SIMULTANEOUS LOCALIZATION AND MAPPING FOR SEMI-SPARSE POINT CLOUDS

Payam Shokrzadeh[1]

[1] Leo Muhendislik, Hacimimi Mah. Beyoglu Istanbul, Turkey - payam.shokrzadeh@leomuhendislik.com

KEY WORDS: SLAM, Point cloud, Semi-sparse point cloud, Laser scanner

ABSTRACT:

3D representation of the environment is a piece of vital information for most of the engineering sciences. However, providing such information in classical surveying approaches demands a considerable amount of time for localizing the sensor in a desired coordinate frame to map the environment. Simultaneous Localization And Mapping (SLAM) algorithm is capable of localizing the sensor and do the mapping while the sensor is moving through the environment. In this paper, SLAM will be applied on the data of a lightweight 3D laser scanner in which we call semi-sparse point cloud, because of the unique specifications of the point cloud which comes from various resolutions in vertical and horizontal directions. In contrast to most of the SLAM algorithms, there is no aiding sensor to provide prior information of motion. The output of the algorithm would be a high-density full geometry detailed map in a short time. The accuracy of the algorithm has been estimated in a medium scale simulated outdoor environments in Gazebo and Robot Operating System (ROS). Considering Velodyne Puck accuracy which is 3cm, the map was generated with approximately 6 cm accuracy.

1. INTRODUCTION

According to Cadena et al (Cadena et al., 2016) the first 20 years of the SLAM history which is from 1986 to 2004 is the classical age of this problem. Researchers were mostly focused on the main probabilistic formulations such as Extended Kalman Filters (EKF), Particle Filters (PF), etc. The subsequent period has named as algorithmic-analysis age from 2004-2015 which was mainly focused on the study of the fundamental properties of SLAM including observability, convergence, and consistency. Table 1 gives information about researches and developments in the second age of SLAM.

Year	Topic
2006	Probabilistic approaches and data association
2008	Filtering approaches
2011	SLAM back-end
2011	Observability, consistency and convergence
2012	Visual Odometry
2016	Multi robot SLAM
2016	SLAM in the Handbook of Robotics
2016	Theoretical aspects

Table 1. Researches of the second age
(Cadena et al., 2016)

Younes et al in (Younes et al., 2017) divided the SLAM algorithms into two different categories which are filter-based (before 2010) and keyframe-based (after 2010) architectures. Keyframe-based algorithms are mostly based on the pose graph optimization or Bundle Adjustment. Most of the SLAM algorithms are based on the visual sensors as the main sensor. The reason is that cameras are inexpensive, lighter and it is easier to extract more information from images rather than laser scanner data or RADAR information. Each kind of sensor has its own pros and cons. For example, Cameras have some limitation such as limited field of view (FOV), scale ambiguity, intrinsic calibration estimation, etc but also they will provide a dense tensor of the environment which is very similar to the visual information of the human. For those laser scanners such as Velodyne or Hokuyo, it is possible to have 3D information of the surrounding area as the output of the sensor however the density of the points are not comparable with the cameras. As another option, stereo cameras or RGB-D sensors like Kinect can provide a dense 3D coordinate of the surrounding environment but the range of these kinds of sensors are not more than a couple of meters.

Zhang and Singh (Zhang, Singh, 2014) have proposed a method to estimate the odometry information by rotating a 2D line laser scanner around its axis and matching the receiving points of each scan. In order to reach that aim, they had to use an encoder in order to calculate the orientation of the laser scanner according to the rotating axis and then register each consecutive scan with respect to the previous one.

In all of the mentioned algorithms, the most important step of the algorithm is to register the new coming data with respect to the previous data. Therefore, using the best method of registration is crucial to achieving the best estimation.

According to Holz et al (Holz et al., 2015), to register a point cloud with respect to another one, there are two main approaches. Iterative Closest Point (ICP) which is a well-known approach for point cloud registration and key point extraction and feature matching. However, not all the point clouds have the same specifications to perform either one or both of those approaches.

The most important application of the SLAM is in robotics sciences. However, nowadays, it is used to generate the 3D model from complex objects and/or environments such as archeological sites/objects, forests and plants, real-time change detection, Building Information Modeling (BIM), As-built construction maps, and any other application which needs real-time or post-processed High Density maps (HD Maps). This study is about applying the SLAM on a mobile system in real time, using a specific type of point cloud which in this paper named as the semi-sparse point cloud.

2. METHODS

SLAM does the localization by comparing some features which are available on the map with those which have found by the algorithm. As a matter of fact, the final step of the localization is actually solving a transformation problem, which means, we have two sets of points in different coordinate systems and by knowing the relations between the points, transformation problem is solvable. Therefore we need a map for localization. However, as discussed in the previous section, SLAM should be able to generate the map. For map generation, we need transformation parameters in order to move from the sensor coordinate frame to the global coordinate frame. Considering these explanations, we need a map for localization and we need localization to generate the map.

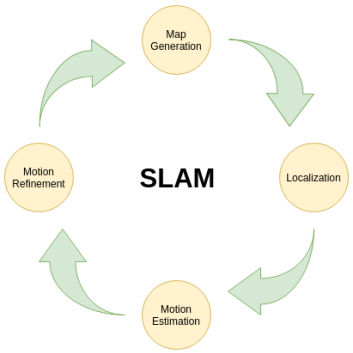

Figure 1. SLAM big picture

Figure 1 shows four main steps of the SLAM, which are initial motion estimation (Odometry), motion refinement, map generation and localization.
It is common to use aiding sensors such as Inertial Measurement Unit (IMU), Odometer, etc. to provide initial motion estimation. However, in this paper initial motion will be estimated from the data itself. As a common approach to initialize the coordinate system, local information of the sensor at time t_0 will be a starting point.

$$C_{global_{t_0}} = T_{senor_{t_0}} \tag{1}$$

In the above equation, C_{global} is the reference point of the SLAM agorithm and $T_{\text{sensor}_{t_0}}$ is the transformation parameter if the sensor at time t_0. It is common to select the $T_{\text{sensor}_{t_0}}$ as Identity matrix like equation 2.

$$T_{senor_{t_0}} = Identity \tag{2}$$

In any registration problem, in order to estimate the transformation parameters, it is necessary to find common features or objects (In this study edge and planar points). Almost all available methods try to minimize the differences between common features. However, finding correspondence feature and treating them is not the same in different methods.

$$T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

In equations 3, T is transformation matrix, t_1 to t_3 are translation parameters and R_{11} to R_{33} are rotation parameters.

2.1 Initial estimation

A feature can be defined as a point which is being described by a cluster of points (usually neighbor points) which can describe some specifications of that position. Those specifications can be normal vectors, curvatures, etc. In this paper, in order to simplify the feature detection problem, points divided into two different categories, edge and planar. In the raw point clouds, each point has 4 dimensions such as X, Y, Z coordinates and intensity. Intensity is highly dependant on the surface texture, the incidence angle of the laser beam and the distance. The surface texture is not likely to change while data recording. However, because of using a mobile system, distance and incidence angle will change in each scan. Therefore, the intensity is not a reliable information. Finding a correlation between data is the key to feature detection. Equation 4 provide curvature value for each point by comparing the position of each point according to its neighbors.

$$c = \frac{1}{2n \cdot \|X_i\|} \sum_{j=-n, j!=i}^{n} \|X_j\| \tag{4}$$

By applying a threshold to curvature value, it is possible to divide the points into a potential edge and planar categories. It is obvious that not all of the potential feature points are useful so, we are going to use those potential edge points which have the highest curvature and those planar points which have the lowest curvature. Figure 2 shows edge points detected in a scan.

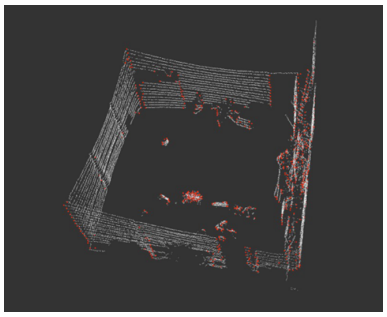

Figure 2. Detecting Edge points in semi-sparse point cloud

By assuming that we are confident that each point has a correspondent and the position of each point in 3D space will not change, using 3D to 3D transformation methods will be helpfull. In another word, we need be able to measure the same point from different positions which is almost impossible in the real world situation. 2D to 3D correspondences method used to solve the problem.
As it is obvious, planar points will mostly appear on the ground or flat (surface) shaped objects and also edge point can be found on linear shaped objects. Therefore, 2D objects can be defined as surfaces and lines and the aim is to find a line correspondence for an edge point and/or a surface correspondence for a planar point.
Rotary mobile laser scanners are scanning the surrounding environment in much more faster frequency rather than other scanners in geomatics applications. The scanning rate is usually between 5 to 20 Hz. Therefore, the displacement between

two consecutive scans should not exceed more than 1 meter if the maximum speed limits is up to 70 Km/h. This specification helps the algorithm to find the correspondences. The first step is to iterate through each interest point and try to find closest possible feature points which fulfill the neighbor selection conditions. Figure 3 shows the big picture of the procedure.

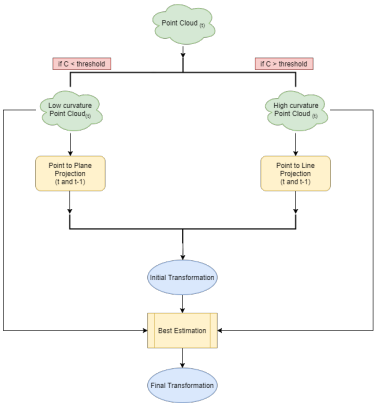

Figure 3. Flowchart of Initial motion estimator

The Velodyne Puck Laser scanner which is used in this thesis can provide 300000 points per second. Iterating through all of the points and calculate the distance in order to compare the interest point and select the closest ones is a time-consuming procedure. Therefore, using an optimal way to search for neighbor points is highly important. Kd-tree search is one of the fastest algorithms in order to find K nearest neighbors according to (W. Moore, 2004).

Edge Points:

On one hand, two points needed to create a line and on the other hand, it is desired to select the points as far as possible regarding each other. The further the points are the better the line parameters can be estimated. To satisfy both cases, it is better to select closest points which do not have the same lase ID that the interesting point has. Then, Kd-tree will provide us sorted closest potential neighbor points which are not further than a certain radius threshold and after checking the laser ID of the sorted points, 2 of them which have all criteria, will be selected to form 2d correspondent feature. For Kd-tree radius threshold, as we discussed before, displacement of the scanner is not going to be more than 1 meter in two consecutive scans however, the scanner may rotate according to each axis, the displacement can increase relatively. The further the point from the scanner, the more displacement according to the previous scan. Practically, setting 5 meters as radius threshold of Kd-tree for this sensor will provide enough inliers to form the features. Now there are three points (One interest point and two neighbor points) available for calculating the point to line distance by using equations 5 to 7.

$$P_i = \begin{bmatrix} X_i & Y_i & Z_i \end{bmatrix}^T \tag{5}$$

$$\vec{A} = P_1 - P_2 \qquad \vec{B} = P_1 - P_3 \qquad \vec{C} = P_2 - P_3 \tag{6}$$

$$D = \frac{\|\vec{A} \times \vec{B}\|}{\|\vec{C}\|} \tag{7}$$

In the above equation, P_1 is interest point, P_2 and P_3 are first closest and second closest points respectively. $\vec{A}$, $\vec{B}$ and $\vec{C}$ are 3D vectors generated by subtracting the points as it is shown. D is the distance between interest point to the correspondent line.

