视觉定位与感知-HW3

516021910700 郭远帆

1.

设:

$$\boldsymbol{\phi_1} = \boldsymbol{x}^\wedge = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

$$\boldsymbol{\phi_2} = \boldsymbol{y}^\wedge = \begin{bmatrix} 0 & -y_3 & y_2 \\ y_3 & 0 & -y_1 \\ -y_2 & y_1 & 0 \end{bmatrix}$$

$$\boldsymbol{\phi_3} = \boldsymbol{z}^\wedge = \begin{bmatrix} 0 & -z_3 & z_2 \\ z_3 & 0 & -z_1 \\ -z_2 & z_1 & 0 \end{bmatrix}$$

首先验证封闭性:

$$\boldsymbol{\phi_1}\boldsymbol{\phi_2} - \boldsymbol{\phi_2}\boldsymbol{\phi_1} = \begin{bmatrix} -x_2y_2 - x_3y_3 & x_2y_1 & x_3y_1 \\ x_1y_2 & -x_3y_3 - x_1y_1 & x_3y_2 \\ x_1y_3 & x_2y_3 & -x_2y_2 - x_1y_1 \end{bmatrix}$$

$$- \begin{bmatrix} -x_2y_2 - x_3y_3 & x_1y_2 & x_1y_3 \\ x_2y_1 & -x_3y_3 - x_1y_1 & x_2y_3 \\ x_3y_1 & x_3y_2 & -x_2y_2 - x_1y_1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & x_2y_1 - x_1y_2 & x_3y_1 - x_1y_3 \\ x_1y_2 - x_2y_1 & 0 & x_3y_2 - x_2y_3 \\ x_1y_3 - x_3y_1 & x_2y_3 - x_3y_2 & 0 \end{bmatrix}$$

可以看到结果也是反对称矩阵，封闭性得以验证。

接下来验证双线性:

$$\begin{aligned}
[\mathrm{a}\boldsymbol{\phi_1} + \mathrm{b}\boldsymbol{\phi_2}, \boldsymbol{\phi_3}] &= (\mathrm{a}\boldsymbol{\phi_1} + \mathrm{b}\boldsymbol{\phi_2})]\boldsymbol{\phi_3} - \boldsymbol{\phi_3}(a\boldsymbol{\phi_1} + b\boldsymbol{\phi_2}) \\
&= a(\boldsymbol{\phi_1}\boldsymbol{\phi_3} - \boldsymbol{\phi_3}\boldsymbol{\phi_1}) + b(\boldsymbol{\phi_2}\boldsymbol{\phi_3} - \boldsymbol{\phi_3}\boldsymbol{\phi_2}) \\
&= \mathrm{a}[\boldsymbol{\phi_1}, \boldsymbol{\phi_3}] + \mathrm{b}[\boldsymbol{\phi_1}, \boldsymbol{\phi_2}] \\
[\boldsymbol{\phi_3}, \mathrm{a}\boldsymbol{\phi_1} + \mathrm{b}\boldsymbol{\phi_2}] &= \boldsymbol{\phi_3}(\mathrm{a}\boldsymbol{\phi_1} + \mathrm{b}\boldsymbol{\phi_2}) - (a\boldsymbol{\phi_1} + b\boldsymbol{\phi_2})\boldsymbol{\phi_3} \\
&= a(\boldsymbol{\phi_3}\boldsymbol{\phi_1} - \boldsymbol{\phi_1}\boldsymbol{\phi_3}) + b(\boldsymbol{\phi_3}\boldsymbol{\phi_2} - \boldsymbol{\phi_2}\boldsymbol{\phi_3}) \\
&= a[\boldsymbol{\phi_3}, \boldsymbol{\phi_1}] + b[\boldsymbol{\phi_3}, \boldsymbol{\phi_2}]
\end{aligned}$$

双线性得以验证。

接下来验证自反性。

$$[\boldsymbol{\phi_1}, \boldsymbol{\phi_1}] = \begin{bmatrix} -x_2x_2 - x_3x_3 & x_2x_1 & x_3x_1 \\ x_1x_2 & -x_3x_3 - x_1x_1 & x_3x_2 \\ x_1x_3 & x_2x_3 & -x_2x_2 - x_1x_1 \end{bmatrix}$$

$$- \begin{bmatrix} -x_2x_2 - x_3x_3 & x_1x_2 & x_1x_3 \\ x_2x_1 & -x_3x_3 - x_1x_1 & x_2x_3 \\ x_3x_1 & x_3x_2 & -x_2x_2 - x_1x_1 \end{bmatrix}$$

$$= \boldsymbol{0}$$

自反性得以验证。

最后验证雅可比等价性。

由之前的性质，该运算在向量形式上可以看作叉积(观察$\boldsymbol{\phi_1\phi_2}-\boldsymbol{\phi_2\phi_1}$结果可知复原的向量是原向量叉积的结果)

而在向量叉积中，有如下性质:

$$\vec{x}\times(\vec{y}\times\vec{z})+\vec{z}\times(\vec{x}\times\vec{y})+\vec{y}\times(\vec{z}\times\vec{x})=0$$

该性质与雅可比等价性一致，由此一致性可以得到雅可比等价性的验证，因此不再展开。

以上，验证了该李代数。实际上，该李代数即是三维向量$\mathbf{R^3}$上定义的叉积。

2.

对三维向量 x，我们假设其四元数表示为$[0,\mathrm{x},\mathrm{y},\mathrm{z}]^{\mathrm{T}}$，其中 x,y,z 为其三个坐标轴的坐标。根据四元数的运算法则，对于单位四元数$\mathbf{q}$，有:

$$\mathbf{q^{-1}}=\frac{\boldsymbol{q}^*}{||\boldsymbol{q}||^2}=\boldsymbol{q}^*$$

有$\mathbf{q}^*=\left[q_w,-q_x,-q_y,-q_z\right]^{\boldsymbol{T}}$

由四元数的乘法计算公式，有:

$$\begin{aligned}\mathbf{q}\otimes\mathbf{x}^{\sim}=-\mathrm{q_x}x-q_yy-q_zz\\+\left(q_wx+q_yz-q_zy\right)i\\+(q_wy-q_xz+q_zx)j\\+(q_wz+q_xy-q_yx)k\end{aligned}$$

要证明该变换是旋转变换，需要证明变换后的四元数$\mathbf{x}^{\sim\prime}$的实部为 0。根据四元数的乘法计算公式,有:

$$\begin{aligned}\mathrm{Re}\{\mathbf{q}\otimes\mathbf{x}^{\sim}\otimes\mathbf{q}^*\}=\mathrm{q_w}\left(-q_xx-q_yy-q_zz\right)\\+\mathrm{q_x}\left(q_wx+q_yz-q_zy\right)\\+\mathrm{q_y}(q_wy-q_xz+q_zx)\\+q_z(q_wz+q_xy-q_yx)\end{aligned}$$

最终所有项均可两两消去，有$\mathrm{Re}\{\mathbf{q}\otimes\mathbf{x}^{\sim}\otimes\mathbf{q}^*\}=0$,故该变换可以表示为旋转变换。利用乘法公式对虚部进行同样的操作，有:

$$\begin{aligned}\mathrm{I}\{\mathbf{q}\otimes\mathbf{x}^{\sim}\otimes\mathbf{q}^*\}=\mathrm{q_x}\left(q_xx+q_yy+q_zz\right)\\+\mathrm{q_w}\left(q_wx+q_yz-q_zy\right)\\+(-q_z)(q_wy-q_xz+q_zx)\\+q_y(q_wz+q_xy-q_yx)\end{aligned}$$

解得最终结果为: $\mathrm{I}\{\mathbf{q}\otimes\mathbf{x}^{\sim}\otimes\mathbf{q}^*\}=\left(1-2q_y^2-2q_z^2\right)x+\left(2q_xq_y-2q_wq_z\right)y+(2q_xq_z+2q_wq_y)z$

这实际上对应旋转矩阵 **R** 的第一行。对其它两个虚数部分进行同样的计算，可以得到最终旋转矩阵:

$$\mathbf{R}=\begin{bmatrix}1-2q_y^2-2q_z^2 & 2q_xq_y-2q_wq_z & 2q_xq_z+2q_wq_y\\2q_xq_y+2q_wq_z & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_wq_x\\2q_xq_z-2q_wq_y & 2q_yq_z+2q_wq_x & 1-2q_x^2-2q_y^2\end{bmatrix}$$

3.

实验原理:

设旋转矩阵 **R** 以及其扰动$\Delta\mathbf{R}$, 设扰动的李代数为$\boldsymbol{\theta}$, 有:

$$\frac{\partial(\mathbf{Rx})}{\partial\boldsymbol{\theta}} = -(\mathbf{Rx})\hat{}$$

$\boldsymbol{\theta}$为$\Delta\mathbf{R}$的李代数表示，将$\boldsymbol{\theta}$表示为旋转向量的形式:$\boldsymbol{\theta} = \theta\mathbf{n}$，其中$\mathbf{n}$为旋转轴（单位向量），$\theta$为旋转角。由此定义参数更新 $\mathbf{R} := \mathbf{R} \oplus \boldsymbol{\theta}$ 如下:

$$\mathbf{R} \oplus \boldsymbol{\theta} = \mathbf{R}(\mathbf{I} + [\boldsymbol{\theta}]\hat{})$$

其中符号^为向量的反对称矩阵表示(或者表示为$[\boldsymbol{\theta}]_\times$)。严格上来说，上式为近似公式，参数更新也可用下式表达:

$$\mathbf{R} \oplus \boldsymbol{\theta} = \mathbf{R}(\cos\theta\,\mathbf{I} + (1-\cos\theta)\boldsymbol{\theta}\boldsymbol{\theta}^{\mathbf{T}} + \sin\theta\,[\boldsymbol{\theta}]\hat{})$$

如果采用四元数来代表旋转，那么参数更新的定义$\mathbf{q} := \mathbf{q} \oplus \boldsymbol{\theta}$:

$$\mathbf{q} \oplus \boldsymbol{\theta} = \mathbf{q} \otimes \{\cos\frac{\theta}{2}, \mathbf{n}\sin\frac{\theta}{2}\}$$

当迭代到θ足够小时，采用近似公式表示:

$$\mathbf{q} \oplus \boldsymbol{\theta} = \mathbf{q} \otimes \{1, \frac{\boldsymbol{\theta}}{\mathbf{2}}\}$$

这里的乘符号是四元数的乘法。

根据高斯牛顿法的思想，每一轮迭代的近似优化目标为:

$$\min\sum_{\mathrm{i}}^{3}(\vec{y_i} - \boldsymbol{R}\exp([\Delta\theta]_\times)\,\vec{x_i})^2$$

其中$\Delta\theta$即扰动$\Delta\mathbf{R}$的李代数表示。根据扰动模型，近似为求解此方程:

$$\min\sum_{\mathrm{i}}^{3}(\vec{y_i} - \boldsymbol{R}\vec{x_i} + [\boldsymbol{R}\boldsymbol{x_i}]_\times\Delta\theta)^2$$

定义$\mathrm{z_i} = \vec{y_i} - \boldsymbol{R}\vec{x_i}$, $\mathbf{J_i} = [\mathbf{Rx_i}]_\times$，上式关于$\Delta\theta$ 导数为 0 可得:

$$\sum_{\mathrm{i}}^{3}\boldsymbol{J}_i^T\boldsymbol{J}_i\Delta\theta = -\sum_{i}^{3}\boldsymbol{J}_i^T\boldsymbol{z}_i$$

求解此方程，得:

$$\Delta\theta = -\left(\sum_{\boldsymbol{i}}^{\mathbf{3}}\boldsymbol{J}_i^T\boldsymbol{J}_i\right)^{-1}(\sum_{i}^{3}\boldsymbol{J}_i^T\boldsymbol{z}_i)$$

根据之前的参数更新定义，更新 **R**（或者 **q**）即可。最后对于更新后的 **R** 需要保证其正交性，对于 **q** 需要保证其单位性。采用 SVD 分解保证 **R** 的正交性，采用归一化保证 **q** 的单位性即可。

除此之外，参考《视觉 SLAM 十四讲》，3D 位姿估计问题还有一种 SVD 解法，其原理如

下:

定义两组点的质心如下:

$$\vec{y}_g=\frac{1}{3}\sum_{i=1}^{3}\vec{y}_i,\vec{x}_g=\frac{1}{3}\sum_{i=1}^{3}\overrightarrow{x_i}$$

然后得到这两组点的去质心坐标:

$$\overrightarrow{q_i}:=\overrightarrow{y_i}-\overrightarrow{y_g},\ \overrightarrow{p_i}:=\overrightarrow{x_1}-\overrightarrow{x_g}$$

定义矩阵 **W:**

$$\mathbf{W}=\sum_{\mathbf{i}=1}^{3}\boldsymbol{q}_i\boldsymbol{p}_i^T$$

对 **W** 进行 SVD 分解，得到:

$$\mathbf{W}=\mathbf{U\Sigma V^T}$$

然后可以解得 **R:**

$$\mathbf{R}=\mathbf{UV^T}$$

该方法是一种几何的方法，将多点的变换“浓缩”为质心之间的变换，在样本数大的情况下其效果不一定比高斯牛顿法好，但由于本次实验中样本点只有 3 个，我们采用这个方法来与高斯牛顿法进行对比。

实现关键思路:

代码实现通过 Matlab 语言，以下为实现的关键思路:

基础模块实现:

1. 为了计算向量的反对称矩阵，实现 X = skew(x) 函数，接受 3*1 向量 x 作为输入，输出其反对称矩阵 X
2. 实现 R = q2r(q) 以及 q = r2q（R）函数，实现四元数到旋转矩阵的转换
3. 误差计算函数 target(R,x,y) 计算目标函数
4. 实现 q = Qmult(q1,q2) 实现四元数的乘法

基础模块实现后，根据高斯牛顿法的迭代思想持续更新参数并记录误差即可。Matlab 对矩阵运算的支持相当完善，代码语言与数学语言近似，在此不再进行冗余的描述。

在本实验中分别实现了四元数表达方式、旋转矩阵表达方式下的高斯牛顿法，其功能接近，可用于验证计算是否正确。

实验结果与分析:

采用 SVD 方法得到的旋转矩阵为:

$$\mathbf{R_m}=\begin{matrix}0.8660 & -0.3536 & 0.3536\\ 0.3536 & 0.9330 & 0.0670\\ -0.3536 & 0.0670 & 0.9330\end{matrix}$$

给出的误差约为2^{-31}，在此基础上加入随机扰动项，然后在不同扰动的情况下开始迭代，观察扰动大小不同情况下的收敛速度如表 1(迭代 x 次项中为得到的误差)。

表 1 实验数据表

扰动大小	初始误差	迭代 2 次	迭代 5 次	迭代 10 次	迭代 20 次	迭代 50 次	迭代 100 次
0.01	1.21e-4	7.82e-6	7.78e-8	1.42e-15	2.32e-20	2.27e-31	8.51e-32
0.1	6.52	0.093	2.61e-4	1.1e-6	2.17e-12	2.27e-29	2.54e-32
1	7.13	0.014	2.42e-4	3.02e-7	5.52e-13	6.45e-30	4.22e-31
10	7.66	0.031	1.27e-4	1.81e-7	3.22e-13	1.51e-30	2.50e-32
随机正交基	10.31	0.85	0.013	8.81e-5	1.43e-10	1.15e-27	1.91e-31

随机正交基下的误差变换图:

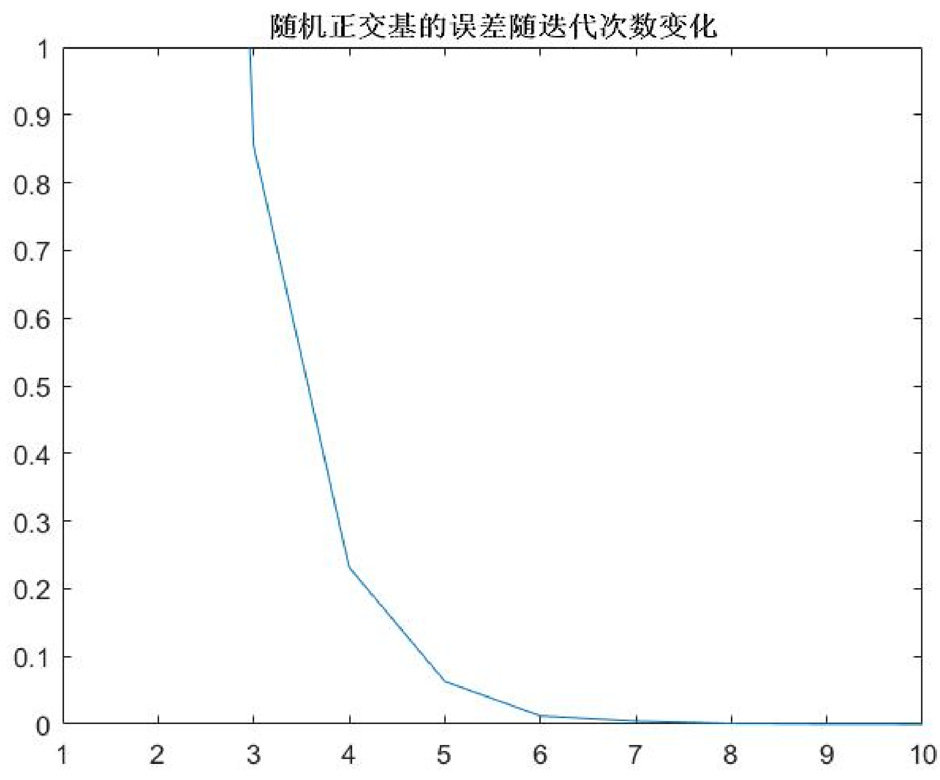

图 1

可以看到，初始误差越大，收敛的速度也就较慢，然而无论如何，10 代之后已经有了相当可观的结果，迭代到 50 代已经接近最优解。

总结:

本实验中对高斯牛顿法在 3D-3D 位姿估计问题中的应用进行了实现，对比了不同初值情况下的高斯牛顿法收敛结果，并比较了 SVD 方法与高斯牛顿法的误差结果。在样本点较小的情况下，SVD 方法可以较快地得到精确的结果，但相比高斯牛顿法，计算开销较大。

本实验的代码由 Matlab 编写，实现了基于旋转矩阵和四元数表达的高斯牛顿法模型以及 SVD 求解方法。

感想和体会:

第三题是踩了不少的坑啊！由于理论知识不够扎实，对于李代数的扰动模型不熟悉，对于参数更新的不理解，一开始得到的结果千奇百怪。但也正因为如此，去查阅了各种资料，将高翔博士的《视觉 SLAM 十四讲》看了一遍又一遍，将老师的 PPT 温习了一遍又一遍，最后终于成功地分别实现了四元数表达的高斯牛顿法、旋转矩阵表达的高斯牛顿法，还把 SVD 方法（基于几何学）也实现了一遍，历时约两个星期，终于将这个问题解了出来。

感谢老师的指导和支持！SLAM 真的很有趣，本次作业相当精彩