观测和结果的不确定性估计

在状态估计中，不确定性估计的是核心问题，直接影响滤波器（或者其他算法系统）的收敛性和鲁棒性，需要系统性的（也许也只是随性的）了解。

主要可以分为**观测不确定性（观测噪声协方差 *R*）**、**过程不确定性（过程噪声协方差 *Q*）**和 **状态估计不确定性（状态协方差 *P*）**

观测不确定性一般是指外部接收到的数据的不确定性，例如各类传感器信息；过程不确定性指在算法处理过程中的影响；状态估计不确定性是指最终估计结果的不确定性

一、观测不确定性

观测噪声协方差 *R* 表示传感器测量的不确定性，需准确建模以避免过拟合或欠拟合。

h是观测矩阵，观测到地标x时就会产生对应的观测输入z

1. 理论基础

- 观测模型：$z_k = h(x_k) + v_k$，其中 $v_k \sim \mathcal{N}(0, R_k)$。
- R 是正定对称矩阵，对角线元素为各观测分量的方差，非对角线元素表示噪声间的相关性（如多传感器间的耦合误差）。

2. 确定方法

(1) 传感器标定法

- **静态实验**：在静止状态下采集 N 次观测数据 $\{z_i\}$，计算：

$$R = \frac{1}{N-1}\sum_{i=1}^{N}(z_i - \bar{z})(z_i - \bar{z})^T, \quad \bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i$$

- **动态实验**：通过已知真实轨迹的对比，拟合残差 $z_k - h(x_k^{\text{true}})$ 的协方差。

(2) 厂商参数法

- 直接使用传感器手册提供的噪声特性（如GPS水平误差 $\sigma = 1.5\,\text{m} \Rightarrow R = \text{diag}(1.5^2, 1.5^2)$）。

(3) 自适应估计法

- **新息协方差匹配**：利用新息序列 $\tilde{z}_k = z_k - h(\hat{x}_{k|k-1})$ 在线更新 R：

$$\hat{R}_k = \frac{1}{W}\sum_{i=k-W+1}^{k} \tilde{z}_i \tilde{z}_i^T + H_k P_{k|k-1} H_k^T$$

其中 W 为滑动窗口大小。

总结：观测模型的协方差通常来自传感器的厂家提供的参数，例如相机、陀螺仪等

二、过程不确定性

比如：pnp过程中的相机内参误差、3d点误差、还有不同初始值的不确定性。但是这一类的不确定性在单帧的估计中影响极小，一般不考虑在最终结果之中。

过程不确定性更多考虑的是长时系统中的误差累计，

过程噪声协方差 Q 表示系统模型的不确定性（如未建模动态或线性化误差）。

1. 理论基础

- 状态预测模型：$x_k = f(x_{k-1}, u_k) + w_k$，其中 $w_k \sim \mathcal{N}(0, Q_k)$。
- Q 的维度与状态量相同，需反映模型误差的统计特性。

2. 确定方法

（1）物理模型推导

- 分析未建模动态（如忽略的空气阻力、摩擦）的扰动范围。
  - **示例**：车辆模型中，假设加速度扰动服从 $\mathcal{N}(0, \sigma_a^2)$，则：

$$Q = \text{diag}(0, 0, \sigma_a^2 \Delta t^2, \sigma_a^2 \Delta t)$$

（对应位置、速度的噪声传递）

（2）经验调参法

- 通过仿真调整 Q，使得新息序列 $\tilde{z}_k$ 满足：

$$\frac{1}{N} \sum_{k=1}^{N} \tilde{z}_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \tilde{z}_k \approx \dim(z_k)$$

（即标准化新息平方和接近观测维度）。

（3）自适应方法

- 基于残差或状态变化动态调整 Q，适用于模型参数时变场景。

三、状态估计不确定性

状态估计的置信度由状态协方差 P表示，可以有以下几种方式来确定：

1、基于残差和雅可比计算协方差

$$\Sigma_\theta = \sigma^2 (J^T J)^{-1}$$

- J：残差对位姿的 Jacobian（可自动或手动求导）
- σ^2：残差的方差估计（来自 inlier 的重投影误差）

📌 优点：反映了实际数据情况，适合你这种系统
📌 缺点：要计算 J，但可以用 Ceres、g2o 或自写自动导数

2、基于误差传播（观测协方差 + Jacobian）

从误差传播公式推导（线性近似）：

$$\Sigma_\theta = (J^T \Sigma_x^{-1} J)^{-1}$$

其中：

- Σ_x：观测误差协方差矩阵，通常为 $2N \times 2N$，假设每个图像点有一定像素误差（比如 $\sigma = 1.0$ 像素）

📌 优点：理论严谨，可建模不同点的权重

📌 缺点：难以获取准确的 Σ_x，尤其在深度学习特征匹配中很难合理建模真实误差分布

状态协方差 *P* 表示状态估计的置信度，通过预测-更新循环递推。

1. 预测阶段

- 协方差传播：

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

其中 $F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}}$ 为状态转移雅可比矩阵。

2. 更新阶段

- 卡尔曼增益：

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$$

其中 。

- 协方差更新：

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T$$

（Joseph形式，保证数值对称正定）。

物理意义：

- 对角线元素：状态各分量的方差（如P_11表示位置估计的误差方差）。
- 非对角线元素：状态分量间的相关性（如位置与速度的协方差）。

四、完整的不确定性传递流程

以EKF为例，展示不确定性如何从传感器到最终估计传递：

1. **初始化**：设置初始状态 $\hat{x}_0$ 和协方差 P_0（通常基于先验知识）。
2. **预测**：
   - 状态预测：$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$。
   - 协方差预测：$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$。
3. **更新**：
   - 计算新息：$\tilde{z}_k = z_k - h(\hat{x}_{k|k-1})$。
   - 更新协方差：$P_{k|k} = (I - K_k H_k) P_{k|k-1}$。
4. **不确定性可视化**：
   - 状态估计的 3σ 边界：$\hat{x}_k \pm 3\sqrt{\text{diag}(P_{k|k})}$。

3、采样或者蒙特卡罗方法估计协方差

步骤：

1. 多次对匹配点采样（带噪声扰动）
2. 每次求解 PnP 位姿
3. 统计输出位姿的协方差

📌 优点：适用于系统难以显式建模误差时

📌 缺点：开销大、慢，但可以结合 SuperGlue 的匹配 score 做概率采样

大量计算得到一个平均的方差

四、实例----室内空间视觉定位的cov

有一套定位的算法流程如下：

根据预先建好的地图（包括图片、深度、特征点，3d点使用2d特征点+深度来得到），在场景中拍摄一张图片，从图片中提取superpoint特征点，然后根据训练好的词袋找出topk图片，得到topk图片的候选3d点集，然后使用pnp得到图片位姿。

我现在想要输出这个位姿的可信度，也就是旋转（四元数）和平移的covariance

公式：

🧠 协方差估计方法（融合多个因素）

我们使用**加权雅可比法**估计协方差：

$$\Sigma_{\text{pose}} = \sigma^2 (J^\top W J)^{-1}$$

其中：

- J：由 `cv2.projectPoints(..., jacobian=True)` 提供
- W：权重对角矩阵，元素是 `superpoint_score * match_score / (1 + reproj_error)`
- σ^2：重投影残差的方差（均方误差）

误差传播理论中的协方差传播：

1. 线性变换的误差传播

设 $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$，其中：

- $\mathbf{x}$ 是随机向量，均值 $\mathbb{E}[\mathbf{x}] = \mu_{\mathbf{x}}$，协方差 $\mathrm{Cov}(\mathbf{x}) = \mathbf{\Sigma}_{\mathbf{x}}$。
- $\mathbf{A}$ 是常数矩阵，$\mathbf{b}$ 是常数向量。

推导步骤：

1. **计算 $\mathbf{y}$ 的均值：**

$$\mathbb{E}[\mathbf{y}] = \mathbf{A}\mathbb{E}[\mathbf{x}] + \mathbf{b} = \mathbf{A}\mu_{\mathbf{x}} + \mathbf{b}$$

2. **计算 $\mathbf{y}$ 的协方差：**

$$\mathrm{Cov}(\mathbf{y}) = \mathbb{E}\left[(\mathbf{y} - \mathbb{E}[\mathbf{y}])(\mathbf{y} - \mathbb{E}[\mathbf{y}])^\top\right]$$

代入 $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$：

$$\mathbf{y} - \mathbb{E}[\mathbf{y}] = \mathbf{A}(\mathbf{x} - \mu_{\mathbf{x}})$$

因此：

$$\mathrm{Cov}(\mathbf{y}) = \mathbb{E}\left[\mathbf{A}(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{x} - \mu_{\mathbf{x}})^\top \mathbf{A}^\top\right] = \mathbf{A}\mathbf{\Sigma}_{\mathbf{x}}\mathbf{A}^\top$$

结论：

对于线性变换 $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$，协方差传播公式为：

$$\mathbf{\Sigma}_{\mathbf{y}} = \mathbf{A}\mathbf{\Sigma}_{\mathbf{x}}\mathbf{A}^\top$$

2. 非线性变换的误差传播（一阶近似）

设 $\mathbf{y} = f(\mathbf{x})$ 是非线性函数，在 $\mu_{\mathbf{x}}$ 处进行一阶泰勒展开：

$$\mathbf{y} \approx f(\mu_{\mathbf{x}}) + \mathbf{J}(\mathbf{x} - \mu_{\mathbf{x}})$$

其中：

- $\mathbf{J}$ 是 f 的雅可比矩阵（Jacobian），$\mathbf{J}_{ij} = \frac{\partial y_i}{\partial x_j}$ 在 $\mathbf{x} = \mu_{\mathbf{x}}$ 处计算。

推导步骤：

1. **近似 $\mathbf{y}$ 的均值：**

$$\mathbb{E}[\mathbf{y}] \approx f(\mu_{\mathbf{x}}) + \mathbf{J}\mathbb{E}[\mathbf{x} - \mu_{\mathbf{x}}] = f(\mu_{\mathbf{x}})$$

2. **近似 $\mathbf{y}$ 的协方差：**
   由于 $\mathbf{y} - \mathbb{E}[\mathbf{y}] \approx \mathbf{J}(\mathbf{x} - \mu_{\mathbf{x}})$，类比线性情况：

$$\mathrm{Cov}(\mathbf{y}) \approx \mathbf{J}\mathrm{Cov}(\mathbf{x})\mathbf{J}^\top = \mathbf{J}\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{J}^\top$$

结论：

对于非线性变换 $\mathbf{y} = f(\mathbf{x})$，协方差的**一阶近似**为：

$$\boldsymbol{\Sigma}_{\mathbf{y}} \approx \mathbf{J}\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{J}^\top$$

对于重投影这个场景来说，已知因变量冲投影误差的协方差sigma， 需要求解的是自变量位姿x的cov，所以根据误差传播理论（没有搜到具体推导，暂时先这么记住吧！），此时cov的方程为（T是位姿）：

$$\boldsymbol{\Sigma}_{\mathbf{T}} = \left(\mathbf{J}^\top \boldsymbol{\Sigma}_{\text{reproj}}^{-1} \mathbf{J}\right)^{-1}$$

对于当前场景下的情况来说，观测的协方差来自权重矩阵和冲投影误差的均方根的组合，目的是把协方差转化为实际的误差尺度（像素和米）

1、估计观测协方差

观测是图片上检测到的2d特征点，因此没有厂家固定给的参数，需要自己来给一个估计。

那么就可以结合以下几点给定一个大致的估计：

- superpoint的特征点置信度
- 匹配的数量

- 匹配的质量（通过重投影误差反映）

记得需要归一化！才能不影响最终的数值结果

2、过程不确定性

计算误差等在这个单帧定位系统中可以略而不记。

3、pnp误差传递

使用PnP算法来得到位姿，需要考虑投影过程中的误差传递，具体来说，就是投影方程对位姿状态的导数，维度是(2N, 6)，N是点的数量，2表示投影结果的维度（x和y的二维像素值），而6表示旋转和平移各三个维度

可以通过opencv的投影函数获得，取其前六维

Code block

```
projected, jacobian = cv2.projectPoints(
    matched_3d_inliers, rot_vec, tran_vec, intrinsic[0], intrinsic[1])
```

附：这是一个典型的场景，也可以手动计算jacobian的表达式

问题建模：

🌟 PnP 观测模型

对于每个点：

$$\mathbf{u} = \pi\big(R(\mathbf{r})\mathbf{X} + \mathbf{t}\big)$$

其中：

- $\mathbf{X}$：3D地图点
- $R(\mathbf{r})$：旋转向量对应的旋转矩阵
- $\pi(\mathbf{P})$：相机投影函数

$$\pi(\mathbf{P}) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix}$$

🌟 误差项

对每个点：

$$\mathbf{e} = \mathbf{u}_{\text{obs}} - \pi(R(\mathbf{r})\mathbf{X} + \mathbf{t})$$

PnP 就是最小化：

$$\sum \mathbf{e}^\top W \mathbf{e}$$

过程计算

🌟 PnP 雅可比矩阵

我们需要：

$$\frac{\partial \mathbf{e}}{\partial T}$$

也就是：

$$\frac{\partial \mathbf{e}}{\partial T} = -\frac{\partial \pi}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial T}$$

其中：

$$\mathbf{P} = R(\mathbf{r})\mathbf{X} + \mathbf{t}$$

1️⃣ 投影对相机点的导数

$$\frac{\partial \pi}{\partial \mathbf{P}} = \begin{bmatrix} f_x/Z & 0 & -f_x X/Z^2 \\ 0 & f_y/Z & -f_y Y/Z^2 \end{bmatrix}$$

