Cheaper Alternative to LiDAR Based Surveying for 3d Mapping

Michael Hernandez and James Doherty

Introduction

Camera and sensor aided technology has been utilized increasingly by civil engineers and architects, and affordable 2d LiDAR (Light Detection and Ranging) systems have also become accessible to consumers and hobbyists to use for a variety of purposes from surveying to various applications in robotics. However, affordable(<$500) 3d LiDAR scanners are not widely available on the consumer market. Our project consists of rotating a readily available and cheap RPLIDAR A1 scanner with a hobbyist servo motor. By combining this data with an IMU (inertial measurement unit) we measure rotational acceleration experienced by the LiDAR. By combining the sensor inputs we can form a point cloud of a room for analysis. Our semantic segmentation deep learning algorithm parses through a mesh of point clouds and detects objects. The device was tested in hallways and rooms by measuring the location of objects and walls in a room and then comparing this data with information that the LiDAR records and calculates. The project provides a low cost option for an automatic comprehensive 3d view and measurements of an indoor space to be used by engineers and architects to gauge the environment and approach solutions. This 3d LiDAR scanner is useful for robotics applications such as mapping/localization as well as 3d scanning for surveying and architecture.

Objectives

1. Create a low cost 3D LiDAR scanner
2. Successfully merge several scans from the 3D scanner to create an accurate map of the interiors of buildings.
3. Test the efficacy of the scanner under different light level conditions and determine a practical conditions to scan and length of time for scan.
4. Implement the PointNet++ Architecture for segmentation of point clouds to detect objects in the environment.

Materials

Hardware

- Adafruit Slamtec RPLIDAR A1
- 35 kg Servo Motor from DS Servo
- HiLetgo GY521 MPU-6050 6 Axis Accelerometer and Gyroscope
- ESP-WROOM-32 Development Board
- Camera Tripod
- Various 3D Printed parts
- RTX 3070 GPU with 5888 CUDA Cores

Software

- Arduino IDE for Controlling ESP32
- Slamtec C++ Library for RPLiDAR
- PointNet++ Deep Learning on Point Sets for 3D Segmentation
- CloudCompare for Visualizing Point Clouds.
- Anaconda Python Virtual Environment

Procedure

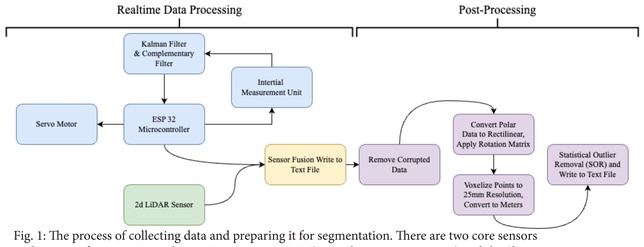


Fig. 1: The process of collecting data and preparing it for segmentation. There are two core sensors to the sensor fusion process: the MPU-6050 6 axis IMU(inertial measurement unit) and the Slamtec RPLiDAR A1 2d LiDAR scanner. The LIDAR scanner is on a servo motor controlled by the ESP-32 microcontroller and rotated on a horizontal axis. The microcontroler also polls the IMU and processes the rotation readings through a Kalman Filter before writing these to the sensor fusion program. This program combines the LIDAR and IMU data into a text file with one point per line. After raw data is collected, corrupted data as a result of high polling speeds is removed and then the raw data from the LIDAR is converted to rectilinear and a rotation matrix applies the IMU rotation readings to the LIDAR points. Next, voxelization is used to reduce noise, and statistical outlier removal is used to further reduce unwanted outlier points that result from an occasional faulty readings. The cleaned points are outputted into a text file.

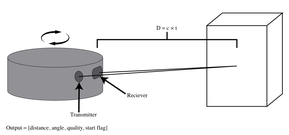


Fig. 2: The Slamtec RPLIDAR A1M8 Scanning Process. The lidar scanner is rotated by a DC motor. The scanner has a transmitter and reciever to send and detect incoming light. The distance that the lidar measures a point at is determined from the length of time for the light to travel. This allows the scanner to have 360 degree FOV and by rotating this on a servo motor, a comprehensive 3D scan is made.

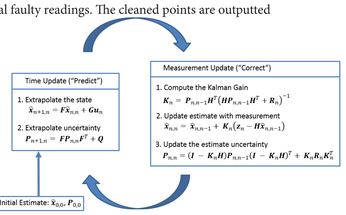


Fig. 3: The Process of the Kalman Filter (obtained from kalmanfilter.net, see citations). The process of the Kalman filter involves an initial estimate based on the previous measurement, a prediction based on the previous states, and a measurement update based on "Kalman Gain". The update is used to estimate the uncertainty of the filter. The filter is run on every iteration of IMU readings to provide a more steady angle.

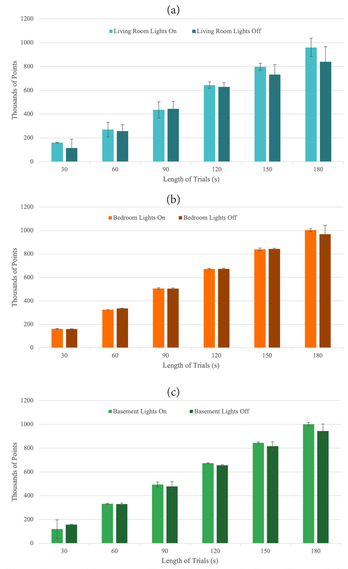


Fig. 4: Average Number of Points Collected in Three Rooms With Lights On and Lights Off. For each room (basement, bedroom, and living room), 3 trials were conducted under the same conditions. Lights off scans were classified by scans taken when no light source was present, meaning all lights in the room were turned off and scans were taken at night. Lights on scans were taken at noon allowing for extra light from windows and all the light sources from lamps and ceiling lights were turned on. The difference between number of scan points shows a slightly higher amount of points recorded from scans taken when the light level was high. Additionally, larger rooms such as the living room (4a) show a greater disparity in point quantities suggesting a higher light level influences a broader scan range.

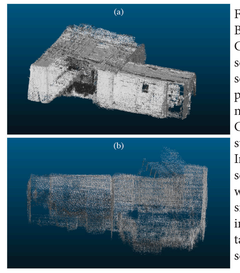


Fig. 8: A Full Scan of A Basement From Several Point Clouds. For the basement, the scanner was used to take six, 60 second scans around different parts of the room. Fig. 8a uses normals computed in Cloud Compare to more easily see the surfaces found by the scanner. In this figure stairs can be clearly seen, as well as a door and clear walls. Fig. 8b has smaller point size on the same scan to look into the scan. A couch and a table can be made out in this scan.

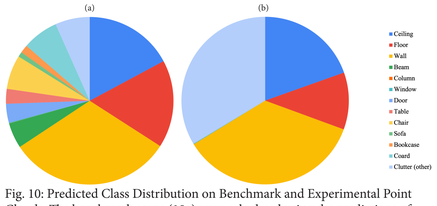


Fig. 10: Predicted Class Distribution on Benchmark and Experimental Point Clouds. The benchmark scans (10a) were calculated using the predictions of the Stanford 3D Indoor Scene Dataset, while the experimental scans (10b) were calculated on the mean of scans of a residential basement and school hallway. The chart shows that the point clouds are high quality and contain adequate points for walls, floors, and ceilings. The model is able to reliably detect these features. The chart also shows that the model struggles at recognizing smaller objects like furniture and windows, and marks these objects as clutter.

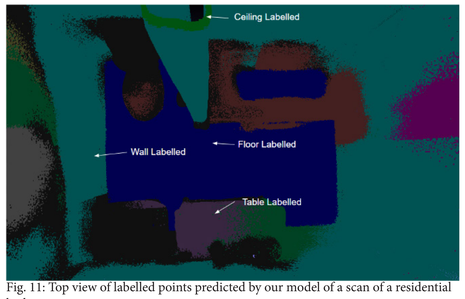


Fig. 11: Top view of labelled points predicted by our model of a scan of a residential bathroom.

