




Landmark-based distributed topological mapping and navigation in GPS-denied urban environments using teams of low-cost robots

Mohammad Saleh Teymouri¹ · Subhrajit Bhattacharya¹ 

Accepted: 9 January 2024
© The Author(s) 2024

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 team 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 which a team of robots, with no a priori knowledge of the environment, can use to 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 balanced strategy involving exploration and exploitation. We use tools from homology theory for identifying “holes” in the coverage/exploration of the unknown environment and hence guide the robots towards achieving a complete exploration and mapping of the environment. Our simulation results demonstrate the effectiveness of the proposed metric-free topological (simplicial complex) representation in achieving exploration, localization and navigation within the environment.

Keywords Mapping · Exploration · Navigation · Multi-robot system · Low-cost · Low-fidelity sensors · Simplicial complex · Topological mapping

Extended author information available on the last page of the article

1 Introduction

1.1 Motivation

Consider an unknown, GPS-denied urban/indoor environment in which we send out a large, fast-moving team of resource-constrained robots with extremely limited sensing capabilities, with no odometry information, with 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 5 G antennae identified by their MAC ids, allowing simple wireless/5 G receivers to detect only their presence, but not their precise range. Another example of such landmarks are passive RFIDs that are low-cost, do not need external power and can be easily installed in an environment. For communication and distributed exploration and mapping, each robot can potentially transmit only tens of bytes of data at a time.

For a given onboard sensor model, it is necessary that the landmarks in the environment be present with sufficient density and in appropriate locations. An urban/indoor environment can be constructed/designed to aid such teams of robots to effectively explore, map, localize and navigate in. This is relevant in the context of search and rescue type operations in such environments where facilities are constructed to aid such operations when necessary. While the team 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. 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 team 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 (such as in nuclear power plants), having strategically placed landmarks in the environment allows robots to perform maintenance operations autonomously.

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 team 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 robot team).

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. In this section we present an overview of existing literature on mapping, localization & navigation in GPS-denied environments, existing simplicial complex based representations, as well as existing work on landmark placement for that purpose.

1.2.1 SLAM and related literature

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, 7, 10, 13, 18, 37]. 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 [26, 47]. State-of-the-art visual odometry based localization, mapping and navigation require significant sensing capabilities (such as stereo or monocular cameras) [19, 20, 34, 38, 41] in order to determine relatively precise metric information about detected features and landmarks in the environment, and focus on precise pose estimation of the robots. Various implementations of SLAM can be found in different industries nowadays such as self-driving autonomous vehicles [23, 32, 39, 50] and consumer robot vacuum cleaners [30]. These state-of-the-art methods need meticulous metric measurement tools like range measuring sensors, relatively precise odometry measurements, and expensive cameras.

Some existing SLAM literature also perform active exploration of the unknown environment for map-building using information-driven methods [5, 27, 48, 51]. SLAM, when combined with active mapping (where robots actively explore the environment to construct the map), is classified as ASLAM (Active SLAM), and existing work in this area also focus on constructing high-fidelity metric maps (often as occupancy grid representations), high-precision pose estimation (through computationally-expensive optimization processes), and uses active exploration (often frontier-based exploration) [35]. Pure active mapping (without the problem of simultaneous localization) is usually achieved using grid-based representations, and multiple robots share the exploration task using Voronoi partitions, and is often guided using entropy-based heuristics [3] or deep reinforcement learning based frontier exploration [29].

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 robot team to build the map collectively and in a distributed fashion. We also perform active exploration of the environment using landmark observation count as a means to guide the exploration, since in a topological representation as ours there is no metric embedding of the map for identifying map *frontiers*. This active exploration strategy also requires us to develop a metric-free navigation algorithm that allows

the robots to move within the partially-explored environment using only the landmarks as reference.

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 a 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.

Closely related to SLAM literature is an extensive literature on information-driven exploration. While this body of research does not necessarily assume high-fidelity sensing, they assume the availability of some type of global localization or assume a partially-known map. For example [4, 12, 40] use location/pose of the robots during exploration of an unknown environment for building a map. We, on the other hand, assume that no localization information is available to the robots.

1.2.2 Simplicial complexes for topological representation

While *landmark complex* [24, 43, 44], and more generally, simplicial complexes [14–16, 42] 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 and only marginally addresses the problem of planning navigation for robots for the construction of the landmark complex. For example, although [24] uses landmark complex as the topological representation of an environment, the authors do not explicitly address the problem of computing robot paths or finding strategies for exploring an environment using the robots. Robot trajectories in this work are predefined, and the paper focuses more on theoretical properties of the landmark complex. Although our prior work [44] does address the problem of exploration for constructing landmark complex, it uses a frontier-based exploration method that is computationally more expensive and requires more capable sensors (for example, robots needs to know the bearing to the landmarks detected within their sensor footprints). Furthermore, the sensor model in either of these works are disk-shaped, and hence the implementations were made with \mathbb{R}^2 as the robot configuration space. However, in our present work, we consider directional sensors in general, thus requiring us to consider $SE(2)$ as the robot configuration space. Although [44] does give some condition on landmark density requirements for disk-shaped sensor footprints, the paper does not explicitly address the problem of landmark placement in an environment to ensure that the landmark complex is homotopy equivalent to the environment.

In [15, 16], the authors use an *encounter complex* for the topological representation of an environment, where agents keep track of encounters with other agents in disk-shaped sensor footprints, instead of observing landmarks in the environment. This requires that a much larger number of agents be used for computing the representation. For example, [15] uses 250 agents in a simple polygonal/annular environment, while in our method we can achieve complete exploration of a complex indoor environment using about 4 robots.

Although an improved metric in [16] allowed exploration with tens of robots, the environments considered were very simple, with one or two square-shaped obstacles. Also, the encounter complex use time as an independent coordinate, and hence the complex is usually much larger in size for similar environments. Furthermore, no active exploration of an environment is performed for constructing the encounter complex—robots only perform random walk or wall-following. We, on the other hand, provide systematic algorithms for exploration in large, complex, indoor environments using robots with directional sensors. [42] uses noisy global position measurements of robots in a swarm performing random walk to construct topological representations of simple environments. In our work we do not rely on the availability of any global position measurement for the robots.

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.

1.2.3 Landmark placement

In this paper we design an explicit algorithm for placement of landmarks in a environment that would ensure certain density conditions and thus allow effective construction of the landmark complex for a given sensor footprint for the robots. While the problem of landmark placement in an environment has been studied in the past, many existing work assume robot sensor and motion models that are relevant to traditional SLAM-type algorithms. For example, in [8, 36], the authors consider robots with camera-like sensors and noisy odometer that can detect the pose of landmarks in the robots' local frame. With such capable sensors, the authors consider the problem of sparse landmark placement that guarantee bounds on location uncertainty. In a related setup, the problem of external sensor placement in a known environment is considered in [49] such that these sensors, which are capable of taking noisy measurement of the poses of robots in the environment, can be used for localization of the robots while minimizing uncertainty. In all these work, the underlying assumptions about the capabilities of the sensors that detect the landmarks (or robots) are high, and the other capabilities of the robot (such as odometer, onboard computational capability for filtering the sensed data) are assumed to be high. Furthermore, while many of these methods perform optimization to extremize some form of cost function, the guarantees that such methods give on localization are at best probabilistic.

In contrast, [22] uses a genetic algorithm approach to perform optimization of a cost function for RB beacon placement, where the evaluation of the cost function is performed through high-fidelity simulation of radio signal propagation in an environment. While the scalar cost function is developed to optimize the beacon placement, the optimization is not directly related to any guarantees and the GA based algorithm does not guarantee a global optimization of the cost function.

In context of localization using wi-fi network routers, [9] uses a more formal optimization approach using a least square method, the theoretical/algorithmic formulation of which assumes an Euclidean domain and isotropic (non-directional) landmark sensing. Along similar lines, minimization of expected localization error at one (or a finite number of) points in a (or a sequence of) convex domain(s), is achieved using a gradient descent of a highly nonlinear optimization objective [28]. Similarly, for mobile robots navigating along a fixed trajectory, landmark placement along the trajectory is achieved using an iterative approach in [2]. [33] considers a similar error-minimizing optimization process for

landmark placement for directional sensors in spatial domains, but in absence of obstacles. In most of these works, the ability of sensors to measure real-valued quantities (for example, range to landmarks) allow the formulation of the said optimization problems in convex or Euclidean domains.

In our setup, on the other hand, because robots have such limited sensing capabilities (binary detection of whether or not a landmark is visible in a directional sensor footprint) and no odometry information, the criteria for landmark placement is significantly more stringent. Furthermore, we explicitly consider the obstacles in the environment and the occlusion of the landmarks caused by such obstacles. To satisfy the required visibility-based condition (described in Sect. 3.1), we develop a filtration-based method for the landmark placement algorithm. This guarantees that at least one landmark is visible from every robot pose (and this guarantee is not probabilistic).

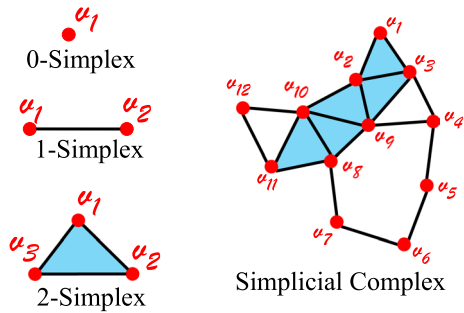
1.3 Contribution and organization

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 novelty of the current work lies in the fact that the sensing and onboard computation capabilities of the robots is extremely limited (a robot can only detect if a landmark is present in its directional sensor footprint, and can tell whether it is to its right or to its left). Our novel topological representation not only allows the collaborative construction of a topological map, but also allows robots to use it for navigation. While the problem of mapping, localization and navigation is solved more precisely using high-fidelity SLAM algorithms that use high-fidelity sensing and computationally-expensive filtering/post-processing, no existing methods is capable of achieving any meaningful mapping or localization with such limited sensing and computation capabilities as ours. Our novel filtration-based landmark placement algorithm is suitable for construction of the topological representation using such limited sensing abilities, and is applicable to indoor, non-convex environments.

The main technical advantage of having low-fidelity sensors on board robots and an algorithm that needs low-fidelity sensing information is that of *robustness*. An algorithm that uses low-fidelity sensors (as in our proposed algorithm) and does not require precise real-valued measurements (such as range or bearing) is more robust to errors. Real-valued measurements require constant filtering of the sensed data in order to get rid of the measurement noise. Furthermore, the high-fidelity, real-valued information is incorporated in mapping and exploration algorithms that construct precise metric maps, and hence such constructions are highly sensitive to measurement errors, which need to be eliminated through extensive post-processing. Also, communication of high-fidelity measurements/information between robots requires high-bandwidth communication channels and requires reasoning about changes in reference frame between the robots. The robustness of a topological algorithm and the ease of communicating the low-fidelity observations between robots allows fast deployment of a large number of robots for performing collaborative exploration and mapping.

Fig. 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



The paper is organized as follows: In Sect. 2 we introduce the basic definitions of *Simplicial Complex* and *Landmark Complex*. In Sect. 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 team which they can use to explore an unknown environment for constructing the Landmark complex and then 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 develop control algorithms for filling them. In Sect. 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

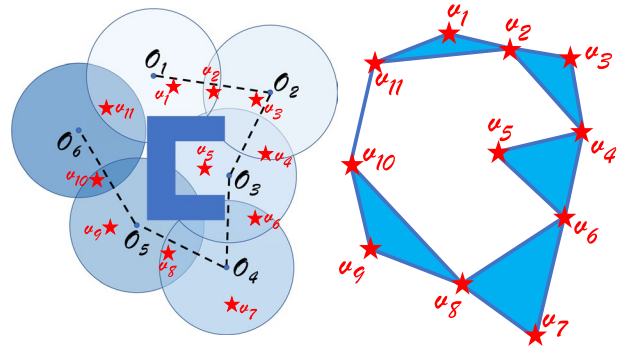
A *simplicial complex* can be thought of as a higher-dimensional extension of a graph, in which not only there are 0 and 1-dimensional entities (vertices and edges), but also 2 and potentially higher dimensional ones (called *simplices*).

