The Exploration of a Partially Self-Navigating Golf Cart

Courtney Nelson
Department of Computer Science
Occidental College
cnelson2@oxy.edu

2022-02-02

Abstract

The project aimed to explore the computer science and engineering behind indoor autonomous vehicles. The focus of the project was an upgrade to the Occidental College's Engineering Club's solar powered golf cart that would make progress towards having the golf cart to navigate simple indoor spaces autonomously. The purpose of this project was to take the first step towards creating an autonomous sustainable vehicle that can transport students around the campus. A principle component of autonomous navigation is the utilization of sensors for mapping. The HDL-32 LiDAR from Velodyne was used for mapping with primary processing enabled by the Point Cloud Library. With the point cloud library fast triangulation of surfaces was employed for the mapping of a hallway and key point identification was used to process a simple room. For hardware, it was discovered that the best way to communicate with the solar powered golf-cart would be to connect to the motor controllers that control forward and backward motion through the solenoid using a raspberry pi. For steering, without an affordable way to connect with the front wheels, it was decided that attaching a motor to the steering column that directly communicates with the raspberry pi would be the most feasible option. The results of the project were the successful generation of point clouds and the completion of the first steps of point cloud processing, in addition to the development of a feasible plan for the computer hardware interface and further point cloud processing.

1 Introduction

The world is on the edge of an exciting new technological revolution. This revolution is the change from human operated mobile robots to self navigating systems. This project proposes an opportunity for Occidental's campus to utilize this emerging technology and knowledge. Although the primary focus of autonomous vehicles is achieving a safe self-driving automobile, there are many practical and useful results of the operation of self-navigating robots in indoor or less dynamic spaces. When a robot is able to guide itself through a task, mundane processes can be automated: liberating already busy schedules from the chores that arise in daily life and work. For these reasons and many others, indoor self-navigating robots are currently on the verge of an exciting upheaval and are likely to play an important role in our future. Whether by assisting us in taking out the trash, synergistically working together in factories, or providing assistance to people with disabilities, self navigating robots will have an active role in our lives. One future application of an extension of the proposed project, to achieve partial autonomous navigation of a solar-powered golf cart, is to provide rides between buildings for students and faculty with injuries or disabilities. This could help make Occidental's campus more accessible to all students.

2 Background

2.1 Localization

A key component of designing an automated mobile robot is establishing its position and orientation as accurately as possible. The main method for accomplishing this uses landmarks. With a sufficient number of landmarks the robot is able to accurately estimate its position and orientation. Igarashi et al. (2001) proposes a solution that accounts for a diverse number of constraints on path-planning and navigation. Igarashi (2001) breaks up the challenge of robot self-navigation into two main problems, estimation of the robot's location and action decision. For determining the robot's location an approximation of Markov localization was used. Markov localization is a probabilistic algorithm that estimates the probability of key locations being occupied by a specified object from sensor data. Igarashi et al. (2001) used the minimization of an objective function, a mathematical representation of preferable motion, that depends on data, constraints, and predictions to determine location. Their navigation algorithm is structured in the order of sensing the enviroment, estimating the robot's location, planning a path, taking action, and then checking if the goal was achieved. If the goal was achieved the robot is instructed to stop. If the goal was not achieved then the process is repeated. A 2D localization function for the robot is described in Equation 1.

$$\begin{aligned} E_1(R_t\alpha_t; R_t^{obs}, r_{t-1}, v_{t-1}) = a_1E_{data} \\ + a_2E_{cnst} + a_3E_{prdt} \end{aligned} \quad (1)$$

Where E_1 is the objective function of the robots position, r_1, a direction in two dimensions. The constants, a_1,a_2, a_3 are the weights of each term. For example, E_{data} is the disparity, square of the difference, between the sensed range data at the time t and the map, data pattern, observed at the location r_t and direction α_t. This objective function compares the input data with an established map and is one of the methods that can be used to determine the robots location. It is similar to the landmark method because it compares its inputted data with the landmarks established in its map. This method is a more sophisticated version of using sensors such as ultra-sonic range finders to triangulate a location from landmarks. In addition to determining its location the robot needs to be able to map the rest of its environment. This will be discussed in the next section.

2.2 Mapping

When there is not a prepared map of the area that the robot will be inhabiting it is important that the robot can build its own map of the environment. The main methods of mapping an environment include geometric mapping, grid mapping, and maps generated through machine learning. The following methods of generating maps are discussed below.

2.2.1 Geometric Mapping

Geometric maps use geometric objects to identify obstructions in the environment. Austin et al. used primitive geometric objects for identifying constraints such as line segments, arc segments, cubes, and planes. Primitive objects can be described by a number of parameters including, shape, size, clearance distance and the error in each measurement. The goal of Austin et al.'s research was to minimize the error for accurate mapping. The advantage of geometric maps is that they require less memory and therefore less time to compute a path. Additionally, geometric maps scale well from two to three dimensions. Although it does not scale without issues. One main issue with three-dimensional modeling is primitive object orientation causing errors in object identification.

2.2.2 Grid Mapping

A basic mode of mapping is dividing a space in a grid pattern. Each element in the grid will contain a value. For example, a 1 or 0. If an object exists in the cell it is assigned a 1. If an object does not exist in the cell it is assigned a 0. One issue with this use of a binary system is that sensors are not always accurate, and it may be difficult to identify whether a cell contains an object. Therefore, instead of a assigning an absolute value to each cell a probability can be assigned. Now, the path of the robot can be determined by adjacent cells with the lowest probability of being occupied by an object. To generate a grid map efficiently you want to explore the frontier of the space while avoiding objects. Occupying cells with a low probability containing an object that are at the edge of the map allows for efficient exploration of the frontiers. Priority in the frontier algorithm will be given to cells, with a low probability of containing an object that is adjacent to the largest number of unknown cells. Once the environment is mapped the robot should be able to efficiently navigate the space. One major limitation of the method is that it is not effective in dynamic environments.

2.2.3 Machine Learning for Mapping

Machine learning can be used to generate priorities when creating a map. For example, Igarashi et al. (2002) formulates a path by maximizing a discrete optimization problem for a given time step. To optimize the path the objective function has a term that accounts for the final goal position, smoothness of path, and possibility of collision. Igarashi et al. (2002) proposes that reinforcement learning is a useful tool for adjusting the weight factor of each term. Igarashi et al. (2002) applied Williams' learning algorithm, episodic REINFORCE, to establish a learning rule that would optimize the weights of the terms of the objective function.

A challenge of this method is that the optimal path may keep the object in its current position. Since the optimization function is looking to minimize the objective function the mobile robot can get trapped in local minima. Therefore, when minimizing the objective function continues to point towards the same location, methods need to be established to move forward along the path. Igarashi et al. (2002) suggests inserting attractive sub goals to the objective function to guide the mobile robot out of regions of local minima. Another approach proposed by Oshiro and Kurata is a self-organizing map for robot navigation data from visual sensors. The model us based on Kohonen's self-organization model of a cortex. Kohonen's model uses a two-dimensional array of neuron-like feature detection units for mapping. During the learning phase the robot is set to move randomly through the room. Then the position and direction are recorded to map two layers, one of position and the other of direction. In addition, Abhishek Roy and Mathew Mithra Noel successfully developed a line-following robot using a hybrid artificial neural network. The robot that can smoothly follow a randomly curved path.

Klingspor et al. applied machine learning to sensor data and robot action concepts. Their strategy was to break up the complex task into clear learning steps that could be combined. The learning steps were broken down into basic features, sensor features, group sensor features perceptual features and action features. Klingspor et al. used data collected by sensors on a robot moving in a known path to start training the algorithm.

Using machine learning to map a space requires random movement throughout the space. Although this may be a strong method when there time and space to train the algorithm, machine learning is limited in its usability in complex environments or in environments where random movement is hazardous.

2.2.4 Corner and Edge Detection

Another method of generating maps is explicit corner and edge detection. Stereo vision can match the edges of images on a pixel-by-pixel basis. This is useful for stationary imaging, but can be problematic when the motion of the camera is unknown. With a tracked edge connectivity, supplemented with some 3D locations of junctions and discontinuities a wire-frame representation of the image can be constructed and used to map the 3D space. Another method of corner and edge detection uses changes in intensity to determine features in the map. The algorithm monitors changes in intensity in small changes in location of a local window. If the position of the window is slightly adjusted with no edge in the window, then there should only be small changes in the intensity of the pixels. If the window contains an edge, then small shifts in the window places will result in small changes of intensity parallel to the edge and large changes in intensity perpendicular to the edge. If the window contains a point or a corner and shifts will cause a large change in intensity. Therefore a localized window that scans an image and be used to determine the locations of corners and edges in the environment. This can then be used to generate paths.

