深蓝学院
shenlanxueyuan.com

第三章作业讲解

主讲人 王家浩

公式推导

●线特征

残差：

$$d_\varepsilon = |\boldsymbol{d}_\varepsilon| = \sqrt{\boldsymbol{d}_\varepsilon^T \boldsymbol{d}_\varepsilon}$$

其中

$$\boldsymbol{d}_\varepsilon = \frac{(\tilde{p}_i - p_b) \times (\tilde{p}_i - p_a)}{|p_a - p_b|}$$

$$\tilde{p}_i = \boldsymbol{R} p_i + \boldsymbol{t}$$

根据链式求导法则，残差对位移和姿态的雅可比写为

$$\frac{\partial d_\varepsilon}{\partial \boldsymbol{T}} = \frac{\partial d_\varepsilon}{\partial \boldsymbol{d}_\varepsilon} \frac{\partial \boldsymbol{d}_\varepsilon}{\partial \tilde{p}_i} \frac{\partial \tilde{p}_i}{\partial \boldsymbol{T}}$$

其中 $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t} \\ \delta\boldsymbol{\theta} \end{bmatrix}$，旋转采用SO3左扰动模型，即 $\boldsymbol{R}' = \exp(\boldsymbol{\theta})\boldsymbol{R}$

公式推导

●线特征

对雅可比中的3个因子依次求解：

$$\begin{aligned}\frac{\partial d_\varepsilon}{\partial \boldsymbol{d}_\varepsilon} &= \frac{\partial \sqrt{\boldsymbol{d}_\varepsilon^T \boldsymbol{d}_\varepsilon}}{\partial \boldsymbol{d}_\varepsilon} \\ &= \frac{1}{2}\frac{1}{\sqrt{\boldsymbol{d}_\varepsilon^T \boldsymbol{d}_\varepsilon}} 2\boldsymbol{d}_\varepsilon^T \\ &= \frac{\boldsymbol{d}_\varepsilon^T}{d_\varepsilon}\end{aligned}$$

$$\begin{aligned}\frac{\partial \boldsymbol{d}_\varepsilon}{\partial \tilde{p}_i} &= \frac{\partial \frac{(\tilde{p}_i - p_b)\times(\tilde{p}_i - p_a)}{|p_a - p_b|}}{\partial \tilde{p}_i} \\ &= \frac{1}{|p_a - p_b|}\frac{\partial\left((\tilde{p}_i - p_b)\times(\tilde{p}_i - p_a)\right)}{\partial \tilde{p}_i} \\ &= \frac{1}{|p_a - p_b|}\frac{\partial\left(\tilde{p}_i - p_b\right)}{\partial \tilde{p}_i}\times(\tilde{p}_i - p_a) \\ &\quad + \frac{1}{|p_a - p_b|}(\tilde{p}_i - p_b)\times\frac{\partial\left(\tilde{p}_i - p_a\right)}{\partial \tilde{p}_i} \\ &= -\frac{1}{|p_a - p_b|}(\tilde{p}_i - p_a)\times + \frac{1}{|p_a - p_b|}(\tilde{p}_i - p_b)\times \\ &= \frac{1}{|p_a - p_b|}(p_a - p_b)_\times\end{aligned}$$

$$\frac{\partial \tilde{p}_i}{\partial \boldsymbol{t}} = \boldsymbol{I}$$

$$\begin{aligned}\frac{\partial \tilde{p}_i}{\partial \delta\boldsymbol{\theta}} &= \frac{\partial\left(\exp(\delta\boldsymbol{\theta})\boldsymbol{R}p_i + \boldsymbol{t}\right)}{\partial \delta\boldsymbol{\theta}} \\ &= \frac{\partial\left(\left(\boldsymbol{I} + \delta\boldsymbol{\theta}^\wedge\right)\boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}} \\ &= \frac{\partial\left(\delta\boldsymbol{\theta}^\wedge \boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}} \\ &= \frac{-\left(\boldsymbol{R}p_i\right)_\times \delta\boldsymbol{\theta}}{\partial \delta\boldsymbol{\theta}} \\ &= -\left(\boldsymbol{R}p_i\right)_\times\end{aligned}$$

公式推导

●线特征

最后得到

$$\frac{\partial d_{\mathcal{E}}}{\partial \boldsymbol{t}} = \frac{\boldsymbol{d}_{\mathcal{E}}^T}{d_{\mathcal{E}}} \frac{1}{|p_a - p_b|} (p_a - p_b)_\times$$

$$\frac{\partial d_{\mathcal{E}}}{\partial \delta\boldsymbol{\theta}} = -\frac{\boldsymbol{d}_{\mathcal{E}}^T}{d_{\mathcal{E}}} \frac{1}{|p_a - p_b|} (p_a - p_b)_\times (\boldsymbol{R}p_i)_\times$$

公式推导

●面特征

残差：

$$d_{\mathcal{H}} = |\boldsymbol{d}_{\mathcal{H}}|$$

其中

$$\boldsymbol{d}_{\mathcal{H}} = (\tilde{p}_i - p_j) \cdot \frac{(p_l - p_j) \times (p_m - p_j)}{|(p_l - p_j) \times (p_m - p_j)|}$$

根据链式求导法则，残差对位移和姿态的雅可比写为

$$\frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{T}} = \frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{d}_{\mathcal{H}}} \frac{\partial \boldsymbol{d}_{\mathcal{H}}}{\partial \tilde{p}_i} \frac{\partial \tilde{p}_i}{\partial \boldsymbol{T}}$$

其中 $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t} \\ \delta\boldsymbol{\theta} \end{bmatrix}$，旋转采用SO3左扰动模型，即 $\boldsymbol{R}' = \exp(\boldsymbol{\theta})\boldsymbol{R}$

公式推导

●面特征

对雅可比中的3个因子依次求解:

$$\frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{d}_{\mathcal{H}}} = \frac{\boldsymbol{d}_{\mathcal{H}}^T}{d_{\mathcal{H}}} \qquad \frac{\partial \boldsymbol{d}_{\mathcal{H}}}{\partial \tilde{p}_i} = \frac{\left((p_l - p_j) \times (p_m - p_j)\right)^T}{\left|(p_l - p_j) \times (p_m - p_j)\right|}$$

$$\frac{\partial \tilde{p}_i}{\partial \boldsymbol{t}} = \boldsymbol{I}$$

$$\begin{aligned}
\frac{\partial \tilde{p}_i}{\partial \delta\boldsymbol{\theta}} &= \frac{\partial \left(\exp(\delta\boldsymbol{\theta})\boldsymbol{R}p_i + \boldsymbol{t}\right)}{\partial \delta\boldsymbol{\theta}} \\
&= \frac{\partial \left(\left(\boldsymbol{I} + \delta\boldsymbol{\theta}^{\wedge}\right)\boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}} \\
&= \frac{\partial \left(\delta\boldsymbol{\theta}^{\wedge}\boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}} \\
&= \frac{-\left(\boldsymbol{R}p_i\right)_{\times}\delta\boldsymbol{\theta}}{\partial \delta\boldsymbol{\theta}} \\
&= -\left(\boldsymbol{R}p_i\right)_{\times}
\end{aligned}$$

公式推导

- 面特征

最后得到

$$\frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{t}} = \frac{\boldsymbol{d}_{\mathcal{H}}^T}{d_{\mathcal{H}}} \frac{((p_l - p_j) \times (p_m - p_j))^T}{|(p_l - p_j) \times (p_m - p_j)|}$$

$$\frac{\partial d_{\mathcal{H}}}{\partial \delta \boldsymbol{\theta}} = \frac{\boldsymbol{d}_{\mathcal{H}}^T}{d_{\mathcal{H}}} \frac{((p_l - p_j) \times (p_m - p_j))^T}{|(p_l - p_j) \times (p_m - p_j)|} (-\boldsymbol{R} p_i)_{\times}$$

编程实现新模型的解析式求导

1) 将激光里程计(aloam_laser_odometry_node)部分CERES优化问题的因子改为解析式求导。

2) 可以基于SE3扰动模型，此时优化参数块只有位姿，需要自定义类，继承自ceres::LocalParameterization，GlobalSize为7，LocalSize为6。

3) 也可以基于SO3扰动模型，此时优化参数块分为平移和旋转，旋转需要自定义类，继承自ceres::LocalParameterization，GlobalSize为4，LocalSize为3。

4) 框架里是分为两个参数块实现的，优化问题中需要分别添加，如下所示

```
ceres::Problem problem(problem_options);
problem.AddParameterBlock(para_q, 4, q_parameterization);
problem.AddParameterBlock(para_t, 3);
```

5) 框架中残差的定义可放在aloam_factor.hpp中，通过以下方式加入到优化问题中。

```
problem.AddResidualBlock(cost_function, loss_function, para_q, para_t);
```

6) 另外注意去畸变参数默认为s=1即可。

Ceres 入门笔记

深蓝学院
shenlanxueyuan.com

初级入门可参考高博slam14讲，ceres 曲线拟合。

解析求导需要对雅克比进行赋值，因为ceres默认的更新形式是加减方式，而so3的四元数更新方式不同，所以需要自定义旋转参数块（顾名思义：自定义delta_q更新运算法则）

Ceres 入门笔记
自定义旋转参数块：LocalParameterization参数化

参考博客：

Ceres详解（一） Problem类

Ceres（二）LocalParameterization参数化

优化库——ceres（二）深入探索ceres::Problem

对于四元数或者旋转矩阵这种使用过参数化表示旋转的方式，它们是不支持广义的加法（因为使用普通的加法就会打破其 constraint，比如旋转矩阵加旋转矩阵得到的就不再是旋转矩阵），所以我们在使用ceres对其进行迭代更新的时候就需要自定义其更新方式了，具体的做法是实现一个参数本地化的子类，需要继承于LocalParameterization，LocalParameterization是纯虚类，所以我们继承的时候要把所有的纯虚函数都实现一遍才能使用该类生成对象。

除了不支持广义加法要自定义参数本地化的子类外，如果你要对优化变量做一些限制也可以如法炮制，比如ceres中slam2d example中对角度范围进行了限制.

Ceres 入门笔记
自定义旋转参数块：LocalParameterization参数化

LocalParameterization　自定义实现

这里以四元数为例子，解释如何实现参数本地化，需要注意的是，QuaternionParameterization中表示四元数中四个量在内存中的存储顺序是[w, x, y, z]，而Eigen内部四元数在内存中的存储顺序是[x, y, z, w]，但是其构造顺序是[w, x, y, z]（不要被这个假象给迷惑），所以就要使用另一种参数本地化类，即EigenQuaternionParameterization，下面就以QuaternionParameterization为例子说明，如下：

Ceres 入门笔记
自定义旋转参数块：LocalParameterization参数化

LocalParameterization　自定义实现

```
class CERES_EXPORT QuaternionParameterization : public LocalParameterization {
 public:
  virtual ~QuaternionParameterization() {}
  virtual bool Plus(const double* x,
                    const double* delta,
                    double* x_plus_delta) const;
  virtual bool ComputeJacobian(const double* x,
                               double* jacobian) const;
  virtual int GlobalSize() const { return 4; }
  virtual int LocalSize() const { return 3; }
};
```

Ceres 入门笔记
自定义旋转参数块：LocalParameterization参数化

LocalParameterization　自定义实现

1.GlobalSize()

　表示参数的自由度（可能有冗余），比如四元数的自由度是4，旋转矩阵so3的自由度是3

Ceres 入门笔记
自定义旋转参数块：LocalParameterization参数化

LocalParameterization 自定义实现

2.LocalSize()

表示 Δx 所在的正切空间（tangent space）的自由度，那么这个自由度是多少呢？下面进行解释:

正切空间是流形（manifold）中概念,我们可以这么理解manifold：

A manifold is a mathematical space that is not necessarily Euclidean on a global scale, but can be seen as Euclidean on a local scale

上面的意思是说，SO3 空间是属于非欧式空间的，但是没关系，我们只要保证旋转的局部是欧式空间，就可以使用流形进行优化了. 比如用四元数表示的3D旋转是属于非欧式空间的，那么我们取四元数的向量部分作为优化过程中的微小增量（因为是小量，所以不存在奇异性）。

Ceres 入门笔记
自定义旋转参数块：LocalParameterization参数化

LocalParameterization　自定义实现

3.Plus()

自定义优化变量更新函数

4、 ComputeJacobian()

参考[2]中的公式24，使用链式法则我们知道 $\tilde{\mathbf{J}}_{ij}$ 由两部分构成，第一部分 $\frac{\partial \mathbf{e}_{ij}(\check{\mathbf{x}})}{\partial \check{\mathbf{x}}_i}$ 是对原始过参数化的优化变量（比如，四元数）的导数，这个很容易求得，直接借助ceres的 AutoDiffCostFunction() 计算即可，或者自己计算雅可比矩阵，实现一个costfunction，关键是第二部分是如何求得的呢？

$$\tilde{\mathbf{J}}_{ij} = \left.\frac{\partial \mathbf{e}_{ij}(\check{\mathbf{x}} \boxplus \boldsymbol{\Delta}\tilde{\mathbf{x}})}{\partial \Delta\tilde{\mathbf{x}}}\right|_{\Delta\tilde{\mathbf{x}}=0} = \frac{\partial \mathbf{e}_{ij}(\check{\mathbf{x}})}{\partial \check{\mathbf{x}}_i} \cdot \left.\frac{\check{\mathbf{x}}_i \boxplus \boldsymbol{\Delta}\tilde{\mathbf{x}}_i}{\partial \boldsymbol{\Delta}\tilde{\mathbf{x}}_i}\right|_{\Delta\tilde{\mathbf{x}}=0}$$

现在求解 $\left.\frac{\partial \check{\mathbf{x}}_i \boxplus \boldsymbol{\Delta}\tilde{\mathbf{x}}_i}{\partial \boldsymbol{\Delta}\tilde{\mathbf{x}}_i}\right|_{\boldsymbol{\Delta}\tilde{\mathbf{x}}=\mathbf{0}}$，以四元数为例：

$$\frac{\partial \check{\mathbf{q}}_i \boxplus \boldsymbol{\Delta}\tilde{\mathbf{q}}_i}{\partial \boldsymbol{\Delta}\tilde{\mathbf{q}}_i} = \frac{\partial \check{\mathbf{q}}_i \otimes e^{\boldsymbol{\Delta}\tilde{\mathbf{q}}_i}}{\partial \boldsymbol{\Delta}\tilde{\mathbf{q}}_i} \approx \frac{\partial \check{\mathbf{q}}_i \otimes \begin{bmatrix} 1 \\ \boldsymbol{\Delta}\tilde{\mathbf{q}}_i \end{bmatrix}}{\partial \boldsymbol{\Delta}\tilde{\mathbf{q}}_i} = \frac{\partial [\check{\mathbf{q}}_i]_L \begin{bmatrix} 1 \\ \boldsymbol{\Delta}\tilde{\mathbf{q}}_i \end{bmatrix}}{\partial \boldsymbol{\Delta}\tilde{\mathbf{q}}_i}$$

注意上面符号的变化，先从 $\boxplus$ 变成四元数乘法 $\otimes$，最后变成普通的乘法，进一步得到，

$$\begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & -q_z & q_y \\ q_y & q_z & q_w & -q_x \\ q_z & -q_y & q_x & q_w \end{bmatrix} \frac{\partial \begin{bmatrix} 1 \\ \boldsymbol{\Delta}\tilde{\mathbf{q}}_i \end{bmatrix}}{\partial \boldsymbol{\Delta}\tilde{\mathbf{q}}_i} = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & -q_z & q_y \\ q_y & q_z & q_w & -q_x \\ q_z & -q_y & q_x & q_w \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -q_x & -q_y & -q_z \\ q_w & -q_z & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \end{bmatrix}$$

最后得到雅可比矩阵是4*3维的，代码里使用一个size为12的数组存储.

Ceres 入门笔记
自定义旋转参数块：LocalParameterization参数化

LocalParameterization 自定义实现

5.Problem::AddParameterBlock()

自定义参数块后，需要通过 Problem::AddParameterBlock 自定义参数快加载

```
void Problem::AddParameterBlock(double *values, int size, LocalParameterization *local_parameterization)
```

Ceres 入门笔记
CostFunction 定义

深蓝学院
shenlanxueyuan.com

重点参考博客：

Ceres详解（一） Problem类

Ceres详解（二） CostFunction

基于线面特征解析求导的实现

深蓝学院 shenlanxueyuan.com

代码详见 README.md

主要完成三个模块：

1.自定义线特征残差块

$$\boldsymbol{d}_{\mathcal{E}} = \frac{(\tilde{p}_i - p_b) \times (\tilde{p}_i - p_a)}{|p_a - p_b|}$$

2.自定义面特征残差块

$$d_{\mathcal{H}} = (\tilde{p}_i - p_j) \cdot \frac{(p_l - p_j) \times (p_m - p_j)}{|(p_l - p_j) \times (p_m - p_j)|}$$

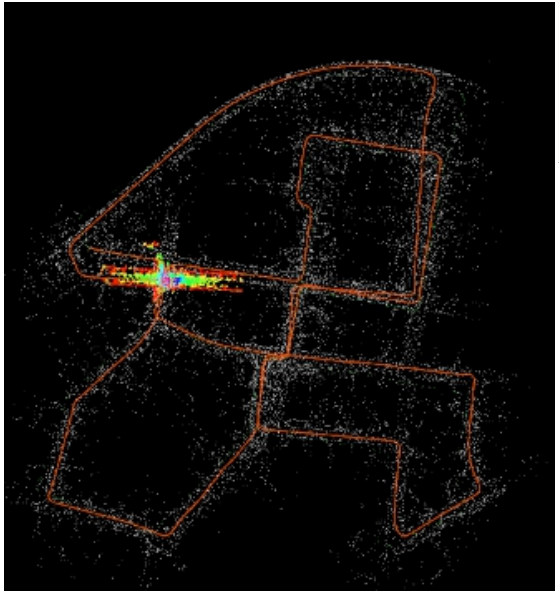
3.自定义旋转残差块

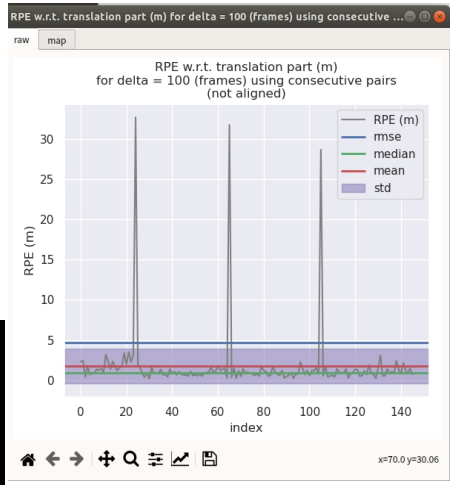
可使用ceres 自带的一个自定义四元数参数快 EigenQuaternionParameterization

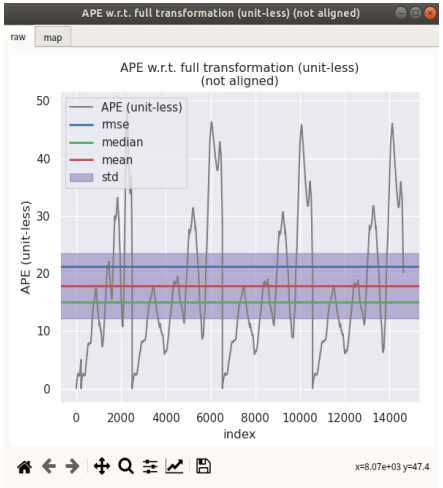
也可以自己订阅四元数更新参数块（详细见第三章作业参考.pdf)

evo 评估

自动求导结果







evo_rpe

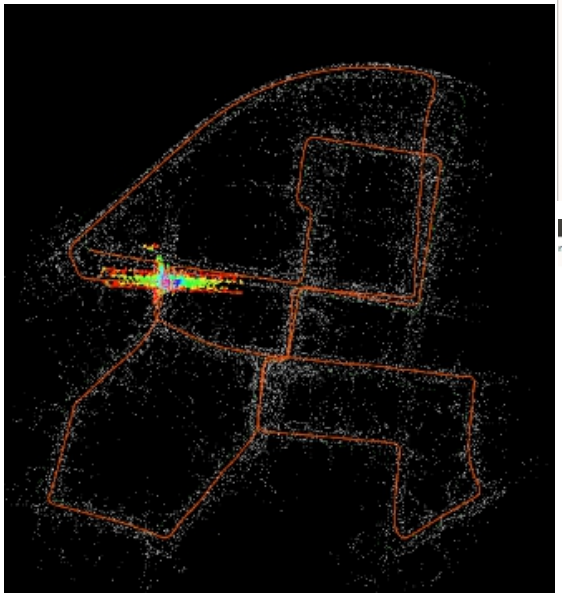
max	32.674307
mean	1.727705
median	0.949400
min	0.174067
rmse	4.633799
sse	3134.926234
std	4.299667

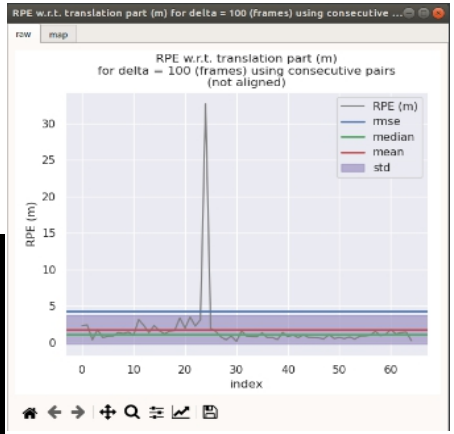
evo_ape

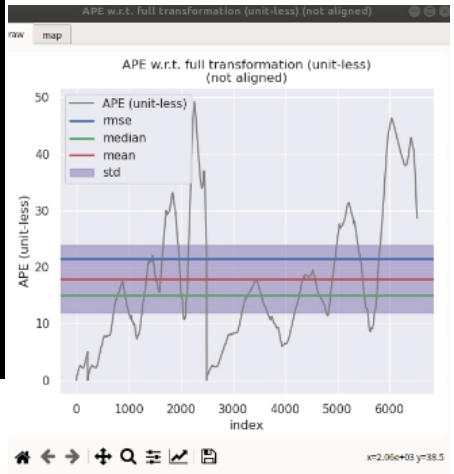
max	49.216907
mean	17.845561
median	15.060993
min	0.000000
rmse	21.164576
sse	6550216.305710
std	11.378720

evo 评估

解析求导结果







evo_rpe

max	32.674307
mean	1.752770
median	1.073263
min	0.174067
rmse	4.309392
sse	1207.105821
std	3.936833

evo_ape

max	49.216907
mean	17.908280
median	15.086003
min	0.000000
rmse	21.509767
sse	3021235.643653
std	11.914847

总结

图中结果可以看出，缺乏回环矫正的情况下在拐弯的情况下由于实现流程和缺乏回环导致误差会偏大，解析求求导和自动求导的结果相似。总体来看轨迹的趋势还是正确的，算法应该没有问题，但是参数调整还有改进的余地。

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听

Thanks for Listening !