ECE276A: SLAM

Ashish Maknikar
PID: A59018297
UC San Diego

***Abstract*—A robot is made to move in an environment and data from sensors mounted on the robot is captured. The captured data represents robot motion parameters and a representation of its environment. This project aims to manipulate and operate on this data to perform simultaneous localisation and mapping of the robot as it moves.**

I. Introduction

It is easy to envisage robot motion in an unknown environment where the robot must develop its own map for motion planning and navigation. These situations are more of a norm than the exception. In such situations, the ability of the system to generate its own map, whilst simultaneously predict its location within the generated map, is essential to execute the mission/task objectives.

Our experiment involves the motion of a differential drive robot to collect data. The following sensors are used to collect data:

- Encoders measure the rotation of each of the four wheels of the robot against time.
- The IMU collects the linear and angular acceleration data of the robot.
- The Hokuyo UTM-30LX LiDAR measures depth of the obstacles on a 270 degree field of view.
- The Kinect RGBD camera collects RGB and their respective disparity images.

This data gives us sufficient information about the robot motion and its environment such that SLAM can be performed. We can surmise the pose and location of the robot from the IMU and encoder data. The LiDAR points allow us to generate the map as the robot moves.

Simultaneously generating the map while localising the robot opens up another opportunity to improve the localisation precision by leveraging the fact that the map is constant and the new sections of the map generated at each time step must be correlated with the map generated until that step. This is achieved by using the Bayes Particle Filter to localise the robot in conjunction with the map generated using LiDAR.

The final portion of the project involves creating a texture for the map using the RGBD images that are recorded. Given camera parameters and its pose with respect to the robot, we can create a RGB color point cloud of the image in the world frame. It is then trivial to fill the map with the color/texture obtained above given the location, pose and map information.

II. Problem formulation

ALSO CONTAINS PARTS OF TECHNICAL APPROACH since the problem cannot be formulated succintly without part of the technical approach

After analysing the sensor data (explained in the technical approach) at time t, we obtain the following:

- time interval τ_t between time step t and $t+1$
- robot linear velocity at time t v_t
- the robot yaw angular velocity at time t ω_t
- coordinates of the lidar endpoints in the LiDAR frame $p_{lidar}^{lidar} \in R^{3\times N_{scans}}$.
- coordinates of the texture in the camera frame $p_{rgb}^{cam} \in R^{3\times N_{pixels}}$

We know the following from the robot configuration:

- pose of the LiDAR frame w.r.t. to robot body frame T_{lidar}^{body}
- pose of the camera frame w.r.t. to robot body frame T_{cam}^{body}

A. Motion Model

We use the differential drive motion model for this step. Assuming the position starts at origin at zero angle of orientation,

$$\begin{aligned} x_0 &= [0,0] \\ \theta_0 &= 0 \\ R(\theta) &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \end{aligned}$$

where x represents the robot position on map. We can then get the position and orientation angle of the robot at $t+1$

$$x_{t+1} = x_t + v_t * [cos(\theta_t), sin(\theta_t)] * \tau_t \tag{1}$$
$$\theta_{t+1} = \theta_t + \omega_t * \tau_t \tag{2}$$

B. Mapping

We first transform the lidar endpoints to the world frame at time step t(the subscript is not written for brevity)

$$p_{lidar}^{body} = T_{lidar}^{body}\, p_{lidar}^{lidar} \tag{3}$$
$$p_{lidar}^{world} = R(\theta)\, p_{lidar}^{body} + x \tag{4}$$
$$m_{occ} = M(p_{lidar}^{world}) \tag{5}$$
$$m_{free} = Bresenham(x, m_{occ}) \tag{6}$$

Here the function M converts the Cartesian coordinates to map cell coordinates and m_{occ} represents the occupied cells on our grid map. We will further need the points m_{free} that are free accourding to the LiDAR observations along the line joining the robot position to the lidar endpoint. We can get these points by using the Bresenham algorithm.

We now have the free and occupied cells given the LiDAR observation Z_t at time t. Let us assume we know robot position x_t(This will be calculated using the Bayes Particle-Filter). We know by Bayes Rule in slides

$$\gamma_{i,t} = \frac{1}{\eta_t} p_h(z_t|m_i \in m_{occ}, x_t)\gamma_{i,t-1}$$

where $\gamma_{i,t}$ is the probability that cell i is occupied given observation. This gives us the odds ratio of the cell m_i on the map as :

$$\begin{aligned} o(m_i|z_{0:t}, x_{0:t}) &= \frac{\gamma_{i,t}}{1-\gamma_{i,t}} \\ &= \frac{p_h(z_t|m_i \in m_{occ}, x_t)}{p_h(z_t|m_i \in m_{free}, x_t)} \, \frac{\gamma_{i,t}}{(1-\gamma_{i,t})} \\ &= g_h(z_t|m_i, x_t)\, o(m_i|z_{0:t-1}, x_{0:t-1}) \end{aligned}$$

We now have a recursive function and must calculate g_h. Again employing the Bayes rule, we obtain

$$g_h(z_t|m_i, x_t) = \frac{p(m_i \in m_{occ}|z_t, x_t)}{p(m_i \in m_{free}|z_t, x_t)} \, \frac{p(m_i \in m_{occ})}{p(m_i \in m_{free})}$$

The second fraction is the prior probability which can be taken as 1 since it is equally likely that a cell is occupied or free. The first term gives signifies the accuracy of the sensor and depends on the ratio of true positives versus false positives of the sensor. We can take this value to be 4.

We take the log of $o(m_i|z_{0:t}, x_{0:t})$ to ease our calculations and find the log odds $\lambda_{i,t} = o(m_i|z_{0:t}, x_{0:t})$.

$$\begin{aligned} \lambda_{i,t} &= \lambda_{i,t-1} + \Delta\lambda_{i,t} - \lambda_{i,0} \\ \Delta\lambda_{i,t} &= log(\frac{p(m_i \in m_{occ}|z_t, x_t)}{p(m_i \in m_{free}|z_t, x_t)}) \\ &= \begin{cases} +log(4), & \text{if } m_i \in m_{occ} \\ -log(4), & \text{if } m_i \in m_{free} \end{cases} \\ \lambda_{i,0} &= \frac{p(m_i \in m_{occ})}{p(m_i \in m_{free})} \\ &= 0 \end{aligned}$$

After updating the log-odds of each cell,we can recover the map cell pmf by the sigmoid function

$$\begin{aligned} \gamma_{i,t} &= \sigma(\lambda_{i,t}) \\ &= \frac{\exp(\lambda_{i,t})}{1+\exp(\lambda_{i,t})} \end{aligned}$$

The above procedure helps us generate the map based on the lidar measurements. We however assumed we knew the location and pose of the robot. The next steps details how we must use the motion model and the lidar correlation to predict the robot position.

We are interested in the $\gamma_{i,t}$ term to threshold and plot the map.

C. Prediction

We will use the Particle filter to predict the pose of the robot. In summary the filter involved representing the pose of the robot by N particles with weights proportional to their probability.

We use the motion model to give a general direction of the motion of the robot. But we the sensors are noisy and we must model the same. We introduce Gaussian noise of standard deviation σ_v and σ_ω respectively into the linear and angular velocity measurements respectively and sample N values from this distribution. Starting at origin, we then move the N particles according to the sampled v_t and ω_t values.

At each step we must assign weights $\alpha_t[k], k \leq N$ to each particles. The weights are initially all equal.

$$\alpha_0[k] = \frac{1}{N}$$

We know that

$$\alpha_{t+1|t+1}[k] \propto p_h(z_t|m, x_k)\alpha_{t+1|t}[k]$$

The likelihood model for the LiDAR scan is proportional to the correlation between the scans world frame projection onto the map and the occupancy grid map

$$\begin{gathered} p_h(z|x, m) \propto corr(m_{occ}, m) \\ \therefore \alpha_{t+1|t+1}[k] \propto corr(m_{occ}, m)\alpha_{t+1|t}[k] \end{gathered}$$

We then choose the particle with the highest $\alpha_{t+1|t+1}[k]$ and its associate pose to map the lidar points on the grid. Thus the updated $\alpha_{t+1|t+1}[k]$ is of interest to us.

D. Texture Mapping

We can garner the coordinates in the camera frame of the rgb pixels according to the steps enunciated in the problem statement.

The coordinates of the image pixels are transformed into the world frame.

$$\begin{aligned} p_{cam}^{body} &= T_{lidar}^{cam}\, p_{cam}^{lidar} \\ p_{cam}^{world} &= R(\theta)\, p_{ca,}^{body} + x \end{aligned}$$

The rotation and translation component of T_{lidar}^{cam} are roll 0 rad, pitch 0.36 rad, and yaw 0.021 rad and (0.18, 0.005, 0.36) m respectively.

The p_{cam}^{world} are than projected onto the grid map according to the pose and trajectory recorded in the Particle Filter method above to get the texture map. This term is of interest to us.

III. Technical Approach

A. Data processing

The data from the inputs is processed as follows:

- **Encoders**: We use the encoders to obtain the linear velocity of the robot wheel. The wheel travels 0.0022m/tic and the number of tics second give us the velocity of the wheel. Averaging the velocity of all the wheels gives us the linear velocity v_t of the robot. The velocity can be seen in 1

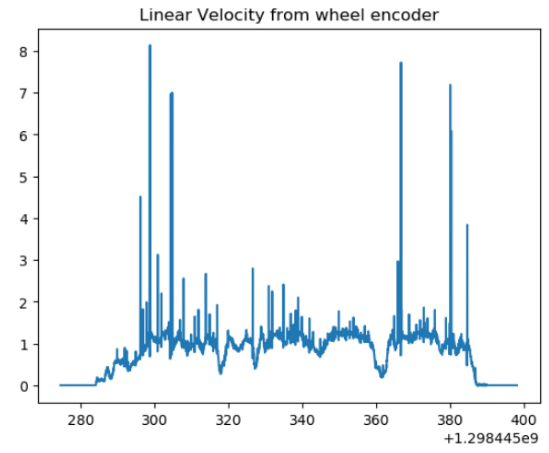

Fig. 1. Velocity derived from the wheel encoders

- **IMU**: We are given the IMU information in rad/s. We utilise only the yaw component to obtain the yaw angular velocity of the robot ω_t. The IMU data can bee seen in 2

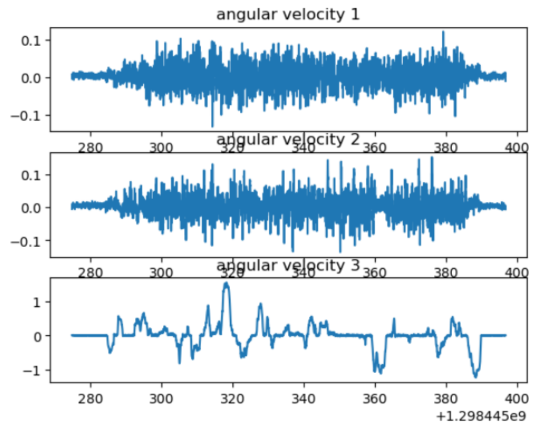

Fig. 2. Velocity derived from the wheel encoders

- **Hokuyo UTM-30LX LiDAR** The LiDAR data gives us the distances of the closest obstacle equally spaced in a 270° field of view. Given the location of the LiDAR on the robot, we can easily compute the Cartesian coordinates of these points in the robot frame. A snapshot of lidar measurement can be seen in 3.
- **Kinect**: The data provides the RGB and the disparity images. We can use them to get the coordinates of the pixels in the robot and consequently the world frame. The RGB data for a frame transformed in 3D cartesian coordinates in the camera frame can be seen in 4

The MAP we will use in all our experiments will have the following a resolution of 0.05m and range of -30m to 30m.

B. Dead Reckoning

To begin, we will first move the robot as a single entity and use the angular and linear velocity measurements without any

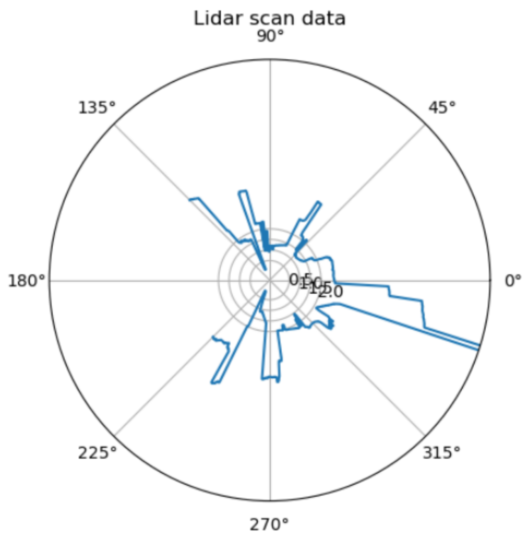

Fig. 3. Lidar Snapshot

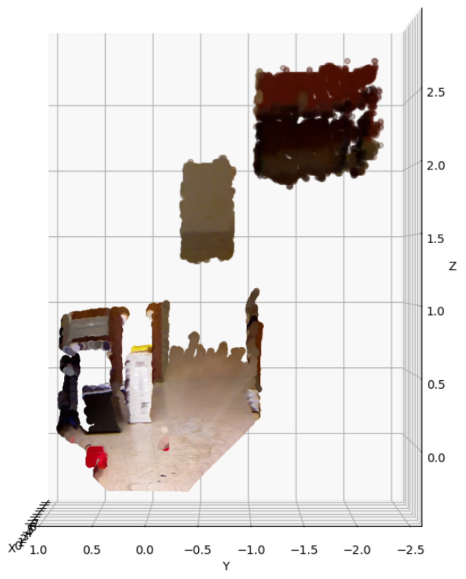

Fig. 4. Lidar Snapshot

noise. This ensures that our code will work properly in the particle filter. The robot orientation and position is updated at each time step according to 1. These calculated values are used to convert the LiDAR endpoint to the world frame according to 3.

These points are then converted to their cell values on the map and the log odds are updated according to **??** by increasing by $+log(4)$ on occupied cells and decreasing by $log(4)$ for the free cells observed in lidar measurement and calculated through Bresenham algorithm.

After the robot completes its traversal, the map is converted in probabilies using the sigmoid **??** function. A threshold of

0.2 is applied to get the final map. 5 and 6 are the dead reckoning trajectories for the two experiments in the problem statement.

C. *Particle Filter SLAM*

We initialise N particles to origin. The angular and linear velocities are converted into normal distributions of 0.1 and 0.01 respectively. We sample N values from these distributions and apply them to the motion model in 1. Each particle is assigned a weight of 1/N initially.

For each particle, the lidar points are projected onto the map and correlation is calculated. The correlation is calculated by passing a grid of (4,4) with a step size of the map resolution around the particle location and the correlation is calculated at each of these points.The maximum correlation among these is selected and the particle moved to the location on the grid with the maximum correlation(not done in our experiment since it results in a fuzzy map). The correlation is multiplied with the weights and the one with the maximum weight is chosen as the position of the robot. The lidar projection on the map at this point is chosen and plotted on the map.

There might be situations where the weights fall too low for some points. in such cases we check the following condition

$$N_{eff} = \frac{1}{\sum \alpha_i^2} < \frac{N}{10}$$

If this condition is satisfied we resample the points with α_i as weights.

D. *Texture Mapping*

We first generate the rgb Cartesian coordinates in the camera frame. After transforming them into the world frame, we threshold all values with z <wheel_radius. These values correspond to the floor texture. They x and y coordinates are converted to the map grid cell coordinates and colored accordingly to get the texture of the map. We then apply the occupancy map mask to remove all colored pixels outside the free space cells in the grid.

IV. Results

A. *Dead Reckoning*

Dead Reckoning has very few parameters. 5 and 6 show the trajectory and map generated by dead reckoning on Dataset 20 and 21 respectively. 7 and 8 plot the change in the orientation angle of the bot wrt time.

