# SLAM初学者手册

作者：孔大庆 日期：2019-7-9

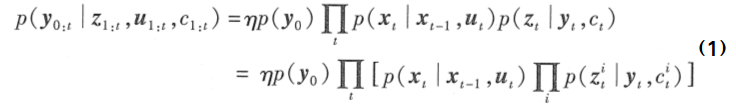
## 前言

SLAM初学者一般很难获得较好的SLAM学习资料，原因一是关于SLAM领域的研究在上世纪末和本世纪初才有较大的进展，因此没有较完整和全面的教学材料；另一方面，SLAM涉及的知识体系庞杂、理论艰深难懂，SLAM在自身不断发展的同时与AI等领域的融合使得初学者望而生畏、无从下手。本文结合自身自学SLAM的历程，将经验分享给大家，仅供参考。

本文主要介绍SLAM的知识结构以及个人的学习方法，将SLAM所涉及的知识点不分大小一一罗列，如有错误欢迎指正。

## SLAM发展历程

上个世界80年代到现在，围绕SLAM问题解决大致可以划分为两个时代，主要是在解决式1）时采用了两种不同的方法：滤波器和非线性优化。1）中状态X包括机器人的位姿和路标点的位置，Z通过传感器对外界环境的观测，求解SLAM问题实际上是最大后验问题，使得在观测Z的条件下X的概率最大。使用贝叶斯估计，后验 = 似然x先验，而在处理先验更新的时候出现了分歧：其一是假设马尔科夫性，简单的一阶马氏性任务，k时刻状态只与k-1时刻的状态有关，而与之前的状态无关，这样假设即可得到以EKF为代表的滤波器方法；其二是依然考虑k时刻状态与之前所有状态的关系，此时将得到非线性优化方法。

 SLAM前期研究者们基本上都是使用EKF滤波器的方法，当然也有其他形式的滤波器如IF(信息滤波)、UKF和PF（粒子滤波器）等。然而研究者们逐渐意识到滤波器的弊端：主流的EKF本身存在很多问题，一个是实际的运动方程和量测方程都是高度非线性的模型，EKF所维持的均值和协方差只是真实非线性分布的线性高斯近似，实际的估计效果受到模型的非线性程度制约，解决非高斯分布的问题代表的方法有粒子滤波，用粒子分布来近似表达真实的状态分布，但是实际使用粒子滤波也会遇到计算量庞大和粒子耗散等问题。EKF另一个致命问题在于它的马尔科夫性假设，实际我们处理SLAM问题时，经常会碰见机器人经过一段时间探索回到原先位置的情形，在SLAM上称为“回环”，一旦出现“回环”我们可以当前位置与原先位置关系修正累计误差，然而对于建立在马氏假设的EKF来说，这种状态重叠的约束无从加入。

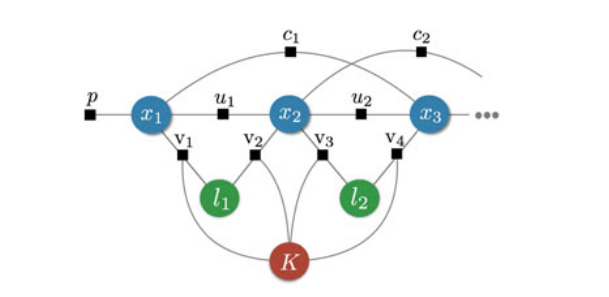
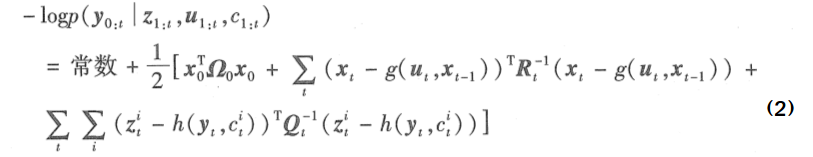
为什么要用前一个时刻的值来估计下一个时刻呢？我们可以把所有状态看成变量，把运动方程和观测方程看成变量间的约束，构造误差函数，然后最小化这个误差的二次型。这样就会得到非线性优化的方法，在SLAM里就走向图优化这条路上去了。如图1，我们将所

图1 SLAM图优化示意图

有的状态和路标点看做是图中的node，位姿间node是通过运动方程实现约束（如图中u1,u2），位姿和路标node则是通过量测方程(如图中v1-v4)形成约束，当形成闭环的时候间隔很远的位姿node也会形成约束（如c1，c2），甚至可以观测方程来估算相机固有参数K。具体实现的时候，实际将（1）式转化为负对数形式，讲最大后验估计问题转化为非线性最小二乘问题。



它与迭代递推的滤波器方法有很大的差别，滤波器是实时对每一帧数据进行迭代处理，更新状态的置信度信息，而上图1中的方式是类似SFM中BA优化的离线批处理（batch）方式。实际图优化就是受到BA的启发，为了实现这种优化同时达到实时估计状态量的目的，逐渐形成了标准的结构——前端+后端。

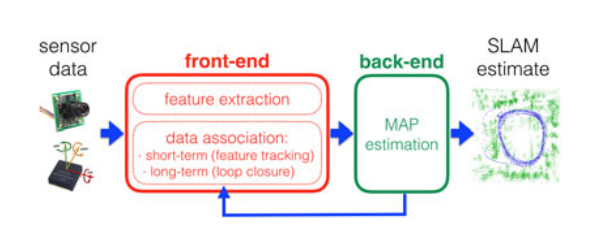
后端其实就是执行式2）非线性最小二乘优化的模块，而为什么要有前端呢，其实很简单，实际上我们使用的测量传感器有很多种，如IMU、GPS、摄像头、激光等，这些传感器测量的数据并不能直接给后端处理，比如我们给后端一堆图像有什么意义呢？因此，这时候我们需要能够将传感器数据翻译为模型中数据的模块，更重要的是这个模块还要负责数据关联即告诉后端所有node存在的约束，通常可以分为短期和长期的，短期指帧间或者局部帧间的约束如u1-u2,v1-v4，而长期指回环的约束如c1、c2。事实上，考虑到SLAM长时间运行，计算资源很难允许图优化的节点无限制增长，处理这种情况一般是使用滑动窗口的方法，只维持一定数据量的节点及其约束信息，这要求我们删除之前的节点信息。为了保留要删除点的信息需要对信息矩阵边缘化处理，这个操作实际上会将剔除节点的约束转到保留节点对应的信息矩阵位置上。

图2 非线性优化中的前端+后端结构

虽然现在SLAM主流的方法是非线性图优化的方法，但针对不同的方案和平台或者环境，滤波方法可能会更好。比如MSCKF（Multistate Constraint Kalman Filter）和VINS（Visual-Inertial Navigation System）,这些方法在考虑了线性化误差、不一致性等因素表现不比非线性优化方法差，所以要灵活运用。

## SLAM知识结构

目前SLAM根据环境感知传感器的不同，大体分为激光和视觉SLAM。激光slam感知环境的数据直接、简单且可靠，可以直接获得环境物体到传感器的距离和方位数据。同时利用ICP，ICL等帧间匹配算法可以获得帧间位姿信息。视觉SLAM相对复杂一点，图像数据必须通过多视图几何的方式来提取其中的运动信息，而这建立在正确的特征匹配上，因此这部分要求SLAM开发者具备一定CV知识。而本人在学习的时候也是偏向视觉SLAM方面，所以优先把视觉SLAM的知识结构介绍给大家，可参照知识星球SLAM知识树，如图3所示。这里主要关注知识树右侧的知识点，分为数学知识、专业知识和编程环境。

1. 数学知识

首先是矩阵知识。矩阵是图像的载体，我们对图像的处理基本上都是对图像矩阵的操作。同时，学习投影几何的人也知道，投影变换都是用矩阵形式表达的，而后端优化时主要的操作是对信息矩阵的处理，特别是做非线性优化求解增量方程的时候，需要用到矩阵的QR分解、Cholesky分解等知识，因此矩阵知识无可或缺。

其次是李群与李代数的知识，这部分知识比较抽象。不过实际上是一种映射，我们在进行非线性优化的时候，需要用迭代求解增量的形式来求解位姿信息，而旋转矩阵并没有对应的加法，且自身还有很多约束，为了解决这个问题，需要用旋转矩阵正切空间上对应的李代数来表示旋转，因为李代数良好的性质使得非线性优化能够进行。

Cost Function: F(x) = ||f(x)||

还有最重要的是非线性优化方面的数学知识，该知识是图优化的核心。知识树上给了五中非线性优化方法，其实还有很多变种。非线性优化的通用流程是选取一个初始状态值x0，然后求取∆x使得cost function最小，如果∆x足够小则停止，否则继续进行，直到到达要求。如果cost function是凸函数，经过有限步迭代一定可以找到全局最优点。然而实际cost function有多个波峰和波谷，迭代寻优很可能找到局部最优解，因此需要一个比较好的初始值x0。因为cost function是非线性的难以求解析解，这里有两类方法：直接对F(x)进行泰勒展开，保留到一阶的成为梯度下降法，保留到二阶的成为牛顿法；另一种是对f(x)进行一阶泰勒展开，然后求解F(x)。这些方法都是将求解∆x转化为线性方程，不过这其中的差异很大。比如梯度下降法，只要取∆x=-J’(x) (J(x)为F(x)的一阶导数矩阵，称为雅克比矩阵)，那么就能确保cost function按照收敛方向迭代，但是实际迭代路径并不是最快的，而且越靠近最优解迭代速度就越慢。而牛顿法取∆x=-J’(x)/H(x)（H(x)是F(x)二阶矩阵，称为海森矩阵），牛顿法因为考虑了二阶信息，所以收敛速度快，但是海森矩阵求逆是很复杂的，而且函数曲率很小即H(x)接近0时候，∆x变化很大，x可能会跳出最优解的区域。为了解决海森矩阵计算复杂问题，可以直接对f(x)进行一阶泰勒展开，然后求解F(x)，这时候可以得到增量方程为∆x=-J’(x)f(x)/(J’(x)J(x))，相当于用J’(x)J(x)替代海森矩阵，只需要求雅克比矩阵即可，这种方法成为高斯牛顿法。但实际还是无法避免当曲率小的时候∆x变化较大的问题，而这种时候不如使用梯度下降法，因此出现LM优化的方法，取∆x=-J’(x)f(x)/[rI+(J’(x)J(x))]该方法是将高斯牛顿法和梯度下降法进行组合，调整r值可以使∆x偏向梯度下降或者高斯牛顿。

Boudle Adjustment光束法平差，是上个世纪处理图像三维重建SFM使用的方法。它的思想是最小化重投影误差来优化位姿和路标点，实际使用的也就是上面提到的LM等非线性优化的算法。

1. 专业知识

专业方面需要我们掌握概率论、滤波理论（多传感器数据融合）、数学建模以及计算机视觉方面的知识。概率论使我们能够将SLAM问题抽象为数学表达的理论基础，这里需要我们着重掌握贝叶斯理论的思想，对于其中的KF（EKF）、IF和PF原理应该能够理解和推导。

滤波理论方面，我们应该明白最大后验、最大似然、最小方差以及最小二乘等滤波的思想，甚至应该能够推导其中的原理。基于EKF的导航方面的多传感器数据融合算法应该能够推导五个方程，也可以对比分析一下基于IF的一些方法，能够比较其中的优缺点。

数学建模能力也需要培养，不同的机器平台和传感器其运动模型和量测模型有很大的区别，我们应该能够准确的对这些运动和量测关系建立出数学模型，同时要考虑到其中的误差信息。

对于视觉SLAM我们还要了解一部分的CV知识，主要是特征提取和匹配、多视图几何等相关知识。但是每一个部分背后的方法和理论都很深奥，需要潜心学习。

1. 编程环境

最后就是项目实现方面的知识了。首先是开发环境：ubuntu+ros+qt+cmake，Ubuntu+ros是目前slam开发最常用的环境，ros不仅提供了方便的节点通信机制、大量的可视化和调试的工具，同时提供了丰富的导航、控制和路径规划等功能库，是机器人开发首选的软件平台。Qt是个人比较喜欢的编译环境，其针对ros而开发的插件qtcreator-ros让ros开发更加便利。Cmake是编译工具，需要我们了解cmake基本操作和常用编译语言。

其次就是第三方库了，上面提到的计算机视觉要进行的特征提取、匹配等操作，均可以使用opencv库来处理，而做非线性优化的时候需要使用g2o或者ceres库，这三个库建议要详细了解，不仅要学会使用库函数还要明白其中的原理。之后就是用于表达几何变化的Eigen库，很多库都依赖它，以及用于李代数表示的sophus库。最后是PCL库，用于处理点云的功能库，其提供的功能函数也很多，可以用来做点云滤波、特征提取、配准以及模式识别等。

总体来说，SLAM对从业者的数学功底、专业知识和编程能力等要求都非常高，我们需要认清这个实时，踏踏实实循序渐进的进行学习。

## 如何学习SLAM

下面我结合自身的经验谈谈如何学习SLAM。学习SLAM是一个非常巧合的事情，当时是同事买了一本《SLAM十四讲》的书自己无从下手最终弃而学深度学习了（PS：据说深度学习学起来简单的多），我之前研究生阶段师兄做过激光slam，当时觉得很高大上，本人自身是学控制出身，奈何控制方面的资质过浅所以很想转导航或者其他方向领域，所以机缘巧合学起了SLAM。跑题了，言归正传。一开始就是按照十四讲上每一章看，一遍看一遍做，感觉还行吧，除了李代数不是很懂，其他基本上都明白，这样进行到第九章左右，后边提到的后端优化就进行不下去了，你自己会想为什么要前后端分开处理？这种好处是什么？BA到底是个啥？后端非线性优化为什么可以处理SLAM问题？等等吧，所以开始迷茫，在逛各种论坛和网站中发现自己缺少了SLAM理论方面的知识，特别是在看了《概率机器人》这本书以及SLAM发展历程的介绍之后有种豁然开朗的感觉，所以我们对于一个完全未知的行业还是有必要去了解它的发展历程、经典书籍以及经典文献的。虽然说现在系统学习SLAM的书籍和资料很少，但是还是能找到不少大牛总结的资料。总体来说，这是一个循序渐进的过程。

为了后人少走弯路，我建议优先了解SLAM理论知识以及发展历程，这样有个宏观的了解后，学习过程会更有针对性。然后找到适合入门的资料，比如《SLAM十四讲》等，开始真正学习和实践，切勿眼高手低只顾理论理解，实践能够检验你真正的掌握程度，这个过程中可以看一些大牛总结性的资料，你会找到一些疑难问题的答案。这之后就是看一些有名的开源SLAM项目代码了，如ORB-SLAM2、cartographer、gmapping、PTAM、OKVIS、VINS-MONO等等，将之前学到的理论与之对应，查漏补缺。最后就是能够模仿开源项目尝试自己搭建SLAM系统。

个人推荐学习流程：

了解阶段：

可以参照目录“References/slam概述”下所有的文章，其中《slam tutorials1/2》是2006年左右slam研究的概述，而另一篇《past，present and future of slam》是2016年的，两篇都建议仔细研读。研究slam发展过程和关键技术，特别是仔细查找参考文献中对应的关键技术。同时，请仔细研读slam入门经典书籍《概率机器人》，我讲其放在“Books/SLAM专业知识”目录下面。这本书算是SLAM发展中期开始由滤波过渡到非线性优化阶段的作品，不仅对传统滤波器方法做了深入签出的介绍，同时对后面图优化方法也有很详细的原理说明，真的是slam入门首选。读懂这本书的难度还是很大的，个人觉得可以循序渐进，第一次可以了解大概，等对slam学习有了深度后可以重读。

入门阶段：

对slam有了简单了解后，我们开始真真正正的尝试做slam，这里强烈推荐高博《slam十四讲》。这本书是关于视觉slam的理想教程，高博首先介绍了视觉slam的框架，然后对slam的前端、后端以及其中涉及的知识点做了较为全面的介绍，而且很容易理解，理论性没有那么强，更可贵的是，他每一章还提供了代码教程，我们可以边学边做，特别适合入门。入门阶段需要我们讲了解阶段的理论融会贯通，我们可以从原理和实践两方面在自己大脑里形成一个成熟的知识框架，这个阶段需要我们打好slam基础。

这个阶段建议大家阅读目录“Books/SLAM快速入门”下面的两个手册，一个是偏向EKF滤波器的SLAM指导手册，另一个是非线性优化的。本人阅读了非线性优化的slam手册，收益匪浅，再昨晚slam十四讲后用这个手册，完全是从另一个角度来巩固和学习slam，强烈推荐这个手册。

深入阶段：

这个阶段建议学习slam开源代码，前提是你已经学会使用slam的开发平台：能够熟练进行Ubuntu+ros+cmake环境的代码开发。我们可以先从简单的开源slam看起，比如前后端的开创PTAM，然后经典的ORB-SLAM，也可以看结合IMU的VINS-MONO。其实，一定不能陷入代码，我们需要结合他们的论文了解代码的实际目的，可以学习代码的对于slam问题的处理技巧，仅此而已。

从这些经典的开源SLAM中，我们要逐渐形成自己slam代码的开发流程和实现框架，继承其中的优点，然后可以分部分分阶段的写自己的slam代码，最终实现整个完整slam代码开发。

实践阶段：

这部分需要我们讲自己代码用于产品功能的实现，要考虑到可靠性、鲁棒性等，做到slam运行稳定可靠并不容易需要一步一个脚印，因为自己还没有这方面经验，这里就不做说明了。