深蓝学院 VIO 第七次课程作业

温焕宇

2019.8.10;2019.8.17 改

1 作业迭代修改部分

针对有噪声 IMU 数据，第一版作业中当 yaml 文件中 imu 噪声参数和仿真设置的噪声一致时，轨迹误差大的问题。

1.1 错误原因：

由于上一版的代码里发布帧的频率控制部分未改，通过 debug 发现每两帧发布一帧数据（针对仿真数据），此部分代码如下：

```
// frequency control
if (round(1.0 * pub_count / (dStampSec - first_image_time)) <= FREQ)
{
    PUB_THIS_FRAME = true;
    // reset the frequency control
    if (abs(1.0 * pub_count / (dStampSec - first_image_time) - FREQ) < 0.01 * FREQ)
    {
        first_image_time = dStampSec;
        pub_count = 0;
    }
}
else
{
    PUB_THIS_FRAME = false;
}
```

但仿真数据里面的特征点数据是连续的，应当连续发布出来，否则轨迹有偏差。但是为什么无噪声时 IMU 轨迹是正确的呢？因为 IMU 和 Camera 数据都相当于真值，故轨迹正确。当有噪声的 IMU 数据时，若还是间隔两帧才发布一帧 Camera 特征点数据，相当于 Camera 信息丢失 2/3，势必会造成轨迹的漂移，如上一版作业的结果。

1.2 解决办法：

将频率控制部分代码注释，保证都是连续的发布每一帧特征点出来。或者将上面代码 PUB-THIS-FRAME = false; 改成如下：

```
    PUB_THIS_FRAME = true;
```

1.3 修改后运行对比结果：

本次评估工具使用 uzh-rpg/rpg-trajectory-evaluation，使用参考 github 上的 README。[1]
四种情况如下表：

name	max/m	mean/m	min/m	rmse/m	轨迹总长/m
A 无噪声	0.436190	0.096559	0.020469	**0.127661**	110.179
B 有噪声	3.203756	1.340572	0.468402	**1.510936**	112.223
C 有噪声 ×3	22.513000	9.931972	0.458918	**11.429269**	126.996
D 有噪声 /3	1.152071	0.345822	0.074846	**0.401324**	112.373

【注】：有噪声数据为 acc-n: 0.019，gyr-n: 0.015，acc-w: 0.0001，gyr-w: 1.0e-5

A 无噪声时：（蓝色为估计值，绿色为真值）

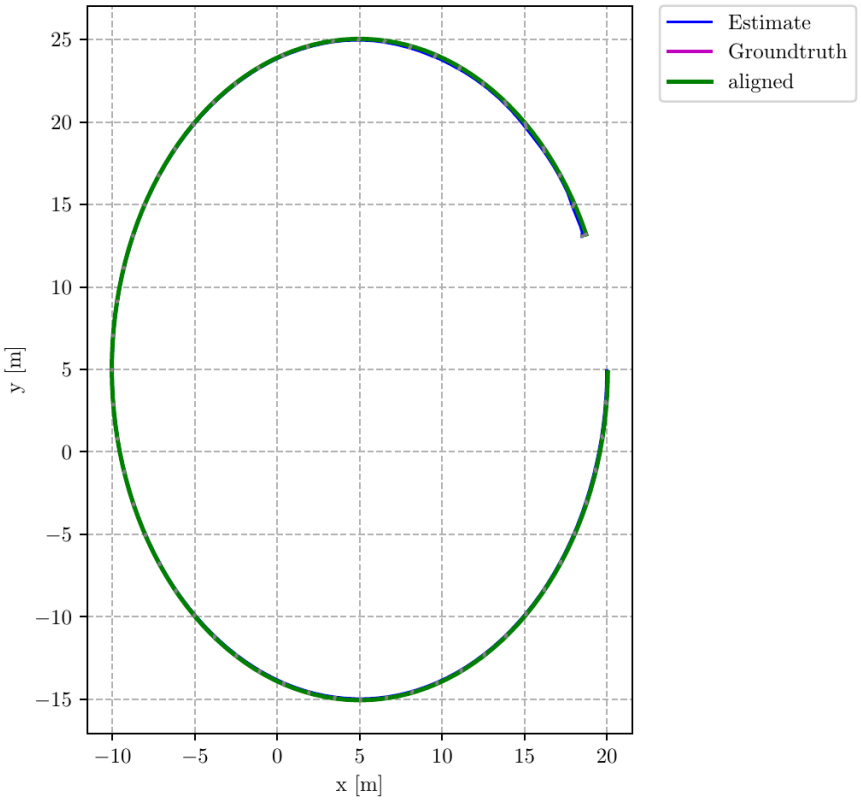


图 1: 无噪声轨迹 top-posyaw

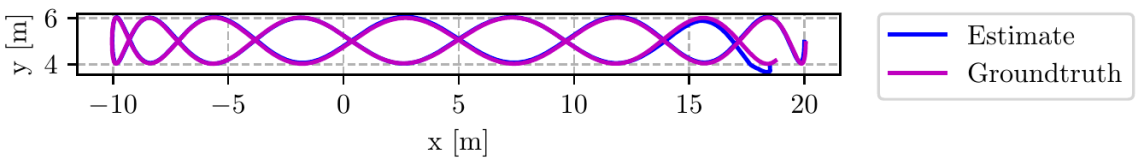


图 2: 无噪声轨迹 side-posyaw

[1] https://github.com/uzh-rpg/rpg-trajectory-evaluation。将最后两个-改为下沉-（由于 latex 编写下沉-会报错，还每找到解决办法）

