16-833 Robot Localization and Mapping

Mid Project Report - Team Slam Dunk

Akshit Gandhi (akgandhi)
Avinash Hemaeshwara Raju (ahemaesh)
Parv Parkhiya (pparkhiy)

Introduction:

Majority of the traditional SLAM problems have been addressed for static environments. Addressing SLAM problem in dynamic environments is quite limited. With the advent of self-driving cars and factory automation robots, impact of robot localization in dynamic environment has become more profound. With this project, we expect to implement the existing Dynamic SLAM approach described in [1] and improve upon this approach. (eg additional cost functions for getting better compatibility score)

Proposed approach utilizes theory originally proposed in political science journal titled Landscape Theory of Aggregation[2] of all places. While traditional SLAM algorithm assumes that all the features are static and hopes that high number of static features will counter the misleading measurements provided by dynamic features, Landscape Theory of Aggregation will allow us to classify the features into static and dynamic set and will enable explicit use of just stationary features in our SLAM pipeline resulting in the improved map and localization output. Also, the structure of the proposed approach allows for integration with any existing SLAM system to get improved results in dynamic environments.

Novelty of the paper is in classifying features in static and dynamic class. The classification is performed by finding a least squares optimization solution to set of correspondences that are to be classified as static or dynamic. Also, proposed approach requires traditional SLAM once classification is performed to get robot trajectory and map. We plan to implement either Lidar Odometry and Mapping(LOAM) or Extended Kalman Filter (EKF) or Factor Graph methods described in the class to work along with the classifier.

Literature Review:

A little background on Landscape Theory of Aggregation in terms of political science as described in paper[2].

"Aggregation means the organization of elements of a system into patterns that tend to put highly compatible elements together and less compatible elements apart. Landscape theory predicts how aggregation will lead to alignments among actors (such as nations), whose leaders are myopic in their assessments and incremental in their actions. The predicted configurations are based upon the attempts of actors to minimize their frustration based upon their pairwise propensities[1] to align with some actors and oppose others."

For implementation of Landscape Theory of Aggregation in SLAM, we see that the frustration score in a set of only static correspondences or in a set of only dynamic correspondences will be the least. We calculate a propensity score which gives us the influences of one correspondence on another correspondence. Also, this influence is a measure of the weight of the correspondence, which is analogous to small nation-big nation theory – "*a source of conflict with a small country is not as important for determining alignments as an equivalent source of conflict with a large country"*. Based on the propensity score, our classifier tries to achieve the minimum energy state by accurately classifying correspondences into static or dynamic classes. Once we have classified correspondences, we omit dynamic correspondences and perform SLAM using the static correspondences.

We minimize following cost function for classification:

$$E(G) = \frac{1}{2}\sum_{i=1}^{n}\sum_{i\neq j=1}^{n}\frac{s_j p_{ij}}{s_i}\sum_{c=1}^{m}\left(u_{ic} - u_{jc}\right)^2$$

s_i – *weight of the* i^{th} *feature point*

u_{ic} – *boolean if* i^{th} *feature is in* c^{th} *class*

p_{ij} – *propensity score between feature point i and j*

u_{jc} – *boolean if* j^{th} *feature is in* c^{th} *class*

m – *number of class*

n – *total number of features*

k – *current timestamp*

t – *window size*

l_i^k – *pose of* i^{th} *feature at* k^{th} *time in global frame*

σ_x^2 – *covariance of x*

[1] an inclination or natural tendency to behave in a particular way.

For our case, size/weight and propensity are defined as,

$$d_{A_i}^{(k)(k+a)} = \left\| l_i^{\,k} - l_i^{\,k+a} \right\|_{\Sigma} \qquad s_i = \frac{1}{\sigma^2_{d_{A_i}}} \qquad p_{ij} = 1 - \frac{1}{t}\sum_{a=1}^{t} \left\| d_{A_i}^{(k)(k+a)} - d_{A_j}^{(k)(k+a)} \right\|$$

Tasks completed so-far:

1. Rather than acquiring our own dataset we decided to move towards using the dataset provided to us in the Particle Filter assignment where we have the robot odometry and the laser scan data for 180 degrees. We have completed the work of porting the dataset from the txt file to a defined structure maintained by our software by parsing through the dataset inorder to have efficient processing of data.
2. In 2D laser scan as it is difficult to have landmark association and finding landmarks in 2D laser scan data is also difficult we moved towards using all the 180 data points at each time-step as our landmarks. In order to find correspondences between landmarks we transform them to the odom frame using the odometry data and then we use nearest neighbor search within a certain radius to establish a correspondence. Currently we have a naive O(n^2) algorithm for the same but we may move to KD-Tree based approach for faster operations later.
3. Based on the various terms defined in the paper like propensity, size, etc, we have defined classes and methods to compute the various terms as per the formulas described.
4. As our main aim is to classify any given landmark as static or dynamic, our optimizer has to produce a binary output. Currently we couldn't find any proper api inside ceres solver for this and thus now we have designed a binary optimizer which uses ceres solver along with multiple custom cost functions which ensure that the output is binary.
5. We have also written a custom visualizer which takes in a vector of points and displays them as static or dynamic based on the property of that point (which is a landmark in our case)

Current progress as compared to the timeline:

1. In the proposal we had promised to acquire the dataset, go through the paper thoroughly and understand the core idea of the landscape theory and implement it's core idea to classify the landmarks as static or dynamic.
2. We have strictly followed the timeline and we are on track as far as our proposed schedule is concerned.

Technical Challenges:

1. As we mentioned that we couldn't find any binary non-linear optimizer and as a turnaround we have written our own cost function. The optimizer works well for toy examples but when we run it on the real dataset, it is extremely slow to converge and also the results are not as expected. Hence to mitigate this challenge we might have to look deep into ceres or any other non-linear solvers (preferred cpp as our current implementation is in c++ but python implementations can also be explored)

Data collected so-far:

So far we have been using the dataset provided to us from Assignment 1. The dataset contains robot odometry data and it also has laserscan data collected as the robot moves. We also have an idea of the position of the sensor on the robot (given in the assignment write-up). Since we are doing SLAM, we **won't** be using provided map information.

Implementation:

For faster run time for debugging and deployment, we have decided to use C++ as our primary coding language for implementing the dynamic SLAM. Implemented code can be divided in 4 parts.

- **ProcessData class:**
  - Process data class provides interface with dataset which is in txt file.
  - It takes dataset filename as input, loads all the data from the text file and given a query for the timestamp and window, process data class provides all the laser data in common global frame for the requested duration in appropriate data structures.
  - Process data also perform feature association/correspondence using nearest neighbour search

- **DynamicSlam class:**
  - This class takes inputs from process data class and computes necessary compatibility score between each corresponding features
  - It also computes reliability score of each feature point
- **Classifier class:**
  - It uses non-linear least square solver called ceres solver and add various constraints as outlined in the algorithm and additional constraints to make the optimization over binary variables instead of continuous variables.
- **Helper functions:**
  - Helper functions includes plotting functions that uses opencv to visualize the laser scan with classified output of dynamic and static

Preliminary Results:

The major contribution of the proposed project is its ability to classify features into dynamic and static. We are able to obtain promising results for this critical part of our system as shown below.

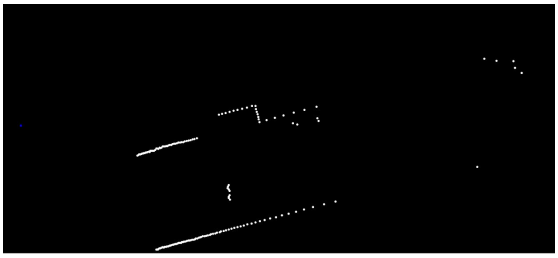

Figure 1: Input laser scan points in global frame that have found correspondence

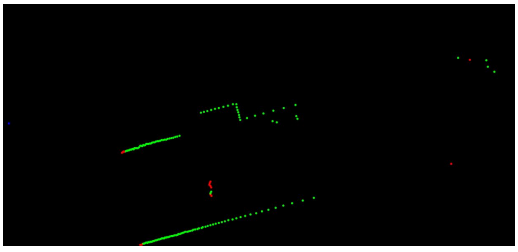

Figure 2: Initialization guess for classification (Red Dynamic, Green Static)

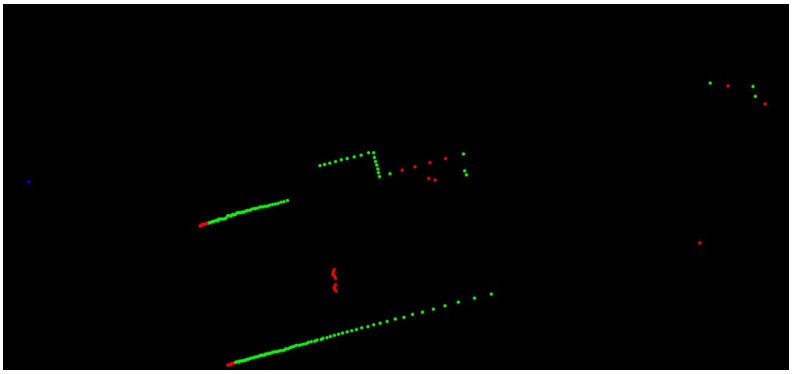

Figure 3: Final Classification output (Red Dynamic, Green Static)

Note:

- Red circular arc above corresponds to human legs which is dynamic in the scene.

- Blue dot corresponds to the origin

Timeline:

3 Nov - 10 Nov	● Improve optimization method for efficient classification
11 Nov - 18 Nov	● Integrate classifier with traditional SLAM with static features
19 Nov - end	● In depth evaluation of the implemented method ● Extensions to improve performance and accuracy

References:

[1] HUA Cheng-hao, DOU Li-hua, FANG Hao, FU Hao, A novel algorithm for SLAM in dynamic environments using landscape theory of aggregation, 2016
[2] Robert Axelrod and D. Scott Bennett, A Landscape Theory of Aggregation, 1993