Planar Points:

Planar points should be projected to a surface to estimate the distance. Therefore, the minimum number of neighbor points will be three. The selection procedure of the closest neighbor planar points is the same as edge points and projecting the planar point to the correspondent surface is as follow. First, calculate the normal of the correspondent plane to estimate the parameters of the plane which is fitted to the neighbor points as described in equations 8 to 10.

$$P_i = \begin{bmatrix} X_i & Y_i & Z_i \end{bmatrix}^T \tag{8}$$

$$\vec{A} = P_3 - P_2 \qquad \vec{B} = P_3 - P_1 \tag{9}$$

$$\vec{N} = \vec{A} \times \vec{B} \tag{10}$$

In the above equation, P_1 is the point of interest, P_2 to P_4 are the neighbor points, $\vec{A}$ and $\vec{B}$ are two vectors on the plane and $\vec{N}$ is the normal of the plane. $\vec{N}$ can provide 3 coefficients out of four coefficients of the plane formula. By considering the following equation as a surface equation, and using one of the neighbors points coordinate, forth coefficient of the plane will be calculated using equations 11 and 12.

$$ax + by + cz + d = 0 \tag{11}$$

$$d = \frac{P_2^T \times \vec{N}}{\|\vec{N}\|} \tag{12}$$

By having all coefficients of the correspondent plane, calculating the difference between the interest point and correspondent plane is just a simple multiplication as described in equation 13 and 14.

$$\hat{\vec{N}} = \frac{\vec{N}}{\|\vec{N}\|} \tag{13}$$

$$D = P_1^2 \times \hat{\vec{N}} + d \tag{14}$$

Estimation:

With the correspondences of the feature points found, sensor motion will be recovered by minimizing the overall distances of the feature points. There are 6 unknown such as 3 orientation parameters and 3 translation parameters which they should be estimated from equation 15.

$$x = (A^T A)^{-1} A^T B \tag{15}$$

In the above equation, A is the coefficient matrix with the dimension of n by 6, B is the numeric term (n by one) and x is a 6 by 1 vector which is the estimated corrections of translations and rotations parameters. Since just two consecutive scans are comparing with each other, accumulation of the errors may affect the registration results rapidly, therefore, the position and orientation of the last scan should be corrected according to the other scans also.

2.2 Refinement

Considering that accumulated errors may cause a considerable drift in the final result, refining the estimated motion from the previous part will be explained in this section.

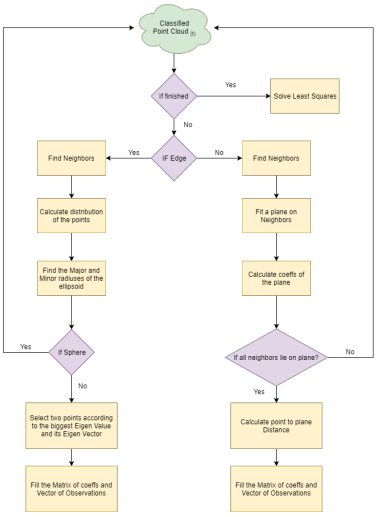

Figure 4. Flowchart of Refining algorithm

SLAM algorithm always updates the map by receiving new data and we are going to use updated maps from previous scans in order to refine the motion model of the new coming scan. The first scan will remain in its coordinate system by this assumption that there is no error in its pos (position and orientation) information. The second scan also will register according to the first one and since it is just the starting of the procedure, there is no accumulated error. The initial pos of the third scan will be estimated by the second scan and from this point, errors of the second scan will be added to the error of the motion estimation of the third scan. Figure 4 shows the steps in this section.

2.2.1 Point to Line Initial motion estimation transfer the point cloud roughly close to its true pos in the global coordinate system. And the algorithm finds closest edge neighbor points that might be correspondent with new edge point. The aim is to find the correspondent line to the selected edge point. One of the ways is to estimate the distribution of the neighbors in 3D space by calculating the covariance matrix (equation 16).

$$Q = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} x_i x_i & \frac{1}{n}\sum_{i=1}^{n} x_i y_i & \frac{1}{n}\sum_{i=1}^{n} x_i z_i \\ \frac{1}{n}\sum_{i=1}^{n} y_i x_i & \frac{1}{n}\sum_{i=1}^{n} y_i y_i & \frac{1}{n}\sum_{i=1}^{n} y_i z_i \\ \frac{1}{n}\sum_{i=1}^{n} z_i x_i & \frac{1}{n}\sum_{i=1}^{n} z_i y_i & \frac{1}{n}\sum_{i=1}^{n} z_i z_i \end{bmatrix} \quad (16)$$

Eigenvalues and eigenvectors of the covariance matrix will give us the parameters of the corresponding line of the selected edge point. Our interest is to have eigenvalues which have different magnitudes to identify the direction of the edge line in 3D space. So, if all of the eigenvalues approximately have the same values it means that either there is no edge or we couldnt find a reliable correspondent edge for the selected edge point.

2.2.2 Point to Plane It is the same approach to find the neighbors for selected planar points. However, treating the neighbors is different.

$$ax + by + cz + d = 0ax + by + cz = -d \quad (17)$$

By calculating the $\frac{df}{da}, \frac{df}{db}, \frac{df}{dc}$ from equation 17 it is possible to form the Jacobian matrix and solve it to have an estimation of plane coefficients. By applying a threshold on the residuals it is possible to make sure if the surface is reliable or not.

2.2.3 Refining the motion estimation Previous two subsections return the distance between each point and its 2D correspondent. Now by having those distances, we can minimize the distances to have a global pos estimation for each new scan by solving the coefficient matrix.

3. DATA AND TEST AREA

3.1 Hardwares and softwares

Figure 5 shows the hardware configurations of the system which is used in this study. The unit has an external battery to power up the computer and laser scanner simultaneously. ROS (Robotics Operating System) kinetic has selected as the software operating system and Intel Nuc has selected as the hardware processing platform which has intel core i7 CPU, 16GB RAM and 250GB SSD hard drive. Operating System (OS) of the computer was Ubuntu 16.04 LT.

Figure 5. Handheld laser scanner configuration

Velodyne sends its data via ethernet cable to the computer and a driver was converting data to Ros messages format (O'Quin, 2018). The driver is available on github which is supporting all models of Velodyne scanners.
On the other hand, in order to asses the accuracy, a small-scale city environment has been created in Blender software and feed to Gazebo simulator to simulate the VLP-16 data in the modeled area.

3.1.1 ROS ROS started as a project in Stanford University in mid-2000. From May 2007 it became public and others also starting to contribute to this project. In a short time, it became one of the most popular platforms for the robotics. Right now ROS consists of thousands of users all over the world.

3.2 Data collection

In order to tune the parameter of the algorithm scans were recorded in "bag" format which is a ROS format to record any type of information. Also by recording the bag file, it is also possible to play the recorded file and simulate the real-time situation.

3.2.1 Simulation A small-scale city environment was modeled in blender software in order to be used in Gazebo simulator. Gazebo can communicate with ROS by publishing the simulated data in ROS message formats. Therefore, in our case, the output of Gazebo simulator would be PointCloud2. The laser scanner sensor was mounted on top of a car with 15 degrees inclination angle in Y-axis, so it can measure higher elevation buildings. The car was guided through the path by a user which was commanding the direction and speed using simulator tools. The area of the modeled environment was 150 meters by 120 meters.

4. RESULTS

In this research, we prepared a complex environment, to check for drifts in pose estimations. Since simulation and scanned environment is not in a unique coordinate system, four reference points were selected and measured in the simulation software, in order to match the laser scanner data and simulated data. The transformation was calculated using Singular Value Decomposition (SVD) method which was implemented in PCL C++. The result of the algorithm is presented in figure 6

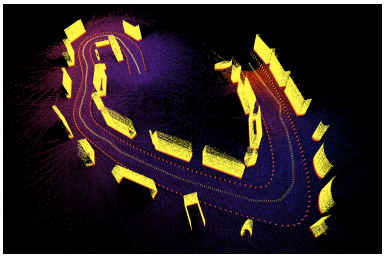

Figure 6. Point cloud from simulation, isometric view

As a comparison between the two different estimators in this algorithm, which are initial motion estimator and refinement, the result of these estimators was visualized as below.

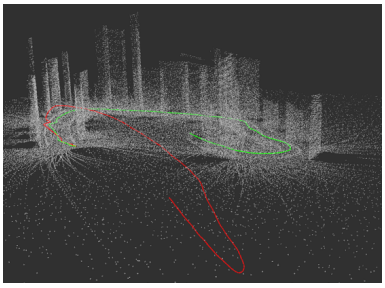

Figure 7. Results of two different estimators in the algorithm isometric view

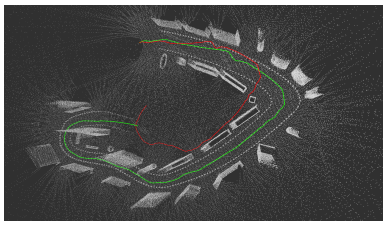

Figure 8. Results of two different estimators in the algorithm top view

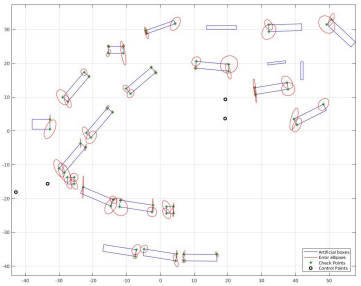

Figure 9. Error ellipses of the check points with scale factor of 50

Figure 7 and 8 visualizes the results of initial motion estimator in red and the refinements in green. As it is obvious, the red path is diverging according to the time since errors are accumulating. In contrast, the green path which is using the red path data as prior knowledge is stable enough to be used as the final pose of the sensor.

To evaluate the accuracy, 62 points which were selected all over the simulated data, also extracted from point clouds as well. PointCab software is being used to vectorize the generated map.

Table 2 shows the mean error, Standard deviations, RMS and variances of the check points which are shown in figure 9.

5. CONCLUSIONS

In this paper, a SLAM method was proposed for mobile mapping systems using a lightweight laser scanner (VLP-16) which can be used without any aiding sensor which is using 2D-3D registration method to minimize the drifting caused by the error accumulation. The performance of the algorithm was tested in a simulated outdoor environment which the sensor was rigidly mounted on top of a car which was driving on a path and it was so promising according to the accuracy of the sensor. The algorithm may fail in such cases as it cannot see the common feature from the previous map. In this case, using aiding sensors may help the algorithm to recover itself.

Type	Value (meter)
X mean Error	0.0166
Y mean Error	0.0451
X STD	0.0590
Y STD	0.0402
X RMS	0.0608
Y RMS	0.0602
X Variance m2	0.0035
Y Variance m2	0.0016

Table 2. Accuracy assessment of data

REFERENCES

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J., 2016. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(1), 1309–1332.

Holz, D., E. Ichim, A., Tombari, F., Rusu, R., Behnke, S., 2015. Registration with the Point Cloud Library - A Modular Framework for Aligning in 3-D. *IEEE Robotics Automation Magazine*, 22, 110-124.

O'Quin, J., 2018. Velodyne Driver. https://github.com/ros-drivers/velodyne.

W. Moore, A., 2004. An Intoductory Tutorial on Kd-Trees.

Younes, G., Asmar, D., Shammas, E., Zelek, J., 2017. Keyframe-based monocular SLAM: Design, survey, and future directions. *Robotics and Autonomous Systems*, 98.

Zhang, J., Singh, S., 2014. LOAM: Lidar Odometry and Mapping in Real-time.