6-29-2020

Design of a Robotic Inspection Platform for Structural Health Monitoring

Jason R. Soto
Florida International University, jsoto103@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

DESIGN OF A ROBOTIC INSPECTION PLATFORM FOR STRUCTURAL HEALTH MONITORING

A thesis submitted in partial fulfillment of

the requirements for the degree of

MASTER OF SCIENCE

in

MECHANICAL ENGINEERING

by

Jason Rolando Soto

2020

To: Dean L. Volakis
College of Engineering

This thesis, written by Jason Rolando Soto, and entitled Design of a Robotic Inspection Platform for Structural Health Monitoring, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Leonel Lagos

Ibrahim Tansel

Dwayne McDaniel, Major Professor

Date of Defense: June 29, 2020

The thesis of Jason Rolando Soto is approved.

Dean John Volakis
College of Engineering

Andres G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2020

DEDICATION

To my parents, who are my inspiration. Love ya both!

ACKNOWLEDGMENTS

I would like to thank various people for their contributions to this project; Dr Dwayne McDaniel, for accepting me as one of his graduate students; Anthony Abrahao, for his mentoring and enthusiasm towards robotics; and Joel Adams, for his part in creating the “jag color” ROS package.

I would also like to acknowledge the National Science Foundation and Indian River State College’s Regional Center for Nuclear Education and Training (RCNET) and DOE EM Cooperative Agreement (No. DE-EM0000598) for providing the funding and equipment to conduct my research.

Finally I thank the FIU Applied Research Center for the experience, opportunity, and for providing the work central to this thesis.

ABSTRACT OF THE THESIS

DESIGN OF A ROBOTIC INSPECTION PLATFORM FOR STRUCTURAL HEALTH MONITORING

by

Jason Rolando Soto

Florida International University, 2020

Miami, Florida

Professor Dwayne McDaniel, Major Professor

Actively monitoring infrastructure is key to detecting and correcting problems before they become costly. The vast scale of modern infrastructure poses a challenge to monitoring due to insufficient personnel. Certain structures, such as refineries, pose additional challenges and can be expensive, time-consuming, and hazardous to inspect.

This thesis outlines the development of an autonomous robot for structural-health-monitoring. The robot is capable of operating autonomously in level indoor environments and can be controlled manually to traverse difficult terrain. Both visual and lidar SLAM, along with a procedural-mapping technique, allow the robot to capture colored-point-clouds.

The robot is successfully able to automate the point cloud collection of straightforward environments such as hallways and empty rooms. While it performs well in these situations, its accuracy suffers in complex environments with variable lighting. More work is needed to create a robust system, but the potential time savings and upgrades make the concept promising.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

1. INTRODUCTION

1.1 MOTIVATION

To ensure that structures remain healthy, they must be routinely inspected. Without inspection (and subsequent maintenance) cracks, wear, and corrosion will reduce their lifespan and may even result in sudden failure. More than ever people are dependent on a complex network of infrastructure to support their modern lives. From roadways and tunnels to factories and airports, each of these structures plays an important role in society. With ever-aging infrastructure and continued expansion, the need to ensure that these structures are properly maintained only increases.

The scale of modern infrastructure poses a significant challenge to inspection. In 2012 the U.S. Energy Information Administration estimated a total of 5.6 million commercial buildings in the United States alone [1]. Inspecting even a percentage of these buildings is a monumental undertaking. While large, this number represents only a fraction of structures requiring inspection as buildings with manufacturing, industrial, or agricultural purposes are excluded from the commercial category.

In addition to the sheer scale, the hazardous nature of some infrastructure acts as a barrier to inspection. This hazardous nature drives up the cost of inspection and sets limits on how long personnel can be exposed. Examples of such infrastructure include

Figure 1.1: Example of concrete degradation [7]

high voltage substations, nuclear waste repositories, and chemical refineries. These structures can become exponentially more hazardous if a problem is detected or if they are affected by a natural disaster.

If the barriers posed by scale and safety risks can be overcome it would cut inspection costs and prevent problems that could end up costing millions of dollars in the long run. The development of a automated inspection tool to monitor the health of infrastructure is a solution to the problems of scale and safety.

1.2 SCOPE

Structural health monitoring can be broken down into three general scales. These can be summarized as survey, general, and targeted monitoring. Each scale collects different types of information, at different rates, and uses different technologies to do so.

Survey monitoring is used for understanding the health of large areas. What this monitoring lacks in detail it makes up for in coverage. Large structures such as levees, roadways, and even cities can be monitored with this method. The same techniques are also applied to evaluate the health of natural areas such as forests and the ocean. These surveys can be conducted manually though sampling or with the assistance of areal drones.

General monitoring can be defined as structural health monitoring of regions at the human scale. This can include the evaluation of the overall health of walls, columns, pipes, and buildings. Monitoring on this scale is often done manually due to the complexity and variety of environments. Personnel are outfitted with protective gear, training, and the tools necessary to perform inspections. These tools can include sensors such as thermal cameras, lidar, and specialized robots but oftentimes visual inspection is enough to detect issues. This method of monitoring typically relies on people's maneuverability and reasoning to identify problems.

Targeted monitoring focuses on small regions with the intent of collecting quantitative information. Comparing a continuous stream of quantitative information with previous history ensures that any changes or problems can be caught as early as possible. This level of monitoring scales poorly and is therefore reserved for the most critical of components. These can include piezoelectric sensors on machinery, skyscraper deflection sensors, or beam corrosion monitors. For these sensors, threshold rules and pattern recognition can be used to identify problems.

Each monitoring scale poses it own challenges, but the safety and scaling problems arise primarily at the general monitoring level. Unlike targeted monitoring, this scale lacks automated methods of data collection. The costs of hiring trained personnel and the danger posed by some environments are barriers to frequent monitoring, even though it might be desired. Developing a robotic platform that performs similar inspections to humans would prove quite useful. If successful it could increase monitoring frequency while mitigating associated health and safety hazards.

1.3 CHALLENGES

Infrastructure is a broad term and includes roadways, buildings, sewers, power plants, etc. These environments vary greatly and pose mobility and inspection challenges of their own. Monitoring such diverse environments is difficult enough for people, let alone robots.

The primary factor preventing the widespread use of autonomous robots is their inability to properly understand their environment. In environments with low uncertainty, autonomous robots have already been used to great effect. Examples include the use of autonomous drones and warehouse robots. In the air environmental complexity is low and the robot is rarely constrained in mobility. In heavily structured environments, such as warehouses, unique markers and hard coded rules provide complete information of the surroundings.

Developing an inspection robot to monitor structures means taking on the challenges of field robotics. These include physical design, localization, and navigation. To be useful for the purpose of inspection, the robot must also collect quality data. Meeting these challenges for multiple environments with one machine is difficult and leads to the cost of generalized robots being quite high. There is also the possibility that a novel combination of environmental factors will negatively impact the robots performance. As a result specialized inspection robots, such as pipe crawlers, are more common in industry as opposed to a generalized platform. While uncommon, general inspection robots do exist such as the "ANYmal" from ANYbotics [10].

Physical design is the first consideration for a mobile robot, as it dictates the terrain the robot can traverse. The physical design of a robot also places limits on the range, mobility, and sensor payload. Larger robots can operate for longer and carry more sensors, but might be unsuitable due to cost and space constraints.

The second most important feature of a mobile robot is its ability to localize in the environment. Knowing the robots current position is crucial to knowing the location of its goals. There are many techniques which can successfully localize a robot, however consistently localizing across a wide range of environments poses a challenge. A wide range of potential environments prevents hard rules or single sensors from uniquely determining position. Localizing within larger regions also adds difficulty as computational resources are taxed and the probability of the robot confusing similar regions increases.

The final key feature of successful mobile robots is their ability to navigate an environment safely. Once given a goal the robot should be able to reach it without intervention from an operator. Once again the wide array of potential environments works against the robot. Obstructions, incomplete maps, and changing terrain can all hinder the path planning of the robot. If proper attention is given to a robots physical design many of the difficulties posed by terrain can be overcome, such as tracked or stair climbing designs.

In addition to the above challenges the robot must also collect accurate data useful for inspection. The definition of useful can vary between environments but at a minimum both geometric and visual information should be collected. This information forms the basis for most general inspections and is easily interpreted by people. The quality of this data is highly dependent on the quality of sensors used, and so a balance must be struck between quality and cost. All collected information must be coherently stored and easily modified to enable future additions or corrections.

1.4 SIMILAR SYSTEMS

Structural health monitoring is useful both during a structure's construction and throughout its lifespan. Monitoring during construction can prevent accidents and produce cost savings if problems are caught early. Monitoring after construction ensures that damage and wear can be identified and corrected before they become a significant problem.

To capture the geometry of structures, lidar stations like the Trimble TX8 [25] are commonly used. These tripod-mounted lidars display exceptional range, accuracy, density, and can capture the environment at millimeter precision. Many of these stations also contain cameras, which enable the mapping of color onto the point cloud. While these stations require manual relocation, the data collected can be processed using automated systems. Software can be used to automatically extract walls, pipes, beams and other features from point clouds. These can then be checked for consistency or compared against blueprints to check for discrepancies.

Going a step further, the collection of the point cloud can be facilitated by mounting the sensors on a mobile platform. The company Scaled Robotics [18] produces robots that collect point clouds for the construction industry. These robots can be driven around a construction site to build a point cloud of the environment. Once again features in the point cloud are extracted and compared to the structures blueprints in order to detect deviations [22]. The goal of these robots is to inspect construction sites so that discrep-

ancies in the buildings construction can be detected and corrected before they affect other parts of the project. These robots typically feature one or more multi-beam lidars for point cloud collection/localization and cameras for capturing images of the environment. These robots correlate their image data with their point cloud by tagging locations and showing images captured at those locations.

Going another step further, the collection of data can be fully automated. The ARGOS challenge [24] was a competition for developing autonomous robots for the inspection of oil and gas platforms. This challenge spawned several autonomous inspection robots capable of performing multiple complex inspection tasks. The FOXIRIS, AIR-K, ARGONAUTS, LIO, and VIKINGS teams (and their respective robots) [23] participated in the finale of the challenge. These robots had to autonomously navigate a multi-level platform and perform routine inspections under adverse conditions. Monitoring tasks included reading gauges, checking valves, detecting leaks, and listening for pump wear. The winner of the 2017 ARGOS challenge was the ARGONAUTS team using a variant of the Taurob Tracker robot. The ARGONAUTS robot featured a tracked design capable of meeting the stair climbing and adverse terrain conditions. It was equipped with a thermal, rgb, and stereo camera mounted on a 5 degree of freedom arm. This allowed it to direct its cameras toward regions of interest and look over/around obstacles. It also featured a spinning dual lidar system for capturing the geometry of its surroundings.

The ANYmal-C from ANYbotics is another inspection robot that has its origins in the ARGOS challenge. Its predecessor, the ANYmal (operated by team LIO), made it to the final rounds of the competition. The ANYmal-C is a commercial quadruped robot designed for autonomously monitoring industrial facilities. Its inspection payload consists of a spotlight, thermal camera, and visual camera along with a lidar for mapping and navigation [3]. It also uses multiple structured light sensors to detect its surroundings and avoid collisions. This robot further increases its autonomous capabilities by having the ability to automatically dock with a charging station when power runs low.

1.5 APPROACH

The goal for the project is to develop an autonomous platform that can robustly map indoor rooms and hallways. The mapping should be detailed enough that damage such as the cracking and flaking of concrete should be apparent if it exists.

This environment was selected to enable the development of a generalized inspection robot within the time-frame of this project. The environment also facilitates testing and is applicable to a large percentage of structures. Limiting the robots design environment to indoors also simplifies the physical requirements of the robot as terrain is primarily flat and little if any weatherproofing will be required. Robust operation in the context of this project means that the robot is unlikely to fail while operating within its design environment.

The primary method of data representation was selected to be a 3D point cloud generated by a lidar sensor. A point cloud is nothing more than a list of xyz-coordinates specifying the location of a point in 3D space. Sufficient points can denote surfaces, with the quality of the surface being positively correlated to the density and accuracy of the point cloud. This form of data is very flexible, scalable, and is easily interpreted by most geometry visualization programs. In addition to the xyz geometric data, the point cloud will also contain rgb color data. This will add detail and make the point cloud easier to interpret.

While a remotely operated vehicle (ROV) with similar capabilities is useful, the goal is to automate the point cloud collection process. This makes data collection scalable and facilitates the monitoring of larger regions. The selection of indoor environments also facilitates the robot automation, as both localization and navigation are simplified on smooth level ground. For the chosen environment, 2D planar navigation should suffice.

