Alvin 第四次作业

第一问: 补充的三角化的代码如下:

确定P1与P2 矩阵的值：

Eigen::Matrix3d Rtemp= frame\_curr.Twc\_.block(0,0,3,3).transpose()\*frame\_last.Twc\_.block(0,0,3,3);

Eigen::Vector3d ttemp=frame\_curr.Twc\_.block(0,0,3,3).transpose()\*(frame\_last.Twc\_.block(0,3,3,1)-frame\_curr.Twc\_.block(0,3,3,1));

cv::eigen2cv(Rtemp, R21);

cv::eigen2cv(ttemp, t21);

R21.copyTo(P2.rowRange(0,3).colRange(0,3));

t21.copyTo(P2.rowRange(0,3).col(3));

P2 = cv\_K\*P2;

其中P2矩阵所使用R，t为相对于last\_frame的相对位姿。

使用TwoViewGeometry中的三角化函数进行三角化：

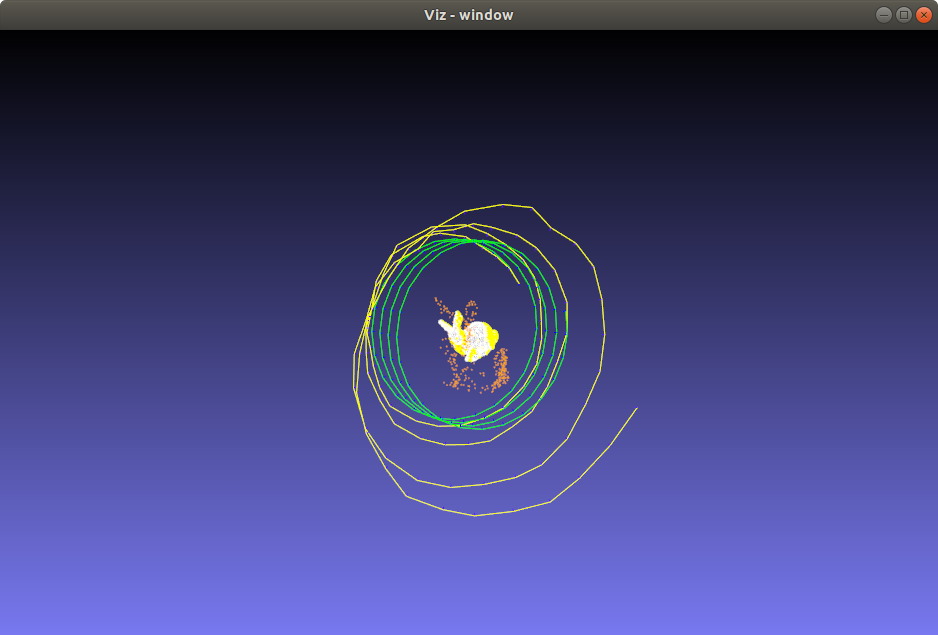
cv::Point2f point\_last,point\_curr;

point\_curr = cv::Point2f(frame\_curr.fts\_[idx\_curr][0], frame\_curr.fts\_[idx\_curr][1]);

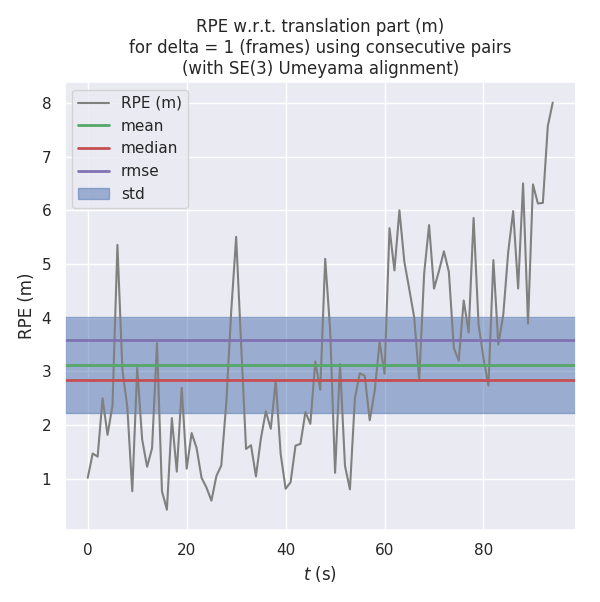
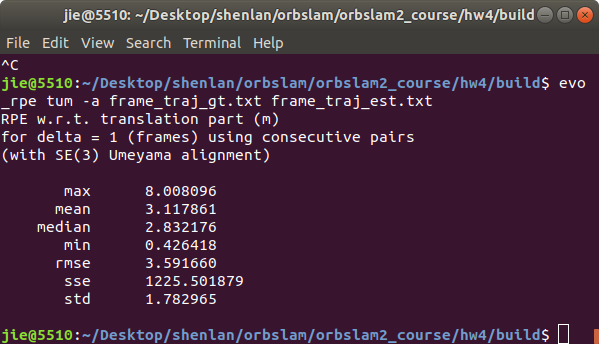
point\_last = cv::Point2f(frame\_last.fts\_[idx\_last][0], frame\_last.fts\_[idx\_last][1]);

TwoViewGeometry::Triangulate(point\_last, point\_curr, P1, P2, p3d\_c1);

运行所得结果如下图所示：



观察可知基本恢复了运动轨迹，但是对比真值明显有比较大的失真。使用rpe评估误差可得：



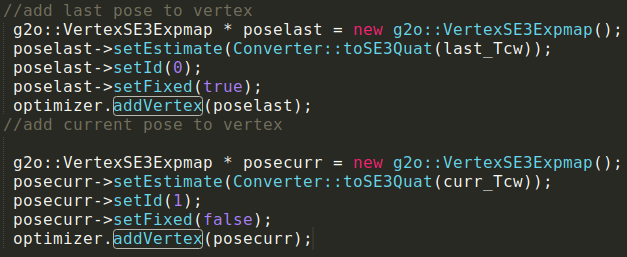
观察可以发现其rmse值较大，且RPE随时间增加有逐渐发散趋势，精度仍然需要提高。

第二问：

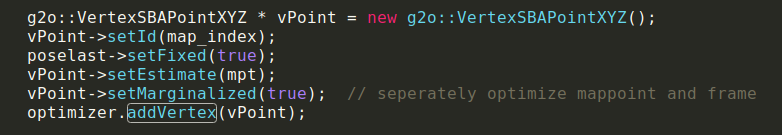
所需补全的g2o的代码，主要功能为向g2o所提供的optimizer中添加顶点(mappoint,pose)和与其中连接顶点的边的关系。

部分代码如下：

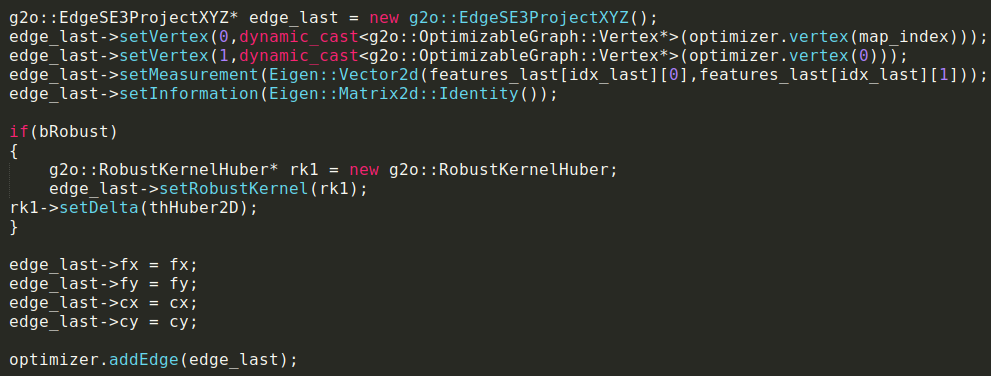
添加顶点两个相机的pose：



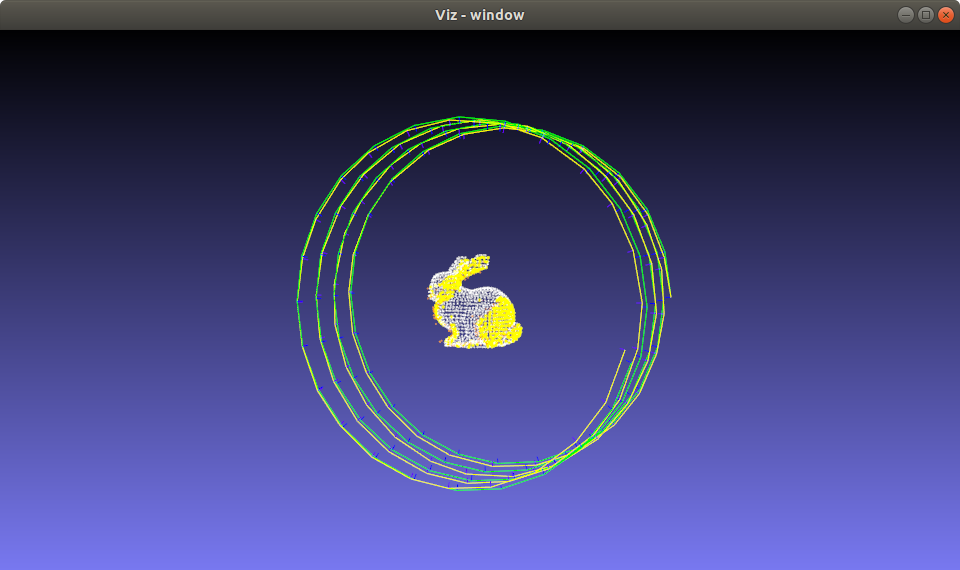
添加顶点所有mappoint的pose：



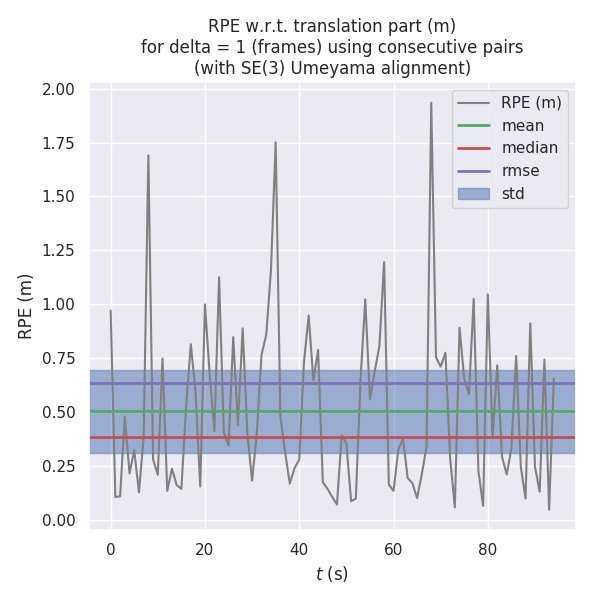
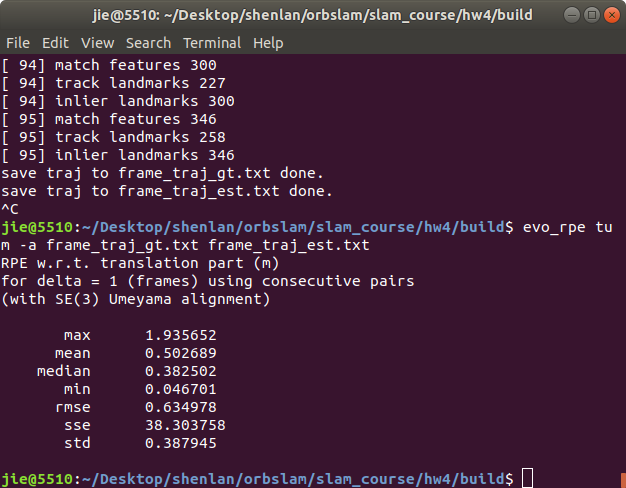
添加边last\_pose和mappoint(第二个pose添加基本相同)：



使用BA优化后的结果如下所示：



使用evo\_rpe评估轨迹的结果如下所示：



可以观察到：优化后所得轨迹其rmse明显优于BA优化之前。且RPE误差的值没有明显的随时间发散的迹象。

第三问：

在feature\_match加入outlier\_rate =0.05 后， slam程序无法完整运行提示vtk内存崩溃。使用重投影误差的评估方式去除外点：

由于mappoint是以 frame\_last 为参考坐标系确立的， 首先转换R，t 为两帧之间位姿：

Eigen::Matrix3d Rtemp= frame\_curr.Twc\_.block(0,0,3,3).transpose()\*frame\_last.Twc\_.block(0,0,3,3);

Eigen::Vector3d ttemp= frame\_curr.Twc\_.block(0,0,3,3).transpose()\*(frame\_last.Twc\_.block(0,3,3,1)-frame\_curr.Twc\_.block(0,3,3,1));

通过相应的重投影公式计算重投影误差：



当重投影误差大于某一确定阈值时则剔除外点，使用如下代码完成：

const float X = Rtemp.row(0).dot(mpt)+ttemp(0);

const float Y = Rtemp.row(1).dot(mpt)+ttemp(1);

const float Z = Rtemp.row(2).dot(mpt)+ttemp(2);

float invz = 1.0/Z;

float u1 = fx\*X\*invz+cx;

float v1 = fy\*Y\*invz+cy;

float errX = u1 - frame\_curr.fts\_[idx\_curr][0];

float errY = v1 - frame\_curr.fts\_[idx\_curr][1];

if ((errX\*errX+errY\*errY)>thr)

{

matches[n].outlier = true;

}

外点剔除在后面的运算中逐渐发散，程序仍有问题。