Autonomous Mobile Robots Coursework 2

Yi Zhou 9971603

Task 1

1.1 The binary mapping

The pose of the robot is located at the world frame, while the measurement are located at the robot frame. So the first step is mapping the measured distance and orientation from robot frame to world frame.

$$m_x = s_{x,k} + z_{\rho,k} \cdot \cos\left(s_{\theta,k} + z_{\alpha,k}\right)$$
$$m_y = s_{y,k} + z_{\rho,k} \cdot \sin\left(s_{\theta,k} + z_{\alpha,k}\right)$$

Then we present the map in occupancy grid. To do this, the position in world frame should be transferred into cells in grid map.

$$i = \left[\frac{m_x - ll_x}{res}\right] + 1$$
$$j = \left[\frac{m_y - ll_y}{res}\right] + 1$$

Each of the cell are allocated with a value to represent its probability of being occupied. In binary mapping, the probability of occupied cell should equal 1, while the probability of unoccupied cell should equal 0. The occupied cell is marked as black and unoccupied cell is white.

1.2 Simulation results

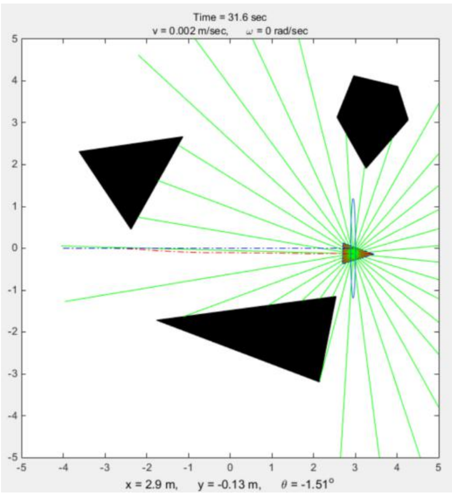


Figure 1.1

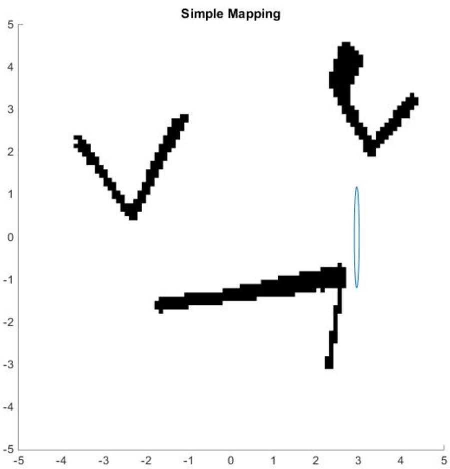


Figure 1.2

In this assignment, we map a 10×10 world frame into 100×100 grid map. Thus the resolution is 0.1. The robot is located at (-4,0) aiming at (3,0) in world frame. The measurement from LIDAR consists of the distance and angle between the detected obstacle and robot in robot frame. The maximum of the detected distance is 7 meters. In the occupancy grid, the surface of the obstacle is discretized into the a series of cells. If the surface is detected within the range, the LIDAR should return the indices of the occupied cells. As the robot drives to goal in the maximum speed of 0.8 m/s, the binary mapping is generated. From figure 1.2.1 and figure 1.2.2, the detected grid map is roughly similar to the original map.

Comparing grid map to original map, the generated positions of obstacle are shifted and overlaid. The uncertainty in measurement makes the surface not so smooth. The uncertainty in position, especially the lateral drift, makes the map rotate at some angle. For example, the surface of lower obstacle is thicker and thicker as the robot moves, due to the rotation.

Task 2

2.1 The probabilistic mapping

As we can see from binary map, there are varieties of uncertainties contributing to the final result. We should try to improve it by minimizing the effect of these uncertainty .From the view of Bayesian probability, the uncertainty in sensor can be taken into account by using probabilistic mapping. The probability is not fixed but is corrected when every new measurement is got. The updating rule are deduced in next section.

2.2 The updating rule

According to the equation of Bayes inference,

$$Posterior = \frac{likelihood \times prior}{normalizing\ factor}$$

$$P(c_{i,j}{=}occupied|z_k) = \frac{P(z_k|c_{i,j}{=}occupied) \cdot P(c_{i,j}{=}occupied)}{P(z_k)}$$

The prior is the probability of the cell is occupied before measurement. The likelihood is the character of sensor, indicating the accuracy of the sensor. The denominator is the normalizing factor which can be computed by law of total probability. At each time step, the posterior,i.e. the probability of the cell is updated by the Bayes' law. The probability of being occupied increases if the cell is detected as occupied, and decreases if unoccupied.

For the sake of computation efficiency, we can use the probability odd and take the log.

$$odd(c_{i,j}{=}occupied|z_k) = \frac{P(z_k|c_{i,j}{=}occupied|z_k)}{P(z_k|c_{i,j}{=} \neq occupied|z_k)}$$

$$= \frac{P(z_k|c_{i,j}{=}occupied) \cdot P(c_{i,j}{=}occupied)}{P(z_k|c_{i,j}{=} \neq occupied) \cdot P(c_{i,j} \neq occupied)} = odd(z_k|c_{i,j}{=}occupied) \cdot odd(c_{i,j}{=}occupied)$$

$$odd(c_{i,j}{=}occupied|z_k) = \exp\Big(\log\big(odd(z_k|c_{i,j}{=}occupied)\big) + \log\big(odd(c_{i,j}{=}occupied)\big)\Big)$$

It should be noticed that the output may be NaN when computing inf/1+inf. Thus the final value should be manipulated as 1-1/(1+odd) to make 1-1/(1+inf) equals one.

2.3 The algorithm to determine if the cell is occupied or not

If the obstacle is detected within the range, the LIDAR should return the indices of the occupied cells. The indices of unoccupied cells between the robot and the obstacle can be computed by some algorithm, for example, the Bresenham Line Algorithm. If nothing is detected in one direction, then all the cells which are passed through by this line can be consider as unoccupied. The algorithm is simply marked unoccupied for every points in this direction. It should be noticed that the rounded indicies should be restricted inside the map. Then using the updating law above to mark the cell with probability of being occupied.

2.4 The simulation results

The sensor properties ,which is the likelihood in Bayes inference, are given. The map is initialized as gray cell which represents 0.5 probability. The higher probability of being occupied the darker the cell is, vice versa. The robot follows the same trajectory as the previous task. The generated probabilistic map is shown in figures below.

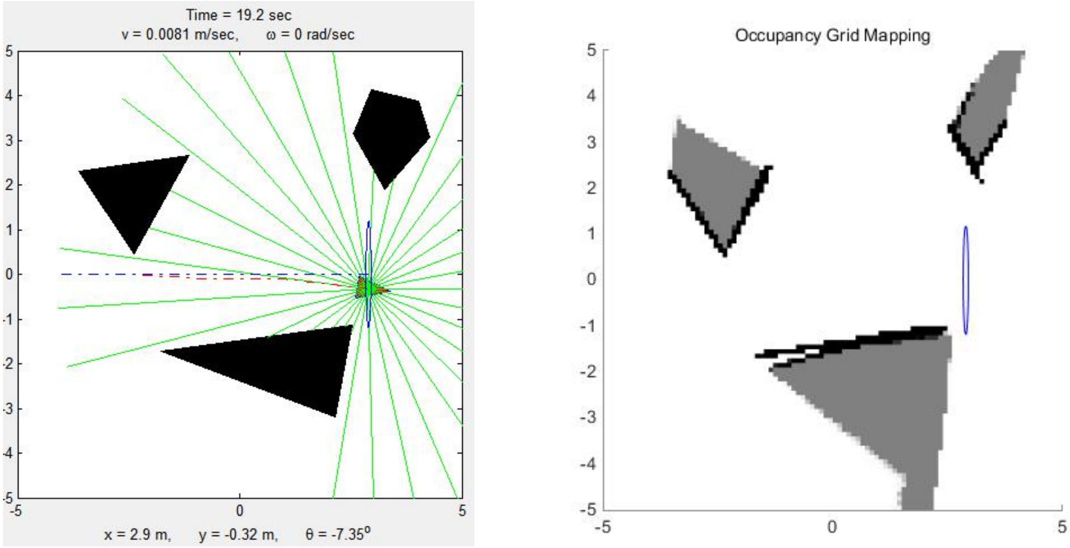


Figure 2.4.1 & 2.4.2 The trajectory and map when robot drifts clockwise

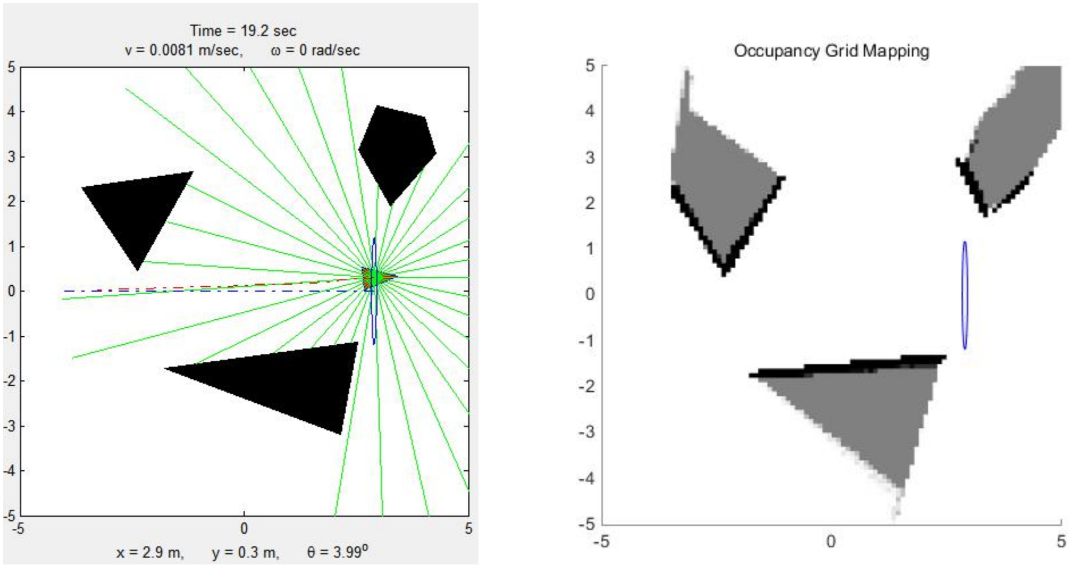


Figure 2.4.3 & 2.4.4 The trajectory and map when robot drifts anticlockwise

Compared to the binary map, the detected surface is thinner. It means that some occupied cells are detected as unoccupied in the later process. However, we do not have much confidence in this phenomenon. Sorts of uncertainties, such as measurement's noise, drift error, computational error .etc, can contribute to this. Among them, only the measurement's error is considered when updating the map. The drift error can affect the results severely. Comparing figure 2.4.2 with figure 2.4.4, the effect of drift error on the map is obvious. The map may rotate at a degree if the lateral drift occurs. One rough but simple way to deal with this rotation is to use the forgetting factor, which will be discussed in next task.

Task 3

As mentioned before, the map may rotate and the obstacle will be shifted or overlaid. If we

only consider about the current environment around the robot, the most direct way to solve this is to forget the past. The forgetting factor can be used to recover the probability of the long-time-ago cells into 0.5.
Design the forgetting factor as

$$\Delta = P - 0.5$$
$$P = P + 0.1 \cdot \left(1 - e^{\Delta}\right)$$

where P denotes the probability of all the cell in the map.
The simulation results is shown below.

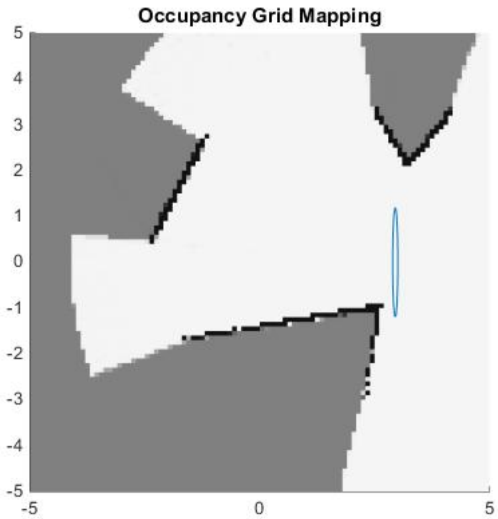


Figure 3.1

Task 4

4.1 Kalman filter localisation

The basis of the Kalman filter is the Bayesian inference under Markov assumption. Moreover, the good quality of Gaussian distribution enables the multiplication of two Gaussian is another Gaussian. The noise is always assumed as a zero mean Gaussian white noise. As a result we can describe the estimated position as a probability with mean value and covariance. Then update them under Bayes inference.

In the Kalman filter localisation, the basic idea is using the measurement to correct the estimation of position. There should be two step. The prediction step is the same as the model based localisation. The correction step takes the difference of the real and estimated measurement into consideration. It needs to know the true position of the landmarks at first. Then use it to correct the position of the robot.

4.2 The algorithm of EKF

Extended Kalman filter is used when the models are nonlinear. Then we can linearise them around the estimated position using Taylor expansion. The following process are the same as Kalman filter. Since the prediction step is illustrated thoroughly in previous assignment, only the correction step is introduced here.
Firstly, the position of landmarks are assumed to be known. Then, the estimated of measurement is given by

$$\hat{z}_{\rho,i,k} = \sqrt{(m_{x,i} - s_{x,k})^2 + (m_{y,i} - s_{y,k})^2} = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\hat{z}_{\alpha,i,k} = \operatorname{atan2}(m_{y,i} - s_{y,k}, m_{x,i} - s_{x,k}) - s_{\theta,k}$$

Linearised it around the estimated pose of robot, the Jacobian matrix G is computed as

$$G = \begin{bmatrix} -\frac{\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}} & -\frac{\Delta y}{\sqrt{\Delta x^2 + \Delta y^2}} & 0 \\ \frac{\Delta y}{\Delta x^2 + \Delta y^2} & \frac{\Delta x}{\Delta x^2 + \Delta y^2} & -1 \end{bmatrix}$$

The Kalman gain is given by

$$Z_k = G_k \hat{\Sigma}_k G_k^T + R_k$$

$$K_k = \hat{\Sigma}_k G_k^T Z_k^{-1}$$

where R is the covariance matrix of nosie.
Then the updating law for the mean and covariance matrix is

$$\mu_k = \hat{\mu}_k + K_k (z_{i,k} - \hat{z}_{i,k})$$

$$\Sigma_k = (I - K_k G_k)\hat{\Sigma}_k$$

Remember to use wrapToPi() to ensure the angle is correct.

4.3 The simulation using EKF

Test the performance of EKF by running several cases. The initial position is (-3,0) and goal is (3,0) for driveStraight2 case and driveToGoal case. The error between the estimated and real position is very small. The ellipsoid which represent the 99% confidence is also very small. The lateral drift is significantly reduced compared to the case without Kalman filter.

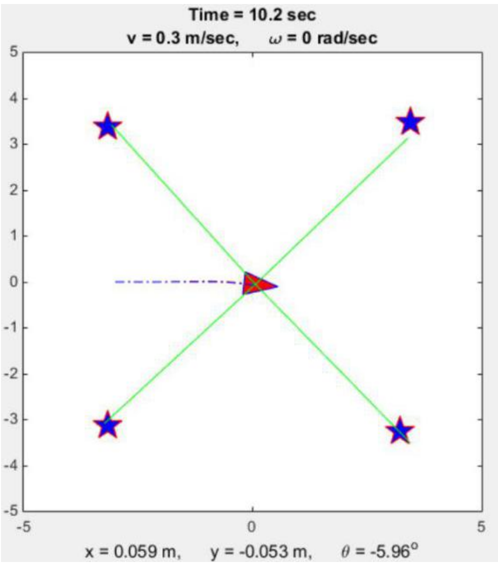


Figure 4.3.1 Drive straight

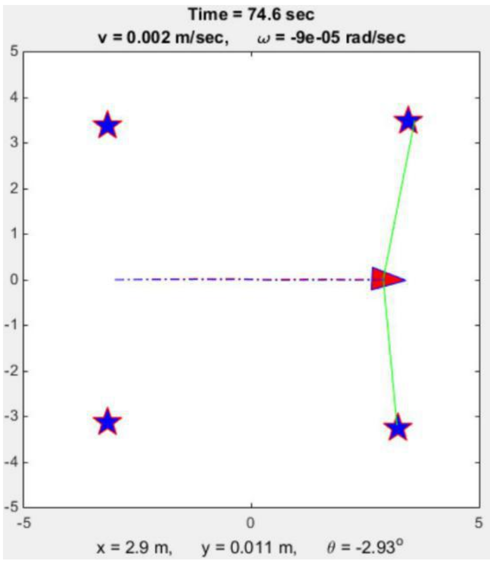
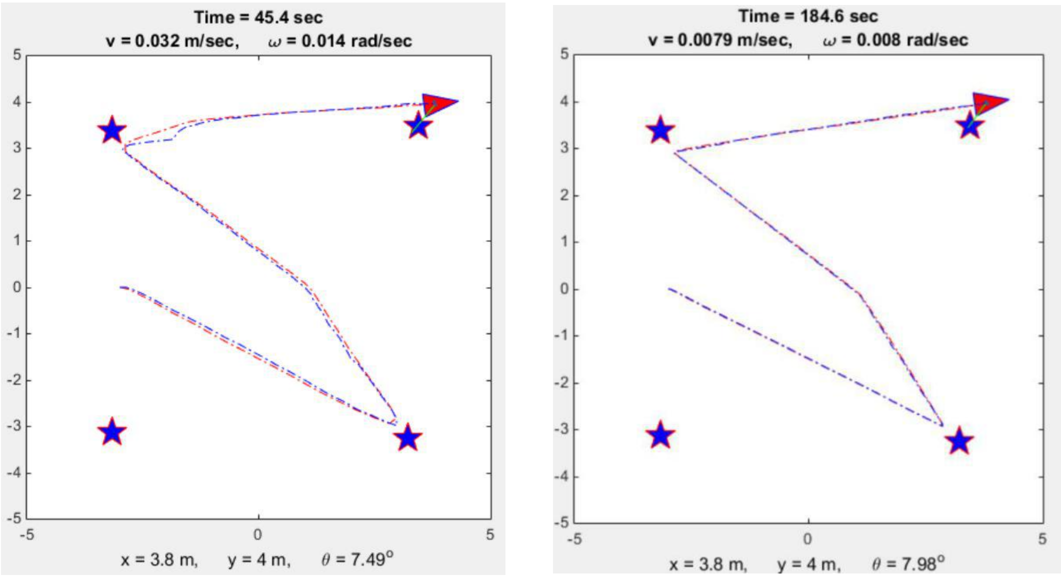


Figure 4.3.2 Drive to goal

Then the multiple goals case are tested. The results are shown in figure 4.4.3 and 4.4.4. In order to get more information from the measurement, the velocity should be slower. As figure 4.3.3 (a) and (b) depict, the error in estimated trajectory is reduced if change velocity from 0.8 m/s to 0.2 m/s. In assignment 1, the covariance is extremely large for multiple goals due to the accumulation of error in such long distance. If using Kalman filter, the trajectory matches well and the covariance is very small.



(a) u=0.8 m/s (b) u=0.2 m/s

Figure 4.3.3 Drive to multiple goals

Finally, the mapping method are tested together with Kalman filter localisation.
In the binary mapping method, the covariance of the estimated position of obstacle is large. In probabilistic mapping method, as mentioned previously, the uncertainty in measurement is taken into account. As we already known, the Kalman filter can minimize the position error of robot. Thus, in figure 4.3.5, the detected position of obstacle point is much more concentrated than that in binary mapping.

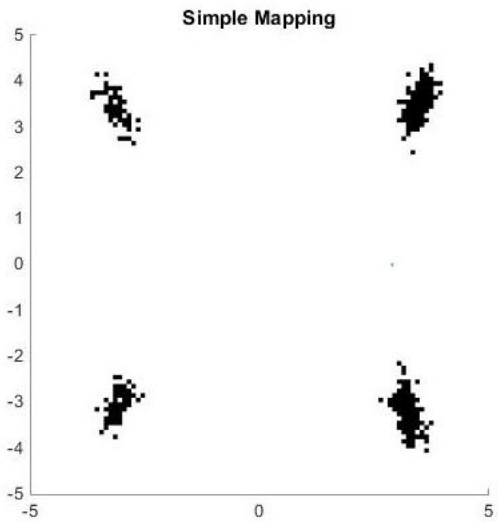


Figure 4.3.4 Binary mapping

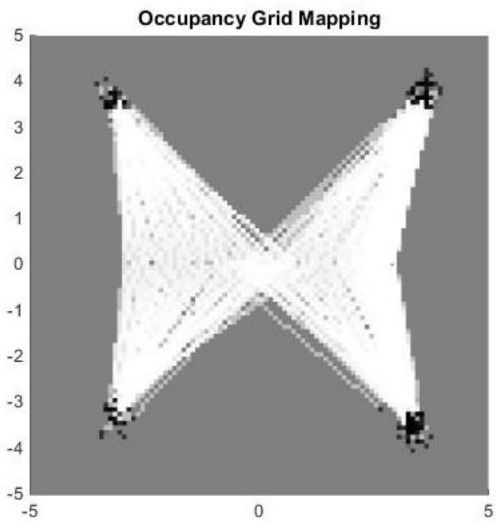


Figure 4.3.4 Probabilistic mapping