3D Modeling and Mapping of Indoor Environments

Walter Hilderbrand, Joshua Moretto, John Murdock Jr., Steven Kligman, Mahdi Norouzi

2020 | University of Cincinnati:

College of Engineering and Applied Science

ABSTRACT

In this report, students discuss the capability of Light Detecting and Ranging (LiDAR/LIDAR/lidar) and its properties; the ability to use the Intel Realsense camera system and its operations; the cloud based streaming services designed and tested; and the integration of all systems to create an interior mapping system to complement the resources available and to create a 3D Virtual Reality (VR) space that is accurate to real building space. This document will provide information about similar case study experiments and technical papers discussing the physics behind LiDAR, the implementation of Robot Operating System (ROS), RPLiDAR, Intel Realsense and robot platforms. The intent of this project was to create a mobile platform that would be capable of creating a virtual model and map of any interior spaces. While the tests conducted showed promise of this project, due to the COVID-19 Pandemic, our testing and final implementation was cut short. Code and explanations are included later for use and implementation.

INTRODUCTION

The premise of this project is to use Light Detecting and Ranging (LIDAR/Lidar), a 360 degree camera, other sensors, software, and a live website to create real time 3-D models of interior spaces. Using an oscillating mount, and data capture software, the interior of a building was mapped, modelled and added to a virtual world that can be explored using Virtual Reality (VR) goggles. With the underlying data points and a 360 degree image overlay, the user is able to explore exact virtual replicas of the existing building. This document explores the mathematics behind the capture program, the theory of LIDAR, the system used when stitching 360° videos and images, and the method for integration of the LIDAR data, the 360° video overlay and the viewing software.

THEORY

Case Studies

From our research, a group from John Hopkins conducted similar experiments and equipment designs to this project. They custom made a housing/mount for the PrimeSense and RGB camera. A total of 4 sensor cameras (2 normal, 1 RGB and the PrimeSense) used to collect data. The group had a number of problems with the algorithm working since the platform is continually moving. When choosing the cameras and equipment, the researchers made the decision based on frame rate, field of view, bandwidth, resolution and cost. They seem to choose the best cameras for the mid-low price range. And finally, when choosing the main computing system for the drone, the group also used the matrix of computing power, size, and price. They seem to use consistent decision matrices as we have when determining which equipment to use and which to pass over.

Another article, *LOAM: Lidar odometry and mapping* authored by Ji Zhang and Sanjiv Singh, overviews how the authors created a system using a 2 axis lidar to create 3-d lidar maps. The benefit of using lidar is that it is not affected by the ambient light. The benefit of this system over others is that this will minimize drift in lidar odometry estimation. The rest of this article gives an overview of the odometry algorithms used, a basic outline for their lidar odometry code, and how lidar mapping works. Their method has been tested indoors and outdoors with success and uses a battery and laptop to capture data. The system uses two algorithms in parallel and compares the data between the two the mapping algorithm and the odometry algorithm.

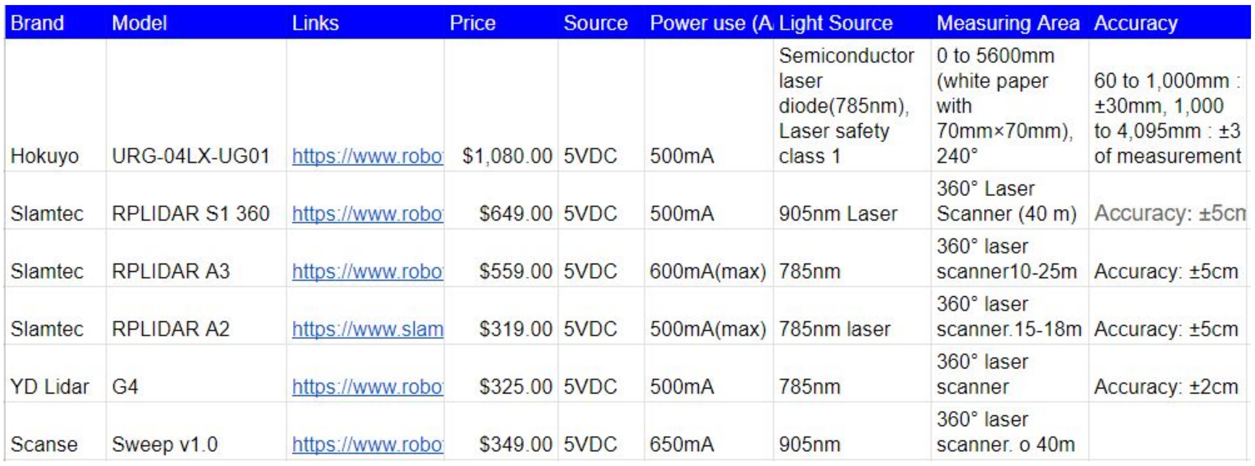
From *A Guide for 3-D Mapping with Low-Cost Sensors Using ROS* the authors present an overview and instructions on how to setup a 3-d mapping system using the xbox kinect and the Rplidar A2. This article gave students good information as how to set up and gather data from the rplidar and gives some insight on how to combine the two systems using ros kinetic, the same ros currently used to run the project's lidar system. This project will differ by using the intel real sense, a more capable sensor system.

Finally, in *Simultaneous Localisation and Mapping (SLAM) Part I The Essential Algorithms*, the research article covers the basic algorithms of SLAM. First the article defines what SLAM is, Simultaneous Localisation and Mapping. This algorithm gives a machine the ability to track its position in an environment and also the ability to map a new environment it has not yet explored. SLAM works by using variations of the Kalman filter and the Blackwellized particle filter. This article further gives an explanation of how both of these filters work, it overviews the equations necessary for measurement updates and point tracking. It then discusses what points are used for tracking. This article also gives an insight into the future of slam.

Equipment Research

Lidar:

After deciding to use lidar for the room imaging low cost and easily available lidar sensors were compared. The lidar was chosen based on the following criteria. Power Consumption: it was necessary to have low power consumption so that the lidar could be run off of a raspberry pi so the final package could be small and easily portable. Programmability: this was the most important factor in choosing a lidar. The lidar needed to have an open software development kit (SDK) and have support and code readily available. This will reduce the amount of time it takes to integrate multiple components and increase the usability of the lidar data. Accuracy: accuracy was also very important in the decision making process. The lidar chosen needs to be very accurate as it will be used to create a dimensionally accurate 3-D building model.



Brand	Model	Links	Price	Source	Power use (A	Light Source	Measuring Area	Accuracy
Hokuyo	URG-04LX-UG01	https://www.robo	$1,080.00	5VDC	500mA	Semiconductor laser diode(785nm), Laser safety class 1	0 to 5600mm (white paper with 70mm×70mm), 240°	60 to 1,000mm : ±30mm, 1,000 to 4,095mm : ±3 of measurement
Slamtec	RPLIDAR S1 360	https://www.robo	$649.00	5VDC	500mA	905nm Laser	360° Laser Scanner (40 m)	Accuracy: ±5cn
Slamtec	RPLIDAR A3	https://www.robo	$559.00	5VDC	600mA(max)	785nm	360° laser scanner10-25m	Accuracy: ±5cm
Slamtec	RPLIDAR A2	https://www.slam	$319.00	5VDC	500mA(max)	785nm laser	360° laser scanner.15-18m	Accuracy: ±5cm
YD Lidar	G4	https://www.robo	$325.00	5VDC	500mA	785nm	360° laser scanner	Accuracy: ±2cm
Scanse	Sweep v1.0	https://www.robo	$349.00	5VDC	650mA	905nm	360° laser scanner. o 40m	

Table 1: LIDAR selection and information chart

Upon careful consideration, the decision was made to choose the RPLIDAR A3 as our LiDAR for the project. The device was high enough quality, had an SDK, had a good accuracy, and was power effective enough for the price. From the point the RPLIDAR A3 was chosen, the project began to take shape and the testing/ familiarization process could take place. Many of the other options were not selected because of their prices, power consumption or their accuracy were not to standard.

Robot Platform:

The next area to determine was how to map the rooms in a systematic way. The ideas of using a robot platform or remote control car/platform was presented. The robots below were considered for the project.

Category	**TurtleBot3 Burger**	**Turtlebot3 Waffle**	**Turtlebot3 Burger Pi**	**DJI RoboMaster**

Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C	Lithium Polymer 10.8(11.1)V 2400mAh/ 25.92Wh
Battery Life	2h 30m	2h	2h	35min
LDS(Laser Distance Sensor)	360 Laser Distance Sensor LDS-01	360 Laser Distance Sensor LDS-01	360 Laser Distance Sensor LDS-01	-
Camera	-	Intel® Realsense™ R200	Raspberry Pi Camera Module v2.1	Proprietary
SBC (Single Board Computers)	Raspberry Pi 3 Model B and B+	Intel® Joule™ 570x	Raspberry Pi 3 Model B and B+	Proprietary-> uses
Price	549$ new	X	1400$ new	549$

Table 2: TurtleBot3 comparison between Burger, Waffle, and Burger Pi bots.

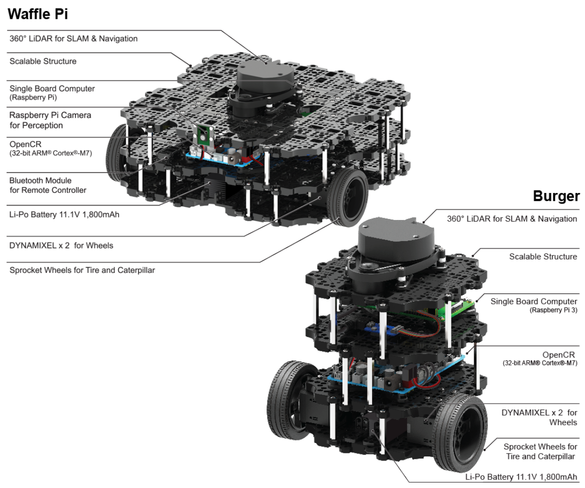

Waffle Pi
360° LiDAR for SLAM & Navigation
Scalable Structure
Single Board Computer (Raspberry Pi)
Raspberry Pi Camera for Perception
OpenCR (32-bit ARM® Cortex®-M7)
Bluetooth Module for Remote Controller
Li-Po Battery 11.1V 1,800mAh
DYNAMIXEL x 2 for Wheels
Sprocket Wheels for Tire and Caterpillar
Burger
360° LiDAR for SLAM & Navigation
Scalable Structure
Single Board Computer (Raspberry Pi 3)
OpenCR (32-bit ARM® Cortex®-M7)
DYNAMIXEL x 2 for Wheels
Sprocket Wheels for Tire and Caterpillar
Li-Po Battery 11.1V 1,800mAh

Figure 1: These are the burger and Waffle Pi bot platforms considered for the project.

