Line-feature-based SLAM Algorithm[1)]

HAN Rui LI Wen-Feng

(*Department of Logistic Engineering, Wuhan University of Technology, Wuhan* 430063)

(E-mail: coolhanrui@tom.com)

Abstract A line-feature based SLAM algorithm is presented in this paper to resolve the conflict between the requirements of computational complexity and information-richness within the point-feature based SLAM algorithm, All operations required for building and maintaining the map, such as model-setting, data association, and state-updating, are described and formulated. This approach has been programmed and successfully tested in the simulation work, and results are shown at the end of this paper.

Key words Line-feature, SLAM, algorithm

1 Introduction

Simultaneous localization and mapping (SLAM) is a hot area in robotics. There are several kinds of SLAM algorithms, among which two are more commonly used than others. They are the grid-based SLAM algorithm and the feature-based one. And the latter develops rapidly. So far, the point-feature based SLAM algorithm has been well studied including the computational complexity, management of the map and so on. It is a dilemma that more point-features should be extracted to get a rich map, while fewer should be utilized in order to save the computational resources. While many ways are developed to solve this problem such as SLAM with compressed filter, and SLAM with local maps[1], there is still a lot to improve. Furthermore, the point-feature based SLAM algorithm provides a map with scattered points which is usually hard to meet the needs of information richness. This paper presents a line-feature based SLAM algorithm which may give a richer map with less computational complexity compared to the point-feature based algorithm.

2 Brief introduction to the feature-based SLAM algorithm

The feature-based SLAM algorithm is an important branch of SLAM algorithms, which uses a filter to update its state. The Kalman filter is the most popular one among all the filters, whose main idea is predicting and updating. To predict the state of a robot at the next time, the process model should be built. To update the state, firstly, an observation model is needed to predict the observation. Then compare the predictions with the estimations of the actual measuring, and update the state with the innovations from the comparison. The formulations used for updating are as follows

$$\hat{x}(k+1) = \hat{x}(k+1|k) + W(k+1)[z(k+1) - h(\hat{x}(k+1|k))] \tag{1}$$

$$P(k+1) = P(k+1|k) - W(k+1)S(k+1)W^{\mathrm{T}}(k+1) \tag{2}$$

$$W(k+1) = P(k+1|k)\nabla^{\mathrm{T}} h_x(k+1)S^{-1}(k+1) \tag{3}$$

$$S(k+1) = \nabla h_x(k+1)P(k+1|k)\nabla^{\mathrm{T}} h_x(k+1) + Q'(k+1) \tag{4}$$

To find the meanings of these parameters, see [1,2].

3 Introduction to the line-feature based SLAM algorithm

3.1 The process mode

Considering that the vehicle is controlled through a demanded velocity v_c and steering angle α, the process model that predicts the trajectory of the centre of the back axle is given by

$$\boldsymbol{X}'_r(k+1) = \begin{bmatrix} x'_r(k+1) \\ y'_r(k+1) \\ \phi'(k+1) \end{bmatrix} = \begin{bmatrix} x'_r(k) + \Delta T v_c(k) \cdot \cos\phi'(k) - \dfrac{v_c}{L}(a \cdot \sin\phi'(k) + b \cdot \phi'(k)) \cdot \tan(\alpha(k)) \\ y'_r(k) + \Delta T v_c(k) \cdot \sin\phi'(k) + \dfrac{v_c}{L}(a \cdot \cos\phi'(k) - b\sin\phi'(k)) \cdot \tan(\alpha(k)) \\ \dfrac{v_c}{L} \cdot \tan\phi'(k) \end{bmatrix}$$

1) Supported by National Natural Science Foundation of P. R. China (60475031)
Received May 23, 2005; in revised form October 19, 2005

where ΔT is the sampling time, a, b, L are parameters shown in Fig. 1. Notice that there are two coordinates in the system: the sensor frame, and the robot frame, respectively. To be simple, we consider the robot as a 'point' and take the sensor frame as the local frame used later. After a simple translation, the final discrete model in the global coordinates can be approximated with the following set of equations

$$\boldsymbol{X}_r(k+1) = \begin{bmatrix} x_r(k+1) \\ y_r(k+1) \\ \phi(k+1) \end{bmatrix} = \begin{bmatrix} x'_r(k+1) + a \cdot \cos(\phi'(k)) - b \cdot \sin(\phi'(k)) \\ y'_r(k+1) + a \cdot \sin(\phi'(k)) + b \cdot \cos(\phi'(k)) \\ \phi'(k+1) \end{bmatrix} + \gamma \tag{5}$$

where γ is the process noise. As the features considered are stationary, the "state transition equation" for the features is $F_i(k+1) = F_i(k)$. Now the state vector for time $k+1$ is $\boldsymbol{X}(k+1) = [X_r(k+1); F_i(k+1)]^{\mathrm{T}}$.

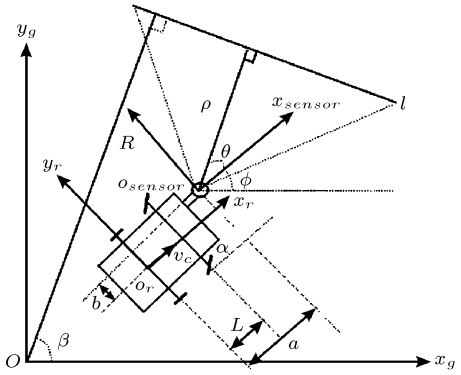

Fig. 1 The process model

3.2 The observation model

Algorithms developed for extracting line features in the environment with sensors is shown in [3]. Here, the line hypotheses are parameterized by perpendicular distance from the origin, and the angle between the line normal and the $\boldsymbol{x}$-axis, (shown in Fig. 1) according to which we get the observation model as follows:

$$h = (x(k+1|k)) = [\rho_i(k+1|k); \theta_i(k+1|k)]^{\mathrm{T}} + \gamma_h = \begin{bmatrix} R_i - \sqrt{x_r^2(k+1|k) + y_r^2(k+1|k)}\cos(\beta_i - a\tan\left(\dfrac{y_r(k+1|k)}{x_r(k+1|k)}\right) \\ \beta_i - \phi(k+1|k) \end{bmatrix} + \gamma_h \tag{6}$$

where γ_h is the white Gauss noise with zero mean and covariance σ^2.

3.3 Building the map

3.3.1 The initialization

When a robot moves in surroundings which are initially completelys unknown, the choice of a world coordinates is arbitrary. As there is no or little prior knowledge of the environment, the coordinate frame can be defined to have its origin at the robot′s starting position, and the initial uncertainty relating to the robot′s position in $P_{rr}(0)$ is set to zero or assigned a proper value. If there is prior knowledge of some feature locations, then it is put into the map explicitly and the feature positions should be assigned suitable initial covariance values $P_{mm}(0)$. The typical initialization would be to have several well known feature positions with low covariance effectively pinning down the coordinate frame, with a more uncertain robot starting location. And the whole state covariance can be presented by $\boldsymbol{P}(0) = [P_{rr}(0), P_{rm}(0); P_{rm}^{\mathrm{T}}(0), P_{mm}(0)]^{\mathrm{T}}$.

3.3.2 Data association

Suppose that there are i features extracted, fixed and put into the map $M(k)$ till time k and at time $k+1$, a set of features are detected, denoted as $L(k+1)$. According to the main idea of the feature-based SLAM algorithm, we usually have $L(k+1) \cap M(k) \neq \Phi$, and the data association is actually a

course of finding this intersection. To consider the effect of the parameters R and β simultaneously, we design the comparing function as

$$f_{n,obsm}(R_n, \beta_n, R_{obsm}, \beta_{obsm}) = R_n^2 + R_{obsm}^2 - 2R_n R_{obsm}\cos(\beta_{obsm} - \beta_n)$$

Consequently, the measurement to the Mahalanobis distance is

$$D_{n,obsm}^2 = f_{n,obsm}^{\mathrm{T}} C_{n,obsm}^{-1} f_{n,obsm} < \chi_\alpha^2(n) \tag{7}$$

where the covariance $C_{n,obsm} = H_{n,obsm}P(k)H_{n,obsm}^{\mathrm{T}} + G_{n,obsm}S(k)G_{n,obsm}^{\mathrm{t}}$, $H_{n,obsm}$ and $G_{n,obsm}$ are Jacobian matrixes of the function with respect to the variables R_n, β_n and R_{obsm}, β_{obsm}, respectively. $P(k)$ and $S(k)$ are calculated according to (2) and (4).

In the case where $n = 1$ and $\alpha = 0.95$, the threshold is 3.841.It must be noticed that there are three cases which satisfy the criterion: two features are nearly overlapped; or they are different sections of one feature; or they are different features located linearly. As to the first two situations, we call them 'true match', and the third one is called 'false match' which should be added on some restrictions to avoid.

3.3.3 Updating

Here, we still use the Kalman filter to update the states. At time k, prediction of the observations to the fixed features in the existing map is made according to the formulations in (7). At time $k+1$, suppose that we have $L(k+1) \cap M(k) = L'$ after the data association and we will get the estimations of the observations to the features in the set L'. With the estimations and their corresponding predictions, the state may be updated according to the formulations (1), (2), (3) and (4). As R and β are used to parameterize line-features, only they are updated. But in practice, what we usually attain are line segments due to the properties of sensors. So there is a need to make compensations to the extreme points.

3.3.4 Augmenting the state

As what was assumed before, the set of features detected is denoted as L and the set of matched features as L'. If the condition $L_{new} = L' \neq \Phi$ meets, then the elements in the set L_{new} will be added into the state vector as new features. And the updated state covariance can be calculated with the measurement noise and robot position errors[4].

4 The simulation

Simulation is done mainly on the localization using Matlab. The environment simulated is shown in Fig. 2, where features are presented by the solid line segments, and the dotted line is used to provide steering angles. And the velocity remains 0.4m/s during the whole implementation. Results are shown in Figs. 3 and 4. The errors about robot′s position in the eastern and northern directions are shown in Fig. 3, from which we can see that the maximum error along the X-axis is about 2mm, while along the Y-axis, 5mm. This is probably because few features were available for a full updating at that time. The evolution of the state covariance is shown in Fig. 4 which is convergent.

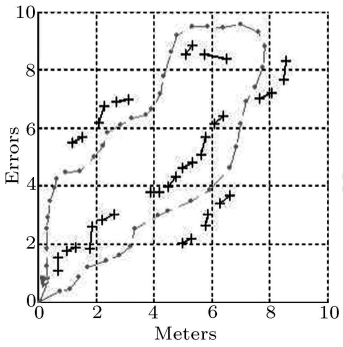

Fig. 2 The simulated environment

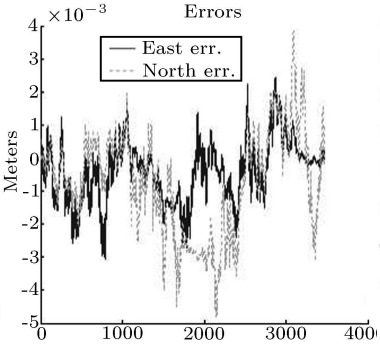

Fig. 3 The errors in X and Y axis

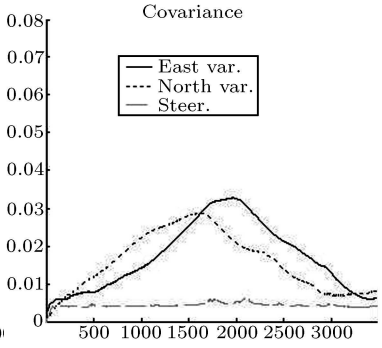

Fig. 4 The state covariance

In the SLAM algorithm, the main killer of the computing sources is the state-updating[1]. Because of the dispersivity of point features, at least 3 point-features which all belong to a line-feature are

needed in order to articulate this line-feature. Contrarily, the presented line-feature based SLAM avoids considering all the points belong to a line in the updating stage, and this is why it is of less computations than the point-feature based one.

5 Conclusion

This paper describes the line-feature based SLAM algorithm in detail and shows the validity and reliability of it according to the simulation. It is proved that the algorithm works well especially in the environment rich of lines or where lines are easily extracted. However, the filter diverges sometimes, because of the incorrect models or accumulated computing errors. So, much finer models should be built and new ways should be developed to make it convergent. Mismatching may happen because during the association we consider each matching between sensor observations and features independently and ignore the fact that measurement prediction errors are correlated. In future work, we should take the joint compatibility into account to find better approach to reject spurious matchings. Codes will be optimized, and the simulation of mapping will be done as well.

References

1 Guivant J, Nebot E. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Journal of Robotic and Automation*, 2001, **17**(3): 242∼257

2 Leonard J, Durrant-Whyte H. Simultaneous map building and localization for an autonomous mobile robot. In: Proceedings of IEEE International Workshop Intelligent Robot System, Osaka, Japan: IEEE/RSJ International Workshop, 1991. 1442∼1447

3 Sack D, Burgard W. A comparison of methods for line extraction from range data. In: Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles, 2004. **1**

4 Smith R, Self M, Cheeseman P. Estimating uncertain spatial relationships in robotics. *Uncertainty in Artificial Intelligence*, New York: Elsevier Science, 1988. **2:** 435∼461

HAN Rui Master student at Wuhan University of Technology. His research interests include simultaneous localization and mapping, multi-sensor information processing and sensor network.

LI Wen-Feng Professor at Wuhan University of Technology. His research interests include Image processing, computer vision, and sensor network.