0. 说明

本PDF文档为自动生成，如有遗漏的格式错误请及时告知！

---

1. 群的性质

群具有四种性质：封闭性、结合律、幺元、逆

问1-1：$\{\mathbb{Z},+\}$ 是否为群?若是,验证其满足群定义;若不是,说明理由。（$\mathbb{Z}$为整数集）

答： 是

- 封闭性

$$\forall a_1,a_2\in\mathbb{Z},\quad a_1+a_2\in\mathbb{Z}$$

- 结合律

$$\forall a_1,a_2,a_3\in\mathbb{Z},\quad (a_1+a_2)+a_3=a_1+(a_2+a_3)$$

- 幺元

$$\exists 0\in\mathbb{Z},\quad s.t. \forall a\in\mathbb{Z},\quad 0+a=a+0=a$$

- 逆

$$\forall a\in\mathbb{Z},\quad \exists a^{-1}\in\mathbb{Z},\quad s.t.\quad a+a^{-1}=a+(-a)=0$$

问1-2：$\{\mathbb{N},+\}$ 是否为群?若是,验证其满足群定义;若不是,说明理由。（$\mathbb{N}$为自然数集）

答： 否，不满足性质4。

$$\forall a\in\mathbb{N},\text{要使得}\,a+a^{-1}=0,\text{则}\,a^{-1}=-a,\text{但是}\,-a\notin\mathbb{N}$$

---

2. 验证向量叉乘的李代数性质

现取集合$\mathbb{V}=\mathbb{R}^3$，数域 $\mathbb{F}=\mathbb{R}$，李括号为：

$$[a,b]=a\times b$$

请验证 $g=(\mathbb{R}^3,\mathbb{R},\times)$构成李代数。

答：

- 封闭性

$$\forall X,Y\in\mathbb{R}^3,\quad [X,Y]=X\times Y\in\mathbb{R}^3$$

- 双线性

  $\forall X,Y,Z\in\mathbb{R}^3,\quad a,b\in\mathbb{F}$，有：

$$\begin{aligned}&[aX+bY,Z]=(aX+bY)\times Z=a(X\times Z)+b(Y\times Z)=a[X,Z]+b[Y,Z]\\&[Z,aX+bY]=Z\times(aX+bY)=a(Z\times X)+b(Z\times Y)=a[Z,X]+b[Z,Y]\end{aligned}$$

- 自反性

$$\forall X\in\mathbb{R}^3,\quad [X,X]=X\times X=\mathbf{0}$$

- 雅克比等价

  $\forall X, Y, Z \in \mathbb{R}^3$

  $$[X,[Y,Z]]+[Z,[X,Y]]+[Y,[Z,X]] = X\times(Y\times Z)+Z\times(X\times Y)+Y\times(Z\times X)$$
  $$=(X\cdot Z)Y-(X\cdot Y)Z+(Z\cdot Y)X-(Z\cdot X)Y+(Y\cdot X)Z-(Y\cdot Z)X=\mathbf{0}$$

---

3. 推导 SE(3) 的指数映射

设$\xi=[\rho,\phi]^T$,有：

$$\xi^\wedge=\begin{bmatrix}\phi^\wedge & \rho\\ \mathbf{0}^T & 0\end{bmatrix}$$

$exp(\xi^\wedge)=\sum_{n=0}^{\infty}\frac{1}{n!}(\xi^\wedge)^n=\sum_{n=0}^{\infty}\frac{1}{n!}\begin{bmatrix}\phi^\wedge & \rho\\ \mathbf{0}^T & 0\end{bmatrix}^n$

$=I+\sum_{n=1}^{\infty}\frac{1}{n!}\begin{bmatrix}\phi^\wedge & \rho\\ \mathbf{0}^T & 0\end{bmatrix}^n$

$=\begin{bmatrix}1+\sum_{n=1}^{\infty}\frac{1}{n!}(\phi^\wedge)^n & \sum_{n=1}^{\infty}\frac{1}{n!}(\phi^\wedge)^{n-1}\rho\\ \mathbf{0}^T & 1\end{bmatrix}=\begin{bmatrix}\sum_{n=0}^{\infty}\frac{1}{n!}(\phi^\wedge)^n & \sum_{n=1}^{\infty}\frac{1}{n!}(\phi^\wedge)^{n-1}\rho\\ \mathbf{0}^T & 1\end{bmatrix}$

$=\begin{bmatrix}\sum_{n=0}^{\infty}\frac{1}{n!}(\phi^\wedge)^n & \sum_{n=0}^{\infty}\frac{1}{(n+1)!}(\phi^\wedge)^{n}\rho\\ \mathbf{0}^T & 1\end{bmatrix}$

令$\rho=\theta\mathbf{a}$ ($\mathbf{a}$为单位向量)

$\sum_{0}^{\infty}\frac{1}{(n+1)!}(\phi^\wedge)^n=\sum_{0}^{\infty}\frac{1}{(n+1)!}(\theta\mathbf{a}^\wedge)^n$

$=I+\frac{1}{2!}\theta(\mathbf{a}^\wedge)+\frac{1}{2!}\theta^2(\mathbf{a}^\wedge)^2+\frac{1}{4!}\theta^3(\mathbf{a}^\wedge)^3+...+\frac{1}{(n+1)!}\theta^n(\mathbf{a}^\wedge)^n$

$=I+(\frac{1}{2!}\theta-\frac{1}{4!}\theta^3...)(\mathbf{a}^\wedge)+(\frac{1}{3!}\theta^3-\frac{1}{5!}\theta^5...)(\mathbf{a}^\wedge)^2 \qquad (\text{式}\,3-1)$

$\because cos\theta=1-\frac{\theta^2}{2!}+\frac{\theta^4}{4!}...$

$\Rightarrow 1-cos\theta=\frac{\theta^2}{2!}-\frac{\theta^4}{4!}...$

所以（式3-1）$=aa^T-a^\wedge a^\wedge+\frac{1-cos\theta}{\theta}a^\wedge+(\frac{1}{3!}\theta^3-\frac{1}{5!}\theta^5...)(\mathbf{a}^\wedge)^2$

$=aa^T+\frac{1-cos\theta}{\theta}a^\wedge+\frac{1}{\theta}(-\theta+\frac{1}{3!}\theta^3-\frac{1}{5!}\theta^5...)(\mathbf{a}^\wedge)^2$

$=aa^T+\frac{1-cos\theta}{\theta}a^\wedge-\frac{sin\theta}{\theta}\mathbf{a}^{\wedge 2}$

$=aa^T+\frac{1-cos\theta}{\theta}a^\wedge-\frac{sin\theta}{\theta}(aa^T-I)$

$=\frac{sin\theta}{\theta}I+(1-\frac{sin\theta}{\theta})aa^T+\frac{1-cos\theta}{\theta}a^\wedge\triangleq J$

综上可得：

$exp(\xi^\wedge)\triangleq\begin{bmatrix}\sum_{n=0}^{\infty}\frac{1}{n!}(\phi^\wedge)^n & J\rho\\ \mathbf{0}^T & 1\end{bmatrix}$

---

4. 伴随

证4-1：对于SO(3)，有：$R\cdot exp(p^\wedge)\cdot R^T=exp((Rp)^\wedge)$

根据泰勒展开可知：

$R \cdot exp(p^\wedge) \cdot R^T = R\sum_0^\infty \frac{(p^\wedge)^n}{n!}R^T = \sum_0^\infty \frac{(Rp^\wedge R^T)^n}{n!} = exp(Rp^\wedge R^T)$

所以要证明$R \cdot exp(p^\wedge) \cdot R^T = exp((Rp)^\wedge)$ 其实就是证明：

$Rp^\wedge R^T = (Rp)^\wedge$

$\Leftrightarrow Rp^\wedge = (Rp)^\wedge R \Leftrightarrow \forall u \in \mathbb{R}^3, Rp^\wedge u = (Rp)^\wedge Ru$

$\Leftrightarrow \forall u \in \mathbb{R}^3, R(p \times u) = (Rp) \times (Ru)$

根据向量叉乘的旋转变换不变性，即$\forall v, u \in \mathbb{R}^3$，$(Rv) \times (Ru) = R(v \times u)$ 可证上式成立。

几何角度来理解：两个向量v,u，其叉乘得到一个与v,u两者垂直的三维向量(v x u)，将这三个向量都经过同一个旋转，它们的相对位姿和模长都不会改变。

证4-2：对于SE(3)，有：$T\ exp(\xi^\wedge)T^{-1} = exp((Ad(T)\xi)^\wedge)$

$$Ad(T) = \begin{bmatrix} R & t^\wedge R \\ \mathbf{0} & R \end{bmatrix}$$

这里不做严格证明，只简单介绍以下证明思路：

- 首先证明$T\ exp(\xi^\wedge)T^{-1} = exp(T\xi^\wedge T^{-1})$ ；
- 然后证明$T\xi^\wedge T^{-1} = (Ad(T)\xi)^\wedge$

  $T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$，$T^{-1} = \begin{bmatrix} R^T & -R^Tt \\ 0 & 1 \end{bmatrix}$，$\xi = [\rho, \phi]^T$，$\xi^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ \mathbf{0}^T & 0 \end{bmatrix}$

  $T\xi^\wedge T^{-1} = \begin{bmatrix} R\phi^\wedge R^T & -R\phi^\wedge R^Tt + R\rho \\ \mathbf{0}^T & 0 \end{bmatrix}$

  其中：$R\phi^\wedge R^T = (R\phi)^\wedge$ ，$(R\phi)^\wedge t = -t^\wedge R\phi$

  所以：

$$T\xi^\wedge T^{-1} = \begin{bmatrix} (R\phi)^\wedge & t^\wedge R\phi + R\rho \\ \mathbf{0}^T & 0 \end{bmatrix}$$

$$(Ad(T)\xi)^\wedge = (\begin{bmatrix} R & t^\wedge R \\ \mathbf{0} & R \end{bmatrix} \begin{bmatrix} \rho \\ \phi \end{bmatrix}) = \begin{bmatrix} R\rho + t^\wedge R\phi \\ R\phi \end{bmatrix}^\wedge = \begin{bmatrix} (R\phi)^\wedge & t^\wedge R\phi + R\rho \\ \mathbf{0}^T & 0 \end{bmatrix}$$

---

5. 轨迹的描述

问5-1：T_{WC} 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 T_{WC} 的平移部分就得到了机器人的轨迹。

答：T_{WC} 的平移部分主要表示旋转过后的机器人坐标系原点在世界坐标系下的位置，而机器人坐标系就表示机器人的位姿，从而 T_{WC} 的平移部分可以表示机器人的位置变化轨迹。

问5-2：编写程序画出机器人轨迹

- 安装Sophus

这里安装非模板类的Sophus库，模板类的Sophus库要求Eigen 版本>=3.3.0，而我电脑上的版本是3.2.92。

```
git clone https://github.com/strasdat/Sophus.git --recursive
cd Sophus
git checkout a621ff
mkdir build
cd build
cmake ..
make install
```

- draw_trajectory.cpp

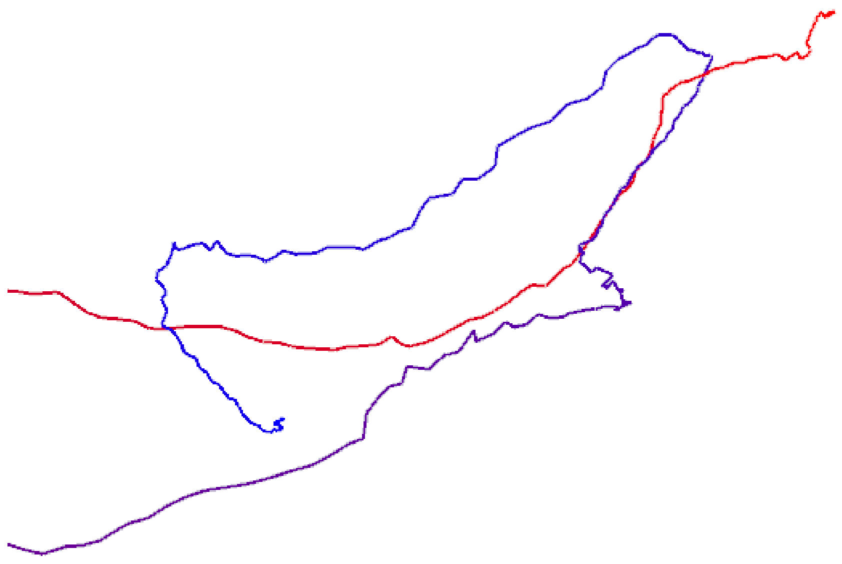
```
#include <sophus/se3.h>
#include <string>
#include <iostream>
#include <fstream>

// need pangolin for plotting trajectory
#include <pangolin/pangolin.h>

using namespace std;

// path to trajectory file
string trajectory_file = "./trajectory.txt";

// function for plotting trajectory, don't edit this code
// start point is red and end point is blue
void DrawTrajectory(vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>>);

int main(int argc, char **argv) {

    vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>> poses;

    /// implement pose reading code
    // start your code here (5~10 lines)
    ifstream trajectory(trajectory_file);

    if(!trajectory){
        cout<<"open file fail!"<<endl;
        return -1;
    }

    string oneline;
    double timestamp;
    Eigen::Vector3d t;
    Eigen::Quaterniond q;
    Sophus::SE3 T;

    while(getline(trajectory,oneline)){
        istringstream temp(oneline);
        temp >>timestamp>>t[0]>>t[1]>>t[2]>>q.x()>>q.y()>>q.z()>>q.w();
        T = Sophus::SE3(q.normalized(),t);
        poses.push_back(T);
    }
    // end your code here

    // draw trajectory in pangolin
    DrawTrajectory(poses);
    return 0;
}

/*******************************************************************************
****************/
void DrawTrajectory(vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses) {
    if (poses.empty()) {
        cerr << "Trajectory is empty!" << endl;
        return;
    }

    // create pangolin window and plot the trajectory
    pangolin::CreateWindowAndBind("Trajectory Viewer", 1024, 768);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    pangolin::OpenGlRenderState s_cam(
            pangolin::ProjectionMatrix(1024, 768, 500, 500, 512, 389, 0.1,
1000),
            pangolin::ModelViewLookAt(0, -0.1, -1.8, 0, 0, 0, 0.0, -1.0,
0.0)
    );

    pangolin::View &d_cam = pangolin::CreateDisplay()
            .SetBounds(0.0, 1.0, pangolin::Attach::Pix(175), 1.0, -1024.0f /
768.0f)
            .SetHandler(new pangolin::Handler3D(s_cam));


    while (pangolin::ShouldQuit() == false) {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        d_cam.Activate(s_cam);
        glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

        glLineWidth(2);
        for (size_t i = 0; i < poses.size() - 1; i++) {
            glColor3f(1 - (float) i / poses.size(), 0.0f, (float) i /
poses.size());
            glBegin(GL_LINES);
            auto p1 = poses[i], p2 = poses[i + 1];
            glVertex3d(p1.translation()[0], p1.translation()[1],
p1.translation()[2]);
            glVertex3d(p2.translation()[0], p2.translation()[1],
p2.translation()[2]);
            glEnd();
        }
        pangolin::FinishFrame();
        usleep(5000);   // sleep 5 ms
    }
```

```
}
```

- CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
SET(CMAKE_BUILD_TYPE "Release")
PROJECT (Chapter3)

add_compile_options(-std=c++11)

INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/include)
INCLUDE_DIRECTORIES("/usr/include/eigens3")

find_package(Pangolin REQUIRED)
INCLUDE_DIRECTORIES(${Pangolin_INCLUDE_DIRS})

find_package(Sophus REQUIRED)
include_directories(${Sophus_INCLUDE_DIRS})

SET(SRC_LIST ${PROJECT_SOURCE_DIR}/src/draw_trajectory.cpp)
ADD_EXECUTABLE(draw_trajectory ${SRC_LIST})
target_link_libraries(draw_trajectory ${Sophus_LIBRARIES}
${Pangolin_LIBRARIES})
```

- 运行结果



6. 轨迹的误差

- 代码

```
#include <sophus/se3.h>
#include <string>
#include <iostream>
#include <fstream>

// need pangolin for plotting trajectory
#include <pangolin/pangolin.h>

using namespace std;

// path to trajectory file
string groundtruth_file = "./groundtruth.txt";
string estimated_file = "./estimated.txt";

// start point is red and end point is blue
void DrawTrajectory(vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_g,
                    vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_e);

// read data
vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>>
ReadData(string FileName);

// compute error
double ComputerError(vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_g,
                     vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_e);


int main(int argc, char **argv){

    vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>> poses_g;
    vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>> poses_e;

    poses_g=ReadData(groundtruth_file);
    poses_e=ReadData(estimated_file);

    double trajectory_error_RMSE = ComputerError(poses_g, poses_e);
    cout<<"Trajectory_error_RMSE : "<<trajectory_error_RMSE<<endl;

    DrawTrajectory(poses_g, poses_e);
    return 0;
}

vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>>
ReadData(string FileName){
    vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>> poses;
    ifstream trajectory(FileName);

    if(!trajectory){
        cout<<"open file fail!"<<endl;
    }

    string oneline;
    double timestamp;
    Eigen::Vector3d t;
    Eigen::Quaterniond q;
    Sophus::SE3 T;

    while(getline(trajectory,oneline)){
        istringstream temp(oneline);
        temp >>timestamp>>t[0]>>t[1]>>t[2]>>q.x()>>q.y()>>q.z()>>q.w();
        T = Sophus::SE3(q.normalized(),t);
        poses.push_back(T);
    }
    return poses;
}


double ComputerError(vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_g,
                     vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_e){

    Eigen::Matrix<double,6,1> T;
    double errors = 0;
    double RMSE = 0;
    int n = poses_g.size();

    for(int i =0; i< n ; i++){
        T = (poses_g[i].inverse()*poses_e[i]).log();
        errors += T.transpose()*T;
    }

    RMSE = sqrt(errors/n);
    return RMSE;
}

void DrawTrajectory(vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_g,
                    vector<Sophus::SE3,
Eigen::aligned_allocator<Sophus::SE3>> poses_e){
    if (poses_g.empty()||poses_e.empty()) {
        cerr << "Trajectory is empty!" << endl;
        return;
    }

    // create pangolin window and plot the trajectory
    pangolin::CreateWindowAndBind("Trajectory Viewer", 1024, 768);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    pangolin::OpenGlRenderState s_cam(
            pangolin::ProjectionMatrix(1024, 768, 500, 500, 512, 389, 0.1,
1000),
            pangolin::ModelViewLookAt(0, -0.1, -1.8, 0, 0, 0, 0.0, -1.0,
0.0)
    );

    pangolin::View &d_cam = pangolin::CreateDisplay()
            .SetBounds(0.0, 1.0, pangolin::Attach::Pix(175), 1.0, -1024.0f /
768.0f)
```

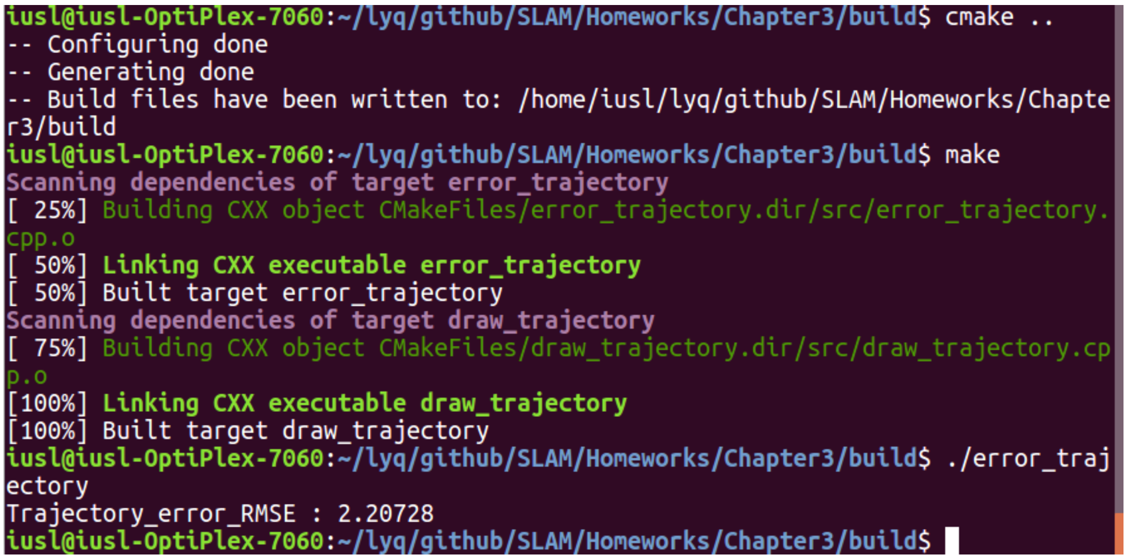
```
            .SetHandler(new pangolin::Handler3D(s_cam));

    while (pangolin::ShouldQuit() == false) {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        d_cam.Activate(s_cam);
        glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

        glLineWidth(2);
        for (size_t i = 0; i < poses_g.size() - 1; i++) {
            glColor3f(1 - (float) i / poses_g.size(), 0.0f, (float) i /
poses_g.size());
            glBegin(GL_LINES);
            auto p1 = poses_g[i], p2 = poses_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 < poses_e.size() - 1; i++) {
            glColor3f(1 - (float) i / poses_e.size(), 0.0f, (float) i /
poses_e.size());
            glBegin(GL_LINES);
            auto p1 = poses_e[i], p2 = poses_e[i + 1];
            glVertex3d(p1.translation()[0], p1.translation()[1],
p1.translation()[2]);
            glVertex3d(p2.translation()[0], p2.translation()[1],
p2.translation()[2]);
            glEnd();
        }
        pangolin::FinishFrame();
        usleep(5000);   // sleep 5 ms
    }

}
```

- CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
SET(CMAKE_BUILD_TYPE "Release")
PROJECT (Chapter3)

add_compile_options(-std=c++11)


INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/include)
INCLUDE_DIRECTORIES("/usr/include/eigens3")

find_package(Pangolin REQUIRED)
INCLUDE_DIRECTORIES(${Pangolin_INCLUDE_DIRS})
```

```
find_package(Sophus REQUIRED)
#set(Sophus_LIBRARIES /usr/local/lib/libSophus.so libSophus.so)
include_directories(${Sophus_INCLUDE_DIRS})


SET(SRC_LIST ${PROJECT_SOURCE_DIR}/src/draw_trajectory.cpp)
ADD_EXECUTABLE(draw_trajectory ${SRC_LIST})
target_link_libraries(draw_trajectory ${Sophus_LIBRARIES}
${Pangolin_LIBRARIES})


ADD_EXECUTABLE(error_trajectory
${PROJECT_SOURCE_DIR}/src/error_trajectory.cpp)
target_link_libraries(error_trajectory ${Sophus_LIBRARIES}
${Pangolin_LIBRARIES})
```

- 运行结果



```
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter3/build$ cmake ..
-- Configuring done
-- Generating done
-- Build files have been written to: /home/iusl/lyq/github/SLAM/Homeworks/Chapte
r3/build
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter3/build$ make
Scanning dependencies of target error_trajectory
[ 25%] Building CXX object CMakeFiles/error_trajectory.dir/src/error_trajectory.
cpp.o
[ 50%] Linking CXX executable error_trajectory
[ 50%] Built target error_trajectory
Scanning dependencies of target draw_trajectory
[ 75%] Building CXX object CMakeFiles/draw_trajectory.dir/src/draw_trajectory.cp
p.o
[100%] Linking CXX executable draw_trajectory
[100%] Built target draw_trajectory
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter3/build$ ./error_traj
ectory
Trajectory_error_RMSE : 2.20728
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter3/build$
```