses a Frequency Modulated Continuous Wave (FMCW) laser to indirectly measure an object's distance and speed using the Doppler effect [20].

The LiDAR system used in this project is a typical kind of ToF LiDAR, whose working principal is shown in Figure 1.

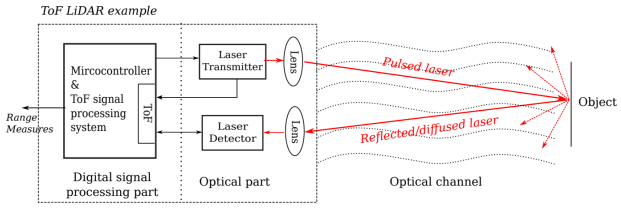

Figure 1: ToF LiDAR Example, adapted from [14]

.

2.2 Depth Camera

Common depth cameras, such as the Intel RealSense Camera, use three lenses to assess depth: a camera, an infrared (IR) camera, and an IR projector. They detect infrared light reflected from the object or body in front of them. The resulting visual data is combined with software to generate a depth estimate, which is then returned as a depth image [23].

Figure 2: Intel RealSense Camera

Different depth cameras achieve depth values through various techniques and mechanisms. Stereo vision systems use two spaced apart cameras to capture images from slightly different angles, similar to human binocular vision [12]. The system can calculate depth by comparing these two images and using the difference between corresponding points in the images. Structured light technology is the process of projecting a pattern of light (usually infrared) onto a scene and photographing it. The depth values are calculated by analysing the distortion of the light pattern caused by the surfaces in the scene [7]. The ToF camera emits pulses of light (usually infrared) and measures the time it takes for the light to travel to the object and return to the camera [8]. This is very similar to the principle of section 2.1 outlined above.

2.3 SLAM

The availability of accurate maps enables system designs to operate in complex environments using only on-board sensors, rather than external reference systems (e.g., GPS). Over the last few decades, the robotics community has focused heavily on developing maps of indoor environments, often without the use of GPS [5]. The simultaneous localization and map building (SLAM) problem is commonly used to describe the process of learning maps under attitude uncertainty [2]. It is a technique that allows robots to map unknown environments while also localising themselves based on the maps created, often in the absence of an external positioning system [3].

SLAM combines data from various sensors (e.g., cameras, LiDAR, inertial measurement units) to create a consistent map of the environment and calculate the robot's position in relation to that map. The process consists of two major steps: localization (estimating the robot's position in the environment) and mapping (building a model or map of the environment's layout).

The SLAM algorithm is heavily reliant on odometry data, which measures changes in position over time. In this project, detailed odometry information is obtained by tracking the rotation of each wheel using a differential wheel. Differential steering provides the mobility and manoeuvrability required to navigate complex environments, and when combined with SLAM, it allows for precise navigation and obstacle avoidance in dynamic settings [6]. Also, using only differential wheels as SLAM odometers amplifies the importance of obstacle avoidance for robots, as they are very sensitive to collisions. This is mentioned in following section 4.

2.4 ROS2 Humble

ROS 2 Humble Hawksbill is a version of the Robotics Operating System (ROS 2). It is the successor to the original ROS (Robotics Operating System) and is intended to improve on its predecessor's functionality, focusing on new robotics use cases, more robust communications, and support for real-time systems [17]. ROS2 Humble is used throughout this project for the following reasons: ROS 2 Humble is an LTS release, which means it will be supported and updated for a longer period of time. And when this project started, this version was the latest LTS version. Whilst it may be easier to find solutions within the ROS community when the project encounters problems using an earlier version, using a newer platform means that the project is more in line with the ongoing developments in the field of robotics, and thus easier to integrate with future technologies and standards emerging in the community. This is also a mission of this project.

2.5 Gazebo

Gazebo is a popular open-source robotics simulator with a powerful physics engine, high-quality graphics, and a wide range of sensors and objects that can be customised to simulate real-world scenarios. Furthermore, Gazebo includes a large library of sensor models and plug-ins for simulating cameras, LiDAR, and other common robotic sensors. The simulator is completely integrated with the Robotics Operating System (ROS), making it an excellent tool for developing and testing ROS-based applications in a simulated environment [10].

This project uses the Gazebo simulation world throughout to run, test and correct the algorithms. This reduces the risks and costs associated with physical prototyping. And algorithms and robot interactions can be tested in a variety of scenarios without causing expensive hardware damage. Through simulation, it is possible to quickly iterate on algorithm design, test changes, and see the results immediately. This rapid feedback loop significantly accelerates the development process.

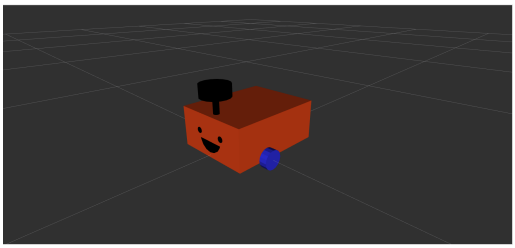

Figure 3: Simulated Indoor Environment in Gazebo

2.6 RViz2

RViz is a powerful 3D visualisation tool for representing sensor data, robot state, and behaviour. RViz2's displays can be customised to meet specific requirements, allowing it to adapt to a wide range of robotic applications [21]. It is also fully integrated with ROS2 and compatible with Gazebo.
RViz2 is used throughout this project to visualise data including robot models, trajectories, and sensor data streams (e.g. laser scans and cameras). It provides visual insights into what the robot is "seeing" and "thinking", which is essential for tuning algorithms and ensuring they perform as expected.

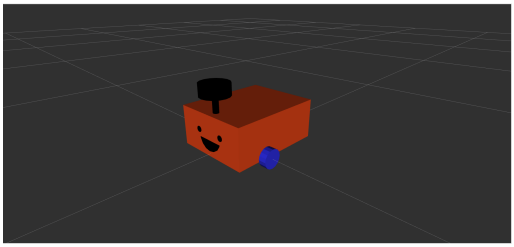

Figure 4: The Robot Model and LaserScan Data visualised in RViz2

2.7 Depthimage_to_Laserscan

Depthimage_to_Laserscan is a ROS package that provides a basic solution for converting depth images from depth cameras to 2D laser scan data [4].
The principal of algorithm for converting depth image data to LaserScan data in the following section is adapted from this package. The details are explained in more detail below.

2.8 Navigation2

Because the primary goal of this project is to address the shortcomings of 2D LiDAR, it is important to employ an auto-navigation method that is well-suited to a wide range of sensors, simple to use, and provides reliable performance. The Navigation2 framework is a cutting-edge, open-source navigation stack that includes a comprehensive set of tools and libraries for enabling autonomous navigation on a variety of robotic platforms, particularly in complex and dynamic environments [16].
Nav2 is user-friendly, with plug-and-play components that make it easier to set up and configure advanced navigation systems in robots. It offers a variety of navigational functions, such as path planning, obstacle avoidance, and map management. Its modular design makes it extremely scalable [18]. In this project, the navigation function of Navigation2 is used for simple robot navigation. The robot is only given a target position and then can plan an obstacle avoidance route through the given map.

2.9 Slam_toolbox

As mentioned in section 2.3, SLAM is a central part of robot navigation. SLAM_toolbox is a package for ROS. It provides tools to perform SLAM using various sensor data sources, both on real robots and in the simulated world [15].

This package is used throughout this project in conjunction with the other tools mentioned above to implement robot navigation and map building.

2.10 Proposed Solutions

In the early stages of the project, two unique proposed solutions were also proposed and they are:

2.10.1 YOLO-V8

YOLO (You Only Look Once) is a deep learning model. It can detect objects, especially a wide variety of obstacles, with high accuracy and speed. Its superior efficiency enables robots to make fast, informed decisions about obstacle avoidance and path planning [19].

However, whilst YOLOv8 provides powerful target detection capabilities, it relies heavily on visual data from the camera. The aim of this project was to explore sensor fusion, in particular the integration of camera and LiDAR data, to take advantage of the complementary strengths of the two sensing modalities. YOLOv8 focuses only on camera input, which is not in line with the sensor fusion goals of this project.

2.10.2 ORB-SLAM3

ORB-SLAM (Oriented FAST and Rotated BRIEF SLAM) is a versatile and accurate visual SLAM method that utilises a feature-based approach for simultaneous localisation and mapping. It can construct detailed environmental maps and accurately locate vehicles in environments not recognised by GPS.

While ORB-SLAM is somewhat able to address the problem of map-building crashes due to collisions for robots using only differential wheel odometers (the reason is explained in following section 4), it is not primarily designed for dynamic obstacle detection and avoidance. Its main goal is to create the map of the environment, rather than consistently recognising and bypassing transient obstacles. This is at odds with the goals of the project.

3 THEORY AND CALCULATION

3.1 Depth Image to Laser Scan

In this section, the object distance information from the depth image is converted into laser scanning information (sensor_msgs/msg/LaserScan) [22], where the information that is important for sensor fusion is:

- **angle_increment**: The angle between each measurement, which can be used to determine the exact direction of each distance measurement within the scan.
- **ranges**: A float32 array representing the distance measurement from the scanner to the nearest object for each angle increment. In Gazebo, the LiDAR application is visualised as a circle (or sector) made up of equal parts of laser data slots within a given range (usually 360 degrees). Two-dimensional laser scanning is typically defined in the plane using polar coordinates, each of which is represented by a distance r and an angle φ.

By default, the centre row of the depth image is used to extract the data that is converted to laser scanning information. This maximises the maximum range at which objects can be detected. This will be mentioned in the next section: "Trade off in maximum range and object depth". Since the goal of this project is to address the disadvantage of LiDAR in detecting low-lying objects, the lower part of the depth image is the main focus of the process. Therefore, how to convert the coordinates and distance information of any row in the depth image to world coordinates so that it can be fused with LaserScan information is the key to this part.

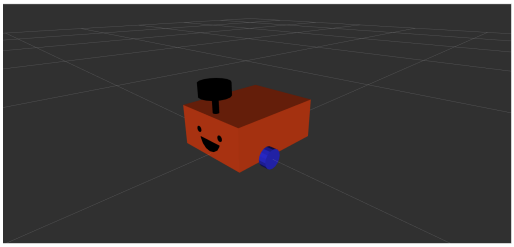

Figure 5: LiDAR Visuliaztion in Gazebo

A common 2D LiDAR has a resolution of 666, which means that 666 pieces of data will be collected over a 360-degree area. A common depth camera produces an image with a resolution of 1920×1080 and a field of view of 666 degrees, which means that in a selected row, about 666 pixels will be acquired over a 666-degree field of view. Therefore, there is no need to worry that the LiDAR data converted from the depth camera data will be limited in the amount of information.

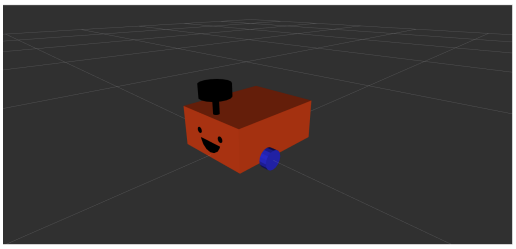

Figure 6: Comparison of Depth and Normal Image

In a depth image, the colour of each pixel point represents the distance of this point in real-world coordinates from the camera's optical centre. Different depth cameras have different ways of colour coding as shown in the figure 7 below.

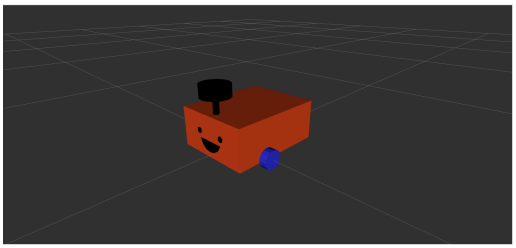

Figure 7: Different Depth Image Encoding Methods

In this project, depth-gray is chosen as a way to encode distance information in depth images. The distance information is expressed in terms of intensity by mapping the depth values to different pixel intensities. The further away the object is, the lighter the shadow will be. Also, since the resolution of the depth image is known (1920×1080 used in this project), the coordinates of any point in the image can be easily represented. Thus, from the depth map it is known: 1. the coordinates $p(x, y)$ of the desired point on the image. 2. the distance from the camera origin O_c to the plane where the object is located, Z_c, which is the depth value mentioned above. 3. the focal length, f. These are indicated in following figure 8.

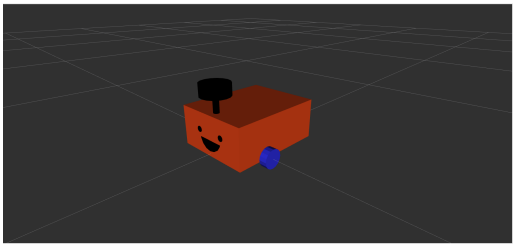

Figure 8: Depth Image Coordinate System

In order to fuse the depth image data with the Laser-

Scan data, the one in the depth image needs to be extracted and processed into the form of laser data slots, i.e. polar coordinates. As shown in figure 8, the camera coordinates (X_c, Y_c, Z_c) of point P and its world coordinates (X_w, Y_w, Z_w) can be found according to the principle of triangle similarity. Then the projection angle of the coordinate point, i.e., the angle AO_cB between the lines AO_c and BO_c, can be calculated as

$$\theta = \arctan\left(\frac{x}{z}\right) \tag{1}$$

The next step is to map the angle AO_cB to the corresponding laser data slot. The minimum and maximum range $[\alpha, \beta]$ of the converted laser information is set by the range of the camera field of view, and this range is subdivided into N laser data slots, represented by the array $Laser[n]$. The index value n of the point P projected to the array $Laser$ can be calculated by Equation 2:

$$n = \frac{\theta - \alpha}{(\beta - \alpha)/N} = N \times \frac{\theta - \alpha}{\beta - \alpha} \tag{2}$$

The value of $Laser[n]$ is the distance r from the point B projected by the point P on the $X_cO_cZ_c$ plane to the centre of the camera's optical center O_c, which can be calculated by Equation 3:

$$Laser[n] = r = O_cB = \sqrt{Z_c^2 + X_c^2} \tag{3}$$

The above calculation of index values can be quickly understood by a simplified example. Assuming that there are $N = 4$ laser data slots in a max range 60 degree and min range -60 degree, the Index of each data slot is shown in following Figure 9.

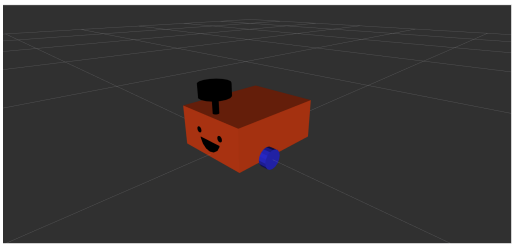

Figure 9: Simplified Example of Laser Data Slot Index

In order to calculate the index of the laser slot at the 30 degree position, there is:

$$n = N \times \frac{\theta - \alpha}{\beta - \alpha} = 4 \times \frac{30° - (-60°)}{60° - (-60°)} = 3$$

And the logic in coding for this part is shown in the following Algorithm 1.

Algorithm 1 Depth image to LaserScan

Require: $depth\_image$

Ensure: A converted LaserScan message from one line of depth image

Variables: scan_time, range_min, range_max, scan_height, output_frame_id

Initialize class variables **with** provided parameters

function MAGNITUDE_OF_RAY(ray)

return $\sqrt{ray.x^2 + ray.y^2 + ray.z^2}$

end function

function ANGLE_BETWEEN_RAYS(ray1, ray2)

dot_product $\leftarrow$ $ray1.x \cdot ray2.x + ray1.y \cdot ray2.y + ray1.z \cdot ray2.z$

magnitude_1 $\leftarrow$ MAGNITUDE_OF_RAY(ray1)

magnitude_2 $\leftarrow$ MAGNITUDE_OF_RAY(ray2)

return $\arccos(\frac{dot\_product}{magnitude1 \cdot magnitude2})$

end function

function USE_POINT(new_value, old_value, range_min, range_max)

new_finite $\leftarrow$ isfinite(new_value)

old_finite $\leftarrow$ isfinite(old_value)

if not new_finite and not old_finite **then**

return not isnan(new_value)

end if

range_check $\leftarrow$ (range_min $\leq$ new_value $\leq$ range_max)

if not range_check **then**

return false

end if

if not old_finite **then**

return true

end if

return (new_value $<$ old_value)

end function

3.2 Tradeoff in Maximum Range and Object Depth

In order to maximise the maximum distance at which objects could be detected, the centre row of the depth image was used to extract the data converted to LaserScan information. This is due to the fact that although it was implemented to extract any row of the depth image to be converted to LaserScan data, the selection of non-centre rows can cause the algorithm to incorrectly consider the floor (or ceiling if taking the top half of the depth image) as an obstacle for long maximum range. This will result in the map not being constructed correctly. The simulation in RViz2 for this particular situation is shown in Figure 10.

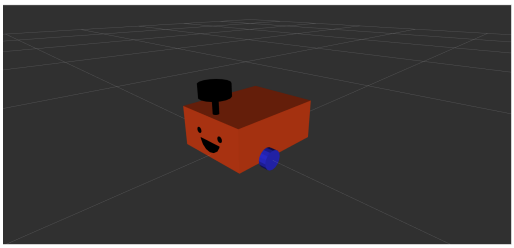

Figure 10: Situation of LiDAR considering the Floor as an Obstacle.

As the aim of this project is to address the disadvantage of LiDAR on low lying objects, the lower the obstacle that can be detected is the better, therefore selecting the centre horizontal line of the depth image and convert it to LaserScan message is not going to maximise the demand. The simplest solution is to place the depth camera on a lower position of the robot, so that even if the centre line in the image is chosen to be extracted, the resulting scanning height is still acceptable. However, since this is a hardware solution to the problem, it is difficult to implement on different, already assembled robots. Algorithm-based approaches are still needed to obtain a generic solution to this problem.

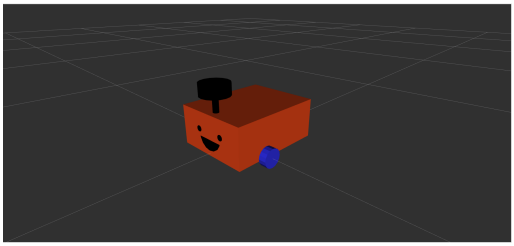

Figure 11: Typical Indoor Robot with Camera on Top

The figure 12 shows how to calculate the maximum distance l from a point when the chosen horizontal line is offset from the original centre line by h, which will be uniformly called to as the "line of sight". When the line of sight meets the ground, the length of the line of sight is the maximum distance, in which case choosing a larger distance would cause the map in the above figure to fail to be constructed. This method therefore ensures that the scan is maximised in a given situation while avoiding the wrong identification. Assuming that there is an obstacle close to the ground, its horizontal height difference from the depth camera's optical center can be seen as equivalent to the distance h from the depth camera's optical center to the ground. The angle α is the angle by which the line of sight is offset from the horizontal line at the centre of the frame to the selected horizontal line immediately above the ground. At this point it can be seen that the maximum distance l of the original horizontal line of sight is too long and crosses the horizon after a clockwise rotation α, i.e., it will lead to an erroneous situation of judging the ground as an obstacle.

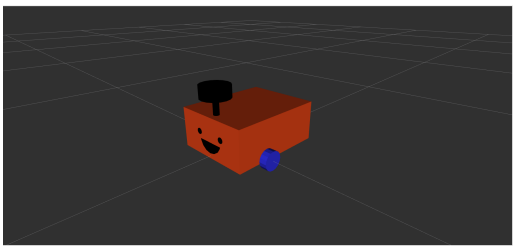

Figure 12: The Tradeoff in Maximum Range and Object Depth

In Figure 13, the maximum distance l in this case is obtained by assuming that after the line of sight is rotated α, it just intersects the ground. The depth camera optical centre is still at a distance h from the ground. Let there be a wall as an obstacle in the horizontal line of sight, the point p is the position scanned by the depth camera in the horizontal line of sight, and the point p' is the position scanned by the depth camera after the line of sight has been rotated α.

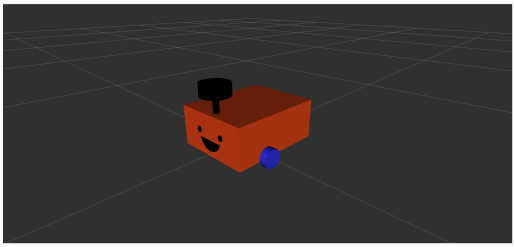

Figure 13: The Tradeoff in Maximum Range and Object Depth

At this point the line of sight is rotated δ further downwards in order to scan at a lower height. Although it is still possible to scan the obstacle point p'' where the wall is lower, the line of sight crosses the horizon, causing an error condition to occur. The length l' of the line of sight that does not cross the horizon needs to be calculated so that the problem can be solved by reducing the maximum distance of the line of sight. The length l' of the line of sight that does not cross the horizon can be calculated by equation 4 below

$$l' = \frac{h}{\sin(\alpha + \beta)} \tag{4}$$

where $\alpha + \beta$ can be expressed as $\arctan(\frac{h'+x}{c})$.
The reduction in scanning height h' caused by the line-of-sight rotation α can be expressed as, $\tan\alpha = \frac{h'}{c}$, since $\sin\alpha = \frac{h}{l}$. The angle α can be expressed as

$\alpha = \arcsin(\frac{h}{l})$, which gives

$$h' = \tan(\arcsin(\frac{h}{l})) \times c \tag{5}$$

Finally, substituting into the original equation yields:

$$l' = \frac{h}{\sin\left(\arctan(\frac{\tan(\arcsin(\frac{h}{l}))\times c + x}{c})\right)} \tag{6}$$

where l' is the new max range after tradeoff. Define it into the programme, it can be seen that the error was fixed successfully.

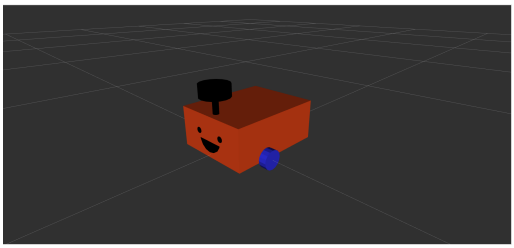

Figure 14: The Same Situation after Fixed

Furthermore, from the representation of the diagonal α above, it can be seen that α can be represented by an inverse $\sin$ wave which is a positively proportional function. It confirms that the more the line of sight is offset downward from the centre horizon, the greater the maximum detection distance sacrificed. Although adjusting the height of the depth camera from the hardware is not recommended, if the depth camera is placed too high, the maximum detection distance sacrificed will be large, thus significantly reducing the effectiveness of this fusion algorithm. Therefore it is necessary to reduce the height of the depth camera even though the algorithm can be adjusted to detect objects in lower lying areas of the field of view.

3.3 Merge of LaserScan data and Depth Image Data

By the above method, the obstacle information within a reasonable range in any row within the depth camera's field of view can be obtained and presented in RViz2 as LaserScan data, as shown in Figure 15 and Figure 16.

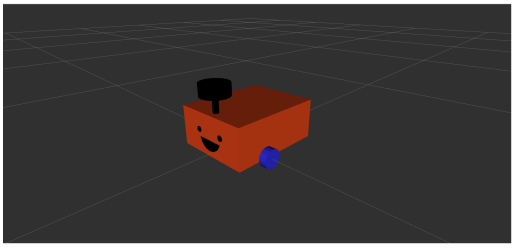

Figure 15: Converted LaserScan Data from Depth Image using the Center Row

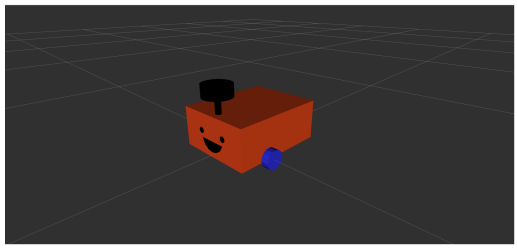

Figure 16: Converted LaserScan Data from Depth Image using the Lower Row

The author of slam_toolbox points out that this package does not support multi-laser configs, so two sets of lidar information must be combined into one. Although the author suggests a solution: use the laser_assembler package [9]. However, this package is not well ported for ROS2 and has problems with not working properly or performing a lot of unnecessary calculations in tests. Therefore, a simple and efficient fusion method is designed for this project. There are two sets of LiDAR data, one with a range of 360 degree and one with a range of $[\alpha, \beta]$ converted from depth images. These two sets of LiDAR data are not at the same height, but this is not important in the 2D map, as shown in Figure 17 and Figure 18.

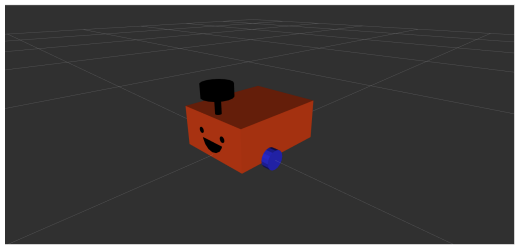

Figure 17: Scan Range of LiDAR

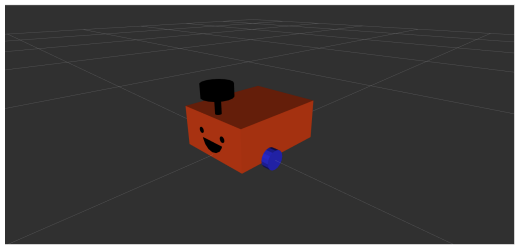

Figure 18: Scan Range of Depth Image

A new dummy LaserScan array is created. And since there is more LaserScan data with a larger range, the $range$ values are compared at the location where the two sets of data overlap, based on the LaserScan resolution. The smaller value, which is the closer obstacle, is entered into the dummy array. The logic of this part is shown in the following Algorithm 2.

Algorithm 2 Merge two LaserScan messages into one

Require: $lidar\_msg$, $depth\_msg$ ▷ LaserScan messages from LiDAR and depth image

Ensure: A merged LaserScan message with the minimum ranges from both inputs

Initialize merged ranges list: $merged\_ranges \leftarrow [\,]$
for $index = 0$ **to** length of $lidar\_msg.ranges$ **do**
  $range_l \leftarrow lidar\_msg.ranges[index]$
  $range_d \leftarrow depth\_msg.ranges[index]$
  Append $\min(range_l, range_d)$ **to** $merged$
end for
Create a new LaserScan message $merged$ **with**
$header \leftarrow$ lidar.header
$angle\_min \leftarrow$ lidar.angle_min
$angle\_max \leftarrow$ lidar.angle_max
$angle\_increment \leftarrow$ lidar.angle_increment
$time\_increment \leftarrow$ lidar.time_increment
$scan\_time \leftarrow$ lidar.scan_time
$min \leftarrow$ min(lidar.range_min, depth.range_min)
$max \leftarrow$ max(lidar.range_max, depth.range_max)
ranges $\leftarrow merged$
return merged

In this way, the algorithm automatically determines the greater range of obstruction for the same obstacle, regardless of whether the obstacle is more obstructed at a lower or higher location. The fusion of the two LaserScan information does not affect the information returned by the original LiDAR.

3.4 Resample of Depth Image Data

However, the fusion of the two LaserScan messages in the previous section is not ideal, as shown in Figure:

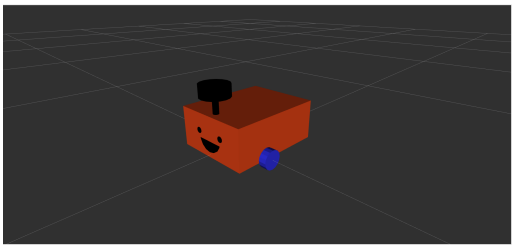

Figure 19: Fusioned LaserScan Data before Resample

This is because the resolution of the LaserScan information converted from LiDAR and depth images is not the same. As there are 1024 pixels per line in a 1920×1080 depth image, all 1024 pixels are used to convert to LaserScan information. However, there are only about 666 LaserScan data in the overlapping portion of the LiDAR LaserScan information, which makes the fusion between the two contradictory.

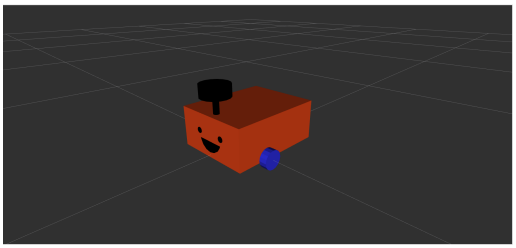

Figure 20: The Contradictory in Resolution

In order to solve this problem without degrading the accuracy of the depth image conversion, the resolution of the two LaserScan's need to be harmonised, based on the data preprocessing in section 3.2 followed by a Resample operation.

Resample the LaserScan data for depth image conversion. First, this method receives two parameters: the original (converted from depth image) LaserScan data, and the LaserScan data of the target (LaserScan information from the LiDAR). These two sets of data contain, but are not limited to, the corresponding angle increments, minimum angle, and maximum angle.

After that, a new LaserScan message is created to store the resampled scan data. The header information, time increments, scan time, minimum and maximum ranges of the new scan data remain the same as the original scan data, while the angular minimum, angular maximum and angular increments are to use the new target values. Interpolation is used to obtain the corresponding $range$ values of the target angles.

Then, the angles of the original and target scan data are stored in two arrays of values generated at specified intervals within the specified range. And then the target angles are interpolated to obtain the corresponding $range$ values. The target angle is linearly interpolated based on the original angle and the $range$ value. If the target angle is out of the range of the original angle, the corresponding $range$ value is set to NaN. finally, the new scan data is returned.

In other words, the new scan data still uses the data converted from the depth image, but only the corresponding amount of data that overlaps with the LaserScan data is retained. At the same time, the $range$ of the scan is changed to that of the LiDAR LaserScan data (360 degrees), except for the portion outside the field of view of the depth image, where $range$ is set to NaN.

The logic of this part is shown in the following Algorithm 3.

Algorithm 3 Resample the converted LaserScan message

Require: $lidar\_msg$, $depth\_msg$ ▷ LaserScan messages from LiDAR and depth image

Ensure: A resampled converted LaserScan message with LiDAR resolution

l_angles ← from lidar.min to lidar.max with step of lidar.increment

d_angles ← from depth.min to depth.max with step of depth.increment

i_ranges ← Interpolate ranges from original_scan to match target_angles

if $range$ **is** out of bounds **then**

$range \leftarrow NAN$

end if

Create resampled_scan **with**

$header$ ← depth.header

$angle\_min$ ← lidar.angle_min

$angle\_max$ ← lidar.angle_max

$angle\_increment$ ← lidar.angle_increment

$time\_increment$ ← depth.time_increment

$scan\_time$ ← depth.scan_time

$range\_min$ ← depth.range_min

$range\_max$ ← depth.range_max

$ranges$ ← i_ranges

return resampled_scan

4 RESULT AND DISCUSSION

With the methodology used above, this project is tested in Gazebo Classic and RViz2 based on ROS2 Humble. The first is a reproduction of the problem to be solved in this project. A differential robotic robot using LiDAR to build a map and avoid obstacles is formulated with a destination point in an indoor environment. However it accidentally collided on the way. The figure below shows that after the collision, the constructed map becomes chaotic.

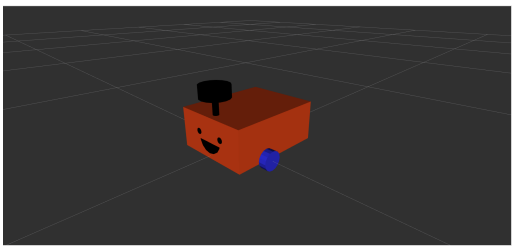

Figure 21: Chaotic map after collision

This is because the robot considers that the robot is still moving during the collision, so the map is updated with the "new position" of the robot. An easy-to-understand example is the skidding that occurs in a moving car on ice. When the car skids, the wheels are still spinning normally, but there is no actual displacement of the car. Two obstacles that are useful to this problem were added to the simulated world to be tested separately:

1. **Bookshelf**: Due to the thin partitions on each level of the bookshelf, the LiDAR scanning surface that is not at the same level as one of the partitions will not detect the partition, but only the baffles on either side. To visualise this more, the back plate of the shelf is removed and the shelf is then placed directly in front of the robot and mapped using only LiDAR. At this point it can be seen in RViz2 that only the baffles appear as obstacles in the generated map. If the robot is given a forward navigation command in this case, it will move straight ahead and hit the bookshelf. As shown in Figure 22. It will not be able to navigate correctly and at the same time it will destroy the original map.

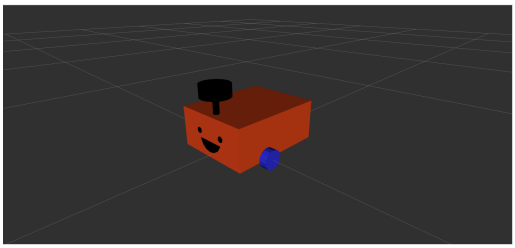

Figure 22: LiDAR cannot detect the shelf correctly.

After using the merged LaserScan data returned by the fusion algorithm, it is clear in RViz2 that the baseboard of the bookshelf is successfully recognised as an obstacle and mapped.

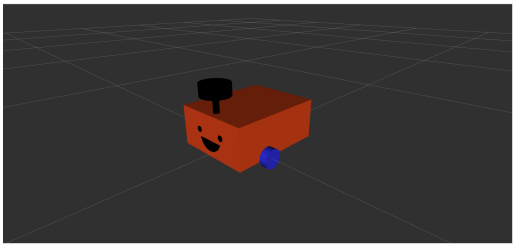

Figure 23: The shelf is detected correctly after the fusion

2. **Barricades**: Another very intuitive test is this cylinder barricade with a polygonal bottom cross-section and a rounded upper cross-section. The different shapes of such barricades at different heights make it very straightforward to see whether low-lying obstacles are detected or not, even without the need to build a map. Both the LiDAR LaserScan data and the merged LaserScan data returned by the fusion algorithm, distinguished by different colours, are turned on in RViz2:

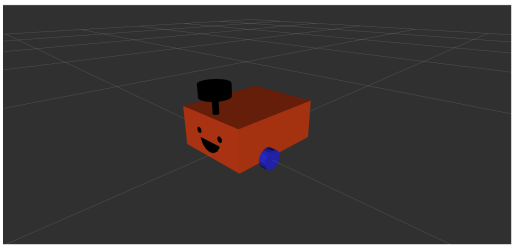

Figure 24: Different LaserScan data for same barricade

It can be seen that what is scanned before the fusion is the higher part of the top half with a circular cross-section. And after the fusion, the scanning of this area is replaced with the lower part of the bottom cross-section with a polygonal shape. This also represents the fulfilment of the objectives of this project.

References

[1] Mary B. Alatise and Gerhard P. Hancke. "A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods". In: *IEEE Access* 8 (2020), pp. 39830–39846. DOI: 10.1109/ACCESS.2020.2975643.

[2] Bashar Alsadik and Samer Karam. "The Simultaneous Localization and Mapping (SLAM)-An Overview". In: *Journal of Applied Science and Technology Trends* 2 (Nov. 2021), pp. 120–131. DOI: 10.38094/jastt204117.

[3] T. Bailey and H. Durrant-Whyte. "Simultaneous localization and mapping (SLAM): part II". In: *IEEE Robotics & Automation Magazine* 13.3 (2006), pp. 108–117. DOI: 10.1109/MRA.2006.1678144.

[4] Chad Rockey. *depthimage_to_laserscan*. https://wiki.ros.org/depthimage_to_laserscan.

[5] Giorgio Grisetti et al. "A Tutorial on Graph-Based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43. DOI: 10.1109/MITS.2010.939925.

[6] Giorgio Grisetti et al. "A tutorial on graph-based SLAM". In: *IEEE Transactions on Intelligent Transportation Systems Magazine* 2 (Dec. 2010), pp. 31–43. DOI: 10.1109/MITS.2010.939925.

[7] Peter Henry et al. "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments". In: *Int. J. Rob. Res.* 31.5 (2012), pp. 647–663. ISSN: 0278-3649. DOI: 10.1177/0278364911434148. URL: https://doi.org/10.1177/0278364911434148.

[8] Radu Horaud et al. "An Overview of Depth Cameras and Range Scanners Based on Time-of-Flight Technologies". In: *CoRR* abs/2012.06772 (2020). arXiv: 2012.06772. URL: https://arxiv.org/abs/2012.06772.

[9] jonbinney. *ROS2 port Issue*. URL: https://github.com/ros-perception/laser_assembler/issues/17.

[10] 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. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[11] David Kortenkamp. "Perception for mobile robot navigation: A survey of the state of the art". In: 1994. URL: https://api.semanticscholar.org/CorpusID:64273728.

[12] Riyad El-laithy, Jidong Huang, and Michael Yeh. "Study on the use of Microsoft Kinect for robotics applications". In: *Record - IEEE PLANS, Position Location and Navigation Symposium* (Apr. 2012), pp. 1280–1288. DOI: 10.1109/PLANS.2012.6236985.

[13] You Li and Javier Ibanez-Guzman. "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems". In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61. DOI: 10.1109/MSP.2020.2973615.

[14] You Li et al. "What Happens for a ToF LiDAR in Fog?" In: *IEEE Transactions on Intelligent Transportation Systems* 22 (2020), pp. 6670–6681. URL: https://api.semanticscholar.org/CorpusID:212725277.

[15] Steve Macenski and Ivona Jambrecic. "SLAM Toolbox: SLAM for the dynamic world". In: *Journal of Open Source Software* 6 (May 2021), p. 2783. DOI: 10.21105/joss.02783.

[16] Steve Macenski et al. *The Marathon 2: A Navigation System*. Feb. 2020.

[17] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. "Exploring the performance of ROS2". In: *Proceedings of the 13th International Conference on Embedded Software*. EMSOFT '16. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2016. ISBN: 9781450344852. DOI: 10.1145/2968478.2968502. URL: https://doi.org/10.1145/2968478.2968502.

[18] *Nav2 — Nav2 1.0.0 documentation*. URL: https://navigation.ros.org/.

[19] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[20] Felix Rohrbach. *An introduction to LiDAR*. Jan. 2020. URL: https://felix.rohrba.ch/en/2015/an-introduction-to-lidar/.

[21] *rviz - ROS Wiki*. URL: http://wiki.ros.org/rviz.

[22] *sensor_msgs/LaserScan Documentation*. URL: https://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/LaserScan.html.

[23] Vladimir Tadic et al. "Application of Intel RealSense Cameras for Depth Image Generation in Robotics". In: *WSEAS Transactions on Computers* 18 (Sept. 2019), pp. 107–112.