

视觉SLAM：从理论到实践

第七次课 后端



主讲人 高翔

清华大学 自动控制与工程 博士
慕尼黑工业大学计算机视觉组 博士后
Email: gao.xiang.thu@gmail.com



第七讲 后端

1. 递归方法
2. 批量方法
3. Pose Graph

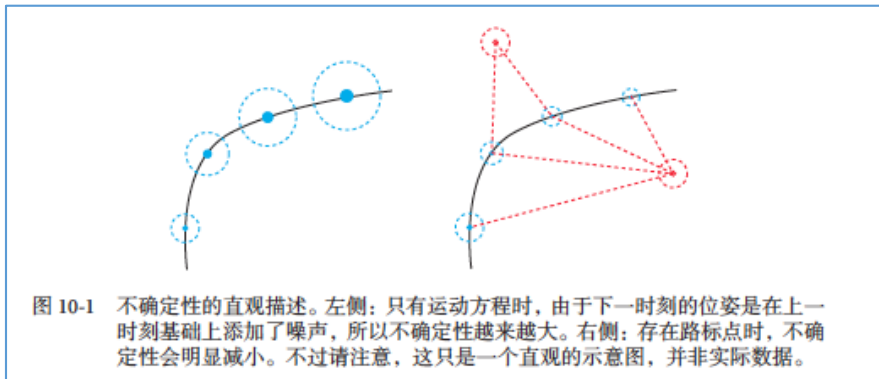
1. 递归方法

1. 递归方法

- 后端 (Backend)
 - 从带噪声的数据估计内在状态——状态估计问题
 - Estimated the inner state from noisy data
- 渐进式 (Incremental/Recursive)
 - 保持当前状态的估计，在加入新信息时，更新已有的估计（滤波）
 - 线性系统+高斯噪声=卡尔曼滤波器
 - 非线性系统+高斯噪声+线性近似=扩展卡尔曼
 - 非线性系统+非高斯噪声+非参数化=粒子滤波器
 - Sliding window filter & multiple state Kalman (MSCKF)
- 批量式 (Batch)
 - 给定一定规模的数据，计算该数据下的最优估计（优化）

1. 递归方法

- 简单的例子
 - 当我们蒙着眼走路时：
 - 一开始我们知道自己在哪
 - 能粗略估计每步的距离，但不确定
 - 不确定性随时间累积
 - 导致自身的位置不确定性越来越大
 - 某个时刻睁开眼睛时
 - 能够观测到周围
 - 尽管也有不确定性，但比估计步长小很多
 - 能够将不确定性保持在一定范围内



同理，可以用渐进式和批量式两种方式描述这个过程

1. 递归方法

- 数学描述

- 符号定义：
- $t = 0 \dots N$ 时间内，机器人的位姿为： x_0 到 x_N ，同时有路标 y_1, \dots, y_M
- 运动和观测方程记为：

$$\begin{cases} x_k = f(x_{k-1}, u_k) + w_k \\ z_{k,j} = h(y_j, x_k) + v_{k,j} \end{cases} \quad k = 1, \dots, N, j = 1, \dots, M.$$

- 注意：运动和观测都受噪声影响；可以没有运动方程（纯视觉SLAM）
- 我们从贝叶斯滤波器方法来推导卡尔曼滤波

1. 递归方法

- 用随机变量表示状态，估计其概率分布
- k 时刻所有待估计的量组成 k 时刻状态： $x_k \triangleq \{x_k, y_1, \dots, y_m\}$.
- k 时刻观测统一记成： z_k 方程简化为：符号稍微有点冲突

$$\begin{cases} x_k = f(x_{k-1}, u_k) + w_k \\ z_k = h(x_k) + v_k \end{cases} \quad k = 1, \dots, N.$$

- 我们从批量方法开始，引出递归方法

1. 递归方法

$$\begin{cases} x_k = f(x_{k-1}, u_k) + w_k \\ z_k = h(x_k) + v_k \end{cases} \quad k = 1, \dots, N.$$

- 根据过去0时刻到k时刻的数据

- 估计当前的状态: $P(x_k | x_0, u_{1:k}, z_{1:k})$.

- Bayes展开: $P(x_k | x_0, u_{1:k}, z_{1:k}) \propto P(z_k | x_k) P(x_k | x_0, u_{1:k}, z_{1:k-1})$.

似然

先验

- 先验部分按条件概率展开:

$$P(x_k | x_0, u_{1:k}, z_{1:k-1}) = \int P(x_k | x_{k-1}, x_0, u_{1:k}, z_{1:k-1}) P(x_{k-1} | x_0, u_{1:k}, z_{1:k-1}) dx_{k-1}.$$

运动模型的预测

K-1时刻估计 (10.6)

1. 递归方法

- 注意先验部分的组成：

$$P(x_k | x_0, u_{1:k}, z_{1:k-1}) = \int P(x_k | x_{k-1}, x_0, u_{1:k}, z_{1:k-1}) \overset{\text{k-1时刻的状态估计}}{P(x_{k-1} | x_0, u_{1:k}, z_{1:k-1})} dx_{k-1}. \quad (10.6)$$

k时刻受先前状态的影响

- 分歧：
 - 可以假设k时刻状态只和k-1时刻有关（假设了一阶马尔可夫性）
 - 或假设k时刻状态与先前所有时刻均有关（不假设马尔可夫性）

1. 递归方法

- 假设一阶马尔可夫性的情况： $P(x_k|x_{k-1}, x_0, u_{1:k}, z_{1:k-1}) = P(x_k|x_{k-1}, u_k)$.
 - 第二项： $P(x_{k-1}|x_0, u_{1:k}, z_{1:k-1}) = P(x_{k-1}|x_0, u_{1:k-1}, z_{1:k-1})$.
- 于是，该公式指出了如何从k-1时刻的状态分布递推至k时刻的分布
 - 只是现在我们还没有代入具体的分布形式
- 在线性模型、高斯状态分布下，我们将得到卡尔曼滤波器
 - 在非线形模型、高斯状态分布下，可以在工作点附近线性展开，得到扩展卡尔曼滤波器

1. 递归方法

- 卡尔曼滤波器的推导

- 线性模型和高斯噪声：
$$\begin{cases} x_k = A_k x_{k-1} + u_k + w_k \\ z_k = C_k x_k + v_k \end{cases} \quad k = 1, \dots, N.$$

$$w_k \sim N(0, R). \quad v_k \sim N(0, Q).$$

- 状态的高斯分布（区别先后验）
$$P(x_{k-1}) = N(\hat{x}_{k-1}, \hat{P}_{k-1})$$

\bar{x}_k, \bar{P}_k
后验

\bar{x}_k, \bar{P}_k
先验

1. 递归方法

- 预备知识：高斯分布的线性变换
 - 假定 $x \sim N(m, S), y = Ax + b$ 则 y 亦服从高斯分布
 - 有： $E[y] = E[Ax + b] = AE[x] + b = Am + b$

$$\begin{aligned} \text{Cov}[y] &= E[(y - E[y])(y - E[y])^T] \\ &= E[A(x - m)(x - m)^T A^T] = ASA^T \end{aligned}$$

1. 递归方法

- 预测（k-1时刻后验通过运动方程推算k时刻先验）

$$P(x_k | x_0, u_{1:k}, z_{1:k-1}) = N(A_k \hat{x}_{k-1} + u_k, A_k \hat{P}_{k-1} A_k^T + R).$$

需要用到高斯
分布的线性组
合性质

- 并记成： $\bar{x}_k = A_k \hat{x}_{k-1} + u_k, \quad \bar{P}_k = A_k \hat{P}_{k-1} A_k^T + R.$

先验

- 根据观测方程，可知： $P(z_k | x_k) = N(C_k x_k, Q).$

- 根据前面的Bayes展开： $P(x_k | x_0, u_{1:k}, z_{1:k}) \propto P(z_k | x_k) P(x_k | x_0, u_{1:k}, z_{1:k-1}).$

- 需要计算两个分布的乘积

1. 递归方法

$$P(x_k | x_0, u_{1:k}, z_{1:k}) \propto P(z_k | x_k) P(x_k | x_0, u_{1:k}, z_{1:k-1}).$$

- 技巧：已经两侧均是高斯分布，所以： $N(\hat{x}_k, \hat{P}_k) = N(C_k x_k, Q) \times N(\bar{x}_k, \bar{P}_k)$.
- 既然都是高斯分布，我们比较指数部分的二次项和一次项部分即可。
- 指数部分展开：

$$(x_k - \hat{x}_k)^T \hat{P}_k^{-1} (x_k - \hat{x}_k) = (z_k - C_k x_k)^T Q^{-1} (z_k - C_k x_k) + (x_k - \bar{x}_k)^T \bar{P}_k^{-1} (x_k - \bar{x}_k).$$

- 比较 x_k 的二次和一次项系数，对于二次项，有：

$$\hat{P}_k^{-1} = C_k^T Q^{-1} C_k + \bar{P}_k^{-1}.$$

1. 递归方法

- 再考虑一次项： $(x_k - \hat{x}_k)^T \hat{P}_k^{-1} (x_k - \hat{x}_k) = (z_k - C_k x_k)^T Q^{-1} (z_k - C_k x_k) + (x_k - \bar{x}_k)^T \bar{P}_k^{-1} (x_k - \bar{x}_k)$.
- 比较一次项系数： $-2\hat{x}_k^T \hat{P}_k^{-1} x_k = -2z_k^T Q^{-1} C_k x_k - 2\bar{x}_k^T \bar{P}_k^{-1} x_k$
- 整理之： $\hat{P}_k^{-1} \hat{x}_k = C_k^T Q^{-1} z_k + \bar{P}_k^{-1} \bar{x}_k$
- 两侧同乘 \hat{P}_k 并定义： $K = \hat{P}_k C_k^T Q^{-1}$ ，得：

$$\begin{aligned}\hat{x}_k &= \hat{P}_k C_k^T Q^{-1} z_k + \hat{P}_k \bar{P}_k^{-1} \bar{x}_k \\ &= K z_k + (I - K C_k) \bar{x}_k = \bar{x}_k + K (z_k - C_k \bar{x}_k).\end{aligned}$$

称为卡尔曼滤波的更新式

1. 递归方法

- 卡尔曼增益: $K = \hat{P}_k C_k^T Q^{-1}$ 需要计算 \hat{P}_k
- 另一种形式 (不需要计算后验协方差):
$$K = (C_k^T Q^{-1} C_k + \bar{P}_k^{-1})^{-1} C_k^T Q^{-1}$$
$$= \bar{P}_k C_k^T (Q + C_k \bar{P}_k C_k^T)^{-1}$$
- 转换需要引入 Sherman-Morrison-Woodbury 恒等式:

$$(A^{-1} + BD^{-1}C)^{-1} \equiv A - AB(D + CAB)^{-1}CA \quad (2.75a)$$

$$(D + CAB)^{-1} \equiv D^{-1} - D^{-1}C(A^{-1} + BD^{-1}C)^{-1}BD^{-1} \quad (2.75b)$$

$$AB(D + CAB)^{-1} \equiv (A^{-1} + BD^{-1}C)^{-1}BD^{-1} \quad (2.75c)$$

$$(D + CAB)^{-1}CA \equiv D^{-1}C(A^{-1} + BD^{-1}C)^{-1} \quad (2.75d)$$

1. 递归方法

• 小结:

1. 预测:

$$\bar{x}_k = A_k \hat{x}_{k-1} + u_k, \quad \bar{P}_k = A_k \hat{P}_{k-1} A_k^T + R$$

2. 校正:

- 计算卡尔曼增益 $K = \bar{P}_k C_k^T (Q + C_k \bar{P}_k C_k^T)^{-1}$
- 更新状态估计

$$\begin{aligned} \hat{x}_k &= \bar{x}_k + K(z_k - C_k \bar{x}_k) \\ \hat{P}_k &= (I - K C_k) \bar{P}_k \end{aligned}$$

经典卡尔曼滤波器的五个公式
给出了线性高斯系统的最优无偏估计

1. 递归方法

- Kalman Filter的非线性扩展：EKF

- 当 f, h 为非线性函数时：
$$\begin{cases} x_k = f(x_{k-1}, u_k) + w_k \\ z_k = h(x_k) + v_k \end{cases} \quad k = 1, \dots, N.$$

- 在工作点附近进行一阶Taylor展开：

$$x_k \approx f(\hat{x}_{k-1}, u_k) + \left. \frac{\partial f}{\partial x_{k-1}} \right|_{\hat{x}_{k-1}} (x_{k-1} - \hat{x}_{k-1}) + w_k.$$

记F

$$z_k \approx h(\bar{x}_k) + \left. \frac{\partial h}{\partial x_k} \right|_{\bar{x}_k} (x_k - \hat{x}_k) + n_k.$$

记H

1. 递归方法

- 预测部分: $\bar{x}_k = f(\hat{x}_{k-1}, u_k), \quad \bar{P}_k = F \hat{P}_k F^T + R_k.$
- 卡尔曼增益: $K_k = \bar{P}_k H^T (H \bar{P}_k H^T + Q_k)^{-1}.$
- 更新部分: $\hat{x}_k = \bar{x}_k + K_k (z_k - h(\bar{x}_k)), \hat{P}_k = (I - K_k H) \bar{P}_k.$

1. 递归方法

- EKF优点

- 推导简单清楚，适用各种传感器形式
- 易于做多传感器融合

- EKF缺点

- 一阶马尔可夫性过于简单
- 可能会发散（要求数据不能有outlier）
- 线性化误差
- 需要存储所有状态量的均值和方差，平方增长

2. 批量方法

2. 批量方法

- Bundle Adjustment已经在之前介绍过了
 - 事实上BA属于批量式的优化方法
 - 给定很多个相机位姿与观测数据，计算最优的状态估计
 - 定义每个运动/观测方程的误差，并从初始估计开始寻找梯度下降

$$\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n \|e_{ij}\|^2 = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n \|z_{ij} - h(\xi_i, p_j)\|^2.$$

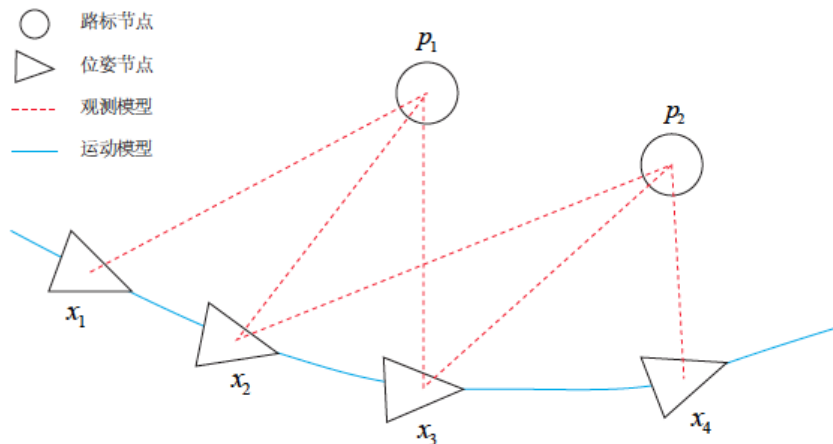


图 6-2 图优化的例子。

2. 批量方法

- BA问题与图结构的关系

- BA虽然是个纯优化问题，但亦可以用图模型清晰地表述出来
- 顶点为优化变量，边为运动/观测约束
- 本身还有一些特殊的结构

- 考虑在位姿 i 处对路标 j 的一次观测 z_{ij} :

$$e_{ij} = z_{ij} - h(x_i, y_j)$$

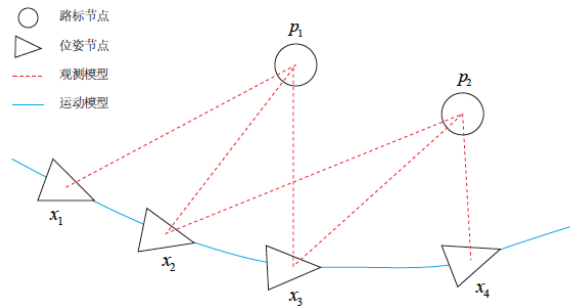


图 6-2 图优化的例子。

特点:

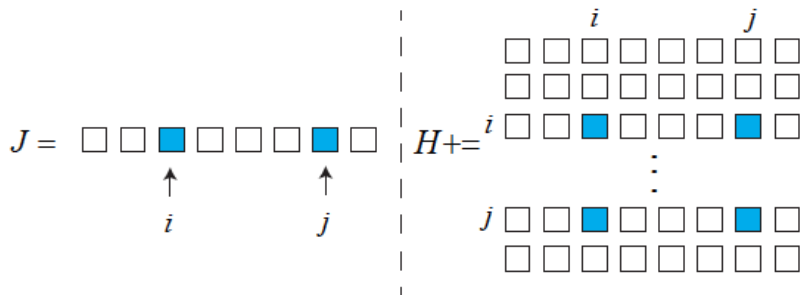
- 每个观测只关系两个变量，其中一个一个是相机，一个是路标
- 纯视觉Ba中，不存在相机与相机/路标与路标之间的关联
- 整个误差函数由许多个这样小的项组成

2. BA与图优化

- 根据Gauss-Newton或Levenberg-Marquardt, 最终我们会求解 $Hx = -b$ 形式的方程
- 每一个误差项会对整体线程方程产生贡献:
- 而误差仅和 i, j 相关, 导致它的雅可比为稀疏矩阵: $H = \sum_{i,j} J_{ij}^T J_{ij}$

$$J_{ij}(x) = \left(\mathbf{0}_{2 \times 6}, \dots, \mathbf{0}_{2 \times 6}, \frac{\partial e_{ij}}{\partial \xi_i}, \mathbf{0}_{2 \times 6}, \dots, \mathbf{0}_{2 \times 3}, \dots, \mathbf{0}_{2 \times 3}, \frac{\partial e_{ij}}{\partial p_j}, \mathbf{0}_{2 \times 3}, \dots, \mathbf{0}_{2 \times 3} \right).$$

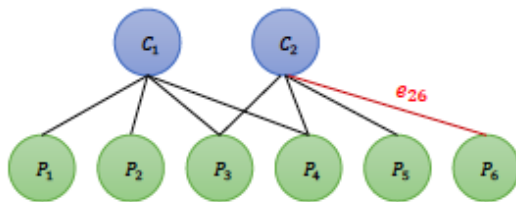
- 示意图:



2. BA与图优化

- 如果对变量做好排序，例如所有相机位姿在前，路标在后，那么 H 有一定的特殊结构：

$$H = \begin{matrix} & \begin{matrix} C_1 & C_2 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \end{matrix} \\ \begin{matrix} C_1 \\ C_2 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{matrix} & \begin{bmatrix} \text{blue block} & & & & & & & \\ & \text{blue block} & & & & & & \\ \text{red arrow} & & \text{blue block} & & & & & \\ & & & \text{blue block} & & & & \\ & & & & \text{blue block} & & & \\ & & & & & \text{blue block} & & \\ & & & & & & \text{blue block} & \\ & & & & & & & \text{blue block} \end{bmatrix} \end{matrix}$$



图模型与 H 矩阵存在对应关系：

- 图模型中存在边 $\Rightarrow H$ 相应地方出现非零块

2. BA与图优化

- 实际当中的 H ，路标数量远大于位姿数量（箭头形矩阵或镐子形矩阵）

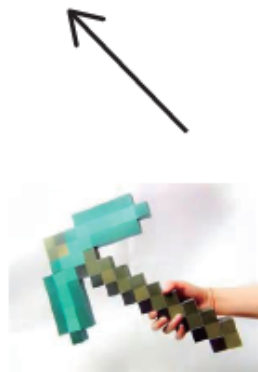
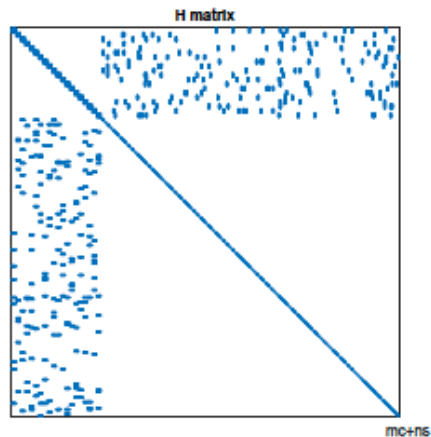
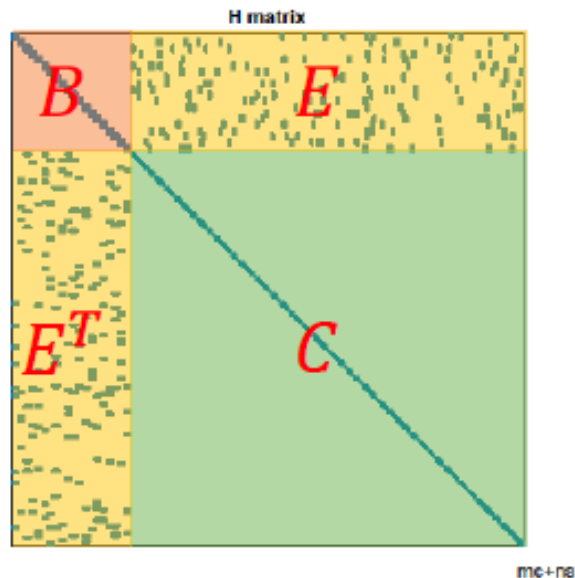


图 10-8 一般情况下的 H 矩阵。



2. BA与图优化

- 利用H的特殊结构，可以大幅加速 $Hx=-b$ 线性方程的求解
- 加速手段又称边缘化（Marginalization）[意义有很多种]
- $Hx=-b$ 的结构：

$$\begin{bmatrix} B & E \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}.$$

B：小，对角块；C，大，对角块
E和E转置：与图模型对应，可稠密

- 因为 C 是对角块，所以可以用C对E进行消元（高斯消元），得到：

$$\begin{bmatrix} I & -EC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} B & E \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} I & -EC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \quad \text{从而} \quad \begin{bmatrix} B - EC^{-1}E^T & 0 \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} v - EC^{-1}w \\ w \end{bmatrix}.$$

2. BA与图优化

- 该方程组分为两步来求：

1. 求解上半部分，规模较小，得到 Δx_c
2. 将结果代入下半部分，得到 Δx_p

- 这个做法称为Marginalization或Schur消元

- 从消元角度来讲，亦可使用Cholesky等其他消元方式解此稀疏方程
- 从Marginalization角度来讲，是我们把所有的路标信息边缘化到了相机的信息中

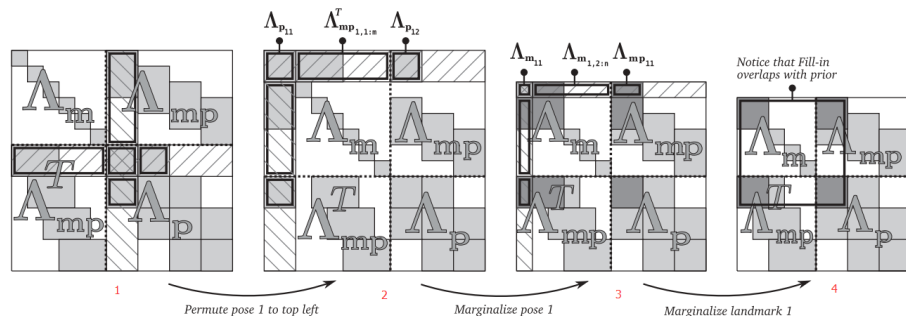
$$\begin{bmatrix} B - EC^{-1}E^T & 0 \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} v - EC^{-1}w \\ w \end{bmatrix}.$$

2. BA与图优化

- Marginalization(marg)

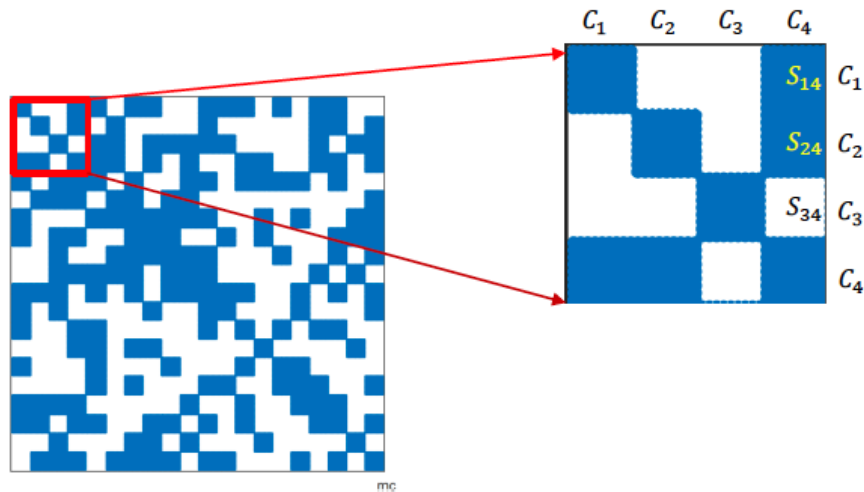
- 联合分布=边缘分布乘条件分布: $P(x_c, x_p) = P(x_c) \cdot P(x_p|x_c)$.
- 上述的做法marg掉了所有的路标点, 让概率全跑到相机位姿中, 这是为了加速求解 $Hx=-b$
- 然而, 在sliding window等其他做法中, 也可以选择边缘化一部分相机/路标, 做法同上

- 但是, 通常情况下的Marg操作会给H矩阵带来填充(fill-in), 使其不再具有稀疏结构;
- 所以, 要么刻意选择特定的marg策略, 要么仅使用稠密的H求解线性方程, 但这样效率会降低



2. BA与图优化

- Marginalization之后，左上角矩阵不再具有稀疏结构：
- 但它反映了相机之间的共视关系（Co-visibility）



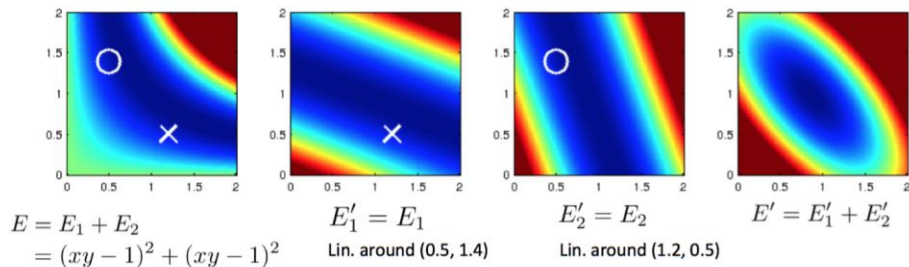
S14/S24处的非零块表示1,4相机，
2,4相机看到了同一个路标点

2. BA与图优化

- Marg操作与FEJ
- 在处理Marg时，被Marg的部分要使用First-estimate-Jacobian
- 否则可能导致整个问题零空间被降维

Windowed, real-time optimization: Consistency.

(for now, let's assume we have initializations, and know which points to use and where they are visible.)



nullspace disappears!

never combine linearizations around different linearization points,
especially in the presence of non-linear nullspaces!
It will render unobservable dimensions observable, and corrupt the system.

2. 批量方法

- 递归方法和批量方法的对比

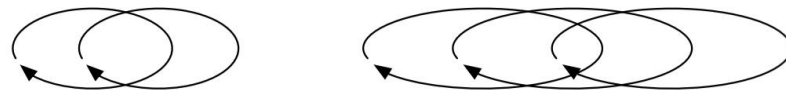
Gauss-Newton iterates over the entire trajectory, but runs offline and not in constant time

$\mathbf{x}_0 \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_{k-2} \quad \mathbf{x}_{k-1} \quad \mathbf{x}_k \quad \mathbf{x}_{k+1} \quad \mathbf{x}_{k+2} \quad \cdots \quad \mathbf{x}_K$



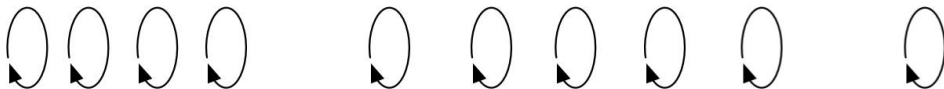
Sliding-window filters iterate over several timesteps at once, run online and in constant time

$\mathbf{x}_0 \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_{k-2} \quad \mathbf{x}_{k-1} \quad \mathbf{x}_k \quad \mathbf{x}_{k+1} \quad \mathbf{x}_{k+2} \quad \cdots \quad \mathbf{x}_K$



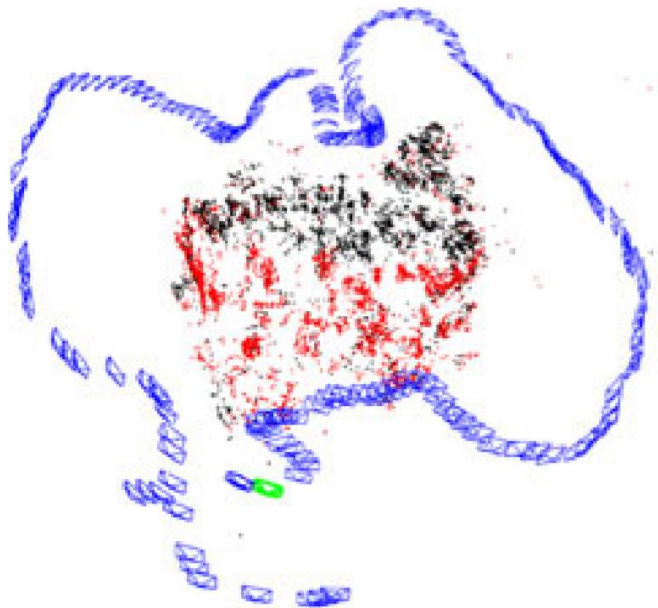
IEKF iterates at only one timestep at a time, but runs online and in constant time

$\mathbf{x}_0 \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_{k-2} \quad \mathbf{x}_{k-1} \quad \mathbf{x}_k \quad \mathbf{x}_{k+1} \quad \mathbf{x}_{k+2} \quad \cdots \quad \mathbf{x}_K$



2. 批量方法

- 在SLAM中使用BA
 - 关键帧和地图的管理
- 批量方法
 - 用BA优化一部分图
 - 其余的固定
- 递归方法（滑动窗口法）
 - 保留一定数量的关键帧
 - 使用BA来优化窗口内的关键帧
 - 新的关键帧到来时，边缘化老的关键帧



3. Pose Graph

3. Pose Graph

- 实际当中BA的计算量很大
 - 通常放在单独的后台线程中计算而无法实时
 - 主要计算来自于大量的特征点
- Pose Graph即是省略了特征点的Bundle Adjustment

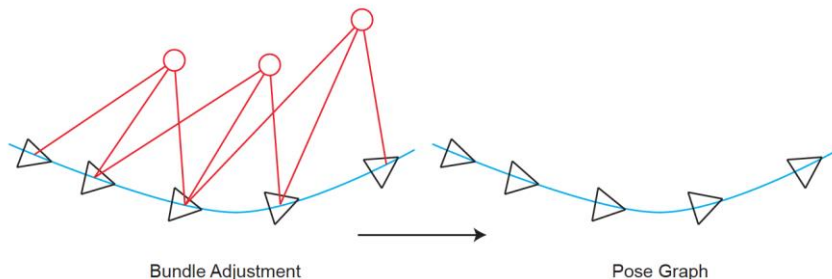


图 11-1 Pose Graph 示意图。当我们不再优化 Bundle Adjustment 中的路标点，仅把它们看成对姿态节点的约束时，就得到了一个计算规模减小很多的 Pose Graph。

3. Pose Graph

- Pose Graph

- 顶点仅由相机位姿组成
- 边为位姿与位姿间的约束

$$\Delta \xi_{ij} = \xi_i^{-1} \circ \xi_j = \ln \left(\exp \left((-\xi_i)^\wedge \right) \exp \left(\xi_j^\wedge \right) \right)^\vee,$$

$$\Delta T_{ij} = T_i^{-1} T_j.$$

- 不同的观测组成了不同的边
- 误差项:

$$\begin{aligned} e_{ij} &= \ln \left(\Delta T_{ij}^{-1} T_i^{-1} T_j \right)^\vee \\ &= \ln \left(\exp \left((-\xi_{ij})^\wedge \right) \exp \left((-\xi_i)^\wedge \right) \exp \left(\xi_j^\wedge \right) \right)^\vee. \end{aligned}$$

3. Pose Graph

- 误差项关于两个顶点的雅可比：

伴随

$$\exp((\text{Ad}(T)\xi)^\wedge) = T \exp(\xi^\wedge) T^{-1}.$$

$$\begin{aligned}\hat{e}_{ij} &= \ln(T_{ij}^{-1} T_i^{-1} \exp((- \delta \xi_i)^\wedge) \exp(\delta \xi_j^\wedge) T_j)^\vee \\ &= \ln(T_{ij}^{-1} T_i^{-1} T_j \exp((- \text{Ad}(T_j^{-1}) \delta \xi_i)^\wedge) \exp((\text{Ad}(T_j^{-1}) \delta \xi_j)^\wedge))^\vee \\ &\approx \ln(T_{ij}^{-1} T_i^{-1} T_j [I - (\text{Ad}(T_j^{-1}) \delta \xi_i)^\wedge + (\text{Ad}(T_j^{-1}) \delta \xi_j)^\wedge])^\vee \\ &\approx e_{ij} + \frac{\partial e_{ij}}{\partial \delta \xi_i} \delta \xi_i + \frac{\partial e_{ij}}{\partial \delta \xi_j} \delta \xi_j\end{aligned}$$

$$\frac{\partial e_{ij}}{\partial \delta \xi_i} = -\mathcal{J}_r^{-1}(e_{ij}) \text{Ad}(T_j^{-1}).$$

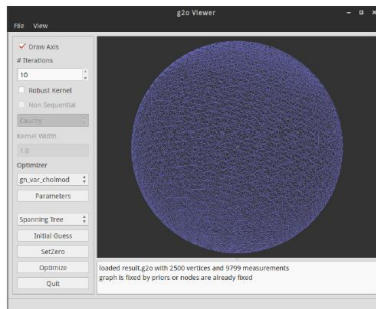
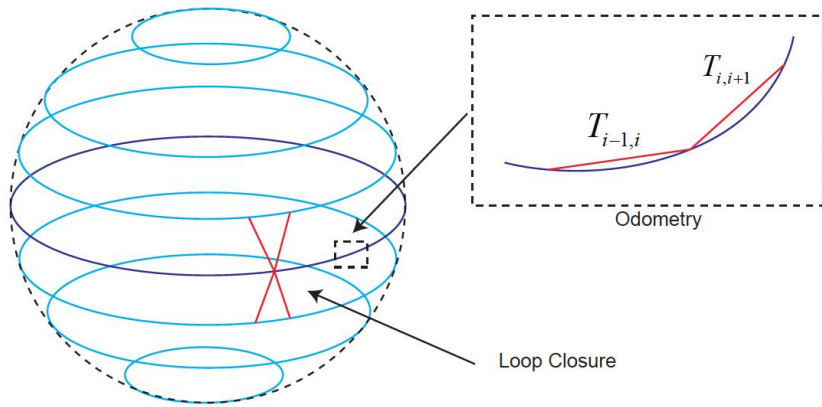
$$\frac{\partial e_{ij}}{\partial \delta \xi_j} = \mathcal{J}_r^{-1}(e_{ij}) \text{Ad}(T_j^{-1}).$$

$$\mathcal{J}_r^{-1}(e_{ij}) \approx I + \frac{1}{2} \begin{bmatrix} \phi_e^\wedge & \rho_e^\wedge \\ \mathbf{0} & \phi_e^\wedge \end{bmatrix}.$$

3. Pose Graph

- 实践：Pose球的例子

仿真的球形轨迹
Odom边和Loop边



Add Noise