B. *Particle Filter SLAM and Texture mapping*

In our first experiment we apply particle filter slam with N=5 to the two datasets without changing the location of the particle from the correlation matrix. The texture maps are then projected onto the map(Maps: 9 and 10; Trajectories: 13 and 14; Texture Map: 11 and 12)

I attempted to change the location of the particle to the position with the higher correlation, but it resulted in more drift and fuzzier image as shown in 15

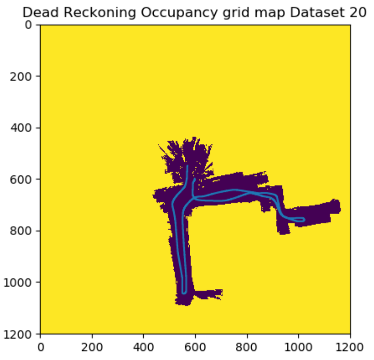

Fig. 5. Dead Reckoning for Dataset 20

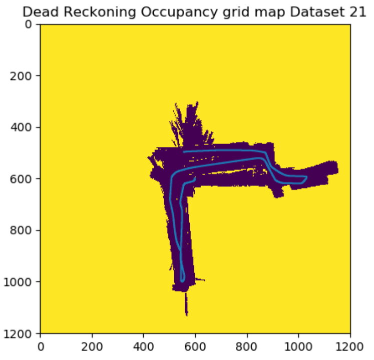

Fig. 6. Dead Reckoning for Dataset 21

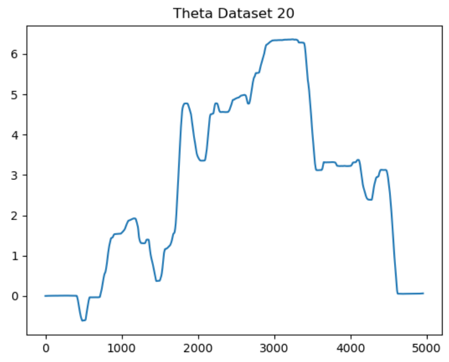

Fig. 7. Dead Reckoning Thetas for Dataset 20

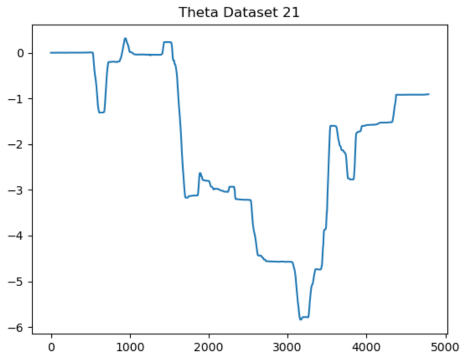

Fig. 8. Dead Reckoning Thetas for Dataset 21

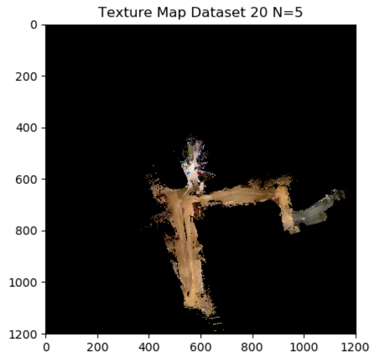

Fig. 11. Texture Map for N=5 Dataset 20

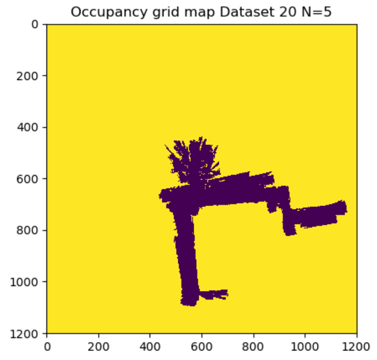

Fig. 9. Occupancy frid for N=5 Dataset 20

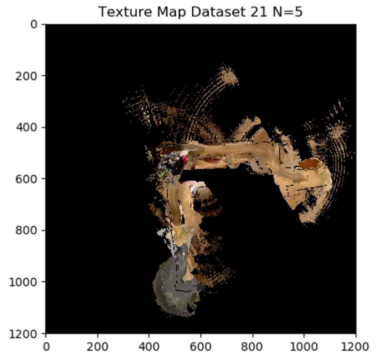

Fig. 12. Texture Map for N=5 Dataset 21

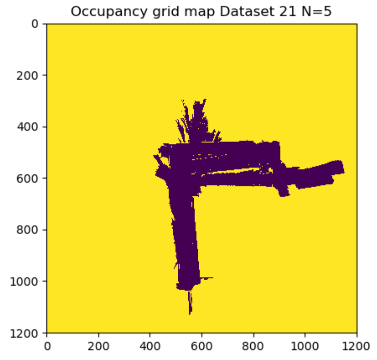

Fig. 10. Occupancy Grid N=5 Dataset 21

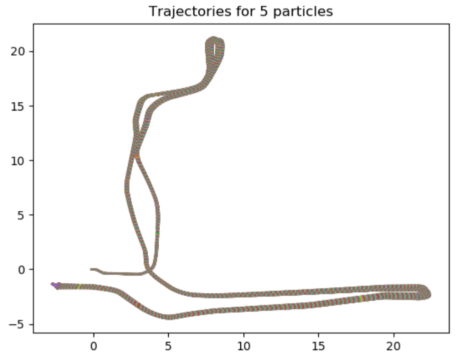

Fig. 13. Trajectories for N=5 Dataset 20

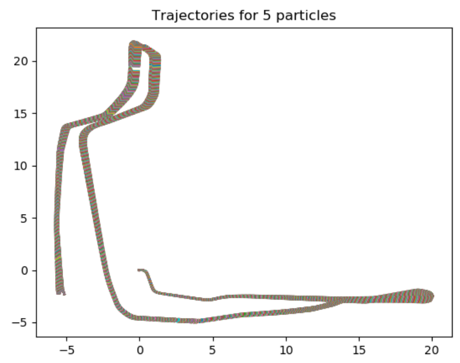

Fig. 14. Trajectores for N=5 Dataset 21

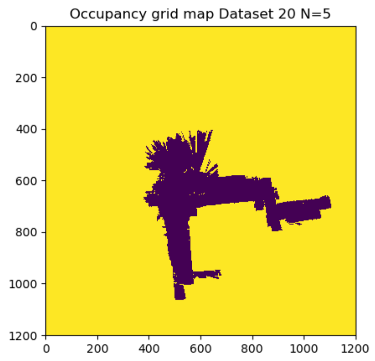

Fig. 15. Trajectores for N=5 Dataset 20 with correlation drift

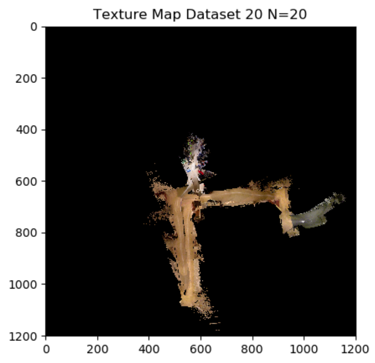

Fig. 16. Texture Map for N=20 Dataset 20 without correlation drift

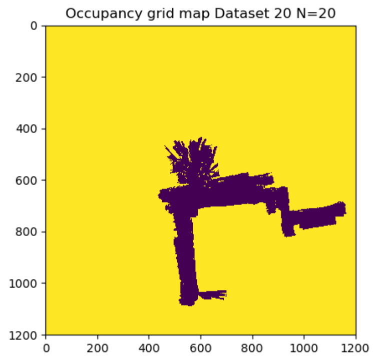

Fig. 17. Occupancy Map for N=5 Dataset 20 with correlation drift

I also experimented with N=20 particles. The texture map for Dataset 20 improved only marginally with no substantial change 16 and 17.

In conclusion, changing position according to correlation map did not work and resulted in a poorer map and image. this might be because the robot was initially static and results in a cloud which results in a random motion near the cloud. Large corridors do not give good textures as we can see for Dataset 21.