第五讲-习题

姜帆

2019 年 7 月 30 日

目录

1 代码实践

1.1 完善求解器代码

1.1.1 Problem::MakeHessian

补充信息矩阵 H 的计算。

```
298         assert(jacobians.size() == vertices.size());
299         for (size_t i = 0; i < vertices.size(); ++i) {
300             auto v_i = vertices[i];
301             if (v_i->IsFixed()) continue;    // Hessian 里不需要添加它的信息，也就是它的雅克比为 0
302 
303             auto jacobian_i = jacobians[i];
304             ulong index_i = v_i->OrderingId();
305             ulong dim_i = v_i->LocalDimension();
306 
307             MatXX JtW = jacobian_i.transpose() * edge.second->Information();
308             for (size_t j = i; j < vertices.size(); ++j) {
309                 auto v_j = vertices[j];
310 
311                 if (v_j->IsFixed()) continue;
312 
313                 auto jacobian_j = jacobians[j];
314                 ulong index_j = v_j->OrderingId();
315                 ulong dim_j = v_j->LocalDimension();
316 
317                 assert(v_j->OrderingId() != -1);
318                 MatXX hessian = JtW * jacobian_j;
319                 // 所有的信息矩阵叠加起来
320                 // TODO:: home work. 完成 H index 的填写.
321                 // H.block(?,?, ?, ?).noalias() += hessian;
322                 H.block(index_i,index_j,dim_i,dim_j).noalias() += hessian;
323                 if (j != i) {
324                     // 对称的下三角
325                     // TODO:: home work. 完成 H index 的填写.
326                     // H.block(?,?, ?, ?).noalias() += hessian.transpose();
327                     H.block(index_j,index_i,dim_j,dim_i).noalias() += hessian.transpose();
328                 }
329             }
330             b.segment(index_i, dim_i).noalias() -= JtW * edge.second->Residual();
331         }
332 
```

图 1: 信息矩阵 H 计算补充

1.1.2 Problem::SolveLinearSystem()

补充求解器代码。

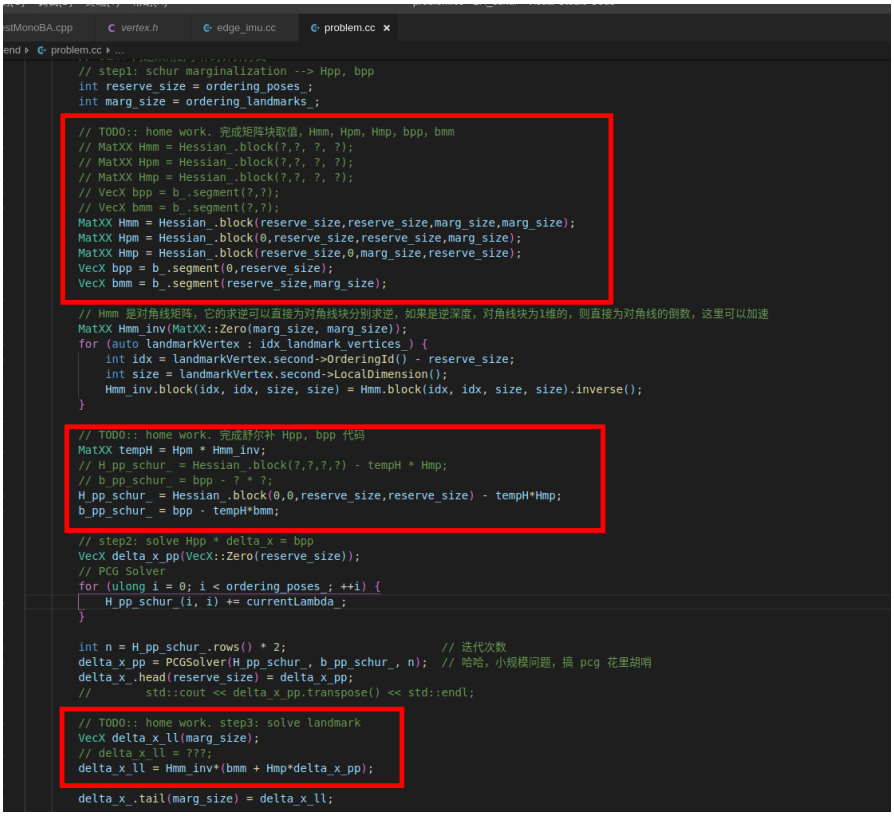

图 2: 求解器代码补充

BA 求解结果 (未固定第一个相机位姿):

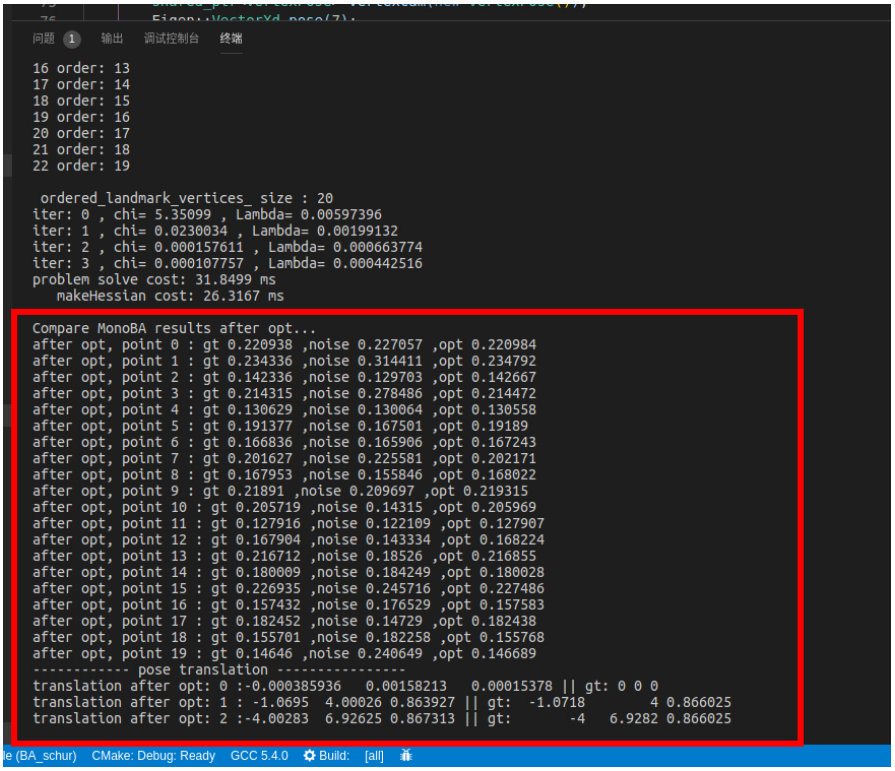

图 3: 程序运行结果

固定第一个相机的位姿后，得到结果：

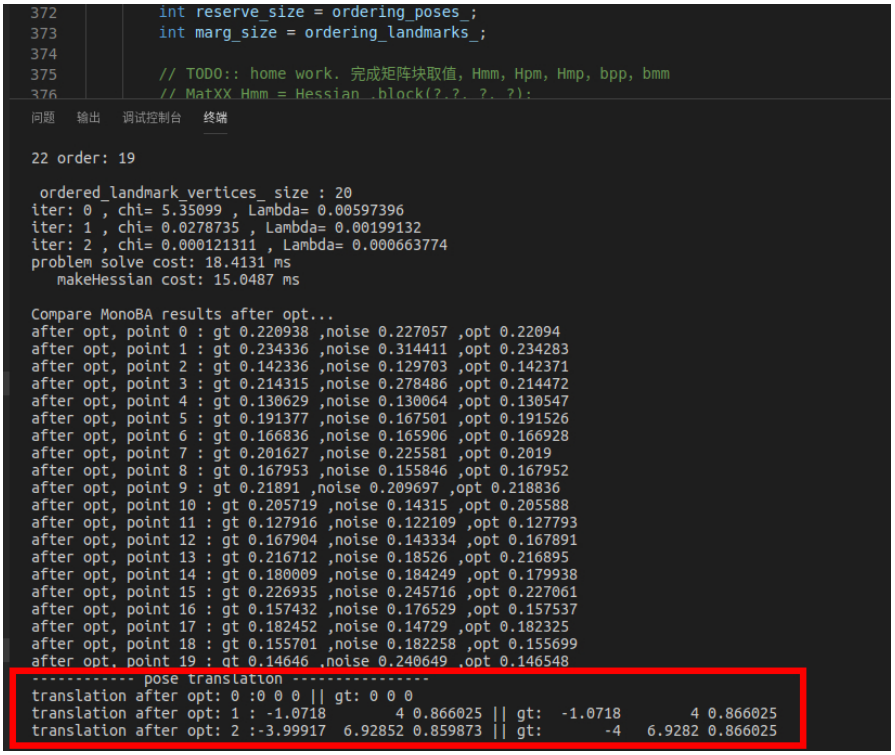

图 4: 固定第一个相机的位姿

1.2 完善滑动窗口代码

完善滑动窗口代码：

```
int cols = 3;
MatXX H_marg(MatXX::Zero(cols, cols));
H_marg << 1./delta1, -1./delta1, 0,
          -1./delta1, 1./delta1 + 1./delta2 + 1./delta3, -1./delta3,
          0.,  -1./delta3, 1/delta3;
std::cout << "---------- TEST Marg: before marg-----------"<< std::endl;
std::cout << H_marg << std::endl;

// 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;

// 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.eigenvalues().array().inverse(), 0)).asDiagonal() *
                          saes.eigenvectors().transpose();

// 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);

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

图 5: 滑动窗口代码补全

marg 第二个变量后的信息矩阵变为：

```
问题 1   输出   调试控制台   终端

after opt, point 10 : gt 0.205719 ,noise 0.14315 ,opt 0.205969
after opt, point 11 : gt 0.127916 ,noise 0.122109 ,opt 0.127907
after opt, point 12 : gt 0.167904 ,noise 0.143334 ,opt 0.168224
after opt, point 13 : gt 0.216712 ,noise 0.18526 ,opt 0.216855
after opt, point 14 : gt 0.180009 ,noise 0.184249 ,opt 0.180028
after opt, point 15 : gt 0.226935 ,noise 0.245716 ,opt 0.227486
after opt, point 16 : gt 0.157432 ,noise 0.176529 ,opt 0.157583
after opt, point 17 : gt 0.182452 ,noise 0.14729 ,opt 0.182438
after opt, point 18 : gt 0.155701 ,noise 0.182258 ,opt 0.155768
after opt, point 19 : gt 0.14646 ,noise 0.240649 ,opt 0.146689
------------ pose translation ----------------
translation after opt: 0 :-0.000385936   0.00158213   0.00015378 || gt: 0 0 0
translation after opt: 1 : -1.0695  4.00026 0.863927 || gt:  -1.0718        4 0.866025
translation after opt: 2 :-4.00283  6.92625 0.867313 || gt:       -4   6.9282 0.866025

---------Test Marginal----------
---------- 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
[1] + Done                       /usr/bin/gdb --interpreter=mi --tty=${DbgTerm} 0</tmp/Microso
jiangfan@jiangfan:~/vio_course/ch5/BA_schur$

(BA_schur)  CMake: Debug: Ready  GCC 5.4.0  Build:  [all]
```

图 6: 滑动窗口运行结果

2 论文阅读

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

视觉惯性里程计有四个自由度，包括三个平移和一个 Z 方向的旋角。一般有三种处理这三个自由度的方法：（1）gauge fixation: 固定第一个相机的位姿中不客观的四个量；（2）gauge prior: 添加先验约束, 增加系统的可观性；（3）free gauge：令不可观的状态量在优化过程中不受约束，然后优化介绍后统一转换。文章经过实验得出如下结论：三种方法精度和计算视觉基本相同，free gauge approch 较其他两种方法稍微快一点，free gauge 的协方差估计与前两种方式表现不同，但可以通过转换与前两种方法产生联系。

2.1 视觉惯导问题构建

视觉惯性里程计（VIO）的目标函数可以写为：

$$\begin{aligned} J(\theta) &= \left\|r^V(\theta)\right\|^2_{\Sigma_V} + \left\|r^I(\theta)\right\|^2_{\Sigma_I} \\ \theta &= \{p_i, R_i, v_i, X_j\} \end{aligned} \tag{1}$$

式（1）中 $J(\theta)$ 第一项为视觉重投影误差目标函数，第二项为 imu 测量误差目标函数，文章使用预积分误差。

2.1.1 状态不可观

目标函数（1）对于 $\theta' = g(\theta)$ 的转换是不变量：

$$J(\theta) = J(g(\theta)) \tag{2}$$

$$g = \begin{bmatrix} R_z & t \\ 0 & 1 \end{bmatrix} \tag{3}$$

因此以上说明使得（1）式成立的解不唯一，也就是说存在一个参数空间 M_θ 使得目标函数都可以达到最小。

$$M_\theta = \{g(\theta) | g \in G\} \tag{4}$$

2.1.2 增加约束

因此需要为参数空间增加一个约束，使得目标函数最小有唯一解。

$$c(\theta) = 0 \tag{5}$$

例如选择第一帧相机位移作为参考坐标系原点，并将第一帧相机的 yaw 设置为 0。

2.2 三种处理方法

2.2.1 gauge fixation

在整个优化过程中固定第一帧相机的位移和 yaw：

$$p_0 = p_0^0, \qquad \delta\phi_{0z} = e_x^T \delta_0 = 0 \tag{6}$$

实际操作中将其对应的 Jacobian 设置为 0:

$$J_{p_0} = 0, \qquad J_{\delta\phi_{0z}=0} \tag{7}$$

2.2.2 gauge prior

添加先验约束相当于给目标函数增加一个惩罚项:

$$\left\|r_0^P\right\|_{\Sigma_0^p}^2, \qquad where \quad r_0^P(\theta) = (p_0 - p_0^0, \delta\phi_{0z}) \tag{8}$$

我们需要自己选择先验协方差 Σ_0^p。一种常用的选择方式是 $\Sigma_0^p = \sigma_0^2 I$，因此先验变成 $\left\|r_0^P\right\|_{\Sigma_0^p}^2 = w^P \left\|r_0^P\right\|^2$, 其中 $w^P = 1/\sigma_0^2$。当 $w^P = 0$ 时 gauge prior 等效与 free gauge approch，当 $w^P \to \infty$ 则等效于 fixation gauge。对于先验的选取，作者做了一系列实验，实验发现先验值选取的不同并不会大致最后解的精度的大幅度变化，但是合理的选择先验能够使得计算代价变小。因此文章通过实验选择了先验权重为 10^5，以便于后面的实验。

2.2.3 free gauge

free gauge 方法使得参数空间在优化过程中不受约束，但是为了处理奇异的 Hession 矩阵，这里采用伪逆进行求解。

对 H 进行奇异值分解:

$$\begin{aligned} H &= UDV^T \\ H^+ &= VD^+U^T \end{aligned} \tag{9}$$

下面是三种方法的一个总结以及示意图:

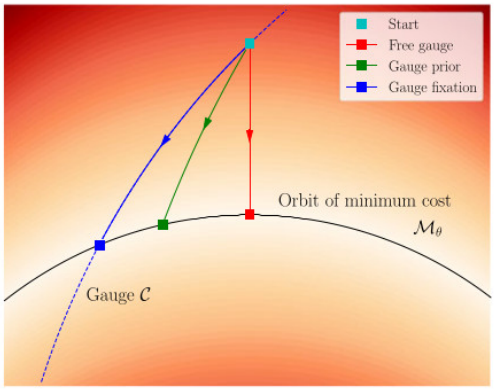

Fig. 2. Illustration of the optimization paths taken by different gauge handling approaches. The gauge fixation approach always moves on the gauge $\mathcal{C}$, thus satisfying the gauge constraints. The free gauge approach uses the pseudoinverse to select parameter steps of minimal size for a given 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).

TABLE I
THREE GAUGE HANDLING APPROACHES CONSIDERED)

	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, $n \times n$

($n = 9N + 3K$ is the number of parameters in (2))

2.3 实验部分

因为 fixation gauge 和 gauge prior 方法效果相同，因此只对 fixation gauge 和 free gauge 方法进行比较。

2.3.1 仿真数据

仿真数据精度比较结果：

TABLE II
RMSE ON DIFFERENT TRAJECTORIES AND 3D POINTS CONFIGURATIONS

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**

The smallest errors (e.g., p gauge fixation vs. p free gauge) are highlighted.
Position, rotation and velocity RMSE are measured in m, deg and m/s, respectively.

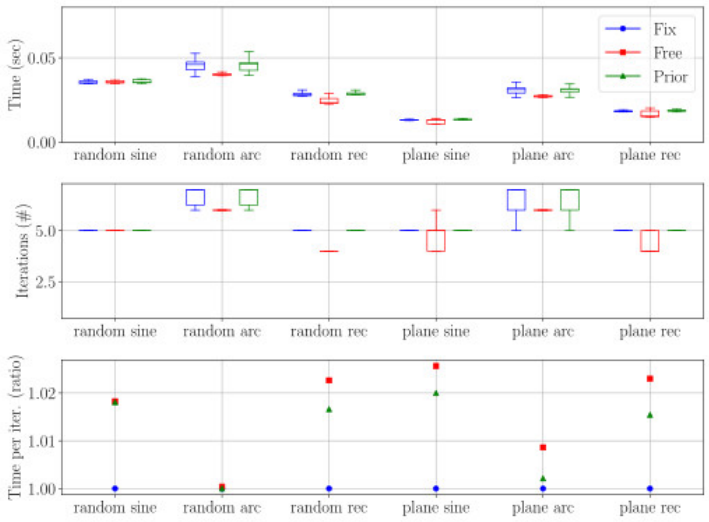

Fig. 7. Number of iterations, total convergence time and time per iteration for all configurations. The time per iteration is the ratio with respect to the gauge fixation approach (in blue), which takes least time per iteration.

对于协方差来说，free gauge 的协方差与其他两种方法并没有一个明显的几何意义：

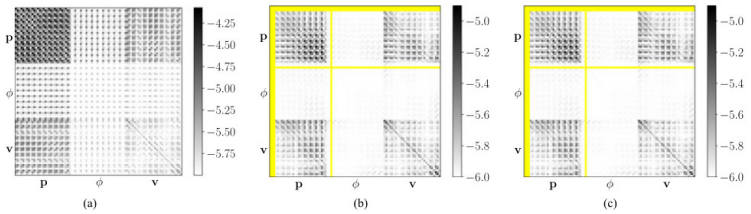

Fig. 9. Covariance of free gauge (Fig. 9) and gauge fixation (Fig. 9) approaches using $N = 10$ keyframes. In the middle (Fig. 9), the free gauge covariance transformed using (12) shows very good agreement with the gauge fixation covariance: the relative difference between them is $\|\Sigma_b - \Sigma_c\|_F / \|\Sigma_c\|_F \approx 0.11\%$ ($\|\cdot\|_F$ 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). (a) Free gauge covariance. (b) Transformed free gauge covariance. (c) Gauge fixation covariance.

但是经过一定的转换后，可以将其与其他两种方法联系起来：

$$Cov(\theta_C) \approx= (Q_{\theta_C}^C \frac{\partial \theta_C}{\partial \theta}) Cov^*(\theta) (Q_{\theta_C}^C \frac{\partial \theta_C}{\partial \theta}) \tag{10}$$

协方差效果图，如图：

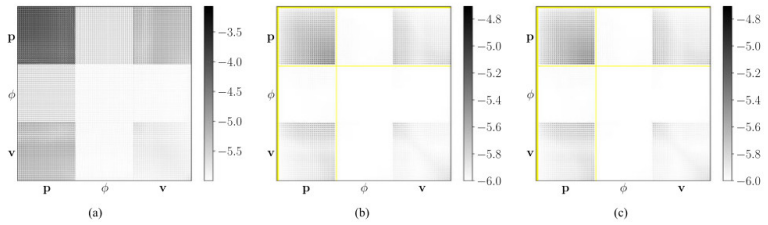

Fig. 10. Covariance comparison and transformation using $N = 30$ keyframes of the EuRoC Vicon 1 sequence (VI1). Same color scheme as in Fig. 9. The relative difference between (b) and (c) is $\|\Sigma_b - \Sigma_c\|_F / \|\Sigma_c\|_F \approx 0.02\%$. Observe that, in the gauge fixation covariance, the uncertainty of the first position and yaw is zero, and it grows for the rest of the camera poses (darker color), as illustrated in Fig. 1(b). (a) Free gauge covariance. (b) Transformed free gauge covariance. (c) Gauge fixation covariance.

2.3.2 真实数据

在真实数据上测试的精度的结果如下： 协方差效果图，如图：

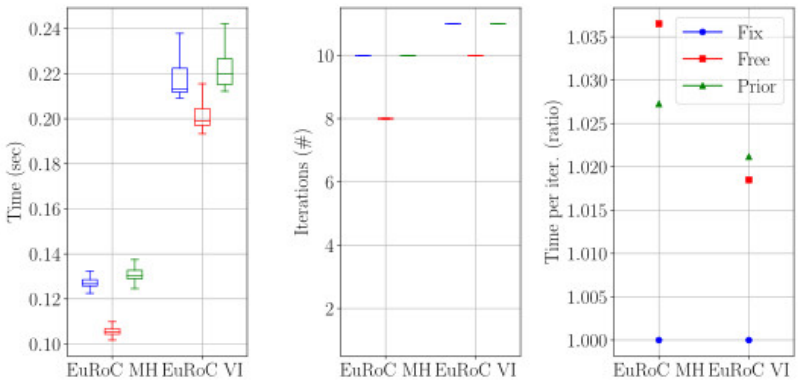

Fig. 11. Computational cost of the three different methods for handling gauge freedom on two sequences from the EuRoC dataset. The time per iteration is the ratio with respect to the gauge fixation approach.

TABLE III
RMSE ON EUROC DATASETS

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

Same notation as in Table II.

2.4 结论

(1) 这三种方法的精度与效率基本相同。

(2) 在 gauge prior 方法中选择合适的先验权重能够减少计算量。并且能够得到同于 fixation gauge 方法的结果。

(3)free gauge 方法比其他两种方法速度略快，原因是因为这种方法的迭代次数更少。

(4)free gauge 的协方差可以通过一定的转换形式得到与 fixation gauge 协方差相同的形式。