Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental requirement for mobile robots to operate autonomously in unstructured environments. SLAM addresses the following question:

“Is it possible for an autonomous vehicle placed in an unknown environment to incrementally build a consistent map of this environment while simultaneously computing its location based on this map?”

Using this technique, a robot can operate in an environment without a priori knowledge of a map and its location in the world. This is a critical skill that can be utilized in a wide range of applications that require a robot to be able to explore a region and return to its starting location, learn different paths to key locations in a world, and to traverse a region completely. Such applications include: search and rescue operations, mine exploration, underwater navigation, crime scene investigation, playing robot soccer etc.

Project Objectives

This project is divided into two main parts. The first part involves 2D mapping and localization of several indoor environments using odometry information as well as data from an IMU and a 2D laser range scanner (LIDAR). In the first section, the wheel encoders are used to estimate the robot’s translation in the world while the IMU data is used to determine the robot’s orientation. A particle filter-based approach to SLAM was taken in this project to provide better estimates of the robot’s pose and to better align the map via scan matching. An occupancy grid mapping technique which uses the best particle pose and which incorporates the Bresenham ray tracing algorithm was used to update the map of the world at each time step.

The second part of the project consists of augmenting the 2D map with camera and depth imagery from an RGB-D sensor (the Kinect). In this section, plane fitting was performed on the Kinect’s disparity image using the RANSAC algorithm. The RGB data corresponding to the estimated ground plane was then used to color the ground in the occupancy grid map. Details and analysis of the techniques and algorithms used in this project are presented in this report.
The resulting 2D occupancy grid map of the walls and obstacles in the environment which was further augmented with the color of the ground plane as well as visualizations of the robot moving within the 2D map prove that the algorithms implemented were effective in successfully performing SLAM.

Data and Platform Used in the SLAM Project

Fig 1. Robot used for SLAM
The 4-wheeled UGV pictured above was used in this project. We were provided with time-stamped range data from the onboard Hokuyo UTM-30LX LIDAR, odometry information from the 4 wheel encoders, RGB-D data from the onboard Kinect sensor, and raw data from the robot’s IMU unit consisting of a 3-axial accelerometer and a 3-axial rate gyro.

PART I: Probabilistic SLAM using LIDAR, IMU and Odometry Measurements

Wheel Odometry for a Differential Drive Robot

Before delving into the details of SLAM, an overview of the odometry for a differential robot will be given here. The robot’s pose (its position and orientation) at time t is represented by the following state vector:

\[ x_t = (x, y, \theta)^T \] (1)
The robot's pose at time $t$ is computed using the following equations:

$$\theta = \frac{\Delta R - \Delta L}{d} \quad (2)$$

$$\Delta S = \frac{\Delta R + \Delta L}{2} \quad (3)$$

$$x_t = x_{t-1} + \Delta S \cos \theta \quad (4)$$

$$y_t = y_{t-1} + \Delta S \cos \theta \quad (5)$$

$$\theta_t = \theta_{t-1} + \Delta \theta \quad (6)$$

where $R$ and $L$ are the right and left wheel encoder counts respectively, $d$ is the distance between the left and right wheels, and $\theta$ is yaw (the robot's heading). For this project, changes in yaw were estimated using the IMU while $x$ and $y$ were estimated using the wheel encoder measurements.

**Formulation and Structure of the SLAM Problem**

The goal of SLAM is to build a map of an environment while simultaneously estimating the robot's location in this map. In a probabilistic (Bayesian) framework, SLAM is modeled by the following probability distribution which is computed for all times $t$ (in practice, computations are done recursively):

$$P(x_t, m | z_{1:t}, u_{1:t})$$

This probability distribution describes the joint posterior density of selected features in the world, $m$ (e.g. walls/obstacles) and the state of the robot at time $t$, $x_t$, given observations, $z_{1:t}$ (laser scan/RGB-D measurements in this case) and controls, $u_{1:t}$ (odometry/IMU measurements).
In general, there are two main phases of SLAM:

1. The Prediction Phase which addresses the Localization Problem via a Motion Model
2. The Measurement Update Phase which addresses the Mapping Problem

In this project, SLAM was implemented using particle filters. Particle filters model the probability distribution as a set of discrete particles which occupy the state space. Each particle represents a different pose hypothesis and has an associated weight. Compared with an EKF-based SLAM which uses unimodal Gaussians to model a non-Gaussian probability density function, particle filters are more suitable in this case, as they allow measurement updates to be done more efficiently and are able to capture the multimodality of the probability distribution in the motion model.

**The Prediction Phase: Solving the Localization Problem**

In the prediction phase of the Particle Filter-based SLAM, a map of the environment is known a priori and the objective is to determine the 2D pose of each particle, $x_t$, based on the most recent controls ($u_t$) and current laser scan measurements, $z_t$.
The robot’s knowledge about its state is reflected in its belief, $P(x_t)$ which is given by:

$$P(x_t) = P(x_t | z_{1:t}, u_{1:t}) \quad (7)$$

The particles are samples drawn from $P(x_t)$ as illustrated in figure 3. The set of M particles $X_t$ are defined by:

$$X_t: x_t[1], x_t[2], ..., x_t[M] \quad (8)$$

Fig 3. Particle representation of a PDF [1].
The Basic Particle Filtering Algorithm is as follows:

Generate Particles

- Each particle is generated by adding a zero-mean Gaussian noise to the pose at time $t$, $x_t$, such that sigma is proportional to each component ($x$, $y$, $\theta$) of the pose:

$$x_i[m] = x_i + N(0, \sigma)$$ (9)

**Make Measurements**: generate laser scans for each particle

Update each Particle’s Weight

- Use a correlation function to compare each particle’s measurement predictions to the actual map measurements
- Assign a weight to each particle such that particles with good predictions have higher weights

**Normalize the weights**: the weight of all the particles should sum to 1.

Resample (if necessary):
- If the Effective Sample Size (ESS) falls below a particular threshold (e.g. 70%), generate a new set of Particles

**Computing Particle Weights with Map Correlation**

A weight or importance factor is assigned to each particle which is used in the critical resampling step. This is computed as follows:

$$w_t = p(z_t | x_t)$$  \hspace{1cm} (10)

According to equation 10, each weight is equivalent to the probability that a measurement is observed given the particle’s current pose. In this project, an occupancy grid map is used to model the physical environment. The map correlation function compares the laser scans generated by each particle to the ones that currently occupy the grid cells. An occupational likelihood value is computed for each particle by taking the average probability of occupancy by the cells in which the laser scans hit.

**Resampling**

Resampling helps to improve the performance of the particle filter by ensuring that only the best particles (particles with highest weights) propagate forward at each time step. During resampling, particles are randomly drawn from the particle set such that the highest-weighting particles are duplicated while particles with small (near-zero) weights are eliminated from the particle set. Resampling is executed only when the Effective Sampling Size (ESS) falls below 70%
The ESS for all M particles is computed by:

$$\text{ESS} = \left(\sum_m w[m] \right)^2 / \left(\sum_m w[m]^2 \right) \quad (11)$$

The figure below illustrates the effectiveness of particle filters in computing pose estimates. In the first 3 figures, the robot is turning the first corner into the corridor. The map misalignments (indicated by the red arrow in the first 3 figures) and pose estimate errors are fairly large when only 3 particles are used to compute the robot’s pose in this case. As the number of particles increases from 3 to 100 (in the 4th figure), the pose estimates are greatly improved which is evident in the map alignment.
Simultaneous Localization and Map Construction of Indoor Environments by a Mobile UGV using a LIDAR, IMU, Kinect Sensors

The Measurement Update Phase: Solving the Mapping Problem

In the measurement update phase, the best particle (particle with the best pose estimate) is used to update the occupancy grid map. The occupancy map is discretized into cells and each cell represents the likelihood of being occupied. The occupational likelihood of each cell, \( p(m | z_{1:t}, x_{1:t}) \) is computed using the Log Odds Ratio:

\[
p(m) = \log \left( \frac{p(m | z_{1:t}, x_{1:t})}{1 - p(m | z_{1:t}, x_{1:t})} \right)
\]  

(12)

The Bresenham Line Algorithm is also used to update the occupational likelihood of each cell. Since the laser scanner measures the distance from the sensor to an obstacle, we can assume that the space between the sensor and the obstacle is most probably free. The Bresenham line algorithm connects the 2 cells (the one occupied by the robot and the one occupied by a laser scan point) with a line. The occupational likelihood of all cells through which the lines pass is then decremented by a certain value. Overtime, as these cells are repeatedly
identified as probably empty, their occupational likelihood value approaches 0.

**Fig 5.** Illustration of the Bresenham Line Algorithm. Lines connect robot’s current cell to the laser scans.
PART II: RGB-D Mapping

In this phase, the RANSAC algorithm was used to detect the dominant plane from the Kinect’s disparity (depth) image [2]. Once the ground points were found, their corresponding RGB values were mapped onto the floor in the occupancy grid map during the robot’s traversal through the environment.

Figure 6. Example of Ground Plane Found using RANSAC. Original RGB Image (left), Green
Simultaneous Localization and Map Construction of Indoor Environments by a Mobile UGV using a LIDAR

Ground Pixels (middle), 2D Mesh representation of Ground Plane (right)

SLAM Results

Fig 7. SLAM Map - Ground Truth
Fig 8. (a) Robot Trajectory (Red Line) and Map Constructed using Raw Motion Model.

(b) Map built using only 3 particles. The path taken by the robot is shown in yellow.
Fig 9. Map built using 50 particles
Fig 10. Map built using 100 particles
Conclusion

In this report, a particle-filter based SLAM algorithm has been presented. It is shown that this approach was successful at constructing an occupancy grid map of an indoor environment using laser scan matches produced by a 2D laser range scanner. Factors such as the number of particles used in the system affected the accuracy of the robot's pose estimates and subsequently map construction. It was seen that in general a larger number of particles produce better maps but this also increases the computational load of the algorithm. RGBD data from the Kinect was also successfully integrated into the system to augment the map. Further work will be done to assess and improve the performance of the system in cases where the robot makes numerous turns in the environment and traverses ramps.

References

