ECE 276A: Project 2
Particle Filter SLAM

Chaitanya Tambat
Department of Mechanical and
Aerospace Engineering
University of California San Diego
ctambat@ucsd.edu

***Abstract*— The problem of localization and mapping of a robot has many novel uses in Modern Robotics and Control of Autonomous vehicles. In many applications we don't have the luxury of a predetermined map, but the usage of landmarks and correspondence can help to maintain a correct localization. A solution to this problem falls into the family of problems called SLAM. A utility of this method is for navigation and creation of a map for any unknown or changing space where a map is either not available or needs to be updated in real-time. This project discusses an approach to solving this Simultaneous Localization and Mapping problem for an autonomous vehicle and understanding the shortcomings of the given implementation.**

***Keywords*— Trajectory, Mapping, Localization, Particles, Particle Filter, Autonomous vehicle, LiDAR, Encoder, Fiber Optic Gyroscope (FOG), SLAM, Timestamps**

I. Introduction

SLAM stands for Simultaneous Localization and Mapping. While navigating through the environment it is essential to know the exact location of ourselves. Having this location information helps us find a way in an unknown environment and to also understand the surroundings better. Humans have learned to do this task instinctively. But how does a robot find its location in a given environment? In robotics, this problem is termed as Simultaneous Localization and Mapping (SLAM).

In robotics SLAM can be defined as the task of estimating the map of the environment while localizing the robot/sensors in the map which is being built. It is very important for mobile robots' navigation in unknown environments or partially unknown environments.

If we already have a map then we need to perform only localization. And if we already know the poses of the robot (location data) then mapping of the environment becomes easier. Solving both these problems at same time is what we call SLAM and is much harder to achieve and a popular topic of study in the field of Robotics.

While talking about SLAM, people make an analogy with the chicken-egg problem:

- To locate the robot in environment, we need an accurate map of the environment, and
- To generate the map, we need accurate position i.e., location of robot in environment

General process of slam consists of updating the position of the robot by using the inputs from the environment by extracting the features from the environment, storing them in memory and then re observing them while moving around in the environment and then optimizing the pose and map data we collected as per the observations currently being made by the sensors of the robot.

Generally, a SLAM system consists of a frontend and a backend.

Frontend takes in raw sensor data from sensors such as LiDAR, camera, IMU, etc. and transforms it into an intermediate representation such as constraints of an optimization problem.

Backend takes this intermediate representation from the frontend and does the task of solving the optimization problem or state estimation problem i.e. estimating parameters which describe the position of objects/landmarks in the environment or where the robot is in it.

II. Problem Formulation

In this project, the summary of the problem can be stated as predicting the state of the autonomous vehicle over time using the map of its surroundings that we generate along the way. This needs to be done using the data from 3 sensors, namely, a fiber optic gyroscope (FOG), a wheel encoder, and a front scanning LiDAR. The main components of the problem formulation are the following: Localization, Mapping, SLAM, Data Cleaning and Synchronization. We will take a look at each part of the problem and what the problem deals with. The main problem statement is to implement SLAM using odometry, 2-D LiDAR scans, and stereo camera measurements from an autonomous car. Using the odometry and LiDAR measurements to localize the robot and build a 2-D occupancy grid map of the environment and using the stereo to add RGB texture to the map.

A. Data Cleaning and Synchronization

Although the ultimate aim and central stage part of this problem is SLAM, the sensor data that we have is

asynchronous as well as has irrational readings due to real world unforeseen circumstances. This data needs to be synchronized and cleaned before it is used for SLAM. The SLAM steps follow next and are defined below as Mapping, Localization and Simultaneous Localization and Mapping.

B. Localization Problem

This part of the problem deals with localization of the vehicle or in other words finding the current state of the autonomous vehicle.
Localization mainly takes two inputs viz. a map and control inputs *u*.
The solution of the localization problem is a sequence of robot states *x*.
In this project, the localization output is the robot pose in the world frame coordinates.

C. Mapping Problem

Mapping deals with obtaining a map of the surroundings or in other words logging the landmarks of the robot's surroundings. Mapping problem takes two inputs, robot states *x* and the observations *z* of the sensors at each state *x*. The output of mapping is the Map of the robot surroundings.
In our case, the Map is of the form an Occupancy Grid Map.