B 噪声为（acc-n: 0.019，gyr-n: 0.015，acc-w: 0.0001，gyr-w: 1.0e-5）时：

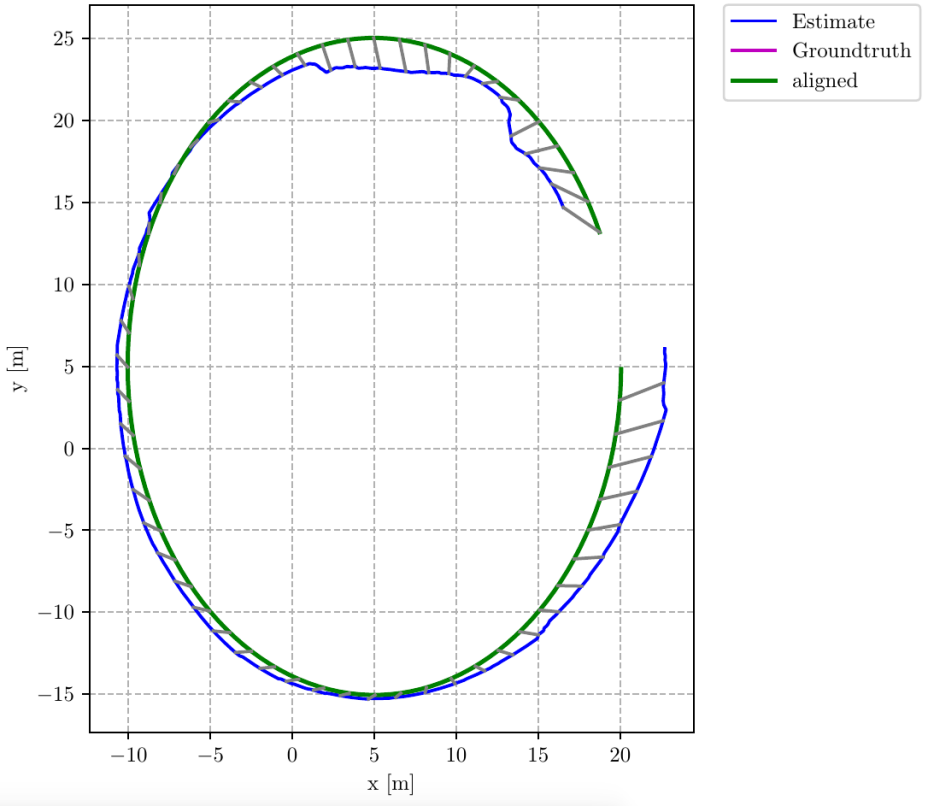


图 3: 原始噪声轨迹 top-posyaw

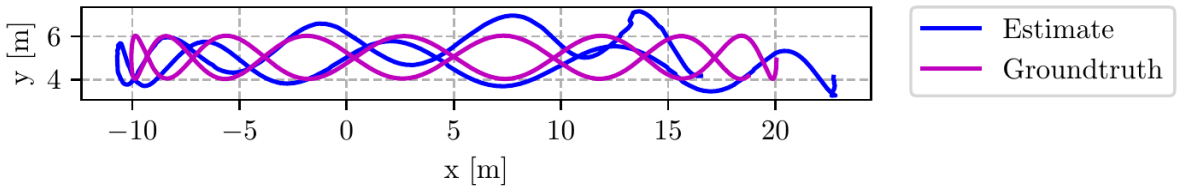


图 4: 原始噪声轨迹 side-posyaw

C 噪声扩大 3 倍为（acc-n: 0.019，gyr-n: 0.015，acc-w: 0.0001，gyr-w: 1.0e-5）*3 时：

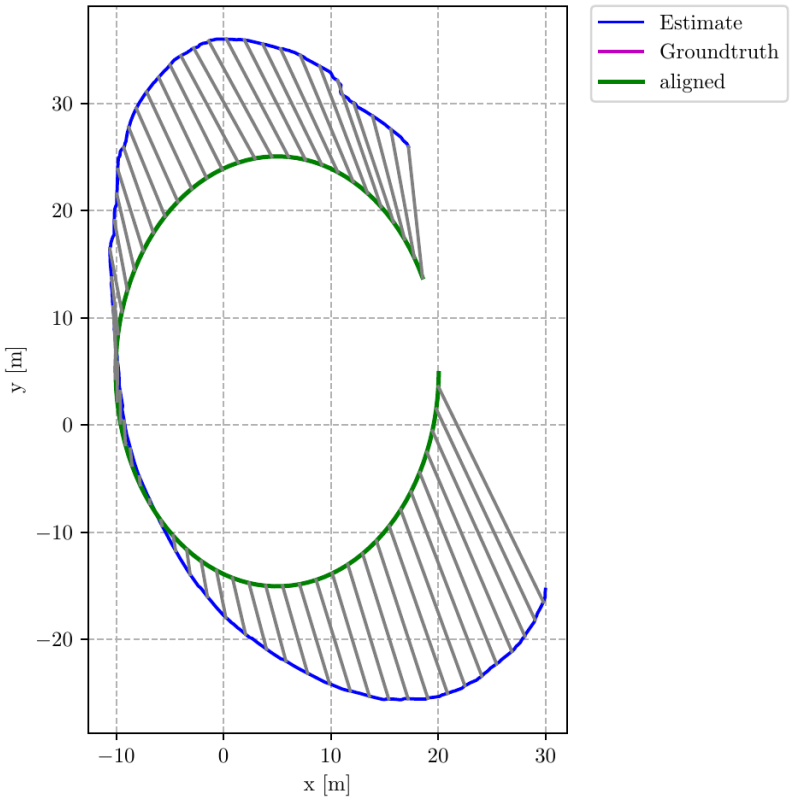


图 5: 扩大 3 倍噪声轨迹 top-posyaw

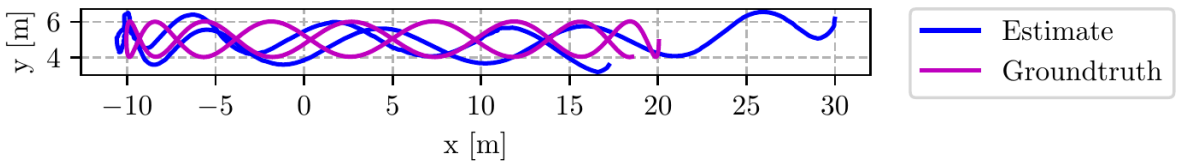


图 6: 扩大 3 倍噪声轨迹 side-posyaw

D 噪声缩小 3 倍为（acc-n: 0.0063，gyr-n: 0.005，acc-w: 0.0000333，gyr-w: 0.333e-5）*3 时：

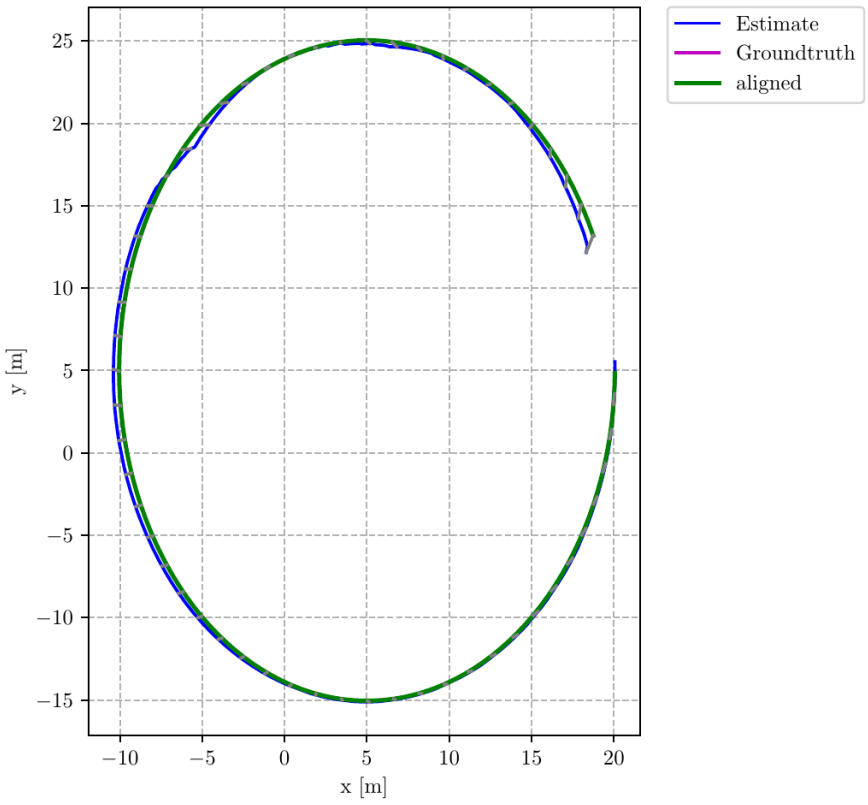


图 7: 缩小 3 倍噪声轨迹 top-posyaw

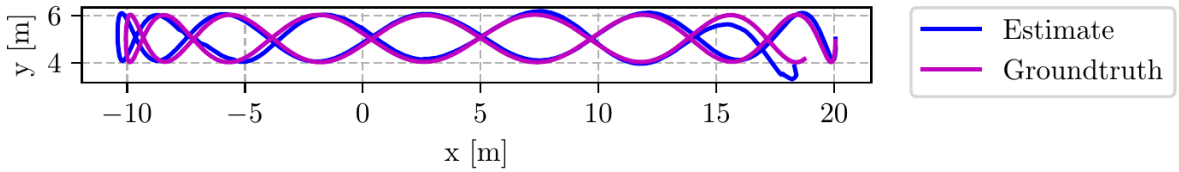
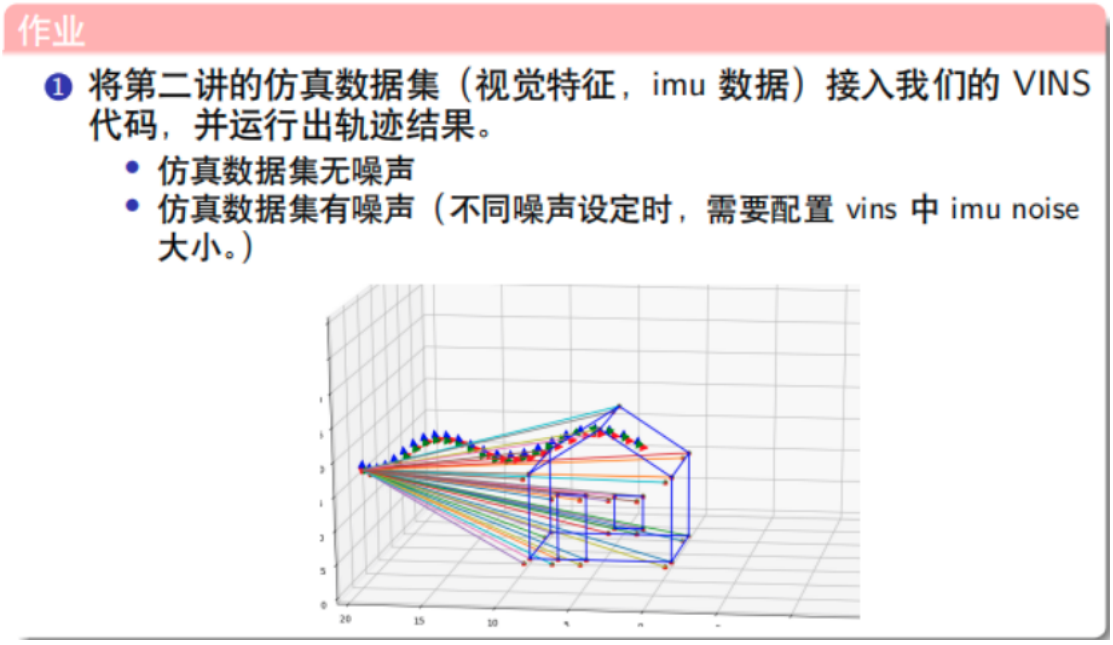


图 8: 缩小 3 倍噪声轨迹 side-posyaw

1.4 结论：

本次修改，仅修改 IMU 数据噪声，不修改 yaml 参数，yaml 参数和仿真程序噪声设置一致。发现当噪声扩大三倍时轨迹误差扩大 10 倍多，噪声缩小 3 倍时轨迹误差缩小 3 倍，由于仅分析三组有噪声数据，结论可能稍有偏差。

2 第一版作业

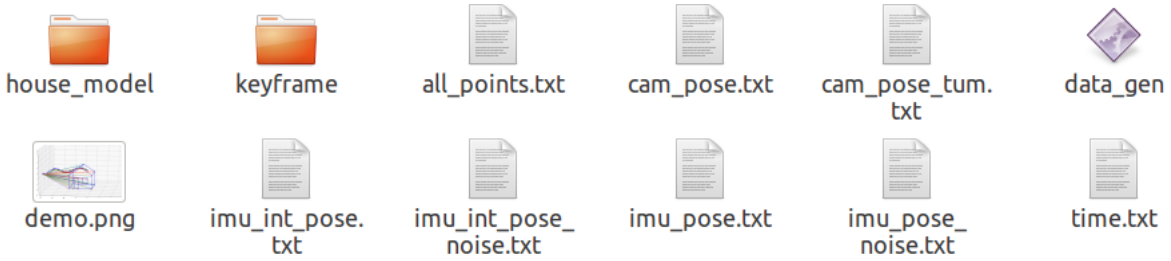


答:

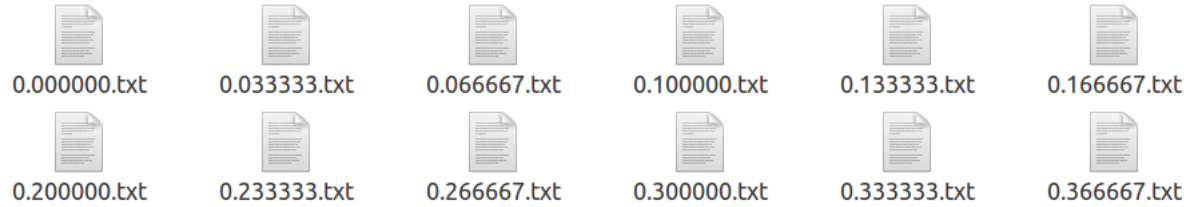
2.1 前期数据处理

a、仿真数据的生成

根据第二讲已知可以生成以下文件。



为了方便接入到 VINS-sys-code 系统里，增加特征点 (即相当于 Camera) 时间戳文件 time.txt。并且将保存特征点的文件名称改为时间戳格式。如下:



b、VINS 读取文件部分代码修改

为了方便修改调试，新建 Simulation-test.cpp 文件，主要代码如下：main 函数:

```
int main(int argc, char **argv)
{
    if(argc != 3)
    {
        cerr << "./sim_data_test 特征点文件路径 配置文件（包括参数、时间戳和imu数据）\n" << endl;
        return -1;
    }

    sData_path = argv[1];
  sConfig_path = argv[2];

    pSystem.reset(new System(sConfig_path));

    std::thread thd_BackEnd(&System::ProcessBackEnd, pSystem);
    std::thread thd_PubImuData(PubImuData);
    std::thread thd_PubPointData(PubPointData);
    std::thread thd_Draw(&System::Draw, pSystem);

    thd_PubImuData.join();
    thd_PubPointData.join();
    thd_BackEnd.join();
  thd_Draw.join();
    std::cout << "end!!!!" << std::endl;
    return 0;
}
```

主要修改 PubPointData() 函数，由于读取的是特征点数据，故将 PubPointData() 函数中的读取图片的 imread() 函数改为 readPoint() 函数，把特征点数据存在一个容器 td::vector<cv::Point2f> 中，代码如下：

```
void readPoint(const string& filename, std::vector<cv::Point2f>& _points)
{
    ifstream f;
    f.open(filename.c_str());

    if(!f.is_open())
    {
        std::cerr << "打不开角点文件 " << std::endl;
        return;
    }
    float data[36][2] = { 0 };
    for(int i=0; i<36; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            f >> data[i][j];
        }
        cv::Point2f tmp=cv::Point2f(data[i][0],data[i][1]);
        _points.push_back(tmp);
    }
}
```

c、传入特征点和 IMU 数据到系统融合优化

首先分析生成的特征点数据集，特征点数据是归一化相机平面坐标，而且顺序可以作为它的 id。然后分析 visn-sys-code 代码，IMG-MSG 存储着特征点信息，原本图片传入到 System::PubImageData 函数中，进行特征提取与跟踪部分，然后将特征点传入到 feature-points(new IMG-MSG()) 里，如下图。

```
for (int i = 0; i < NUM_OF_CAM; i++)
{
    auto &un_pts = trackerData[i].cur_un_pts; // 其实可以将仿真的点直接喂到这里来->tracjerData中的un_pts
    auto &cur_pts = trackerData[i].cur_pts;
    auto &ids = trackerData[i].ids;
    auto &pts_velocity = trackerData[i].pts_velocity;
    for (unsigned int j = 0; j < ids.size(); j++)
    {
        if (trackerData[i].track_cnt[j] > 1)
        {
            int p_id = ids[j];
            hash_ids[i].insert(p_id);
            double x = un_pts[j].x;
            double y = un_pts[j].y;
            double z = 1;
            feature_points->points.push_back(Vector3d(x, y, z));
```

故针对特征点数据集，直接跳过图像处理过程，将特征点归一化坐标传入到 feature-points 中的归一化坐标 x、y，为了方便调试修改，新建 PubPointData() 函数，注释掉 readImage() 函数，修改主要代码如下：

```
// trackerData[0].readImage(img, dStampSec); // 得到cur_pts , prev_un_pts, prev_pts
...
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(int i=0; i<NUM_OF_CAM; i++)
      {
          // auto &un_pts = trackerData[i].cur_un_pts; // 无畸变的点
          // auto &cur_pts = trackerData[i].cur_pts;
          // auto &ids = trackerData[i].ids;
          // auto &pts_velocity = trackerData[i].pts_velocity;
          for(int j=0; j<feature.size(); j++)
          {
              // if(trackerData[i].track_cnt[j] > 1)
              {
                  int p_id = j;
                  hash_ids[i].insert(p_id);
                  double x = feature[j].x;
                  double y = feature[j].y;
                  double z = 1;
                  // cout << "x is " << x << "  y is " << y << endl;
                  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(cur_pts[j].x);
                  // feature_points->v_of_point.push_back(cur_pts[j].y);
                  // feature_points->velocity_x_of_point.push_back(pts_velocity[j].x);
                  // feature_points->velocity_y_of_point.push_back(pts_velocity[j].y);
                  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);
                }
            }
```

d、修改配置文件 yaml

根据第二讲仿真代码主要修改内外参如下：

```
model_type: PINHOLE
camera_name: camera
image_width: 640
image_height: 640
distortion_parameters:
   k1: 0
   k2: 0
   p1: 0
   p2: 0
projection_parameters:
   fx: 460
   fy: 460
   cx: 255
   cy: 255

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]
```

e、最后进行编译运行

2.2 数据结果分析

2.2.1 仿真数据无噪声

将无噪声数据集 imu-pose.txt 文件输入到 vins 系统中，imu 噪声的 yaml 参数如下：

```
#imu parameters       The more accurate parameters you provide, the better performance
acc_n: 0.019          # accelerometer measurement noise standard deviation. #0.2   0.04
gyr_n: 0.015          # gyroscope measurement noise standard deviation.     #0.05  0.004
acc_w: 0.0001         # accelerometer bias random work noise standard deviation.  #0.02
gyr_w: 1.0e-5         # gyroscope bias random work noise standard deviation.     #4.0e-5
g_norm: 9.81007       # gravity magnitude
```

运行结果如图 1:

