ORB-SLAM2 第三次作业

使用 cv::solvePnPRansac 函数实现pnp求解当前帧的位姿.

(见代码)

尝试不同的pnp算法,并对结果做简要分析.

查阅opencv源码, 发现opencv3.2版本, `solvePnPRansac` 函数, `flag` 参数无效(opencv已经注释掉其他pnp方法的实现). 参考opencv源码, 重新封装.
使用 `RANSACPointSetRegistrator` 将算法变为RANSAC方式, 参考 `PnPRansacCallback` , 加入其他算法的实现.

关键代码如下:

```
namespace wegatron
{
    // ransac callback 实现各个EPNP的计算
    class PnPRansacCallback final : public cv::PointSetRegistrator::Callback
    {
        int runKernel( InputArray _m1, InputArray _m2, OutputArray _model ) const override
        {
            Mat opoints = _m1.getMat(), ipoints = _m2.getMat();
                if(flags == cv::SOLVEPNP_UPNP)
            {
                upnp solver(cameraMatrix, opoints, ipoints);
                cv::Mat r, t;
                solver.compute_pose(r, t);
                Rodrigues(r, rvec);
                cv::Mat _local_model;
                hconcat(rvec, t, _local_model);
                _local_model.copyTo(_model);
                return true;
            }
            if(flags == cv::SOLVEPNP_DLS)
            {
                dls solver(opoints, opoints);
                cv::Mat r, t;
                if(!solver.compute_pose(r, t))
                {
                    return false;
                }
                Rodrigues(r, rvec);
                cv::Mat _local_model;
                hconcat(rvec, t, _local_model);
                _local_model.copyTo(_model);
                return true;
            }
            cv::Mat opoints = _m1.getMat(), ipoints = _m2.getMat();
            bool correspondence = solvePnP( _m1, _m2, cameraMatrix, distCoeffs,
                                    rvec, tvec, useExtrinsicGuess, flags);
            cv::Mat _local_model;
            hconcat(rvec, tvec, _local_model);
            _local_model.copyTo(_model);
            return correspondence;
        }
    }
}

// 实现代码
Ptr<PointSetRegistrator::Callback> cb; // pointer to callback
cb = makePtr<wegatron::PnPRansacCallback>( cameraMatrix, distCoeffs,
 ransac_kernel_method, useExtrinsicGuess, rvec, tvec);
RANSACPointSetRegistrator ransac_pnp(
    cb, // 通过model callback 实现不同种pnp算法, 参考PnPRansacCallback
    model_points, // 不同算法用到的模型点的数量, 比如p3p为4, epnp为5
    threshold, // 像素误差阈值
    confidence, // 置信度
    max_iters); // 最大迭代次数

ransac_pnp->run(
    opoints, // obj points 3d 点的坐标
    ipoints, // image points 像素坐标
    local_model, // 3x2的向量, 第一列为rotation vector, 第二列为translation vector
    mask_local_inliers); // 内点 vector<bool>
```

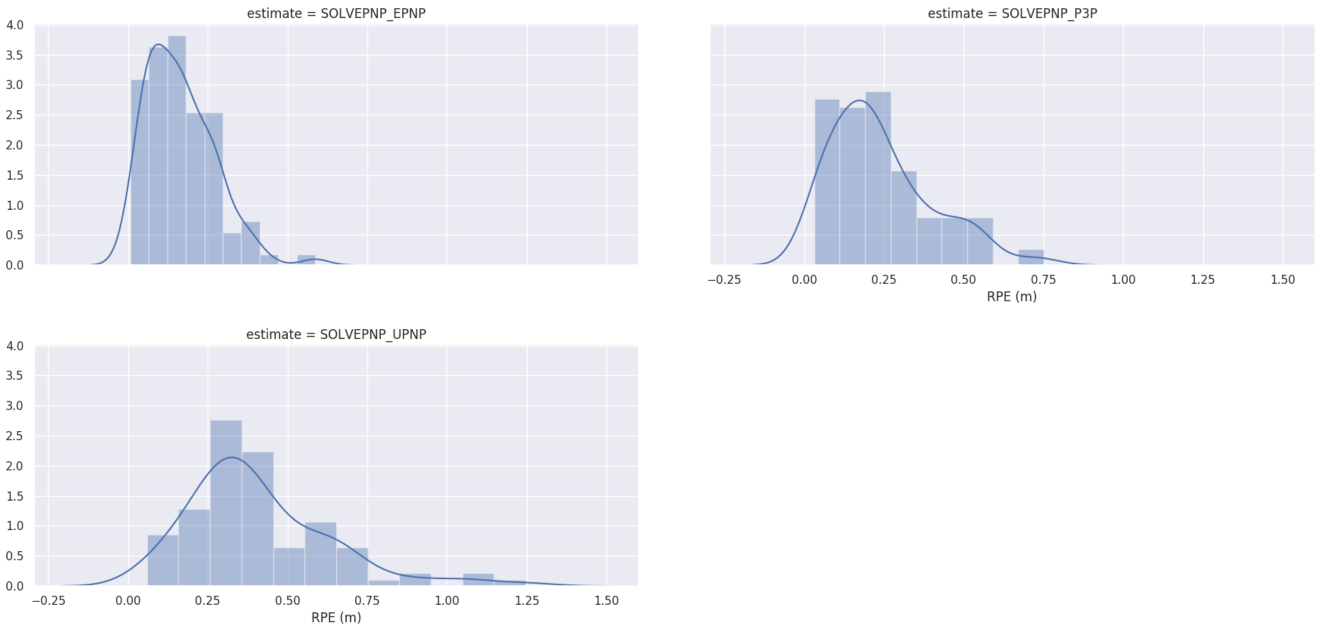
对于其他方法, 参考opencv注释掉的代码, 不使用ransac. 加入了 `DLS` 和 `UPNP` , 实验发现 `DLS` 会一直计算失败, 而 `IPPE` 是针对平面物体这里不适用.

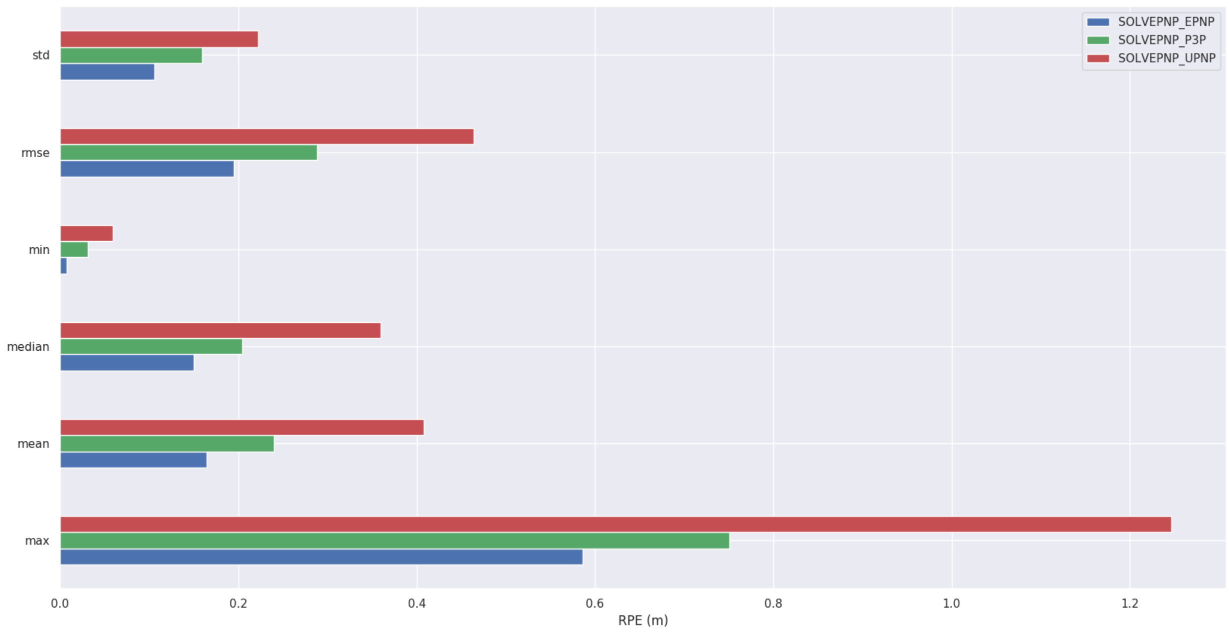
```
cv::Mat opoints(obj_pts);
cv::Mat ipoints(img_pts);
cv::Mat rmat;
if(method == cv::SOLVEPNP_DLS) {
    dls dls_pnp(opoints, ipoints);
    suc = dls_pnp.compute_pose(rmat, t);
} else if(method == cv::SOLVEPNP_UPNP){
    upnp upnp_pnp(cv_K, opoints, ipoints);
    suc = upnp_pnp.compute_pose(rmat, t);
} else {
    throw std::logic_error("unsupported pnp method!");
}
if(suc) Rodrigues(rmat, r);
```

使用python批量生成结果进行对比:

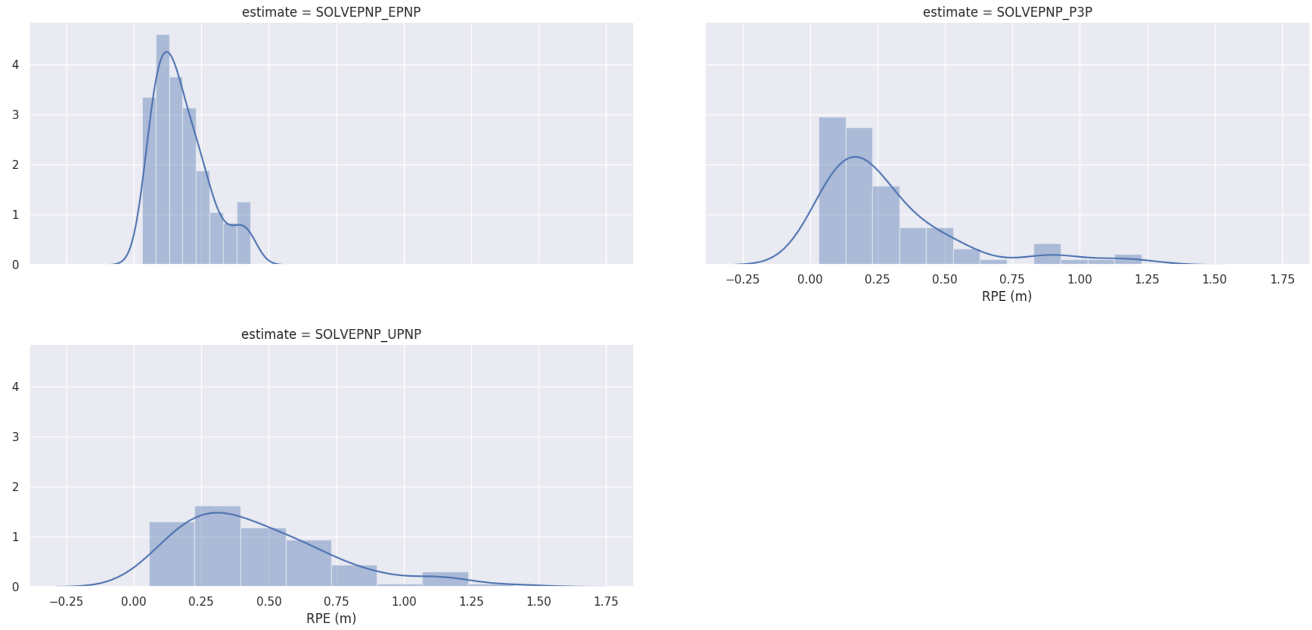
```
from shutil import copyfile
import os

itr = 20
th = 2.0
pixel_sigma = 8.0
cf = 0.95

res = [
    '',
    'SOLVEPNP_EPNP',
    'SOLVEPNP_P3P',
    'SOLVEPNP_DLS',
    'SOLVEPNP_UPNP'
]

os.system('rm -rf *')

for method in range(1, 5):
    if method is 3:
        continue
    append_str = str(itr) + ' ' + str(th) + ' ' + str(cf) + ' ' + str(method) + ' ' + str(pixel_sigma)
    print('/media/wegatron/data/workspace/zsw_work/projects/vio_course/orbslam2_course/build-hw3-Desktop_Qt_5_12_2_GCC_64bit
    os.system('/media/wegatron/data/workspace/zsw_work/projects/vio_course/orbslam2_course/build-hw3-Desktop_Qt_5_12_2_GCC_64
    traj_file = res[method] + '.txt'
    copyfile('frame_traj_est.txt', traj_file)
    print('evo_rpe tum -a frame_traj_gt.txt ' + traj_file + ' --save_result ' + res[method]+'.zip')
    os.system('evo_rpe tum -a frame_traj_gt.txt ' + traj_file + ' --save_result ' + res[method]+'.zip')

os.system('evo_res ' + res[1]+'.zip ' + res[2]+'.zip ' + res[4]+'.zip -p --save_table res.csv')
```

pixel_sigma = 0, 迭代20次



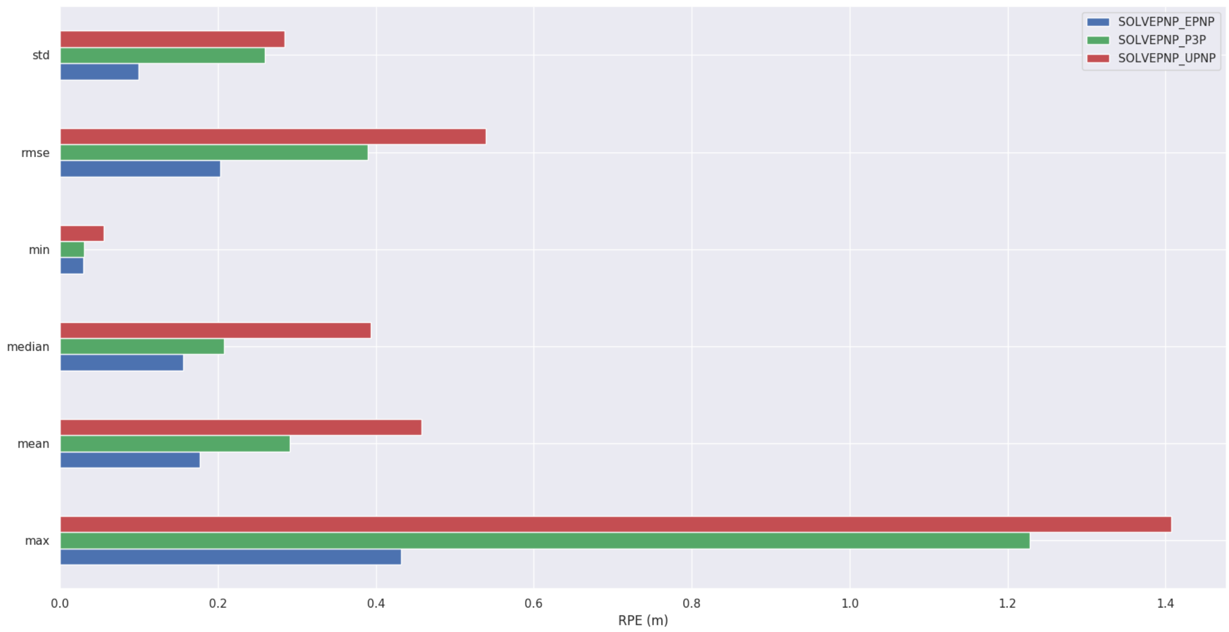


pixel_sigma = 1.0, 迭代20次

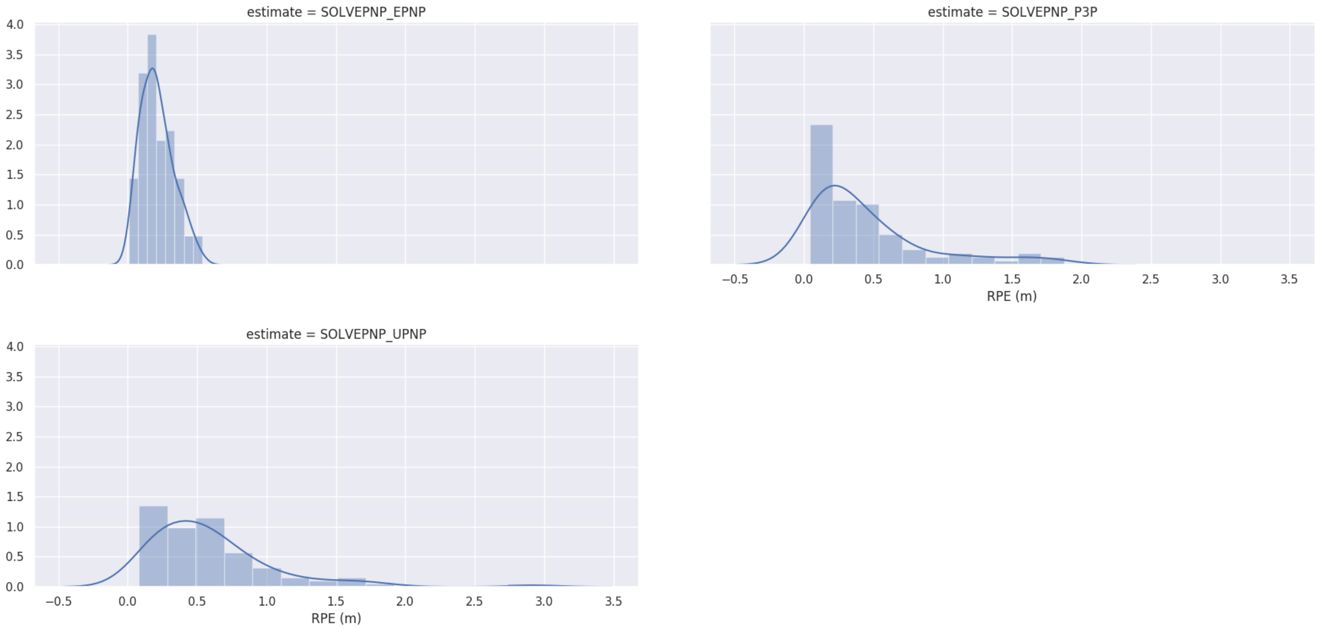

estimate = SOLVEPNP_EPNP

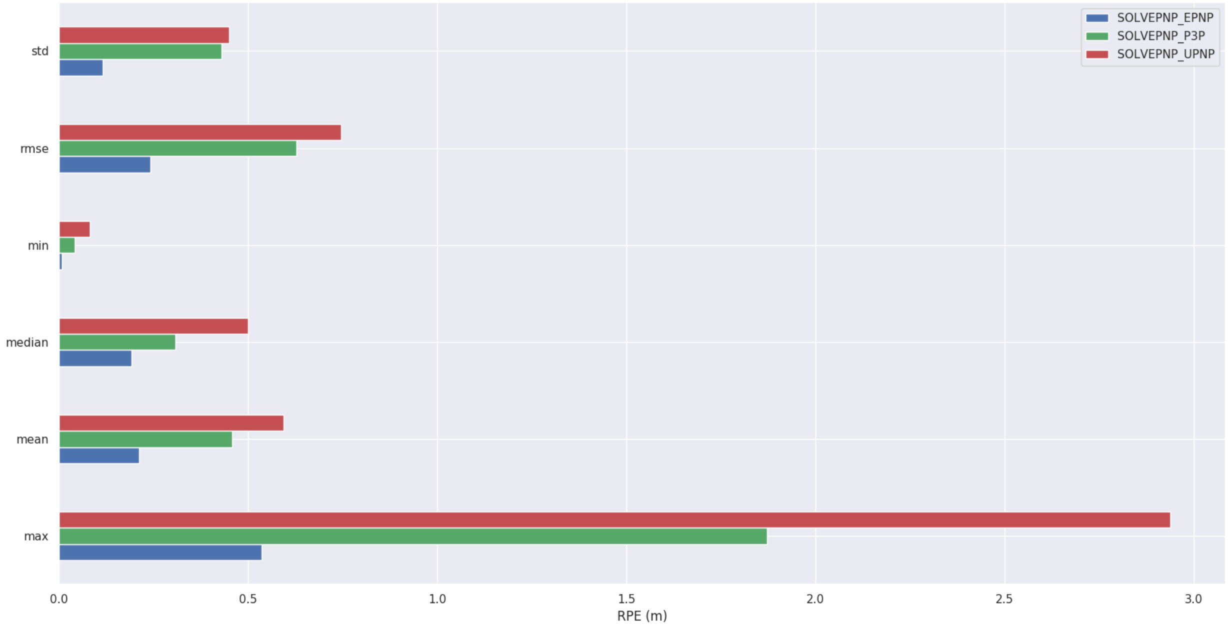
estimate = SOLVEPNP_P3P

estimate = SOLVEPNP_UPNP

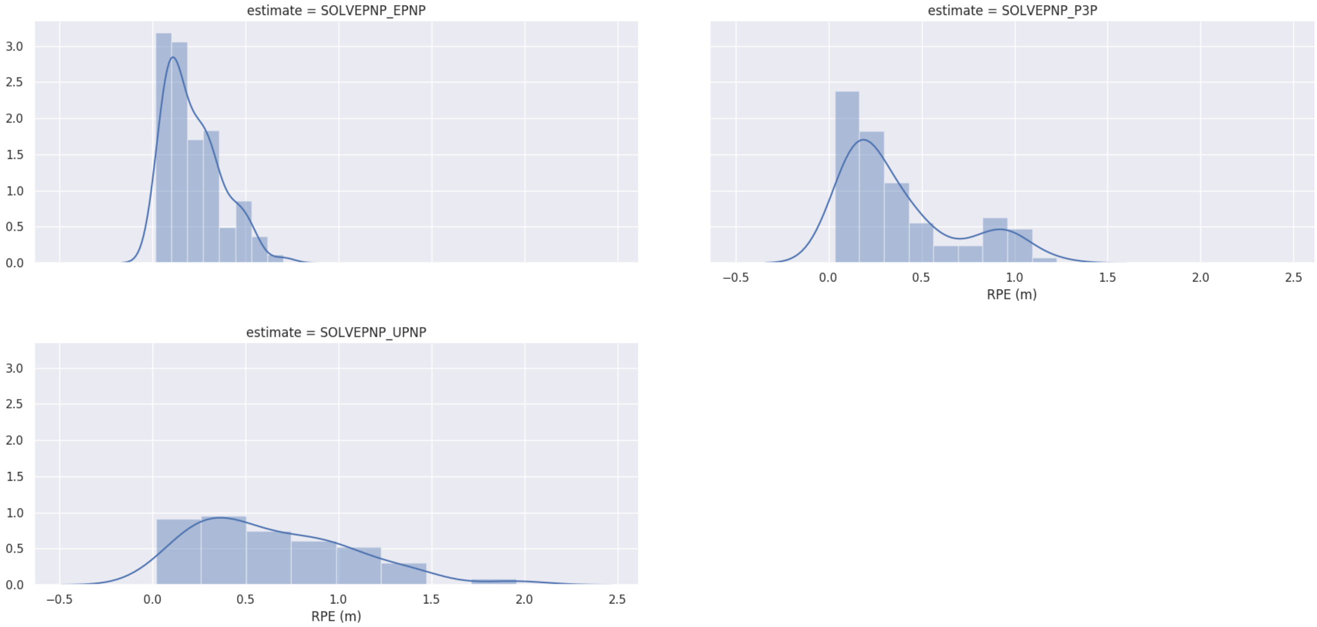


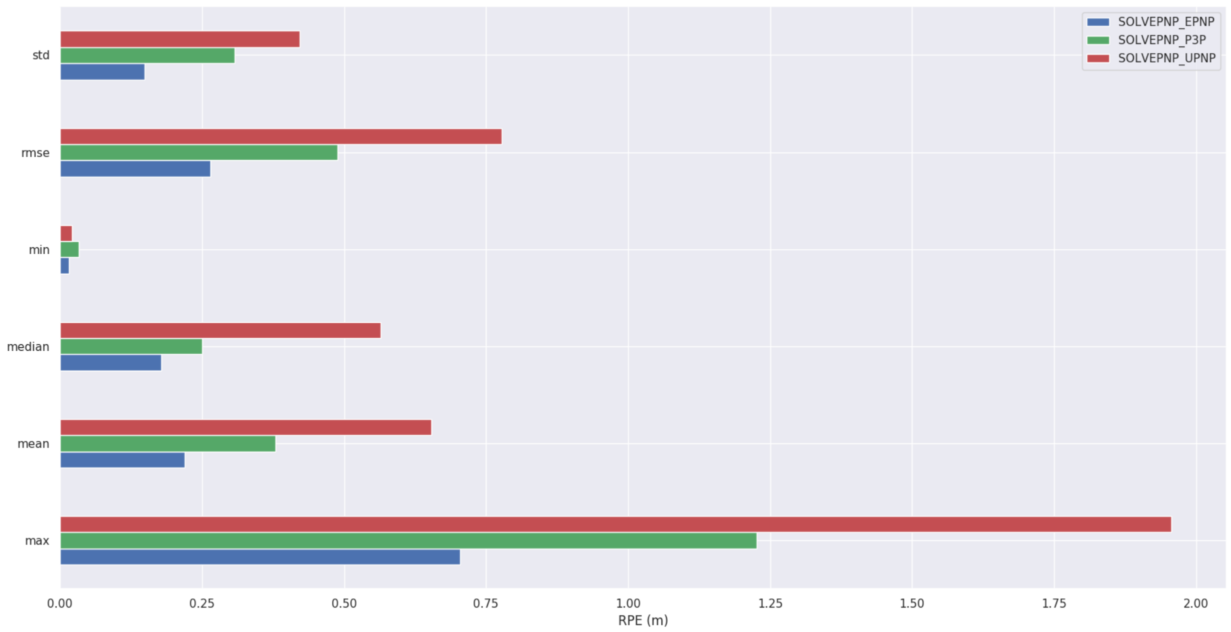
pixel_sigma = 2.0, 迭代20次



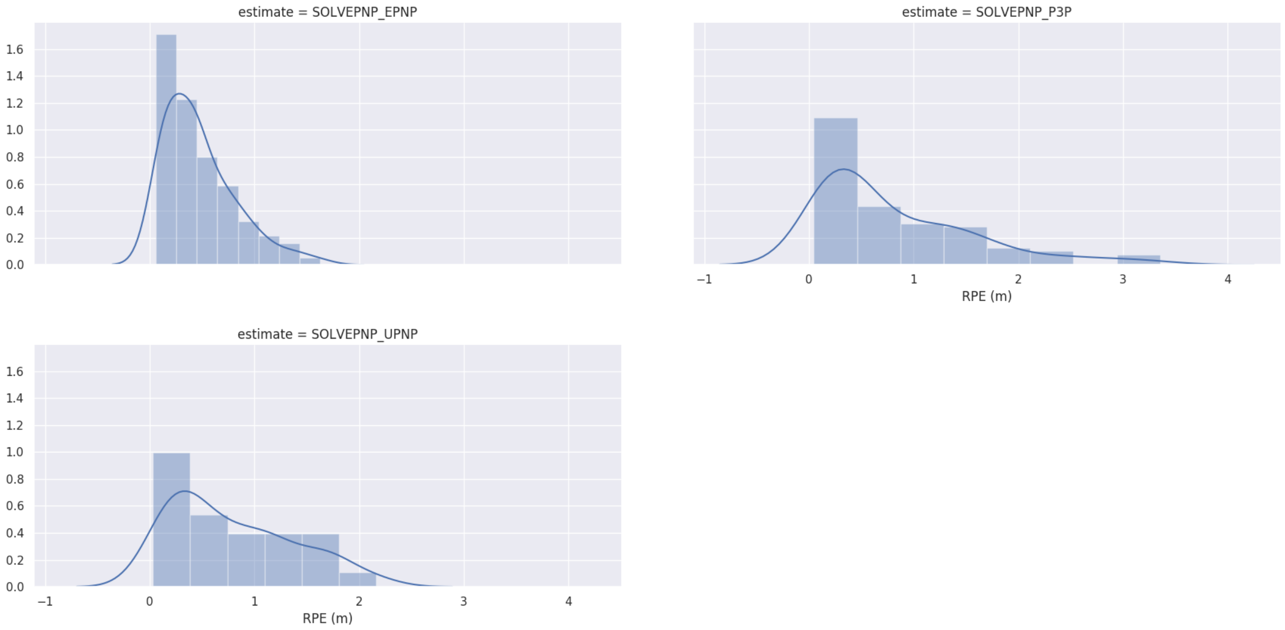


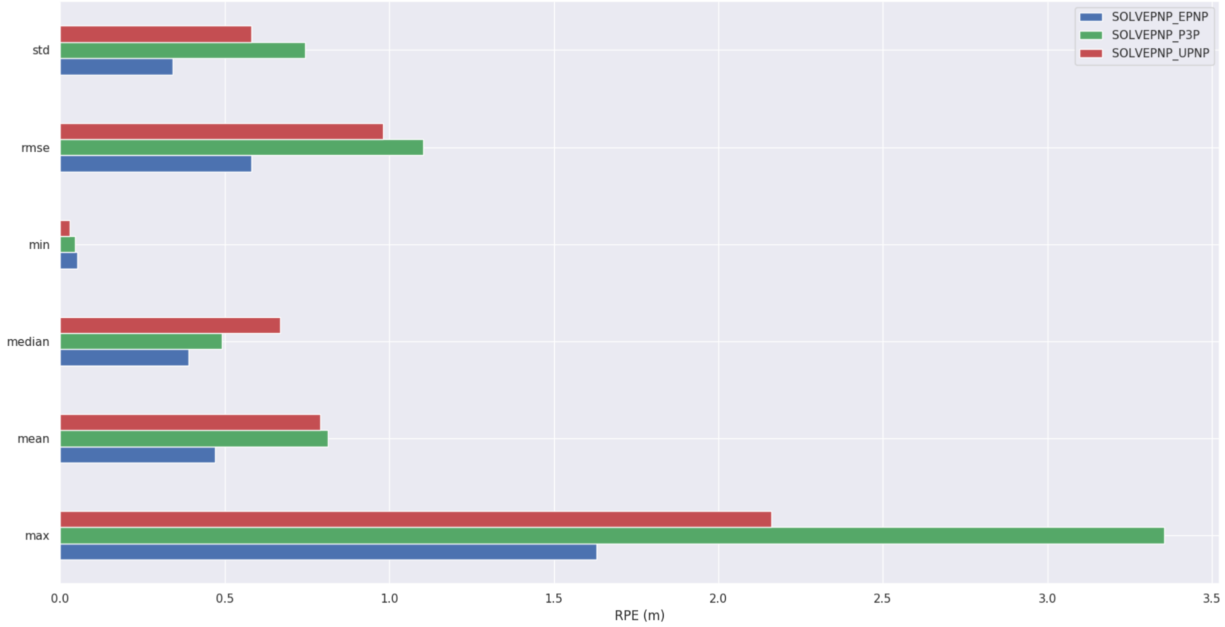
pixel_sigma = 4.0, 迭代20次,





pixel_sigma = 8.0, 迭代20次





EPNP方法对噪声的鲁棒性, 优于P3P. 而UPNP方法假定焦距未知(在计算相机位姿的同时计算相机的焦距), 其算法大部分与EPNP相同.

另外, 在使用UPNP时第一版本没有使用Ransac策略, 当噪声加到2.0时结果基本完全不对了, 上图是加入了Ransac策略后的结果, 而DLS算法计算一直失败, 这里就没加入比较.

阅读ORB-SLAM2源码中初始化以及PNP相关代码

(略)