Northeastern University

EECE 5554: Robotics Sensing and Navigation

Professor: Thomas Consi

Project Report

December 2024

Generation Of 3D Point Cloud From 2D LIDAR

Group I

Ajith Sylvester, Athmavidya Venkataraman, Kamalnath Bathirappan,
Ket Ashvinbhai Bhikadiya, Richik Sinha Choudhury

Abstract

Most mobile robots utilizing the Robot Operating System (ROS) rely on LiDAR-based Simultaneous Localization and Mapping (SLAM), with indoor robots often employing 2D SLAM. While some robots are designed to operate both indoors and outdoors, navigating in dynamically changing environments presents challenges. For instance, 2D LiDAR can only detect obstacles in the plane of the sensor, often missing vertical obstacles such as chairs or hanging objects, which can lead to collisions despite careful placement of the LiDAR. One potential solution is to use 3D LiDAR, which provides a full 360° detection range. However, 3D LiDAR sensors are significantly more expensive.

In this project, we propose a novel approach to generate 3D point cloud data using a 2D LiDAR by adapting the sensor's position and movement. Specifically, we suggest two methods:

1. **Perpendicular Rotation Method**: The LiDAR is mounted perpendicular to the environment, with the base of the LiDAR rotating back and forth by 180° to capture 2D data at different vertical angles, creating a pseudo-3D point cloud.
2. **Parallel Rotation Method**: The LiDAR is placed parallel to the environment and moves vertically (up and down) while the base rotates 180° horizontally, again collecting 2D LiDAR data at various heights to simulate a 3D scan.

While both methods can generate a 3D representation of the environment, the rotation speed and update frequency of the LiDAR are important factors in capturing dynamic obstacles. In the first method, the LiDAR takes longer to complete a full rotation, which may result in delayed updates when dynamic objects enter the detection range. In contrast, the second method, which involves less than 180° rotation, can potentially yield faster updates by reducing the time for each cycle.

In this project, we have implemented both methods and compared the resulting 3D point clouds in terms of accuracy, scan quality, and update frequency. The primary objective is to optimize the generation of 3D point clouds using a cost-effective 2D LiDAR, enabling mobile robots to more effectively navigate complex, dynamic environments.

Code and recorded data for our paper can be found on our GitHub repository at https://github.com/bathirappan-k/2D-Laser-To-3D-PointCloud.

Background

The high cost of 3D LIDAR sensors, starting at several thousand dollars for a mid-range unit (such as the Velodyne Puck) to $10,000+ for high end units make 3D SLAM techniques inaccessible for low-cost robots, including robots that are deployed at mass scale, such as delivery, warehouse, and agricultural robots. Without 3D environment scanning, these robots' ability to avoid vertical obstacles is severely limited. A method to enable three-dimensional environment mapping with low-cost hardware would greatly open up performance improvements for low-cost robots and could make complex capabilities more widely accessible in educational settings.

Methodology

Our project uses a relatively inexpensive 2D LIDAR sensor, the Youyeetou FHL-LD19P, mounted on a custom-designed 3D printed bracket that holds the LIDAR sensor perpendicular or parallel to the floor. This bracket is mounted on a servo, which gimbals the sensor horizontally or vertically to capture a sweep of the environment. The bracket is designed such that the LIDAR sensor's center is directly in line with the rotational center of the servo. The servo body will remain stationary during a scan of the environment. The servo is controlled by an Arduino Uno, which subscribes to the servo angle from a ROS topic. From this servo angle, a ROS node can reconstruct the transformation between the LIDAR's reference frame and the base reference frame. Since the base is stationary, the transformation between the environment reference frame and base reference frame is fixed. The LIDAR sensor will publish its 2D scan data to a LaserScan ROS topic. As the servo sweeps the LIDAR sensor across its range of motion, the primary computational ROS node will subscribe to both the LaserScan and servo angle topics, pairing each scan and angle by their timestamps. The computational node will then begin to construct the point cloud by first projecting each 2D LIDAR scan into a combined 3D space by transforming each scan based on the servo angle. These projected points collected together represent the 3D point cloud. The points are then wrapped by an Open3D point cloud object, enabling interactive 3D visualization and conversion to other formats, e.g. ROS' PointCloud2.

After algorithm development, we tested both the perpendicular rotation method and vertical translation methods in an indoor environment on a stationary base, qualitatively evaluating the visualized point clouds and their accuracy in representing the environment, their resolution and fidelity, and scan stability, as well as quantitatively comparing total scan time (and hence update rate). We will also evaluate how each method responds to dynamic obstacles in the environment that may be moving.

Requirements / Experimental Setup

- LIDAR sensor: FHL-LD19P Lidar from Youyeetou
- Arduino Uno
- High-Torque Servo Motor
- Servo Motor Driver
- Battery
- Ubuntu 20.04 and ROS Noetic
- RViz
- Python 3.11 with Rosbag, NumPy, and Open3D Visualizer

Data Collection and Setup

Setup 1 Design Animation: Perpendicular Rotation Method
Setup 2 Design Animation: **Parallel Rotation Method**

Data Collection Video Setup 1: [Perpendicular Setup Data Collection]
Data Collection Video Setup 2: [Parallel Setup Data Collection]

For data collection using a 2D LiDAR in a perpendicular setup, the default ROS drivers provided by the LiDAR manufacturers were utilized to connect the LiDAR to the laptop via the UART module. An Arduino UNO was connected to the laptop, serving as a power source for the Arduino and also sending position control commands for the servo motor from the publisher node, which were subscribed by another ROS node running on the Arduino. These commands were then transmitted to the servo motor using a BILDA servo motor driver. This setup allowed the LiDAR mount to sweep 180 degrees back and forth.

Initially, data was collected by configuring the LiDAR scans to emit perpendicularly to the environment, and the ROS bag data was recorded, primarily capturing the servo angles and 2D LiDAR data. Subsequently, the same environment was recorded with the LiDAR scans set parallel to the environment.

Hardware Setup

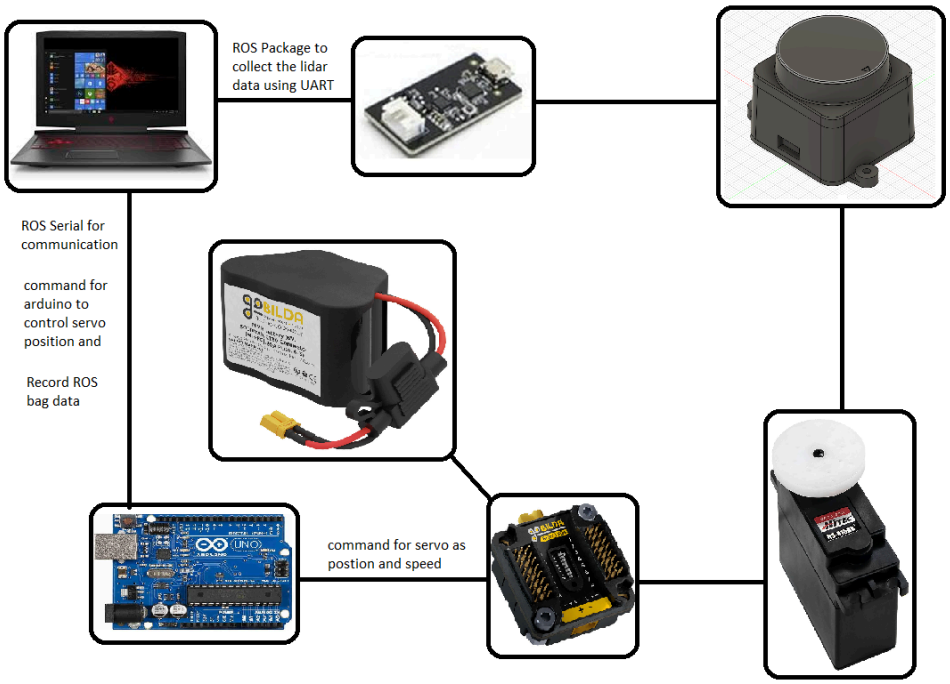

Fig 1: Hardware Diagram

Algorithm: Point Cloud Generation and Visualization

The 2D LIDAR scans from the FHL-LD19P sensor return an array of 455 ranges for each scan. Each item in the range array represents the distance the laser sensor measured at that angle, with the angle increasing clockwise from 0 degrees (marked on the sensor with a triangle). The

range measurements are roughly 0.79 degrees apart in angle, constructing a full 360 degree revolution. We refer to this angle as the LIDAR angle.

After the data is collected in a ROS bag file using the previously described algorithm, the LaserScan and servo angle messages are read from their respective topics two construct two NumPy arrays: an Nx455 array of ranges, and an Nx1 array of servo angles, where N is the number of LIDAR readings / associated angles. This enables us to process the entire batch of LIDAR data in parallel, speeding up computation. We also construct an array of length 455 containing the corresponding LIDAR angle in radians at each index.

We apply some preprocessing to the batched array of ranges, filtering out LIDAR angles outside of our specified useful range. This is done to filter out the readings obstructed by the sensor mounting bracket (below 5 degrees and above 395 degrees, and also to generate an "open-faced" point cloud for easier visualization. We also convert NaN values, which are returned by the LIDAR sensor when it fails to read a distance at a given angle (when the rangefinder doesn't receive a reflection) to 0, effectively ignoring them.

The array of LIDAR ranges can be conceptualized as a series of polar coordinates, with r = the range value and θ = the corresponding LIDAR angle, converted to a right-handed angle (increases counterclockwise). Projecting these points into Cartesian coordinates enables us to easily visualize a single 2D scan. For our project, we combine 2D scans in 3D space by using spherical coordinates, a generalization of polar coordinates in 3D.

In our perpendicular mount setup, the individual scans maintain radius r = the range value, and the polar angle θ = the LIDAR angle, representing the scan on the xz-plane, which is then revolved about the z-axis by setting the azimuthal angle φ = the servo angle. In the parallel mount setup, we simply rotate the axes, with the x-axis representing the axis of servo rotation - the LIDAR starts on the xy-plane [Figure 2].

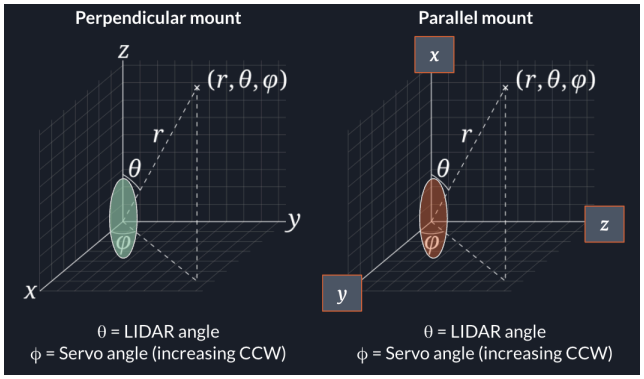

Figure 2: Spherical Coordinate Systems for Perpendicular and Parallel Setups

Once the corresponding spherical coordinates for each scan are computed, we project our points into Cartesian coordinates with parallel NumPy operations, computing x, y, and z coordinates for each LIDAR point (equations for perpendicular mount, change axes accordingly for parallel):

$$x = range * \sin(lidar\ angle) * \cos(servo\ angle)$$
$$y = range * \sin(lidar\ angle) * sin(servo\ angle)$$
$$z = range * cos(lidar\ angle)$$

Each resulting array is then flattened and combined, resulting in an (N*455)x3 array representing the computed point cloud in Cartesian coordinates. The array is then wrapped by an Open3D point cloud object for visualization and format conversion.