When researching options to use as a platform the TurtleBot3 Burger, Waffle, and Waffle Pi arouse as possible solutions. They are sturdy research and open source platform with ability to integrate LiDAR and Intel RealSense cameras easily (actually depicted on the Waffle Pi bot). There have been multiple research teams/graduate thesis publications for LiDAR mapping and other applications with these platforms. They are very modular in design allowing for multiple configurations. They are slightly more expensive (Burger Pi) new, but if we can find a used one with everything but the sensors, possibly cheaper? They are decent platforms if we don't care about costs.

EXPERIMENT

Lidar-

Tried to run the RPlidar with raspberry pi running ubuntu 18.04. ROS applications are not supported by ubuntu 18.04 and will not download properly to the raspberry pi. The solution to this problem will be changing the operating system on the raspberry pi to either an earlier version of ubuntu (16 or earlier) or a version of debian optimized for the raspberry pie

Upon further research and testing ubuntu 16 and ros kinetic was successfully loaded onto the raspberry pie. From here the rplidar software package was loaded onto the pi and the rplidar a2 was successfully ran using rviz. One problem arose when running the lidar off of the raspberry pie. After running the lidar for 30+ minutes the raspberry pie would overheat and would have to be powered down. The next steps will be to get raw data from the lidar scan and to research solutions to the overheating problem.

Lidar Test Parameters

The following tests were used to begin collecting data with the lidar: Stationary Lidar with stationary walls; Stationary Lidar with moving simulated walls; Movin Lidar stationary walls; Moving Lidar and moving the simulated walls. Each test will use the RPLidar, Raspberry pi, and visual monitor. All tests will have the lidar stationary to begin the test for 5 seconds to calibrate and create the basepoint location values. Store those data points as the initial position of the robot/platform. After the baseline is recorded, the commands are executed to begin continuous data collection. Once data begins streaming, move the respective objects to determine the delta of the points collected. Record data as it is collected while time stamping all data collected.

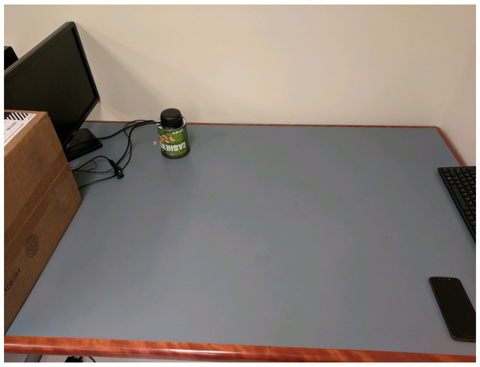
Procedure:

1) Select enclosed location for testing.
2) Set up the RPLidar unit and data collection system.
3) Begin calibration shoot. Record those data points

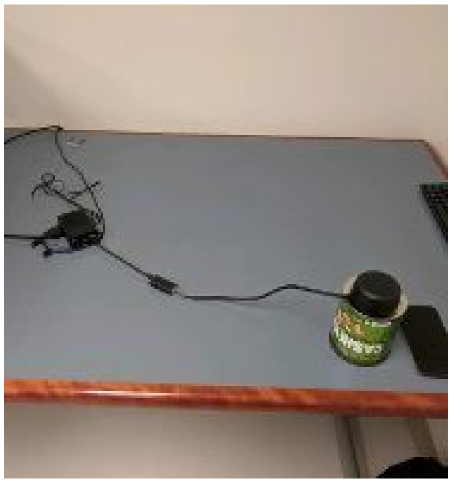
4) Double check calibration complete.

5) Begin moving Lidar/Walls/both to collect deltas

5a) For tests of moving the lidar, move the device 1’ in the x axis and y axis (+/-) on a box like course to accentuate the movement delta.



Position 1 Position 2



Position 3 Position 4

Figures 2 (a-d): Lidar test initial testing

HOLD ON THIS TEST: 5b) for tests of moving the walls, move each wall individually in the respective x-y (+/-) directions 1'. After moving, move it back and then move the next wall.

5c) Tests that have both will begin with moving the Lidar, then the walls, then both in the (+/-) x-y plane. Each individual move will only have one object moving at a time.

5d) Upon successful data collection, storage, and calculation-calibration, begin moving 2 objects at simultaneously.

Results:

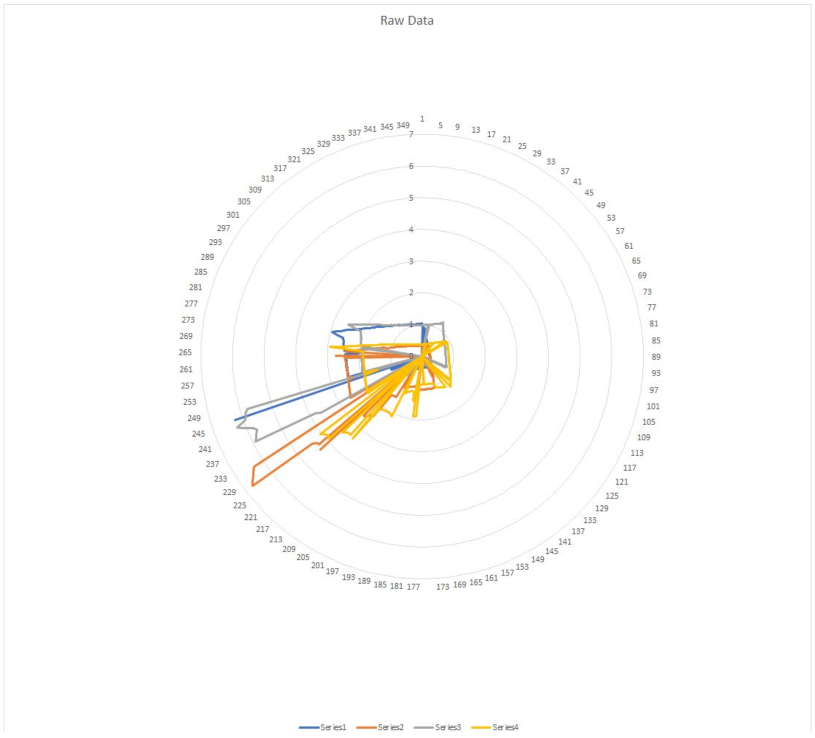


Figure 3: Lidar Initial Test result

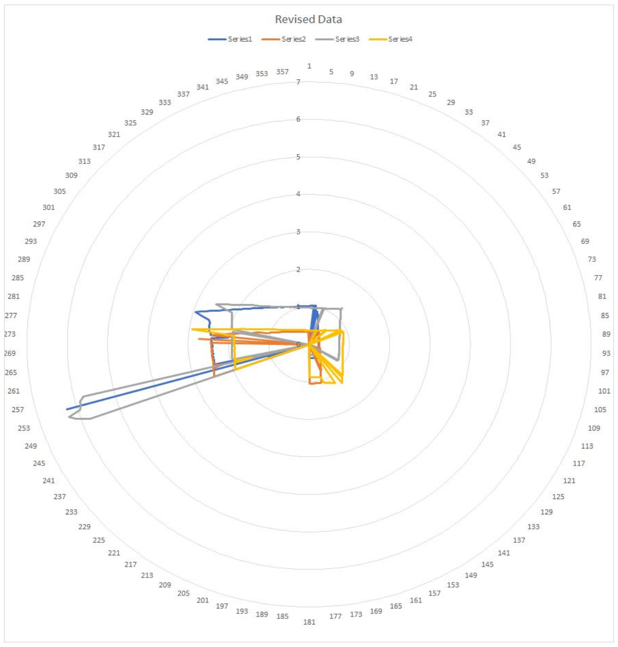


Figure 4: Second Lidar Initial Test result

Though the data can be slightly difficult to understand at first glance, the data captured actually shows the idea of what we are wanting to do. The overlap of the boxes for the series 1-4 represent the movement of the LiDAR and the different tests we conducted. The data points that are outliers are correct because the lidar was actually capturing the data from the open corner in the simulated box. These data points look useless, but they help to explain the idea that the lidar is not stopping at the imaginary fourth wall but going out into the free space and collecting data still.

Experiment 2:

Using the Lidar and Hector SLAM to create a 2-D model of an indoor environment

Following the first experiment it was determined that running the lidar off of a Raspberry Pi 3 was ineffective. In the future it will need to be run from a more powerful Pi 4 or from a laptop. For the ease of usability testing continued on a laptop using Ubuntu 18.04 and ROS Melodic. In order to create a 2-D map of an environment the rplidar was used to gather data and then Hector

SLAM was used on ROS to create a map of the test area. The photos below show the outside of the Engineering Research Center at the university and the map created on the 8th floor inside the building.

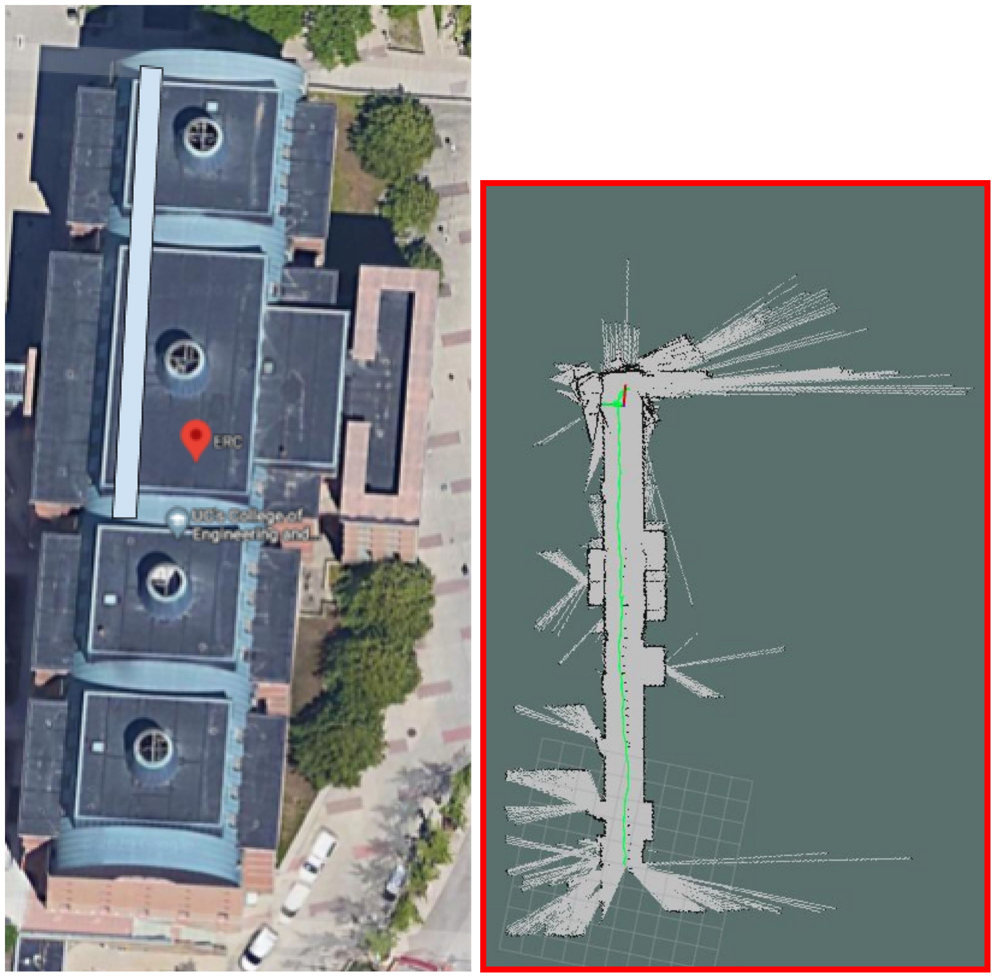


Figure 5: Google maps image and Lidar 2D map of Engineering Research Center

During this experiment there were a few trials. During each of the trials a person would enter or exit one of the rooms attached to the hallway being surveyed. This disturbance would break the map. The map would break because the interference would cause the software to lose track of its position in the environment. This test took place on the 8th floor of the ERC in the highlighted area on the image above. It can be also noted that having the lidar held by a human during the test induces too much error into the measurements as when a human walks their plane may change up or down and the lidar can be rotated severely damaging the map being made. In the future It will need to be placed on a cart and pushed eliminating any uncertainty.

Experiment 3:

Using the Intel RealSense camera and RTAB-Map package to create a 3-D model of an indoor environment

A wrapper for ROS of the RTAB-Map SLAM approach was used to create 3-D models of environments. The launch file included with the package was used to initiate the source files. Arguments were passed into the launch file to update the package on the ROS topics corresponding to the intended variable such as the depth topic produced by the camera.

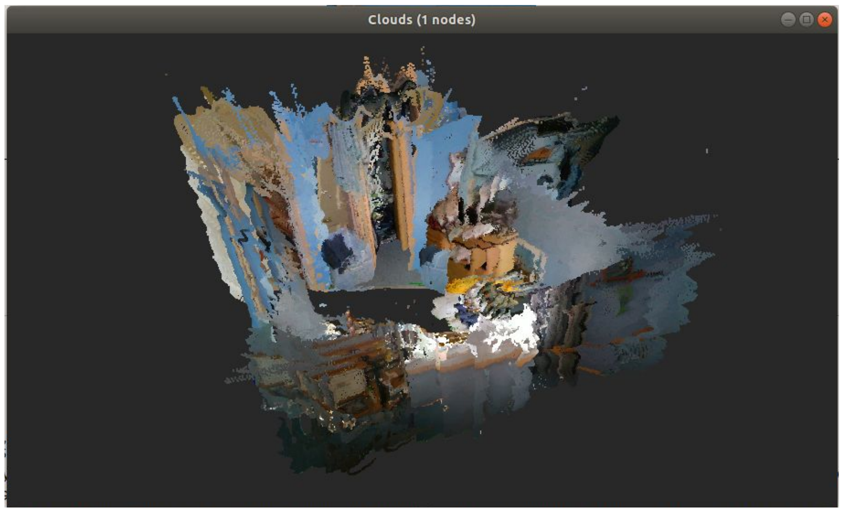


Figure 6: 3-D Model of an environment visualized with RTAB-Map Database Viewer

There were difficulties during experimentation such as the fragility of the camera's motion. If the camera was moved too quickly or bumped during mapping, the map creation would be thrown off balance and would need to be restarted. A human arm was used to steady the camera as it mapped the room resulting in natural unsteadiness when performing. This caused numerous restarts of mapping and also produced an uneven 3-D Model of the intended environment.

<u>Implementation</u>:

Package Sources:

RTAB-Map: https://github.com/introlab/rtabmap_ros

Librealsense (prerequisite for Realsense2_camera package): https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md

- May need to download source from development instead of master branch. Latest Ubuntu versions may not be compatible with master branch progress.

Realsense2_camera:
https://github.com/IntelRealSense/realsense-ros#installation-instructions

Commands:

1) roslaunch realsense2_camera rs_camera.launch align_depth:=true
    - "Realsense2_camera" ROS package needed
    - This launch file starts the RealSense camera to print to base ROS topics

2) roslaunch rtabmap_ros rtabmap.launch rtabmap_args:="--delete_db_on_start" depth_topic:=/camera/aligned_depth_to_color/image_raw rgb_topic:=/camera/color/image_raw camera_info_topic:=/camera/color/camera_info
    - "Rtabmap_ros" ROS package needed
    - Start command 1 in a separate terminal before this command

3) rtabmap-databaseViewer
    - Open "rtabmap.db" file inside software to visualize model
    - RTABMap package automatically creates "rtabmap.db" file in ".ros" folder in home folder (file is possibly elsewhere)

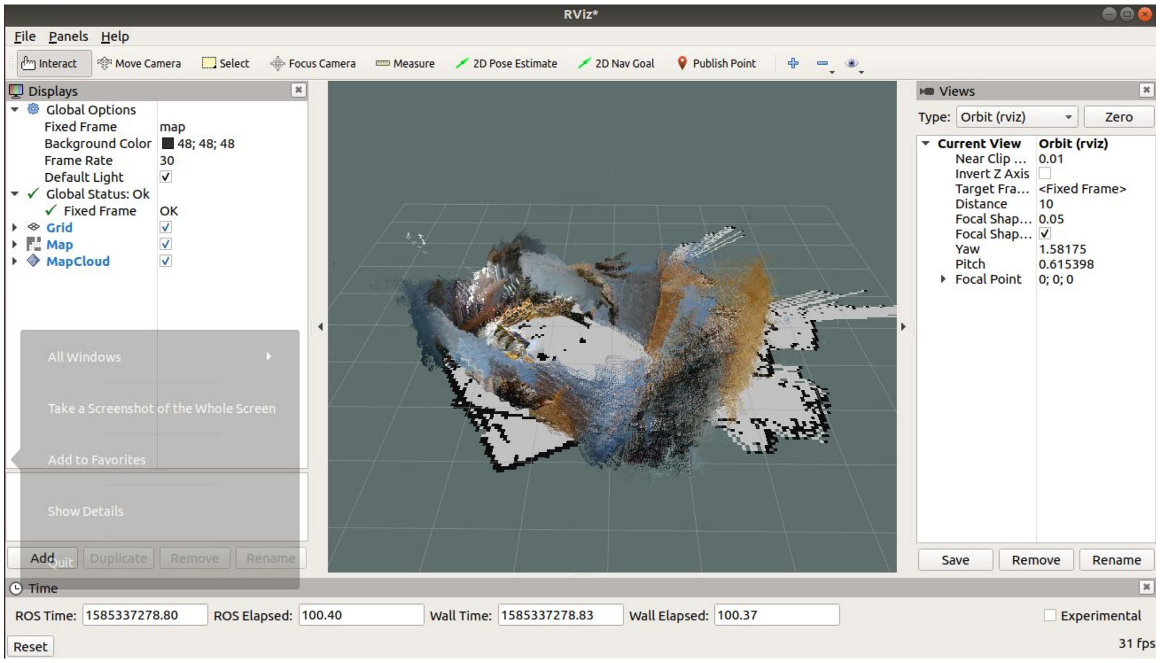
4) Using RViz

- “Rosrun rviz rviz” command to start rviz (may also be started by argument to rtabmap.launch file)
- Visualize cloud by adding “/rtabmap/MapData” topic to RViz

Experiment 4:

Using the Intel RealSense camera and RTAB-Map package with Lidar and Hector SLAM to create model with 2-D and 3-D representation

An attempt to simultaneously use both the RealSense camera and Lidar sensor was performed by utilizing both libraries used when performed individually. The RTAB-Map package allowed for inclusion of laser scan by passing appropriate arguments to the launch file. Arguments ,“subscribe_scan” and “icp_odometry”, caused the map generation to integrate new technologies instead of solely relying on RealSense measurements. The mapping package based itself on being installed inside a robot so manipulation of real-world coordinate frames of the two camera locations were required. A ROS package called tf maintains relationships between frames seen by all parts of the robot to maintain a realistic consistency of part locations. Since no robot body was used, only a human arm to hold both cameras, the camera/sensors were estimated at a distance of how far off the ground they were.



```
Global Options
  Fixed Frame        map
  Background Color   48; 48; 48
  Frame Rate         30
  Default Light      ✓
Global Status: Ok
  Fixed Frame        OK
Grid                 ✓
Map                  ✓
MapCloud             ✓
```

```
Type: Orbit (rviz)
Current View    Orbit (rviz)
  Near Clip ...   0.01
  Invert Z Axis
  Target Fra...   <Fixed Frame>
  Distance        10
  Focal Shap...   0.05
  Focal Shap...   ✓
  Yaw             1.58175
  Pitch           0.615398
  Focal Point     0; 0; 0
```

```
ROS Time: 1585337278.80   ROS Elapsed: 100.40   Wall Time: 1585337278.83   Wall Elapsed: 100.37   Experimental
31 fps
```

Figure 7: RViz visualization of combined RealSense and Lidar mapping

Issues were immediately present in the visualizations as the 3-D and 2-D models were rotated in different directions. This may have been fixed with additional tinkering of the coordinate frames of both cameras. Also, the lidar seemed to create more than one representation of the environment. Both packages used, RTAB-Map and Hector SLAM, were running simultaneously so this may have caused an overlap of output from the lidar sensor to its desired ROS topic creating different map representations.

<u>Implementation:</u>

Package Sources:

Lidar Hector Slam: https://github.com/NickL77/RPLidar_Hector_SLAM

RTAB-Map: https://github.com/introlab/rtabmap_ros

Librealsense (prerequisite for Realsense2_camera package):

https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md

- May need to download source from development instead of master branch. Latest Ubuntu versions may not be compatible with master branch progress.

Realsense2_camera:
https://github.com/IntelRealSense/realsense-ros#installation-instructions

-

Commands:

1) roslaunch realsense2_camera rs_camera.launch align_depth:=true
    - "Realsense2_camera" ROS package needed
    - This launch file starts the RealSense camera to print to base ROS topics

2) roslaunch rtabmap_ros rtabmap.launch rtabmap_args:="--delete_db_on_start" depth_topic:=/camera/aligned_depth_to_color/image_raw rgb_topic:=/camera/color/image_raw camera_info_topic:=/camera/color/camera_info subscribe_scan:="true" frame_id:="base_link" rtabmapviz:=false
    - "Rtabmap_ros" ROS package needed
    - Start command 1 in a separate terminal before this command

3) sudo chmod 666 /dev/ttyUSB0
    - Needed to give permission for LIDAR use when plugged in with USB

4) roslaunch hector_slam_launch mapTest.launch
    - "mapTest.launch" is a custom launch file based on the launch file given in the "hector_mapping" package launch folder called "mapping_default.launch". Additional commands were added to the given launch file (See Below) and then saved as a different custom file inside the "hector_slam_launch" package folder.

5) rosrun tf view_frames
    - This command prints the tf tree of the robot system. Tf is a ROS library that controls the different coordinate frames or positions of each robot part (RealSense camera, lidar, robot frame, etc.)
    - Commands added below to a launch file tell the mapping package where the RealSense Camera is relative to the rest of the robot.

Commands added to end of “mapping_default.launch” file in Hector_mapping package:

```
<node pkg="tf" type="static_transform_publisher"  name="base_to_color"
args="0 0 0.74 0 0 -1.5707963 /base_link /camera_color_optical_frame 100" />

<node pkg="tf" type="static_transform_publisher" name="color_to_depth"
args="-0.1325  -0.1975  0.0  -1.570796327  0.0  0.0  /camera_color_optical_frame
/camera_depth_optical_frame 100" />
```

Optical Flow

Optical flow is a class of algorithms that track the positions of objects in a series of frames through time. For example, an optical flow algorithm can be used to trace the movement of a particular object in a video. It was originally thought that optical flow could be used to determine the distance between camera and an object. However, based on current research it is only capable of calculating relative distance.

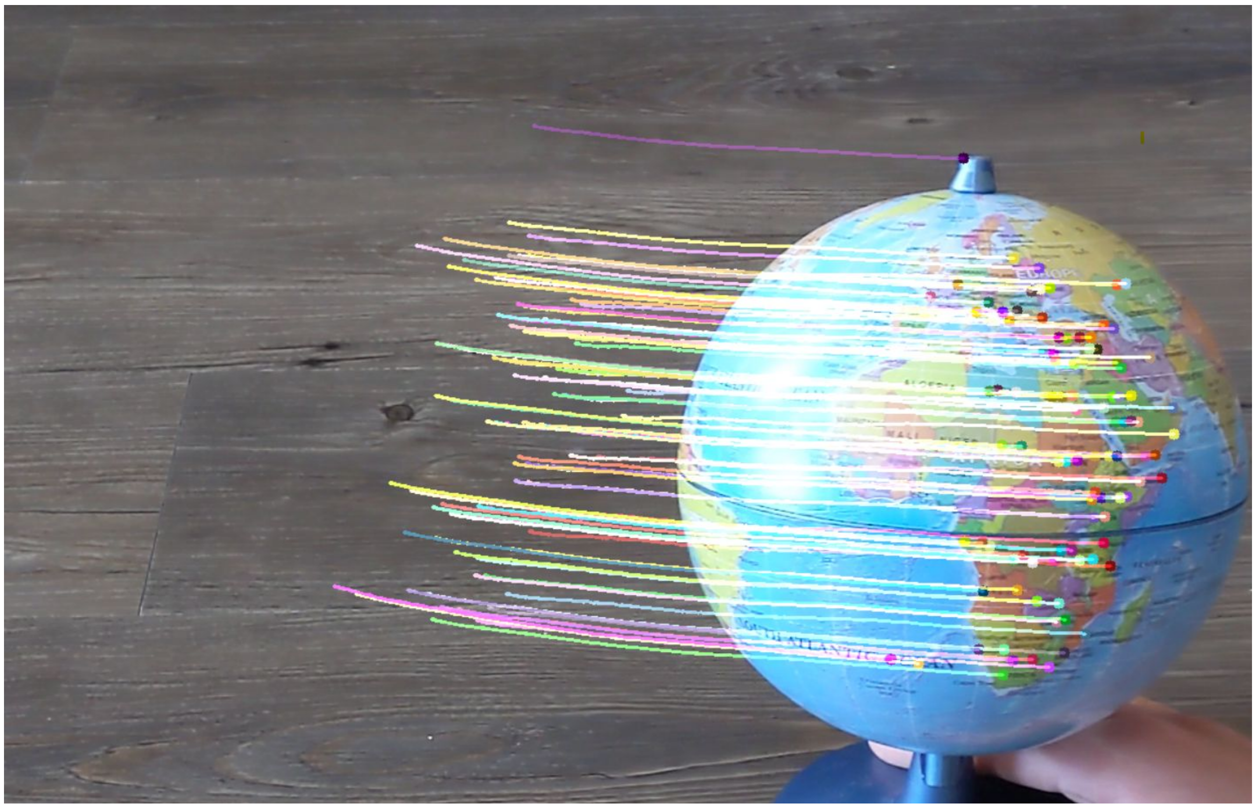


Figure 8: OpenCV Optical Flow Tracking Points on Globe.

Intel Realsense

The Intel Realsense is a series of camera systems that offer a wide variety of features. The Realsense model this project is using is the D435i. The D435i is optimized for 3d modeling and depth sensing. The D435i has two depth sensors, an rgb camera, and an IR projector. It has a large field of view and can be more accurate when it is mounted on a device and is moving fast when compared to other Realsense Models. The D435i has an added inertial measurement unit (IMU) which can track where the unit has moved through space. These features

Experiment 5:

Cloud Video Streaming Platform

This project attempts to create a platform that can take a live video input and output the real-time livestream to a web server for viewing. A private platform to stream videos gives increased freedom with areas such as video encoding, resolution, object recognition, and playback. A Raspberry Pi was used originally to capture live video with a video capture card and to send the stream to a web server hosted on Google Cloud platform. To test and improve the system more efficiently, a computer screen was streamed to the cloud platform directly (No Raspberry Pi or capture card was involved). Below is a list showing the technologies used in the platform divided into sections (front-end, database, and server-end).

Technologies:

- Web Application
    - Google Application Engine
    - Flask Framework (Python language)
    - Video.js (Javascript language library) for playing videos in HLS format on

webpage

- Database
    - Google Cloud Storage Bucket
- Streaming Server
    - FFmpeg (capturing and streaming video)
        - HLS format (.m3u8 playlist file with .ts video chunks)
    - Server-side scripts
        - External Libraries: Watchdog, Google Cloud Storage

Implementation:

Flowchart

1 →	2 →	3 →	4 →
RPi runs python server-side script to initiate video streaming, file organizing, uploading, and playback processes.	FFmpeg software tool to capture video in HLS format .	Script uploads video chunks when created to Google Storage.	Video element on web page reads master video playlist file from Google Storage.

5 →	6
Server stops the stream. Full streamed video uploaded to cloud storage for user playback.	Appropriate files reorganized and/or removed to prepare for future stream.

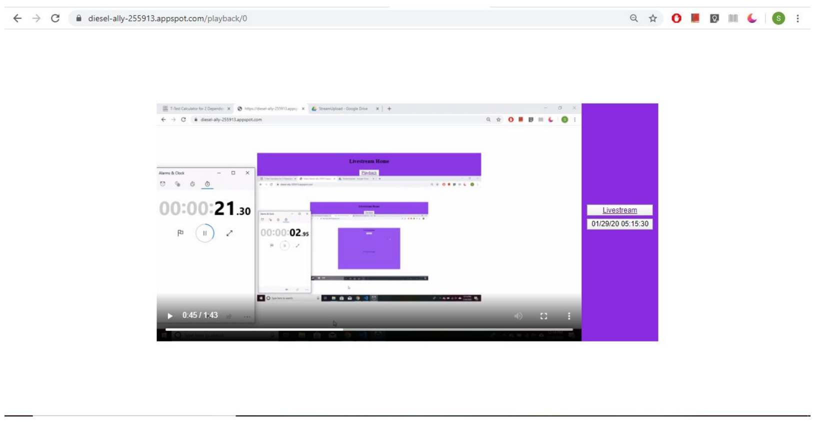


Figure 9: Playback Video Web Page

Difficulties:

The platform was left in a completed state but had imperfections affecting latency and video playback. Using a timer on the server, the streaming video on the web application was found to be around 12 to 20 seconds behind the real-time video being captured. This is an acceptable delay but would be ineffective in scenarios desiring immediate responses. Playback video was also inconsistent, dysfunctional segments often were present in archived videos. This dysfunction was in the form of specific video chunks (near 5 seconds) being blacked out.

Future Work:

- Research into using other web streaming protocols (WebRTC?) instead of HLS
- Research deeper into HLS protocol, using the other playlist (.m3u8) modes such as Live, VOD, etc.
- Try out different video chunk (.ts) times and number of total chunks
- Research into ffmpeg video encoding in the cloud. Send video stream from local computer to cloud server using RTMP.

CONCLUSION

A Intel RealSense camera, LIDAR sensor, and web streaming platform were all finished to a level of acceptability but a cohesive system was left as future work. The RealSense camera and LIDAR sensor both created 3-D and 2-D models, respectively, using the mapping approaches of RTAB-Map and Hector SLAM. Problems arose during linkage of both parties, development of a physical robot frame and additional programming of utilized ROS packages is needed for advancement. A web streaming platform was established but wasn't developed further to integrate with the mapping created inside the ROS framework. Visualizations were left to be viewed on the server or as exports using tools such as RViz. Exploration with VR was also not reached and was left for future development.

References

Center, N. O. and A. A. N. C. S. (2012). Lidar 101 : An Introduction to Lidar Technology , Data , and Applications. *NOAA Coastal Services Center*. https://doi.org/10.5194/isprsarchives-XL-7-W3-1257-2015

Dong, H. L., Kyoung, M. L., & Sang, U. L. (2008). Fusion of lidar and imagery for reliable building extraction. *Photogrammetric Engineering and Remote Sensing*.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping (SLAM): part I The Essential Algorithms. *Robotics & Automation Magazine*. https://doi.org/10.1109/MRA.2006.1638022

Kashani, A. G., Olsen, M. J., Parrish, C. E., & Wilson, N. (2015). A review of LIDAR radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors (Switzerland)*. https://doi.org/10.3390/s151128099

Mecanum Wheels, Drives and Source Code. (n.d.). Retrieved October 15, 2019, from https://www.roboteq.com/index.php/applications/applications-blog/entry/driving-mecanum-wheels-omnidirectional-robots

Portugal, D., Araújo, A., & Couceiro, M. S. (2020). A Guide for 3D Mapping with Low-Cost Sensors Using ROS. In *Studies in Computational Intelligence* (Vol. 831, pp. 3–23). https://doi.org/10.1007/978-3-030-20190-6_1

Rajesh Desai, P., Nikhil Desai, P., Deepak Ajmera, K., & Mehta, K. (2014). A Review Paper on Oculus Rift-A Virtual Reality Headset. *International Journal of Engineering Trends and Technology*. https://doi.org/10.14445/22315381/ijett-v13p237

Zhang, J., & Singh, S. (2015). *LOAM: Lidar Odometry and Mapping in Real-time*. https://doi.org/10.15607/rss.2014.x.007