2️⃣ 相机点对位姿的导数

平移：

$$\frac{\partial \mathbf{P}}{\partial \mathbf{t}} = I_{3\times 3}$$

旋转：

$$\frac{\partial \mathbf{P}}{\partial \mathbf{r}} = -R(\mathbf{r})[\mathbf{X}]_\times$$

这里 $[\mathbf{X}]_\times$ 是 X 的叉乘矩阵：

$$[\mathbf{X}]_\times = \begin{bmatrix} 0 & -X_z & X_y \\ X_z & 0 & -X_x \\ -X_y & X_x & 0 \end{bmatrix}$$

3️⃣ 最终雅可比

对每个点：

$$J_i = -\frac{\partial \pi}{\partial \mathbf{P}} \begin{bmatrix} -R[\mathbf{X}]_\times & I \end{bmatrix}$$

拼成整体：

$$J = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \end{bmatrix}$$

每个 J_i 是 2×6

4、状态估计不确定性

最终的状态的不确定性，也就是输出位姿的协方差矩阵，由前两个过程传递而来。sigma2也需要加权以保证对应。

Code block

```
# 4. Estimate cov by weighted Jocobian
'''
    cov = sigma^2 * (J^T @ W @ J).inv, in which:
    - sigma: weighted variance of the reprojection loss
    - W: weight diagonal matrix
    - J: Jocobian in projection, (2N, 6)
'''
matched_2d_inliers = matched_2d[inliers.squeeze()]
matched_3d_inliers = matched_3d[inliers.squeeze()]
projected, jacobian = cv2.projectPoints(matched_3d_inliers, rot_vec, tran_vec, intrinsic[0], intrinsic[1])
reproj_errors = np.linalg.norm(projected.squeeze() - matched_2d_inliers, axis=1)

weights = matched_feature_scores[inliers.squeeze()] * match_scores[inliers.squeeze()]
weights += 1e-6
weights = weights / np.sum(weights)
sigma2 = (weights @ (reproj_errors ** 2)) / (np.sum(weights) - 6)  # variance of reprojection error

J = jacobian[:, :6]  # (2N, 6), only use rot and tran part
W = np.diag(np.repeat(weights, 2))  # (2N, 2N)
cov_pose = np.linalg.inv(J.T @ W @ J + 1e-6 * np.eye(6)) * sigma2
```

```
cov_rot = cov_pose[:3, :3]
cov_tran = cov_pose[3:, 3:]
```

5、过程中的异常值分析

1. cov数值分析

- **cov对角线上的元素表示该点的方差大小，可以开平方得到这个维度估计的标准差**（也就是稳定性，数值越大越不稳定），例如角度中标准差根号1=1 rad，约为50度的标准差
- 如果平移对角线元素过大，很有可能是尺度标准没有统一，比如毫米为单位的3d点坐标用在米级系统，导致数值爆炸

✅ 分析原因

可能原因	说明
单位不一致	有些系统中点坐标是以毫米为单位，而你用的是米，或反之
匹配数量太少 / 分数太低	如果匹配点少（如 < 10），或者匹配质量差，PnP 会极度不稳定，协方差就大
特征点分布不均匀	如果所有匹配都集中在图像某个角落，PnP 解是病态的（就像用一只眼测深度）
重投影误差高	如果误差大，则 `W`（权重矩阵）小，对应协方差就大
雅可比矩阵的缩放问题	如果你没对 `J.T @ W @ J` 做适当单位缩放，导致尺度漂移大
相机内参问题	若相机内参设错，会让像素投影到空间点的位置偏差很大 → PnP 不可信

2. jacobian数值分析

在优化问题中，雅可比矩阵（Jacobian matrix）是目标函数相对于其参数的一阶偏导数矩阵。它在许多优化算法中起着至关重要的作用，特别是在基于梯度的优化方法中，如梯度下降、牛顿法和拟牛顿法等。雅可比矩阵提供了参数变化对目标函数变化的敏感度信息。

异常的雅可比矩阵代表什么？

1. **数值不稳定**：雅可比矩阵中的元素如果包含非常大的值或者非常小的值（接近于零），可能导致数值不稳定，影响优化过程的收敛性和稳定性。
2. **模型错误**：异常的雅可比矩阵可能表明模型或函数本身存在问题，如模型参数化不当、函数定义错误或不连续等。
3. **奇异性**：雅可比矩阵的奇异性（非满秩）可能导致优化算法无法正确地更新参数，因为雅可比矩阵的逆不存在或不稳定。
4. **梯度爆炸或消失**：在深度学习中，雅可比矩阵的异常值可能导致梯度爆炸（梯度值非常大）或梯度消失（梯度值非常小），影响网络的训练。

如何观察雅可比矩阵是否异常？

1. **检查雅可比矩阵的范数**：计算雅可比矩阵的Frobenius范数、1-范数或无穷范数，检查是否存在异常大的值。
2. **条件数**：计算雅可比矩阵的条件数（最大奇异值与最小奇异值的比值）。条件数很大表明矩阵接近奇异，可能导致数值不稳定。
3. **可视化**：对于小规模问题，可以可视化雅可比矩阵的元素，检查是否存在异常值。
4. **梯度检查**：通过有限差分方法计算梯度，并与雅可比矩阵计算的梯度进行比较，检查是否存在显著差异。
5. **正则化**：在优化过程中添加正则化项，如L1或L2正则化，可以帮助稳定雅可比矩阵。
6. **使用鲁棒的优化算法**：选择对雅可矩阵异常值鲁棒的优化算法，如信赖域方法或自适应学习率的梯度下降法。
7. **检查数据和模型**：确保输入数据的质量和模型的正确性，避免因数据错误或模型不当导致雅可比矩阵异常。

3. 矩阵奇异性

奇异值和奇异性

奇异值（Singular Values）

奇异值是矩阵奇异值分解（Singular Value Decomposition, SVD）的组成部分。对于任意矩阵 $A \in \mathbb{R}^{m \times n}$，都可以分解为：

$$A = U\Sigma V^T$$

其中：

- $U \in \mathbb{R}^{m \times m}$ 是一个正交矩阵，其列向量称为左奇异向量（left singular vectors）。
- $V \in \mathbb{R}^{n \times n}$ 是一个正交矩阵，其列向量称为右奇异向量（right singular vectors）。
- $\Sigma \in \mathbb{R}^{m \times n}$ 是一个对角矩阵，对角线上的非负实数称为矩阵 A 的奇异值（singular values），并且通常按从大到小的顺序排列。

奇异值提供了矩阵在不同方向上的“尺度”或“拉伸因子”的信息，它们是非负的。

奇异性（Singularity）

奇异性描述的是矩阵不可逆的性质。一个矩阵是奇异的，当且仅当它没有满秩（即矩阵的秩小于其行数或列数），或者说，当且仅当它的行列式（determinant）为零。奇异矩阵没有逆矩阵。

奇异值和奇异性的关系

奇异值和奇异性之间的关系体现在奇异值分解中。具体来说：

- **非奇异矩阵**：如果矩阵 A 的所有奇异值都不为零，则 A 是非奇异的（invertible），意味着 A 可逆。
- **奇异矩阵**：如果矩阵 A 至少有一个奇异值为零，则 A 是奇异的（singular），意味着 A 不可逆。

奇异值分解提供了一种判断矩阵是否奇异的方法：检查 Σ 中是否有零元素。如果有至少一个零奇异值为零，则矩阵是奇异的。

物理和几何解释

- **非奇异矩阵**：可以视为没有“塌陷”或“压缩”到更低维度，能够唯一地表示线性变换。
- **奇异矩阵**：至少有一个方向被“塌陷”到更低维度，导致矩阵不能唯一地表示线性变换，可能存在无限多解或无解。

jacobian的奇异性影响

Jacobian的奇异性及其影响

雅可比矩阵的奇异性意味着矩阵不可逆，这在优化和数值分析中有重要影响：

1. **优化问题**：在优化问题中，如果雅可比矩阵是奇异的，那么基于梯度的优化算法可能无法正确更新参数，因为无法计算逆矩阵来调整步长。
2. **数值稳定性**：奇异雅可比矩阵可能导致数值不稳定，因为它们可能导致算法中的除零操作或病态问题。
3. **误差传播**：在误差分析中，奇异雅可比矩阵可能导致误差被放大，从而影响最终结果的准确性。
4. **线性系统**：在解决线性系统时，如果雅可比矩阵是奇异的，那么系统可能没有唯一解，或者没有解。
5. **机器学习**：在机器学习中，奇异雅可比矩阵可能导致梯度消失或爆炸，影响神经网络的训练效果。

处理奇异性

处理雅可比矩阵奇异性的方法包括：

- **正则化**：添加一个小的正则化项来避免奇异性。
- **使用伪逆**：如果雅可比矩阵是奇异的，可以使用其伪逆来代替逆矩阵。
- **特征值分解**：通过特征值分解来识别和处理奇异值。
- **重新参数化**：重新定义问题，以避免雅可比矩阵的奇异性。