深蓝学院
shenlanxueyuan.com

标题 手写VIO第七章作业分享

主讲人 海滩游侠

纲要

深蓝学院
shenlanxueyuan.com

第一部分：题目分析

第二部分：仿真数据特点

第三部分：代码框架理解

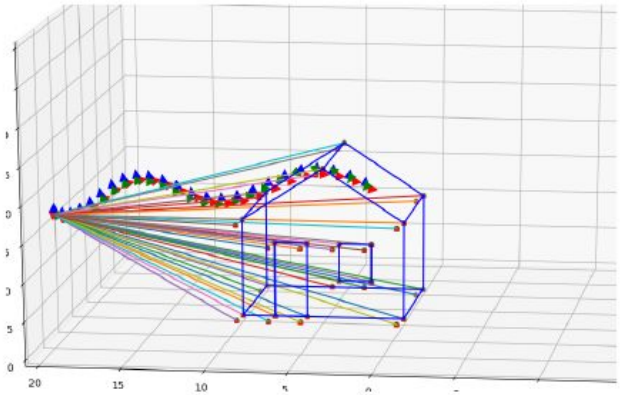
第四部分：代码细节

题目分析

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。）



图一: VIO作业要求截图

题目分析

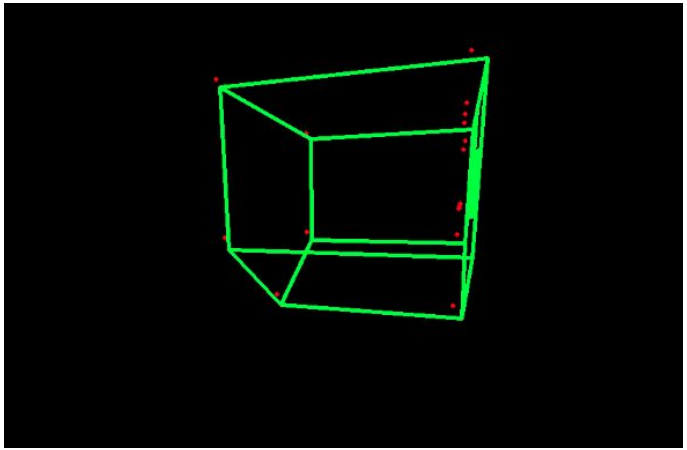
深蓝学院
shenlanxueyuan.com

作业目标:
修改项目前端接口: 从euroc数据转为仿真数据

思路:

1. 直接将仿真数据转换为图像文件,调整参数,然后用vins mono进行处理
   ==>没有采用这种方法, 因为并不有助于理解前端的 结构（放弃）

2. 理解仿真数据特征，然后根据数据性 质，删减vins mono框架中不需要的部分,让系统工作

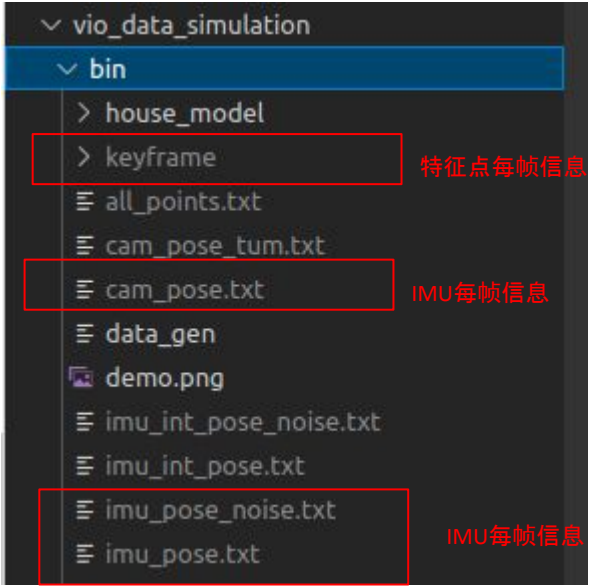


图二：仿真数据可视化效果

仿真数据特点

第二章作业中,
在vio_data_simulation中, 我们可以找到需要的文件, 其中主要包括keyframe, 和imu_pose.txt

建议大家回顾一下生成特征点的源 码文件gener_alldata.cpp,param.h等文件,因为这样可以熟悉生成数据的流程



图三：第二章主要文件

仿真数据特点

这里要注意一下生成数据的一些特征:

- 所有(36个)特征点在每一帧都会被观测
- 每一帧imu之间和每一帧图像之间的时间差都是固定的
- 得到的图像特征位于归一化平面
- 特征点始终按特定顺序排列,这样可以简化特征点ID的处理

Vins-mono 代码框架理解

深蓝学院 shenlanxueyuan.com

step 1: 看cmake, 找到执行文件代码 run_euroc.cpp

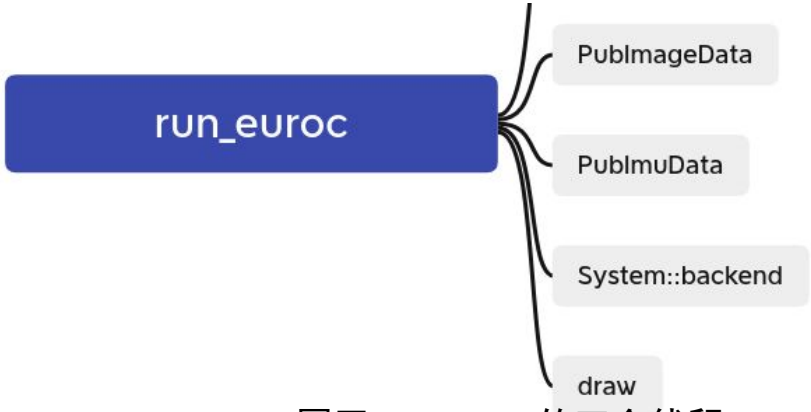
```
add_executable(run_euroc test/run_euroc.cpp)
target_link_libraries(run_euroc
  MyVio
  -lpthread)

add_executable(testCurveFitting test/CurveFitting.cpp)
target_link_libraries(testCurveFitting MyVio)
```

图四 CMakelists截图

Vins-mono 代码框架理解

step 2: 通过run_euroc.cpp理解代码逻辑

代码启动了四个线程, 对应的功能分别是: 获取图片,imu,作图以及后端



图五 run_euroc的四个线程

Vins-mono 代码框架理解

step 3: 寻找接口

在对于代码理解后,其实可以发现,我们的目标就是把image和imu信息分别放到image_buf 和imu_buf中.

注意image_buf存放的就是每一帧的特征点构成的vector

理解框架,并且寻找到接口后,我们就可以改代码了

代码细节

深蓝学院
shenlanxueyuan.com

1. 修改cmakelists.txt

   不建议修改run_euroc, 新建一个可执行文件,并用cmake关联上更合理

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

add_executable(testCurveFitting test/CurveFitting.cpp)
target_link_libraries(testCurveFitting MyVio)

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

图六: cmakelists截图

代码细节

深蓝学院 shenlanxueyuan.com

2. 修改imu读取接口

```
void ParseImuData()
{
    string sImu_data_file = sConfig_path + "imu_pose_noise.txt";
    cout << "I parse imu data from file : " << 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 StampNSec = 0.0;
    Vector3d vAcc;
    Vector3d vGyr;
    double _;//read useless data
    while (std::getline(fsImu, sImu_line) && !sImu_line.empty()) // read imu data
    {
        std::istringstream ssImuData(sImu_line);
        ssImuData >> StampNSec >>_>>_>>_>>_>>_>>_>> >>vGyr.x() >> vGyr.y() >> vGyr.z() >>vAcc.x() >> vAcc.y() >> vAcc.z();
        //cout << "Imu t: " <<StampNSec << " gyr: " << vGyr.transpose() << " acc: " << vAcc.transpose() << endl;
        //cout<<"imu pub time "<<StampNSec << endl;
        pSystem->PubImuData(StampNSec, vGyr, vAcc);
        usleep(5000*nDelayTimes);
    }
    fsImu.close();
}
```

代码细节

2. 修改图像读取接口

```
void ParseImageData()
{
    You, a day ago • update
    vector<double> ts_data = ParseImageTimestamp();
    //define a image pair
    vector<pair<double,double>> feature_pts;

    int ts_size= ts_data.size();
    cout << "In total timestamp number of imgs is  " <<ts_size << endl;


    std::string sImage_line;
    double dStampNSec;
    string sImgFileName;

    // cv::namedWindow("SOURCE IMAGE", CV_WINDOW_AUTOSIZE);
    //while (std::getline(fsImage, sImage_line) && !sImage_line.empty())
    for (int i=0;i<ts_size;i++)
    {
        dStampNSec = ts_data[i];

        std::stringstream file;
        file<<"/keyframe/all_points_"<<i<<".txt";
        string filename = file.str();
        string imagePath = sConfig_path +filename;
        //Sstd::cout<<imagePath<<std::endl;
        ifstream fsImg;
        fsImg.open(imagePath.c_str());
        if (!fsImg.is_open())
        {
            cerr << "Failed to open image file! " << imagePath << endl;
            return;
        }
        //std::string sImg_point;
        std::string sImg_point;
```

```
double _;//read useless data
int image_w = 1000;
Mat image_frame = Mat::zeros( image_w, image_w, CV_8UC3 );
    double cx= 500;
    double f = 460;
vector<cv::Point2f> feature_vec ;

while (std::getline(fsImg, sImg_point) && !sImg_point.empty()) // read imu data
{
    cv::Point2f feature_pt;
    std::istringstream ssImgData(sImg_point);
    ssImgData >>_>>_>>_>>_>>feature_pt.x >> feature_pt.y;
    feature_vec.push_back(feature_pt);
    auto center = cv::Point(feature_pt.x*f+cx,feature_pt.y*f+cx);
    cv::circle(image_frame, center,1,CV_RGB(255,0,0),3);
    //pSystem->PubImuData(StampNSec, vGyr, vAcc);
    //usleep(5000*nDelayTimes);
    //cout<<center.x<<" "<<center.y<<", ";
}
fsImg.close();

cout<<"img pub ts "<<dStampNSec<<endl;

pSystem->PubImageFeature(dStampNSec, feature_vec);
cv::imshow("SOURCE IMAGE", image_frame);
cv::waitKey(1);
usleep(5000*20/3*nDelayTimes);
```

代码细节

深蓝学院 shenlanxueyuan.com

```
void System::PubImageFeature(double dStampSec, vector<cv::Point2f>& img_features)
{
    if (!FREQ)
    {
        FREQ =30;
        cout<<"frequency is "<<FREQ<<endl;
    }
    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;       You, a day ago • update
        first_image_time = dStampSec;
        last_image_time = dStampSec;
        return;
    }
```

3.调整image数据读取流程

在原项目中,system处理的image数据为Mat格式,而我们希望system处理的则是由特征点组成的vector,因此在system中,我们需要重新定义一个函数来处理接收到的仿真数据,并且在`PubImageData`这个函数中,涉及到时间戳检查的相关代码其实也可以去掉,因为得到的仿真数据,频率是规则的

代码细节

深蓝学院 shenlanxueyuan.com

```
void Estimator::setSimParameter()
{
    You, a day ago • update

    FOCAL_LENGTH = 460;
    SOLVER_TIME = 0.04 ;
    NUM_ITERATIONS = 8 ;
    MIN_PARALLAX =0.02;
    MIN_PARALLAX = MIN_PARALLAX / FOCAL_LENGTH;

    string OUTPUT_PATH;
    //fsSettings["output_path"] >> OUTPUT_PATH;
    VINS_RESULT_PATH = OUTPUT_PATH + "/vins_result_no_loop.txt";


    ACC_N = 0.08;
    ACC_W = 4e-05;
    GYR_N = 0.015;
    GYR_W = 2e-06;
    G.z()=9.81;
    ROW = 1000;
    COL = 1000;
    // ROS_INFO("ROW: %f COL: %f ", ROW, COL);

    ESTIMATE_EXTRINSIC = 0;
    {
        if (ESTIMATE_EXTRINSIC == 0){
            std::cout << " fix extrinsic param " << endl;
        }
        cv::Mat cv_R, cv_T;
        float R_data[9] = { 0, 0, -1,-1, 0, 0,0, 1, 0};
        float t_data[3] ={ 0.05,0.04,0.03};
        cv_R = cv::Mat(3, 3, CV_32F, R_data);
        cv_T = cv::Mat(3, 1, CV_32F, t_data);
        Eigen::Matrix3d eigen_R;
        Eigen::Vector3d eigen_T;
        cv::cv2eigen(cv_R, eigen_R);
        cv::cv2eigen(cv_T, eigen_T);
        Eigen::Quaterniond Q(eigen_R);
        eigen_R = Q.normalized();
        RIC.push_back(eigen_R);
        TIC.push_back(eigen_T);

    }

    INIT_DEPTH = 5.0;
    BIAS_ACC_THRESHOLD = 0.1;
    BIAS_GYR_THRESHOLD = 0.1;

    TD = 0;
    ESTIMATE_TD = 0;
    TR = 0;//Rolling shutter

    MAX_CNT = 150;
    MIN_DIST = 30;
    ROW = 1000;//todo
    COL = 1000;//todod
    FREQ =  30;
    F_THRESHOLD =1;
    SHOW_TRACK = 1;
    EQUALIZE =1;
    FISHEYE =  0;
```

4.超参的读取

原项目用yaml文件的形式读取很多超参,并且对于后端优化至关重要,我们可以模仿yaml的写法,然后读取对应仿真项目的相关参数,这里我为了更直观的实现(省事),在estimator.cpp中直接把仿真环境的参数hardcode进去, 注意,初始化system类的时候,**可以建立一个新的构造函数**,在这个构造函数内setParameter()

代码细节

深蓝学院
shenlanxueyuan.com

1. 在vins-mono中实现的feature_tracker这个特征点的跟踪功能(对应vinsmono项目的一个node)可以被抛弃,因为我们在仿真环境下已经可以拿到了特征点数据
2. vins mono需要有acc_n,acc_w, gyr_n, gyr_w这几个参数,但是原始的仿真参数没有 给全
3. 功能的实现有尽量从易到难, 比如刚开始先完成测试文件的链接运行和hello world,然后实现imu,img读取,再然后更深入的修改代 码

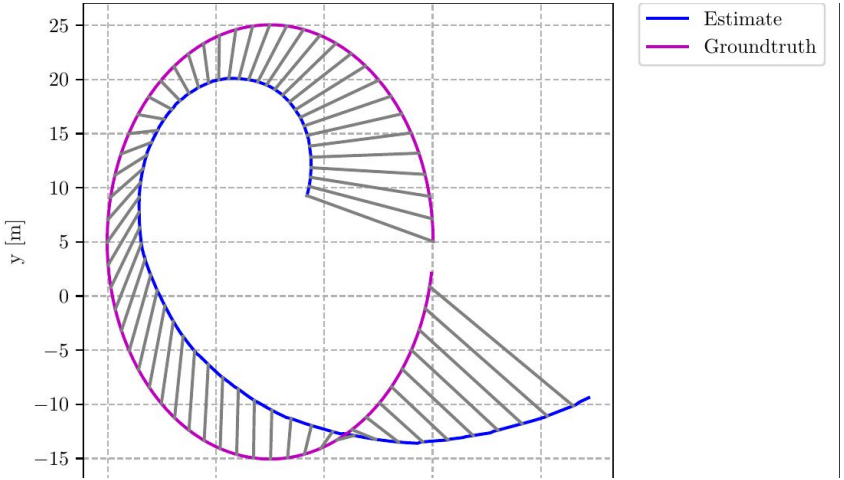
NO BUG FREE/TODO

4. 代码在debug和release模式下效果会有区 别,因此建议统一在release模式下观察效果
5. 如果join全部线程,运行处理完所有数据后不能自 动结束

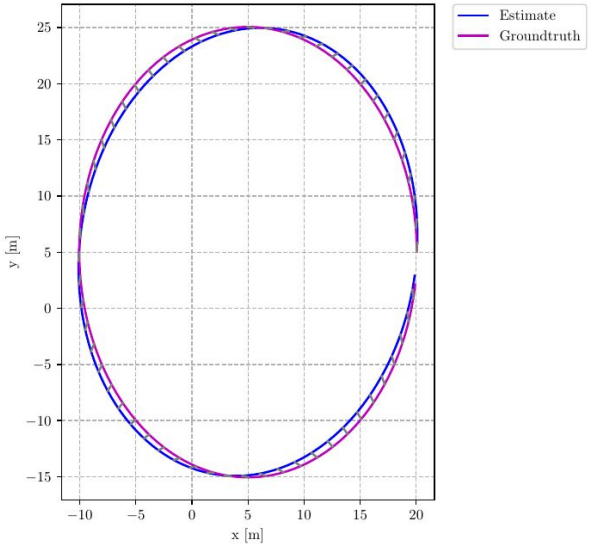
数据分析

除了kitti evo之外,还可以使用Uzh提供的工具:

https://github.com/uzh-rpg/rpg_trajectory_evaluation



imu有噪声时的俯视图



imu无噪声时的俯视图

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening