2017

A 3D Data Acquisition Cart with Applications to Warehouse Automation

Vibhor Sood
Lehigh University

A 3D Data Acquisition Cart with Applications to Warehouse Automation

By

Vibhor Sood

A Thesis
Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for Degree of
Master of Science

in

Electrical Engineering

Lehigh University

Dec 2017

This thesis is accepted and approved in partial fulfilment of the requirements for the Master of Science.

Date

John Spletzer
(Thesis Advisor)

Co-Advisor

Chengshan Xiao
(Chairperson of Department)

Contents

List of Figures

List of Tables

Abstract

Automated Guided Vehicles (AGV) are increasingly being adopted for warehouse automation. This work focuses on the design and fabrication of a 3D Data Acquisition Cart (3D-DAC) with applications to warehouse automation. The 3D-DAC facilitates acquiring large scale data sets without the overhead of requiring an AGV. It integrates on-board computing and power, optical wheel encoders, and a Velodyne VLP-16 Puck LiDAR for exteroceptive sensing. Three-dimensional (3D) LiDARs like the Velodyne Puck are becoming the sensor of choice for not only robot navigation, but also for other tasks such as pallet detection and picking and dropping to name but a few. In this thesis, we demonstrated real-time mobile data logging with the 3D-DAC. Results were validated in a Simultaneous Localization and Mapping (SLAM) task. Preliminary results indicate the potential to map warehouses on the order of 10,000 m^2 with an accuracy of several centimeters.

1 INTRODUCTION

Warehouses are the new malls, except people don't walk in and out of stores buying what they like. Instead, they shop online and companies like Amazon or Alibaba fulfill their orders with the help of robots [44]. This year saw a record $5.3 billion spent on online sales associated with Black Friday, and a staggering $25 billion spent on Singles Day in China - a day for individuals to show their pride in being single [43]. Furthermore, a recent article from Bloomberg [41] cites $2.7 billion spent on building warehouses in October of this year alone in the United States. The average size of warehouses is also increasing with a report published by CBRE Group Inc, noting that "the average warehouse completed this year was more than double the size in 2001." Given the increasing ubiquity of online shopping and larger warehouses, companies have a vested interest in leveraging autonomous robots to carry out low-value tasks such as fetching and transporting inventory for order fulfillment.

Larger warehouses present numerous opportunities and challenges, specifically when it comes to robot navigation. According to Leonard et al., the problem isn't the process of navigation, but the "reliable acquisition or extraction of information from sensor data [42]" particularly in known environments. This validates the need for acquiring large scale data sets for algorithm development and classifier training. To do this efficiently we have developed a 3D Data Acquisition Cart (3D-DAC) to log LiDAR (Light Detection and Ranging) and odometry data. We subsequently validated the data logging approach in a Simultaneous Localization And Mapping (SLAM) application.

1.1 Background

This work focussed on the use of a 3D LiDAR sensor and optical wheel encoders attached to a modular utility cart to facilitate mapping warehouse facilities. To ground the terms in this field and give information relevant to this work, a brief introduction to Velodyne VLP-16 Puck LiDAR, SLAM algorithms, Robot Operating System, and optical wheel encoder based odometry is provided.

1.1.1 Odometry

Odometry is the use of data from sensors such as wheel encoders, to estimate change in position and orientation of the robot over time. An optical wheel encoder is a type of incremental rotary encoder used for identifying change in position or the motion of the shaft (wheel hub in our case), which is used to calculate speed, distance and position. There are 2 primary types of encoders: magnetic or optical.

Magnetic rotary encoders rely on three components that are: a disk, sensors, and a conditioning circuit. The disk is magnetized, with *n* number of poles around its circumference. The sensors detect change in the magnetic field as the disk is being rotated, and convert this information into a sine wave. The sensors are often Hall effect devices which can detect changes in voltage. The conditioning circuit is used to condition (multiply or divide) the signal to produce the desired output. In contrast, an optical encoder identifies position change as light passes through a patterned encoder disk mounted on the wheel. As the disk revolves, windows in a masked pattern are covered or opened, showing the movement and position of the optical wheel encoder as shown in Figure 1.2. To provide additional information such as higher resolution or rotation direction, a track may have two sensors which are offset to give two copies of the signals produced at the same time, but 90 degrees out of phase. We used high resolution quadrature optical encoders for the 3D-DAC.

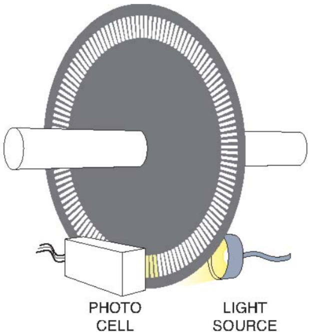


Figure 1.1: Pattern of mast on Encoder disk and configuration of scanning heads [24].

1.1.2 LiDAR

A LiDAR is similar to RADAR (RAdio Detection And Ranging) in that it uses the concept of time of flight to calculate distances. However, instead of using radio waves, a LiDAR uses ultraviolet, visible or infrared light to image objects. By taking advantage of the speed of light, a 2D LiDAR is able to acquire multiple points in a very short time. 3D LiDAR uses the same concept in 3D by integrating multiple 2D lasers. The LiDAR fires rapid pulses of laser light at a surface, and the amount of time it takes for each pulse to bounce back is calculated. Light moves at a constant and known speed, so the LiDAR instrument can calculate the distance between itself and the target with high accuracy. A key advantage of using a LiDAR compared to a camera (for example) is its insensitivity to ambient lighting and optical texture in the scene. The difference between a 2D LiDAR scan and a 3D LiDAR scan is shown in Figure 1.3.

$$Distance = \frac{(\text{Speed of Light x Time of Flight})}{2}$$

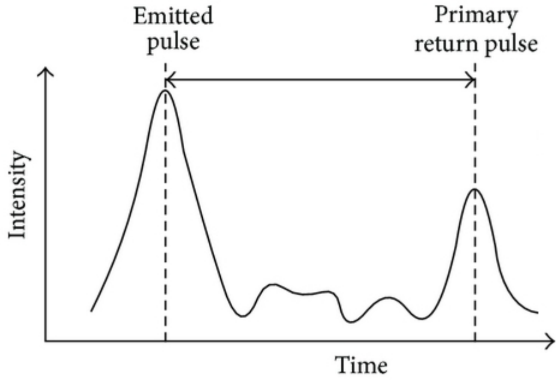
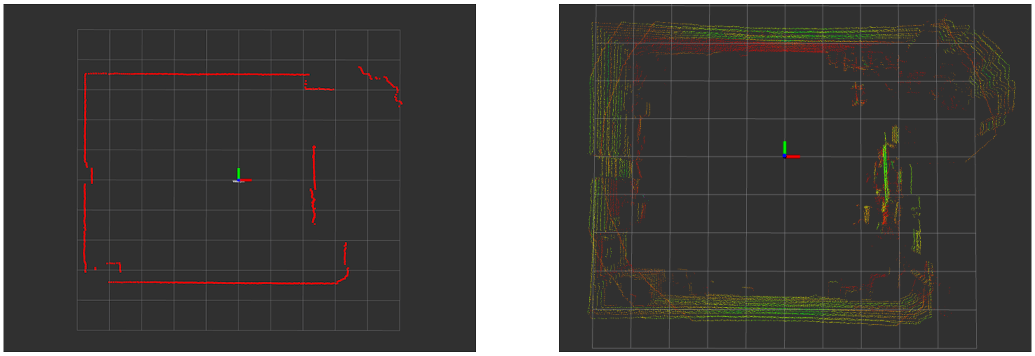


Figure 1.2: Time of flight measurement system operation principle, where the distance is calculated dividing by 2 the time elapsed between the emitted pulse [23].

