第七章作业

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

```cpp
void PubImuData()
{
    // change the path
    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;
    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();
}
```

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

```cpp
void PubImageData()
{

    string sImage_file = "../data/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;

    // cv::namedWindow("SOURCE IMAGE", CV_WINDOW_AUTOSIZE);
    while (std::getline(fsImage, sImage_line) && !sImage_line.empty())
    {
        std::istringstream ssImgData(sImage_line);
        ssImgData >> dStampNSec;
        cout<<"cam time: "<<fixed<<dStampNSec<<endl;
        string all_points_file_name = "../data/keyframe/all_points_" + to_string(n)+ ".txt";
        cout<<"points_file: "<<all_points_file_name<<endl;

        vector<Point2f> FeaturePoints;
        std::ifstream f;
        f.open(all_points_file_name);
```

```
    // file content in each line: x, y, z, 1, u, v
    while(!f.eof())
    {
        std::string s;
        std::getline(f,s);
        if(!s.empty())
        {
            std::stringstream ss;
            ss << s;

            double ignored;
            for(int i=0;i<4;i++)
                ss>>ignored;

            float px,py;
            ss >> px;
            ss >> py;
            cv::Point2f pt( px, py);

            FeaturePoints.push_back(pt);
        }
    }

    pSystem->PubImageData(dStampNSec, FeaturePoints);

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

```
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->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);
        }
    }
```