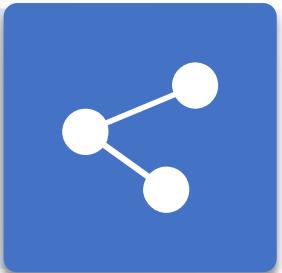




# Distributed Trajectory Estimation with Privacy and Communication Constraints: a Two-Stage Distributed Gauss-Seidel Approach

**Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Henrik I. Christensen, Frank Dellaert**  
**IEEE International Conference on Robotics and Automation (ICRA), 2016**

智能网络与优化实验室





1

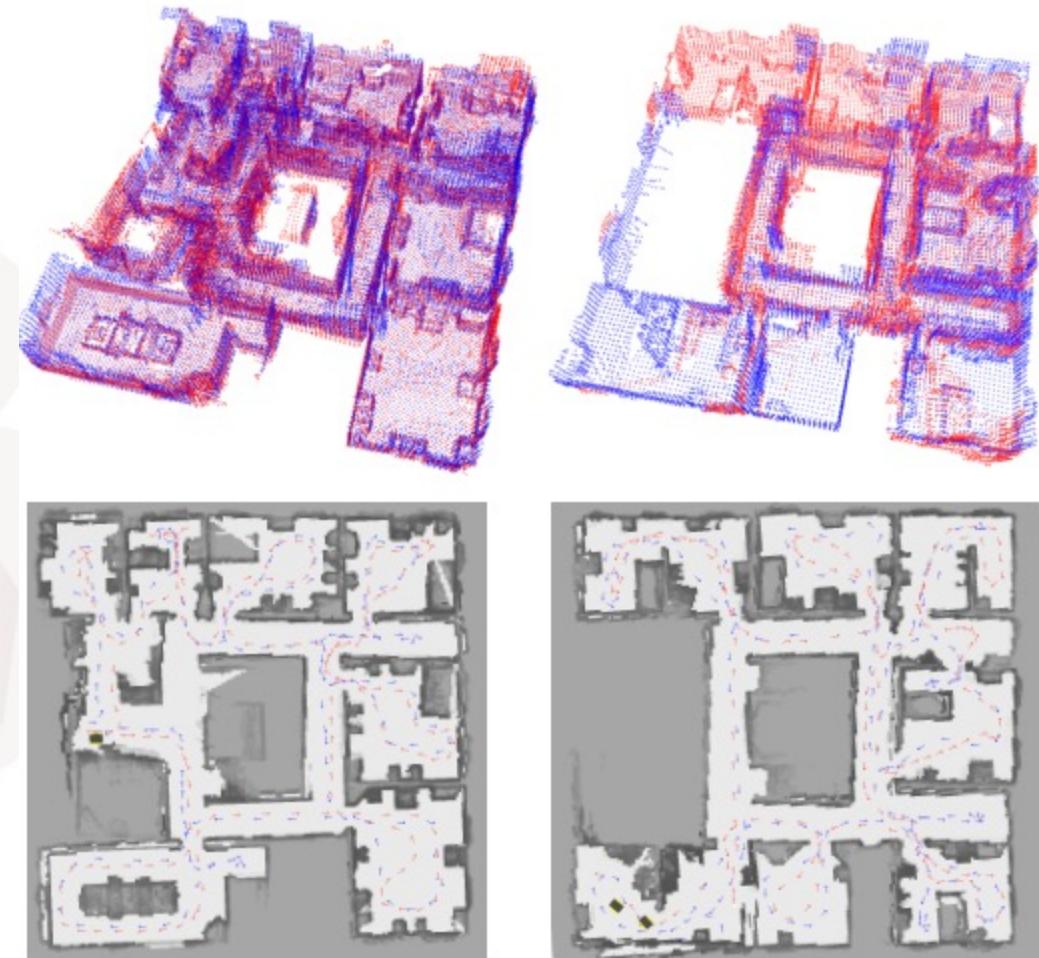
# 问题提出



## 问题提出

本文解决了传感器融合问题的一个具体实例：考虑一个机器人团队探索未知环境的情况，每个机器人必须根据自己的传感器数据并利用与其他机器人交换的信息来估计其轨迹。轨迹估计是许多估计和控制任务（例如，地理标记传感器数据、3D 地图重建、位置感知任务分配）的支柱。事实上，在我们的应用中，轨迹估计可以实现分布式 3D 重建和定位（右图）。

图中为两个机器人（红色和蓝色）协同建图结果，每一列为一次建图，上为重建点云，下为机器人的估计轨迹。





## 问题提出

考虑一个现实场景：

- 其中机器人仅在给定距离内进行通信；
- 在会合期间（即，当机器人距离足够近以进行通信时），由于带宽限制，它们也无法交换大量信息。

目标是设计一种技术，允许每个机器人估计自己的轨迹，同时要求对队友的轨迹有最少的了解。这种“隐私约束”在军事应用中具有明确的动机：如果一个机器人被捕获，它就无法提供有关其他机器人所覆盖区域的敏感信息。

- 考虑一个分布式 MLE 轨迹估计问题，其中机器人必须协同估计它们的轨迹，同时最小化交换信息的数量；
- 考虑一个完全分布式的情况，其中机器人只能在发生交会时进行通信和获取数据。



2

# 论文思路

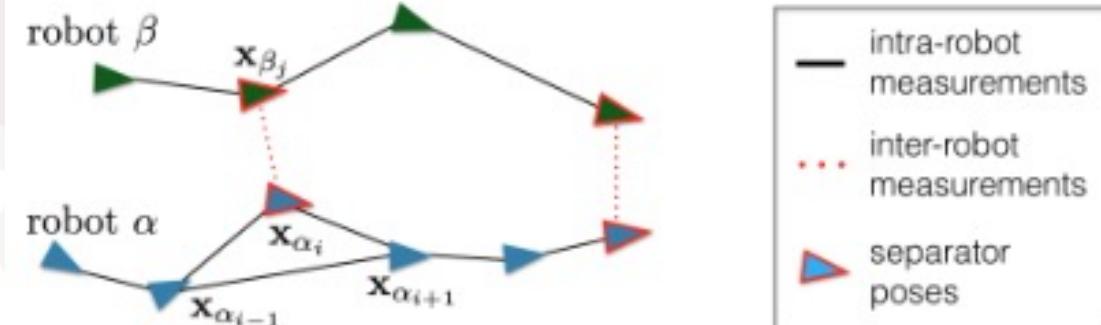


# 多机器人轨迹估计

## 1. 符号

设机器人系统中包含的机器人为  $\Omega = \{\alpha, \beta, \gamma, \dots\}$ ；将轨迹模型化为一个有限的位姿集合（如图）；记机器人  $\alpha$  在时刻  $i$  的位姿为  $x_{\alpha_i} \in SE(3)$ ，在某些情况下也记为  $x_{\alpha_i} = (R_{\alpha_i}, t_{\alpha_i})$ ,  $R_{\alpha_i} \in SO(3)$  and  $t_{\alpha_i} \in \mathbb{R}^3$ ，机器人  $\alpha$  的位姿则可以记为

$$x_\alpha = \{x_{\alpha_1}, x_{\alpha_2}, \dots\}$$





## 多机器人轨迹估计

### 2. 位姿计算

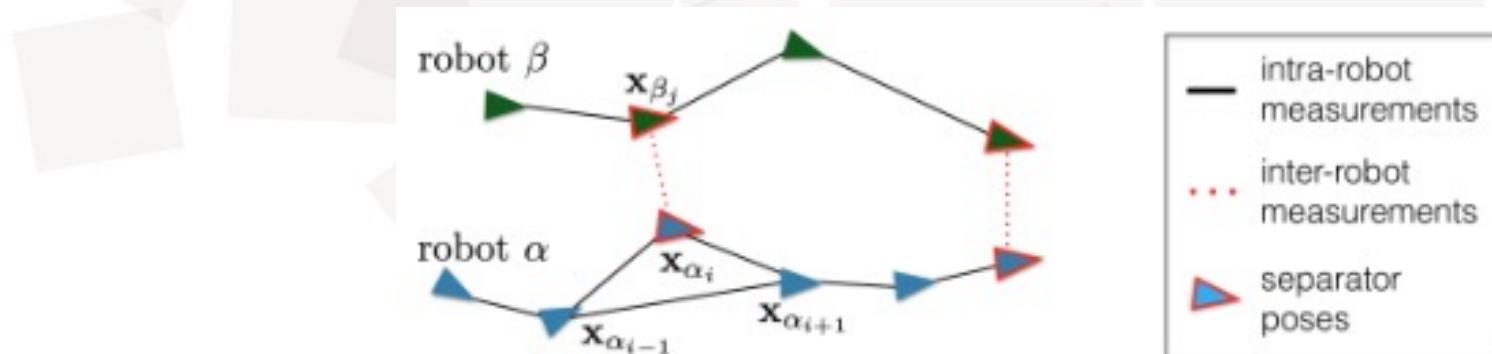
- intra-robot : 机器人内位姿计算

- 惯导 ( 图中  $x_{\alpha_i}$  和  $x_{\alpha_{i+1}}$  )

- 回环检测 ( 图中  $x_{\alpha_{i-1}}$  和  $x_{\alpha_i}$  )

- inter-robot : 机器人间位姿计算

- 相对位姿 ( 图中  $x_{\alpha_i}$  和  $x_{\beta_j}$  ) : 使用分离子 ( separator ) 描述





## 多机器人轨迹估计

### 2. 位姿计算

虽然位姿测量（内部与外部）是基于测量过程中涉及的机器人，但所有相关测量都可以在同一测量模型中构建。可以将带有噪声的位姿计算模型写为下式

$$\bar{\mathbf{z}}_{\beta_j}^{\alpha_i} \doteq \left( \bar{\mathbf{R}}_{\beta_j}^{\alpha_i}, \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} \right), \quad \text{with: } \begin{cases} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} = (\mathbf{R}_{\alpha_i})^T \mathbf{R}_{\beta_j} \mathbf{R}_\epsilon \\ \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} = (\mathbf{R}_{\alpha_i})^T (\mathbf{t}_{\beta_j} - \mathbf{t}_{\alpha_i}) + \mathbf{t}_\epsilon \end{cases}$$

式中  $\bar{\mathbf{z}}_{\beta_j}^{\alpha_i}$  表示相对位姿计算， $\bar{\mathbf{R}}_{\beta_j}^{\alpha_i}$  表示了  $\mathbf{R}_{\beta_j}$  相对帧  $\alpha_i$  下的姿态， $\mathbf{R}_\epsilon$  为随机噪声（ $\bar{\mathbf{t}}_{\beta_j}^{\alpha_i}$  和  $\mathbf{t}_\epsilon$  的定义类似于  $\bar{\mathbf{R}}_{\beta_j}^{\alpha_i}$  和  $\mathbf{R}_\epsilon$ ）。

如上式定义，则 intra-robot 相对位姿可以写为  $\bar{\mathbf{z}}_{\alpha_k}^{\alpha_i} (i \neq k)$ ；而 inter-robot 写为  $\bar{\mathbf{z}}_{\beta_j}^{\alpha_i} (\alpha \neq \beta)$ 。



## 多机器人轨迹估计

### 2. 位姿计算

在后续过程中，标记  $\mathcal{E}_I^\alpha$  为机器人  $\alpha$  的 intra-robot 位姿计算集合，所有的机器人的位姿计算集合构成

$$\mathcal{E}_I = \bigcup_{\alpha \in \Omega} \mathcal{E}_I^\alpha$$

标记  $\mathcal{E}_S^\alpha$  为包含机器人  $\alpha$  的 inter-robot 位姿计算集合，同理有

$$\mathcal{E}_S = \bigcup_{\alpha \in \Omega} \mathcal{E}_S^\alpha$$

所有需要计算的位姿构成的集合为

$$\mathcal{E} = \mathcal{E}_I \cup \mathcal{E}_S$$



## 多机器人轨迹估计

### 3. 轨迹的 MLE ( 极大似然 ) 估计

记  $x = \{x_\alpha, x_\beta, \dots\}$  为所有机器人带估计的位姿集合，则对  $x$  的计算使用 MLE

$$\bar{\mathbf{z}}_{\beta_j}^{\alpha_i} \doteq \left( \bar{\mathbf{R}}_{\beta_j}^{\alpha_i}, \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} \right), \quad \text{with: } \begin{cases} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} = (\mathbf{R}_{\alpha_i})^T \mathbf{R}_{\beta_j} \mathbf{R}_\epsilon \\ \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} = (\mathbf{R}_{\alpha_i})^T (\mathbf{t}_{\beta_j} - \mathbf{t}_{\alpha_i}) + \mathbf{t}_\epsilon \end{cases}$$

$$x^* = \operatorname{argmax}_{x \in SE(3)} \prod_{(\alpha_i, \beta_j) \in \mathcal{E}} \mathcal{L}(\bar{\mathbf{z}}_{\beta_j}^{\alpha_i} | x) = \operatorname{argmin}_{x \in SE(3)} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} -\ln \mathcal{L}(\bar{\mathbf{R}}_{\beta_j}^{\alpha_i} | x) - \ln \mathcal{L}(\bar{\mathbf{t}}_{\beta_j}^{\alpha_i} | x)$$

上式的似然函数依赖于噪声  $\mathbf{R}_\epsilon$  和  $\mathbf{t}_\epsilon$ 。文中，假设：

- $\mathbf{t}_\epsilon$  服从均值为 0，信息矩阵为  $\omega_t^2 \mathbf{I}_3$  的高斯分布
- $\mathbf{R}_\epsilon$  服从  $\mu = \mathbf{I}_3, \kappa = \omega_R^2$  的冯·米塞斯 ( Von Mises ) 分布

注：Von Mises 分布：

$$\text{数值表示 : } f(x|\mu, \kappa) = \frac{e^{\kappa \cos(x-\mu)}}{2\pi I_0(\kappa)}, \quad \text{矩阵表示 : } \mathbb{P}(A) = \frac{\exp(\kappa \operatorname{tr}(\mu^T A))}{c_n(\kappa)}$$

其中  $I_0(x)$  为 0 阶 Bessel 修正函数， $c_n(\kappa)$  为常量。



## 多机器人轨迹估计

### 3. 轨迹的 MLE ( 极大似然 ) 估计

在前面的假设下，对  $x = \{(\mathbf{R}_{\alpha_i}, \mathbf{t}_{\alpha_i}) | \forall \alpha \in \Omega, \forall i\}$  的 MLE 估计可以转化为求下面的优化问题

$$\min_{\substack{\mathbf{t}_{\alpha_i} \in \mathbb{R}^3 \\ \mathbf{R}_{\alpha_i} \in SO(3) \\ \forall \alpha \in \Omega, \forall i}} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \mathbf{t}_{\beta_j} - \mathbf{t}_{\alpha_i} - \mathbf{R}_{\alpha_i} \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} \right\|^2 + \frac{\omega_R^2}{2} \left\| \mathbf{R}_{\beta_j} - \mathbf{R}_{\alpha_i} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} \right\|_F^2$$

式中  $\left\| \mathbf{R}_{\beta_j} - \mathbf{R}_{\alpha_i} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} \right\|_F^2$  使用 Frobenius 范式  $\|\cdot\|_F$  来表示旋转误差。

- 注 : Frobenius 范式 : 对于矩阵  $A \in \mathbb{R}^{m \times n}$ ,  $\|A\|_F = \sqrt{\text{tr}(A^T A)} = \sqrt{\sum_{i,j} a_{ij}^2}$ 。

在 centralized 环境下解决此问题通常是收集所有的测量值  $\mathcal{E}$ ，然后使用迭代优化或快速近似来解决，而本文考虑的情况下，将原本的问题描述为：( 下页 )

$$\bar{\mathbf{z}}_{\beta_j}^{\alpha_i} \doteq \left( \bar{\mathbf{R}}_{\beta_j}^{\alpha_i}, \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} \right), \quad \text{with: } \begin{cases} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} = (\mathbf{R}_{\alpha_i})^T \mathbf{R}_{\beta_j} \mathbf{R}_\epsilon \\ \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} = (\mathbf{R}_{\alpha_i})^T (\mathbf{t}_{\beta_j} - \mathbf{t}_{\alpha_i}) + \mathbf{t}_\epsilon \end{cases}$$



## 多机器人轨迹估计

### 3. 轨迹的 MLE ( 极大似然 ) 估计

分布式轨迹估计问题 ( Distributed Trajectory Estimation ) :

设计一个算法，使得机器人  $\alpha$  在与其他机器人 ( $\Omega_r \subseteq \Omega \setminus \{\alpha\}$ ) 发生会和时执行，并且算法满足：

- 输入：

( i ) intra-robot 位姿计算  $\varepsilon_I^\alpha$

( ii ) inter-robot 位姿计算  $\varepsilon_S^\alpha$

( iii ) 机器人  $\beta \in \Omega_r$  轨迹的部分估计

- 输出：对  $x_\alpha^*$  的 MLE 估计，其中  $x^* = \{x_\alpha^*, x_\beta^*, \dots\}$ 。

注意到， $\varepsilon_I^\alpha$  和  $\varepsilon_S^\alpha$  是机器人  $\alpha$  所已知的，而 ( iii ) 中的内容则需要通过通信来获取，因此还要使通信的轨迹估计数据尽可能少。

下一部分将提出此问题的解决方法。



## 两阶段分布式轨迹估计

$$\min_{\substack{\mathbf{t}_{\alpha_i} \in \mathbb{R}^3 \\ \mathbf{R}_{\alpha_i} \in SO(3) \\ \forall \alpha \in \Omega, \forall i}} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \mathbf{t}_{\beta_j} - \mathbf{t}_{\alpha_i} - \mathbf{R}_{\alpha_i} \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} \right\|^2 + \frac{\omega_R^2}{2} \left\| \mathbf{R}_{\beta_j} - \mathbf{R}_{\alpha_i} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} \right\|_F^2$$

第一阶段：解决优化问题的松弛版本，得到所有机器人  $R_{\alpha_i}$  的估计；

第二阶段：利用 Gauss-Newton 法迭代计算完整的位姿。



## 两阶段分布式轨迹估计（集中式）

### 1. 第一阶段（松弛和映射）

通过下式计算机器人的旋转估计

$$\min_{\mathbf{R}_{\alpha_i} \in SO(3)} \sum_{\substack{(\alpha_i, \beta_j) \in \mathcal{E} \\ \forall \alpha \in \Omega, \forall i}} \omega_R^2 \left\| \mathbf{R}_{\beta_j} - \mathbf{R}_{\alpha_i} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} \right\|_F^2$$

通过丢弃约束  $\mathbf{R}_{\alpha_i} \in SO(3)$ ，先解决此问题的二次松弛：

$$\min_{\mathbf{R}_{\alpha_i}, \forall \alpha \in \Omega, \forall i} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_R^2 \left\| \mathbf{R}_{\beta_j} - \mathbf{R}_{\alpha_i} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} \right\|_F^2$$

我们可以将上式重写为下式的形式，将矩阵  $\mathbf{R}_{\alpha_i}$  堆叠为向量  $r$

$$\min_r \|A_r r - b_r\|^2$$



## 两阶段分布式轨迹估计（集中式）

### 1. 第一阶段（松弛和映射）

由于是求解最小二乘问题，可以通过求解下式得到解

$$(A_r^T A_r)r = A_r^T b_r$$

$$\min_r \|A_r r - b_r\|^2$$

假设  $\check{r}$  为上式的解，将  $\check{r}$  写为矩阵形式，可以得到  $\check{R}_{\alpha_i}$ 。

由于  $\check{R}_{\alpha_i}$  来源于问题的松弛问题，因此无法保证满足约束  $\check{R}_{\alpha_i} \in SO(3)$ ，因此通过 SVD 分解将结果映射到  $SO(3)$  当中，从而得到旋转的估计  $\hat{R}_{\alpha_i} = \text{project}(\check{R}_{\alpha_i})$ 。



## 两阶段分布式轨迹估计（集中式）

### 2. 第二阶段 (Gauss-Newton 迭代)

用第一阶段的结果  $\hat{\mathbf{R}}_{\alpha_i}$  重新为问题确定参数，将未知的  $\mathbf{R}_{\alpha_i}$  表示为  $\hat{\mathbf{R}}_{\alpha_i}$  乘上一个扰动项

$$\mathbf{R}_{\alpha_i} = \hat{\mathbf{R}}_{\alpha_i} \exp(\theta_{\alpha_i}), \theta_{\alpha_i} \in \mathbb{R}^3$$

其中  $\theta_{\alpha_i}$  是为旋转所加入的新的参数， $\exp(\cdot)$  为  $\text{SO}(3)$  上的指数映射。从而原优化函数化为

$$\min_{\mathbf{t}_{\alpha_i}, \theta_{\alpha_i} \in \mathbb{R}^3} \sum_{\substack{(\alpha_i, \beta_j) \in \mathcal{E} \\ \forall \alpha \in \Omega, \forall i}} \omega_t^2 \left\| \mathbf{t}_{\beta_j} - \mathbf{t}_{\alpha_i} - \hat{\mathbf{R}}_{\alpha_i} \exp(\theta_{\alpha_i}) \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} \right\|^2 + \frac{\omega_R^2}{2} \left\| \hat{\mathbf{R}}_{\beta_j} \exp(\theta_{\beta_j}) - \hat{\mathbf{R}}_{\alpha_i} \exp(\theta_{\alpha_i}) \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} \right\|_F^2$$

为求解上式，先将  $\exp(\cdot)$  一阶展开为  $\exp(\theta_{\alpha_i}) = \mathbf{I}_3 + S(\theta_{\alpha_i})$ ， $S(\theta_{\alpha_i})$  为向量  $\theta_{\alpha_i}$  的反对称矩阵，上式化为

$$\min_{\mathbf{t}_{\alpha_i}, \theta_{\alpha_i} \in \mathbb{R}^3} \sum_{\substack{(\alpha_i, \beta_j) \in \mathcal{E} \\ \forall \alpha \in \Omega, \forall i}} \omega_t^2 \left\| \mathbf{t}_{\beta_j} - \mathbf{t}_{\alpha_i} - \hat{\mathbf{R}}_{\alpha_i} \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} - \hat{\mathbf{R}}_{\alpha_i} S(\theta_{\alpha_i}) \bar{\mathbf{t}}_{\beta_j}^{\alpha_i} \right\|^2 + \frac{\omega_R^2}{2} \left\| \hat{\mathbf{R}}_{\beta_j} - \hat{\mathbf{R}}_{\alpha_i} \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} + \hat{\mathbf{R}}_{\beta_j} S(\theta_{\beta_j}) - \hat{\mathbf{R}}_{\alpha_i} S(\theta_{\alpha_i}) \bar{\mathbf{R}}_{\beta_j}^{\alpha_i} \right\|_F^2$$

通过使用类似第一阶段的求解过程，可以求解出  $\mathbf{t}_{\alpha_i}$  和  $\mathbf{R}_{\alpha_i}$ 。



## 两阶段分布式轨迹估计（集中式）

### 2. 第二阶段 (Gauss-Newton 迭代)

通过使用类似第一阶段的求解过程，将  $\theta_{\alpha_i}$  和  $t_{\alpha_i}$  重排至向量  $p$  中，原问题化为求解线性最小二乘问题：

$$\min_p \|A_p p - b_p\|^2$$

通过求解下式得到  $p$

$$(A_p^T A_p) p = A_p^T b_p$$

从而解出  $R_{\alpha_i}$  和  $t_{\alpha_i}$ 。



## 两阶段分布式轨迹估计（分布式）

### 1. 改写为分布式形式

将  $(A_r^T A_r)r = A_r^T b_r$  中的  $r$  分割  $r = [r_\alpha, r_\beta, \dots]$ ，其中  $r_\alpha$  描述了  $\alpha$  的旋转；同理，将  $(A_p^T A_p)p = A_p^T b_p$  中  $p$  分割  $p = [p_\alpha, p_\beta, \dots]$ ，其中  $p_\alpha$  描述了  $\alpha$  的轨迹。这两个式子被写为一般形式

$$H\mathbf{y} = \mathbf{g} \Leftrightarrow \begin{bmatrix} H_{\alpha\alpha} & H_{\alpha\beta} & \cdots \\ H_{\beta\alpha} & H_{\beta\beta} & \cdots \\ \cdots & \cdots & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{y}_\alpha \\ \mathbf{y}_\beta \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{g}_\alpha \\ \mathbf{g}_\beta \\ \vdots \end{bmatrix}$$

即在给定  $H$  和  $\mathbf{g}$  的前提下，求解  $\mathbf{y}$ 。为引出分布式 Gauss-Siedel ( DGS )，先将上面的线性式写为

$$\sum_{\delta \in \Omega} H_{\alpha\delta} \mathbf{y}_\delta = \mathbf{g}_\alpha, \quad \forall \alpha \in \Omega$$

取出含  $\mathbf{y}_\alpha$  的部分，有

$$H_{\alpha\alpha} \mathbf{y}_\alpha = - \sum_{\delta \in \Omega \setminus \{\alpha\}} H_{\alpha\delta} \mathbf{y}_\delta + \mathbf{g}_\alpha$$



## 两阶段分布式轨迹估计（分布式）

### 1. 改写为分布式形式

DGS 算法开始于一个随机初始值  $\mathbf{y}^{(0)} = [\mathbf{y}_\alpha^{(0)}, \mathbf{y}_\beta^{(0)}, \dots]$ ，在第  $k$  轮迭代，使用如下的更新规则：

$$\mathbf{y}_\alpha^{(k+1)} = H_{\alpha\alpha}^{-1} \left( - \sum_{\delta \in \Omega_\alpha^+} H_{\alpha\delta} \mathbf{y}_\delta^{(k+1)} - \sum_{\delta \in \Omega_\alpha^-} H_{\alpha\delta} \mathbf{y}_\delta^{(k)} + \mathbf{g}_\alpha \right)$$

其中  $\Omega_\alpha^+$  表示已经完成第  $k+1$  次迭代的机器人， $\Omega_\alpha^-$  表示除  $\alpha$  外还需要等待迭代更新的机器人。

有了上面的式子，接下来要考虑的就是数据通信问题和收敛问题。



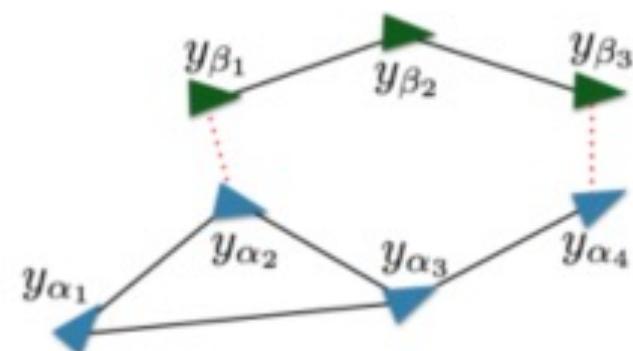
## 两阶段分布式轨迹估计（分布式）

### 2. DGS 中的数据通信

执行 DGS 迭代，机器人  $\alpha$  只需知道 intra-robot 和 inter-robot 的位姿  $\varepsilon_I^\alpha$  和  $\varepsilon_S^\alpha$ 。如下图， $\alpha$  除知道自身的位姿外，仅需知道  $y_{\beta_1}$  和  $y_{\beta_3}$ 。

对于矩阵  $H$ ，在下图右边给出了矩阵的数值分布情况，非对角块元素  $(\alpha_i, \beta_j)$  在  $\alpha_i$  和  $\beta_j$  之间存在边（位姿计算）时取非零值。利用矩阵的稀疏性，可以将 DGS 迭代方程变为

$$\mathbf{y}_\alpha^{(k+1)} = H_{\alpha\alpha}^{-1} \left( - \sum_{(\alpha_i, \delta_j) \in \mathcal{E}_S^{\alpha+}} H_{\alpha_i \delta_j} \mathbf{y}_{\delta_j}^{(k+1)} - \sum_{(\alpha_i, \delta_j) \in \mathcal{E}_S^{\alpha-}} H_{\alpha_i \delta_j} \mathbf{y}_{\delta_j}^{(k)} + \mathbf{g}_\alpha \right)$$



	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\beta_1$	$\beta_2$	$\beta_3$
$\alpha_1$	$H_{\alpha\alpha}$						
$\alpha_2$					$H_{\alpha\beta}$		
$\alpha_3$							
$\alpha_4$							
$\beta_1$					$H_{\beta\alpha}$		
$\beta_2$						$H_{\beta\beta}$	
$\beta_3$							



## 两阶段分布式轨迹估计（分布式）

### 2. DGS 中的数据通信

$$\mathbf{y}_\alpha^{(k+1)} = H_{\alpha\alpha}^{-1} \left( - \sum_{(\alpha_i, \delta_j) \in \mathcal{E}_S^{\alpha+}} H_{\alpha_i \delta_j} \mathbf{y}_{\delta_j}^{(k+1)} - \sum_{(\alpha_i, \delta_j) \in \mathcal{E}_S^{\alpha-}} H_{\alpha_i \delta_j} \mathbf{y}_{\delta_j}^{(k)} + \mathbf{g}_\alpha \right)$$

式中， $\mathcal{E}_S^{\alpha+}$  表示已经完成迭代的 inter-robot 计算， $\mathcal{E}_S^{\alpha-}$  则表示未完成，二者满足  $\mathcal{E}_S^{\alpha+} \cup \mathcal{E}_S^{\alpha-} = \mathcal{E}_S^\alpha$ 。

透过上式可以发现，DGS 只需对 inter-robot 涉及的位姿进行计算，因此机器人  $\delta$  只需要在机器人会合时发送会合点的位姿估计。



## 两阶段分布式轨迹估计（分布式）

### 2. DGS 中的数据通信

$$\mathbf{y}_\alpha^{(k+1)} = H_{\alpha\alpha}^{-1} \left( - \sum_{(\alpha_i, \delta_j) \in \mathcal{E}_S^{\alpha+}} H_{\alpha_i \delta_j} \mathbf{y}_{\delta_j}^{(k+1)} - \sum_{(\alpha_i, \delta_j) \in \mathcal{E}_S^{\alpha-}} H_{\alpha_i \delta_j} \mathbf{y}_{\delta_j}^{(k)} + \mathbf{g}_\alpha \right)$$

式中， $\mathcal{E}_S^{\alpha+}$  表示已经完成迭代的 inter-robot 计算， $\mathcal{E}_S^{\alpha-}$  则表示未完成，二者满足  $\mathcal{E}_S^{\alpha+} \cup \mathcal{E}_S^{\alpha-} = \mathcal{E}_S^\alpha$ 。

透过上式可以发现，DGS 只需对 inter-robot 涉及的位姿进行计算，因此机器人  $\delta$  只需要在机器人会合时发送会合点的位姿估计。



## 两阶段分布式轨迹估计（分布式）

### 3. DGS 的收敛

DGS 可以从任意初始化值  $\mathbf{y}^{(0)} = [\mathbf{y}_\alpha^{(0)}, \mathbf{y}_\beta^{(0)}, \dots]$  开始迭代，因此需要选择一个好的初始值来减少迭代次数。为找到好的初始值，本文使用了标记初始化（Flagged Initialization）的方法：

- 在第一轮迭代开始前，所有机器人标记为“未初始化”。
- $\alpha$  在不考虑 inter-robot 的情况下使用  $\mathbf{y}_\alpha^{(k+1)} = H_{\alpha\alpha}^{-1} \mathbf{g}_\alpha$  迭代，然后  $\alpha$  将自己标记为“已初始化”。
- $\beta$  开始迭代时，仅使用来自已初始化的 inter-robot 位姿，完成迭代后也将自己标为“已初始化”。
- 重复此过程，所有机器人都会在完成迭代后被初始化。



3

# 实验结果

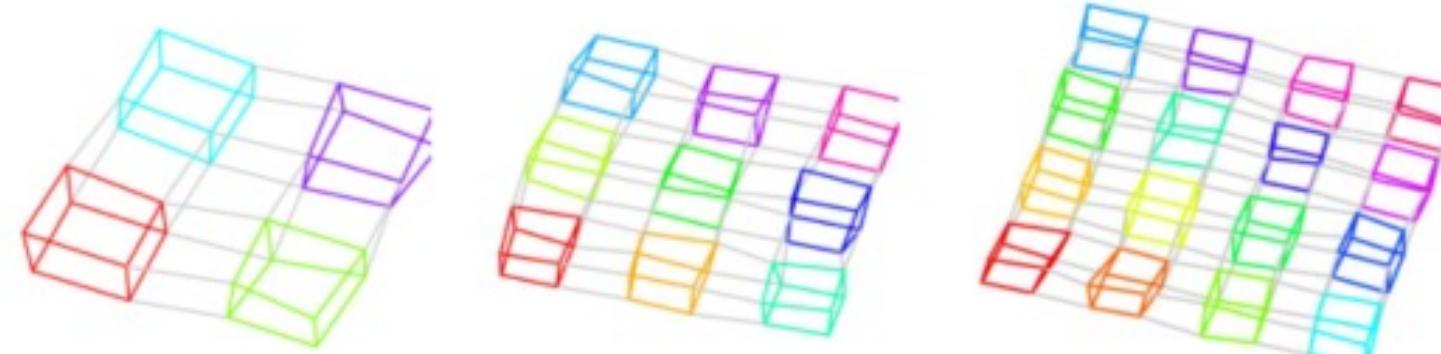


## 实验设置（仿真）

- 分别使用 4, 9, 16, 25, 36 和 49 个机器人进行实验，测量可扩展性。
- 机器人被放置在一个 3D 网格中（如图），机器人将在这个 3D 立方中移动。
- 在机器人到达角落时，它们将发生数据交换（图中灰色连接边）。
- 从均值为 0 的高斯分布中产生噪声，其中旋转的标准差为  $\sigma_R = 5^\circ$ ，平移为  $\sigma_t = 0.2m$ 。（可能是用于生成  $\omega_R^2$  和  $\omega_t^2$ ）
- 运算结果将在 10 次 Monte Carlo 下运行。

设  $m_r = \min_r \|A_r r - b_r\|^2$ ,  $m_p = \min_p \|A_p p - b_p\|^2$ ，利用下面两个式子来定义第 k 轮迭代计算结果的旋转和平移误差

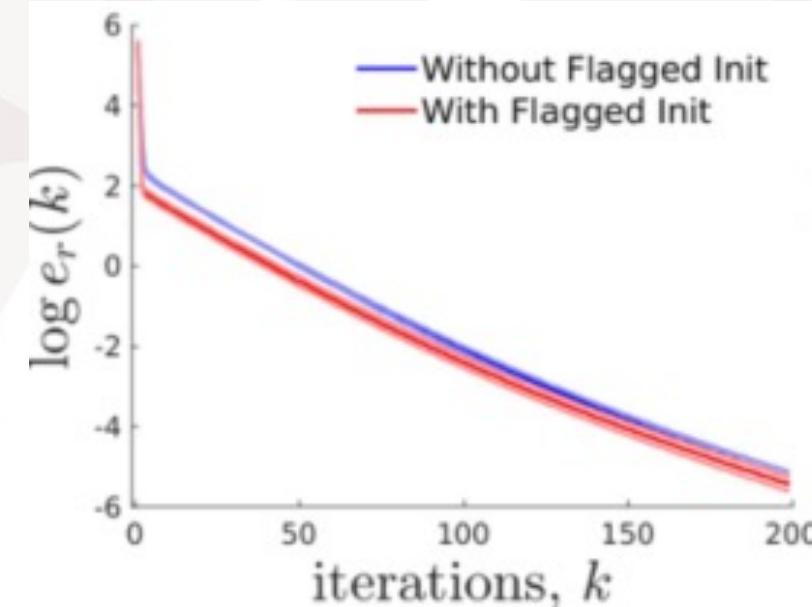
$$e_r(k) = \|A_r r^{(k)} - b_r\|^2 - m_r, \quad e_p(k) = \|A_p p^{(k)} - b_p\|^2 - m_p$$



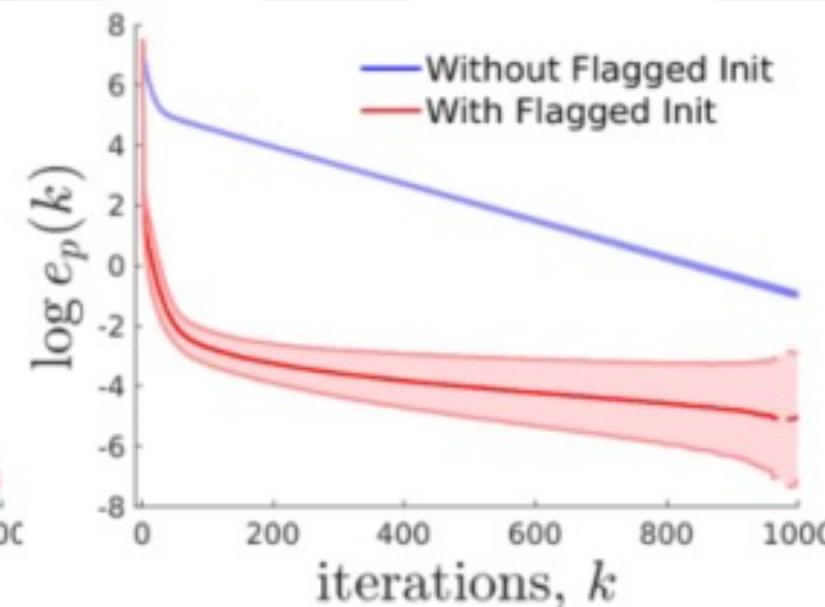


## Flagged Initialization 的结果 ( 仿真 )

如图给出了 49 个机器人的前提下，是否使用 flagged initialization 的迭代收敛情况，在未使用的情况下，初始值均设为 0。发现使用了 flagged initialization 明显加快了收敛速度。



(a) Rotation Estimation Error

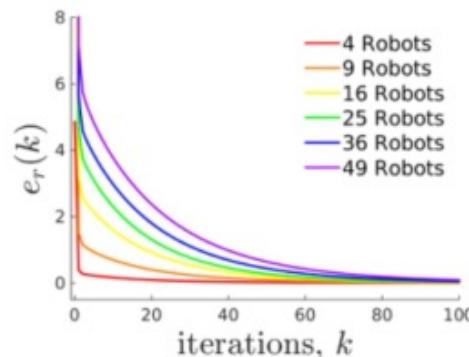


(b) Pose Estimation Error

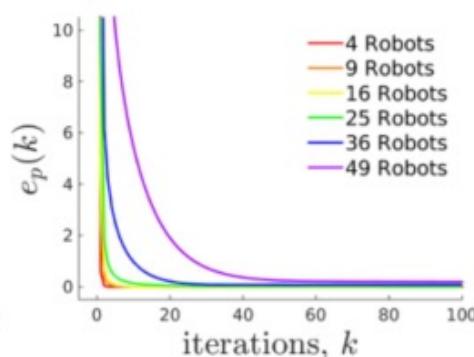


## 收敛性（仿真）

- Fig.6 给出了不同数量机器人环境下的收敛情况（机器人少时收敛快，多时虽然迭代次数增加，但也能快速收敛）
- 下图-2 给出了不同高斯噪声下，49 个机器人的收敛情况（噪声较大时收敛迭代次数变多）
- 下图-3 给出了较少的迭代次数就能将轨迹估计基本还原（Odometry 文中未使用，仅用于视觉效果的对比）

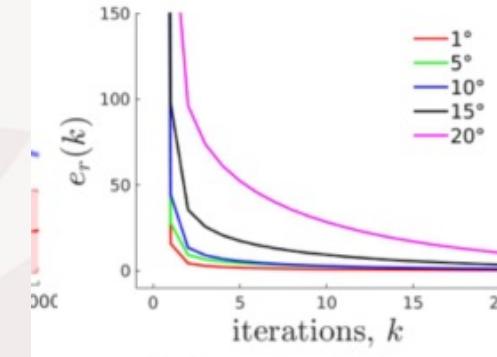


(a) Rotation Estimation Error

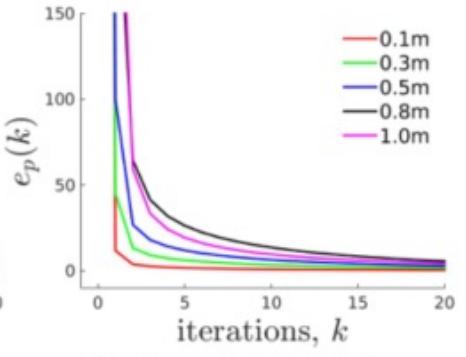


(b) Pose Estimation Error

Fig. 6. Convergence for scenarios with increasing number of robots.

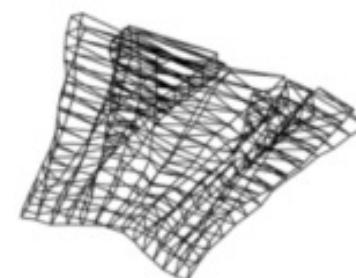


(a) Rotation Noise

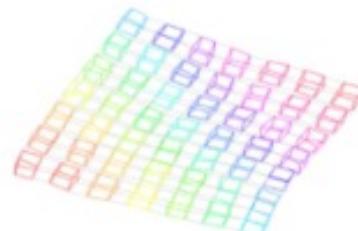


(b) Translation Noise

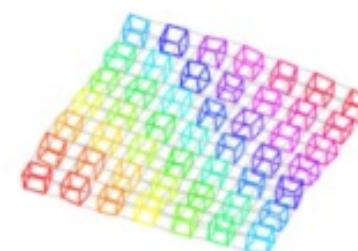
Fig. 7. Convergence for increasing levels of noise (scenario with 49



(a) Odometry



(b) 10 iterations



(c) 1000 iterations

Fig. 8. Trajectory estimates for the scenario with 49 robots. (a) Odo-



## 准确度和可扩展性（仿真）

将本文方法与集中化的两阶段方法和高斯牛顿法对比，对于本文的方法，设置阈值使得满足下条件时停止迭代：

$$\|r^{(k+1)} - r^{(k)}\| \leq \eta_r, \quad \|p^{(k+1)} - p^{(k)}\| \leq \eta_p$$

Table-1 给出了不同阈值下 DGS 和集中化方法的对比，表中 cost 值越低表明效果越好，数据说明 DGS 在多机器人的条件下也能在较少的迭代下达到较好的效果。

#Robots	Distributed Gauss-Seidel				Centralized	
	$\eta_r = \eta_p = 10^{-1}$		$\eta_r = \eta_p = 10^{-2}$		Two-Stage Cost	GN Cost
	#Iter	Cost	#Iter	Cost		
4	10	1.9	65	1.9	1.9	1.9
9	14	5.3	90	5.2	5.2	5.2
16	16	8.9	163	8.8	8.8	8.7
25	17	16.2	147	16.0	16.0	15.9
36	28	22.9	155	22.7	22.6	22.5
49	26	35.1	337	32.9	32.7	32.5

TABLE I



## 对噪声的敏感性（仿真）

Table-2 说明了 DGS 在不同的噪声下也能达到与集中式基本相同的效果：

Measurement noise $\sigma_r(^{\circ})$	$\sigma_t(\text{m})$	Distributed Gauss-Seidel				Centralized	
		$\eta_r = \eta_p = 10^{-1}$		$\eta_r = \eta_p = 10^{-2}$		Two-Stage Cost	GN Cost
		#Iter	Cost	#Iter	Cost		
1	0.05	8.5	2.1	51.0	1.8	1.8	1.8
5	0.1	21.8	14.8	197.8	14.0	14.0	13.9
10	0.2	35.6	58.4	277.7	56.6	56.6	56.0
15	0.3	39.8	130.5	236.8	128.4	129.3	126.0

TABLE II



## separator 的可扩展性（仿真）

separator 用于表示机器人发生数据交换的点。实验中设置两个机器人平行移动 10 个时间步，分别发生数据交换的点从 1 个到 10 个。Fig.9a 说明了在  $\eta_r = \eta_p = 10^{-1}$  下所需的迭代次数。

Fig.9b 说明了 DGS 和 DDF-SAM 在实验环境下数据交换的字节数。

- DDF-SAM 字节数为： $K_{GN} [sB_p + (sB_p)^2]$ ， $K_{GN}$  为计算高斯牛顿法所用的迭代次数， $s$  为 separator 的数量， $B_p$  为位姿的字节数。

- DGS 字节数为： $K_{DJ}^r(sB_r) + K_{DJ}^p(sB_p)$ ， $K_{DJ}^r$  和  $K_{DJ}^p$  为迭代次数， $B_r$  为位姿的旋转部分字节数（9 个 double 共 72B）， $B_p$  为计算 DGS 第二阶段时  $p$  的字节数（6 个 double 共 48B）。

实验说明了 DGS 的数据交换负担较为稳定。

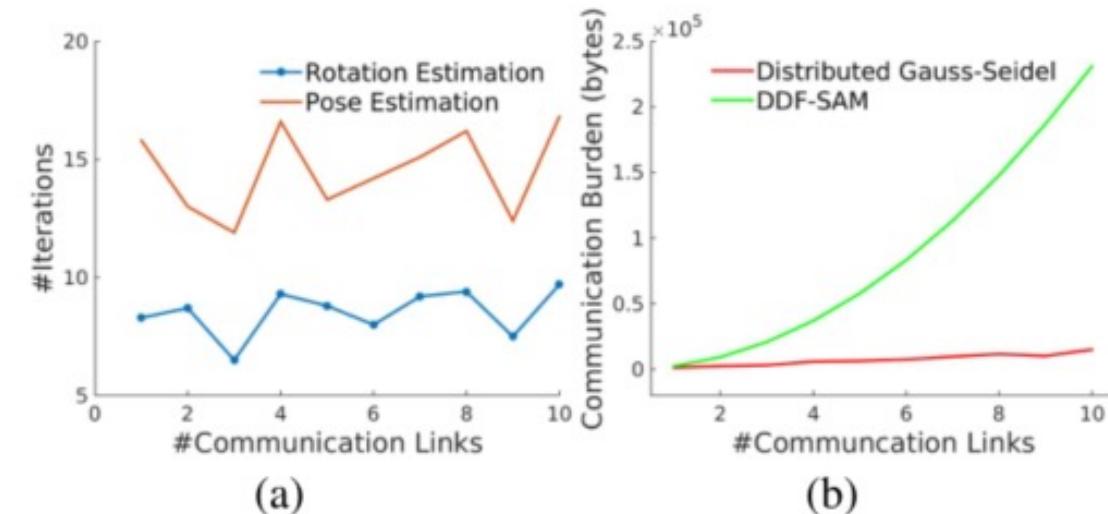


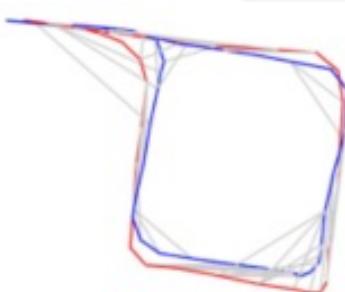
Fig. 9. (a) Number of iterations versus number of separators for the DGS



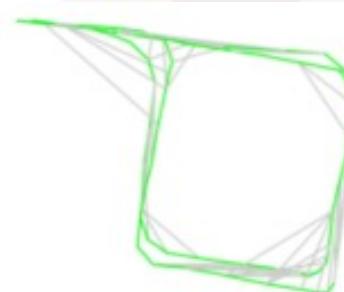
## 真实环境实验

两个机器人在真实的环境下测试：

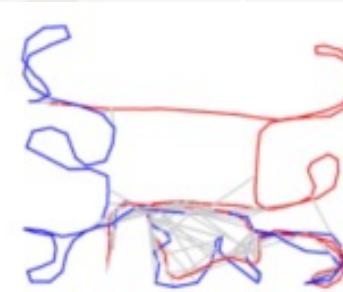
#Test	Distributed Gauss-Seidel				Centralized	
	$\eta_r = \eta_p = 10^{-1}$		$\eta_r = \eta_p = 10^{-2}$		Two-Stage Cost	GN Cost
	#Iter	Cost	#Iter	Cost		
1	10	0.30	78	0.24	0.23	0.23
2	16	0.62	511	0.56	0.54	0.54
3	28	1.20	606	0.87	0.84	0.84



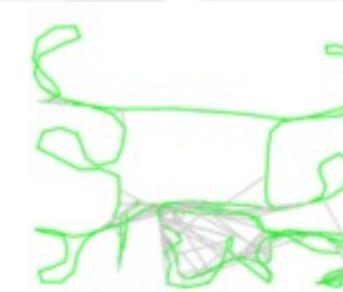
Distributed



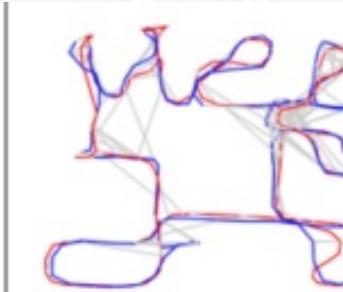
Centralized



Distributed



Centralized



Distributed



Centralized



4

源码



## 源码

- <https://github.com/CogRob/distributed-mapper>





謝謝

Thank You

THANKS

