CSE 468/568 Lab 4: Grid Localization using Bayes Filter

The objective of this assignment is to make you familiar with Grid Localization which is a variant of discrete Bayes Localization. You will work with actual data using ROS bags. **NOTE:** This lab is quite challenging. You will get three weeks instead of two for this one. But please get started early.

Grid Localization

Grid Localization is a variant of discrete Bayes Localization. In this method, the map is an occupancy grid. At each time step, the algorithm finds out the probabilities of the robot presence at every grid cell. The grid cells with maximum probabilities at each step, characterize the robot's trajectory. Grid Localization runs in two iterative steps — Movement and Observation. This is very similar to the example we did in class.

After each movement, you should compute if that movement can move the robot between grid cells. For each observation, you should find the most probable cells where that measurement could have occurred.

ROSbag files

Rosbag is used for recording the messages published on certain topics to files. For this assignment, we have provided a bag file for you (`lab4.bag`) which includes the movements and observation information from one scenario. You need to read the bag file to simulate your localization algorithm running on a robot executing this scenario. For this purpose, read through the Rosbag tutorials.

Description of the Dataset

The data assets for lab 4 include the following files:

- lab4.bag
- tag_groundtruth.csv
- transformation.py
- transformation.cc

lab4.bag

This bag file contains messages from two topics: /odom and /tag_detections. Information about the bagfile can be gathered using the `rosbag info <bagfile>` command. Information about the structure of published messages can be gathered using the `rostopic list`, `rostopic info <topic_name>` and `rosmsg show <message_name>` commands.

The /odom topic receives periodic messages about the location of the robot. This location is calculated using the robots own sensors and can be inaccurate. The objective of this lab is to refine the location information using the landmark positions. When a tag is detected, its position, relative to the pose of the robot and in the camera frame, is published in the /tag_detections topic.

tag_groundtruth.csv

This file includes the actual locations of the landmarks. These locations are precise and were measure using an external system.

transformation.py

The poses of the detected tags are in the camera frame. For the robot to process this information consistently with the odometry data, the poses of the tags must be transformed from the camera frame to the robot frame. `transformation.py` file contains the method to transform co-ordinates from the camera frame to the robot frame. Simple running `python transformation.py` will print out the poses of all the detected tags in the robot frame. The method used for the transformation can be used as well.

This file has three methods: `camera2Robot, transformTagsToRobotFrame` and `main`. Given the $(x_c; y_c; z_c)$ co-ordinates in the camera frame, the `camera2Robot` method returns the $(x_r; y_r; z_r)$ in the robot frame. This method can be used to transform the detected tags in the camera frame to the robot frame. In the measurement step, these co-ordinates in the robot frame could further be transformed into the world frame. There is further discussion in the following sections.The other two methods `main` and `transformTagsToRobotFrame` are helper methods to read the tag detection messages from the bag file and iterate through each of the detections. They are included mainly for usage reference and for the completeness of the example code.

transformation.cc

This file has the c++ counterpart methodscamera2Robot and transformTagsToRobotFrameto the methods in the python script and they perform the same function.

Motion Model, Observation Model and Map

For localization, you need a map. There are twenty-two landmarks in the robot map. Their ground truth locations are provided for you in the file `tag_groundtruth.csv`.

The robot moves in the area within these landmarks, observing some at any given time. You should make a grid for 10m*8m coverage. Your cell size should be 10cm*10cm. You also should assume a dimension for the robot's heading. So your grid is 3 dimensional. The third dimension covers the robot's heading. The robot's initial pose is (17, 2, 2). (Notice that I have assumed there is no cell with index zero).

Motion Step

Each new `/odom` topic message gives us a pose after motion. To update our belief, we simply calculate the change in the x and y positions and the change in the heading angle (dx, dy and dθ) between two consecutive odom messages and move the belief in each cell by the changes in each of the dimensions dx, dy and dθ. We must also add a suitable motion noise depending on the motion model parameters we choose.

Assuming white Gaussian noise, we can add motion noise for the movement of each cell, by producing a Gaussian with the new position as the mean and the chosen motion model variance as the variance. This gives us $P(x_t \mid x_{t-1}, u_{t-1})$ which we can use in the Baye's filter motion step. Remember, the method to integrate motion is given by the equation below.

$\mathrm{P}(\mathrm{x}_t \mid x_{t-1}, u_{t-1}) = \Sigma_{x_i} P(x_t \mid x_i^{t-1}, u_{t-1}).P(x_i^{t-1})$

Pick a reasonable value for variance for the noise. It should not be too large to completely throw off the motion or two small to not affect it at all. I recommend experimenting with the noise after you have implemented the motion step with some initial value.

Measurement Step

Each new `tag_detections` topic messages gives us the pose of the tag, measured relative to the robots's position when the tag was detected. We also know the actual positions of the landmarks in the world from the `tag_groundtruth.csv` file. We use this information along with the detected position and a suitable noise model in the measurement step. First, we would need to transform the detected position from the robot frame to the world frame. Then, using the ground truth position and measurement model parameters (noise variance), we can compute the probability density function for the measurement. We can now use this in the measurement step of the Baye's filter.

In Grid Localization, for the purpose of moving the robot between cells, the motion and observation noise should be adjusted. You need a translation and rotation noise for movement and range and bearing noise for your observation. A good selection for this purpose is half the cell size. So for the example of 10*10 cells and 10 degree discretization, range and translation noises are 5cm, and bearing and rotation noise variances are 5 degrees.

Assuming a Gaussian measurement noise, for a single landmark detection, the mean would be the ground truth location of the landmark and the variance would be the variance chosen in our measurement noise model. This gives us a probability mass around the expected location that reflects our confidence in the measurement. Then, we transform the detected location of the tag from the robot frame to the the world frame. For a transformation, we require a translation (T) and a rotation (R). The translation (T) is the distance moved from the origin of the world frame. Fortunately, this can be obtained from the indices of the Baye's filter grid. In fact, each cell with a value in the grid corresponds to a belief of this translation. For the rotation matrix (R), we require the heading direction. Again, this can also be obtained from the θ dimension of the Baye's grid. With the heading direction, we can construct the rotation matrix as follows:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Now, we can compute the transformed point using the following formula:

$P_{world} = T + R * P_{robot}$

Once we have transformed our robot pose estimate prior to the world frame, we can now apply our sensor model to the sensor reading. Please remember, this was

$$P(x|z) = \frac{P(z|x).P(x)}{\Sigma_{x_i} P(z|x_i).P(x_i)}$$

You will calculate this for every possible grid cell just like we did in the one-dimensional example in class. **Note:** When the robot starts, it begins at a 90 degree angle with the world x-axis.

Your main ROS node should iterate through these two steps based on the messages received until there are no more messages. For each computation, you should calculate the posterior probability distribution and output the most likely cell where the robot is located. The output must be a series of locations output to the console. From our data collection, we have ground truth information about the robot location. We will compare your output with that to see if your calculations are correct.

Submission Instructions

You will submit lab4.tar.gz, a compressed archive file containing the lab4 folder. The folder should contain a launch file which running it should read the bag

file and run the grid localization algorithm. The folder should compile if I drop it into my catkin workspace and call `catkin_make`. Please take care to follow the instructions carefully so we can script our tests, and not have to dig into each submission. Problems in running will result in loss of points. Please use the submit script for submission using the syntax

```
$ submit cse468 lab4.tar.gz
```

or

```
$ submit cse568 lab4.tar.gz
```

depending on whether you are taking cse468 or cse568 respectively.

Details on the usage of the submit script can be found here.

The assignment is due Tuesday, Apr 24 before midnight. This lab involves learning various concepts. So, I highly recommend getting started early.