深蓝学院
shenlanxueyuan.com

从零手写VIO-第六期
第五次作业 思路讲解

主讲人 吴少腾

第五次作业

作业

基础题

1. 完成单目 Bundle Adjustment (BA) 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

- 请总结论文[a]：优化过程中处理 $\mathbf{H}$ 自由度的不同操作方式。内容包括：具体处理方式，实验效果，结论。(加分题，评选良好)
- 在代码中给第一帧和第二帧添加 prior 约束，并比较为 prior 设定不同权重时，BA 求解收敛精度和速度。(加分题，评选优秀)

[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.

基础作业-1

深蓝学院
shenlanxueyuan.com

- 完成BA求解器problem.cc中的部分代码

Problem::MakeHessian()

- 与第四章作业相似，组装H矩阵

$$Hessian(i,j) = J_i^T w J_j$$

- 注意维度

```
MatXX JtW = jacobian_i.transpose() * edge.second->Information();
for (size_t j = i; j < verticies.size(); ++j) {
    auto v_j = verticies[j];

    if (v_j->IsFixed()) continue;

    auto jacobian_j = jacobians[j];
    ulong index_j = v_j->OrderingId();
    ulong dim_j = v_j->LocalDimension();

    assert(v_j->OrderingId() != -1);
    MatXX hessian = JtW * jacobian_j;
    // 所有的信息矩阵叠加起来
    // TODO:: home work. 完成 H index 的填写.
    // H.block(?,?, ?, ?).noalias() += hessian;
    if (j != i) {
        // 对称的下三角
// TODO:: home work. 完成 H index 的填写.
        // H.block(?,?, ?, ?).noalias() += hessian.transpose();
    }
}
b.segment(index_i, dim_i).noalias() -= JtW * edge.second->Residual();
```

基础作业-1

深蓝学院 shenlanxueyuan.com

Problem::SolveLinearSystem()

- 第一步 Schur 三角化

$$\begin{bmatrix} H_{pp} & H_{pm} \\ H_{mp} & H_{mm} \end{bmatrix} \begin{bmatrix} \Delta x_p^* \\ \Delta x_m^* \end{bmatrix} = \begin{bmatrix} b_p \\ b_m \end{bmatrix}$$

两边同时左乘 $\begin{bmatrix} I & -H_{pm}H_{pp}^{-1} \\ 0 & I \end{bmatrix}$

$$\begin{bmatrix} H_{pp} - H_{pm}H_{pp}^{-1}H_{mp} & 0 \\ H_{mp} & H_{mm} \end{bmatrix} \begin{bmatrix} \Delta x_p^* \\ \Delta x_m^* \end{bmatrix} = \begin{bmatrix} b_p - H_{pm}H_{pp}^{-1}b_m \\ b_m \end{bmatrix}$$

```
    // SLAM 问题采用舒尔补的计算方式
    // step1: schur marginalization --> Hpp, bpp
    int reserve_size = ordering_poses_;
    int marg_size = ordering_landmarks_;

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

    // Hmm 是对角线矩阵，它的求逆可以直接为对角线块分别求逆，如果是逆深度，对角线块为1维的，则直接为对角
    MatXX Hmm_inv(MatXX::Zero(marg_size, marg_size));
    for (auto landmarkVertex : idx_landmark_vertices_) {
        int idx = landmarkVertex.second->OrderingId() - reserve_size;
        int size = landmarkVertex.second->LocalDimension();
        Hmm_inv.block(idx, idx, size, size) = Hmm.block(idx, idx, size, size).inverse();
    }

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

基础作业-1

$$\begin{bmatrix} H_{pp} - H_{pm}H_{pp}^{-1}H_{mp} & 0 \\ H_{mp} & H_{mm} \end{bmatrix} \begin{bmatrix} \Delta x_p^* \\ \Delta x_m^* \end{bmatrix} = \begin{bmatrix} b_p - H_{pm}H_{pp}^{-1}b_m \\ b_m \end{bmatrix}$$

- 第二步 先解

$$\left(H_{pp} - H_{pm}H_{pp}^{-1}H_{mp}\right) \color{red}{\Delta x_p^*} = b_p - H_{pm}H_{pp}^{-1}b_m$$

- 第三步 再解

$$H_{mm}\color{red}{\Delta x_m^*} = b_m - H_{mp}\Delta x_p^*$$

```
// step2: solve Hpp * delta_x = bpp
VecX delta_x_pp(VecX::Zero(reserve_size));
// PCG Solver
for (ulong i = 0; i < ordering_poses_; ++i) {
    H_pp_schur_(i, i) += currentLambda_;
}

int n = H_pp_schur_.rows() * 2;                       // 迭代次数
delta_x_pp = PCGSolver(H_pp_schur_, b_pp_schur_, n);  // 哈哈，小规模问题，搞 pcg 花里胡哨
delta_x_.head(reserve_size) = delta_x_pp;
//        std::cout << delta_x_pp.transpose() << std::endl;

// TODO:: home work. step3: solve landmark
VecX delta_x_ll(marg_size);
// delta_x_ll = ???;
delta_x_.tail(marg_size) = delta_x_ll;
```

基础作业-1

深蓝学院
shenlanxueyuan.com

- 优化结果接近真值，但是由于单目系统具有7个不可观的状态，优化结果在零空间发生漂移

- 可以通过testMono.cc中的如下代码fix前两帧位姿

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

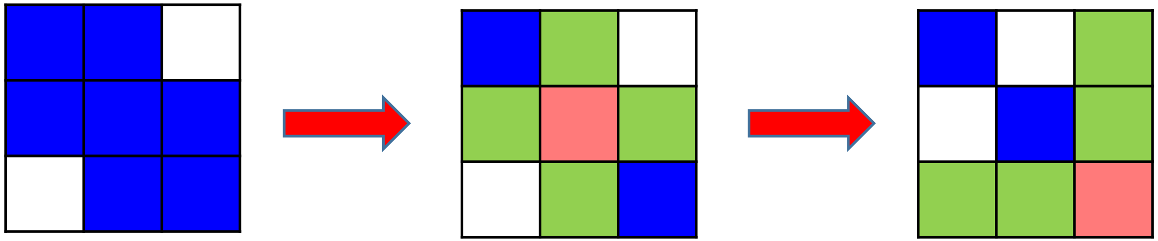
```
wushaoteng@wushaoteng-Inspiron-7559:~/slam/vio/hw5/hw_course5_new/build$ ./app/testMonoBA
0 order: 0
1 order: 6
2 order: 12

 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: 1.68735 ms
   makeHessian cost: 0.990675 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
```

基础作业-2

- 完成滑动窗口算法测试函数　Problem::TestMarginalize()

- 第一步 将被边缘化的变量移到右下角

```cpp
// 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;
```



```
---------- 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
```

基础作业-2

- 第二步 边缘化 Schur补过程与上一题一致

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

```
---------- TEST Marg: after marg------------
 26.5306 -8.16327
-8.16327  10.2041
```

提升作业

Zhang Z, Gallego G, Scaramuzza D. On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation[J]. IEEE Robotics and Automation Letters, 2018, 3(3): 2710-2717.

- 目的

➢ 讨论使用最小二乘法解决VIO问题时处理不可观自由度(Gauge Freedom)的三种处理方法和性能对比

- 需要关注的问题

➢ 三种方法的具体处理方式

➢ 性能对比

提升作业

- global position和global yaw不可观的后果

对于代价函数 $J(\boldsymbol{\theta}) \doteq \underbrace{\|\mathbf{r}^V(\boldsymbol{\theta})\|^2_{\Sigma_V}}_{\text{Visual}} + \underbrace{\|\mathbf{r}^I(\boldsymbol{\theta})\|^2_{\Sigma_I}}_{\text{Inertial}},$

$$J(\boldsymbol{\theta}) = J(g(\boldsymbol{\theta})). \quad \text{where} \quad g \doteq \begin{pmatrix} \mathrm{R}_z & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

$$g(\boldsymbol{\theta}) = \boldsymbol{\theta}' \equiv \{\mathbf{p}'_i, \mathrm{R}'_i, \mathbf{v}'_i, \mathbf{X}'_j\}$$

$$\mathbf{p}'_i = \mathrm{R}_z \mathbf{p}_i + \mathbf{t} \qquad \mathrm{R}'_i = \mathrm{R}_z \mathrm{R}_i$$

$$\mathbf{v}'_i = \mathrm{R}_z \mathbf{v}_i \qquad \mathbf{X}'_j = \mathrm{R}_z \mathbf{X}_j + \mathbf{t}$$

➢ 最优解$\boldsymbol{\theta}^*$，任意对全局z轴的旋转变换和全局平移都不改变代价函数大小，即存在多个最优解

提升作业

深蓝学院
shenlanxueyuan.com

- global position和global yaw不可观的后果

对于代价函数 $J(\boldsymbol{\theta}) \doteq \underbrace{\|\mathbf{r}^V(\boldsymbol{\theta})\|^2_{\Sigma_V}}_{\text{Visual}} + \underbrace{\|\mathbf{r}^I(\boldsymbol{\theta})\|^2_{\Sigma_I}}_{\text{Inertial}},$

$$J(\boldsymbol{\theta}) = J(g(\boldsymbol{\theta})). \quad \text{where} \quad g \doteq \begin{pmatrix} \mathrm{R}_z & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

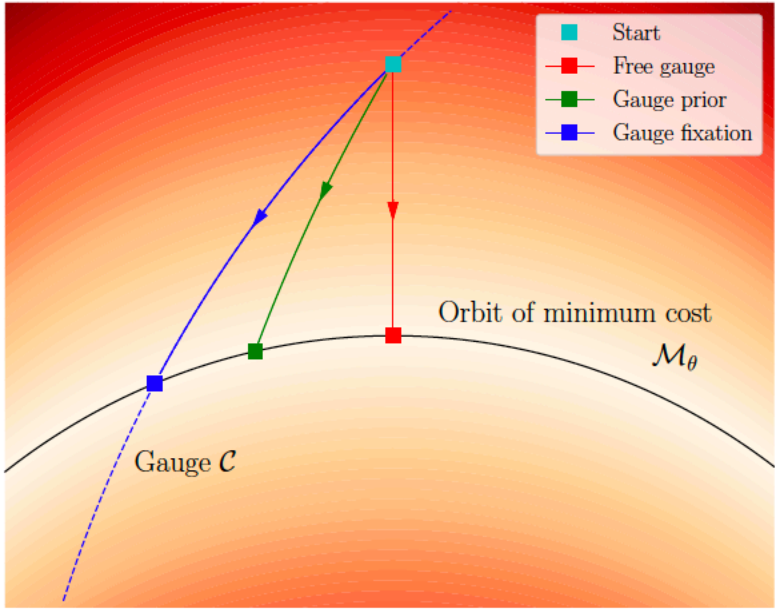
$$g(\boldsymbol{\theta}) = \boldsymbol{\theta}' \equiv \{\mathbf{p}'_i, \mathrm{R}'_i, \mathbf{v}'_i, \mathbf{X}'_j\}$$

$$\begin{aligned} \mathbf{p}'_i &= \mathrm{R}_z \mathbf{p}_i + \mathbf{t} & \mathrm{R}'_i &= \mathrm{R}_z \mathrm{R}_i \\ \mathbf{v}'_i &= \mathrm{R}_z \mathbf{v}_i & \mathbf{X}'_j &= \mathrm{R}_z \mathbf{X}_j + \mathbf{t} \end{aligned}$$

➢ 最优解$\boldsymbol{\theta}^*$，任意对全局z轴的旋转变换和全局平移都不改变代价函数大小，即存在多个最优解

$$\mathcal{M}_{\boldsymbol{\theta}} \doteq \{g(\boldsymbol{\theta}) \mid g \in \mathcal{G}\},$$

提升作业



- 这些最优解可以组成一个具有4自由度的流形 $\mathcal{M}_{\boldsymbol{\theta}} \doteq \{g(\boldsymbol{\theta}) \,|\, g \in \mathcal{G}\},$

提升作业

深蓝学院 shenlanxueyuan.com

- 处理gauge freedom的3种方法

	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

固定第一帧的global position和global yaw。相当于引入了新的测量信息，使原本不可观的状态变得可观。信息矩阵H满秩，可求逆得到唯一最优解。

➢ Free Gauge :

不处理不可观的状态，得到的解会在零空间漂移。信息矩阵H不满秩，可用Moore-Penrose广义逆求解得到范数最小的最小二乘解。

➢ Gauge Prior :

介于上面两个方法之间，改变残差权重，添加额外的信息以处理不可观状态。信息矩阵满秩，可通过求逆得到唯一最优解。

提升作业

- 三种方法的图形化对比

➢ Fixed Gauge

将解限定在某一个流形$\mathcal{C}$上，最优解：

$$\boldsymbol{\theta}_C = \mathcal{C} \cap \mathcal{M}_{\boldsymbol{\theta}}$$

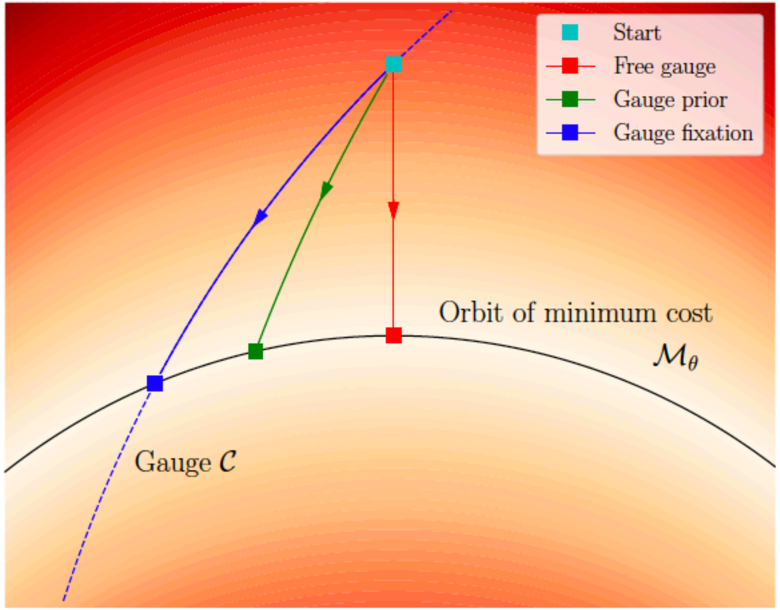
➢ Free Gauge

获得最小范数的最小二乘解 。

“垂线段”最短

➢ Prior Gauge

介于二者之间



提升作业

- Gauge Fixation和Gauge Prior的处理方法
  - 使用LM算法得到增量$\delta\phi^q$，那么第Q次迭代后的旋转量为：

$$\mathrm{R}^Q = \prod_{q=0}^{Q-1} \mathrm{Exp}(\delta\phi^q)\mathrm{R}^0,$$

  - 即使限制$\delta\phi^q$的z轴分量为0，每一次迭代都会在上一次的基础上改变roll和pitch，使得这一帧的z轴与最初的z轴不重合，总的来看还是改变了yaw。
  - 为了更好的限制初始帧的yaw，对初始帧的位姿更新都是对0时刻的更新。

$$\mathrm{R}_0 = \mathrm{Exp}(\Delta\phi_0)\mathrm{R}_0^0,$$

0时刻

初始帧

提升作业

深蓝学院
shenlanxueyuan.com

●Gauge Fixation和Gauge Prior的处理方法

➢ 为了更好的限制初始帧的yaw，对初始帧的位姿更新都是对0时刻的更新。

$$\mathrm{R}_0 = \mathrm{Exp}(\Delta\phi_0)\mathrm{R}_0^0,$$

0时刻

初始帧

➢ Gauge Fixation

$$\mathbf{p}_0 = \mathbf{p}_0^0, \qquad \Delta\phi_{0z} \doteq \mathbf{e}_z^\top \Delta\phi_0 = 0,$$ 等价于对应jocabian置零。

➢ Gauge Prior

$$\|\mathbf{r}_0^P\|_{\Sigma_0^P}^2, \quad \text{where} \quad \mathbf{r}_0^P(\boldsymbol{\theta}) \doteq (\mathbf{p}_0 - \mathbf{p}_0^0,\ \Delta\phi_{0z}).$$ 增加惩罚项，信息矩阵设置权重。

提升作业

深蓝学院
shenlanxueyuan.com

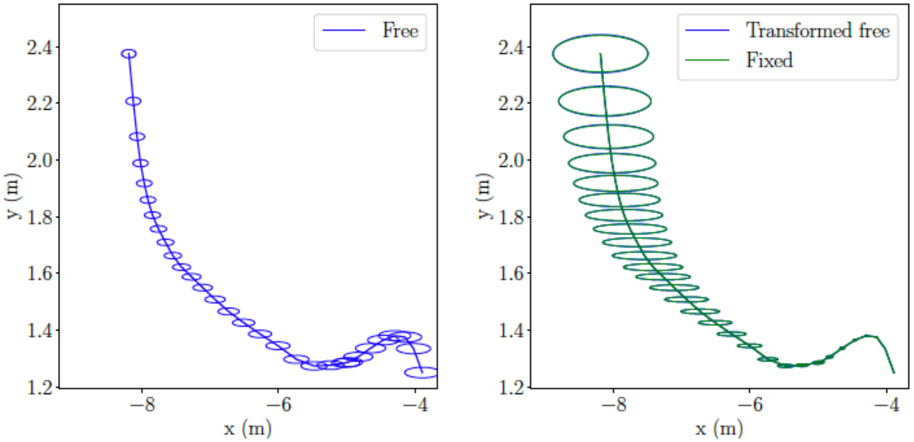
- 性能对比

➢ 评价方式：IV.C
SE3对齐第一帧，用旋转向量的角度差和平移向量的二范数的RMSE评价

➢ 性能：
- 精度
- 适宜的权重
- 迭代次数与时间
- **协方差**

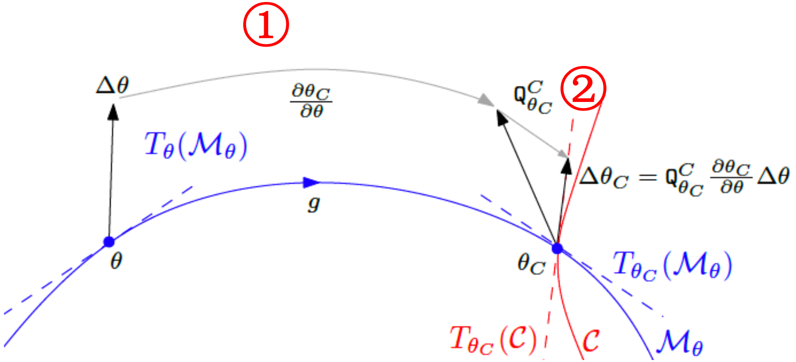
提升作业

- 协方差也需要像评价旋转和平移一样“对齐”



(a) Free gauge approach (b) Gauge fixation approach

➢ Gauge Fixation 不确定性不断增长
➢ Free Gauge 将不确定性“平摊”到每一帧



$$\mathrm{Cov}(\boldsymbol{\theta}_C) \approx \left(\mathtt{Q}^C_{\boldsymbol{\theta}_C}\frac{\partial\boldsymbol{\theta}_C}{\partial\boldsymbol{\theta}}\right)\overset{*}{\mathrm{Cov}}(\boldsymbol{\theta})\left(\mathtt{Q}^C_{\boldsymbol{\theta}_C}\frac{\partial\boldsymbol{\theta}_C}{\partial\boldsymbol{\theta}}\right)^\top$$

➢ 协方差变换后才能对比

① 将θ沿$\mathcal{M}_\theta$对齐到θ_C, $\Delta\theta$变换成$\Delta\theta_C$

② 取$\Delta\theta_C$ 沿流形$\mathcal{C}$在θ_C切空间$T_{\theta_C}(\mathcal{C})$的分量

提升作业

深蓝学院 shenlanxueyuan.com

- 代码实现 Prior Gauge

➢ Gauge Prior

$\|\mathbf{r}_0^P\|^2_{\Sigma_0^P}$, where $\mathbf{r}_0^P(\boldsymbol{\theta}) \doteq (\mathbf{p}_0 - \mathbf{p}_0^0, \Delta\phi_{0z})$. 增加惩罚项，信息矩阵设置权重。

$$r^P = \begin{bmatrix} r_R^P \\ r_p^P \end{bmatrix} = \begin{bmatrix} \Delta\phi_z \\ p - p^0 \end{bmatrix} = \begin{bmatrix} Ln((R^0)^{-1}R) \\ p - p^0 \end{bmatrix}$$

其中R^0和p^0 是设定的先验值， R和p是更新的值省略了下标0.

$$J_{r^P} = \frac{\partial r^P}{\partial \begin{bmatrix} R \\ p \end{bmatrix}} = \begin{bmatrix} \frac{\partial r_R^P}{\partial R} & \frac{\partial r_R^P}{\partial p} \\ \frac{\partial r_p^P}{\partial R} & \frac{\partial r_p^P}{\partial p} \end{bmatrix} = \begin{bmatrix} \frac{\partial r_R^P}{\partial R} & 0 \\ 0 & \frac{\partial r_p^P}{\partial p} \end{bmatrix} = \begin{bmatrix} \frac{\partial Ln((R^0)^{-1}R)}{\partial R} & 0 \\ 0 & \frac{\partial (p-p^0)}{\partial p} \end{bmatrix} = \begin{bmatrix} J_r^{-1}((R^0)^{-1}R) & 0 \\ 0 & I \end{bmatrix}$$

提升作业

- 使用提供的EdgeSE3Prior类，仿照其他残差增加边

$$J_{rP} = \begin{bmatrix} J_r^{-1}((R^0)^{-1}R) & 0 \\ 0 & I \end{bmatrix}$$

```cpp
void EdgeSE3Prior::ComputeJacobians() {

    VecX param_i = verticies_[0]->Parameters();
    Qd Qi(param_i[6], param_i[3], param_i[4], param_i[5]);

    // w.r.t. pose i
    Eigen::Matrix<double, 6, 6> jacobian_pose_i = Eigen::Matrix<double, 6, 6>::Zero();

#ifdef USE_SO3_JACOBIAN
    Sophus::SO3d ri(Qi);
    Sophus::SO3d rp(Qp_);
    Sophus::SO3d res_r = rp.inverse() * ri;
    // http://rpg.ifi.uzh.ch/docs/RSS15_Forster.pdf  公式A.32
    jacobian_pose_i.block<3,3>(0,3) = Sophus::SO3d::JacobianRInv(res_r.log());
#else
    jacobian_pose_i.block<3,3>(0,3) = Qleft(Qp_.inverse() * Qi).bottomRightCorner<3, 3>();
#endif
    jacobian_pose_i.block<3,3>(3,0) = Mat33::Identity();

    jacobians_[0] = jacobian_pose_i;
//    std::cout << jacobian_pose_i << std::endl;
}
```

- 评价结果时需要参考论文将第一帧和真值对齐，用RMSE评价

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening