NTNU

Norwegian University of
Science and Technology

Ouster-1 3D LiDAR SLAM

Litterature and method review

Vebjørn Fossheim Eklo

December 21, 2018

Abstract

This report is a literature review of the problem known as the simultaneous localization and mapping (SLAM) problem. This report aims to widen the knowledge of the process of running a SLAM algorithm on an autonomous system. The rapport also visits different methods that are applicable for LiDAR SLAM, which is SLAM where the main sensor is a laser range scanner. The result of the literature review is a broader understanding of the subject and an intuition for which method best fits in a system that is planned as a master thesis, here described in a future work chapter.

Preface

This report in its finished product is the coursework for TTK4551 - Engineering Cybernetics, Specialization Project, which is credited 7.5 *cp*, counting towards the master's degree program in Cybernetics and Robotics at the Norwegian University of Science and Technology. The choice of project is supervised by Thor Inge Fossen, Professor of Guidance, Navigation and Control at the NTNU Department of Engineering Cybernetics.

Contents

List of Figures

1 Introduction

This introductory chapter will briefly provide context for the material presented in this report and give motivation and a description of the problem that is to be solved in the master thesis after this literature review project. The contributions of the thesis are defined and elaborated. Finally, the outline of the report is presented in short.

1.1 Motivation

In many applications of robotics, such as industrial automation, and autonomous mobility, there is a need to implement a model that can localize the robot, and simultaneously map its surroundings. This is by many considered as the "holy grail" of mobile robotics. The motivation behind this project is to review the work and development in this subject and use this knowledge to make a plan for future work in a master thesis. The subject is interesting because it is evolving, and no perfect solution to the problem is found. The tools given to accomplish the task is a new and cheaper product than the state of the art sensor, and may, therefore, inflict how future research is made more available. Aside from this, the main motivation behind this project is to learn the theory of SLAM, explore different methods of solving this problem and make a good guess to which of these methods best suit the plan for future work.

1.2 Background and Contributions

This report is mainly a literature review and is therefore mostly a tool for the writer to get a good overview of the theory and methods behind the problem. The report is also written in a way so that the reader should be able to understand what the SLAM problem implies, and how this process can be handled. The main contribution of this report is, therefore, an overview of possible methods, and an in-depth description of key parts of the system.

Most of the parts presented in the report has been touched upon in earlier reports

and studies, and these are well cited. The background material for the general solving of the SLAM problem is therefore broad, but in terms of background material for the tools presented in this report, there hasn't been much development, as the provider of the sensor is a start-up company. The sensor technology is by no means new, and the use of background material from similar products should, therefore, be of helpful use.

1.3 Outline

The report is organized as follows. After this intro chapter, Chapter 2 presents the theory and origins behind the SLAM problem. It gives an in-depth description of how the process works and presents the theory on how to solve the problem. The chapter also touches upon the sensor that is to be included in the future work of this process and describes the most important aspects of its features. The theory behind the communication system is also presented in this chapter. Chapter 3 explains the main algorithms used to solve the SLAM problem, while Chapter 4 presents some methods where these algorithms are included, and is more a chapter presenting possible implementations for future work with the process. Lastly, Chapter 5 presents an overview of the key subjects for future work in the master thesis.

2 Theory

2.1 The SLAM problem

The Springer Handbook of Robotics gives a great statement of what the SLAM problem represent [19]. A mobile robot roams the unknown environment, starting at an initial location. Its motion is uncertain, making it gradually more difficult to determine its current pose in global coordinates. As it roams, the robot senses its environment with a noisy sensor. The SLAM problem is the problem of building a map of the environment while simultaneously determine the robot's position relative to this map given noisy data.

Simultaneously Localization and Mapping or SLAM has its origin in studies done by Smith, Self and Cheeseman back in 1986 on estimation for spatial relationships and covariance between coordinate frames [17]. They introduced a method on how to localize an object relative to another by manipulating the uncertainty associated with spatial information, in the form of sensed relationships, prior constraints, and relative motion. They made it possible to estimate the probability of certain events, based on the uncertainty of the robot and the surrounding objects relative location. Smith, Self and Cheeseman made an effort to develop the methods in the context of state-estimation and filtering theory [18] to provide a solid basis for numerous extensions. This led to the introduction of SLAM in 1991 by Leonard and Durrant-Whyte [12].

Leonard and Durrant-Whyte introduced an algorithm for localization, based on current state, position estimates and sensor observation. They were able to develop a mobile robot localization system that integrated a variety of beacon observations as input to a Kalman filter to maintain a robust vehicle location estimate. Although their algorithm had a number of limitations, most importantly the need for a priori environment description and infinite data limits, they set the standard for further work with more computationally tractable algorithms.

Durrant-Whyte, Fellow and Bailey have written one of the most cited papers on the general SLAM topic [2]. To illustrate their points, figure 1 shows the general idea of

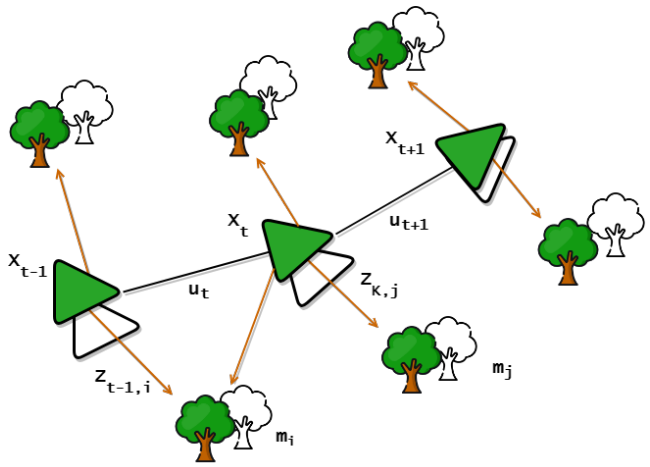


Figure 1: SLAM illustration. Objects in green are the real locations, the white objects are the estimated equivalents

how SLAM works. The true locations are never known or measured directly in SLAM. Instead, observations are made between true robot and landmark locations, and an estimated result is presented in both robot and landmark location. In the illustration, a robot is moving through an environment taking relative observations of a number of unknown landmarks. In the figure objects in green represent real locations of robot and landmarks, while white is the estimated equivalents. To further understand this we need to define the notations used in figure 1 at time instant t. The notations used in SLAM varies in different papers, but the ones presented below is fairly common.

x_t: The state vector. Describes the robot's location and orientation. Put in consecutive order $X_{0:T}$, and we get the path of the robot.

u_t: The control vector. Describes the controls the robots receives in the prior state to drive it to the current state. Put in consecutive order $u_{1:T}$, and we get the robot controls.

m_i: Landmark vector. The environment may be comprised of landmarks, objects,

surfaces, etc., This vector describes the location of these. Put together, the landmarks create a map m of the environment.

$z_{t,i}$: The observation vector. At time instant t, $z_{t,i}$ describes the observation of landmark m_i. Put in consecutive order, $z_{1:T}$ describes a set of observations, where e.g z_T could be a laser scan image.

