Landmark-based Distributed Topological Mapping and Navigation in GPS-denied Urban Environments Using Teams of Low-cost Robots

Mohammad Saleh Teymouri Subhrajit Bhattacharya∗

March 8, 2021

Abstract

In this paper, we address the problem of autonomous multi-robot mapping, exploration and navigation in unknown, GPS-denied indoor or urban environments using a swarm of robots equipped with directional sensors with limited sensing capabilities and limited computational resources. The robots have no a priori knowledge of the environment and need to rapidly explore and construct a map in a distributed manner using existing landmarks, the presence of which can be detected using onboard sensors, although little to no metric information (distance or bearing to the landmarks) is available. In order to correctly and effectively achieve this, the presence of a necessary density/distribution of landmarks is ensured by design of the urban/indoor environment. We thus address this problem in two phases: 1) During the design/construction of the urban/indoor environment we can ensure that sufficient landmarks are placed within the environment. To that end we develop a filtration-based approach for designing strategic placement of landmarks in an environment. 2) We develop a distributed algorithm using which a team of robots, with no a priori knowledge of the environment, can explore such an environment, construct a topological map requiring no metric/distance information, and use that map to navigate within the environment. This is achieved using a topological representation of the environment (called a Landmark Complex), instead of constructing a complete metric/pixel map. The representation is built by the robot as well as used by them for navigation through a balance between exploration and exploitation. We use tools from homology theory for identifying “holes” in the coverage/exploration of the unknown environment and hence guiding the robots towards achieving a complete exploration and mapping of the environment. Our simulation results demonstrates the effectiveness of the proposed metric-free topological (simplicial complex) representation in achieving exploration, localization and navigation within the environment.

1 Introduction

1.1 Motivation

Consider an unknown, GPS-denied urban/indoor environment in which we send out a large, fast-moving swarm of resource-constrained robots with extremely limited sensing capabilities and limited communication bandwidth, and with no a priori knowledge of the environment. Precise range or bearing measurement to the landmarks in the environment may not be available (for example, in an urban environment such landmarks may be wireless routers or 5G antennae identified by their MAC ids, allowing simple wireless/5G receivers to detect only their presence, but not their precise range). For communication and distributed exploration and mapping, each robot can potentially transmit only tens of bytes of data at a time.

An urban/indoor environment can be constructed/designed to aid such teams of robot to effectively explore, map, localize and navigate in. This is relevant in context of search and rescue type operations in such environments where facilities are constructed to aid such operations when necessary. While the swarm

∗MS. Teymouri and S. Bhattacharya are with the Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA, 18015 USA e-mail: mot317@lehigh.edu, sub216@lehigh.edu (see https://www.lehigh.edu/ sub216/)
of robots itself may not have a blueprint of the environment, they can rely on the structured placement of reliable landmarks to aid with the process of exploration, mapping and navigation.

Without global localization or coordinate charts, our objective is to attain this in a metric-free and coordinate-free manner that does not rely on precise distance or bearing measurements, is fast, robust to errors, and does not require extensive filtering or post-processing in order to compensate for sensing and actuation noise. In this paper we propose the use of simplicial complexes as the metric-free, coordinate-free topological representations of the environment. In particular, the robot swarm constructs an abstract simplicial complex representation, known as the landmark complex, using the landmarks detected by them as they navigate through the environment. This is a low-fidelity but correct (homotopically equivalent) topological representation of the free space (under appropriate assumptions on the density of landmarks and the coverage attained by the swarm).

1.2 Related Works

While GPS is broadly available to users around the world for localization and navigation, and so are maps of urban environments, a reliance on such information is not practical in many contexts. For example, GPS or maps may not be reliably available inside buildings with thick concrete walls. Since they require complex infrastructure to operate, GPS or a global map database may not be available for underground or extraterrestrial (such as future Lunar or Martian) colonies.

Most state-of-the-art methods for construction of maps of unknown environments without localization fall under the SLAM literature and require precise metric information (such as range or bearing measurements), rely on relatively precise odometry measurements, and in order to build a complete map, require extensive post-processing for correcting accumulated errors [1, 3–5, 9, 20]. Such methods usually construct a grid-based coordinate representation, making the amount of information that need to be shared between robots extremely large, and a precise transformation between the different robots' grid maps difficult to compute [15, 26]. State-of-the-art visual odometry based localization, mapping and navigation require significant sensing capabilities (such as stereo or monocular cameras) [10, 11, 19, 21, 23] in order to determine relatively precise metric information about detected features and landmarks in the environment, and focuses on precise pose estimation of the robots. Various implementations of SLAM can be found in different industries nowadays such as self-driving autonomous vehicles [12, 18, 22, 27] and consumer robot vacuum cleaners [16]. These state-of-the-art methods need meticulous metric measurement tools like range measuring sensors, relatively precise odometry measurements, and expensive cameras.

We, on the other hand, use topological methods to address the problem where multiple robots with only on-board limited-range sensors can detect presence of landmarks within their respective sensing disks (the binary information of whether a landmark is present or not), but no distance measurements to the landmarks are used. We consider a multi-robot setup in which a large number of robots need to cooperatively build the topological representation of the environment. Without knowing its own location, nor the locations of other robots in the environment (globally or relative to itself), the robots only communicate to other robots the identity of the landmarks that they observe – an extremely small amount of data – which allows the swarm to build the map collectively and in a distributed fashion.

While our method does not intend to match or compete with the metric precision of high-fidelity state-of-the-art SLAM techniques, the strength of our method lies in the use of extremely low-fidelity and inexpensive sensing and computational capabilities that allow the robots to perform mapping, localization and navigation tasks without requiring such precision. We consider directional sensors on robots such that the binary information of existence of a landmark can be detected by the robot in its sensor footprint. This makes the configuration space in which the robots need to construct the simplicial complex representation (“landmark complex”) a subset of $SE(2)$. Although we consider constrained resources for robots, landmarks are assumed to be available in necessary density for the topological exploration and mapping of the environment. The robots however cannot measure the bearing or the distance to the landmarks. The only information a robot uses for local control, is whether a detected landmarks is to its left or to its right side.

While landmark complex [13, 24], and more generally, simplicial complexes [6, 7] have been used for topological representation of environments, most of the existing work in this field has been for robots with disk-shaped sensor footprints in a planar domain. Furthermore, current literature does not explicitly address the problem of planning navigation of robots using the landmark complex, either to gather information that
would allow construction of the complex itself or to execute tasks. In this paper we address these practical issues concerning the construction and exploitation of a landmark complex representation. We also design navigation algorithms for the robots constructing the landmark complex representation through exploration of the environment as well as for robots exploiting the landmark complex for goal-directed navigation.

1.3 Contribution and Organization

The main contributions of the paper can be broadly classified into three parts: 1) The development of a filtration-based algorithm for determining placement of landmarks in an urban/indoor environment during the design/construction of the environment; 2) Design a set of controllers that would allow a team of robots to perform exploration of the environment for constructing the Landmark complex representation; 3) Design a set of controllers that would allow the team of robots to exploit the partially or fully constructed Landmark complex to perform informed exploration or navigation within the environment.

The paper is organized as follows: In Section 2 we introduce the basic definitions of Simplicial Complex and Landmark Complex. In Section 3 we describe the filtration-based algorithm for strategic placement of landmarks in an urban/indoor environment during its design and construction. Section 4 provides a set of control algorithms for the robots in the swarm which they can use to explore an unknown environment for constructing the Landmark complex and exploit the (fully or partially) constructed Landmark complex for further exploration and navigation. In this section we also use tools from homology theory for detecting ‘holes’ in the exploration and hence filling them. In Section 5 we provide detailed simulation results and evaluation of the proposed algorithms.

2 Preliminaries

In this section we provide brief definitions of Simplicial Complex and Landmark Complex.

2.1 Simplicial Complexes and Simplices

**Definition 1** (An abstract Simplicial Complex [14]). An abstract simplicial complex, $\mathcal{K}$, constructed over a set $V$ (the vertex set) is a collection of sets $C_n$, $n = 0, 1, 2, \ldots$, such that:

\begin{enumerate}
  \item An element in $C_n$, $n \geq 0$ is a subset of $V$ and has cardinality $n+1$ (i.e. for all $\sigma \in C_n$, $\sigma \subseteq V, |\sigma| = n+1$. $\sigma$ is called a “$n$-simplex”), and,
  \item If $\sigma \in C_n$, $n \geq 1$, then $\sigma \setminus v \in C_{n-1}, \forall v \in \sigma$. Such a ($n$-1)-simplex, $\sigma \setminus v$, is called a “face” of the simplex $\sigma$. The simplical complex is the collection $\mathcal{K} = \{C_0, C_1, C_2, \ldots\}$.
\end{enumerate}

Moreover, in Figure 1 an example of simplicial complex is provided. While a graph constitutes of only two sets ($V$ and $E$), in this example a simplicial complex with three sets is presented. In this figure, $\mathcal{K} = \{C_0, C_1, C_2\}$, where $C_0$ is the set of all 0-simplices (vertex set), $C_1$ is the set of all 1-simplices (edge set) and $C_2$ is the set of all 2-simplices, representing a triangle that connects three vertices. In this example, the $C_0, C_1$ and $C_2$ are as following:

\begin{enumerate}
  \item $C_0 = V = \{v_1, v_2, \ldots, v_{12}\}$.
  \item $C_1 = E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \ldots, \{v_{11}, v_{12}\}\}$.
  \item $C_2 = \{\{v_1, v_2, v_3\}, \{v_2, v_3, v_9\}, \{v_2, v_9, v_{10}\}, \{v_8, v_{10}, v_{11}\}\}$.
\end{enumerate}

2.2 Landmark Complex

A Landmark Complex [13, 24], $\mathcal{X}$, is an abstract simplicial complex, composed of the set of landmarks observed by a robot navigating in its configuration space. Assuming the environment has sufficient landmarks (the precise criteria is described in Section 3), a robot’s task is to collect the information of the landmarks, and create an abstract simplicial complex which is called Landmark Complex. Every time a robot observes a $n$-tuple of landmarks, it inserts the corresponding $(n-1)$-simplex constituting of the observed landmarks in $C_{n-1}$ (along with all its faces and sub-faces in $C_i, i < n-1$). Furthermore, the information contained in the landmark complex can be interpreted to generate the map of the environment.
Figure 1: An abstract simplicial complex consists of simplices of different dimensions. In this figure, the simplicial complex on the right, constructed from 0-simplices, 1-simplices and 2-simplices, depicted individually on left.

Figure 2: On the left there is an environment with 11 landmarks depicted as stars, and the corresponding landmark complex constructed from 6 observations with omni-directional sensor is shown on the right.

Figure 2 illustrates an example of creating a landmark complex by 6 observations with omni-directional sensor and use it to generate the topological map of the environment. In this example, for the first observation point \( O_1 \), the robot observes the landmarks \( \{v_2, v_3, v_4\} \) which actually is a 2-simplex. But, whenever a 2-simplex is observed, the connecting lines will also be considered as a 1-simplex, so in this case there exists 1-simplex as following, \( C_1 = \{\{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\} \) and as the same way the vertices are forming 0-simplex as well. This means, every time that we have an observation forming an \( n \)-simplex the lower dimension simplices will be inserted into the landmark complex. So, after 6 observations the created landmark complex would be \( \mathcal{X} = \{C_0, C_1, C_2\} \) where the 0-simplex or the Vertices set is equal to \( C_0 = \{v_1, v_2, v_3, ..., v_{11}\} \), the Edges set or 1-simplex is \( C_1 = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, ..., \{v_{10}, v_{11}\}\} \) and the 2-simplex is \( C_2 = \{\{v_2, v_3, v_4\}, \{v_4, v_5, v_6\}, \{v_6, v_7, v_8\}, \{v_8, v_9, v_{10}\}, \{v_{11}, v_1, v_2\}\} \).

The Landmark Complex \( \mathcal{X} \) can be immersed on a plane to create a visual representation of the topological map as shown in Figure 2. As visible in the image of \( \mathcal{X} \), the landmark complex captures the obstacle in the environment in terms of a hole in the complex.

3 Landmark Placement Algorithm

The ability to plan and implement strategic landmark placement in the environment allows faster robot explorations and precise topological mapping of the environment. For instance, in situations like search and rescue missions, if a building is equipped with sufficient landmarks (beacons) during its construction, firefighters can use a swarm of distributed robots to explore the environment and locate survivors more effectively and safely. Moreover, in hazardous places where the environment is too harsh for humans to operate
3.1 Domain of Visibility

We define the workspace, \( W = \mathbb{R}^2 - O \), to be the set of all possible landmark locations (where \( O \) is the set of all obstacles in the environment) and the configuration space \( C \) which is the set of all robot’s configurations. We also define the visibility domain of a landmark, \( D_{p,S} \subset C \), to be the set of all robot configurations in which the landmark at location \( p \in W \) will be visible to a robot using sensor footprint \( S \subset W \). Figure 3a shows a situation where both \( W \) and \( C \) are subset of \( \mathbb{R}^2 \). Therefore, In this figure, the sensor footprint is in shape of a disk. However, in this paper, robots with directional sensors (sector of a disk) are being considered, and accordingly the configuration space \( C \) for is a subset of \( SE(2) \). Figure 3b shows the visibility domain of a landmark in \( SE(2) \). In this figure, the workspace \( W \) is still \( \mathbb{R}^2 \) representing the possible landmark’s positions.

3.2 Nerve Lemma and Visibility Condition

The overall objective of this section is to establish the visibility and density condition for the presenting Landmark Placement Algorithm (LPA). First we show that the Čech complex [14], \( \mathcal{H} \), constructed by the visibility domain of the landmarks will be homotopy equivalent to the Landmark Complex \( \mathcal{X} \) constructed during the robot explorations. According to the definition of the Čech complex, whenever there is a \( n \)-way intersection of visibility domains, there would be a corresponding \((n-1)\)-simplex in \( \mathcal{H} \). Furthermore, by definition, a visibility domain is a set of all configurations that a landmark is visible to a robot. Therefore, whenever there is a \((n-1)\)-simplex in \( \mathcal{H} \), it implies that if a robot have a configuration in the intersection of those visibility domains, then all of the \( n \) landmarks will be visible to the robot and there would be the corresponding simplex in \( \mathcal{X} \) too. Therefore, \( \mathcal{H} \cong \mathcal{X} \). In Figure 4, an example is shown for more elaboration.

Suppose \( \mathcal{P} = \{p_1, \ldots, p_n\} \) is the set of landmark positions, and \( \mathcal{M} = \{D_{p,S} \subset C \mid p \in \mathcal{P}\} \) is the set of landmark visibility domains. Using the Nerve Theorem from algebraic topology [14], if the open cover \( \mathcal{M} \) of a topological space \( \mathcal{C} \) is a good cover, then the nerve of the cover \( \mathcal{N}(\mathcal{R}) \) is homotopy equivalent to the topological space \( \mathcal{C} \) without missing any topological information of the space. By definition, an open cover \( \mathcal{M} \) is considered a good cover when not only every nonempty sets of \( \mathcal{M} \), but every two and three way intersections are contractible. In other words, the Čech Complex constructed by the visibility domain of the landmarks \( \mathcal{H} \) is homotopy equivalent to the configuration space \( \mathcal{C} \ (\mathcal{H} \cong \mathcal{C}) \) and previously it has been shown that \( \mathcal{H} \cong \mathcal{X} \) consequently \( \mathcal{X} \cong \mathcal{C} \). Therefore, during the landmark placement phase, there should be at least one landmark visible to the robot for every configuration in the configuration space to obtain the metric-free
3.3 Landmark Placement Algorithm using Filtration over Sensor Footprints

In this section an iterative algorithm is presented to place the landmarks such that the Čech Complex $\mathcal{H}$ covers the entire configuration space $\mathcal{C}$. To this end, we use a filtration over sensor footprints – if $S^s$ is the true sensor footprint of a robot’s sensor, we define $S^1 \supset S^2 \supset \ldots \supset S^s$ to be a sequence of sensor footprints such that $S = \{S^i \subset \mathcal{C} \mid 1 \leq i \leq s\}$, where $S^i$ is the sensor footprint to be used at $t^{th}$ iteration.

The overall algorithm is to place a set of new landmarks in the uncovered regions of the environment at the $t^{th}$ iteration, considering $S^t$ to be the sensor footprint at that iteration. Starting with a relatively large sensor footprint, $S^1$, we gradually decrease the size of the footprint until we attain $S^s$, while placing landmarks in the uncovered regions in every iteration (Figure 7 illustrates the idea for disk-shaped sensor footprints). More details of the algorithm are provided below.

Denote the position of the $i^{th}$ landmarks by $p_i$ and $P^t = \{p_i \in \mathcal{W} \mid 1 \leq i \leq n\}$ is the set of all landmark positions at iteration $t$, where $\mathcal{W}$ is the workspace. Furthermore, we define $D_{p_i, S^t} = \bigcup_{p \in P^t} D_{p, S^t}$ as the union of all landmarks’ visibility domains, such that $D_{p_i, S^t} \subset \mathcal{C}$. Suppose $U^t$ is the complement of the covered space $D_{p_i, S^t}$ at $t^{th}$ iteration where $U^t = \mathcal{C} - D_{p_i, S^t}$ and $U^t \subset \mathcal{C}$. At every iteration, the Landmark Placement Algorithm (LPA) detects $U^t$, $i = 1, 2, \ldots, m$, the $i^{th}$ connected component of the uncovered area $U^t$ at $t$ such that $U^t = \bigcup_{i=1}^m U^t_i$, and suppose $\bar{U}^t$ is the set of all connected components $\bar{U}^t = \{U^t_1, U^t_2, U^t_3, \ldots, U^t_m\}$. In order to cover $U^t_j$ for $j = 1, 2, \ldots, m$, LPA places a landmark at $p_j$ such that $D_{p_j, S^{t+1}}$ covers most of the $U^t_j$ and inserts $p_j$ into the $P^t$. Afterwards LPA will compute the new $U^t$ with newly placed landmarks and continues the same procedure until $U^t = \emptyset$. In the next iteration, LPA goes to the next sequence of the sensor footprints, $S^{t+1}$ such that $S^{t+1} \subset S^t$ and $D_{p_i, S^{t+1}} \subset D_{p_i, S^t}$. Since the net cover of $D_{p_i, S^{t+1}}$ is smaller than the cover domain with $S^t$, new connected components of uncovered area may open up in the configuration space $\mathcal{C}$. Therefore, new landmark positions need to be populated in $P^{t+1}$. We start with $P^{t+1} = P^t$ and then the LPA process restarts by identifying the connected components in $U^{t+1} = \mathcal{C} - D_{p_i, S^{t+1}}$.

At the very first iteration, the LPA starts with a relatively large visibility domain, $D_{p_1, S^1}$, placed at the center of the workspace $\mathcal{W}$ such that in an obstacle-free environment $D_{p_1, S^1} \cong \mathcal{C}$. In circumstances that the sensor footprint sequence is consisted of concentric disks ($\mathcal{W} = \mathcal{C} \subset \mathbb{R}^2$), $S^1$ is a big circle where its radius is larger than the diameter of the environment. During the filtration, the radius of $S^1$ gets decreased until the target sensor footprint $S^s$ is reached. On the other hand, in case of directional sensor footprints where $\mathcal{C} \subset SE(2)$, LPA starts with a large half disk sensor footprint, $S^1$, where its radius and the angle of the circular sector reduces during the filtration until it reaches $S^s$. Figure 5 shows a sequence of sensor footprints in $SE(2)$.

In Algorithm 1 the Landmark Placement Algorithm (LPA) is presented. LPA takes the sequence of sensor footprints $S$ as an input, along with the workspace $\mathcal{W}$ and configuration space $\mathcal{C}$. At $t^{th}$ iteration, LPA computes the uncovered area $U^t$ (Lines 3 and 4). While the uncovered area $U^t$ is not fully covered...
with the union of all landmarks visibility domains $D_{P^t,S^t}$, LPA detects the connected components $\bar{U}^t$ and places a landmark at $U^t_1$ (Lines 6 and 7) and inserts the placed landmark into the set of all landmarks at $t^{th}$ iteration $P^t$ (Line 8). Afterwards, the algorithm updates $U^t$ (Lines 9 and 10) and repeats this process until $U^t = \emptyset$.

For the case $C \subset \mathbb{R}^2$, we used the pixel based representation of the image processing package Open-CV to detect connected components of the uncovered area. Moreover, the Place_Landmark($S^t, U^t_1$) function, places a landmark at the centroid of the $U^t_1$ such that the visibility domain of the placed landmark $D_{P^t,S^t}$ covers $U^t_1$ as much as possible.

On the other hand, when $C \subset SE(2)$, LPA detects a pixel based representation of $\theta$-slice for the uncovered region $U^t$ and for each $\theta$-slice, it detects the connected components $\bar{U}^t$. In other words, the lines 6 through 10 in Algorithm 1 will be repeated for each $\theta$-slice when $C \subset SE(2)$. Moreover, the algorithm finds the centroid of the $\theta$-slice and the Place_Landmark($S^t, U^t_1$) function places a landmark on a fixed distance $\Delta S$, away from the computed centroid in the opposite direction of $\theta$. The reason of doing this, is to enable the visibility domain of the placed landmark $D_{P^t,S^t}$ to cover the $U^t_1$ to the maximum. In order to find the optimum value of $\Delta S$ for minimum number of landmarks placed in an environment, we tested LPA with different values of $\Delta S$ which the results are presented in Figure 6.

Figure 7, shows an example over different iterations of LPA to populate a simple environment with landmarks for $C \subset \mathbb{R}^2$. In this figure, images on each row represent one iteration on filtration over sensor footprint. Moreover, Figure 8 shows the same environment populated with landmarks using LPA where $C \subset SE(2)$. In Figure 9, results of performing the Landmark Placement Algorithm on two different complex environments are presented.
Figure 7: An example of LPA in $\mathbb{R}^2$ with omni-directional sensor footprints. Each row demonstrates populating the environment with landmarks for a fixed sensor footprint (from a chosen filtration over the sensor footprints) at a particular iteration. In the first row sensor footprint is $S^1$ and in the last row it is the target footprint $S^*$. Environments are shown where $C \subset SE(2)$. Figures 9b and 9d show the placed landmarks in each environment. Figures 9e and 9f depict the visual representation of Čech complex constructed by considering every $(x, y, \theta)$ in the configurations space.

4 Multi Robot Exploration and Navigation

In this section we describe algorithms for constructing the Landmark complex through exploration of the environment, as well as algorithms for exploiting the constructed Landmark complex for informed exploration and navigation within the environment.

4.1 Sensor Model

At any instance of time, a robot can detect a landmark, if it falls into the robot’s sensor footprint. However, the robot cannot measure the bearing or the distance to the landmarks. The only information a robot has, is whether the detected landmarks are to its left or to its right side. This is a model for two low cost sensors attached to each side of the robot, with each sensor being able to detect the presence of a landmark only
Algorithm 1 Landmark Placement Algorithm

Require:
  a. Workspace \( W = \mathbb{R}^2 - O \)
  b. Configuration Space \( C \)
  c. Sensor Footprint \( S = \{S_1, S_2, S_3, \ldots, S_s\} \)

Ensure:
  a. Set of all Landmarks’ Position \( \mathcal{P} \)

1. Initiate Set \( \mathcal{P}^0 = \{\} \)
2. for \( t = 1, 2, 3, \ldots, s \) do
3.   \( \mathcal{D}_{\mathcal{P}^t, S^t} = \bigcup_{p \in \mathcal{P}^t} \mathcal{D}_{p, S^t} \)
4.   \( \mathcal{U}^t = C - \mathcal{D}_{\mathcal{P}^t, S^t} \)
5.   while \( \mathcal{U}^t \neq \emptyset \) do
6.     \( \bar{\mathcal{U}}^t = \text{detect}\_\text{connected}\_\text{components}(\mathcal{U}^t) \)
7.     \( p = \text{Place}\_\text{Landmark}(S^t, \bar{\mathcal{U}}^t_1) \)
8.     \( \mathcal{P}^t \leftarrow \mathcal{P}^t \cup p \)
9.     \( \mathcal{D}_{\mathcal{P}^t, S^t} = \bigcup_{p \in \mathcal{P}^t} \mathcal{D}_{p, S^t} \)
10.    \( \mathcal{U}^t = C - \mathcal{D}_{\mathcal{P}^t, S^t} \)
11.   end while
12. end for
13. return \( \mathcal{P}^s \)

when it is present on the sensor’s side of the robot. No other range or bearing information is assumed to be available.

4.2 Landmark Observation

Assuming \( C \subset SE(2) \) (for directional sensors) and \( W \subset \mathbb{R}^2 \), at time \( t \) the location of \( i^{th} \) robot is denoted by \( r_i^t = (x_i, y_i, \theta_i) \) and \( R^t = \{r_i^t \in C \mid 1 \leq i \leq N\} \) is the set of all robots’ locations. At each time step, each robot will make an observation to create a simplex from the observed landmarks. Every time a robot observes a \( n \)-tuple of landmarks, it inserts the corresponding \( (n-1) \)-simplex constituting of the observed landmarks in \( C_{n-1} \) (along with all its faces and sub-faces in \( C_{i}, i < n - 1 \)). Algorithm 2 takes the robot’s location \( r_i = (x_i, y_i, \theta_i) \) as an input. Each time the \( i^{th} \) robot will detect landmarks falling into its sensor footprint and store them in the simplex set \( S \). The subroutine \( \text{Make}\_\text{Observation}(r_i) \) inserts not only the observed simplex \( S \), but also all of the faces and sub faces of \( S \) into the landmark complex \( K \) recursively. Moreover, the landmark complex \( K \) is a global variable stored on a centralized server.

4.3 Robot Short-Term-Trajectory (STT) Modeling

In this section, a non-holonomic model for generating robot short-term-trajectories is presented, which can be used for directing the robot towards a specific landmark in its domain of visibility. This short-term-
trajectory generation will be used as a low-level controller for the different modes of walk described in Section 4.4.

Since the location of landmarks and robots are unknown in the environment, our strategy is to generate short paths (short-term-trajectories) described by Dubins curves [8] for each robot in order to attain a short term control objective. In other words, we modeled the short-term-trajectories of the robots as arcs of circles with distinct radius $\rho$ and arc length $s$, tangent to the robot’s current orientation (See Figure 10). Given these two parameters, there would be two choices of Dubins curves. One make the robot to turn towards its left and the other one to its right. This binary choice is described by a two-state variable $\beta$ which can either be +1 (right turn) or -1 (left turn). Consequently, the short-term-trajectories are described by three variables, $(\rho, s, \beta)$.

It is notable that $\rho$ and $s$ are limited to upper bounds and lower bounds. $\rho$ can assume value in range of $[0, \infty)$ where $\rho = 0$ allowing rotations at the robot’s position and $\rho = \infty$ corresponding to a straight line. However for computational purposes, we assume a large finite $\rho_{\text{max}}$ as the upper bound on $\rho$. Moreover,

\begin{algorithm}
\caption{Make Observation($r_i$)} //Updates the Landmark Complex, $\mathcal{K}$, using observations.
\begin{algorithmic}[1]
\Require
\begin{itemize}
\item a. Robot’s location $r_i = (x_i, y_i, \theta_i)$
\item b. Landmark Complex $\mathcal{K}$ (Global Variable)
\end{itemize}
\State $\mathcal{S} = \text{Detect Landmarks($r_i$)}$ //Set of 0-simplices observed by $i^{th}$ robot at $(x_i, y_i, \theta_i)$
\State $\mathcal{K} = \text{Update Landmark Complex($\mathcal{S}$)}$ //Recursively insert simplex $\mathcal{S}$ and its faces to $\mathcal{K}$
\end{algorithmic}
\end{algorithm}
Figure 10: **Robot Short-Term Trajectory Modeling:** This example shows a robot following the trajectories modeled as Dubins curves to reach a goal point.

$s$ can assume value in range of $[0, \min\{s_{\text{max}}, \pi \rho\})$ where $s_{\text{max}}$ is a constant value. Choosing the upper bound from the minimum of $s_{\text{max}}$ and $\pi \rho$ will avoid arc lengths greater than half a circle. Hence, $\rho$ will have a value from $[0, \rho_{\text{max}})$ and $s$ from $[0, \min\{s_{\text{max}}, \pi \rho\})$.

We assume robots can detect obstacles when they are in contact with them. In such case, if robots collide with an obstacle while executing the short-term-trajectory, they will turn on their position to acquire a new orientation and resume exploring. Figure 10 shows an example of robot short-term-trajectory sequence modeled as Dubins curves. In this example, the position of the $i^{th}$ robot at the $t^{th}$ time step is denoted as $r_{t_i}^i$. At each time step, robot will choose a Dubins curve as the short-term-trajectory and follows the created path to reach the goal. In this case the followed path is $T = \{(\rho_1, s_1, \beta_1), (\rho_2, s_2, \beta_2), (\rho_3, s_3, \beta_3)\}$. In this figure the path $(\rho_1, s_1, \beta_1')$ also can be found which implies that with a same $\rho_1$ and $s_1$ there exists two paths. This shows the importance of $\beta$.

### 4.4 Modes of Walk

In this section we describe different modes of walk that the robots can assume depending on the desired balance between exploration of the unknown regions of the environment and exploitation of the partially-constructed Landmark complex. These include *random walk* which is the only feasible option when the robots know nothing about the environment or when the robots are venturing into completely unexplored sections of the environment. But with a partially-constructed Landmark complex the robots can exploit it to perform an *Informed Systematic Walk* for more efficient exploration of the environment at the frontiers to the unexplored regions. Finally, when the Landmark complex is mostly complete, we can use tools from homology theory to detect "holes" in complex (small islands of unexplored regions) and use the robots to perform targeted exploration of such regions in order to complete the Landmark complex using a *Homology Informed Walk*.

#### 4.4.1 Random Walk

Since at the beginning of the exploration the Landmark complex is empty and there is no reference for the robots to use as a guide (recall that the location of landmarks and robots are unknown), robots need to perform *Random Walk (RW)* to construct a partial Landmark complex in order to localize themselves with respect to the observed landmarks. In this section, a description of the Random Walk algorithm is provided.

To develop a Random Walk, all three variables for generating the robots’ short-term-trajectory $(\rho, s, \beta)$, are randomly chosen with $\rho$ and $s$ being selected from a uniform probability distribution. Therefore, $\rho$ will be sampled from $[0, \rho_{\text{max}})$ and $s$ from $[0, \min\{s_{\text{max}}, \pi \rho\})$. This will enable the robots to choose between a vast variety of paths at each time step. For instance Figure 11 shows some of these possible paths. In this figure, in time $t$, the $i^{th}$ robot is in $r_{t_i}^i$ and randomly picks one path which in here is $(\rho_3, s_3, \beta_3')$ and moves to $r_{t+1}^{i+1}$. We refer to a single step of Random Walk as the execution of single short-term-trajectories explained above.
Algorithm 3 \( r_i^* := RW_{\text{Observe}}(r_i) \)

Require:
\ a. Robot’s location \( r_i = (x_i, y_i, \theta_i) \)

Ensure:
\ a. Robot’s updated location \( r_i^* = (x_i^*, y_i^*, \theta_i^*) \)

1: \( [\rho, s, \beta] := \text{UniRandSample}(\rho_{\text{max}}, s_{\text{max}}) \)
2: Set \( J := \text{GeneratePath}(\rho, s, \beta) \)
3: for \( k \in (1, 2, \cdots |J|) \) do
4: \( r_i \leftarrow J[k] \quad // J[k] = (x_k^i, y_k^i, \theta_k^i) \in C \)
5: \( \text{Make Observation}(r_i) \)
6: end for
7: return \( r_i^* \)

Figure 11: An illustrative example of Random Walk algorithm. In this figure, different short-term-trajectories (STT) have been shown where \( STT_i \) is \((\rho_i, s_i, \beta_i)\) and \( STT_i' \) is \((\rho_i, s_i', \beta_i')\).

The pseudo code given in Algorithm 3, describes the Random Walk and Observation subroutine \( RW_{\text{Observe}}(r_i) \). The input to this function is the \( i^{th} \) robot’s location \( r_i \). It will move the robot to the new random location \( r_i^* \) while observing new simplices and updating the landmark complex \( \mathcal{K} \). In line 1, the variables \((\rho, s, \beta)\) are sampled randomly from a uniform probability distribution. Furthermore, the function \( \text{GeneratePath} \) returns a set of points \( J \) in the configuration space \( C \subset SE(2) \) (Line 2). The \( i^{th} \) robot’s location is updated with these configurations, while the robot makes an observation to update \( \mathcal{K} \) each time along the path (Lines 3 to 6).

4.4.2 Informed Systematic Walk

Once robots have created a partial Landmark complex, they can exploit that information to perform a more systematic form of walk which result in faster landmark complex construction through increased exploration at the boundaries/frontiers of the unexplored regions.

The key insight behind this mode of walk is to navigate the robots to the landmarks that are observed fewer times in comparison to other landmarks. Those landmarks correspond to regions that are more likely to have remained unexplored (frontier or boundary regions). We refer to these landmarks as frontier landmarks. In other words, frontier landmarks are expected to be the boundaries of the unexplored regions since they are observed fewer number of times. The Informed Systematic Walk (ISW) is designed to navigate the robots to the least observed landmarks. Moreover, ISW partitions a set of landmarks based on the number of robots and navigates each robot to the least observed landmark within its partition. In following subsections, the descriptions of individual components of ISW are given.

i. **Voronoi partitioning based landmark assignment**: Since there are multiple robots, we used graph search-based Voronoi partitioning to assign landmarks to the robots. In other words, ISW will partition the environment into cells centered around the robots and assign the least observed landmark in
Each cell to the corresponding robot.

We construct the graph \( G \) from the partial Landmark complex built by robots during the exploration, such that the vertex set \( V(G) \) is the 0-simplices and the edge set \( \mathcal{E}(G) \) is the 1-simplices of the constructed landmark complex. We use a Dijkstra type wave front propagation algorithm to construct Voronoi Partitioning on the graph \( G \) [2]. In order to do so, we initiate the open list with the vertices corresponding to the landmarks currently being observed by the robots. Moreover, the cost on every edge \( (C_G) \) is equal to 1 unit. The pseudo-code given in Algorithm 4, describes our Voronoi partitioning algorithm which returns the tessellation map, \( \tau \), of landmarks to the robots. In line 6 the open list is initiated with the landmarks observed by the \( i^{th} \) robot at current time step for every \( i = 1,2, \ldots, N \) and in line 7 these landmarks are assigned a partition identity of \( i \) which means the landmark is assigned to the \( i^{th} \) robot. Lines 11 through 21 corresponds to the main loop of the algorithm. With every iteration, the vertex with minimum \( g \)-score in \( Q \) (the unexpanded vertices), is expanded. Furthermore, the algorithm checks for improvement in \( g \)-score for the neighboring vertices of the expanded vertex (line 14 to 20). If the potential \( g \)-score of a neighbor is less than the current one, the algorithm updates the \( g \)-score of that vertex and when a vertex is updated with an improved \( g \)-score, the tessellation identity of that vertex also gets updated (line 18).

Figure 12 depicts Voronoi partitioning and shows the path that the robots need to take in order to reach the least observed landmark in the corresponding partition.

ii. Dijkstra’s search for shortest path in 1-skeleton: Once the tessellation \( \tau \) is computed, each robot find the landmark with least observation count within its own partition. We refer to this landmark as goal landmark. We construct the shortest path from the robots’ current location to the goal landmark using Dijkstra algorithm on the graph \( G \). The output of Dijkstra search algorithm is a sequence of landmarks that a robot need to navigate along. Suppose the sequence of landmarks (path landmarks) for \( i^{th} \) robot is \( L_i = \{S_i, l_1, l_2, \ldots, G_i\} \). Each robot will use the obtained path landmarks to reach its assigned goal landmark in its own partition. Algorithm 5 describes our \( ISW(R, i) \) subroutine. Inputs to this function are the set of robots location \( R \) and the identity of the robot running this function \( i \). It also uses the global variable \( K \) to construct the graph \( G \) (Line 1). In line 2, the algorithm computes the tessellation map \( \tau \) by using the \( Voronoi(R, G) \) subroutine, described in Algorithm 4. Afterwards, the least observed landmark (goal landmark) in the \( i^{th} \) robot partition will be identified \( i^* \) (Line 3), and by using Dijkstra algorithm on graph \( G \), the shortest distance \( (g score) \) of the \( i^{th} \) robot to the goal landmark \( i^* \) is computed (Line 4). Therefore, the outputs of \( ISW \) are the identity of the \( i^{th} \) robot’s goal landmark \( i^* \) and the shortest distance from the robot to its goal landmark \( g score \).

iii. Navigation: In order to navigate along the path landmarks \( L_i \), the \( i^{th} \) robot needs to generate short-term-trajectories that will take it from one landmark in the sequence into the next. Suppose the \( i^{th} \) robot observed \( l_k \) in the sequence. In order to navigate to \( l_{k+1} \), it first need to make sure \( l_{k+1} \) is within its sensor footprint. Based on our assumption on the sensor model, a robot cannot measure the bearings or the distance to the landmarks. However, it has the information on whether \( l_{k+1} \) is to its left or to its right side. Based on this information, robot will choose the proper \( \beta \) to create the appropriate short-term-trajectory to
Algorithm 4 Voronoi Partitioning Algorithm
\( \tau := \text{Voronoi}(R, G) \)

**Require:**
- a. Graph \( G \) (with vertex set \( V(G) \), edge set \( E(G) \subseteq V(G) \times V(G) \) and cost function \( C_G \))
- b. Set of Agents location:
  \[ R = \{ r_i \in \mathcal{C} \subset SE(2) \mid 1 \leq i \leq N \} \]

**Ensure:**
- a. The tessellation map \( \tau : V(G) \to \{ 1, 2, \cdots, N \} \)

1: Initiate \( g \): Set \( g(v) := \infty, \forall v \in V(G) \) //Shortest distances
2: Initiate \( \tau \): Set \( \tau(v) := -1, \forall v \in V(G) \) //Tessellation
3: Detect landmarks for \( i^{th} \) robot, \( \{ p_{i,1}, \cdots, p_{i,M_i} \} \), \( i = 1, 2, \cdots, N \). \( M_i \) is the number of landmarks observed by each robot
4: **for** \( i \in \{ 1, \cdots, N \} \) **do**
5:   **for** \( k \in \{ 1, \cdots, M_i \} \) **do**
6:     Set \( g(p_{i,k}) := 0 \)
7:     Set \( \tau(p_{i,k}) := i \)
8:   **end for**
9: **end for**
10: Set \( Q := V(G) \) //Set of unexpanded vertices
11: **while** \( Q \neq \emptyset \) **do**
12:   \( q := \arg \min_{q' \in Q} g(q') \) //Maintained by a heap data structure
13:   Set \( Q = Q - q \) //Remove \( q \) from \( Q \)
14:   **for each** \( \{ w \in N_G(q) \} \) **do**
15:     Set \( g'(q) := g(q) + C_G([q, w]) \)
16:     **if** \( g'(q) < g(w) \) **then**
17:         Set \( g(w) := g' \)
18:         Set \( \tau(w) := \tau(q) \)
19:     **end if**
20: **end for**
21: **end while**
22: **return** \( \tau \)

Algorithm 5 Informed Systematic Walk Algorithm
\([i^*, g_{\text{score}}] := ISW(R, i)\)

**Require:**
- a. Set of Agents location:
  \[ R = \{ r_i \in \mathcal{C} \subset SE(2) \mid 1 \leq i \leq N \} \]
- b. Identity of the robot \( i \)
- c. Landmark Complex \( \mathcal{K} \) (Global Variable)

**Ensure:**
- a. Identity of the goal landmark \( i^* \)
- b. Distance to the goal landmark \( g_{\text{score}} \)

1: Graph \( G := \text{Construct\_Graph}(\mathcal{K}) \)
2: The tessellation map \( \tau := \text{Voronoi}(R, G) \)
3: \( i^* := \text{Least\_Observed\_Landmark}(\tau, i) \)
4: \( g_{\text{score}} := \text{Search}(i^*, G) \)
5: **return** \([i^*, g_{\text{score}}]\)
navigate towards its goal landmark. $\beta = +1$ if $l_{k+1}$ is to the robot’s right side and $\beta = -1$ if $l_{k+1}$ is to its left side. However, we sample $\rho$ and $s$ randomly as before.

On the other hand, if $l_{k+1}$ is not visible to the $i^{th}$ robot, few steps of random walk will be taken in order to find $l_{k+1}$. In case that even after taking few steps of random walk, $l_{k+1}$ was not visible, a new Dijkstra search will be executed to generate new sequence of path landmarks to the goal.

Figure 13 illustrates an example of the navigation algorithm. In this example the $i^{th}$ robot is depicted in different time steps. Moreover, $L_i = \{S_i, l_1, l_2, l_3, l_4, ..., l_{11}, G_i\}$ is the sequence of path landmarks provided by the Dijkstra algorithm, that directs the robot from starting landmark $S_i$ towards the goal landmark $G_i$. At each time step, robot chooses $\beta$ according to the orientation of the nearest path landmark.

**Algorithm 6** $r_i^* := ESTT_{\text{Observe}}(r_i, l_k)$

**Require:**
- a. Robot’s location $r_i = (x_i, y_i, \theta_i)$
- b. Observed landmark $l_k$

**Ensure:**
- a. Robot’s updated location $r_i^* = (x_i^*, y_i^*, \theta_i^*)$

1: $[\rho, \ s] := \text{UniRand\_Sample}(\rho_{\text{max}}, \ s_{\text{max}})$
2: $\beta := \text{Left\_Right}(r_i, \ l_k)$
3: Set $\mathcal{J} := \text{Generate\_Path}(\rho, \ s, \ \beta)$
4: for $k \in (1, 2, \cdots, |\mathcal{J}|)$ do
5: \hspace{1cm} $r_i \leftarrow \mathcal{J}[k]$ \hspace{0.5cm} // $\mathcal{J}[k] = (x_i^k, y_i^k, \theta_i^k) \in C$
6: \hspace{1cm} $\text{Make\_Observation}(r_i)$
7: end for
8: return $r_i^*$

In Algorithm 6, an outline of Execute Short-Term Trajectory and Observation subroutine $ESTT_{\text{Observe}}(r_i, l_k)$ is presented. Similar to $RW_{\text{Observe}}(r_i)$ subroutine, this function also takes the the $i^{th}$ robot’s location $r_i$ and returns the updated location $r_i^*$ while observing new simplices and updating the landmark complex $\mathcal{K}$. As described in Algorithm 3, $RW_{\text{Observe}}(r_i)$ generates the short-term-trajectories completely randomly by sampling all $(\rho, \ s, \ \beta)$ variables from uniform probability distribution. However, in line 2 of $ESTT_{\text{Observe}}(r_i, l_k)$ subroutine, the Left\_Right function takes the observed landmark $l_k$ and the $i^{th}$ robot’s location $r_i$ as inputs and checks whether $l_k$ is to the left or right side of the $i^{th}$ robot and return $\beta = +1$ if $l_k$ is to the robot’s right and $\beta = -1$ if it is to the left. Nevertheless, $\rho$ and $s$ are still randomly chosen (Line 1). Afterwards, similar to $RW_{\text{Observe}}(r_i)$, the set of points $\mathcal{J}$ in the configuration space $C \subset SE(2)$ is generated by taking the $(\rho, \ s, \ \beta)$ variables (Line 3). Ultimately, for each configuration in $\mathcal{J}$, the $i^{th}$ robot’s location is updated while making observations to update $\mathcal{K}$ (Lines 4 to 7).

In Algorithm 7, the pseudo-code for the $\text{Navigate}(r_i, i^*)$ subroutine is presented. Inputs of this function are the position of the $i^{th}$ robot and the identity of the goal landmark $i^*$. In line 5, the function $\text{Shortest\_Path}$
Algorithm 7 $r^*_i := \text{Navigate}(r_i, i^*)$

**Require:**

a. Robot’s location $r_i = (x_i, y_i, \theta_i)$  
   b. Identity of the goal landmark $i^*$  
   c. Landmark Complex $\mathcal{K}$ (Global Variable)

**Ensure:**

a. Robot’s updated location $r^*_i = (x^*_i, y^*_i, \theta^*_i)$

1: Initiate Bool $\text{successful} := $ false
2: while $(\text{successful})$ do
3:   Initiate Bool $\text{executed}_{rw} := $ false
4:   Graph $G := \text{Construct\_Graph}(\mathcal{K})$
5:   Set $L_i := \text{Shortest\_Path}(i^*, G)$  //
6:   while $L_i \neq \emptyset$ do
7:     Initiate Bool $\text{visible} := $ false
8:     for $j \in ([L_i], \cdots, 1)$ do  //
9:       if $L_i[j]$ is visible to $r_i$ then
10:          $r_i \leftarrow \text{ESTT\_Observe}(r_i, L_i[j])$
11:          for $k \in \{1, 2, \cdots, j\}$ do
12:            Set $L_i := L_i - L_i[k]$
13:          end for
14:          $\text{visible} := $ true
15:          break
16:     end if
17:   end for
18:   if $(\neg\text{visible})$ then
19:     if $(\text{executed}_{rw})$ then
20:       break
21:     else
22:       for $q \in \{1, 2, \cdots, \sigma\}$ do
23:         $r_i \leftarrow \text{RW\_Observe}(r_i)$
24:       end for
25:       Bool $\text{executed}_{rw} := $ true
26:     end if
27:   end if
28: end while
29: if $L_i = \emptyset$ then
30:   Bool $\text{successful} := $ true
31: end if
32: end while
33: return $r^*_i$
takes the graph $G$ and goal landmark’s identity $i^*$ as inputs and returns the set of path landmarks $L_i$ for the $i^{th}$ robot to the goal landmark. In lines 8 through 18 of the algorithm, $i^{th}$ robot tries to find the furthest landmark in path landmarks set $L_i$ in order to make a shortcut if there is any available (Line 8). Once the furthest visible landmark in $L_i$ is identified ($j^{th}$ path landmark $L_i[j]$), the algorithm executes a short-term-trajectory based on the orientation of the $j^{th}$ landmark by using the $ESTT \_\text{Observe}(r_i, L_i[j])$ subroutine described in Algorithm 6 (Line 10). This function moves the $i^{th}$ robot to the next location while making observations along the path to update landmark complex $K$. Afterwards, all the path landmarks until $L_i[j]$ will be removed from $L_i$ (Lines 11 to 13). However, if non of the path landmarks in $L_i$ are visible to the $i^{th}$ robot, the algorithm will perform few steps ($\sigma$) of Random Walk (Lines 22 to 25) and checks for the visibility of the landmarks in $L_i$ one more time. Even if after the execution of Random Walk non of the remaining path landmarks are visible to the robot, a new Dijkstra search will be performed to generate new set of path landmarks $L_i$.

iv. Interleaving between ISW and RW: A noteworthy point in ISW is to not use it repeatedly. Since ISW always navigates the robots to the least observed landmarks, there might be landmarks with fewer observation count which may be far away from the current frontier and ISW tries to navigate the robots to them. Moreover, since ISW will only increase the landmark’s observation count by one unit, robots may need to visit the current landmark again soon after they observed the next landmark. In other words, after observing the least observed landmark, the next step of ISW will leave this obtained landmark for one that has been observed less. Therefore, the consecutive steps of ISW will cause robots jumping between different landmarks. On the other hand, since robots have directional sensors, few observations are needed to capture the simplices which are connected to that landmark. In order to capture all the landmark’s neighbors, robots need to approach to the corresponding landmark with different orientations.

Another issue caused by having consecutive ISW is that, the frontier of the simplicial complex could grow non-uniformly in one particular direction. Essentially, ISW tries to grow the boundaries and it always do that with the closest landmark. Consequently, after too many consecutive steps of ISW the boundary will have a horn-like shape as depicted in Figure 14. In other words, ISW is only useful for acquiring new frontiers.

Hence, to solve both of these issues, few steps of RW is needed to explore the adjacent landmarks to ensure they have been observed sufficient number of times before switching back to ISW.

4.4.3 Homology Informed Walk

Once the robots performed RW and ISW, there might still remain some holes in the constructed landmark complex coverage $K$ due to the insufficient number of observations. We call them false holes (holes in the Landmark complex due to insufficient exploration – See Figure 16a), as opposed to the holes generated in the landmark complex due to the presence of obstacles. In order to localize them and navigate the robots to these false holes, we used homology theory to detect 1-simplices that bound them (See Figure 16d). In the following subsections, we briefly describe how tools from homology theory can be used to identify these 1-simplices and navigate robots to them. Fore more details on these methods the reader can refer to [6,14].
i. **Boundary matrices and higher order Laplacian**: In order to use the homology theory, first we need to define the boundary matrices $B_1$ and $B_2$ to be $n \times m$ and $m \times p$ matrices, where $n$, $m$, and $p$ respectively are the number of 0-simplices, 1-simplices, and 2-simplices.

Consider the simplicial complex in Figure 15. To construct $B_1$ and $B_2$ we need to assign arbitrary orientation to the 1-simplices and 2-simplices as shown in the figure. The $(i,j)$th element of $B_1$ matrix, where $i$ is the identity of the vertex and $j$ is the identity of the 1-simplex, can only have 0, +1, and -1 values. It will be 0 when the $i$th vertex is not one of the ends of the $j$th 1-simplex, +1 when $j$th 1-simplex points towards the $i$th vertex, and -1 if $j$th 1-simplex points away the $i$th vertex. For instance, in the given example, the first column of $B_1$ which corresponds to $e_1$, will have +1 for the $(1,1)$th element, -1 for the $(2,1)$th element, and the rest will remain zero.

![Figure 15: A directed simplicial complex of dimension 2.](image)

Similarly, the $(j,k)$th element of $B_2$, where $j$ is the identity of the 1-simplex and $k$ is the identity of the 2-simplex, also can only have 0, +1, and -1 values. It will be 0 when the $j$th 1-simplex is not adjacent to the $k$th 2-simplex, +1 when they are adjacent and their orientations are aligned, and -1 if their orientation do not match. For instance the $(2,1)$th element of $B_2$ will be -1, whereas $(4,1)$th and $(6,1)$th elements of $B_2$ will have a +1 value while the rest is zero. $B_1$ and $B_2$ matrices of the simplicial complex given in Figure 15 are provided in the following.

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \\ 0 & 0 & e_1 \\ -1 & 0 & e_2 \\ 0 & 1 & e_3 \\ 1 & 1 & e_4 \\ 0 & -1 & e_5 \\ 1 & 0 & e_6 \\ 0 & 0 & e_7 \\ 0 & 0 & e_8 \end{bmatrix}$$

It can be shown that the product of $B_1$ and $B_2$ is always equal to 0. Furthermore, the 1st order Laplacian matrix $L_1$ is defined as following.

$$L_1 = B_1^T B_1 + B_2 B_2^T$$

(1)

We are interested in a specific element from the kernel of $L_1$ which corresponds to a set of 1-simplices forming tight cycles around the holes in the complex.

ii. **Laplacian dynamics and $\ell_1$-norm minimization**: In order to identify the 1-cycles that bound the false holes tightly, we used the results of [6]. In this work, the authors considered the combinatorial Laplacian flow

$$\dot{x}(t) = -L_1 x(t), x(0) \in \mathbb{R}^m$$

(2)
(a) As shown, there is a false hole in the simplicial complex

(b) The corresponding result after solving the Laplacian Dynamics

(c) Result after $\ell_1$-Norm Minimization

(d) Result after choosing vertices that bound the false hole

Figure 16: The Laplacian Dynamic and $\ell_1$-norm minimization can localize the holes in the simplicial complex

Starting with any $x(0) \in \mathbb{R}^m$, where $m$ is the number of 1-simplices in the landmark complex $K$, it can be shown that the Laplacian flow will converge to $\ker L_1$ where $x$ is a linear combination of 1-cycles that bounds the 1-dimensional holes in the complex (See Figure 16b). In order to get the tightest 1-cycle around a hole we need to solve the following $\ell_1$-norm optimization problem.

\[
\min_{z \in \mathbb{R}^p} \|x + B_2 z\|_1
\]  

(3)

To solve this problem, a subgradient method is used [25].

\[
z^{k+1} = z^k - \alpha_k B_2^T \text{sgn}(B_2 z^k + x)
\]  

(4)

The initial condition used in equation (4) is $z^0 = 0$, $z \in \mathbb{R}^p$ where $p$ is the number of 2-simplices in the landmark complex $K$. Moreover, $\alpha_k$ is the step size which by picking a small enough $\alpha_k$, the converged $z$ gets close to the optimal solution (See Figure 16c).

iii. HIW algorithm and navigation: The pseudo-code given in Algorithm 8, presents the outline of the HIW algorithm and navigation.

In line 3 and 4 of this pseudo-code, $x^*$ and $z^*$ respectively are the converged solutions of equations (2) and (4). In line 6 through 10, the algorithm checks the absolute value of every element in vector $y = x^* + B_2 z^*$ ($y \in \mathbb{R}^m$), and if it is greater than a computed threshold ($\zeta$), the identity of that 1-simplex will be inserted to $Q$. The higher the value of $|y[i]|$ for the $i^{th}$ 1-simplex, it is more likely to be adjacent to a hole. It is notable that $\zeta$ is computed using the standard deviation of $|y[i]|, i = 1, 2, \cdots, m$. Since a large number of elements in vector $y$ have values close to zero and only a small fraction have absolute values greater than zero (See Figure 16c), if we order the elements in vector $y$ by absolute values, we would be able to see a jump. In order to find the appropriate thresholding value at where the jump occurs, we use the standard deviation of vector $|y[i]|, i = 1, 2, \cdots, m$ to compute $\zeta$, independent from size of the vector $y$.

Since we selected the 1-simplices with highest absolute value in vector $y$, these edges constitute the tightest 1-cycle that bound holes in the landmark complex $K$ (See Figure 16d). Furthermore, since these 1-simplices constitute of connected components surrounding isolated holes, we need to identify each of them. Therefore, we construct graph $G'$ such that the vertex set $V(G')$ and the edge set $E(G')$ are the 0-simplices and 1-simplices in $Q$ (Line 11). Afterwards in line 12, the function $\text{Identify Connected Components}$ takes the graph $G'$ as an input and by using Dijkstra algorithm on $G'$, is able to identify these connected components.
**Algorithm 8 Homology_Informed_Walk(ℝ)**

**Require:**
- a. Landmark Complex \( K \) (Global Variable) with \( n \) number of 0-simplices, \( m \) number of 1-simplices, and \( p \) number of 2-simplices.
- b. Graph \( G \) (with vertex set \( \mathcal{V}(G) \), edge set \( \mathcal{E}(G) \subseteq \mathcal{V}(G) \times \mathcal{V}(G) \) and cost function \( C_G \))
- c. Set of Agents location: \( \mathcal{R} = \{ r_i \in \mathcal{C} \subset SE(2) | 1 \leq i \leq N \} \)

```plaintext
1: Initiate \( Q = \{ \} \) //Set of 1-simplices identities
2: Initiate \( C_c = \{ \} \) //Global variable of connected components’ identity
3: \( x^* := \text{Converged solution of } \dot{x}(t) = -\mathcal{L}_1 x(t) \)
4: \( z^* := \arg\min_{z \in \mathbb{R}^p} \|x + B_2 z\|_1 \)
5: Set \( y = x^* + B_2 z^* \)
6: for \( i \in \{1, 2, \cdots, m\} \) do //\( m \) is the number of 1-simplices in \( K \)
7:   if \( |y[i]| > \zeta \) then //The \( i^{th} \) element of vector \( y \)
8:     Set \( Q \leftarrow Q \cup i \)
9:   end if
10: end for
11: Graph \( G' := \text{Construct\_Graph}(Q) \)
12: Set \( C_c := \text{Identify\_Connected\_Components}(G') \) //By using Dijkstra algorithm on graph \( G' \)
13: for all \( i \in \{1, 2, \cdots, N\} \) on individual threads do
14:   while \( C_c \neq \emptyset \) do
15:     \( C_t := \text{Compute\_Cost\_Matrix}(\mathcal{R}, C_c, G) \) //Cost matrix of size robots \((N) \) times clusters \((M) \)
16:     \( j^* := \text{Hungarian\_Assignment}(C_t, i) \)
17:     for \( k \in \{1, 2, \cdots, w\} \) do //\( w \) is the number of 0-simplices in \( C_c[j^*] \)
18:       \( r_i \leftarrow \text{Navigate}(r_i, C_c[j^*][k]) \)
19:     end for
20:     Set \( C_c = C_c - C_c[j^*] \) //Remove \( C_c[j^*] \) from \( C_c \)
21:   end while
22: end for
```
Figure 17: Hungarian assignments with corresponding shortest paths to the false holes in the simplicial complex. In Figure (a) some of the false holes and the assigned robots with the shortest paths to each assignment are shown, and Figure (b) shows the corresponding Landmark complex (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in $\mathbb{R}^2$ just for the purpose of visualization).

The output of this function is the set $Cc$ where the $i^{th}$ element in the set itself is a set of 0-simplices corresponding to the $i^{th}$ connected component.

In lines 13 to 20, each robot will find its own assignment to a connected component and will navigate to explore and observe the 0-simplices in order to cover the holes. This is executed on an individual thread for each robot. In order to assign the connected components to the robots, we used the Hungarian Algorithm, since it is of a cubic complexity [17]. To use Hungarian Algorithm, we compute the cost matrix $Ct$ (line 15) which is of size $N \times M$, where $N$ is number of robots and $M$ is the number of connected components. The $(i,j)^{th}$ element of $Ct$ constitutes of the distance from the $i^{th}$ robot to the $j^{th}$ connected component and it is computed by running the Dijkstra algorithm on the graph $G$ constructed over the landmark complex $K$.

Furthermore, in line 16, the Hungarian Assignment function, takes the cost matrix $Ct$ and the identity of the robot running this thread as inputs, and returns the identity of the assigned connected component ($j^*$). Lines 17 to 19 of Algorithm 8, navigate the $i^{th}$ robot to each 0-simplices in $Cc[j^*]$ using the function Navigate($r_i$, $i^*$) described in Algorithm 7. By navigating each robot to these connected components, we explore the regions corresponding to the holes in the coverage, hence we are able to cover the false holes much faster than combined $RW$ and $ISW$. In Figure 17 the Hungarian assignment with the corresponding shortest path to the assigned connected component for each robot is depicted.

Results of the $HIW$ algorithm is presented in Figure 18. In Figure 18a, 4 robots have performed combined $RW$ and $ISW$ to construct the simplicial complex. Afterwards, robots switch to $HIW$ to detect the holes in the simplicial complex. We explain the switching strategy in detail in Section 5, where we experimentally discover the optimal criteria for switching. We assume that robots are able to detect whether a landmark is adjacent to an obstacle or not. Therefore by using this information robots are able to distinguish the false holes from the holes that are corresponding to the obstacles in the environment.

4.5 Landmark Complex Construction Algorithm (LCCA)

So far we have explained necessary concepts and subroutines for the overall Landmark Complex Construction Algorithm (LCCA). In this section we put these concepts together and develop Algorithm 9 for enabling a group of robots to create a Landmark complex representation of an environment through a balanced strategy of exploration and exploitation. At each time step, each robot will make an observation to create a simplex from the observed landmarks and to update the landmark complex $K$. Moreover, this algorithm runs for each robot on a separate thread (Line 1 through 21).

Since at the beginning of the exploration, the location of landmarks and robots are unknown, robots will perform a fixed number of steps ($\gamma$) of $RW$, to construct a partial landmark complex in order to localize
Figure 18: Detection and exploration of false holes using HIW (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in $\mathbb{R}^2$ just for the purpose of visualization)
themselves with respect to the landmarks (Lines 2 through 4). Once robots have created a sufficiently large landmark complex, they switch to the combined RW and ISW (Lines 6 through 20) until the growth rate of landmark complex $K$ become less than $\varepsilon_1$. Furthermore, RW takes the location of the $i^{th}$ robot running the thread as an input ($r_i$), where ISW will take the set of all robots’ location $\mathcal{R}$ and the identity of the $i^{th}$ robot to find the goal landmark and the distance to it for the $i^{th}$ robot. However, ISW also uses the established landmark complex $K$, which is a global variable, to construct graph $G$ for Voronoi partitioning algorithm, described in Algorithm 4, and to find the $g_{score}$ to the goal landmark.

In addition, LCCA interleaves between ISW and RW to avoid issues caused by too many consecutive steps of ISW. As described in Informed Systematic Walk subsection, these issues are, first, navigating the robots to the far frontier landmarks back and forth, and second, growing the landmark complex non-uniformly. To this end, in line 7, the algorithm checks whether the $i^{th}$ robot is at the boundary of the landmark complex or not, by checking the $g_{score}$ of the acquired goal landmark. Also it checks the number of which ISW has been performed consecutively. If the $g_{score}$ is less than a fixed number $\omega$, or if the counter on consecutive steps of ISW is greater than a fixed number $\eta$, LCCA will execute $\delta$ number of RW (Lines 15 through 19).

At the final stages of the exploration, LCCA will switch to Homology Informed Walk, explained in Algorithm 8. This will enable the robots to locate holes in the landmark complex $K$ and to cover the false holes much faster than ISW and RW. When the growth rate of the landmark complex $K$ becomes less than $\varepsilon_2$, where $\varepsilon_2 < \varepsilon_1$, the exploration will be stopped.

Algorithm 9 Landmark Complex Construction Algorithm

Require:
- a. Set of Agents location: $\mathcal{R} = \{r_i \in \mathcal{C} \subset SE(2) \mid 1 \leq i \leq N\}$

Ensure:
- a. Landmark Complex $K$

1: for $i \in \{1, 2, 3, \cdots, N\}$ on individual thread do
2:     for $j \in (1, 2, \cdots, \gamma)$ do
3:         $r_i \leftarrow RW\_Observe(r_i)$
4:     end for
5:     while rate of growth $K \leq \varepsilon_1$ do
6:         $[i^*, g_{score}] := ISW(\mathcal{R}, i)$
7:         if $g_{score} \leq \omega$ or $ISW\_counter \geq \eta$ then
8:             for $k \in (1, \cdots, \delta)$ do
9:                 $r_i \leftarrow RW\_Observe(r_i)$
10:            end for
11:         $ISW\_counter = 0$
12:     else
13:         $r_i \leftarrow Navigate(r_i, i^*)$
14:         $ISW\_counter++$
15:     end if
16: end while
17: while rate of growth $K \leq \varepsilon_2$ do
18:     Homology\_Informed\_Walk($\mathcal{R}$)
19: end while
20: return $K$

5 Results

Our proposed Landmark Placement Algorithm (LPA) described in Section 3 was implemented in C++. The algorithm allows us to design strategic placement of landmarks in an environment during its de-
(a) observations count: 3000.  (b) observations count: 7000  (c) observations count: 10000 (d) observations count: 20000
Beginning of the combined RW and ISW

(e) observations count: 30000  (f) observations count: 50000.  (g) observations count: 100000.  (h) observations count: 180000.
Final result after combined RW and ISW.

Figure 19: LCCA in first complex environment (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in $\mathbb{R}^2$ and superimpose that on top of a map of the actual environment just for the purpose of visualization).

Figure 19: LCCA in first complex environment (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in $\mathbb{R}^2$ and superimpose that on top of a map of the actual environment just for the purpose of visualization).

sign/construction in order to ensure that at least one landmark is visible to a robot sensor for every configuration in $\mathcal{C} \subset \mathbb{R}^2$ (for disk-shaped sensor footprints) or $\mathcal{C} \subset SE(2)$ (for directional sensors). In Figures 7 and 8, results for populating a simple environment with landmarks are provided for $\mathcal{C} \subset \mathbb{R}^2$ and $\mathcal{C} \subset SE(2)$ respectively. Moreover, Figure 9 shows the results of Landmark Placement Algorithm (LPA) on two different complex environments. These environments are populated with the landmarks where $\mathcal{C} \subset SE(2)$. In this figure, the Čech complexes of the visibility domain of the landmarks for these environments are also depicted.

We also implemented the Landmark Complex Construction Algorithm (LCCA) described in Section 4 using C++. Our implementation is a multi-threaded one, with each thread emulating a robot, and with limited inter-thread communication to construct and update the Landmark complex maintained in the cloud (a central server). We evaluated the proposed algorithm using two different complex environments referred to as the first complex environment and the second complex environment. Results of the Landmark Complex Construction Algorithm is presented in Figures 19 and 20 for the first and second complex environments respectively.

To evaluate the role of HIW in improving exploration of the environment, we executed the LCCA with varying amounts of HIW and recorded the number of observations required. As a reference/benchmark, we first counted all the 2-simplices in the Čech complex of the visibility domain of the landmarks. Then we ran LCCA to construct 98% of those 2-simplices in the simplicial complex. We ran 5 simulations for each of the complex environments, in which 4 robots performed combined RW and ISW to construct 98%, 97%, 95%, 93% and 91% of the simplices respectively, following which HIW was used to complete the
Figure 20: LCCA in second complex environment (note that the Landmark complex itself is an abstract simplical complex. We immerse it in $\mathbb{R}^2$ and superimpose that on top of a map of the actual environment just for the purpose of visualization).
HIW contribution in constructing 2-simplices

![Different HIW contributions on constructing the 2-simplices in the Čech complexes of the visibility domain of the landmarks, to reach the target amount for the first and second complex environments.](image)

remaining 0%, 1%, 3%, 5% and 7% respectively of the simplices to reach the target 98%. Figure 21 shows the results of this experiment. We can deduce that HIW improves the algorithm by reducing the total number of observations needed, to complete 98% of the total 2-simplices, however its effectiveness starts diminishing beyond a certain percentage. In fact Figure 21 suggests that the optimum level of HIW use is to construct the last 5% of total 2-simplices with HIW.

Since the total number of 2-simplices is unknown to the robots, we need a strategy for switching to HIW without using this information. A useful data that robots have, is the growth rate of the simplicial complex. We define the growth rate of the simplicial complex \( r \), equal to the number of the new 2-simplices added to the simplicial complex at each iteration divided by the total number of 2-simplices that are already existing in the simplicial complex. Figure 22 shows the growth rate of the simplicial complex with time. As the exploration continues, it decreases exponentially indicating that the number of new 2-simplices added to the simplicial complex decreases with time. We experimentally find the optimal value of \( r \) at which the switch to HIW results in completion of the exploration with the minimum number of observations. To this end, we ran hundreds of simulations for each environment and recorded the number of total observations, as well as the growth rate value that the HIW switch happened. The results of these simulations are given in Figure 23. The x-axis denotes the growth rate value at which the switch happened and the y-axis is the total number of observations that robots made to complete the exploration. It is noteworthy that switching on smaller \( r \) corresponds to smaller contribution of HIW in constructing the simplicial complex. While some HIW does help with completing exploration with fewer observation, too much of HIW increases the number of observations. This is expected since the purpose of HWI is to fill holes within the complex, and not complete exploration when there are unexplored frontiers. In fact, as evident from the plots in Figure 23, a growth rate of about 0.004 is the optimal value at which switching to HIW minimizes the total observation count.

References

[1] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037, 2008.

[2] Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Multi-robot coverage and exploration on riemannian manifolds with boundary. *International Journal of Robotics Research*, 33(1):113–137, January 2014. DOI: 10.1177/0278364913507324.
Figure 22: Growth rate of the simplicial complex ($r$).

Figure 23: The number of the observations required to complete the simplicial complex plotted against the growth rate value at which the switch to HIW is performed. As evident, a switch at the growth rate of about 0.004 gives an optimal performance in either of the environments.
[3] Jose A. Castellanos, JMM Montiel, José Neira, and Juan D. Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952, 1999.

[4] Howie Choset and Keiji Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, 2001.

[5] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[6] J. Derenick, V. Kumar, and A. Jadababaie. Towards simplicial coverage repair for mobile robot teams. In *2010 IEEE International Conference on Robotics and Automation*, pages 5472–5477, 2010.

[7] A. Dirafzoon and E. Lobaton. Topological mapping of unknown environments using an unlocalized robotic swarm. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 5545–51, 2013.

[8] L. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497, 1957.

[9] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[10] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-slam: Large-scale direct monocular slam. In David Fleet, Tomas Pajdla, Bern Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing.

[11] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE.

[12] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. IEEE, 2014.

[13] R. Ghrist, D. Lipsky, J. Derenick, and A. Speranzon. Topological landmark-based navigation and mapping. *University of Pennsylvania, Department of Mathematics, Tech. Rep.*, 8, 2012.

[14] A. Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2001.

[15] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[16] Will Knight. The roomba now sees and maps a home. *MIT Technology Review*, 2015.

[17] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[18] H. Lategahn, A. Geiger, and B. Kitt. Visual slam for autonomous ground vehicles. In *2011 IEEE International Conference on Robotics and Automation*, pages 1732–1737, 2011.

[19] Shuoyuan Liu, Peng Guo, Lihui Feng, and Aiying Yang. Accurate and robust monocular slam with omnidirectional cameras. *Sensors*, 19(20):4494, 2019.

[20] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2002.

[21] Rodrigo Munguia and Antoni Grau. Monocular slam for visual odometry. In *2007 IEEE International Symposium on Intelligent Signal Processing*, pages 1–6. IEEE, 2007.
[22] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[23] Meixiang Quan, Songhao Piao, Minglang Tan, and Shi-Sheng Huang. Tightly-coupled monocular visual-odometric slam using wheels and a mems gyroscope. *IEEE Access*, 7:97374–97389, 2019.

[24] Rattanachai Ramaithitima and Subhrajit Bhattacharya. Landmark-based exploration with swarm of resource constrained robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 21-25 2018.

[25] A. Tahbaz-Salehi and A. Jadbabaie. Distributed coverage verification in sensor networks without location information. *IEEE Transactions on Automatic Control*, 55(8):1837–1849, 2010.

[26] Sebastian Thrun and Yufeng Liu. Multi-robot slam with sparse extended information filers. *Robotics Research*, pages 254–266, 2005.

[27] Junqiao Zhao, Yewei Huang, Xudong He, Shaoming Zhang, Chen Ye, Tiantian Feng, and Lu Xiong. Visual semantic landmark-based robust mapping and localization for autonomous indoor parking. *Sensors*, 19(1), 2019.