第七讲-习题

姜帆

2019 年 8 月 11 日

目录

1 vins 代码仿真数据接口修改

在 test 文件夹中新 *simulation.cpp* 作为运行仿真数据的函数接口。主要重写 $PubImuData()$ $PubImageDat$ 函数。IMU 数据按照仿真生成数据格式读取便可，图像数据由于仿真生成的是每帧图像所得到的固定的 36 个特征点，因此读入特征点的像素坐标，并导入系统中，生成对应的 $feature_buf$ 即可，不需要进行匹配跟踪。

```
void PubImageData()
{
    string sImage_file = sConfig_path + "keyframe/all_points_";
    cout << "1 PubImageData start sImage_file: " << sImage_file << endl;
    string config_file = sConfig_path + "simu_config.yaml";
    cv::FileStorage fsSettings(config_file, cv::FileStorage::READ);
    if (!fsSettings.isOpened())
    {
        cerr << "1 readParameters ERROR: Wrong path to settings!" << endl;
        return;
    }
        cv::FileNode n = fsSettings["projection_parameters"];
        double fx = static_cast<double>(n["fx"]);
        double fy = static_cast<double>(n["fy"]);
        double cx = static_cast<double>(n["cx"]);
        double cy = static_cast<double>(n["cy"]);

    string featuer_file;
    vector<cv::Point2f> points;
    for(int i =0; i<600;i++)
    {
        featuer_file = sImage_file + to_string(i) + ".txt";
        ifstream fsImage;
        fsImage.open(featuer_file.c_str());
        if (!fsImage.is_open())
        {
            cerr << "Failed to open image feature file! " << featuer_file << endl;
            return;
        }
        t += 1.0/30;
        cv::Point3d p;
        cv::Point2f feature;
        double u,v;
        int num;
        Mat image(image_w, image_h, CV_8UC1, Scalar(0,0,0));
        cv::Vec3b a(255,255,255);
        while(!fsImage.eof())
        {
            fsImage >> p.x >> p.y >> p.z >> num >> feature.x >> feature.y;

            u=fx*feature.x+cx;
            v=fy*feature.y+cy;
            feature.x = u;
            feature.y = v;
            points.push_back(feature);
            if(u< 640 && u > 0 && v > 0 && v < 640 )
                image.at<cv::Vec3b>(u,v) = a;
        }
        fsImage.close();

        pSystem->PubSimuImageData(t,image, points);
        //pSystem->PubImageData(t,image);
        cv::namedWindow("IMAGE2", CV_WINDOW_AUTOSIZE);
        cv::imshow("IMAGE2", image);
        cv::waitKey(1);
        usleep(50000*nDelayTimes);
        points.clear();
    }
}
```

图 1: 代码修改

跟踪模块作出如下修改，直接对特征点进行赋值，不需要对图像进行角点提取与光流跟踪。

```cpp
void FeatureTracker::readSimuImageFeature(const cv::Mat &_img, const vector<cv::Point2f> points, double _cur_time)
{
    cv::Mat img;
    TicToc t_r;
    cur_time = _cur_time;

    if (EQUALIZE)
    {
        cv::Ptr<cv::CLAHE> clahe = cv::createCLAHE(3.0, cv::Size(8, 8));
        TicToc t_c;
        clahe->apply(_img, img);
        //ROS_DEBUG("CLAHE costs: %fms", t_c.toc());
    }
    else
        img = _img;

    if (forw_img.empty())
    {
        prev_img = cur_img = forw_img = img;
    }
    else
    {
        forw_img = img;
    }

    forw_pts.clear();
    if (cur_pts.size() > 0)
    {
        TicToc t_o;
        vector<uchar> status;
        vector<float> err;
        forw_pts = points;
    }

    for (auto &n : track_cnt)
        n++;

    if (PUB_THIS_FRAME)
    {
        rejectWithF();
        //ROS_DEBUG("set mask begins");
        TicToc t_m;
        setMask();
        //ROS_DEBUG("set mask costs %fms", t_m.toc());

        //ROS_DEBUG("detect feature begins");
        TicToc t_t;
        forw_pts.clear();
        n_pts = points;
        //ROS_DEBUG("detect feature costs: %fms", t_t.toc());

        //ROS_DEBUG("add feature begins");
        TicToc t_a;
        addPoints();
        //ROS_DEBUG("selectFeature costs: %fms", t_a.toc());
    }
    prev_img = cur_img;
    prev_pts = cur_pts;
    prev_un_pts = cur_un_pts;
    cur_img = forw_img;
    cur_pts = forw_pts;
```

图 2: 代码修改