# SLAM 学习教程

# 日志：

p1-p15

p16-p25

# 1. 数学基础

线性代数： 矩阵，向量，矩阵求逆，特征值分解，奇异值计算，最小二乘法

卡尔曼滤波理论，

非线性优化：最小二乘法，梯度下降，优化机器人位置和姿态估计，地图构建

微积分：包括导数，积分，微分方程，用于机器人运动模型和传感器模型

概率论和统计学： 贝叶斯推断，高斯分布，最大似然估计，常用于状态估计和传感器数据融合

图论与优化：最小生成树，图遍历，用于图优化

## 1. 向量

向量： 有长度的箭头，或是一个有序的数组，它定义在一组基坐标系中，满足可加性以及缩放性

基向量： 向量空间的一组基是张该空间的一个线性无关的向量集，所以任意两个不同线的二维向量可以作为一个二维空间的一组基

```
#include <iostream>
#include <Eigen/Dense>

using  namespace std;
int main() {
    //定义一个三维向量， 3 代表 3维  d代表  double 类型
    Eigen::Vector3d v1(1.0,2.0,3.0);

    Eigen::Vector3d v2(1.0,2.0,3.0);

    cout<<"v1 = "<<endl<<v1;
    cout<<endl;
    cout<<"v2 = "<<endl<<v2;
    Eigen::Vector3d sum=v1+v2;
    cout<<endl;
    cout<<"v1+v2 = "<<endl<<sum;

    return 0;
}
```

## 2. 内积与外积

内积： 也称点积，指的是两个向量之间的乘积，例如 对于长度为n的向量a和b，他们的内积为

$$a * b = a1 * b1 + a2 * b2 + a3 * b3.....an * bn \tag{3}$$

其中,ai和bi 分别为 向量a和b在第i个维度上的分量

**内积满足 交换律，结合律，分配律**，向量长度的平方等于向量的内积（自己与自己的内积）

外积： 也称叉积 ， 是指可以得到以一个与两个向量都垂直的新向量，对于长度为3的向量a和b，他们的外积为

$$aXb = （a2 * b3 - a3 * b2, a3 * b1 - a1 * b3, a1 * b2 - a2 * b1) \tag{4}$$

**外积满足反交换律 结合律，** （反交换律是指在某些情况下，两个元素的交换顺序会改变运算结果）

```
    double dot_product = v1.dot(v2);
    cout<< "v1*v2 = " <<dot_product<<endl;
    //外积 叉乘
    Eigen::Vector3d corss_product = v1.cross(v2);
    cout<<endl;
    cout<<"a X b = "<<endl<<corss_product;

```

## 3. 插值

根据有限数据，希望得到一个连续的函数（曲线）；或者更密集的离散方程与已知数据互相吻合，这个过程叫做拟合，插值是曲线必须通过已知点的拟合。

插值的方法：

- 最邻近插值
- 线性插值
- 双线性插值
- 三次插值
- 三次样条线插值

## 4. 矩阵

特殊矩阵

- 对角矩阵
- 上（下） 三角矩阵

- 对阵矩阵
- 反对称矩阵
- 行列式
- 伴随矩阵 按行求得代数余子式按列放置构成伴随矩阵

# 5. 特征值与特征向量

定义： 假设我们有一个n阶的矩阵以及一个实数 r 使得我们可以找到一个非零向量

满足： Ax= r x ， 我们就称 r 是矩阵A的特征值，非零向量x是矩阵A的特征向量

几何意义：

矩阵和向量作乘法，向量会变成另一个方向或者长度的新向量，主要会发生旋转，伸缩的变化，如果矩阵乘以某一些向量后，向量不发生旋转，

只产生，伸缩变换，那么就说这些向量是矩阵的特征向量，伸缩比例就是特征值

```
#include <iostrema>
#include <Eigen/Dense>

using namespace Eigen;
using namespace std;

int main() {
    //定义一个3x3的对称矩阵
    Matrix3d A;
    A<<3,-1,2,-1,2,0,2,0,1;
    cout << "Matrix3d A=\n "<<A<<endl;
    //求矩阵A的特征向量和特征值
    SelfAdjointEigenSolver<Matrix3d> eigensolver(A);

    if(eigensolver.info()!=Success) abort();

    cout<<"特征值为"<<eigensolver.eigenvalues()<<endl;
    cout<<"特征向量为"<<eigensolver.eigenvectors()<<endl;

    return 0;
}

```

## 特征值分解：

特征值分解 就是讲一个矩阵分解成： A= P^P -1

P 是这个矩阵A的特征向量组成的矩阵

^ 是特征是组成的对角矩阵，里面的特征值是由大到小排列

这些特征值对应的特征向量就是描述这个矩阵的变化方向

A 为n*n的矩阵

### 奇异值分解：

特征值分解，A是方阵，若A不是方阵，需要使用奇异值分解SVD

SVD也是对矩阵进行分解，和特征分解不同，SVD并不要求分解的矩阵为方阵

## 6. 概率

## 7. 矩

## 8. 最小二乘法

要解决的问题：

虽然没有确定解，但是我们能不能求出近似解，使得模型能够在各个观测点上达到“最佳”拟合，“最佳”的准确可以是

所有观测点到直线的聚类和最小，也可以是所有观测点到直线的误差（真实值-理论值——绝对值和最小等

求具体参数：

- 导数法
- 几何法
- 梯度下降法
- 线性回归

观测值-预测值 的平方 为目标函数，求目标函数最小 得到的参数就是结果

## 9. 坐标系的变化

旋转矩阵R是一个正交矩阵，且 R的行列式为1

## 10. 二阶导数与海森矩阵

## 11. 贝叶斯定理

- 先验概率： 在不知道B事件的前提下，我们对A事件概率的一个主观判断
- 可能性 函数：
- 后验概率 ：

如果我能掌握一个事情的全部信息，我当然能计算出一个客观概率，可是很多情况决策面临的信息都是不全的，我们手中只有有限的信息

，既然无法得到全面的信息，我们就在信息有限的情况下，尽可能做出一个好的预测，也就是，在主观判断的基础上，你可以先估计一个值

（先验概率），然后根据观测的新信息不断修正（可能性函数)

## 12. 回环检测

回环检测，又称闭环检测，是指机器人识别出曾到达的常见，使得地图闭环的能力，在SLAM建图过程中，位姿的估计仅考虑相邻时间

上的关键帧，这期间产生的误差会逐步累积，形成累计误差，这样长期估计的结果将不可靠，所以通过回环检测的方法，发现潜在的回环，

用它修正误差可构建全局一致的轨迹和地图

检测方法：

- 词袋模型　常用于视觉SLAM　ORB-SLAM VINS-Mono
  1. 构建字典： 描述子聚类过程，用K近邻算法，或者使用已经探索过的环境中的特征在线动态生成词袋模型
  2. 建立字典树： 因为字典过大，使用k叉树 的方式来表达字典以建立字典树
  3. 计算词袋向量，关键帧和差项帧的相似度是通过词袋向量之间的距离来衡量的
  4. 相似度计算： 一些词用来识别两个图像是否显示同一个地方，比其他词更有用，而有一些词对识别贡献不大，为了区分

     这些词的重要性，可为每个词分配特定泉州，常见的方案是TF-IDF，它综合了图像中词的重要性，和收集过程

中词的重要性，可以用于评估一个词对于一个文件或者一个语料库中一个领域文件集的重复程度

5. 回环验证

- 基于深度学习

## 13. 关键帧-前端

作用：

- 降低局部相邻关键帧中的信息冗余度
- 防止无效和错误信息进入优化过程
- 提高计算资源的效率

TPCRKJ

## 14. 点云配准

点云配准分为两步： 粗配准 精配准

- 粗匹配：在点云相对位姿完全未知的情况下对点云进行配准，找到一个可以让两块点云近似的旋转平移变换矩阵，进而将配准点云数据转换

  到同一坐标系内，可以为精配准提高良好的初始值。
- 精匹配： 需要初始位姿，精配准是在粗配准的基础上，让点云之间的空间位置差异最小化，得到一个更加精确的旋转平移变换矩阵。

## 15. 相机畸变

- 枕形畸变
- 桶形畸变
- 切向畸变： 安装畸变

## 16. kmeans

kmeans 是一种非监督的聚类方法，是最常用的聚类技术之一，kmeans 尝试找到数据的自然类别，通过用户设定的类别个数k

它可以快速的找到 ”好的“类别 中心， "好的" 意味着聚类中心位于数据的自然中心

算法步骤：

1. 输入有样本数量集合和用户指定的类别数K
2. 分配类别初始化中心点的位置（随机或指定）
3. 将每个样本点放入离它最近的聚类中心所在的集合
4. 移动聚类中心到它所在集合的中心
5. 转到第3步，知道满足给定收敛条件

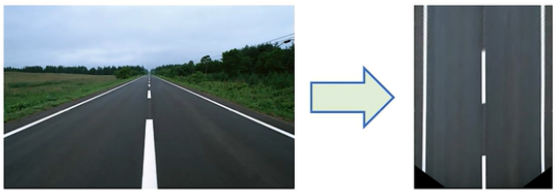
## 17. 齐次坐标变换

用N+1 维矢量 表示N维位置矢量的方法称为齐次坐标表示法

在三维直角坐标系中，一个点表示为[Px,py,pz]T ， 它的齐次坐标为[wPx,wPy,wPz,w]T.

齐次变换矩阵是4x4矩阵，完整形式由4个子矩阵组成：

$$T=\begin{bmatrix} R_{3x3} & P_{3x1} \\ f_{1x3} & \omega_{1x1} \end{bmatrix}=\begin{bmatrix} \text{旋转变换} & \text{位置矢量} \\ \text{透视变换} & \text{比例变换} \end{bmatrix}$$



平移 + 旋转

机器人研究中的齐次变换矩阵：　平移 + 旋转

$$T=\begin{bmatrix} R_{3x3} & P_{3x1} \\ 0 & 1 \end{bmatrix}=\begin{bmatrix} \text{旋转变换} & \text{位置矢量} \\ 0 & 1 \end{bmatrix}$$

- 位置矢量　平移

## 2. 平移齐次坐标变换

{B}分别沿{A}的X、Y、Z坐标轴平移a、b、c距离的平移齐次变换矩阵写为：

$$\underset{4\times 4}{Trans(a,b,c)}=\begin{bmatrix}1&0&0&a\\0&1&0&b\\0&0&1&c\\0&0&0&1\end{bmatrix}$$

R

用非零常数乘以变换矩阵的每个元素，不改变特性。

**例**2-3:求矢量2i+3j+2k被矢量4i-3j+7k平移得到的新矢量.

$$\begin{bmatrix}1&0&0&4\\0&1&0&-3\\0&0&1&7\\0&0&0&1\end{bmatrix}\begin{bmatrix}2\\3\\2\\1\end{bmatrix}=\begin{bmatrix}6\\0\\9\\1\end{bmatrix}$$

矢量和矩阵相乘，矢量放在右边

## 3. 旋转齐次坐标变换

$$\mathbf{R}(x,\theta)=\begin{bmatrix}1&0&0\\0&c\theta&-s\theta\\0&s\theta&c\theta\end{bmatrix}\quad \mathbf{R}(y,\theta)=\begin{bmatrix}c\theta&0&s\theta\\0&1&0\\-s\theta&0&c\theta\end{bmatrix}\quad \mathbf{R}(z,\theta)=\begin{bmatrix}c\theta&-s\theta&0\\s\theta&c\theta&0\\0&0&1\end{bmatrix}$$

将上式变换为齐次式：（以下公式要记住！）

$$\mathbf{R}(x,\theta)=\begin{bmatrix}1&0&0&0\\0&c\theta&-s\theta&0\\0&s\theta&c\theta&0\\0&0&0&1\end{bmatrix}\quad \mathbf{R}(y,\theta)=\begin{bmatrix}c\theta&0&s\theta&0\\0&1&0&0\\-s\theta&0&c\theta&0\\0&0&0&1\end{bmatrix}\quad \mathbf{R}(z,\theta)=\begin{bmatrix}c\theta&-s\theta&0&0\\s\theta&c\theta&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

- 例子:

引入齐次变换后，连续的变换可以变成矩阵的连乘形式。计算简化。

例2-4 ：U=7i+3j+2k，绕Z轴转90度后，再绕Y轴转90度，再平移（4，-3，7）。

$$R(z,90)=\begin{bmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$Trans(4,-3,7)=\begin{bmatrix}1 & 0 & 0 & 4\\ 0 & 1 & 0 & -3\\ 0 & 0 & 1 & 7\\ 0 & 0 & 0 & 1\end{bmatrix}$$

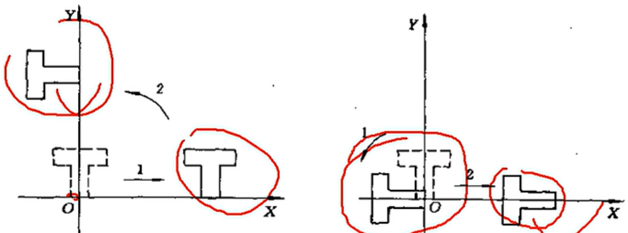
$$R(y,90)=\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ -1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$${}^A_BT=Trans(4,-3,7)Rot(y,90)Rot(z,90)=\begin{bmatrix}0 & 0 & 1 & 4\\ 1 & 0 & 0 & -3\\ 0 & 1 & 0 & 7\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$${}^A_B\mathrm{T}\begin{bmatrix}7\\ 3\\ 2\\ 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 1 & 4\\ 1 & 0 & 0 & -3\\ 0 & 1 & 0 & 7\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}7\\ 3\\ 2\\ 1\end{bmatrix}$$

$${}^A\mathbf{P}={}^A_B\mathbf{R}\cdot{}^B\mathbf{P}+{}^A\mathbf{P}_{B0}$$

由矩阵乘法没有交换性，可知变换次序对结果影响很大。




(a) 先平移后旋转　　(b) 先旋转后平移

- 矩阵乘法没有交换性，可知变换次序对结果影响很大