

【边缘化及滑动窗口算法】

1. SLAM问题的概率建模

状态 ξ 观测 r

$$P(\xi | r) = \frac{P(r | \xi) P(\xi)}{P(r)} \rightarrow \text{先验 priori, GNSS ...}$$

$P(r | \xi)$ → 与状态多元关
系
后验 posteriori

Objective: (解释: 在取某一状态时, 能使该观测下此状态出现的概率最大)

$$\begin{aligned} \underset{\xi}{\operatorname{argmax}} P(\xi | r) &= \underset{\xi}{\operatorname{argmax}} P(r | \xi) P(\xi) \\ &= \underset{\xi}{\operatorname{argmax}} \exp \left(-\frac{1}{2} (r - \mu_r)^T \Sigma_r^{-1} (r - \mu_r) + \left(-\frac{1}{2} (\xi - \mu_\xi)^T \Sigma_\xi^{-1} (\xi - \mu_\xi) \right) \right) \\ &= -\frac{1}{2} \underset{\xi}{\operatorname{argmin}} \left(\|r - \mu_r\|^2_{\Sigma_r} + \|\xi - \mu_\xi\|^2_{\Sigma_\xi} \right) \end{aligned}$$

多维高斯分布:

$$x \sim N(\mu, \Sigma)$$

$$p(x) = \frac{1}{(2\pi)^n \det \Sigma} \exp \left[-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right]$$

在有多个独立观测时:

$$\begin{aligned} \text{Objective: } \underset{\xi}{\operatorname{argmax}} P(r_1, r_2, r_3, \dots, r_n | \xi) P(\xi) &\quad \downarrow \text{最小二乘} \\ \underset{\xi}{\operatorname{argmin}} \left(\sum_{i=1}^n \|r_i - \mu_{r_i}\|^2_{\Sigma_{r_i}} + \|\xi - \mu_\xi\|^2_{\Sigma_\xi} \right) &\quad H \xi = b \end{aligned}$$

2. 钩子补在状态估计中的应用

[Schur 原理]

$$\text{对于矩阵 } M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

若 A 或 D 是逆矩阵, 可由如下变换为上三角矩阵:

$$\begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & D_A \end{bmatrix}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} I & 0 \\ D^{-1}C & I \end{bmatrix} = \begin{bmatrix} D_B & B \\ 0 & D \end{bmatrix}$$

Δ_A 和 Δ_D 就是钩子补:

$$\Delta_A = D - CA^{-1}B$$

$$\Delta_D = A - BD^{-1}C$$

用钩子补解决SLAM问题:

$$\begin{bmatrix} H_{ii} & H_{ij} \\ H_{ji} & H_{jj} \end{bmatrix} \begin{bmatrix} \delta \xi_i \\ \delta \xi_j \end{bmatrix} = \begin{bmatrix} b_i \\ b_j \end{bmatrix}$$

$$\begin{bmatrix} I & 0 \\ H_{ji} H_{ii}^{-1} & I \end{bmatrix} \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ji} & H_{jj} \end{bmatrix} \begin{bmatrix} \delta \xi_i \\ \delta \xi_j \end{bmatrix} = \begin{bmatrix} b_i \\ b_j \end{bmatrix}$$

$$\begin{bmatrix} H_{ii} & H_{ij} \\ 0 & -H_{ji} H_{ii}^{-1} H_{ij} + H_{jj} \end{bmatrix} \begin{bmatrix} \delta \xi_i \\ \delta \xi_j \end{bmatrix} = \begin{bmatrix} b_i \\ b_j \end{bmatrix}$$

$$(-H_{ji} H_{ii}^{-1} H_{ij} + H_{jj}) \delta \xi_j = b_j$$

就此言, 就能从问题中剥离。

写得真好 ↓

<https://xiaotaoguo.com/p/slam-marginalization>

第10周：ORB_SLAM2 课程课件

本周课程重点：

全网最详细的SIM3 算法原理详解

什么是SIM3变换？

目的

热身：3对点计算旋转可以吗？

计算SIM3平移

计算SIM3尺度因子

计算旋转

迭代次数的估计

四元数到旋转向量的转换

Sim3的逆变换矩阵

参考

第10周：ORB_SLAM2 课程课件

本课件是公众号 计算机视觉life 旗下课程 [《全网最详细的ORB-SLAM2精讲：原理推导+逐行代码分析》](#)

(点击可跳转课程详情) 的课程课件。谢谢各位学员的支持！

本课程对应的注释代码：https://github.com/electech6/ORBSLAM2_detailed_comments

由于源码注释和课件在持续更新，所以：

如视频课程中注释与上述GitHub中有不同，**以GitHub上最新源码为准。**

如视频课程中课件与本课件不同，**以本课件为准。**

本周课程重点：

1. 理解Sim3的原理（重要），能看懂推导过程。
2. 掌握Sim3代码（重要）。

全网最详细的SIM3 算法原理详解

什么是SIM3变换？

Sim3 (**Similarity Transformation**)的提出就是为了解决两个坐标系之间的相似变换问题，只要我们能得到**3对**匹配好的不共线的三维点在两个坐标系下的坐标，我们就能解出Sim3相似变换。这个也是Sim3中数字3的来源。计算Sim3 实际就是计算这三个参数：旋转 R 、平移 t 、尺度因子 s 。

为什么三对不共线点就可以求解？

我们来感性的理解一下，我们有三对匹配的不共线三维点可以构成两个三角形。我们根据三角形各自的法向量可以得到他们之间的旋转，通过相似三角形面积能够得到尺度，用前面得到的旋转和尺度可以把两个三角形平行放置，通过计算距离可以得到平移。

以上是直观感性的理解，实际在计算的时候需要有严格的数学推导。我们这里使用的方法是来自 Berthold K. P. Horn 在 1987 年发表的论文 "Closed-form solution of absolute orientation using unit quaternions"。该文提出了用三维匹配点构建优化方程，不需要迭代，直接用闭式解求出了两个坐标系之间的旋转、平移、尺度。该方法的优点非常明显：

- 1、给定两个坐标系下的至少 3 个匹配三维点，只需一步即可求得变换关系，不需要迭代，速度很快。
- 2、因为不是数值解，不需要像迭代方法那样需要找一个好的初始解。闭式解可以直接求得比较精确的结果。

数值解(numerical solution)是在特定条件下通过近似计算得出来的一个数值，比如数值逼近。

闭式解也称为解析解，就是给出解的具体函数形式，从解的表达式中就可以算出任何对应值。

实际上，在 SLAM 问题中，我们通常能够得到大于 3 个的三维匹配点，该论文推导了该情况下最小二乘得到最优解的方法。

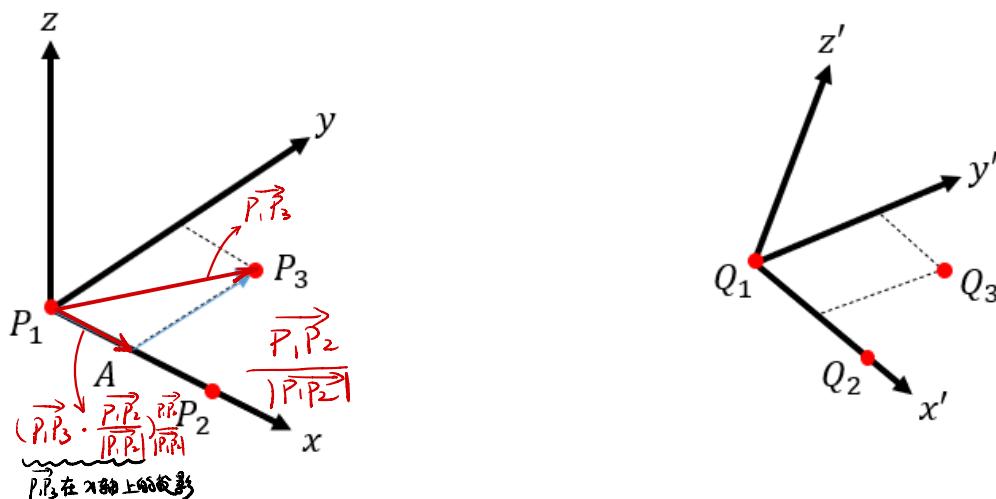
另外，论文中利用单位四元数表示旋转，简化了求解的推导。

目的

已知至少三个匹配的不共线三维点对，求他们之间的相对旋转、平移、尺度因子。

热身：3 对点计算旋转可以吗？

假设坐标系 1 下有三个不共线三维点 P_1, P_2, P_3 ，他们分别和坐标系 2 下的三个不共线三维点 Q_1, Q_2, Q_3 —— 匹配。



首先，我们根据坐标系 1 下的三个不共线三维点来构造一个新的坐标系。

沿着 x 轴上的单位向量 \hat{x}

$$\begin{aligned} x &= P_2 - P_1 \\ \hat{x} &= \frac{x}{\|x\|} \end{aligned} \tag{1}$$

沿着 y 轴的单位向量 \hat{y}

$$\begin{aligned}
y &= \overrightarrow{AP_3} \\
&= \overrightarrow{P_1P_3} - \overrightarrow{P_1A} \\
&= (P_3 - P_1) - [(P_3 - P_1)\hat{x}]\hat{x} \\
\hat{y} &= \frac{y}{\|y\|}
\end{aligned} \tag{2}$$

沿着 z 轴的单位向量 \hat{z}

$$\hat{z} = \hat{x} \times \hat{y} \tag{3}$$

同理，我们对于坐标系2下的 Q_1, Q_2, Q_3 也可以得到沿着3个坐标轴的单位向量 $\hat{x}', \hat{y}', \hat{z}'$

我们现在要计算坐标系1到坐标系2的旋转，记坐标系单位向量构成的基底矩阵为

$$\begin{aligned}
M_1 &= [\hat{x}, \hat{y}, \hat{z}] \\
M_2 &= [\hat{x}', \hat{y}', \hat{z}']
\end{aligned} \tag{4}$$

假设坐标系1下有一个向量 v_1 ，它在坐标系2下记为 v_2 ，因为向量本身没有变化，根据坐标系定义有

$$\begin{aligned}
M_1 v_1 &= M_2 v_2 \\
v_2 &= M_2^T M_1 v_1
\end{aligned} \tag{5}$$

那么从坐标系1到坐标系2的旋转就是

$$R = M_2^T M_1 \tag{6}$$

看起来好像没什么问题，但是实际上我们不会这样使用，因为存在如下问题：

- 1、这个旋转的结果和选择点的顺序关系密切，我们分别让不同的点做坐标系原点，得到的结果不同。
- 2、这种情况不适用于匹配点大于3个的情况。

因此实际上我们不会使用以上方法。我们通常能够拿到远大于3个的三维匹配点对，我们会使用最小二乘法来得到更稳定、更精确的结果。

下面进入正文。

计算SIM3平移

假设我们得到了 $n > 3$ 组匹配的三维点，分别记为 $\{P_i\}, \{Q_i\}$ ，其中 $i = 1, \dots, n$ 我们的目的是对于每对匹配点，找到如下的变换关系：

$$Q_i = sRP_i + t \tag{7}$$

其中 s 是尺度因子， R 是旋转， t 是平移。

如果数据是没有任何噪音的理想数据，理论上我们可以找到满足上述关系的尺度因子、旋转和平移。但实际上数据是不可避免会有噪音和误差，所以我们转换思路，定义一个误差 e_i ，我们的目的就是寻找合适的尺度因子、旋转和平移，使得它在所有数据上的误差最小。

$$\begin{aligned}
e_i &= Q_i - sRP_i - t \\
\min_{s,R,t} \sum_{i=1}^n \|e_i\|^2 &= \min_{s,R,t} \sum_{i=1}^n \|Q_i - sRP_i - t\|^2
\end{aligned} \tag{8}$$

在开始求解之前，我们先定义两个三维点集合中所有三维点的均值（或者称为质心、重心）

$$\begin{aligned}\bar{P} &= \frac{1}{n} \sum_{i=1}^n P_i \\ \bar{Q} &= \frac{1}{n} \sum_{i=1}^n Q_i\end{aligned}\tag{9}$$

我们对每个三维点 P_i, Q_i 分别减去均值，得到去中心化后的坐标 P'_i, Q'_i ，则有

$$\begin{aligned}P'_i &= P_i - \bar{P} \\ Q'_i &= Q_i - \bar{Q} \\ \sum_{i=1}^n P'_i &= \sum_{i=1}^n (P_i - \bar{P}) = \sum_{i=1}^n P_i - n\bar{P} = 0 \\ \sum_{i=1}^n Q'_i &= \sum_{i=1}^n (Q_i - \bar{Q}) = \sum_{i=1}^n Q_i - n\bar{Q} = 0\end{aligned}\tag{10}$$

上面的结论很重要，我们在后面推导的时候要使用。

下面开始推导我们的误差方程：

g2] CP - 拼

$$\begin{aligned}\sum_{i=1}^n \|e_i\|^2 &= \sum_{i=1}^n \|Q_i - sRP_i - t\|^2 \\ &= \sum_{i=1}^n \|Q'_i + \bar{Q} - (sRP'_i + sR\bar{P}) - t\|^2 \\ &= \sum_{i=1}^n \|(Q'_i - sRP'_i) + \underbrace{(\bar{Q} - sR\bar{P}) - t}_{t_0}\|^2 \quad \text{已知} \\ &= \sum_{i=1}^n \|(Q'_i - sRP'_i)\|^2 + 2t_0 \sum_{i=1}^n (Q'_i - sRP'_i) + n\|t_0\|^2\end{aligned}\tag{11}$$

为了推导不显得那样臃肿，其中我们简记

$$t_0 = \bar{Q} - sR\bar{P} - t \tag{12}$$

根据前面的推导可得等式右边中间项

$$\sum_{i=1}^n (Q'_i - sRP'_i) = \sum_{i=1}^n Q'_i - sR \sum_{i=1}^n P'_i = 0 \tag{13}$$

这样我们前面的误差方程可以化简为：

$$\sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|(Q'_i - sRP'_i)\|^2 + n\|t_0\|^2 \tag{14}$$

等式右边的两项都是大于等于0的平方项，并且只有第二项里的 t_0 和我们要求的平移 t 有关，所以当 $t_0 = 0$ 时，我们可以得到平移的最优解 t^*

$$\begin{aligned}t_0 &= \bar{Q} - sR\bar{P} - t = 0 \\ t^* &= \bar{Q} - sR\bar{P}\end{aligned}\tag{15}$$

也就是说我们知道了旋转 R 和尺度 s 就能根据三维点均值做差得到平移 t 了。注意这里平移的方向是 $\{P_i\} \rightarrow \{Q_i\}$ 。

计算SIM3尺度因子

我们的误差函数也可以进一步简化为：

$$\begin{aligned} \sum_{i=1}^n \|e_i\|^2 &= \sum_{i=1}^n \|Q'_i - sRP'_i\|^2 \\ &= \sum_{i=1}^n \|Q'_i\|^2 - 2s \sum_{i=1}^n Q'_i RP'_i + s^2 \sum_{i=1}^n \|RP'_i\|^2 \end{aligned} \quad (16)$$

由于向量的模长不受旋转的影响，所以 $\|RP'_i\|^2 = \|P'_i\|^2$

为了后续更加清晰的表示，我们用简单的符号代替上述式子里的部分内容，所以有

$$\begin{aligned} \sum_{i=1}^n \|e_i\|^2 &= \underbrace{\sum_{i=1}^n \|Q'_i\|^2}_{S_Q} - 2s \underbrace{\sum_{i=1}^n Q'_i RP'_i}_{D} + s^2 \underbrace{\sum_{i=1}^n \|P'_i\|^2}_{S_P} \\ &= S_Q - 2sD + s^2 S_P \end{aligned} \quad (17)$$

$\frac{d(S_Q - 2sD + s^2 S_P)}{ds} = -2D + 2sS_P = 0$
 $s = \frac{D}{S_P} = \frac{D}{\frac{1}{s} \cdot \frac{1}{s} \|P'_i\|^2}$

由于 R 是已知的，我们很容易看出来上面是一个以 s 为自变量的一元二次方程，要使得该方程误差最小，我们可以得到此时尺度 s 的取值：

$$s = \frac{D}{S_P} = \frac{\sum_{i=1}^n Q'_i RP'_i}{\sum_{i=1}^n \|P'_i\|^2} \quad (18)$$

ORB-SLAM2和3里都是使用上述公式求尺度。注意这里尺度的方向是 $\{P_i\} \rightarrow \{Q_i\}$ 。

但是，到这里还存在一个问题，我们对 P, Q 做个调换后得到

$$\frac{\sum_{i=1}^n P'_i R^T Q'_i}{\sum_{i=1}^n \|Q'_i\|^2} \neq \frac{1}{s} \quad (19)$$

我们看到尺度并不具备对称性，也就是从 $\{P_i\} \rightarrow \{Q_i\}$ 得到的尺度并不等于从 $\{Q_i\} \rightarrow \{P_i\}$ 得到的尺度的倒数。这也说明我们前面方法得到的尺度并不稳定。所以需要重新构造误差函数，使得我们得到的尺度是对称的、稳定的。

当然我们不用自己绞尽脑汁去构造，直接搬运论文里大佬的构造方法即可：

$$\begin{aligned} \sum_{i=1}^n \|e_i\|^2 &= \sum_{i=1}^n \left\| \frac{1}{\sqrt{s}} Q'_i - \sqrt{s} RP'_i \right\|^2 \\ &= \frac{1}{s} \sum_{i=1}^n \|Q'_i\|^2 - 2 \underbrace{\sum_{i=1}^n Q'_i RP'_i}_{D} + s \sum_{i=1}^n \|RP'_i\|^2 \\ &= \frac{1}{s} S_Q - 2D + s S_P \end{aligned} \quad (20)$$

论文里是 $(\sqrt{s} S_P - \frac{1}{\sqrt{s}} S_Q)^2 + (2\sqrt{s} S_Q - D)$

$$= (\sqrt{s} S_P - \sqrt{\frac{S_Q}{s}})^2 + 2(\sqrt{S_P S_Q} - D)$$

$$= \sqrt{s} S_P - \sqrt{\frac{S_Q}{s}} + \frac{S_Q \sqrt{s}}{s} - 2D$$

论文中此处有误，但最后结果是对的
 但你给的也有误吗？

上面等式右边第一项只和尺度 s 有关的平方项，第二项和 s 无关，但和旋转 R 有关，因此令第一项为 0，我们就能得到最佳的尺度 s^*

$$s^* = \sqrt{\frac{S_Q}{S_P}} = \sqrt{\frac{\sum_{i=1}^n \|Q'_i\|^2}{\sum_{i=1}^n \|P'_i\|^2}} \quad (21)$$

同时，第二项里的 S_P, S_Q 都是平方项，所以令第二项里的 $D = \sum_{i=1}^n Q'_i R P'_i$ 最大，可以使得剩下的误差函数最小。

这里我们总结下对称形式的优势：

- 1、使得尺度的解和旋转、平移都无关
- 2、反过来，旋转的确定不受数据选择不同的影响

我们直观理解一下，尺度就是三维点到各自均值中心的距离之和。

计算旋转

高阶复数

下面我们考虑用四元数来代替矩阵来表达旋转。

为什么用四元数而不是矩阵表达旋转？

- 1、因为直接使用矩阵必须要保证矩阵的正交性等约束，这个约束太强了，会带来很多困难。
- 2、四元数只需要保证模值为1的约束，简单很多，方便推导。

开始之前，先来看下四元数的性质。大家可以自行证明。

性质1、用四元数来对三维点进行旋转

实部为0

假设空间三维点 $P = [x, y, z]$ ，用一个虚部四元数来表示为 $\dot{p} = [0, x, y, z]^T$ 。

旋转用一个单位四元数 \dot{q} 来表示，则 \dot{p} 旋转后的三维点用四元数表示为：

$$\dot{p}' = \dot{q}\dot{p}\dot{q}^{-1} = \dot{q}\dot{p}\dot{q}^* \quad (22)$$

四元数 \dot{p}' 的虚部取出即为旋转后的坐标。其中 \dot{q}^* 表示取 \dot{q} 的共轭。

性质2：

$x_1x_2 + y_1y_2$

三个四元数满足如下条件。直接相乘的形式，表示四元数乘法，中间的 · 表示向量点乘。

$$\dot{p} \cdot (\dot{r}\dot{q}^*) = (\dot{p}\dot{q}) \cdot \dot{r} \quad (23)$$

性质3：

假设四元数 $\dot{r} = [r_0, r_x, r_y, r_z]$ ，则有

$$\begin{aligned} \dot{r}\dot{q} &= \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & -r_z & r_y \\ r_y & r_z & r_0 & -r_x \\ r_z & -r_y & r_x & r_0 \end{bmatrix} \dot{q} = \mathbb{R}\dot{q} \\ \dot{q}\dot{r} &= \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{bmatrix} \dot{q} = \overline{\mathbb{R}}\dot{q} \end{aligned} \quad (24)$$

和 i 实部相乘，虚部交换，负号是复数 $= -1$ 得到的
右侧除 r_0 (i 的实部) 不变，其它全变号 (由于点乘改变了乘的顺序)

其中 $\mathbb{R}, \overline{\mathbb{R}}$ 都是 4×4 的对称矩阵。

左侧不变，因为与 i 的实部相乘。

下面进入正题。

利用前面的性质，我们现在的代价函数可以做如下变换：

$$\begin{aligned}
\sum_{i=1}^n Q'_i R P'_i &= \sum_{i=1}^n (\dot{Q}'_i) \cdot (\dot{q} \dot{P}'_i \dot{q}^*) \\
&= \sum_{i=1}^n (\dot{Q}'_i \dot{q}) \cdot (\dot{q} \dot{P}'_i) \\
&= \sum_{i=1}^n (\mathbb{R}_{Q,i} \dot{q}) \circ (\overline{\mathbb{R}_{P,i}} \dot{q}) \\
&= \sum_{i=1}^n \dot{q}^T \underline{\mathbb{R}_{Q,i}^T} \overline{\mathbb{R}_{P,i}} \dot{q} \\
&= \dot{q}^T \left(\sum_{i=1}^n \mathbb{R}_{Q,i}^T \overline{\mathbb{R}_{P,i}} \right) \dot{q} \\
&= \dot{q}^T N \dot{q}
\end{aligned} \tag{25}$$

其中：

$$\begin{aligned}
Q'_i &= [Q'_{i,x}, Q'_{i,y}, Q'_{i,z}]^T & (26) \\
P'_i &= [P'_{i,x}, P'_{i,y}, P'_{i,z}]^T \\
\dot{Q}'_i \dot{q} &= \begin{bmatrix} 0 & -Q'_{i,x} & -Q'_{i,y} & -Q'_{i,z} \\ Q'_{i,x} & 0 & -Q'_{i,z} & Q'_{i,y} \\ Q'_{i,y} & Q'_{i,z} & 0 & -Q'_{i,x} \\ Q'_{i,z} & -Q'_{i,y} & Q'_{i,x} & 0 \end{bmatrix} \dot{q} = \mathbb{R}_{Q,i} \dot{q} \\
\dot{q} \dot{P}'_i &= \begin{bmatrix} 0 & -P'_{i,x} & -P'_{i,y} & -P'_{i,z} \\ P'_{i,x} & 0 & P'_{i,z} & -P'_{i,y} \\ P'_{i,y} & -P'_{i,z} & 0 & P'_{i,x} \\ P'_{i,z} & P'_{i,y} & -P'_{i,x} & 0 \end{bmatrix} \dot{q} = \overline{\mathbb{R}_{P,i}} \dot{q} \\
\mathbb{R}_{Q,i}^T &= \begin{bmatrix} 0 & \mathbb{Q}'_{i,x} & \mathbb{Q}'_{i,y} & \mathbb{Q}'_{i,z} \\ -\mathbb{Q}'_{i,x} & 0 & \mathbb{Q}'_{i,z} & -\mathbb{Q}'_{i,y} \\ -\mathbb{Q}'_{i,y} & -\mathbb{Q}'_{i,z} & 0 & \mathbb{Q}'_{i,x} \\ -\mathbb{Q}'_{i,z} & \mathbb{Q}'_{i,y} & -\mathbb{Q}'_{i,x} & 0 \end{bmatrix}
\end{aligned}$$

我们定义

$$\begin{aligned}
M &= \sum_{i=1}^n P'_i Q'^T_i \\
&= \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{27}
\end{aligned}$$

其中

$$\begin{aligned}
S_{xx} &= \sum_{i=1}^n P_{i,x} Q_{i,x} \\
S_{xy} &= \sum_{i=1}^n P_{i,x} Q_{i,y}
\end{aligned} \tag{28}$$

引入 M 是为了方便用其元素来表示 N ，我们将上面的结果代入整理，则有：

$$\begin{aligned}
N &= \sum_{i=1}^n \mathbb{R}_{Q,i}^T \overline{\mathbb{R}_{P,i}} \\
&= \begin{bmatrix} (S_{xx} + S_{yy} + S_{zz}) & S_{yz} - S_{zy} & S_{zx} - S_{xz} & S_{xy} - S_{yx} \\ S_{yz} - S_{zy} & (S_{xx} - S_{yy} - S_{zz}) & S_{xy} + S_{yx} & S_{zx} + S_{xz} \\ S_{zx} - S_{xz} & S_{xy} + S_{yx} & (-S_{xx} + S_{yy} - S_{zz}) & S_{yz} + S_{zy} \\ S_{xy} - S_{yx} & S_{zx} + S_{xz} & S_{yz} + S_{zy} & (-S_{xx} - S_{yy} + S_{zz}) \end{bmatrix} \tag{29}
\end{aligned}$$

然后我们对 N 进行特征值分解，求得最大特征值对应的特征向量就是待求的用四元数表示的旋转，注意这里旋转的方向是 $\{P_i\} \rightarrow \{Q_i\}$ 。

论文中写反了。

至此，我们就得到Sim3 的三个参数：旋转 R 、平移 t 、尺度因子 s 。

我们总结一下计算 Sim3 的步骤。

1、先计算旋转 R 。

具体来说，先构建 M 矩阵

$$\begin{aligned} M &= \sum_{i=1}^n P'_i Q'^T_i \\ &= \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \end{aligned} \quad (30)$$

然后得到矩阵 N，对 N 进行特征值分解，求得最大特征值对应的特征向量就是待求的用四元数表示的旋转 R 。注意这里旋转的方向是 $\{P_i\} \rightarrow \{Q_i\}$ 。

2、根据上面计算的旋转 R 来计算尺度 s 。

具体来说，可以使用以下两种方法来计算，第一种是具有对称性的尺度（推荐）

$$s = \sqrt{\frac{S_Q}{S_P}} = \sqrt{\frac{\sum_{i=1}^n \|Q'_i\|^2}{\sum_{i=1}^n \|P'_i\|^2}} \quad (31)$$

第二种是不具有对称性的尺度（ORB-SLAM 使用）

$$s = \frac{D}{S_P} = \frac{\sum_{i=1}^n Q'_i R P'_i}{\sum_{i=1}^n \|P'_i\|^2} \quad (32)$$

3、根据旋转 R 和尺度 s 计算平移 t 。

$$t = \bar{Q} - sR\bar{P} \quad (33)$$

以上就是Sim3 推导过程。如有不理解强烈建议看原论文。

迭代次数的估计

ϵ 表示在 N 对匹配点中，随便抽取一对点是内点的概率。为了计算Sim3，我们需要从 N 对点里取三对点，假设是有放回的取，在一次采样中，同时取这三对点都为内点的概率是 ϵ^3 ，相反的，这三对点中至少存在一对外点的概率是 $1 - \epsilon^3$ 。假设RANSAC 连续进行了 K 次采样，每一次采样中三对点中至少存在一次外点的概率 $p_0 = (1 - \epsilon^3)^K$ ，那么， K 次采样中，至少有一次采样中三对点都是内点的概率 $p = 1 - p_0$

代入，得到

$$K = \frac{\log(1 - p)}{\log(1 - \epsilon^3)} \quad (34)$$

四元数到旋转向量的转换

假设四元数为 $q = [q_0, q_1, q_2, q_3]$, q_0 为实部, q_1, q_2, q_3 为虚部, 旋转向量 θn 的旋转轴 $n = [n_x, n_y, n_z]^T$, 旋转角度 θ , 旋转向量到四元数的转换公式:

$$q = [\cos \frac{\theta}{2}, n \sin \frac{\theta}{2}] \quad (35)$$

四元数到旋转向量的转换公式:

$$\begin{aligned} \theta &= 2 \arccos q_0 \\ [n_x, n_y, n_z]^T &= [q_1, q_2, q_3]^T / \sin \frac{\theta}{2} \end{aligned} \quad (36)$$

Sim3的逆变换矩阵

假设 Sim3 的变换矩阵为:

$$Sim3 = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix} \quad (37)$$

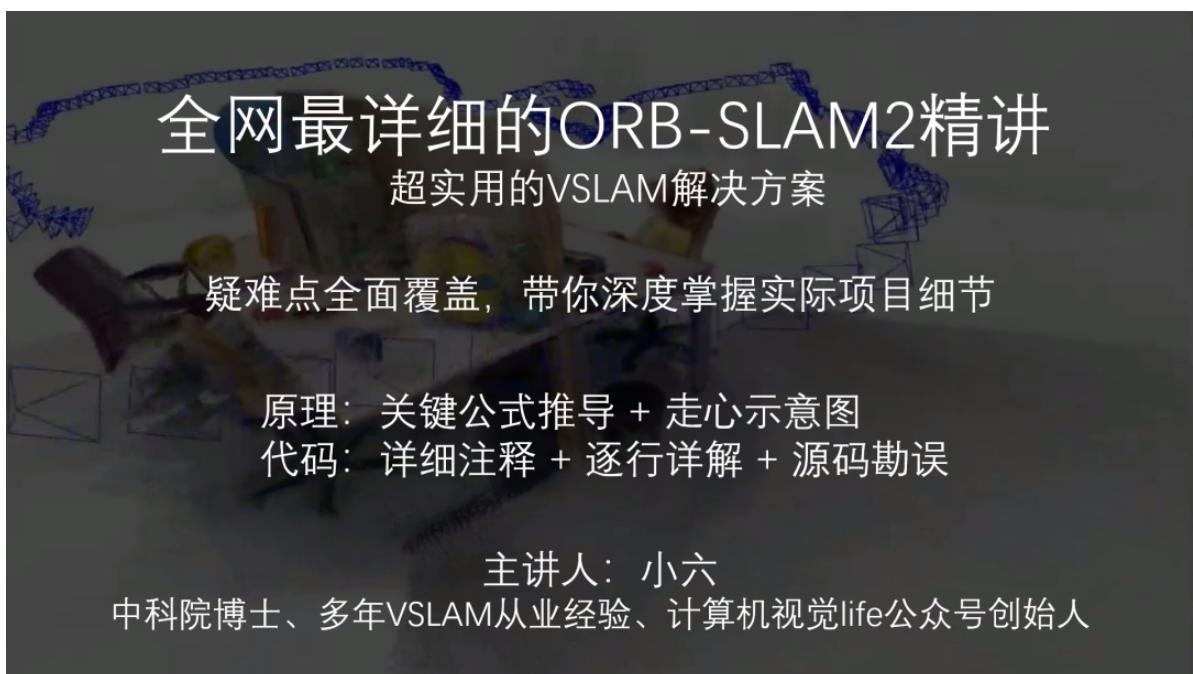
那么, 它的逆变换矩阵为:

$$(Sim3)^{-1} = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{s} R^T & -\frac{1}{s} R^T t \\ 0 & 1 \end{bmatrix} \quad (38)$$

参考

论文: *Closed-form solution of absolute orientation using unit quaternions*

[《全网最详细的ORB-SLAM2精讲：原理推导+逐行代码分析》](#) (点击可跳转课程详情)



长按或扫描二维码查看课程介绍和购买方式: