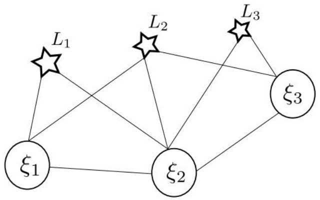
基于滑动窗口算法的 VIO 系统：可观性和一致性

主讲人 陈康

纲要

作业

❶ 某时刻，SLAM 系统中相机和路标点的观测关系如下图所示，其中 ξ 表示相机姿态，L 表示观测到的路标点。当路标点 L 表示在世界坐标系下时，第 k 个路标被第 i 时刻的相机观测到，重投影误差为 $\mathbf{r}(\xi_i, L_k)$。另外，相邻相机之间存在运动约束，如 IMU 或者轮速计等约束。

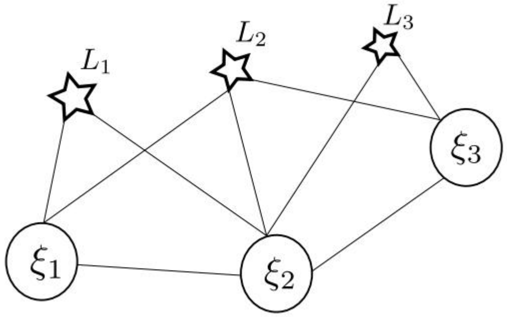


1. 请绘制上述系统的信息矩阵 $\mathbf{\Lambda}$.
2. 请绘制相机 ξ_1 被 marg 以后的信息矩阵 $\mathbf{\Lambda}'$.

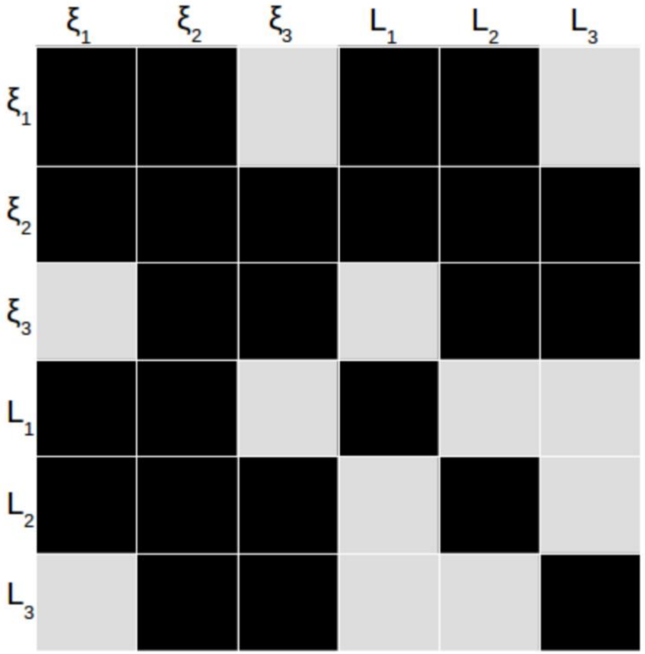
❷ 阅读《Relationship between the Hessian and Covariance Matrix for Gaussian Random Variables》. 证明信息矩阵和协方差的逆之间的关系。

❸ 请补充作业代码中单目 Bundle Adjustment 信息矩阵的计算，并输出正确的结果。正确的结果为：奇异值最后 7 维接近于 0，表明零空间的维度为 7.

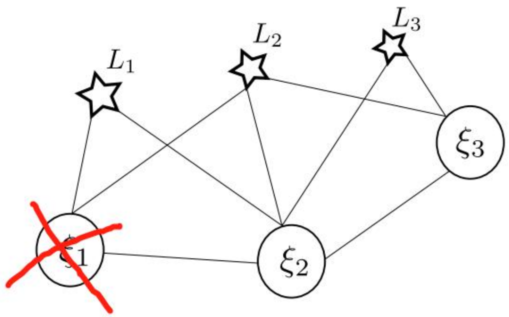
第1题（第1小题）



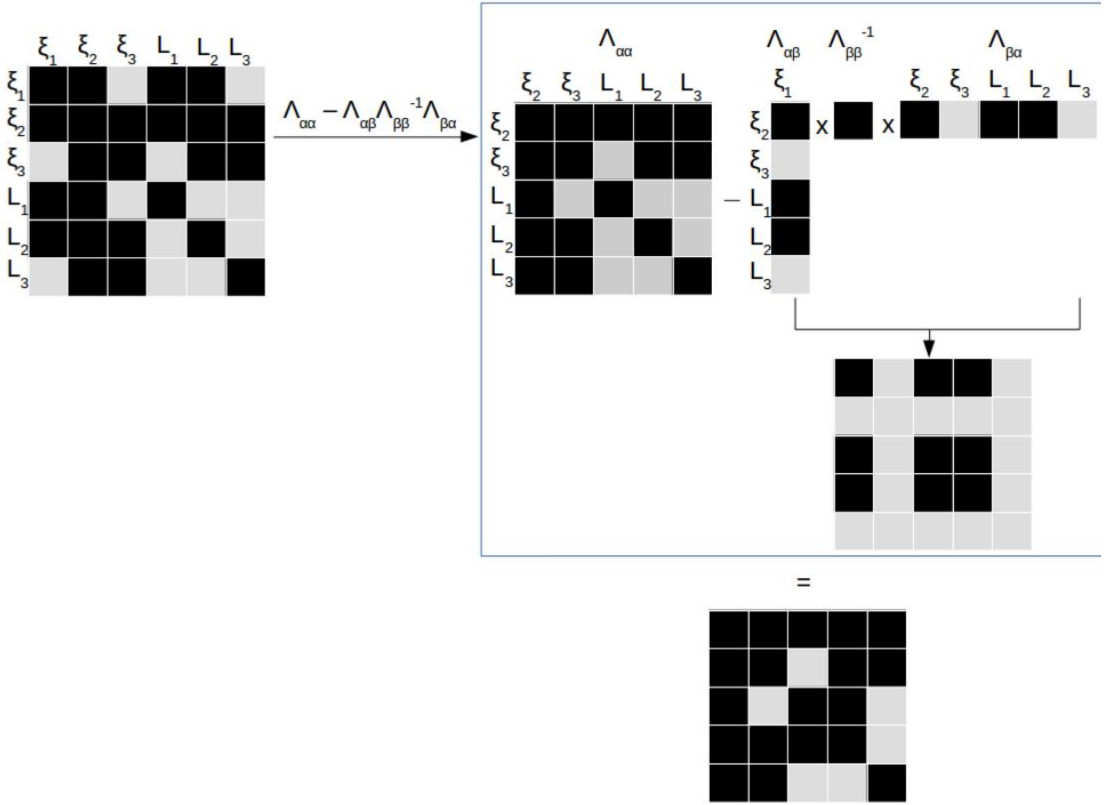
根据相机和路标点的观测关系图



第1题（第2小题）



L_1
L_2
L_3
ξ_1
ξ_2
ξ_3



ξ1 ξ2 ξ3 L1 L2 L3
$\Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\beta\alpha}$
$\Lambda_{\alpha\alpha}$
$\Lambda_{\alpha\beta}$
$\Lambda_{\beta\beta}^{-1}$
$\Lambda_{\beta\alpha}$
ξ2 ξ3 L1 L2 L3
ξ1
x
x
=

第2题

根据论文《《Relationship between the Hessian and Covariance Matrix for
Gaussian Random Variables》中的内容，有：
假设高斯随机向量θ为n维行向量（即可以看成nx1矩阵），均值向量为θ^*，协方差矩阵为Σ_θ，则其联合概率密度函数为：

$$p(\theta) = (2\pi)^{-\frac{N_\theta}{2}}|\Sigma_\theta|^{-\frac{1}{2}}\exp[-\frac{1}{2}(\theta-\theta^*)^T\Sigma_\theta^{-1}(\theta-\theta^*)]$$

则目标函数可以定义为该联合概率密度函数的负对数：

$$J(\theta) =-\ln p(\theta) = \frac{N_\theta}{2}\ln 2\pi + \frac{1}{2}\ln|\Sigma_\theta| + \frac{1}{2}(\theta-\theta^*)^T\Sigma_\theta^{-1}(\theta-\theta^*)$$

该目标函数是变量θ的二次函数，通过对θ_l和$\theta_{l'}$求部分偏导，即可得到在(l,l')上的Hessian矩阵：

$$H^{(l,l')}(\theta^*) = \frac{\partial^2 J(\theta)}{\partial\theta_l\partial\theta_{l'}}|_{\theta=\theta^*} = (\Sigma_\theta^{-1})^{(l,l')}$$

第2题

但是论文中直接就给出了Hessian矩阵（即信息矩阵）等于协方差的逆，并未给出中间证明过程，下面将自行推导证明过程：

$$H^{(l,l')}(\theta^*) = \frac{\partial^2 J(\theta)}{\partial\theta_l \partial\theta_{l'}}|_{\theta=\theta^*} = [\frac{\partial}{\partial\theta_l}\left(\frac{\partial J(\theta)}{\partial\theta_l}\right)]_{\theta=\theta^*} = [\frac{\partial J'(\theta)}{\partial\theta_l}]_{\theta=\theta^*}$$

由上式可知，首先对$J(\theta)$求一次导得到$J'(\theta)$，然后对$J'(\theta)$再求一次导即为Hessian矩阵。

先使用矩阵乘法法则对$J(\theta)$求微分（$J(\theta)$式中只有$\frac{1}{2}(\theta-\theta^*)^T\Sigma_\theta^{-1}(\theta-\theta^*)$项与$\theta$有关），则：

$$dJ = (\frac{1}{2}d\theta)^T\left(\Sigma_\theta^{-1}\theta - \Sigma_\theta^{-1}\theta^*\right) + \frac{1}{2}(\theta-\theta^*)^T(\Sigma_\theta^{-1}d\theta)$$

第2题

深蓝学院
shenlanxueyuan.com

由一开始的假设可知，θ为n维行向量，则$\Sigma_\theta^{-1}\theta$仍为n维行向量，$d\theta$也为n维行向量，则根据两个向量的内积公式$u^T v = v^T u$，可将上式变换为：

$$dJ = \left(\Sigma_\theta^{-1}\theta - \Sigma_\theta^{-1}\theta^*\right)^T (\frac{1}{2} d\theta) + \frac{1}{2}(\theta - \theta^*)^T (\Sigma_\theta^{-1} d\theta)$$

$$= \left(\Sigma_\theta^{-1}(\theta - \theta^*)\right)^T (\frac{1}{2} d\theta) + \frac{1}{2}(\theta - \theta^*)^T (\Sigma_\theta^{-1} d\theta)$$

$$= (\theta - \theta^*)^T (\Sigma_\theta^{-1})^T (\frac{1}{2} d\theta) + \frac{1}{2}(\theta - \theta^*)^T (\Sigma_\theta^{-1} d\theta)$$

$$= \frac{1}{2}(\theta - \theta^*)^T (\Sigma_\theta^{-1})^T (d\theta) + \frac{1}{2}(\theta - \theta^*)^T \Sigma_\theta^{-1} (d\theta)$$

第2题

由于Σ_θ为协方差矩阵，则Σ_θ为对称矩阵，则有$(\Sigma_\theta^{-1})^T=\Sigma_\theta^{-1}$，则上式可以进一步简化为：

$$dJ=\frac{1}{2}(\theta-\theta^*)^T(\Sigma_\theta^{-1})^T(d\theta)+\frac{1}{2}(\theta-\theta^*)^T\Sigma_\theta^{-1}(d\theta)$$

$$=\frac{1}{2}(\theta-\theta^*)^T\Sigma_\theta^{-1}(d\theta)+\frac{1}{2}(\theta-\theta^*)^T\Sigma_\theta^{-1}(d\theta)=(\theta-\theta^*)^T\Sigma_\theta^{-1}(d\theta)$$

又根据导数与微分的关系$dY=\frac{\partial Y^T}{\partial X}dx$，则有：

$$\frac{\partial J}{\partial\theta}=(\Sigma_\theta^{-1})^T(\theta-\theta^*)=J'(\theta)$$

则对上式$J'(\theta)$再次进行θ求导，则非常容易看到结果就是Σ_θ^{-1}，即：

$\frac{\partial J'}{\partial\theta}=\Sigma_\theta^{-1}$，则证明完毕。

第3题

$$\begin{bmatrix} \mathbf{H}_{pp} & \mathbf{H}_{pl} \\ \mathbf{H}_{lp} & \mathbf{H}_{ll} \end{bmatrix}$$

```cpp
// 随机数生成三维特征点
std::default_random_engine generator;
std::vector<Eigen::Vector3d> points;
for(int j = 0; j < featureNums; ++j)
{
    std::uniform_real_distribution<double> xy_rand(-4, 4.0);
    std::uniform_real_distribution<double> z_rand(8., 10.);
    double tx = xy_rand(generator);
    double ty = xy_rand(generator);
    double tz = z_rand(generator);

    Eigen::Vector3d Pw(tx, ty, tz);
    points.push_back(Pw);

    for (int i = 0; i < poseNums; ++i) {
        Eigen::Matrix3d Rcw = camera_pose[i].Rwc.transpose();
        Eigen::Vector3d Pc = Rcw * (Pw - camera_pose[i].twc);

        double x = Pc.x();
        double y = Pc.y();
        double z = Pc.z();
        double z_2 = z * z;
        Eigen::Matrix<double,2,3> jacobian_uv_Pc;
        jacobian_uv_Pc<< fx/z, 0 , -x * fx/z_2,
                0, fy/z, -y * fy/z_2;
        Eigen::Matrix<double,2,3> jacobian_Pj = jacobian_uv_Pc * Rcw;
        Eigen::Matrix<double,2,6> jacobian_Ti;
        jacobian_Ti << -x* y * fx/z_2, (1+ x*x/z_2)*fx, -y/z*fx, fx/z, 0 , -x * fx/z_2,
                        -(1+y*y/z_2)*fy, x*y/z_2 * fy, x/z * fy, 0,fy/z, -y * fy/z_2;

        H.block(i*6,i*6,6,6) += jacobian_Ti.transpose() * jacobian_Ti;
        /// 请补充完整作业信息矩阵块的计算
        //H22
        H.block(poseNums*6+j*3,poseNums*6+j*3,3,3) += jacobian_Pj.transpose() * jacobian_Pj;
        //H12
        H.block(i*6,poseNums*6+j*3,6,3) += jacobian_Ti.transpose() * jacobian_Pj;
        //H21
        H.block(poseNums*6+j*3,i*6,3,6) += jacobian_Pj.transpose() * jacobian_Ti;
//  H.block(?,?,?,?) += ?;
```

第3题

程序运行后截图，最后7维数值非常非常小，接近0，表明零空间的维度为7。

```
 0.00734249
 0.00701361
 0.00634341
 0.00608493
 0.00547299
  0.0053236
 0.00520788
 0.00502341
  0.0048434
 0.00451083
  0.0042627
 0.00386223
 0.00351651
 0.00302963
 0.00253459
 0.00230246
 0.00172459
0.000422374
1.25708e-16
8.63763e-17
5.18689e-17
4.38809e-17
2.98776e-17
1.45304e-17
1.59456e-18
root@ck-virtual-machine:/home/ck/working/slam/work/13/nullspace_test/build#
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening