Broadcast your weaknesses: cooperative active pose-graph SLAM for multiple robots
Supplementary Material

Yongbo Chen*, Liang Zhao*, Ki Myung Brian Lee*,
Chanyeol Yoo*, Shoudong Huang* and Robert Fitch*

December 11, 2019

This document provides supplementary material to the paper [1]. Therefore, it should not be considered as a self-contained document, but instead regarded as an appendix of [1], and cited as:

"Y. Chen, L. Zhao, S. Huang, R. Fitch, K. M. B. Lee and C. Yoo, Broadcast your weaknesses: cooperative active pose-graph SLAM for multiple robots, (Supplementary Material), 2019."

Throughout this document, notation corresponds to that used in [1].

1 Fisher information matrix (FIM) of the 3D pose-graph SLAM corresponding to (1) in [1]

The FIM of the 3D pose-graph SLAM problem is given by [2]:

$$\boldsymbol{\mathcal{I}}_{3D} = \left[\begin{array}{c:c} \boldsymbol{L}_w^{\mathbb{R}^3} & \triangle_w^{3D\top} \\ \hdashline \triangle_w^{3D} & \boldsymbol{L}_w^{SO(3)} + \mathrm{diag}\{\boldsymbol{\Psi}_1, \cdots, \boldsymbol{\Psi}_{n_p}\} \end{array}\right]. \tag{1}$$

where

$$\boldsymbol{L}_{w\ ii_1}^{\mathbb{R}^3} = \begin{cases} \sum_{j_1 \in V_i^+} \delta_{ij_1}^{-2} \boldsymbol{I}_{3\times 3} + \sum_{j_2 \in V_i^-} \delta_{j_2 i}^{-2} \boldsymbol{I}_{3\times 3} & i = i_1 \\ -\delta_{ii_1}^{-2} \boldsymbol{I}_{3\times 3} & (i, i_1) \in \mathcal{E} \\ -\delta_{i_1 i}^{-2} \boldsymbol{I}_{3\times 3} & (i_1, i) \in \mathcal{E} \\ \boldsymbol{0}_{3\times 3} & \text{else,} \end{cases} \tag{2}$$

$$\triangle_{w\ ii_1}^{3D} = \begin{cases} \begin{bmatrix} \sum_{j \in V_i^+} \delta_{ij}^{-2} (\boldsymbol{x}_j - \boldsymbol{x}_i)^\top \boldsymbol{R}_i \boldsymbol{E}_1 \boldsymbol{R}_i^\top \\ \sum_{j \in V_i^+} \delta_{ij}^{-2} (\boldsymbol{x}_j - \boldsymbol{x}_i)^\top \boldsymbol{R}_i \boldsymbol{E}_2 \boldsymbol{R}_i^\top \\ \sum_{j \in V_i^+} \delta_{ij}^{-2} (\boldsymbol{x}_j - \boldsymbol{x}_i)^\top \boldsymbol{R}_i \boldsymbol{E}_3 \boldsymbol{R}_i^\top \end{bmatrix} & i = i_1 \\ \begin{bmatrix} \delta_{ii_1}^{-2} (\boldsymbol{x}_{i_1} - \boldsymbol{x}_i)^\top \boldsymbol{R}_i \boldsymbol{E}_1 \boldsymbol{R}_i^\top \\ \delta_{ii_1}^{-2} (\boldsymbol{x}_{i_1} - \boldsymbol{x}_i)^\top \boldsymbol{R}_i \boldsymbol{E}_2 \boldsymbol{R}_i^\top \\ \delta_{ii_1}^{-2} (\boldsymbol{x}_{i_1} - \boldsymbol{x}_i)^\top \boldsymbol{R}_i \boldsymbol{E}_3 \boldsymbol{R}_i^\top \end{bmatrix} & (i, i_1) \in \mathcal{E} \\ \boldsymbol{0}_{3\times 3} & \text{else,} \end{cases}. \tag{3}$$

*All authors are with Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, 2007 Australia, e-mail: Yongbo.Chen@student.uts.edu.au.

$$
\boldsymbol{L}_w^{SO(3)}{}_{ii_1} = \begin{cases} \sum_{j\in V_i} \frac{\omega_{ij}}{3}\boldsymbol{I}_{3\times 3} & i = i_1 \\ -\frac{\omega_{ii_1}}{3}\boldsymbol{I}_{3\times 3} & (i,i_1)\in\mathcal{E} \\ -\frac{\omega_{i_1 i}}{3}\boldsymbol{I}_{3\times 3} & (i_1,i)\in\mathcal{E} \\ \boldsymbol{0}_{3\times 3} & \text{else,} \end{cases}
$$

$$
\boldsymbol{\Psi}_i = \begin{bmatrix} \psi_i^{11} & \psi_i^{12} & \psi_i^{13} \\ \psi_i^{12} & \psi_i^{22} & \psi_i^{21} \\ \psi_i^{13} & \psi_i^{21} & \psi_i^{33} \end{bmatrix}, \tag{4}
$$

$$
\psi_i^{kl} = \sum_{j\in V_i^+} \delta_{ij}^{-2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{R}_i \boldsymbol{I}_{3\times 3}^{k,l} \boldsymbol{R}_i^\top (\boldsymbol{x}_i - \boldsymbol{x}_j),
$$

$$
\boldsymbol{E}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \boldsymbol{E}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \boldsymbol{E}_3 = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{5}
$$

where $\boldsymbol{I}_{3\times 3}^{k,l} = \boldsymbol{E}_k \boldsymbol{E}_l^\top$, $\star_{ij}$ means the (i,j)-th block of the matrix $\star$.

2 Linear-algebraic operations for fast covariance recovery

Using a recursive formula [3], one can efficiently recover the covariance of the last pose, which is a block-diagonal sub-matrix of the entire covariance matrix. Let $\boldsymbol{H}$ be the square root information matrix such that $\boldsymbol{F} = \boldsymbol{\mathcal{I}}_{nD} = \boldsymbol{H}^\top \boldsymbol{H}$. The recursion is given by:

$$
\begin{aligned}
\boldsymbol{C}_{ii} =& \frac{1}{\boldsymbol{H}_{ii}}\Big(\frac{1}{\boldsymbol{H}_{ii}} - \sum_{k=i+1,\boldsymbol{H}_{ik}\neq 0}^{n} \boldsymbol{H}_{ik}\boldsymbol{C}_{ki}\Big), \\
\boldsymbol{C}_{ij} =& \frac{-1}{\boldsymbol{H}_{ii}}\Big(\sum_{k=i+1,\boldsymbol{H}_{ik}\neq 0}^{j} \boldsymbol{H}_{ik}\boldsymbol{C}_{kj} + \sum_{k=j+1,\boldsymbol{H}_{ik}\neq 0}^{n} \boldsymbol{H}_{ik}\boldsymbol{C}_{jk}\Big),
\end{aligned} \tag{6}
$$

Similar to the fast recovery method in SLAM++ [4], we use the approximate minimum degree permutation (AMDP) and a hash table for sparse Cholesky decomposition. The entire process is shown in Algorithm 1.

Algorithm 1: Fast covariance recovery for last pose

Input: Fisher information matrix $\boldsymbol{F}$
Output: The covariance block corresponding to the last pose
Create $\boldsymbol{F}' = \boldsymbol{F}$, delete its column and row corresponding to the last pose, and get $\boldsymbol{F}'_{reduce}$;
$\boldsymbol{p}_{order} = AMDP(\boldsymbol{F}'_{reduce})$;
$\boldsymbol{p}_{order} = [\boldsymbol{p}_{order}; \boldsymbol{p}_{order}^{last}]$; // $\boldsymbol{p}_{order}^{last}$ is the column/row order for the last pose.
$\boldsymbol{H} = Cholesky(\boldsymbol{F}(\boldsymbol{p}_{order}, \boldsymbol{p}_{order}))$; // Sparse Cholesky decomposition.
Covariance recovery by (6) using a hash table;

3 Further discussion on Problem 1

In this section, we focus on Problem 1 in [1]. We first consider the case where the added edge has weight value 1, and then discuss the general case.

Problem 1 asks the following question. For a pose graph $\mathcal{G}^0$, if we add an edge (i.e. a new measurement) as in the case of 1-ESP, can we find equivalent addition of a node with 2 edges, as in the case of 1-NESP, in terms of information? Or, can we estimate their differences in terms of information? Fig. 1 illustrates the situation, where the base pose graphs are the same.

3.1 Solution

3.1.1 Case where weight value is equal to 1

Eq.(8) in [1] tells us that the D-optimality metric of 1-ESP is approximately equal to the tree-connectivity of the whole graph. Thus, we consider tree-connectivity instead of the original metric.

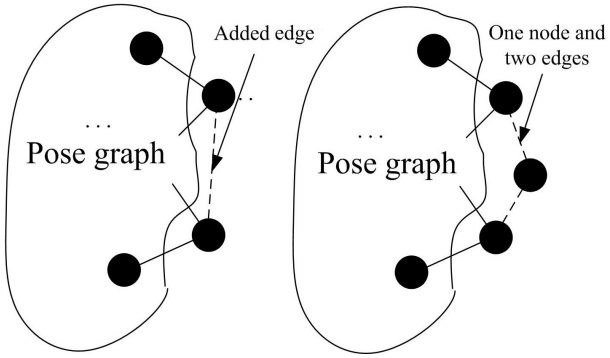

Figure 1: 1-ESP and 1-NESP problems

First, let $\mathcal{G}^0_{\mathbb{R}^n}$ and $\mathcal{G}^0_{SO(n)}$ be the rotation and translation graphs of the base pose graph $\mathcal{G}^0$. Then, the tree connectivity is $t_w(\mathcal{G}^0) = n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^0_{SO(n)}))$. We then add a new edge e to $\mathcal{G}^0$ to obtain a new pose graph $\mathcal{G}^e$. Setting the weight of the new edge to be $w(e) = 1$, the tree-connectivity of $\mathcal{G}^e$ can be computed as follows.

Adding e can result in two possible situations for spanning trees $\mathcal{T} \in \mathcal{T}_{\mathcal{G}^e}$, as shown in Fig. 2. In the first situation, $\mathcal{T}$ does not include e (i.e. $e \notin \mathcal{E}(\mathcal{T})$), and the value of the spanning tree $\mathbb{V}(\mathcal{T}$ remains unchanged. In the second situation, $e \in \mathcal{T}$, and $\mathbb{V}(\mathcal{T})$ changes. It is set as $t_w(\mathcal{G}^1))$. Thus the tree connectivity of the new graph is:

$$\begin{aligned}&\log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^e))) \approx t_w(\mathcal{G}^e) = n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))\\ &= n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + t_w(\mathcal{G}^1_{SO(n)}))\end{aligned} \tag{7}$$

For 1-NESP, the tree-connectivity of the new pose graph falls into three cases, which are shown in Fig. 3. Because of the added node, at least one measurement between edges 1 and 2 is required. The first two cases mean that the spanning tree only includes one of these two edges. The weight of the spanning tree in these two cases remains equal to that of the original graph $\mathcal{G}^0$.

In the third case, the spanning tree includes both new edges. We can view these two edges and the associated new node as a single edge, which implies that the weight of the spanning tree is equal to that of the second case of 1-ESP. Thus the weight of the spanning tree of pose graph $\mathcal{G}^n$ after adding one node and two edges is:

$$\begin{aligned}&\log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^n))) \approx n\log(t_w(\mathcal{G}^n_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^n_{SO(n)}))\\ &= n\log(\underbrace{2t_w(\mathcal{G}^0_{\mathbb{R}^n})}_{\text{add edge 1 or edge 2}} + \underbrace{t_w(\mathcal{G}^1_{\mathbb{R}^n})}_{\text{add edge 1 and 2}})\\ &+ d\log(\underbrace{2t_w(\mathcal{G}^0_{SO(n)})}_{\text{add edge 1 or edge 2}} + \underbrace{t_w(\mathcal{G}^1_{SO(n)})}_{\text{add edge 1 and 2}})\end{aligned} \tag{8}$$

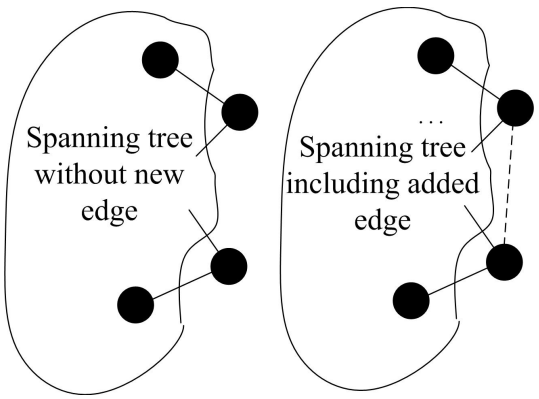

Figure 2: Two kinds of spanning trees in 1-ESP

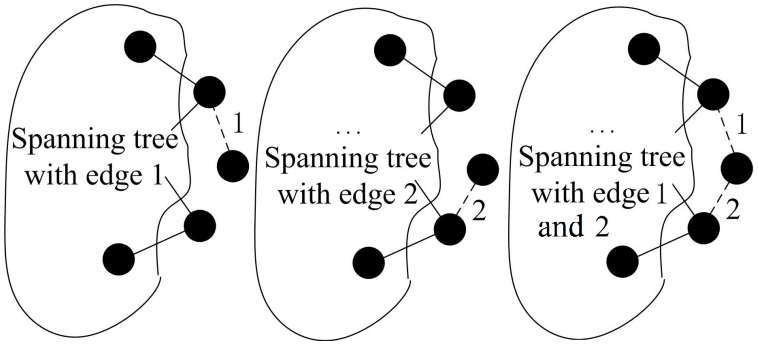

Figure 3: Two kinds of spanning trees in 1-NESP

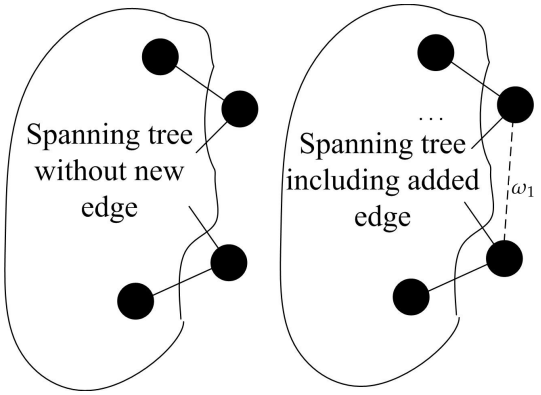

Figure 4: Two kinds of spanning trees with non-unit weight edge in 1-ESP

Because $t_w(\mathcal{G}^0_{\mathbb{R}^n}) > 0$, $t_w(\mathcal{G}^0_{SO(n)}) > 0$, $t_w(\mathcal{G}^1_{\mathbb{R}^n}) > 0$ and $t_w(\mathcal{G}^1_{SO(n)}) > 0$, from (13) we have:

$$\begin{aligned}
&n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)})) \\
&= n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + t_w(\mathcal{G}^1_{SO(n)})) \\
&< n\log(t_w(\mathcal{G}^n_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^n_{SO(n)})) \\
&= n\log(2t_w(\mathcal{G}^0_{\mathbb{R}^n}) + t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(2t_w(\mathcal{G}^0_{SO(n)}) + t_w(\mathcal{G}^1_{SO(n)})) \\
&< n\log(2t_w(\mathcal{G}^0_{\mathbb{R}^n}) + 2t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(2t_w(\mathcal{G}^0_{SO(n)}) + 2t_w(\mathcal{G}^1_{SO(n)})) \\
&= n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n}) + d\log(t_w(\mathcal{G}^e_{SO(n)})) + n\log 2 + d\log 2
\end{aligned} \tag{9}$$

Lastly, we obtain lower bound LB and upper bound UB on the tree-connectivity of the pose graph $\mathcal{G}^n$ as follows:

$$\begin{aligned}
LB &= n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)})) \\
UB &= n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)})) + (n+d)\log 2
\end{aligned} \tag{10}$$

3.1.2 Non-unit edge weight

In this section, we consider the more general case where new edges have non-unit weights. The weight of a single new edge is defined as ω_1, where $\omega_1 > 1$. For a pair of new edges, their weights are both equal and defined as ω_2, where $2\omega_1 \geq \omega_2 \geq \omega_1 > 1$. We address the same questions posed in the previous section for this case.

Based on Definition 3.1 in [1] and (7), the tree connectivity of the graph $\mathcal{G}_0$ after adding a non-unit weight edge is:

$$\begin{aligned}
&\log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^e))) \approx n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)})) \\
&= n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \omega_1 t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + \omega_1 t_w(\mathcal{G}^1_{SO(n)}))
\end{aligned} \tag{11}$$

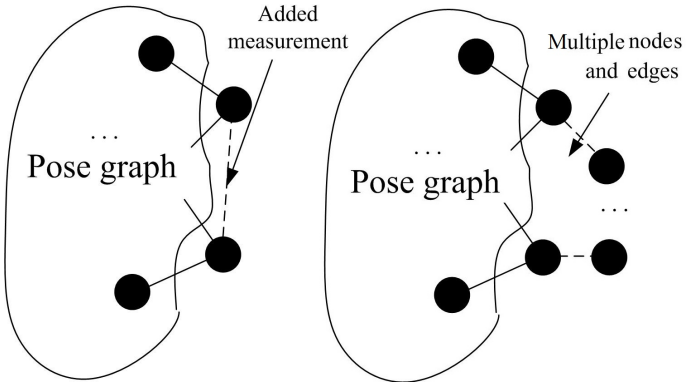

Figure 5: 1-ESP and multi-NESP

The tree-connectivity of the corresponding graph after adding one node and two edges is:

$$
\begin{aligned}
&\log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^e))) \approx n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))\\
&= n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \omega_1 t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + \omega_1 t_w(\mathcal{G}^1_{SO(n)}))\\
&\leq \log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^n))) \approx n\log(t_w(\mathcal{G}^n_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^n_{SO(n)}))\\
&= n\log(2\omega_2 t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \omega_2^2 t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(2\omega_2 t_w(\mathcal{G}^0_{SO(n)}) + \omega_2^2 t_w(\mathcal{G}^1_{SO(n)}))\\
&= n\log(\omega_2) + n\log(2) + n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \frac{\omega_2}{2} t_w(\mathcal{G}^1_{\mathbb{R}^n}))\\
&+ d\log(\omega_2) + d\log(2) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + \frac{\omega_2}{2} t_w(\mathcal{G}^1_{SO(n)}))\\
&\leq n\log(\omega_2) + n\log(2) + n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \omega_1 t_w(\mathcal{G}^1_{\mathbb{R}^n}))\\
&+ d\log(\omega_2) + d\log(2) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + \omega_1 t_w(\mathcal{G}^1_{SO(n)}))\\
&= n\log(\omega_2) + n\log(2) + d\log(\omega_2) + d\log(2) + n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))\\
&\approx n\log(\omega_2) + n\log(2) + d\log(\omega_2) + d\log(2) + \log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^e)))
\end{aligned}
\tag{12}
$$

The upper and lower bounds become:

$$
\begin{aligned}
&LB_1 \leq n\log(t_w(\mathcal{G}^n_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^n_{SO(n)})) \leq UB_1\\
&LB_1 = n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))\\
&UB_1 = (n+d)\log(\omega_2) + (n+d)\log(2) + n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))
\end{aligned}
\tag{13}
$$

4 Problem 2: Further discussion

Problem 2 involves the addition of more than one node. Can we find an equivalent operation using m nodes with at least $m+1$ edges, named multi-NESP, to replace 1-ESP with respect to information? Or can we estimate their differences with respect to information? Two different pose-graph structures are shown in Fig. 5. Their pose graphs are the same.

4.1 Solution methods

4.1.1 1 chain structure with m nodes and $m+1$ edges

Assume the new $m+1$ edges have equal weights defined as ω_3, satisfying $C^1_{m+1}\omega_1 \geq \omega_3 \geq \omega_1 \geq 1$, where C^1_{m+1} implies that the permutation selected 1 element from $m+1$ elements. Further assume that the new nodes and edges form a chain structure where the beginning and ending nodes are connected to the corresponding nodes of the 1-ESP problem.

Similar to (8), we can compute the weight of the spanning tree of the pose graph after adding

m nodes and $m+1$ edges $\mathcal{G}^m$ as:

$$\begin{aligned}
&\log(\det(\boldsymbol{\mathcal{I}_{nD}}(\boldsymbol{\mathcal{G}^m}))) \approx n\log(t_w(\mathcal{G}^m_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^m_{SO(n)}))\\
&= n\log(\underbrace{C^1_{m+1}\omega_3^m t_w(\mathcal{G}^0_{\mathbb{R}^n})}_{\text{add a open edge line}} + \underbrace{\omega_3^{m+1} t_w(\mathcal{G}^1_{\mathbb{R}^n})}_{\text{add all edges}})\\
&+ d\log(\underbrace{C^1_{m+1}\omega_3^m t_w(\mathcal{G}^0_{SO(n)})}_{\text{add a open edge line}} + \underbrace{\omega_3^{m+1} t_w(\mathcal{G}^1_{SO(n)})}_{\text{add all edges}})
\end{aligned} \tag{14}$$

The tree-connectivity of the corresponding graph after adding one node and two edges is:

$$\begin{aligned}
&\log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^e))) \approx n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))\\
&= n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \omega_1 t_w(\mathcal{G}^1_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + \omega_1 t_w(\mathcal{G}^1_{SO(n)}))\\
&\le \log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^m))) \approx n\log(t_w(\mathcal{G}^m_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^m_{SO(n)}))\\
&= n\log(C^1_{m+1}\omega_3^m t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \omega_3^{n+1} t_w(\mathcal{G}^1_{\mathbb{R}^n}))\\
&+ d\log(C^1_{m+1}\omega_3^m t_w(\mathcal{G}^0_{SO(n)}) + \omega_3^{m+1} t_w(\mathcal{G}^1_{SO(n)}))\\
&= n\log(\omega_3^m) + n\log(C^1_{m+1}) + n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \frac{\omega_3}{C^1_{m+1}} t_w(\mathcal{G}^e_{SO(n)}))\\
&+ d\log(\omega_3^m) + d\log(C^1_{m+1}) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + \frac{\omega_3}{C^1_{m+1}} t_w(\mathcal{G}^1_{SO(n)}))\\
&\le n\log(\omega_3^m) + n\log(C^1_{m+1}) + n\log(t_w(\mathcal{G}^0_{\mathbb{R}^n}) + \omega_1 t_w(\mathcal{G}^e_{SO(n)}))\\
&+ d\log(\omega_3^m) + d\log(C^1_{m+1}) + d\log(t_w(\mathcal{G}^0_{SO(n)}) + \omega_1 t_w(\mathcal{G}^1_{SO(n)}))\\
&= n\log(\omega_3^m) + n\log(C^1_{m+1}) + n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))\\
&+ d\log(\omega_3^m) + d\log(C^1_{m+1})\\
&\approx (nm+dm)\log(\omega_3) + (n+d)\log(C^1_{m+1}) + \log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^e)))
\end{aligned} \tag{15}$$

4.1.2 Complex structure with more than $m+1$ edges

The chain structure shown in Section 4.1.1 is just one type of complex structure with more than $m+1$ edges. From Theorem 4 in [5], we know that the tree-connectivity gain for a connected graph is normalized, monotone, and sub-modular. The tree-connectivity of the case in Section 4.1.1 is thus a lower-bound on the general case. Assume that the pose-graphs corresponding to the general case are $\mathcal{G}^{m*}$, $\mathcal{G}^{m*}_{\mathbb{R}^n}$ and $\mathcal{G}^{m*}_{SO(n)}$. We have:

$$\begin{aligned}
&\log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^e))) \approx n\log(t_w(\mathcal{G}^e_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^e_{SO(n)}))\\
&\le \log(\det(\boldsymbol{\mathcal{I}_{nD}}(\mathcal{G}^{m*}))) \approx n\log(t_w(\mathcal{G}^{m*}_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^{m*}_{SO(n)})),
\end{aligned} \tag{16}$$

which shows that the solution of the 1-ESP problem can give a performance guarantee for the solution of the multi-NESP problem.

5 Performance bounds

We first consider the simplest method in Section 3.1.1. For most complex pose graphs, because the weights of spanning trees that include both new edges is smaller than the other case $t_w(\mathcal{G}^1_{\mathbb{R}^n}) << t_w(\mathcal{G}^0_{\mathbb{R}^n}), t_w(\mathcal{G}^1_{SO(n)}) << t_w(\mathcal{G}^0_{SO(n)})$, we have:

$$\begin{aligned}
&n\log(t_w(\mathcal{G}^n_{\mathbb{R}^n})) + d\log(t_w(\mathcal{G}^n_{SO(n)})) - LB >>\\
&UB - n\log(t_w(\mathcal{G}^n_{\mathbb{R}^n})) - d\log(t_w(\mathcal{G}^n_{SO(n)})) > 0
\end{aligned} \tag{17}$$

We observe that, compared with the lower bound, the tree-connectivity $\log(t_w(\mathcal{G}^n_{\mathbb{R}^n}) + t_w(\mathcal{G}^n_{SO(n)}))$ usually approaches UB. In order to verify this result, we performed several simulations where we added two randomly selected nodes using well-known datasets including city10000, intel, CSAIL and manhattan. These two nodes are used to add one edge or a node with two edges. Based on (17), the ratio $Ratio = \frac{\log(t_w(\mathcal{G}^n_{SO(n)})) + \log 2 - \log(t_w(\mathcal{G}^e_{SO(n)}))}{\log(t_w(\mathcal{G}^n_{SO(n)})) - \log(t_w(t_w(\mathcal{G}^n_{SO(n)})))}$ is shown in Figs. 6 through 9. The results show that, as an example, the rotation graphs satisfy our assumptions about $t_w(\mathcal{G}^1_{SO(n)}) <<$

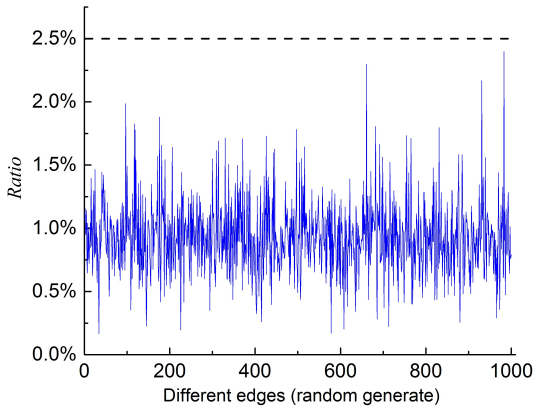

Figure 6: city10000

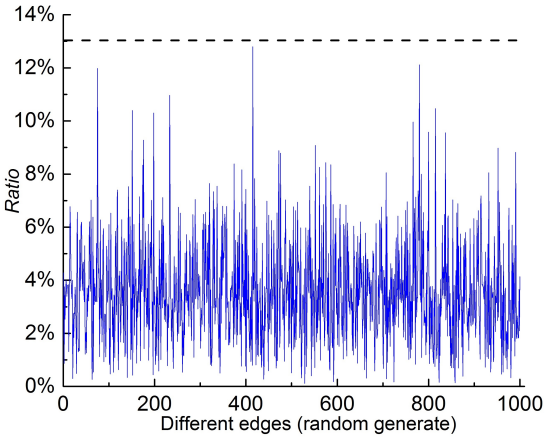

Figure 7: intel

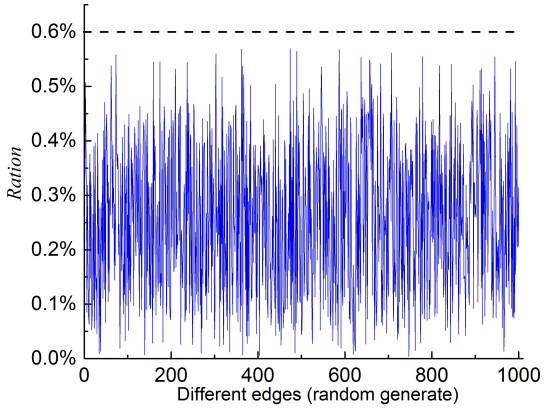

Figure 8: CSAIL

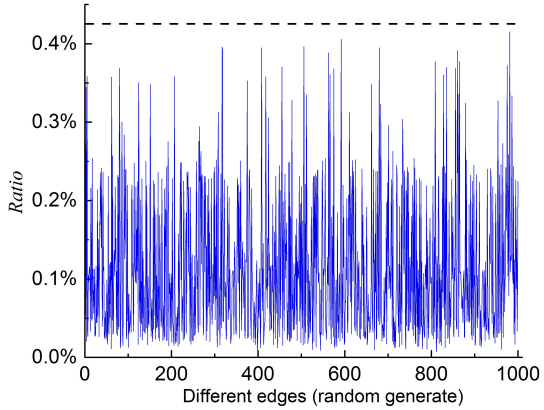

Figure 9: manhattan

$t_w(\mathcal{G}^0_{SO(n)})$, which helps to support our conclusion in (17). Even though these bounds may not be tight, they approach their upper bounds for most pose-graphs. For most datasets, we have: $n\log(t_w(\mathcal{G}^n_{\mathbb{R}^n}))+d\log(t_w(\mathcal{G}^n_{SO(n)})) \in (0.98UB,\ UB)$. This conclusion is also suitable for the normal weighted value in Section 3.1.2 and the multi-NESP in Section 4.1.1.

6 Setting the $Threshold_2$ parameter

In the 1-ESP problem, there are three situations for setting $Threshold_2$, as shown in Fig. 10. Every two nodes in the pose graph can be connected to improve the SLAM result. Our task is thus to define the possible set $\mathcal{E}_c$. We consider the 1-ESP problem whose two nodes are two blue nodes shown in Fig. 10.

For $Ls < Rs$, normally the common features between two blue nodes are enough to compute the relative poses. There will be an edge between them. Because of the random placement of features, however, it is possible to find that there is no edge between them. In this situation, our active SLAM target will be the midpoint of the straight line. When the other robot visits this point, it will be possible to enlarge the sensed area and introduce one additional node and two edges to the new pose graph.

For $R_s < Ls < 2R_s$, normally there are few common features between the two blue nodes. These are typically not sufficient to compute the relative poses. Our potential target located at the green point can help to introduce more common features. In this way, we hope that the relative measurements can be built between our additional pose and two original poses.

Candidate edges can be too long, especially longer than twice the sensor range $2R_s$, for one additional pose to be enough to strengthen the pose graph. Multiple poses need to be introduced on the interval $v_t\triangle t$. When too many poses are added, we need a longer additional active SLAM trajectory and, meanwhile, the graph will be generally become more complicated and less like a chain structure. There is also a greater chance of generating a multi-NESP problem instead of an

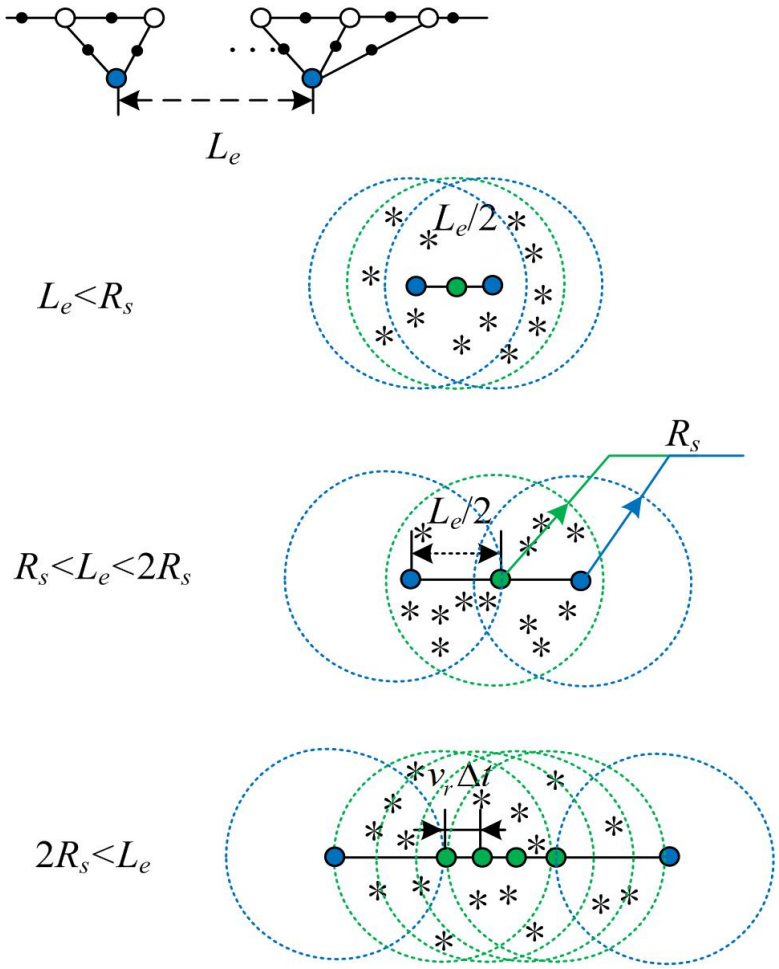

Figure 10: Three possible situations based on $Threshold_2$

m-NESP problem. This graph is hard to be analyzed. We can only offer a loose lower bound for it in Theorem 3.5 in our paper. Based on our analysis in the above Section 5, the tree-connectivity of the graph after adding both nodes and edges approaches its upper bound. Thus if too many poses are added, we can not ensure the bound performance. In short, because of the limitation of the additional active SLAM trajectory and the complication of the multiple nodes situation, we prefer a suitable index $Threshold_2 = 2R_s + 3v_r \triangle t$.

7 SVM-based corridor generation in the horizontal direction

For the horizontal direction, the hard-margin SVM formulation is applied. We can obtain many features of the obstacles based on the map representations. Their sets are divided based on their associated obstacle. Based on the SVM, we can divide the waypoints with features on obstacles by one margin. Let the features belonging to the j-th obstacle be $\{\boldsymbol{x}_l^{o,j}\} = \{x_{x,l}^{o,j}, x_{y,l}^{o,j}, x_{z,l}^{o,j}\}, l = 1, \cdots, N_j^o$, where N_j^o is the number of the features of the j-th obstacle. We have:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}_k,\ \boldsymbol{\beta}_k^o,\ \boldsymbol{\beta}_k^w} \quad \boldsymbol{\alpha}_k^\top \boldsymbol{E} \boldsymbol{\alpha}_k \\ &s.t. \quad \boldsymbol{A}_k^o \boldsymbol{\alpha}_k - \boldsymbol{\beta}_k^o \leq -\mathbf{1},\ \boldsymbol{A}_k^w \boldsymbol{\alpha}_k - \boldsymbol{\beta}_k^w \geq -\mathbf{1}, \end{aligned} \tag{18}$$

Here, $\boldsymbol{\alpha}_k = (\cdots, a_j^k, b_j^k, \cdots)^\top$ is the set of coefficients of the 2D margin equations between the k-th waypoint and the j-th obstacle that is close to the waypoint. We only consider N_k^{obs} obstacles near the k-th waypoint as the environment may be large.

Vectors $\boldsymbol{\beta}_k^o$ and $\boldsymbol{\beta}_k^w$ have dimensions $\sum_{j=1}^{N_k^{obs}} N_j^o$ and N_k^{obs}, respectively. Block matrices $\boldsymbol{A}_k^o$ and $\boldsymbol{A}_k^w$ such that the l-th row of every j-th diagonal is the x- and y-coordinates of the features on the j-th obstacle $(x_{x,l}^{o,j}, x_{y,l}^{o,j})$, and the x and y-coordinates of the k-th way-point $(x_{x,k}^{r,i}, x_{y,k}^{r,i})$ respectively.

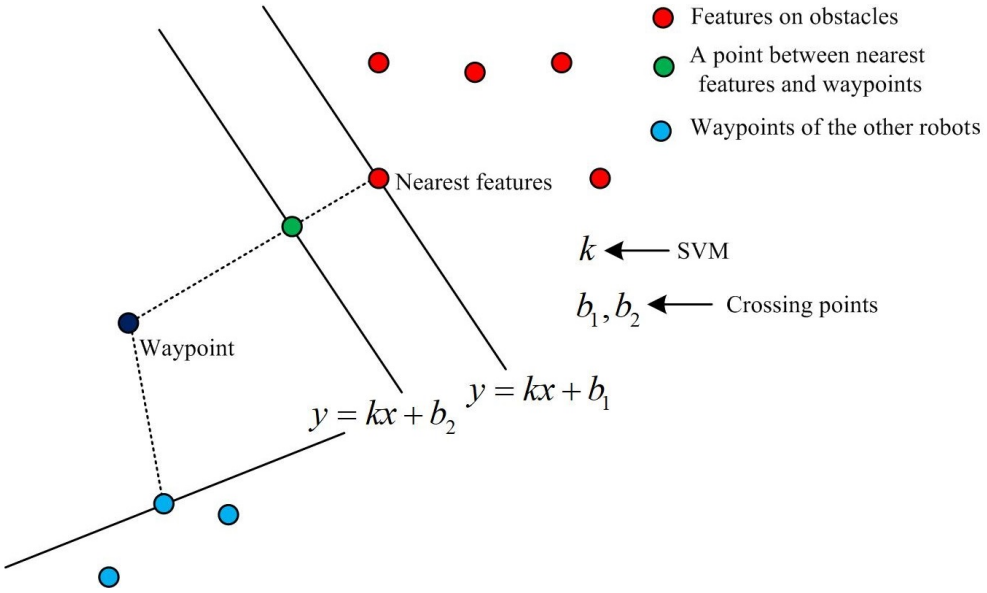

Figure 11: SVM-based corridor generation on the horizontal direction

The SVM result shows the maximum-margin hyperplane. The slope of the hyperplane of the k-th way-point corresponding to the j-th obstacle is obtained by $-b_j^k/a_j^k$. In this paper, the y-intercept is obtained by crossing the nearest feature point $\{\boldsymbol{x}_{near}^{o,j}\}$ on the j-th obstacle. It can be set in other points between $\{\boldsymbol{x}_{near}^{o,j}\}$ and $\{\boldsymbol{x}_k^{r,i}\}$ to give a safer corridor for the robot. We note that, if the trajectory of the other robot comes close to the waypoints, they also need to be seen as obstacles and formulated in a similar form in (18). By solving (18) for all waypoints, we can obtain safe corridors in the horizontal direction. The entire process is shown in Fig. 11.

8 Bezier curve-based trajectory planning

In the previous section, a safe corridor is built. It remains to plan a smooth trajectory for the robot. We use a Bezier-curve parameterization and a quadratic program to find such a trajectory.

Piecewise polynomials, like Bezier curves, are widely used to represent complex trajectories with an arbitrary number of continuous derivatives for trajectory planning by selecting a suitable choice of curve degree and number of pieces. In every dimension (x, y and z), the robot trajectory is represented as an N-th order Bezier curve whose variable is time t:

$$f_\mu(t) = \sum_{j=0}^{N} t^j c_j^\mu,\ c_j^\mu = \frac{n!}{(n-j)!} \sum_{i=0}^{j} \frac{(-1)^{i+j} p_i^\mu}{i!(j-i)!} \tag{19}$$

where p_i^u is the $\mu = (x,\ y,\ z)$ axis of control point $\boldsymbol{p}_i$. The trajectory planning problem is to minimize snap:

$$J = \int_{t_0}^{t_N} \sum_{\mu=x,y,z} \left\| \frac{d^k f_\mu(t)}{dt^k} \right\|_2^2 dt, \tag{20}$$

In our simulation, the coefficient k is set as 4. Rewriting (20) in matrix form, we obtain the final trajectory planning problem in the safe corridor with the ending constraints, including start and goal position constraints, using the convex quadratic program:

$$\begin{aligned} \max_{\boldsymbol{p}} \quad & \boldsymbol{p}^\top (\boldsymbol{M}^\top \boldsymbol{Q} \boldsymbol{M}) \boldsymbol{p} \\ s.t. \quad & \boldsymbol{f}(0) = \boldsymbol{x}_s,\ \boldsymbol{f}(T) = \boldsymbol{x}_m \\ & (p_i^x, p_i^y, p_i^z) \in \mathcal{P}\ \forall i \end{aligned} \tag{21}$$

where $\boldsymbol{p} = (p_1^x, \cdots, p_N^x, p_1^y, \cdots, p_N^y, p_1^z, \cdots, p_N^z)^\top$, $\boldsymbol{M}$ is a block-diagonal matrix transforming control points into polynomial coefficients according to (19), $\boldsymbol{Q}$ is the Hessian matrix of J, and T is the expected time period to finish the planning trajectory.

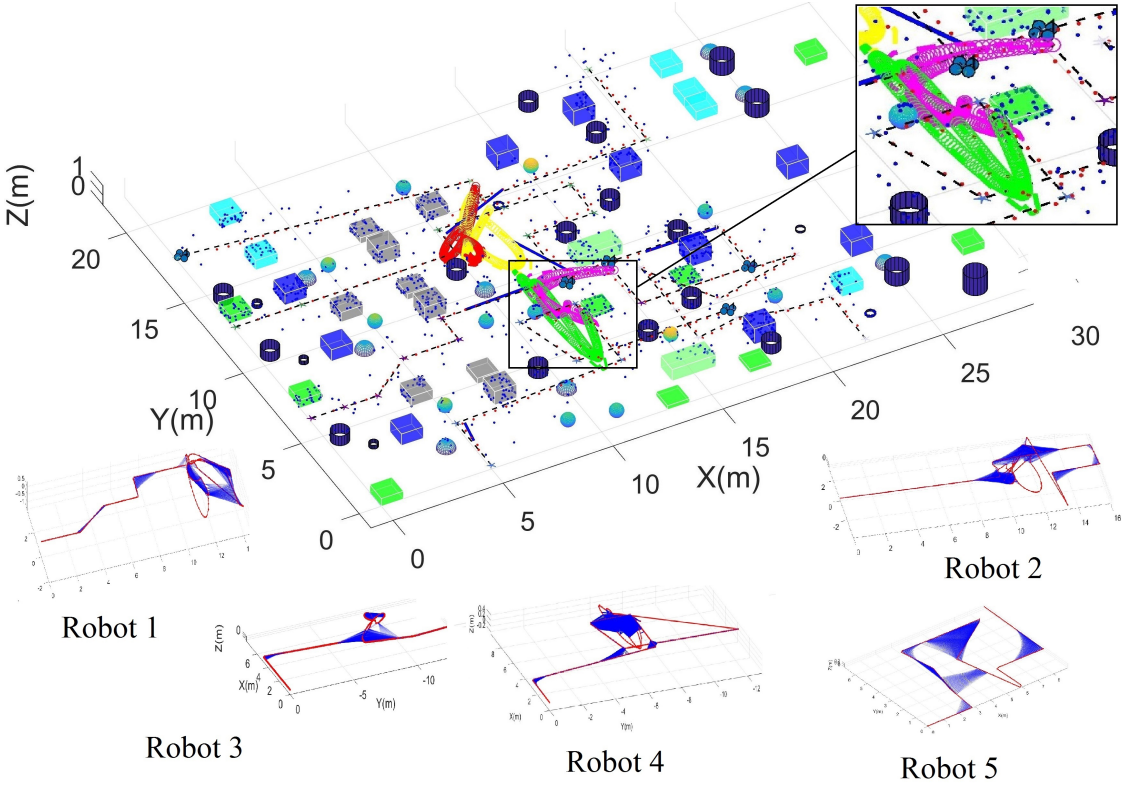

Figure 12: Active SLAM trajectory (colorful curves), estimated pose graph results (red points and lower parts) and relative measurements (lower parts) in an environment simulation

Finally, this convex trajectory optimization problem is solved using the interior-point method. After the i-th robot obtains significant measurements, it will send a measurement to the j-th robot using the communication system. This additional information can help the j-th robot to augment its pose graph, which repairs the weakness with low communication cost.

9 Figure 4

For better readability, we provide a larger version of Fig. 4 in [1]. This figure is reproduced here as Fig. 12.

References

[1] Y. Chen, L. Zhao, S. Huang, R. Fitch, K. M. B. Lee and C. Yoo, Sell your weakness: a low-cost high-efficiency active pose-graph SLAM method for multiple robots, IEEE Robotics and Automation Letters (RA-L), 2018, submitted.

[2] Y. Chen, K. Khosoussi, S. Huang, L. Zhao, and G. Dissanayake, "Cramér-Rao bounds and optimal design metrics for pose-graph SLAM," 2018. https://github.com/cyb1212/A-submitted-paper/blob/master/main.pdf

[3] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. Robotics and autonomous systems, 57(12), pp.1198-1210, 2009.

[4] V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik, Fast covariance recovery in incremental nonlinear least square solvers, in IEEE International Conference on Robotics and Automation (ICRA), pp. 4636-4643, 2015.

[5] K. Khosoussi, M. Giamou, G.S. Sukhatme, S. Huang, G. Dissanayake, and J.P. How, "Reliable graph topologies for SLAM," *The International Journal of Robotics Research*, 2018.