LOAM: LiDAR Odometry and Mapping in Real Time

Aayush Dwivedi (14006), Akshay Sharma (14062), Mandeep Singh (14363)
Indian Institute of Technology Kanpur

1 Abstract

This project deals with online simultaneous localization and mapping (SLAM) problem without taking any assistance from Global Positioning System (GPS) and Inertial Measurement Unit (IMU). The main aim of this project is to perform online odometry and mapping in real time using a 2-axis lidar mounted on a robot. This involves use of two algorithms, the first of which runs at a higher frequency and uses the collected data to estimate velocity of the lidar which is fed to the second algorithm, a scan registration and mapping algorithm, to perform accurate matching of point cloud data.

2 Introduction

In this project we aim to perform accurate mapping of a given environment using a using a lidar which itself is moving in the environment. The reason behind using lidars is their high scan rates and ability to keep the error bounded irrespective of the distances. Mapping using moving lidars requires accurate knowledge of the pose of the lidar too, but to know the pose we in turn need the map. Though GPS and IMU can be used to obtain the pose of the lidar but here we aim to perform odometry using the data collected using the lidar itself. To solve this problem we use two different algorithms to achieve both odometry and accurate scan matching of the point cloud. The two algorithms run at different frequencies, but both the algorithms use points around sharp edges and planar patches as feature points to find transformations relating the poses and scans at different time steps. The odometry algorithm runs at a higher frequency and use feature points from the collected point cloud to localize the bot, but as it runs at high frequency the belief of this algorithm is low. The scan matching algorithm runs at a lower frequency which allows it to use a larger number of feature points to achieve high accuracy while making the map. Also the odometry calculations are known to accumulate error in the pose over time causing a drift, to minimize which loop closure is required. But in this project we aim to minimize this drift without taking any help of loop closure.

3 Notation

We will use following notation in our paper for clarity and to simplify expressions.
P_k : Point cloud perceived during k^{th} sweep
L : Lidar Coordinate System
$X^L_{(k,i)}$: Coordinates of point $i : i \in P_k$
t_k : Starting time of sweep k
$\overline{P}_k$: Reprojected point cloud of P_k to time stamp of t_{k+1}
ϵ_k : Set of edge points in P_k

H_k : Set of edge points in P_k
$\bar{\epsilon}_k$: ϵ_k reprojected to the beginning of k^{th} sweep
$\overline{H}_k$: H_k reprojected to the beginning of k^{th} sweep
W : World frame coordinate system
Q_k : Map after the k^{th} sweep
$\overline{Q}_{k+1}$: Point cloud, P_{k+1} reprojected in the world frame, W
T_k^W : pose transform of the LiDAR at t_{k+1} in the world frame, W

4 System Hardware

The referred paper uses a custom built 3D lidar based on a Hokuyo UTM-30LX laser scanner.
Laser Scanner Specifications:

- Field of view - 180°
- Resolution - 0.25°
- Scan Rate - 40 lines/sec

This laser scanner is connected to a motor to get the 3D point cloud.
Motor Specifications:

- Angular Speed - $180°/s$
- Sweep Rotation - $-90°$ to $90°$
- Sweep Duration - $1sec$

5 Method

5.1 Feature Point Extraction

First, feature point is extracted from the point cloud P_k.
It should be noted that the laser scanner has a resolution of 0.25°, but since the laser scanner rotates at an angular speed of $180°/s$ with a frequency of 40 Hz, the resolution in the direction perpendicular to the scan plane is $180°/40 = 4.5°$

Feature points are defined as the points in our scan that correspond to sharp edges or planar surface patches. To identify whether a point in our scan is a feature point or not, we take help from the *smoothness* of the local surface of the point. It should be noted that a planar patch is very smooth whereas an edge point forms a discontinuity as the distance of its surrounding points from the lidar changes abruptly.

More formally, in case of a planar patch, if we want to check a point for being a planar patch, then we should look for the distance of its neighbouring points in the available scan. If the distances of neighbouring points is very close to the distance of the point then the point most likely lies on a surface patch. Similarly, if the distances of neighbouring points vary very much from the point of interest then the point is considered to be an edge point.

Formulating the above statements:

$$c = \frac{1}{|S| \cdot ||X^L_{(k,i)}||} || \sum_{j \in S, j \neq i} (X^L(k,i)) - X^L(k,j)||$$

where, i is the point of interest and S is a set of neighbouring points of i.
$|S|$ represents the cardinality of S.
Now, for every point in the point cloud P_k, we calculate its smoothness value by the above stated formula. All the points are sorted in order of their $smoothness(c)$ value. All points whose smoothness is greater than a predetermined threshold (say c_{max}) are classified as edge points and all those points with smoothness lesser than a predetermined threshold (say c_{min}) are characterized as planar points.

These values of c_{max} and c_{min} are tuned by mostly hit and trial and they depend on the type of environment we are mapping and the amount of detail we need. Too high value of c_{max} causes a lack of feature points whereas too low value causes some not-so-important points to become feature points.

More number of feature points causes the algorithm to be computationally expensive, so we limit their number by saying that a point can not be a feature point if a surrounding point is already a feature point.

5.2 LiDAR Odometry

5.2.1 Finding feature point correspondences

For edge point: We will find an edge line corresponding to an edge point. For any edge point $i \in \bar{\epsilon}_{k+1}$ we look for its corresponding egde line in $\overline{P}_k$. We search for j, the nearest point of i in $\overline{P}_k$ and l in the two consecutive scans of j. Here (j, l) form the corresponding edge line to i. Now as the known transformation is not correct we find the distance of the edge point from the edge line and use this as an error estimate. The distance can be calculated using vectors as:

$$d_\epsilon = \frac{|(\overline{X}^L_{(k+1,i)} - \overline{X}^L_{(k,j)}) \times (\overline{X}^L_{(k+1,i)} - \overline{X}^L_{(k,l)})|}{|\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,l)}|}$$

For planar point: To find the planar patch corresponding to a planar point $i \in \overline{H}_{k+1}$ we will need three non collinear points from $\overline{P}_k$ Similar to edge point we find the closest point to i in $\overline{P}_k$ and call it j. Now for the other two points we look for the closest points in two consecutive scans to j and name them m and l. Selecting them from different scans ensures non collinearity among the points. Here also we find the distance between the point i alnd the plane formed by (j, m, l) and use it as an error estimate. The distance can be calculated as:

$$d_H = \frac{|(\overline{X}^L_{(k+1,i)} - \overline{X}^L_{(k+1,i)}) \cdot (\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,l)}) \times (\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,m)})|}{|(\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,l)}) \times (\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,m)})|}$$

5.2.2 Motion Estimation

We take the transform from time stamp t_{k+1} to t as $T^L_{k+1} = [t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]^T$. Now for any point $i \in P_{k+1}$ with its time stamp as t_i we can write the transform from t_{k+1} to $t_i$using linear interpolation as

$$T^L_{(k+1,i)} = \frac{t_i - t + k + 1}{t - t_{k+1}} T^L_{k+1}$$

. Now we can use Rodrigues' formula to convert rotation about the three axes into rotaion about a single resultant axes, such that the nte rotaion is $\theta = ||T^L_{(k+1,i)}(4:6)||$ and the axis of

rotation is $\omega = T^L_{(k+1,i)}(4:6)/||T^L_{(k+1,i)}(4:6)||$. Now we can use the transform to match the feature points as :

$$X^L_{(k+1,i)} = RX^L_{(k+1,i)} + T^L_{(k+1,i)}(1:3)$$

Now using these elations and the previous mentioned distance relations for each feature points we can formuate two error functions as $f_\epsilon(X^L_{(k+1,i)}, T^L_{k+1}) = d_\epsilon, i \in \epsilon_{k+1}$ and $f_H(X^L_{(k+1,i)}, T^L_{k+1}) = d_H, i \in H_{k+1}$. Combining all the equations for all the feature points we can write $f(T^L_{k+1}) = d$ and then use the Levenberg-Marquardt non linear optimization technique to minimize d:

$$\mathbf{T}^L_{k+1} \leftarrow \mathbf{T}^L_{k+1} - (\mathbf{J}^T\mathbf{J}) + \lambda diag(\mathbf{J}^T\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{d}$$

where λ is a factor determined by the Levenberg-Marquardt method.
Also motion estimation is adapted for robust fitting and the feature point which gives less information with huge error i.e. have a large distance from its corresponding feature, greater than a fixed threshold are treated as outlier. So, a bisquare weight, $(1-d_i^2)^2$ is assigned to each feature points which removes the outliers from the dataset.

5.3 LiDAR Mapping

5.3.1 Integration of T^L_{k+1} to T^W_{k+1} and projection of $\overline{P}_{k+1}$ in the world frame:

After the $(k+1)^{th}$ sweep, we have the undistorted point cloud $\overline{P}_{k+1}$ reprojected on the time stamp t_{k+2} and the pose transform T^L_{k+1} generated from the odometry algorithm during the sweep between $[t_{k+1}, t_{k+2}]$. Now the mapping algorithm runs at the end of the $(k+1)^{th}$ sweep to match and register the point cloud $\overline{P}_{k+1}$ into the world frame, W. It extents the pose transform T^L_{k+1} to T^W_{k+1} using the pose of the LiDAR, T^W_k at t_{k+1} in the world frame, W. It then projects $\overline{P}_{k+1}$ onto the world frame, W as $\overline{Q}_{k+1}$ which is then matched with map, Q_k obtained at the end of the k^{th} sweep.

5.3.2 Correspondence of the feature points of $\overline{Q}_{k+1}$ with the map, Q_k:

The point cloud is stored onto the map, Q_k in 10 m cubic areas and the points that intersect with the $\overline{Q}_{k+1}$ is extracted and stored. We now find the set of surrounding points, S' in Q_k around the feature points of $\overline{Q}_{k+1}$ and only keep those feature points that lie on the corresponding feature (edge line or the planar patch in Q_k) in set S'. Now, we can find the orientation of the features using the Principle Component Analysis (PCA) by calculating the covariance matrix, E of S'. For the set of edge points, the eigenvector corresponding to the maximum eigenvalue of E defines the direction of the edge line which is passing through the centroid of S'. Similarly, for the set of planar points, the eigenvector corresponding to the minimum eigenvalue of E gives the direction normal to the planar patch which passes through the geometric center of S'.

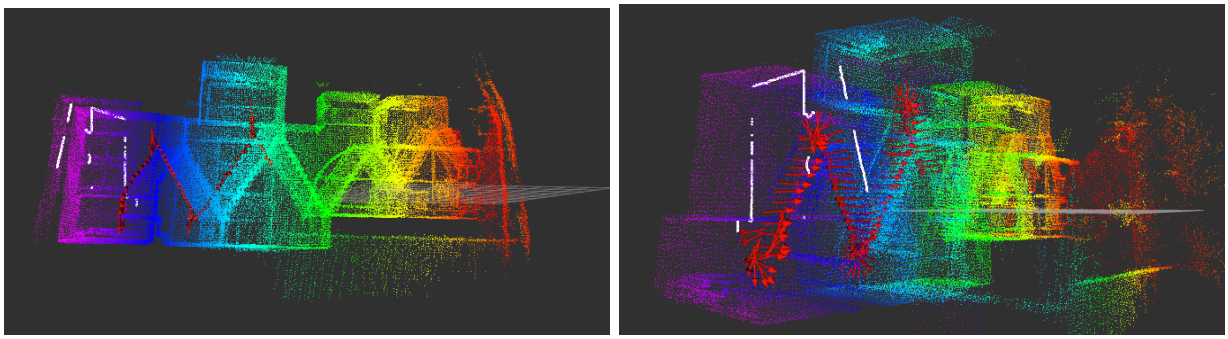
5.3.3 Optimization of the pose transform, T^W_{k+1} after $(k+1)^{th}$ sweep:

The pose transform of the $\overline{Q}_{k+1}$ in the world coordinate system is integrated using the pose of map after k^{th} sweep, T^W_k and the transform output, T^L_{k+1} from the LiDAR odometry of $(k+1)^{th}$ sweep. Now, we can optimize it using the same method as applied in the LiDAR odometry algorithm. First, the distance is computed from all the feature points to the corresponding updated features (i.e. edge lines and planar patches) which gives us the similar non-linear function, $\mathbf{f}(\mathbf{T}^W_{k+1}) = \mathbf{d}$. Then, Levenberg-Marquardt method is applied to optimize the required non-linear function which gives the final pose transform.

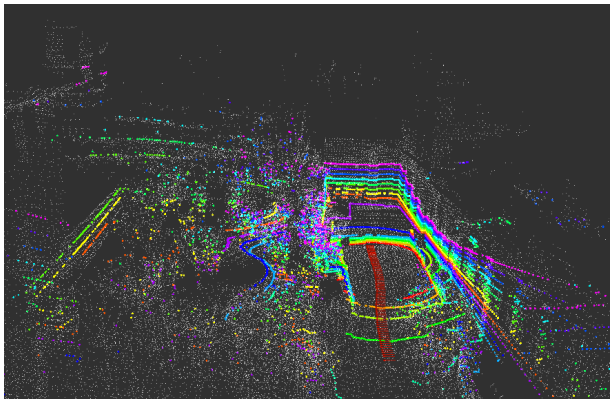
6 Results

The LOAM algorithm is implemented on the two datasets: staircase and indoor-outdoor environment on the Robot Operating System (ROS). The pose of the robot at different time stamp t_k is published in the *rviz* environment using the red coloured arrows. Also, the last scan of the LiDAR is registered with the white coloured patch in case of the staircase dataset.

6.1 For Staircase dataset:



6.2 For Indoor Outdoor dataset:



7 Conclusions and Future Work

The strategy of using two separate algorithms to perform fast and accurate SLAM seems to work properly as can be seen in the results. The overall drift in the LiDAR pose (experimental error[1] in motion estimation drift is calculated out to be 1% for 50 m in indoor environment) has also been minimized without any help from loop closure, indicating that Levenberg-Marquardt non linear optimization efficiently reduces the error. The method to extract feature points by calculating the smoothness around them accurately identifies edges and planar patches.

The method can be integrated for the camera images along with the LiDAR data for more accurate results to map and calculate the pose of the robot in the environment. Also, the loop closure can be deployed into the algorithm for more robust results. This could result in lesser motion estimation drift.

References

[1] Ji Zhang, Sanjiv Singh, *"LOAM: Lidar Odometry and Mapping in Real-time"*

[2] Open Source Code: https://github.com/daobilige-su/loam_back_and_forth

[3] Staircase dataset: http://wiki.ros.org/loam_back_and_forth

[4] Indoor Outdoor Dataset: http://wiki.ros.org/loam_velodyne