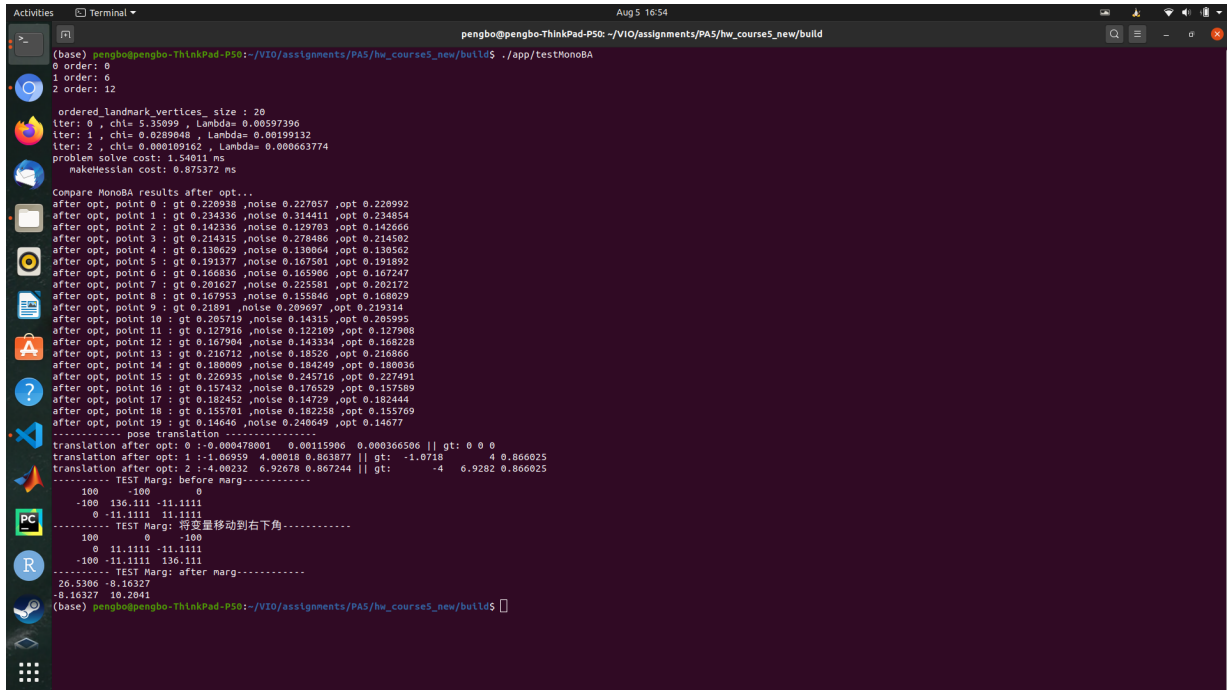


# 从零开始手写 VIO - 作业 5

peng00bo00

August 10, 2020

1. 补全代码后程序运行截图可参见 Fig.1



```
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA5/hw_course5_new/build$ ./app/testMonoBA
0 order: 0
1 order: 6
2 order: 12

ordered_landmark_vertices_size : 20
iter: 0 , chl= 5.35099 , Lambda= 0.00597396
iter: 1 , chl= 0.0289048 , Lambda= 0.00199132
iter: 2 , chl= 0.000102462 , Lambda= 0.000603774
problem solve cost: 1.54011 ms
makeHessian cost: 0.075372 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 0.142660
after opt, point 3 : gt 0.214315 ,noise 0.270406 ,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 0.160836 ,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 0.209097 ,opt 0.219314
after opt, point 10 : gt 0.205719 ,noise 0.143315 ,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 0.216722 ,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 0.102452 ,noise 0.10729 ,opt 0.102444
after opt, point 18 : gt 0.155701 ,noise 0.102258 ,opt 0.155709
after opt, point 19 : gt 0.14646 ,noise 0.240649 ,opt 0.14677

----- pose translation -----
translation after opt: 0 :-0.000478001 0.00115906 0.000365006 || gt: 0 0 0
translation after opt: 1 :-1.06959 4.00018 0.063877 || gt: -1.0718 4 0.066025
translation after opt: 2 :-4.00232 0.92678 0.007244 || gt: -4 0.9282 0.066025
----- TEST Marg: before marg-----
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
-100 11.1111 -11.1111
-100 -11.1111 136.111
----- TEST Marg: after marg-----
26.5306 -8.16327
-8.16327 10.2041
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA5/hw_course5_new/build$
```

Figure 1: 程序截图

2. 论文对比了 3 种不同的处理 H 矩阵方式:

- (a) gauge fixation: 固定一部分待优化参数, 令 Jacobbian 矩阵的相应部分为 0
- (b) gauge prior: 在优化目标函数中额外添加一项  $\|r_0^p\|_\Sigma^2$  来施加约束
- (c) free gauge: 使用 Hessian 矩阵的伪逆或是 LM 算法等来求解

实验结果显示 3 种方法在精度上基本没有差别; 使用 gauge prior 需要设置好先验权重  $\Sigma$ , 合适的权重可以避免增加计算开销; 使用合适的先验权重时 gauge prior 方法和 gauge fixation 方法有类似的计算效率和精度; free gauge 方法比另外 2 种方法要快一一点, 这主要是因为这种方法的迭代次数相对小一些。

3. 给第 1 帧和第 2 帧的 Hessian 矩阵加上先验项  $wI_n$  作为 gauge prior，其中  $w$  为先验项权重。使用不同权重  $w$  运行程序得到结果如所示 Table.1, Fig.2, Fig.3。结果显示不同的权重会对计算迭代次数以及最后的误差产生影响：当  $w$  趋于 0 时 gauge prior 接近于 free guage 方法；而当  $w$  比较大时效果则类似与 gauge fixation 方法；实验中当  $w > 1e6$  时算法的迭代次数和误差基本收敛。

Table 1: 实验对比

权重 $w$	迭代次数	误差
1e-5	2	0.0123678
1e-4	2	0.0123862
1e-3	3	0.0136865
1e-2	4	0.0185144
1e-1	4	0.0213409
1	3	0.0172074
1e1	4	0.0124609
1e2	2	0.0072258
1e3	2	0.00791445
1e4	3	0.0075842
1e5	4	0.00643306
1e6	2	0.00222307
1e7	2	0.0019815
1e8	2	0.00197189
1e9	2	0.00197171
1e10	2	0.0019717

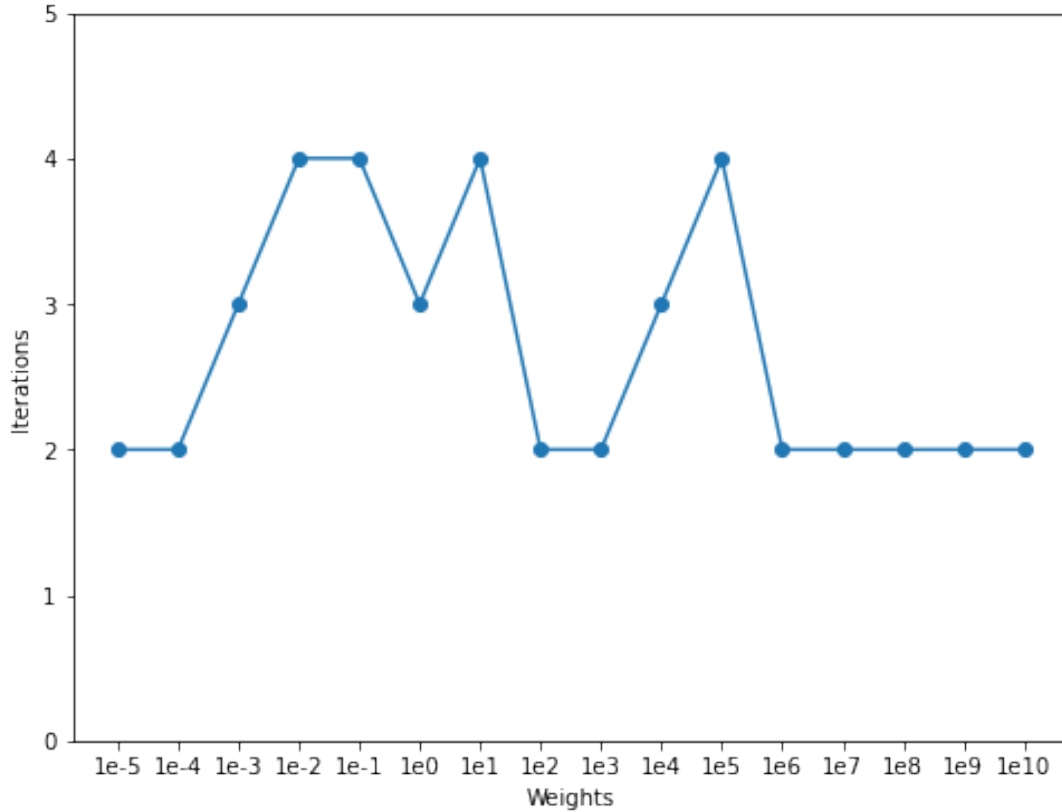


Figure 2: 迭代次数曲线

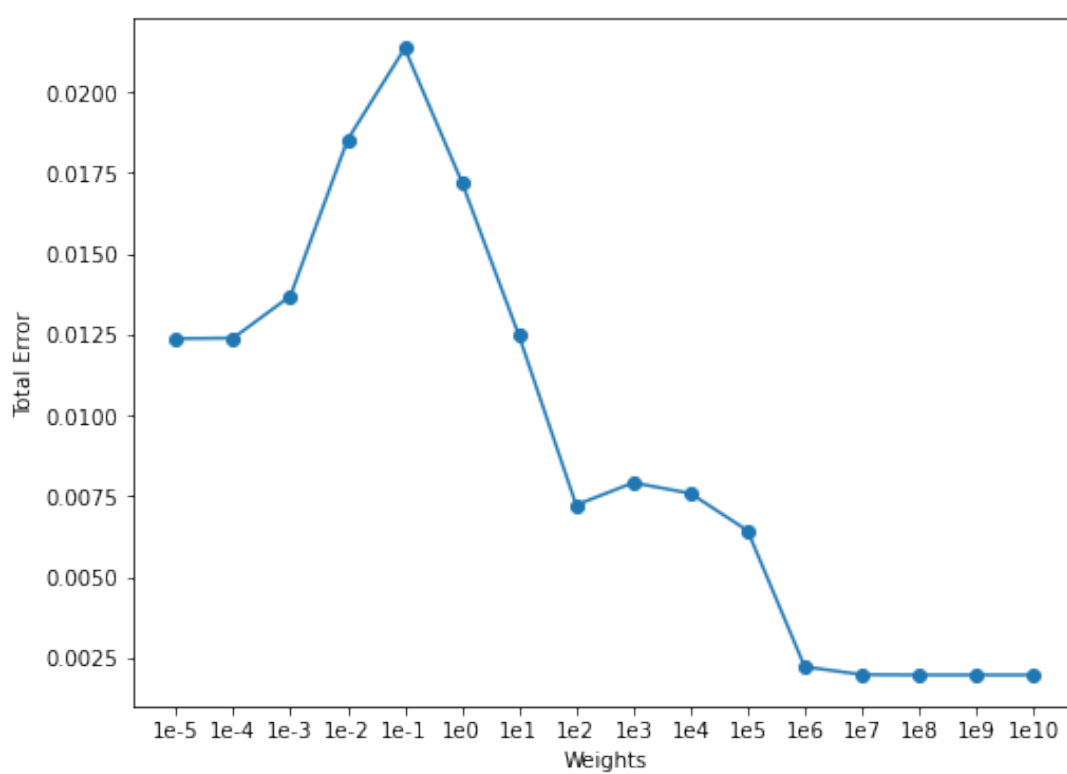


Figure 3: 误差曲线