Jindong_Shi
2022-02-11

第 16 节课习题

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。

新建源文件 test/run_vio_sim.cpp, 修改该源文件中的函数如下:

首先修改 void PubImuData():

```
void PubImuData()
{
    // string sImu_data_file = sData_path + "imu_pose.txt";
    string sImu_data_file = sData_path + "imu_pose_noise.txt";
```

.....

```
    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( str: sImu_line);

        ssImuData >> dStampNSec;
        double temp;
        for(int i = 0; i < 7; ++i)
        {
            ssImuData >> temp;
        }
        ssImuData >> vGyr.x() >> vGyr.y() >> vGyr.z() >> vAcc.x() >> vAcc.y() >> vAcc.z();
        pSystem->PubImuData( dStampSec: dStampNSec, vGyr, vAcc);

        usleep( useconds: 5000*nDelayTimes);
    }
```

然后修改 void PubImageData():

```
void PubImageData()
{
    string sImage_file = sData_path + "cam_pose.txt";
```

.....

```
std::string sImage_line;
double dStampNSec;

int k = 0;
while (std::getline( & fsImage,  & sImage_line) && !sImage_line.empty())
{
    std::istringstream ssImuData( str: sImage_line);
        ssImuData >> dStampNSec;
    // cout << "Image t : " << fixed << dStampNSec << " Name: " << sImgFileName << endl;
    string imagePath = sData_path + "keyframe/all_points_" + std::to_string( val: k) + ".txt";
    ++k;

    std::ifstream fFP( s: imagePath);
    if (!fFP.is_open())
    {
        cerr << "Failed to open image file! " << imagePath << endl;
        return;
    }
```

```
    string sFP_line;
    std::vector<cv::Point2f> FeaturePoints;
    while(std::getline( &: fFP,  &: sFP_line) && !sFP_line.empty())
    {
        std::stringstream ss;
        ss << sFP_line;
        double temp;
        for(int i = 0; i < 4; ++i)
        {
            ss >> temp;
        }
        float px, py;
        ss >> px >> py;
        cv::Point2f pt( x: px,  y: py);
        FeaturePoints.push_back(pt);
    }
    pSystem->PubImageData( dStampSec: dStampNSec, FeaturePoints);
}
```

文件 cam_pose.txt 记录相机在每一个时间戳的位置，我们这里只需要获取它的时间戳信息。对应每一个时间点，在文件夹 keyframe 下都会存储有一个文件 all_point_k.txt，其中 k 代表时间戳从 0 开始的 id，该文件内记录了每个时间点获取的每一帧图像上所有特征点的归一化平面坐标。我们创建容器 std::vector<cv::Point2f> FeaturePoints，用来存储每一帧图像上所有特征点的归一化平面坐标。

我们需要重载 System 类内的函数 void System::PubImageData()，使其可以接收 std::vector<cv::Point2f> 类型的参数。我们将特征点的归一化平面坐标和对应的像素平面坐标保存到 feature_points 内。

```
// Overload the function "PubImageData" for "run_vio_sim"
void System::PubImageData(double dStampSec, const std::vector<cv::Point2f> &FeaturePoints)
{
```

……

```
for (int i = 0; i < NUM_OF_CAM; i++)
{
    auto &un_pts :vector<Point_<float>>& = trackerData[i].cur_un_pts;
    auto &cur_pts :vector<Point_<float>>& = trackerData[i].cur_pts;
    auto &ids :vector<int>& = trackerData[i].ids;
    auto &pts_velocity :vector<Point_<float>>& = trackerData[i].pts_velocity;
    for (unsigned int j = 0; j < FeaturePoints.size(); j++)
    {
        // if (trackerData[i].track_cnt[j] > 1)
        {
            int p_id = j;
            hash_ids[i].insert( x: p_id);
            double x = FeaturePoints[j].x;
            double y = FeaturePoints[j].y;
            double z = 1;
            feature_points->points.push_back(Vector3d(x, y, z));
            feature_points->id_of_point.push_back(p_id * NUM_OF_CAM + i);
            feature_points->u_of_point.push_back(460 * x + 255);
            feature_points->v_of_point.push_back(460 * x + 255);
            feature_points->velocity_x_of_point.push_back(0);
            feature_points->velocity_y_of_point.push_back(0);
        }
    }
```

当我们修改仿真程序中的 imu 噪声参数时，我们也需要修改 config 文件内对应的参数:

```
extrinsicRotation: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data:  [ 0, 0, -1,
           -1, 0,  0,
            0, 1,  0]
#Translation from camera frame to imu frame, imu^T_cam
extrinsicTranslation: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [-0.05, -0.04, 0.03]
```

……

```
#imu parameters       The more accurate parameters you provide, the better performance
acc_n: 0.0019          # accelerometer measurement noise standard deviation. #0.2   0.04
gyr_n: 0.0015         # gyroscope measurement noise standard deviation.     #0.05  0.004
acc_w: 1.0e-5       # accelerometer bias random work noise standard deviation.  #0.02
gyr_w: 1.0e-6    # gyroscope bias random work noise standard deviation.     #4.0e-5
g_norm: 9.81007     # gravity magnitude
```

1) 仿真数据集无噪声

我们在 run_vio_sim.cpp 源文件内使用不带噪声的 imu 数据文件 imu_pose.txt，将 config 文件内的 imu 参数设成非常小的值，比如 1e-9;

绘制出轨迹如下,我们发现轨迹并没有跟丢:

2) 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。)

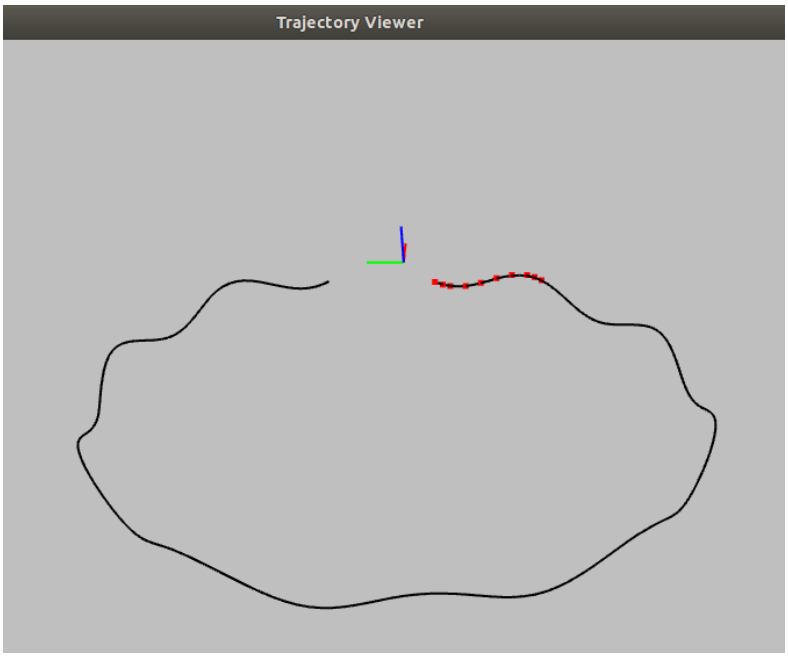
我们在 run_vio_sim.cpp 源文件内使用带噪声的 imu 数据文件 imu_pose_noise.txt。

首先我们模拟较小噪声的情况:

```
// noise
double gyro_bias_sigma = 1.0e-6;
double acc_bias_sigma = 1.0e-5;

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

绘制出的轨迹如下，与不带噪声的轨迹差不多:



然后我们模拟较大噪声的情况:

```
// noise
double gyro_bias_sigma = 1.0e-5;
double acc_bias_sigma = 1.0e-4;

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

绘制出轨迹如下，我们发现轨迹很明显估计得不准确: