深蓝学院
shenlanxueyuan.com

视觉SLAM进阶课程
从零开始手写VIO-第5期

第5讲 滑动窗口算法实践 作业讲评

评讲人 吕华龙

【作业】

深蓝学院 shenlanxueyuan.com

基础题

1. 完成单目 Bundle Adjustment (BA) 求解器 problem.cc 中的部分代码。
   - 完成 Problem::MakeHessian() 中信息矩阵 H 的计算。
   - 完成 Problem::SolveLinearSystem() 中 SLAM 问题的求解。
2. 完成滑动窗口算法测试函数。
   - 完成 Problem::TestMarginalize() 中的代码，并通过测试。

说明：为了便于查找作业位置，代码中留有 TODO:: home work 字样.

提升题

- 请总结论文[a]：优化过程中处理 H 自由度的不同操作方式。内容包括：具体处理方式，实验效果，结论。(加分题，评选良好)
- 在代码中给第一帧和第二帧添加 prior 约束，并比较为 prior 设定不同权重时，BA 求解收敛精度和速度。(加分题，评选优秀)

[a] Zichao Zhang, Guillermo Gallego, and Davide Scaramuzza. "On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2710–2717.

题目一 BA求解器

深蓝学院 shenlanxueyuan.com

(1) 完成Problem::MakeHessian()中信息矩阵H的计算

```
    // 所有的信息矩阵叠加起来
    // TODO:: home work. 完成 H index 的填写.
    // H.block(?,?, ?, ?).noalias() += hessian;
    H.block(index_i,index_j, dim_i, dim_j).noalias() += hessian;
    if (j != i) {
        // 对称的下三角
// TODO:: home work. 完成 H index 的填写.
        // H.block(?,?, ?, ?).noalias() += hessian.transpose();
        H.block(index_j,index_i, dim_j, dim_i).noalias() += hessian.transpose();
```

题目一 BA求解器

深蓝学院 shenlanxueyuan.com

(2) 完成 Problem::SolveLinearSystem()中 SLAM 问题的求解

```
// TODO:: home work. 完成矩阵块取值，Hmm，Hpm，Hmp，bpp，bmm
// MatXX Hmm = Hessian_.block(?,?, ?, ?);
// MatXX Hpm = Hessian_.block(?,?, ?, ?);
// MatXX Hmp = Hessian_.block(?,?, ?, ?);
// VecX bpp = b_.segment(?,?);
// VecX bmm = b_.segment(?,?);
MatXX Hmm = Hessian_.block(reserve_size,reserve_size, marg_size, marg_size);
MatXX Hmp = Hessian_.block(reserve_size,0, marg_size, reserve_size);
MatXX Hpm = Hessian_.block(0, reserve_size, reserve_size, marg_size);
VecX bpp = b_.segment(0,reserve_size);
VecX bmm = b_.segment(reserve_size,marg_size);
```

题目一 BA求解器

深蓝学院 shenlanxueyuan.com

(2) 完成 Problem::SolveLinearSystem()中 SLAM 问题的求解

```
// TODO:: home work. 完成舒尔补 Hpp, bpp 代码
MatXX tempH = Hpm * Hmm_inv;
// H_pp_schur_ = Hessian_.block(?,?,?,?) - tempH * Hmp;
// b_pp_schur_ = bpp - ? * ?;
H_pp_schur_ = Hessian_.block(0,0,reserve_size,reserve_size) - tempH * Hmp;
b_pp_schur_ = bpp - tempH * bmm;

// TODO:: home work. step3: solve landmark
VecX delta_x_ll(marg_size);
// delta_x_ll = ???;
delta_x_ll = Hmm_inv*(bmm-Hmp*delta_x_pp);
delta_x_.tail(marg_size) = delta_x_ll;
```

题目二 滑动窗口算法测试

完成Problem::TestMarginalize()中的代码，并通过测试

```
// TODO:: home work. 将变量移动到右下角
/// 准备工作: move the marg pose to the Hmm bottown right
// 将 row i 移动矩阵最下面
Eigen::MatrixXd temp_rows = H_marg.block(idx, 0, dim, reserve_size);
Eigen::MatrixXd temp_botRows = H_marg.block(idx + dim, 0, reserve_size - idx - dim, reserve_size);
// H_marg.block(?,?,?,?) = temp_botRows;
// H_marg.block(?,?,?,?) = temp_rows;
H_marg.block(idx, 0, dim, reserve_size) = temp_botRows;
H_marg.block(idx + dim, 0, reserve_size - idx - dim, reserve_size) = temp_rows;

// TODO:: home work. 完成舒尔补操作
//Eigen::MatrixXd Arm = H_marg.block(?,?,?,?);
//Eigen::MatrixXd Amr = H_marg.block(?,?,?,?);
//Eigen::MatrixXd Arr = H_marg.block(?,?,?,?);
Eigen::MatrixXd Arm = H_marg.block(0,n2,n2,m2);
Eigen::MatrixXd Amr = H_marg.block(n2,0,m2,n2);
Eigen::MatrixXd Arr = H_marg.block(0,0,n2,n2);
```