The data that is obtained is not necessarily accurate. Uncertainty in the robot's motion and observations is common. SLAM is best described by a probabilistic approach. The robot or landmark is not seen in an exact position but has a probability distribution to its location. The use of the mean and variance from this distribution is a common way to estimate the positions in SLAM. Even though probabilistic distribution models vary depending on the system, there are mainly two forms of the SLAM problem. The full and online SLAM problem.

The full SLAM problem aims to estimate the entire path together with the map as shown in equation 1. Written this way, the full SLAM problem is the problem of calculating the probability distribution of the path x_T and the map m given observed sensor data and robot controls. Figure 2 illustrates a graphical model of the full SLAM problem. Algorithms for the full SLAM problem often process all data at the same time. The online SLAM problem seeks to recover only the recent pose and map, marginalizing out the previous poses. Usually done incremental one at a time, as shown in equation 2. A graphical model of the online SLAM problem would look familiar to the full problem figure 2, but with an unknown coloured field only for the current state and not the entire path.

$$\boldsymbol{p}(\boldsymbol{x}_{0:T}, \boldsymbol{m} \mid \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) \tag{1}$$

$$\boldsymbol{p}(\boldsymbol{x}_t, \boldsymbol{m} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) = \int_{\boldsymbol{x}_0} \dots \int_{\boldsymbol{x}_{t-1}} \boldsymbol{p}(\boldsymbol{x}_{0:t}, \boldsymbol{m} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) d\boldsymbol{x}_{t-1} \dots d\boldsymbol{x}_0 \tag{2}$$

$$\boldsymbol{p}(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \tag{3}$$

$$p(z_{t+1} \mid x_{t+1}, m) \tag{4}$$

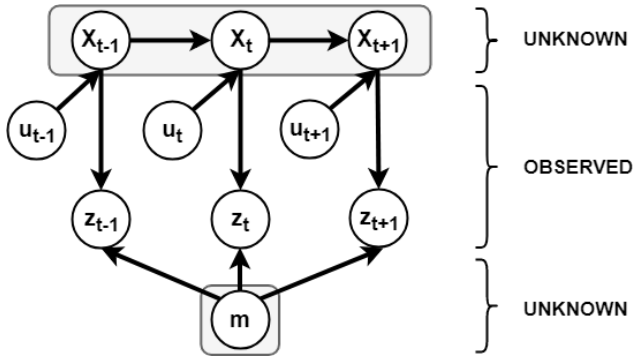


Figure 2: Graphical Model of the full SLAM-Problem

These two generalizations of the SLAM problem is usually structured into two models. The motion model and the observation model. The motion model describes the relative motion of the robot, while the observation model relates measurements with the robots pose. The models corresponds to the arcs in figure 3. Probabilistic motion models comprise the state transition probability shown in equation 3. It is an essential part of the prediction step of the Bayes filter which calculates the belief parameter that helps the robot infer its position and innovation. See chapter 2 in [20] for Bayes filters algorithms and mathematical derivation. There are several different techniques to motion models designed for different use cases. Thrun, Fox, Burgard and Dallaert introduced a robust motion model called Monte Carlo Localization in 2001 [21], able to estimate the pose in dynamic systems. Examples of motion models in the plane is given in [20] chapter 5.

As for motion models, observation models offer vastly different models depending on the measurement devices and the system platform. The specifics of the model depends on the sensor. Cameras are best modelled by projective geometry, sonars by describing the sound wave and its reflection on surfaces in the environment. The

system planned in this paper uses a laser range scanner, the observation model should therefore be applicable to this sensor. Chapter 6 in Thrun [20] presents different approaches with different distributions for the probability estimates. Solutions using the RANSAC-algorithm is used in [6] and ICP (iterative closest point) methods in [9].

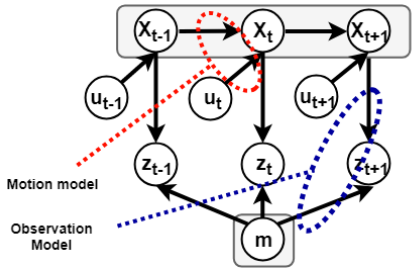


Figure 3: Grapical representation of the motion and observation model

SLAM is best explained as a concept rather than a simple algorithm, where all the blocks that are shown in figure 2 can be solved with different approaches. The basic idea is that after enough measurements, errors caused by vibrations and robot movements will be nullified. Each measurement is expected to be the same relative to the environment, meaning the position can be estimated over a large enough data set.

The data association problem is a problem worth mentioning when talking about SLAM. It is, in fact, one of the hardest problems to solve when you wish to track your position. The data association problem is the problem of deciding which target generated which observation. This would be easy in single-target tracking problems, but when the association is more ambiguous and tracks multiple observation in one scan or sample, the assignment of the observations become much harder to solve. One simple strategy to solve this is to pay attention to the measurement that is closest to the prediction. This again becomes a problem when areas suffer for clutter in the landscape. A more sophisticated method is to keep track of multiple states/observations hypothesis and consider the possibilities of these hypotheses. This is a common method for particle filter strategies, where each particle can be a hypothesis of the current observation.

Some suggested solutions is presented in [8][14][16].

Another common problem in SLAM is the loop-closing problem. This is the problem of realizing that the observed measurements have been observed at an earlier stage. If one with certainty can conclude that this, in fact, has accrued, a loop-closing algorithm can be used to overcome the drift accumulated in the robot trajectory over time. Possible solutions with descriptors to the problem is presented in [11][5].

2.2 LIDAR - OUSTER1

A LiDAR is a laser range sensor. The sensor shines a light at a surface and measures the time it takes for the light beam to return to its source. Since light moves at constant speed, a LiDAR can measure the distance with high accuracy. LiDARs fires rapid pulses of light every second and receives these as points. By repeating this process in quick succession, the LiDAR can build a complex map of the surface it is measuring. The number of points received per second varies greatly with the price of the sensor, but more expensive LiDARs can receive millions of points per second [1].

There are different kind of LiDARs for different use cases, including detection in all three different dimensions. A 1D LiDAR is usually a cheap sensor that is used for detecting the distance to a single point. A 2D LiDAR usually rotates to measure the distance to objects in the 2D plane. The last and more expensive 3D LiDAR also has a vertical field of view, giving it 3D measuring capabilities. In SLAM both 2D and 3D LiDARs is a common measurement tool. A 3D LiDAR will give you more information but is usually more costly in both price and processing power. Compared to vision-based SLAM, LiDAR SLAM has the advantage of working in areas were distinct features and light is absent.

The Ouster-1 LiDAR planned used in this project is a 16-beam 3D LiDAR with a 16.6 deg vertical field of view. It is able to capture 327 680 points per second. Every point received contains data information including range, intensity, reflectivity, ambient data, angles and time-stamp. Ouster-1 also has a built-in IMU for gyro and accelerometer readings.

See appendix A for the Ouster-1 datasheet. Since the Ouster-1 LiDARs output depth images, signal images, and ambient images in real time, all without a camera, they add a solution with one sensor that competitors need a LiDAR/camera fusion to accomplish. Together with the cheap price compared to other 3D LiDAR products, the OUSTER-1 sensor makes a good candidate for SLAM-related use.

2.3 ROS

Robot Operating System (ROS) is an open-source, meta-operating system. ROS was made to support the reuse of code in robotics research and development. The point is to have an environment where many modules can run concurrently, and communicate without being aware of each other. Meta-operating system means that ROS is built and based on Ubuntu Linux, and shares it's process management system, file system, user interface and programming utilities. In addition, it also provides tools and libraries for obtaining, building, writing and running code across multiple computers. In other words, instead of redefining and changing the programming vocabulary and grammar, ROS only adds features and libraries to the traditional C++ program. You can therefore simply use some function calls and classes instead of rewrite major parts of code.

In ROS, programs are called nodes. Nodes can communicate with each other by sending messages. These messages are sent to what is called a message topic. A topic of a message must have a defined message type. This is so that ROS can convert them from data structures to byte streams at sender's end, transport them to the recipient, and then convert them back to data structures. To receive and send certain topic of messages the nodes needs to include what is called ROS Subscriber and ROS Publisher functions. These are functions included in the ROS library and are necessary for the nodes to communicate. An example of a node used in ROS is the ouster_ros node that can visualize recorded or real-time point cloud messages. Most open-source robot systems use ROS nodes to run different nodelets in the system. The nodelet package in ROS is designed to provide a way to run multiple algorithms in the same process with zero copy transport between algorithms.

3 SLAM Algorithms

There are several different paradigms to solve the SLAM problem. This chapter will review the three main paradigms used for SLAM, for which most others are derived. EKF(extended Kalman filter)-SLAM is historically the first introduced, but it has become less popular due to its limiting computational properties. Particle filtering is a popular method for online-SLAM and provides a perspective on addressing the data association problem in SLAM. Graph-Based SLAM is based on graphical representations and successfully applies sparse nonlinear optimization methods for solving the full SLAM problem.

3.1 EKF SLAM

As mentioned in chapter 2, Smith, Self and Cheeseman were the first to propose the use of a single state vector to estimate the locations of a robot and a set of features in the environment. Together with this, an error covariance matrix represented the uncertainty in these estimates, including a correlation between the vehicle and feature state estimates. Based on this the EKF-SLAM algorithm was developed.

Simply put, the EKF-SLAM algorithm applies the extended Kalman Filter to online SLAM using maximum likelihood data association. As any EKF-algorithm, EKF-SLAM makes a Gaussian noise assumption for the robot motion and observed surroundings. The system covariance matrix mentioned above grows quadratically with the number of landmarks. Therefore, EKF-SLAM uses a feature-based map, composed of a relatively small number of point landmarks (>1000). The number of landmarks is preferably kept low for computational reasons.

The EKF algorithm represents the robot estimate by a multivariable Gaussian shown in equation 5. The vector μ_t represents the state vector including the estimate of the robot's location and location of the features in the map. The dimension of μ_t depends on the system. If the system were a planar surface robot, the dimension would be $3+2N$, as three variables are needed to represent the location and $2N$ for the N landmarks.

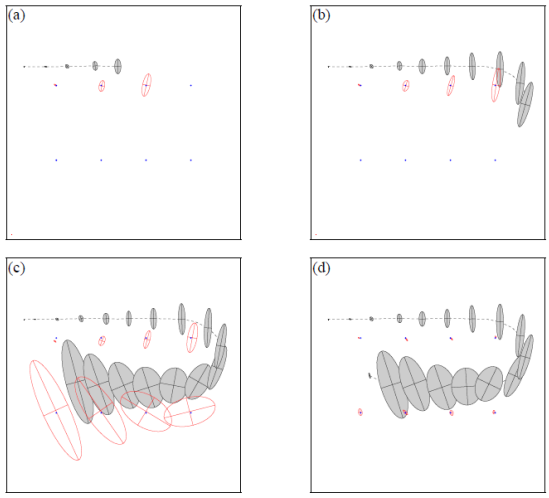


Figure 4: EKF-SLAM (*image courtesy of Michael Montemerlo, Standford University*)

The matrix Σ_t is the error covariance matrix used as the expected error for the guess μ_t. With the same example as above, the covariance matrix is of size $(3+2N)\cdot(3+2N)$, making the problem quadratic. The off-diagonal elements in the covariance matrix represent the correlations in the estimates of different variables. Since the robot's location is uncertain, and therefore the landmarks locations are uncertain, the nonzero correlations are included. In appendix A the complete algorithm for updating μ_t and Σ_t is shown, see [20] for an complete review. The algorithm uses an incremental maximum likelihood (ML) estimator to determine the correspondences.

$$\boldsymbol{p}(\boldsymbol{x}_t, \boldsymbol{m} \mid \boldsymbol{Z}_t, \boldsymbol{U}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \tag{5}$$

Figure 4 illustrates the EKF-SLAM algorithm. The ellipses shown in the figure represents uncertainty in position (grey) and in landmark location (transparent). As the system moves, the pose uncertainty increases because of the errors in odometry mea-

surements. The system also senses landmarks in the environment and maps these with an uncertainty that combines the uncertainty in the measurement and the systems pose. This result in increasing uncertainty also for the landmark positions. The most important part happens in the last frame of the figure. Here the system observes the same landmark it did in the beginning. Through this observation, the systems state error is reduced, along with the uncertainty of the landmark locations. This happens because the uncertainty in the location estimates of the landmarks is vastly affected by the error in the position. The correlation in the covariance matrix spreads through the whole matrix and updates the previous landmark estimates. This effect is probably the most important in EKF-SLAM. As mentioned above, to solve the problem of uncertain data association, the system can use an ML-estimator to approximate which of the landmarks in the map most likely corresponds to the landmark just observed [22].

A key limitation to the method is the computational complexity. Sensor updates require time quadratic in the number of landmarks N to compute. This complexity stems from the fact that the covariance matrix maintained by the Kalman filters has $O(N^2)$ elements of which must be updated even if just a single landmark is observed. This limits the method for sets with a large number of landmarks [19][20].

3.2 Particle filtering

Particle filtering has become popular in recent decades. Particle filters is a recursive Bayes filter, where it represents a posterior through a set of particles. The posterior is the distribution of possible unobserved values conditional on the observed values. In SLAM, each particle is best taught as a concrete guess as to what the true value of the state may be, see figure 5 for an illustrated example. By collecting many such guesses into a set of guesses, or a set of particles, the particle filter approximates the posterior distribution. Under mild conditions, the particle filter has been shown to approach the true posterior values as the particle set size goes to infinity. The particle filter is also not limited to Gaussian distributions.

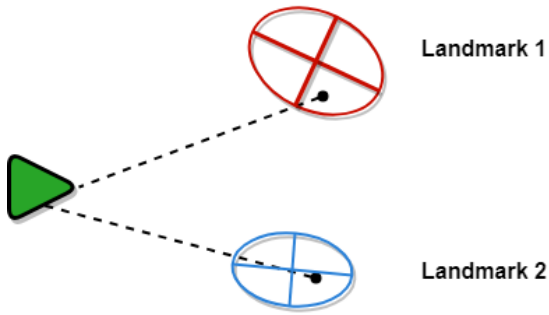


Figure 5: Particle illustration, making guesses of obstacle states

$$\boldsymbol{x} = (\boldsymbol{x}_{x:t}, \boldsymbol{m}_{1,x}, \boldsymbol{m}_{1,y} \ldots . \boldsymbol{m}_{N,x}, \boldsymbol{m}_{N,y}) \tag{6}$$

The key problem with the particle filter in the context of SLAM is that the path and the map tend to be huge. Particle filters scale exponentially with the dimension of the underlying state space. Three or four dimensions are thus acceptable, but e.g. 100 dimensions are generally not, as seen from equation 6. The trick in making particle filters applicable to the SLAM problem is to use the particle set only to model the system's path, then each sample of particle sets is a path hypothesis. Now an individual map of landmarks can be computed. This key idea is known as the *Rao-Blackwellization* or *fastSLAM*.

Rao-Blackwellization performs a marginalization over the probability distribution in the state space. Instead of using sampling to represent the multivariate probability distribution of the state space, marginalizing out a subset of the state space, is a much more efficient method when using a Gaussian distribution, as seen in figure 6 where the map is marginalized to several smaller maps. This marginalization has become very popular in SLAM problems because jointly sampling over position and map is impractical. When the creators of FastSLAM realized that Rao-Blackwellization could help to marginalize the maps from the joint distribution, the SLAM problem became

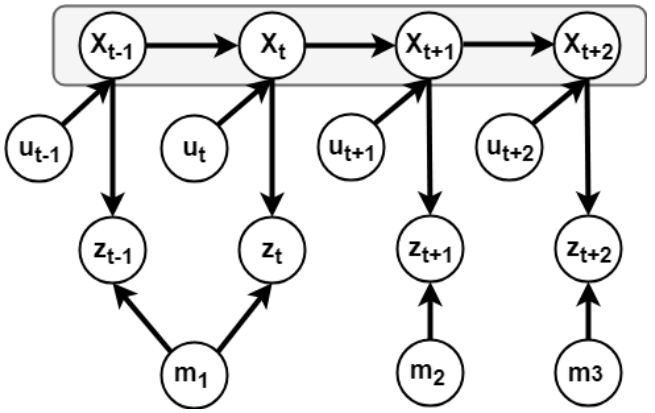


Figure 6: Graphical model of fast SLAM map marginalization

much more tractable.

FastSLAM is an algorithm that recursively estimates the full posterior distribution over the systems pose and landmark locations, yet scales logarithmically with the number of landmarks in the map. The algorithm has been run successfully on as many as 50000 landmarks [13]. FastSLAM decomposes the SLAM problem into a robot localization problem and a collection of landmark estimation problems that are conditioned on the robots pose estimate. Each particle in FastSLAM processes N Kalman filters that estimate the N landmark locations conditioned on the path estimate. This results in an algorithm requiring $O(MN)$ time, where M is the number of particles. By developing a tree-based data structure, the FastSLAM algorithm obtains $O(MlogN)$ running time.

$$\boldsymbol{x}_t^{[k]} \sim \boldsymbol{p}(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{[k]}, \boldsymbol{u}_t) \tag{7}$$

$$\boldsymbol{\omega}^{[k]} = |2\pi \boldsymbol{Q}|^{[k]}\ exp(-\frac{1}{2}(\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]})^T\ \boldsymbol{Q}^{-1}\ (\boldsymbol{z}_t - \hat{\boldsymbol{z}}^{[k]}) \tag{8}$$

The key steps of the fastSLAM algorithm are shown in equation 7 and 8. Equation 7 extends the path posterior by sampling a new pose for each sample. Equation 8 computes the particle weighting parameter $\omega^{[k]}$ that helps the algorithm update the belief of each observed landmarks for each sample, and then resample. k represents the particle number, Q is the measurement covariance matrix, and $\hat{z}^{[k]}$ is the expected observation. See appendix A for the full algorithm, and [13] for a complete review.

3.3 Graph-Based SLAM

In Graph-Based SLAM, landmarks and robot locations can be thought of as nodes in a graph. Every consecutive pair of locations x_{t-1}, x_t is tied together by an edge/spatial-constraint that represents the information conveyed by the control reading u_t. Edges also exist between the nodes that correspond to locations u_t and landmarks m_i, assuming a landmark is observed, represented by the orange arcs in figure 7. Edges in this graph are soft constraints. Relaxing these constraints and finding a node configuration that minimizes the error in the edges yields the robot's best estimate for the map and the full path.

The construction of the graph is illustrated in figure 7. Suppose at time t_{t-1}, the robot senses landmark m_1. The edge between these two nodes is added to the yet incomplete graph, as shown by the red line. When caching the edges in a matrix format (which happen to correspond to a quadratic equation defining the resulting constraints), a value is added to the elements between x_{t-1} and m_1. If the system now moves, the control readings will create a new edge between x_{t-1} and x_t. By doing this in consecutive fashion, edges from observations and controls lead to a graph-matrix of increasing size. Nevertheless, this graph is sparse, in that each node is only connected to a small number of other nodes (assuming a sensor with limited sensing range). The number of constraints in the graph is (at worst) linear in the time elapsed and in the number of nodes in the graph.

In figure 8 and figure 9 an implementation of a Graph-based SLAM algorithm is shown. Every node in the graph corresponds to a robot position and a laser measurement. We

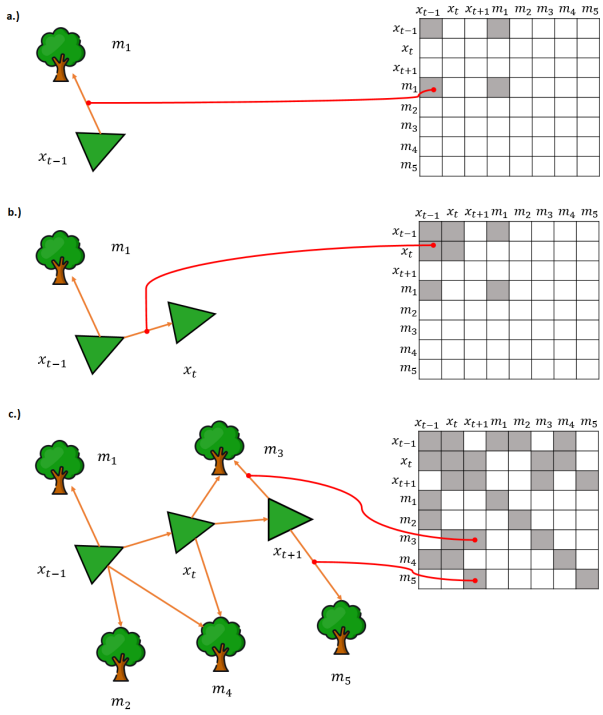


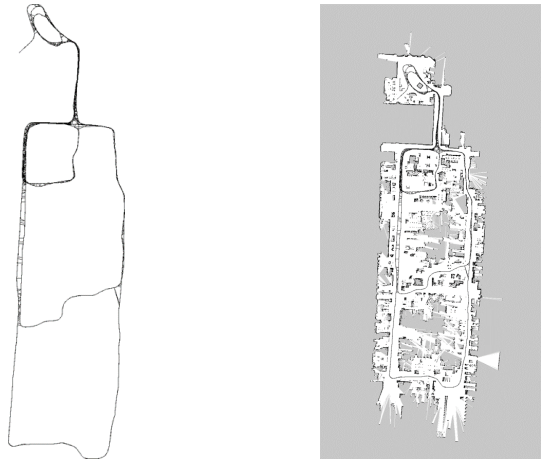
Figure 7: Caption

can see in the first figure that the uncertainty from the measurements and controls is huge. By letting edges between two nodes be represented by spatial constraints, a graph is made in the last frame of figure 8. Once the graph is obtained the most likely map is obtained by correcting the nodes, figure 9. Finally, the map can then be rerendered based on the now known poses.

Graph-SLAM methods were originally used to solve the full SLAM problem offline, but more recent techniques handles the online-SLAM problem by incrementing and re-using the previously computed solutions. In comparison to EKF-SLAM, graphical methods scale to higher-dimensional maps, exploiting the sparsity in the graphs.

(a) (b) (c)

Figure 8: Graph-Based SLAM, KUKA Halle 22, courtesy of P. Pfaff & G. Grisetti



(a) (b)

Figure 9: Graph-Based SLAM, KUKA Halle 22, courtesy of P. Pfaff & G. Grisetti

4 Methods

4.1 HDL Graph Slam

High Definition LiDAR (HDL) graph SLAM, is an open source ROS package for real-time 6DOF(degrees of freedom) SLAM using a 3D Lidar. It is a 3D Graph-based SLAM method with normal distribution transform (NDT) [3] scan matching-based odometry estimation and loop detection. It also supports several, graph constraints including GPS, IMU acceleration and orientation, and floor constraints detected in point clouds. Information and code is obtained from the hdl_graph_slam Github branch [10].

The HDL Graph SLAM algorithm consists of four ROS nodelets.

- **Prefiltering**
- **Scan Matching**
- **Floor detection**
- **HDL_Graph_SLAM**

The input cloud from the LiDAR measurements is firstly downsampled by the prefiltering nodelet. The filtered points from the prefiltering nodelet are passed through to the scan matching nodelet which estimates the sensor pose by iteratively applying a scan matching between consecutive frames. Consecutively the floor detection nodelet detects floor planes by RANSAC [6]. The estimated odometry and the detected floor planes are sent to HDL Graph SLAM. To compensate for the accumulated error of scan matching, the algorithm performs loop detection and optimizes a pose graph which takes various constraints into account.

4.2 Cartographer

Cartographer is a real-time 2D and 3D SLAM system working on multiple platforms developed by Google. Cartographer builds globally consistent maps in real-time across

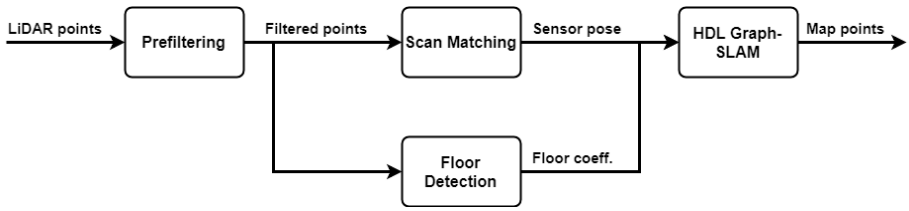


Figure 10: Illustration of HDL Graph SLAM nodelets

a broad range of sensor configurations. Google released Cartographer as an open-source SLAM library in 2016. Although the system works with a range of sensors, the algorithm is developed with a heavy focus in LiDAR SLAM.

A high-level system overview is illustrated in figure 11, were the system is divided into three main parts.

- **Input Sensor Data**
- **Local SLAM**
- **Global SLAM**

The first part provides the system with sensor information. The local SLAM part matches range data with IMU and pose estimates and uses this to update a Voxel grid. A voxel grid represents a value on a regular grid in three-dimensional space. In SLAM voxel grids are used to create the map from the observed scan. Cartographer uses OctoMap [7], which is a common framework for 3D mapping, and has the advantage of lossless compression and compact map files in addition to being open-source. After producing the 3D point map, the Global SLAM part adds constraints to the map and optimizes the robots position [15].

4.3 SegMap & SegMatch

SegMap is a segment based approach for map representation in localization and mapping with 3D sensors. The robot's surroundings are decomposed into a set of

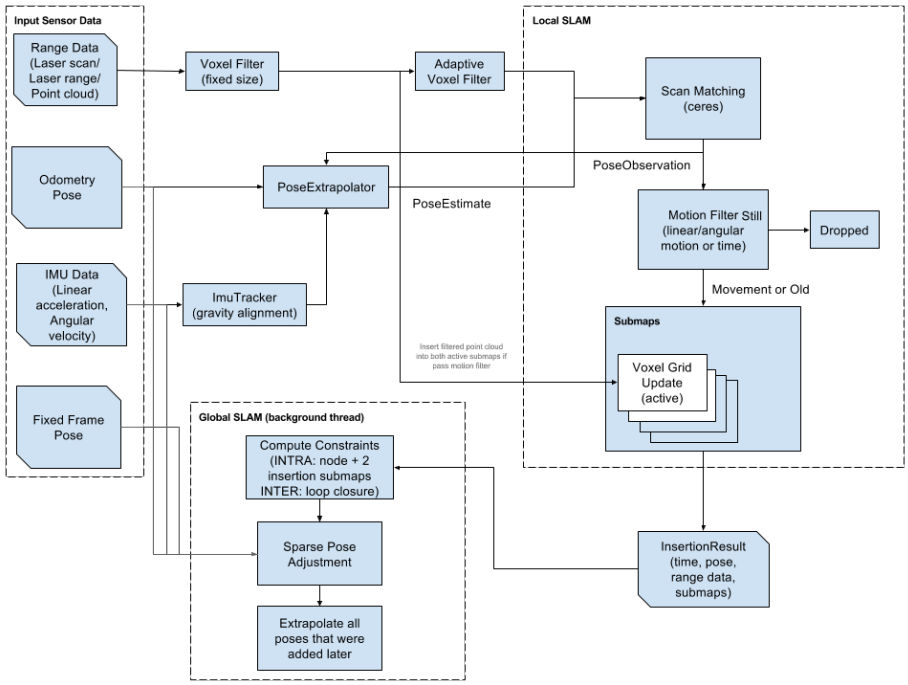


Figure 11: High-level system overview. Courtesy of Google.

segments, and each segment is represented by a distinctive, low dimensional learning based descriptor. Segmap is able to reconstruct the environment while achieving a high compression rate allowing for SLAM with 3D LiDARS at a large scale [4].

The developers of SegMap also developed an algorithm called SegMatch. SegMatch is a loop-closing algorithm designed for loop-closing from 3D laser data. In figure 12 a block diagram of SegMatch is presented. The Segmentation block segments point clouds into distinct elements for matching. Feature extraction from the distinct elements is then performed by a descriptor. Similar features from different segments are matched in the next step using the RANSAC approach. A detailed description is given in [5].

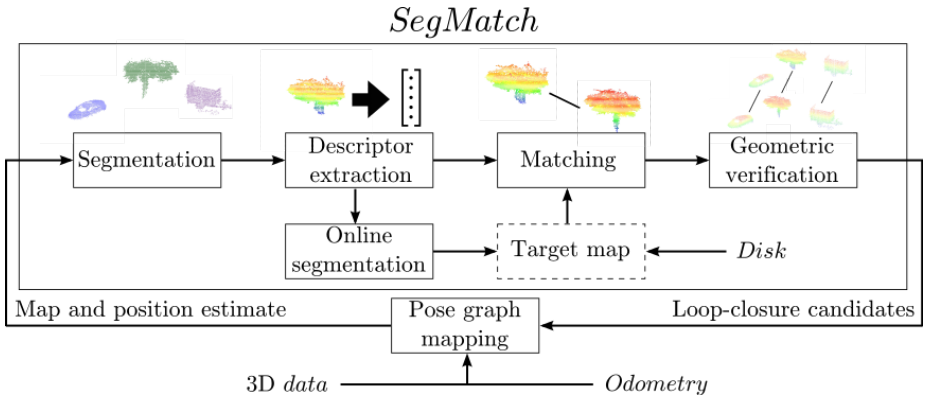


Figure 12: SegMatch block diagram. Courtesy of Dubé, Dugas, Stumm, Nieto, Siegwart and Cadena

5 Future work

5.1 Master Thesis

As this literature review concludes, the building of a system is set to take place from January - June 2019 as a master thesis. There is no set task, other than to apply the knowledge touch upon in this report to develop a test platform for a system that runs autonomously using SLAM with the Ouster1 LiDAR. There are nonetheless several important subjects that need to be touched upon in the master thesis for this to happen. The most obvious being:

- Obtaining datasets of some local area
- Visualize obtained point clouds and localize the measuring device
- Attach the measurement device to some platform and develop control structures
- Test the platform with different SLAM approaches in different types of environments

The first task has some fairly straightforward procedures developed by Ouster and should be no problem to complete in the early stages of the master thesis time span. As for the second part, the visualizing is already tested with Ousters ROS package, while the localizing is more of a challenge. At this time there are no sources that has developed this part, except for Ouster that have shown a video of their system running on SLAM. Ouster has not provided any information on how to solve the SLAM problem using their sensors, but similar systems with a similar product should be sufficient enough to develop a platform for this task. As the first two parts progress, we need to decide upon a platform for the system to run on. There are many applicable alternatives, but this decision needs a lot of planning as it will in a large majority decide the outcome of the thesis. The last and probably the most important part is to test the system in different environments and test how the system responds to the use of the Ouster-1 LiDAR. If time allows it, and the faculties resources allow for testing of similar products, comparisons of the solution on different sensors would pay off

as a great way of presenting the results. Figure 13 shows a draft of a Class Diagram, visualizing how the system could be modeled, containing the most important sections that the proposed plan above needs to comprise.

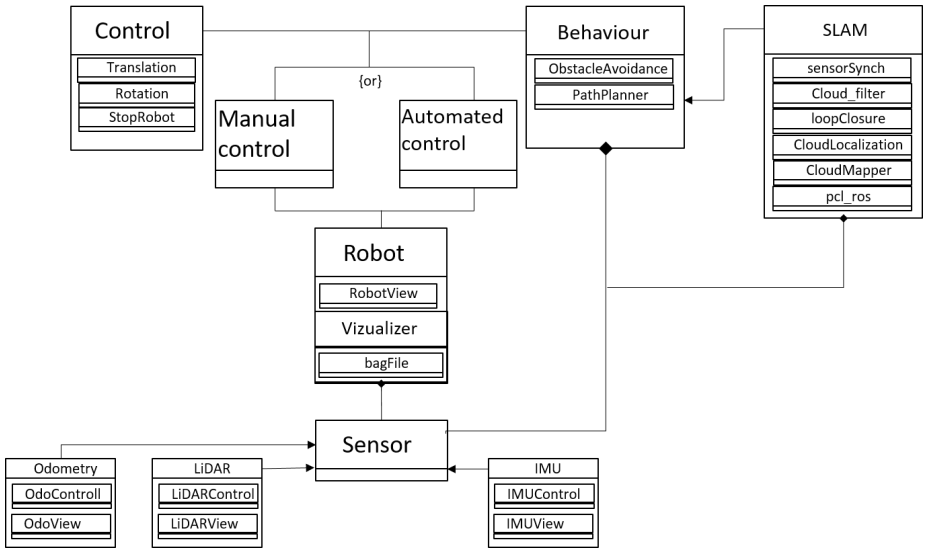


Figure 13: Class Diagram

5.2 Stakeholders

The technology and development of SLAM is a very important part of true autonomous navigation. If a well-developed platform works with the OUSTER-1 LiDAR, which is a cheaper alternative to its competitors, works in different environments the possibility for growth of the technology is huge. Development in this field of technology will have stakeholders in industrial solutions, autonomous driving and further development at this institute.

6 Conclusion

After reviewing different methods for solving the problem, only some of the many has been highlighted. The only testing that has been done in this project is the visualization of some provided test data by Ouster-1. It is therefore difficult to conclude on which method that will fit in future work before they all are tested. Aside from this, I think that the HDL_Graph_SLAM method presented in subsection 4.1 fits the proposed process described in Chapter 5 Future work, and should on paper be a great candidate to solve the visualization and mapping problem of the system. I say this because the method is well documented and specifically aims to solve the problem using 3D LiDARs. The package is tested on the Velodyne VLP16 sensor, which is the very similar counterpart to the Ouster-1 16 beam sensor.

The Cartographer solutions is also an option and should be considered as it is developed by Google, which is known for their innovative and solid solutions. I have heard from other students with similar projects, that the Cartographer algorithm may be hard to implement as it requires fine-tuned parameters. In spite of that, a test of this solution should be fitting in the master thesis.

In addition to the methods targeted above, this report concludes that SLAM is a complex and difficult problem, but with the right tuning and constraints a solvable and applicable solution for a vast majority of different scenarios.

References

[1] [n.d.].
URL: *http://www.lidar-uk.com/how-lidar-works/*

[2] Bailey, T. and Durrant-Whyte, H. [n.d.]. Simultaneous Localisation and Mapping (SLAM): Part II State of the Art, p. 10.

[3] Biber, P. and Strasser, W. [2003]. The normal distributions transform: a new approach to laser scan matching, *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, Vol. 3, IEEE, Las Vegas, Nevada, USA, pp. 2743–2748.
URL: *http://ieeexplore.ieee.org/document/1249285/*

[4] Dubé, R., Cramariuc, A., Dugas, D., Nieto, J., Siegwart, R. and Cadena, C. [2018]. SegMap: 3d Segment Mapping using Data-Driven Descriptors, *Robotics: Science and Systems XIV*, Robotics: Science and Systems Foundation.
URL: *http://www.roboticsproceedings.org/rss14/p03.pdf*

[5] Dubé, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R. and Cadena, C. [2016]. SegMatch: Segment based loop-closure for 3d point clouds, *arXiv:1609.07720 [cs]* . arXiv: 1609.07720.
URL: *http://arxiv.org/abs/1609.07720*

[6] Fontanelli, D., Ricciato, L. and Soatto, S. [2007]. A Fast RANSAC-Based Registration Algorithm for Accurate Localization in Unknown Environments using LIDAR Measurements, *2007 IEEE International Conference on Automation Science and Engineering*, IEEE, Scottsdale, AZ, USA, pp. 597–602.
URL: *http://ieeexplore.ieee.org/document/4341827/*

[7] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. and Burgard, W. [2013]. OctoMap: an efficient probabilistic 3d mapping framework based on octrees, *Autonomous Robots* **34**(3): 189–206.
URL: *http://link.springer.com/10.1007/s10514-012-9321-0*

[8] Hähnel, D., Thrun, S., Wegbreit, B. and Burgard, W. [2005]. Towards Lazy Data Association in SLAM, *in* B. Siciliano, O. Khatib, P. Dario and R. Chatila (eds), *Robotics Research. The Eleventh International Symposium*, Vol. 15, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 421–431.
URL: *http://link.springer.com/10.1007/11008941_45*

[9] Kohlbrecher, S., von Stryk, O., Meyer, J. and Klingauf, U. [2011]. A flexible and scalable SLAM system with full 3d motion estimation, *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, IEEE, Kyoto, Japan, pp. 155–160.
URL: *http://ieeexplore.ieee.org/document/6106777/*

[10] koide3 [2018]. 3d LIDAR-based Graph SLAM. Contribute to koide3/hdl_graph_slam development by creating an account on GitHub. original-date: 2018-01-01T07:35:43Z.
URL: *https://github.com/koide3/hdl_graph_slam*

[11] Labbé, M. and Michaud, F. [2018]. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation: LABBÉ and MICHAUD, *Journal of Field Robotics* .
URL: *http://doi.wiley.com/10.1002/rob.21831*

[12] Leonard, J. J. and Durrant-Whyte, H. F. [1991]. Mobile robot localization by tracking geometric beacons, *IEEE Transactions on Robotics and Automation* **7**(3): 376–382.

[13] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B. [n.d.]. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, p. 6.

[14] Neira, J. and Tardos, J. [2001]. Data association in stochastic mapping using the joint compatibility test, *IEEE Transactions on Robotics and Automation* **17**(6): 890–897.
URL: *http://ieeexplore.ieee.org/document/976019/*

[15] Nüchter, A., Bleier, M., Schauer, J. and Janotta, P. [2018]. Continuous-Time SLAM—Improving Google's Cartographer 3d Mapping, *Latest Developments in Reality-Based 3D Surveying and Modelling*, MDPI.
URL: *http://www.mdpi.com/books/pdfdownload/edition/786/3*

[16] Olson, E., Leonard, J. and Teller, S. [2006]. Fast iterative alignment of pose graphs with poor initial estimates, *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, Orlando, FL, USA, pp. 2262–2269.
URL: *http://ieeexplore.ieee.org/document/1642040/*

[17] Smith, R. C. and Cheeseman, P. [1986]. On the Representation and Estimation of Spatial Uncertainty, *The International Journal of Robotics Research* **5**(4): 56–68.
URL: *http://journals.sagepub.com/doi/10.1177/027836498600500404*

[18] Smith, R., Self, M. and Cheeseman, P. [n.d.]. Estimating Uncertain Spatial Relationships in Robotics, p. 26.

[19] Stachniss, C., Leonard, J. J. and Thrun, S. [2016]. Simultaneous Localization and Mapping, *in* B. Siciliano and O. Khatib (eds), *Springer Handbook of Robotics*, Springer International Publishing, Cham, pp. 1153–1176.
URL: *https://doi.org/10.1007/978-3-319-32552-1_46*

[20] Thrun, S. [2002]. Probabilistic robotics, *Communications of the ACM* **45**(3).
URL: *http://portal.acm.org/citation.cfm?doid=504729.504754*

[21] Thrun, S., Fox, D., Burgard, W. and Dellaert, F. [2001]. Robust Monte Carlo localization for mobile robots, *Artificial Intelligence* **128**(1-2): 99–141.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S0004370201000698*

[22] Tipaldi, G. D., Spinello, L. and Burgard, W. [2013]. Geometrical FLIRT phrases for large scale place recognition in 2d range data, *2013 IEEE International Conference on Robotics and Automation*, IEEE, Karlsruhe, Germany, pp. 2693–2698.
URL: *http://ieeexplore.ieee.org/document/6630947/*

A Appendix

1: **Algorithm EKF_SLAM**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, N_{t-1})$**:**

2: $N_t = N_{t-1}$

3: $F_x = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$

4: $\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$

5: $G_t = I + F_x^T \begin{pmatrix} 0 & 0 & \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & \frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$

6: $\bar{\Sigma}_t = G_t\ \Sigma_{t-1}\ G_t^T + F_x^T\ R_t\ F_x$

7: $Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$

8: for all observed features $z_t^i = (r_t^i\ \ \phi_t^i\ \ s_t^i)^T$ do

9: $\begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \\ \bar{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$

10: for $k = 1$ to N_t+1 do

11: $\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix}$

12: $q_k = \delta_k^T \delta_k$

13: $\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \text{atan2}(\delta_{k,y}, \delta_{k,x}) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{k,s} \end{pmatrix}$

14: $F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$

15: $H_t^k = \frac{1}{q_k} \begin{pmatrix} \sqrt{q}_k\delta_{k,x} & -\sqrt{q}_k\delta_{k,y} & 0 & -\sqrt{q}_k\delta_{k,x} & \sqrt{q}_k\delta_{k,y} & 0 \\ \delta_{k,y} & \delta_{k,x} & -1 & -\delta_{k,y} & -\delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$

16: $\Psi_k = H_t^k\ \bar{\Sigma}_t\ [H_t^k]^T + Q_t$

17: $\pi_k = (z_t^i - \hat{z}_t^k)^T\ \Psi_k^{-1}\ (z_t^i - \hat{z}_t^k)$

18: endfor

19: $\pi_{N_t+1} = \alpha$

20: $j(i) = \underset{k}{\text{argmin}}\ \pi_k$

21: $N_t = \max\{N_t, j(i)\}$

22: $K_t^i = \bar{\Sigma}_t\ [H_t^{j(i)}]^T \Psi_{j(i)}^{-1}$

23: endfor

24: $\mu_t = \bar{\mu}_t + \sum_i K_t^i\ (z_t^i - \hat{z}_t^{j(i)})$

25: $\Sigma_t = (I - \sum_i K_t^i\ H_t^{j(i)})\ \bar{\Sigma}_t$

26: return μ_t, Σ_t

1: **FastSLAM1.0_known_correspondence**$(z_t, c_t, u_t, \mathcal{X}_{t-1})$**:**

2: *for* $k = 1$ *to* N *do* // *loop over all particles*

3: *Let* $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots \right\rangle$ *be particle* k *in* $\mathcal{X}_{t-1}$

4: $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // *sample pose*

5: $j = c_t$ // *observed feature*

6: *if feature* j *never seen before*

7: $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // *initialize mean*

8: $H = h'(\mu_{j,t}^{[k]}, x_t^{[k]})$ // *calculate Jacobian*

9: $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // *initialize covariance*

10: $w^{[k]} = p_0$ // *default importance weight*

11: *else*

12: $\hat{z}^{[k]} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *measurement prediction*

13: $H = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *calculate Jacobian*

14: $Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // *measurement covariance*

15: $K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // *calculate Kalman gain*

16: $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}^{[k]})$ // *update mean*

17: $\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // *update covariance*

18: $w^{[k]} = |2\pi Q|^{-\frac{1}{2}}\ \exp\left\{-\frac{1}{2}(z_t - \hat{z}^{[k]})^T\, Q^{-1}\, (z_t - \hat{z}^{[k]})\right\}$ // *importance factor*

19: *endif*

20: *for all unobserved features* j' *do*

21: $\langle \mu_{j',t}^{[k]}, \Sigma_{j',t}^{[k]} \rangle = \langle \mu_{j',t-1}^{[k]}, \Sigma_{j',t-1}^{[k]} \rangle$ // *leave unchanged*

23: *endfor*

24: *endfor*

25: $\mathcal{X}_t = \text{resample}\left(\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, w^{[k]} \right\rangle_{k=1,\ldots,N}\right)$

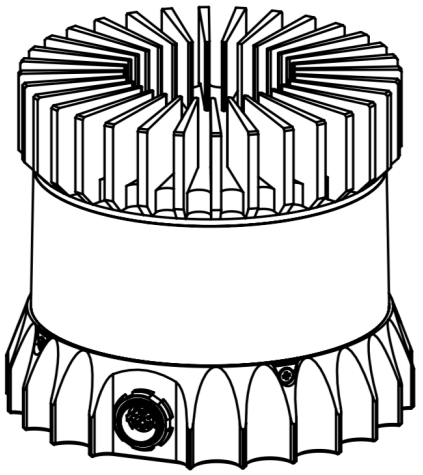
26: *return* $\mathcal{X}_t$

OUSTER

Ouster, Inc.
350 Treat Ave
San Francisco, CA 94110
LIDAR@ouster.io

OS-1

High Resolution Imaging LIDAR



SUMMARY

The OS-1 offers a market leading combination of price, performance, reliability and SWAP. It is designed for indoor/outdoor all-weather environments and long lifetime. As the smallest high performance LIDAR on the market, the OS-1 can be directly integrated into vehicle facias, windshield, side mirrors, and headlight clusters.

HIGHLIGHTS

- Fixed resolution per frame operating mode
- Camera-grade ambient and intensity data
- Multi-sensor crosstalk immunity
- Industry leading intrinsic calibration
- Open source drivers

OPTICAL PERFORMANCE

Range	0.5 m - 120 m @ 80% reflective lambertian target, 225 w/m2 sunlight, SNR of 12 0.5 m - 40 m @ 10% reflective lambertian target, 225 w/m2 sunlight, SNR of 12 * range of 0.0-80m (min range of 0.0m) in enhanced low range mode
Range Accuracy	Zero bias for lambertian targets, slight bias for retroreflectors
Range Resolution	1.2 cm
Range Repeatability (1 sigma / standard deviation)	SNR >250: ± 1.5cm SNR 100: ± 3 cm SNR 12: ± 10 cm
Vertical Resolution	64 or 16 beams
Horizontal Resolution	2048, 1024, or 512 (configurable)
Field of View	Vertical: +16.6° to -16.6 ° (33.2°) - uniform spacing / Horizontal: 360°
Angular Sampling Accuracy	Vertical: ±0.01° / Horizontal: ±0.01°
Rotation Rate	10 to 20 hz (configurable)
# of Returns	1 (strongest)

LASER

Laser Product Class	Class 1 eye-safe per **[IEC 60825-1:2007 & 2014]**
Laser Wavelength	850 nm
Beam Diameter Exiting Sensor	10 mm
Beam Divergence	0.13° (FWHM)

LIDAR OUTPUT

Connection	UDP over gigabit ethernet
Point Per Second	1,310,720 (64-channel) 327,680 (16-channel)
Data Per Point	Range, intensity, reflectivity, ambient, angle, time stamp
Time Stamp Resolution	10 ns
Data Latency	< 10 ms

IMU OUTPUT

Connection	UDP over gigabit ethernet
Samples Per Second	1,000
Data Per Sample	3 axis gyro, 3 axis accelerometer
Time Stamp Resolution	10 ns

Data Latency	< 10 ms

CONTROL INTERFACE

Connection	TCP over gigabit ethernet
Time Synchronization	Input sources: • IEEE1588 precision time protocol • External PPS • Internal 10 ppm drift clock Output sources: • Configurable 1-60Hz output pulse
LIDAR Operating Modes	Hardware triggered angle firing (guaranteed fixed resolution per rotation): • 64 x 2048 @ 10hz • 64 x 1024 @ 10hz or 20hz • 64 x 512 @ 10hz or 20hz Fixed timing firing: • Configurable measurement period between 50 µsec and 1 second
Additional Programmability	Multi-sensor rotation phase tuning Queryable intrinsic calibration information: • Beam angles • IMU pose correction matrix

MECHANICAL/ELECTRICAL

Power Consumption	14-16 W typical, 18 W peak
Operating Voltage	22-26 V, 24 V nominal
Connector	Proprietary pluggable connector (Power + data + DIO)
Dimensions	Diameter: 85 mm (3.34 in) Height: 73 mm (2.87 in)
Weight	380 g (13.4oz)
Mounting	4 M3 screws / 2 locating 3mm pins

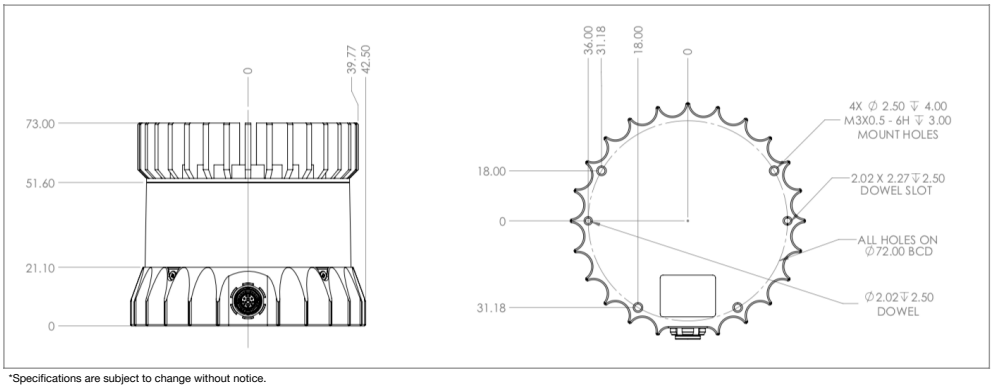
OPERATIONAL

Operating Temperature	-20C to +50C (with Mount)
Storage Temperature	-40C to +105C
Ingress	IP67
Shock	500 m/s2 amplitude, 11 ms duration
Vibration	5 Hz to 2,000 Hz, 3 Grms
Certification	FCC, CE, RoHS

ACCESSORIES

Included Interface Box	PolyCarb/FR4, 100g, 75mm x 50mm x 25mm (LxWxH), 2m CAT6 cable, 24V power adapter, 5m sensor cable
Optional Mount	Aluminum, 530g, 110mm x 110mm x 20.5mm (LxWxH), 4x M8 thru holes

EXTERIOR DIMENSIONS



*Specifications are subject to change without notice.