2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Probabilistic Active Loop Closure for Autonomous Exploration

He Yin*, Jong Jin Park*, Marcelino Almeida*, Martin Labrie, Jim Zamiska, Richard Kim

Abstract— **When a mobile robot autonomously explores an indoor space to produce a localization and navigation map, it is important to create both a stable pose graph and a high-quality occupancy map that covers all the navigable areas. In this work, we propose a novel probabilistic active loop closure framework which attempts to maximally reduce pose graph uncertainty during exploration and improves occupancy map quality. We calculate a probabilistic reward of getting a loop closure at any pose on a pose graph, which considers both how much pose graph uncertainty would be reduced by getting a loop closure there, and the robot's travel cost to navigate to that pose. By choosing poses that provide the largest rewards, we can maximally reduce pose graph uncertainty while avoiding long travel times. The effectiveness of the method is illustrated through on-device testing in various floor plans.**

I. Introduction

Mobile robots that operate in indoor settings typically need a high quality map of the operating environment in order to navigate and localize well within that environment. Manually constructing such maps can be time-consuming and error-prone, especially for large floor plans, and requires specialized skills. The objective of autonomous exploration is to construct a high quality map without any supervision or human intervention. Traditional information-theoretic exploration strategies including [1]–[4] can explore maps quickly but do not take the resulting map quality into account.

The final map quality depends on the quality of localization during exploration [5]. Since pose estimates from a Simultaneous Localization and Mapping (SLAM) system (especially a visual SLAM system) drift during exploration, it is necessary for the robot to perform active loop closure (ALC) during exploration by guiding the robot to previously visited areas with low uncertainty so SLAM can get a loop closure there. Performing ALC during exploration prevents distortions of the occupancy map which could, for example, seal off narrow passageways and prevent the robot from exploring the entire floor

In this work, we present a probabilistic ALC algorithm for indoor robots to maintain a stable pose graph and enhance map quality during exploration. Our main contributions are

1) We propose two types of computationally lightweight metrics to measure the uncertainty in a pose graph: one is distance based, and other is posterior covariance matrix based. Using the proposed uncertainty metrics we provide a method to predict the pose graph uncertainty reduction we can obtain by getting a loop closure at a given cluster of keyframes.
2) We model loop closure as a probabilistic event, and propose a model for capturing the probability of getting a loop closure at a given cluster of keyframes.
3) By combining the probability and uncertainty reduction of getting a loop closure at a cluster of keyframes, we propose a probabilistic reward function for active loop closure planning. This ensures that the planning behavior is probabilistically robust.
4) We optimize/reduce the computational burden by providing a branch and bound based method to search for the cluster of keyframes that achieves the largest reward at low computation cost.

* authors contributed equally.
All authors are with Amazon Lab126. {heyinz,jongpark,mmalmeid,labrieml,jzamiska,richk}@amazon.com

II. Related Work

The active SLAM and active perception communities have produced numerous studies utilizing a diverse range of sensors, SLAM algorithms, uncertainty metrics, and exploration termination criteria. In [6], these works are categorized into three sets of strategies: belief-space planning strategies [7]–[11], deep reinforcement learning strategies [12]–[16], and partially-observable Markov Decision Process (POMDP) strategies. Our approach fits into the POMDP category.

The POMDP Strategies [17]–[24] formulate the Active SLAM problem as POMDPs, and commonly divide the problem into three stages for the purpose of facilitating its resolution: (i) generation of a set of discrete candidate locations, (ii) evaluation of the utility at those locations, (iii) selection of the optimal location. For (i), an efficient way of generating candidate locations for active loop closure is proposed in [17], where the minimum map and graph distances from the robot to the candidate locations are required. For (ii), expected reduction in entropy of the map has been used as the utility function in [18]–[20]. Besides entropy, covariance matrices of the pose graph have also been utilized in the utility function, e.g. [21] introduces a method to roughly estimate full graph D-optimality, reducing computation time by up to 90% (computation time still scales cubically with the pose graph's number of keyframes). This estimation is subsequently used as the utility function within the paper's framework. However, this full graph D-optimality estimation could still be computationally challenging for robots operating in moderately large spaces. For (iii), an active visual SLAM system is introduced in [22], which leverages the full graph D-optimality estimate from [21] as the utility function, and proposes a decision-making mech-

anism that switches between exploration and exploitation. Full graph A-optimality is used as the utility function in [24] to compute the optimal trajectory. Selection of the optimal location has also been studied in [23], which considers the match effect, the expected impact of a loop closure on global optimization.

The most related work is presented in our previous paper [5], where we introduced the concept of lighthouses, a visually informative location with a panoramic view, that we create during exploration. The ALC planner chooses the nearest lighthouses as the optimal locations and drives the robot periodically to those locations to reduce robot pose uncertainty. However, since lighthouses are the only locations for the ALC planner to choose from, for floor plans that have few lighthouses the robot may need to travel a long way to reach a lighthouse. This increases the risk of the map getting distorted while traveling. Also going to a lighthouse doesn't necessarily maximally reduce pose graph uncertainties, and sometimes introduces map quality issues.

III. Notations

Let $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ be a pose graph where $\boldsymbol{\mathcal{V}} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ is the set of vertices (also called graph nodes). Each graph node has a corresponding keyframe. $\boldsymbol{\mathcal{E}}$ is the set of edges. We denote the pose of the vertex $\boldsymbol{v}_i$ as $\boldsymbol{p}_{\boldsymbol{v}_i}$. We assume that the latest node $\boldsymbol{v}_r$ on the pose graph corresponds to the robot's current pose $\boldsymbol{p}_r$. $l(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}, \boldsymbol{\mathcal{M}})$ denotes the length of the shortest path between the poses of the nodes $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ calculated on a 2D occupancy grid map $\boldsymbol{\mathcal{M}}$ using algorithms like A^*. $l(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\mathcal{G}})$ denotes the length of the shortest path between nodes $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ in $\boldsymbol{\mathcal{G}}$ computed by the sum over the lengths of the traversed edges. $d(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j})$ denotes the Euclidean distance between the positions of poses $\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}$. With a slight abuse of notation, we also use $l(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}, \boldsymbol{\mathcal{G}})$ to represent the same meaning as $l(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\mathcal{G}})$. We refer to $l(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}, \boldsymbol{\mathcal{M}})$ and $l(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}, \boldsymbol{\mathcal{G}})$ as shortest map and graph distances.

IV. Proposed Pose Graph Uncertainty Metrics

Over time, many metrics have been proposed to compute uncertainties in a pose graph. Notably, [21] presents a set of modern optimality criteria in Active SLAM. The metric that has been more prominently used in recent literature is the D-optimality [22], [25], which entails computing the determinant of the pose graph's covariance matrix (i.e., the inverse of the Hessian). However, computing the D-optimality criterion can be quite expensive, as it scales with the graph size in $\mathcal{O}(n^3)$, making it very costly to compute D-optimality even for reasonably sized pose graphs [21].

Another problem that arises is that we need to have an estimated uncertainty of the pose graph if we actually add a loop closure. This is needed so we can find vertices whose loop closures would maximally reduce uncertainty.

To overcome the problems mentioned above and to avoid hefty computations, we perform some simplifications on the formulation, and propose two types of computationally lightweight uncertainty metrics to determine best candidates for loop closure. In Section IV-A, we introduce a posterior covariance matrix-based uncertainty metric, and provide the definition of Neighbor D-optimality. In Section IV-B, we provide a Distance-Based uncertainty metric that further diminishes the need for computational resources.

A. Posterior Covariance Based Uncertainty Metric

1) Linear Least Squares Posterior Covariance Update: In this section, we deal with the posterior covariance update for a linear problem (hence, it should not apply directly to a nonlinear problem such as pose graph optimization (PGO)). Assume that we have measurements $\boldsymbol{z}$ of a quantity $\boldsymbol{x}$, and that those measurements are polluted by measurement noise $\boldsymbol{\eta}$, with $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_\eta)$:

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\eta}, \tag{1}$$

where $\boldsymbol{H}$ is a known mapping matrix. Additionally, assume that we have an initial rough estimation of $\boldsymbol{x}$, denoted as $\boldsymbol{x}^- \sim \mathcal{N}(\boldsymbol{x}, \boldsymbol{\Sigma}_-)$. We would like to obtain the posterior covariance of the estimate $\hat{\boldsymbol{x}}$, denoted as $\boldsymbol{\Sigma}_+$.

This problem is classically known as the "Recursive Least Squares" problem [26]. The optimal linear estimator $\hat{\boldsymbol{x}}$ for this problem (in the Least Squares sense) and its associated covariance $\boldsymbol{\Sigma}_+$ are given as:

$$\hat{\boldsymbol{x}} = \boldsymbol{x}^- + \boldsymbol{\Sigma}_+ \cdot \boldsymbol{H}^T \cdot \boldsymbol{\Sigma}_\eta^{-1} \left(\boldsymbol{z} - \boldsymbol{H}\boldsymbol{x}^-\right), \tag{2}$$

$$\boldsymbol{\Sigma}_+ = \left(\boldsymbol{\Sigma}_-^{-1} + \boldsymbol{H}^T \boldsymbol{\Sigma}_\eta^{-1} \boldsymbol{H}\right)^{-1}. \tag{3}$$

2) Pose Graph Posterior Covariance Update: The PGO problem is not a linear problem and it does not have a measurement model of the type of Eq. 1. Still, let's assume that we have an estimate of the relative pose between two nodes i and j, given by $\boldsymbol{p}_{i,j}^- \in SO(3) \sim \mathcal{N}(\boldsymbol{p}_{i,j}, \boldsymbol{\Sigma}_-)$, and that we have a new loop closure between these poses. Let's make the simplifying assumption that this new loop closure can be regarded as a new measurement:

$$\boldsymbol{z} = \boldsymbol{p}_{i,j} + \boldsymbol{\eta}_{i,j}, \tag{4}$$

where $\boldsymbol{\eta}_{i,j} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{i,j})$ is a noise vector in $SO(3)$ and represents the odometry error that the robot accumulates when navigating between nodes i and j. The equation above is similar to Eq. 1 with $\boldsymbol{H}$ being an identity matrix.

Assuming that this problem is locally linear, then we can estimate the posterior covariance after a loop closure between vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ as $\hat{\boldsymbol{\Sigma}}_{i,j+}^{-1} = \boldsymbol{\Sigma}_-^{-1} + \boldsymbol{\Sigma}_{i,j}^{-1}$. If we use information matrices instead, we can use the expression:

$$\hat{\boldsymbol{\Sigma}}_{i,j+}^{-1} = \boldsymbol{I}_- + \boldsymbol{I}_{i,j}, \tag{5}$$

where we have defined $\boldsymbol{I}_- \triangleq \boldsymbol{\Sigma}_-^{-1}$ and $\boldsymbol{I}_{i,j} \triangleq \boldsymbol{\Sigma}_{i,j}^{-1}$. Note how this is computationally cheap to compute, since all of the matrices in Eq. 5 have dimensions 6×6.

3) Neighbor D-optimality: Now that we can compute an estimate of the posterior covariance matrix $\hat{\boldsymbol{\Sigma}}_{i,j+}$ between a candidate pair of source/target vertices, we define the Neighbor D-optimality as $D_{opt} = \det\left(\hat{\boldsymbol{\Sigma}}_{i,j+}\right)$ and we will use it as the uncertainty metric. The uncertainties between

the pair of nodes i and j before and after establishing a loop closure edge between them are defined as follows

$$U_-(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}) := \det\left(\boldsymbol{\Sigma}_-\right), \; U_+(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}) := \det\left(\hat{\boldsymbol{\Sigma}}_{i,j+}\right).$$

The justification for using just Neighbor D-optimality as opposed to the whole graph D-optimality is that by trying to find the pair of neighbors whose uncertainty reduces maximally, then one can expect that it would also maximally reduce graph uncertainty. Although this will not always be true, we have noted that this is a good approximation. Our method is more appropriate for constrained systems, being computed with $\mathcal{O}(1)$ complexity. Graph D-optimality's complexity, on the other hand, grows cubically with the number of vertices.

4) Uncertainty Reduction: Throughout the paper, we use $\boldsymbol{p}_r$ to represent the pose of the node $\boldsymbol{v}_r$ where the robot is located during exploration *before* heading out to a vertex $\boldsymbol{v}$ on the pose graph to get a loop closure. To choose the optimal vertex, we use Neighbor D-optimality to predict how much relative uncertainty between $\boldsymbol{v}_r$ and $\boldsymbol{v}$ can be reduced when having a new loop closure edge between them

$$\begin{aligned}\Delta U(\boldsymbol{p}_{\boldsymbol{v}}) &= U_-(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}) \;/\; U_+(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}) \\ &= \det\left(\boldsymbol{\Sigma}_-\right) / \det\left(\hat{\boldsymbol{\Sigma}}_+\right). \end{aligned} \tag{6}$$

B. Distance Based Uncertainty Metric

1) Neighbor T-optimality and Its Approximation: Similar to Neighbor D-optimality, the scaled trace of covariance matrices between a pair of vertices can also approximate their relative uncertainty. We refer to it as the Neighbor T-optimality, as opposed to the full graph T-optimality described in [25], [27]. Since this type of relative pose uncertainty cannot be greater than the uncertainty in odometry of the travelled path between them, we can also write

$$\frac{1}{\ell}\text{trace}(\boldsymbol{\Sigma}(\boldsymbol{v}_i, \boldsymbol{v}_j)) \leq f(l(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\mathcal{G}})) := c_o \; l^2(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\mathcal{G}}),$$

where ℓ is the dimension of the state vector ($\ell = 6$ in our case), and $c_o > 0$ is a coefficient that converts the shortest graph distance to the scalar odometry uncertainty. Thus we can also write

$$\sqrt{\frac{1}{\ell \; c_o} \; \text{trace}(\boldsymbol{\Sigma}(\boldsymbol{v}_i, \boldsymbol{v}_j))} \leq l(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\mathcal{G}}), \tag{7}$$

which makes the shortest graph distance $l(\cdot)$ over the graph $\boldsymbol{\mathcal{G}}$ also a measure of relative pose uncertainty between any pair of nodes. It is useful for robots without access to covariance matrices to estimate pairwise relative uncertainties.

2) Uncertainty Reduction: Since the robot traveled from $\boldsymbol{p}_{\boldsymbol{v}}$ to $\boldsymbol{p}_r$, the shortest graph distance between them is $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}})$. By assuming the robot will (i) now travel along the shortest path in the occupancy map from $\boldsymbol{p}_r$ back to $\boldsymbol{p}_{\boldsymbol{v}}$, (ii) the robot's future pose $\boldsymbol{p}'_r$ will coincide with $\boldsymbol{p}_{\boldsymbol{v}}$, and (iii) will be able to get a loop closure at $\boldsymbol{p}_{\boldsymbol{v}}$, the shortest graph distance between $\boldsymbol{p}_r$ and $\boldsymbol{p}_{\boldsymbol{v}}$ on the graph $\boldsymbol{\mathcal{G}}'$ with the hypothetical edge then becomes $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{M}})$. This process is illustrated in Fig. 1. By using the newly defined Distance

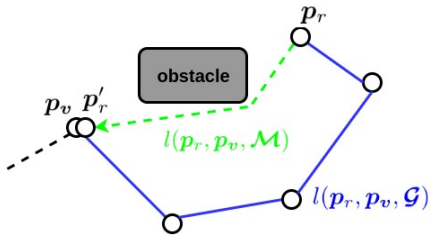


Fig. 1: By adding an edge between $\boldsymbol{p}_r$ and $\boldsymbol{p}_{\boldsymbol{v}}$, their shortest graph distance reduces from $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}})$ to $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{M}})$.

Based uncertainty metric, the uncertainty reduction we get from this loop closure can be approximately predicted as

$$\begin{aligned}\Delta U(\boldsymbol{p}_{\boldsymbol{v}}) &= U_-(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}) - U_+(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}) \\ &= l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}}) - l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}}') \\ &= l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}}) - l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{M}}). \end{aligned} \tag{8}$$

V. ALC Target Selection

A. ALC Candidate Definition and Construction

Generally speaking, the robot can choose to go back to any keyframe on the pose graph to attempt a loop closure, but not all of the keyframes could provide enough uncertainty reduction. Hence, we present a method to create loop closure candidates $\mathcal{T}_{alc}$ from chosen keyframes, selecting the ALC target from this set.

Similar to our previous work [5], we actively create lighthouses during exploration by rotating 360° in-place when we reach a feature rich location to emulate a panoramic view for narrow field of view robots. A lighthouse is defined by the set of generated keyframes during rotation. However, unlike [5], we lift the restriction of only going back to lighthouses.

An ALC candidate $\boldsymbol{\tau}$ is a structure made up of a cluster of keyframes with high information gain for loop closures. ALC candidates come from two sources. The first one is the set of proactively created lighthouses, and the second is a set of passively identified clusters of keyframes.

The process of constructing ALC candidates is summarized in Alg. 1. First, we convert all the lighthouses into ALC candidates. Then we downsample a keyframe from a pose graph every few keyframes, which satisfies the following criteria: (a) it should be within a maximum range c_E from the robot, (b) its graph distance to robot should be above a minimum threshold c_G, (c) its view score should be above a threshold c_s to increase the probability of getting loop closures. Then we group the keyframes in the small neighborhood of the downsampled keyframe, and treat this spatial cluster of keyframes $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N\}$ as an ALC candidate.

An ALC candidate $\boldsymbol{\tau}_i$ has the following attributes:

- representative pose $\boldsymbol{p}_{\boldsymbol{\tau}_i}$
- distances: $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i}, \boldsymbol{\mathcal{M}})$, $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i}, \boldsymbol{\mathcal{G}})$, $d(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i})$
- $\boldsymbol{\mathcal{V}}(\boldsymbol{\tau}_i)$: the set of keyframes in its neighborhood
- view scores at those keyframes $\{s_v(\boldsymbol{v}_1), \cdots, s_v(\boldsymbol{v}_N)\}$
- $P_{LC}(\boldsymbol{p}_{\boldsymbol{\tau}_i})$: the probability of getting loop closure at $\boldsymbol{\tau}_i$ defined in Sec. V-B
- $\Delta U(\boldsymbol{p}_{\boldsymbol{\tau}_i})$: pose graph uncertainty reduction by creating a loop closure at $\boldsymbol{p}_{\boldsymbol{\tau}_i}$ defined in Sec. IV-A & IV-B

- $R(\boldsymbol{\tau}_i)$: reward the robot can get by going to $\boldsymbol{p}_{\boldsymbol{\tau}_i}$ defined in Sec. V-C
- $R_{ub}(\boldsymbol{\tau}_i)$: upper bound on $R(\boldsymbol{\tau}_i)$ defined in Sec. V-D.1.

View score is the number of detected visual features at a certain pose normalized by the regions of non-maximal suppression, and it is within the range of $[0, 1]$. It correlates with the probability of getting a loop closure at that pose.

Algorithm 1 ALC candidates Construction

Input: pose graph $\boldsymbol{\mathcal{G}}$, lighthouses $\mathcal{L}$, robot pose $\boldsymbol{p}_r$
for lighthouse L_i in $\mathcal{L}$ **do**
  create an ALC candidate object $\boldsymbol{\tau}$, and $\boldsymbol{\tau} \leftarrow L_i$
  insert $\boldsymbol{\tau}$ into $\mathcal{T}_{alc}$
end for
for every few keyframe in $\boldsymbol{\mathcal{G}}$ **do**
  downsample a keyframe $\boldsymbol{v}_i$
  if $d(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}_i}) > c_E$ or $s_v(\boldsymbol{v}_i) < c_s$ or $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{\mathcal{G}}) < c_G$ **then**
    continue
  end if
  create an ALC candidate object $\boldsymbol{\tau}$
  $\boldsymbol{\mathcal{V}}(\boldsymbol{\tau}) \leftarrow$ keyframes in the neighborhood of $\boldsymbol{v}_i$
  $\boldsymbol{p}_{\boldsymbol{\tau}} \leftarrow \boldsymbol{p}_{\boldsymbol{v}_i}$, and updateAttributes($\boldsymbol{\tau}$)
  insert $\boldsymbol{\tau}$ into $\mathcal{T}_{alc}$
end for
Output: set of ALC candidates $\mathcal{T}_{alc}$

B. Probabilistic Loop Closure Event

When the robot has travelled sufficiently far away from a node, the possibility of the robot being able to go back to that node decreases as function of the travel distance. This problem is particularly pronounced for robots that have a narrow (esp. less than $180°$) field of view, as the sensor views on the way back will be often completely different from the views on the way out. This results in localization accuracy not improving (i.e. pose uncertainty will grow) while the robot is traveling back to the target node.

Based on this idea, we define the probability of getting loop closure at an ALC candidate $\boldsymbol{\tau}$ with a single keyframe $\boldsymbol{v}$ in the neighborhood as

$$P_{LC}(\boldsymbol{p}_{\boldsymbol{\tau}}) = \tanh\left(c_v s_v(\boldsymbol{p}_{\boldsymbol{v}})\right) \times \exp\left(-\frac{l_{-}(\boldsymbol{p}'_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}}')^2}{c_l^2}\right)$$

where c_v and c_l are tuning parameters, $s_v(\boldsymbol{p}_{\boldsymbol{v}})$ is the view score at pose $\boldsymbol{p}_{\boldsymbol{v}}$, and $l_{-}(\boldsymbol{p}'_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}}') = l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{v}}, \boldsymbol{\mathcal{G}}) + l(\boldsymbol{p}_r, \boldsymbol{p}'_r, \boldsymbol{\mathcal{M}})$ represents the estimated minimum relative pose uncertainty between the robot's future pose $\boldsymbol{p}'_r$ and the keyframe $\boldsymbol{v}$ after the robot had attempted to move to $\boldsymbol{v}$ to create a loop closure, but before the loop closure event. The expression of P_{LC} captures the fact that probability of getting loop closure is a monotonic increasing function of the view score at $\boldsymbol{v}$; and it is a monotonic decreasing function of the relative uncertainty between the robot's future pose and $\boldsymbol{v}$. In future works, we plan to use data-driven methods to come up with a more accurate model.

The probability of getting loop closure at an ALC candidate $\boldsymbol{\tau}$ with a cluster of keyframes $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_N\}$ in the neighborhood can then be derived as

$$P_{LC}(\boldsymbol{p}_{\boldsymbol{\tau}}) = 1 - \Pi_{i=1}^{N}\left(1 - P_{LC}(\boldsymbol{p}_{\boldsymbol{v}_i})\right) \tag{9}$$

C. Probabilistic Reward Function

From the current robot pose $\boldsymbol{p}_r$, the reward function of creating a loop closure to an ALC candidate $\boldsymbol{\tau}_i$ can be defined

$$R(\boldsymbol{\tau}_i) = -c_t\ l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i}, \boldsymbol{\mathcal{M}}) + P_{LC}(\boldsymbol{p}_{\boldsymbol{\tau}_i})\ \Delta U(\boldsymbol{p}_{\boldsymbol{\tau}_i}), \tag{10}$$

where the first term $-l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i}, \boldsymbol{\mathcal{M}})$ is the traveling cost for traveling from robot pose to the ALC candidate $\boldsymbol{\tau}_i$. The second term $P_{LC}(\boldsymbol{p}_{\boldsymbol{\tau}_i})\ \Delta U(\boldsymbol{p}_{\boldsymbol{\tau}_i})$ is the expected uncertainty reduction at the ALC candidate $\boldsymbol{\tau}_i$. c_t is a tuning parameter that reflects the relative importance between the traveling cost and expected uncertainty reduction.

D. ALC Target Selection

Having the reward function defined, the most straightforward way of finding the ALC target $\boldsymbol{\tau}^*$ is to run A^* algorithm to find minimum-distance paths from the robot pose to all the ALC candidates, evaluate the reward at each ALC candidate, and then choose the one that maximizes the reward.

However, this might be time-consuming for a robot with limited computation power since A^* is computationally heavy especially in larger maps. For the Distance-Based uncertainty metric, we can make use of its property and find the ALC target without evaluating reward at each ALC candidate by using a branch and bound (BNB) based method.

1) Reward Upper Bound: For the ease of describing the BNB based ALC target selection method in the following subsection, we provide a way of computing the reward upper bound $R_{ub}(\boldsymbol{\tau}_i)$: to replace $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i}, \boldsymbol{\mathcal{M}})$ by $d(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i})$ everywhere in the reward computation in Eq. 10. $R_{ub}(\boldsymbol{\tau}_i)$ has noticeable properties: it is an upper bound on $R(\boldsymbol{\tau}_i)$, and is computationally much cheaper to estimate than a reward.

2) Branch and Bound Based Method: A BNB based method is summarized in Alg. 2 to solve for the ALC target with minimal computation burden. The first step is to initialize the ALC target by choosing the one from the set $\mathcal{T}_{alc}$ with the largest reward upper bound. We compute the minimum map distance from the initialized ALC target $\boldsymbol{\tau}^*$ to the robot, and compute its reward $R(\boldsymbol{\tau}^*)$.

Then we enter the main algorithm of BNB. While the set $\mathcal{T}_{alc}$ is not empty, we perform the following steps. First we compare the reward upper bound of each ALC candidate with $R(\boldsymbol{\tau}^*)$. If the reward upper bound of an ALC candidate is smaller than $R(\boldsymbol{\tau}^*)$, then the reward of that ALC candidate can not be larger than $R(\boldsymbol{\tau}^*)$. Therefore, this ALC candidate can not be the ALC target, and will be removed from $\mathcal{T}_{alc}$. After comparing the reward upper bound of all the ALC candidates with $R(\boldsymbol{\tau}^*)$, if $\mathcal{T}_{alc}$ is empty, then the ALC target is found and we can break the while loop. If $\mathcal{T}_{alc}$ is not empty, then we choose the ALC candidate with the largest reward upper bound as the potential ALC target $\boldsymbol{\tau}'$ and compute its reward $R(\boldsymbol{\tau}')$. Then we can compare $R(\boldsymbol{\tau}')$ with $R(\boldsymbol{\tau}^*)$. If $R(\boldsymbol{\tau}')$ is larger, then $\boldsymbol{\tau}'$ becomes the ALC target.

(a) ALC triggered for the 1^{st} time (b) Getting loop closure at 1^{st} $\boldsymbol{\tau}^*$ (c) ALC triggered for 2^{nd} time (d) Getting loop closure at 2^{nd} $\boldsymbol{\tau}^*$

Fig. 2: Case study in Home 1. (a) We show the selected $\boldsymbol{\tau}^*$ (a big pink box) when the ALC planner triggered for the 1^{st} time. The shortest map and graph distances between the robot and $\boldsymbol{\tau}^*$ are 16.95 m and 8.55 m. By getting a loop closure at the $\boldsymbol{\tau}^*$, we can reduce the uncertainty between $\boldsymbol{v}_r$ and $\boldsymbol{\tau}^*$ by half. (b) We show the pose graph after getting a loop closure at the 1^{st} $\boldsymbol{\tau}^*$. (c) When ALC planner triggered for the 2^{nd} time, our algorithm selects a $\boldsymbol{\tau}^*$ on the left side of the home, which can reduce the uncertainty by around 18.2 m. If the method from [5] was used, it would choose the nearest lighthouse (pink sphere at the top of the home) as $\boldsymbol{\tau}^*$, which can only reduce uncertainty by around 7.4 m. (d) Upon creating a loop closure at the 2^{nd} $\boldsymbol{\tau}^*$, the pose graph now exhibits two compact loops, demonstrating the stability of the overall structure.

Algorithm 2 ALC Target selection by Branch and Bound

Input: set of ALC candidates $\mathcal{T}_{alc}$, robot pose $\boldsymbol{p}_r$
$\boldsymbol{\tau}^* \leftarrow \arg\max_{\boldsymbol{\tau}_i \in \mathcal{T}_{alc}} R_{ub}(\boldsymbol{\tau}_i)$ ▷ Initialize ALC target
compute $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}^*}, \boldsymbol{\mathcal{M}})$, $R(\boldsymbol{\tau}^*)$. Remove $\boldsymbol{\tau}^*$ from $\mathcal{T}_{alc}$
while $\mathcal{T}_{alc}$ is not empty **do**
  for $\boldsymbol{\tau}_i$ in $\mathcal{T}_{alc}$ **do**
    if $R_{ub}(\boldsymbol{\tau}_i) < R(\boldsymbol{\tau}^*)$ **then**
      Remove $\boldsymbol{\tau}_i$ from $\mathcal{T}_{alc}$
    end if
  end for
  if $\mathcal{T}_{alc}$ is empty **then**
    break the while loop
  end if
  $\boldsymbol{\tau}' \leftarrow \arg\max_{\boldsymbol{\tau}_i \in \mathcal{T}_{alc}} R_{ub}(\boldsymbol{\tau}_i)$ ▷ potential target
  compute $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}'}, \boldsymbol{\mathcal{M}})$, $R(\boldsymbol{\tau}')$. rm $\boldsymbol{\tau}'$ from $\mathcal{T}_{alc}$
  if $R(\boldsymbol{\tau}') > R(\boldsymbol{\tau}^*)$ **then**
    $\boldsymbol{\tau}^* \leftarrow \boldsymbol{\tau}'$
  end if
end while
Output: ALC target $\boldsymbol{\tau}^*$

E. Overall ALC Target Computation Algorithm

By combing the algorithm for constructing ALC candidates (Alg. 1), with the one for selecting an ALC target from the ALC candidates (Alg. 2), we establish the overall process for computing the ALC target as outlined in Alg. 3.

Algorithm 3 ALC Target Computation Overview

Input: pose graph $\boldsymbol{\mathcal{G}}$, lighthouses $\mathcal{L}$, robot pose $\boldsymbol{p}_r$
$\mathcal{T}_{alc} \leftarrow$ constructCandidatePoses($\boldsymbol{\mathcal{G}}, \boldsymbol{p}_r, \mathcal{L}$) ▷ Alg. 1
compute $R_{ub}(\boldsymbol{\tau}_i)$ for all $\boldsymbol{\tau}_i \in \mathcal{T}_{alc}$
$\boldsymbol{\tau}^* \leftarrow$ targetPoseSelection($\mathcal{T}_{alc}, \boldsymbol{p}_r$) ▷ Alg. 2
Output: ALC target $\boldsymbol{\tau}^*$

VI. Active Loop Closure Planning

The ALC planner guides the robot to target $\boldsymbol{\tau}^*$, rotating 360° there to enhance loop closure chances for robots with limited field of view. Like in [5], our strategy toggles between frontier and ALC planners during exploration. Afterward, the refinement stage deploys the path coverage planner to further stabilize the pose graph. This paper introduces new ALC target computation and revises ALC's triggering conditions.

A. Triggering Conditions

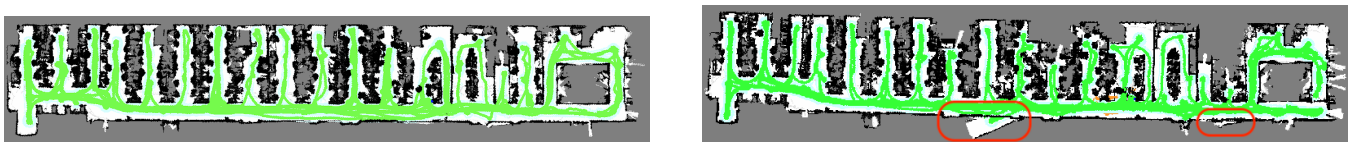
We collect perception data, including the occupancy map and pose graph, and compute an ALC target $\boldsymbol{\tau}^*$ using Algorithm 3. Differently from the periodic ALC triggering in [5], we gauge uncertainty reduction at $\boldsymbol{\tau}^*$ against a threshold and activate the ALC planner if it surpasses this threshold. Our paper employs an adaptive threshold, which declines monotonically with time since last ALC, making ALC activation easier as time elapses.

VII. Experiments

In Section IV, we have provided two metrics for computing the uncertainty reduction: Neighbor D-optimality and Distance-Based Metric. In Section VII-A, we show how the Neighbor D-optimality compares with traditional graph D-optimality. In Section VII-B we present results obtained on a real robot in exploration with a low compute budget and narrow-FOV depth sensor using the Distance-Based Metric.

A. Validate Neighbor D-optimality

To validate the Neighbor D-optimality-based candidate selection criterion (Section IV-A.3), we conducted GTSAM experiments [28]. We employed the *w100* example with 100 vertices and 300 factors. These factors consist of 99 odometry, 200 loop closures, and 1 prior at the origin keyframe. In this scenario, the robot begins at the origin $(0, 0)m$, moves right, loops thrice in the upper-right region (with loop closures), shifts left, loops once in the upper left, then loops at the bottom before returning to the origin.



(a) Occupancy map obtained using the method from this work

(b) Occupancy map obtained using the method from [5]

Fig. 3: Exploration results at Office 1. The proposed method's occupancy map (Fig. 3a) shows better quality than the previous work's (Fig. 3b), with no observed distortion or drift. Conversely, Fig. 3b exhibits significant distortion in the areas marked in red.

To evaluate the Neighbor D-optimality criterion, we compare it against pure D-optimality. Our goal is to show how our method differs from what is commonly used in recent literature. To have a fair comparison, we use the reward function of Eq. 10 with $c_t = 0$ and $P_{LC}(\boldsymbol{p}_{r_i}) = 1, \forall i \in \{1, \cdots, n\}$. We pick the candidate whose ΔU is the largest, where ΔU is defined in Eq. 6. We obtain the posterior covariance matrix by estimating the required odometry covariance $\boldsymbol{\Sigma}_{i,j}$ to link candidate source pose $\boldsymbol{p}_{\boldsymbol{v}_i}$ and target pose $\boldsymbol{p}_{\boldsymbol{v}_j}$. This estimation is a function of the inter-node distance $l(\boldsymbol{p}_{\boldsymbol{v}_i}, \boldsymbol{p}_{\boldsymbol{v}_j}, \boldsymbol{\mathcal{M}})$. We then calculate the posterior neighbor covariance matrix using Eq. 5.

To compute D-optimality on the whole graph, we use the method [21]: given a full graph posterior covariance matrix $\boldsymbol{\Sigma}_+ \in \mathbb{R}^{\ell \times \ell}$, the D-optimality is computed as follows:

$$D_{opt}(\boldsymbol{\Sigma}_+) \triangleq \exp\left(\frac{1}{\ell}\sum_{k=1}^{\ell} \log\left(\lambda_k\right)\right), \tag{11}$$

where $\lambda_1, \lambda_2, \cdots, \lambda_\ell$ are the eigenvalues of the covariance matrix $\boldsymbol{\Sigma}_+$. Notice that computing D-optimality on the full graph covariance is quite expensive, as we need to compute the eigenvalues of the covariance matrix as in Eq. 11, which has computational complexity of $\mathcal{O}(n^3)$.

We compared Neighbor D-optimality with full graph D-optimality by evaluating the ideal loop closure candidates for all source poses in the *w100* pose graph. Interestingly, we observed that both methods produced identical decisions. Calculating the Neighbor D-optimality candidates for all 100 source nodes took just 0.048 seconds, whereas achieving the same with full graph D-optimality consumed 295.0 seconds. The D-optimality validation was done on a laptop with a 5.1 [GHz] Intel Core i7 processor and 32 [GB] of RAM.

It would be tempting to claim that Neighbor D-optimality is equivalent to graph D-optimality. However, we note that the former minimizes local uncertainty, whereas the latter aims at overall graph uncertainty reduction. Though it is logical that decreasing node pair uncertainty minimizes overall graph uncertainty, this is only validated for our assessed $w100$ example. Further investigations are needed to confirm equivalence of the two methods in a broader sense.

B. Real world Exploration Results

Due to lack of cheap access to covariance matrices in our onboard SLAM software, we only performed on-device exploration experiments using the Distance-Based uncertainty metric (Eq. 8). In the future works, we plan to utilize Neighbor D-optimality metric (Eq. 6) on device.

Selection of ALC targets during exploration in Home 1 (2000 ft^2) and the resulting pose graph are showcased in Fig. 2. Exploration experiments were also performed at Office 1 ($5,000$ ft^2) using the methods from both this paper and [5], and the results are shown in Fig. 3. Fig. 4 illustrates the effectiveness of the proposed BNB algorithm for computing $\boldsymbol{\tau}^*$ (Alg. 2). Finally, further experiments in various floor plans indicate that the proposed method (ALCv2) explores faster than [5] (ALCv1) as shown in Fig. 5.

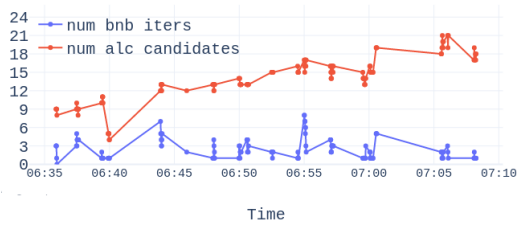


Fig. 4: Number of ALC candidates $N_{\boldsymbol{\tau}}$ v.s. Number of iterations in the while loop of Alg. 2 N_{bnb} at different timestamps in one exploration run in Office 1. Average N_{bnb} and $N_{\boldsymbol{\tau}}$ across all timestamps are 2.36 and 13.70, respectively. If BNB method was not used, every time when we calculate the ALC target $\boldsymbol{\tau}^*$, we would need to iterate through every single ALC target $\boldsymbol{\tau}_i$, and invoke A^* algorithm to calculate $l(\boldsymbol{p}_r, \boldsymbol{p}_{\boldsymbol{\tau}_i}, \boldsymbol{\mathcal{M}})$ for $N_{\boldsymbol{\tau}}$ times. By using our method, we only need to invoke A^* for N_{bnb} times. This saves computation power and time.

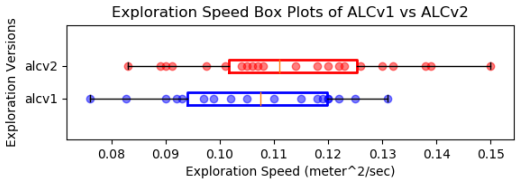


Fig. 5: Box plots of exploration speed (explored area divided by exploration duration) for ALCv1 (18 runs) v.s. ALCv2 (22 runs). Because ALCv2 is triggered opportunistically to reduce pose graph uncertainty rather than being triggered periodically and frequently, exploration can be concluded more efficiently.

VIII. Conclusion and Future Works

The paper introduces a probabilistic ALC framework for autonomous exploration to improve map quality. It models loop closure as a probabilistic event and introduces a reward function to guide ALC planning, aiming to reduce pose graph uncertainty. A branch and bound method is proposed to find the target pose efficiently, reducing computational burden.

The future works include (1) using data-driven methods to model the probability of getting loop closures; (2) showing the equivalence between Neighbor and full graph D-optimalities; (3) using Neighbor D-optimality on device.

Acknowledgments

The authors would like to thank Rajasimman Madhivanan, Xuewei Qi, and Arnie Sen for sharing their ideas that went into this work and for supporting the delivery of this work.

REFERENCES

[1] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[2] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[3] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3105–3112.

[4] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 179–185.

[5] M. Deshpande, R. Kim, D. Kumar, J. J. Park, and J. Zamiska, "Lighthouses and global graph stabilization: Active slam for low-compute, narrow-fov robots," in *2023 IEEE international conference on robotics and automation (ICRA)*, 2023.

[6] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *IEEE Transactions on Robotics*, 2023.

[7] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A pomdp approach," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[8] M. Kontitsis, E. A. Theodorou, and E. Todorov, "Multi-robot active slam with relative entropy optimization," in *2013 American Control Conference*. IEEE, 2013, pp. 2757–2764.

[9] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," 2010.

[10] J. Van Den Berg, S. Patil, and R. Alterovitz, "Efficient approximate value iteration for continuous gaussian pomdps," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 1832–1838.

[11] J. M. Porta, M. T. Spaan, and N. Vlassis, "Robot planning in partially observable continuous domains," 2005.

[12] L. Tai and M. Liu, "Mobile robots exploration through cnn-based reinforcement learning," *Robotics and biomimetics*, vol. 3, no. 1, pp. 1–8, 2016.

[13] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. in 2017 ieee," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31–36.

[14] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, 2016.

[15] J. A. Placed and J. A. Castellanos, "A deep reinforcement learning approach for active slam," *Applied Sciences*, vol. 10, no. 23, p. 8386, 2020.

[16] F. Chen, J. D. Martin, Y. Huang, J. Wang, and B. Englot, "Autonomous exploration under uncertainty via deep reinforcement learning on graphs," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6140–6147.

[17] C. Stachniss, D. Hahnel, and W. Burgard, "Exploration with active loop-closing for fastslam," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. IEEE, 2004, pp. 1505–1510.

[18] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters." in *Robotics: Science and systems*, vol. 2, 2005, pp. 65–72.

[19] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[20] C.-Y. Wu and H.-Y. Lin, "Autonomous mobile robot exploration in unknown indoor environments based on rapidly-exploring random tree," in *2019 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2019, pp. 1345–1350.

[21] J. A. Placed and J. A. Castellanos, "A general relationship between optimality criteria and connectivity indices for active graph-slam," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 816–823, 2022.

[22] J. A. Placed, J. J. G. Rodríguez, J. D. Tardós, and J. A. Castellanos, "Explorb-slam: Active visual slam exploiting the pose-graph topology," in *Iberian Robotics conference*. Springer, 2022, pp. 199–210.

[23] H. Lehner, M. J. Schuster, T. Bodenmüller, and S. Kriegel, "Exploration with active loop closing: A trade-off between exploration efficiency and map quality," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6191–6198.

[24] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 661–666.

[25] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active slam," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2080–2087.

[26] Y.-B. Jia, "Recursive least squares estimation," https://faculty.sites.iastate.edu/jia/files/inline-files/recursive-least-squares.pdf, 2019, accessed: 08/13/2022.

[27] J. Kiefer, "General equivalence theory for optimum designs (approximate theory)," *The annals of Statistics*, pp. 849–879, 1974.

[28] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.