ROB312-TP4

PAN Mengyu

December 2020

1 Introduction

In this TP, we will apply the Extended Kalman Filter (EKF) in the Simultaneous Localization and Mapping (SLAM) method as the traditional Kalman Filter cannot be applied in the non-linear condition.

2 Influence of the environment

2.1 Question 1

In the code, we need to set the number and the position of the landmark as well as the robot trajectory. There are four situations:

1) default situation (a long loop and a sparse map) in the figure 1

2) a short loop and a dense map with many landmarks inside the robot perception radius in the figure 2

3) a long loop and a dense map with many landmarks all along the loop in the figure 3

4) a long loop and a sparse map with only few landmarks near the start position in the figure 4

In the 1, we can see that with the map becoming sparser and the loop getting longer, both translation error and rotation error grow bigger. This is because with fewer landmark it's more difficult to get the position accurately and the long loop means there is fewer landmark to be used in the perception.

2.2 Question 2

We change the Mahalanobis distance (KNOWN DATA ASSOCIATION = 0) and do the same as the question 1. The result is showed in the 2 and we can observe the similar tendency as in the question 1.

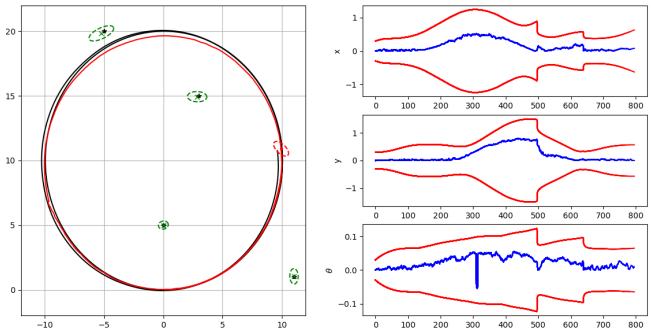

Figure 1: Original Situation

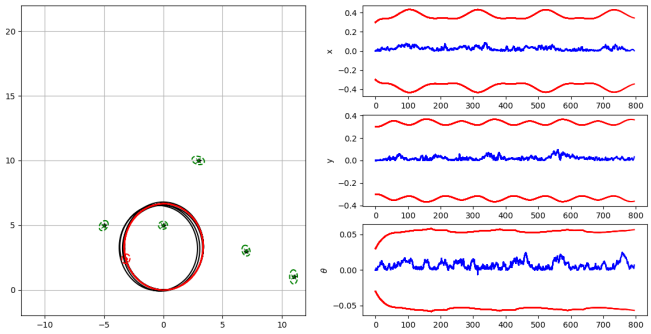

Figure 2: a short loop and a dense map with many landmarks inside the robot perception radius

3 Probabilistic models

3.1 Question 3

We need to change the estimated noise values Q and Py to make them 1) bigger 2)equal 3)small than the values used for simulation (Q Sim and Py Sim).

4 Undelayed initialization

4.1 Question 4

I am not sure what I modify is correct. In this question, we cannot know the exact position of landmark but we can know the direction between our position

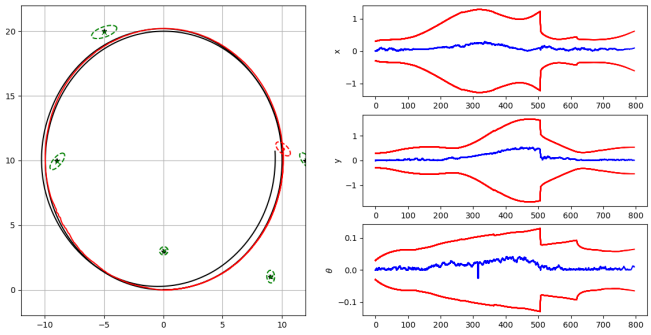

Figure 3: a long loop and a dense map with many landmarks all along the loop

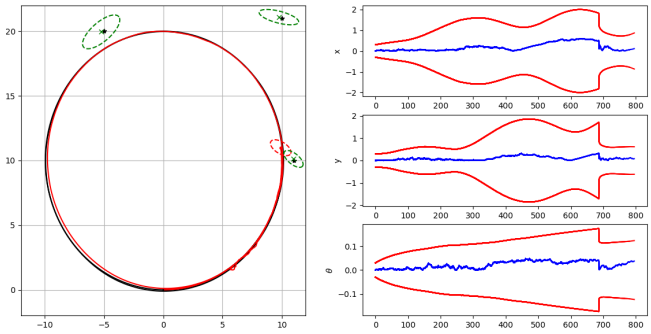

Figure 4: a long loop and a sparse map with only few landmarks near the start position

and landmark. So I modify the code in the observation as shown below but don't change the structure of the filter. The result is shown in the figure12.

```
dx = Landmarks[i, 0] - xTrue[0, 0]
dy = Landmarks[i, 1] - xTrue[1, 0]
distance = sqrt(dx*dx+dy*dy)
d = math.hypot(dx/distance, dy/distance)
angle = pi_2_pi(math.atan2(dy, dx) - xTrue[2, 0])
```

References

Situation	Mean (var) translation error	Mean (var) rotation error
default situation	0.2836789	0.02383337
a short loop and a dense map	0.03728238	0.006986068
a long loop and a dense map	0.1790156	0.01252366
a long loop and a sparse map	0.2307779	0.02198571

Table 1: Error in each situation for Q1

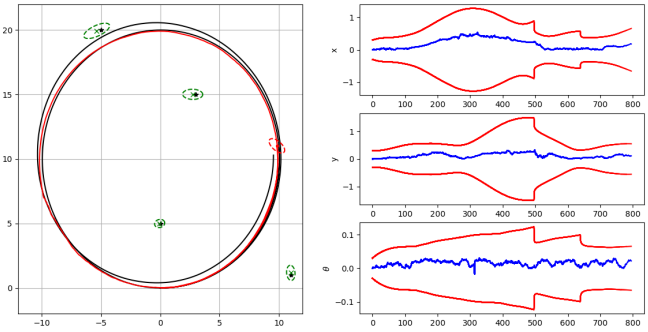

Figure 5: Original Situation

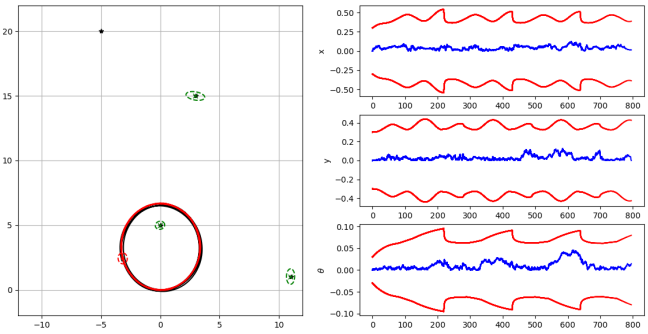

Figure 6: a short loop and a dense map with many landmarks inside the robot perception radius

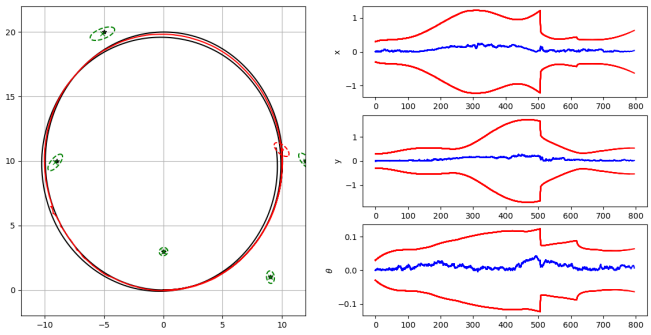

Figure 7: a long loop and a dense map with many landmarks all along the loop

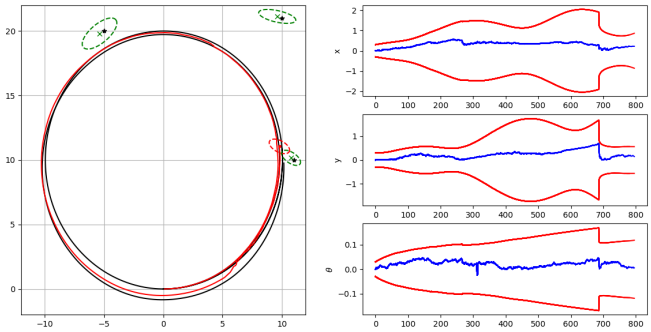

Figure 8: a long loop and a sparse map with only few landmarks near the start position

Situation	Mean (var) translation error	Mean (var) rotation error
default situation	0.2272802	0.01521054
a short loop and a dense map	0.05595261	0.01084257
a long loop and a dense map	0.1137069	0.01126761
a long loop and a sparse map	0.3933408	0.02152286

Table 2: Error in each situation for Q2

Relation	Mean (var) translation error	Mean (var) rotation error
big	0.2272802	0.01521054
equal	0.1186991	0.009140911
small	0.2719709	0.02644083

Table 3: Error in each situation for Q3

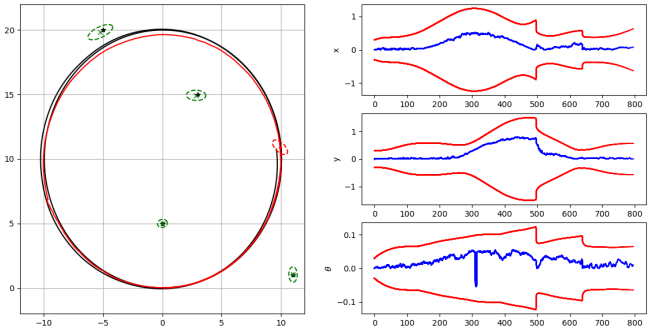

Figure 9: noise values Q and Py bigger than the values used for simulation

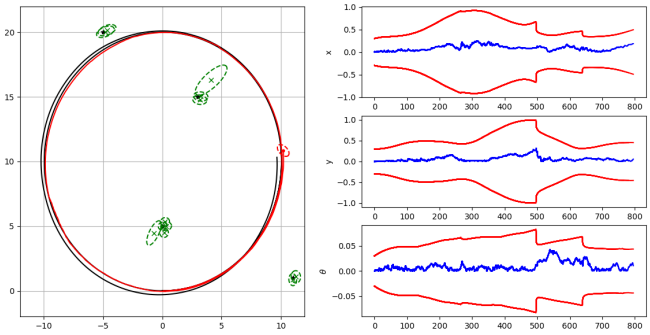

Figure 10: noise values Q and Py equal to the values used for simulation

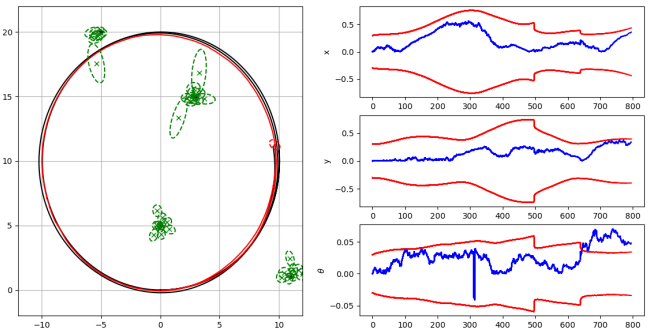

Figure 11: noise values Q and Py smaller than the values used for simulation

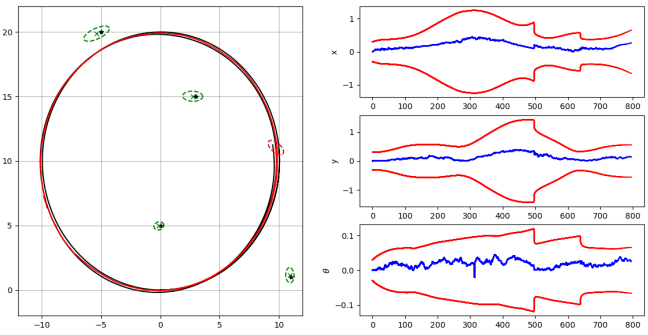

Figure 12: result for Q4