ORBSLAM HW2

1.使用evo评估轨迹精度误差

单目:

```
# 执行命令
./Examples/Monocular/mono_euroc Vocabulary/ORBvoc.txt
Examples/Monocular/EuRoC.yaml
/media/wegatron/data/data/euroc/mav0/cam0/data Examples/Monocular/EuRoC_TimeStamps/MH03.txt
# 评估命令
evo_rpe euroc -as
/media/wegatron/data/data/euroc/mav0/state_groundtruth_estimate0/data.csv KeyFrameTrajectory.txt
--save_results result/orbslam2_mono.zip
```

误差结果:

```
RPE w.r.t. translation part (m)
for delta = 1 (frames) using consecutive pairs
(with Sim(3) Umeyama alignment)

       max      6.337917
      mean      0.773712
    median      0.521276
       min      0.002196
      rmse      1.257088
       sse      257.583911
       std      0.990777
```

双目:

```
# 执行命令
./Examples/Stereo/stereo_euroc Vocabulary/ORBvoc.txt
Examples/Stereo/EuRoC.yaml /media/wegatron/data/data/euroc/mav0/cam0/data
/media/wegatron/data/data/euroc/mav0/cam1/data
Examples/Stereo/EuRoC_TimeStamps/MH03.txt
# 轨迹评估命令
evo_rpe euroc
/media/wegatron/data/data/euroc/mav0/state_groundtruth_estimate0/data.csv CameraTrajectory.txt
--save_result result/orbslam2_stero.zip
```

误差结果:

```
RPE w.r.t. translation part (m)
for delta = 1 (frames) using consecutive pairs
(with SE(3) Umeyama alignment)

       max      0.210384
      mean      0.055828
    median      0.046963
       min      0.000075
      rmse      0.071583
       sse      13.476425
       std      0.044804
```

2.四叉树筛点函数ComputeKeyPointsOctTree改为网格筛点函数ComputeKeyPointsOld.

单目:

```
RPE w.r.t. translation part (m)
for delta = 1 (frames) using consecutive pairs
(with Sim(3) Umeyama alignment)

       max      7.085651
      mean      0.862042
    median      0.513167
       min      0.003775
      rmse      1.326824
       sse      258.787739
       std      1.008635
```

双目:

```
RPE w.r.t. translation part (m)
for delta = 1 (frames) using consecutive pairs
(with SE(3) Umeyama alignment)

       max      0.200413
      mean      0.055753
    median      0.047040
       min      0.000013
      rmse      0.071424
       sse      13.416570
       std      0.044643
```

3.使用OpenCV的ORB特征提取与描述子计算函数.

在 `ComputePyramid(image);` 后加入如下代码:

```
    // Pre-compute the scale pyramid
    ComputePyramid(image);

    // extract nfeatures by opencv
    cv::Ptr<cv::ORB> orb = cv::ORB::create(nfeatures, scaleFactor, nlevels);
    _keypoints.clear(); _descriptors.clear();
    orb->detectAndCompute(image, cv::Mat()/*mask*/, _keypoints, _descriptors);
    return;
```

单目:

```
RPE w.r.t. translation part (m)
for delta = 1 (frames) using consecutive pairs
(with Sim(3) Umeyama alignment)

       max      6.976903
      mean      0.844388
    median      0.531142
       min      0.011711
      rmse      1.294183
       sse      262.960865
       std      0.980775
```

双目:

```
RPE w.r.t. translation part (m)
for delta = 1 (frames) using consecutive pairs
(with SE(3) Umeyama alignment)

       max      0.370378
      mean      0.055556
    median      0.046974
       min      0.000008
      rmse      0.071447
       sse      13.425223
       std      0.044924
```

4. 比较分析

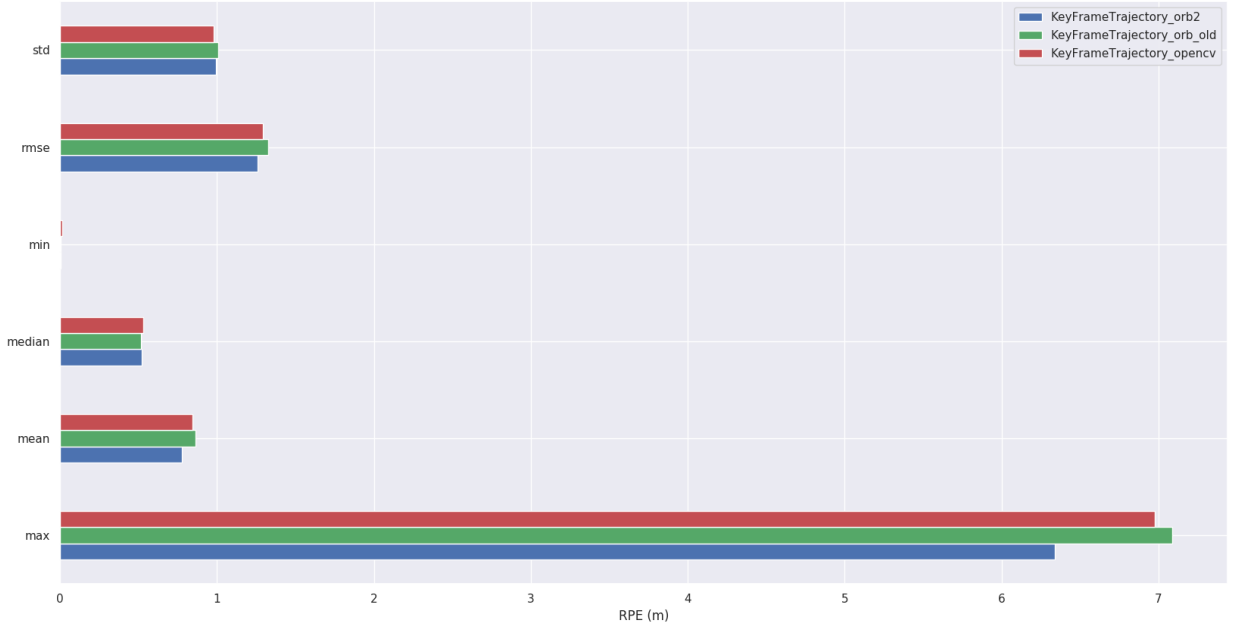
算法对比:

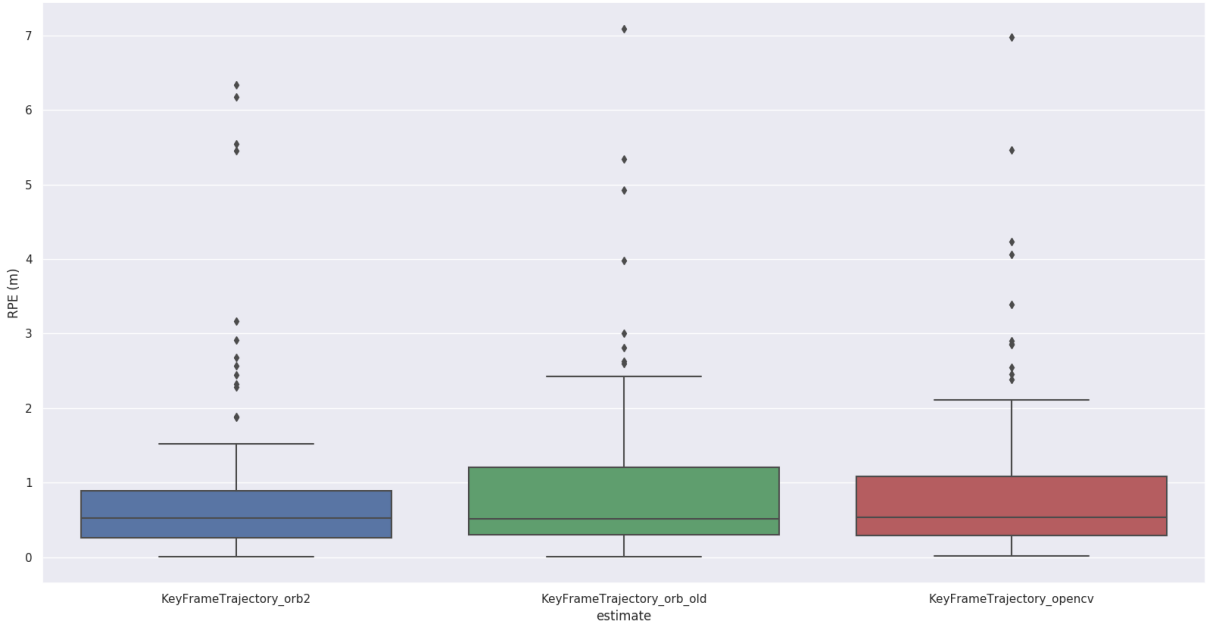
- opencv选特征点, 根据特征点的Fast阈值进行排序, 选取特征最明显的点.
- ComputeKeyPointsOld的方法选特征点, 将图像先进行区域划分, 先以较大阈值选取特征点, 若特征点数量少, 再以较大阈值选取特征点. 最后循环遍历每一个块, 选取每一个块中最好的特征点(如果有的话), 直到特征点数量足够, 或全部特征点选完为止.
- 四叉树筛点方法选取特征点, 是上一个方法的升级版, 上一个方法在一个区块内, 特征点任然可能会过于集中, 使用四叉树, 往下继续划分, 使得特征点更加均匀.

数据对比命令:

```
evo_res result/orbslam2_mono.zip result/orbslam2_old_mono.zip
result/orbslam2_cv_mono.zip -p --save_table result/mono_table.csv
```

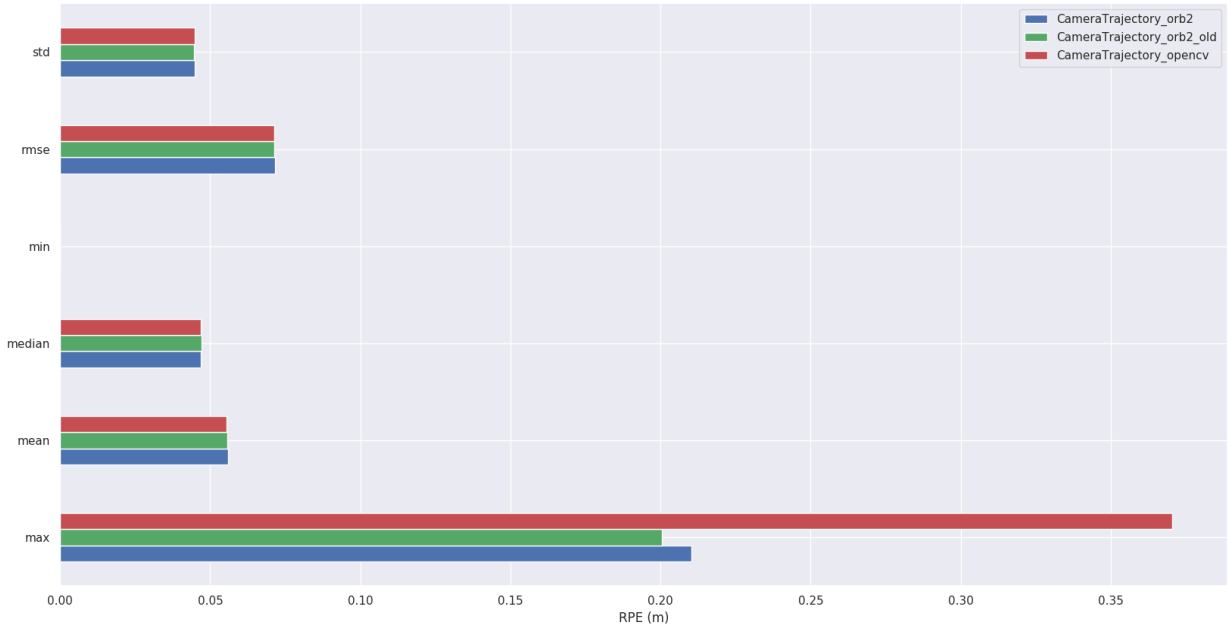
单目结果对比:

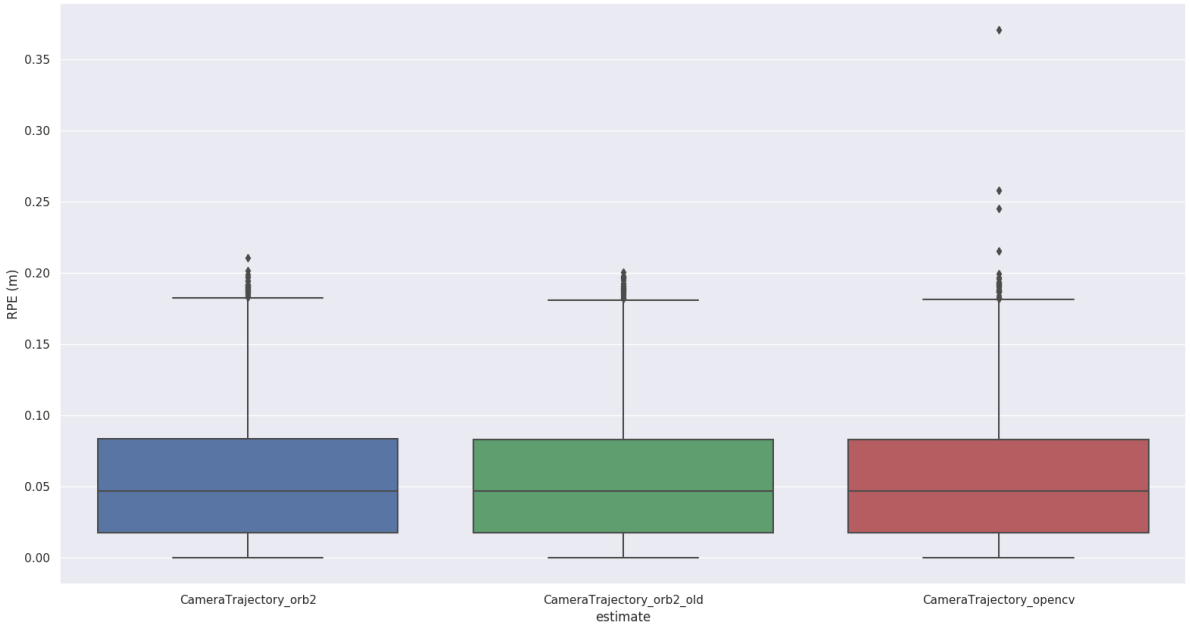




由上图可以得知, 使用四叉树筛点方式计算特征点, SLAM的整体精度和稳定性比其他两种均占有一定优势. 误差整体下降, 且更集中. 在特征点选择时, 特征更好(更明显)能够提供更精准的匹配, 而距离相近的特征点, 提供的约束相似. 因此, 四叉树筛点的方法, 综合考虑了这两个因素, 取得了更优的结果.

双目结果对比:





双目整体结果差异相对较小, 但使用opencv求特征点的方法, SLAM结果中有更多的异常值(outlier), 说明使用四叉树筛点的方法占优.

Reference

Python package for the evaluation of odometry and SLAM