ArUco EKF SLAM

Dongsheng Yang, ydsf16@buaa.edu.cn, 2018 年 9 月 23 日星期日

引言

这篇文章实现了《概率机器人》第 10 章中提到的 EKF-SLAM 算法，更确切的说是实现了已知一致性的 EKF-SLAM 算法。

EKF-SLAM 一般是基于路标的系统。本文就使用了一种人工路标——ArUco 码。每个 ArUco 码有一个独立的 ID，通过 PnP 方法可以计算出码和相机之间的相对位姿。OpenCV 扩展库中集成了 ArUco 码库，提供了检测和位姿估计的功能。大家可以参考：https://docs.opencv.org/3.3.0/d9/d6d/tutorial_table_of_content_aruco.html。

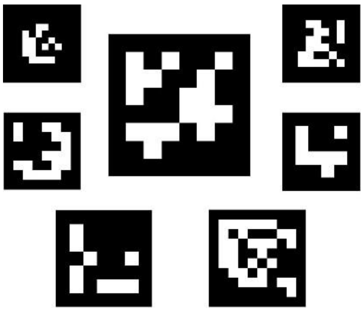

图 1　ArUco 码

首先在房间的地面上贴若干 ArUco 码作为路标，然后遥控一个带有摄像头+编码器的机器人在房间内运动。本文的目标就是通过 EKF 算法同时估计出这些码的位置和机器人的位姿。

图 2　实验环境

要实现 EKF-SLAM，最关键的就是建立运动模型和观测模型，将这两个模型直接带进 EKF 算法框架就是 EKF-SLAM。EKF-SLAM 算法使用扩展的状态空间：

$$\mathbf{X} = \begin{bmatrix} x & y & \theta & m_{x,1} & m_{y,1} & m_{x,2} & m_{y,2} & \cdots & m_{x,N} & m_{y,N} \end{bmatrix}^T$$

前 3 项是机器人位姿，后 $2N$ 项是 N 个路标点的位置。

运动模型

里程计模型

我比较喜欢采用《自主移动机器人导论》中的里程计模型作为运动模型。具体的，如果 t-1 时刻机器人的位姿是 $\xi_{t-1}=\begin{bmatrix}x & y & \theta\end{bmatrix}_{t-1}^{T}$，那么 t 时刻的机器人位姿为：

$$\begin{bmatrix}x\\y\\\theta\end{bmatrix}_t=\begin{bmatrix}x\\y\\\theta\end{bmatrix}_{t-1}+\begin{bmatrix}\Delta s\cos(\theta+\Delta\theta/2)\\\Delta s\sin(\theta+\Delta\theta/2)\\\Delta\theta\end{bmatrix},$$
$$\begin{cases}\Delta\theta=\dfrac{\Delta s_r-\Delta s_l}{b}\\\Delta s=\dfrac{\Delta s_r+\Delta s_l}{2}\end{cases},\Delta s_{l/r}=k_{l/r}\cdot\Delta e_{l/r} \tag{1}$$
$$\Delta s_{l/r}\sim N\left(\Delta s_{l/r},\quad\left\|k\Delta s_{l/r}\right\|^2\right).$$

$k_{l/r}$ 左右轮系数，把编码器增量 $\Delta e_{l/r}$ 转化为左右轮的位移，b 是轮间距。左右轮位移的增量 $\Delta s_{l/r}$ 服从高斯分布，均值就是编码器计算出的位移增量，标准差与增量大小成正比。如果 t-1 时刻机器人位姿的协方差为 $\mathbf{\Sigma}_{\xi,t-1}$，控制的协方差也就是编码器增量的协方差为 $\mathbf{\Sigma}_u$，那么机器人位姿在 t 时刻的协方差就是：

$$\mathbf{\Sigma}_{\xi,t}=\mathbf{G}_\xi\mathbf{\Sigma}_{\xi,t-1}\mathbf{G}_\xi^{\ T}+\mathbf{G}'_u\mathbf{\Sigma}_u\mathbf{G}'^{T}_u \tag{2}$$

$\mathbf{G}_\xi$ 是(1)式关于机器人位姿 ξ_{t-1} 的雅克比：

$$\mathbf{G}_\xi=\frac{\partial\xi_t}{\partial\xi_{t-1}}=\begin{bmatrix}1 & 0 & -\Delta s\sin(\theta+\Delta\theta/2)\\0 & 1 & \Delta s\cos(\theta+\Delta\theta/2)\\0 & 0 & 1\end{bmatrix} \tag{3}$$

$\mathbf{G}_u$ 是(1)式关于控制（编码器增量）$\mathbf{u}=\begin{bmatrix}\Delta s_r & \Delta s_l\end{bmatrix}^T$ 的雅克比：

$$\mathbf{G}'_u=\frac{\partial\xi_t}{\partial\mathbf{u}}=\begin{bmatrix}\frac{1}{2}\cos\left(\theta+\frac{\Delta\theta}{2}\right)-\frac{\Delta s}{2b}\sin\left(\theta+\frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta+\frac{\Delta\theta}{2}\right)+\frac{\Delta s}{2b}\sin\left(\theta+\frac{\Delta\theta}{2}\right)\\\frac{1}{2}\sin\left(\theta+\frac{\Delta\theta}{2}\right)+\frac{\Delta s}{2b}\cos\left(\theta+\frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta+\frac{\Delta\theta}{2}\right)-\frac{\Delta s}{2b}\cos\left(\theta+\frac{\Delta\theta}{2}\right)\\\frac{1}{b} & -\frac{1}{b}\end{bmatrix} \tag{4}$$

EKF-SLAM 运动更新

上面说的还是只考虑机器人位姿的情况，但是 SLAM 系统还需要考虑路标点。扩展路标点之后，运动方程为：

$$\underbrace{\begin{bmatrix} x \\ y \\ \theta \\ m_{x,1} \\ m_{y,1} \\ \vdots \\ m_{x,N} \\ m_{y,N} \end{bmatrix}_t}_{\mathbf{X}_t} = \underbrace{\underbrace{\begin{bmatrix} x \\ y \\ \theta \\ m_{x,1} \\ m_{y,1} \\ \vdots \\ m_{x,N} \\ m_{y,N} \end{bmatrix}_{t-1}}_{\mathbf{X}_{t-1}} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} \Delta s\cos(\theta+\Delta\theta/2) \\ \Delta s\sin(\theta+\Delta\theta/2) \\ \Delta\theta \end{bmatrix}}_{g(\mathbf{X}_{t-1},\mathbf{u}_t)} \tag{5}$$

系统状态的均值$\bar{\mathbf{u}}_t$更新利用(5)式，下面看状态的方差$\bar{\boldsymbol{\Sigma}}_t$更新。

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{G}_t\boldsymbol{\Sigma}_{t-1}\mathbf{G}_t^{\ T} + \mathbf{G}_u\boldsymbol{\Sigma}_u\mathbf{G}_u^{\ T} \tag{6}$$

$\mathbf{G}_t$是$g\left(\mathbf{X}_{t-1},\mathbf{u}_t\right)$ 关于$\mathbf{X}_{t-1}$的雅克比：

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{G}_\xi & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{7}$$

$\mathbf{G}_u$是$g\left(\mathbf{X}_{t-1},\mathbf{u}_t\right)$关于$\mathbf{u}_t$的雅克比：

$$\mathbf{G}_u = \mathbf{F}\,\mathbf{G}_u' \tag{8}$$

把(6)式展开看一下：

$$\begin{aligned} \bar{\boldsymbol{\Sigma}}_t &= \begin{bmatrix} \mathbf{G}_\xi & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \boldsymbol{\Sigma}_t \begin{bmatrix} \mathbf{G}_\xi^{\ T} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} + \mathbf{F}\mathbf{G}_u'\boldsymbol{\Sigma}_u\mathbf{G}_u'^{\,T}\mathbf{F}^T \\ & \begin{bmatrix} \mathbf{G}_\xi\boldsymbol{\Sigma}_{xx}\mathbf{G}_\xi^{\ T} & \mathbf{G}_\xi\boldsymbol{\Sigma}_{xm} \\ \left(\mathbf{G}_\xi\boldsymbol{\Sigma}_{xm}\right)^T & \boldsymbol{\Sigma}_{mm} \end{bmatrix} + \mathbf{F}\mathbf{G}_u'\boldsymbol{\Sigma}_u\mathbf{G}_u'^{\,T}\mathbf{F}^T \end{aligned} \tag{9}$$

可以看出，控制量的更新同时影响了机器人位姿的协方差，以及位姿与地图之间的协方差。

测量模型

首先解决测量值的问题。虽然可以获得 ArUco 码相对于机器人的 6 自由度位姿信息，但是为了与书上的观测统一，本文还是把相机作为 Range-bearing 传感器使用，也就是转换成距离r和角度ϕ。1 个 ArUco 码作为一个路标点$\mathbf{m}$，坐标为$\begin{bmatrix} m_x & m_y \end{bmatrix}^T$。

先说一下如何转化成距离和角度。下图是示意图，码与相机的相对位姿为${}_m^c\mathbf{T}$，相机与机器人的相对位姿为${}_c^r\mathbf{T}$，那么码相对于机器人的位姿为${}_m^r\mathbf{T} = {}_c^r\mathbf{T}\,{}_m^c\mathbf{T}$。${}_m^r\mathbf{T}$的平移项$x$和$y$就是码的原点在机器人坐标系下的坐标。转化成距离信息就是$r=\sqrt{x^2+y^2}$，角度就是ϕ=atan2$\left(y,x\right)$。这样就得到了测量值$z=\begin{bmatrix} r & \phi \end{bmatrix}^T$。这里再做一个近似假设，认为观测的标准差与距离和角度成线性关系：

$$\mathbf{Q} = \begin{bmatrix} \left\|k_r r\right\|^2 & \\ & \left\|k_\phi \phi\right\|^2 \end{bmatrix} \tag{10}$$

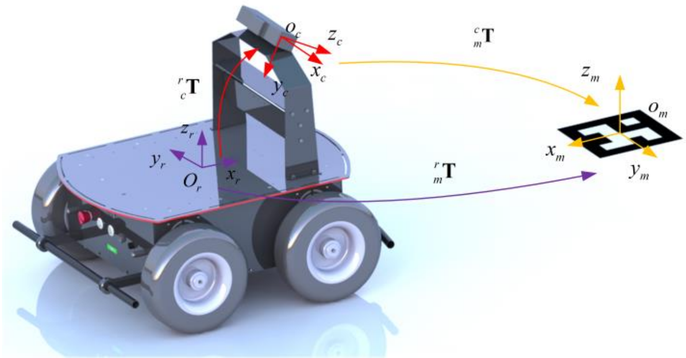

图 3 观测模型示意

第 i 个路标点的观测模型为：

$$\mathbf{z}_t^i = h\left(\mathbf{X}_t\right) + \boldsymbol{\delta}_t^i,\ \boldsymbol{\delta}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_t\right) \tag{11}$$

展开来看：

$$\begin{cases} r_t^i = \sqrt{\left(m_{x,i} - x\right)^2 + \left(m_{y,i} - y\right)^2} \\ \phi_t^i = \operatorname{atan2}\left(m_{y,i} - y,\ m_{x,i} - x\right) - \theta \end{cases} \tag{12}$$

根据扩展卡尔曼滤波，需要求解观测 $\mathbf{z}_t^i$ 相对于 $\mathbf{X}_t$ 的雅克比 $\mathbf{H}_t^i$，实际上一个路标点观测只涉及到机器人的位姿和这个路标点的坐标，组合在一起就是五个量：$v = \begin{bmatrix} x & y & \theta & m_{x,i} & m_{y,i} \end{bmatrix}$。于是，观测 $\mathbf{z}_t^i$ 相对于 v 的雅克比是：

$$\mathbf{H}_v = \frac{\partial h}{\partial v} = \frac{1}{q}\begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix}, \left(\begin{cases} \delta_x = m_{x,i} - x \\ \delta_y = m_{y,i} - y \end{cases},\ q = \delta_x^{\ 2} + \delta_y^{\ 2}\right) \tag{13}$$

由于实际的状态空间是3+2N 维的，要求的观测雅克比应该是 $2\times\left(3+2N\right)$ 维的。对(13)进行转换得到观测 $\mathbf{z}_t^i$ 相对于全状态空间 $\mathbf{X}_t$ 的雅克比：

$$\mathbf{H}_t^i = \mathbf{H}_v\mathbf{F}_i = \frac{1}{q}\begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & \underbrace{0\cdots0}_{2i-2} & 0 & 1 & \underbrace{0\cdots0}_{2N-2i} \end{bmatrix} \tag{14}$$

下面就可以按照 EKF 的框架进行操作了。

$$\begin{aligned} \mathbf{K}_t^i &= \bar{\boldsymbol{\Sigma}}_t\mathbf{H}_t^{iT}\left(\mathbf{H}_t^i\bar{\boldsymbol{\Sigma}}_t\mathbf{H}_t^{iT} + \mathbf{Q}_t\right)^{-1} \\ \boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t^i\left(\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i\right) \\ \boldsymbol{\Sigma}_t &= \left(\mathbf{I} - \mathbf{K}_t^i\mathbf{H}_t^i\right)\bar{\boldsymbol{\Sigma}}_t \end{aligned} \tag{15}$$

其中，

$$\hat{\mathbf{z}}_t^i = \begin{bmatrix} \sqrt{\left(\bar{m}_{x,i} - \bar{x}\right)^2 + \left(\bar{m}_{y,i} - \bar{y}\right)^2} \\ \operatorname{atan2}\left(\bar{m}_{y,i} - \bar{y},\ \bar{m}_{x,i} - \bar{x}\right) - \bar{\theta} \end{bmatrix} \tag{16}$$

就是由路标点和机器人位姿的均值获取。对每个观测到的路标点进行上述操作就完成了观测更新。

地图构建

上文所说的操作都是假设路标点的数量是已知的，这个值也可以认为是不知道的，可以边运行边加入路标点：当看到一个新的地图点时就扩展状态空间和协方差。当观测到一个新的路标点，其观测为 $z=\begin{bmatrix} r & \phi \end{bmatrix}^T$，根据机器人的位姿可以计算地图点的坐标为：

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} r\cos(\phi) \\ r\sin(\phi) \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} = r\begin{bmatrix} \cos(\theta+\phi) \\ \sin(\theta+\phi) \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \tag{17}$$

地图点的协方差为：

$$\mathbf{\Sigma}_m = \mathbf{G}_p \mathbf{\Sigma}_\xi \mathbf{G}_p^T + \mathbf{G}_z \mathbf{Q} \mathbf{G}_z^T \tag{18}$$

$\mathbf{G}_p$ 是(17)式关于机器人位姿的雅克比：

$$G_p = \begin{bmatrix} 1 & 1 & -r\sin(\theta+\phi) \\ 1 & 1 & r\cos(\theta+\phi) \end{bmatrix} \tag{19}$$

$\mathbf{G}_z$ 是(17)式关于观测 z 的雅克比：

$$G_z = \begin{bmatrix} \cos(\theta+\phi) & -r\sin(\theta+\phi) \\ \sin(\theta+\phi) & r\cos(\theta+\phi) \end{bmatrix} \tag{20}$$

通过以上各式，算出新路标的均值和协方差，加入到均值向量和协方差矩阵中即可。至此，EKF 算法中所有的模型都已建立完毕。下面给出具体的实施代码。

算法实现

全部工程代码：

https://github.com/ydsf16/aruco_ekf_slam

我用 Falconbot 机器人，采集了两组实验数据，大家可以在这里下载：

https://pan.baidu.com/s/1EX9CYmdEUR2BJh7v5dTNfA

参考文献：

《概率机器人》

《自主移动机器人导论》

Freiburg SLAM Course: http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/