Title of the Paper

Student Name and Number
Mechanical Engineering Department, University College London

ABSTRACT

Mobile robots have gained popularity in many application areas over the last few decades. In order to sense the environment in which the robot is moving autonomously, various sensors such as depth cameras are required. Due to the huge amount of data, the computational cost of processing 3D information in depth images is very high. However, for many mobile robots, navigation in a simplified two-dimensional world is sufficient. For this reason, depth images of the environment can be converted into 2D information in the form of laser scan lines and then processed using localisation and mapping algorithms. In this paper, an existing algorithm for converting depth images into laser scans is improved and tested on a real robot in a real-world scenario. Compared to the original algorithm, the improved algorithm takes into account 3D information such as the robot's height and obstacles when creating the laser scan line. This allows the mobile robot to navigate in a simplified 2D world without colliding with obstacles in the real 3D world. Due to the high computational cost of processing 3D information, the algorithm was optimised so that it could be executed on a low-cost single board computer.

Keywords: Mobile Robot, Depth Camera, Laser Scan, 2D Mapping, Localisation, Mapping, Navigation, Single Board Computer, Optimisation

NOMENCLATURE

α	alpha

1 INTRODUCTION

The emergence and popularity of indoor mobile robots has dramatically changed various fields such as logistics, healthcare and domestic services, facilitating the automation of navigation tasks in indoor environments. The use of autonomous mobile robots is emerging in diverse fields such as companies, industry, hospitals, institutions, agriculture and households to improve services and daily activities.

These robots rely heavily on advanced navigation systems that are able to manoeuvre safely in complex and changing indoor spaces. An important aspect of such systems is their ability to detect and avoid obstacles, which is critical to ensuring operational efficiency and safety. Navigation for indoor mobile robots involves autonomously determining the robot's position in the environment and plotting a collision-free path to its destination. It is important for autonomous mobile robots to acquire information from the environment and perceive objects or relative positions around themselves. Perception is an essential aspect of mobile robot research. Performing tasks such as accurately estimating the position of an object may become problematic if the mobile robot is unable to observe its environment correctly and efficiently. This process requires complex interactions between sensor inputs, data processing and actuation mechanisms.

Among the range of sensors used for this purpose, laser radar (LiDAR) stands out for its accuracy in measuring distance and detecting obstacles. However, despite these advantages, LiDAR has inherent limitations in its detection capabilities. These limitations include, in particular, the cost of 3D LiDAR due to its high price and huge arithmetic requirements, and the limitations of 2D LiDAR in terms of detection dimensions.

The 2D LIDAR system is mounted on top of the robot as standard so that obstacles in the horizontal plane of its laser rays can be detected. Obstacles that do not intersect this plane, however, especially those close to the ground (e.g., chair supports or lamp bases) are therefore difficult to detect. This gap in detection capability highlights the need for 2D golden radar navigation systems to be able to recognise obstacles in more dimensions. In order to address the challenge of detecting such low-lying obstacles, this paper proposes the fusion of a LiDAR and camera system by converting information from a depth camera into a LiDAR message (LaserScan Message). This approach utilises the 3D visual information provided by the camera and the 360-degree measurements provided by the LIDAR, creating a synergistic effect and enhancing environmental awareness. The camera captures detailed images of the environment, including objects below the LIDAR detection plane, while the LIDAR facilitates spatial understanding by accurately mapping obstacles at greater distances and over a wider area. The software package ”Depthimage_to_Laserscan” is included in ROS2, however the algorithm is applied to use only the depth camera as a single sensor for obstacle avoidance information.

Therefore, based on this package converting depth information from the camera into LiDAR information and then integrating it with data from the LiDAR through a series of algorithms, a comprehensive environment model can be generated. This approach greatly enhances the robot's ability to detect and bypass direct and low-lying obstacles, thereby improving the safety and efficiency of indoor navigation. The LIDAR-camera fusion approach resolves a key sensor selection conflict, bridges the gap in detection capabilities of 2D LIDAR systems, and improves the autonomy and efficiency of indoor robots, thus advancing the field of autonomous robot navigation.

2 LITERATURE REVIEW

2.1 LiDAR

The sensing system for automated vehicle navigation consists of a combination of active and passive sensors, i.e. camera, radar and lidar. One such device, LIDAR, is an advanced active sensor system that illuminates and detects its surroundings by emitting lasers. Such devices accurately measure distances to various objects 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 relies on a sophisticated beam control system. The lasers are typically generated by modulated laser diodes that emit light at near-infrared (NIR) wavelengths. These laser beams are then reflected back by objects in the environment and the returned signals are captured by photodetectors on the scanner. Fast electronics in the lidar system filter these returned signals and measure the time difference between the emitted laser light and the received laser light. This time difference is the key to calculating the distance, as it is proportional to the distance. This time difference allows the system to accurately estimate the distance between the laser and the object. In addition, any variations in reflected intensity due to differences in surface materials or changes in environmental conditions between the transmitter and receiver can be compensated for by advanced signal processing techniques. The output of LiDAR is typically a point cloud of data that exhibits the structure of the scanned environment and includes information on the location of each point and the intensity of the reflected laser energy. This makes LiDAR extremely effective in environmental modelling and object recognition. LiDAR ranging can work in two main ways: direct detection and coherent detection. Devices that use a pulsed laser directly measure distance by measuring the laser's time of flight (ToF); this type of device is called a direct detection laser rangefinder. The other type of device uses a Frequency Modulated Continuous Wave (FMCW) laser, which indirectly measures the distance and speed of an object through the Doppler effect, and this type of device is known as a coherent detection laser rangefinder.

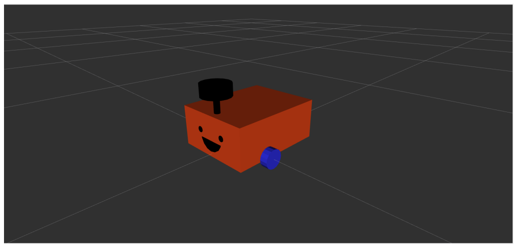

Figure 1: Example figure caption.

2.2 Depth Camera

Common depth cameras such as the Intel RealSense Camera utilise three lenses - a camera, an infrared (IR) camera and an IR projector - to assess depth by detecting infrared light reflected from the object/body in front of it. The resulting visual data is combined with software to create a depth estimate, i.e., a depth image is returned.

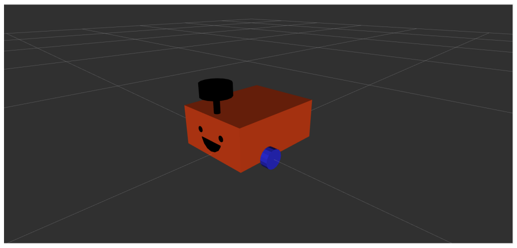

Figure 2: Example figure caption.

Different depth cameras use different techniques and mechanisms to achieve depth values. Stereo vision systems use two spaced apart cameras to capture images from slightly different angles, similar to human binocular vision. By comparing these two images, the system can calculate depth based on the difference between corresponding points in the images. Structured light technology involves projecting a pattern of light (usually infrared) onto a scene and capturing an image of it with a camera. Depth values are calculated by analysing the distortion of the light pattern caused by the surfaces in the scene, while 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. This is very similar to the principle of LIDAR above

2.3 SLAM

The availability of accurate maps allows system designs to operate in complex environments based solely on their on-board sensors, without relying on external reference systems (e.g., GPS). Acquiring maps of indoor environments, often without GPS, has been a major research focus of the robotics community over the past decades. Learning maps under attitude uncertainty is often referred to as the simultaneous localisation and map building (SLAM) problem. It is a technique that enables robots to map unknown environments while simultaneously localising themselves based on the maps drawn, often in the absence of an external positioning system. SLAM combines data from various sensors (e.g., cameras, LIDAR, inertial measurement units) to create a consistent map of the environment and determine the robot's position relative to that map. The process involves two main steps: localisation (estimating the robot's position in the environment) and mapping (constructing a model or map of the environment's layout).The SLAM algorithm relies heavily on odometry data, which involves measuring changes in position over time. In this project, detailed odometry information is provided by tracking the rotation of each wheel through a differential wheel. Differential steering provides the mobility and manoeuvrability required to navigate complex environments, and when used in conjunction with SLAM, enables precise navigation and obstacle avoidance in dynamic settings. Also, using only differential wheels as SLAM odometers can amplify the importance of obstacle avoidance for robots, which are very sensitive to collisions. This will be mentioned in the last part of Test.

2.4 ROS2 Humble

ROS 2 Humble Hawksbill is one of the versions of the Robotics Operating System (ROS 2), a successor to the original ROS (Robotics Operating System) designed to improve on the functionality of its predecessor, with a focus on new use cases in robotics, more robust communications, and support for real-time systems. 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 Nav2

As the main objective of this project is to address the shortcomings of 2D LiDAR, it is important to use an auto-navigation method that is highly adapted to a wide range of sensors, is easy to use and has a robust performance.The Navigation2 framework is a state-of-the-art, open-source navigation stack that provides a comprehensive set of tools and libraries designed to enable autonomous navigation for a wide range of robotic platforms, especially in complex and dynamic environments. Nav2 is user-friendly and offers plug-and-play components that simplify the setup and configuration of advanced navigation systems in robots. It provides a wide range of navigation functions, including path planning, obstacle avoidance and map management. Its modular design makes it highly scalable.

2.6 Gazebo

Gazebo is a well-known open-source robotics simulator that offers a powerful physics engine, high-quality graphics, and a variety of sensors and objects that can be customised to mimic real-world scenarios. Moreover, Gazebo comes with a comprehensive library of sensor models and plug-ins to simulate cameras, LIDAR, and other common robot sensors. The entire simulator is fully integrated with the Robotics Operating System (ROS), making it an ideal tool for developing and testing ROS-based applications in a simulated environment. This project uses the Gazebo simulation world throughout to run, test and practice 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.

2.7 RViz2

RViz is a powerful 3D visualisation tool for visualising sensor data, robot state and behaviour. The displays in RViz2 can be customised to meet different requirements, giving it the flexibility to adapt to all types of robotic applications. RViz2 is also fully integrated with ROS2 and can be used 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.

2.8 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. Part of the code 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.

2.9 Slam_toolbox

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. 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-SLAM

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, it is not primarily designed for dynamic obstacle detection and avoidance. Its focus on creating a static map of the environment, rather than consistently recognising and bypassing transient obstacles, is also 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), 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.
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 1920x1080 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.

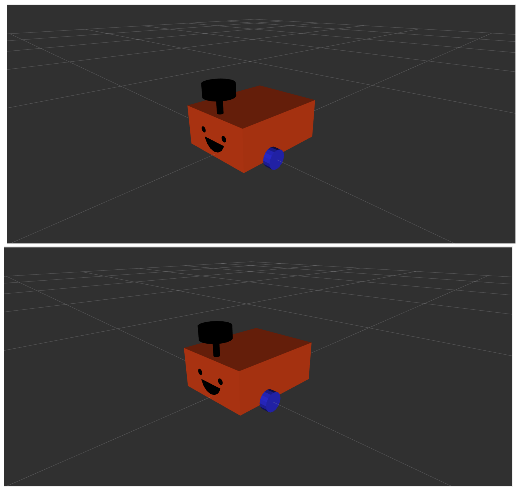

Figure 3: Example figure caption.

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.

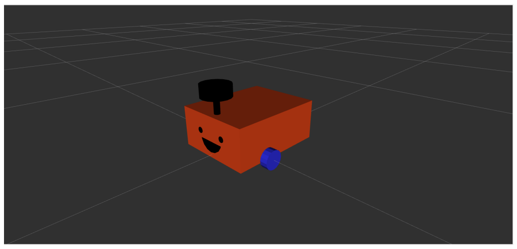

Figure 4: Example figure caption.

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 (1920x1080 is 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. This is indicated by the red, green, and blue lines in the following figure:

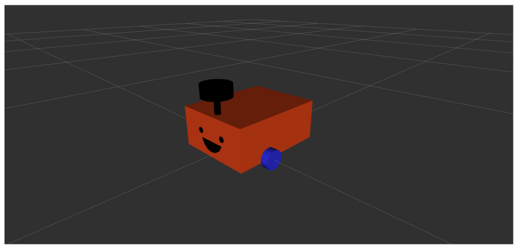

Figure 5: Example figure caption.

In order to fuse the depth image data with the **LaserScan** data, the data in the depth image needs to be extracted and processed into the form of laser data slots, i.e. polar coordinates. As shown in the figure, 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, is 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 the following formula:

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

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 light O_c, which can be calculated by the following formula:

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

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 the following figure.

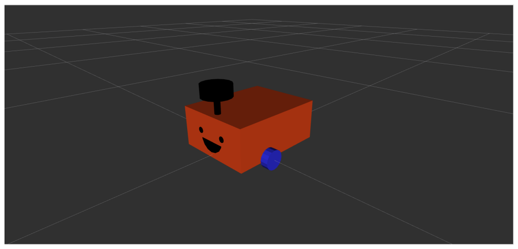

Figure 6: Example figure caption.

In order to calculate the index of the laser slot at the 30 degree position, you have the equation:

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

The simple collinearity theorem is used in the calculation of the distance r, as shown by the following figure:

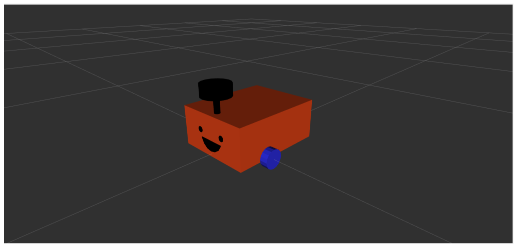

Figure 7: Example figure caption.

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 ← $ray1.x \cdot ray2.x + ray1.y \cdot ray2.y + ray1.z \cdot ray2.z$
      magnitude_1 ← MAGNITUDE_OF_RAY(ray1)
      magnitude_2 ← 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 ← isfinite(new_value)
      old_finite ← isfinite(old_value)
      if not new_finite and not old_finite then
          return not isnan(new_value)
      end if
      range_check ← (range_min ≤ new_value ≤ 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 the previous section it was mentioned that 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 in the previous section it was implemented to extract any row of the depth map to be converted to LaserScan data, the selection of non-centre rows can cause the algorithm to incorrectly judge the floor (or ceiling if taking the top half of the depth image) as an obstacle at longer distances, resulting in the map not being constructed correctly. The simulation in RViz2 is shown below.

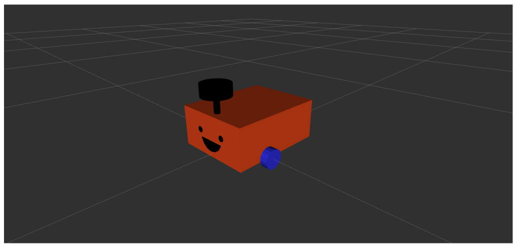

Figure 8: Example figure caption.

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 is not going to maximise the demand. To avoid this, 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. The following shows how to compute 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 referred to as the "line of sight" in the text. 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 sweep is maximised in a given situation while avoiding the occurrence of misidentification. In the figure, assuming that there is an obstacle close to the ground, then its horizontal height difference from the depth camera's centre of gravity can be seen as equivalent to the distance h from the depth camera's centre of gravity 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.

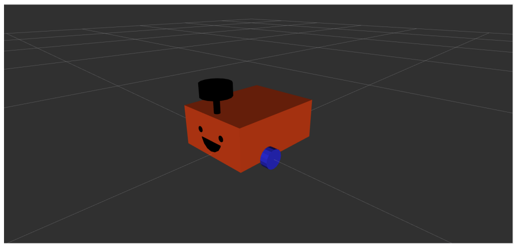
Figure 9: Example figure caption.

In the figure below, 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. 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 the following equation:

$$l' = \frac{h}{\sin{(\alpha + \beta)}} \tag{5}$$

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

$$h' = \tan(\arcsin(\frac{h}{l})) \times c \tag{6}$$

Finally, substituting into the original equation yields:

$$l' = \frac{h}{\sin{(\arctan(\frac{\tan(\arcsin(\frac{h}{l}))\times c + x}{c}))}} \tag{7}$$

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

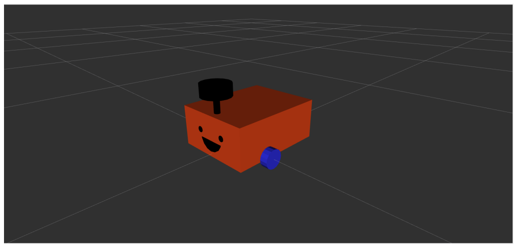
Figure 10: Example figure caption.

Furthermore, from the representation of the diagonal α in equation (666) 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 important 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 following figures.

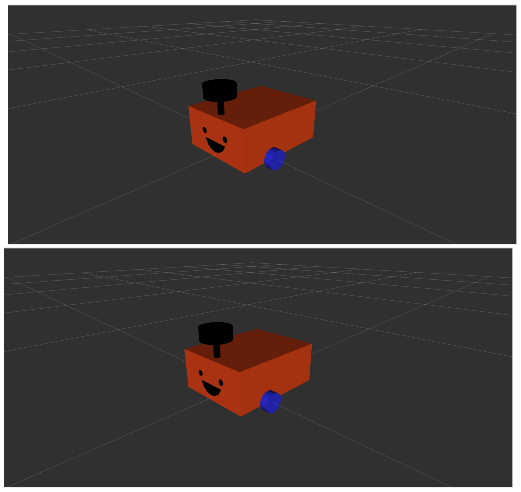
Figure 11: Example figure caption.

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. 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 was customised 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:

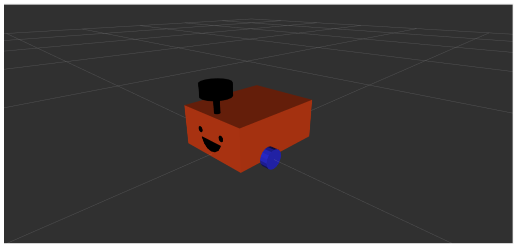

Figure 12: Example figure caption.

A new dummy LaserScan array is created. 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. 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.

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
```

3.4 Resample of Depth Image Data

However, the fusion of the two LaserScan messages mentioned in the previous section is not ideal, as shown in Figure:
This is because the resolution of the LaserScan information converted from LiDAR and depth images is not the same. Since there are 1024 pixels per line in a 1920x1080 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.

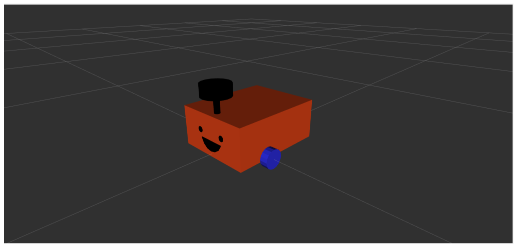

Figure 13: Example figure caption.

In order to solve this problem without degrading the accuracy of the depth image conversion, it is proposed to unify the resolution of the two LaserScan's based on the second part of the data preprocessing described above 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.

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. The resampled range values are obtained by converting the interpolated numpy array into a list.

First, the angles of the original and target scan data are stored in two arrays of values generated at specified intervals within the specified range. 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.

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

In line with the methodology used above, this project was tested in Gazebo Classic and RViz2 based on ROS2 Humble. The first is a replication of the problem to be solved in this project. Two obstacles that are prominent 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 shelf will not detect the shelf, but only the baffles on either side. To visualise this more, the back plate of the shelf was removed and the shelf was 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. It will not be able to navigate correctly and at the same time it will destroy the original map as shown in the figure.

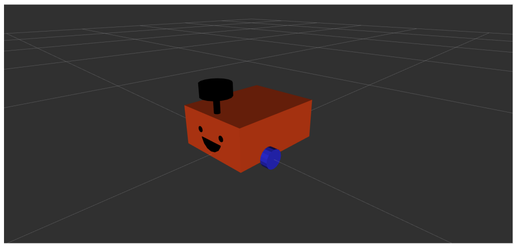

Figure 14: Example figure caption.

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

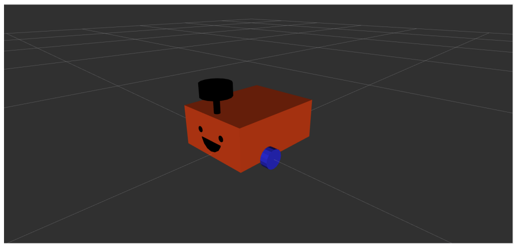

Figure 15: Example figure caption.

2. Barricades: Another very intuitive test is this cylinder barricade with a polygonal bottom cross-section and a rounded upper cross-section. As shown in the figure.

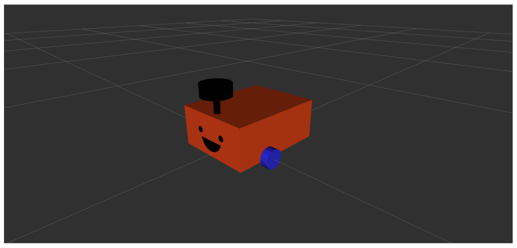

Figure 16: Example figure caption.

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:

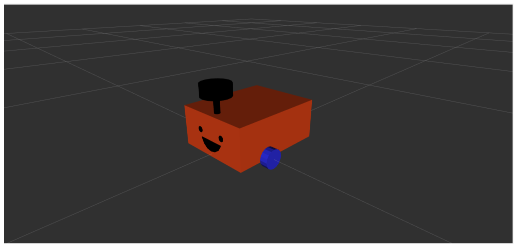

Figure 17: Example figure caption.

It can be seen that what was scanned before the fusion was the higher part of the top half with a circular cross-section, whereas after the merger the scanning of this area was 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.