Autonomous Mobile Robots Coursework 1

Yi Zhou 9971603

Task 2

The pose is given by the Gaussian distribution

$$s_k \sim N(\mu_k, \Sigma_k)$$

Assume the error q_k is of zero mean Gaussian distribution.

$$q_k \sim N(0, Q_k)$$

Then the discrete motion model is

$$s_k = h(s_{k-1}, u_k) + q_k$$

where

$$h(s_{k-1}, u_k) = \begin{bmatrix} s_{x,k-1} + \Delta t \cdot v_k \cdot \cos(s_{\theta,k-1}) \\ s_{y,k-1} + \Delta t \cdot v_k \cdot \sin(s_{\theta,k-1}) \\ s_{\theta,k-1} + \Delta t \cdot \omega_k \end{bmatrix}$$

The Jacobian matrix of the equation above is

$$H_K = \begin{bmatrix} \frac{\partial h_1}{\partial s_{x,k}} & \frac{\partial h_1}{\partial s_{y,k}} & \frac{\partial h_1}{\partial s_{\theta,k}} \\ \frac{\partial h_2}{\partial s_{x,k}} & \frac{\partial h_2}{\partial s_{y,k}} & \frac{\partial h_2}{\partial s_{\theta,k}} \\ \frac{\partial h_3}{\partial s_{x,k}} & \frac{\partial h_3}{\partial s_{y,k}} & \frac{\partial h_3}{\partial s_{\theta,k}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta t \cdot v_k \cdot \sin(\mu_{\theta,k-1}) \\ 0 & 1 & \Delta t \cdot v_k \cdot \cos(\mu_{\theta,k-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

Thus the linearised motion model using Taylor expansion around μ_k is

$$s_k \approx \mu_k + H_K(s_{k-1} - \mu_{k-1})$$

Because the s and q are both Gaussian, according to affine transformation and sum rule of independent Gaussian the updated law for covariance matrix takes the form of

$$\Sigma_k = H_k \Sigma_{k-1} H_k^T + Q_k$$

The Q_k is the source of the increasing uncertainty. It is absorbed into the covariance matrix every time step. The linear velocity determine the value of change in one time step.

For the ease of illustration, we can build the coordinate based on the pose of robot. The equations above can be thought as the projection into the global coordinate through a similar transformation,(rotation matrix) like figure 2.1 shows.

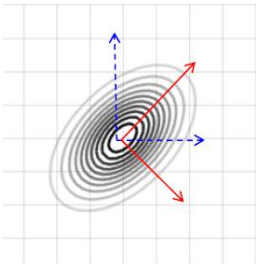

Figure 2.1

From a very intuitive view, the uncertainty is quite smaller than the traveling distance in the heading direction. However, the lateral speed should be zero, which means the uncertainty in angle and lateral velocity have relative large effect on it. Moreover, in reality, the turn error and drift error both should have bigger effect on the lateral direction. Thus, it is easy to infer that the uncertainty is more likely to affect the covariance matrix in lateral direction. That's why the shape of the 99% confidence of pose is always ellipsoid.

The true path, the estimated path and the covariance ellipsoid are shown below. As discussed before, the in standard deviation in lateral direction is larger than that of the heading direction.

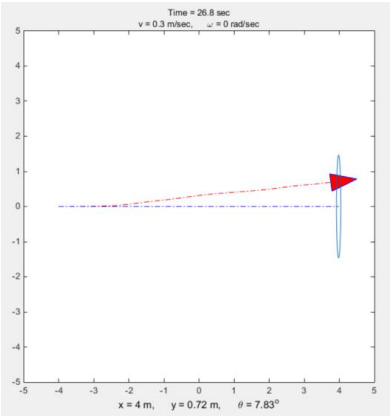

Figure 2.2

Autonomous Mobile Robots Coursework 1

Yi Zhou 9971603

Task 3

3.1 The control law

To drive the robot towards the goal, using two proportional feedback controller on distance and orientation separately,

$$v = k_d \cdot e_d$$
$$\omega = k_t \cdot e_t$$

where,

$$e_d = \sqrt{e_x^2 + e_y^2}$$
$$e_\theta = \text{atan2}(e_y, e_x) - \theta_r$$

3.2 Tuning the parameters

3.2.1 The effect of k_d

Too small k_d means very slow velocity. Too large k_d will make the velocity unstable. It should also be noticed that the larger the velocity, the larger error may occurs at every time step. Thus, we want the robot head to the goal in a moderate speed.

The simple idea is to set a maximum for velocity. The more smooth trajectory can be achieved by the function

$$k_d = v_0 \cdot \frac{1 - e^{-\alpha \cdot e_d^2}}{e_d}$$
$$v = k_d \cdot e_d = v_0 \cdot \left(1 - e^{-\alpha \cdot e_d^2}\right)$$

Thus the velocity will always less than the maximum v_0. The dynamics of the error in distance and the velocity, when α=1 and v_0=1, is shown in figure 3.2.1. The error will never become zero due to the measurement's accuracy. Moreover, the dynamics infers the robot will adjust itself with small action near zero. Thus, a threshold of error in distance should be made in order to stop the robot.

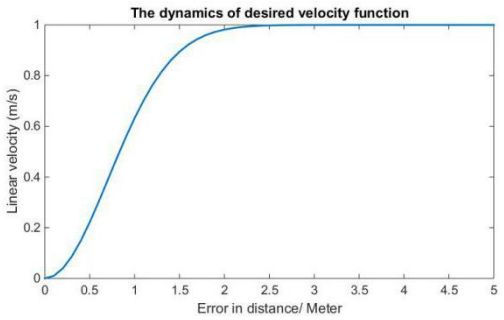

Figure 3.2.1

3.2.2 The effect of k_θ

If k_θ is too small, for example, equals 3, then the trajectory may become curved, due to the angle cannot reach the desired value in one time step.

If k_θ is too large, for example, greater than 12, the heading angle may oscillate a lot and even become unstable. Too much overshoot can lead to this phenomenon.

The appropriate k_θ I choose is 6. The tracking performance of the angle is show in figure 3.2.2 It heads to the desired position after 3 time steps. The overshoot is small. The trajectory looks acceptable as well.

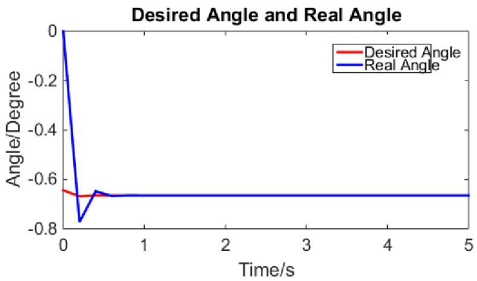

Figure 3.2.2

3.3 performance and improvement

In this assignment, the performance is acceptable. The system is fast enough with little tracking error.

Autonomous Mobile Robots Coursework 1

Yi Zhou 9971603

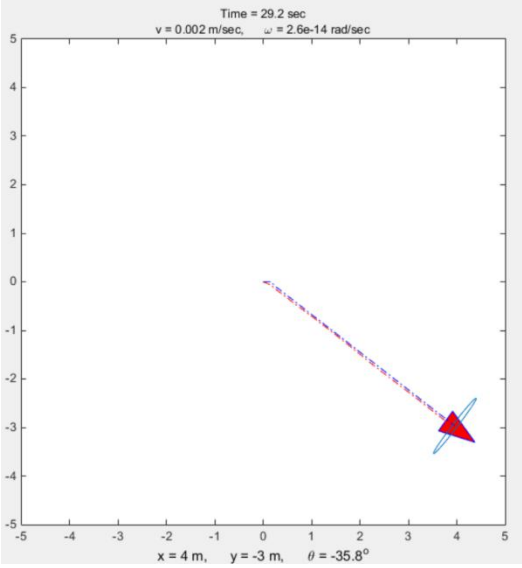

Figure 3.3.1

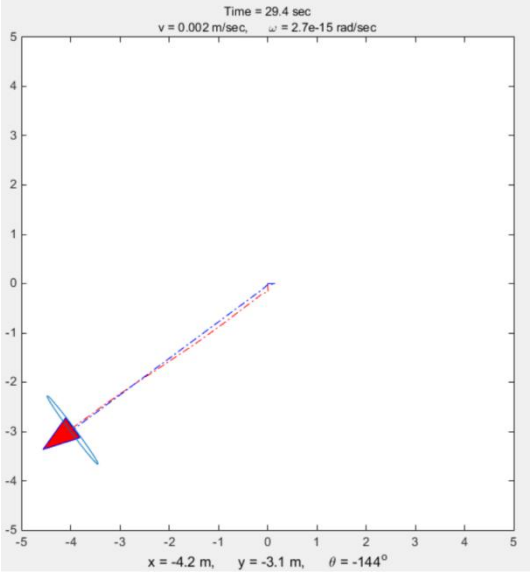

Figure 3.3.2

However, in reality, using simple proportional control may lead to problem with steady state error. A PI controller should be used instead to eliminate that error.

Task 4

The same control algorithm and parameters are used for multiple goals. Only the difference is changing its goal after reaching one. From figure 4.1 and 4.2, the estimated trajectory is good. However, the uncertainty is large due to the accumulation of error in different orientation.

The shape of ellipsoid, i.e. the covariance on the x and y direction, is different from the single goal case. For every goal, the ellipsoid on the lateral direction will always be larger. The orientation of the robot is different, so does the projection of the covariance matrix. As the robot moves, the uncertainty in x direction and y direction both accumulate. Therefore, the ellipsoid tends to be different shape.

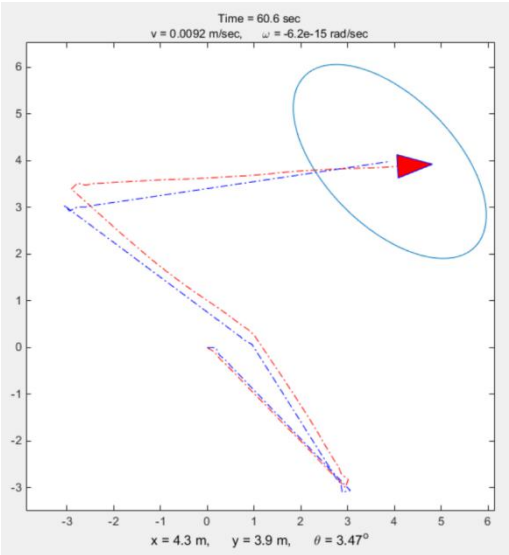

Figure 4.1

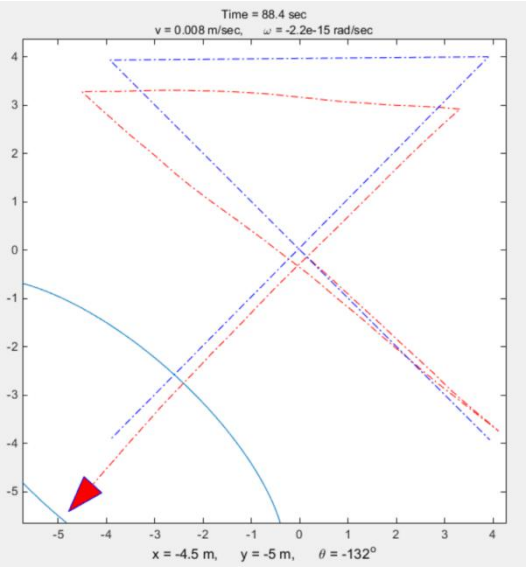

Figure 4.2

Autonomous Mobile Robots Coursework 1

Yi Zhou 9971603

Task 5

5.1 The algorithm

The algorithm can be separated into two sections, the goal_to_goal algorithm and the following_wall algorithm.

The goal-to-goal algorithm is the same as previous task. There are two steps in following-wall algorithm, i.e. when to start and when to end.

Step 1: Detect the obstacle and follow the wall.

If the shortest distance measured by the sensor is less than a threshold, the robot should consider whether the collision would happen. If the obstacle and the goal are on the two sides of the robot, like figure 5.1.2 , it is safe to continue to head to the goal. If the obstacle and the goal are on the same side of the robot, like figure 5.1.1, the robot should start to follow the wall with velocity of 0.1m/s. The desired velocity is perpendicular to the wall. It is obvious that there are two directions we can choose. The first option is turning at acute angle (if possible). The second option is always circling in one direction, clockwise or anticlockwise. The first one, which is I choose, is smarter because less angle means less effort and less error. The second method is more reliable, because it can always cycle a full round of a convex obstacle in order to find the leaving point. However, both these methods will get stuck in closed trajectory in some special concave case, which will be discussed later.

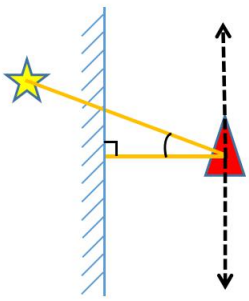

Figure 5.1.2

Step 2: Leave the wall and head to goal

When the obstacle and the goal are again on the same side, the robot should find its way to goal. It should stop circling and head to goal.

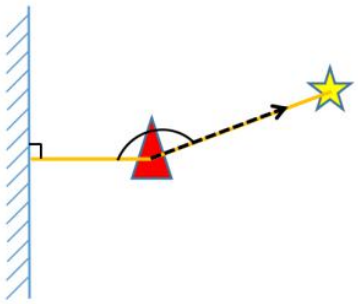

Figure 5.1.2

The trajectory is shown below. The same parameters of goal-to-goal is used here.

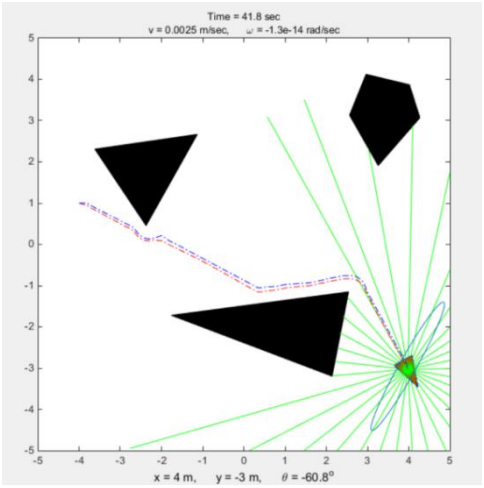

Figure 5.1.3

5.2 Comparison between the algorithms

The Bug 1 in figure 4.8(a) in the text book is time consuming, but it can always find a shortest way between the obstacle and goal. The Bug 2 in figure 4.8(b) in the text book is faster but may fail in some concave case, which will be discussed below. The algorithm which I use is similar but faster than Bug 2.

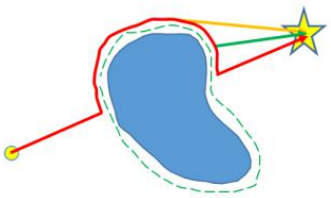

Figure 5.2

Autonomous Mobile Robots Coursework 1

Yi Zhou 9971603

5.3 The deficiency in my algorithm

There are two serious deficiency in my algorithm if applied to the reality. First one, it cannot deal with the obstacle shaped like figure 5.3.1. This can be solved by using bug 1 algorithm.

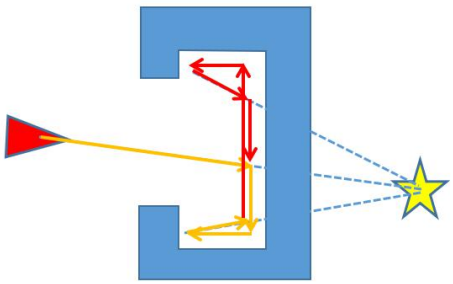

Figure 5.3.1

Second one, it cannot deal with the large drift error if the cycling velocity is large. In this assignment, the velocity when cycling around the obstacle is set to 0.1 m/s and the threshold of the minimum distance is set to a relative large value. However, in reality, this drift error cannot be solved via easily tuning the parameters. So a velocity which is perpendicular to the wall is necessary to avoid the obstacle.

In my algorithm, as figure 5.3.2, a rough idea to realize the same function is to add an additive term to ensure the robot will always turn a little bit more to avoid obstacle.

The desired angle = the following-wall angle + α* (following-wall angle divided by its normal)

Where α is a small tuning number.

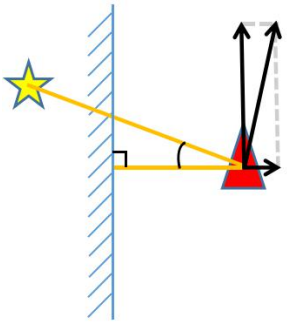

Figure 5.3.2

The resulted trajectory is depicted in figure 5.3.3. The trajectory tends to leave the wall little by little while cycling.

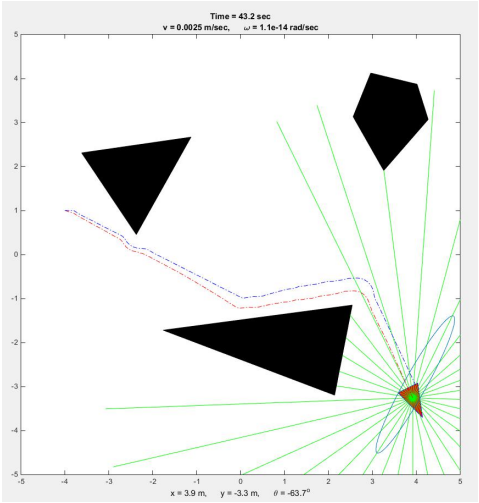

Figure 5.3.3

An alternative control algorithm, which is easy to implemented in reality ,is demonstrated in figure 5.3.4. The desired velocity is the vector addition of the goal-to-goal velocity and the avoid-obstacle velocity with some weighting numbers. However, this method can also have problem when the goal-to-goal velocity and the avoid-obstacle velocity are collinear.

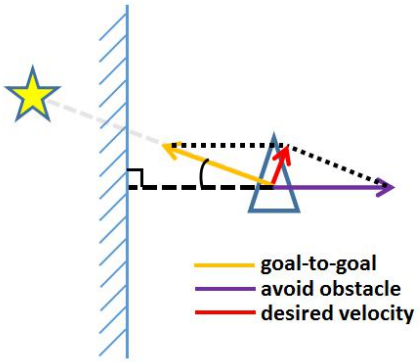

Figure 5.3.4

Reference:

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

Autonomous Mobile Robots Textbook