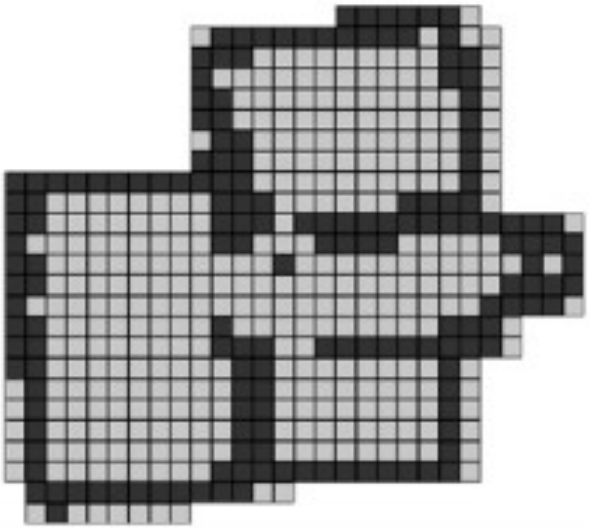


Fig 1: Occupancy Grid Map

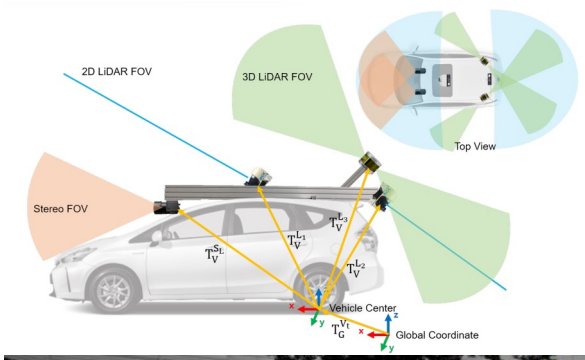
D. Simultaneous Localization and Mapping

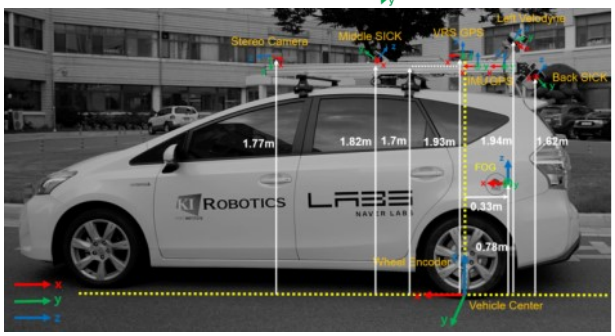
The problem of Simultaneous Localization and Mapping deals with doing step A and B i.e., Localization and Mapping simultaneously with limited data and knowledge about both.
The inputs that a SLAM problem takes are control input *u* which is an acceleration in our case and observation *z* which is the LiDAR scans in this project.

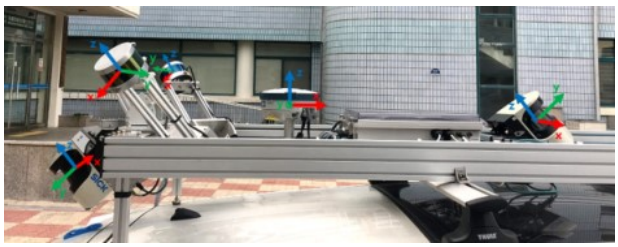
The ultimate output in our case is obtaining the vehicle location and the Map at a given time.

To summarize our problem statement, we have LiDAR readings in terms of timestamps against the lengths it reaches at each timestamp. Although, these readings are in the LiDAR frame and need to be converted to the world frame to get a map of the world. Similarly, we have encoder readings which give us the distance travelled by the vehicle between two timestamps in terms of ticks of an encoder. The third sensor viz. the Fiber Optic Gyroscope (FOG) data also needs to be processed before it is used. The sensor layout on the vehicle is given by the images below in figure 1.

The mathematical formulation of the SLAM problem can be given by coordinate formulation in terms of the world frame and a grid map. The useful mathematical formulas are the distance formula between two points and the coordinates given by L(cos Ф) and L(sin Ф) where Ф(theta) is the angle made by the line with the x-axis of the world frame.







