基础题：

1.完成单目 Bundle Adjustment 求解器 problem.cc 中的部分代码。

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

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

解：1）.首先阅读一下代码把握全局：

a. Problem::MakeHessian()的信息矩阵 H

```
// 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();
}
```

b. Problem::SolveLinearSystem()求解代码

- 第一步：

```
// 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);
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);
```

- 第二步：

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

- 第三步：

```
// 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. 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。

.Problem::TestMarginalize() 代码补全

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

// 将 col i 移动矩阵最右边
Eigen::MatrixXd temp_cols = H_marg.block(0, idx, reserve_size, dim);
Eigen::MatrixXd temp_rightCols = H_marg.block(0, idx + dim, reserve_size, reserve_size - idx - dim);
H_marg.block(0, idx, reserve_size, reserve_size - idx - dim) = temp_rightCols;
H_marg.block(0, reserve_size - dim, reserve_size, dim) = temp_cols;

std::cout << "---------- TEST Marg: 将变量移动到右下角------------" << std::endl;
std::cout << H_marg << std::endl;

/// 开始 marg ： schur
double eps = 1e-8;
int m2 = dim;
int n2 = reserve_size - dim; // 剩余变量的维度
Eigen::MatrixXd Amm = 0.5 * (H_marg.block(n2, n2, m2, m2) + H_marg.block(n2, n2, m2, m2).transpose());

Eigen::SelfAdjointEigenSolver<Eigen::MatrixXd> saes(Amm);
Eigen::MatrixXd Amm_inv = saes.eigenvectors() * Eigen::VectorXd((saes.eigenvalues().array() > eps).select(saes.
                          saes.eigenvectors().transpose();

// TODO:: home work. 完成舒尔补操作
Eigen::MatrixXd Arm = H_marg.block(0,2,2,1);
Eigen::MatrixXd Amr = H_marg.block(2,0,1,2);
Eigen::MatrixXd Arr = H_marg.block(0,0,2,2);

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

std::cout << "---------- TEST Marg: after marg------------" << std::endl;
std::cout << H_prior << std::endl;
```

最后运行通过：

同时验证是否 fixed 前两个位姿：

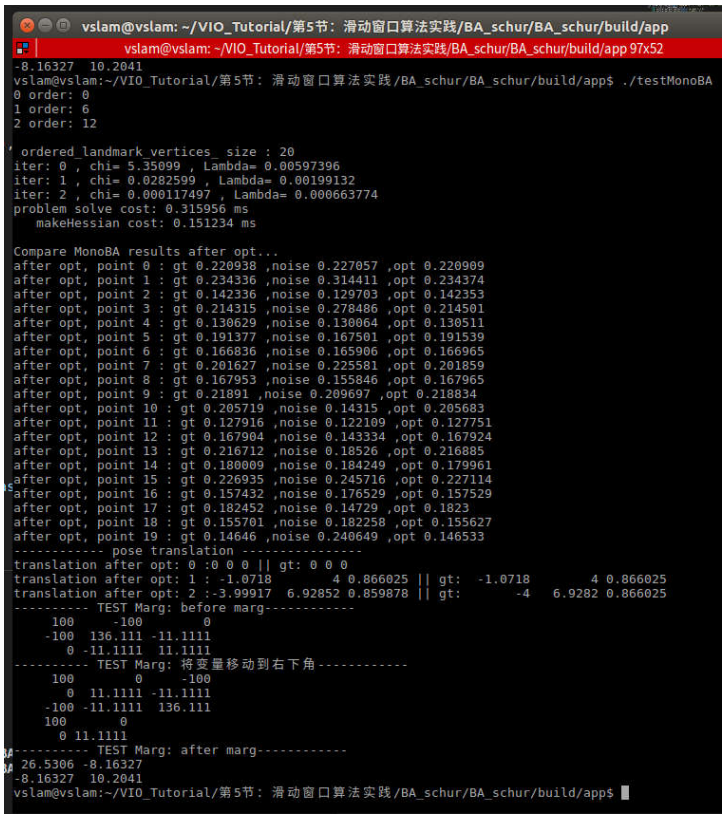
```
if(i < 2)
vertexCam->SetFixed();
```

见下页图：

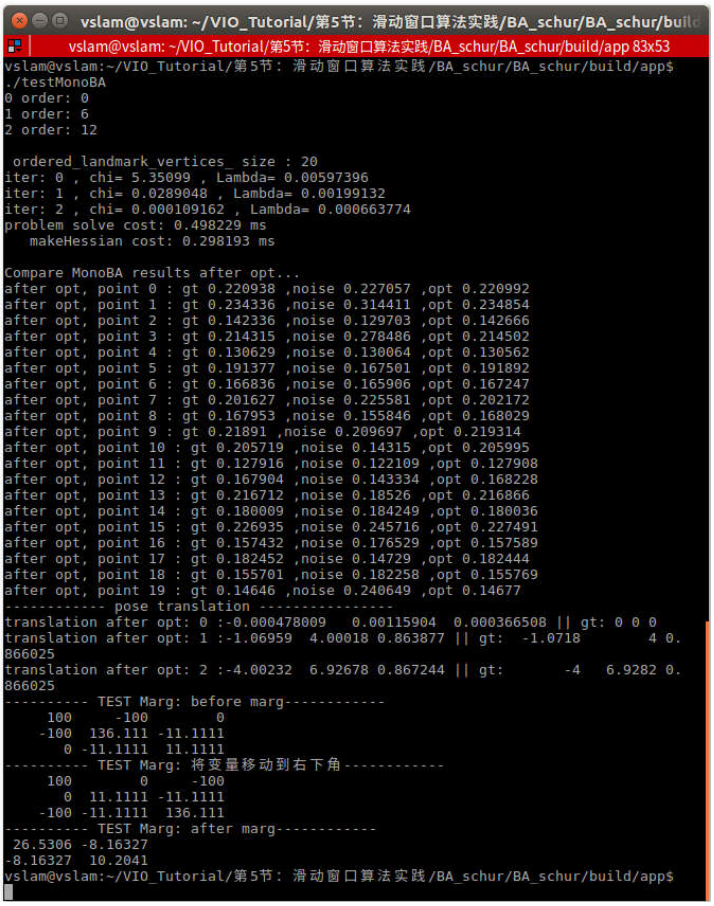
结论：

如果不固定的话，初始位置有偏移，后面的优化变量也有偏移。

a 固定：


vslam@vslam: ~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/BA_schur/build/app
vslam@vslam: ~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/BA_schur/build/app 97x52
-8.16327 10.2041
vslam@vslam:~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/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= 0.0282599 , Lambda= 0.00199132
iter: 2 , chi= 0.000117497 , Lambda= 0.000663774
problem solve cost: 0.315956 ms
makeHessian cost: 0.151234 ms
Compare MonoBA results after opt...
after opt, point 0 : gt 0.220938 ,noise 0.227057 ,opt 0.220909
after opt, point 1 : gt 0.234336 ,noise 0.314411 ,opt 0.234374
after opt, point 2 : gt 0.142336 ,noise 0.129703 ,opt 0.142353
after opt, point 3 : gt 0.214315 ,noise 0.278486 ,opt 0.214501
after opt, point 4 : gt 0.130629 ,noise 0.130064 ,opt 0.130511
after opt, point 5 : gt 0.191377 ,noise 0.167501 ,opt 0.191539
after opt, point 6 : gt 0.166836 ,noise 0.165906 ,opt 0.166965
after opt, point 7 : gt 0.201627 ,noise 0.225581 ,opt 0.201859
after opt, point 8 : gt 0.167953 ,noise 0.155846 ,opt 0.167965
after opt, point 9 : gt 0.21891 ,noise 0.209697 ,opt 0.218834
after opt, point 10 : gt 0.205719 ,noise 0.14315 ,opt 0.205683
after opt, point 11 : gt 0.127916 ,noise 0.122109 ,opt 0.127751
after opt, point 12 : gt 0.167904 ,noise 0.143334 ,opt 0.167924
after opt, point 13 : gt 0.216712 ,noise 0.18526 ,opt 0.216885
after opt, point 14 : gt 0.180009 ,noise 0.184249 ,opt 0.179961
after opt, point 15 : gt 0.226935 ,noise 0.245716 ,opt 0.227114
after opt, point 16 : gt 0.157432 ,noise 0.176529 ,opt 0.157529
after opt, point 17 : gt 0.182452 ,noise 0.14729 ,opt 0.1823
after opt, point 18 : gt 0.155701 ,noise 0.182258 ,opt 0.155627
after opt, point 19 : gt 0.14646 ,noise 0.240649 ,opt 0.146533
------------ 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 :-3.99917 6.92852 0.859878 || 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
100 0
0 11.1111
---------- TEST Marg: after marg------------
26.5306 -8.16327
-8.16327 10.2041
vslam@vslam:~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/BA_schur/build/app$

b 不固定：


vslam@vslam: ~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/BA_schur/build
vslam@vslam: ~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/BA_schur/build/app 83x53
vslam@vslam:~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/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: 0.498229 ms
makeHessian cost: 0.298193 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 :-0.000478009 0.00115904 0.000366508 || 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
vslam@vslam:~/VIO_Tutorial/第5节：滑动窗口算法实践/BA_schur/BA_schur/build/app$

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。

文献题目为；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.

文章提出：

针对基于优化视觉惯性状态估计, 作者针对 gauge freedom（翻译为规范自由度）展示了三种不同的处理方式，分别是 Fixed gauge(固定不客观变量的状态)，Gauge prior（在某些状态上设置先验值），和 Free gauge（在优化中状态自由变化）。

第一部分：三种不同处理方式：

方法	处理方式
Fixed gauge	gauge fixation 采用的方式是在优化中固定 yaw 角和第一个相机的位置，采用的方式的是 $p_0 = p_0^0,\ \Delta\phi_{0z} = e_z^T \Delta\phi_0 = 0$ 等价于 $J_0 = 0,\quad J\Delta\phi_{0z} = 0$
Gauge prior	The *gauge prior* approach 增加一个惩罚系数 $\left\|r_0^P\right\|^2_{\sum_0^P}$
Free gauge	*free gauge* approach 采用在优化算法中优化向量自由变化，为了处理奇异 H 矩阵，采用伪逆或增加阻尼。

三种方式的迭代形式方式：

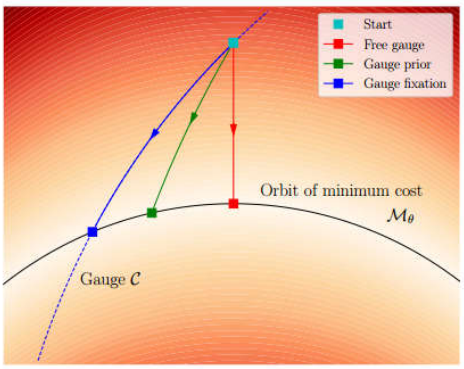


Fig. 2: Illustration of the optimization paths taken by different gauge handling approaches. The gauge fixation approach always moves on the gauge $\mathcal{C}$, thus satisfying the gauge constraints. The free gauge approach uses the pseudoinverse to select parameter steps of minimal size for a given cost decrease, and therefore, moves perpendicular to the isocontours of the cost (1). The gauge prior approach follows a path in between the gauge fixation and free gauge approaches. It minimizes a cost augmented by (11), so it may not exactly end up on the orbit of minimum visual-inertial cost (1).

第二部分：仿真实验设置

1)数据生成：

Pose： sine, arc and rec

Landmark:

采用两种路标点，平面点，分布在几个平面上 3D 点，和随机点，沿着轨迹的形成的点

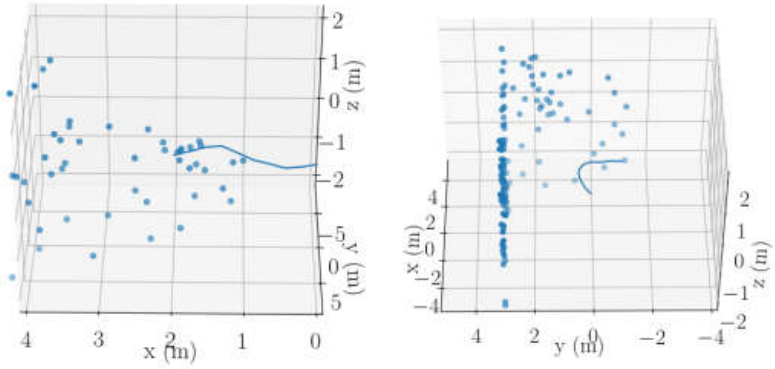


Fig. 3: Sample simulation scenarios. The left one shows a *sine* trajectory with randomly generated 3D points, and the right one shows an *arc* trajectory with the 3D points distributed on two planes.

IMU: 从 B 样条上获取 imu 数据;

2)评估方法

Evaluation（准确度）

Computational Efficiency（计算代价）

Covariance（协方差）

3.仿真实验

A.Gauge Prior:：(选择合适的优先权重，如下图 4 和 5)

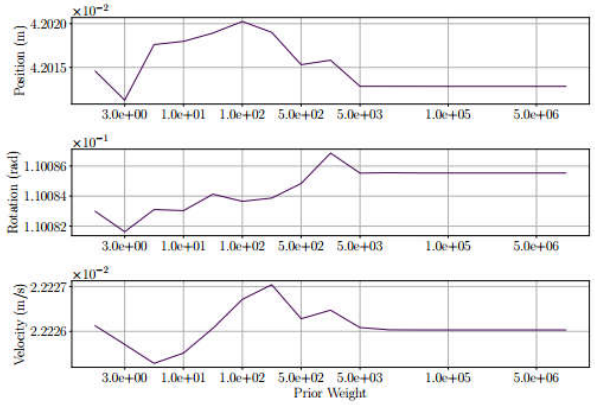


Fig. 4: RMSE in position, orientation and velocity for different prior weights

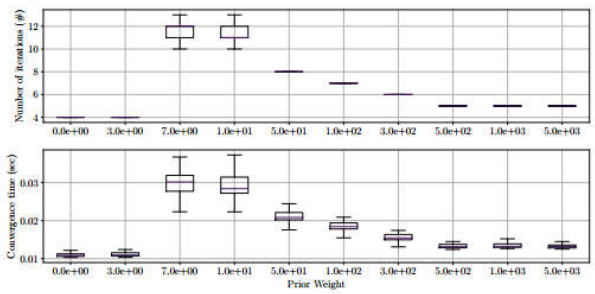


Fig. 5: Number of iterations and computing time for different prior weights.

图 4 可以看出不同的先验权重 prior weights 对准确度影响不大，但是图 5 需要，需要悬着合适的先验权重保持计算代价小。大的权重有时会使优化不稳定。

B. 准确度和计算代价

实验在 6 种方式比较三种方式的表现。

TABLE II: RMSE on different trajectories and 3D points configurations. The smallest errors (e.g., $\mathbf{p}$ gauge fixation vs. $\mathbf{p}$ free gauge) are highlighted.

Configuration	Gauge fixation			Free gauge		
	$\mathbf{p}$	ϕ	$\mathbf{v}$	$\mathbf{p}$	ϕ	$\mathbf{v}$
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.

表二，可以看出 50 次实验得平均误差，free gauge approach 和 the gauge fixation approach 有较小差异，而且他们都没有较好的准确度

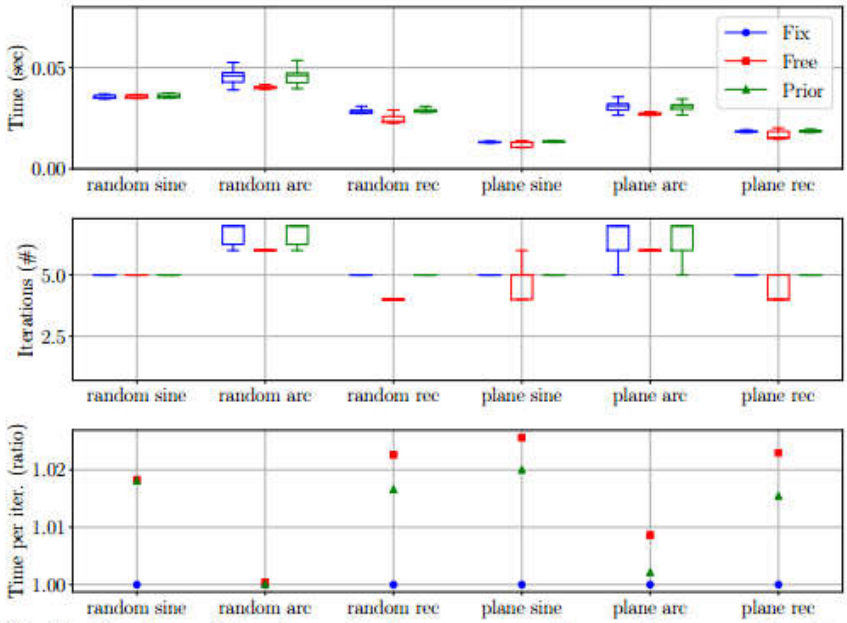


Fig. 7: Number of iterations, total convergence time and time per iteration for all configurations. The time per iteration is the ratio with respect to the gauge fixation approach (in blue), which takes least time per iteration.

从图 7 可以看出收敛时间和迭代次数， gauge prior approach 和 gauge fixation approach 计算时间几乎是一样的。free gauge approach 是稍微比其他的两种计算快。特别的，除了带有随机点和 sine 曲线，free gauge approach 花费较少的迭代次数和较少的时间，注明 gauge fixation approach 花费较少的时间因为较少的优化变量。

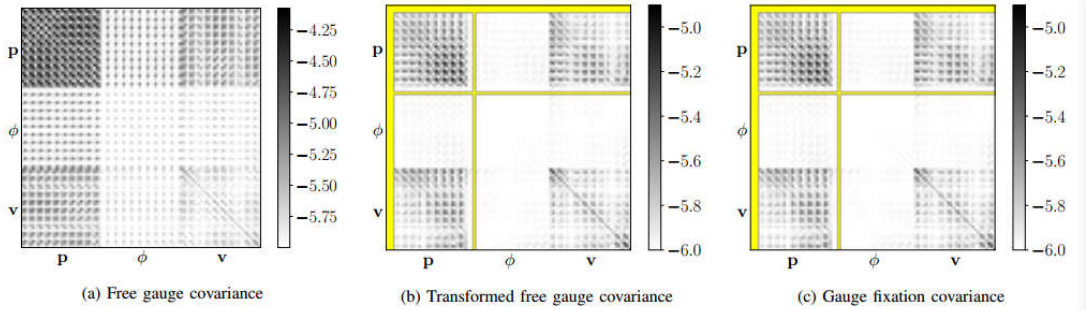
C. 讨论

1) 三种处理方式有相同的准确度。
2) **在** In the gauge prior approach 中，需要选择合适的权重来避免计算代价
3) 采用合适的权重，In the gauge prior approach 和 the gauge fixation approach. 在准确度和计算代价上有相同的表现

4) The free gauge approach 相对其他两种方式，应为较少的迭代次数

3．协方差

在给定先验权重的情况下，gauge prior approach 和 gauge fixation approach 的协方差矩阵是相似的，他们和 the free gauge approach 不同，如图 9.a 和 9.c，但是他们可以通过某种方式转化。



(a) Free gauge covariance (b) Transformed free gauge covariance (c) Gauge fixation covariance

Fig. 9: Covariance of free gauge (Fig. 9a) and gauge fixation (Fig. 9c) approaches using $N = 10$ keyframes. In the middle (Fig. 9b), the free gauge covariance transformed using (12) shows very good agreement with the gauge fixation covariance: the relative difference between them is $\|\Sigma_b - \Sigma_c\|_F / \|\Sigma_c\|_F \approx 0.11\%$ ($\|\cdot\|_F$ denotes Frobenius norm). For better visualization, the magnitude of the covariance entries is displayed in logarithmic scale. The yellow bands of the gauge fixation and transformed covariances indicate zero entries due to the fixed 4-DoFs (the position and the yaw angle of the first camera).

第三部分：实物实验

实物实验时在数据集 *Machine Hall 1* (MH1) 和 *Vicon Room 1*(VI1)进行相同的实验方式。采用 SVO 在优化中采用初始化参数，用于算法消除尺度问题。在全轨迹中不开启优化。

1) 计算代价如，图 11，计算代价和实物一样，The free gauge approach 相对其他两种方式，应为较少的迭代次数

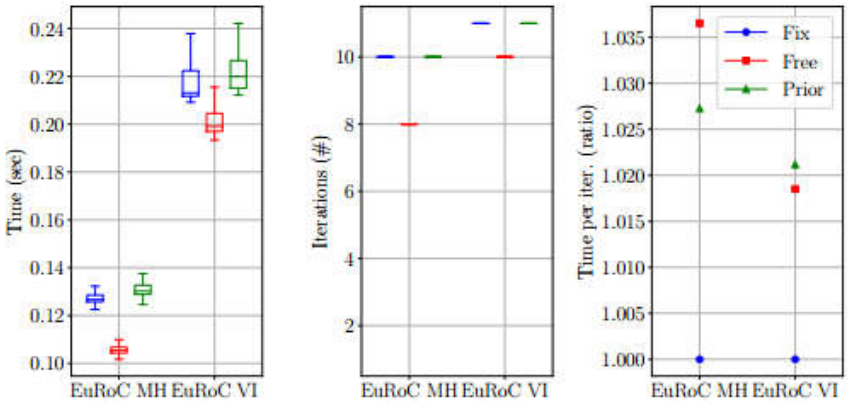


Fig. 11: Computational cost of the three different methods for handling gauge freedom on two sequences from the EuRoC dataset. The time per iteration is the ratio with respect to the gauge fixation approach.

准确性的比较报告在表三，三种处理方式有相同的估计误差

TABLE III: RMSE on EuRoC datasets. Same notation as in Table II.

Sequence	Gauge fixation			Free gauge		
	p	ϕ	v	p	ϕ	v
EuRoC MH	0.06936	**0.07845**	0.03092	**0.06918**	0.07857	**0.03091**
EuRoC VI	0.07851	0.4382	0.04644	0.07851	0.4382	0.04644

第四部分：总结

针对基于优化视觉惯性状态估计,作者针对 gauge freedom（翻译为规范自由度）展示了三种不同的处理方式，分别是 Fixed gauge(固定不客观变量的状态)，Gauge prior（在某些状态上设置先验值），和 Free gauge（在优化中状态自由变化）。

1. 在仿真和实物中有相同的准确性和效率（计算代价），同时 Free gauge 有较快的计算代价，因为较小的迭代次数，
2. 主要差异是在优化算法的时候，估算的协方差是不同的