# 多源信息滤波与融合
# 课程设计报告

## 题目：基于 EKF 的 SLAM 方法研究


学　　院：　　自动化

专业名称：导航、制导与控制

学生姓名：　　樊振辉

学　　号：　　2120170888

指导教师：　邓志红、戴亚萍


二〇一八年五月

# 摘要

卡尔曼滤波（KF）是一种利用线性系统状态方程，通过输入输出的观测数据对系统状态进行估计的技术，扩展卡尔曼滤波（EKF）是一种在卡尔曼滤波技术基础上，针对非线性系统提出信息融合方法，其可以利用系统的状态方程与观测方程对系统的状态进行最优估计。

即时定位与地图构建（SLAM）是导航领域近几年中快速发展的技术之一，它解决机器人在未知环境中的定位与地图构建问题。机器人智能化是未来发展趋势，整体可以分为环境感知、定位与构图、路径规划三部分，分别解决“我在那儿”、“我要到哪里去”、“如何去”的问题，SLAM 作环境感知与路径规划的链接过程发挥着不可替代的作用。由于单一传感器的误差及环境局限性，SLAM 过程往往融合多种传感器的信息进行位置估计。

摄像头作为视觉传感器可以提取丰富的环境信息，而编码器作为里程计传感器可以对机器人进行精确的航位推算，融合这两种信息可以实现机器人的局部定位，并记录自身的运行轨迹，相比采用单一传感器在定位精度与定位方式上有所提高。EKF 作为经典的信息融合算法在一些工程应用领域有着成熟的应用成果，本文采用 EKF 算法融合摄像头与编码器信息，利用摄像头观测标签计算相对位置，利用编码器计算里程信息，并采用 Matlab 进行仿真分析。

**关键词**：扩展卡尔曼滤波、即时定位与地图构建、信息融合

# Abstract

Kalman filter (KF) is a technique that uses the state equation of the linear system to estimate the state of the system through observations of the input and output. The extended Kalman filter (EKF) is based on the Kalman filter technique and is aimed at nonlinearity. The system proposes an information fusion method that can use the system's state equations and observation equations to make an optimal estimation of the state of the system.

Instant positioning and map building (SLAM) is one of the fast-growing technologies in the navigation field in recent years. It solves the problem of robot positioning and map construction in an unknown environment. Intelligent robotics is the future development trend. The whole can be divided into three parts: environment perception, positioning and composition, and path planning. They can solve the problems of “I'm there,” “Where do I go,” and “How to go?” The linking process between perception and path planning plays an irreplaceable role. Due to the error of single sensor and environmental limitations, SLAM process often fuses the information of multiple sensors to estimate the position.

The camera as a visual sensor can extract a wealth of environmental information, and the encoder can be used as an odometer sensor for accurate dead reckoning of the robot. Fusion of these two kinds of information can achieve local positioning of the robot and record its own running trajectory. A single sensor has improved positioning accuracy and positioning. EKF, as a classical information fusion algorithm, has mature application results in some engineering applications. This paper uses EKF algorithm to fuse camera and encoder information, uses the camera to measure the relative position of the tag, uses the encoder to calculate mileage information, and uses Matlab to perform simulation analysis.

**Keywords**: Extended Kalman Filter, Simultaneous Localization And Mapping, Information Fusion

# 目录

# 第 1 章 绪论

## 1.1 研究背景与现状

卡尔曼滤波技术（KF）基于美国学者 R．E．Kalman 和 R．S．Bucy 在 1961 年提出的线性滤波理论[1]，经过 NASA 在阿波罗计划轨道预测项目与火星探测器的成功应用，得到世界各地学者的广泛研究，并作为信息融合的经典技术在导航领域[2]、目标跟踪[3][4]等领域有着广泛的应用。

KF 技术主要应用于信息融合，其假定难以直接测量的状态噪声与观测噪声服从高斯分布，根据线性系统的状态方程以及传感器的观测方程，不断循环递推得到系统当前状态的最优估计，并且 Kalman 从数学角度证明了该过程的合理性。经过几代学者们的不懈研究，在卡尔曼滤波技术的基础上，针对非线性系统的状态估计研究出扩展卡尔曼滤波方法（EKF）。由于其较高的估计精度与鲁棒性，经过近 80 年的发展仍然得到学者们不断的研究。

即时定位与地图构建（SLAM）是导航领域的关键技术之一，自 1988 年有 Smith、Self 等人初步提出完整的框架以来，已经逐渐发展出多种工程领域的 SLAM，如陆地环境中激光 SLAM、视觉 SLAM，水下环境中融合声呐的 SLAM、融合多普勒技术的 SLAM 等。其中，由于视觉传感器较高的性价比与环境适应性而得到学者们广泛的研究[5]。

由于 SLAM 技术涉及多种传感器的信息融合，采用 EKF 算法在 SLAM 领域占据一定地位，并得到一些学者的研究。EKF-SLAM 利用路标作为观测向量的变量、利用移动装置的状态作为状态向量进行估计，在一定环境下有着较高的精度与处理速度[6]。

## 1.2 国内外发展趋势

目前基于滤波方法的 SLAM 得到世界各地诸多学者们的研究，并且在实时性、定位精度等方面取得较好的实验效果，其中最具代表性的算法包括 EKF-SLAM[7]（基于扩展卡尔曼滤波算法的 SLAM）、UKF-SLAM（基于无迹卡尔曼的 SLAM）、FAST-SLAM1.0[8]，2.0[9]（基于 EKF 与 PF 混合滤波算法的 SLAM）。由于 EKF-SLAM 是一种基于滤波方法的即时定位与地图构建算法，其定位精度与效率取决于环境中特征点数，在大范围环境中的实时应用有着局限性，因此如

何对其进行优化得到是目前诸多学者进行研究所要解决的难题。

## 1.3 主要内容与章节安排

本文以单目视觉 SLAM 技术为基础，研究了 EKF 在其中的应用。在 Matlab 仿真环境中，以四轮小车模型为载体，结合小车的运动模型通过编码器原理建立小车的运动方程，利用视觉感知标签建立观测方程，并采用 EKF 对编码器信息与标签信息进行信息融合，来对小车进行定位，并从定位精度方面验证了 EKF 的合理性。

论文内容及章节安排如下：

第 1 章，绪论。介绍了基于 EKF 的 SLAM 研究方法的背景与研究意义，并分别介绍了 EKF 与 SLAM 的研究历史与现状，介绍了国内外采用 EKF 进行 SLAM 研究的进展与成果。

第 2 章，理论。介绍了 KF 的理论基础与应用场景，在系统方程满足线性条件并且噪声服从高斯分布的前提下，推到了 KF 的一步状态预测方程、状态预测误差协方差、卡尔曼增益、状态更新方程、状态更新协方差五个方程。并推导了 EKF 的理论基础，介绍了状态方程观测方程线性化方法，介绍了 SLAM 的架构。

第 3 章，建立模型。结合小车的运动模型，以小车的差速模型建立系统的状态方程，以标签的距离与角度建立了观测模型，为实际仿真实验奠定理论基础。

第 4 章，仿真实验。在 Matlab 仿真环境中分析采用 EKF 对编码器信息与标签信息进行融合与定位的过程，通过仿真分析了单纯采用编码器信息的误差累计效应，验证了采用 EKF 融合编码器信息与标签信息可以提高小车定位精度，并验证了丰富的标签数量更有利于提高小车自身的定位精度。

# 第 2 章 基于 EKF 的 SLAM 方法

SLAM 过程是通过移动装置携带的某种传感器对环境进行感知，利用周围环境的信息进行定位，在移动的过程中不断迭代并构建局部地图，这个过程可以由运动模型与观测模型进行数学建模。在离散的时间点 $t=1,2,...,K$ 对应的位姿为 $x_1,x_2,...,x_k$，假设地图中路标点的个数为 $N$，则各个路标位置记为 $y_1,y_2,...,y_N$。从 $k-1$ 到 $k$ 时刻，移动装置的位姿由 $x_{k-1}$ 变为 $x_k$，之间的关系式可以由（2-1）进行描述：

$$x_k = f(x_{k-1},u_k,\omega_k) \tag{2-1}$$

其中，$u_k$ 为编码器在 $k$ 时刻的示数，$\omega_k$ 为噪声，利用函数 $f$ 可以对下一时刻的运动位置进行航位推算。

视觉 SLAM 过程中，通过相机对周围环境中的路标进行观测，在 $k$ 时刻看到某个路标 $y_j$，产生一个观测量 $z_{k,j}$，其观测模型可以由式（2-2）进行描述。

$$z_{k,j} = h(y_j,x_k,v_{k,j}) \tag{2-2}$$

其中，$v_{k,j}$ 为观测噪声，利用观测模型 $h$ 可以得到路标的相对位置。采用视觉传感器的 SLAM 的运动模型可以由式（2-3）进行描述。

$$\begin{cases} x_k = f(x_{k-1},u_k)+\omega_k \\ z_{k,j} = h(y_j,x_k)+,v_{k,j} \end{cases} \quad k=1,2,...,K,\ j=1,2,...,N \tag{2-3}$$

令 $x_k=\{x_k,y_1,...,y_m\}$，即取 $x_k$ 为 $k$ 时刻位姿及观测到的路标坐标。同时令 $z_k=\{z_{k,1},z_{k,2},...,z_{k,m}\}$，即取 $z_k$ 为 $k$ 时刻能够观测到的路标观测值，根据式（2-3）可以得到式（2-4）。

$$\begin{cases} x_k = f(x_{k-1},u_k)+\omega_k \\ z_{k,j} = h(x_k)+v_k \end{cases} \quad k=1,2,...,K \tag{2-4}$$

$x_k$ 是带有噪声的变量，可以看成是服从某种概率分布的随机变量，用概率分布表示为 $P(x_k|x_0,u_{1:k},z_{1:k})$，根据贝叶斯法则可以得到式（2-5）。

$$P(x_k|x_0,u_{1:k},z_{1:k}) \propto P(x_k|z_k)P(x_k|x_0,u_{1:k},z_{1:k-1}) \tag{2-5}$$

将 $P(x_k|x_0,u_{1:k},z_{1:k-1})$ 在 $x_{k-1}$ 进行展开可以得到式（2-6）。

$$P(x_k|x_0,u_{1:k},z_{1:k-1}) = \int P(x_k|x_{k-1},x_0,u_{1:k},z_{1:k-1})P(x_{k-1}|x_0,u_{1:k},z_{1:k-1})dx_{k-1} \tag{2-6}$$

式（2-6）为利用编码器信息、摄像头观测信息及初始状态对之后每一个状态进行估计，当假设其为马尔科夫过程时可以采用 EKF 进行信息融合，这种假设忽略编码器的误差累计，但可以利用摄像头的数据进行修正，具有一定的合理性[7]，这也是 EKF-SLAM 的研究基础。

## 2.1 KF 算法

当假设式（2-6）为服从马尔科夫过程，即 $k$ 时刻的状态只与 $k-1$ 时刻相关，与之前时刻无关，则可以得到时（2-7）。

$$P(x_k \mid x_0, u_{1:k}, z_{1:k-1}) = \int P(x_k \mid x_{k-1}, u_k) P(x_{k-1} \mid x_0, u_{1:k}, z_{1:k-1}) dx_{k-1} \tag{2-7}$$

其中，$P(x_k \mid x_0, u_{1:k}, z_{1:k-1})$ 为 $k-1$ 时刻的概率分布。则在服从马尔科夫过程的前提下，SLAM 过程转变为从 $k-1$ 时刻推导 $k$ 时刻的状态分布。

KF 算法针对线性高斯系统，及系统的状态方程为线性方程，并且噪声分布服从高斯分布，其运动方程与观测方程为线性方程如式（2-8）。

$$\begin{cases} x_k = A_k x_{k-1} + B_k u_k + \omega_k \\ z_k = C_k x_k + v_k \end{cases} \tag{2-8}$$

状态与噪声分布如式（2-9）。

$$\begin{aligned} &\omega_k \sim N(0, R_k)\ v_k \sim N(0, Q_k) \\ &R_k = cov(\omega_k)\ \ Q_k = cov(v_k) \end{aligned} \tag{2-9}$$

$cov(\omega_k)$ 是对 $cov(\omega_k, \omega_k)$ 简写，下文同理。

假设已知 $k-1$ 时刻的状态估计 $\hat{x}_{k-1}$ 及误差协方差矩阵 $\hat{P}_{k-1}$，根据 $k$ 时刻的编码器数据 $u_k$ 可以得到先验状态估计 $\bar{x}_k$ 如式（2-10）。

$$\bar{x}_k = A_k \hat{x}_{k-1} + B_k u_k \tag{2-10}$$

根据高斯分布的特性可以推导得到先验误差协方差矩阵 $\bar{P}_k$ 如式（2-11）。

与视觉传感器的观测数据 $z_k$，可以得到 $k$ 时刻状态估计 $\hat{x}_k$ 及误差协方差矩阵 $\hat{P}_k$。

$$\begin{aligned} \bar{P}_k &= cov(x_k, \bar{x}_k) \\ &= cov(A_k x_{k-1} + B_k u_k + \omega_k - A_k \hat{x}_{k-1} - B_k u_k) \\ &= A_k \hat{P}_{k-1} A_k^T + R_k \end{aligned} \tag{2-11}$$

式（2-9）利用 $k-1$ 时刻的状态进行状态估计的精度采用与实际测量值之间的误差进行衡量，引入残差 $e_k$ 进行描述如式（2-12）所示。

$$e_k = z_k - C_k \bar{x}_k \tag{2-12}$$

同理残差的误差协方差 $\tilde{P}_k$ 如式（2-13 所示）

$$\begin{aligned} \tilde{P}_k &= cov(z_k - C_k \bar{x}_k) \\ &= C_k P_k C_k^T + Q_k \end{aligned} \tag{2-13}$$

利用先验状态估计 $\bar{x}_k$ 和观测噪声残差 $e_k$ 对 $k$ 时刻状态进行估计得到式（2-14）。

$$\hat{x}_k = \bar{x}_k + K_k e_k \tag{2-14}$$

其中 $K_k$ 是卡尔曼增益，其定义如式（2-15）。

$$K_k = \bar{P}_k C_k^T \tilde{P}_k^T \tag{2-15}$$

其由先验误差协方差与残差协方差矩阵唯一确定。状态估计 $\hat{x}_k$ 的误差协方差 $\hat{P}_k$ 由式（2-16）确定。

$$\hat{P}_k = (I - K_k C_k)\bar{P}_k \tag{2-16}$$

综上所述，卡尔曼滤波算法过程如下：根据 $k-1$ 时刻的系统状态无偏估计 $\hat{x}_{k-1}$ 及误差协方差 $\hat{P}_{k-1}$、$k$ 时刻的编码器信息 $u_k$ 及观测值 $z_k$，根据状态方程得到先验估计值 $\bar{x}_k$ 及误差协方差 $\bar{P}_k$。通过观测数据进行校正信息计算卡尔曼增益 $K_k$，进一步得到后验状态估计 $\hat{x}_k$ 及误差协方差矩阵 $\hat{P}_k$。

## 2.2 EKF 算法

实际应用中系统的状态方程往往是非线性的，KF 方法不再适用，因此需要将非线性系统进行线性化，将状态噪声与观测噪声近似为高斯分布进行处理，然后再利用卡尔曼滤波算法进行处理，这就是 EKF 的思想。

$$\begin{cases} x_k = f(x_{k-1}, u_k) + \omega_k \\ z_{k,j} = h(x_k) + v_k \end{cases} \rightarrow \begin{cases} x_k = A_k x_{k-1} + B_k u_k + \omega_k \\ z_k = C_k x_k + v_k \end{cases} \tag{2-17}$$

在 $k$ 时刻，将运动方程在 $\bar{x}_{k-1}$ 处进行泰勒展开可以得到式（2-18）。

$$x_k = f(\hat{x}_{k-1}, u_k) + \frac{\partial f(x_{k-1}, u_k)}{\partial x_{k-1}}\Big|_{\hat{x}_{k-1}} (x_{k-1} - \hat{x}_{k-1}) + \omega_k \tag{2-18}$$

记 $F_k = \frac{\partial f(x_{k-1}, u_k)}{\partial x_{k-1}}\Big|_{\hat{x}_{k-1}}$。同理将观测方程在 $\bar{x}_{k-1}$ 处进行泰勒展开可以得到式（2-19）。

$$z_k \approx h(\hat{x}_k) + \frac{\partial h(x_k)}{\partial x_k}\Big|_{\hat{x}_k} + v_k \tag{2-19}$$

记 $H_k = \frac{\partial h(x_k)}{\partial x_k}\Big|_{\hat{x}_k}$，则根据式（2-18）、（2-19）可以得到 EKF 的运动方程如式（2-20）。

$$\begin{cases} x_k \approx F_k x_{k-1} + f(\hat{x}_{k-1}, u_k) - F_k \hat{x}_{k-1} + \omega_k \\ z_k \approx H_k x_k h(\hat{x}_k) - H_k \bar{x}_k + v_k \end{cases} \tag{2-20}$$

由式（2-20）可以看出 EKF 在 KF 的基础上首先对状态方程与观测方程进行线性化，然后采用与 KF 相同的方法进行状态估计。算法流程图如图 2-1。

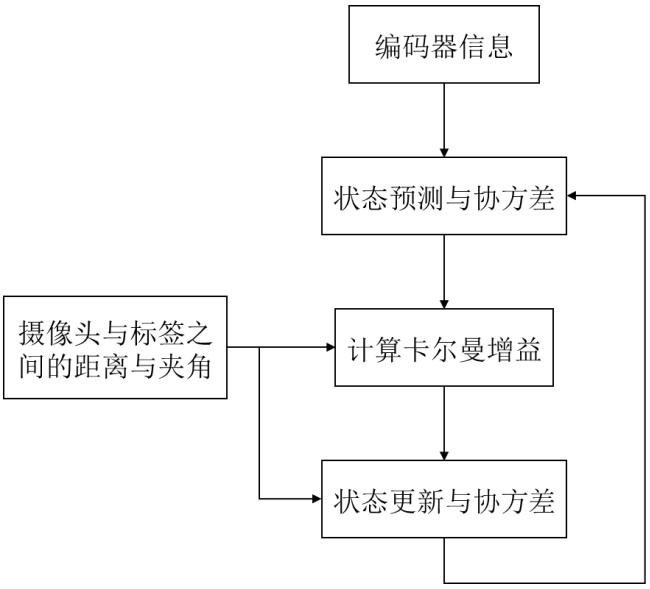



图 2-1 EKF-SLAM 算法流程图

## 2.3 本章小结

本章首先介绍了 KF 的使用条件，推到了 KF 应用的一般步骤，即在已知线性系统状态方程观测方程，并且噪声服从高斯分布的情况下，需要利用上一步的状态更新值与状态更新误差协方差确定一步状态预测方程、状态预测误差协方差矩阵、卡尔曼滤波增益、状态更新方程与状态更新误差协方差，通过这五个方程不断循环迭代得到连续的状态估计。但由于实际应用中大部分系统是非线性的，这种方法具有局限性。

其次介绍了非线性系统的线性化方法与 EKF 算法的流程，即针对非线性系统采用一阶泰勒展开进行线性化，通过将非线性的状态方程与观测方程线性化并利用 KF 的过程进行状态估计。

# 第 3 章 模型构建与分析

本章对基于标签的 EKF-SLAM 技术进行研究，以移动四轮小车作为载体模型，利用单目摄像头对标签信息进行提取与辨认，计算当前捕捉到的标签与摄像头之间的夹角与距离作为观测量；同时以编码器信息作为里程计信息建立当前状态的状态方程，通过融合标签信息与里程计信息进行状态估计与轨迹记录。利用里程计信息需要的到小车的运动模型，利用单目摄像头观测标签需要得到摄像头的小孔成像模型，本章对这两种模型进行构建与分析。

## 3.1 运动与观测模型

SLAM 过程的数学模型由运动方程与观测方程进行描述，本节介绍了用于构建运动方程的车辆差速转向模型。

车辆差速模型如图 3.1 所示。图中，右下角坐标系 x-y 为固定在世界中的固定坐标系，$\theta$ 为转角正方向。$(x_{k-1},y_{k-1})$ 和 $\theta_{k-1}$ 为 $k-1$ 时刻两后轮中心在固定坐标系下坐标及车辆航向角。$(x_k,y_k)$ 和 $\theta_k$ 为 $k$ 时刻两后轮重心在固定坐标系下坐标及车辆航向角。$r_k$ 为 $k$ 时刻瞬时转弯半径。$L_k,R_k,D_k$ 分别为左右轮及两轮中心从 $k-1$ 时刻到 $k$ 时刻的行驶距离。

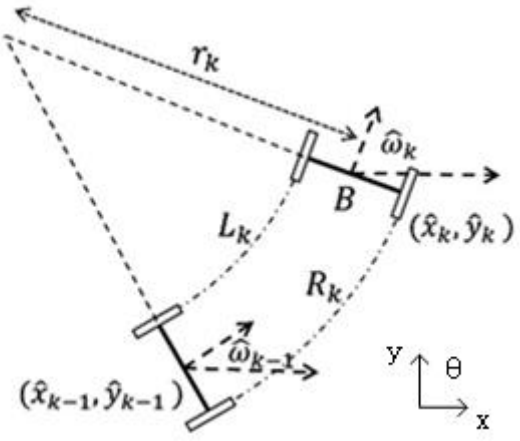



图 3.1 车辆差速转向模型

根据文献[10]中的车辆差速模型，结合图 3.1 可以得到在转弯过程中，从 $k-1$ 时刻过渡到 $k$ 时刻时的状态关系如式（3-1）。

$$\begin{cases} x_k = x_{k-1} + r_k(\sin\theta_{k-1} - \sin\theta_k) \\ y_k = y_{k-1} + r_k(\cos\theta_{k-1} - \cos\theta_k) \\ \theta_k = \theta_{k-1} + \omega_k \end{cases} \tag{3-1}$$

其中，$r_k=\frac{D_k}{\omega_k}=\frac{\frac{L_k+R_k}{2}}{\frac{R_k-L_k}{B}}=-\frac{B}{2}\cdot\frac{L_k+R_k}{L_k-R_k}$，为小车在转弯过程中前轮中线与转弯中心的距离。式（3-1）即构建的运动状态模型，在已知前一时刻坐标位置的情况下，通过当前时刻观测得到的编码器信息 $\omega_k$ 可以对下一时刻的坐标进行预测。

## 3.2 相机测距模型

相机模型是对三维空间中的点投影到二维图片平面过程的数学描述。针孔相机模型是一种简单而有效的相机模型，三维空间中的物体反射或自己发出的光线，经过相机光心投影到成像平面上，成像平面接收到光点后感光器件产生测量值，形成照片中的像素值。

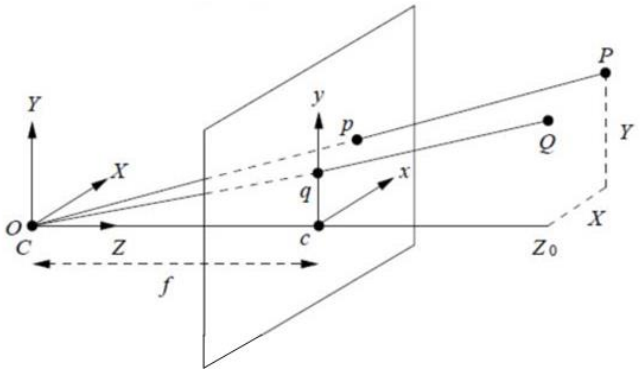



图 3.2 相机成像模型

针孔相机模型投影过程如图 3.3 所示。图中，$C-X-Y-Z$ 为相机坐标系，$c-x-y$ 为图像坐标系。三位空间中一点 P 在相机坐标系下的坐标为$(X,Y,Z)$，投影至成像平面上的点 $p$，设点 $p$ 在相机坐标系下的坐标为$(X',Y',f)$，点 $p$ 在图像坐标系下的坐标为$(x,y)$，显然有 $x=X',y=Y'$。有空间几何关系可以得到式（3-2）。

$$\frac{Z}{f}=\frac{X}{X'}=\frac{Y}{Y'} \tag{3-2}$$

即：

$$\begin{cases} X'=f\dfrac{X}{Z} \\ Y'=f\dfrac{Y}{Z} \end{cases} \tag{3-3}$$

相机有呈矩阵排列的像素点组成，因此在相机平面更适合采用像素坐标表示其所在位置。像素坐标为在物理成像平面上固定的坐标系，横轴 $u$ 与图像坐标轴

$x$ 轴同相平行，纵轴 $v$ 与图像坐标系 $y$ 轴同相平行。点 $p$ 在图像坐标系下的坐标与在像素坐标系下的坐标关系如式（3-4）。

$$\begin{cases} u = \alpha X' + c_x \\ v = \beta Y' + c_y \end{cases} \tag{3-4}$$

由式（3-3）、（3-4）可以得到式（3-5）。

$$\begin{cases} u = f_x \dfrac{X}{Z} + c_x \\ v = f_y \dfrac{Y}{Z} + c_y \end{cases} \tag{3-5}$$

其中，$f_x = \alpha f, f_y = \beta f$，即对焦距在两个坐标轴上进行缩放。$c_x$、$c_y$ 为图像坐标系原点在像素坐标系下的坐标。采用齐次公式表示可以得到式(3-6).

$$Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = KP \tag{3-6}$$

其中 $K$ 为相机内参矩阵，需要进行标定得到。$P$ 为三维空间点在相机坐标系下的坐标。利用坐标变换与旋转，在已知某一点世界坐标系的前提下可以将其转化到像素坐标系如式（3-7）。

$$Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \end{bmatrix}\begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = KTP_W \tag{3-7}$$

公式中 $T$ 是固定坐标系到相机坐标系的变换矩阵。利用式（3-7）可以得到世界坐标系到像素坐标系的转化关系。在利用式（3-7）进行坐标系至像素坐标西的坐标转化后需要根据像素中的坐标计算摄像头至标签的距离与角度信息，其计算公式如（3-8）。

$$\begin{cases} d = \sqrt{t_1^2 + t_3^2} \\ \zeta = arctan\dfrac{t_1}{t_3} \end{cases} \tag{3-8}$$

公式中，$d$ 是摄像头至标签的距离，$\zeta$ 是摄像头与标签之间的角度。

## 3.3 本章小结

本章对所用到的车辆差速转向模型与相机模型进行介绍，为 EKF 融合奠定

理论基础。

# 第 4 章 Matlab 仿真实验

本章基于第 3 章的模型与第 2 章的滤波理论搭建 Matlab 仿真环境，以移动小车为载体，采用 EKF 算法对摄像头与编码器信息进行融合，实现自身定位。

## 4.1 基于标签的 EKF-SLAM 方法

设 $k$ 时刻状态向量 $X_k=(x,y,\theta,m_{1,x},m_{1,y},...,m_{j,x},m_{j,y},...,m_{N,x},m_{N,y})_k^T$，$x,y,\theta$ 为两后轮中心在固定坐标系下横纵坐标及车辆航向角，$m_{j,x},m_{j,y}$ 为标签 j 在固定坐标系下横纵坐标，$j=1,2,...,N$。取 $u_k$ 为从 $k-1$ 到 $k$ 时刻左右轮行驶距离，$u_k=(L_k,R_k)$。由于 EKF 的一步预测及误差协方差的计算与小车的运动模型有关，因此分为转弯与直线运动两种情况分别进行讨论。

（1）转弯运动过程

当车辆转弯时，根据式（2-17）、（3-1）可以得到其一步预测方程 $\bar{X}_k$ 满足式（4-1）。

$$\begin{cases} x_k = x_{k-1} + r_k(\sin\theta_{k-1} - \sin\theta_k) \\ y_k = y_{k-1} + r_k(\cos\theta_{k-1} - \cos\theta_k) \\ \theta_k = \theta_{k-1} + \omega_k \\ m_{1,x}^k = m_{1,x}^{k-1} \\ m_{1,y}^k = m_{1,y}^{k-1} \end{cases} \tag{4-1}$$

一步预测误差协方差如式（4-2）所示。

$$\bar{P}_k = F_k\hat{P}_{k-1}F_k^T + R_k \tag{4-2}$$

$$F_k = \frac{\partial f(X_{k-1},u_k)}{\partial X_{k-1}}\bigg|\hat{X}_{k-1} = \begin{bmatrix} F_k^1 & 0_{3\times N} \\ 0_{N\times 3} & I_{N\times N} \end{bmatrix} \tag{4-3}$$

$$F_k^1 = \begin{bmatrix} 1 & 0 & r_k(\cos\theta_{k-1} - \cos\theta_k) \\ 0 & 1 & r_k(\sin\theta_{k-1} - \sin\theta_k) \\ 0 & 0 & 1 \end{bmatrix} \tag{4-4}$$

（2）直行运动过程

当小车直行时，标签的一步预测方程如式（4-5）。

$$\begin{cases} x_k = x_{k-1} + D_k \cos\theta_{k-1} \\ y_k = y_{k-1} + D_k \sin\theta_{k-1} \\ \theta_k = \theta_{k-1} \\ m_{1,x}^k = m_{1,x}^{k-1} \\ m_{1,y}^k = m_{1,y}^{k-1} \end{cases} \tag{4-5}$$

一步预测协方差方程如式（4-2），其中 $F_k$ 满足式（4-4）、（4-6）。

$$F_k^1 = \begin{bmatrix} 1 & 0 & -D_k \sin\theta_{k-1} \\ 0 & 1 & D_k \cos\theta_{k-1} \\ 0 & 0 & 1 \end{bmatrix} \tag{4-6}$$

由式（3-8）可以得到观测方程如式（4-7）。

$$\begin{cases} d_j^k = \sqrt{(x_k - m_{j,x}^k)^2 + (y_k - m_{j,y}^k)^2} \\ \zeta_j^k = arctan\dfrac{y_k - m_{j,y}}{x_k - m_{j,x}} - \theta_k \end{cases} \tag{4-7}$$

$d_j^k, \zeta_j^k$ 分别为 $k$ 时刻相机与第 $j$ 个二维码之间的距离与角度。根据式（2-19）、（4-7）得到 EKF 中的卡尔曼增益如式（4-8）所示。

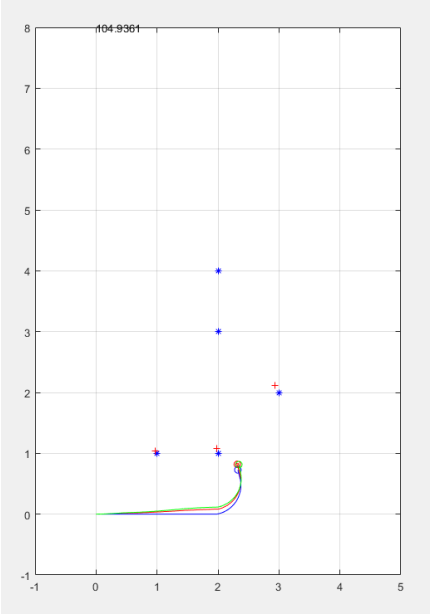
$$K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + Q_k)^{-1} \tag{4-8}$$

公式中 $H_k = \dfrac{\partial h(X_k)}{\partial X_k}\Big|\bar{X}_k$，即式（4-7）的一阶偏导数。由此得到 EKF 的后验概率分布如式（4-9）所示。

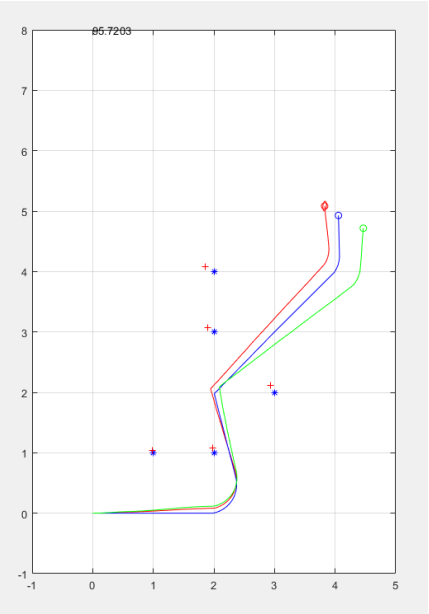
$$\begin{cases} \hat{x}_k = \bar{x}_k + K_k(z_k - H_k \bar{x}_k) \\ \hat{P}_k = (I - K_k H_k)\bar{P}_k \end{cases} \tag{4-9}$$

## 4.2 仿真分析与结论

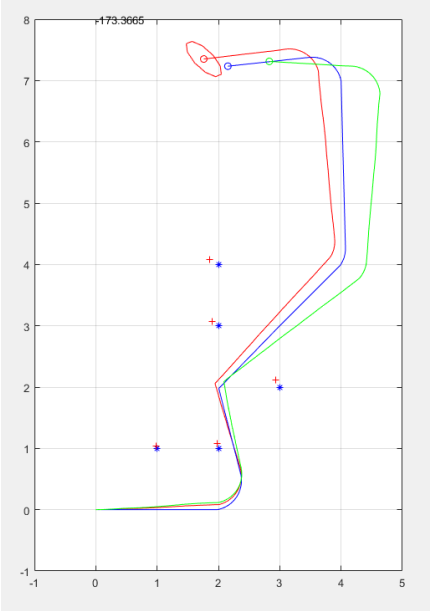
根据 4.1 节卡尔曼滤波处理过程，在 Matlab 搭建仿真环境进行仿真。采用四轮小车为载体，利用编码器记录其历程信息，利用摄像头对周围的标签进行观测，记录标签的位置信息，采用 EKF 对这两种信息进行融合对小车进行定位。数据融合过程中为了简化计算过程利用式（3-8）直接得到摄像头对标签的观测信息，在仿真过程中，设定小车轨迹为固定轨迹，首先采用 6 个标签进行局部定位得到如图 4-1 所示的仿真结果。



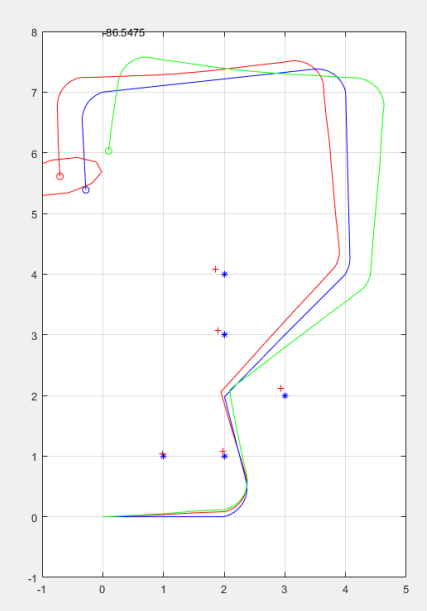

$t_1$



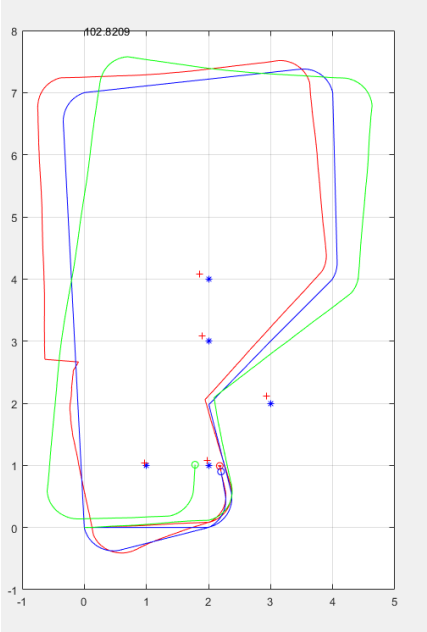

$t_2$




$t_3$




$t_4$




$t_5$

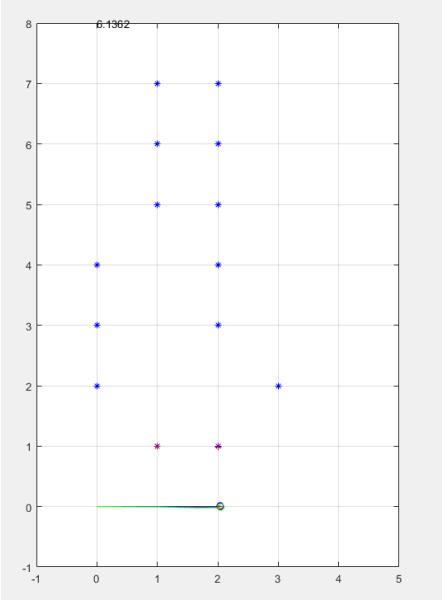
图 4-1 6 个标签位置的 EKF-SLAM 过程

图中，蓝色轨迹为左右编码器在无噪声的情况下得到的小车运行轨迹，为小车行走的真实路径；绿色轨迹为左右编码器在有噪声的情况下由运动模型得到的轨迹，为小车航位推算的结果；红色直线轨迹为 EKF 融合编码器信息与周围标签新的得到的轨迹，为小车实际运行轨迹；蓝色的点为设定的标签位置，红色的点为采用摄像头进行估计的标签位置。

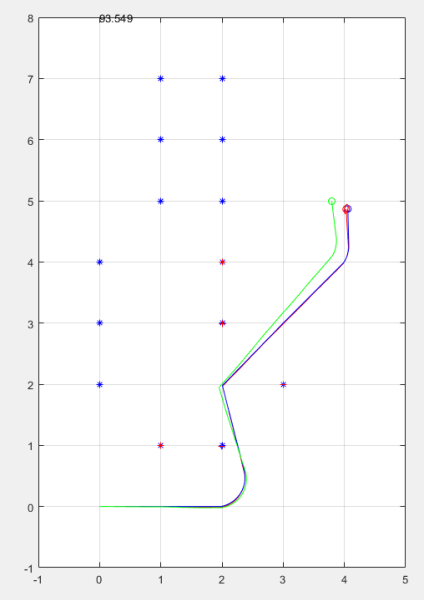
从图 4-1 可以看出在初始时刻标签信息比较丰富的情况下，小车的定位有着较高的精度，但是在标签较少的环境中其定位精度较差。

由编码器进行航位推算得到的轨迹看出在小车运行过程中里程计的误差不断积累；由红色轨迹可以看出采用 EKF 对编码器信息与摄像头信息进行融合相比单独采用里程计信息进行航位推算具有较高的误差，在相同噪声影响的前提下，采用 EKF-SLAM 可以提高定位精度。但是基于少量标签信息的 SLAM 过程仍然存在较大误差，因此可以通过提高标签数量提高定位精度。

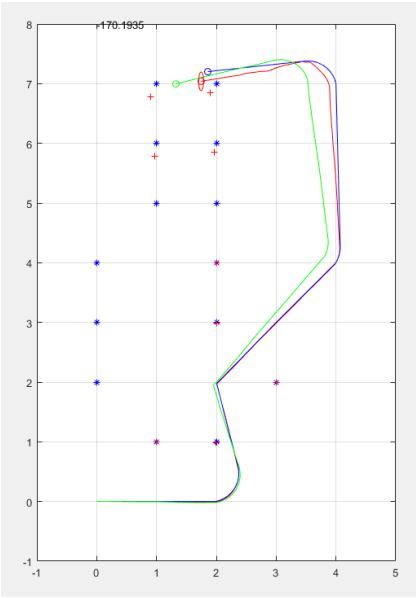
根据红色轨迹的运行结果可以看出，采用 EKF 算法融合编码器信息与标签信息进行定位的精度取决于周围标签的数量，在（-0.7，2.6）附近存在突变，原因在于在这之前的一段过程仅有编码器信息而没有标签信息。有红色点的状态变化可以看出标签的定位精度取决于 EKF 对自身定位的精度。



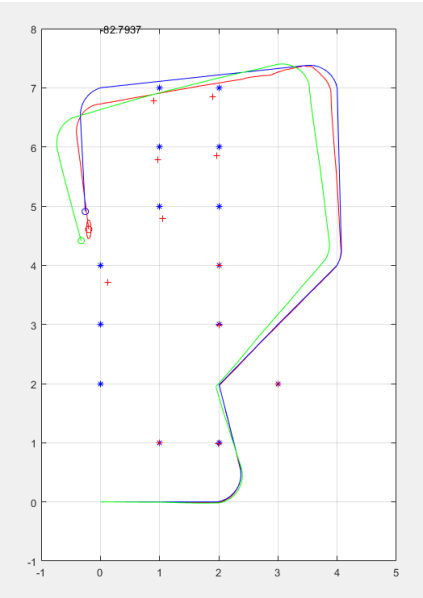

$t_1$



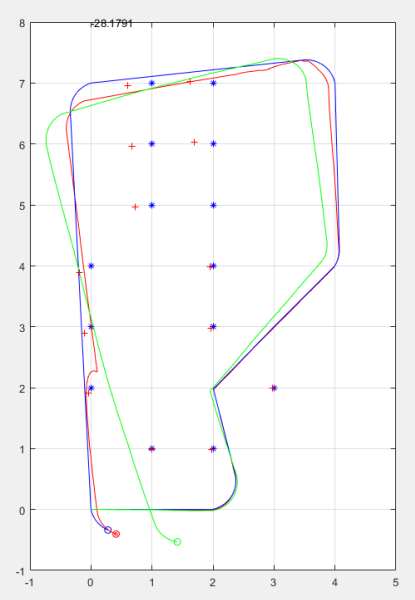

$t_2$




$t_3$




$t_4$




$t_5$

图 4-2 12 个标签位置的 EKF-SLAM 过程

从图 4-3 可以看出相对于 6 个标签位置的 EKF-SLAM 过程，在相同路径下基于 12 个标签位置的 EKF-SLAM 可以获得更高的精度。由于利用单目摄像头获取标签信息并与编码器信息进行融合可以提高定位精度，更多的标签信息可以提高定位精度，并可以提高标签的位置精度。但在实际应用中标签的数量越多需要进行的计算越大，大多数情况下摄像头感知环境中的特征点数成千计，对 SLAM 的实时性影响很大。因此学者们往往针对实时性问题对特征点数进行筛选，采用后端优化的方法减小计算时间。

## 4.3 本章小结

本章对 EKF-SLAM 方法进行仿真实现，以小车为实验载体，首先结合小车的运动模型建立状态方程，并利用摄像头观测标签的距离与夹角得到观测方程，采用 EKF 算法对两种信息进行融合。通过 Matlab 仿真验证了单纯利用编码器信息进行航位推算的误差累积性，与其进行对比在相同环境噪声的情况下，采用 EKF 对编码器信息与摄像头信息进行融合可以提高定位精度。并且分析了标签的数量对定位精度的影响，验证了标签数量越多则定位精度越高的结论。

# 总结与展望

KF 信息融合技术有着严格的数学基础，并且 EKF 算法经过近 80 年的发展在诸多工程实践领域具有成熟的应用。本文介绍了 EKF 算法的流程与理论基础，对 EKF 算法在 SLAM 过程中的应用进行仿真分析，以四轮小车为载体，结合其运动学模型，根据编码器原理建立状态方程，根据周围环境中的标签信息建立观测方程，通过 EKF 算法对编码其信息与标签信息进行融合，实现了小车的定位。

在仿真过程中首先，分析了单纯采用编码器信息进行航位推算具有误差累积效应，由于环境中的噪声对编码器信息的影响，其定位误差随着时间的累积越来越大。其次，分析了采用 EKF 算法融合编码器信息与环境中的标签信息进行定位的精度，通过对比验证了 EKF 算法具有较高的定位精度。 最后，分析了标签的数量对于定位精度的影响，通过仿真证明了丰富的标签数量有助于提高定位精度。

# 参考文献

[1] Kalman R E. A new approach to linear filtering and prediction problems[J]. Journal of basic Engineering, 1960, 82(1): 35-45.

[2] Carlson N A. Federated filter for fault-tolerant integrated navigation systems[C]// Position Location and Navigation Symposium, 1988. Record. Navigation Into the, Century. IEEE Plans '88. IEEE. IEEE, 1988:110-119.

[3] Comanciu D. Real-time tracking of non-rigid objects using mean shift[J]. Proc Cvpr, 2000, 2:2142.

[4] Comaniciu D, Ramesh V, Meer P. Kernel-Based Object Tracking[M]. IEEE Computer Society, 2003.

[5] Fuentes-Pacheco J, Ruiz-Ascencio J, Rendón-Mancha J M. Visual simultaneous localization and mapping: a survey[J]. Artificial Intelligence Review, 2015, 43(1):55-81.

[6] Bailey T, Nieto J, Guivant J, et al. Consistency of the EKF-SLAM Algorithm[C]// Ieee/rsj nternational Conference on Intelligent Robots and Systems. IEEE, 2006:3562-3568.

[7] Kwang-Jin K, Yu M J, Young-Bum P, et al. A Performance Comparison of Extended and Unscented Kalman Filters for INS/GPS Tightly Coupled Approach[J]. Journal of Institute of Control, 2006, 12(8):780-788.

[8] Montemerlo M, Thrun S. Simultaneous localization and mapping with unknown data association using FastSLAM[C]// IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA. IEEE, 2003:1985-1991 vol.2.

[9] Chen S M, Yuan J F, Chen X L, et al. A FastSLAM2.0 algorithm based on electromagnetism-like mechanism[J]. Control Theory & Applications, 2015.

[10] Klančar G, Matko D, Blažič S. A control strategy for platoons of differential drive wheeled mobile robot[J]. Robotics & Autonomous Systems, 2011, 59(2):57-64.

# 附录 A： 关键代码

```
function data = ekfslam(lm, wp)
    format compact;
    configfile;
    %setup animation
    screensize = get(0, 'ScreenSize')*0.75;
    fig = figure('Position', [0 0 screensize(3), screensize(4)]);
    plot(lm(1,:),lm(2,:),'b*')
    hold on, axis equal, grid on
    MAXX = max([max(lm(1,:)) max(wp(1,:))]);
    MAXY= max([max(lm(2,:)) max(wp(2,:))]);
    MINX = min([min(lm(1,:)) min(wp(1,:))]);
    MINY= min([min(lm(2,:)) min(wp(2,:))])
    axis([MINX-1 MAXX+1 MINY-1 MAXY+1])
    h = setup_animations;
    veh = [0 -WHEEL_BASE -WHEEL_BASE; 0 -2 -2]
    global vtrue XX PX DATA
    vtrue = zeros(3,1);
    XX = zeros(3,1);
    DR = zeros(3,1);
    PX = zeros(3);

    dt = DT_CONTROLS;
    dtsum = 0;
    iwp = 1;
    W = 0;
    counter = 0;
    path_count = 0;
    associations = [];
    sim_time = 0;
    DATA = init_storage(XX, PX, vtrue, DR);

    while iwp ~= 0
        sim_time =sim_time+ dt;
        counter = counter +1;
        [W, iwp] = computerotation(wp, iwp, AT_WAYPOINT, W, MAXW, dt);
```

```
vtrue = vehiclemodel(vtrue, V, W, dt);
%Control noise
Vn = V + randn(1)*sqrt(Q(1,1));
Wn = W + randn(1)*sqrt(Q(2,2));

[Gt, Gu] = calcJacobian(XX, Vn, Wn, dt);
PX(1:3, 1:3) = Gt*PX(1:3, 1:3)*Gt.' + Gu*Q*Gu.';
if length(PX) > 3
    PX(1:3,4:end) = Gt*PX(1:3,4:end);
    PX(4:end,1:3) = PX(1:3,4:end)';
end

XX(1:3,:) = vehiclemodel(XX, Vn, Wn, dt);
DR = vehiclemodel(DR, Vn, Wn, dt);

z = []; zindex = [];
znew = true;
for i=lm
    dist = sqrt((i(1) - vtrue(1))^2 + (i(2) - vtrue(2))^2);
    if dist <= MAX_RANGE
        %All landmarks which can be observed described by
range,bearing,id
        bearing = atan2((i(2) - vtrue(2)),(i(1) - vtrue(1)));

        l = [dist; pilimit(bearing - vtrue(3)); i(3)];
        z = [z l]
        %Is this a new landmark? If not, get it's index in the map
        for y = 1:length(associations)
            if i(3) == associations(y)
                zindex = [zindex y];
                znew = false;
            end
        end
        %Never seen this lm before, add it's ID to the association table
        if znew
            associations = [associations i(3)];
            %Increase the size of our mean and covar
            muJ = [XX(1) + l(1)*cos(l(2) + XX(3)); XX(2) + l(1)*sin(l(2)
```

```
+ XX(3))];
                        XX = [XX; muJ(1); muJ(2)];
                        covars = length(PX);
                        PX(covars + 1, covars + 1) = 1;
                        PX(covars + 2, covars + 2) = 1;
                    end
                end
                znew = true;
            end

            %observation jacobian
            j = 0;
            for zi = 1:size(z,2)
                %Calculated with current waypoint zloc
                zloc = z(:,zi);
                for r = 1:length(associations)
                    if zloc(3) == associations(r)
                        j = r;
                    end
                end
                if (j==0)
                    disp('Cannot find landmark');
                end

                jind = j*2 + 2;
                [zhat, H] = observemodel(XX, zloc, jind);

                PHt = PX*H.';
                S = H*PHt + R;
                SInv = inv(S);
                %SInv = (SInv + SInv')*0.5;
                %PSD_check = chol(SInv);
                K = PHt*SInv;
                diff = [zloc(1) - zhat(1); zloc(2) - zhat(2)];
                %K*[zloc(1) - zhat(1); zloc(2) - zhat(2)]
                XLast = XX(1:3);

                XX = XX + K*[zloc(1) - zhat(1); zloc(2) - zhat(2)];
```

```
            L = K*S*K';
            PX = (eye(length(PX)) - K*H)*PX;
            %PX = PX - ((L + L')*0.5)

            j=0;
        end

        store_data(XX, PX, vtrue, DR);
        %calc covar ellipse
        vehEl = make_veh_ellipse(XX,PX);
        set(h.vehStat, 'string',    num2str(XX(3)*180/pi));
        try
            set(h.cova, 'xdata', vehEl(1,:), 'ydata', vehEl(2,:))
            set(h.xv, 'xdata', XX(1,:), 'ydata', XX(2,:))
            set(h.dr, 'xdata', DR(1,:), 'ydata', DR(2,:))
            set(h.pthtrue,          'xdata',          DATA.truth(1,1:DATA.i),          'ydata',
DATA.truth(2,1:DATA.i))
            set(h.pthdead,          'xdata',          DATA.dead(1,1:DATA.i),          'ydata',
DATA.dead(2,1:DATA.i))
            set(h.pth,          'xdata',          DATA.path(1,1:DATA.i),          'ydata',
DATA.path(2,1:DATA.i))
            set(h.xf, 'xdata', XX(4:2:end), 'ydata', XX(5:2:end))
            set(h.xt, 'xdata', vtrue(1,:), 'ydata', vtrue(2,:))
            drawnow
        catch
            disp('whoops')
        end
    end
        %animations
end

function h = setup_animations()
    h.xv = plot(0,0,'ro','erasemode','xor');
    h.xt = plot(0,0,'bo','erasemode','xor');
    h.dr = plot(0,0,'go','erasemode','xor');
    h.pth = plot(0,0,'r','markersize',2,'erasemode','background');
    h.pthtrue = plot(0,0,'b','markersize',2,'erasemode','background');
```

```
    h.pthdead = plot(0,0,'g','markersize',2,'erasemode','background');
    h.obs = plot(0,0,'k','erasemode', 'xor');
    %h.timeelapsed = annotation('textbox', [0.89 0.9 0.1 0.05]);
    h.xf = plot(0,0,'r+');
    h.cova = plot(0,0,'r','erasemode', 'xor');
    h.vehStat = text(0, 8, 'ERROR');
end

function p= make_veh_ellipse(x, P)
    N = 10;
    inc = 2*pi/N;
    phi = 0:inc:2*pi;
    circ = 2*[cos(phi); sin(phi)];

    r = sqrtm_2by2(P(1:2,1:2));
    a = r*circ;
    p(2,:) = [a(2,:)+x(2) NaN];
    p(1,:) = [a(1,:)+x(1) NaN];
end

function data = init_storage(x, P, vtrue, DR)
    data.i=1;
    data.path = x;
    data.truth = vtrue;
    data.dead = DR;
    data.state(1).x = x;
    data.state(1).P = diag(P);
end

function store_data(x, P, vtrue, DR)
    global DATA
    CHUNK = 500;
    len = size(DATA.path,2);
    if (DATA.i == len)
        disp('increase')
        if len < CHUNK, len= CHUNK; end
        DATA.path = [DATA.path zeros(3,len)];
        DATA.truth = [DATA.truth zeros(3,len)];
```

```
            DATA.dead = [DATA.dead zeros(3,len)];
            pack
        end
        i = DATA.i + 1;
        DATA.i = i;
        DATA.path(:,i) = x(1:3);
        DATA.truth(:,i) = vtrue;
        DATA.dead(:,i) = DR;
        DATA.state(i).x = x;
        DATA.state(i).P = diag(P);
end
```