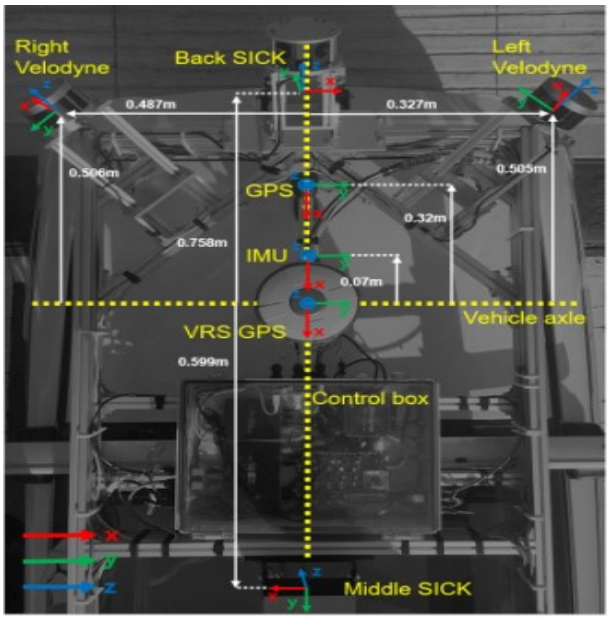


Figure 2: Sensor Layout on the Autonomous Vehicle

III. Technical Approach

A. Data Cleaning and Synchronization

Before starting with the actual SLAM problem, we need to process the available data such that we have synchronous and tractable data such that it makes our problem easier to solve. We will deal with data processing of each sensor as below.

i) Fiber Optic Gyroscope (FOG)

The data from the FOG is in w˙ rad/s which is not immediately useful to us. It's also sampled 10x faster than both the lidar and encoders. So, in this project, we matched the timestamps of the FOG with the LiDAR timestamps and summed the angle turned by the vehicle between two timestamps to get the synchronized data with the LiDAR readings.

ii) Wheel Encoders

The wheel encoders give us readings in terms of clicks of the encoder on each wheel. The wheel encoder readings were synced similarly with the LiDAR timestamps. The distance travelled by the vehicle between two timestamps was calculated using the formula below.

$$\dot{x} = \frac{\Delta \text{Ticks}\, \pi\, \text{D}_{\text{wheel}}}{\Delta\tau\, \text{TPR}}$$

Figure 3: Encoder Distance travelled formula.

In our model, the ticks per revolution is 4096 and the wheel diameters are 0.623 and 0.622 m respectively for the left and right wheel.

iii) LiDAR

LiDAR stands for Light Detection and Ranging. As the name suggests, the distance or the range is measured using a light beam, wherein, the LiDAR emits a light of known frequency and it measures the time of return of the light rays thereby giving us the distance of the object from the LiDAR.
The LiDAR is therefore used to map the surroundings given a robot pose at a given timestamp.

Once I had all sensors cleaned up and at the same sampling rate, I trimmed the data such that I had the largest data set where all sensors were sampled and times were inline. The data from all the sensors was matched to the LiDAR timestamp data thereby giving us 115865 timestamps and corresponding LiDAR, encoder and FOG readings.

This data was used further to implement our Particle Filter based SLAM project. We will see in the following sections what Particle Filter SLAM means.

B. Dead Reckoning

Although dead reckoning is not necessary for the particle filter SLAM, it's a good step to build some intuition of the data and many of the steps translate well into the SLAM predict step. The integration steps done here are the base of updating the particle states in the prediction step.

i) Motion Model

The motion model in the autonomous vehicle is a differential drive model whose motion model in mathematical form is given by:

$$\dot{X} = [v\cos(\theta)\ ,\ v\sin(\theta),\ \dot{\theta}]$$

where v is the velocity of the center of the axle and θ is the angle made by the vehicle with the x-axis of the world frame. We have plotted the vehicle trajectory and observed it as below.

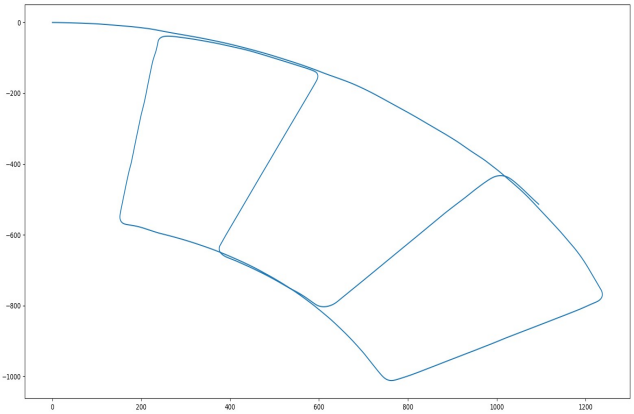


Figure 4: Vehicle Trajectory

ii) LiDAR Mapping

In the dead reckoning step, we have also used the LiDAR scans to create a map using the vehicle trajectory and the LiDAR scans.

The dead reckoning MAP is as shown below.

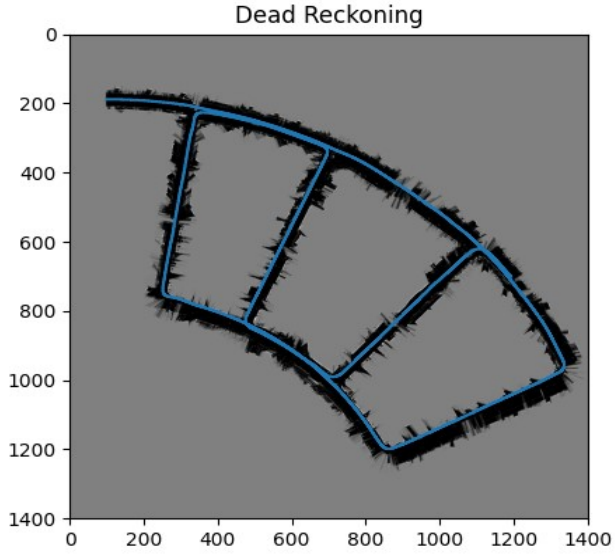


Figure 5: Dead Reckoning Map and Trajectory

C. Particle Filter SLAM

The method we will be using to solve this SLAM problem is the particle filter SLAM. Particle is a term we have used to represent our autonomous vehicle and all its possible states at each timestamp. Hence, particle filter is the method of SLAM wherein, we represent the possible locations of our vehicle using the number of particles and decide the best particle location based on how well the LiDAR map of each particle positions relates and matches with the map we already have from the previous state. The four major steps in Particle Filter SLAM are Prediction, Correlation, Updation and Resampling. They are further divided as mentioned below.

Particle filter SLAM involves the following steps:

i) Particle Initialization

First step taken is to initialize N particles to the initial state of the vehicle [0, 0, 0] with weights 1/N per particle. As we are not trying to localize the robot in a given map it does not make sense to disperse the particles initially but to instead start them on top of the robot. In our project, we have taken 5 particles due to computational limitations of the machine on which Python was used.

ii) Map Initialization

We initialize a map of all zeros and apply the first lidar scan where endpoints increase the value in the grid by 4 and open space decreases the value of the grid by 4. The sign conventions of these changes are based on applying a sigmoid to the map later to determine the probabilities that a space if occupied.

$$\gamma_{i,t} = p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \sigma(\lambda_{i,t}) = \frac{\exp(\lambda_{i,t})}{1 + \exp(\lambda_{i,t})}$$

Figure 6: Sigmoid Function

Using the following steps, we initialize the map.

a) convert the scan to cartesian coordinates

b) transform the scan from the lidar frame to the body frame and then to the world frame

c) convert the scan to cells (via bresenham2D or cv2.drawContours) and update the map log-odds

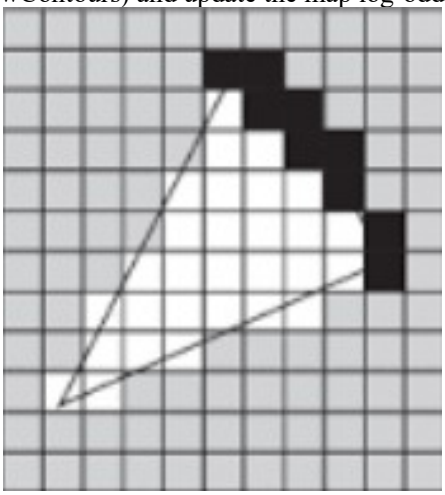


Figure 7: Lidar Occupancy Grid Map

iii) Prediction

Using the differential drive model with velocity from the encoders and the angular velocity from the gyroscope, we predict the new state of the particles. But before using the values of velocity and angular velocity we add Gaussian noise to these values to obtain variable values for each particle.

For every particle, u represents a possible robot 2-D position. In the prediction step, we compute u(t+1) for each particle k=1,….N using

$$\boldsymbol{\mu}_{t+1|t}^{(k)} = f\left(\boldsymbol{\mu}_{t|t}^{(k)}, \mathbf{u}_t + \boldsymbol{\epsilon}_t\right) \qquad \alpha_{t+1|t}^{(k)} = \alpha_{t|t}^{(k)}$$

Where f(x,u) is a differential drive motion model.

u(t) =(vt,wt) is the linear and angular velocity input.

iv) Map Correlation

We use a laser correlation model i.e., a model for a laser scan z obtained from sensor pose x in occupancy grid m based on correlation between z and m.

We transform the scan z(t+1) to the world frame using u(t+1) and find all cells y(t+1) in the grid corresponding to the scan. Scan grid correlation steps are as below:

a) Transform the scan z from the laser frame to the world frame using the robot pose x (transformation from the body frame to the world frame)
b) Find all grid coordinates y that correspond to the scan, i.e., y is a vector of grid cell indices i which are visited by the laser scan rays (e.g., using Bresenham's line rasterization algorithm)
c) Let y = r(z, x) be the transformation from a lidar scan z to grid cell indices y. Define a similarity function corr((z, x), m) between the transformed and discretized scan y and the occupancy grid m:

$$\text{corr}(\mathbf{y}, \mathbf{m}) = \sum_i \mathbb{1}\{y_i = m_i\}$$

v) Weight Update
Based on the laser correlation model, we update the weight of the particle as below

$$p_h(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}^{(k)}_{t+1|t}, \mathbf{m}) \propto \text{corr}\left(\mathbf{y}^{(k)}_{t+1}, \mathbf{m}\right)$$

vi) Weight Normalization

For the following step, we need to normalize the weights such that they sum to 1. This is done using the softmax activation method as follows:

$$W = \frac{e^{w_i}}{\sum_{j=1}^{N} e^{w_j}}$$

Where W is the vector of weights for all particles being overwritten by the softmax activation over each prior weight (Note: overwriting occurs after all new elements have been computed), and N is the number of particles.

vii) Resampling

A further aspect that has a major influence on the performance of a particle filter is the resampling step. During resampling, the particles with a low importance weight w(i) are typically replaced by samples with a high weight. On the one hand, resampling is necessary since only a finite number of particles are used. On the other hand, the resampling step can delete good samples from the sample set, causing particle depletion [19]. Accordingly, it is important to find a criterion when to perform a resampling step. Liu [11] introduced the so-called effective number of particles Neff to estimate how well the current particle set represents the true posterior. This quantity is computed as

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w^{(i)}\right)^2}.$$

The intuition behind Neff is as follows. If the samples were drawn from the true posterior, the importance weights of the samples would be equal to each other. The worse the approximation the higher the variance of the importance weights. Since Neff can be regarded as a measure of the dispersion of the importance weights, it is a useful measure to evaluate how well the particle set approximates the true posterior. Our approach follows the one proposed by Doucet to determine whether or not a resampling should be carried out. We resample each time Neff drops below a given threshold of N/2 where N is the number of particles. In our experiments we found that this approach drastically reduces the risk of replacing good particles, because the number of resampling operations is reduced and resampling operations are only performed when needed.

To summarize our approach:

a) Initialize particle set u = (0,0,0) with weights 1/N.
b) Use the first LiDAR scan to initialize the Map
c) Prediction step: Use the differential-drive model with velocity from the encoders and angular velocity from the gyroscope to predict the motion of each particle and add noise
d) Update step: combines robot state and map update. Use the laser scan from each particle to compute map correlation (via getMapCorrelation) and update the particle weights I Choose the particle with largest weight α (k) t|t , project the laser scan zt to the world frame and update the map log-odds
e) Textured map: use the RGBD images from the largest-weight particle's pose to assign colors to the occupancy grid cells

IV. RESULTS

Comparing the dead reckoned map with our plotted map, we notice that there is a flip in the plot. This is due to the error in plotting axis in the dead reckoning trajectory.

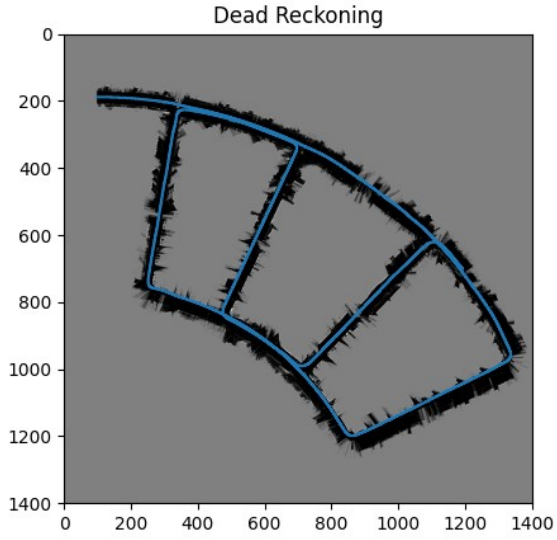


Figure 8: Dead Reckoning and Trajectory

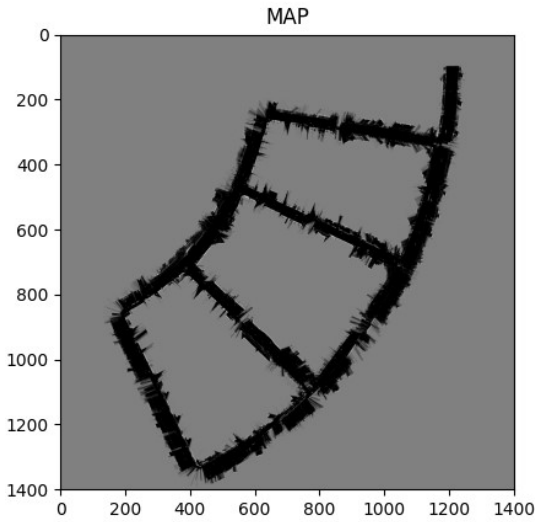


Figure 9: MAP after completion of time

As we see below. By superimposing dead reckoned map and Particle Filtered MAP, due to a manual error in specifying the axis of the dead reckoning, we observe a flip in it.

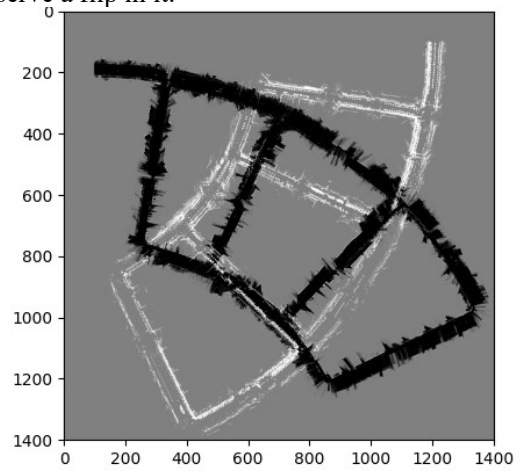


Figure 10: Superimposed Map images

Although I did not reach the texture mapping step, I referenced the Map with the given images and observed how the Map changes when the vehicle stops at a signal or how the LiDAR readings change when a big vehicle like a truck passes by and thus leads to small readings from the LiDAR.

Due to computational limitations of my Laptop, I could not plot Map images over time and the whole mapping procedure took 2 hours to complete due to these limitations.

Due to asynchronous data, some readings came out unparalleled to what was observed in the stereo images.

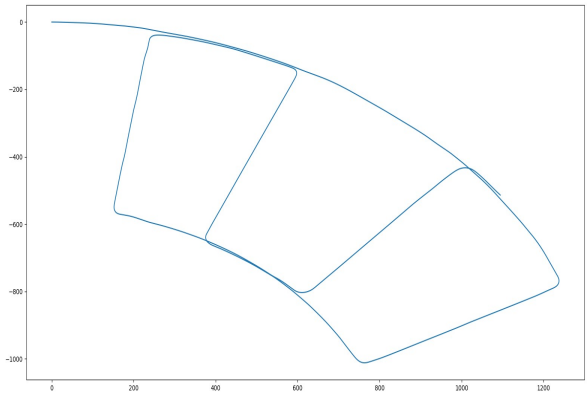


Figure 10: Trajectory

V. REFERENCES

A. Nikolay Atanasov, University of California San Diego, Lecture Notes- ECE276A, Winter 2022

B. A Doucet, On sequential simulation-based methods for bayesian f iltering. Technical report, Signal Processing Group, Departement of Engeneering, University of Cambridge, 1998.

C. Giorgio Grisetti, Cyrill Stachniss Wolfram Burgard, Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling

D. https://www.youtube.com/c/CyrillStachniss/videos