Results

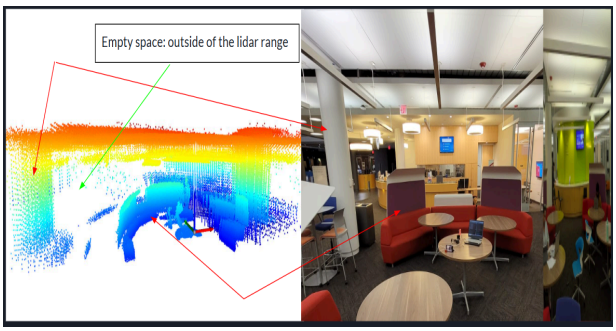

Figure 3 : LIDAR Map in Open Areas Using a Perpendicular Sensor Setup (for front 0 to 180 degrees)

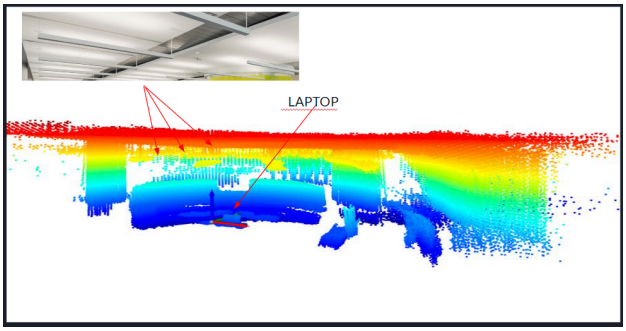

Figure 4 : LIDAR Map in Open Areas Using a Perpendicular Sensor Setup (for back 180 to 360 degrees)

Fig. 3 presents a LiDAR-generated map of an indoor space, capturing seating arrangements, structural pillars, and unmapped empty areas outside the sensor's range. The map aligns well with the real-world image, offering a clear representation. Fig. 4 showcases additional details captured by the LiDAR, such as the laptop on the table and ceiling lights. This demonstrates the accuracy achieved through precise data collection and algorithms to map both key structures and finer details effectively.

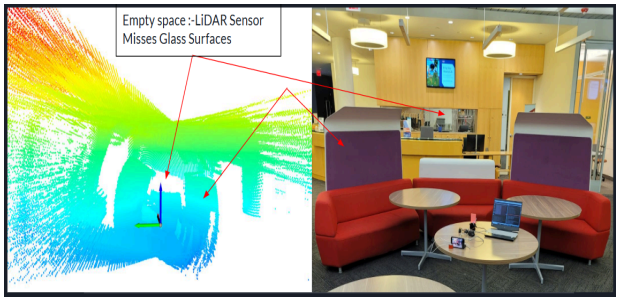

Figure 5 : Map in Open Areas Using a Parallel Sensor Setup

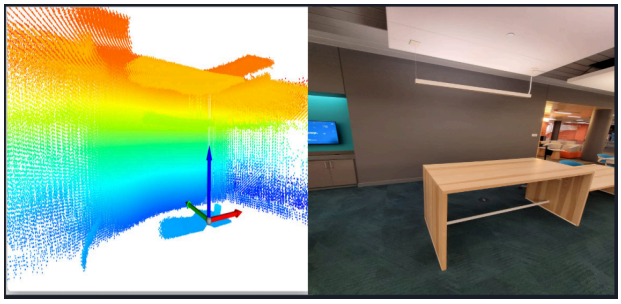

Figure 6 : Map in Front of Wall Areas Using a Perpendicular Sensor Setup

Fig. 5 shows a parallel setup for data collection. While it is less precise than the perpendicular setup, objects in the map remain easily recognizable. This setup also emphasizes the LiDAR's limitation in detecting glass surfaces. Fig. 6 displays data collected in front of a wall, where reflections for nearby objects are dense, but for distant objects, they become sparser due to beam divergence.

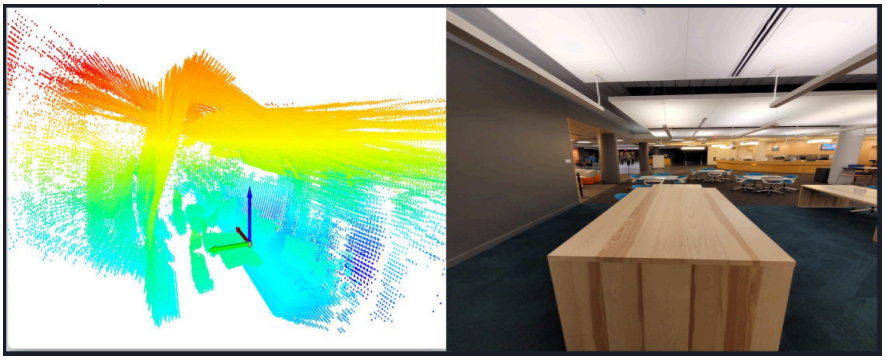

Figure 7 : LiDAR Map in Front of Wall Areas Using a Parallel Sensor Setup

Fig. 7 presents a parallel setup for LiDAR data from a straight and an open environment. In the map, The left side depicts the structured, straight environment, while the right side shows the more complex, open environment. This configuration shows this setup can perform in varying conditions. Despite limitations in LiDAR frequency and servo resolution, the system achieved impressive results. As demonstrated in the figures, the LiDAR effectively captured the environment's structure, even in the challenging open environment.

Conclusion

This project successfully demonstrated the generation of pseudo 3D point clouds using a cost-effective 2D LiDAR sensor through two distinct methodologies: the perpendicular rotation method and the parallel rotation method. By leveraging mechanical adjustments and software-based data processing, the project showcased an innovative approach to creating pseudo-3D representations of an environment. The results highlighted the trade-offs between the methods, with considerations for accuracy, scan time, and responsiveness to dynamic obstacles. While the perpendicular rotation method provided comprehensive scans, it faced limitations in update frequency, whereas the parallel rotation method offered faster updates but with worse resolution on vertical features and a smaller effective range. These findings underscore the potential of adapting low-cost sensors for advanced applications, paving the way for more accessible solutions in robotics and automation. Future improvements in real-time performance, sensor fusion with cameras and IMUs, and hardware optimization can further enhance the utility of this approach in diverse scenarios, from indoor navigation to outdoor exploration.

References

1. M. G. Ocando, N. Certad, S. Alvarado and Á. Terrones, "Autonomous 2D SLAM and 3D mapping of an environment using a single 2D LIDAR and ROS," 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), Curitiba, Brazil, 2017, pp. 1-6, doi: 10.1109/SBR-LARS-R.2017.8215333.
2. Mohd Yusuf, Aman Zaidi, Abid Haleem, Shashi Bahl, Mohd Javaid, Sonu Bala Garg, Jatinder Garg. IoT-based low-cost 3D mapping using 2D Lidar for different materials, Materials Today: Proceedings, Volume 57, Part 2, 2022, Pages 942-947, ISSN 2214-7853,https://doi.org/10.1016/j.matpr.2022.03.161. (https://www.sciencedirect.com/science/article/pii/S2214785322014857)
3. M. Chen, S. Yang, X. Yi and D. Wu, "Real-time 3D mapping using a 2D laser scanner and IMU-aided visual SLAM," 2017 IEEE International Conference on Real-time Computing and Robotics (RCAR), Okinawa, Japan, 2017, pp. Sew 297-302, doi: 10.1109/RCAR.2017.8311877. keywords: {Three-dimensional displays;Two dimensional displays;Measurement by laser beam;Visualization;Simultaneous localization and mapping;Laser fusion;Pose estimation},
4. Paul J. Besl, Neil D. McKay, "Method for registration of 3-D shapes," Proc. SPIE 1611, Sensor Fusion IV: Control Paradigms and Data Structures, (30 April 1992); https://doi.org/10.1117/12.57955