图 9: 轨迹图

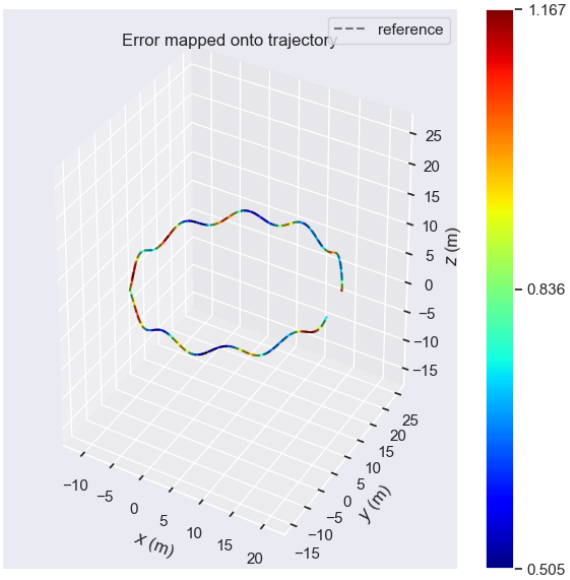
【注】由于初始化部分未画出轨迹，故在 1.233 秒之前无轨迹显示。

针对无噪声 imu 数据集进行分析，修改 yaml 配置文件中的 imu 标定噪声及 bias 参数（扩大），并分析其对结果的影响。

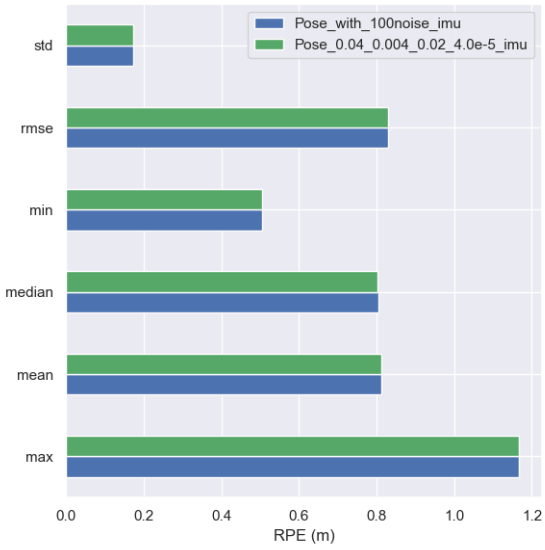
扩大 100 倍的 yaml 参数如下:

```
#imu parameters       The more accurate parameters you provide, the better performance
acc_n: 1.90           # accelerometer measurement noise standard deviation. #0.2   0.04
gyr_n: 1.50          # gyroscope measurement noise standard deviation.     #0.05  0.004
acc_w: 0.01           # accelerometer bias random work noise standard deviation.  #0.02
gyr_w: 1.0e-3         # gyroscope bias random work noise standard deviation.     #4.0e-5
g_norm: 9.81007       # gravity magnitude
```

结果如图 2:



(a) 原始 yaml 的 imu 标定参数结果与真值误差对比结果



(b) 将 imu 标定参数扩大 100 倍，两者与真值误差结果对比

图 10: 两种不同 imu 标定参数下轨迹与真值误差结果对比

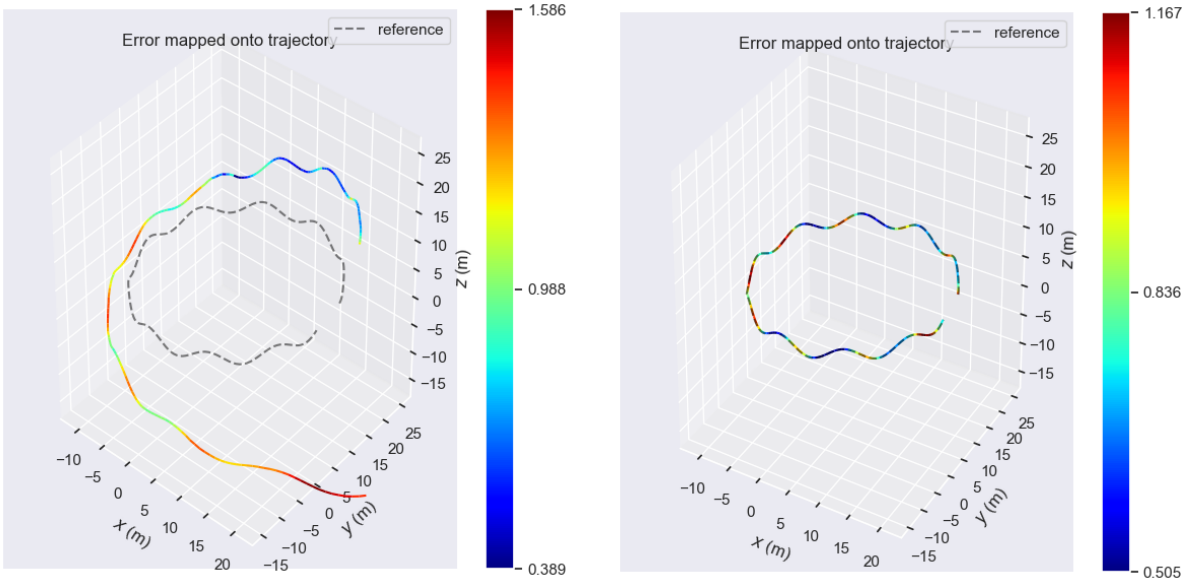
不带噪声 IMU 数据两种情况下与真值误差数值结果如下：

name	max/m	mean/m	rmse/m	sse	轨迹总长/m
原始 yaml	1.167762	0.812515	0.830518	128.985	110.266
imu 参数扩大 100 倍	1.167276	0.812294	0.830292	128.915	110.239

结果分析：针对【无噪声的 IMU 数据集】，由于 IMU 信息和视觉信息两者都接近真值，故无论怎么样修改 imu 标定噪声参数，即无论如何修改视觉和 IMU 权重，对结果影响都不是很大。

2.2.2 仿真数据有噪声（此部分误差不应该这么大，修改看（1、迭代修改））

答：针对带噪声的 IMU 数据 imu-pose-noise.txt。运行结果如图 3：



(a) 带噪声 imu 数据集，yaml 参数为仿真程序设置的参数　　(b) 将 imu 标定参数扩大 100 倍的结果轨迹

图 11: 带噪声 imu 数据集下两种不同 imu 标定参数轨迹与真值结果对比

带噪声 IMU 数据两种情况下与真值误差数值结果如下：

name	max/m	mean/m	rmse/m	sse	轨迹总长/m
原始 yaml	1.58592	1.00874	1.05132	206.685	139.993
imu 参数扩大 100 倍	1.16664	0.81244	0.830471	128.971	110.183

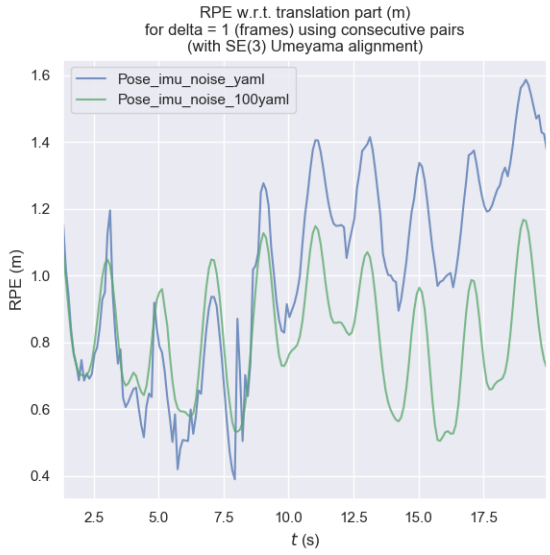


图 12: 不同 imu 标定参数结果误差对比，蓝色是 100 倍 imu 标定参数，绿色是原始标定参数

结果分析：针对【含噪声的 IMU 数据集】，原始标定参数，由于 IMU 噪声导致轨迹偏移严重，在增大 yaml 中 IMU 噪声参数情况下，噪声项的对角协方差矩阵增大，则协方差的逆信息矩阵减小，IMU 的权重变小，更加相信视觉测量，故轨迹正常。

2.3 总结

针对本次作业，对 VINS 前端代码部分更深入了解，本来计划方案是将特征点数据放入 cv::Mat 中，但是在恢复坐标或者进行光流跟踪会引入额外误差，而且会增大工作量。故修改方案将特征点数据直接传入到系统进行优化。

对于有噪声的 IMU 数据，其结果漂移严重，但存在可靠视觉信息的情况下，降低 IMU 的权重可使系统精度提高。在具体应用场景中应结合实际对系统进行优化。

【疑问】图 3(a) 的 evo 结果图显示和数值好像不太匹配，不知是何原因。