# SLAM

# SLAM

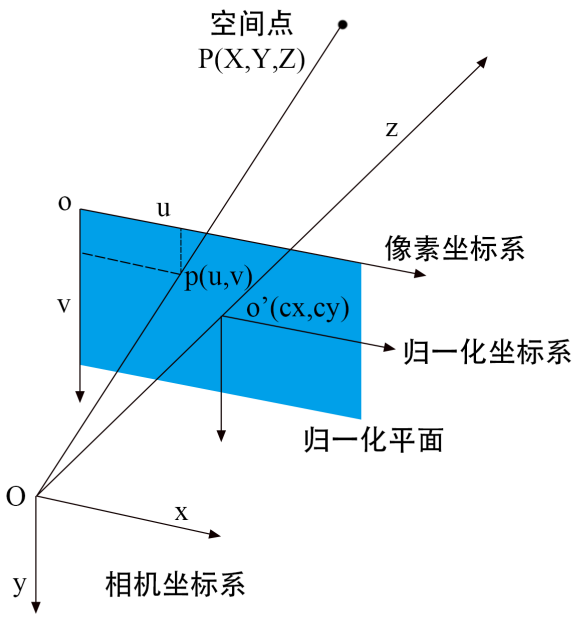
概述：https://www.it610.com/article/1548921238750040064.htm

# 1. 传感器

## 1.1. 相机

### 1.1.1. 相机模型

#### 1.1.1.1. 针孔相机模型




$$s\boldsymbol{u} = \boldsymbol{K}\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix}\begin{bmatrix}\boldsymbol{P}\\ 1\end{bmatrix} \tag{1}$$

$$\boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},\ \boldsymbol{u} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix},\ \boldsymbol{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{\text{world}} \tag{2}$$

畸变模型**

#### 1.1.2. 相机分类

https://zhuanlan.zhihu.com/p/371410573

### 1.2. 激光雷达

#### 1.2.1. 机械旋转式激光雷达

#### 1.2.2. 固态激光雷达

### 1.3. IMU

## 2. 几何变换与位姿轨迹对齐

### 2.1. 旋转的几种表示方式

| 表示方法 | Eigen类 | 初始化 |
| --- | --- | --- |
| 欧拉角 | | |
| 旋转矩阵（3×3） | Eigen::Matrix3d | |
| 旋转向量（3×1） | Eigen::AngleAxisd | |
| 四元数（4×1） | Eigen::Quaterniond | |

pitch、yaw、roll

### 2.2. 几何变换

几何变换位于Eigen的Geometry模块

| 变换名称 | 矩阵形式 | 自由度 | Eigen类（以double数据类型为例） |
| --- | --- | --- | --- |
| 旋转变换（SO3） | $\boldsymbol{R}$ | | |
| 欧式变换（SE3）、刚体变换 | $\boldsymbol{T}=\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ \boldsymbol{0} & 1\end{bmatrix}$ | | |
| 相似变换（Sim3） | $\boldsymbol{T}=\begin{bmatrix}c\boldsymbol{R} & \boldsymbol{t}\\ \boldsymbol{0} & 1\end{bmatrix}$ | | |
| 仿射变换（线性变换+平移） | $\boldsymbol{T}=\begin{bmatrix}\boldsymbol{A} & \boldsymbol{t}\\ \boldsymbol{0} & 1\end{bmatrix}$ | | Eigen::Affine3d |
| 射影变换 | | | Eigen::Projective3d |

## 2.3. Umeyama算法求解sim3变换

### 2.3.1. 理论推导

Umeyama算法、ICP算法、Sim3：https://zhuanlan.zhihu.com/p/532444005，

https://zhuanlan.zhihu.com/p/274504142，

https://blog.csdn.net/weixin_42823098/article/details/111308627?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-111308627-blog-108800676.pc_relevant_layerdownloadsortv1&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-111308627-blog-108800676.pc_relevant_layerdownloadsortv1&utm_relevant_index=5

参考文献：* "Least-squares estimation of transformation parameters between two point patterns",

* Shinji Umeyama, PAMI 1991, DOI: 10.1109/34.88573

$$e^2(\boldsymbol{R},\boldsymbol{t},c)=\frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{y}_i-(c\boldsymbol{R}\boldsymbol{x}_i+\boldsymbol{t})\right\|^2 \tag{3}$$

### 2.3.2. Umeyama算法实现流程

**(1) 分别计算两组点的质心$(\bar{\boldsymbol{x}},\ \bar{\boldsymbol{y}})$、去质心坐标$(\boldsymbol{x}',\ \boldsymbol{y}')$**

$$\bar{\boldsymbol{x}}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i,\ \bar{\boldsymbol{y}}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y}_i \tag{4}$$

$$\boldsymbol{x}'=\boldsymbol{x}_i-\bar{\boldsymbol{x}},\ \boldsymbol{y}'=\boldsymbol{y}_i-\bar{\boldsymbol{y}} \tag{5}$$

**(2) 分别计算两组点的方差系数$(\sigma_x,\ \sigma_y)$、协方差矩阵$\boldsymbol{H}$**

$$\sigma_x=\frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{x}_i'\right\|^2,\ \sigma_y=\frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{y}_i'\right\|^2 \tag{6}$$

$$\boldsymbol{H}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}'_i\boldsymbol{y}'^{\mathrm{T}}_i \tag{7}$$

**(3) 对$H$进行SVD分解，求解最优旋转矩阵$\boldsymbol{R}^*$**

$$\boldsymbol{H}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} \tag{8}$$

$$\boldsymbol{M}=\begin{bmatrix}1&0&0\\0&1&0\\0&0&\det(\mathbf{V}\mathbf{U}^{\mathrm{T}})\end{bmatrix} \tag{9}$$

$$\boldsymbol{R}^*=\boldsymbol{V}\boldsymbol{M}\boldsymbol{U}^{\mathrm{T}} \tag{10}$$

**(4) 利用最优解$\boldsymbol{R}^*$求解尺度参数$c$**

$$c^*=\frac{\boldsymbol{D}^*}{\sigma_x}=\frac{\mathrm{tr}(\boldsymbol{M}\boldsymbol{\Sigma})}{\sigma_x} \tag{11}$$

**(6) 利用最优解计算$t$**

$$\boldsymbol{t}^* = \bar{\boldsymbol{y}} - s^* \boldsymbol{R}^* \bar{\boldsymbol{x}} \tag{12}$$

**Umeyama算法位于Eigen源码Geometry模块的Umeyama.h头文件中**

# 3. 多视图几何

## 3.1. 对极约束

## 3.2. 三角化

## 3.3. PnP

### 3.3.1. 迭代非线性优化的PnP

$$\boldsymbol{T}^* = \arg\min_{\boldsymbol{T}} \|\boldsymbol{e}_i\|^2 = \arg\min_{\boldsymbol{T}} \sum_{i=1}^{n} \left\| \boldsymbol{u}_i - \frac{1}{s_i} \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \\ 1 \end{bmatrix} \right\|^2 \tag{13}$$

### 3.3.2. EPnP

#### 3.3.2.1. 理论推导

使用世界坐标系中四个虚拟控制点的线性组合来表示全体$n$个地图点坐标 $\boldsymbol{p}_i^w (i = 1, 2, \cdots, n)$，根据相机外参得到相机坐标系下地图点坐标$\boldsymbol{p}_i^c$和控制点坐标$\boldsymbol{c}_j^c$之间的关系

$$\boldsymbol{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \boldsymbol{c}_j^w, \ \boldsymbol{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \boldsymbol{c}_j^c \tag{14}$$

式中：$\sum_{j=1}^{4} \alpha_{ij} = 1$，$\alpha_{ij}$称为齐次重心坐标（homogeneous barycentric coordinates）。

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \tag{15}$$

消去$s_i$得到如下等式

$$\begin{cases} \sum_{j=1}^{4} \alpha_{ij} f_x x_j^c + \alpha_{ij} (c_x - u_i) z_j^c = 0 \\ \sum_{j=1}^{4} \alpha_{ij} f_y y_j^c + \alpha_{ij} (c_y - v_i) z_j^c = 0 \end{cases} \tag{16}$$

若有$n$个点，则构成线性方程组

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{0} \tag{17}$$

式中：$\boldsymbol{M}_{2n \times 12}$，$\boldsymbol{x} = \begin{bmatrix} (\boldsymbol{c}_1^c)^{\mathrm{T}} & (\boldsymbol{c}_2^c)^{\mathrm{T}} & (\boldsymbol{c}_3^c)^{\mathrm{T}} & (\boldsymbol{c}_4^c)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$，上述方程的解是属于系数矩阵$\boldsymbol{M}$的零空间，可以由线性组合的形式表示

$$\boldsymbol{x} = \sum_{i=1}^{N} \beta_i \boldsymbol{v}_i \tag{18}$$

式中：$\boldsymbol{v}_i(i=1,2,\cdots,N)$为矩阵$\boldsymbol{M}$的$N$个零奇异值对应的右奇异向量，
$\beta_i(i=1,2,\cdots,N)$为系数项，$N$可以是1~4之间的任意值，对此原文作者分别详细给出了$N=1$、$2$、$3$、$4$时$\beta_i$的具体求解方法。

**注意：向量v的维数是多少？？？ci怎么用vi线性表示？？？怎么解释原文公式 （16）？？？**

通过最小化两个坐标系下控制点间距差来计算系数项 $\boldsymbol{\beta}=[\beta_1 \quad \beta_2 \quad \beta_3 \quad \beta_4]^{\mathrm{T}}$

$$\mathrm{Error}(\boldsymbol{\beta})=\sum_{(i,j)\ s.t.\ i<j}\left(\left\|\boldsymbol{c}_i^c-\boldsymbol{c}_j^c\right\|^2-\left\|\boldsymbol{c}_i^w-\boldsymbol{c}_j^w\right\|^2\right) \tag{19}$$