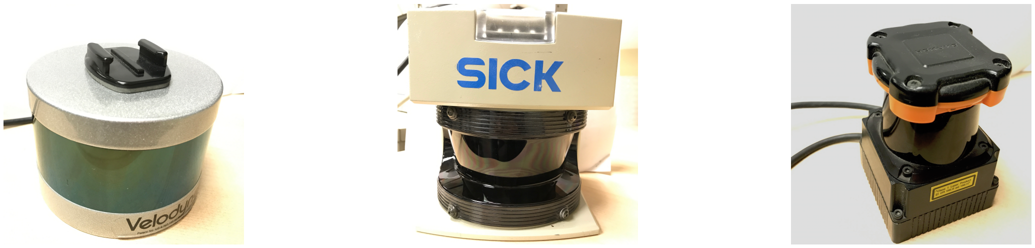
The container used for 2D or 3D LiDAR scans is referred to as a *point cloud* Point clouds contain not only the 3D point data, but also time, intensity (how much light returns back from a point), color, and other meta data. In figure 1.4, examples of 2D and 3D LiDARs are shown. The difference between point cloud size can be dramatic. A single point cloud from the VLP-16 Puck 3D LiDAR can contain up to 57,600 points, while SICK LMS291 contains 181 points, and the Hokuyo UTM-30LX has up to 1,081 points. LiDARs are used extensively in AGV systems for tasks such as localization, pallet detection, and pick and drop systems as shown in Figure 1.5.



(a) 2D Laser scan (Top View)

(b) 3D point cloud (Top view)

Figure 1.3: 2D Laser scan & 3D point cloud of Packard Lab room 450.

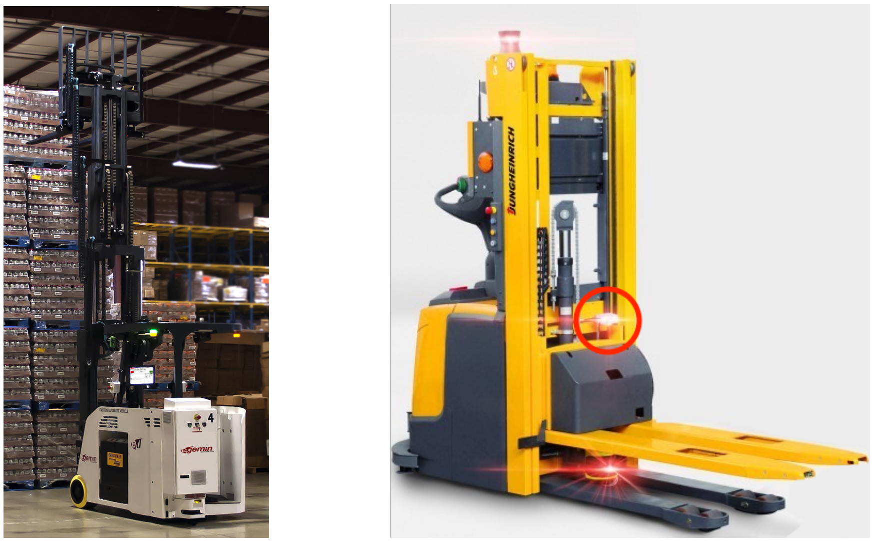


(a) VLP-16 Puck 3D LiDAR

(b) SICK LMS291 2D LiDAR

(c) Hokuyo UTM-30LX 2D LiDAR

Figure 1.4: The different types of 2D and 3D LiDARs in VADER lab



(a) AGV Forklift picking and dropping items

(b) AGV Forklift with LiDAR being used for pallet detection (encircled)

Figure 1.5: The different applications of AGVs using LiDARs being shown such as picking and dropping, and pallet detection

1.1.3 Simulataneous Localization And Mapping (SLAM)

Simultaneous Localization And Mapping is the task of constructing or reconstructing a map of an environment while simultaneously keeping track of the location of the 3D-DAC within the map. In our case, the data acquired by the Puck and wheel encoders are used to make a 2D map and localize the 3D-DAC to implement **SLAM**.

There are multiple steps involved in the implementation of SLAM, and the goal of the process is to find the pose of the robot relative to a map being built on-the-fly. A predictor-corrector algorithm is used to estimate the position and orientation (pose) of the robot. The initial, *prediction* of the 3D-DAC's pose is done by odometry data as measured by calculating the individual wheel velocities. Those velocities are then mapped to robot velocities using the kinematic model of the cart. Then the rotation and translation in the $x-y$ direction are calculated, and this serves as the initial guess of the robot's pose. The *correction* step refines the initial pose estimate with the help of LiDAR data.

SLAM algorithm can be divided into two approaches: feature-based and scan matching. In the feature-based approach, salient features in the environment are explicitly extracted from the LiDAR scan. The robot then attempts to associate these observations to landmarks it previously had seen. Re-observed landmarks are then used to update the robots position. In contrast, scan matching approaches to SLAM extract features implicitly by finding the rigid transformation necessary to align two consecutive scans. In this work, we employ a scan matching approach. A more in-depth review of the scan matching technique is discussed below.

1.1.4 Robot Operating System (ROS)

Robot Operating System (ROS) is a robotics middleware used for robot software development, and not an Operating System (OS) as the name suggests. It provides inter-process communication, hardware abstraction, device drivers, libraries, package management, and much more. It also provides packages for SLAM, path planning, and for a wide range of robotics algorithms. ROS is not only being used for Research and Development purposes, but companies such as Rethink Robotics [39], and many more are using it to create products in the robotics space. The packages that were used for this project are listed in Table 1.1.

Package Name	**Description**
pointcloud_to_laserscan	It is used to convert the 3D point cloud received from the LiDAR to a 2D laser scan
laser_scan_matcher	It is a scan matching algorithm based on [18], allows for scan matching between 2 consecutive scans received from the above package
gmapping	It is used to create a 2D occupancy grid map from the data received from scan matching and odometry
hector_trajectory_server	It is used to convert the pose data from the odometry to display a path
phidgets	Used to get the count from Left and Right *high_speed_encoders*

Table 1.1: ROS packages used for the implementation of SLAM

1.2 Objective

The objective of this project is to find a efficient, effective approach for data acquisition in large-scale warehouse environments. Many applications can be imagined but our demonstration application is SLAM using the Puck LiDAR and optical wheel encoders. Other applications of the data acquisition system may include pallet detection i.e. imaging pallets on upper rack shelves, using the telescoping mast mounted on the cart. Thus, the data acquisition system needs to be adaptable to these and other tasks.

2 RELATED WORK

In warehouse environments, fetching the order is responsible for more than 60% of the operational cost [2]. Yet, warehouse navigation flexibility is still impaired by embedding infrastructure into the workspace, for example, using a series of barcode stickers on the floor [3]. For robots operating in a warehouse or even just a large dynamic environment, with just onboard sensors, a "*natural*" behaviour would consist of storing some major characteristics of an already driven path, and to employ these references as checkpoints for future navigation task along the same route. Scenarios where this approach would be extremely useful and could extend beyond warehouse use include planetary sample return missions [4], charging station homing for electric wheelchairs [5], and autonomous underground tramming for mining [6].

LiDAR has increasingly become an important range sensor in robot navigation[10] to implement SLAM, and most applications require 2D LiDARs [11]. If the LiDAR scan rate is sufficiently high relative to the velocity of the robot, distortion from within scans can often be disregarded [25]. In [12], standard Iterative Closest Point (ICP) methods [13] can be used to calculate the rigid transformation between 2 consecutive scans. Moreover, a two-step method is proposed to remove the distortion [14]: an ICP based velocity estimation step is followed by a distortion compensation step, using the computed velocity.

Newman et al. in [4], propose an infrastructure-free framework that makes use of a topological map instead of a globally consistent metric map for warehouse navigation. They use a monocular camera to track the floor texture and the map is a locally consistent pose graph representation where each pose in the graph has an corresponding image. Localization is implemented by matching the current frame to a probable subset of the pose graph.

Chong et al. [15], presented a precise localization algorithm for vehicles in 3D urban environment with only one 2D LIDAR and odometry information. They presented an idea of synthetic 2D LiDAR to solve the localization problem on a virtual 2D plane. A Monte Carlo Localization scheme was adopted for vehicle position estimation, based on synthetic LIDAR measurements and odometry information from an IMU. While this solution achieves good global localization, it is not designed for precise estimation relative to the local environment, an ability that is highly desirable in a number of cases.

Zhang et al.[16] propose a real-time method using a two-axis LiDAR moving in six-Degrees Of Freedom (DOF) for odometry and mapping. Their suggested method achieved both low-drift and low-computational complexity without using high accuracy ranging or inertial measurements. The key idea mentioned to obtain this comes from dividing SLAM, which seeks to optimize a large number of variables simultaneously, into two algorithms. One performs odometry at a high frequency but low fidelity to estimate the velocity, while the other runs at a lower frequency for fine matching and registration of the point cloud. By combining the two, mapping can be achieved in real time.

