

视觉SLAM中的数学基础 第一篇 3D空间的位置表示

前言

转眼间一个学期又得过去，距离我上次写《一起做RGBD SLAM》已经半年之久。《一起做》系列反响很不错，主要由于它为读者提供了一个可以一步步编码、运行的SLAM程序。为读者理解SLAM实现的细节作了详细的介绍。但是我也有很多对它不满意的地方。作为面向实现的介绍，它的代码不够稳定可靠。例如，甚至没有对匹配丢失的情况进行处理。因而只能用于教学。另一方面，对SLAM研究来说，我只是介绍了编码方面如何调用一些常见的库函数，而没有对这些函数进行深入的、原理上的讲解。这就导致了读者只了解了函数的接口，而没法根据数学原理进行创新。归根到底，研究机器人相关问题，一是要有扎实的数学基础。二是要有强大的动手编程能力。这对大多数刚入门的研究者来说，极具挑战性。我也希望，通过阅读我的博客，你能走进SLAM研究的门庭，有朝一日自己也写出优秀的程序和论文。

有鉴于此，我准备写一篇SLAM相关的数学知识，包括代数、几何、概率、运筹等等。对于重要的算法例如ICP、EKF，细数讨论它的原理，并给出它的实现（原生的代码或在某个库的实现）。由于它们的原理较复杂，我会从最基本的东西开始讲起。但是我毕竟不是在写数学书，我不会像数学书那样写成“`定义——定理——推论”的结构。我们不会纠缠于一些定理的严格证明，相反的，我们只在必要的情况下加以说明，告诉读者这些数学公式在SLAM中有何应用，如何应用。

由于博客编辑器的限制，我们以斜体 \mathbf{x} 表示变量，以粗正体 \mathbf{X} 表示矢量和字母，以黑板粗体 $\mathbf{\hat{X}}$ 表示空间。希腊字母没有粗体所以保持原样。向量默认为列向量。其余和普通的数学书一致。

小萝卜：师兄，这么严肃不是你的风格啊！

师兄：啊，数学嘛.....

刚体运动

本篇我们讨论一个很基础的问题：如何描述机器人的位姿。这也是研究SLAM的第一个问题。注意这里“位姿”的用语包含了位置和姿态。描述位置是很简单的。如果机器人在平面内运动，那么用两个坐标来描述它的位置：

$$= [x, y]$$

相应的，如果它在三维空间中，我们就用三个空间坐标来表示：

$$= [x, y, z]$$

姿态的表达比坐标为复杂。2D的姿态可以只用一个旋转角 θ 表达。3D姿态的表达方式则有多种。常见的如欧拉角、四元数、旋转矩阵等。我们将在后文详细介绍。有了位置和姿态，我们就可以描述一个坐标系。进一步，还能描述坐标系间的变换关系。常见的问题如：机器人视野中某个点，对世界坐标系的（或地图的）哪个点？这时，就需要先得该点针对机器人坐标系坐标值，再根据机器人位姿变换到世界坐标系中。

齐次坐标系

在位姿变换中，通常采用射影空间的齐次坐标表示。齐次坐标是什么呢？记 维射影空间为 \mathbb{P}^n ，其中一个空间点的坐标为普通的3D坐标加一个齐次分量：

$$= [x_1, \dots, x_n, 1]$$

例如，在2维和3维射影空间中的点，分别表示为：

$$\begin{matrix} \mathbf{p}_2 = [x, y, 1] \\ \mathbf{p}_3 = [x, y, z, 1] \end{matrix} \quad (1)$$

小萝卜：既然一个空间点只有3个坐标，为啥非要用四个数表示呢？

师兄：想，四个数表示点，说明点和坐标肯定不是一一对应的。没错，在齐次坐标中，某个点的每个分量同乘一个非零常数后，仍然表示的是同一个点。因此，一个点的齐次坐标值不是唯一的。如 $[1, 1, 1, 1]$ 和 $[2, 2, 2, 2]$ 是同一个点，但在 $\neq 0$ 时，我们可以对每个坐标除以最后一项，强制最后一项为1，从而得到一个点唯一的坐标表示：

$$\mathbf{p}_3 = [x, y, z, 1] \quad (2)$$

这时，忽略掉最后一项，这个点的坐标和欧氏空间是一样的。那么，为什么要用齐次坐标呢？原因有以下几条。

- 齐次坐标下点和直线（高维空间里为超平面）能够使用同样的表达。

例如，3D空间 \mathbb{R}^3 中，一个平面可由一个方程定义：

$$ax + by + cz = d \quad (3)$$

则该平面 Π 可以用 \mathbb{P}^3 中的坐标 $\mathbf{\pi} = [a, b, c, d]^T$ 来描述。这样，点位于平面上（2D对应点位于直线上）的事情可以简洁地表示为：

$$\mathbf{p}^T \mathbf{\pi} = 0 \quad (4)$$

把点和超平面采用同样的表示，这种做法一个非常直接的好处，是射影几何里的“对偶原理”。该原理是说，任何有关“点”与“平面”的命题，都可以交换“点”与“平面”的概念，得到一个对偶的命题。对偶命题和原命题是一样的。通过“对偶原理”，射影几何的数学家就可以偷懒，只需要证一半定理，因为对偶命题和原命题有同样的涵义。例如，我们证明了 \mathbb{P}^2 中某条件下三点共线，那么替换概念后的三线共点则自然成立。

小萝卜：数学家真是好偷懒啊！

- 齐次坐标能囊括无穷远点与无穷远超平面。

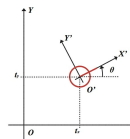
最后一项坐标为零的点称为无穷远点。它们在 \mathbb{P}^n 中真实存在，且能够很方便地参与正常的代数运算。根据式4，易见所有无穷远点都在一个平面 $\omega = [0, 0, 0, 1]^T$ 上，该平面记作无穷远平面（2D对应无穷远直线）。 \mathbb{P}^2 中的无穷远直线较容易理解，它就像是地平线，与所有直线相交于位于它之上的无穷远点。而且，在射影变换中（例如照相），很容易在照片中看到地平线并算出它的方程。这说明2D射影变换会把无穷远直线变成通常的直线。

- 齐次坐标可以方便地将平移与旋转放在一个矩阵中。

师兄：这应该是最明显的好处啦！大家都爱用齐次坐标，包括我。有关坐标系怎么用齐次坐标进行变换，后文会详细解释。现在我们能表达点了，还剩下一个姿态。由于2D与3D差别较大，我们分而述之。

2D姿态的描述

2D空间中，物体的位姿可用两个平移量 t_x, t_y ，加一个旋转角 θ 表示。如下图所示，此时，设机器人坐标系 $\mathbf{x}' - \mathbf{y}'$ 下某点的坐标为 $[x', y']^T$ ，对应在世界坐标系 $\mathbf{x} - \mathbf{y}$ 下为 $[x, y]^T$ ，那么由直观得：



$$\begin{cases} x = t_x + x' \cos \theta - y' \sin \theta \\ y = t_y + x' \sin \theta + y' \cos \theta \end{cases} \quad (5)$$

读者可以自行尝试推导一下，在虚线处建立一个中间坐标系即可。若将该式写成矩阵形式，则有：

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \quad (6)$$

其中

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

称为旋转矩阵， \mathbf{t} 为平移矢量。注意到 \mathbf{R} 是一个正交矩阵，且只有一个自由度。加上平移矢量后，一共有三个自由度。

此定义下的旋转阵必是正交阵。而正交阵并非全是旋转矩阵。事实上，行列式为+1的正交阵才是旋转矩阵，行列式为-1的正交阵是镜像后的旋转矩阵。

式6中， \mathbf{R} 和 \mathbf{t} 还不是线性关系。下面我们用齐次坐标表示它们，即：

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

则有：

$$\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}' + \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix} \quad (7)$$

为便于理解，我们在矩阵下方标出了它的维数。可以看出使用齐次坐标满足了线性关系，记作：

$$\mathbf{p} = \mathbf{H} \mathbf{p}' \quad (8)$$

其中， \mathbf{H} 表示从世界坐标系到机器人坐标系的变换矩阵。我们也可以轻松地写出反向的变换矩阵：

公告

昵称：半闲居士
园龄：8年10个月
粉丝：3062
关注：0
+加关注

2023年1月									
<	日	一	二	三	四	五	六		>
	1	2	3	4	5	6	7		
	8	9	10	11	12	13	14		
	15	16	17	18	19	20	21		
	22	23	24	25	26	27	28		
	29	30	31	1	2	3	4		
	5	6	7	8	9	10	11		

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

视觉SLAM(17)
机器人(14)
SLAM(13)
一起做RGB-D SLAM(7)
Kinect(4)
rgbdslam(3)
计算机视觉(2)
图像处理(2)
视觉SLAM漫谈(2)
李群(2)
更多

随机分类

随笔(2)
一起做rgbd slam(2)

随机标签

2016年8月(1)
2016年7月(1)
2016年6月(2)
2016年3月(2)
2016年2月(2)
2016年1月(8)
2015年12月(1)
2015年8月(4)
2015年7月(4)
2015年4月(2)
2014年6月(1)
2014年4月(1)

阅读排行榜

- 视觉SLAM漫谈(211589)
- 一起做RGB-D SLAM (1)(137824)
- 一起做RGB-D SLAM (2)(118148)
- 深入理解图优化与g2o: g2o篇(115570)
- 视觉SLAM实战(一): RGB-D SLAM V2(110972)

评论排行榜

- 一起做RGB-D SLAM (2)(77)
- 一起做RGB-D SLAM (5)(66)
- 一起做RGB-D SLAM (3)(66)
- 一起做RGB-D SLAM (6)(64)
- 一起做RGB-D SLAM (4)(62)

推荐排行榜

- 视觉SLAM漫谈(71)
- 一起做RGB-D SLAM (2)(34)
- 一起做RGB-D SLAM (1)(31)
- 深入理解图优化与g2o: g2o篇(25)
- 视觉SLAM漫谈(三): 研究点介绍(25)

最新评论

- Re: 一起做RGB-D SLAM (5)
@going_go 你好 我现在也是碰到这个问题。想问下次佬是怎么解决的...
--养朱的锅
- Re: 一起做RGB-D SLAM (4)
大家好，我在执行bin/detectFeatures时出现以下错误：oodboy@goodboy-virtual-machine: ~/slam\$ bin/detectFeatures bin/dete...
--喵喵喵喵喵
- Re: 一起做RGB-D SLAM (1)
博主，您好，我在执行第三部分bin/detectFeatures时出现bin/detectFeatures: symbol lookup error: bin/detectFeatures: unde...
--喵喵喵喵喵
- Re: 视觉SLAM实战(二): ORB-SLAM2 with Kinect2

= 0 = -1 = 1 [0 1] (9)

既然如此，我们就可用 表示机器人的位姿，那么机器人在时刻 的位姿就可以记作 。当然，从存储上来讲，存储 是不经济的。在2D运动中，它有九个变量，但实际上自由度只有三个，所以我们可以只存储位移向量 与旋转角 ，而在需要计算的时候再构建出 ，称2D欧几里得变换。它对矩阵乘法构成群(群是一个集合加一种运算，且运算在该集合上满足封闭性、结合律、有单位元和逆元。)，该群记作(2) 。相应的，二维旋转构成二维旋转群(或称特殊正交群)(2) 。有关它们进一步的性质，我们会在以后的李群、李代数中提到。

5. Re:一起做RGB-D SLAM (3)

请问为什么我的旋转矩阵和您的差一个负号呀

--Mai_Ql

3D变换

3D的旋转可以由旋转矩阵、欧拉角、四元数等若干种方式描述，它们也统称为三维旋转群(3) 。而3D的变换即旋转加上位移，是为(3) 。为了和2D变换统一起见，我们首先介绍旋转矩阵表示法。

旋转矩阵描述

旋转矩阵是一种3 × 3 的正交矩阵，它对变换的描述十分类似于2D情形。参照上一节的数学符号，我们有：

$$= \begin{bmatrix} 3 \times 3 & 3 \times 1 \\ \mathbf{0}_{1 \times 3} & 1_{1 \times 1} \end{bmatrix} \quad (10)$$

这里 为3D的旋转矩阵，同样的， 为3D的平移矢量。

由于3D旋转都可以归结成按照某个单位向量 进行大小为 的旋转。所以，已知某个旋转时，可以推导出对应的旋转矩阵。该过程由罗德里格斯公式表明，由于过程比较复杂，我们在此不作赘述，只给出转换的结果：

$$(\cdot) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (11)$$

这里 $\begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ， $\cos(\theta)$ ， $\sin(\theta)$ ，公式虽然较为复杂，但实际写成程序后，只需知道旋转方向和角度后即可完成计算。另一件有趣的事是，如果用

$$\Lambda = \begin{bmatrix} 0 & -\theta & 0 \\ \theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

表示与 对应的一个反对称矩阵，那么有：

$$(\cdot) = \cos(\theta) I + \sin(\theta) \Lambda = \exp(\Lambda \theta) \quad (12)$$

最后那个指数，读者若不理解，可以暂时不管它，这将在之后的李代数中会讲到。根据此式，我们也可以从任意给定的旋转矩阵，求出对应的转轴与转角。关于转角，我们对上式两边求矩阵的迹，可得：

$$\begin{aligned} \text{tr}(\cdot) &= \cos(\theta) + (1 - \cos(\theta)) \cos^2(\alpha) + \sin^2(\alpha) \\ &= 3 \cos(\theta) + (1 - \cos(\theta)) \\ &= 1 + 2 \cos(\theta) \end{aligned} \quad (13)$$

因此：

$$\theta = \arccos\left(\frac{\text{tr}(\cdot) - 1}{2}\right) \quad (14)$$

关于转轴，由于旋转轴上的向量在旋转后不发生改变，说明

$$\Lambda \cdot \mathbf{v} = 0$$

因此，只要求此方程的解向量即可。这也说明 是 特征值为1的一个特征向量。

总之，读者应当明白在3D时，旋转和平移仍可用转移矩阵 来描述，其结构也与2D类似。而 4×4 构成了三维欧氏变换群(3) ，注意到 虽然有16个变量，但真正的自由度只有6个，其中3个旋转，3个位移。

旋转矩阵描述是一种比较适合数学推导及计算机实现的方式，但它不符合人类的思维方式。当你看到一个3 × 3 的矩阵时，很难想象物体实际上发生了怎样的旋转。反之，给定一个旋转方式，人也很难直接写出矩阵的数值。所以，为了便于人类理解，人们还使用了其他方法来表示三维旋转。

欧拉角

欧拉角是一种广为使用的姿态描述方式，以直观见长。在最常用的欧拉角表达式中，我们把旋转分解成沿三个轴转动的量：滚转角—俯仰角—偏航角(roll-pitch-yaw)。它的好处是十分的直观，且只有三个参数描述。缺点是会碰到著名的万向锁问题：在俯仰为±90°时，表达某个姿态的形式不唯一。此外，它也不易于插值和迭代。

我们并不会详细介绍欧拉角，因为它在SLAM中用处也很有限。我们既不会在数学推导中使用它，也不会在程序中用欧拉角表示机器人的姿态。不过，在想要验证自己算法输出的姿态是否有误时，转换成欧拉角能让你快速辨认结果是否有误。

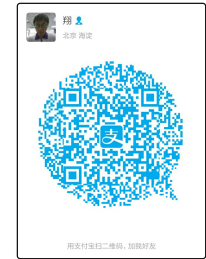
四元数

四元数原理和各种运算将在下一篇博客中提到。

本篇小结

本篇博客介绍了2D和3D空间中刚体运动的表示方法，以旋转矩阵为主。下一篇我们将介绍四元数表示法，然后演示如何用Eigen3对这些矩阵进行操作。敬请期待。

如果你觉得我的博客有帮助，可以进行几块钱的小额赞助，帮助我把博客写得更好。



标签: SLAM, 视觉SLAM, 数学基础



半闲居士
粉丝 - 3062 关注 - 0
+加关注

« 上一篇: SLAM拾遗(2).doxygen
» 下一篇: 视觉SLAM中的数学基础 第二章 四元数

posted @ 2016-01-08 16:18 半闲居士 阅读(37765) 评论(9) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

推荐阅读：

- 深入理解 Linux 物理内存分配全链路实现
- 巧用视觉惯导法，还原 3D 文字特效
- MassTransit | 基于 StateMachine 实现 Saga 编排式分布式事务
- 一次 SQL 调优，聊一聊 SQLSERVER 数据页
- 终于弄明白了 RocketMQ 的存储模型

阅读排行：

- 巧用视觉惯导法，还原 3D 文字特效
- 火热的低代码到底是啥？
- C#开发的超级屏幕类库 - 开源研究系列文章
- SQLSERVER 居然也能调 C# 代码？
- MongoDB从入门到实战之.NET Core使用MongoDB开发ToDoList系统(2)-Sw