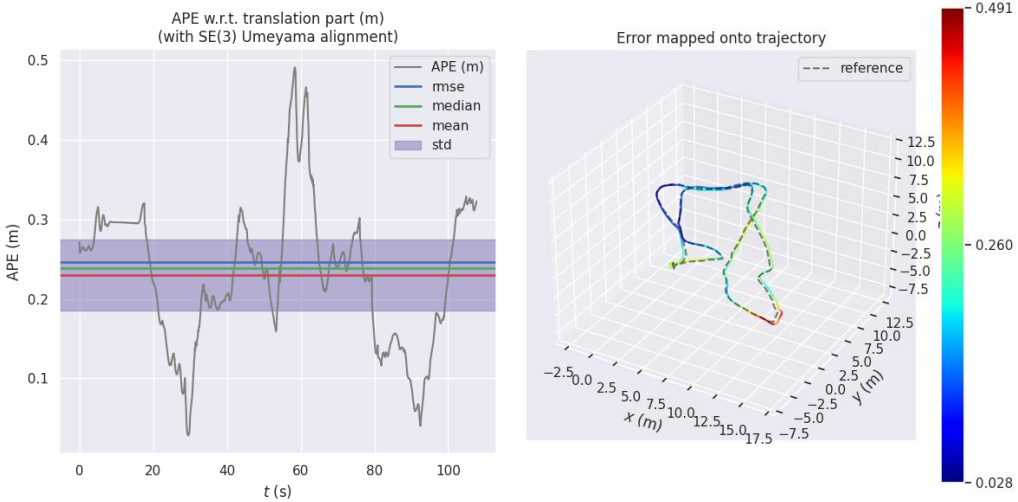
深蓝学院
shenlanxueyuan.com

大作业分享

第一题 实现更优的优化策略

深蓝学院 shenlanxueyuan.com

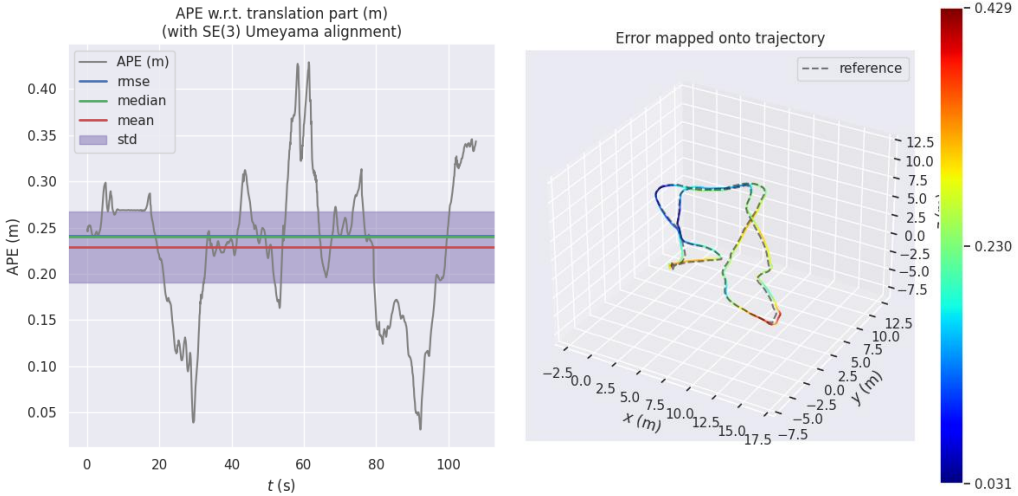
- 大作业基准代码结果



第一题 实现更优的优化策略

深蓝学院 shenlanxueyuan.com

- LM算法-迭代策略1结果



初始阻因尼子：$\mu_0 = 10^{-5}$

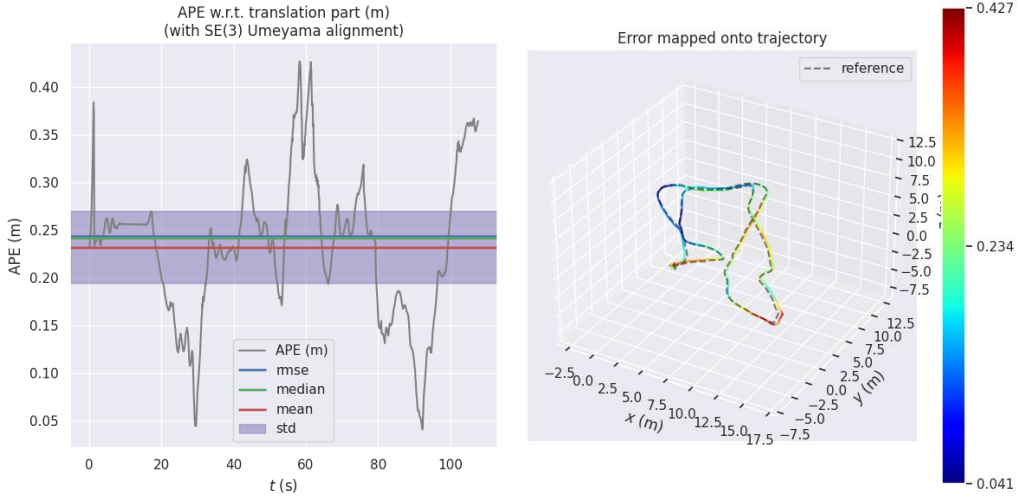
优化精度比较：LM策略1稍优。最大绝对误差下降12%，平均误差相差不明显。

优化耗时比较：差异不明显

第一题 实现更优的优化策略

深蓝学院 shenlanxueyuan.com

●DogLeg算法结果



初始信赖半径：$\Delta_0 = 1$

优化精度比较：与LM策略1接近

优化耗时比较：

- 总BA优化耗时有些微下降（约3%）
- H矩阵构建时间缩短约10%
- DogLeg算法并未针对H矩阵构建进行优化，时间缩短推测是迭代次数减少的结果。
- DogLeg需要计算最速下降以及其与GN整合的部分，有额外的计算量

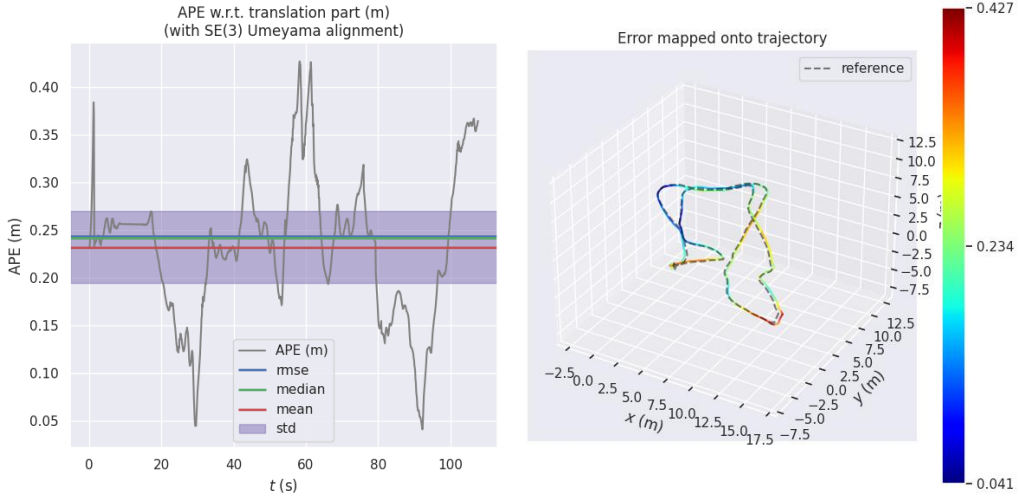
第一题 实现更优的优化策略

- DogLeg算法细节
  - DogLeg算法中高斯牛顿的求解涉及到解正规方程$\boldsymbol{H}\boldsymbol{\Delta x}_{\mathrm{gn}} = \boldsymbol{b}$
  - $\boldsymbol{H}$矩阵可能不满秩，无法利用逆矩阵求解
  - 使用广义逆，但效率可能较差
  - 参考LM算法，在$\boldsymbol{H}$矩阵上添加一个$\mu\boldsymbol{I}$（我的作业中$\mu = 10^{-3}$）
    - ceres的方法：在$\boldsymbol{H}$矩阵上添加$\mu\boldsymbol{I}$，检查结果是否确实是原正规方程的解，如果不是，逐渐增大μ的取值直到求解正确（或优化失效），并记录最后一次使用的μ，在下一次迭代求解时直接使用该μ值。此外当迭代求解成功时，将μ值减半。

第一题 实现更优的优化策略

深蓝学院 shenlanxueyuan.com

- DogLeg算法-更多的迭代停止条件



初始信赖半径：$\Delta_0 = 1$

优化精度比较：变化不大

优化耗时比较：

- 总BA优化耗时明显下降（平均耗时下降23%）
- 迭代次数减少是耗时下降的主要原因

停止条件：

- 误差绝对值变化小于阈值
- 状态变化绝对值小于阈值
- 梯度中系数绝对值最大的值（无穷范数）小于阈值

第一题 实现更优的优化策略

- 结果汇总

方法 指标	原始LM	LM策略1	DogLeg	DogLeg 带附加停止条件
最大绝对误差/m	0.491266	0.429017	0.426565	0.426095
最小绝对误差/m	0.028452	0.031461	0.040659	0.040103
绝对误差均值/m	0.239110	0.229060	0.231865	0.231518
均方误差/m	0.246862	0.241462	0.243628	0.243298
平方误差和/m^2	65.694118	62.851776	63.984134	63.810808
BA优化总耗时/ms	97917	96279	93468	71573
BA优化平均耗时/ms	89.915	88.410	85.829	65.724
H矩阵构建最大耗时/ms	63.033	60.6418	58.3707	61.5927
H矩阵构建最小耗时/ms	3.34494	3.61197	2.75379	2.4703
H矩阵构建平均耗时/ms	40.017	40.064	35.984	27.490
正规方程求解最大耗时/ms	144.454	149.259	153.624	115.713
正规方程求解最小耗时/ms	33.393	31.5723	33.928	14.8209
正规方程求解平均耗时/ms	78.680	77.030	74.473	54.049

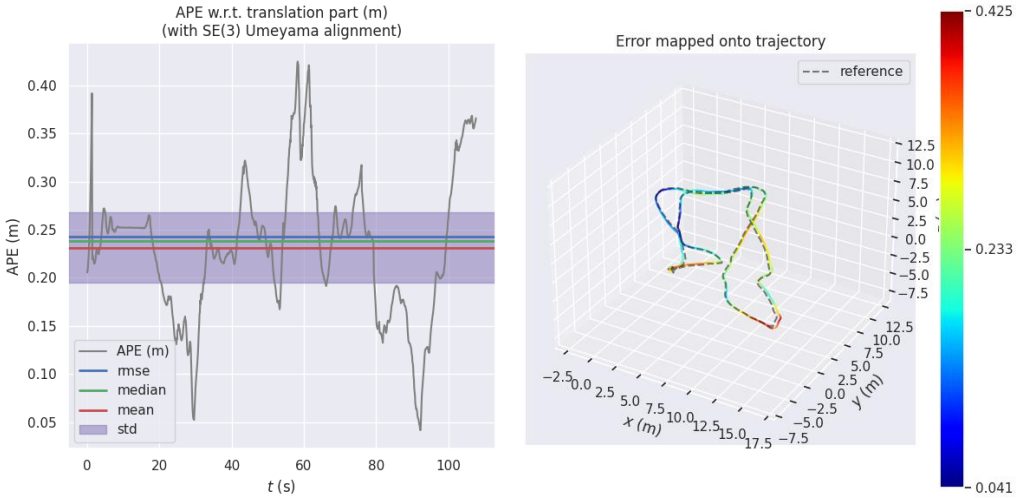
第二题 实现更快的H矩阵构建

- 使用AVX2指令集
  - Eigen中有直接的支持。使用AVX2指令集无需修改任何代码，只需在编译选项中增加-mavx2
  - 对数据对齐的要求较为严格。默认的代码直接编译会出现运行错误。最简单的处理方法是使用C++17来编译代码

第二题 实现更快的H矩阵构建

深蓝学院 shenlanxueyuan.com

●H矩阵构建优化-使用AVX2指令集



优化精度比较：差异不明显

优化耗时比较：

- BA总体耗时下降约17%
- H矩阵构建和正规方程求解均涉及矩阵运算，因此两个部分都有提升
- 正规方程求解速度的提升非常明显，耗时下降接近20%
- H矩阵构建获益不太明显，耗时仅下降3.5%

原因猜测：

- H矩阵构建中有较多小尺寸矩阵，提升空间有限
- H矩阵构建可能有其它操作更为耗时

第二题 实现更快的H矩阵构建

- **H**矩阵构建优化-残差计算优化
  - 构建**H**矩阵时需要执行一个三层循环：对每个残差遍历所有的顶点对，以构建H矩阵的一小部分，从而形成最终的完整的H矩阵
  - 每一轮迭代，带鲁棒核的残差实际上与顶点无关，可以抽作公共部分，减少计算量
  - 优化的结果：精度基本不变，**H**矩阵构建耗时明显下降（约15%）

```
for (auto &edge: edges_) {

    edge.second->ComputeResidual();
    edge.second->ComputeJacobians();

    // TODO:: robust cost
    auto jacobians = edge.second->Jacobians();
    auto verticies = edge.second->Verticies();
    assert(jacobians.size() == verticies.size());
    for (size_t i = 0; i < verticies.size(); ++i) {
        auto v_i = verticies[i];
        if (v_i->IsFixed()) continue;    // Hessian 里不需要添加它的信息，也就是它的雅克比为 0

        auto jacobian_i = jacobians[i];
        ulong index_i = v_i->OrderingId();
        ulong dim_i = v_i->LocalDimension();

        // 鲁棒核函数会修改残差和信息矩阵，如果没有设置 robust cost function，就会返回原来的
        double drho;
        MatXX robustInfo(edge.second->Information().rows(),edge.second->Information().cols());
        edge.second->RobustInfo( & drho, & robustInfo);

        MatXX JtW = jacobian_i.transpose() * robustInfo;
        for (size_t j = i; j < verticies.size(); ++j) {
            auto v_j = verticies[j];
```

第二题 实现更快的H矩阵构建

●结果汇总

方法 指标	DogLeg 带附加停止条件	DogLeg 使用AVX2指令集	DogLeg AVX2+优化残差计算
最大绝对误差/m	0.426095	0.424682	0.424682
最小绝对误差/m	0.040103	0.041445	0.041445
绝对误差均值/m	0.231518	0.231279	0.231279
均方误差/m	0.243298	0.242413	0.242413
平方误差和/m^2	63.810808	63.347578	63.347578
BA优化总耗时/ms	71573	58870	54201
BA优化平均耗时/ms	65.724	54.059	49.772
H矩阵构建最大耗时/ms	61.5927	60.4102	59.973
H矩阵构建最小耗时/ms	2.4703	3.33983	3.143
H矩阵构建平均耗时/ms	27.490	26.086	21.867
正规方程求解最大耗时/ms	115.713	100.5	96.350
正规方程求解最小耗时/ms	14.8209	13.3034	11.5866
正规方程求解平均耗时/ms	54.049	43.326	39.068

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening