第七章作业提示

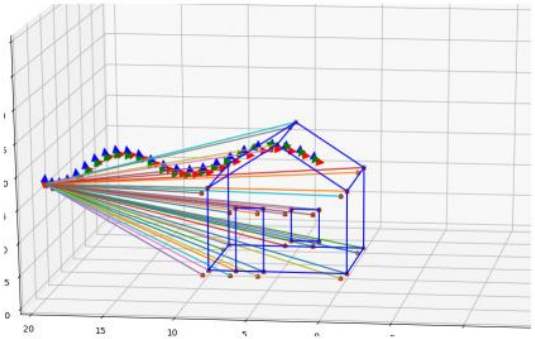
主讲人 会打篮球的猫

第一题

深蓝学院 shenlanxueyuan.com

作业

1. 将第二讲的仿真数据集（视觉特征，imu 数据）接入我们的 VINS 代码，并运行出轨迹结果。
   - 仿真数据集无噪声
   - 仿真数据集有噪声（不同噪声设定时，需要配置 vins 中 imu noise 大小。）



1、编译；

2、熟悉流程，配置接口；

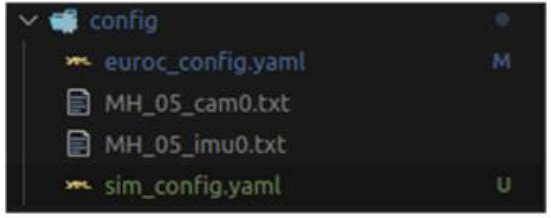
3、测试。

第一题

2.1、首先仿照 EuRoC 数据集代码，在 test 下新建了 run_sim.cpp 文件来跑仿真数据，并在 CMakeLists 中添加。


```
add_executable(run_sim test/run_sim.cpp)
target_link_libraries(run_sim
  MyVio
  -lpthread)
```

同时，仿照 config 下的 euroc_config.yaml 创建了 sim_config.yaml，并修改文件中的相机参数、外参、IMU 噪声。



第一题

深蓝学院 shenlanxueyuan.com

2.2.1、首先是 PubImuData 函数，如下图所示：

```
void PubImuData()
{
    // string sImu_data_file = sData_path + "imu_pose.txt";
    string sImu_data_file = sData_path + "imu_pose_noise.txt";
    cout << "1 PubImuData start sImu_data_filea: " << sImu_data_file << endl;
    ifstream fsImu;
    fsImu.open(sImu_data_file.c_str());
    if (!fsImu.is_open())
    {
        cerr << "Failed to open imu file! " << sImu_data_file << endl;
        return;
    }

    std::string sImu_line;
    double dStampNSec = 0.0;
    Vector3d vAcc;
    Vector3d vGyr;
    double qp[7];  // not used
    while (std::getline(fsImu, sImu_line) && !sImu_line.empty()) // read imu data
    {
        // timestamp (1), imu quaternion(4), imu position(3), imu gyro(3), imu acc(3)
        std::istringstream ssImuData(sImu_line);
        ssImuData >> dStampNSec;
        for (size_t i = 0; i < 7; ++i) {
            ssImuData >> qp[i];
        }
        ssImuData >> vGyr.x() >> vGyr.y() >> vGyr.z() >> vAcc.x() >> vAcc.y() >> vAcc.z();
        // cout << "Imu t: " << fixed << dStampNSec << " gyr: " << vGyr.transpose() << " ac
        pSystem->PubImuData(dStampNSec, vGyr, vAcc);
        usleep(5000*nDelayTimes);
    }
    fsImu.close();
}
```

时间戳(1)、q(4)、p(3)、gyro(3)、acc(3)

我们不需要 q 和 p 这 7 维，因此我保存在了一个临时数组中。

第一题

深蓝学院 shenlanxueyuan.com

2.2.2、其次是 PubImageData 函数，如下图所示：

```
while (std::getline(fsImage, sImage_line) && !sImage_line.empty())
{
    std::istringstream ssImuData(sImage_line);
    ssImuData >> dStampNSec;
    // points file
    string sPoints_file = sData_path + "keyframe/all_points_" + to_string(m) + ".txt";
    m++;
    ifstream fsPoints;
    fsPoints.open(sPoints_file.c_str());
    if (!fsPoints.is_open())
    {
        cerr << "Failed to open points file! " << sPoints_file << endl;
        return;
    }

    std::string sPoints_line;
    std::vector<cv::Point2f> featurePoints;
    double pt3d[4];  // not used
    while (std::getline(fsPoints, sPoints_line) && !sPoints_line.empty()) {
        // x, y, z, 1, u, v
        std::istringstream ssPointData(sPoints_line);
        for (size_t i = 0; i < 4; ++i) {
            ssPointData >> pt3d[i];
        }
        cv::Point2f pt2d;
        ssPointData >> pt2d.x >> pt2d.y;  // uv(归一化平面)
        featurePoints.emplace_back(pt2d);
    }

    pSystem->PubFeatureData(dStampNSec, featurePoints);

    usleep(50000*nDelayTimes);
}
```

对于每一帧图像，首先读取时间戳，再读取该帧对应的全部特征点信息。

参考所提供的特征点数据格式：

x、y、z、1、u、v

我们需要保存的是u和v，并将其保存至vector中。

最后，相比于处理image，我们直接拿到了观测信息，因此需要创建新接口来处理。

第一题

深蓝学院
shenlanxueyuan.com

```
204     // FIXME:
205     if (PUB_THIS_FRAME)
206     {
207         pub_count++;
208         shared_ptr<IMG_MSG> feature_points(new IMG_MSG());
209         feature_points->header = dStampSec;
210         vector<set<int>> hash_ids(NUM_OF_CAM);
211         for (int i = 0; i < NUM_OF_CAM; i++)
212         {
213             for (unsigned int j = 0; j < featurePoints.size(); j++)
214             {
215                 int p_id = j;
216                 hash_ids[i].insert(p_id);
217                 double x = featurePoints[j].x;
218                 double y = featurePoints[j].y;
219                 double z = 1;
220                 feature_points->points.push_back(Vector3d(x, y, z));
221                 feature_points->id_of_point.push_back(p_id * NUM_OF_CAM + i);
222
223                 float u,v;
224                 u = 460 * x + 255;
225                 v = 460 * y + 255;
226                 feature_points->u_of_point.push_back(u);
227                 feature_points->v_of_point.push_back(v);
228                 feature_points->velocity_x_of_point.push_back(0);   // 姑且设为 0
229                 feature_points->velocity_y_of_point.push_back(0);
230             }
```

此时，我们已经有了每一帧的特征点集合，不需要再对图像提取和追踪特征点，因此直接遍历，并输入信息即可。

此外，我们前面拿到的u和v是并不是像素坐标，而是归一化坐标，因此需要进行简单的投影变换。

第一题

深蓝学院
shenlanxueyuan.com

2.2.4、System.cpp 中的数据保存

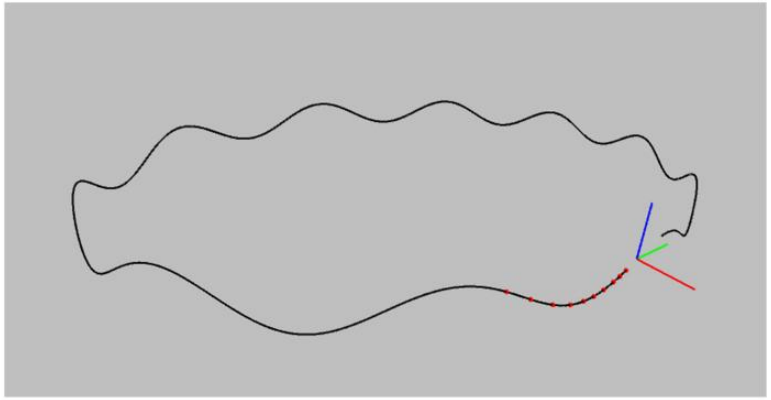
为了方便后续使用 evo 工具对 VINS 输出结果与真值进行更好的对比，期望保存成 TUM 格式的数据。而默认的保存方式因为空格问题导致格式不对，因此修改了保存数据这里的代码：

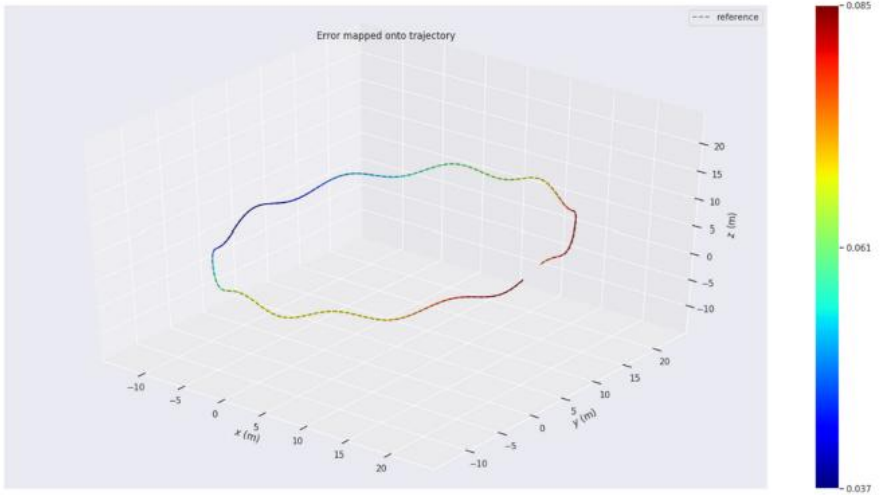
```
425                 // ofs_pose << fixed << dStamp << " " << p_wi
426                 ofs_pose.precision(18);
427                 ofs_pose <<dStamp<<" ";
428                 ofs_pose.precision(10);
429                 ofs_pose <<p_wi(0)<<" "
430                     <<p_wi(1)<<" "
431                     <<p_wi(2)<<" "
432                     <<q_wi.x()<<" "
433                     <<q_wi.y()<<" "
434                     <<q_wi.z()<<" "
435                     <<q_wi.w() <<std::endl;
436             }
```

第一题

深蓝学院
shenlanxueyuan.com

无噪声





可以看出 VIO 输出的轨迹与真值几乎是完全贴合。

```
APE w.r.t. translation part (m)
(with SE(3) Umeyama alignment)

       max      0.084906
      mean      0.063872
    median      0.066940
       min      0.037071
      rmse      0.065462
       sse      2.515458
       std      0.014339
```

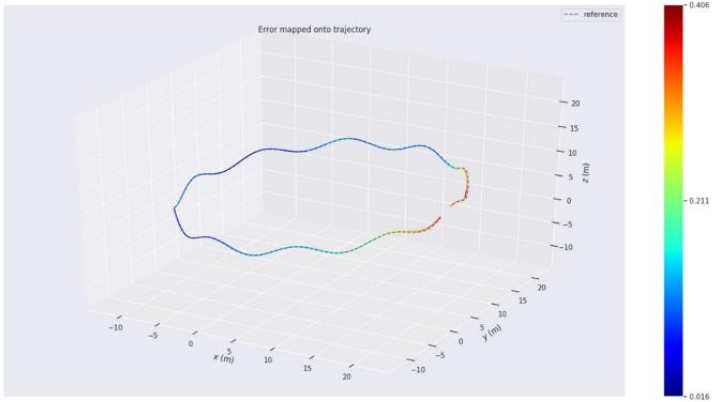
第一题

深蓝学院 shenlanxueyuan.com

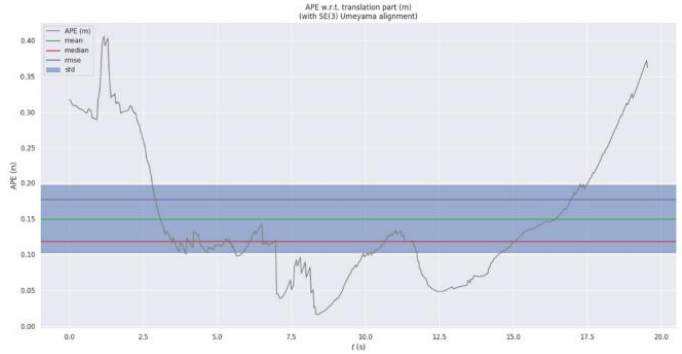
小噪声

修改生成数据的噪声项，并使用带噪声的 IMU 数据（imu_pose_noise.txt）。同时将该噪声配置到 VINS 中。

```
// noise
double gyro_bias_sigma = 1.0e-6;
double acc_bias_sigma = 0.00001;

double gyro_noise_sigma = 0.0015;
double acc_noise_sigma = 0.0019;
```



可以看出，加入噪声项以后，轨迹和真值之间出现了明显的不一致，误差也变大了。



APE w.r.t. translation part (m) (with SE(3) Umeyama alignment)	
max	0.406453
mean	0.150086
median	0.118658
min	0.016449
rmse	0.177015
sse	18.393166
std	0.093853

上两张图进一步展示了加入噪声的 IMU 数据的测试结果，相比于没有噪声，rmse 有了大幅提高，误差变大。

第一题

深蓝学院 shenlanxueyuan.com

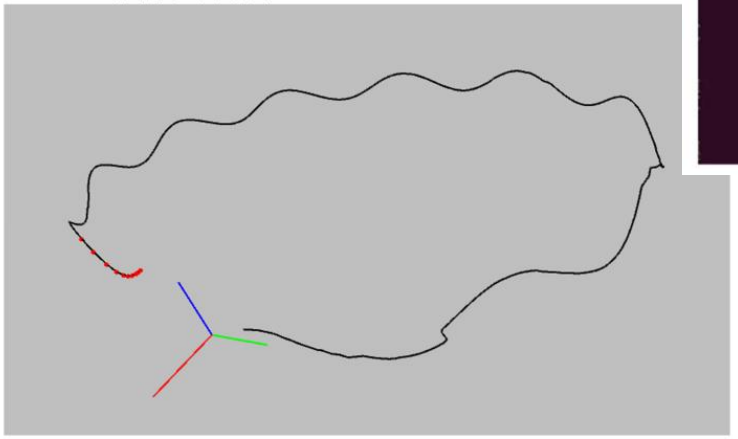
大噪声

3.4、仿真数据集的噪声进一步加大

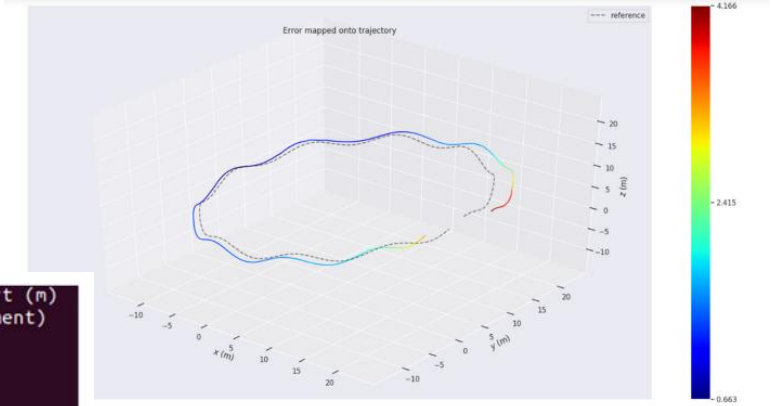
继续加大 IMU 噪声项（数值上是一开始的 10 倍），如下图所示：

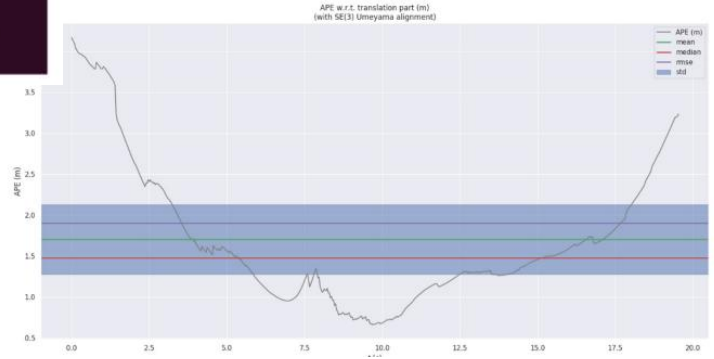
```
// noise
double gyro_bias_sigma = 1.0e-5;
double acc_bias_sigma = 0.0001;

double gyro_noise_sigma = 0.015;
double acc_noise_sigma = 0.019;
```

运行结果如下图所示：

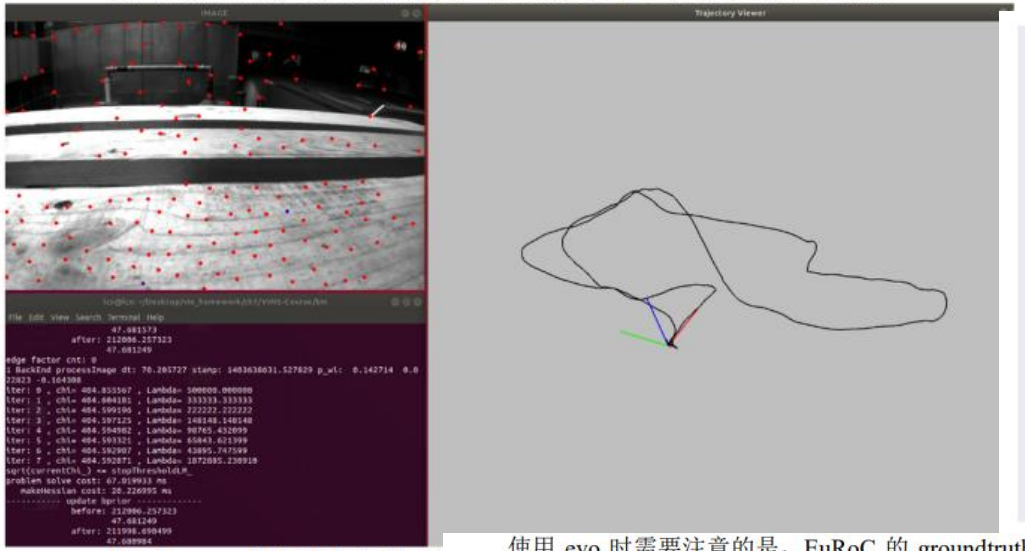


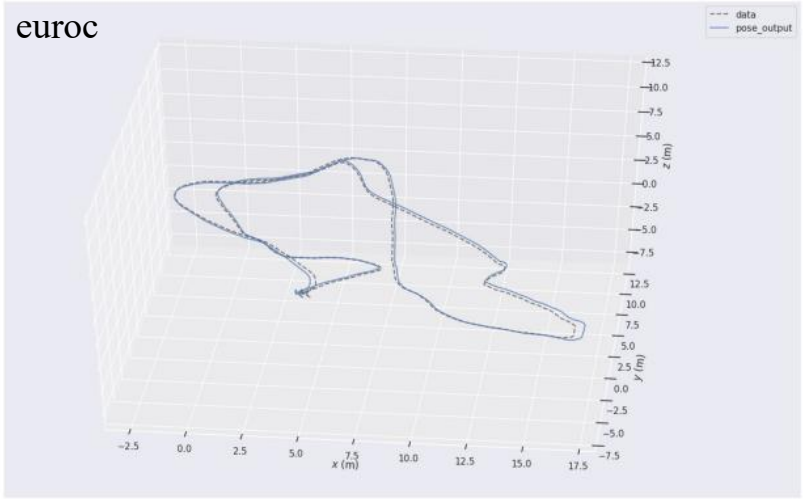
```
APE w.r.t. translation part (m)
(with SE(3) Umeyama alignment)

       max      4.166487
      mean      1.706450
    median      1.478792
       min      0.662635
      rmse      1.905695
       sse      2131.793219
       std      0.848354
```





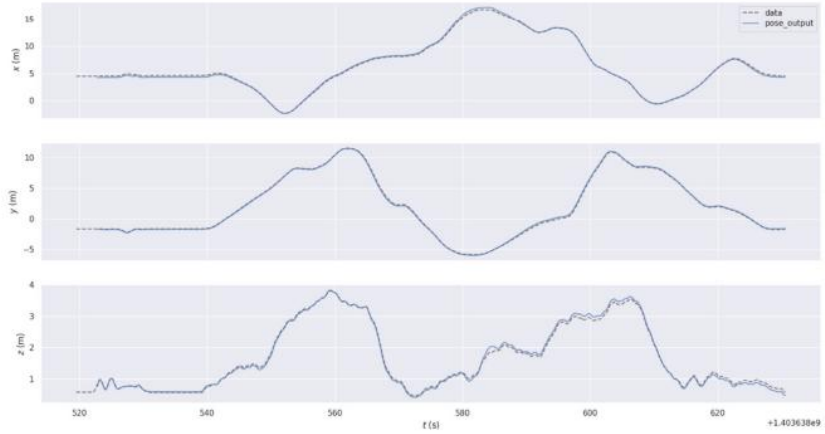
第一题

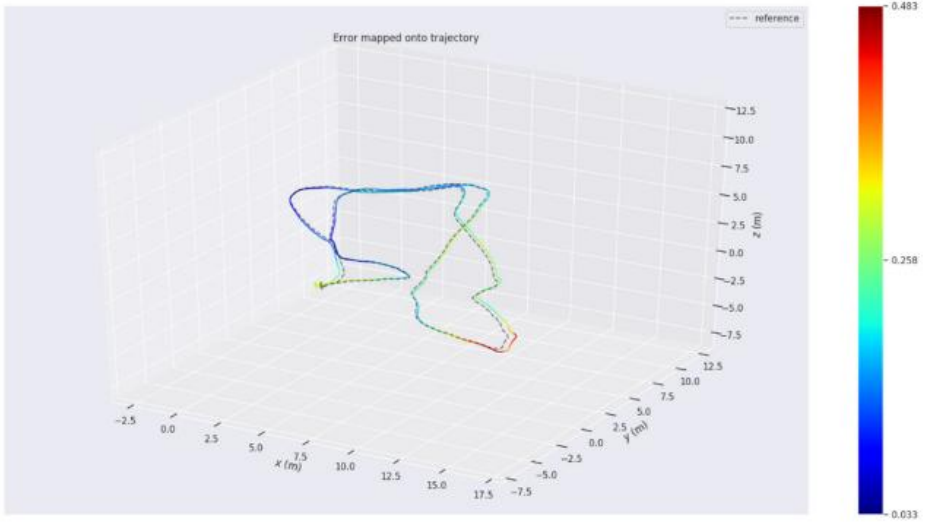




使用 evo 时需要注意的是，EuRoC 的 groundtruth 格式并不是 TUM 格式，但是 evo 工具提供了两者直接的转换功能，因此将 gt 转化成了 TUM 格式，如下图所示：

第一题





与真值相比，总体上看的 VIO 的估计结果还是比较准确的，轨迹较为贴合，误差不大。

第一题

深蓝学院
shenlanxueyuan.com

个人遇到的编译问题：

主要问题出在编译环境，花费了好多额外时间。如：①gflags 库在源码编译完以后一定要将其改为动态库链接（或使用 apt 方式安装），否则会出现链接不到的问题；②opencv 特别早安装的，其中 gtk 版本过低导致了运行出问题，需要安装新版并重新编译 opencv；③最新的 glog 编译需要 cmake 版本 3.16 以上；④最新的 ceres 需要 C++ 14 等等……

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening