深蓝学院 VIO 课程第五课作业

基础题

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。
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

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.

1. 完成 problem.cc 中的部分代码

参考第三课的作业、第四课与本课的 PPT 资料和公式，补全 problem.cc 中 TODO 部分的代码如下所示，详情可见附件 problem.cc 文件。程序运行输出如图 1 所示。

```
279  void Problem::MakeHessian() {
...  ...
314    // 所有的信息矩阵叠加起来
315    // TODO:: home work. 完成 H index 的填写.
316    H.block(index_i, index_j, dim_i, dim_j).noalias() += hessian;
317      if (j != i) {
318        // 对称的下三角
319        // TODO:: home work. 完成 H index 的填写.
320        H.block(index_j, index_i, dim_j, dim_i).noalias() +=
     hessian.transpose();
321      }
...  ...
348  void Problem::SolveLinearSystem() {
...  ...
366    // TODO:: home work. 完成矩阵块取值，Hmm，Hpm，Hmp，bpp，bmm
367    MatXX Hmm = Hessian_.block(reserve_size, reserve_size, marg_size,
     marg_size);
368    MatXX Hpm = Hessian_.block(0, reserve_size, reserve_size, marg_size);
```

```
369    MatXX Hmp = Hessian_.block(reserve_size, 0, marg_size, reserve_size);
370    VecX bpp = b_.segment(0, reserve_size);
371    VecX bmm = b_.segment(reserve_size, marg_size);
...    ...
381    // TODO:: home work. 完成舒尔补 Hpp, bpp 代码
382    MatXX tempH = Hpm * Hmm_inv;
383    H_pp_schur_ = Hessian_.block(0,0,reserve_size,reserve_size) - tempH*Hmp;
384    b_pp_schur_ = bpp - tempH * bmm;
...    ...
398    // TODO:: home work. step3: solve landmark
399    VecX delta_x_ll(marg_size);
400    delta_x_ll = Hmm_inv * (bmm - Hmp * delta_x_pp);
401    delta_x_.tail(marg_size) = delta_x_ll;
402    ...
```

应用程序输出

testMonoBA

```
23:38:06: Starting /home/vance/vio_ws/src/vio_homework_code/L5/BA_schur/build/app/testMonoBA ...
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= 0.0289048 , Lambda= 0.00199132
iter: 2 , chi= 0.000109162 , Lambda= 0.000663774
problem solve cost: 19.0242 ms
   makeHessian cost: 15.8027 ms

Compare MonoBA results after opt...
after opt, point 0 : gt 0.220938 ,noise 0.227057 ,opt 0.220992
after opt, point 1 : gt 0.234336 ,noise 0.314411 ,opt 0.234854
after opt, point 2 : gt 0.142336 ,noise 0.129703 ,opt 0.142666
after opt, point 3 : gt 0.214315 ,noise 0.278486 ,opt 0.214502
after opt, point 4 : gt 0.130629 ,noise 0.130064 ,opt 0.130562
after opt, point 5 : gt 0.191377 ,noise 0.167501 ,opt 0.191892
after opt, point 6 : gt 0.166836 ,noise 0.165906 ,opt 0.167247
after opt, point 7 : gt 0.201627 ,noise 0.225581 ,opt 0.202172
after opt, point 8 : gt 0.167953 ,noise 0.155846 ,opt 0.168029
after opt, point 9 : gt 0.21891 ,noise 0.209697 ,opt 0.219314
after opt, point 10 : gt 0.205719 ,noise 0.14315 ,opt 0.205995
after opt, point 11 : gt 0.127916 ,noise 0.122109 ,opt 0.127908
after opt, point 12 : gt 0.167904 ,noise 0.143334 ,opt 0.168228
after opt, point 13 : gt 0.216712 ,noise 0.18526 ,opt 0.216866
after opt, point 14 : gt 0.180009 ,noise 0.184249 ,opt 0.180036
after opt, point 15 : gt 0.226935 ,noise 0.245716 ,opt 0.227491
after opt, point 16 : gt 0.157432 ,noise 0.176529 ,opt 0.157589
after opt, point 17 : gt 0.182452 ,noise 0.14729 ,opt 0.182444
after opt, point 18 : gt 0.155701 ,noise 0.182258 ,opt 0.155769
after opt, point 19 : gt 0.14646 ,noise 0.240649 ,opt 0.14677
------------ pose translation ----------------
translation after opt: 0 :-0.00047801  0.00115904 0.000366507 || gt: 0 0 0
translation after opt: 1 :-1.06959  4.00018 0.863877 || gt:  -1.0718        4 0.866025
translation after opt: 2 :-4.00232  6.92678 0.867244 || 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
23:38:07: /home/vance/vio_ws/src/vio_homework_code/L5/BA_schur/build/app/testMonoBA exited with code 0
```

图 1 testMonoBA 程序运行结果截图

2. 完成滑动窗口算法测试函数

补充的代码如下所示，详情可见附件 problem.cc 文件。程序运行输出同样如上页图 1 所示。

```
554  void Problem::TestMarginalize() {
...  ...
572      // TODO:: home work. 将变量移动到右下角
573      /// 准备工作： move the marg pose to the Hmm bottown right
574      // 将 row i 移动矩阵最下面
575      Eigen::MatrixXd temp_rows = H_marg.block(idx, 0, dim, reserve_size);
576      Eigen::MatrixXd temp_botRows = H_marg.block(idx + dim, 0, reserve_size
     - idx - dim, reserve_size);
577      H_marg.block(idx, 0, reserve_size - idx - dim, reserve_size) =
     temp_botRows;
578      H_marg.block(reserve_size - dim, 0, dim, reserve_size) = temp_rows;
     ...
600      // TODO:: home work. 完成舒尔补操作
601      Eigen::MatrixXd Arm = H_marg.block(0,n2,n2,m2);
602      Eigen::MatrixXd Amr = H_marg.block(n2,0,m2,n2);
603      Eigen::MatrixXd Arr = H_marg.block(0,0,n2,n2);
604
605      Eigen::MatrixXd tempB = Arm * Amm_inv;
606      Eigen::MatrixXd H_prior = Arr - tempB * Amr;
...  ...
```

3. 论文总结

VIO 有四个不可观的自由度，优化的时候需要特别处理，这篇论文就这个问题进行了详细讨论。通常有三种方法：①是固定这四个自由度，②是给这四个自由度加先验，③是任意优化这四个自由度再 reset。三种方法的对比如表一所示，这里先给出结论：

- 三种方法的精度相当；
- Gauge Prior 法需要选择一个合适的先验权重，以避免计算开销上升；
- Gauge Prior 法在合适权重的情况下，精度和计算效率可以跟 Fixed Gauge 法相当；
- Free Gauge 法计算速度比另外两种方法略快，因为迭代次数更少；
- Free Gauge 法不需要对旋转参数做特殊处理；

表 1 VIO 协方差矩阵对四个不可观自由度的处理方法对比

方法名称	具体处理方式	实验效果对比
Fixed Gauge	在小参数空间中优化，使其状态量均可观，则 H 可逆。优化时固定第一个相机的位置和 yaw 角。	● 变量维度最少，单次迭代耗时最短 ● 精度相当
Gauge Prior	增加目标函数额外的误差项，满足确定性的约束，使 H 可逆，目标函数变为： $J(\boldsymbol{\theta}) \doteq \underbrace{\|\mathbf{r}^V(\boldsymbol{\theta})\|^2_{\Sigma_V}}_{\text{Visual}} + \underbrace{\|\mathbf{r}^I(\boldsymbol{\theta})\|^2_{\Sigma_I}}_{\text{Inertial}} + \|\mathbf{r}_0^P\|^2_{\Sigma_0^P}$ 其中$\Sigma_0^P = \sigma_0^2 \mathrm{I}$	● 单次迭代耗时中等 ● 精度相当
Free Gauge	用奇异 H 的伪逆提供额外约束获得唯一解。 在优化过程中让参数向量自由变化。	● 单次迭代耗时最长 ● 迭代次数最少 ● 总耗时最快 ● 精度相当

综上来看，Free Gauge 方法应该是最好的。但是 Free Gauge 方法不足之处在于他没有一个参考系，信息矩阵的逆得到的协方差没有太多的意义，可以参考引文[14]中的方法解决。将 Free Gauge 法的协方差通过线性变换成 Fixed Gauge 的协方差形式，这就解决了这个方法的不足。