EKF-SLAM implementation with unknown data association

Report Author: Jingsheng Chen
(jinch@kth.se)

***Abstract*-The EKF-SLAM algorithm is a widely used SLAM algorithm based on Extended Kalman Filter, and it is also a very classic and simple implementation in SLAM. This report mainly discusses the specific implementation of the EKF-SLAM algorithm in the case of unknown data association, which is observation order of the landmark is unknown, and we will use normalized Mahalanobis distance to judge whether the observed landmark is new or old.**
The conclusion of this report is that when we do not know the data association, the stability of the algorithm is not strong, and the use of Mahalanobis distance to identify landmarks may fail, so the basic algorithm may not be applicable to more complex environments.

I. INTRODUCTION

SLAM (Simultaneous Localization And Mapping) technology occupies a very important position in today's robotics technology, and is mainly used to solve the problems of positioning, navigation and map construction when mobile robots run in unknown environments.

SLAM is more like a concept than an algorithm, usually it includes the following parts, feature extraction, data association, state estimation, state update and feature update. For each of these parts, a different algorithm can be used.

In this report, we will try to use the EKF-SLAM algorithm to implement some of the functions of the SLAM technology, which is a typical lightweight SLAM system. The implementation of EKF-SLAM is based on the EKF (Extended Kalman filter) algorithm.

We can use EKF to estimate the pose of the robot, but the estimated pose at this moment may be quite different from the actual one. Therefore, we cannot simply rely on the motion of the robot to estimate the position of the robot. In addition to estimating the pose of the robot, the EKF-slam algorithm also estimates the coordinates of all landmarks encountered in the path, that is, estimates the environment, thereby updating the current pose of the robot.

When the robot moves, its position will change. At this time, according to the observation of the position sensor, the robot will extract the feature points in the observation information, and then use the EKF to combine the position of the feature point currently observed, the movement distance, and the position of the feature point observed before the movement. Finally, the robot will estimate its current position and current environment information.

A. Contribution

In this paper, we provide the following contributions:

- Write and implement the EKF-SLAM algorithm in Matlab
- Design a steering car model and several map environment data containing landmarks, and use this data to test the performance of the algorithm

B. Outline

In the next section the work of other authors on the EKF-SLAM algorithm will be presented. In section Ⅲ The implementation of the EKF-SLAM algorithm will be explained. And sections 4 and 5 will present the results we get with this algorithm, along with an analysis and discussion of the results.

II. RELATED WORKS

EKF-SLAM algorithm is not complicated, and it has been proposed for many years, the concept itself has reached a state of maturity sufficient. During these years, many authors have done a lot of related and interesting research on it.

The consistency of the algorithm was studied by Bailey et al. in 2006[1], after that, Shoudong et al. also put forward a study on the consistency of the algorithm in 2007, and discussed its convergence in the nonlinear two-dimensional case[2].

In addition to consistency, algorithm complexity is also a research direction. A Divide and Conquer SLAM algorithm was proposed by Lina et in 2008[3], this algorithm computes a solution without relying on approximations or simplifications, and the complexity of each step is reduced from O(n2) to O(n).

Besides, EKF-SLAM has also been applied to many practical projects, such as lidar-based differential model of vehicle motion [4](similar to what is discussed in this report), indoor use of fusing ultrasonic sensors and stereo cameras[5], and Sonar-scanned based AUV navigation[6], et.

Due to the author's limited personal ability and time, this report does not discuss the in-depth research issues on EKF-SLAM like mentioned above, but mainly discusses the implementation of EKF-SLAM algorithm in Matlab.

III. Implementation and Theory

The implementation of the algorithm mainly depends on the framework of the EKF algorithm, and there is a key difference: in addition to estimating the pose of the robot, the EKF-slam algorithm also has to estimate the coordinates of all landmarks encountered in the path, that is, the estimation of the environment.

In order to avoid this long and boring report, we will not repeat the specific content of the EKF algorithm. For the specific implementation, readers can refer to the content of the EKF SLAM chapter in Joan's guide with matlab code[7]. This part only describes some of the interesting functions implementation details.

A. *Car dynamics*

Figure 1 is the pose of the mobile car at two adjacent moments, where θ1is the angle at which the mobile robot moves around the arc at the two adjacent moments, and θ3is the heading angle (towards the head) of the mobile machine at the two adjacent moments. The amount of change, l is the distance between the left and right wheels, d is the distance that the right wheel travels more than the left wheel, and r is the radius of the mobile robot's arc motion.

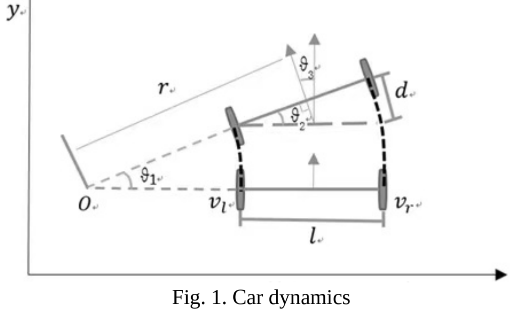


Fig. 1. Car dynamics

In this system, the pose of the car refers to (x, y, θ), which is controlled by two control quantities (Vn, Gn). Among them, the meaning of each representation is:

1)Vn: The forward linear speed of the car, which is preset in the system, here is 3m/s

2)Gn: The steering angle of the car, which is used to control the steering of the car, and it is determined by the distance of the next new road sign.

The pose model of the car can be listed as three equations, as follows:

$$\begin{cases} x' = x + V_n * dt * cos(G_n + \theta) \\ y' = y + V_n * dt * sin(G_n + \theta) \\ \theta' = \theta + \dfrac{V_n * dt * sin(G_n)}{l} \end{cases}$$

Among them, dt is the system sampling time, here is 0.025 seconds.

B. *Calculate the real pose*

In this step, we calculate the observation information through the simulator, add noise and send it to the EKF.

First, we calculate the Euclidean distance between the car's true pose and the current waypoint (car does not reach the waypoint, but is nearby). Then, we judge whether their distance is less than the square of *at_waypoint*. This parameter is set by ourselves and can be adjusted. If it is, it means that it is near the waypoint, and it starts to replace the next waypoint. Here, it will be checked whether the waypoint code exceeds the maximum value, and if it exceeds, it will be set to -1. Next, calculate the angle *deltaG* between the previous waypoint and the current pose. The program here limits the maximum value, and finally accumulates it into the corner G to complete the update of the corner.

Next, the program performs a simple loopback detection. In this project, we judge whether the currently reached path point returns to the origin. If it returns to the origin, the loopback times *number_loops* is reduced by 1. If the *number_loops* times are used up, *iwp* is set to 0, to exit the program.

C. *State observation*

In this system, the state observation has a certain observation period, which is controlled by the *dt_observe* parameter. When certain observation conditions are met, the simulator will first update the state observation. For the simulator, the observation update mechanism in the system is also very interesting. In fact, it is very simple.

In one sentence, it is to find the landmark in the detection circle with a certain radius in the real pose. The idea that the program looks for landmarks is:
1) Calculate the difference between all landmarks and the real pose;
2) Screen and save the id codes of all landmarks whose difference is within the scanning radius;
3) Extract the coordinate information of the landmark according to the id code;
4) According to the real pose and landmark coordinates, the landmark information is converted into the observation information format (observation distance, observation angle), and saved in the observation matrix of the EKF.

D. *Data association*

In this project, the observation order of landmarks is unknown. We use the normalized Mahalanobis distance to judge whether the observed landmarks are new or previously observed. The principle of Mahalanobis distance will not be repeated here. The implementation steps are as follows:

First, we use the observation model to calculate the observation matrix z_hat of each landmark and the corresponding covariance matrix H.
Then calculate the difference z_bar between the current observation matrix and the observation matrix of each known landmark, and then calculate the sample covariance S, which is the data Covariance distance.

Next, we calculate the Mahalanobis distance according to the formula, and then calculate the normalized Mahalanobis distance through the log function:

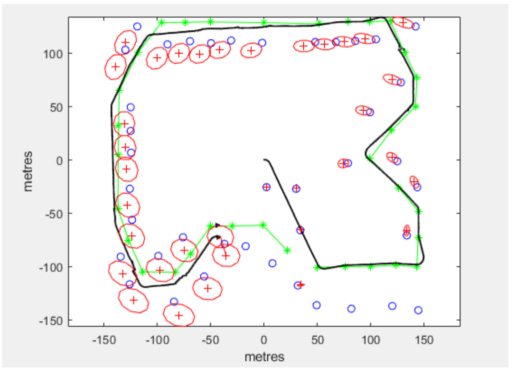
$$d_{nis} = \bar{z}^T * S^{-1} * \bar{z}$$
$$d_{nd} = d_{nis} + \log(\det(S))$$

Next, we compare the Mahalanobis distance and the normalized distance to determine whether the currently observed landmark is a new landmark, and then perform operations such as updating the observation matrix.

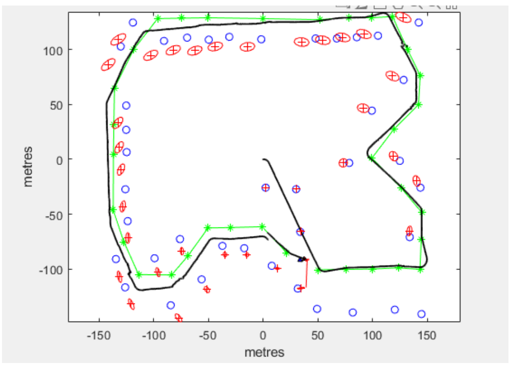
IV. Result and Discussing

This part tests two different sets of data, and records the actual pose of the car and the error of the predicted pose. The blue circle in the figure is the landmark, the green line is the trajectory formed by the waypoint, and the green * is the waypoint. The black is the curve the car actually walks, and the red x is the prediction of the landmark and its corresponding uncertainty ellipse.

Test data1:



(a)



(b)

Fig. 2. The first running circle

In the first lap of the car's motion(Figure2), we can find that the uncertainty ellipse keeps increasing with the car's movement, which is due to the increasing uncertainty of the landmark prediction by the intervening car that measures the noise, until the car completes the first lap and observe previously observed landmark for the second time. In this moment the loop closing and the uncertainty is instantly reduced.

This is because when the car re-observes this landmark, its uncertainty is much smaller than that of the later landmark, and through the association between this landmark and the previous landmark, the uncertainty reduction will propagate forward through loop closing and reduce other uncertainty about landmarks. We can observe that the landmarks that are farthest from the initial landmark (that is, near the diagonal of the map) have the greatest uncertainty after loop closing.

When the car moves to the beginning of the second lap (Fig.3), its pose has a large deviation, because it does not observe any landmarks to correct its position (the landmark

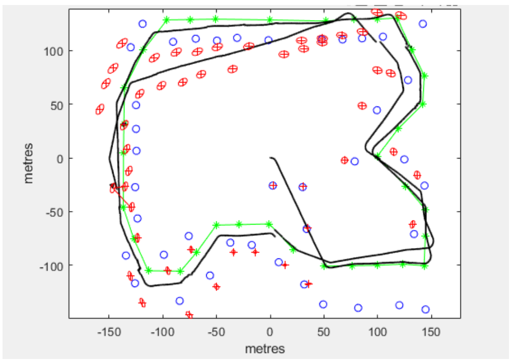


Fig. 3. The second running circle

in the lower right corner of the map is not observed, in Fig.3), and then the error accumulates more and more large, resulting in a complete failure of the second lap. In the few tests I repeated, sometimes the car would correct its position, and sometimes it wouldn't, and it mainly because of the different random noise, resulting in inconsistent error each time.

I think this may be because when the error exceeds a certain limit, the data association will fail, causing the car to not correctly calculate its own pose. This also shows that the EKF-SLAM algorithm is highly dependent on landmarks. When the car does not observe landmarks during a period of operation and encounters large noise, it may fail to run.

Test data2:

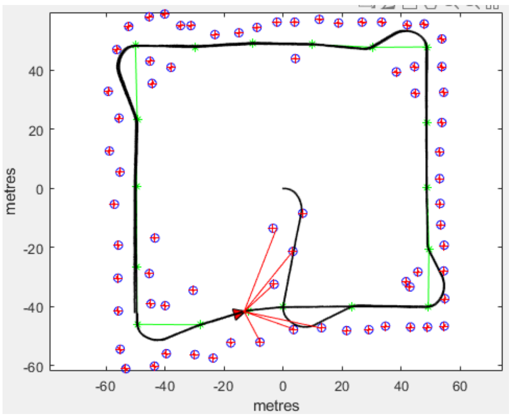


Fig. 4. Running result of data 2

It can be seen from the second set of data that the EKF-SLAM algorithm can get a better result when it runs in a small size, enough landmarks and a relatively simple environment.

When analyzing the error, we can see that when the car re-observed the previous landmark for the first time, that is, when the loop closing was completed, it corrected its own posture and reduced the error. After that, the car has been running in a smaller error. within the range.

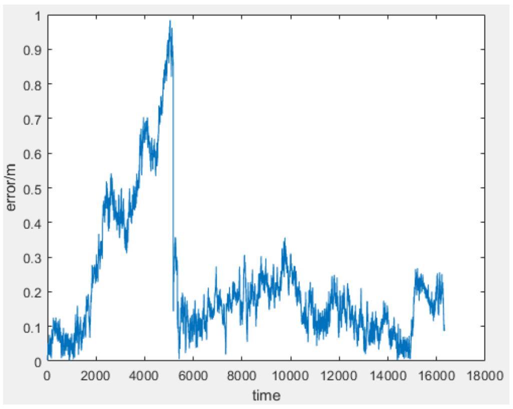


Fig. 5. Error of data 2

V. Conclusion

We have shown that the EKF-SLAM algorithm can get a good result when it runs in a relatively standard environment, but its processing ability in the complex environment is poor. One problem is that when we do not know the data association, the stability of the algorithm is not strong, and the use of Mahalanobis distance to identify landmarks may fail. So we can infer that in the face of some landmarks with weak features or symmetric environments, The probability of errors in the EKF-SLAM algorithm will increases.

EKF-SLAM itself is only a very basic algorithm. As mentioned in the second part of this report, some more complex optimization work is required if we want to apply it to specific engineering projects.

REFERENCES

[1] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, “Consistency of the EKF-SLAM algorithm,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3562–3568.

[2] S. Huang and G. Dissanayake, “Convergence and consistency analysis for extended Kalman filter based SLAM,” *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1036–1049, 2007.

[3] L. M. Paz, J. D. Tardós, and J. Neira, “Divide and Conquer: EKF SLAM in \$ O (n) \$,” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1107–1120, 2008.

[4] D. Wang, H. Liang, T. Mei, H. Zhu, J. Fu, and X. Tao, “Lidar Scan matching EKF-SLAM using the differential model of vehicle motion,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 908–912.

[5] S. Ahn, J. Choi, N. L. Doh, and W. K.

Chung, “A practical approach for EKF-SLAM in an indoor environment: fusing ultrasonic sensors and stereo camera,” *Auton. Robots*, vol. 24, no. 3, pp. 315–335, 2008.

[6] A. Mallios, P. Ridao, D. Ribas, F. Maurelli, and Y. Petillot, “EKF-SLAM for AUV navigation under probabilistic sonar scan-matching,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4404–4411.

[7] J. Sola, “Simulataneous localization and mapping with the extended Kalman filter,” *Avery Quick Guide MATLAB Code*, 2013.