1.6 CONTRIBUTIONS

Three general technical contributions are made in this thesis. The first is the design of a Robot Operating System (ROS) based colored-point-cloud collection robot constructed using off-the-shelf components. The second is the development of a ROS package (a computer program) capable of mapping camera colors onto point clouds. The final contributions are the development of a ROS package for recording ROS colored point clouds. For a brief overview of the Robot Operating System (ROS) refer to section 3.1.1.

The first contribution was the development of a ground robot capable of maneuvering indoors and mapping the interior of rooms. It is capable of capturing 360 degree colored point clouds and autonomously navigating to new locations while scanning. The robot is designed to operate in environments which are both geometrically and visually rich and have a relatively level ground plane. The ideal environment also assumes that all objects considered obstacles (to the robot) intersect the scan plane used for obstacle avoidance. The robot can operate continuously for approximately two hours and the best performance can be obtained if an accurate 2D map of the environment is loaded into the robot. If no existing map is available the robot can generate its own, though it is recommended to correct this map using loop closure and/or other methods.

The second contribution is the "jag color" package which was developed at the Florida International University (FIU) Applied Research Center (ARC). This package allows ROS image topics to be mapped on to point clouds regardless of where on the robot the camera and point cloud sensors are placed. The only caveat is that the camera Field of View (FoV) must intersect the generated point cloud.

The last contribution, the "pc2 record" package, was developed in order to manage the recording of large point clouds. This package was also developed at FIU ARC and works with ROS to record colored point clouds and store them as x,y,z,r,g,b columnar

data in a text file. The program operates by writing data continuously so as to prevent large point clouds from being stored in memory.

2. THEORY

This research effort aims to develop a robot that can autonomously perform indoor structural health monitoring. Defects in concrete normally detectable by people should also be detectable in the point cloud captured by the robot. To achieve this, the robot must have accurate localization, data collection, and reliable navigation capabilities. This chapter describes the theory behind implementing those capabilities and acts as a guide for the development of this effort.

2.1 LOCALIZATION

Localization is a robot's ability to determine its pose (position and orientation) relative to the environment. Self localization is critical for robots as knowing their current position establishes a relation towards potential goals. Most mobile robots require some form of localization to be able to complete tasks autonomously. Even a simple floor vacuuming robot must at least recognize its home station and navigate towards it to recharge.

Localization begins with an estimate of the robot's starting location. This could be the last known pose, a hard coded value, or an estimate based on the surroundings. If an initial estimate cannot be obtained then, by convention, the estimate is set to zero. With the initial estimate established, the next step is obtaining the current position of the robot.

2.1.1 APPROXIMATE LOCALIZATION

If no environmental relations can be established, then internal relations can be used as long as those have correlations to the environment. This is called dead-reckoning localization and relies on integrating velocity and acceleration measurements to estimate position [20].

Acceleration and velocity measurements can be obtained by many sensors, the most common being encoders and inertial measurement units (IMU's). Encoders operate by

counting marks on an encoder wheel and correlating the number of marks to a displacement or rotation. IMU's can measure linear, radial, gravitational, and even magnetic acceleration. This form of localization does not use knowledge of the environment and so localization can only be related to the initial position of the robot.

Calculations of dead-reckoning localization are very fast due to the simple algorithms used. This produces smooth and continuous localization which is good for applications like autonomous navigation, at least in the short term. Due to the nature of integration, small errors in the velocity and acceleration measurements accumulate over time causing the position estimate to drift. Without a means of correction, this error can grow unbounded [20] and eventually results in the localization estimate becoming unusable. Approximate localization is highly dependent on the quality of the sensors in use. The smaller the error the longer the robot can operate using only this form of localization.

Both encoder's and IMU's are sensors commonly found on robotic platforms. Depending on their accuracy, the combination can produce basic functionality and continuous localization.

2.1.2 ABSOLUTE LOCALIZATION

Absolute localization operates by estimating the absolute pose of a robot relative to its environment. Because the measurement is derived from features of the environment, error is bounded to that of the localization process and the sensor used for measurement. There are four general methods for obtaining absolute localization; these are fields, beacons, observation, and SLAM (Simultaneous Localization and Mapping). At their core, these methods rely on comparing incoming data to an internal representation of the environment.

Field based localization operates by comparing the strength and direction of forces to an internal model. IMU's and magnetometer's can be used for this but only if the robot and its environment meet specific criteria. If the robot is constrained to the ground

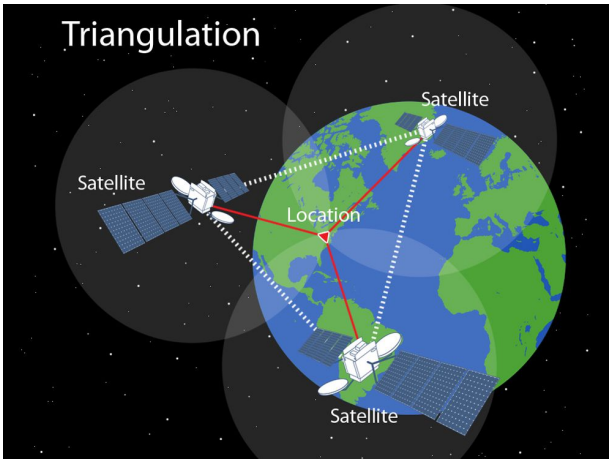


Figure 2.1: Satellite triangulation [8].

plane, the downward acceleration of gravity can be used to identify the absolute roll and pitch of a robot. Similarly, the direction of the earth's magnetic field can define the absolute yaw. This form of localization also includes altitude/depth estimates derived from pressure gradients.

Beacons operate by emitting pulses of information at regular intervals. These pulses are timestamped and a robot receives them at varying times depending on the distance to the beacons. A beacon-based setup must utilize a minimum of three beacons to enable localization. By knowing the position of each beacon, and comparing when signals were sent, the robot can triangulate its position. Beacon type sensors include GPS, router, cellular, and acoustic position systems. These systems are mostly used to obtain xyz positioning, but can provide orientation if a second receiver is used.

Observation based localization is pose information derived from an external source. This external observer then relays the information to the robot. A simple example would be an overhead camera observing a planar robot. In this case the (x,y) positioning of the robot would correspond to the (u,v) pixels on the camera. This technique is commonly used in the movie industry to localize the movement of an actor wearing a special suit. The external camera system recognizes the spherical markers and stores the positioning data of each. By knowing the position of the observer(s) in the environment the absolute

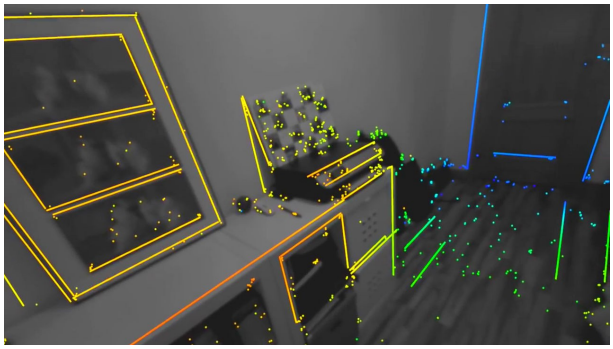


Figure 2.2: Example of visual SLAM feature detection [4].

pose of a system can be calculated. Sonar and radar are also examples of observation-based localization, as they can be used to determine the coordinates of boats and planes.

Lastly, SLAM techniques operate by observing and recording features in the environment. A SLAM program records sensor data, identifies key features, and compares them to an internal model. SLAM systems typically store the more prominent environmental features such as the positioning of walls and corners. If the incoming data displays the same (or similar) features to the internal model, then the robot must currently occupy a pose which reproduces those features. If new features are detected, then these are added to the internal representation. Many forms of SLAM exist including visual, lidar, and sonar based SLAM. Unique visual markers in the environment (such as QR codes) can also be used for slam, but these must be set up in the environment beforehand.

While absolute localization guarantees bounded error for a know map, it cannot guarantee the actual accuracy of said map. During SLAM the internal model of the robot is built up over time. Because SLAM sensors rarely see the entire environment at once, multiple views must be combined to create a map. This combination process introduces error into the map in a similar fashion as dead-reckoning. To correct this error a process called loop closure can be implemented.

Loop closure is the process of correcting an environment representation (a map) based on whether a location has been previously observed [9]. There are two compo-

nents to loop closure; detecting whether a location has been previously observed, and altering the internal map. If two regions have been previously observed loop closure forces them to overlap by modifying the map along the path looping back to the observed region.

For the purposes of this research effort, all localization equipment must be located on the platform. This simplifies the robot setup by making the system self contained and also prevents the need to setup localization infrastructure at new locations. This does not exclude global infrastructure such as GPS or the earth's magnetic field but, due to the design environment (indoors), these method are less reliable. GPS has trouble penetrating roofs while magnetometers can be affected by large metal fixtures, which can distort the local magnetic field. This limits the applicable absolute localization methods to SLAM and gravity. Of the SLAM sensors, the two most practical for indoor use are lidars and cameras.

2.1.3 SENSOR FUSION

Sensor measurements are not perfect. Noise in both the sensor readings and computations add uncertainty to a measurement. In addition, sensors often provide incomplete pose data, requiring multiple sensors to properly localize the robot. Sensor fusion is the act of combining multiple sensor measurements to obtain a better localization estimate. These multiple measurements can originate from the same sensor, multiple sensors, or a mathematical model of the system.

The simplest form of sensor fusion involves averaging the returns of a single sensor. This effectively lowers the sensor frequency but results in "smoother" sensor behavior. More complex sensor fusions use multiple types of sensors along with knowledge of physical systems to further improve localization estimates.

One of the best known methods for fusing sensor data is the Kalman filter. This algorithm can be used to combine sensor signals, knowledge of sensor error, and a mathematical model of the robot to estimate pose. The Kalman filter works by first

passing knowledge of the robots latest "known state" (position, velocity, acceleration) through a mathematical model representing the robot. This model contains knowledge of the robot's mass and inertia; allowing it to predict the next state of the robot. This prediction is treated as a measurement and combined with new sensor measurements to calculate the most likely pose of the robot. Many versions of the Kalman filter exist including (but not limited to) the original Kalman Filter (KF), Extended Kalman Filter (EKF) [13], and Unscented Kalman Filter (UKF).

Kalman filters work well for fusing data, but rely on the assumption that the incoming data (including knowledge of uncertainty) is correct. Sometimes, whether due to damage or internal error, a sensor measurement is drastically wrong. A worse situation occurs when a sensor returns erroneous data while functionally intact. This failure mode is called sensor degeneracy and results from a sensor's inability to properly interpret the environment.

Sensor degeneracy occurs when a sensor cannot properly interpret reality due to the nature of its environment. In localization this means that the position, velocity, or acceleration estimates no longer reflect reality. A perfect example of such a phenomenon would be the loss of traction while relying on encoders. If traction is lost, such as on mud or ice, the wheels of a robot will spin yet result in little to no translation. The encoders do not detect that the wheel is slipping and so would return a velocity even though the robot is not moving. More complex sensors, such as tracking cameras and lidar, can also experience degeneracy. In environments with repetitive features, the sensor could interpret seeing the same features as undergoing no motion. Environments with many moving features can also cause problems as small shifts in feature position cause SLAM to converge to different position estimates. If enough features move in one direction, the robot may interpret this as motion even though the platform remains static.

These cases require the temporary removal of the sensor or specific degeneracy direction [27] from the fusion equation. To mitigate the impact of sensor removal on localization, the robot should carry redundant sensors to compensate while degeneracy

persists. Determining whether a sensor is degenerate can be difficult, especially if it can happen at any time. If enough different sensors measure a single property (such as x velocity), then it becomes possible to determine the degenerate sensor using statistics. Alternatively, one can rely on an inherently robust sensor (such as an IMU) to act as the baseline and compare it against other less robust measurements.

Degeneracy filtering is a form of sensor fusion that removes bad sensor measurements from localization estimates. To create a robust robot, both sensor fusion and degeneracy filtering should be utilized.

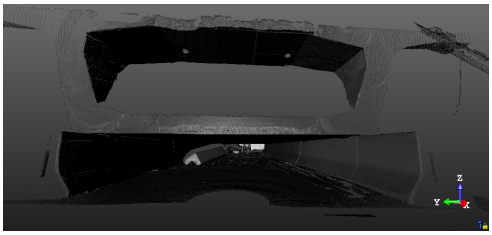
2.2 POINT CLOUD COLLECTION

2.2.1 GEOMETRY CAPTURE

To perform structural health monitoring, this research effort aims to collect information on the geometry and location of structural features. If geometric resolution is accurate and dense enough, then it becomes possible to identify certain forms of structural damage by analyzing the quality of collected surfaces. Geometry also provides an intuitive representation of the environment, making it easy for operators to understand.

A basic but very useful form of geometric representation is the point cloud. A point cloud is a list of xyz coordinates representing points in space. In a point cloud the shape and surface of objects are left to the interpretation of the operator. While not as storage efficient as other data formats, point clouds are very flexible and easy to modify. This makes them an ideal data structure for continuous updates. More detailed point clouds contain additional fields such as “i” for “intensity” resulting in a xyzi point cloud. Point clouds are also useful because their simple nature allows other geometric representations (such as meshes, surfels, and voxels) to be constructed from them.

Lidar is perhaps the most straightforward approach for capturing a point cloud. Lidar sensors operate by emitting a pulse of laser light and then analyzing the return signal bouncing off a surface. Using the time of flight (or beam spread) the distance to the



(a) Lidar

(b) Photogrammetry [16]

Figure 2.3: Example point clouds captured with lidar and photogrammetry.

surface can be calculated. This pulsed laser can be swept across the environment, resulting in a planar (2D) point cloud. If the lidar is swept through an additional axis it produces a 3D point cloud. Multiple laser emitter/receiver pairs can be part of the same lidar, thereby, increasing the rate of data collection. Lidar works well on matte surfaces due to the reflected laser light spreading evenly, ensuring some makes it back to the receiver. Highly reflective or absorbent materials pose a challenge because the beam is either deflected away or absorbed completely by the surface.

Photogrammetry is another method of point cloud collection and operates by extracting 3D data from images using stereo vision. This technique works by comparing two images and identifying the same features in both. By knowing the transform between two camera reference frames, and observing the resulting displacement features within the images, each feature can be converted into xyz data. The density of this point cloud is determined by the number of features in the image. If the object in question has no distinguishing visual features, photogrammetry will not work. This can be remedied by adding visual features to the surface with paint or projecting these features using light [2].

Each of these methods generates a point cloud, but a single point cloud capture is unlikely to contain enough data about the environment to be useful. Parts of the environment are often occluded and some surfaces are too distant to yield sufficient detail. To remedy this, multiple point clouds must be collected and combined to for sufficient representation of the environment. Point clouds can be combined if the spatial

transformation between the clouds is known, allowing the points to be shifted from one reference frame to another. This transform can be obtained from physical measurements, knowledge of the scanning system, or localization estimates. Certain rigidly defined devices, such as scanning tables, can construct the transformation using solely encoder and servo positioning.

If the transform is unknown or inaccurate, then it must obtained by a process called registration. Registration works by taking two point clouds, comparing them, and finding the spatial transform which minimizes the error between them. Registration programs often extract planar and spherical surfaces from the point clouds to facilitate finding the correct transform.

A shortcoming of the registration process is that it requires the point clouds in question to have overlap. This ensures that the algorithm can converge without attempting to match unrelated features. In most cases registration algorithms also require a rough estimate of the desired spatial transform in order to function. This speeds up the registration process by lowering the number of iterations required and constraining the possible point cloud orientations.

For this project, a procedural mapping technique will be used to capture point clouds. This method will obtain individual lidar scans and combine them using transforms obtained from localization. Lidar was chosen instead of photogrammetry due to the sensors usually being more accurate at longer ranges. Many lidar sensors also have the benefit of measuring a complete 360 degrees, allowing a complete capture of the surroundings with a single sensor.

2.2.2 COLOR CAPTURE

Capturing the color of a structure can add another dimension of information to the data. Problems such as corrosion and weathering cannot be detected easily with geometry, but stand out readily in images. Collecting color data would allow operators to make more informed decisions about the health of a structure. The most direct approach would

involve only the collection of images. This would preserve the full detail of the images but would split the data into separate categories. A more integrated solution is needed.

The most common way images are combined with geometry is through texture mapping. This process overlays images onto polygons constructed from a point cloud. This technique is standard in the gaming industry and is able to combine geometry and image data regardless of resolution disparities. A less used, but more straightforward, approach is the direct mapping of color onto point clouds. This technique be done procedurally without the need to extract polygons from the point cloud.

The simplest method for mapping color to a point cloud is directly by using a pinhole camera model. This is an idealized camera model where light rays pass through a infinitesimally small aperture and are projected to scale within the camera. The resulting image can be linearly correlated to the point cloud as long as the spatial transform between the camera and lidar is known. This makes it straightforward to determine which pixels map to specific 3D points. Other color mapping approaches exist, such as correlating images to point cloud intensity images [12], but the pinhole camera technique has the advantage of not requiring registration.

The pinhole camera model is an idealized system, and while pinhole camera apertures exist they are uncommon and still only an approximation. More often cameras use some version of a lens to focus light, and depending on the curvature of the lens it can distort an image far from the idealized model. This issue can be corrected using a technique called camera calibration which mitigates most of the distortion. During camera calibration, markers of known size and configuration are held in front of the camera. By observing the distorted markers and comparing them to their real shape a matrix that corrects the original image can be created. This is called a rectified image and more closely resembles the pinhole model.

Once a rectified image is obtained it is possible to extract the appropriate pixel color. This can be done by transforming the point cloud into the reference frame of the image. This transform can be obtained by measuring the distance and rotation between the point

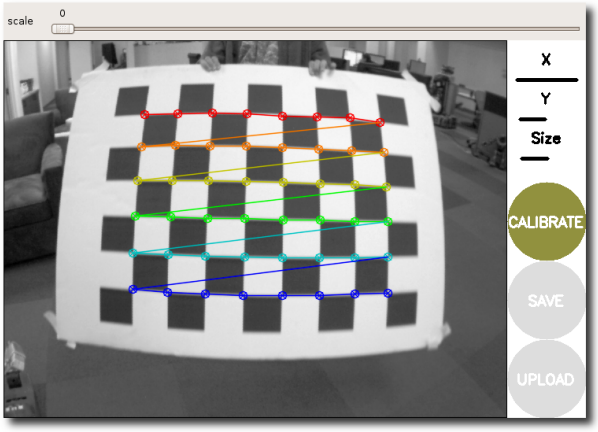


Figure 2.4: Example of camera calibration grid [14].

cloud source and camera origin. Points that fall outside the FoV of the camera can be excluded from the process. Using the image resolution a grid representing the pixels can be projected into the 3D space following the camera FoV. Where the projected grid intersects a point(s), then the corresponding pixel color can be attributed to said point(s).

Most cameras have higher resolutions than lidar; therefore it is likely that important image information could be missed. Additional image data can be added to the point cloud using interpolation. If additional points are interpolated at every pixel location, they can be used to record color just like regular points. This has the drawback of creating "false" geometry within the point cloud which should be distinguished from properly captured points.

2.2.3 POINT CLOUD MANAGEMENT

Point clouds are very flexible data structures, but can be inefficient for storing large amounts of information. With the robot intended to collect dense colored-point-clouds, it is likely that several gigabytes of data can be generated per scan. If the robot is to capture multiple point clouds autonomously this data must be managed so that it does not adversely affect the robots performance. Additionally the raw data can be processed to improve both its quality and storage characteristics.

As the robot scans the environment it will inadvertently collect point clouds of uneven density and duplicate data, especially if multiple passes are necessary. Voxelization [26] is a form of re-sampling meant to reduce the size of a data-set by allotting no more than one point per specified unit of volume. This filter divides the entire volume of a point cloud into small cubes (voxels) of a desired size. This is typically done using an octree, which selectively subdivides the point cloud depending on whether larger cells contain points. The final subdivision of an octree is called the "leaf" and is equivalent to a voxel. If multiple points land within a voxel they can be averaged together, resulting in a single point. If the center of the voxel is used instead, it decreases the accuracy of the decimation but converts the unstructured point cloud into structured data. Structured data has many uses, one of them being further compression to reduce file size. Depending on the desired resolution this process can substantially reduce file size at a cost to surface quality.

Smoothing is the process of mathematically adjusting point normal's to improve the surface quality. A point normal is the direction orthogonal to the "surface" implied by a point cloud (normal to the surface). This technique operates by shifting point locations so that there is less variation between neighboring point normals. This can help smooth noisy data or subtle imperfections caused by registration. While this method can arguably increase surface quality, it removes the finer details from a point cloud, which is not always desirable.

Sometimes the point cloud capture process is met with an error. A lidar beam could reflect improperly off a surface or stereo vision could match incorrect features. This will cause an erroneous point to be recorded by the sensor which does not represent real geometry. Often these erroneous points deviate significantly from the nearest neighbors, making them possible to detect and remove using a statistical outlier removal (SOR) filter. This can improve point cloud quality but can also remove real data as well.

The above processes require the point cloud to be compared against itself. As such, the point cloud in question must be held in the memory, which would consume the

platforms resources. To avoid resource issues any post processing techniques should be applied to the point using resources external to the robot.

2.3 AUTONOMY

To increase the frequency of infrastructure monitoring automation is necessary. This will ensure that the robot can collect information without the need of an operator present, thereby saving time. In the field of autonomous driving there exists a 5 tier scale that categorizes the level of autonomy a system displays. Based on this scale, the robot should achieve level 3 automation (conditional automation) in order to meet development goals. This level of automation describes a robot that can navigate autonomously within a specified environment. For conditional automation a human is still required for some driving tasks as well as general supervision.

To achieve this level of automation, the robot must be able to plan a path though the environment. Path planning can be done by using a path-planning algorithm [5] in combination with a known map of the environment. These map representations can either be geometric, metric, or topological [6]. For robots restricted to the ground plane, 2D metric maps are often used to represent non-traversable obstacles. These maps are commonly used instead of 3D maps as they are easy to compute, recognize, and create. More complex path planning systems separate the known map into "global" and "local" components. While the global map computes the general path of the robot, the smaller local map allows path segments to be recomputed efficiently in case obstacles or people change position.

Many path planning algorithms can be used to navigate a known map, but most employ some form of a cost map [21] to determine the optimal path. A cost map is a representation of the environment where discrete locations in space have a cost associated with the "difficulty" of traveling through that location. A 2D cost-map typically resembles an image where black pixels are obstacles (having infinite cost) and gray-

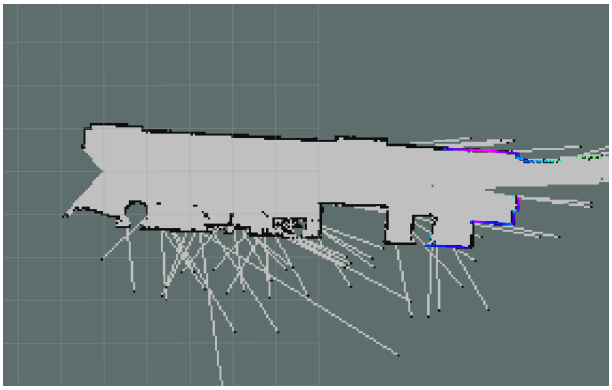


Figure 2.5: ROS 2D metric obstacle map.

scale represents a cost gradient of difficulty. If a part of the map is known to be more "difficult" to traverse than another (muddy terrain, steep slope, closeness to an obstacle), it can be represented as an increased cost. By calculating multiple paths though the cost-map, the path-planning algorithm converges on the path with the lowest cost. This is typically the shortest path which avoids terrain difficulty. By varying the weights of the cost map, a robot's behavior can be modified to traverse more "costly" terrain if it provides sufficient savings in path length.

Path planning algorithms work on a known map of the environment, but can still operate if this map is unknown. If a SLAM algorithm is used in conjunction with a path planning algorithm, the SLAM can continuously update the map as the robot traverses the environment. This is more resource-intensive than a single calculation as it requires the path to be recomputed with each map update. This capability is useful for exploring unknown regions and finding cost effective paths.

Path planning algorithms work well, but their performance is heavily dependent on the sensors used to identify obstacles and terrain. If the proper algorithms and sensors are not implemented, a robot will fail to identify traversable and non-traversable terrain. For ground robots it is common to implement a 2D lidar for map making and obstacle detection. This sensor provides a good understanding of the environment in the xy-plane, but has a very narrow understanding of the z-axis. Obstacles below or above the lidar scan plane are completely invisible to the robot and navigation using only this sen-

sor could result in damage to the robot and the surroundings. Furthermore, basic lidar obstacle detection does not discriminate between sensor readings. This can lead completely benign terrain, such as grass stalks or access ramps, to be treated as impassable obstacles.

3. SYSTEM DESCRIPTION

This chapter describes both the physical and software components implemented on the robot. These components were incorporated in order to meet the localization, mapping, and navigation goals outlined in Chapter 2. The combination of these systems describes a robot which is able to autonomously navigate indoor environments and collect colored point clouds.

3.1 BASE PLATFORM

3.1.1 ROS OVERVIEW

Before describing the robot it is important to understand the underlying system which allows its operation, the Robot Operating System (ROS).

ROS is a middle-ware created with the goal of facilitating robot development. ROS resides "on top" of an operating system and manages communications between sensors and programs by implementing a node structure for every program. These nodes pass information to one another using standardized messages and are each programmed with a specific task in mind. This structure promotes code re-usability and allows for programs which were developed independently to work together.

The versatility of ROS has allowed for the development of a large community codebase which makes it easy to add functionality to a robot. Because of its wide spread use many manufactures create ROS packages (programs) for their sensors and robotic platforms. A ROS package is the most atomic (fundamental) component of a ROS system that provides useful functionality. These packages typically deploy one or multiple nodes with the intent of accomplishing a specific task. If multiple packages are meant to work together, then these can be grouped into a ROS meta-package.

In addition to the availability of ROS compatible code, the program includes many tools to facilitate testing and development. One of the more prominent tools is called

RVIZ (short for robot visualization) and allows an operator to visualize robot data such as point clouds, images, and maps. Another key feature of ROS is that it can support distributed processing across multiple computers as long as they all share a local area network (LAN).

While using ROS has many benefits, the program is not suitable for all applications. ROS requires an operating system to run, limiting the applicable systems to those incorporating full-on computers. ROS is also not built for security, as anyone with the authority to run ROS code also has the ability to view and manipulate it. Finally, ROS is unsuitable for real time applications such as flight controllers which must guarantee a response within narrow time constraints.

For this project none of the ROS limitations apply, making it a good system with which to control the robot. Small computers, such as the Raspberri-Pi, allow the incorporation of ROS into many robotic systems. Security is not a concern, as access to the robot will be very limited and the platform is still under development. Lastly, the project does not require the meeting of real-time processing constraints.

3.1.2 BASE PLATFORM

The starting point for this project was a Clearpath Jackal robotic platform. This platform was not chosen in order meet specific design criteria. Rather, this platform was available at the start of the project and its capabilities guided the development of the robot.

The Jackal is a relatively small robotic platform measuring 508x430x250 mm and weighing 17 kg (see Appendix A). The platform maneuvers using a differential-drive system with one motor driving two wheels on either side of the robot. Included with the base platform was a SICK-Tim551 LiDAR, DualShock-3 joystick controller, and GPS receiver. This platform contains an internal computer running Linux 16.04 and can be controlled using the Robot Operating System (ROS).

Like many robotic components, the Clearpath Jackal comes equipped with ROS packages that provided basic functionality. This primarily consists of a “jackal base”

Figure 3.1: Jackal base platform [17].

and “jackal control” packages which enable control of the robot via the DualShock-3 controller. The ROS node structure of the base platform is depicted in Appendix B.

To obtain more control over the platform a wireless router and Linux laptop were added to the ROS system. By connecting both the laptop and the robot to the router a local area network (LAN) was created. This allowed access to the robot via the laptop using the Secure Shell (SSH) program. Running ROS on the laptop also enabled ROS features such as the RVIZ program.

3.2 LOCALIZATION

In ROS the odometry message is used to describe a robots localization estimate. This message consists of position (pose) and velocity (twist) vectors, each with linear and rotational components. Additionally, the messages include a covariance matrix, which estimates the uncertainty and correlation of the vectors.

To begin, the localization estimate generated by the default Jackal platform was evaluated. This estimate was generated by using the “robot localization” ROS package to combine wheel encoder and IMU measurements. This package is able to combine an arbitrary number of sensor measurements into a single estimate using Kalman filters. While the default localization produced continuous odometry the estimate drifted over time due to lacking a fixed reference frame.

Figure 3.2: Sick TIM551 2D lidar [19].

To obtain better performance from the "robot localization" package an absolute measurement is required for each pose component along with at least one other velocity measurement. If multiple absolute pose measurements are used for a vector component, it degrades the performance of the package over time. To produce more accurate localization, the existing encoder measurements were combined with new IMU, lidar, and visual SLAM measurements.

From the base platform the encoder estimates were able to provide z-rotation and x-linear velocities. The default IMU however produced noisy measurements, therefore it was replaced by a LORD 3DM-CX5-25 IMU. The new IMU, like its predecessor, provided xyz-rotational velocities and was selected due to its high accuracy and available ROS package.

To obtain absolute position measurements the SICK TIM 551 lidar mounted on the platform was enabled (Figure 3.2). This 2D lidar is able to provide xy-linear and z-rotation positions when coupled with a SLAM algorithm. Three SLAM ROS packages "cartographer", "gmapping", and "hector mapping" were tested to determine which would yield the desired performance. The SLAM package should provide good utility, produce frequent pose estimates, and generate accurate feature maps.

The cartographer package yielded the best performance over all by providing good localization and accurate feature maps. The package achieved this by using loop closure and obstacle correction to generate accurate maps of the environment. Loop closure is a map correction process that joins two regions of a map if they have been previously observed. This reduces the localization error of a map significantly by resetting the map drift at loop closure locations. The obstacle correction also improved the feature map by removing temporary obstacles if they were no longer observed. These features were very useful but proved too resource intensive for the platform.

Upon testing the "gmapping" package it demonstrated robust map-making capabilities but converged on map updates quite slowly. This slow convergence produced "jumping" localization which, while accurate, would have yielded poor performance for the planned procedural-scanning technique. The package was also relatively simple and lacked additional functionality besides localization.

The "hector mapping" package performed similarly to "gmapping" but converged on map updates much more quickly. This made the localization less robust but provided quicker pose estimates. Through testing it was discovered that rapidly rotating the platform could cause the generated map to distort. Due to the slow robot velocities required by the planned procedural-scanning technique, it was determined that the error was not a problem. This package was implemented into the robot as it provided the best balance of resource use, utility, and update rate. A 2D metric map generated by the "hector mapping" package can be seen in Figure 3.3. Due to lacking loop closure the start and endpoints of this map are misaligned.

With three pose components defined, a second source of absolute localization was required to provide the remaining three. To complete the pose estimate a RealSense 265 Tracking camera (RS-T265) was obtained and integrated into the robot (see Figure 3.4). This sensor uses visual SLAM to estimate its pose in space and was chosen for the small form factor and dedicated ROS package. Adding this kind of sensor had the added benefit of providing absolute localization using a sensor different from lidar.

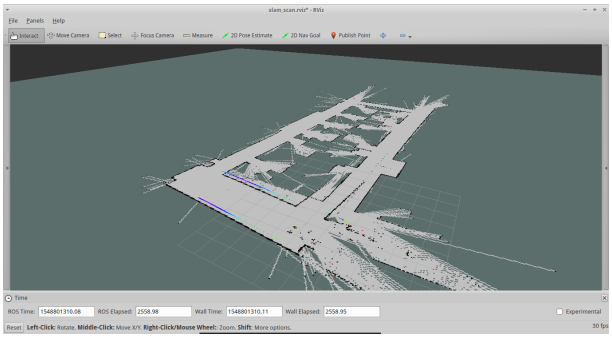


Figure 3.3: Map generated by hector mapping package.

Figure 3.4: Intel RealSense T265 tracking camera [11].

Unlike the 2D lidar, the RS-T265 sensor localizes using 3D information. This means that the sensor is able to provide a fully defined pose (xyz-linear and xyz-rotation) by itself. Instead of using the entire pose, only the z-linear and xy-rotation components were selected for the sensor fusion. This was done to prevent duplicate pose measurements, which would have negatively impacted the performance of the "robot localization" package. This sensor also had the benefit of providing a complete set of velocity estimates (xyz-linear and xyz-rotation) which were also added to the sensor fusion.

The RS-T265 generated very consistent data and so was considered for providing xy-linear and z-rotation pose components instead of the lidar. After testing the behavior of the system using this configuration, the option was decided against due to unsatisfactory behavior of the sensor. Firstly, the accuracy of the visual SLAM was lower than the lidar SLAM over longer distances. Secondly, the feature maps created by the lidar SLAM could be saved and reused by other programs as opposed to the purely internal RS-T265

feature map. Lastly, the RS-T265 implemented a variant of loop closure that repeatedly "teleported" the robot localization estimate instead of altering the feature map.

With the integration of the RS-T265, the absolute pose was fully defined. This sensor, along with the encoders and IMU, ensured that velocity estimates were obtained for each pose component. Combining these measurements using the "robot localization" ROS package yielded a fully defined odometry estimate.

3.3 POINT CLOUD COLLECTION

The objective of this project is to successfully map infrastructure so that operators can inspect the generated model for damage. To achieve this it was determined that the robot should collect high-resolution colored-point-clouds. The first step in collecting a colored-point-cloud is the collection of a non-colored one. Photogrammetry and registration techniques were considered, but for this project, a procedural mapping method using lidar was chosen.

The SICK-Tim551 mounted on the robot is a planar lidar oriented in the xy-plane. Because the robot travels on the xy-plane, this lidar is minimally useful for obtaining the geometry of the surroundings. To capture the surroundings a second lidar was needed, preferably one with a 360 degree FoV and mounted on the yz-plane (relative to the robot). In this configuration the lidar profile could sweep completely through the surroundings as the robot traveled forward (in the x-direction).

To collect this point cloud a RPLIDAR-A2 lidar (Figure 3.5) was obtained and mounted onto the robot on the yz-plane. This lidar was selected for its dedicated ROS package, high accuracy, and high scan rate relative to its price point. To attach the RPLIDAR-A2 to the robot, a mount was designed that would not interfere with the SICK-Tim551 and would enable future camera placement. The lidar was attached vertically at the end of the mount as this configuration provided the best point density for

Figure 3.5: RPLIDAR A2 lidar sensor.

mapping. With the sensor installed and the "rplidar ros" package operational it was possible to collect point clouds and observe them using RVIZ.

Due to using a single 2D lidar for data collection, the density of the point cloud was highly dependent on the velocity of the robot. The faster the robot moved, the sparser the data, and so a balance had to be struck between cloud density and scanning speed. Through testing, a robot velocity of 0.05 m/s was eventually selected and used for tests as the speed produced relatively dense scans while still moving the robot at a reasonable (if slow) pace.

To obtain colored point clouds a RealSense D435 (RS-D435) structured-light camera was mounted on the robot. This device produces colored point clouds by projecting infrared points onto a surface, extracting depth using stereo vision, and overlaying color using an rgb camera. While the sensor yielded great accuracy at close distances, point cloud quality rapidly declined at distances over a meter. This limited the usefulness of the sensor, and led to the decision of mapping color onto the RPLIDAR point cloud.

In order to project images onto the collected point cloud, cameras were required to collect the color data. To provide complete coverage of the 360 degree RPLIDAR-A2, four ELPUSBFHD01ML21 cameras were obtained. These cameras were selected for their FoV of greater than 90 degrees and USB interface. Mounting these cameras

orthogonally around the robot's x-axis provided complete coverage of the FoV of the lidar. Four cameras also allowed each one to pay attention to an important plane; ground, ceiling, left, and right walls. ROS operation was enabled on these cameras though the use of the "usb cam" ROS package. This package converts the video output of USB cameras into ROS Image messages.

With cameras installed it was necessary to correct for lens distortion using rectification. Developing a rectification algorithm was outside the scope of this project and so the ROS "image processing" meta-package was used. This package can be used to both calibrate and rectify images so that they better approximate a pinhole camera model. By using identical cameras with identical lenses the calibration process only needed to be performed once. The resulting calibration parameters were saved and reused for the rectification of the other three cameras.

To project the rectified images onto the point cloud, the "jag color" package was developed. This package operates by utilizing a pinhole camera model to correlate points in 3D space to pixels in an image. This package was developed at FIU, as literature review yielded no existing ROS packages with the desired functionality. The "jag color" package is a key development as it allows any rectified ROS image to be mapped onto a point cloud. This means that other image sources such as thermal, spectral, and radiation can all be mapped to a point cloud as long as they can be converted to a rectified ROS Image.

Running four instances of the "usb cam", "image rect", and "jag color" packages consumed a lot of resources and bogged down the robots primary computer. It was determined that a second computer was necessary to perform the color mapping process. The computer selected for this task was an Nvidia Jetson-Nano (Figure 3.6) due to its small form factor and high processing power. The computer was enclosed in a custom case for protection and mounted on the top of the robot. To support this addition a wireless access point, network switch, and additional power distribution system were also incorporated into the Jackal. This upgrade added the computer to the ROS network

Figure 3.6: Nvidia Jetson Nano computer [15].

and effectively provided the robot with an additional 2 USB channels, 4 cores, and 4 GB of RAM.

With the upgrades in place it was theoretically possible to collect colored-point-clouds, however, a couple of technical challenges had to first be solved. Firstly, all the cameras had to be reduced in resolution down to 320x240p. This is due to a Linux limitation preventing large bandwidths of data from passing though USB interfaces. While USB 3 interfaces have plenty of bandwidth to support four 640x480p cameras, this Linux limitation prevented the interfaces from operating properly

The second challenge was the addition of a timeserver to the system. The Jetson Nano computer lacks a timekeeping battery and so becomes desynchronized in time from the main platform when powered off. If there is no internet when the robot is powered back on this discrepancy in time prevents many ROS processes from functioning correctly. To remedy this, a timeserver was added to the Jackal robot. This allowed the Jetson Nano to fetch time from the Jackal if there was no internet available.

With the upgrades complete it was possible to move all image related processing to the Jetson Nano. Running at full capacity, the color mapping process consumed about 50 percent of the resources on the Jetson Nano. The completed Jackal robot can be seen in Figure 3.7.

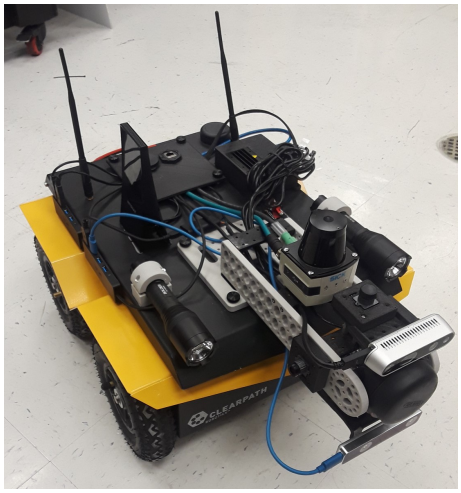


Figure 3.7: Final Jackal Platform with incorporated sensors.

3.4 DATA MANAGEMENT

To manage the collected point clouds, a method of processing and recording the point cloud information was needed. ROS natively supports recording information as "bag-files" (a format for saving streamed data), but this is not an ideal solution for the application. Bagfiles are an uncommon datatype for storing point clouds, and are more often used for completely reconstructing a robot's behavior. ROS also features an in-built point cloud exporting method which saves point cloud messages using the ".pcd" file format. This format is a more common file-type for point clouds, but unfortunately the inbuilt method stores each generated message as its own file. This creates thousands of point clouds that would later have to be combined into a single file.

To obtain the desired point cloud storage functionality, a point cloud recording package called "pc2 record" was created. This package was developed at FIU and transforms a PointCloud2 ROS message into xyzrgb columnar data written to a text (.txt) file. The conversion is performed by first separating the ROS message into xyz and rgb data. The rgb data is then unpacked into individual r,g,b values and cast to the uint8 datatype. The x,y,z and r,g,b values are then procedurally written to a text file. The writing is done continuously to prevent data storage in memory. This allows the recording of large point cloud datasets without depleting the computational resources of the platform.

With the raw point cloud data recorded, it became possible to perform post-processing to improve its characteristics. Using the Meshalab and CloudCompare programs, several post processing techniques were evaluated including meshing, smoothing, decimation, voxelization and filtering. While many post processing techniques were tested, it was determined that most were counterproductive to the goals of the project. Many of the techniques altered the surface quality of the point clouds, which would result in a loss of finer detail. The most useful process for the collected data was the application of statistical outlier removal (SOR) filters. These cleaned up the data by removing many erroneous points without greatly affecting the overall point cloud.

3.5 AUTONOMY

To enable autonomous navigation the "move base" ROS package was installed on the robot. This package is part of a well know meta-package called "navigation", and provides basic 2D navigation capabilities to robotic platforms. The "move base" package detects obstacles using an input LaserScan message. When an obstacle is detected, the program assigns high cost-map values to the regions around it (blue regions in Figure 3.8). This prevents the robot from planning a path too close to the obstacle and risking collision. If a map topic is also available, then the package can calculate a global cost-map for longer range navigation.

The "move base" package allows navigation from the robot's current position to a specified goal. This is very useful, but does not allow complex routes (such as scanning multiple rooms) to be automated. To solve this problem, the "follow waypoints" ROS package was installed. This package stores a list of goal points and publishes the next point to "move base" as each goal is reached. This enables a robot to enter multiple rooms and complete circuits by returning back to its starting position.

Finally, to improve mapping repeatability, the "map server" ROS package was installed. This program enables the saving and loading of obstacle maps, allowing maps

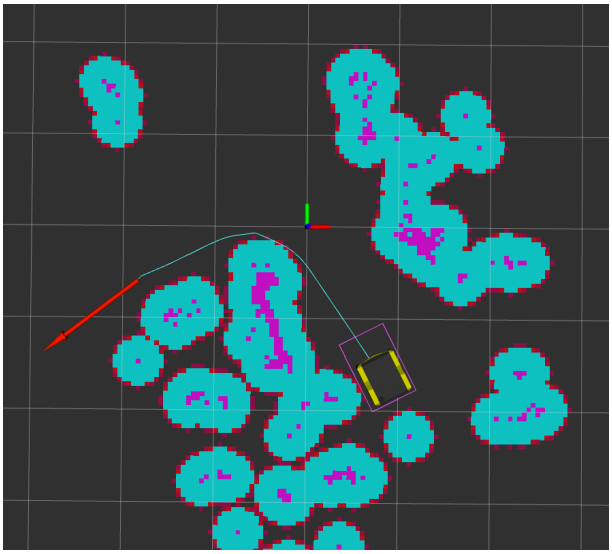


Figure 3.8: Costmap generated by the move base package.

generated by SLAM packages such as “hector mapping” to be reused. Reusing maps lowers deviation between subsequent scans as the map is not constantly recreated, which could introduce new errors. This also allowed the reusing of goal lists from the “follow waypoints” package. Implementing “map server” requires an operator to specify the initial pose of the robot on the loaded map. If a SLAM algorithm was running, a rough initial pose estimate would rapidly converged to a more accurate estimate.

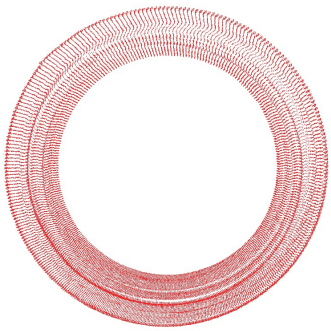
4. RESULTS AND DISCUSSION

This chapter focuses on evaluating the scanning performance of the robot in different environments. Many environments were tested, but only those displaying exceptional behavior are covered in the following chapter. Point cloud results will be depicted and the reasons for accuracy/error will be discussed.

4.1 CULVERT SCANS

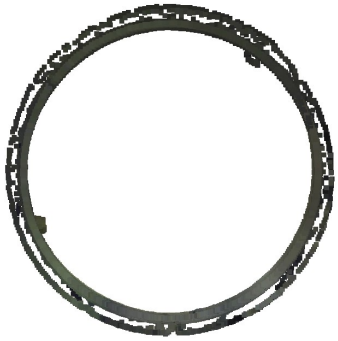
In this section we evaluate the scanning performance of the robot by mapping a culvert mock-up. While the culvert does not match the design environment of the robot, its symmetrical shape provides a clear understanding of the point clouds intended shape. To further test the scanning performance of the robot, the platform was forced to travel over “debris” to observe the effects on the captured point cloud.

(a) photo front.



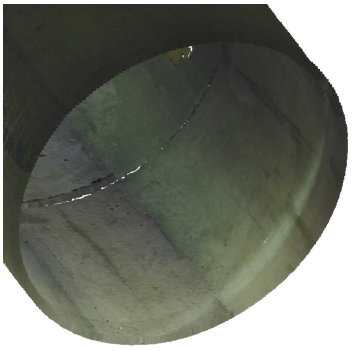
(b) xyz-cloud front.

(c) xyzrgb-cloud front.



(d) xyzrgb-cloud projection.

(e) xyzrgb-cloud segment.



(f) xyzrgb-cloud detail.

Figure 4.1: Culvert scans.

Figure 4.1 depicts the point cloud of the culvert while traveling along its length at a constant speed of 0.05 meters per second. This environment is unsuitable for autonomous navigation as the ground plane was not flat and suddenly drops off at either end of the culvert. Over all, the point clouds came out nicely due to the prismatic nature of the environment. This allowed the robot to capture detailed point clouds in a single pass without casting lidar "shadows" on the environment. These images showcase uniform, dense, and accurate point clouds without geometric abnormalities. The only abnormalities being mismatch of concrete color and some camera artefacts.

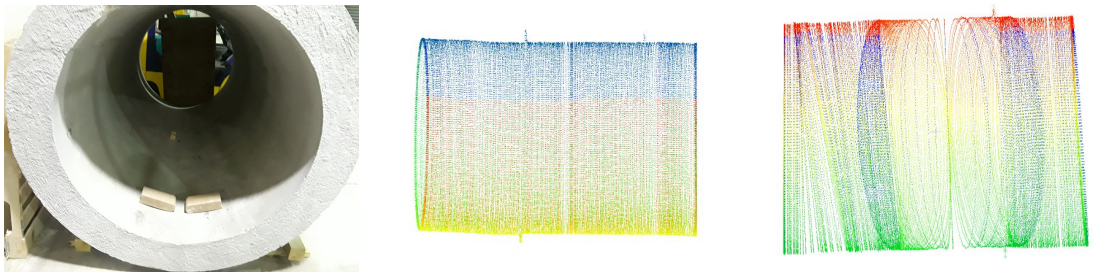
Comparing Figure 4.1a and 4.1f shows that the color of the concrete is not properly represented in the point cloud. This is due to the lack of proper lighting within the culvert, causing the cameras to register darker coloring than a light grey. Camera artefacts can also be seen in the point cloud as two dark lines running the length of the tunnel in Figures 4.1c, 4.1e, and 4.1f.These visual artefacts originate at the intersection of the four camera FoV's and are thought to be caused by minor discrepancies in color and exposure between the cameras.

While color matching was inaccurate and camera artefacts were present, the robot was successful in capturing concrete scoring present on the interior of the culvert. This scoring can be seen on the floor and sides of the culvert in Figures 4.1a, 4.1e, and 4.1f. The robot was also able to capture the seams where the segments of the culvert meet. This can be observed as a protruding ring running along the perimeter of the culvert (seen in Figure 4.1e and 4.1d). The protrusion is due to the depth change of the seam. The robot also captured the 1.5 inch wooded cubes (Figure 4.1d) placed on the interior of the culvert. These were used to observe the geometric and color mapping accuracy of the robot.

To test the effect of debris on the point cloud quality, two bricks were placed inside the culvert for the robot to traverse (See Figure 4.2a). Figure 4.2b depicts a point cloud captured without debris while Figure 4.2c depicts the effect that traversing the debris had on the point cloud. These scans retain much of the accuracy of the non-debris scan

but demonstrate a uneven point cloud density. This is due to the scan plane tilting as the robot crossed the debris, causing the scan plane to rotate past regions at varying speed. Of note is that the overall shape of the culvert is only slightly distorted. This is a good indicator that the internal transformations are responding properly and keeping the relations between the robot and its environment.

While the robot can produce decent scans while traveling over debris it must do so slowly or risk error. If the platform experiences a sudden acceleration (a sudden drop, twist, or jump when traversing debris) this distorts the point cloud. This produced a discontinuity in the point cloud at the location of sudden acceleration, as several scan profiles of the culvert were offset from their real locations. This is thought to be caused by the slight delay in data capture and the use of a Kalman filter for localization, which estimates future positions based on the current velocities of the robot.



(a) Debris photo. (b) Without-debris xyz-cloud. (c) With-debris xyz-cloud.

Figure 4.2: Culvert debris traversal test.

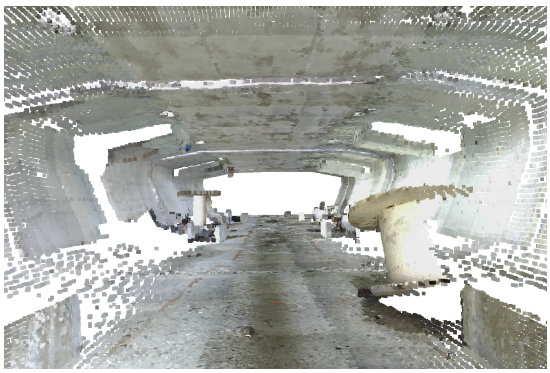
4.2 OVERPASS SEGMENT

To better understand how well the platform captured concrete defects, the interior of an overpass segment was scanned. Due to the low height of concrete reinforcements within the overpass, along with sudden drop-offs at either end of the tunnel, this environment was unsuitable for autonomous navigation. While this environment does not meet our autonomy criteria it was selected because it provides a good analog to the environments the platform is meant to monitor.

Due to the semi-prismatic nature of the environment it was possible to obtain a good understanding of the surroundings using a single scan. Unlike the tunnel however, this environment is not fully prismatic and would require multiple passes by the robot to fully capture. This would ensure that any lidar shadows are filled in, and orthogonal surfaces are captured, and result in an overall increase in the point cloud density (see Figures 4.3b and 4.3c).

Both the geometry capture and color mapping performed well in this environment. Figure 4.4 depicts a colored point cloud of the roof of the overpass. On this roof, imperfections in the concrete (such as color mottling) are present and captured by the scan. These scans also demonstrate more accurate color-matching between the photo of the overpass and the point cloud. This is due to increased ambient lighting when compared to the culvert scans. Similarly to the culvert, camera artefacts are also present in this scan where the FoV's of the cameras intersect. These artefacts show up as lines on the floor running parallel to and along the entire length of the tunnel (see Figure 4.3b).

(a) photo front.



(b) xyz-cloud side.

(c) xyzrgb-cloud side.

Figure 4.3: Overpass scans.

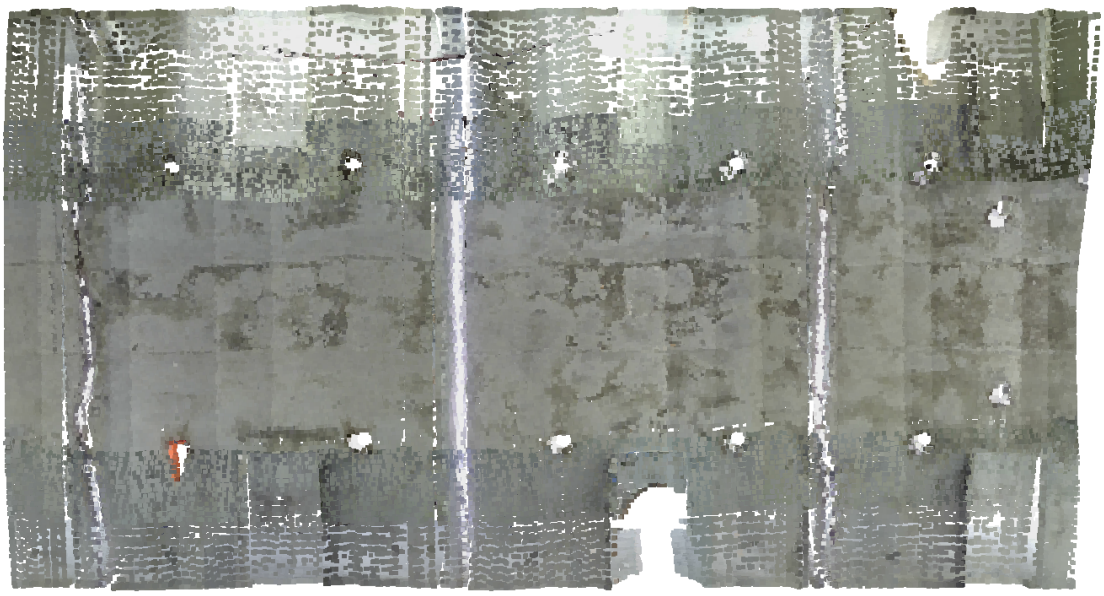


Figure 4.4: Overpass xyzrgb-cloud top.

4.3 HALLWAY SCANS

This section evaluates the scanning behavior of the robot in hallways, one of its primary design environments. The robot was tested in several hallways to understand how different geometries affected scanning behavior. The testing demonstrated that autonomous scanning was possible if at least one prominent geometric feature was detectable by the lidar SLAM.

Figure 4.5 depicts both the front and side views of a successful hallway scan. No problems were encountered during the scan due to the ample geometric features present in the environment. While more complex than culverts, hallways are still generally prismatic in geometry. This allows the robot to build a nearly complete point cloud of the environment using a single scanning pass.

While prismatic environments allow the robot to easily capture the surrounding, they can also cause the lidar SLAM to enter a degenerate state. If the hallway has too few geometric features along its principal axis it prevents the robots localization estimate from updating. As the robot travels down a geometrically degenerate hallway, the lidar “sees” the exact same features no matter how far it travels. This incoming

data is identical to what the robot would experience if staying still, and so publishes the corresponding odometry. This results in the collected scans "piling-up" in a single location, corrupting both the point cloud and the SLAM feature map.

(a) photo front.

(b) xyzrgb-cloud front.

(c) xyzrgb-cloud side.

Figure 4.5: Hallway scans.

Not only is SLAM degeneracy problematic, but its effect is exacerbated by the lidar implemented on the robot. The robot SLAM uses a forward facing SICK-Tim551 lidar with a 270 degree FoV. When entering a degenerate environment the 90 degree gap in data causes the robot loose sight of perfectly valid features located behind it. This significantly shortens the distance the robot can travel before entering a degenerate state. Navigation maneuvers (such as traveling in reverse or zig-zag pattern) can postpone the onset of degeneracy, but an improvement of the lidar would be more effective in increase the robustness of the robot.

For the platform to capture good color data, it requires both evenly lit and brightly lit environments. The tested indoor environments satisfied the brightness constraint but not the evenness constraint. As the robot traveled the hallways the high brightness contrast

of the light sources caused problems with the color mapping. Whenever the robot passed under a light, the upward facing camera would alter its exposure to compensate. This caused the point cloud to record a darker color for that segment of the point cloud. This color mismatch can be seen as brightness banding on the roof of the hallway in Figure 4.5b point cloud.

4.4 WALL SCANS

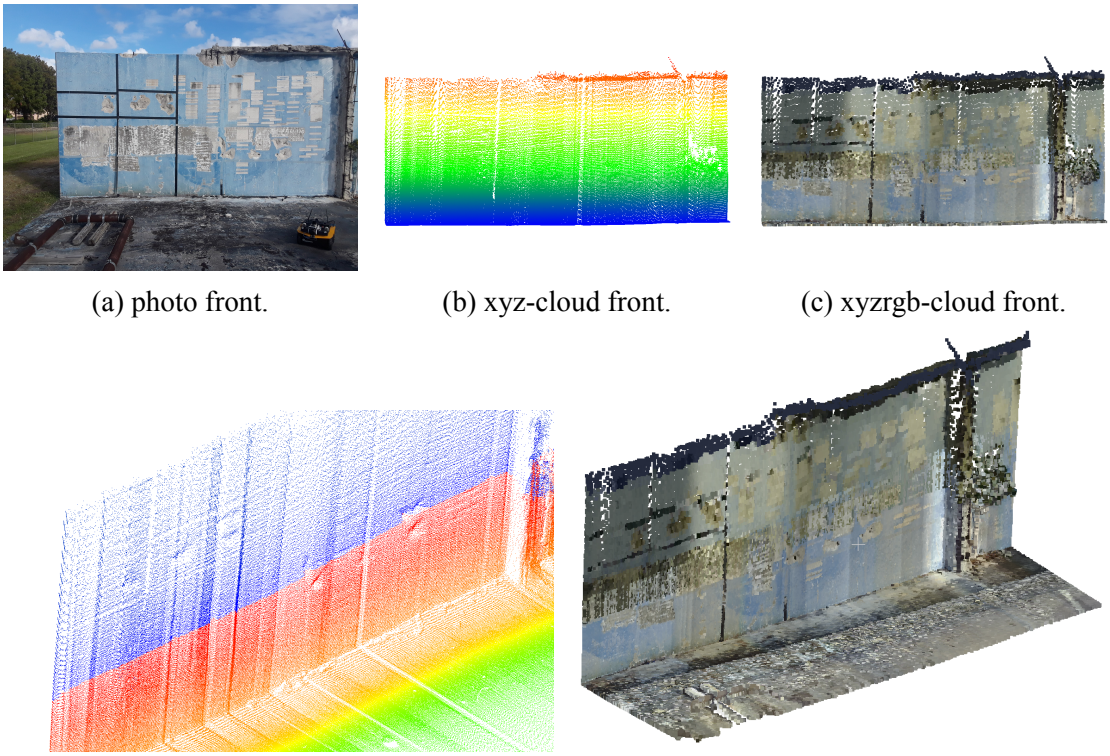
This section evaluates the performance of the robot while scanning a damaged wall. Figure 4.6a depicts an exterior concrete wall which has undergone mechanical scoring. Blue regions of the wall in Figures 4.6a, 4.6c, 4.6e indicate undamaged sections while gray regions indicate exposed concrete. While outdoor environments do not meet the design criteria of the robot this structure was selected as it featured both prominent damage and a prismatic geometry. While walls typically meet the autonomy criteria for the robot, the pipes observed in the lower left corner of Figure 4.6a are too low for the robot to detect and so prevent autonomous navigation during these tests.

After examining scans taken with the robot it was observed that the platform reliably captured the damage present on the wall. The features observed in the point cloud can be readily identified as damage as they were recorded in both the geometric and color data. Figure 4.6d shows damage as irregularities in the point cloud geometry while Figure 4.6e shows damage as point cloud discoloration.

While outdoor scanning did not negatively impact the xyz point cloud, it did pose a challenge to accurately recording color data. As opposed to well lit indoor environments (which the robot was designed for), outdoor lighting is directional and changes with time. This leads to shadows being recorded in the point cloud which have significant color differences to their surroundings. While operators can readily identify shadows it is likely that automated forms of damage detection would confuse the sudden color

discrepancy with damage. This is further exacerbated if scans were to be captured at different times throughout the day, as shadow positioning would shift significantly.

In addition to daily lighting changes, outdoor brightness can vary significantly on short timescales, such as when a cloud passes over the sun. The results of such an occurrence can be seen in the upper left corner of Figure 4.6e, which displays a darker coloration than the rest of the wall. Below the discoloration, the camera artefacts due to FoV intersection appear once again. Once the cloud covers shifts however, the overall ambient lighting was more consistent than indoors. This allowed for surprisingly good color matching for the remainder of the scan.



(a) photo front. (b) xyz-cloud front. (c) xyzrgb-cloud front.

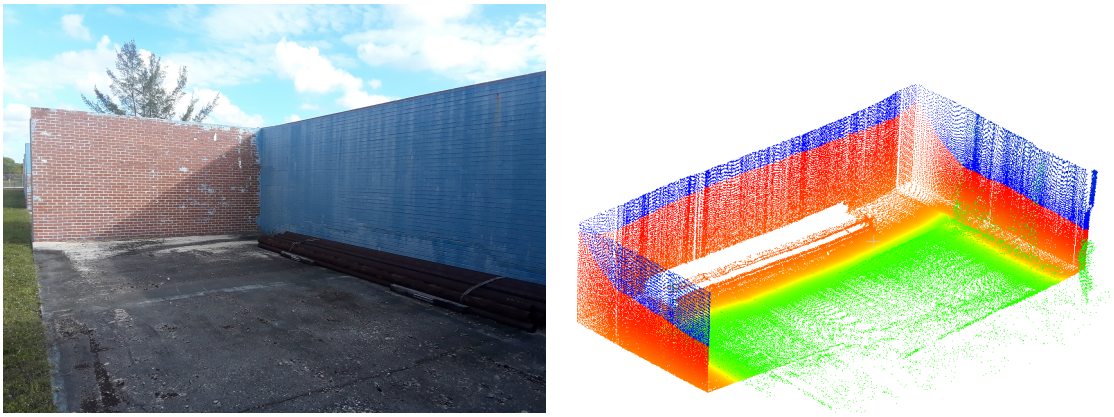
(d) xyz-cloud isometric. (e) xyzrgb-cloud isometric.

Figure 4.6: Damaged wall scans.

4.5 OPEN ROOM SCANS

In this section we evaluate the performance of the robot operating in a three walled room. This environment was chosen to determine whether the robot could capture small

imperfections present on the brick wall. The room also meets autonomy criteria and is overall an environment that meets the use cases of the robot. Figure 4.7 depicts a photo the room and a xyz point cloud. The point cloud shadow observed in Figure 4.7b is cast by several pipes stacked along the long wall of the room.



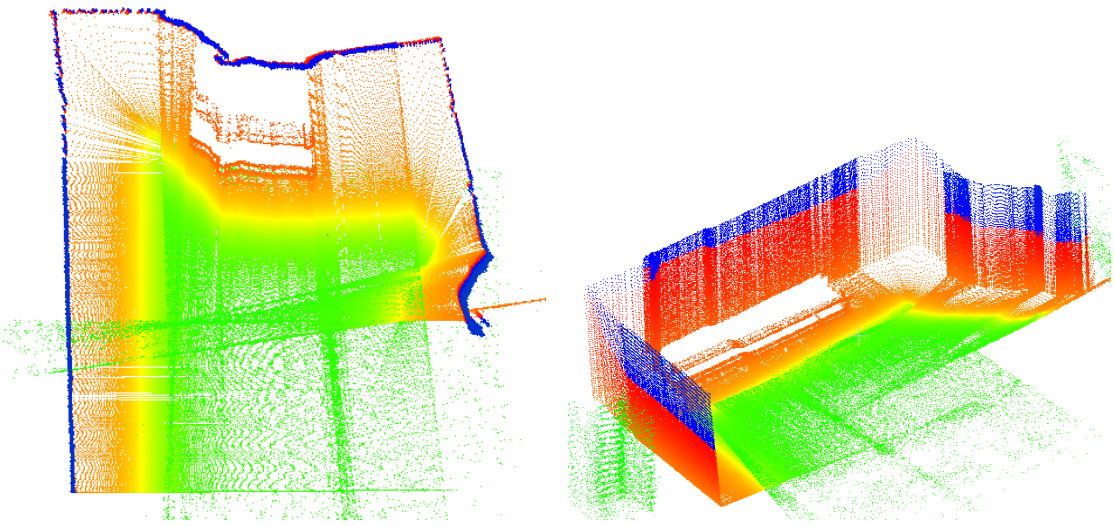
(a) photo isometric. (b) xyz-cloud isometric.

Figure 4.7: Open room scans.

While everything points to the room being ideal for scanning this is not actually the case. This environment is actually visually degenerate to the RS-T265 tracking camera used for visual SLAM. This degeneracy caused severe errors in both the map-making and scanning of the environment. Figure 4.8 depicts point clouds captured while the robot experienced degeneracy.

By examining these corrupted point clouds it is possible to gain some understanding about the failure mode of the robot. The "piled-up" nature of the point cloud indicates that the sensor believes that the robot is traveling at a slower speed than in actuality. This is due to the sensor observing no new features and therefore assuming it is not moving. While the exact mechanism is unknown, it is hypothesized that the repetitive patterns of the brick walls confuse the visual SLAM. This resulted in either the robot localization estimate halting completely or progressing at a fraction of the robots real velocity.

Once the degeneracy was identified the robots settings were manually configured to rely solely on lidar SLAM for localization. This enabled the capturing of the point cloud depicted in Figure 4.7b. Using lidar SLAM guaranteed that point clouds of the



(a) degenerate xyz-cloud top. (b) degenerate xyz-cloud isometric.

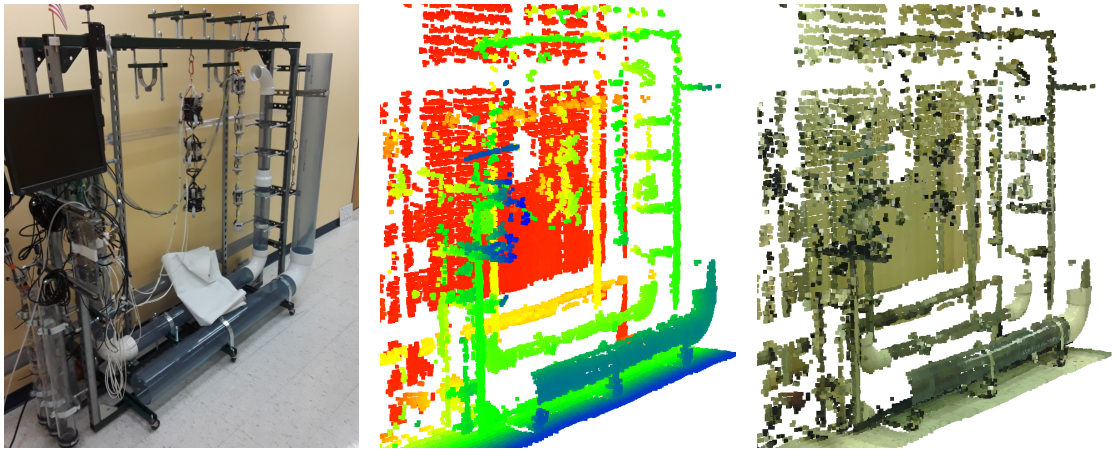
Figure 4.8: Clouds captured during degeneracy.

environment could be captured consistently, however in rare cases the visual SLAM was also able to generate an un-distorted point cloud. The primary takeaway form these tests is that degenerate environments cannot always recognized by operators. Due to the lack of a degeneracy filter, the current robot failed to localize properly in the environment.

4.6 OTHER SCANS

This section depicts attempts at capturing environments with features more complex than prismatic geometry. While only moderately successful these scans provide useful insight onto the robot capabilities.

Figure 4.9 depicts a complex metal frame with a series of plastic pipes running along its perimeter. This structure was chosen as it provides an analog to more industrial settings where piping and framing are of interest. Similarly to the scans featuring prismatic geometry, this point cloud was captured with a single pass of the robot. Observing the scan in Figure 4.9b it is obvious that the point cloud is missing information behind the metal frame. This is effectively the "shadow" of the frame being cast by the lidar onto the back wall and results in a substantial amount of information missing from this point



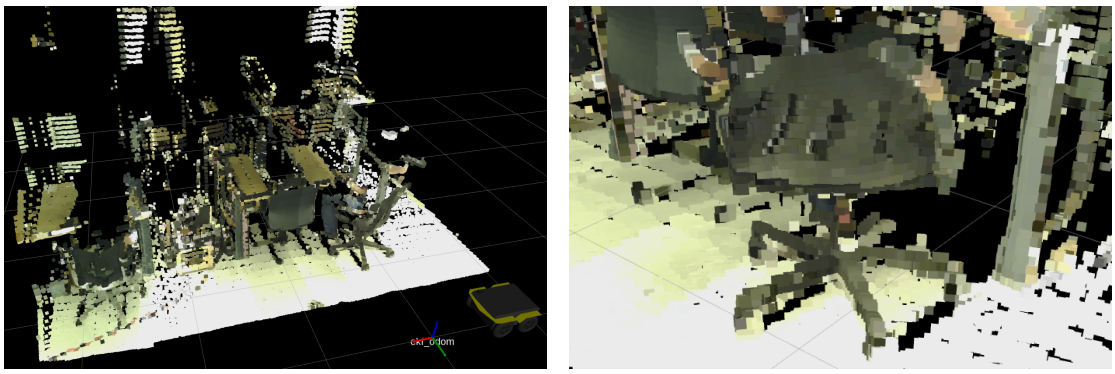
(a) photo isometric. (b) xyz-cloud isometric. (c) xyzrgb-cloud isometric.

Figure 4.9: Scan comparison of a complex metal frame.

cloud. To correct this, multiple passes of the region would have to be conducted in order to build a more complete 3D point cloud.

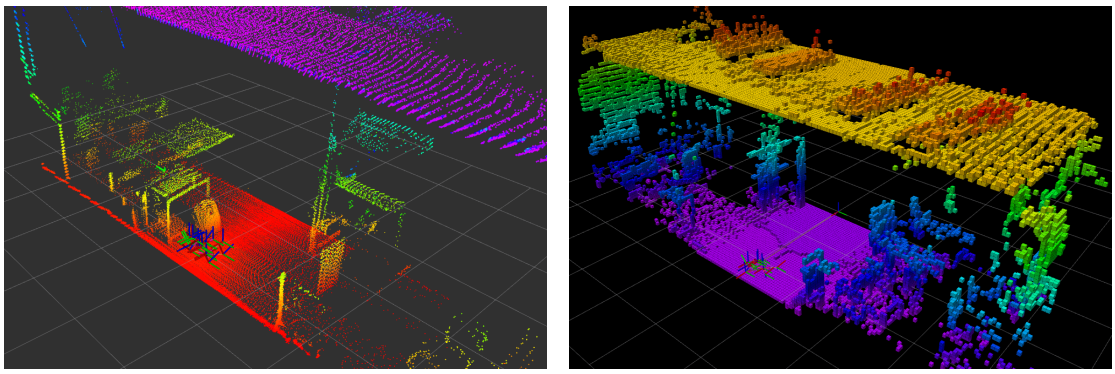
Figure 4.10 depicts a desk with rolling chairs captured using a single pass of the robot. This scan does a good job at showcasing the accuracy of the color mapping, as on Figure 4.10b color was accurately mapped to the legs of the chair. While this scan may appear complete, in actuality there is a lot of information missing from the point cloud. This is because the robot can only capture the undersides of desks and chairs due to being so low to the ground. To capture complex environments such as these, multiple passes as well as a second lidar mounted higher on the robot should be used.

Figure 4.11 shows an attempt at continuous voxelization of a point cloud using the "octomap" ROS package. The program was run on the Jackal primary computer but was too resource intensive at high resolutions. To maintain the platform performance the voxel resolution had to be set at 5 cm or above. This robbed the cloud of all fine detail and would eventually and would still slow down the platform if a large enough region was scanned.



(a) xyzrgb-cloud isometric. (b) xyzrgb-cloud detail.

Figure 4.10: Color mapping fidelity on desk chairs.



(a) xyz-cloud isometric. (b) xyz-voxel isometric.

Figure 4.11: Voxelization using the Octomap ROS package.

5. CONCLUSION AND RECOMMENDATIONS

This section will list the conclusions reached for each of the principle components of the project. Future work will be limited to two recommendations per section as robots can continuously be improved.

5.1 PHYSICAL DESIGN

The physical design of the robot was not a primary concern throughout most of the project. The starting Jackal platform had no problems traversing the design environment (indoors) and could carry a payload of up to 20 kg. Adding unprotected sensors to the platform negated its inbuilt weatherproofing but did not otherwise impact the performance of the robot.

While physical design was not the greatest concern, several parts were still designed and manufactured for the robot. Using a 3D printer, the sensor mount, camera cases, and a Jetson-Nano case were produced. The sensor mount was designed to allow for the re-positioning of cameras and frontal lidar. The cases were produced in order to protect the exposed PCB's of the cameras and Jetson-Nano from shorts and scratches.

Future upgrades to the robot's physical design should be aimed at protecting the incorporated sensors. During development, it is likely that the robots autonomy will be reconfigured and tested multiple times. Protecting the sensors from collisions could help prevent downtime and costly repairs. Increasing the configurability of the sensor mount would also facilitate the incorporation of additional lidars and cameras to the robot.

5.2 LOCALIZATION

The robot developed for this project required both continuous and accurate localization. This was driven primarily by the procedural mapping technique implemented to capture point clouds. Localization was obtained using a combination of lidar SLAM,

visual SLAM, IMU, and encoder measurements. Using an Extended Kalman Filter the disparate sensor measurements were combined into a single localization estimate. This yielded consistent and accurate localization under most indoor conditions.

While the robot is able to localize accurately it suffers from a lack of robustness. Rapid rotations of the platform confuse the SLAM algorithm and corrupt the generated feature map. The robot also experiences problems with sensor degeneracy due to using a 2D lidar for planar navigation. Degenerate environments for this robot include open spaces and featureless hallways. In open spaces the laser range is insufficient to detect features while in hallways the features repeat along the hallway axis. Under both these conditions the localization estimate becomes stuck while the robot continues to move.

Future work on localization should focus on detecting and filtering sensor degeneracy. This would allow the robot to properly localize even if the environment causes a sensor to fail due to degeneracy or damage. Subsequent improvements should attempt to solve the "kidnapped robot" problem by incorporating map recognition to the robot. This would allow the platform to automatically localize its position within a known map, thereby removing the need for an operator to manually input starting coordinates or always assuming them to be 0.

5.3 POINT CLOUD COLLECTION

Capturing colored-point-clouds with this robot yields good results. Under ideal conditions the robot can obtain point clouds that accurately represent the environment. While the geometric accuracy of the point cloud is relatively stable, the performance of the color mapping suffers if lighting is inconsistent.

The robot uses a 360 degree planar lidar and a procedural-mapping technique to capture point clouds. This combination allows the robot to collect point clouds of its surroundings while moving through the environment. The robots accurate localization even allows the capturing of point clouds while traversing uneven terrain. Using four

cameras the robot is able to project colors onto the point cloud. This is possible through the use of the "jag color" package, developed at FIU. This package allows for the mapping of ROS Image messages onto point clouds using a pinhole camera model. The captured point cloud can then be recorded using the "pc2 record" package (also developed at FIU).

While the point cloud collection system works, detailed data collecting requires the robot to move slowly though the environment. The robot also has trouble overlaying point clouds generated by multiple passes through the environment. This is due to subtle discrepancies in localization and the lack of point cloud registration techniques.

Color mapping is imperfect as it suffers from problems caused by object occlusion and variable lighting. Because the lidar scan plane is offset from the camera origin, complex environments can occlude 3D points from the view of the camera. This results in the colors from the occluding object being mapped onto the occluded points. Lighting exposure also affects the quality of the color mapping as the four cameras independently modulate their brightness. This causes regions of the point cloud to be darker or lighter depending on whether a particular camera is looking at lighter or dimmer regions of the environment. These discrepancies are especially apparent where the cameras' FoV's intersect as there is often a sudden discrepancy in color.

For future work an effort should be made to capture data from thermal cameras onto the point-cloud. While it is tempting to correct the color mapping discrepancies, adding this sensor would make new damage indicators (such as leaks and short circuits) detectable by the system. This upgrade would yield a significant increase in the information content of the collected point clouds, allowing for operators to make more informed decisions. Another excellent upgrade would be increasing the overall density of captured point clouds. This could be done through the addition of a second planar lidar or multi-beam lidar. This upgrade would also allow the platform to travel faster while still collecting the same density of data.

5.4 AUTONOMY

The project succeeded in implementing level 3 autonomous navigation on the robot, but only for idealized environments. Complex paths can be specified by setting multiple goal locations on a known map. When activated the robot plans the shortest path to each goal and navigates while avoiding known and new obstacles. Prior to starting navigation, the point cloud collection package can be run. This allows the robot to capture a point cloud of the environment while autonomously traveling towards the specified destination.

While the system is able to navigate autonomously, it can only do so safely in an idealized environment. This environment can be described as regions permitting planar navigation and lacking objects ranging from 2" to 12" in height. Planar navigation is terrain that can be adequately approximated by using a single plane. This criterion is specified as the robot cannot distinguish between inclines and obstacles. The specified obstacle height describes objects which cannot be traversed or detected by the robot. If obstacles in the height range are present, the robot may crash and cause damage to itself or surroundings. These factors are very limiting and greatly reduce the applicability of the current platform.

For future work, a major improvement would be the addition of extra obstacle-detection sensors to the robot. This would expand the environments suitable for autonomous navigation as previously undetectable obstacles could be identified. Another upgrade would be the implementation of scanning goals to the robot. These goals would behave identically to the current goal markers but would be able to start/stop point cloud recording. This would allow the robot to navigate to regions of interest before collecting data.

5.5 EASE OF USE

While the robot can accomplish autonomous mapping tasks, in its current state it is not a user-friendly device. Operating the platform requires substantial knowledge of ROS, Linux, and the developed system. Several of the ROS packages also lack usability, requiring the manual modification of files to operate. This is somewhat expected from a platform under development, but makes it difficult for anyone wanting to use the tool.

The first notable change would be allowing the entire system to be launched using a single command. Currently the ROS packages on the Jackal, Jetson-Nano, and laptop computers are launched separately. Changing the launch procedure so as to be able to run off a single command would save a significant amount of setup time and make it easy for operators to use.

The second useful upgrade for usability involves general improvements to the "pc2 record" ROS package. Currently this package requires manual renaming of files to prevent overwrites when saving. Saving multiple file-types could also prove useful, especially ones containing headers to identify point cloud data. Lastly, ROS should be able to interact with this package to allow the automated starting and stopping of point cloud recording.

BIBLIOGRAPHY

[1] U.S. Energy Information Administration. *A Look at the U.S. Commercial Building Stock: Results from EIA's 2012 Commercial Buildings Energy Consumption Survey (CBECS)*. URL: https://www.eia.gov/consumption/commercial/reports/2012/buildstock/. (accessed: 05.06.2020).

[2] Ali [Hosseininaveh Ahmadabadian], Ali Karami, and Rouhallah Yazdan. "An automatic 3D reconstruction system for texture-less objects". In: *Robotics and Autonomous Systems* 117 (2019), pp. 29–39. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2019.04.001. URL: http://www.sciencedirect.com/science/article/pii/S0921889017307431.

[3] AnyBotics. *ANYmal C*. URL: https://www.anybotics.com/anymal-legged-robot/. (accessed: 05.30.2020).

[4] Kade Barranco. *S.L.A.M — Tracking for MR*. URL: https://medium.com/desn325-emergentdesign/s-l-a-m-tracking-for-mr-2a0face9d97a. (accessed: 04.26.2020).

[5] Norlida Buniyamin et al. "A simple local path planning algorithm for autonomous mobile robots". In: *International journal of systems applications, Engineering & development* 5.2 (2011), pp. 151–159.

[6] František DuchoĖ et al. "Path planning with modified a star algorithm for a mobile robot". In: *Procedia Engineering* 96 (2014), pp. 59–69.

[7] Wikimedia Foundation. *Concrete degradation*. URL: https://en.wikipedia.org/wiki/Concrete_degradation. (accessed: 05.07.2020).

[8] National Geographic. *Triangulation*. URL: https://www.nationalgeographic.org/photo/triangulation-sized/. (accessed: 05.07.2020).

[9] Karl Granström et al. "Learning to close loops from range data". In: *The International Journal of Robotics Research* 30.14 (2011), pp. 1728–1754. DOI: 10.1177/0278364911405086. eprint: https://doi.org/10.1177/0278364911405086. URL: https://doi.org/10.1177/0278364911405086.

[10] Marco Hutter et al. "ANYmal-toward legged robots for harsh environments". In: *Advanced Robotics* 31.17 (2017), pp. 918–931.

[11] Intel. *Intel RealSense Tracking Camera T265*. URL: https://www.intelrealsense.com/tracking-camera-t265/. (accessed: 05.04.2020).

[12] N Meierhold et al. "Automatic feature matching between digital images and 2D representations of a 3D laser scanner point cloud". In: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 38 (2010), pp. 446–451.

[13] Thomas Moore and Daniel Stouch. "A generalized extended kalman filter implementation for the robot operating system". In: *Intelligent autonomous systems 13*. Springer, 2016, pp. 335–348.

[14] n/a. *How to Calibrate a Monocular Camera*. URL: http://library.isr.ist.utl.pt/docs/roswiki/camera_calibration(2f)Tutorials(2f)MonocularCalibration.html. (accessed: 05.08.2020).

[15] Nvidia. *Jetson Nano Developer Kit*. URL: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. (accessed: 05.04.2020).

[16] pixelmitherer. *Tree trunk photogrammetry*. URL: https://sketchfab.com/3d-models/tree-trunk-photogrammetry-86de736e3fc044aeac2e4be59fd7a1e2. (accessed: 05.08.2020).

[17] Clearpath Robotics. *Jackal Unmanned Ground Vehicle*. URL: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/. (accessed: 05.02.2020).

[18] Scaled Robotics. *Scaled Robotics Digitising Construction*. URL: https://www.scaledrobotics.com/. (accessed: 05.31.2020).

[19] Sick. *2D LiDAR sensors TiM5xx / Outdoor*. URL: https://www.sick.com/us/en/detection-and-ranging-solutions/2d-lidar-sensors/tim5xx/tim551-2050001/p/p343045. (accessed: 05.04.2020).

[20] Roland Siegwart and Illah R Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Cambridge, Massachusetts: MIT Press, 2004.

[21] Anthony Stentz. "Map-based strategies for robot navigation in unknown environments". In: *AAAI spring symposium on planning with incomplete information for robot problems*. 1996, pp. 110–116.

[22] TechCrunch. *Scaled Robotics - Winner of Startup Battlefield Berlin 2019*. URL: https://www.youtube.com/watch?v=SStjervx5hs. (accessed: 05.30.2020).

[23] Total. *Argos Project*. URL: http://argos.ep.total.com/en/. (accessed: 05.31.2020).

[24] Total. *ARGOS, building robots for safety | Total*. URL: https://www.youtube.com/watch?v=HOPo6hAPcIo. (accessed: 05.30.2020).

[25] Trimble. *Trimble TX8*. URL: https://geospatial.trimble.com/products-and-solutions/trimble-tx8. (accessed: 05.30.2020).

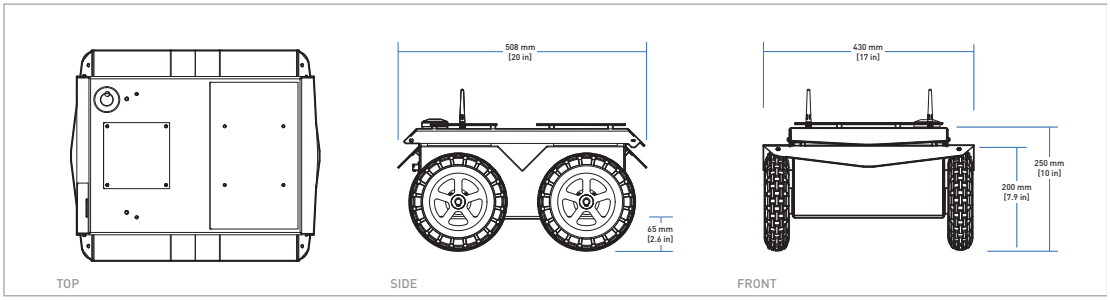
[26] Anh-Vu Vo et al. "Octree-based region growing for point cloud segmentation". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 104 (2015), pp. 88–100.

[27] Weikun Zhen and Sebastian Scherer. "Achieving Robust Localization in Geometrically Degenerated Tunnels". In: 2018.

JACKAL™

UNMANNED GROUND VEHICLE

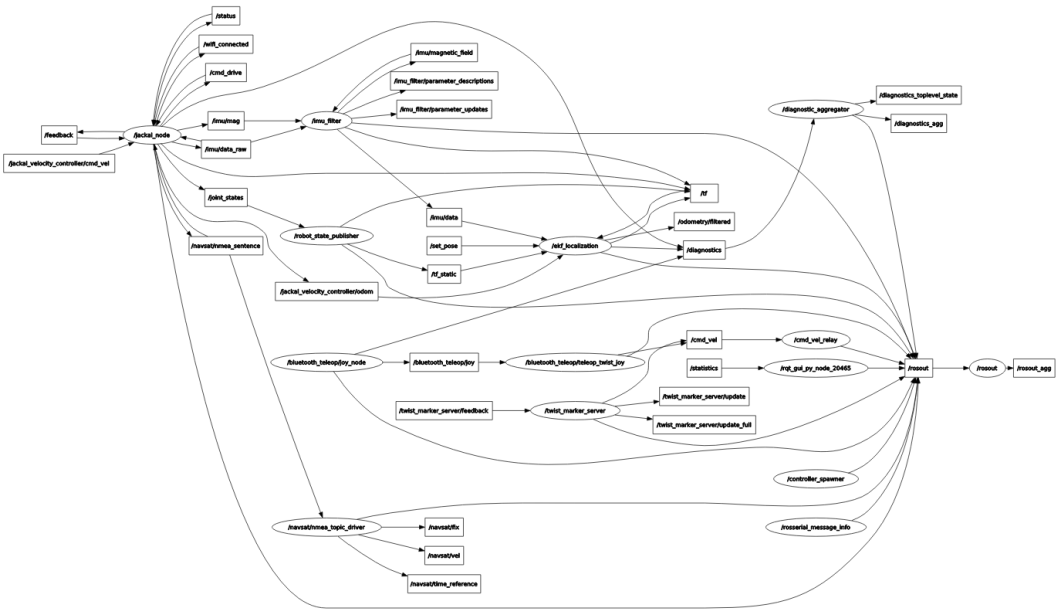
CLEARPATH ROBOTICS™



TECHNICAL SPECIFICATIONS

SIZE AND WEIGHT				
EXTERNAL DIMENSIONS (L x W x H)	508 x 430 x 250 mm (20 x 17 x 10 in)			
INTERNAL STORAGE DIMENSIONS	250 x 100 x 85 mm (10 x 4 x 3 in)			
WEIGHT	17 kg (37 lbs)			
GROUND CLEARANCE	65 mm (2.6 in)			
SPEED AND PERFORMANCE				
MAX. PAYLOAD	20 kg (44 lbs)			
ALL-TERRAIN PAYLOAD	10 kg (22 lbs)			
MAX. SPEED	2.0 m/s (6.6 ft/s)			
DRIVE POWER	500 W			
BATTERY AND POWER SYSTEM				
BATTERY CHEMISTRY	Lithium Ion			
CAPACITY	270 Watt hours			
CHARGE TIME	4 hours			
RUN TIME	Heavy usage: 2 hours		Basic Usage: 8 hours	
USER POWER	5 V at 5 A, 12 V at 10 A, 24 V at 20 A			
INTERFACING AND COMMUNICATION				
CONTROL MODES	Kinematic Commands — velocity, angular velocity Open Loop Motor Driver Commands — voltage Wheel Velocity Commands			
FEEDBACK	Battery and motor current Wheel velocity and travel		Integrated GPS receiver Integrated gyroscope and accelerometer	
COMMUNICATION	Ethernet, USB 3.0, RS232. (IEEE 1394 available)			
DRIVERS AND APIs	Packaged with ROS Kinetic			
INTEGRATED ACCESSORIES (included)	Wireless Game controller, GPS, IMU, On-Board Computer, WIFI Adapter, Accessory Mounting Plates			
COMPUTER	**Standard**		**Performance**	
	i3-4330TE Dual core, 2.4GHz 4 GB RAM	WIFI Adapter 120 GB Hard Drive	Intel Core i5 4570T Dual core, 2.9GHz 4 GB RAM	WIFI Adapter 128GB Hard Drive
ENVIRONMENTAL				
OPERATING AMBIENT TEMPERATURE	-20 to 45 °C (-4 to 113 °F)			
IP RATING	62 (65 available on select configurations)			

B. STARTING ROS NODES



C. ENDING ROS NODES