In [17], a method for infrastructure-free localization in a warehouse environment is presented for an Automated Guided Vehicle. A 3D LiDAR was used to accomplish this. To construct a 3D map of the warehouse, 3D data were used for both feature segmentation and mapping. The upright shelving was used as landmarks, after which a map based localization algorithm was implemented using 3D feature to landmark matching, thus reducing the data association errors.

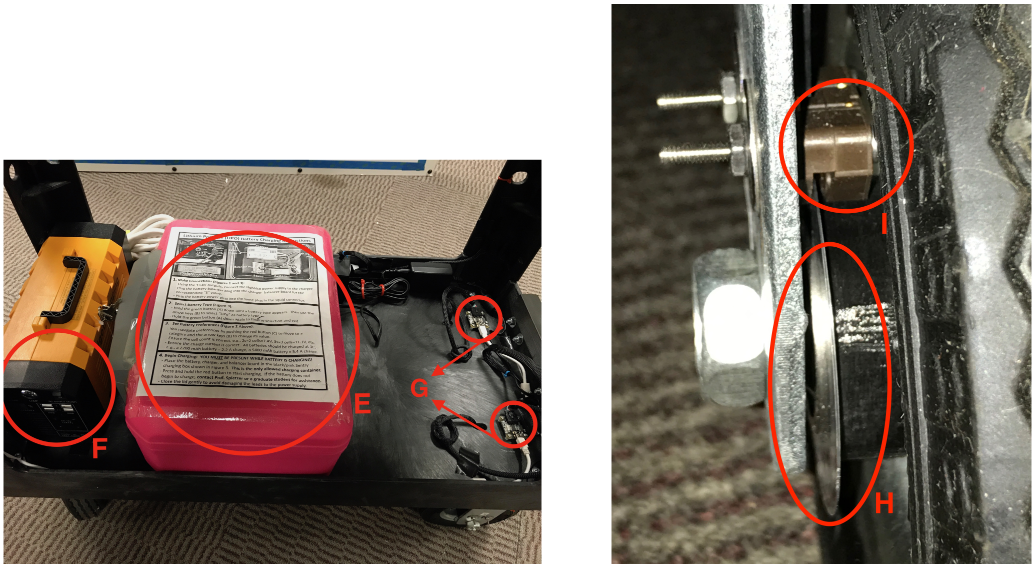
From the work presented above, we can conclude that LiDAR has become the sensor of choice to help robots/vehicles navigate, not only for large warehouse or indoor environment but even for outdoor environment, such as self-driving vehicles. Most of the work focuses on building solutions after the initial data has been acquired. It again motivates the need for efficient data collection for development of algorithms.

3 TECHNICAL APPROACH

3.1 Development platform

The chassis for this project is a utility cart, which can support up to 500lb [Utility Cart]. The cart has swivel casters at the back for steering and fixed casters at the front where we mounted the wheel encoders. Generally solid wheels are preferred when using wheel encoders as the diameter is constant. However, we deliberately chose pneumatic wheels since they provide improved ground contact and yielded better results in our case.

Odometry data is obtained from high-resolution quadrature encoder readers that can detect incremental changes in wheel position, and track these changes with respect to time. To get the odometry data, a transmissive optical encoder module is used to detect rotary or linear position. It consists of a lensed LED source and a monolithic detector IC and uses a phased array detector



(a) Bottom shelf which houses, F) Li-Ion power supply, E) Fireproof charging case, and G) High speed encoder reader

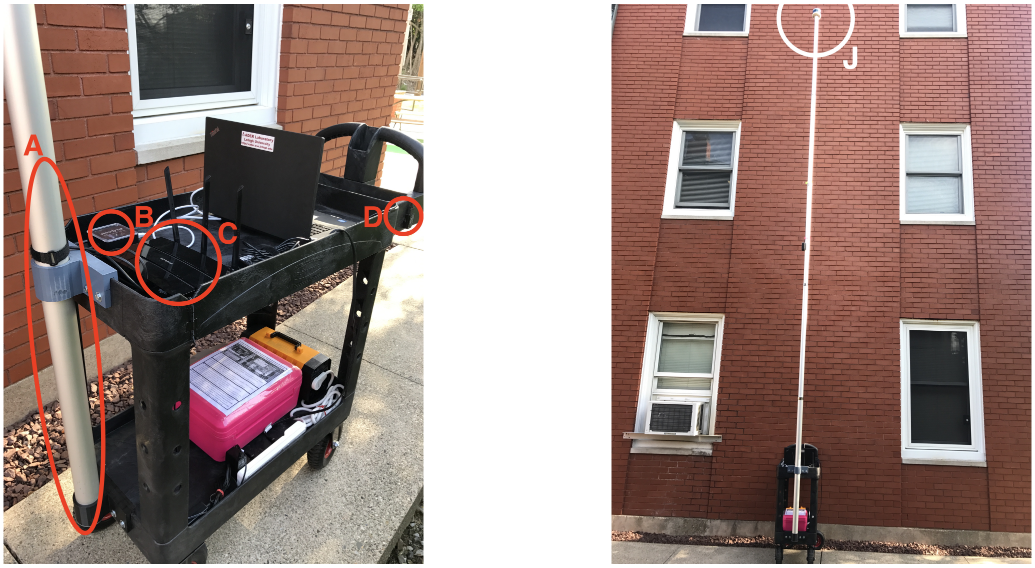
(b) Front fixed wheels with H) encoder disk, and I) encoder module

Figure 3.1: Prototype utility cart for warehouse data acquisition

technology. The rotary encoder disks [Disk] are made from Mylar polyester film. Each optical encoder module is resolution specific and is matched to the resolution of the encoder disk. In our case, it was 2048 Counts Per Revolution (CPR). The mounting of the encoders is shown in Figure 3.1.b.

For exteroceptive sensing, a Velodyne VLP-16 Puck LiDAR [Velodyne] was used, it has a range of 100 m, supports 16 channels, 360°horizontal field of view and a 30°vertical field of view, with ±15°up and down. Also, the returning UDP packets contain time of flight data, reflectivity measurements, and rotation angles all of which have synchronized time stamps with μs resolution. To mount the Puck, a telescopic aluminum mast [Mast] is used which can be extended up to 6 m as shown in Figure 3.2.a-b. The ability to dramatically change the LiDAR's height is important, as AGV localization LiDARs can be anywhere from < 1 m to > 3 m above the ground. Furthermore, the LiDAR could be used for pallet detection on higher level racks, shelf clearing, inventory management and so on as illustrated in Figure 1.5. Custom mounts for the mast and LiDAR were designed and 3D printed; these secure the mast to the front of the cart, and mount the LiDAR on top of the mast.

On the top shelf of the cart, a Gigabit wireless router is mounted to provide wireless network access for a mast mounted 3D camera along with a wireless bridge and a Li-Ion battery that would be installed in the future. On the side, a Microstrain 3DM-GX3 IMU [IMU] is mounted



(a) 3D-DAC with all the components: A) telescopic mast, B) Puck Interface box, C) Wi-Fi router, D) Microstrain IMU

(b) 3D-DAC with fully extended, 6 m high telescopic mast

Figure 3.2: Different labeled parts used to design the 3D-DAC

for gyro-corrected odometry which would be implemented in the future.

On the bottom shelf of the 3D-DAC, shown in Figure 3.1.a, a fireproof charging case is placed for safely charging and storing Li-Po batteries which would be kept on board for powering a 3D camera in the future. A Portable high capacity (120V AC/12V DC, 31 AH and 346WH) Lithium-ion power supply [UPS] was chosen which provides an impressive data collection time of 10 hours when all the sensors and the laptop are powered up. A Li-Ion was chosen since it has higher density, improved resiliency, and longer cycle life as compared to Lead-acid batteries [40]. Figure 3.3 depicts the weight and density of Li-Ion batteries as compared to Lead-acid:

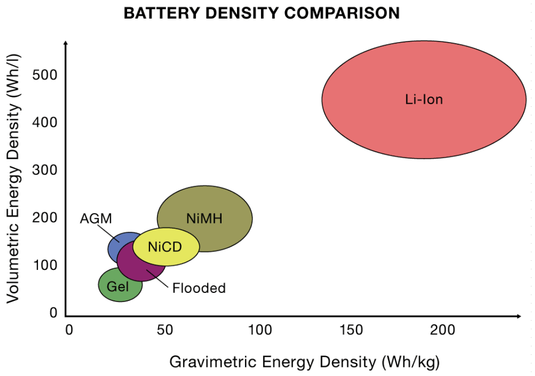


Figure 3.3: Lithium-Ion vs Lead-acid batteries [40]

3.1.1 Pre-logging check

As a pre-logging check, we generated a frequency plot for the LiDAR to ensure it was operating at the desired frequency. We found this useful, as the frequency needs to be set consistently in both the ROS driver and the Velodyne firmware. If either is set incorrectly, the data will be corrupted (observed in the course of our work). Representative plots are shown in Figure 3.3 and Figure 3.4. We can extrapolate from Figure 3.4 that the firmware sets the actual rotation speed since the frequency was set differently in the driver and firmware. To validate, we plotted not only the histogram, but also an individual scan as illustrated in Figure 3.4. When the scan rate is correctly set to nominally 10 Hz, we see a 2π wrap. However, if the scan rate is set to 10 Hz in the driver and the firmware is set to 20 Hz, a 4π wrap is observed. This is indicative of the LiDAR spinning at twice the rate the ROS driver is reading/publishing the scan data.

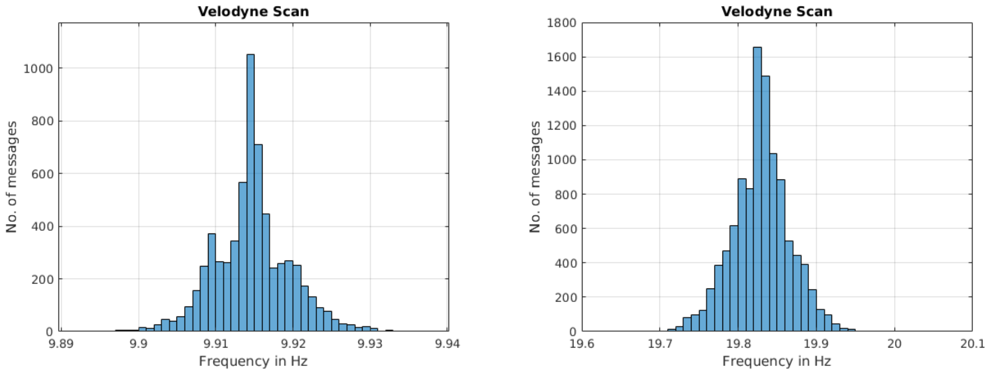
$$1\ RPM = \frac{1}{60} Hz$$

Thus,

$$600\ RPM = \frac{600}{60} = 10Hz$$

Similarly,

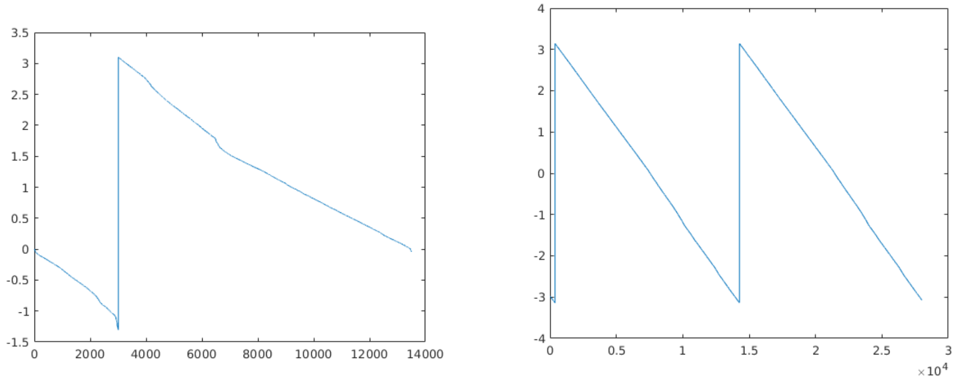
$$1200\ RPM = \frac{1200}{60} = 20Hz$$



(a) LiDAR operating at 10Hz

(b) LiDAR operating at 20Hz

Figure 3.4: LiDAR frequency plot showing histograms of the scan rate for 10 Hz (left) and 20 Hz (right).



(a) Azimuth angle when LiDAR is operating at 10Hz in the driver as well as Velodyne firmware.

(b) Azimuth angle when frequency is set to 20Hz in firmware but 10Hz in ROS driver

Figure 3.5: LiDAR Azimuth angle plot

3.2 Pose Estimation using Encoders

The first and foremost task of the project is determining the pose of the 3D-DAC with respect to the environment as it being pushed around. The combination of position and orientation is referred to as the *pose*. Dead reckoning is although a primitive but well known method for pose estimation over short distances. To implement dead reckoning, we assume a kinematic model of the cart, which includes

- Movement on a horizontal plane
- Point contact of the wheels
- Pure rolling ($v = 0$ at contact point)
- Wheels connected by rigid frame

The two casters wheels on the cart do not impose any constraint on the carts chassis, since they can rotate freely in any direction. On the other hand, the two fixed wheels in the front each impose two constraints on the cart's movement,

- The **sliding** constraint for this wheel enforces that the component of the wheel's motion orthogonal to the wheel plane must be zero.
- The **rolling** constraint for this wheel enforces that all motion along the direction of the wheel plane must be accompanied by the appropriate amount of wheel spin so that there is pure rolling at the contact point.

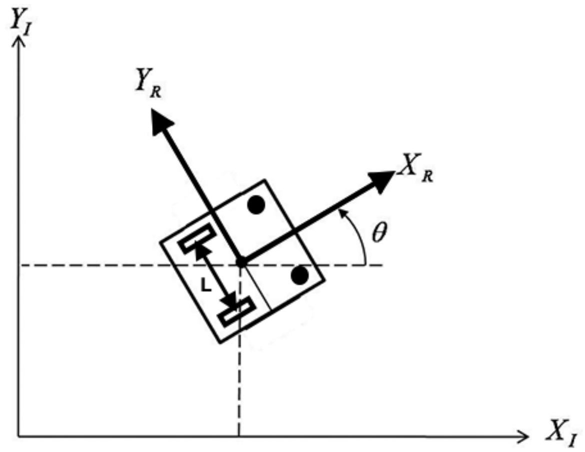


Figure 3.6: Kinematic model

The model parameters include

- r_L = Left wheel radius
- r_R = Right wheel radius
- L = wheelbase i.e. distance between wheels

and the **kinematic equations** are

Initial frame: $\{X_I, Y_I\}$

Robot frame:$\{X_R, Y_R\}$

- V_R = linear velocity of right wheel
- V_L = linear velocity of left wheel
- $\dot{\phi}_R$= angular velocity of right wheel
- $\dot{\phi}_L$ = angular velocity of left wheel

$$V_R = r_R \dot{\phi}_R \tag{3.1}$$

$$V_L = r_L \dot{\phi}_L \tag{3.2}$$

$$\omega(t) = \frac{V_R(t) - V_L(t)}{L} \tag{3.3}$$

$$v(t) = \frac{V_R(t) + V_L(t)}{2} \tag{3.4}$$

Since we are assuming no sliding,

$$v_y = 0 \tag{3.5}$$

so the velocities in our world frame (X_I, Y_I) are

$$\dot{x}(t) = v(t)\cos\theta(t) \tag{3.6}$$

$$\dot{y}(t) = v(t)\sin\theta(t) \tag{3.7}$$

$$\dot{\theta}(t) = \omega(t) \tag{3.8}$$

3.3 Calibration and Mounting

Calibration is necessary in odometry to reduce navigational errors. This is especially important since the diameter of pneumatic wheels can change with temperature. The main parameter needed to calibrate this is the measure of Distance per "encoder ticks" for each wheel. It is defined as the distance traversed by each robot wheel during each encoder tick. We can calibrate the robot by pushing the cart on a straight line for a distance of (for example) 5 m and monitoring encoder counts of each wheel The average of these counts is divided by the total distance traveled to get a starting value for the encoder tick, which happens per millimeter.

$$\text{Counts per millimeter } = \frac{\text{right count}}{\text{total distance(in mm) traveled}}$$

The ideal diameter of both the wheels should be 0.1524 m, but since we're using pneumatic wheels, the left and right wheel radii will vary. To verify the radius of the left and right wheel

$$\text{Distance travelled} = \frac{\text{Encoder ticks}}{CPR} 2\pi r \tag{3.9}$$

Since we're using quadrature encoders, the number of ticks will be divided by 4, thus

$$r = \frac{\text{Distance travelled x } CPR \text{ x 4}}{\text{Encoder ticks x 2 x } \pi} \tag{3.10}$$

- $D_L =$ Calculated diameter of left wheel

- $D_R =$ Calculated diameter of right wheel

For our calibration, distance travelled = 5 m, and the encoder count per revolution (CPR) = 2048
Left encoder ticks = 83,630

$$r_L = \frac{5m \text{ x } 2048 \text{ x } 4}{83630 \text{ x } 2 \text{ x } \pi} = 0.07795m \tag{3.11}$$

$$D_L = 0.1559m$$

Right encoder ticks = 85,889

$$r_R = \frac{5m \text{ x } 2048 \text{ x } 4}{85889 \text{ x } 2 \text{ x } \pi} = 0.0759m \tag{3.12}$$

$$D_R = 0.1519m$$

3.4 Software Architecture

In the ROS programming paradigm, processes are nodes. ROS provide inter-process communication between nodes. By composing multiple nodes, we construct significant applications. The QT graph for our project is depicted in Figure 3.7, and was generated using the rqt_graph package in ROS.

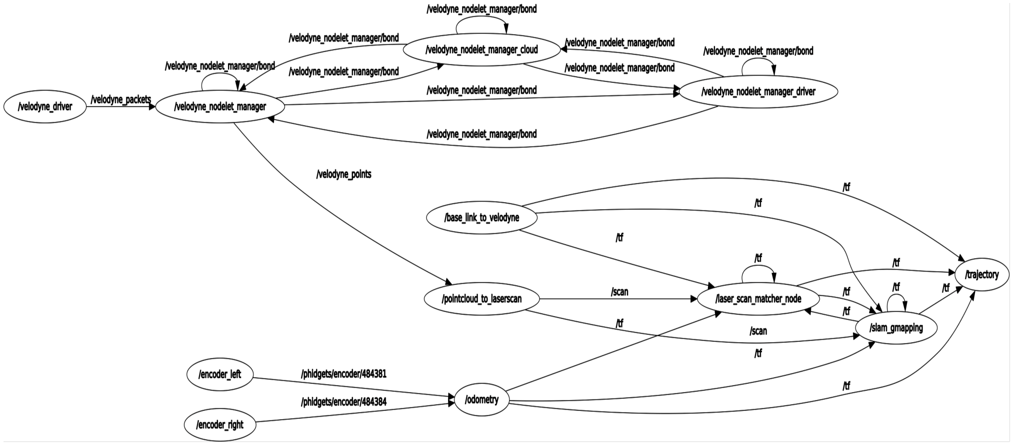

/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager_cloud
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager/bond
/velodyne_driver
/velodyne_packets
/velodyne_nodelet_manager
/velodyne_nodelet_manager/bond
/velodyne_nodelet_manager_driver
/velodyne_nodelet_manager/bond
/velodyne_points
/tf
/tf
/base_link_to_velodyne
/tf
/tf
/tf
/trajectory
/tf
/tf
/tf
/pointcloud_to_laserscan
/scan
/laser_scan_matcher_node
/tf
/tf
/slam_gmapping
/tf
/scan
/tf
/encoder_left
/phidgets/encoder/484381
/tf
/odometry
/phidgets/encoder/484384
/encoder_right

Figure 3.7: QT graph

The QT graph depicts the flow of data, and the source of the data, the odometry data is generated from the left and right encoder. The odometry data is then transformed to the robot frame and used by three nodes, laser_scan_matcher for ICP, slam_gmapping to build the 2D map, and trajectory to show trajectory robot took.

Similarly, the Puck generates velodyne_packets which contain the 3D point cloud. The 3D point cloud is then converted to a 2D laser scan using the pointcloud_to_laserscan node which is used for ICP. The 2D laser scan is then transformed to the robot frame from the Puck LiDAR frame to be used for building the map.

3.5 Localization using scan matching

Let's assume our robot takes scan x_0 at t_0 and another scan at x_1 at t_1. We assume that the distance travelled is small, and the same environment will be visible to the Puck from both locations i.e. the scans will largely overlap. Using the overlap in the obtained scans, it is possible to find rigid transformation to align the two consecutive scans. Matching of the above mentioned scans is referred as *scan matching*. An example depicting scan matching is shown in Figure 3.8 below.

3.5.1 Iterative Closest Point Algorithm

ICP is an algorithm employed to calculate the translation t and rotation R that minimizes the sum of the squared error. In ICP the reference, or *target*, is kept fixed, while the *source*, is transformed to best match the reference. The algorithm is terminated based on the maximum number of iterations or when the relative error is below threshold. The ICP algorithm iteratively performs the following steps

- Matching: for every data point the nearest neighbor in the model point set is found.
- Minimization: the error metric is minimized.
- Transformation: data points are transformed using the minimization result.

Although, in many cases the algorithm converges rapidly, there might be several problems:

- The algorithm may converge towards one of the local minima instead of the global minimum.
- Outliers may cause faulty results, especially at quadratic error weighting.
- The point cloud's may not resemble the same parts of an object. Partial overlap is required but it may cause error as well.

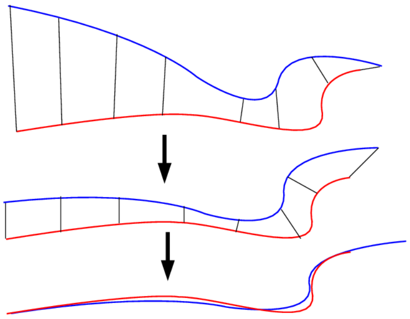


Figure 3.8: An instance of ICP algorithm to align 2 scans. A set of points are selected along each line. One of the point sets is iteratively moved and transformed to minimize the distance between each point set [28].

In [27] 6 distinct stages of the ICP algorithm were identified, which have been mentioned below:

3.5.1.1 Selection

In selection it may be beneficial to consider only some of the model and data points before applying the ICP algorithm. To reduce computational complexity, the amount of points may be reduced by random or uniform subsampling. This will speed up computations mostly in the matching step. The idea behind random sampling is to sample differently in every iteration of the algorithm in order to prevent any bias towards outliers.

To achieve a good distribution of surface normals among the sampled points, these should be put into buckets of similar normal direction. In [29], buckets were based on the angles to the basis vectors. For a normal of unit length

$$\vec{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

the angles to the x, y and z-axis are calculated simply as

$$a\cos(n_x)\; a\cos(n_y)\; a\cos(n_z)$$

3.5.1.2 Matching

Matching accounts for the pairing of points from the data point cloud to the model point cloud. Finding the nearest neighbors is usually the most computational intensive step in the ICP algorithm. Techniques might be employed to speed up nearest neighbor searching. Nearest neighbor may be defined differently depending on the implementation, one is closest point in euclidean distance as show in Figure 3.9 (left). The other is normal shooting where we pick the point on the plane that is closest as shown in Figure 3.9 (right).

Alternatively, point matching can be done by finding the line-surface intersection. Letting l define the line originating in data point p with direction of normal $\vec{n}_p$

$$l : \begin{bmatrix} x \\ y \\ z \end{bmatrix} = p + t.\vec{n}_p$$

Every data point p is paired with the intersection of l with the surface of the model point cloud. If for a particular point, the line l does not intersect any triangle, the point is not taken into account. This approach scales linearly with the number of points in the model point cloud.

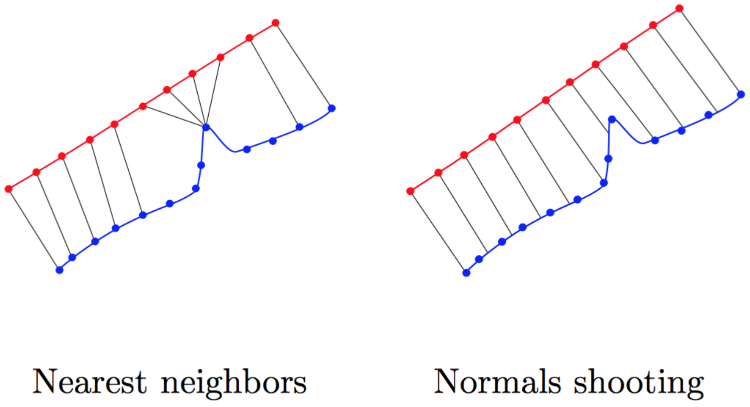


Figure 3.9: ICP matching strategies in two dimensions. Blue points represent the model point cloud's to which red data points are matched [34].

3.5.1.3 Weighting

Matched point pairs may be weighted differently based on their compatibility in some sense. Practically this means scaling every term in the error metric with a specific factor w. The factor could be based on distance, color, curvature and so on.

Normal compatibility weighting for a point pair with normals $\vec{n}_p$ and $\vec{n}_q$ can be done using the weight [34]:

$$w = \vec{n}_p.\vec{n}_q$$

For points with curvature values c_p and c_q in the range $[0;1]$, a Gaussian weight is proposed by [34]:

$$w = e^{-(c_p - c_q)^2}$$

To weight the point pair (p,q) according to their distance, Godin suggested [30]

$$w = 1 - \frac{dist(p,q)}{dist_{max}}$$

where $dist_{max}$ is the maximum distance of all point pairs.

3.5.1.4 Rejection

Point pairs may be rejected after the matching step. This can be done on a statistical evaluation of the nearest neighbor distances, the 5% or 10% worst point pairs may be rejected. For point cloud's that have only partial overlap, point pairs involving edge vertices might be rejected. This requires at least one of the point cloud's to be triangulated. Two points are both edge vertices if the pair only appears in one triangle. If the point pair is present in two triangles, then it can only be concluded that one of them is not an edge vertex.

3.5.1.5 Error metrics

The error metric defines the objective function that is minimized in every iteration of the algorithm. Two metrics commonly deployed are:

Point to point minimization [31] sums the squared distances of data points to model points, i.e.

$$E = \sum_{i=0}^{N} ||Rp_i + \vec{T} - q_i||^2$$

Point to plane minimization sums the distances of data points to the tangent planes in which the matched model points reside.

$$E = \sum_{i=0}^{N} \left[\left(Rp_i + \vec{T} - q_i^2 \right) .\vec{n}_i \right]^2$$

where $\vec{n}_i$ denotes the estimated tangent normals at the i^{th} model point.

3.5.1.6 Minimization

A closed form solution exists for the minimization of the point to point error metric [32]. A closed form solution also exists for the equivalent quaternion formulation [33, 34].

The point to plane error metric only has a closed form solution after linearization of the rotation matrix R. The linearization introduces an error, but in iterations occurring later, rotations are expected to be small, and the linearized rotation matrix is close to the real one. The minimization was proposed and derived by [35].

Existing methods for estimating the covariance of the ICP algorithm are either inaccurate or are computationally too expensive to be used. Thus we used the ICP variant implemented in [18], it uses a point-to-line metric optimized for scan matching and is based on the analysis of the error function being minimized and considers that the correspondences are not independent, and explicitly utilizes the covariance matrix of the measurements, which are not assumed to be independent either.

4 EXPERIMENTAL RESULTS

4.1 Odometry

To calibrate the odometry, our calibration routine was repeated 10 times, and the average parameter values were used. To validate the performance of encoders, we localized the 3D-DAC using odometry alone. The results are shown in Figure 4.3. The error increases with time as expected. This is mostly caused by the sharp turns along the route which cause wheel slippage - violating the kinematic model discussed above. There is also a difference in count per mm values for right and left wheel, caused by wheel slippage when the cart turns, and slight manufacturing defects.

For the run shown here,

$$\text{Count per mm}_{Rightwheel} = 17.1$$

$$\text{Count per mm}_{Leftwheel} = 17.2$$

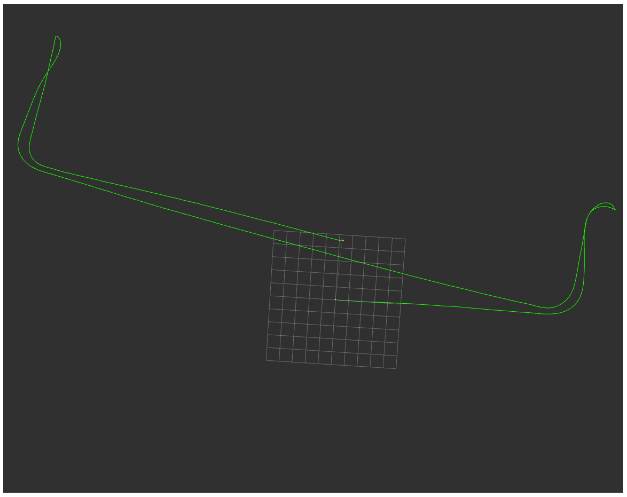


Figure 4.1: The path taken by the cart according to the encoders.

The distance travelled by the left wheel is

$$distance_{Left} = \frac{Count_{Left}}{count/mm_{Left}} = \frac{2,225,538}{17.2}$$

$$distance_{Left} = 129.39m$$

The distance travelled by the right wheel is

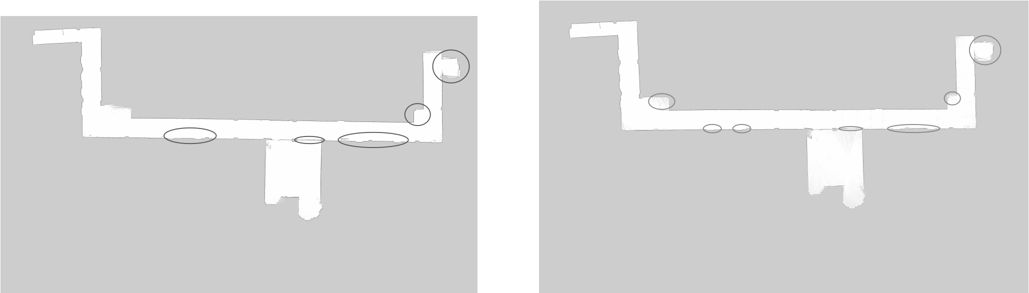
$$distance_{Right} = \frac{Count_{Right}}{count/mm_{Right}} = \frac{2,202,449}{17.1}$$

$$distance_{Right} = 128.79m$$

Although, the difference between the distances measured by the left and right encoders is 0.6 m, the pose error is ≈4.5 m as discernable from the map above (each square is 1 meter on the side). The cart travels in a straight line for ≈16 m with low error. The error increases when the cart makes a turn. The left or the right wheel will slip or slide in order to make the turn, and this dramatically violates the kinematic constraint. As can be inferred from this result, the error will grow unbounded over time.

4.2 ICP Mapping

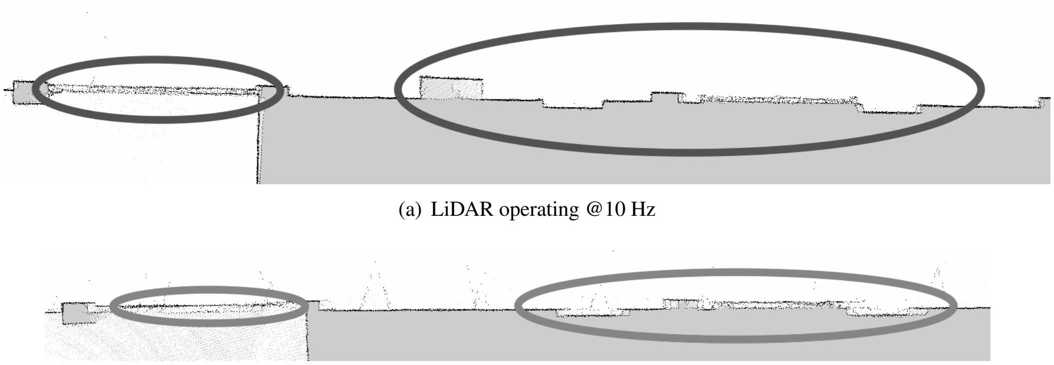
Several tests were conducted to validate the performance of the 3D-DAC. Ultimately, the goal was to validate the data acquisition through an implementation of SLAM. The first experiment was to map the 4th floor of Packard Lab on the Lehigh University campus. Both LiDAR and wheel encoder data were logged simultaneously and time-stamped for post processing. In our experiments, the LiDAR scan rate was set to 10 Hz as this qualitatively yielded better results than 20 Hz. This can be seen in Figure 4.2 and 4.3.



(a) LiDAR operating @10 Hz

(b) LiDAR operating @20 Hz

Figure 4.2: 2D map of 4^{th} floor Packard lab and room PA 450 for 10 Hz and 20 Hz. The LiDAR performed better at 10 Hz frequency. The circles parts indicate which features were better mapped.



(a) LiDAR operating @10 Hz

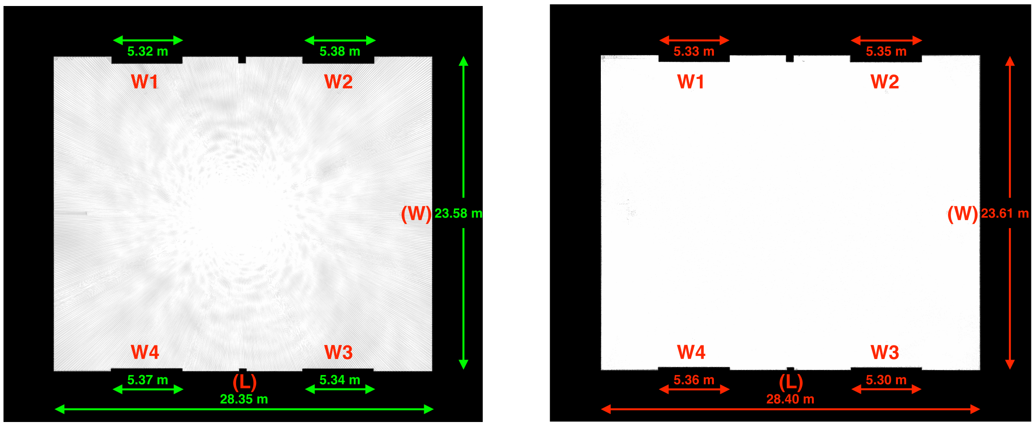
(b) LiDAR operating @20 Hz

Figure 4.3: 2D map of 4^{th} floor Packard lab for 10 Hz and 20 Hz. The LiDAR performed better at 10Hz frequency. The circles parts indicate which features were better mapped.

4.3 ICP Mapping Performance

To test the scan matching performance of the ICP algorithm and the cart itself, the 3D-DAC was taken to the basketball court in Taylor Gym at Lehigh University. Our first step was to obtain ground truth. To do this, the Puck was raised to a height of 2 m from the ground, and the maximum range of the Puck was set to 100 m. From this perspective, the entire court could be mapped with a single scan and the accuracy of the dimensions of the court were only limited to the accuracy of the Puck. The cart was placed at center of the court, and dimensions are shown in figure 4.4.a.

To test the SLAM performance, the Puck was kept at the same height of 2m but the maximum range of the LiDAR was decreased to 10 m in software. The cart was then started from the top left corner of the court and kept at a distance of 4 m from the wall, it was pushed at a speed of ≈ 1.4 m/s, around the perimeter of the court and once through the middle. Initially the error is 0.01 m but it reaches up to 0.04 m by the time it reaches the front wall. The resulting map has been shown in figure 4.4.b



(a) Actual dimensions when Pucks maximum range set to 100 m

(b) Calculated dimensions when Pucks range decreased to 10 m

Figure 4.4: 2D map of basketball court in Taylor Gym showing the actual and calculated dimensions.

Error calculation

$$\%error_{W1} = \frac{5.33 - 5.32}{5.32} = 0.18\%$$

$$\%error_{W2} = \frac{5.35 - 5.38}{5.38} = 0.55\%$$

$$\%error_{W3} = \frac{5.30 - 5.34}{5.34} = 0.74\%$$

$$\%error_{W4} = \frac{5.36 - 5.37}{5.37} = 0.18\%$$

$$\%error_{Length} = \frac{28.40 - 28.35}{28.35} = 0.176\%$$

$$\%error_{Width} = \frac{23.61 - 23.58}{23.58} = 0.127\%$$

The area of this facility is 668 square meters, and the maximum error percentage is 0.74% which includes the ±0.02 m error of the Puck itself. We can deduce that the actual error of the SLAM approach would be lower if Puck error is taken into account.

We can also extrapolate SLAM performance. If we were to increase the maximum range of the Puck to 50 m from 10 m, we would expect to be able to map an area with dimensions 5 times larger. In other words, we would expect to be able to reliably map an area 25X the size of the Basket ball court - an area of 16,712 square meters - with similar accuracy. This size is typical for large warehouses.

The parameters for the Laser Scan Matcher package that were used to obtain the above result are mentioned in Table 4.1.

Paramters	**Description**	**Value**
max_iterations	Max. ICP cycle iterations	40
max_correspondence_dist	Max. distance for a correspondence to be valid	0.4
max_angular_correction_deg	Max. angular displacement between scans°	45.0
max_linear_correction	Max. translation between scans (m)	0.5
ε_{xy}	A threshold for stopping (m)	0.000001
ε_{θ}	A threshold for stopping (rad)	0.000001
outliers_maxPerc	Percentage of correspondences to consider	0.90

Table 4.1: Laser Scan Matcher Parameters for the package mentioned in Table 1.1

5 FUTURE WORK

Odometry performance has an impact on SLAM performance when we're trying to localize the cart and map the facility. The limitations of odometry for pose estimation were shown in Figure 4.3. Once the cart makes a turn, the error starts accumulating quickly and grows unbounded in time. As mentioned previously, odometry is used for the initial alignment of LiDAR scan. The better the initial alignment, the greater probability of the scan matching algorithm converging to the global minimum. Thus, if we can improve the odometry performance we can improve SLAM performance. One way to improve odometry performance is to use gyro-corrected odometry, i.e. using a gyroscope to better estimate the angular velocity/displacement of the 3D-DAC. Gyro-corrected odometry works as the angular velocity measurements are not affected by wheel slippage. In fact, gyros can determine which of the two wheels has slipped, and provide corrections to the linear velocity of the cart as a result [45]. This is what we would like to pursue in the future. In terms of testing, the next step would be to take the cart to an actual warehouse to validate the SLAM performance in a representative environment.

6 CONCLUSION

In this paper, a flexible and low-cost 3D-DAC was designed and fabricated. We believe it will be extremely beneficial in application to warehouse automation tasks. We demonstrated the effectiveness of the 3D-DAC in a SLAM application where it was able to map an area of 660 square meters with an accuracy of approximately 0.05 m. Based upon these results, we believe the 3D-DAC would be suitable to map warehouse facilities with areas of 10,000 square meters or more. Our next step is to extend these results to representative, real-world environments.

7 REFERENCES

[1] Peter R. Wurman, Raffaello D'Andrea, Mick Mountz,"Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses", Association for the Advancement of Artificial Intelligence,vol.29, no.1,pp. 9-21, 2008.

[2] J. P. van den Berg and W. Zijm, "Models for warehouse management: Classification and examples," International Journal of Production Economics, vol. 59, no. 1, pp. 519–528, 1999.

[3] J. P. van den Berg and W. Zijm, "Models for warehouse management: Classification and examples," International Journal of Production Economics, vol. 59, no. 1, pp. 519–528, 1999.

[4] Gadd, M. & Newman, P. A framework for infrastructure-free warehouse naviga- tion. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, 3271–3278 (IEEE, 2015).

[5] T. Goedeme, M. Nuttin, T. Tuytelaars, and L. Van Gool, "Omnidirectional vision based topological navigation," International Journal of Computer Vision, vol. 74, no. 3, pp. 219–236, 2007.

[6] J. Marshall, T. Barfoot, and J. Larsson, "Autonomous underground tramming for center-articulated vehicles,"Journal of Field Robotics, vol. 25, no. 6-7, pp. 400–421, 2008.

[7] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," Journal of Field Robotics, vol. 27, no. 5, pp. 534– 560, 2010.

[8] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision ve- hicle localization in urban environments," in Proceedings of Robotics: Science and Systems, Atlanta, GA, USA, June 2007.

[9] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in ICRA'10, 2010, pp. 4372– 4378.

[10] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM– 3D mapping outdoor environments," Journal of Field Robotics, vol. 24, no. 8-9, pp. 699–722, 2007.

[11] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, “A flexible and scalable SLAM system with full 3D motion estimation,” in IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan, September 2011.

[12] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, “Comparing ICP variants on real-world data sets,” Autonomous Robots, vol. 34, no. 3, pp. 133–148, 2013.

[13] S. Hong, H. Ko, and J. Kim, “VICP: Velocity updating iterative closest point algorithm,” in IEEE International Conference on Robotics and Automation (ICRA), Anchorage, Alaska, May 2010.

[14] F. Moosmann and C. Stiller, “Velodyne SLAM,” in IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, June 2011.

[15] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, ”Synthetic 2D LIDAR for Precise Vehicle Localization in 3D Urban Environment, IEEE International Conference on Robotics and Automation (ICRA), 2013.

[16] J. Zhang and S. Singh, “LOAM: Lidar Odometry and Mapping inReal-time,” in Robotics: Science and Systems Conference (RSS 2014),2014.

[17] Dylan Schwesinger and John Spletzer,”A 3D Approach to Infrastructure-Free Localization in Large Scale Warehouse Environments”, IEEE International Conference on Automation Science and Engineering (CASE), added to IEEE Xplorer, November 2016.

[18] A. Censi, ”An ICP variant using a point-to-line metric” Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2008.

[19] Ibrahim Ozdur, Paul Toliver and T. K. Woodward,”Photonic-lantern-based coherent LIDAR system”, Optical Society of America (2015).

[20] Kenji Omasa, Fumiki Hosoi, and Atsumi Konishi, ”3D lidar imaging for detecting and understanding plant responses and canopy structure”, Journal of Experimental Botany, Volume 58, Issue 4, pp. 881–898, March 2007.

[21] Patric Beinschob and Christoph Reinke, ”Graph SLAM based mapping for AGV localization in large-scale warehouses”, Intelligent Computer Communication and Processing (ICCP), IEEE International Conference, 2015.

[22] Adolfo Comerón, Michaël Sicard, Francesc Rocadenbosch, Constantino Muñoz-Porcar, Alejandro Rodríguez and David García-Vizcaíno,"Atmospheric aerosol measurement with a network of advanced lidar instruments", SPIE Newsroom, March 2013.

[23] Leonardo Campos Inocencio, Mauricio Roberto Veronez, Francisco Manoel Wohnrath Tognoli, Marcelo Kehl de Souza, Reginaldo Macedônio da Silva, Luiz Gonzaga Jr, and César Leonardo Blum Silveira, "Spectral Pattern Classification in Lidar Data for Rock Identification in Outcrops", The Scientific World Journal, vol. 2014, Article ID 539029, 10 pages, February 2014.

[24] Brian Winter, "How optical Encoders work... ", http://blog.nidec-avtron.com/encoders/how-optical-encoders-work, April 2014.

[25] Ji Zhang, Sanjiv Singh, "Low-drift and real-time lidar odometry and mapping" Springer Science and Business Media, February 2016.

[26] Ahmad Kamal Nasir, Hubert Roth, "Pose Estimation By Multisensor Data Fusion Of Wheel Encoders, Gyroscope, Accelerometer And Electronic Compass", IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control (CESCIT), 2012.

[27] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm", Proceedings Third International Conference on 3-D Digital Imaging and Modeling, pp. 145–152, 2001.

[28] Smistad E, Falch TL, Bozorgi M, Elster AC, and Lindseth F," Medical image segmentation on GPUs - A comprehensive review", Medical Image Analysis 2015; 20: 1–18, https://doi.org/10.1016/j.media, 2014, 10.012 PMID: 2553428239.

[29] T. Masuda, K. Sakaue, and N. Yokoya, "Registration and integration of multiple range images for 3-d model construction", Proceedings of 13th In- ternational Conference on Pattern Recognition, 1:879–883 vol.1, 1996.

[30] G. Godin, M. Rioux, and R. Baribeau, "Three-dimensional registration using range and intensity information", Proceedings of the SPIE - The In- ternational Society for Optical Engineering, 2350:279–290, 1994.

[31] P.J. Besl and H.D. McKay, "A method for registration of 3-d shapes", IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2):239–256, 1992.

[32] K. S. Arun, T. S. Huang, and S. D. Blostein, ”Least-squares fitting of two 3-d point sets”, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9(5):698–700, 1987.

[33] O.D. Faugeras and M. Hebert, ”The representation, recognition, and locating of 3-d objects”, International Journal of Robotics Research, 5(3):27, 1986.

[34] B.K.P. Horn ”Closed-form solution of absolute orientation using unit quaternions” Journal of the Optical Society of America A (Optics and Image Science), 4(4):629–642, 1987.

[35] Y. Chen and G. Medioni, ”Object modeling by registration of multiple range images”, IEEE International Conference on Robotics and Automation, vol.3 pp. 2724–2729, 1991.

[36] Sam Shead,”Amazon now has 45,000 robots in its warehouses”, Business Insider, 2017.

[37] Jungheinrich AG, ”Complete solution in core shooting area”, The automobile manufacturer BMW Group counts on the reliability of Jungheinrich Automated Guided Vehicles (AGVs).

[38] Frank Tobe , ”Robotics industry growing faster than expected”, therobotreport, July 10.

[39] Rethink Robotics, ”Introduces high performance Sawyer robot with software development kit”, RethinkRobotics, October 2016.

[40] Joe O’Connor, ”Battery Showdown: Lead-Acid vs. Lithium-Ion”, Medium Corp., Jan 2017.

[41] Ingrid Lunden, ”Black Friday racks up 5.03$Binonlinesales$,2B on mobile alone”, TechCrunch, Nov. 2017.

[42] Patrick Clark and Kim Bhasin,”Warehouses Are Getting Bigger, Taller, Faster”, Bloomberg, Dec 2017.

[43] John J. Leonard and Hugh F. Durant-Whyte, ”Mobile Robot Localization By Tracking Geometric Beacons”, IEEE transactions on robotics and automation, Vol. 7 No. 3, June 1991.

[44] Tom Green, ”China Upgrades Logistics as Alibaba Goes Amazon”, Asian Robotics, April 2017.

[45] Carlos Garcia-Saura, ”Self-calibration of a differential wheeled robot using only a gyroscope and a distance sensor”, Imperial College London arxiv.org, Sep 2015

8 APPENDIX A

This appendix contains the table of all the parts that were used in the making of this cart

Part Name	**Description**	**Source**
Utility Cart	This is platform on which all the sensors would get mounted on, it can carry weights upto 500lb	Amazon.com
Uninterruptible power supply	Provides up to 10 hours of backup for data acquisition	Amazon.com
Wi-Fi router	It'll be used to connect the IFM 3D camera wirelessly	Amazon.com
Telescopic mast	This coud be extended to up to 6m and mount the LiDAR	Amazon.com
Encoder Disks	Transmissive Rotary Encoder Disk with CPR of 2048 used for odometry	US Digital
Encoder	EM1 transmissive optical encoder module used to read the encoder disk for odometry	US Digital
PhidgetEncoder	It used to read one EM1 optical encoder module	Phidgets.com
Velodyne VLP-16 LiDAR	16-channel puck LiDAR used for scan matching and more	Velodyne
Pneumatic wheels	6” pneumatic (2)caster and (2)fixed wheels for the cart	Apollo Caster

Table 8.1: Parts used for building the Data Acquisition System

9 VITA

Vibhor Sood was born on November 10, 1992 in New Delhi, India. He is the son of Anuj Sood and Aditi Sood. He attended undergraduate school at Manav Rachna International University in Haryana, India and was awarded a B.E. degree majoring in Electronics and Communication Engineering in May 2015. He worked at Magnum Telesystem (Samsung) India for a duration of 18 months. He did his graduate work in the Computer Science department at Lehigh University and was awarded a M.S. degree in Electrical Engineering in December 2017. His masters research was performed in Lehigh's VADER (Vision, Assistive Devices, and Experimental Robotics) Lab under Professor John R. Spletzer.