System Results

Table 1: Cummulative Rate of Points Recorded After Cleaning Processes (points per second). The RPLIDAR scanner has a theoretical maximum speed of 8000 points per second but due to sensor fusion and cleaning processes, this number is significantly less. Cleaned scanned data is classified as data that was voxelized and filtered through the SOR filter (see Fig. 1). What is also seen is that as the length of time of the scan increases, the cummulative rate of point collection decreases. Rate of point collection is also lower when the light level is lower. From this we determined that it is not necessarily worth it to increase the length of time for scan.

Time of	Living Room		Bedroom		Basement	
Scan (s)	Lights On	Lights Off	Lights On	Lights Off	Lights On	Lights Off
30	2757	2070	2271	2220	1899	2393
60	2114	2078	1892	1827	2241	2181
90	2054	2114	1656	1610	1998	1965
120	2022	2027	1471	1496	1755	1771
150	1920	1826	1356	1346	1629	1112
180	1668	1638	1212	1239	1523	1452

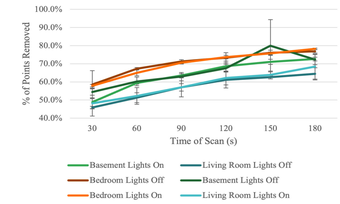


Fig. 5: Percentage of Points Removed During Post-Processing from Raw Scan Data to Cleaned Scan Data. Cleaned scanned data is classified as data that was voxelized and filtered through the SOR filter (see Fig. 1). A substantial number of points collected in the scans is redundant and errors accumulate as time goes on. This leads to a efficiency reducing significantly as time goes on. Also seen with the "Lights Off" groups is that the % of points removed is greater than the "Lights On" group. This shows that more light leads to less noisy and data with fewer outliers.

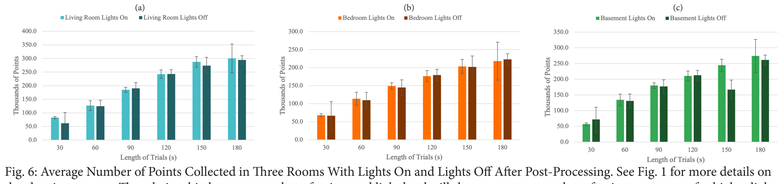


Fig. 6: Average Number of Points Collected in Three Rooms With Lights On and Lights Off After Post-Processing. See Fig. 1 for more details on the cleaning process. The relationship between number of points and light level still shows a greater number of points on average for higher light level scans but the standard deviation also increases. A higher standard deviation suggests that there is a variability in scan quality from scan to scan that is not accounted for by number of points. Therefore, taking multiple scans and merging them together may prove to be a viable strategy for attaining good quality scans.

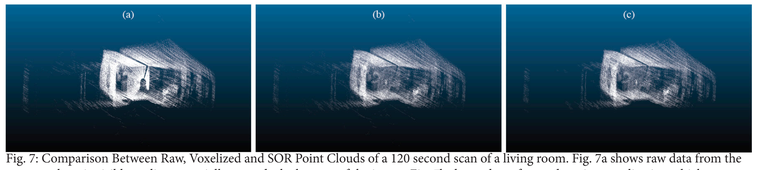
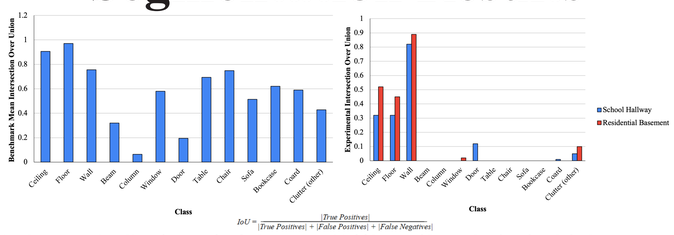


Fig. 7: Comparison Between Raw, Voxelized and SOR Point Clouds of a 120 second scan of a living room. Fig. 7a shows raw data from the scanner, there is visible outliers especially towards the bottom of the image. Fig. 7b shows data after undergoing voxelization which removes noise from the scan. Unnecessarily high density of points (especially towards the middle of the scan) is removed as well as some outliers. Finally, the statistical outlier removal clearly removes many wrongly scanned points that would introduce noise into the point cloud and make it unusable for the segmentation model.

Segmentation Results



$$IoU = \frac{|True\ Positives|}{|True\ Positives| + |False\ Positives| + |False\ Negatives|}$$

Fig. 9: Mean Intersection Over Union by point class for Segmentation Model Processed Data on Benchmark vs. Lidar Scans. The intersection Over Union (IoU) for each class was calculated with the above equation. The mean IoU for the benchmark dataset was calculated using Area 5 of the Stanford 3d Indoor Scene Dataset. The Mean IoU for point clouds was calculated on visual estimates of false negatives, positives and true positives. IoU in both benchmark (9a) and experimental (9b) trials of the segmentation was highest for the floor, ceiling and wall classes, showing a highly accurate classification of important building features. IoU was lower for furniture classes in benchmark data (9a), and was much lower for furniture classes in experimental data (9b). This was due to a high false positive rate for the "clutter" class and demonstrates the model has a lower understanding of smaller objects and furniture in rooms. It should be noted that IoU is not a percent accuracy but is directly proportional to accuracy.

Scan Name	Total Duration (Seconds)	Points	Time Per 1,000,000 Points
Benchmark_Office	67	1136617	58.9
Benchmark_Conference	46	777622	59.2
Benchmark_Hallway	27	510949	52.8
School_Hallway	33	634021	52.0
Basement	36	577339	62.4
Mean	41.8	727310	57.1

Table 2: Pointnet++ Segmentation Run Time on Point Clouds. Run time of the Segmentation Model was calculated by timing the duration of model run time. The model used 8 gigabytes of RAM and used a 5888 CUDA Core Nvidia Geforce RTX 3070 GPU. The table shows that time of the model is linearly proportional to the number of points in the scan. This suggests an O(n) runtime making our program efficient on dense and large point clouds with apt hardware.

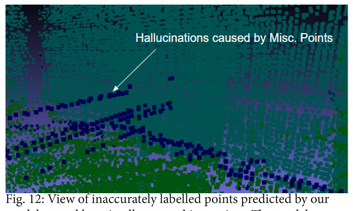


Fig. 12: View of inaccurately labelled points predicted by our model caused by miscellaneous object points. The model labelled these points as a floor instead of a sofa.

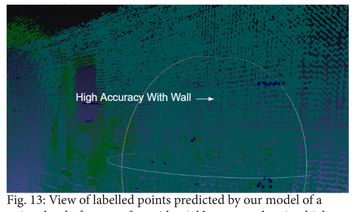


Fig. 13: View of labelled points predicted by our model of a point cloud of a scan of a residential basement showing high accuracy for wall detection.

Procedure (cont.)

After the raw data is collected and stored in a CSV file, it needs to be processed into usable points for data. The data from the LiDAR scanner is transmitted as polar coordinates with θ being represented in degrees. This data is run through conversion into rectilinear coordinates. After being processed, the points are left as rectilinear points. The z value is left as zero for the time being:

$$P_i = \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix}$$

Next, the points are rotated about the roll, pitch, and yaw angles determined by the IMU readings. The rotation matrices used are as follows.

$$R_{roll}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} R_{pitch}(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} R_{yaw}(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A dot product is used to determine the rotation of points.

$$P_{roll} = R_{roll}(\theta) \cdot P_i \qquad P_{roll+pitch} = R_{pitch}(\theta) \cdot P_{roll} \qquad P_f = R_{yaw}(\theta) \cdot P_i$$

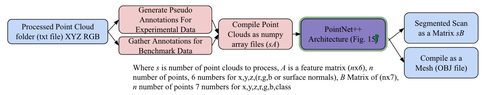


Fig. 14: Post-processing architecture. The processed point cloud is parsed and annotations are compiled into a Numpy Array with features for each point cloud. Annotations shows ground truth for benchmark data, but generation of pseudo annotations is necessary as the PointNet++ compiler expects an annotated pointcloud for evaluation of accuracy. Each point cloud is passed to the PointNet++ segmentation model and a matrix with labelled points is generated. Additionally, we generate a mesh (object file) for useful applications of the point cloud in CAD softwares, robotic usage and data visualization.

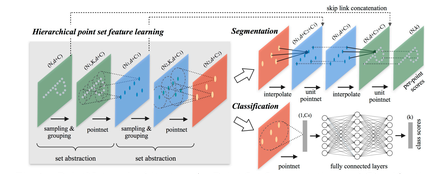


Fig. 15: PointNet++ Architecture for Point Set Segmentation and Classification (obtained from PointNet++ Standford University, see citations). PointNet++ herarcically learns features about the point cloud by recursively applying sampling and grouping to divide the point cloud into local regions and learning geometric features. Each local region is processed using a smaller Pointnet for the region. This helps to extract high level features from multiple scales and groupings. The learned features are then interpolated and upsampled, with skip connections to refine per-point predictions. Finally, a unit PointNet processes the refined features to output a per-point classification score, determining the semantic label of each point.

Conclusion

- It is possible to make a LiDAR scanner with relatively cheap components that can obtain usable scans of real world interiors. Assuming the user has access to a computer with Nvidia CUDA support, our scanner has an estimated cost of $130 dollars. Making it incredibly affordable.
- When it comes to indoor settings, there are clear benefits to having more light available in the room, however practical data is still achievable in the dark, albeit with more noise and fewer points for the same length of time.
- Longer scan time does not necessarily correlate to higher scan quality. Qualitatively, we determined that 60 second scan times were highly usable for segmentation, especially after being combined with multiple other scans. After 90 seconds of scan time, the chances of a scan failing increase, as well as the number of outliers (Fig. 5)
- Point Clouds generated by our solution are reliable enough to be segmented by a pre-trained Pointnet++ model, particularly for non-furniture classes.
- Our solution generates segmented point clouds that can prove helpful for architects, engineers, robotics and any future project that requires an annotated 3d model of an interior space. Our contributions are helpful to enhancing the affordability of the developing field of computer vision.

Future Modifications

1. Using Raspberry Pi, we can make the entire system wireless, making it far more usable in a real world setting.
2. By fusing sensor data with ROS2 (Robotic Operating System) we believe we can increase reliability and speed of real time data processing. ROS2 would also allow us to implement a SLAM (simultaneous localization and mapping) algorithm which would allow us to move the scan around while it is recording scan data. This would potentially give the scanner abilities relative to multi-thousand dollar scanners.
3. We believe we can fine tune the segmentation model with annotations of our scans to be better trained on our LiDAR (which does not include color data).
4. Additioanlly, make a graphical user interface to make the scanner more acessible and usable for consumers.

References

- Becker (www.kalmanfilter.net), A. (n.d.). Online Kalman Filter Tutorial. Www.kalmanfilter.net. https://www.kalmanfilter.net/background.html
- Charles Ruizhongtai Qi, Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.1706.02413
- Comerzan, S. (2021). Lidar as an additional tool for drone based reconnaissance. Institute of Technology Carlow.
- Lauszus. (2012, September 12). TKJ Electronics» A practical approach to Kalman filter and how to implement it. TKJ Electronics. https://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/
- RPLIDAR-A1 360°Laser Range Scanner _ Domestic Laser Range Scanner|SLAMTEC. (n.d.). Www.slamtec.com. https://www.slamtec.com/en/Lidar/A1
- Slamtec RPLIDAR Public SDK for C++. (2022, October 2). GitHub. https://github.com/slamtec/rplidar_sdk
- yanx27. (n.d.). Yanx27/Pointnet_Pointnet2_pytorch: PointNet and pointnet++ implemented by pytorch (pure python) and on ModelNet, ShapeNet and S3dis. GitHub. https://github.com/yanx27/Pointnet_Pointnet2_pytorch
- Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A Modern Library for 3D Data Processing. Open3d.org. https://www.open3d.org/docs/release/introduction.html