深蓝学院
shenlanxueyuan.com

第三章作业思路提示

主讲人 汪正涛

群的性质

深蓝学院 shenlanxueyuan.com

●Q2易错点：逆元的概念

One of the most familiar groups is the set of integers

$$\mathbb{Z} = \{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$$

together with addition.[3] For any two integers a and b, the sum $a + b$ is also an integer; this *closure* property says that + is a binary operation on $\mathbb{Z}$. The following properties of integer addition serve as a model for the group axioms in the definition below.

- For all integers a, b and c, one has $(a + b) + c = a + (b + c)$. Expressed in words, adding a to b first, and then adding the result to c gives the same final result as adding a to the sum of b and c. This property is known as *associativity*.
- If a is any integer, then $0 + a = a$ and $a + 0 = a$. Zero is called the *identity element* of addition because adding it to any integer returns the same integer.
- For every integer a, there is an integer b such that $a + b = 0$ and $b + a = 0$. The integer b is called the *inverse element* of the integer a and is denoted $-a$.

The integers, together with the operation +, form a mathematical object belonging to a broad class sharing similar structural aspects. To appropriately understand these structures as a collective, the following definition is developed.

Inverse element

From Wikipedia, the free encyclopedia

See also: Inversion

In abstract algebra, the idea of an **inverse element** generalises the concepts of negation (sign reversal) (in relation to addition) and reciprocation (in relation to multiplication). The intuition is of an element that can 'undo' the effect of combination with another given element. While the precise definition of an inverse element varies depending on the algebraic structure involved, these definitions coincide in a group.

https://en.wikipedia.org/wiki/Inverse_element

向量叉乘的李代数性质

- Q3大致有两种证明方式：
  - 1. 直接展开进行证明
  - 2. 利用向量叉乘的相关运算法则与公式简化证明。如证明雅克比等价可以用向量叉乘的拉格朗日公式

$$X \times (Y \times Z) = Y(X \cdot Z) - Z(X \cdot Y)$$

SE(3)的指数映射

- SE(3)的指数映射的推导难点主要在于左雅克比，其关键在于泰勒展开与“凑配”。至于整个映射的推导可看右图（表示稍有不同）。

$$\exp\left(\begin{bmatrix}\Omega & p\\ 0 & 0\end{bmatrix}\right)=\sum_{n=0}^{\infty}\frac{1}{n!}\left(\begin{bmatrix}\Omega & p\\ 0 & 0\end{bmatrix}\right)^n$$

$$=I+\begin{bmatrix}\Omega & p\\ 0 & 0\end{bmatrix}+\frac{1}{2!}\begin{bmatrix}\Omega^2 & \Omega p\\ 0 & 0\end{bmatrix}+\frac{1}{3!}\begin{bmatrix}\Omega^3 & \Omega^2 p\\ 0 & 0\end{bmatrix}+\cdots$$

所以:

$$\exp\left(\begin{bmatrix}\Omega & p\\ 0 & 0\end{bmatrix}\right)=\begin{bmatrix}\exp(\Omega) & Vp\\ 0 & 1\end{bmatrix}$$

$$V=I+\frac{1}{2!}\Omega+\frac{1}{3!}\Omega^2+\cdots$$

伴随

深蓝学院 shenlanxueyuan.com

●Q5根据提示可以轻松完成

Recall that for a vector $\mathbf{u}$, the matrix $\mathbf{u}_\times$ is defined as the linear transformation

$$\mathbf{u}_\times :: \mathbf{v} \mapsto \mathbf{u} \times \mathbf{v}.$$

So the identity $\mathbf{R} \cdot \boldsymbol{\omega}_\times \cdot \mathbf{R}^{-1} = (\mathbf{R}\boldsymbol{\omega})_\times$ can be intuitively interpreted as "[unrotating, then crossing with $\boldsymbol{\omega}$, then rotating] is the same as [crossing with a rotated $\boldsymbol{\omega}$]". We can prove this identity by letting the RHS of the identity act upon an arbitrary vector $\mathbf{v}$:

$$(\mathbf{R}\boldsymbol{\omega})_\times \mathbf{v} = (\mathbf{R}\boldsymbol{\omega}) \times \mathbf{v} = (\mathbf{R}\boldsymbol{\omega}) \times (\mathbf{R}\mathbf{R}^{-1}\mathbf{v}) = \mathbf{R}\left[\boldsymbol{\omega} \times (\mathbf{R}^{-1}\mathbf{v})\right] = \mathbf{R}\boldsymbol{\omega}_\times \mathbf{R}^{-1}\mathbf{v},$$

where we have used the fact that for any rotation matrix U and vectors $\mathbf{a}, \mathbf{b}$ we have

$$(U\mathbf{a}) \times (U\mathbf{b}) = U(\mathbf{a} \times \mathbf{b}).$$

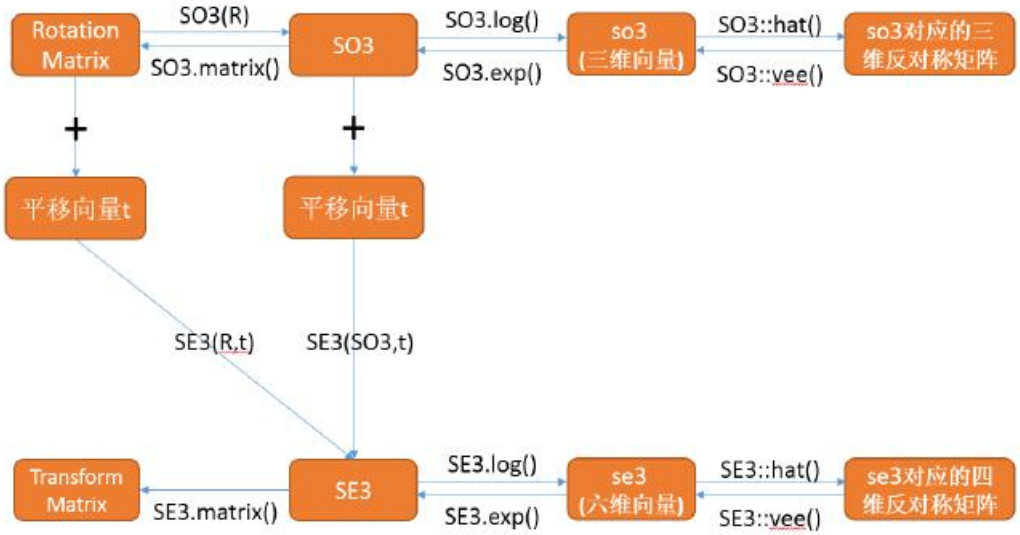
轨迹的描绘与误差

深蓝学院 shenlanxueyuan.com

$$[t, t_x, t_y, t_z, q_x, q_y, q_z, q_w],$$

SE3

$$e_i = \| \log(\boldsymbol{T}_{gi}^{-1} \boldsymbol{T}_{ei})^{\vee} \|_2.$$



SO3, so3, SE3和se3的相互转换关系

——jingguo

轨迹的描绘与误差

深蓝学院 shenlanxueyuan.com

```cpp
void DrawTrajectory(vector<Sophus::SE3d,
Eigen::aligned_allocator<Sophus::SE3d>> poses) {
    if (poses.empty()) {
        cerr << "Trajectory is empty!" << endl;
        return;
    }

    // create pangolin window and plot the trajectory
    pangolin::CreateWindowAndBind("Trajectory Viewer", 1024, 768);  //创建一个显示窗口
    // glEnable 用于启用各种功能。功能由参数决定。与glDisable相对应。glDisable是用来关闭的。两个函数参数取值是一致的。
    // link: https://www.cnblogs.com/icmzn/articles/5741484.html
    glEnable(GL_DEPTH_TEST);    //启用深度测试，根据坐标的远近自动隐藏被遮住的图形
    glEnable(GL_BLEND); //启用颜色混合。例如实现半透明效果
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    //glBlendFunc(GLenum sfactor,GLenum dfactor) 前者sfactor表示源因子，后者dfactor表示目标因子
    //GL_SRC_ALPHA：表示使用源颜色的alpha值来作为因子
    //GL_ONE_MINUS_SRC_ALPHA：表示用1.0减去源颜色的alpha值来作为因子（1-alpha）

    //定义投影，初始化模型视角矩阵
    pangolin::OpenGlRenderState s_cam(
            pangolin::ProjectionMatrix(1024, 768, 500, 500, 512, 389, 0.1, 1000),
            //ProjectMatrix(int h, int w, int fu, int fv, int cu, int cv, int znear, int zfar)
            pangolin::ModelViewLookAt(0, -0.1, -1.8, 0, 0, 0, 0.0, -1.0, 0.0)
            //ModelViewLookAt(double x, double y, double z,double lx, double ly, double lz, AxisDirection Up)
    //进行显示设置
    pangolin::View &d_cam = pangolin::CreateDisplay()
            .SetBounds(0.0, 1.0, pangolin::Attach::Pix(175), 1.0, -1024.0f / 768.0f)
            //SetBounds函数前四个参数依次表示视图在视窗中的范围（下、上、左、右），最后一个参数是显示的长宽比
            .SetHandler(new pangolin::Handler3D(s_cam));
            //创建相机视图句柄，需要使用它来显示前面设置的相机所“拍摄”到的内容


    while (pangolin::ShouldQuit() == false) {   //ShouldQuit()：检测你是否关闭OpenGL窗口
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //清空颜色和深度缓存。这样每次都会刷新显示，不至于前后帧的颜信息相互干扰

        d_cam.Activate(s_cam);  //激活显示并设置状态矩阵
        glClearColor(1.0f, 1.0f, 1.0f, 1.0f);   //red、green、blue、alpha分别是红、绿、蓝、不透明度，值域均为[0,1]。即设置颜色，为后面的glClear做准备，默认值为（0,0,0,0）。此函数仅仅设定颜色，并不执行清除工作

        glLineWidth(2); //设置线段的宽度
        for (size_t i = 0; i < poses.size() - 1; i++) {
            glColor3f(1 - (float) i / poses.size(), 0.0f, (float) i / poses.size());    //绘图颜色：红、绿、蓝
            glBegin(GL_LINES);  //开始绘制直线
            auto p1 = poses[i], p2 = poses[i + 1];  //提取当前和之后的两个位姿poses
            glVertex3d(p1.translation()[0], p1.translation()[1], p1.translation()[2]);
            glVertex3d(p2.translation()[0], p2.translation()[1], p2.translation()[2]);  //直线连接pose[i]与pose[i+1]
            glEnd();    //停止绘图操作
        }
        pangolin::FinishFrame();    //执行后期渲染，事件处理和帧交换，相当于前面设置了那么多现在可以进行最终的显示了
        usleep(5000);   // sleep 5 ms 把调用该函数的线程挂起一段时间以便显示，单位是微秒
```

——gaosq

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening