EKF SALM 算法实践

纪友州

September 29, 2019

摘　要

本篇报告全面说明了 KF，EKF，IEKF 算法的基本原理；构建推导出四轮前驱自主移动机器人的运动学模型和观测模型；详尽全面的解释推导了实验所中使用的 EKF SLAM 算法的系统模型、预测过程、过修正程、已知知 ID 和未知 ID 数据关联算法及状态向量增广的详细过程；基于 EKF SLAM 算法构建了相应的仿真代码，源码已上传 github，可在https://github.com/Nrusher/project1_for_robot_navigation获得；实验内容包括数据生成实验，EKF SLAM 预测与单纯模型预测对比实验，稀疏 landmark 与稠密 landmark 下 EKF SLAM 性能对比实验，预测更新同时进行与非同时进行对比 EKF SLAM 性能对比实验，EKF SLAM 在有色噪声下性能实验，IEKF SLAM 与 EKF SLAM 在高斯噪声和有色噪声下性能对比。

1　EKF 算法原理

EKF 全称 Extended Kalman Filter，即扩展卡尔曼滤波器，一种高效率的递归滤波器 (自回归滤波器)。基本的卡尔曼滤波器在线性系统中有着良好的性能，但在非线性系统中效果不理想，一般适用于线性系统。扩展卡尔曼滤波器通过对非线性系统进行线性化使得卡尔曼滤波器在非线性系统中也可以有良好的性能。

1.1　卡尔曼滤波

对于线性系统：

$$\begin{aligned} x_t &= A_t x_{t-1} + B_t u_t + \varepsilon_t \\ z_t &= C_t x_t + \delta_t \end{aligned} \tag{1}$$

其中 x_t 和 x_{t-1} 是 t 时刻和 $t-1$ 时刻的状态向量，u_t 是时间 t 时刻的控制向量，ε_t 和 δ_t 分别是模型过程噪声和测量噪声，其都假设为均值为 0 的高斯噪声，其方差分别为 R_t, Q_t。对于该线性系统，该卡尔曼滤波算法可表述如下

Algorithm Kalman_filter)$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

predict

$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$

$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

update

$K_t = \overline{\Sigma}_t C_t^T \left(C_t \overline{\Sigma}_t C_t^T + Q_t \right)^{-1}$

$\mu_t = \overline{\mu}_t + K_t \left(z_t - C_t \overline{\mu}_t \right)$

$\Sigma_t = \left(I - K_t C_t \right) \overline{\Sigma}_t$

return μ_t, Σ_t

其中 μ_t 为卡尔曼滤波器对 x_t 的预测，Σ_{t-1} 表示该估计结果的协方差，$\overline{\mu}_t, \overline{\Sigma}_{t-1}$ 则是对 μ_t, Σ_{t-1} 的先验估计，K_t 是卡尔曼滤波算法的中间变量，被称为卡尔曼增益系数。对于一个线性系统而言，使用卡尔曼算法可以较为准确的获得系统的状态估计。

1.2 扩展卡尔曼滤波

考虑更加一般非线性系统：

$$\begin{aligned} x_t &= g\left(u_t, x_{t-1}\right) + \varepsilon_t \\ z_t &= h\left(x_t\right) + \delta_t \end{aligned} \tag{2}$$

上式中的相关变量定义与式1定义相同，$g(.), h(.)$ 为非线性函数。式2其不满足卡尔曼滤波算法对系统必须是线性系统的假设，因而不能对式2使用卡尔曼滤波器。线性系统与非线性系统的差异可见图1和图2。

为对非线性应用卡尔曼滤波算法，可以对该非线性系统进行局部线性化，将线性化点周围的系统状态看做是线性系统，这样对于非线性不太强的系统，可以认为线性化点周围的系统状态是满足卡尔曼滤波算法的条件的。非线性系统线性化的思想可有图2所示

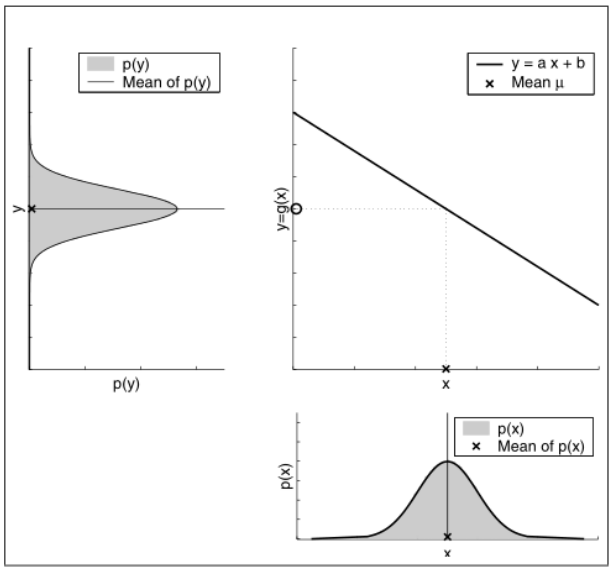

Figure 1: 线性系统

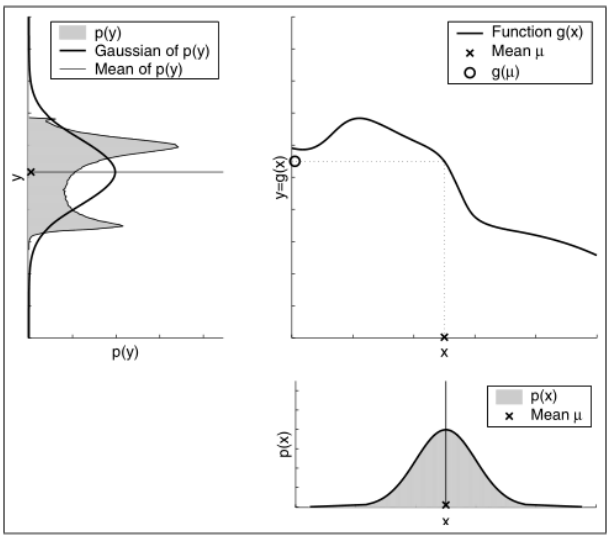

Figure 2: 非线性系统

扩展卡尔曼算法所使用的线性化方法为泰勒展开。对式2中的 $g(u_t, x_{t-1})$ 进行一阶泰勒展开：

$$g'(u_t, x_{t-1}) \quad := \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \tag{3}$$

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=G_t}(x_{t-1} - \mu_{t-1}) \\ &= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \end{aligned} \tag{4}$$

同样对 $h(x_t)$ 进行一阶泰勒展开：

$$h'(x_t) = \frac{\partial h(x_t)}{\partial x_t} \tag{5}$$

$$\begin{aligned} h(x_t) &\approx h(\overline{\mu}_t) + \underbrace{h'(\overline{\mu}_t)}_{=:H_t}(x_t - \overline{\mu}_t) \\ &= h(\overline{\mu}_t) + H_t(x_t - \overline{\mu}_t) \end{aligned} \tag{6}$$

线性化后扩展卡尔曼滤波算法可表述如下

Algorithm Extended_Kalman_filter)$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

predict

$\overline{\mu}_t = g(u_t, \mu_{t-1})$

$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

update

$K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t\right)^{-1}$

$\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$

$\overline{\Sigma}_t = (I - K_t H_t)\overline{\Sigma}_t$

return μ_t, Σ_t

扩展卡尔曼算法扩展了基本的卡尔曼算法的应用范围，使其对非线性系统也具有良好的效果。在本次 SLAM 算法实验便是扩展卡尔曼算法在 SLAM 方面的一个应用实例。

2 IEKF 算法原理

基本的 EKF 算法假设噪声形式为高斯噪声，但实际情况下噪声形式往往为有色噪声形式，并且 EKF 算法的线性化工作点往往不是输入状态真实的均值，而是一个估计的均值，这样的偏差会导致计算出的雅可比矩阵也不是最好的，且线性化过程丢弃了许多高阶项，这会导致 EKF 算法的性能下降，甚至导致滤波器发散。迭代扩展卡尔曼滤波算法 IEKF 能有效的改善这一问题，个人理解其基本的思想与残差逼近类似。IEKF 会将上一步修正后的输出作为下一次迭代修正误差的输入值，通过不断迭代来修正误差减小误差，具体的 IEKF 算法的步骤可表达如下

Algorithm Iterated_Extended_Kalman_filter)$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, times)$

predict

$\overline{\mu}_t = g\left(u_t, \mu_{t-1}\right)$

$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

Iterated update

$\overline{\Sigma}_t^0 = \overline{\Sigma}_t$

$\overline{\mu}_t^0 = \overline{\mu}_t$

for $i = 1$ *to times* do

$K_t^i = \overline{\Sigma}_t^{i-1} H_t^{i-1T} \left(H_t^{i-1} \overline{\Sigma}_t^{i-1} H_t^{i-1T} + Q_t \right)^{-1}$

$\overline{\mu}_t^i = \overline{\mu}_t^{i-1} + K_t \left(z_t - h\left(\overline{\mu}_t^{i-1}\right) \right)$

$\overline{\Sigma}_t^i = \left(I - K_t^i H_t^{i-1}\right) \overline{\Sigma}_t^{i-1}$

endfor

$\mu_t = \overline{\mu}_t^i$

$\Sigma_t = \overline{\Sigma}_t^i$

return μ_t, Σ_t

3 移动机器人模型

本次实验所使用的移动机器人模型如图3所示，为 2 维平面上的自主移动车，其前轮可以进行转向操作，后轮为固定的驱动轮，假设车身中心装有传感器（如激光、声纳、里程计等），传感器固联坐标系与自主移动车辆的本体系一致，可对路标进行观测获得传感数据。车辆状态根据运动学模型定义为 $x_v = [x, y, \varphi]^T$，控制输入量为舵角 γ 和速度 V，相当于实际车辆的方向盘转角和油门。此模型符合实际的汽车模型，因而具有一定的实用性。

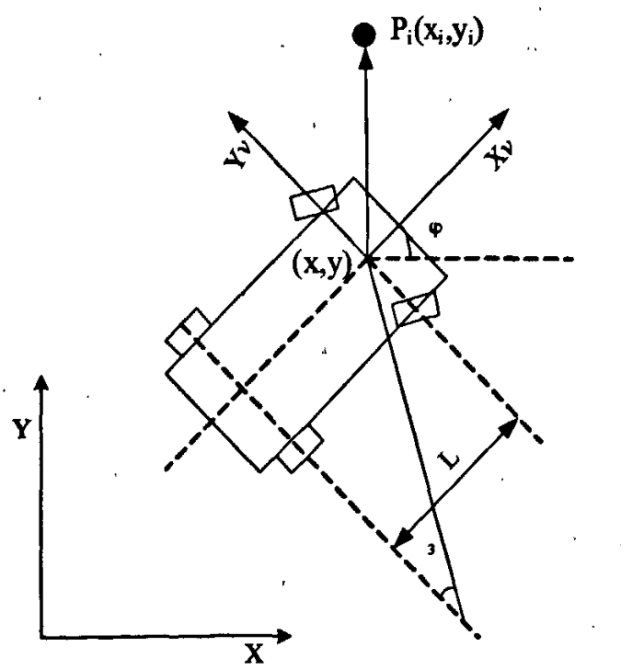

Figure 3: 自主移动车辆模型

3.1 运动模型

为研究方便，可以将图3中的车体模型转换为一长为 L 的连杆，前端两侧车轮的速度归于中心点为 V，既如图4和图5所示，其中5与4差别仅在于舵角的正负问题，故进行了简化。

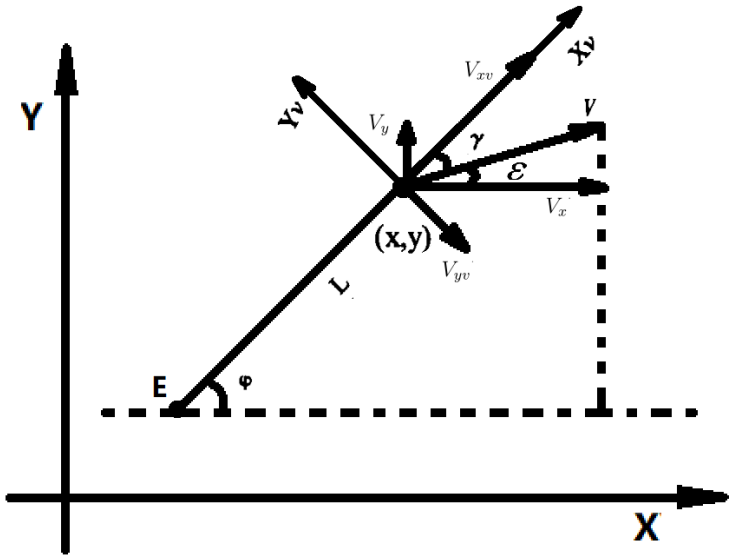

Figure 4: 自主移动车辆简化模型（舵角为负）

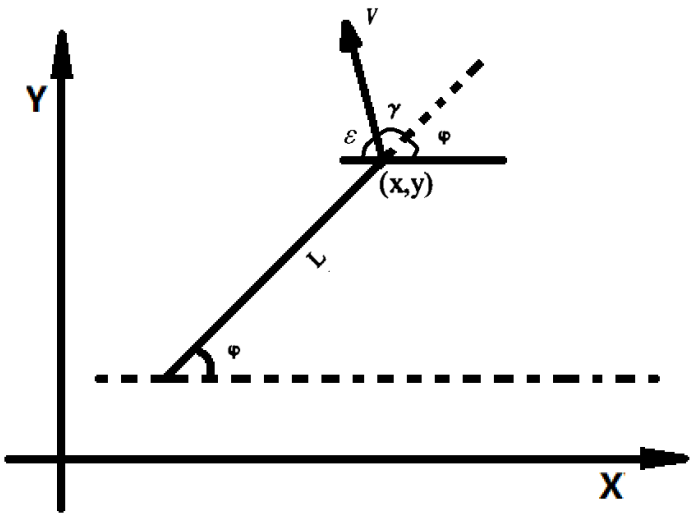

Figure 5: 自主移动车辆简化模型（舵角为正）

以舵角为负时的图4为例，对该自主移动车辆的运动学模型进行建模。
将速度 V 向世界坐标系 $X-Y$ 的 X,Y 方向分解，可得

$$\begin{cases} \dot{x} = V_x = V cos(\epsilon) \\ \dot{y} = V_y = V sin(\epsilon) \end{cases} \tag{7}$$

由图易知

$$\epsilon = \varphi + \gamma \tag{8}$$

既

$$\begin{cases} \dot{x} = V cos(\varphi + \gamma) \\ \dot{y} = V sin(\varphi + \gamma) \end{cases} \tag{9}$$

再将速度 V 向世界坐标系 $X_v - Y_v$ 的 X_v, Y_v 方向分解，可得

$$\begin{cases} V_{xv} = Vcos(\gamma) \\ V_{yv} = Vsin(\gamma) \end{cases} \tag{10}$$

易知，其中 V_{xv} 对车体的姿态角 φ 无影响，V_{yv} 是点 (x, y) 绕点 E 作圆周运动时的线速度，故

$$\dot{\varphi} = \frac{V_{yv}}{L} = \frac{Vsin(\gamma)}{L} \tag{11}$$

综合11，9，可得车体运动模型

$$\begin{cases} \dot{x} = Vcos(\varphi + \gamma) \\ \dot{y} = Vsin(\varphi + \gamma) \\ \dot{\varphi} = \dfrac{Vsin(\gamma)}{L} \end{cases} \tag{12}$$

将其写为离散形式

$$\begin{cases} \dfrac{x(k+1) - x(k)}{\Delta T} = V(k+1)cos(\varphi(k) + \gamma(k+1)) \\ \dfrac{y(k+1) - y(k)}{\Delta T} = V(k+1)sin(\varphi(k) + \gamma(k+1)) \\ \dfrac{\varphi(k+1) - \varphi(k)}{\Delta T} = \dfrac{V(k+1)sin(\gamma(k+1))}{L} \end{cases} \tag{13}$$

进一步的转化为迭代形式，并加入噪声

$$\begin{cases} x(k+1) = x(k) + \Delta TV(k+1)cos(\varphi(k) + \gamma(k+1)) + q_x(k+1) \\ y(k+1) = y(k) + \Delta TV(k+1)sin(\varphi(k) + \gamma(k+1)) + q_y(k+1) \\ \varphi(k+1) = \varphi(k) + \dfrac{\Delta TV(k+1)sin(\gamma(k+1))}{L} + q_\varphi(k+1) \end{cases} \tag{14}$$

既

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \varphi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \varphi(k) \end{bmatrix} + \begin{bmatrix} \Delta TV(k+1)\cos(\varphi(k) + \gamma(k+1)) \\ \Delta TV(k+1)\sin(\varphi(k) + \gamma(k+1)) \\ \frac{\Delta TV(k+1)}{L}\sin(\gamma(k+1)) \end{bmatrix} + \begin{bmatrix} q_x(k+1) \\ q_y(k+1) \\ q_\varphi(k+1) \end{bmatrix} \tag{15}$$

为更加接近实际模型，对最大舵角及其转舵速度进行限制，并且为了仿真便利，将速度 V 设为定值，既进行如下的限制

$$\begin{cases} |\gamma| < MAXG \quad MAXG \in R^+ \\ |\dot{\gamma}| < RATEG \quad RATEG \in R^+ \\ V = V_{rate} \quad V_{rate} \in R^+ \end{cases} \tag{16}$$

以上的公式15和公式16既为本次实验的移动机器人运动学模型。

3.2 观测模型

假设自主移动机器人的传感器为激光雷达，其返回的为路标 i 的相对距离 $z_r^i(k)$ 和方位角 $z_\theta^i(k)$，根据图3其观测模型可表示如下

$$z_i(k) = \begin{bmatrix} z_r^i(k) \\ z_\theta^i(k) \end{bmatrix} = h_i(x(k)) + r_i(k) = \begin{bmatrix} \sqrt{(x_i - x(k))^2 + (y_i - y(k))^2} \\ \arctan\left(\frac{y_i - y(k)}{x_i - x(k)}\right) - \varphi(k) \end{bmatrix} + \begin{bmatrix} r_i^i(k) \\ r_0^i(k) \end{bmatrix} \tag{17}$$

其中 $r_i(k)$ 为观测的噪声向量。为更接近真实的应用场景，另传感器的观测范围仅为车前半径为 MAX_RANGE 半圆内的 landmark。

4 EKF SLAM 算法

4.1 状态变量定义

本次实验采用的是 2D 的 landmark 地图，第 i 个 landmark 的坐标为 $(m_{i,x}, m_{i,y})$; 自主移动机器人有三个状态量 x, y 坐标值和航角 φ，控制量为速度 V 和舵角 γ; 自主移动机器人可以通过激光雷达获取自主移动机器人与第 i 个 landmark 间的相对距离 z_r^i 和相对角度 z_θ^i。

定义系统的状态向量为

$$x_t = (\underbrace{x, y, \varphi}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T \tag{18}$$

定义系统的控制向量为

$$u_t = (V, \gamma)^T \tag{19}$$

定义系统观测向量为

$$z_t = (\underbrace{z_r^1, z_\theta^1}_{\text{landmark 1}}, \ldots, \underbrace{z_r^n, z_\theta^n}_{\text{landmark } n})^T \tag{20}$$

定义系统方程为

$$\begin{aligned} x_t &= g(x_{t-1}, u_t) + +\varepsilon_t \\ z_t &= h(x_t) + \delta_t \end{aligned} \tag{21}$$

根据系统的状态向量 x_t，则 EKF 算法需更新的状态 μ 协方差矩阵 Σ 为

$$\mu = \begin{pmatrix} x \\ y \\ \varphi \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{pmatrix} \tag{22}$$

$$\Sigma = \left[\begin{array}{ccc|ccccc} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \hline \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{m_{1,x}\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{m_{1,y}\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{m_{n,x}\theta} & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{m_{n,y}\theta} & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{array}\right] \tag{23}$$

令 $x_v = (x, y, \varphi)^T, m = (m_{1,x}, m_{1,y}, \ldots, m_{n,x}, m_{n,y})^T$，则式22,23可以简写为分块矩阵形式

$$\mu = \begin{pmatrix} x_v \\ m \end{pmatrix} \tag{24}$$

$$\Sigma = \begin{pmatrix} \Sigma_{x_v x_v} & \Sigma_{x_v m} \\ \Sigma_{m x_v} & \Sigma_{mm} \end{pmatrix} \tag{25}$$

4.2 预测阶段

在 EKF 的预测阶段，根据 EKF 算法，需要按式26更新 $\overline{\mu}_t, \overline{\Sigma}_t$ 的值，由于仿真时采用的 landmark 是静止的，故 $\overline{\mu}_t$ 中的 m_t 及 $\overline{\Sigma}_t$ 中的 Σ_{mm} 不需要更新，仅考虑对 $\overline{x}_v, \overline{\Sigma}_{x_v x_v}, \overline{\Sigma}_{x_v m}, \overline{\Sigma}_{m x_v}$ 的值。

$$\begin{aligned} \overline{\mu}_t &= g\left(u_t, \mu_{t-1}\right) \\ \overline{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + R_t \end{aligned} \tag{26}$$

根据移动机器人的运动模型式16更新 $\overline{x}_v$

$$\begin{aligned} \overline{x}_{v_t} &= g_{x_v}(x_{v_t}, u_t) \\ &= \begin{pmatrix} x(t-1) \\ y(t-1) \\ \varphi(t-1) \end{pmatrix} + \begin{pmatrix} \Delta T V(t) \cos(\varphi(t-1) + \gamma(t)) \\ \Delta T V(t) \sin(\varphi(t-1) + \gamma(t)) \\ \frac{\Delta T V(t)}{L} \sin(\gamma(t)) \end{pmatrix} \end{aligned} \tag{27}$$

由于 landmark 静止，因此系统模型的雅可比矩阵可写为

$$G_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial \mu_{t-1}} = \begin{pmatrix} G_t^{x_v} & 0 \\ 0 & I \end{pmatrix} \tag{28}$$

其中

$$\begin{aligned} G_t^{x_v} &= \frac{\partial g_{x_v}(x_{v_{t-1}}, u_t)}{\partial x_{v_{t-1}}} = \frac{\partial}{\partial (x, y, \varphi)^T} \begin{bmatrix} x + V \cdot dt \cdot \cos(\gamma + \varphi) \\ y + V \cdot dt \cdot \sin(\gamma + \varphi) \\ \varphi + \dfrac{V \cdot dt \cdot \sin(\gamma)}{l} \end{bmatrix} \\ &= I + \frac{\partial}{\partial (x, y, \varphi)^T} \begin{bmatrix} V \cdot dt \cdot \cos(\gamma + \varphi) \\ V \cdot dt \cdot \sin(\gamma + \varphi) \\ \dfrac{V \cdot dt \cdot \sin(\gamma)}{l} \end{bmatrix} \\ &= I + \begin{bmatrix} 0 & 0 & -V \cdot dt \cdot \cos(\gamma + \varphi) \\ 0 & 0 & V \cdot dt \cdot \sin(\gamma + \varphi) \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -V \cdot dt \cdot \cos(\gamma + \varphi) \\ 0 & 1 & V \cdot dt \cdot \sin(\gamma + \varphi) \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{29}$$

根据 EKF 算法,$\overline{\Sigma}_t$ 的更新公式为

$$\begin{aligned} \overline{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + R_t \\ &= \begin{pmatrix} G_t^{x_v} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \Sigma_{x_v x_v} & \Sigma_{x_v m} \\ \Sigma_{m x_v} & \Sigma_{mm} \end{pmatrix} \begin{pmatrix} \left(G_t^{x_v}\right)^T & 0 \\ 0 & I \end{pmatrix} + R_t \\ &= \begin{pmatrix} G_t^{x_v} \Sigma_{x_v x_v} \left(G_t^{x}\right)^T & G_t^{x_v} \Sigma_{x_v m} \\ \left(G_t^{x_v} \Sigma_{x_v m}\right)^T & \Sigma_{mm} \end{pmatrix} + R_t \end{aligned} \tag{30}$$

其中

$$R_t = \begin{bmatrix} R_{x_v} & 0 \\ 0 & 0 \end{bmatrix} \tag{31}$$

$$R_{x_v} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\varphi^2 \end{bmatrix} \tag{32}$$

若已知的是控制量的方差为 $\sigma_v^2, \sigma_\gamma^2$ 则，可以通过下式将控制量的协方差矩阵映射到自主移动机器人的状态变量

$$\begin{aligned} G_t^u &= \frac{\partial g_{x_v}(x_{v_{t-1}}, u_t)}{\partial u_t} = \frac{\partial}{\partial u_t} \begin{bmatrix} x + V \cdot dt \cdot \cos(\gamma + \varphi) \\ y + V \cdot dt \cdot \sin(\gamma + \varphi) \\ \varphi + \dfrac{V \cdot dt \cdot \sin(\gamma)}{l} \end{bmatrix} \\ &= \begin{pmatrix} \dfrac{\partial g_{x_v}(x_v, u)}{\partial v} & \dfrac{\partial g_{x_v}(x_v, u)}{\partial \gamma} \end{pmatrix} \\ &= \begin{pmatrix} dt \cdot \cos(\gamma + \varphi) & -V \cdot dt \cdot \sin(\gamma + \varphi) \\ dt \cdot \sin(\gamma + \varphi) & V \cdot dt \cdot \cos(\gamma + \varphi) \\ \dfrac{dt \cdot \sin(\gamma)}{l} & \dfrac{dt \cdot \cos(\gamma)}{l} \end{pmatrix} \\ R_{x_{v_t}} &= G_t^u \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix} G_t^{uT} \end{aligned} \tag{33}$$

整个预测阶段的算法可表示为

EKF_SLAM_Prediction$(\mu_{t-1}, \Sigma_{t-1}, u_t)$

$$G_t^{x_v} == \begin{bmatrix} 1 & 0 & -V \cdot dt \cdot \cos(\gamma + \varphi) \\ 0 & 1 & V \cdot dt \cdot \sin(\gamma + \varphi) \\ 0 & 0 & 1 \end{bmatrix}$$

$$G_t^u = \begin{bmatrix} dt \cdot \cos(\gamma + \varphi) & -V \cdot dt \cdot \sin(\gamma + \varphi) \\ dt \cdot \sin(\gamma + \varphi) & V \cdot dt \cdot \cos(\gamma + \varphi) \\ \dfrac{dt \cdot \sin(\gamma)}{l} & \dfrac{dt \cdot \cos(\gamma)}{l} \end{bmatrix}$$

$$R_{x_{vt}} = G_t^u \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix} G_t^{uT}$$

$$G_t = \begin{pmatrix} G_t^{x_v} & 0 \\ 0 & I \end{pmatrix}$$

$$R_t = \begin{bmatrix} R_{x_v} & 0 \\ 0 & 0 \end{bmatrix}$$

$$\overline{x}_{v_t} = \begin{pmatrix} x(t-1) \\ y(t-1) \\ \varphi(t-1) \end{pmatrix} + \begin{pmatrix} \Delta TV(t)\cos(\varphi(t-1) + \gamma(t)) \\ \Delta TV(t)\sin(\varphi(t-1) + \gamma(t)) \\ \frac{\Delta TV(t)}{L}\sin(\gamma(t)) \end{pmatrix}$$

$$\overline{\mu}_t = \begin{bmatrix} \overline{x}_{v_t} \\ m_{t-1} \end{bmatrix}$$

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

return $\overline{\mu}_t, \overline{\Sigma}_t$

4.3 更新阶段

更新阶段需要根据传感器获取的数值按式34对 $\overline{mu}_t, \overline{\Sigma}_t$ 进行修正

$$\begin{aligned} K_t &= \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t\right)^{-1} \\ \mu_t &= \overline{\mu}_t + K_t \left(z_t - h\left(\overline{\mu}_t\right)\right) \\ \Sigma_t &= \left(I - K_t H_t\right) \overline{\Sigma}_t \end{aligned} \tag{34}$$

根据自主移动机器人的观测模型式17, 令 $\delta_x = m_{i,x} - x, \delta_y = m_{i,y} - y$, 既分别是第 i 个 landmark 在 X, Y 坐标轴上的坐标差值，再令 $q = \delta_x^2 + \delta_y^2$, 则观测模型可写为

$$\overline{z}^i = \begin{pmatrix} \overline{z}_r^i \\ \overline{z}_\theta^i \end{pmatrix} = h(\overline{\mu}) = \begin{pmatrix} \sqrt{q} \\ \arctan 2(\delta_x, \delta_y) - \varphi \end{pmatrix} \tag{35}$$

计算观测模型的雅可比矩阵 H_t^i

$$\begin{aligned} \overline{\mu}_t &= (x, y, \varphi, m_{i,x}, m_{i,y}) \\ H_t^i &= \frac{\partial h(\overline{\mu}_t)}{\partial \overline{\mu}_t} = \frac{\partial}{\partial \overline{\mu}_t} \begin{pmatrix} \sqrt{\delta_x^2 + \delta_y^2} \\ \arctan 2(\delta_x, \delta_y) - \varphi \end{pmatrix} \\ &= \begin{pmatrix} \dfrac{\partial \sqrt{q}}{\partial x} & \dfrac{\partial \sqrt{q}}{\partial y} & \dfrac{\partial \sqrt{q}}{\partial \varphi} \\ \dfrac{\partial \arctan 2(\delta_x, \delta_y) - \varphi}{\partial x} & \dfrac{\partial \arctan 2(\delta_x, \delta_y) - \varphi}{\partial y} & \dfrac{\partial \arctan 2(\delta_x, \delta_y) - \varphi}{\partial \varphi} \end{pmatrix} \\ &\quad \begin{pmatrix} \dfrac{\partial \sqrt{q}}{\partial m_{i,x}} & \dfrac{\partial \sqrt{q}}{\partial m_{i,x}} \\ \dfrac{\partial \arctan 2(\delta_x, \delta_y) - \varphi}{\partial m_{i,x}} & \dfrac{\partial \arctan 2(\delta_x, \delta_y) - \varphi}{\partial m_{i,x}} \end{pmatrix} \\ &= \frac{1}{q} \begin{pmatrix} -\sqrt{(}q)\delta_x & -\sqrt{(}q)\delta_y & 0 & \sqrt{(}q)\delta_x & \sqrt{(}q)\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix} \end{aligned} \tag{36}$$

在实际观测过程中很多情况下观测的 landmark 不止一个，既当观测到的 landmark 序列为 Z_t, 当前状态向量为 μ_t

$$\begin{aligned} Z_t &= \{z_1, z_2, \ldots, z_m\} \\ \mu_t &= (\underbrace{x, y, \varphi}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T \end{aligned} \tag{37}$$

假设已知观测到的 landmark 序列 Z_t 中的第 i 个观测值对应于 μ_t 中的第 j 个 landmark 状态值 $m_{j,x}, m_{j,y}$, 则可以使用一个数组 c 来表达该映射关系

$$\begin{cases} c[i] = j \quad i \in [1, m] \;\; j \in [1, n] \;\; i, j \in Z^+ & if\ z_i\ can\ be\ found\ in\ \mu_t \\ c[i] = -1 & if\ z_i\ cannot\ be\ found\ in\ \mu_t \end{cases} \tag{38}$$

这里需要注意的是 Z_t 中的每一个 landmark，未必都能找到相应的 $(m_{j,x}, m_{j,y})$, 这种情况发生在移动机器人观测到新的 landmark 时。

定义一个数据关联函数实现该映射关系

$$c = associate(Z_t, \mu_t) \tag{39}$$

在观察到多个 landmark 时，由于 landmark 是静止的，任意 landmark 间无联系，因此对于观测第 i 个 landmark 的雅可比矩阵 H_t^i 而言，其与其它 landmark 相关的偏导相均为 0，既

$$\frac{\partial z_r^i}{\partial m_{k,x}} = 0 \quad \frac{\partial z_r^i}{\partial m_{k,y}} = 0 \quad \frac{\partial z_\theta^i}{\partial m_{k,x}} = 0 \quad \frac{\partial z_\theta^i}{\partial m_{k,y}} = 0 \quad j = c[i] \quad k \neq j \tag{40}$$

由此, 此时的 H_t^i 可以写为

$$\begin{aligned} H_t^i &= \frac{1}{q}\begin{pmatrix} -\sqrt{(}q)\delta_x & -\sqrt{(}q)\delta_y & 0 & \sqrt{(}q)\delta_x & \sqrt{(}q)\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix} F_j \\ F_j &= \begin{pmatrix} 1 & 0 & 0 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 1 & 0 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 0 & 1 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 0 & 0 & 0\ldots0 & 1 & 0 & 0\ldots0 \\ 0 & 0 & 0 & \underbrace{0\ldots0}_{2j-2} & 0 & 1 & \underbrace{0\ldots0}_{2n-2} \end{pmatrix} \end{aligned} \tag{41}$$

由此，整个 EKF 算法的更新过程可以写为如下

EKF_SLAM_Update$(\overline{\mu}_t, \overline{\Sigma}_t, Z_t)$

$c = associate(Z_t, \mu_t)$

$Qt = \begin{bmatrix} \sigma_r & 0 \\ 0 & \sigma_\theta \end{bmatrix}$

for z_i in Z_t and $c[i] \neq -1$ do

j = c[i]

$\delta_x = m_{j,x,t} - x_t, \delta_y = m_{j,y,t} - y_t$

$q = \delta_x^2 + \delta_y^2$

$$F_j = \begin{pmatrix} 1 & 0 & 0 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 1 & 0 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 0 & 1 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 0 & 0 & 0\ldots0 & 1 & 0 & 0\ldots0 \\ 0 & 0 & 0 & \underbrace{0\ldots0}_{2j-2} & 0 & 1 & \underbrace{0\ldots0}_{2n-2} \end{pmatrix}$$

$$H_t^i = \frac{1}{q}\begin{pmatrix} -\sqrt{(}q)\delta_x & -\sqrt{(}q)\delta_y & 0 & \sqrt{(}q)\delta_x & \sqrt{(}q)\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix} F_j$$

$$K_t^i = \overline{\Sigma}_t^i H_t^{iT} \left(H_t^i \overline{\Sigma}_t^i H_t^{iT} + Q_t \right)^{-1}$$

$$\overline{\mu}_t^i = \overline{\mu}_t^i + K_t^i \left(z_i - h\left(\overline{\mu}_t^i\right) \right)$$

$$\overline{\Sigma}_t^i = \left(I - K_t^i H_t^i \right) \overline{\Sigma}_t^i$$

endfor

$\mu_t = \overline{\mu}_t^i$

$\Sigma_t = \overline{\Sigma}_t^i$

return μ_t, Σ_t

4.4 数据关联

在 EKF_SLAM 的更新阶段，需要对观测到的 landmark 和状态向量中的 landmark 相匹配，这一过程被称为数据关联。在本实验的数据关联分为两种，一种是观测时数据自带唯一身份标签的，例如在 landmark 上贴上二维码，使用相机读取二维码便可唯一确定 ID。这种情况下很容易实现数据关联，其中一种实现方式是，维护一个表示 μ_t 中 landmark ID 的数组即可，记该素组为 $\mu\_ID, \mu\_ID[i]$ 表示 μ_t 第 i 个 landmark，也就是坐标为 $(m_{i,x}, m_{i,y})$ 的 landmark ID，记观测到的 landmark 的 ID 的数组为 ZID，则数据关联算法可写为

Associate(μ_ID, Z_ID)
  if find $Z\_ID[i] = \mu\_ID[j]$
    $c[i] = j$
  else
    $c[i] = -1$
  return c

这里的查找算法可以使用遍历，更快的可以使用 HASH 算法进行查找操作。除了维护上述的表外，在 landmark 数目不多，存储资源富裕的情况下可以直接维护一个表达所有 landmark 在 μ_t 中位置的 $table$,$table[id]$ 代表的是 ID 为 id 的 landmark 在 μ_t 中的位置，$table[id] = 0$ 代表该 landmark 未在状态向量中出现，在进行状态向量增广时更新添加到状态向量中新的 landmark id 对应的 $table[id]$ 便可以方便的维护 $table$ 数组。此时的关联的算法可写为

Associate$(table, Z\_ID)$
  for id in Z_ID
    if $table[id] == 0$
      $c[i] = -1$
    else
    $c[i] = table[id]$
  return c

另外一种情况是观测时数据不带身份标签，这时一个可行的数据关联算法是根据观测到的数据与已知的 landmark 间的相似程度来确定。一种简单的方法是直接使用欧式距离进行度量，既计算观测到的 landmark 坐标与所有已知 landmark 间的距离，求出最小距离，若最小距离小于某一阈值，则判定其为最小距离对应的 landmark，若最小距离大于某一阈值则判定其为一新观测到的 landmark，否则丢弃这个观测到的值。其过程可以如下表示

Associate$(\mu, Z, threshold\_old, threshold\_new)$

$$\mu(lamdmark_j) = \begin{pmatrix} m_{j,x} \\ m_{j,y} \end{pmatrix}$$

$$m_j = \text{observe\_model\_conversion}(m_{j,x}, m_{j,y}) = \begin{pmatrix} m_{j,r} \\ m_{j,\theta} \end{pmatrix}$$

$$z_i = \begin{pmatrix} z_r \\ z_\theta \end{pmatrix}$$

$M = \{m_1, m_2, \ldots, m_n\}$
for i from 1 to $length(Z)$ do
  for j from 1 to $length(M)$ do
    $dis[j] = \sqrt{(z_i - m_j)^T (z_i - m_j)}$
  $j_{min} = \arg\min_j dis[j]$
  $min\_dis = dis[j_{min}]$
  if $min\_dis < threshold\_old$
    $c[i] = j$
  else if $min - dis > threshold\_new$
    $c[i] = -1$
  else
    $c[i] = droped$ 后续不再考虑该观测值
return c

但由于噪声影响和距离与角度的量纲不一致的影响，这样简单的欧式距离度量很容易出错，为了解决这一问题可以使用马氏距离代替欧式距离，利用起表示不确定性的协方差矩阵，使其相似性度量更加合理，马氏距离定义如下

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})} \tag{42}$$

其中 $\vec{x}, \vec{y}$ 属于同一分布，S 表示该分布的协方差矩阵。利用马氏距离可以有效的避免尺度影响并考虑到了噪声，相对欧式距离更加合理可靠。由此以上的数据关联算法可以写为以下形式

Associate$(\mu, Z, \Sigma, Q, threshold\_old, threshold\_new)$
$\mu(lamdmark_j) = \begin{pmatrix} m_{j,x} \\ m_{j,y} \end{pmatrix}$
$m_j = \text{observe\_model\_conversion}(m_{j,x}, m_{j,y}) = \begin{pmatrix} m_{j,r} \\ m_{j,\theta} \end{pmatrix}$
$z_i = \begin{pmatrix} z_r \\ z_\theta \end{pmatrix}$
$M = \{m_1, m_2, \ldots, m_n\}$
for i from 1 to $length(Z)$ do
  for j from 1 to $length(M)$ do
    $H_t^j = \frac{1}{q} \begin{pmatrix} -\sqrt{(}q)\delta_x & -\sqrt{(}q)\delta_y & 0 & \sqrt{(}q)\delta_x & \sqrt{(}q)\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix}$
    $S = H_t^j \Sigma H_t^{jT} + Q$ H_t^j 为状态向量中第 i 个 landmark 的雅可比矩阵
    $dis[j] = \sqrt{(z_i - m_j)^T S^{-1} (z_i - m_j)}$
  $j_{min} = \arg\min_j dis[j]$
  $min\_dis = dis[j_{min}]$
  if $min\_dis < threshold\_old$

$c[i] = j$
else if $min - dis > threshold\_new$
$c[i] = -1$
else
$c[i] = droped$ 后续不再考虑该观测值
return c

4.5 状态向量增广

所谓的状态向量增广就是将新观测到的 landmark 信息添加到状态向量及协方差矩阵中，改变 μ, Σ 的维度。若一个新观测到的 landmark 的观测值为 z，其对应的全局坐标为 $m = (m_x, m_y)^T$, 此时的机器人位姿为 $x_v = (x, y, \varphi)^T$, 则根据观测模型可以写出 z 到 m 的转换为

$$m = f(z, x_v) = \begin{pmatrix} m_x \\ m_y \end{pmatrix} = \begin{pmatrix} x + z_r \sin(\varphi + z_\theta) \\ y + z_r \cos(\varphi + z_\theta) \end{pmatrix} \tag{43}$$

对其线性化，求雅可比矩阵为

$$F_{x_v} = \frac{\partial f(z, x_v)}{\partial x_v} = \begin{pmatrix} 1 & 0 & -z_r \cos(\varphi + z_\theta) \\ 0 & 1 & z_r \sin(\varphi + z_\theta) \end{pmatrix} \tag{44}$$

$$F_z = \frac{\partial f(z, x_v)}{\partial z} = \begin{pmatrix} cos(\varphi + z_\theta) & -z_r \cos(\varphi + z_\theta) \\ sin(\varphi + z_\theta) & z_r \sin(\varphi + z_\theta) \end{pmatrix} \tag{45}$$

对状态向量的增广只需要简单的将根据式43解算出的 landmark 坐标拼接到状态向量中既可。对协方差矩阵进行更行时需要更新 Σ 中与该 landmark 相关的部分 $\Sigma_{x_v m}, \Sigma_{m x_v}, \Sigma mm$，该部分的协方差矩阵可以通过上述求得的雅可比矩阵 F_{x_v} 和 F_z 将观测模型的协方差矩阵和机器人位姿的协方差矩阵转换到其对应的相关系数。若一个新观测到的 landmark 的观测值为 z, 则利用其加新状态向量的过程如下所示

Augmented$(\mu_{k-1}, \Sigma_{k-1}, z)$

$$m = \begin{pmatrix} m_x \\ m_y \end{pmatrix} = \begin{pmatrix} x + z_r \sin(\varphi + z_\theta) \\ y + z_r \cos(\varphi + z_\theta) \end{pmatrix}$$

augment μ

$$\mu_k = \begin{pmatrix} \mu_{k-1} \\ m \end{pmatrix}$$

$$\Sigma_{k-1} = \left(\begin{array}{c|c} \Sigma_{x_v x_v,k-1} & \Sigma_{x_v m,k-1} \\ \hline \Sigma_{x_v x_v,k-1} & \Sigma_{mm,k-1} \end{array}\right)$$
$$F_{x_v} = \begin{pmatrix} 1 & 0 & -z_r\cos(\varphi + z_\theta) \\ 0 & 1 & z_r\sin(\varphi + z_\theta) \end{pmatrix}$$
$$F_z = \begin{pmatrix} cos(\varphi + z_\theta) & -z_r\cos(\varphi + z_\theta) \\ sin(\varphi + z_\theta) & z_r\sin(\varphi + z_\theta) \end{pmatrix}$$
$$P^4_{mm} = F_{x_v}\Sigma_{x_v x_v,k-1}F^T_{x_v} + F_z Q F^T_z$$
$$P^2_{mm} = F_{x_v}\Sigma_{x_v m,k-1}$$
$$P^3_{mm} = {P^2_{mm}}^T$$
$$P_{x_v m} = F_{x_v}\Sigma_{x_v x_v,k-1}$$
$$P_{m x_v} = P^T_{x_v m}$$

augment Σ

$$\Sigma_k = \left(\begin{array}{c|cc} \Sigma_{x_v x_v,k-1} & \Sigma_{x_v m,k-1} & P_{x_v m} \\ \hline \Sigma_{x_v x_v,k-1} & \Sigma_{mm,k-1} & P^2_{mm} \\ P_{m x_v} & P^3_{mm} & P^4_{mm} \end{array}\right)$$

return μ_k Σ_k

5 MATLAB 代码实现

5.1 代码结构

实验的代码编写结构如下

```
/
├── data 存放实验数据
│   └── map1.mat 地图及数据
├── pictures 实验结果图
└── functions 相关函数
    ├── EKF_SLAM_core_functions EKF SLAM 算法相关函数
    │   ├── augment.m 增广状态向量
    │   ├── data_associate.m 数据关联
    │   ├── data_associate_known.m 数据关联 (id 未知)
    │   ├── KF_cholesky_update.m cholesky KF 更新
    │   ├── KF_joseph_update.m joseph KF 更新
    │   ├── KF_simple_update.m 简单 KF 更新
    │   ├── KF_IEKF_update.m 迭代更新
    │   ├── EKF_predict.m EKF 预测
    │   ├── EKF_update.m EKF 更新
    │   └── update_iekf.m IEKF 更新
    ├── models 运动模型及观测模型
    │   ├── observe_model.m 观测模型
    │   └── vehicle_model.m 自主移动机器人模型
    └── data_gen_functions 产生数据用到的相关函数
```

```
        ├── gen_color_noise.m 产生有色噪声
        ├── add_control_noise.m 添加控制噪声
        ├── add_observation_noise.m
        ├── compute_steering.m 根据控制点计算舵角
        ├── get_observations.m 获取可以观测到的 landmark
        ├── pi_to_pi.m 角度限制函数
        └── transformtoglobal.m 坐标变换
    └── draw_functions 画图函数
        ├── line_plot_conversion.m 生成直线数据（画激光线用）
        ├── make_covariance_ellipses.m 画方差性椭圆
        ├── make_ellipse.m 画椭圆
        ├── make_laser_lines.m 画激光线
        ├── draw_car.m 画出车
        └── draw_circle.m 画圆
├── gendata.m 用来生成仿真用的地图及数据的函数
├── gen_trajectory2d.m 用来产生地图关键点
├── EKF_SLAM_simulation.m 仿真脚本
├── analysis.m 仿真结果分析
└── simulation_config.m 仿真参数配置文件
```

代码使用方法

1. 运行 EKF_SLAM_simulation.m 仿真脚本，查看仿真过程。
2. 运行 analysis.m 脚本分析仿真结果

备注：代码中 KF_cholesky_update.m，KF_joseph_update.m ，KF_simple_update.m 是三种不同的对矩阵求逆的处理方法，其中前两者的数值稳定性更好。

完整源码可在https://github.com/Nrusher/project1_for_robot_navigation获得

5.2 数据生成代码实现

为方便实验结果的对比，采取先生成数据，再在数据上进行仿真的方法。数据生成时首先会生成一些路径上的关键点，再利用移动机器人的运动模型，通过计算舵角，规划出一条经过这些关键点的曲线，在此过程中收集仿真需要的数据。其基本流程如图6所示

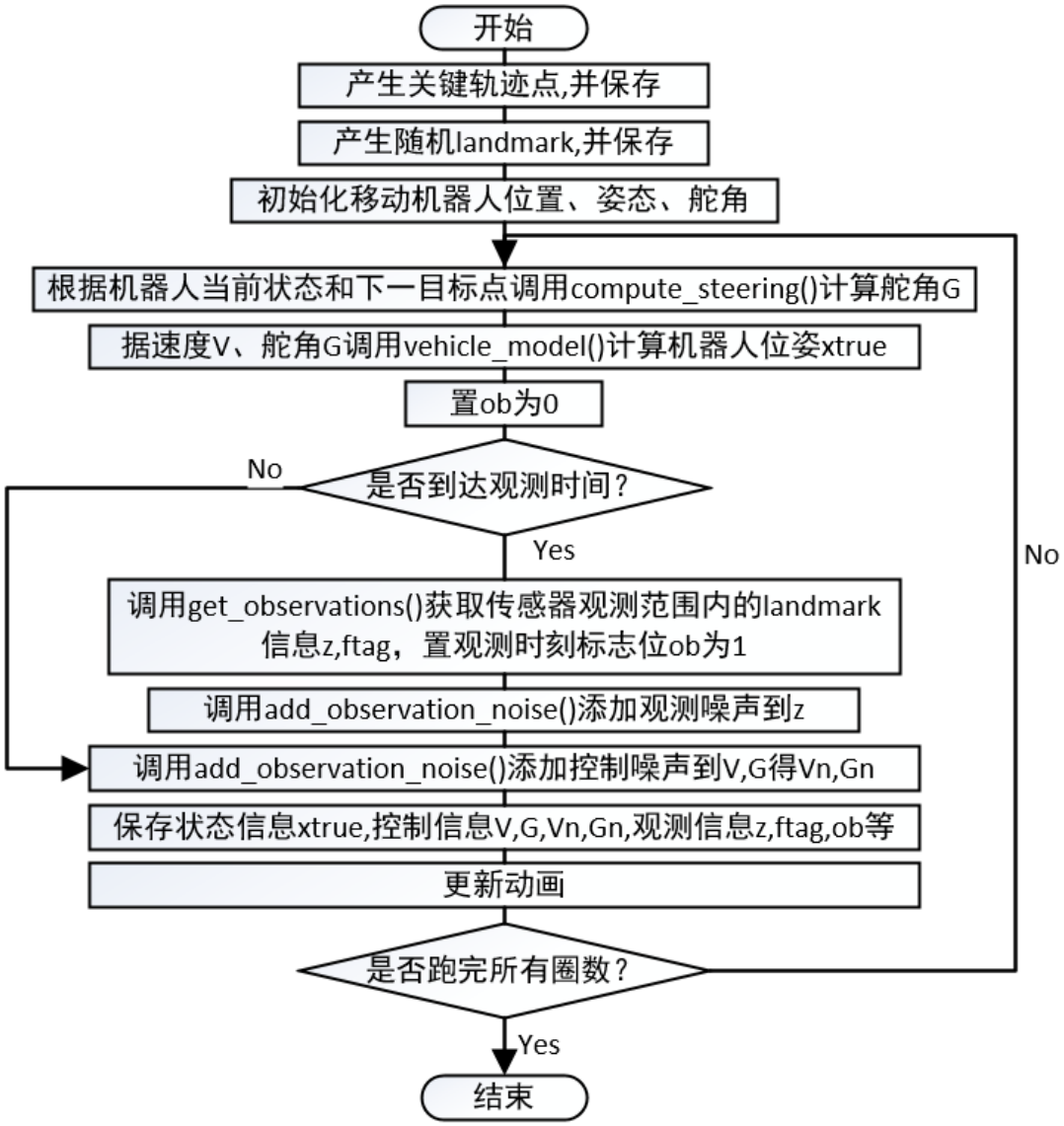

Figure 6: 数据生成流程图

5.3 EKF_SLAM 仿真代码实现

EKF SLAM 仿真的基本流程如图7所示

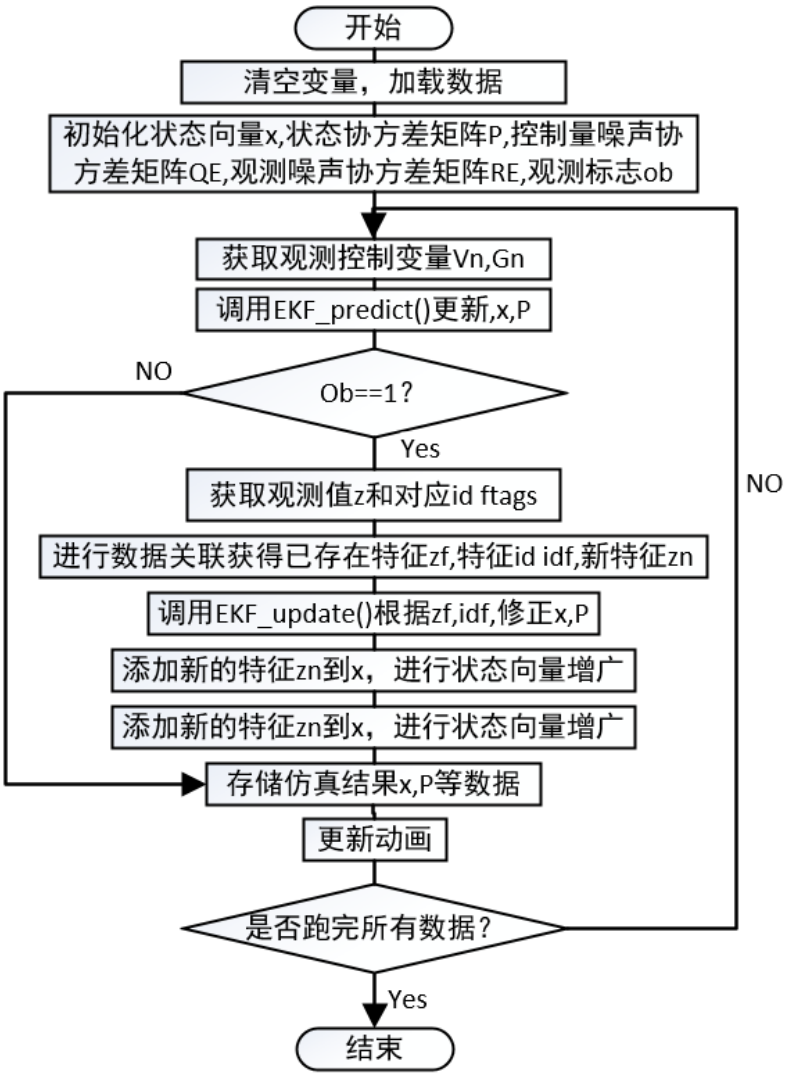

Figure 7: EKF-SLAM 仿真流程图

6 实验结果及分析

6.1 数据实验生成结果

数据生成时自主移动机器人的部分参数为:

- 轮距 $L = 4m$
- 行驶速度 $V = 8m/2$
- 最大舵角 $MAXG = 30°$
- 最大转向速度 $RATEG = 20°/s$
- 控制周期 $DT\_CONTROLS = 0.025s$
- 观测周期 $DT\_OBSERVE = 8 \times 0.025s$

- 最大观测距离 $MAX\_RANGE = 30m$

图22为数据生成的实验过程图，X, Y 轴的单位为米。其中蓝色星号代表 landmark；红色圆点代表路径上的关键点；外围有蓝色圆圈的为移动机器人的下一目标点；黑色虚线代表移动机器人的真实轨迹；黑色圆代表移动机器人，其中心的粉色圆点为当前机器人位置，粉色圆点与黑色圆边上点的连线代表机器人目前航向；有与粉色点连线的 landmark 代表其被机器人传感器观测到；黄色圆代表机器人传感器的观测半径，但传感器仅观测机器人正前方 180 度范围内的 landmark。

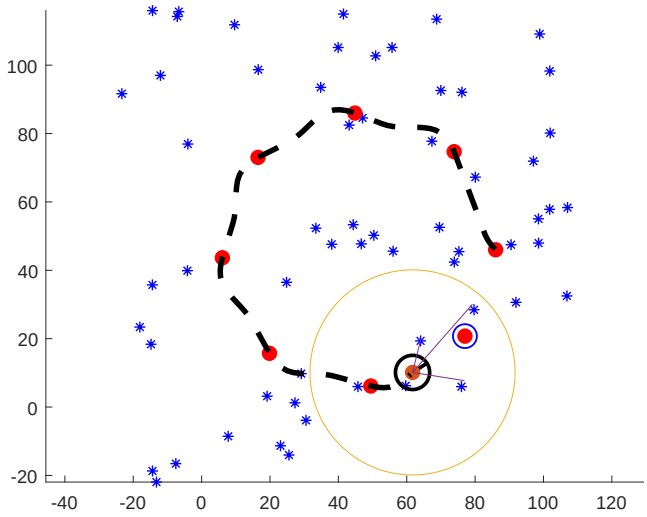

Figure 8: 数据生成

采集的数据格式如下

- key_points - 地图关键点
- lamdmarks - landmark 的坐标
- states(i).G - 真实舵角
- states(i).V - 真实速度
- states(i).Gn - 带有噪声的舵角
- states(i).Vn - 带有噪声的速度
- states(i).xtrue - 真实的位姿
- states(i).ftag_visible - 可观测到的 landmark
- states(i).ztrue - 真实的 landmark 观测值
- states(i).zn - 带有噪声的 landmark 观测值
- states(i).observation_update - 是否进行了观测数据更新
- states(i).next_keypoint - 下一目标点

6.2 EKF SLAM 预测与单纯模型预测对比

仿真的噪声参数参数如下（下文未作说明时，噪声参数于此相同）

- 速度噪声方差 $\sigma_V^2 = 2^2 m$
- 舵角噪声方差 $\sigma_G^2 = 10^{2^\circ}$
- 观测相对距离噪声方差 $\sigma_r^2 = 1^2 m$
- 观测相对角度噪声方差 $\sigma_\theta^2 = 1^{2^\circ}$

仿真结果如下图所示

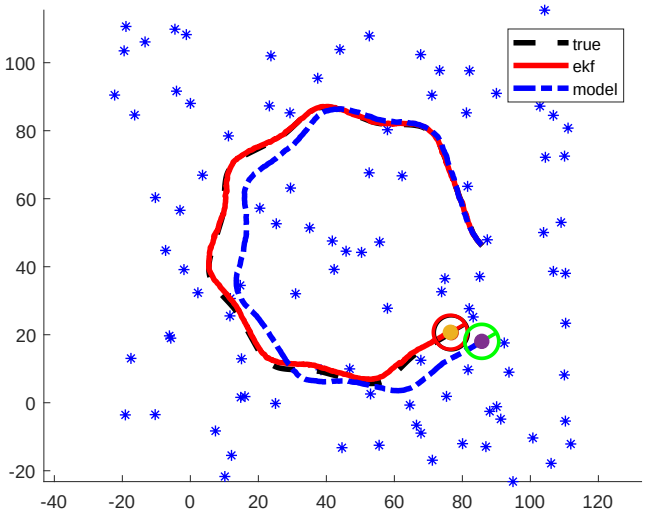

Figure 9: EKF SLAM 预测与单纯模型预测轨迹

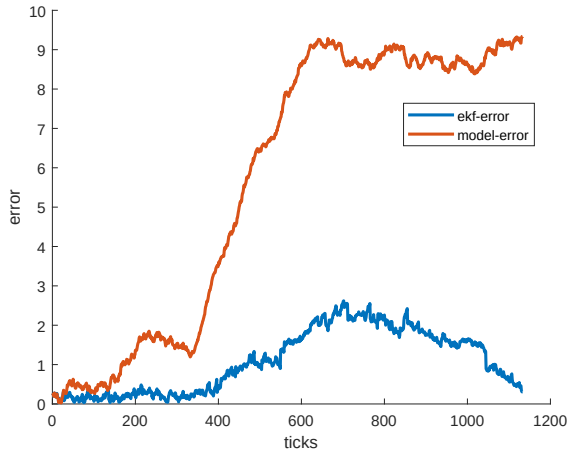

Figure 10: EKF SLAM 预测与单纯模型预测二范数误差

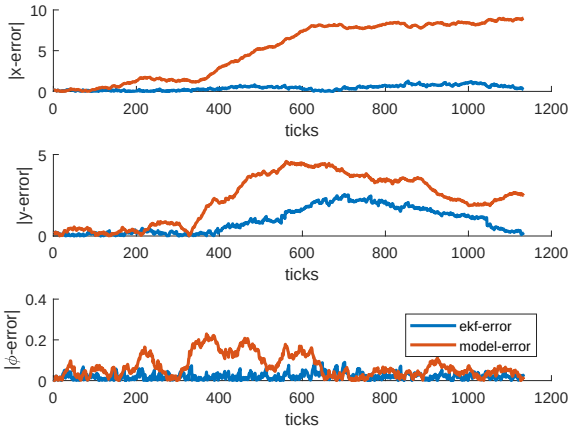

Figure 11: EKF SLAM 预测与单纯模型预测 x, y, φ 误差

可以直观的发现 EKF SLAM 的预测精度明显高于单纯的模型预测的精度，且随时间增长越来越明显。

6.3 稀疏 landmark 与稠密 landmark 对比

为对比观测到的 landmark 数目多少对 EKF SLAM 性能的影响，将移动机器人观测到的 landmark 数量一直保持在一个，将该情况下的轨迹预测结果与正常的仿真结果对比，实验结果如下所示

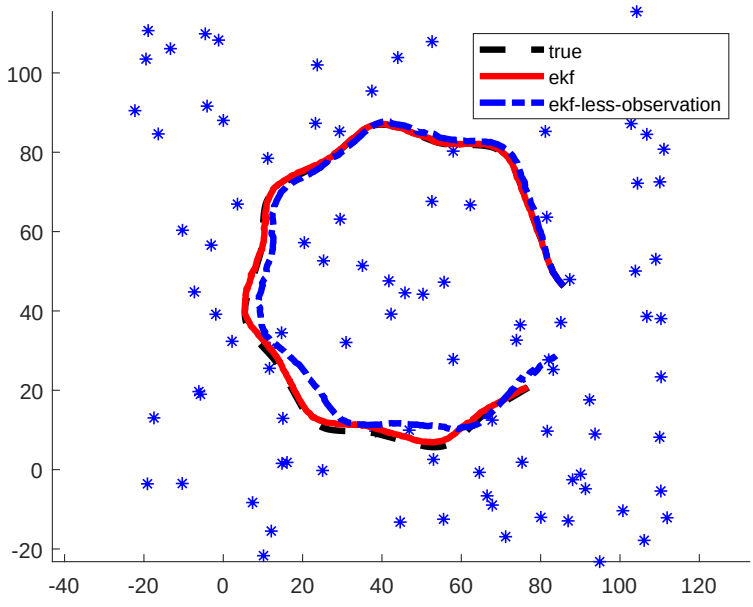

Figure 12: 较多 landmark 与较少 landmark 下 EKF SLAM 预测轨迹

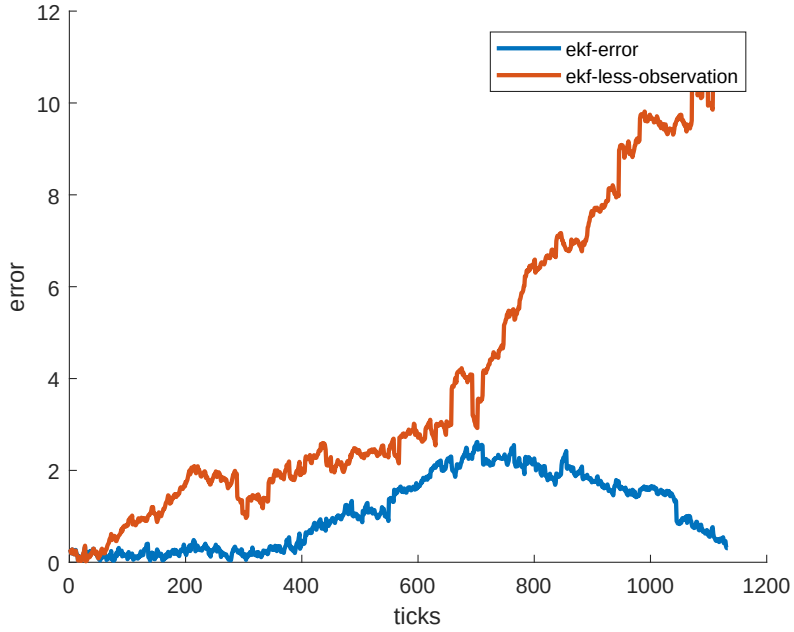

Figure 13: 较多 landmark 与较少 landmark 下 EKF SLAM 二范数误差

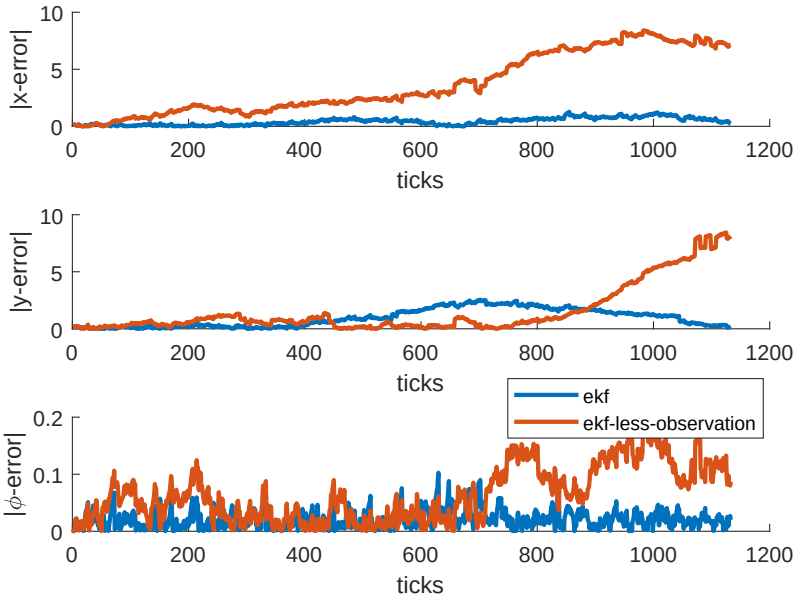

Figure 14: 较多 landmark 与较少 landmark 下 EKF SLAM x, y, φ 误差

由上图可以发现在观测到的 landmark 数目较多的时候 EKF SLAM 的性能要比观测到的 landmark 数目较少时好。

6.4 预测更新同时进行与非同时进行对比

在时间情况下由于控制和测量的周期往往有差异，在本实验中获取 landmark 观测周期是控制周期的 8 倍，该实验验证的是 EKF SLAM 预测过程和更新修正过程异步（也就是在每个控制周期都执行预测过程，仅在获得观测值时进行更新修正过程）与预测过程和更新修

正过程同步（也就是仅在获得观测值时进行预测过程和更新修正过程）下的性能区别。实验结果如下图所示

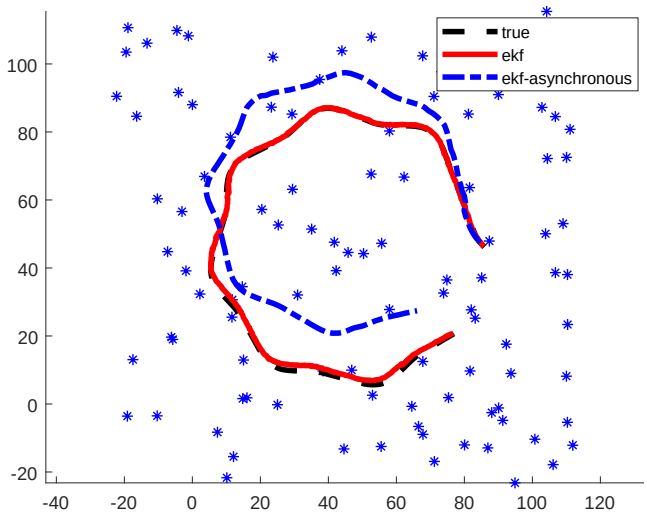

Figure 15: 同步与非同步下的 EKF SLAM 轨迹

从图中可以发现异步的 EKF SLAM 算法性能明显优于同步的 EKF SLAM 算法，个人认为，一方面这是由于同步的预测过程的更新周期远高于异步的，导致预测值本身不够准确，另一方面是因为由于更新频率低，很多控制信息被丢到，使得算法对协方差矩阵的估计也不够准确。为充分利用信息，因使用异步 EKF SLAM 算法更好。

6.5 EKF SLAM 在有色噪声下性能

实验采用的有色噪声序表达式为

$$e_{color}(k) = e_{white}(k) + e_{white}(k-1) * 0.8 - e_{white}(k-2) * 0.6$$

其中 e_{white} 为一白噪声序列。速度的 e_{white} 噪声方差为 $\sigma_V^2 = 2^2 m$ 舵角的 e_{white} 噪声方差为 $\sigma_G^2 = 10^{2^\circ}$。在该噪声序列下 EKF SLAM 的性能如下图所示

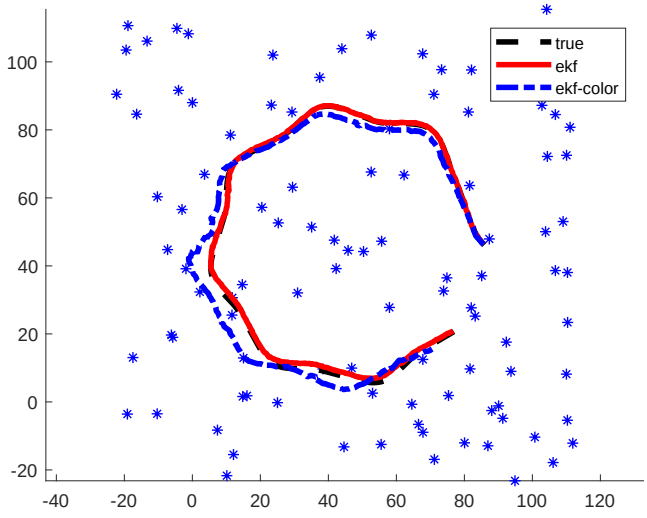

Figure 16: 高斯噪声与有色噪声下 EKF SLAM 轨迹

由上图可以发现 EKF SLAM 算法在噪声为有色噪声时，性能有较为明显的下降。

6.6 IEKF SLAM 与 EKF SLAM 性能对比

在噪声为高斯噪声情况下 5 次迭代的 IEKF 与 EKF 算法性能对比

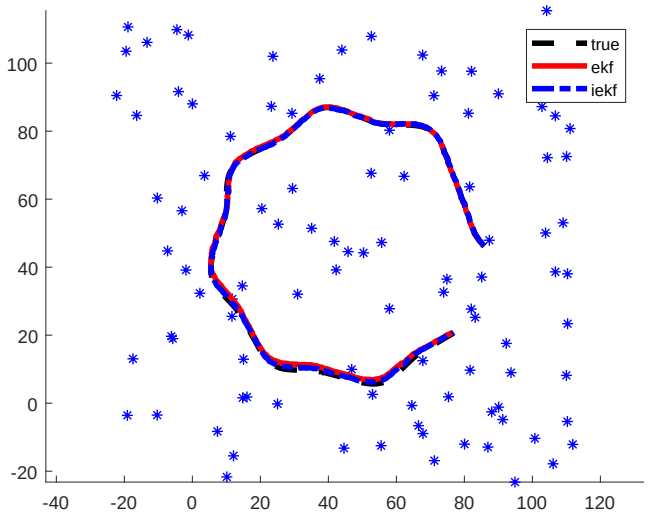

Figure 17: 高斯噪声下 EKF 与 IEKF 算法预测轨迹

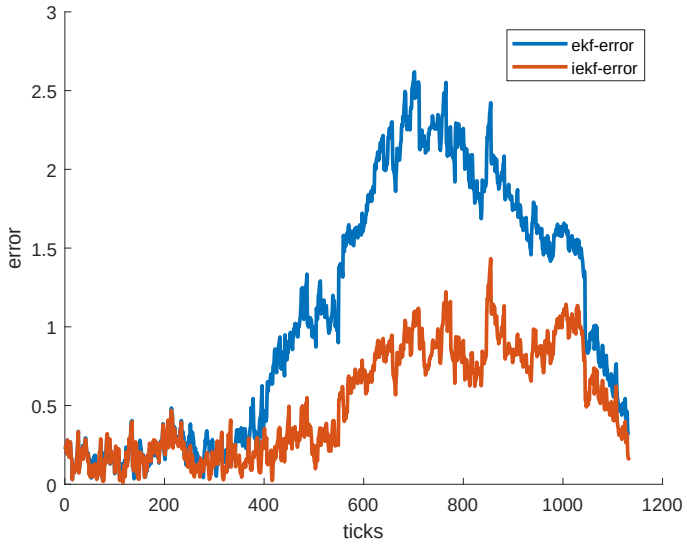

Figure 18: 高斯噪声下 EKF 与 IEKF 算法二范数误差

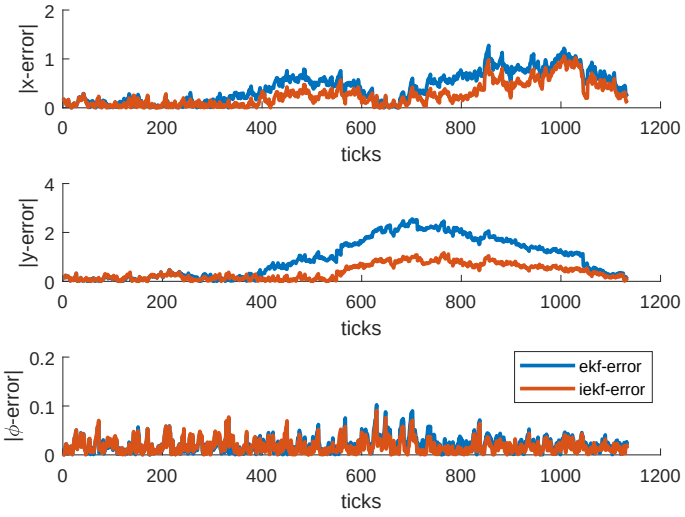

Figure 19: 高斯噪声下 EKF 与 IEKF 算法 x, y, φ 误差

在噪声为有色噪声情况下 5 次迭代 IEKF 与 EKF 算法性能对比（有色噪声同上一节）

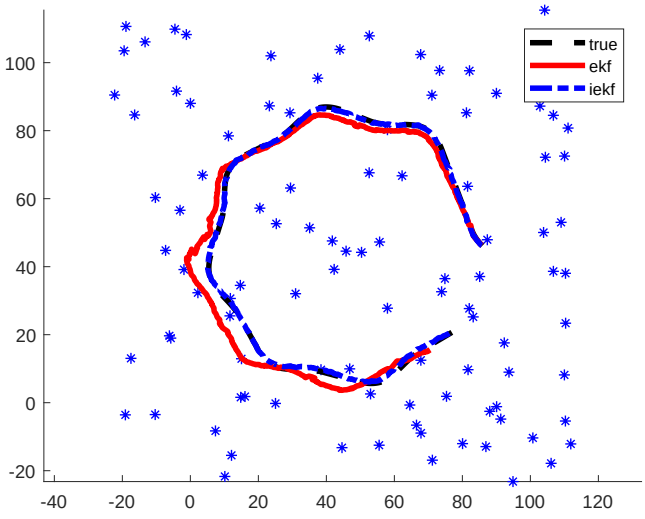

Figure 20: 有色噪声下 EKF 与 IEKF 算法预测轨迹

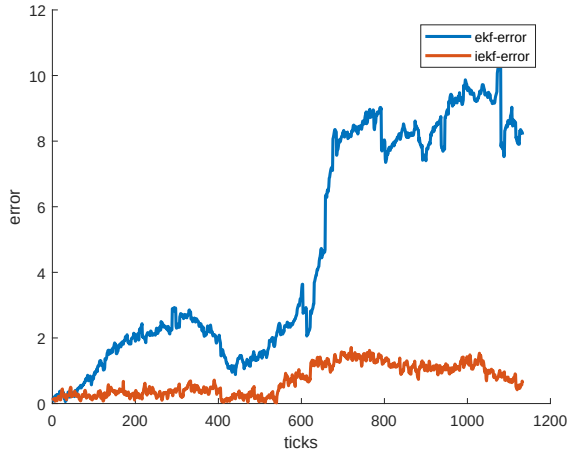

Figure 21: 有色噪声下 EKF 与 IEKF 算法二范数误差

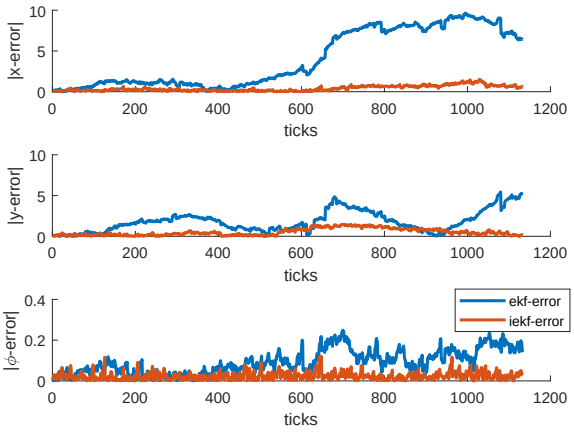

Figure 22: 有色噪声下 EKF 与 IEKF 算法 x, y, φ 误差

7 结论

1. EKF SLAM 算法的定位精度相对于单纯的依靠传感器和模型预测明显高出许多。
2. EKF SLAM 算法在获得更多的观测 landmark 数目有助于其精度的提高，但同时也会带来计算量的上升，降低其运算速度，需要对二者进行取舍。
3. EKF SLAM 算法的预测和修正过程不比同时进行，可根据实际的传感器数据更新频率分开操作，甚至对状态向量和协方差矩阵部分量进行预测和修正，可以很好的协调各个更新频率不同的传感器，具有很好的实际可行性。
4. EKF SLAM 算法在有色噪声下其性能会有所下降。
5. IEKF SLAM 算法无论是在高斯噪声下还是在有色噪声下，其精度都明显高于 EKF SLAM 算法，在有色噪声情况下其任然可以保持很好的性能。但由于 IEKF 需要迭代运行，对其速度有一定影响，需要在精度和实时性二者间做平衡。

8 参考文献

References

[1] 强敏利，张万绪. IEKF 滤波在移动机器人定位中的应用. 39(2), page 74-77 2012.

[2] Thrun S. Probabilistic robotics[M]. 2006.

[3] 李久胜, 李永强, 周荻, et al. 基于 EKF 的 SLAM 算法的一致性分析 [J]. 计算机仿真, 2008, 25(6):155-160.

[4] Tim Bailey EKF SLAM CODE 2004

[5] 相关网络 blog