VIO 第七章作业

曾卓

07 2021

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。）

1.1 代码修改

主要是修改 run_euroc.cpp 文件里的 PubImageData()，和 PubImuData() 以及 system.cpp 里的 PubImageData() 函数

```
void PubImuData()
{
    // change the path
    // string sImu_data_file = sData_path + "imu_pose.txt";
    string sImu_data_file = "../data/imu_pose_noise.txt";
    // string sImu_data_file = "/mnt/hgfs/bwedu/VIO/exercise/ex02/course2_hw_new/vio_data_simulation/bin/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;
    while (std::getline(fsImu, sImu_line) && !sImu_line.empty()) // read imu data
    {
        std::istringstream ssImuData(sImu_line);
        // imu data format: timestamp (1) , imu quaternion(4) , imu position(3) , imu gyro(3) , imu acc(3)
        // we only need timestamp(1), gyro(3), acc(3)
        ssImuData >> dStampNSec;
        double ignored; //ignore quarenion(4),position(3)
        for (size_t i = 0; i < 7; ++i)
            ssImuData >> ignored;
        ssImuData >> vGyr.x() >> vGyr.y() >> vGyr.z() >> vAcc.x() >> vAcc.y() >> vAcc.z();
        // cout << "Imu t: " << fixed << dStampNSec << " gyr: " << vGyr.transpose() << " acc: " << vAcc.transpose() << endl;
        pSystem->PubImuData(dStampNSec, vGyr, vAcc);
        usleep(5000 * nDelayTimes);
    }
    fsImu.close();
}
```

此部分改动较小，注意 imu 格式和 camera 格式是一样的，我们需要的只是最后的 gyro 和 acc 故将中间的 7 个元素略去

run_euroc.cpp 中 PubImageData 的修改

```
void PubImageData()
{
    // open cam_pose.txt only for the timestamps
    string sImage_file = "../data/cam_pose.txt";
    // string sImage_file = "/mnt/hgfs/bwedu/VIO/exercise/ex02/course2_hw_new/vio_data_simulation/bin/cam_pose.txt";
    cout << "1 PubImageData start sImage_file: " << sImage_file << endl;
    ifstream fsImage;
    fsImage.open(sImage_file.c_str());
    if (!fsImage.is_open())
    {
        cerr << "Failed to open image file! " << sImage_file << endl;
        return;
    }
    std::string sImage_line;
    double dStampNSec;
    string sImgFileName;
    int n = 0; // read all_points at each dStampNSec
```

从带时间戳的 cam.txt 文件中读取时间，传入 dSampNSec，然后从 all_points 文件中读取特征点在归一化平面上的坐标。

```
    while (std::getline(fsImage, sImage_line) && !sImage_line.empty())
    {
        std::istringstream ssImgData(sImage_line);
        // only read the timestamp
        ssImgData >> dStampNSec;
        cout << "cam time: " << fixed << dStampNSec << endl;
        string all_points_file_name = "../data/keyframe/all_points_" + to_string(n) + ".txt";
        // string filePath = "/mnt/hgfs/bwedu/VIO/exercise/ex02/course2_hw_new/vio_data_simulation/bin/";
        // string all_points_file_name = filePath + "keyframe/all_points_" + to_string(n) + ".txt";
        cout << "points_file: " << all_points_file_name << endl;

        vector<Point2f> FeaturePoints;
        ifstream fsImgPts;
        fsImgPts.open(all_points_file_name);
        if (!fsImgPts.is_open())
        {
            cerr << "Failed to open image file! " << all_points_file_name << endl;
        }

        string sImgPts_line;

        // read feature points from all_points.txt
        while (getline(fsImgPts, sImgPts_line) && !sImgPts_line.empty())
        {
            istringstream ssImgPoint(sImgPts_line);
            // file format in each line: x, y, z, 1, u, v
            // we only need u, v
            double ignored;
            for (size_t i = 0; i < 4; ++i)
            {
                ssImgPoint >> ignored;
            }
            Point2f feature_point;
            ssImgPoint >> feature_point.x;
            ssImgPoint >> feature_point.y;

            FeaturePoints.push_back(feature_point);

        }

        pSystem->PubImageData(dStampNSec, FeaturePoints);

        usleep(50000 * nDelayTimes);
        n++;
    }
    fsImage.close();
```

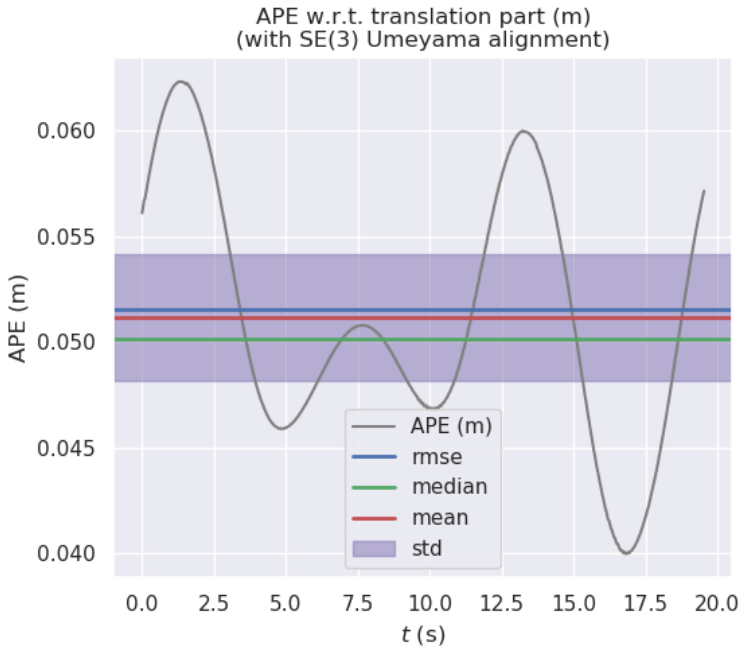
接下来重载了 system::PubImageData 的函数，直接接收由 run_euroc 中 PubImageData 里得到的由 Point2f 组成的特征点队列，而不是之前的 Mat.

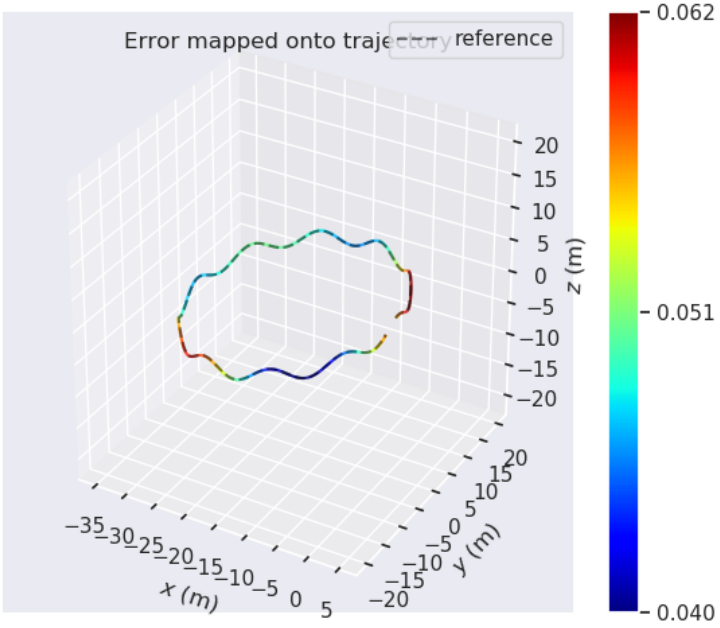
```cpp
/* overloaded function used for the simulated IMU data in ex02
 * parameters:
 *  double dStampSec: time stamp
 *  vector<Point2f> &point_list: all the feature points
 */

void System::PubImageData(double dStampSec, const vector<Point2f> &point_list)
{

    if (!init_feature)
    {
        cout << "1 PubImageData skip the first detected feature, which doesn't contain optical flow speed" << endl;
        init_feature = 1;
        return;
    }

    if (first_image_flag)
    {
        cout << "2 PubImageData first_image_flag" << endl;
        first_image_flag = false;
        first_image_time = dStampSec;
        last_image_time = dStampSec;
        return;
    }
    // detect unstable camera stream
    if (dStampSec - last_image_time > 1.0 || dStampSec < last_image_time)
    {
        cerr << "3 PubImageData image discontinue! reset the feature tracker!" << endl;
        first_image_flag = true;
        last_image_time = 0;
        pub_count = 1;
        return;
    }
    last_image_time = dStampSec;
```

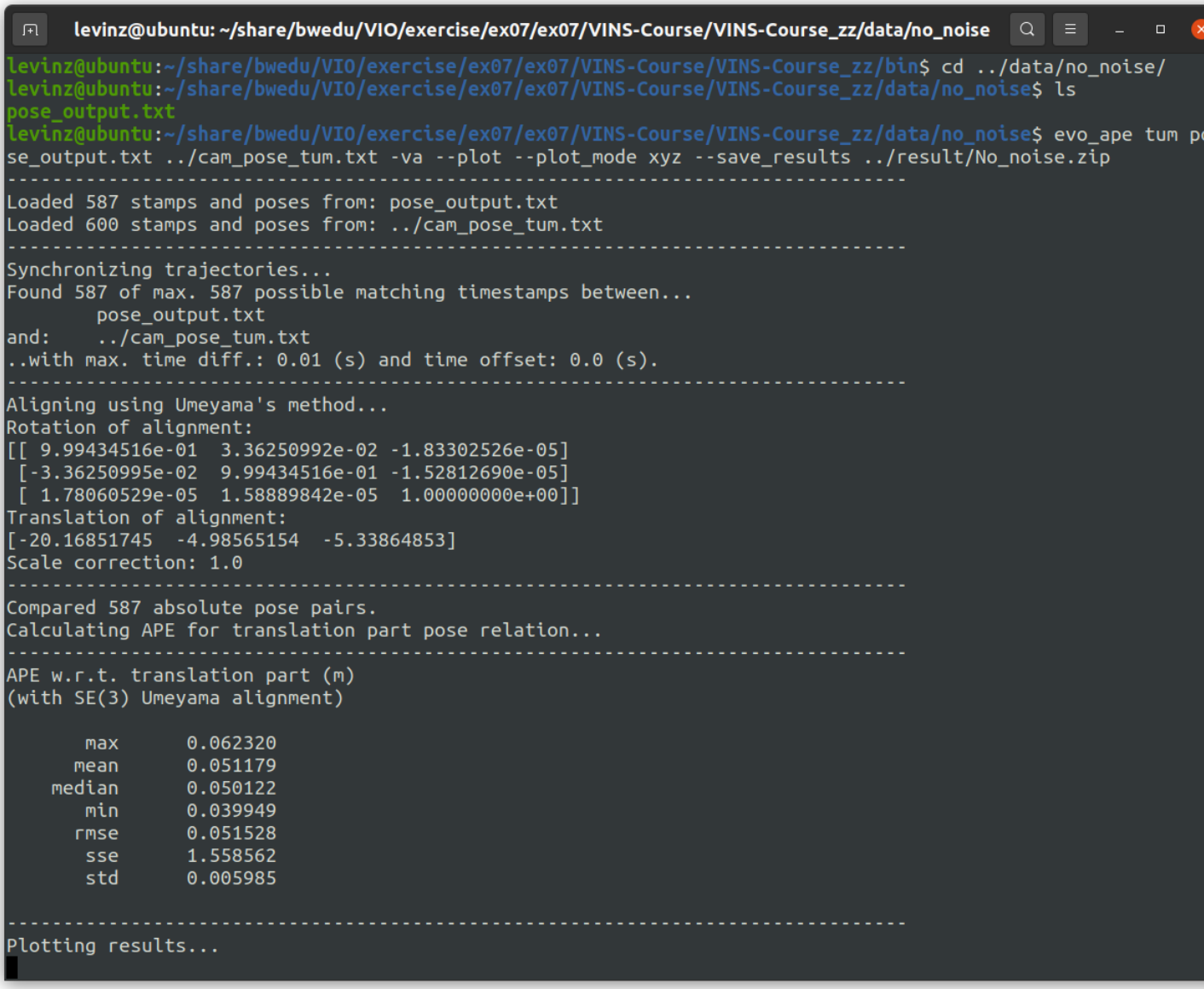
```cpp
    ///------------------------------changed here ----------------------------------//
    PUB_THIS_FRAME = true;
    TicToc t_r;

    if (PUB_THIS_FRAME)
    {
        ++pub_count;
        shared_ptr<IMG_MSG> feature_points(new IMG_MSG());
        feature_points->header = dStampSec;
        vector<set<int>> hash_ids(NUM_OF_CAM);
        for (size_t i = 0; i < NUM_OF_CAM; i++)
        {
            for (size_t j = 0; j < point_list.size(); j++)
            {
                int p_id = j;
                hash_ids[i].insert(p_id);
                double x = point_list[j].x;
                double y = point_list[j].y;
                // feature_points on homogeneous plane
                feature_points->points.push_back(Vector3d(x, y, 1));
                feature_points->id_of_point.push_back(p_id * NUM_OF_CAM + i);
                feature_points->u_of_point.push_back(0);
                feature_points->v_of_point.push_back(0);
                feature_points->velocity_x_of_point.push_back(0);
                feature_points->velocity_y_of_point.push_back(0);
            }
        }

        ///------------------------------unchanged ----------------------------------//
```

1.2 仿真结果

1.2.1 无噪声的情况下



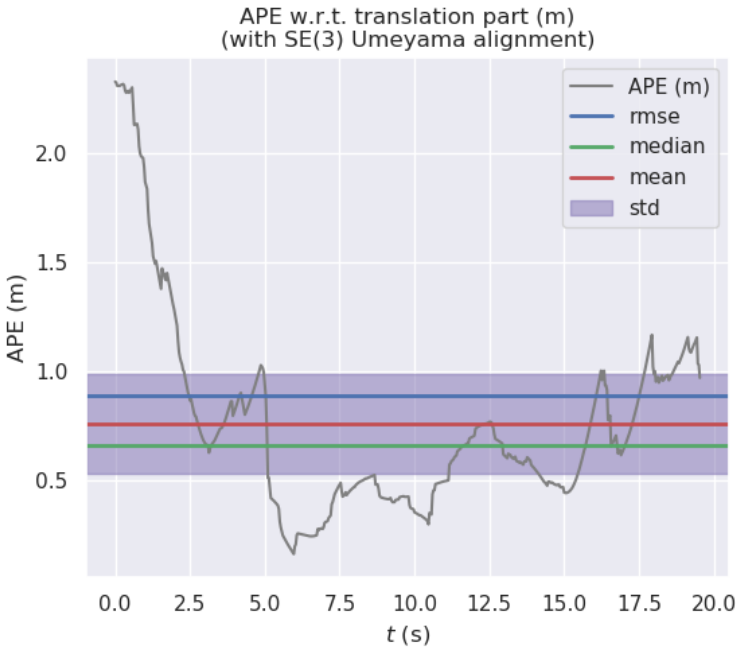


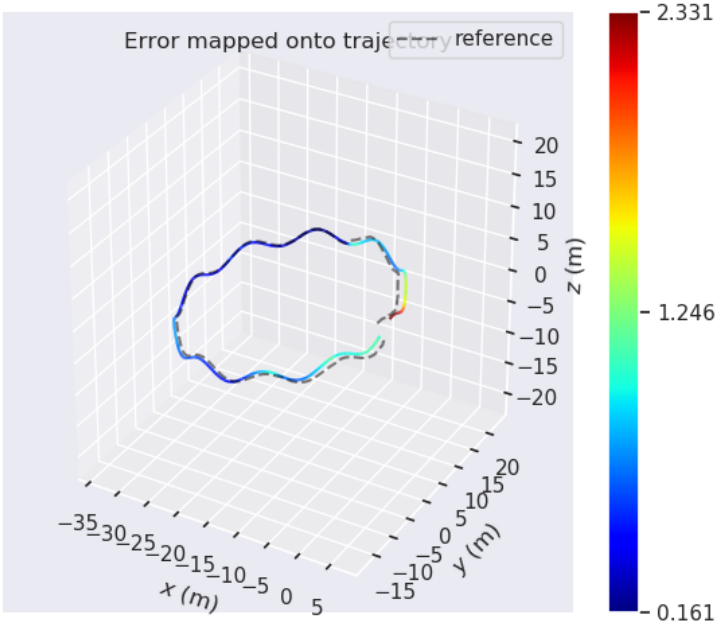


1.2.2 有小噪声的情况下

噪声参数

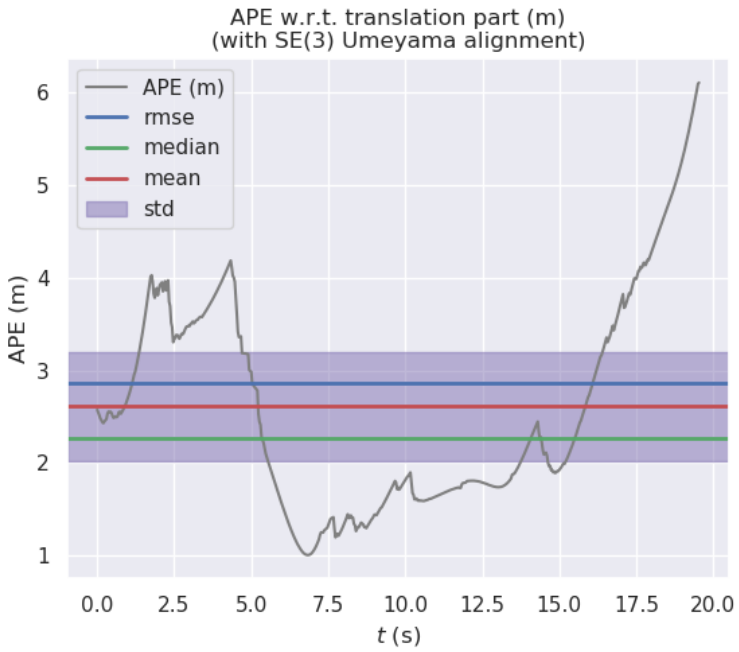
```
double multiple = 1;
double gyro_bias_sigma = 1.0e-5 * multiple;
double acc_bias_sigma = 0.0001 * multiple;

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

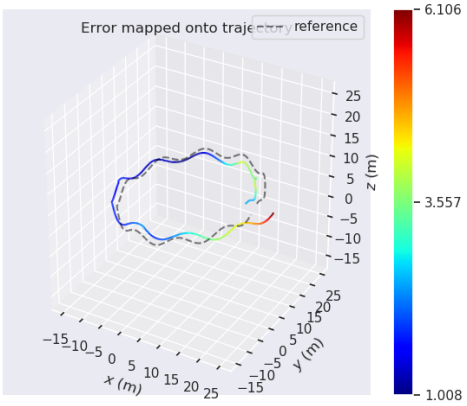





```
APE w.r.t. translation part (m)
(with SE(3) Umeyama alignment)

       max      2.331128
      mean      0.760675
    median      0.660745
       min      0.160673
      rmse      0.886574
       sse      461.389388
       std      0.455397
```

1.2.3 有较大噪声的情况下






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

       max      6.105680
      mean      2.613591
    median      2.262476
       min      1.007710
      rmse      2.864082
       sse      4815.140183
       std      1.171370
```