Control of P3DX and Mapping

To navigate in environment collision free robot needs following things

- Map of environment
- Position of robot in map
- Path (based on task)

1.1. Robot Mapping

The internal representation of the map can be "metric" or "topological":

- The metric framework is the most common for humans and considers a two-dimensional space in which it places the objects. The objects are placed with precise coordinates. This representation is very useful, but is sensitive to noise and it is difficult to calculate the distances precisely.
- The topological framework only considers places and relations between them. Often, the distances between places are stored. The map is then a graph, in which the nodes corresponds to places and arcs correspond to the paths.

Many techniques use probabilistic representations of the map, in order to handle uncertainty.

There are three main methods of map representations, i.e., free space maps, object maps, and composite maps. These employ the notion of a grid, but permit the resolution of the grid to vary so that it can become finer where more accuracy is needed and coarser where the map is uniform.

1.2. Occupancy Grid Mapping

Occupancy Grid Mapping refers to a family of *computer algorithms* in probabilistic robotics for *mobile robots* which address the problem of generating maps from noisy and uncertain sensor measurement data, with the assumption that the robot pose is known.

The basic idea of the occupancy grid is to represent a map of the environment as an evenly spaced field of binary *random variables* each representing the presence of an obstacle at that location in the environment. Occupancy grid algorithms compute approximate posterior estimates for these random variables.

1.2.1. Occupancy Cell and Grid

The term Occupancy is defined as a binary random variable. A random variable is a function from a sample space to the reals. This case Occupancy is defined in the probability space that has two possible states. Free and occupied. The occupancy random variable, then, has two values, 0 and 1

$m_{x,y}$: {free, occupied} → {0, 1}

An Occupancy grid map is just an array of occupancy variables. Each element of the grid can be represented with a corresponding occupancy variable. Figure 5 shows a 2D example of Occupancy grid map. Occupancy grid mapping requires, a Bayesian filtering algorithm to maintain an Occupancy grid map. Bayesian filtering implies a recursive update to the map. A robot can never be certain about the world so we use the probabilistic notion of occupancy instead of the occupancy itself.

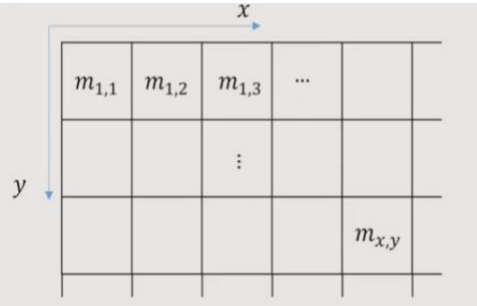


Fig 5. Occupancy grid map

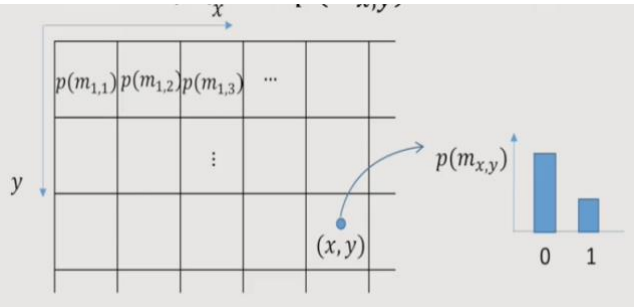


Fig 6. Probabilistic approach for occupancy

1.2.2. Probabilistic Model of Measurement

Occupancy grid mapping algorithms usually incorporate a range sensor. This sensor provides distance information. However in our map cell's point of view there are two possible measurements. A cell could be passed through by the ray. Which means it is free empty space. The light blue cells in the figure are an example of free cells. Also it is possible that a cell is hit by the ray. Which means a cell is occupied by something. The yellow cell where the ray starts at, is an example of occupied cells. We will use 0, for the free measurements. 1, where the Occupied measurement for each cell.

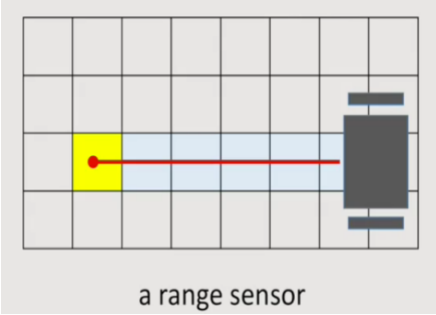


Fig 7. Range Sensor

***z*: {free, occupied} --> {0, 1}**

Now, we're going to think about a probabilistic model of the measurements. Given the occupancy state of each cell. There are only four possible conditional probabilities of measurements, which we can enumerate. Because the variables z and m are all binary, probability that z is 1 given m is 1 is the probability that we have occupied measurements for an occupied cell. Probability that z is 0 given m is 1 is the probability that we have free measurement for an occupied cell. We can define a probabilities of observation given m is 0, in the same way.

$p(z|m_{x,y})$: Probability of measurement conditioned of occupancy state of each cell

$p(z = 1|m_{x,y} = 1)$: True **occupied** measurement

$p(z = 0|m_{x,y} = 1)$: False **free** measurement

$p(z = 1|m_{x,y} = 0)$: False **occupied** measurement

$p(z = 0|m_{x,y} = 0)$: True **free** measurement

These are the measurement parameters we need to set. False measurements from sensor noise, the discretized space representation, moving objects, and uncertain knowledge of the robot motion. So we have four parameters. However, if you remember what the conditional probability is, you may notice that we actually have two parameters for our measurement model.

Now, we have basic understanding of elements of the Occupancy Grid Mapping Algorithm. We have defined the Occupancy variable that represents the state of grid cells. And the measurement model parameters that will be used to update the map. If we had some prior information of the cell, and we may take that into consideration, according to Bayes' rule.

1.2.3. Bayesian Filtering and Log Odd update Algorithm

We will define more mathematical notations to discuss the mapping algorithm. We want to update the occupancy probability of each cell from our measurements in a Bayesian framework. However, keeping track of probabilities directly can be hard. Instead of using occupancy probability itself. If there is a probability of something happening, written as p(X) and the odds can be considered as a ratio. This ratio is the probability of the thing happening over the probability of the thing not happening. We are going to use the odds of a cell occupied, which can be expressed as shown on the slide using the posterior probability notation.

Bayes' Rule: $$p(m_{x,y}|z) = \frac{p(z|m_{x,y})p(m_{x,y})}{p(z)}$$

Fig 8. Bayes' Rule

Applying Bayes' rule, we can rewrite the odds to include the sensor model term, the prior term, or both, the numerator and the denominator. Then the evidence term p(z) naturally goes away. Things get simpler when we take the logarithm of the odds. Note that the left-hand side includes the posterior odds and the right-hand side includes the sensor model and the prior. Because of the characteristics of log functions, the two terms multiplied on the right-hand side gets separated into an addition. This is a formula for the log-odds update of occupancy grid mapping.

Odd: = p(x)/p(x^c)

$Odd_{occupied}$ = p($m_{x,y}$ = 1|z)/p($m_{x,y}$ = 0|z)

Log Odd Update: $\log(odd^{post}) = \log(odd^{measurement}) + \log(odd^{prior})$

Log-Odd: $$\log\frac{p(m_{x,y}=1|z)}{p(m_{x,y}=0|z)} = \log\frac{p(z|m_{x,y}=1)}{p(z|m_{x,y}=0)} + \log\frac{p(m_{x,y}=1)}{p(m_{x,y}=0)}$$

Fig 9. Log-Odd Update formula

The map stores the log-odds values of each cell and the measurement model is represented as a log-odds as well. The computation for map updates then becomes additions of those log odds.

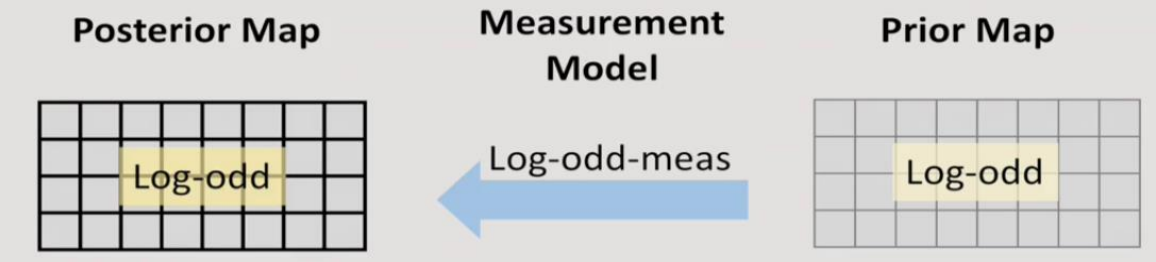


Fig 10. Log Odd Occupancy Grid Algorithm

1.3. Localization

Now, we will consider correlation based matching strategies for location a robot on a map given laser range data. This map registration process provides a very precise complement to odometry based localization.

Now let's introduce the LIDAR depth sensor. LIDAR stands for light detection and ranging and it provides distance measurements. Often engineered in a laser scanner to provide two dimensional data.

First, the occupancy grid map provides a grounds truth knowledge of what the robot should expect to observe in the world. Second, the set of lighter scan measurements provides information on what the robot is observing at the current time. The lighter scan measurements must be discretized according to the map representation, in order to be compared to the information from the occupancy map. With these two pieces of information the goal is to find the best robot pose on the map that explains the measured observations.

Searching over all possible poses of the robot can be difficult. But based on the odometry information, we have some tricks to make the search easier. We can constrain the search to a limited number of poses based on odometry information. Because we track the robot over time, we have the last known position of the robot and odometry information on how far the robot most likely moved. Thus, the most likely pose for the robot is now given a new set of laser data, is probably close to where the odometry predicts the robot to be. This prediction means that we can refine our search to poses near the prediction and be more confident in the validity of our search results. We measure each pose p in the search based on a map registration metric. One metric is to consider the sum of the map values m, at coordinates x and y, where the laser returns r, hit. This correlation metric can be modified to suit the application at hand. In our case, the value of our map cell will be a log odds ratio, so laser returns that

are seen at a map location with high probability of occupancy will strongly increase the registration in the metric score. Laser returns with map locations known as free cells will decrease the metric score. Additionally, the correlation can be scaled where returns from far distances affect the metric calculation less than nearby the laser returns. We register the robot on the map, at the pose that maximizes the registration metric. Thus, when the odometry is calculated, it uses this pose to predict a new position of the robot, in time. In addition to considering merely the laser returns, we can consider points for the laser returns penetrated. This calculation can further corroborate our map registration. It requires considerably more computation however.

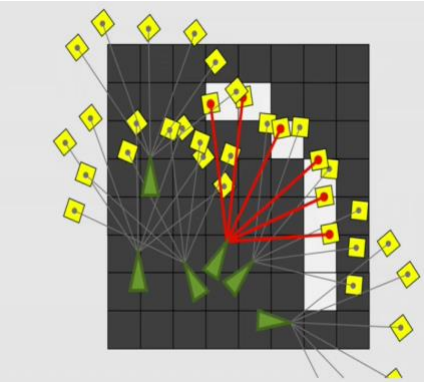


Fig 4. Searching over all possible space

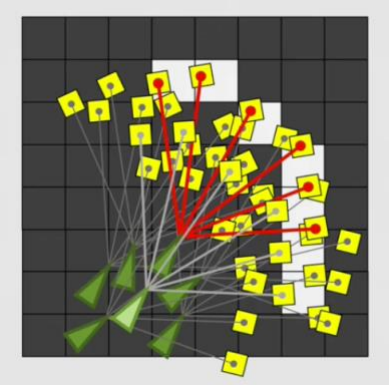


Fig 4. Constraining search based on odometry

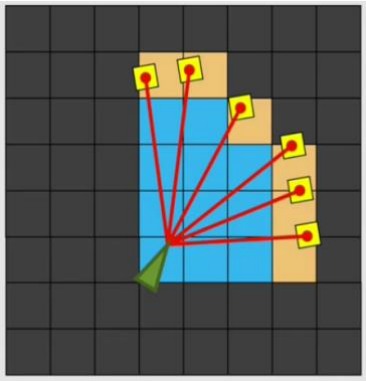
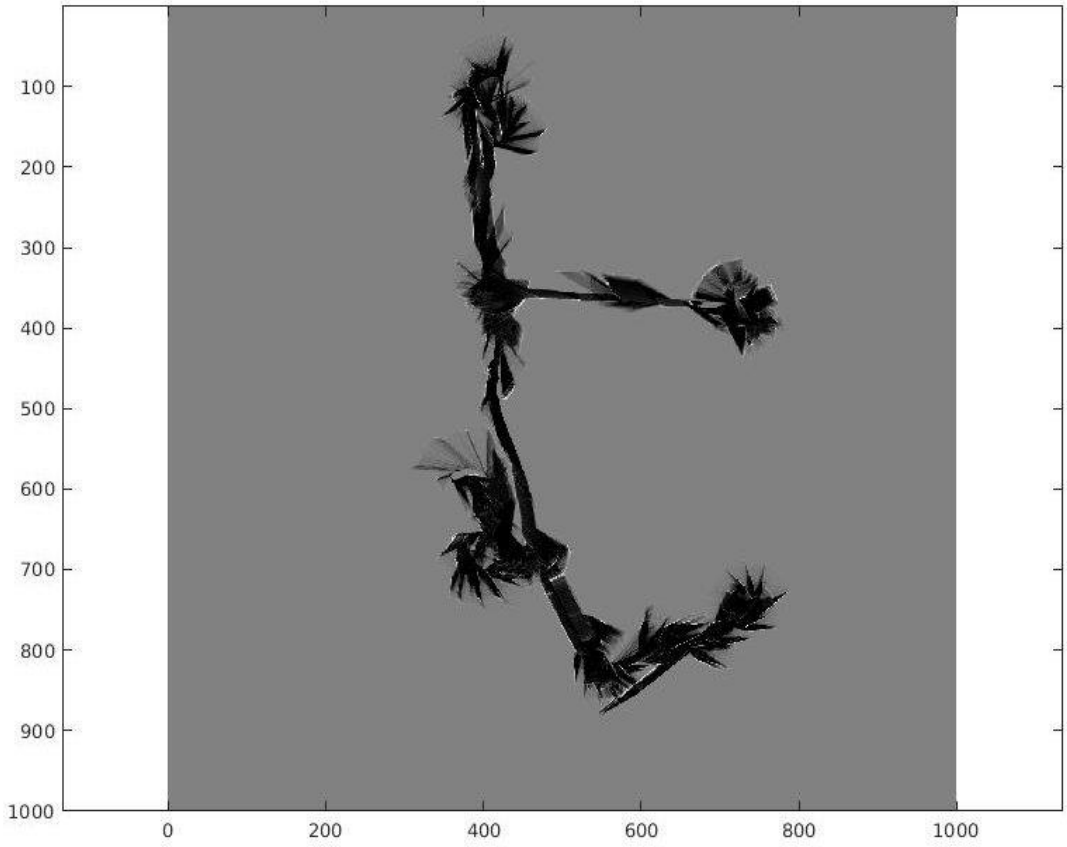


Fig 4. Best possible pose based upon correlation

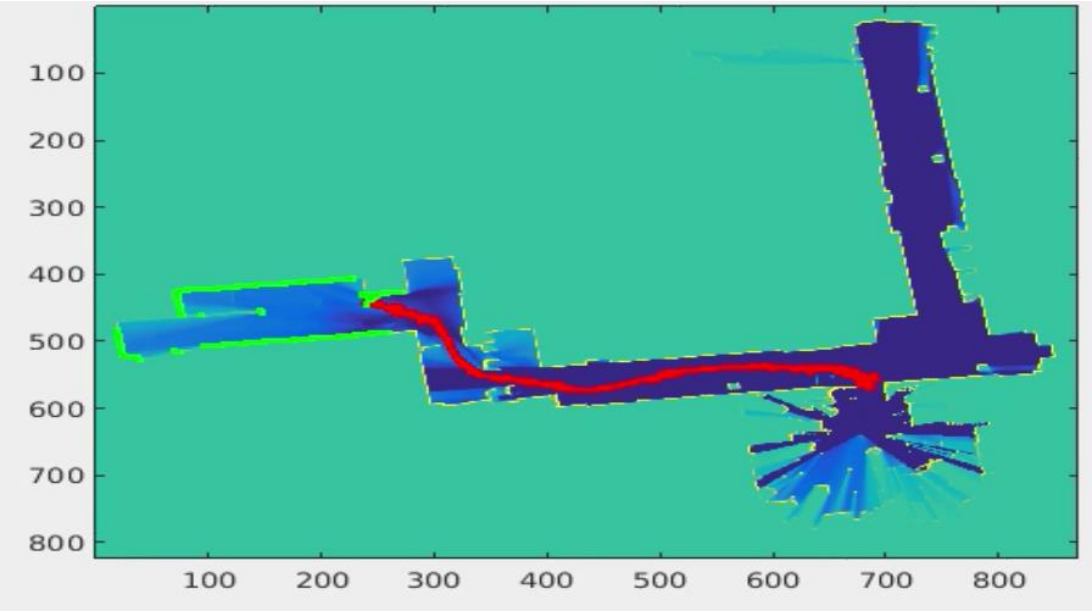
2. Results and Discussions

Map of Corridor in front of robotics lab built using Kinect



2.1. Localization using Particle Filter

Localization done using particle filter



3. References

3.1. https://www.coursera.org/learn/robotics-learning

3.2. https://www.coursera.org/learn/mobile-robot

3.3. wiki.**ros**.org/**Kinect**

3.4. wiki.**ros**.org/**openni**_launch

3.5. https://www.**coursera**.org/learn/**robotics**-flight