求解$\boldsymbol{R}$、$\boldsymbol{t}$

#### 3.3.2.2. 实现流程

EPnP：以OpenCV为例

**参考文献：** [EPnP An Accurate O(n) Solution to the PnP Problem.pdf]

https://zhuanlan.zhihu.com/p/361791835

## 3.4. ICP(属于多视图几何吗？)

# 4. 特征提取与匹配

## 4.1. SIFT

### 4.1.1. 尺度空间极值检测

级联过滤方法检测关键点。通过使用被称为尺度空间的连续尺度函数，在所有可能的尺度上搜索稳定特征，实现对图像尺度变化不变的位置的检测。

尺度空间$L(x,y,\sigma)$定义为变尺度高斯函数$G(x,y)$与图像$I(x,y)$的卷积

$$L(x,y,\sigma)=G(x,y,\sigma)\cdot I(x,y) \tag{20}$$

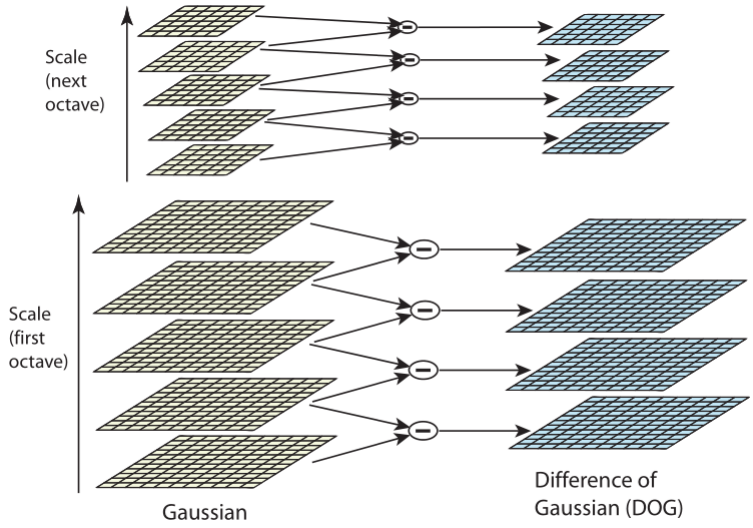
$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2} \tag{21}$$

使用高斯差分函数（Difference of Gaussian, DOG）实现有效检测尺度空间中稳定的关键点位置，该函数是由常数乘因子$k$分隔的两个相邻尺度的差分

$$D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y,\sigma))\cdot I(x,y)=L(x,y,k\sigma)-L(x,y,\sigma) \tag{22}$$

**初始图像与高斯函数进行增量卷积，生成下图左列所示的被常数因子k分隔的尺度空间图像集。我们选择将尺度空间的每个八度(即σ的两倍)分成一个整数s，即k = 21/s**

**对于尺度空间的每个八度，将初始图像与高斯函数进行反复卷积，生成左图所示的尺度空间图像集。对相邻的高斯图像进行相减，得到右边的高斯图像差。在每一个八度之后，高斯图像被降低采样2倍，并重复这个过程**




#### 4.1.2. 关键点定位

#### 4.1.3. 方向分配

#### 4.1.4. 关键点描述子

https://blog.csdn.net/lavender19/article/details/120396145?spm=1001.2101.3001.6650.6&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-6-120396145-blog-118003686.pc_relevant_3mothn_strategy_recovery&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-6-120396145-blog-118003686.pc_relevant_3mothn_strategy_recovery&utm_relevant_index=11

https://blog.csdn.net/Robert_Q/article/details/118003686

### 4.2. ORB

### 4.3. OpenCV特征提取与匹配

cv::keypoint类的成员变量包括哪些：位置、方向、角度、金字塔层数？？

## 5. 线性方程组与非线性优化

### 5.1. 线性方程组

线性方程组的几种分解方法：svd，QR、LU、

### 5.2. 非线性优化问题的数学模型

### 5.3. Ceres非线性优化

```
ceres::Problem
ceres::Solver::Options
ceres::Solver::Summary
ceres::CostFunction

// 求解器类ceres::Solver中的Solve函数来求解非线性优化问题，引用的三个参数中包含三个重要的类ceres::Problem 、ceres::Solver::Options、ceres::Solver::Summary
CERES_EXPORT void Solve(const Solver::Options& options,
                        Problem* problem,
                        Solver::Summary* summary);
```

### 5.3.1. 构造代价函数

### 5.3.2. 构建优化问题

### 5.3.3. 配置并运行求解器

## 5.4. g2o非线性优化