

# Lifelong Mapping

主讲人 曾书格

越凡创新技术负责人  
电子科技大学硕士





# Lifelong Mapping

## Lifelong Mapping



1. Lifelong Mapping的概念及应用



2. 冗余节点的选取



3. Pose-Graph的稠密近似



4. Pose-Graph的稀疏近似



# Lifelong Mapping

## Lifelong Mapping



**1. Lifelong Mapping的概念及应用**



2. 冗余节点的选取



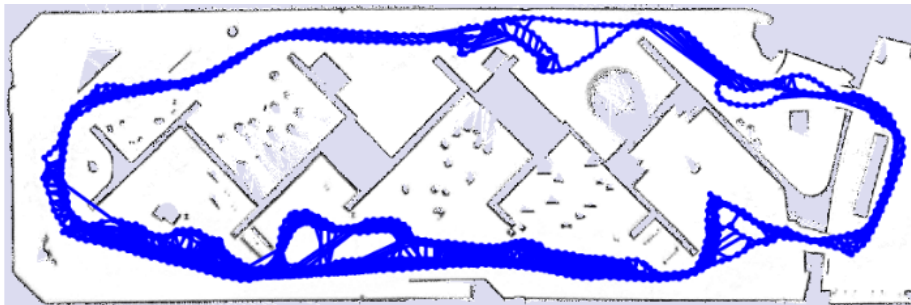
3. Pose-Graph的稠密近似



4. Pose-Graph的稀疏近似



## 示意图



普通SLAM示意图



## SLAM存在的问题

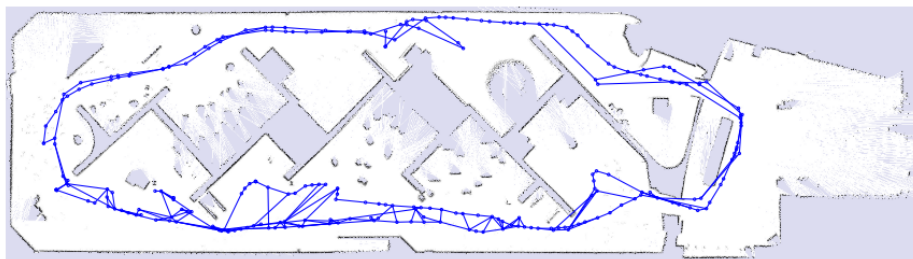
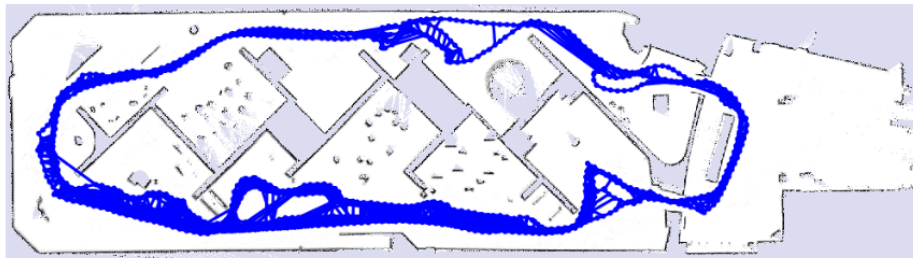
- 节点的增加通过机器人每走一定的距离或者旋转一定的角度
- 随着机器人经过距离的延长，PoseGraph中的节点越来越多，求解的规模不断增大，导致优化所需的时间越来越长。
- PoseGraph的大小和机器人走过的距离长度成正比，经过的距离越长，PoseGraph中的节点越多。



# Lifelong Mapping



## 示意图

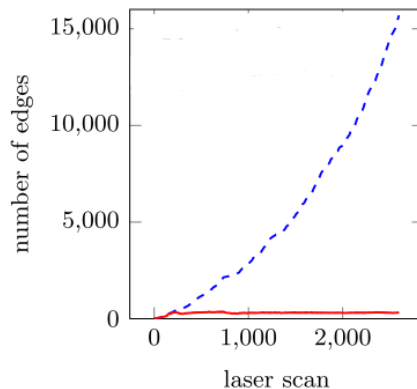


Lifelong Mapping示意图



## 基本思想

- 实现机器人边导航边建图
- PoseGraph的规模不能随着运行轨迹的增长而增加
- PoseGraph的规模应该和建图面积相关

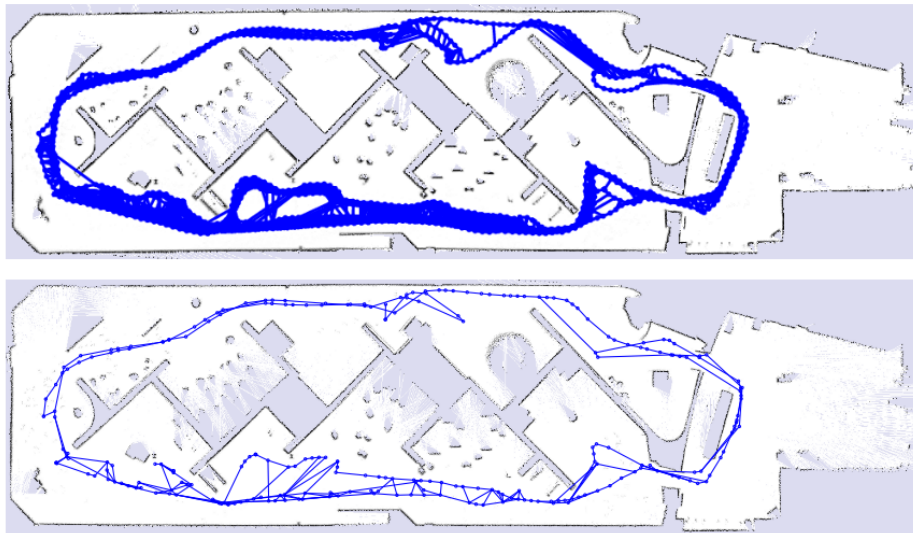




# Lifelong Mapping



## 示意图



Lifelong Mapping示意图



## 实现思路

- 定期删除Pose-Graph中冗余的节点
- 哪个节点是冗余的?
- 从PoseGraph中删除一个冗余节点, PoseGraph需要做什么样的调整?



# Lifelong Mapping

## Lifelong Mapping



1. Lifelong Mapping的概念及应用



**2. 冗余节点的选取**



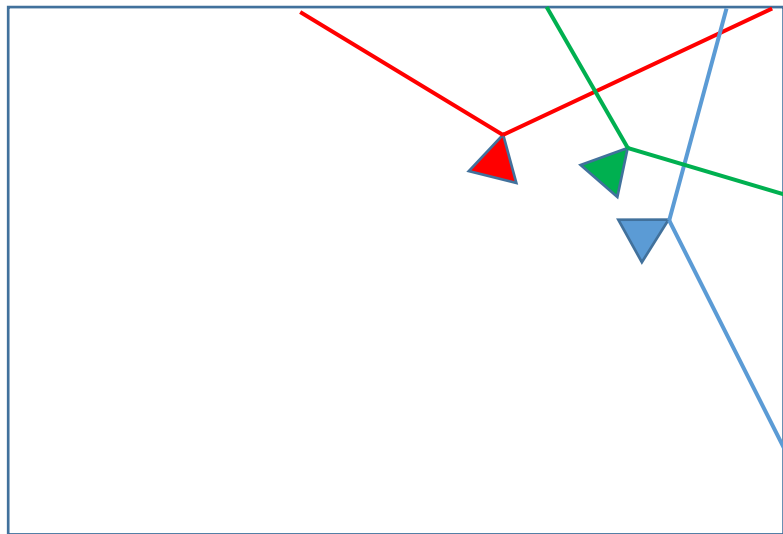
3. Pose-Graph的稠密近似



4. Pose-Graph的稀疏近似



## 冗余节点的定义



冗余节点示意图

- SLAM的目的是构建一个环境的地图
- 如果去除某一个节点，构建出来的地图不发生变化或者变化很小，则认为该节点是一个冗余节点(绿色的节点)。
- 冗余节点的数据被其他节点覆盖住，因此可以直接删除而不影响地图的生成。





# Lifelong Mapping



## 冗余节点的选择

- 构建的地图为栅格地图，地图Cell一共有两种状态：占用(Occupied)，空闲(Free)。
- 每个栅格存储被击中的次数 以及 被穿过的次数，该栅格是障碍物的概率：

$$p(m_i) = \frac{hits(i)}{hits(i) + misses(i)}$$

- 如果某栅格是障碍物的概率大于阈值，则认为该栅格是障碍物。
- PoseGraph中的每一个节点都有一帧对应的激光数据，每个节点中存储该激光数据穿过的栅格以及击中的栅格。

			1,0	1,0	1,0
			0,1	0,1	0,1
0,1	0,1	0,1	1,0	0,1	1,0
0,1	0,1	0,1	1,0	0,1	1,0
0,1	0,1	0,1	0,1	1,0	0,1
			0,1	0,1	1,0
			1,0	0,1	0,1

节点*i*对应的数据

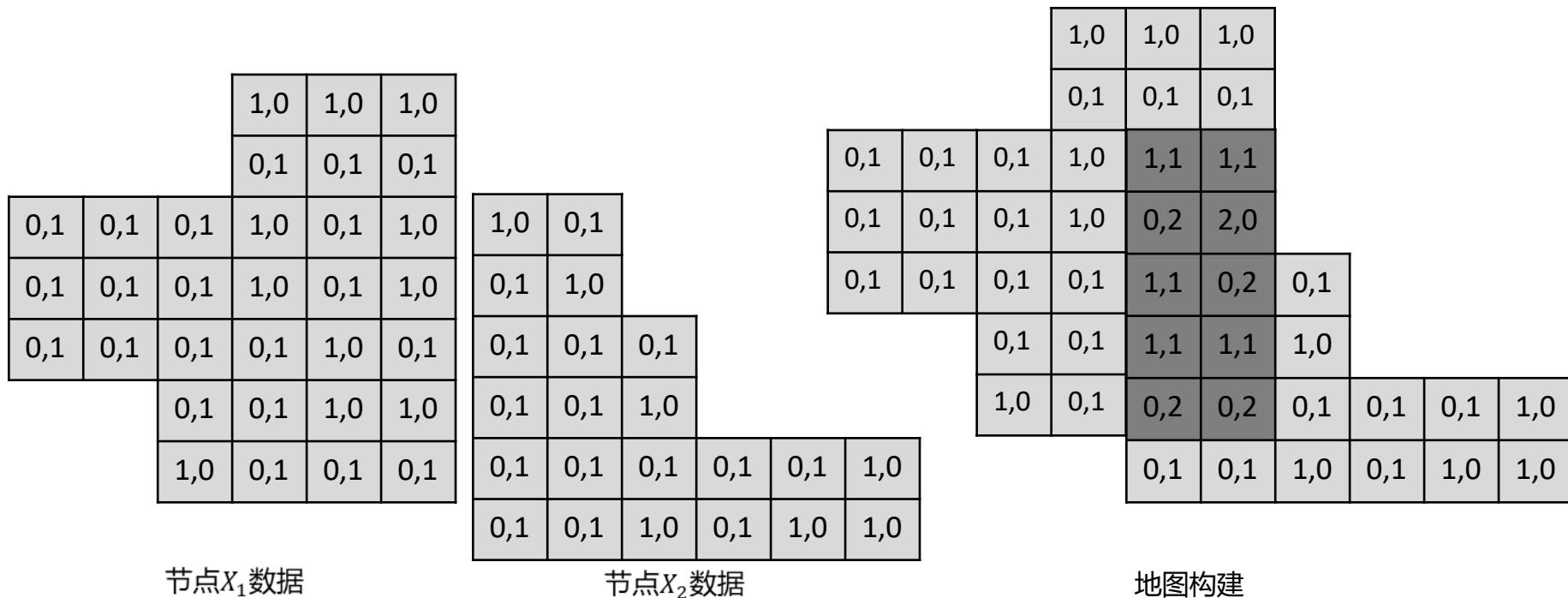


# Lifelong Mapping



## 冗余节点的选择

- 全部的节点数据构建一个完整的地图A，得到每一个栅格**被击中**和**被穿过**的次数。





# Lifelong Mapping



## 冗余节点的选择

- 对于任意一个节点 $i$ ，从地图A中去除该节点的数据，即节点 $i$ 穿过的栅格对应的穿过次数减1，节点 $i$ 击中的栅格对应的击中次数减1。

			1,0	1,0	1,0				
			0,1	0,1	0,1				
0,1	0,1	0,1	2,2	2,1	2,1	2,1	1,0	2,1	
0,1	0,1	0,1	3,0	1,2	2,2	1,2	0,1	0,1	
0,1	0,1	0,1	2,1	2,1	1,2	1,1	1,0	2,2	
		0,1	1,2	3,1	2,1	2,0	0,1	1,0	
		1,0	0,3	1,2	0,2	0,1	0,1	0,1	1,0
				0,1	0,1	1,0	0,1	1,0	1,0

			1,0	1,0	1,0				
			0,1	0,1	0,1				
0,1	0,1	0,1	1,2	1,1	1,1	1,1	1,0	2,1	
0,1	0,1	0,1	1,0	1,1	2,1	0,2	0,1	0,1	
0,1	0,1	0,1	2,0	1,1	0,2	1,1	1,0	2,2	
		0,1	1,1	3,1	1,1	2,0	0,1	1,0	
		1,0	0,3	1,2	0,2	0,1	0,1	0,1	1,0
				0,1	0,1	1,0	0,1	1,0	1,0



# Lifelong Mapping



## 冗余节点的选择

- 统计状态发生变化的栅格个数，作为该节点*i*冗余度的评估值。

2,2	2,1	2,1	2,1
3,0	1,2	2,2	1,2
2,1	2,1	1,2	
1,2	3,1	2,1	

$$p(m_i) = \frac{hits(i)}{hits(i) + misses(i)}$$

0.50	0.67	0.67	0.67
1.00	0.33	0.50	0.33
0.67	0.67	0.33	
0.33	0.75	0.67	

1,2	1,1	1,1	1,1
1,0	1,1	2,1	0,2
2,0	1,1	0,2	
1,1	3,1	1,1	

$$p(m_i) = \frac{hits(i)}{hits(i) + misses(i)}$$

0.33	0.50	0.50	0.50
1.00	0.50	0.67	0.00
1.00	0.50	0.00	
0.50	0.75	0.50	

假设阈值为0.6

	■	■	■
		■	
	■		
		■	

删除节点*i*时，6个栅格的状态发生变化



## 冗余节点的选择

- 地图A中删除节点 $i$ 时，状态发生改变的栅格数量越小，说明节点 $i$ 对应的该帧数据越冗余；状态发生改变的栅格数量越大，说明该帧数据越不冗余。
- 依次计算每一个节点的冗余度，选择最冗余的且冗余度小于一定阈值的节点作为待删除的节点。



# Lifelong Mapping

## Lifelong Mapping



1. Lifelong Mapping的概念及应用



2. 冗余节点的选取



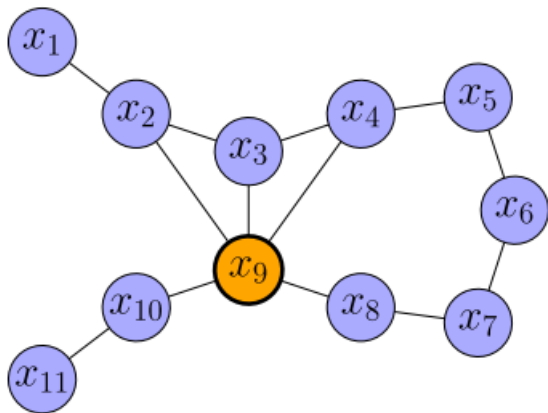
**3. Pose-Graph的稠密近似**



4. Pose-Graph的稀疏近似

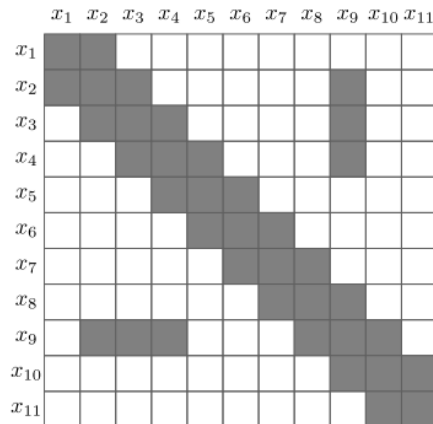


## Pose-Graph的稠密近似



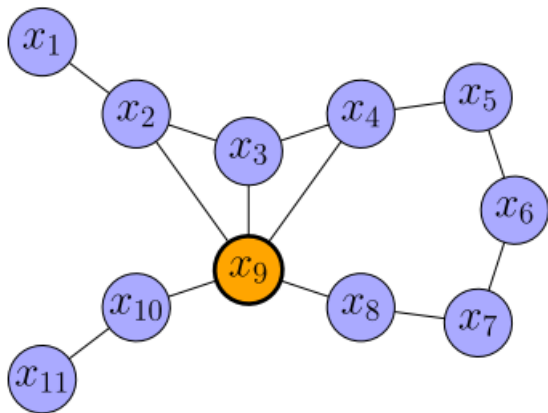
### 冗余节点示意图

- 橙色的节点表示待删除的冗余点，也就是需要边缘化(marginalization)的节点
- 对应的信息矩阵(Hessian矩阵):





## Pose-Graph的稠密近似



冗余节点示意图

- 边缘化的过程：

$$p(\alpha, \beta) = \mathcal{N}^{-1} \left( \begin{bmatrix} \xi_{\alpha} \\ \xi_{\beta} \end{bmatrix}, \begin{bmatrix} \Omega_{\alpha\alpha} & \Omega_{\alpha\beta} \\ \Omega_{\beta\alpha} & \Omega_{\beta\beta} \end{bmatrix} \right)$$



多元高斯分布的边缘分布也是高斯分布（比如  $p(\alpha)$ ），且其信息向量与信息矩阵分别为：

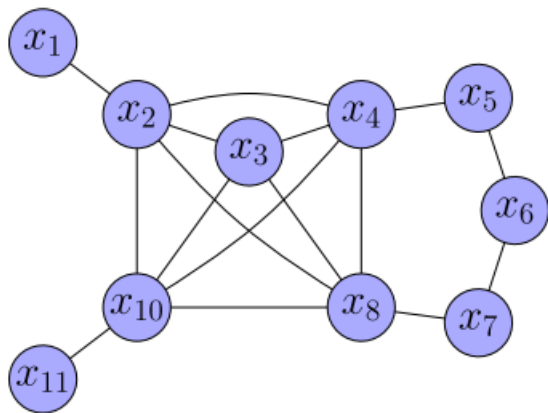
$$\xi = \xi_{\alpha} - \Omega_{\alpha\beta} \Omega_{\beta\beta}^{-1} \xi_{\beta}$$

$$\Omega = \Omega_{\alpha\alpha} - \Omega_{\alpha\beta} \Omega_{\beta\beta}^{-1} \Omega_{\beta\alpha}$$



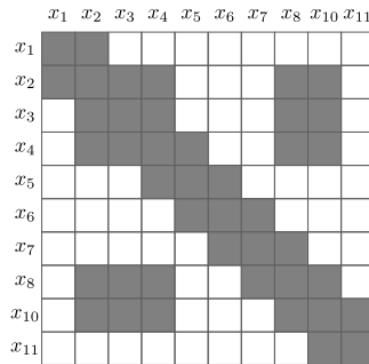


## Pose-Graph的稠密近似



### 精确边缘化后的Pose Graph

- 精确边缘化后, PoseGraph中待删除节点的相邻节点之间, 形成了一个完全子图, 让整个 PoseGraph变得更加稠密。
- 对应的信息矩阵:





## 完全子图的边

- Edge-Reversal:

第 $i$ 个节点与第 $j$ 个节点之间的相对位姿服从高斯分布:  $e_{ij} \sim N(u_{ij}, \Sigma_{ij})$  , 其中  $u_{ij} = (t_{ij}, \theta_{ij})$  , 则第 $j$ 个节点与第 $i$ 个节点之间的相对位姿也服从高斯分布:

$$e_{ji} \sim N(u_{ji}, \Sigma_{ji})$$

其中  $u_{ji} = -Ru_{ij}$  ,  $\Sigma_{ji} = R\Sigma_{ij}R^T$  ,  $R = \begin{bmatrix} R^T(\theta_{ij}) & 0 \\ 0 & 1 \end{bmatrix}$

证明:

$$T_{ij} = \begin{bmatrix} R(\theta_{ij}) & t_{ij} \\ 0 & 1 \end{bmatrix}$$

$$T_{ji} = T_{ij}^{-1} = \begin{bmatrix} R^T(\theta_{ij}) & -R^T(\theta_{ij})t_{ij} \\ 0 & 1 \end{bmatrix}$$

$$u_{ji} = (-R^T(\theta_{ij})t_{ij}, -\theta_{ij}) = -\begin{bmatrix} R^T(\theta_{ij}) & 0 \\ 0 & 1 \end{bmatrix} u_{ij}$$



## 完全子图的边

- Edge-Composition:

第 $i$ 个节点与第 $j$ 个节点之间的相对位姿服从高斯分布  $e_{ij} = (u_{ij}, \Sigma_{ij})$  , 第 $j$ 个节点与第 $k$ 个节点之间的相对位姿服从高斯分布  $e_{jk} = (u_{jk}, \Sigma_{jk})$  , 则第 $i$ 个节点与第 $k$ 个节点之间的相对位姿服从高斯分布:

$$e_{ik} = (u_{ik}, \Sigma_{ik})$$

其中,  $u_{ik} = u_{ij} + Ru_{jk}$

$$\Sigma_{ik} = \Sigma_{ij} + R\Sigma_{jk}R^T$$

$$R = \begin{bmatrix} R(\theta_{ij}) & 0 \\ 0 & 1 \end{bmatrix}$$

证明:

$$T_{ik} = T_{ij}T_{jk} = \begin{bmatrix} R_{ij}R_{jk} & R_{ij}t_{jk} + t_{ij} \\ 0 & 1 \end{bmatrix}$$

$$\begin{aligned} u_{ik} &= T2V(T_{ik}) = \begin{bmatrix} t_{ij} + R_{ij}t_{jk} \\ \theta_{ij} + \theta_{jk} \end{bmatrix} = \begin{bmatrix} t_{ij} \\ \theta_{ij} \end{bmatrix} + \begin{bmatrix} R_{ij} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_{jk} \\ \theta_{jk} \end{bmatrix} \\ &= u_{ij} + Ru_{jk} \end{aligned}$$



# Lifelong Mapping



## 完全子图的边

- Edge-Combine:

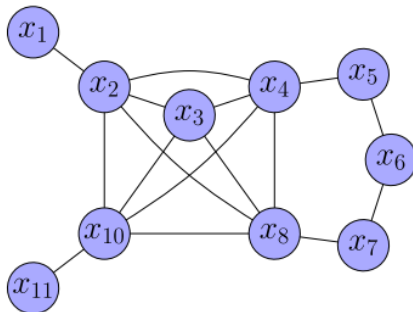
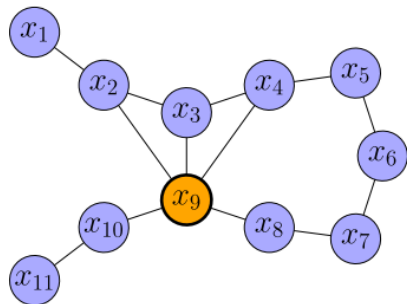
第 $i$ 个节点与第 $j$ 个节点之间的相对位姿服从高斯分布  $e_{ij} = (u_{ij}, \Sigma_{ij})$  , 第 $i$ 个节点与第 $j$ 个节点之间通过组合得到的相对位姿服从高斯分布  $e'_{ij} = (u'_{ij}, \Sigma'_{ij})$  , 则第 $i$ 个节点与第 $j$ 个节点之间的相对位姿服从高斯分布:

$$e = (u, \Sigma)$$

其中,  $u = \Sigma(\Sigma_0^{-1}u_0 + \Sigma_1^{-1}u_1)$

$$\Sigma = (\Sigma_0^{-1} + \Sigma_1^{-1})^{-1}$$

证明: 两者的加权和, 权重为各自的信息矩阵, 显然是成立的。



对比示意图

- 纯组合边:

$$e_{24} = edge\_composition(e_{29}, e_{94})$$

$$e_{28} = edge\_composition(e_{29}, e_{98})$$

$$e_{210} = edge\_composition(e_{29}, e_{910})$$

$$e_{38} = edge\_composition(e_{39}, e_{98})$$

$$e_{310} = edge\_composition(e_{39}, e_{910})$$

$$e_{48} = edge\_composition(e_{49}, e_{98})$$

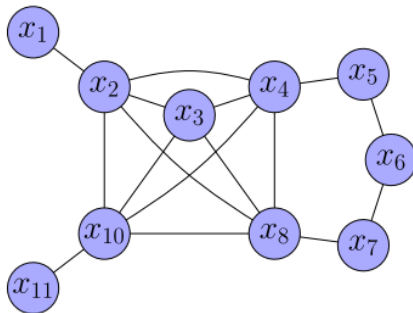
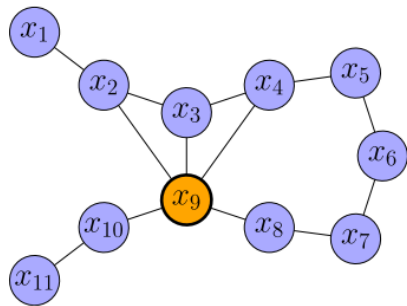
$$e_{410} = edge\_composition(e_{49}, e_{910})$$



# Lifelong Mapping



## Pose-Graph的稠密近似



对比示意图

- 复合边:

$$e'_{23} = edge\_composition(e_{29}, e_{93})$$

$$e_{23} = edge\_combine(e'_{23}, e_{23})$$

$$e'_{34} = edge\_composition(e_{39}, e_{94})$$

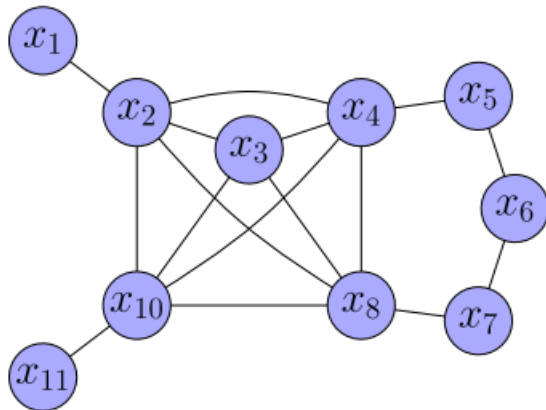
$$e_{34} = edge\_combine(e'_{34}, e_{34})$$



# Lifelong Mapping



## Pose-Graph的稠密近似



## 精确边缘化后的Pose Graph

- 每删除一个节点，就会在pose-graph中形成一个完全子图，让图变得更加的稠密。
- 随着删除的节点越来越多，图就会变得越来越稠密，破坏Hessian矩阵的稀疏性质，无法利用矩阵的稀疏性进行快速求解，会极大的减慢优化的速度。



## 对完全子图进行稀疏近似



# Lifelong Mapping

## Lifelong Mapping



1. Lifelong Mapping的概念及应用



2. 冗余节点的选取



3. Pose-Graph的稠密近似

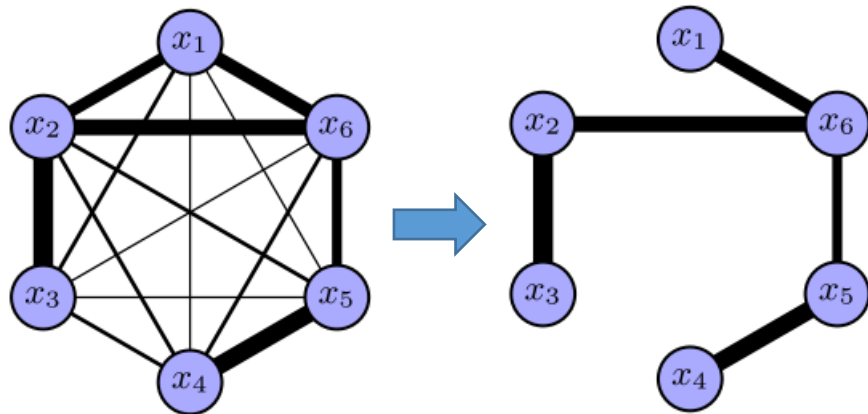


**4. Pose-Graph的稀疏近似**





## Chow-Liu Tree



Chow-Liu Tree 近似

- 图中边的厚度表示两个变量之间互信息的大小
- Chow-Liu Tree近似等价于最大生成树
- 数学表示:

$$p(\tilde{\mathbf{x}}) = p(x_k) \prod_{i=1}^{k-1} p(x_i \mid x_{i+1}, \dots, x_k) \\ \approx p(x_k) \prod_{i=1}^{k-1} p(x_i \mid x_{i+1}) = q(\tilde{\mathbf{x}}).$$



## Lifelong Mapping



### Chow-Liu Tree

- 高斯分布变量的互信息计算方法：

$$I(x_i; x_j) = \frac{1}{2} \log_2 \left( \frac{|\tilde{\Sigma}_{ii}|}{|\tilde{\Sigma}_{ii} - \tilde{\Sigma}_{ij} \tilde{\Sigma}_{jj}^{-1} \tilde{\Sigma}_{ji}|} \right)$$

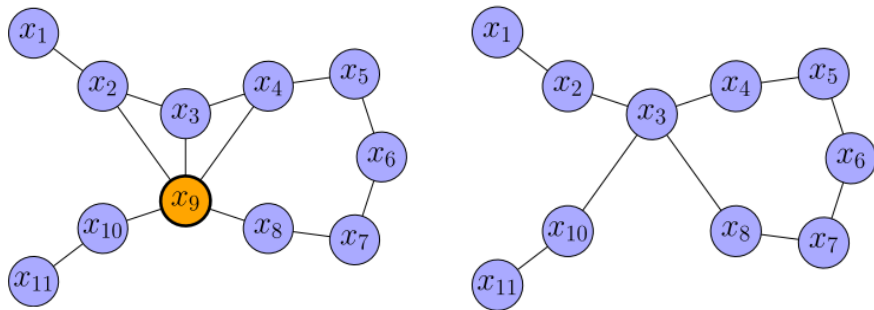
$x_i, x_j$  表示完全子图中的两个节点

$\tilde{\Sigma}$  表示完全子图的协方差矩阵

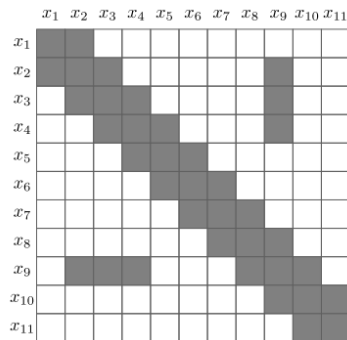
- 完全子图的分布：均值就是目前各个节点的位姿，协方差矩阵通过构造全图的信息矩阵(第六节课程的内容)，然后进行边缘化得到完全子图的信息矩阵，最后求逆得到完全子图的协方差矩阵即可。



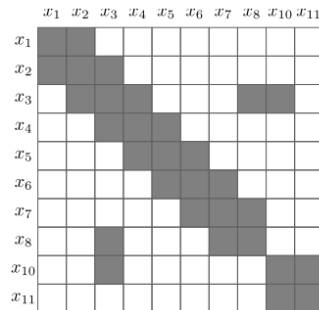
- 信息矩阵(Hessian矩阵):



稀疏近似对比图



原始Hessian矩阵



近似Hessian矩阵



## Lifelong Mapping



### 稀疏近似的流程

- 1. 按照上述方法找到待删除的节点 $i$ ;
- 2. 对于pose-graph进行边缘化, 得到去除节点 $i$ 之后的整体分布A;
- 3. 从整体分布A中继续进行边缘化, 得到完全子图的分布B;
- 4. 计算完全子图的各个节点之间的互信息, 用互信息作为边的权重, 构造图;
- 5. 用最大生成树决定要保留的边。



[1]Information-theoretic compression of pose graphs for laser-based SLAM



作业



详细见作业说明



结语

感谢聆听！  
Thanks for Listening

