一个 EKFSLAM 的仿真例子

Shitong Du *

September 16, 2019

Abstract

这是一个关于 EKFSLAM 的例子。一直以来对经典的 filter-based on framework 不太明白，正好借用这个例子来对 EKFSLAM 进行深入的了解。在网上找了一个 EKFSLAM 的仿真例子，该例子通过仿真小车的运动状态，对真实的运动轨迹增加噪声来模拟小车的运动方程所获得的姿态。通过对 landmark 点增加噪声来模拟传感器获取的量测数据。

1 Kalman Filter

Kalman filter 属于贝页斯滤波 (Bayes filters) 的一种，它的适用条件是线性高斯系统 (Linear Gaussian systems). 所谓的高斯系统指的是服从正态分布的系统。经典的卡尔曼滤波框架由状态方程和量测方程组成，具体公式如下：

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{1}$$

$$z_t = C_t x_t + \delta_t \tag{2}$$

Here x_t and x_{t-1} are state vectors, and u_t is the control vector at time t. A_t is a square matrix of size $n \times n$, where n is the dimension of the state vector x_t. B_t is of size $n \times m$, with m being the dimension of the control vector u_t. The random variable ε_t in (2) is a Gaussian random vector that models the uncertainty introduced by the state transition.

C_t is a matrix of size $K \times n$, where k is the dimension of the measurement vector z_t. The vector δ_t describles the measurenment noise. The distribution of δ_t is a multivariate Gaussian with zero mean and covariance Q_t.

卡尔曼滤波的算法基本公式被呈现在 Algorithm 1中. 具体的推导目前暂且不用关注，现在需要知道的是 KF 框架的输入，输出和如何使用 KF。在使用 Kalman 过程中，主要使用 mean μ_t 和 the covariance Σ_t 来表征系统当前 t 时刻的状态。公式 2 和 3 被称做**状态预测** (state prediction)。该过程会根据上一时刻估计到的状态 (μ_{t-1}) 和运动模型 (u_t) 来估计当前时刻的状态 ($\bar{\mu}_t$). $\bar{\Sigma}_t$ 是用来对状态估计准确度的一种测量。它表征了我们仅仅通过过程控制方程 (Eq. (2)) 来预测当前时刻状态的准确度。公式 4 到 6 为**状态更新** (state updation). K_t 是**卡尔曼增益** (Kalman gain)。**卡尔曼增益的作用，就是分配模型预测的状态和传感器测量的状态之间的权重**. 在计算完卡尔曼增益后，就可以通过该增益对状态进行进一步的更新，之所以称为进一步更新针对的是状态预测阶段而言。在式 5 中，z_t 是当前时刻的量测值，C_t 起到了连接观测量和状态之间的关系。这两个量在不同的应用场景下有不同的意义。如在激光雷达中，z_t 可以表示激光雷达数据直接得到的数据 (range and angle)。此时 C_t 就是当前状态和 range 和 angle 之间的关系；如果是在视觉场景中，z_t 可以表示提取的特许点的像素点坐标，则 C_t 表示状态和对应像素点

*dushitong@hrbeu.edu.cn

Algorithm 1 Kalman filter algorithm

Input: μ_{t-1}, Σ_{t-1}, u_t, z_t

Output: μ_t, Σ_t

function Algorithm Kalman filter(μ_{t-1}, Σ_{t-1},u_t, z_t)

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

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

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

$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$

$\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$

return μ_t,Σ_t

end function

坐标之间的关系。关于卡尔曼滤波的形象解释可以参照博客[1]。卡尔曼滤波算法部分的内容主要来自书籍 Probabilistic Robotics 的 3.2 章节。当前的要求是不用明白如何推导的，只要知道卡尔曼滤波的输入输出和五个公式即可。

2 The Extended Kalman Filter

经典的卡尔曼滤波是基于线性高斯假设的条件下的，**也就是说过程和量测方程函数都是状态的线性函数** (linear function)。要明白之所以要有线性假设是因为高斯模型在经过线性变换以后依然是高斯过程。所以线性假设最终还是为了服务于高斯假设。这里没有深究为什么一定要是高斯过程，肯定是在推导五个公式的过程用到了高斯假设。**经典的卡尔曼滤波基于系统的线性假设是不符合现实条件的，因为现实生活中大部分的实例都是非线性系统，这样自然就引出了卡尔曼滤波的改进版本。扩展卡尔曼滤波是卡尔曼滤波的进阶版，差别不大，但可以应用于非线性系统。** EKF 的关键思想在于**线性化** (linearization). 因为过程方程和量测方程都是状态的非线性函数，所以 Eq. (2)的表示方法不能在使用，故我们重新定义这两个方程。

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{3}$$

$$z_t = h(x_t) + \delta_t \tag{4}$$

以上两个方程是非线性的，为了能继续利用经典的卡尔曼滤波算法。扩展卡尔曼滤波对非线性函数 $g(.)$ 和 $h(.)$ 进行一阶**泰勒展开** (Taylor Expansion)

$$G_t = \frac{\partial g}{\partial x}|_{\hat{x}_{t-1|t-1},u_t} \tag{5}$$

$$H_t = \frac{\partial h}{\partial x}|_{\hat{x}_{t|t-1}} \tag{6}$$

泰勒展开后，把一阶泰勒展开替换经典卡尔曼滤波的对应部分即可得到下面的 EKF 的五个公式。

$$\hat{x}_{t|t-1} = g(u_t, x_{t-1}) \tag{7}$$

$$\hat{\Sigma}_{t|t-1} = G_t\Sigma_{t-1}G_t^T + R_t \tag{8}$$

$$K_t = \hat{\Sigma}_{t|t-1}H_t^T(H_t\hat{\Sigma}_{t|t-1}H_t^T + Q_t)^{-1} \tag{9}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - h(\hat{x}_{t|t-1})) \tag{10}$$

$$\hat{\Sigma}_{t|t} = (I - K_t H_t)\hat{\Sigma}_{t|t-1} \tag{11}$$

[1] https://www.jianshu.com/p/9214a94b26ca

请注意这里五个公式的表达式和经典卡尔曼滤波框架中使用的符号不太一样，但表示的意义是一样，这里之所以这么用是为了清楚的表示出在泰勒展开时给 G_t 和 H_t 赋的值到底是哪一个。

3 EKFSLAM

3.1 EKFSLAM 模型构建

假设现在有一个移动 robot 通过安装在其上的 2D LIDAR 来获取外界信息。下面我们基于这样一个应用场景来列出系统的运动和量测方程。在列出运动方程之前，我们首先要确定系统要估计的状态量的形式。因为本文只关注 2D 问题，所以对于 2DEKFSLAM，状态量 x_t 的表示如下：

$$x_t = (x, y, \theta, m_{1,x}, m_{1,y}, \ldots, m_{n,x}, m_{n,y})^T \tag{12}$$

其中前三项表示机器人的位姿量，后面的 $2 \times n$ 个量表示地图中的 landmark 量。请注意这个状态量的构成已经道出了 SLAM 的本质。因为状态量中不仅包括位姿量还包括了路标点，这就说明了在每次估计状态时，位姿和地图中的 landmark 点也在更新的范围内。这些被估计的 landmark 点即我们所说的地图点。**在每个时刻，算法根据** sensor **看到的** landmark **点的新旧来决定如何算法下面的操作。所谓新的** landmark **点是指在之前的时刻机器人没有看到过的点，也就是说此刻的状态量中不包括的那些** landmark **点；所谓旧的路标点是指机器人在之前的时刻已经看到过这些点，也就是说此刻的状态量** x_t **中已经包括了这些点。基于以上分类，**SLAM **算法使用旧的路标点通过启动** EKF **来更新状态（这些旧的路标点相当于框架中的量测** z**）；而那些新的路标点只是被加入到状态量中，也就是扩充到状态量中，并不输入到卡尔曼滤波框架中。**

依照 Eq. (4)结合上面说的实际应用场景，可以得到具体的运动和量测方程：

$$\begin{cases} \acute{x} = x + V_n \cdot dt \cdot \cos(G_n + \theta) \\ \acute{y} = y + V_n \cdot dt \cdot \sin(G_n + \theta) \\ \acute{\theta} = \theta + \frac{V_n \cdot dt \cdot \sin(G_n)}{l} \end{cases} \tag{13}$$

其中 V_n 是机器人的前进线速度，G_n 是机器人的转向角。总之 V_n 和 G_n 作为系统的控制输入命令即可。dt 是系统采样时间。

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} m_{i,x} - x_{t,x} \\ m_{i,y} - x_{t,y} \end{pmatrix} \tag{14}$$

$$q = \delta^T \delta \tag{15}$$

$$z_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - x_{t,\theta} \end{pmatrix} \tag{16}$$

其中 m_i 为当前时刻所看到的 landmark 点。x_t 为当前时刻机器人的位姿。z_t^i 是 2D LIDAR 所采集到也就是量测信息。以上 Eq. (13)和 (16)分别是该应用场景下的过程和量测方程。

3.2 EKFSLAM 参数求解

Section. 3.1已经给出了系统的运动和量测方程，接下来就是按照 EKF 框架的五个公式来更新状态向量。首先我们要弄明白状态向量 x_t 和对应的协防差矩阵 Σ 的形式。

$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{x} \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \vdots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma} \tag{17}$$

其中 $m*$ 是当前状态下的路标点。接下来就是计算 G_t 和 H_t. 这里有一个问题需要注意的是，**在计算 G_t 的时候，不仅对状态求了一阶导数，对输入状态量 Vn 和 G_n 也进行了求导，分别得到 G_t^v 和 G_t^u。个人不明白为什么要这么做**。 这里暂且存疑且只给出最终的计算结果。中间过程可参照参照博客[2]. 因为过程方程是由小车运动状态得到的，而在移动过程中状态量中只有位姿信息发生了变化。所以 G_t^v 和 G_t^u 是分块矩阵。由以上叙述可知，Eq. (8)可以进一步化为：

$$\hat{\Sigma}_{t|t-1} = G_t^v \Sigma_{t-1} (G_t^v)^T + G_t^u R_t (G_t^u)^T \tag{18}$$

$$= \begin{pmatrix} G_t^x & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix} \begin{pmatrix} (G_t^x)^T & 0 \\ 0 & I \end{pmatrix} + G_t^u R_t (G_t^u)^T \tag{19}$$

$$= \begin{pmatrix} G_t^x \Sigma_{xx} (G_t^x)^T & G_t^x \Sigma_{xm} \\ (G_t^x \Sigma_{xm})^T & \Sigma_{mm} \end{pmatrix} + G_t^u R_t (G_t^u)^T \tag{20}$$

在获得 G_t 以后，还需要求取 H_t。H_t 是一个 $k \times n$ 的矩阵。k 是量测点的数量 $*$ 路标点维数，n 是状态向量此刻的树目。Eq. (16)对状态向量 x 求导可得：

$$H_t^i = \frac{\partial h}{\partial x} \tag{21}$$

$$= \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & \delta_y & \delta_x \end{pmatrix} \tag{22}$$

注意上式只是针对一个 landmark 的情况，也就是量测点只有一个。所以是 H_t^i。H_t^i 是一个 $2 \times n$ 的矩阵，2 行是因为一个路标是 2D 的。H_t 是 $k \times n$ 矩阵，k 是 landmakrs 的数量 $*2$。

上面的所有步骤都是按照 EKF 的 5 个公式的顺序来叙述的，在求得 G_t 和 H_t 后，可以进一步求得卡尔曼增益从而进行状态更新。具体细节可以看程序，这里需要注意的一点是数据关联问题。关于这个问题，会在下一节进行细致描述。

4 数据关联

数据关联 (data association) 技术来源于多目标跟踪环境和传感器的观测过程之间的不确定性。这是因为在目标跟踪过程中，同一个目标可以在名个传感器上产生多个观测值，这些观测值不完全相似，主要是由杂波和传感器的系统误差导致，数据关联的目的就是要判定这些不完全相似的观测值是否来源于同一目标。

在 EKFSLAM 问题中，所谓的数据关联问题就是识别当前的路标是否是之前遇见过的，若是则说明是已知路标，若不是则是新路标。之所以要有这种区分，是为了在状态更新时分情况处理。这里分为两种状况：1) 观测到已知路标时，需要对已有的所有的状态变量和协方差进行更新；即使用卡尔曼滤波框架。2) 观测到未知路标时，需要将新路标的信息添加到状态变量中，同时更新协方差（换句话说，就是需要对状态变量和协方差进行扩充)；只进行状态更新，不进行更新。

[2] https://blog.csdn.net/qq_36355662/article/details/84836293

5 总结

基于卡尔曼滤波框架下的 SLAM 问题，需要知道载体的运动模型和量测模型；在状态更新时，不仅要更新位姿信息，landmark 点也要同时更新。但是这其中涉及到一个数据关联问题，即区分当前看到的 landmarke 点是已经存在在状态向量中还是新的点。这两种不同的范畴决定了算法下一步的操作。所谓的 localization 指的是状态向量中的位姿部分；所谓的 Mapping 指的是状态向量中的 landmarks 部分。