ESE650 - Project 3
Localization and Mapping

Arunkumar Byravan
Department of Mechanical Engineering
University of Pennsylvania

I. Introduction

The aim of the project was to map an unkown environment while at the same time keeping track of the robot's location within that map.This is known as the Simultaneous Localization and Mapping problem. In order to do so, the robot was equipped with a number of sensors, namely

- Wheel encoders to measure translation
- IMU for rotation rates and acceleration
- LIDAR for range information
- Kinect for depth information & images

A rendering of the plafrom is shown in fig 1.The implementation was done in two steps. Initially, the orientation information from the IMU was integrated with range information from the LIDAR to produce a 2D occupancy grid map of the walls and obstacles in the environment. In the second step, additional depth and camera imagery from the Kinect was used to find the "ground" plane and augment the map with the color of the points belonging to the ground plane.

II. 2D occupancy grid

An occupancy grid map can be used to map the environment into an array of cells, where each cell holds the probability of it being an obstacle. Such maps are useful to represent static environments making use of data from a wide range of sensors similar to the ones used here. The various steps in generating the occupancy grid are listed below. A detailed explanation of each step follows.

- Estimate the robot's orientation
  - Obtain Rotation rate from gyros and acceleration from accelerometers
  - Run them through an Unscented Kalman Filter
- Estimate the 2D pose of the robot
  - Use the encoders & gyro yaw rate as a motion model
  - Do scan matching using LIDAR data to obtain 2D pose estimate
- Populate the map based on current 2D pose & orientation
  - Keep track of the robot's pose

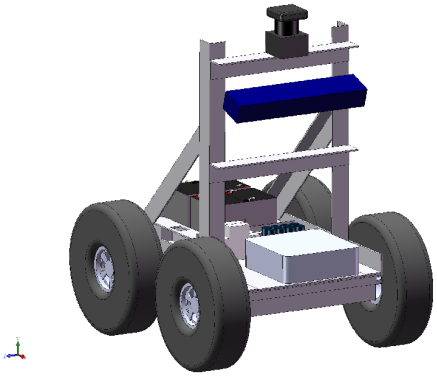

Fig. 1. Project platform

A. Unscented Kalman Filter

The Kalman Filter was implemented based on the paper A Quaternion-based Unscented Kalman Filter for Orientation Tracking by Edgar Kraft. To track the orientation of the object, a unit quaternion representing the orientation of the body frame with respect to world frame was chosen as the state vector. An Unscented Kalman Filter(UKF) was used due to the non-linearities that arise due to the use of unit quaternions as a state representation. The rate gyro data was used for the process updates while the Accelerometer data was used for the measurement Updates. The output of the UKF was then processed to obtain the robot's orientation with respect to the world frame in terms of Euler angles (Yaw/Pitch/Roll).

B. 2D pose estimation

There are three main parts to estimating the 2D pose of the robot, namely the motion model for the robot, processing the laser scans & scan matching.

1) Motion Model: To obtain a simple & good apriori estimate of the robot's motion in each timestep, the robot can be approximated to be a two wheeled differential drive robot. By counting the tics of the encoder in each timestep and by knowing the distance travelled by the robot for each tic, we can estimate the translation of the robot. The angular velocity can be estimated from the instantaneous yaw rate of the robot. This can be integrated to obtain the change in yaw angle during that timestep. Combining these with the pitch and roll estimates from the UKF, we can obtain a good apriori estimate of the robot's motion in the world frame.

This is used as an intializer for the scan matching operation to limit the search space and improve processing speed.

2) Processing the LIDAR scans: The LIDAR gives the distance to different obstacles in a planar slice of the world. Further processing has to be done to obtain meaningful information from these scans. Primarily, the laser scan includes two antennae which are fixed onto the robot. These have to be removed. Also, scans with ranges greater than 40 meters can be removed as they are more than the "trusted" range of the LIDAR. Finally, due to the presence of ramps in the environments, where the robot can pitc up/down the LIDAR occasionally sees the ceilings and ground. These should not be marked as obstacles as they will result in erroneous maps. To remove such points, the following process is used:

- Transform the scans from the laser frame to the world frame using the robot's pitch and roll
- Assume a box centered around the LIDAR
- Discard scan points outside this box

Once all these operations have been completed, scan matching can now be done on the remaining points to recover the 2D pose of the robot.

3) Scan Matching: Scan matching is the process by which the correlation between two laser scans from successive steps is computed to determine the robot's motion in between the two timesteps. The steps involved are:

- Transform the scan from the LIDAR frame to the body frame
- Choose a search space in yaw, x and y positions
- Rotate the body frame to the world frame using pitch,roll from UKF and yaw from search space
- Translate from the robot's centre to world centre using estimates from the Encoder
- Compute correlation between this scan and the existing map
- Repeat this over whole search space in yaw,x & y
- Pose with the maximum correlation is the new 2D pose

Fig 2 shows an example of the correlation between two laser scans. In order to improve the scan matching, a few extras were added:

- A gaussian prior centered around the encoder position estimate with variance equal to the window size was scaled and added to the correlation. This would improve matching in hallways
- A two stage scan matching was implemented with a coarse estimate and finer estimates

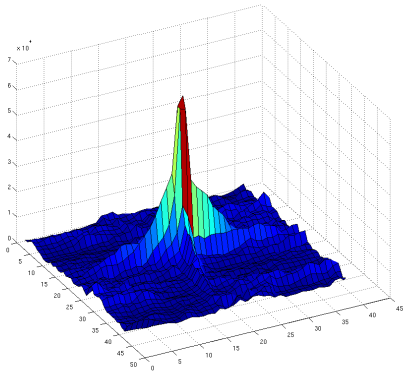

Fig. 2. AN exmaple of correlation

C. Populating the Map

Once scan matching has been done, the map can now be populated with the points declared as obstacles. First, the laser scans are transformed to the world frame using the new estimates from scan matching. Then, the X and Y values are discretized and the value in the map at those points is incremented by "1". This value is bounded to a maximum of 100 meaning that if a cell has a value of 100, it is almost certain to be an obstacle while a cell with a value of "0" is assumed to be free. Once this is done, the state of the robot is stored and the above operations are repeated over time to generate the 2D occupancy grid along with the robot's pose.

Implementation specific details:

- Map
  - Resolution : 5 cm
  - Size : -30m to 30 m in X & Y
  - Obstacle value : 100
- Scan Matching
  - Coarse:
    - X&Y : -0.1:0.05:0.1
    - Yaw(degrees) : current3: 1 :current+3
  - Fine:
    - X&Y : -0.05:0.025:0.05
    - Yaw(degrees) : current1: 0.25 :current+1

D. Results and Discussion

Using the procedure detailed above, results on a number of dataests were obtained and are shown below. Fig 3 shows the reults for dataset 20, which is on flat terrain. Fig 4 and 5 show results for datasets 22 and 22 both of which have ramps. The results show that the setup is able to consistently generate good maps even in the presence of changes in elevation in the system. There is also almost no double walls which indicate that the robot is able to localize its position pretty accurately. The use of two levels for detecting the yaw with an accuracy of 0.25 degrees really improved the results. Also, the addition of the gaussian prior made it more robust in long hallways

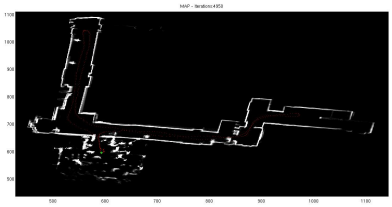

Fig. 3. Results for Dataset 20

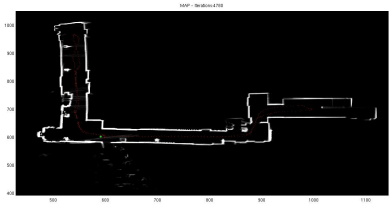

Fig. 4. Results for Dataset 21

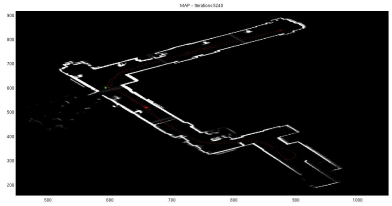

Fig. 5. Results for Dataset 22

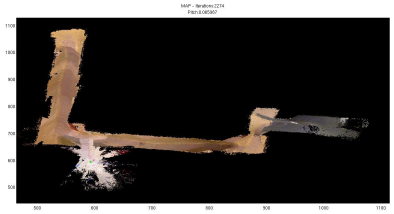

Fig. 6. Floor map for Dataset 20

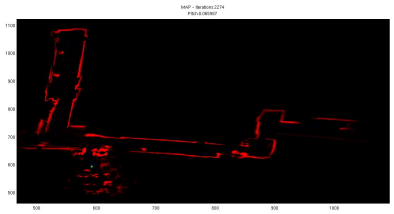

Fig. 7. Obstacle map for Dataset 20

where the robot had good match in only one direction. IN terms of speed, the system is able to run at near real time speeds (40 Hz).

III. SLAM WITH KINECT

The second part was to integrate the data from Kinect along with the 2D occupancy grids generated from the first part to produce a map with the ground colored according to the color from the images. The Kinect sensor has a camera and an IR projector/receiver which gives us RGB images and disparity images respectively. Once the setup is calibrated, once can obtain depth from the disparity images by means of simple calculations. USing the depth and the corresponding X & Y points, one can obtain a 3D point cloud from a single image. There are many ways to use this point cloud such as using an Iterative Closest Point algorithm for two successive point clouds to figure out the motion between the two, using RANSAC to find planes in the world etc. To augment the 2D map with color, two approaches were considered and implemented. The details are laid out below.

A. A naive approach - Using range of "Z" values

To augment the map with the color of points belonging to the ground , one must first figure out which of the points refer to the ground plane and which are obstacles. A simple way of doing this would be to look at the height of the points (Z values). One could simply threshold the value to say that points below a certain height belong to ground and others are obstacles. Another way would be to first discretize the X & Y values into a grid, and then infer based on the range of "Z" values for a single cell in that grid. The latter is implemented here.

- Initially, the image is subsampled to reduce the number of points
- The disparity image is then converted to a point cloud
- Points at a distance greater than 3 meters are discarded
- For each pixel in the disparity image, the corresponding pixel in the RGB image is found
- The Point cloud is transformed to the world frame
- The point cloud is discretized into cells of 5 cm sides
- Each cell is deemed to be an obstacle or not by:
  - RANGE(Z) = max(Z) - min(Z)
  - if(RANGE ¿ 0.2) then OBSTACLE
  - if(RANGE ¡ 0.01) then FLOOR
- The obstacle cells were used to form an obstacle map
- The color of the floor cells was used to create a "floor" map

1) Results and Discussion: Using this simple approach, two maps were generated for each dataset, one an obstacle map and another a map of the floor. Figures 6 - 11 show these maps for datasets 20,21 and 22. The disadvantage of this method is that if a grid has just a single point, it will be considered as belonging to the floor as its RANGE is zero. But the system is able to generate good maps of obstacles as most obstacles usually tend to have a wide range in height over a single cell. COnsequently, if a ramp is steep, it will end up being marked as an obstacle. In terms of speed, this method is quite fast (5-8 Hz) if the image can be subsampled.

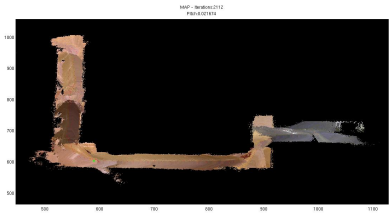

Fig. 8. Floor map for Dataset 21

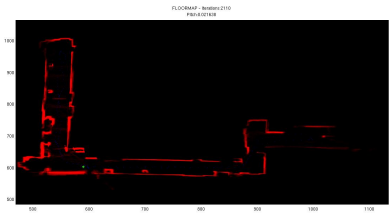

Fig. 9. Obstacle map for Dataset 21

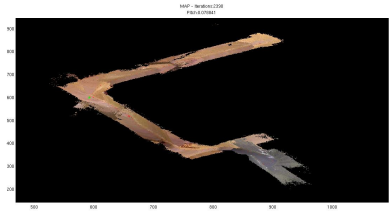

Fig. 10. Floor map for Dataset 22

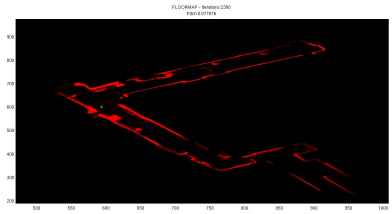

Fig. 11. Obstacle map for Dataset 22

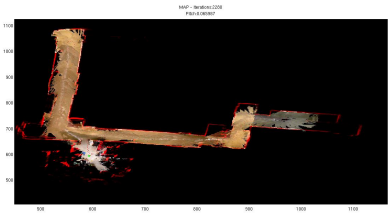

Fig. 12. Map for Dataset 20

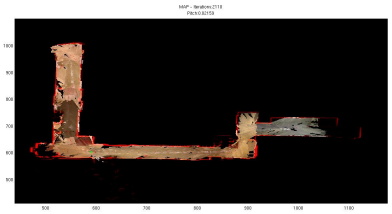

Fig. 13. Map for Dataset 21

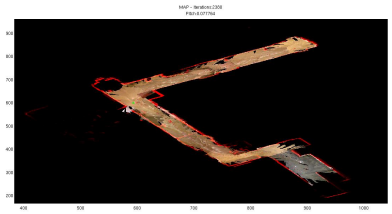

Fig. 14. Map for Dataset 22

B. A better method - RANSAC

A more robust method to detect points belonging to the ground would be to fit a plane to the points in the point cloud. One could then take the inliers for that plane(a ground plane) and use the colour of those points to project onto the ground. THis approach is detailed below:

- Initially, the disparity is converted into depth
- Points with depth greater than 3 m are discarded
- Points which correspond to pixels in the bottom 1/4th of the image are chosen as input to RANSAC
  - The bottom 1/4th is chosen as it was found to contain mostly ground points and it improved speed
  - The RANSAC was done in the image frame woth points of form [depth,Ypix,Zpix]
  - RANSAC in the image frame produces better results than on points in world
- The plane from RANSAC is used to compute inliers from all image points
- The inliers are transformed to the world frame & discretized
- The colors corresponding to inlier points are augmented in the 2D map along with the obstacle map from scan matching

This method was found to be more robust than the previous one to detecting ground points and the map generated could be subsequently used for planning. A few other tricks to the implementation are:

- The cost of the points corresponding to the inliers was reduced in the obstacle map from scan matching
- The cost of the points correspinding to outliers in the bottom 1/4th of image was increased

1) Results and Discussion: Results from the above implementation are shown in figures 12,13 & 14. In these results, the obstacle map from the LIDAR data is shown in Red while the Kinect points are colored. The results indicate that this method is more robust than the previous system. As we are fitting a plane to points rather than just using the elevation of the points, this method should be able to detect ramps more

accurately than the previous method. Also, the edges of the ramps can also be properly detected as obstacles using this method. In terms of speeds, this system is pretty slow(2-3 Hz) due to a number of costly operations (RANSAC, figuring out the colors).

IV. Conclusion

A system for Simultaneous Localization and Mapping making use of IMU, LIDAR, Encoders and Kinect was conceived and implemented. The system was tested out on varying terrain and different speeds and lighting conditions and was found to perform well consistently and close to real time in some cases. Over the course of the project, the problem of sensing on robots was tackled and a favourable solution was found. This paves the way for us to advance to the next level of planning and acting based on the tools that we have now developed.