题目三 H自由度的操作方法比较

深蓝学院 shenlanxueyuan.com

solver 求解中的小疑问

1. 上节课说到信息矩阵 $\mathbf{H}$ 不满秩，那求解的时候如何操作呢？
   - 使用 LM 算法，加阻尼因子使得系统满秩，可求解，但是求得的结果可能会往零空间变化。
   - 添加先验约束，增加系统的可观性。比如 g2o tutorial 中对第一个 pose 的信息矩阵加上单位阵 $\mathbf{H}_{[11]} += \mathbf{I}$.

2. orbslam，svo 等等求 mono BA 问题时，fix 一个相机 pose 和一个特征点，或者 fix 两个相机 pose，也是为了限定优化值不乱飘。那代码如何实现 fix 呢?
   - 添加超强先验，使得对应的信息矩阵巨大（如，10^{15}），就能使得 $\Delta x = 0$;
   - 设定对应雅克比矩阵为 0，意味着残差等于 0. 求解方程为 $(\mathbf{0} + \lambda \mathbf{I})\, \Delta \mathbf{x} = \mathbf{0}$，只能 $\Delta x = 0$。

题目三 H自由度的操作方法比较

On the Comparison of Gauge Freedom Handling in Optimization-based Visual-Inertial State Estimation

1. **Gauge prior：**给不可观的状态上添加先验，通过添加一个惩罚项，使得H矩阵可逆，相当于在目标函数中添加一个由先验项组成的惩罚项。

2. **Gauge fixation：**将不可观测状态固定到某些给定值，在优化过程中固定第一个相机的位置和yaw角。在比较小的参数空间中进行优化，在这个空间中没有不可观测的状态，H是可逆的。

3. **Free gauge：**允许不客观状态量在优化中随意变化，使用奇异的Hessian的伪逆矩阵来提供额外的约束（使用最小范数来更新参数），获得唯一解，或者通过添加一些阻尼项使得最小二乘问题有一个良好定义的参数更新。

题目三 H自由度的操作方法比较

首先比较三种不同处理方式对优化状态量和信息矩阵维度的影响

	Size of parameter vec.	Hessian (Normal eqs)
Fixed gauge	$n-4$	inverse, $(n-4)\times(n-4)$
Gauge prior	n	inverse, $n\times n$
Free gauge	n	pseudoinverse, $n\times n$

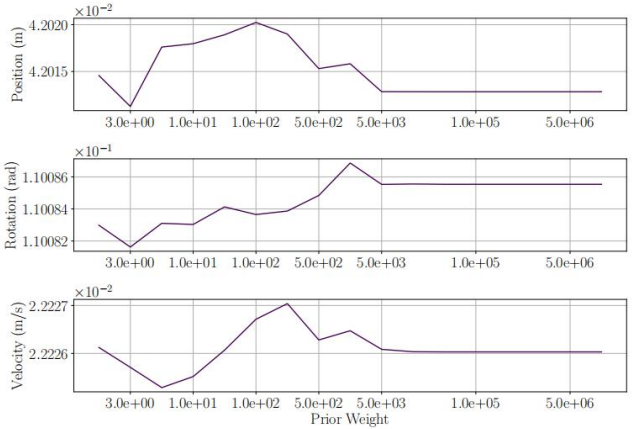

Fig. 4: RMSE in position, orientation and velocity for different prior weights

然后对Gauge prior选取不同的先验权重，比较对精度和计算效率的影响

- 不同的先验值对位姿求解的精度影响并不大
- 当先验权重大于一定阈值时，迭代次数和收敛时间趋于稳定

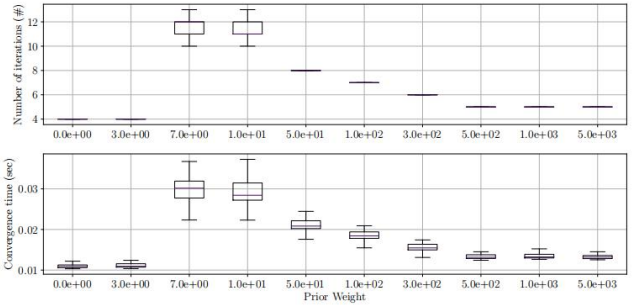

Fig. 5: Number of iterations and computing time for different prior weights.

题目三 H自由度的操作方法比较

分别从精度、计算效率、协方差三个维度对三种方法进行了评估

- 精度：RMSE (位置和速度的欧式距离、旋转的相对角度误差)
- 计算效率：收敛时间和迭代次数
- 协方差：信息矩阵的逆

Configuration	Gauge fixation			Free gauge		
	p	ϕ	v	p	ϕ	v
sine plane	0.04141	0.1084	**0.02182**	0.04141	0.1084	0.02183
arc plane	**0.02328**	0.6987	0.01303	0.02329	0.6987	0.01303
rec plane	**0.01772**	0.1668	0.01496	0.01774	0.1668	**0.01495**
sine random	0.03932	0.0885	0.01902	**0.03908**	**0.0874**	**0.01886**
arc random	0.02680	0.6895	0.01167	**0.02678**	0.6895	**0.01166**
rec random	**0.02218**	0.1330	0.009882	0.02220	0.1330	**0.009881**

Position, rotation and velocity RMSE are measured in m, deg and m/s, respectively.

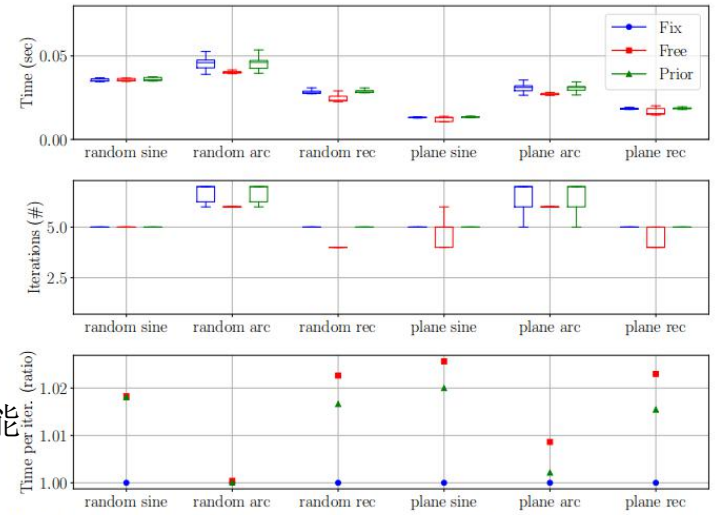

Fig. 7: Number of iterations, total convergence time and time per iteration for all configurations. The time per iteration is the ratio with respect to the gauge fixation approach (in blue), which takes least time per iteration.

结论：

- 三种方法有相同的精度
- Gauge prior方法需要选择合适的先验权值以避免计算量的增加
- 在适当的权重值下，Gauge prior 与Gauge fixed 具有几乎相同性能
- Gauge free 略快于其他方法，需要较少的迭代就能收敛

题目四 添加prior约束

$$r_{prior} = \begin{bmatrix} r_R \\ r_p \end{bmatrix}_{prior} = \begin{bmatrix} \ln(\tilde{R}_{wc}^{-1} R_{wc}) \\ p_{wc} - \tilde{p}_{wc} \end{bmatrix}$$

$\tilde{R}_{wc}$、$\tilde{p}_{wc}$分别为R_{wc}、p_{wc}的先验

$$J = \frac{\partial r_{prior}}{\partial \begin{bmatrix} R_{wc} \\ p_{wc} \end{bmatrix}} = \begin{bmatrix} \frac{\partial r_R}{\partial R_{wc}} & 0 \\ 0 & \frac{\partial r_p}{\partial p_{wc}} \end{bmatrix}$$

$$\begin{aligned} \frac{\partial r_R}{\partial R_{wc}} &= \frac{\partial \ln(\tilde{R}_{wc}^{-1} R_{wc})}{\partial R_{wc}} \\ &= \lim_{\delta\theta \to 0} \frac{\ln(\tilde{R}_{wc}^{-1} R_{wc} \exp(\delta\theta)) - \ln(\tilde{R}_{wc}^{-1} R_{wc})}{\delta\theta} \\ &= \lim_{\delta\theta \to 0} \frac{r_R + J_r^{-1}\delta\theta - r_R}{\delta\theta} \\ &= J_r^{-1}(r_R) \end{aligned}$$

$$\frac{\partial r_p}{\partial p_{wc}} = \frac{\partial (p_{wc} - \tilde{p}_{wc})}{\partial p_{wc}} = I$$

edge_prior.cpp

backend > edge_prior.cpp > ...

```
61
62  void EdgeSE3Prior::ComputeJacobians()
63  {
64
65      VecX param_i = verticies_[0]->Parameters();
66      Qd Qi(param_i[6], param_i[3], param_i[4], param_i[5]);
67
68      // w.r.t. pose i
69      Eigen::Matrix<double, 6, 6> jacobian_pose_i = Eigen::Matrix<double, 6, 6>::Zero();
70
71  #ifdef USE_SO3_JACOBIAN
72      Sophus::SO3d ri(Qi);
73      Sophus::SO3d rp(Qp_);
74      Sophus::SO3d res_r = rp.inverse() * ri;
75      // http://rpg.ifi.uzh.ch/docs/RSS15_Forster.pdf  公式A.32
76      jacobian_pose_i.block<3, 3>(0, 3) = Sophus::SO3d::JacobianRInv(res_r.log()); // JR
77  #else
78      jacobian_pose_i.block<3, 3>(0, 3) = Qleft(Qp_.inverse() * Qi).bottomRightCorner<3, 3>();
79  #endif
80      jacobian_pose_i.block<3, 3>(3, 0) = Mat33::Identity(); //Jp
81
82      jacobians_[0] = jacobian_pose_i;
83      //    std::cout << jacobian_pose_i << std::endl;
84  }
85
86  } // namespace backend
87  } // namespace myslam
```

题目四 添加prior约束

深蓝学院 shenlanxueyuan.com

添加第一帧和第二帧的Prior约束

TestMonoBA.cpp × edge_prior.h

app > TestMonoBA.cpp > ...

```
#include "backend/problem.h"

#include "backend/edge_prior.h"
#include <sophus/se3.hpp>

using namespace myslam::backend;
using namespace std;

int main()
{
    // 准备数据
    vector<Frame> cameras;
    vector<Eigen::Vector3d> points;
    GetSimDataInWordFrame(cameras, points);
    Eigen::Quaterniond qic(1, 0, 0, 0);
    Eigen::Vector3d tic(0, 0, 0);

    // 构建 problem
    Problem problem(Problem::ProblemType::SLAM_PROBLEM);

    // 所有 Pose
    vector<shared_ptr<VertexPose>> vertexCams_vec;
    for (size_t i = 0; i < cameras.size(); ++i)
    {…
    }

    /* ********** 前两帧位姿先验项 ********** */
    //先验项权重系数
    double weight = 1e6; //// 0 3 10 100 500 5000 1e4 1e5 1e6 1e7 1e8 1e9 1e10
    for (size_t k = 0; k < 2; ++k)
    {
        //构建边（残差、雅可比）
        shared_ptr<EdgeSE3Prior> edge_prior(new EdgeSE3Prior(cameras[k].twc, cameras[k].qwc));

        //将第一帧和第二帧的含噪声的估计pose设置为先验的状态量顶点
        std::vector<std::shared_ptr<Vertex>> edge_prior_vertex;
        edge_prior_vertex.push_back(vertexCams_vec[k]);
        //顶点与边相连
        edge_prior->SetVertex(edge_prior_vertex);

        // 设置权重
        edge_prior->SetInformation(edge_prior->Information() * weight);
        //将边添加到problem中
        problem.AddEdge(edge_prior);
    }
    // 所有 Point 及 edge
    std::default_random_engine generator;
```

设计不同Prior权重，BA求解精度和速度的比较

权重	迭代次数	求解时间(ms)	RMSE Translation	RMSE Rotation
0	3	1.08324	0.00261831	0.00301854
3	3	0.645179	0.00678568	0.000992244
10	3	0.638841	0.00315454	0.000519869
100	3	0.630379	0.00375711	0.000575194
500	3	0.928295	0.00422012	0.000623944
5000	4	0.776276	0.00435992	0.000657869
1.00E+04	4	0.799114	0.00400621	0.000631276
1.00E+05	5	1.05863	0.0031826	0.000582881
1.00E+06	3	0.596518	0.000188488	0.000256018
1.00E+07	3	0.611311	1.05E-05	2.30E-05
1.00E+08	3	0.616097	2.51836E-07	5.75E-07
1.00E+09	3	0.62331	2.93E-09	6.74E-09
1.00E+10	3	0.627971	2.98E-11	6.86E-11

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening