深蓝学院 VIO 第五次课程作业

温焕宇

2019.7.20

1 基础题：题一

1. **完成单目 Bundle Adjustment 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。**

1.1 完成 Problem::MakeHessian() 中信息矩阵 H 的计算

答：完成此题首先了解矩阵块的操作：

1）Eigen 库中矩阵块操作有两种方法，定义为：

```
matrix.block(i, j, p, q);  // (1)

matrix.block<p, q>(i, j);  // (2)
```

定义（1）表示返回从矩阵的 i 行、j 列开始，每行取 p 个元素，每列取 q 个元素所组成的临时新矩阵对象，原矩阵的元素不变。

定义（2）中 block（p, q）可理解为一个 p 行 q 列的子矩阵，该定义表示从原矩阵中第 i 行、j 列开始，获取一个 p 行 q 列的子矩阵，返回该子矩阵组成的临时矩阵对象，原矩阵的元素不变。

本文使用的是第一种方式：index-i、index-j 表示从 H 矩阵的第 i、j 行开始，dim-i、dim-j 表示 i 行、j 列的维度即元素的个数。当 i=j 时为 H 对角线元素，当 i!=j 时，为 H 非对角线元素。因此写作如下：

```
        MatXX hessian = JtW * jacobian_j; // hessian矩阵 J_i^T * \Sigma^{-1} * J_j
        // 所有的信息矩阵叠加起来
        // TODO:: home work. 完成 H index 的填写.
        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(index_j, index_i, dim_j, dim_i).noalias() += hessian.transpose();
        }
```

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

答：根据舒尔补

$$\begin{bmatrix} \mathbf{H}_{pp} & \mathbf{H}_{pm} \\ \mathbf{H}_{mp} & \mathbf{H}_{mm} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{pp} \\ \delta\mathbf{x}_{ll} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{pp} \\ \mathbf{b}_{mm} \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \mathbf{I} & -\mathbf{H}_{pm}\mathbf{H}_{mm}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{pp} & \mathbf{H}_{pm} \\ \mathbf{H}_{mp} & \mathbf{H}_{mm} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{pp} \\ \delta\mathbf{x}_{ll} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{H}_{pm}\mathbf{H}_{mm}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{pp} \\ \mathbf{b}_{mm} \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \mathbf{H}_{pp} - \mathbf{H}_{pm}\mathbf{H}_{mm}^{-1}\mathbf{H}_{mp} & \mathbf{0} \\ \mathbf{H}_{mp} & \mathbf{H}_{mm} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{pp} \\ \delta\mathbf{x}_{ll} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{pp} - \mathbf{H}_{pm}\mathbf{H}_{mm}^{-1}\mathbf{b}_{mm} \\ \mathbf{b}_{mm} \end{bmatrix} \tag{3}$$

$$(\mathbf{H}_{pp} - \mathbf{H}_{pm}\mathbf{H}_{mm}^{-1}\mathbf{H}_{mp})\delta\mathbf{x}_{pp} = \mathbf{b}_{pp} - \mathbf{H}_{pm}\mathbf{H}_{mm}^{-1}\mathbf{b}_{mm} \tag{4}$$

由以上得到 $\delta\mathbf{x}_{pp}$，然后代入式（3）求得 $\delta\mathbf{x}_{ll}$。

$$\delta\mathbf{x}_{ll} = \mathbf{H}_{mm}^{-1}(\mathbf{b}_{mm} - \mathbf{H}_{mp}\delta\mathbf{x}_{pp}) \tag{5}$$

根据以上 H 矩阵块的位置，代入下述代码：

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

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

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

2 基础题：题二

❷ 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。**

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

答:

同理修改代码如下:

```
// 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(idx, 0, reserve_size - idx - dim, reserve_size) = temp_botRows;
H_marg.block(reserve_size - dim, 0, dim, reserve_size) = temp_rows;
```

```
// TODO:: home work. 完成舒尔补操作
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);

Eigen::MatrixXd tempB = Arm * Amm_inv;
Eigen::MatrixXd H_prior = Arr - tempB * Amr;
```

编译完运行得到结果如下图:

```
0 order: 0
1 order: 6
2 order: 12

 ordered_landmark_vertices_ size : 20
iter: 0 , chi= 5.35099 , Lambda= 0.00597396
iter: 1 , chi= 0.0283713 , Lambda= 0.00199132
iter: 2 , chi= 0.000137971 , Lambda= 0.000663774
problem solve cost: 1.27446 ms
   makeHessian cost: 0.675663 ms

Compare MonoBA results after opt...
after opt, point 0 : gt 0.220938 ,noise 0.227057 ,opt 0.22107
after opt, point 1 : gt 0.234336 ,noise 0.314411 ,opt 0.23442
after opt, point 2 : gt 0.142336 ,noise 0.129703 ,opt 0.142479
after opt, point 3 : gt 0.214315 ,noise 0.278486 ,opt 0.214553
after opt, point 4 : gt 0.130629 ,noise 0.130064 ,opt 0.130629
after opt, point 5 : gt 0.191377 ,noise 0.167501 ,opt 0.191651
after opt, point 6 : gt 0.166836 ,noise 0.165906 ,opt 0.16701
after opt, point 7 : gt 0.201627 ,noise 0.225581 ,opt 0.202057
after opt, point 8 : gt 0.167953 ,noise 0.155846 ,opt 0.168019
after opt, point 9 : gt 0.21891 ,noise 0.209697 ,opt 0.219008
after opt, point 10 : gt 0.205719 ,noise 0.14315 ,opt 0.205668
after opt, point 11 : gt 0.127916 ,noise 0.122109 ,opt 0.127888
after opt, point 12 : gt 0.167904 ,noise 0.143334 ,opt 0.167982
after opt, point 13 : gt 0.216712 ,noise 0.18526 ,opt 0.216997
after opt, point 14 : gt 0.180009 ,noise 0.184249 ,opt 0.180006
after opt, point 15 : gt 0.226935 ,noise 0.245716 ,opt 0.227189
after opt, point 16 : gt 0.157432 ,noise 0.176529 ,opt 0.157615
after opt, point 17 : gt 0.182452 ,noise 0.14729 ,opt 0.182438
after opt, point 18 : gt 0.155701 ,noise 0.182258 ,opt 0.155825
after opt, point 19 : gt 0.14646 ,noise 0.240649 ,opt 0.146651
------------ pose translation ----------------
translation after opt: 0 :0 0 0 || gt: 0 0 0
translation after opt: 1 : -1.0718        4 0.866025 || gt:  -1.0718        4 0.866025
translation after opt: 2 :-4.00002  6.92819 0.866027 || gt:       -4   6.9282 0.866025
---------- TEST Marg: before marg------------
     100     -100        0
    -100  136.111 -11.1111
       0 -11.1111  11.1111
---------- TEST Marg: 将变量移动到右下角-----------
     100        0     -100
       0  11.1111 -11.1111
    -100 -11.1111  136.111
---------- TEST Marg: after marg------------
 26.5306 -8.16327
-8.16327  10.2041
```

3 提高题

提升题

paper reading[a]，请总结论文：优化过程中处理 H 自由度的不同操作方式。总结内容包括：具体处理方式，实验效果，结论。

[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.

3.1 总结论文: 优化过程中处理 H 自由度的不同操作方式，总结内容包括：具体处理方式，实验效果，结论

答：

大纲:

章 II：介绍了基于优化的视觉惯性状态估计问题及其唯一解

章 III：列举了不同的处理 H 自由度的方法

章 IV：描述仿真步骤

章 V：给出精度/时间和协方差方面的详细比较

章 VI：结果

II：基于优化的视觉惯性状态估计问题及其唯一解

视觉惯性融合估计问题：

$$J(\boldsymbol{\theta}) \doteq \underbrace{\|\mathbf{r}^V(\boldsymbol{\theta})\|^2_{\Sigma_V}}_{\text{Visual}} + \underbrace{\|\mathbf{r}^I(\boldsymbol{\theta})\|^2_{\Sigma_I}}_{\text{Inertial}}, \tag{1}$$

以上可以用完全平滑或者固定滞后平滑方法求解，但是由于 VI 状态估计问题的不确定性和不可观性，导致没有足够的方程来确定唯一解。

III：处理 H 自由度的方法

TABLE I: Three gauge handling approaches considered. ($n = 9N + 3K$ is the number of parameters in (2))

	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, $n \times n$

Fixed gauge: 在更小的可观的状态参数空间优化

Gaugae prior: 通过产生可逆 Hessian 来增加目标函数以满足某些约束

Free gauge: 可以使用 single Hessian 的伪逆来隐式的提供额外的约束（具有最小范数的参数更新）用于唯一解

IV：仿真分析

A、数据生成：使用 3 条 6DoF 轨迹，包括类圆弧、正弦和矩形。一组是由随机生成 3D 点产生的正弦轨迹；另一组是由分布在两个平面 3D 点生成圆弧轨迹。为了生成惯性测量数据，用 B 样条曲线拟合这个轨迹，这个值加上 bias 和高斯噪声，对于视觉测量，通过针孔模型相机投影 3D 点的得到图像坐标，并且添加高斯噪声。

B、优化求解器：使用 Ceres 中的 LM 求解。状态向量中包括关键帧和 3D 点的位置、朝向、速度。初始状态由 Groudtruth 随机分布???。

C、评价标准：

1）精度：首先计算一个变换（由两者轨迹的初始位姿确定，由于重力朝向确定，所以这个变换是 4 自由度的）使估计值和真值对齐。之后，再计算所有关键帧的均方根误差（RMSE）。具体就是使用欧氏距离表示位置和速度误差。对于旋转估计，计算对齐旋转和真值之间的相对旋转（利用角轴形式），并且使用相对旋转角度来表示旋转误差。

2）效率：记录融合时间和迭代次数。每个配置（轨迹和点的组合）运行 50 次，并且计算平均时间和精确度。

3）协方差：估计的协方差为 Hessian 矩阵的逆。For the free gauge approach, the Moore-Penrose pseudoinverseis used, since the Hessian is singula。

V：结果对比：时间、精度、协方差等

A、精度和时间结果对比

1）三种方法的结果精度几乎相同

2）在 gauge prior 方法中，需要选择合适的先验权重以避免增加计算成本，在适当的权重下，gauge prior 方法几乎和 gauge fixation 方法达到相同的精度和计算成本

3）free gauge 方法由于需要更少的迭代次数收敛，故比其它方法更快

B、协方差结果对比（待写）

本文并不能直接以一种有意义的方式解释三种协方差。但是，free gauge 的协方差可以精确的线性变换到 gauge fixation 的协方差矩阵，对于优化类似黑盒子问题计算协方差比较有用（例如那些不能直接从雅可比矩阵计算协方差或者 H 矩阵的逆）。

VI：结论

总结两点就是：free gauge approach 计算时间更短迭代次数更少，但三种方法精度几乎相同，计算协方差方式不同。并且阐述了如何变换 free gauge 的协方差以满足 gauge fixation 条件，表明不同方法的协方差密切相关。