

视觉SLAM理论与实践 第三次作业讲评





2 群的性质



1. {Z, +} 是否为群?若是,验证其满足群定义;若不是,说明理由。

2. {N, +} 是否为群?若是,验证其满足群定义;若不是,说明理由。 其中 Z 为整数集,N 为自然数集。

1. 是

2. 否, 逆不满足。

3 验证向量叉乘的李代数性质



- 1. 封闭性
- 2. 双线性
- 自反性
 上述三个通过差乘的基本性质就可以看出来。
- 4. 雅可比等价 三矢量叉乘展开成点乘:

$$\forall X, Y, Z \in \mathbb{R}^3, X \times (Y \times Z) = (X \cdot Z) \cdot Y - (X \cdot Y) \cdot Z$$

参考: 知乎搜 "三矢量叉乘展开成点乘的公式如何证明?"

4 推导 SE(3) 的指数映射



已知性质:

1. 正弦和余弦函数的泰勒展开

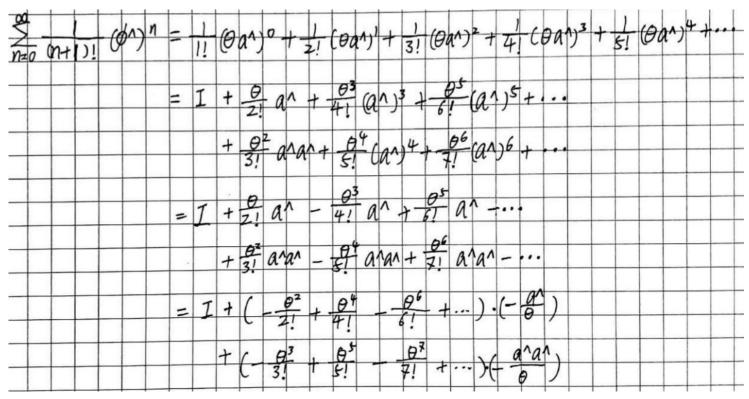
$$\sin x = \sum_{n=0}^{\infty} rac{(-1)^n}{(2n+1)!} x^{2n+1} \qquad \qquad = x - rac{x^3}{3!} + rac{x^5}{5!} - \cdots \qquad \qquad ext{for all } x$$
 $\cos x = \sum_{n=0}^{\infty} rac{(-1)^n}{(2n)!} x^{2n} \qquad \qquad = 1 - rac{x^2}{2!} + rac{x^4}{4!} - \cdots \qquad \qquad ext{for all } x$

2. SLAM14讲的
$$(4.20)$$
和 (4.21) 公式 $a^{\wedge}a^{\wedge}=aa^{T}-I$ $a^{\wedge}a^{\wedge}a^{\wedge}=-a^{\wedge}$

4 推导 SE(3) 的指数映射



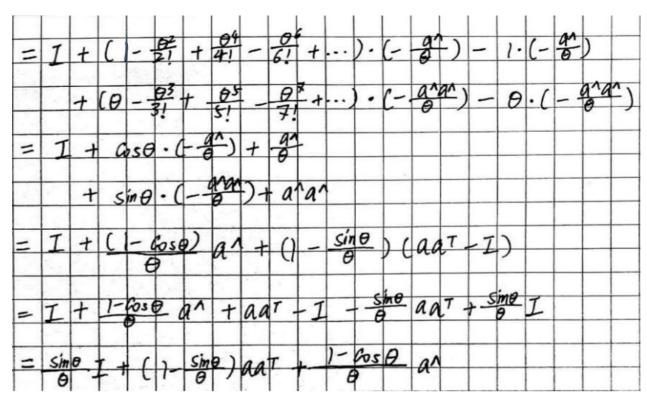
推导:



4 推导 SE(3) 的指数映射



推导:

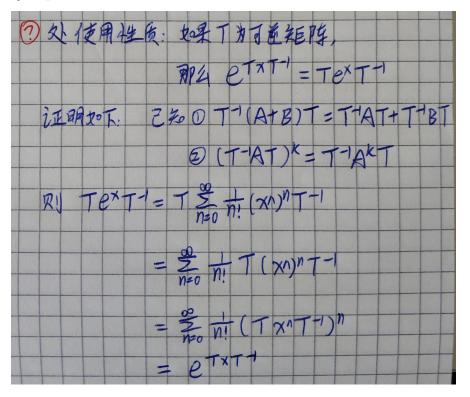


5 伴随



按照题目中的提示一步步证明即可

1. $iIR: \forall \vec{\alpha} \in \mathbb{R}^3, R \vec{\alpha}^{\Lambda} R^{T} = (Ra)^{\Lambda}$
左边: R R R R R R R R R R R R R R R R R R R
$=R(\bar{\alpha}\times\bar{\alpha})$
$ \overline{5} $ $ \overline{R} $
$= \mathcal{R}(\vec{R} \times \vec{u})$
2. IEBA: Rexp(\$^) RT = exp((RP)^)
ZZo: (Rp)N = Rp^RT
RV: exp((RA))) = exp(RP)RT)
$= R \exp(p^n) R^{T} $



6 轨迹的描绘



6.1 事实上,Twc的平移部分即构成了机器人的轨迹。它的物理意义是什么?为何画出 Twc 的平移部分就得到了机器人的轨迹?

Twc的物理意义: 机器人在某时刻相对于世界坐标系原点的位置。

因为在某个时间段内,通过在每一个离散时刻机器人相对于世界坐标系的位置,我们可以估计出它的运动轨迹。轨迹的具体形式(折线或者曲线)取决于算法和离散时刻的间隔。

6 轨迹的描绘



思路: 使用C++的标准库 读入数据, 并用Sophus库 保存成SE3。

TODO部分的代码见右 图。

```
string line;
double t,t_x,t_y,t_z,q_w,q_x,q_y,q_z;
ifstream myfile(trajectory file);
if (!myfile)
cerr << "can't open the file" << endl;
exit(1);
while (!myfile.eof())
getline(myfile, line);
stringstream stream(line);
stream >> t >> t_x >> t_y >> t_z >> q_x >> q_y >> q_z >> q_w;
Eigen::Vector3d t(t x, t y, t z);
Eigen::Quaterniond q(q_w,q_x,q_y,q_z);
q.normalize();
Sophus::SE3 SE3_qt(q,t);
poses.push back(SE3 qt);
```

7 * 轨迹的误差



1. 使用Sophus按照公式计算RMSE

```
double RMSE(vector<Sophus::SE3, Eigen::aligned allocator<Sophus::SE3>> T g,
vector<Sophus::SE3, Eigen::aligned allocator<Sophus::SE3>> T i)
int T_g_length = T_g.size();
double errors = 0;
Eigen::Matrix<double,6,1> se3;
for(int i=0; i< T_g_length; i++)
se3 = (T_g[i].inverse()*T_i[i]).log();
errors += se3.norm()*se3.norm();
errors = sqrt(errors/T g length);
return errors;
```

7*轨迹的误差



2. 画出两个轨迹。修改画图函数,主要是使用了两个 for 循环对 Ti 和 Tg 的轨迹进行绘制:

```
for (size t i = 0; i < T g.size() - 1; i++) {
glColor3f(1 - (float) i / T g.size(), 0.0f, (float) i / T g.size());
glBegin(GL LINES);
auto p1 = T_g[i], p2 = T_g[i + 1];
glVertex3d(p1.translation()[0], p1.translation()[1], p1.translation()[2]);
glVertex3d(p2.translation()[0], p2.translation()[1], p2.translation()[2]);
glEnd();
for (size t i = 0; i < T_i.size() - 1; i++) {
glColor3f(1 - (float) i / T_i.size(), 0.0f, (float) i / T_i.size());
glBegin(GL LINES);
auto p1 = T_i[i], p2 = T_i[i + 1];
glVertex3d(p1.translation()[0], p1.translation()[1], p1.translation()[2]);
glVertex3d(p2.translation()[0], p2.translation()[1], p2.translation()[2]);
glEnd();
```



感谢各位聆听 Thanks for Listening