2.3 Target Tracking

Jia et al. proposes a data fusion-based algorithm to identify and track moving objects to optimize self-navigation. Since both the tracked objects and robots are in motion determining position and trajectory is difficult. Jia et al. uses the robot's motion-sensors to determine the motion of the camera and then applies an optical flow vector field, color features, and differences in visual features in the cameras of a stereo vision system to determine the relative positions of the moving objects and robot. Further, Jia et al. uses an extended Kalman filter for three-dimensional target tracking. A Kalman filter is an algorithm that uses a sensor measurements over time to minimize statistical noise.11 To accurately describe a target object in 3D knowledge of its size, location, velocity and distance from robot are needed. Traditional target-tracking algorithms for autonomous robots modify the environmental conditions by adding sensors that the robot can utilize for tracking. This is not an option for robots looking to traverse new environments or outdoor environments. For accurate tracking in most environments all required sensors need to be on the robot. Jai et al. uses optical flow vectors to represent the motion off target pixels as a sequence of images progresses. Optical flow vectors combined with other visual features of target objects are a means of reliably tracking an object system in three dimensions. Additionally, optical flow vectors can be used as to measure the transformation of non-rigid target bodies. Employing optical flow-based velocity compensation to target monitoring was initially developed by Chang et al.

2.4 Sensors

Self-navigating robots often use sensors such as ulta-sonar, range finder, and visual sensors to estimate its position. This process is called odometry. It is the process of using data from motion sensor to estimate their position over time. Leading sensors that supply data necessary to build maps will be discussed in the sections below.

2.4.1 Vision Based Maps

A visually guided robot that can navigate an environment, like other methods, by planning paths and constructing maps. A stereo vision system allows for the production of 3D images with accurate depth. Stero-vision is the extraction of 3D features by comparing images of the same location from two vantage points. Murray et al. shows that robots guided by stero-vision systems are a viable alternative to leading methods such as sonar and rang finders. One major limitation is that the robot's speed is limited by the time it takes to compute paths from the collected images. Stero-vision systems are useful for dynamic and complicated environments. There large field of sight and continuous mapping makes them a strong candidate for the sensors used in autonomous navigation.

An alternative is an ocellus camera. Using an ocellus camera Hayashi et al. implemented a navigation system that uses feature points to recognize significant obstructions. Hayashi et al. demonstrated that using only the ocellus camera the mobile robot was able to recognize and navigate around an ordinary room. Further, Jaung et al. developed a line-following system for self-navigating robots using image processing. The robot has a camera attachment that periodically captures images. The images are then processed, and the line is extracted using a high-speed rectangular search method.

2.4.2 Internal Sensors

Odometry can be unreliable if the sensor is internal to the robot. A common internal sensor used to measure change in position is an encoder. This device encounters issues when the robot slips and the encoder does not register as a distance traveled. This results in inaccurate self-location. Encoders and other internal sensors like gyroscopes cannot be used as the lone navigating system because they cannot handle complex or dynamic tasks and they are prone to error.

2.4.3 External Sensors

Unlike visual sensors, other external sensors including sonar and range finders have difficulty dealing with outdoor or dynamic environments. The sensors are likely to be frequently obstructed by moving objects. Visual self-location is advantageous in highly dynamic environments. It is a better option because it does not require known objects to reference its position from and the range of mapping is not limited by direct line of sight. Ultrasonic sensor are widely used for robots needing to avoid obstacles because they are simpler than many other methods and when used in parallel with other sensors they can be very effective. For example Choi combines the visual image from the CCD camera with the ultrasonic sensor to determine the optimal path.

Another alternative is a LIDAR (Light Detection and Ranging) sensor. LIDAR is commonly used as a primary sensor to lead self driving cars. Specifically, a LIDAR sensor send out a laser pulse and measures how long does it takes to reflect of a surface and return to the sensor. Most lidars use a time of flight calculation between when the pulse of the laser is sent and when it is reflected and then received. LIDARs can be used for 3D maps by sensing a depth of each obstruction in surrounding the device. Scanning LIDARs have a 360 degree range. They are optimal for generating maps and for use on self-driving vehicles. The biggest deterrent for the use of scanning LIDAR sensors is their high price point. In contrast, ingle point lidars are more affordable but they only can scan a single point. The LIDAR sensor generates a point cloud, which allows for the mapping of 3D field of view.

Additionally, Zunaidi et al used multiple internal sensors to improve the relative position measurements of individual sensors. The mobile robot was equipped with an encoder, gyroscope and accelerometer. By cross checking the information from the encoder with that from the gyroscope and accelerometer they were able to increase their accuracy. Although this method is only reliable for use in laboratory or indoor conditions since a bump or small disturbance in the wheels can cause an unpredictable error in the encoder's count. The optimal sensor for this application is a LIDAR device.

3 Ethical Considerations

In addition to technical challenges, there are also societal challenges in introducing self-navigating robots to daily life. These challenges include job loss for workers whose jobs are based on manual labor or operating machinery. This includes jobs displaced by self-navigating robots that are not self-driving cars. Further, this added mobility comes with large amounts of risk. Including determining liability if someone is injured. As this technology is developed it is important that society is legally and culturally preparing itself for the changes that come with new technology. Although there are many challenges in implementing self-navigation for robots, it is an important research area to pursue. Self navigation has the potential to transform many aspects of work and society for the better.

4 Methods

4.1 Overview

4.2 Materials

- Solar-Powered Golf Cart
- Motor to rotate steering column
- HDL-32E LiDAR by Velodyne
- Raspberry Pi 3 B+
- Laptop
- Power Supply
- Testing area with simple obstacles
- Motor for Steering Column (Future Work)

4.3 Hardware

4.3.1 Golf Cart Drive Train

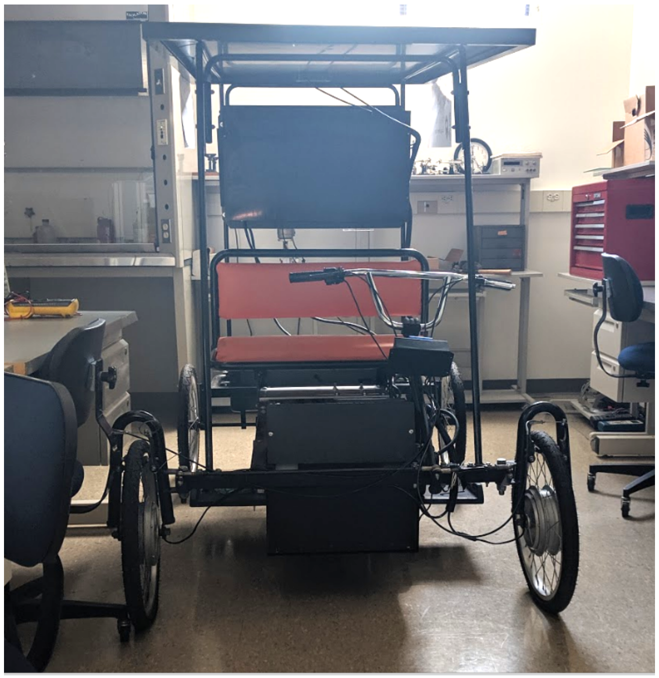

Figure 1: Solar powered golf cart that was built by Occidental College's engineering club and the primary vehicle for this project.

The project was based on a solar powered ”golf cart”, shown in Figure 1, built by the engineering club at Occidental College nine years ago. The golf-cart was built out of a re-purposed electric bike and recycled car and bike parts. Since the drive train of the golf-cart was primarily developed from an electric bike hardware with primary power derived from solar powers it is significantly limited in its power and its hardware compatibility. The golf-cart was equipped with three motor controllers. One motor controller was in use on the two rear wheels to control speed. In addition, there was a motor controller on each front wheel for steering that were not currently in use in favor of manual steering.

The Crystalyte CT4825S motor controllers connected to the front wheels are nine years old and had little manufacturing information. After discussions from the manufacturer, it was decided that the circuitry of the motor controllers were incompatible with any standard micro-computer. The result of this challenge is that the micro-computer will not be able to directly operate the individual wheels. The proposed alternative is to attach a motor to the steering column and drive the motor using the micro-computer. The challenge with this method is implementation and speed control. This does not solve the challenge of communicating with the thrust. The plan to communicate with the thrust includes wiring the micro-computer directly to the servo that is attached to both back wheels. Therefore the speed would be controlled through the rear wheels and the direction would be controlled by rotating the steering column.

4.3.2 On-board Sensor

The mapping sensor in use for the project is a Light Detection and Ranging Sensor (LiDAR). A LiDAR measures the distance and intensity of objects in its line of sight by illuminating a point in space with laser light and measuring the time elapsed and the difference in intensity between the outgoing laser light and the reflected incoming light. The collection of points can then be plotted in three dimensional space that denotes depth and uses color to assign intensity. The collection of the data points with depth and intensity is a point cloud. In conjunction with inertial data such as relative location and velocity a full working map of the space can be developed and paths with low probability of collisions can be constructed.

The LiDAR used for this project is the HDL-32E. Its information packets are 1206 byte payloads which consist of twelve 100 byte records followed by 4 bytes of a 32 bit unsigned integer time stamp in microseconds and 2 blank bytes. For each 100 byte record there is one identifier and the beginning of the record followed by a two byte segment giving the rotational position followed by 32 sets of 3-bytes which report information about each laser fired from the sensor. The next two bits report the distances at which the light is reflected to the nearest .2 cm. The remaining byte in the 100 byte sequence encodes the intensity on a scale of 0 - 255. The laser safe for vision with a 903nm wavelength and a 70m measurement range. The vertical field of vision is +10.67 degrees to -30.67 degrees with a 360 degree horizontal field of view (VeloView 2015). The sensor requires 12V so the golf cart will need to run the sensor off of an additional 12V battery rather than using the solar panels. The output of the sensor is 700,000 points/second and therefore requires significant computational power for on board data storage and processing. The most powerful reasonably priced micro-computer suitable for this project is the Raspberry Pi 3 Model B+, which will be discussed in the next section.

4.3.3 Computing Hardware

For computing hardware the main decision was between an Arduino and a Raspberry Pi. The Raspberry Pi was chosen because it has more computing power than an Aurduino, which is necessary for using the LiDAR. At about 1,000 rotations per minute the LiDAR is in taking a significant amount of data per second as discussed above. The Raspberry Pi 3 Model B+ is a 1.4GHz 64-bit quad-core processor. Although the Raspberry Pi 3 Model B+ is one of the most powerful micro-computer it has still had computational difficulty when trying to install software. It looses its display capabilities and has screen freezing while running at maximum processing power. Additionally, the time it takes to download programs and packages was much longer by hours that the installation on a laptop. Further the installation of useful programs for the display and processing of point clouds are experimental at best on Raspian, the operating system for raspberry pi that is modeled off of Debian and is a Linux based operating system. For Veloview one of the primary programs used for this project the install of VeloView onto Raspian has not been completed successfully and documented. Forums discussing the implementation have few active participants and non signs at this time of success. Therefore to use VeloView for introductory display and processing a laptop was necessary. In future work the installation of VeloView, Point Cloud Library, and the Robot Operating System would be a strong next step towards onboard data processing from the LiDAR.

4.4 Software

4.4.1 Veloview Implementation

VeloView, preforms real-time visualization and processing for Velodyne's HDL sensors. VeloView was used for point cloud visualization, map construction and debugging. VeloView displays distance measurements from the LiDARs point cloud data and supports color mapping. VeloView was installed using a binary package on both Windows and Linux. Then re-installed from source onto Linux operating to get access to processing capabilities that are under development.

To access the experimental processing algorithms through VeloView requires the a manual build with the following dependencies for Linux machines:

- build-essential
- cmake
- git
- flex
- byacc
- python-minimal
- python2.7-dev
- libxext-dev
- libxt-dev
- libbz2-dev
- zlib1g-dev
- freeglut3-dev
- pkg-config

With careful implementation, the above packages were successfully installed. This list lacked one major component Qt5, a framework for software development. The challenge has been in building Qt for the raspberry pi and configuring for the build. There are resources for installing the current Qt onto a raspberry pi but the process has many dependencies and intricacies. The source version of VeloView utilizes the Point Cloud Library (PCL) though a plug. PCL a well known open source C++ library with tools to work with point clouds. The PCL plugin allows users to generate filters to process there data that are specifically relevant to there project. Examples of applications where the PCL Plugins are used is robot-arm guiding, multi-sensor autonomous calibrations, and the automatic detection of objects.

To select the operating system best suited for point cloud processing a dual booted laptop with Linux and Windows was used. Unfortunately both operating systems have issues with the dependencies for PCL. On windows the install of PCL from a pre-made binary for visual studio 2017 was possible. PCL compiled from a binary was able to run basic commands and visualize data but the grabber for Lidar Data is only compatible with a manual build of the newest PCL library. After successfully installing PCL the PCL's dependencies on Linux PCL from source was able to be install with the majority of its features enabled.

4.4.2 Robot Operating System

Additionally ROS the Robot Operating System was installed as a alternative method to do the SLAM processing. Over the duration of the project having point clouds displayed in ROS was achieved, but there were challenges in getting ROS and PCL to communicate with each other.

4.5 Point Cloud Processing

4.5.1 Simultaneous Localization and Mapping

The first step in VeloView's Simultaneous Localization and Mapping (SLAM) algorithm is key point extraction. During key point extraction the algorithm identifies discontinues in boarders, edges, and corners, and continuities in local regions that can be approximated as surfaces. Once all of the boarders, edges, corners, and surfaces have been detected primitive geometric models are constructed. These models in conjunction with GPS or inertial information then fed into a Kalman Filter to more accurately position each point edge and plane of the point cloud. Kalman filters intake an original estimate for the location of a singular point in the point cloud and its uncertainty then compare the prediction to the measurement from the LiDAR and its experimental and instrumental uncertainties to update the position of both the LiDAR and the location of objects in the space. For implementation through an algorithm the main mathematical model uses matrices. The mathematical model of the matrix modifications are included below.

The initial estimate for the position and momentum of each point in the point cloud denoted as X_0 and P_0 where X_0 is the six dimensional state matrix which contains information on the position of the point in three dimension and the velocity of the point in the dimensions. In classical mechanics every object can be uniquely described by its position and momentum. Therefore the X_0 matrix fully describes the necessary components of the state of each particle. P_0 is process covariance matrix, which holds information about the uncertainty of the state of each point. A covariance matrix gives the covariance between pairs of its elements. The variation in the values of the elements gives the uncertainty of the state of a given point. The initial prediction of the state of a point and its uncertainty are then modified by the following equations to update there values to correspond to the current location of the sensor. The modification to the matrices are shown in the following equation.

$$X_k = AX_{k-1} + BU_k + W_k \tag{2}$$

Where X_k resulting state matrix after manipulation, X_{k-1} denotes the state matrix before manipulation, U_k is the control variable matrix which uses GPS or inertial data to quantify the forces acting on the given point in the point cloud and W_k is the predicted noise covariance matrix which measure the error in the manipulation of the previous matrix. Further A and B are adaptation matrices that assist in the conversion process from the previous approximation to the new approximation.

The updated process covarinace P_k is given

$$P_k = AP_{k-1}A^T + Q_k \tag{3}$$

Where P_{k-1} is the previous matrix, Q_k is the process noise covarience matrix, and A is the adaptation matrix that assists in the conversion process. After updating the predicted state it is ready to be used in conjunction with the measured value to make a new estimate of the state of the point in the point cloud. The next step in the Kalman filter process is to calculate the Kalman gain (K) which assigns a weight to the prediction versus the measured value. The Kalman gain is shown below:

$$K = \frac{P_k H}{HP_k H^T + R} \tag{4}$$

Where H is an adaptation matrix and R is the sensor noise covariance matrix which is the error in the measurement.

The Kalman Gain in conjunction with the measured value Y and the predicted value X_{k_p} are used to calculate the new estimate of the state matrix of the point particle.

$$X_k = X_{k_p} + K[Y - HX_k] \tag{5}$$

This process continues for each particle in the point cloud until the process covarience matrix has uncertainty less than the uncertainty threshold set by the program.

This process is highly effective and can be implemented in real time or on prerecorded data in VeloView. There are run time issues with the SLAM algorithm currently used in VeloView which resulted in the SLAM processing for this project needing to be done offline. For this project the hardware included a HDL-32E LiDAR from VeloView, but didn't not include the accompanying GPS unit. Therefore a SLAM processing of the data from the LiDAR unit through VeloView possible was not possible for this project. Therefore as a test for future work the SLAM program was run on existing data from Velodyne.

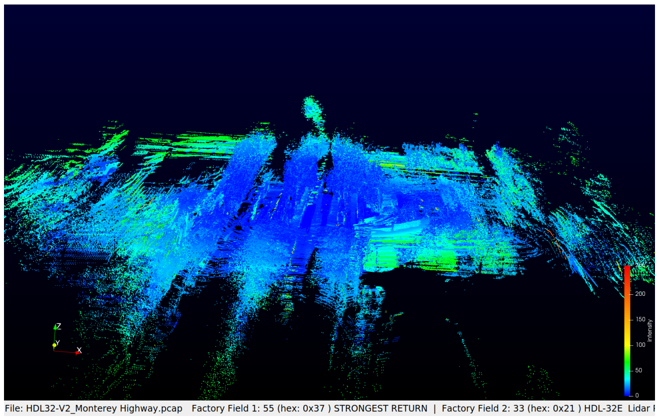
200
150
100
50
0
Intensity
File: HDL32-V2_Monterey Highway.pcap Factory Field 1: 55 (hex: 0x37) STRONGEST RETURN | Factory Field 2: 33 (hex: 0x21) HDL-32E Lidar I

Figure 2: SLAM on data collected from an HDL-32E LiDAR from Velodyne of car driving through a park with the LiDAR attached to the hood of the car. The color corresponds to the intensity of the reflected light.

4.5.2 Point Cloud Library

The second type of data processing done on the point clouds for this project used Point Cloud Library directly. The Fast Triangulation of Unordered Points algorithm by PCL was used with some modifications on the data from the LiDAR. The algorithm works by growing a mesh between points with similar characteristics. The first step of the algorithm identifies key points as discussed in the previous subsection. The result of key point indentification on a room is shown in Figure 3.

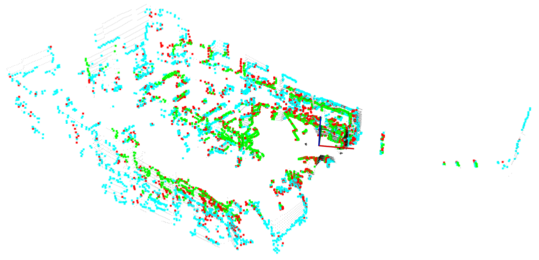

Figure 3: Key point identification using the point cloud library of a room captured by the HDL-32E LiDAR used in this project.

The algorithm uses a covariance matrix of the nearest neighbor points to the point in question to look for edges and surfaces in the variance in the x, y, and z dimensions between nearby points. To identify the neighboring points a k-dtree is created from the data. This is a data structure that finds the median in each dimension of the data and splits that data about the median. This continues in each dimension until the tree is fully formed. An issue with this method of data organization is that nearby points can end up on separate branches of the tree is they are near the median value where the tree is split. Therefore when forming a mesh created by the surface normals of adjacent points some data that should be linked as a surface are not connected. In the case where the density of the point cloud changes rapidly this can lead to difficulty in triangulating a surface. The results of the modified Fast Triangulation of Unordered Points algorithm are shown in Figure 4.

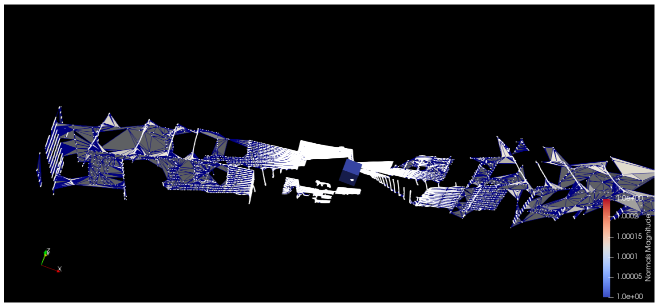

Figure 4: Fast Triangulation of Unordered Points algorithm from the point cloud library run on data of a room captured by the HDL-32E LiDAR used in this project.

5 Results

This project resulted in the successful point-cloud data collection from the HDL-32E LiDAR. In addition to the demonstration of two methods of processing point clouds. The first, a Simultaneous Localization and Mapping Algorithm preformed on data collected by Velodyne using a LiDAR of the same model. The second, the surface mapping of a hallway using an algorithm that preforms fast triangulation of unordered points provided by the Point Cloud Library. In regards to hardware. The drive train of the solar powered golf cart was analyzed and a plan was created for the implementation of computer command based driving for the golf cart.

6 Discussion and Future Work

The purpose of this project was to explore partially autonomous indoor navigation. The work completed made progress towards the main three components of achieving partial autonomy: hardware / computer hardware interface, sensor communication and implementation, and data processing and planning. This project intended to make progress on each major component of autonomous navigation to take the first step towards Occidental having a partially autonomous golf-cart created by students.

The future work for hardware includes the implementation of a motor onto the steering column of the golf cart and the establishment of the communication between the rear wheels of the golf cart and the raspberry pi. For the sensor aspect of the project the next step is to add a GPS or inertial measurement unit to the golf cart for SLAM processing of incoming data. The computational challenge of these upgrades will primarily be in the hardware soft-ware interface. The first challenge will be uploading the necessary software onto a microcomputer that can accommodate the large amount of data from the LiDAR sensor. The next challenging step will be identifying paths in the map generated by the SLAM algorithm and communicating the path to the golf cart. If, in the future, the golf cart becomes operational it will be important to clarify the ethical considerations of having a self-driving vehicle on Occidental's campus further. At this point in the development of partially autonomous vehicles driver control in the case of testing and malfunction is critical.

References

[1] H. Igarashi, K. Loi. Artif Life Robotics, **5**: 72, 2001.

[2] T. Tsukiyama. Mobile robot localization from landmark bearings. In XIX IMEKO World Congress Fundamental and Applied Metrology, September 2009.

[3] A. David, B. McCarragher. Geometric constraint identification and mapping for mobile robots. Robotics and Autonomous Systems. **35**:59-76, 2001.

[4] B. Mordechai, F. Modada. Elements of Robotics. Elements of Robotics, 2018.

[5] H. Igarashi. Separating visual information into position and direction by SOM.*Artificial Life and Robotics*, **8**:5-8, 2004.

[6] V. Klingspor, K. J. Morik, A. D. Rieger. Machine Learning. **23**:305, 1996.

[7] S. Mahlknecht, R. Oberhammer, G. Novak. Real-Time Syst. **29**:247, 2005.

[8] C.G., Harris, M. Stephans. A combined corner and edge detector. Alvey vision conference, **Vol. 15, No. 50**:10-52, 1988.

[9] Z. Jia, A. Balasuriya, S. Challa. Artif Life Robotics **12**:317, 2008.

[10] H. Musoff, P. Zarchan. Fundamentals of kalman filtering: A practical approach. American Institute of Aeronautics and Astronautics, 2009.

[11] JY Chang, WF Hu, MH Cheng, et al. Digital image translational and rotational motion stabilization using optical flow technique. IEEE Trans Consumer Electron **48**:108-115, 2002.

[12] D. Murray, C. Jennings. Stereo Vision Based Mapping and Navigation for Mobile Robots. IEEE Int Conf Robotics Automation **Vol. 2**:1694-1699, 1997.

[13] T. Umeno, E. Hayashi. Navigation system for an autonomous robot using an ocellus camera in indoor environment. 2005.

[14] LH. Juang, JS. Zhang. Artif Intell Rev, 2018.

[15] S. H. Cho, Y. Kyong, S. Hong, W. D. Cho.Self Localization Method Using Parallel Projection Model for Mobile Sensor in Navigation Applications. Journal of Computer Science and Technology **24(3)**:588-603, 2009.

[16] B.S. Choi, J.J. Lee. Localization of a mobile robot based on an ultrasonic sensor using dynamic obstacles. *Artificial Life and Robotics* **12(1-2)**:280-283, 2008.

[17] S. Choi, T. Jin, J. Lee. *Artificial Life and Robotics*. **7**:132, 2003.

[18] Velodyne. User's Manual and Programming Guide: HDL-32E High Definition LiDAR Sensor. July 2015.

[19] I. Zunaidi, K. Norihiko, N, Yoshihiko, M. Hirokazu. Positioning System for 4Wheel Mobile Robot: Encoder, Gyro and Accelerometer Data Fusion with Error Model Method. 2019.