Definition 1 (An abstract Simplicial Complex [25]) An abstract simplicial complex, \mathcal{K} , constructed over a set V (the *vertex set*) is a collection of sets C_n , $n = 0, 1, 2, \dots$, such that:

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

In Fig. 1 an example of a 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 the 0-simplices in the complex (*vertex set*), C_1 is the set of all the 1-simplices in the complex (*edge set*) and C_2 is the set of

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



all the 2-simplices in the complex (representing triangles that connects three vertices). In this example, the C_0 , C_1 and C_2 are as following:

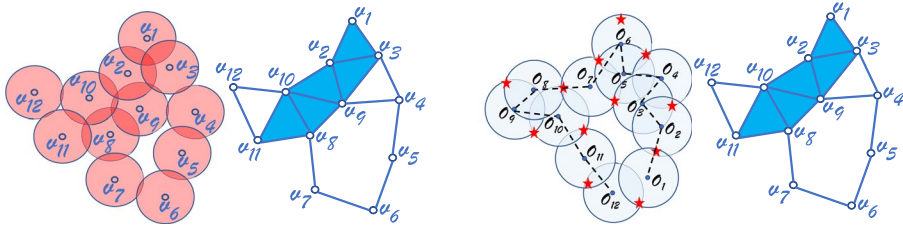
- i. $C_0 = V = \{v_1, v_2, \dots, v_{12}\}$.
- ii. $C_1 = E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \dots, \{v_{11}, v_{12}\}\}$.
- iii. $C_2 = \{\{v_1, v_2, v_3\}, \{v_2, v_3, v_9\}, \{v_2, v_9, v_{10}\}, \{v_8, v_{10}, v_{11}\}\}$.

2.2 Landmark complex

A Landmark Complex [24, 44], \mathcal{K} , 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 Sect. 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 an 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 $C_i, i < n - 1$). The information contained in the landmark complex can be interpreted to generate a topological map of the environment.

Figure 2 illustrates an example of construction of a landmark complex by 6 observations with an omni-directional sensor which serves as a topological map of the environment. In this example, for the first observation point O_1 , the robot observes the landmarks $\{v_1, v_2, v_{11}\}$ which corresponds to a 2-simplex in the Landmark Complex. But, whenever a 2-simplex is observed, the lines connecting the pairs of observations will also be considered as 1-simplices, so in this case there exists 1-simplices, $\{\{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\} \subset C_1$, and likewise the observed landmarks form 0-simplices 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, $\mathcal{K} = \{C_0, C_1, C_2\}$, is described by the set of 0-simplices, $C_0 = \{v_1, v_2, v_3, \dots, v_{11}\}$, the set of 1-simplices, $C_1 = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \dots, \{v_{10}, v_{11}\}\}$, and the set of 2-simplices, $C_2 = \{\{v_2, v_3, v_4\}, \{v_4, v_5, v_6\}, \{v_6, v_7, v_8\}, \{v_8, v_9, v_{10}\}, \{v_{11}, v_1, v_2\}\}$.

The Landmark Complex \mathcal{K} can be *immersed* on a plane to create a visual representation of the topological map as shown in Fig. 2. As visible in the image of \mathcal{K} , the landmark complex captures the obstacle in the environment in terms of a *hole* in the complex. The utility of the landmark complex is that it give an effective way of constructing a simplicial representation of the underlying configuration space that is topologically correct (i.e.



(a) Čech complex (or Nerve of the covering) of the domains of visibility of the landmarks (marked v_1 to v_{12}).

(b) Landmark complex constructed from 12 observations made along a robot trajectory (dashed curve). The landmarks are shown with red stars, while observation points are marked o_1 to o_{12} .

Fig. 3 Illustration of Čech complex versus Landmark Complex for 12 landmarks when the robot sensor footprints are omni-directions (and hence $\mathcal{C} \subseteq \mathbb{R}^2$). With sufficient observations, the landmark complex and the Čech complex are equivalent

topologically equivalent to the underlying configuration space) through a sequence of landmark observations only.

2.3 Čech complex or nerve of a covering

Consider a topological space, \mathcal{X} (this, in context of this paper will be a subset of the robot configuration space, \mathcal{C} , which in turn will be a subset of \mathbb{R}^2 or $SE(2)$ depending on whether the robots’ sensors are omni-directional or not—see Sect. 3.1), and a collection of its subsets $\mathcal{A} = \{\mathcal{A}_i\}_{i \in I}$ (where I is the set of indices) that cover \mathcal{X} . We define the Čech complex or the Nerve of the cover [25], $\mathcal{N}(\mathcal{A})$, to be an abstract simplicial complex with a 0-simplex corresponding to each of the subsets, and a $(n - 1)$ -simplex corresponding to every n -way overlap of the subsets. Figure 3a shows an example of a Nerve or Čech complex of disk-shaped subsets of \mathbb{R}^2 .

The main utility of the Čech complex or Nerve is its use in the Nerve theorem from algebraic topology [25], which states that if the open cover \mathcal{A} of the topological space \mathcal{X} is a good cover¹ then the Nerve of the cover is topologically equivalent (homotopy equivalent) to the covered space, \mathcal{X} , without missing any topological information of the space.

The application of the Nerve theorem in context of the landmark complex is discussed in more details in Sect. 3.2, where it is used to reason about the topology of the space covered by the domains of visibility of the landmarks. While the topology of the space \mathcal{X} may be difficult to compute without precise geometric embedding of the subsets in \mathcal{A} , the Čech complex or Nerve only requires local information about the overlaps of the subsets.

¹ By definition, an open cover, \mathcal{A} , is considered a good cover when not only every nonempty sets of \mathcal{A} are contractible, but every two and three way intersections of those sets are contractible.

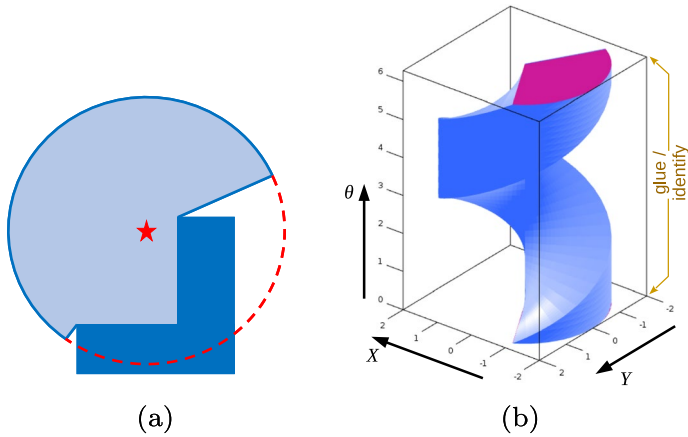


Fig. 4 Domain of Visibility: **a** The domain of visibility of a landmark when there is no directionality in the sensing, and thus $C \subseteq \mathbb{R}^2$. The L-shaped dark blue region is an obstacle and occludes line of sight. Hence, only the robot positions in the light-blue part of the disk-shaped region will be able to see the landmark (red star), and constitutes the domain of visibility for the landmark. **b** The domain of visibility of a landmark when the sensor footprint is directional (sector of a disk) and hence $C \subseteq SE(2)$

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. This is made possible in urban or indoor environments, which, during their design and construction, can be built with the planned placement of landmarks. In this section we describe an algorithm for strategically placing landmarks in an environment in order to attain such objectives.

3.1 Domain of visibility

We define the *workspace*, $\mathcal{W} = \mathbb{R}^2 - O$, to be the set of all possible landmark locations (where O is the set of all obstacles in the environment). We define the *configuration space*, \mathcal{C} , to be the set of all possible configurations of a robot. If robots do not have a directionality in their sensing (i.e., the sensor footprints are disk-shaped), then $\mathcal{C} = \mathcal{W} = \mathbb{R}^2 - O$. However, if the robots have directional sensors (e.g., sector of a disk), then the configuration of a robot is determined not only by its position, $(x, y) \in \mathbb{R}^2 - O$, but also its heading $\theta \in SO(2)$. In that case the configuration space is $\mathcal{C} = (\mathbb{R}^2 - O) \times SO(2) \subseteq SE(2)$ (where, $SE(2) = \{(x, y, \theta) \mid (x, y) \in \mathbb{R}^2, \theta \in SO(2)\}$, is the space of all position and orientations/headings of a directional robot/sensor on a plane).

We also define the *visibility domain* (or *domain of visibility*) of a landmark, $\mathcal{D}_{p,S} \subset \mathcal{C}$, to be the set of all robot configurations in which the landmark at location $p \in \mathcal{W}$ will be visible to a robot using sensor footprint $S \subset \mathcal{W}$. Figure 4a shows a situation where both \mathcal{W} and \mathcal{C} are subsets of \mathbb{R}^2 due to the robot sensor footprints being disk-shaped (i.e., omni-directional). However, in general, in this paper we will consider robots with directional sensors (with sensor footprints shaped as a sector of a disk), and accordingly the configuration

space $C \subseteq SE(2)$. Figure 4b shows the visibility domain of a landmark in $SE(2)$. In this figure, the workspace \mathcal{W} is still \mathbb{R}^2 representing the possible landmark's positions.

3.2 Visibility condition and landmark observation condition

Suppose $\mathcal{P} = \{p_1, \dots, p_n\}$ is the set of landmark positions, and $\mathcal{D} = \{\mathcal{D}_{p,S} \subset C \mid p \in \mathcal{P}\}$ is the set of landmark visibility domains. The following lemma establishes the density condition for landmark placement so that \mathcal{D} is a cover of C .

Lemma 1 *If at least one landmark is visible from every configuration in the configuration space, then \mathcal{D} is a cover of C .*

Proof If \mathcal{D} was not a cover of C , then by definition, there would be some point in C which will not be in any of the visibility domains in \mathcal{D} , and hence no landmark will be visible from that configuration. \square

The following proposition establishes the density condition for landmark observation so that the constructed landmark complex, \mathcal{K} is a correct representation of the configuration space, C .

Proposition 1 *If the visibility domains of the landmarks, $\mathcal{D} = \{\mathcal{D}_{p,S} \mid p \in \mathcal{P}\}$ constitutes a good cover of the configuration space, C , and if a landmark complex, \mathcal{K} , is constructed using sufficiently dense observation of landmarks (with at least one observation from $\cap_{p' \in Q} \mathcal{D}_{p',S}$ for every subset $Q \subseteq \mathcal{P}$ such that $\cap_{p' \in Q} \mathcal{D}_{p',S}$ is non-empty), then the the landmark complex and the configuration space are homotopy equivalent (i.e., $\mathcal{K} \cong C$).*

Proof The landmark complex constructed by taking an observation from every n -way non-empty overlap of the domains of visibility (i.e., an observation for every configuration sets of the form $\cap_{p' \in Q} \mathcal{D}_{p',S}$ whenever it is non-empty for some $Q \subseteq \mathcal{P}$), by definition, is equivalent to the Čech complex or Nerve of the covering \mathcal{D} (because, in the Landmark complex there is a $(n - 1)$ -simplex for every observation taken from an n -way overlap of the visibility domains from which n landmarks are visible, and in the Nerve/Čech complex there is a $(n - 1)$ -simplex corresponding to every n -way overlap of the open sets in \mathcal{D}). That is, $\mathcal{K} \cong \mathcal{N}(\mathcal{D})$ (where $\mathcal{N}(\mathcal{D})$ is the Čech complex or Nerve of \mathcal{D}).

Again, due to the *Nerve Theorem* we have $C \cong \mathcal{N}(\mathcal{D})$.

As a consequence, $\mathcal{K} \cong C$. \square

In Fig. 3, an example is shown for more elaboration.

Corollary 1 *If one or more robots perform random walk for a sufficiently long time in a configuration space, and if the configuration space is connected, then, with a probability greater than zero, they will eventually build a landmark complex, \mathcal{K} , that is homotopy equivalent to C .*

Proof Due to a well-known result on random walks [21, 45], random walks in spaces of dimension 4 or less are guaranteed to visit a point (get arbitrary close to the point) infinitely often with non-zero probability. So in configuration spaces which are subsets of \mathbb{R}^2 or $SE(2)$, the robot(s) will eventually visit at least one configuration from $\cap_{p' \in Q} \mathcal{D}_{p',S}$ (for

every subset $Q \subseteq \mathcal{P}$ such that $\cap_{p' \in Q} \mathcal{D}_{p', S}$ is non-empty) if they are allowed to perform the random walk for a sufficiently long time, and hence construct the landmark complex as described in Proposition 1. \square

3.3 Landmark placement algorithm using filtration over sensor footprints

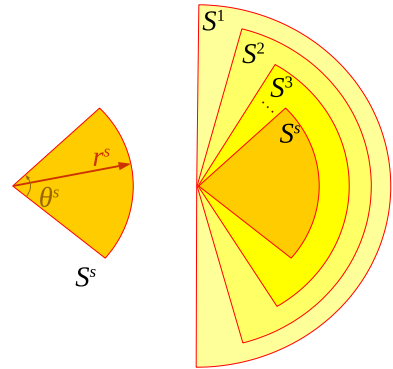
In this section an iterative algorithm is presented to place the landmarks such that the Nerve/Cech Complex $\mathcal{N}(\mathcal{D})$ covers the entire configuration space \mathcal{C} . To this end, we use a filtration over sensor footprints—if S^s is the true sensor footprint of a robot's sensor, we define $S^1 \supset S^2 \supset \dots \supset S^s$ to be a sequence of sensor footprints such that $S = \{S^t \subset \mathcal{C} \mid 1 \leq t \leq s\}$, where S^t 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 (Fig. 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 let $\mathcal{P}^t = \{p_i \in \mathcal{W} \mid 1 \leq i \leq n\}$ be the set of all landmark positions at iteration t , where \mathcal{W} is the workspace. Furthermore, we define $\mathcal{D}_{\mathcal{P}^t, S^t} = \bigcup_{p \in \mathcal{P}^t} \mathcal{D}_{p, S^t}$ as the union of all landmarks' visibility domains, such that $\mathcal{D}_{\mathcal{P}^t, S^t} \subset \mathcal{C}$. Suppose \mathcal{U}^t is the complement of the covered space $\mathcal{D}_{\mathcal{P}^t, S^t}$ at t th iteration where $\mathcal{U}^t = \mathcal{C} - \mathcal{D}_{\mathcal{P}^t, S^t}$ and $\mathcal{U}^t \subset \mathcal{C}$. At every iteration, the Landmark Placement Algorithm (LPA) detects U_j^t , $j = 1, 2, \dots, m$, the j th connected component of the uncovered area \mathcal{U}^t at t such that $\mathcal{U}^t = \bigcup_{j=1}^m U_j^t$. Suppose \bar{U}^t is the set of all the connected components $\bar{U}^t = \{U_1^t, U_2^t, U_3^t, \dots, U_m^t\}$. In order to cover U_j^t for $j = 1, 2, \dots, m$, LPA places a landmark at p_j such that \mathcal{D}_{p_j, S^t} covers most of the U_j^t and inserts p_j into the \mathcal{P}^t . Afterwards LPA will compute the new \mathcal{U}^t with newly placed landmarks and continues the same procedure until $\mathcal{U}^t = \emptyset$. 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 $\mathcal{D}_{\mathcal{P}^t, S^{t+1}} \subset \mathcal{D}_{\mathcal{P}^t, S^t}$. Since the net cover of $\mathcal{D}_{\mathcal{P}^t, S^{t+1}}$ 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 \mathcal{P}^{t+1} . We start with $\mathcal{P}^{t+1} = \mathcal{P}^t$ and then the LPA process restarts by identifying the connected components in $\mathcal{U}^{t+1} = \mathcal{C} - \mathcal{D}_{\mathcal{P}^{t+1}, S^{t+1}}$.

At the very first iteration, the LPA starts with a relatively large visibility domain, \mathcal{D}_{p_1, S^1} , placed at the center of the workspace \mathcal{W} such that in an obstacle-free environment $\mathcal{D}_{p_1, S^1} \cong \mathcal{C}$. In circumstances that the sensor footprint sequence consists 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 $SE(2)$.

Fig. 5 Sensor footprint sequence for directional sensors ($SE(2)$)



Algorithm 1 Landmark Placement Algorithm (LPA) // Centralized.

Require:

- a. Workspace $\mathcal{W} = \mathbb{R}^2 - O$
- b. Configuration Space \mathcal{C}
- c. Sensor Footprint $S = \{S^1, S^2, S^3, \dots, S^s\}$

Ensure:

- a. Set of all Landmarks' Position \mathcal{P}

```

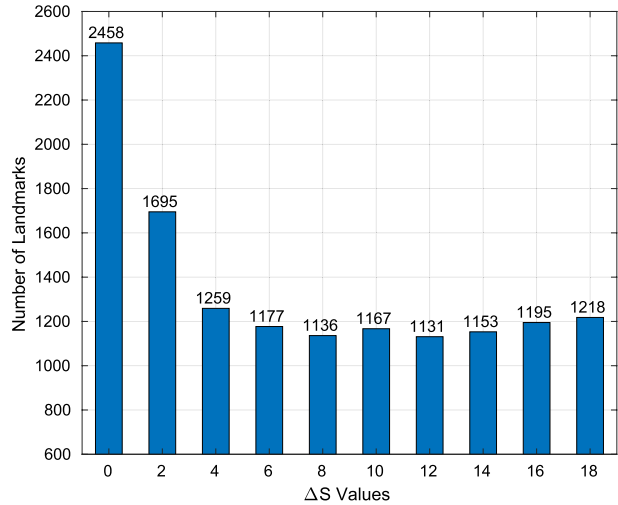
1: Initiate Set  $\mathcal{P}^0 = \{\}$ 
2: for  $t = 1, 2, 3, \dots, s$  do
3:    $\mathcal{P}^t = \mathcal{P}^{t-1}$ 
4:    $\mathcal{D}_{\mathcal{P}^t, S^t} = \bigcup_{p \in \mathcal{P}^t} \mathcal{D}_{p, S^t}$ 
5:    $\mathcal{U}^t = \mathcal{C} - \mathcal{D}_{\mathcal{P}^t, S^t}$ 
6:   while  $\mathcal{U}^t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

```

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 \mathcal{U}^t (Lines 4 and 5). While the uncovered area \mathcal{U}^t is not fully covered with the union of all landmarks visibility domains $\mathcal{D}_{\mathcal{P}^t, S^t}$, *LPA* detects the connected components $\bar{\mathcal{U}}^t$ and places a landmark at $\bar{\mathcal{U}}^t_1$ (Lines 7 and 8) and inserts the placed landmark into the set of all landmarks at t th iteration \mathcal{P}^t (Line 9). Afterwards, the algorithm updates \mathcal{U}^t (Lines 10 and 11) and repeats this process until $\mathcal{U}^t = \emptyset$.

Omni-directional Sensor Case: For the case $\mathcal{C} \subset \mathbb{R}^2$, we used the pixel based representation of the image processing package *Open-CV* to detect connected components of

Fig. 6 Number of landmarks for different ΔS values



the uncovered area. Moreover, the $Place_Landmark(S^t, U_1^t)$ function, places a landmark at the centroid of the U_1^t such that the visibility domain of the placed landmark \mathcal{D}_{p^t, S^t} covers U_1^t as much as possible.

Directional Sensor Case: On the other hand, when $\mathcal{C} \subset SE(2)$, LPA detects a pixel based representation of θ -slices for the uncovered region \mathcal{U}^t , and for each θ -slice, it detects the connected components \bar{U}^t . In other words, the lines 7 through 11 in Algorithm 1 will be repeated for each θ -slice when $\mathcal{C} \subset SE(2)$. Moreover, the algorithm finds the centroid of the θ -slice and the $Place_Landmark(S^t, U_1^t)$ function places a landmark on a fixed distance ΔS , away from the computed centroid in the opposite direction of θ . The reason of doing this, is to enable the visibility domain of the placed landmark \mathcal{D}_{p^t, S^t} to cover the U_1^t to the maximum. In order to find the optimum value of ΔS for minimum number of landmarks placed in an environment, we tested LPA with different values of ΔS which the results are presented in Fig. 6. In computing the uncovered space in the configuration space, \mathcal{U}^t (line 11 of Algorithm 1), for directional sensors, we also remove points that correspond to poses closer than a distance of ϵ from workspace boundaries (walls and obstacles). This is to ensure that certain pathological configurations (configurations in which a robot stands next to a workspace boundary and directly faces the boundary—see Fig. 8b) are not considered in order to avoid infinite density of landmarks at the boundary. In practice, for exploration using robots with directional sensors, this elimination of landmark visibility coverage very close to workspace boundaries is not an issue, since robots can perform short random walks whenever they are directly facing a workspace boundary and not observing any landmark.

Results from LPA: Fig. 7, shows an example over different iterations of LPA to populate a simple environment with landmarks for $\mathcal{C} \subset \mathbb{R}^2$. In this figure, images on each row represent one iteration on filtration over sensor footprint. Moreover, Fig. 8 shows the same environment populated with landmarks using LPA where $\mathcal{C} \subset SE(2)$. In Fig. 9, results of performing the Landmark Placement Algorithm on two different complex environments are shown where $\mathcal{C} \subset SE(2)$. Figure 9b, d show the placed landmarks in each environment. It's worth noting that with directional sensors ($\mathcal{C} \subset SE(2)$), a large majority of the landmarks are *boundary landmarks*—landmarks that are placed along the workspace boundaries or

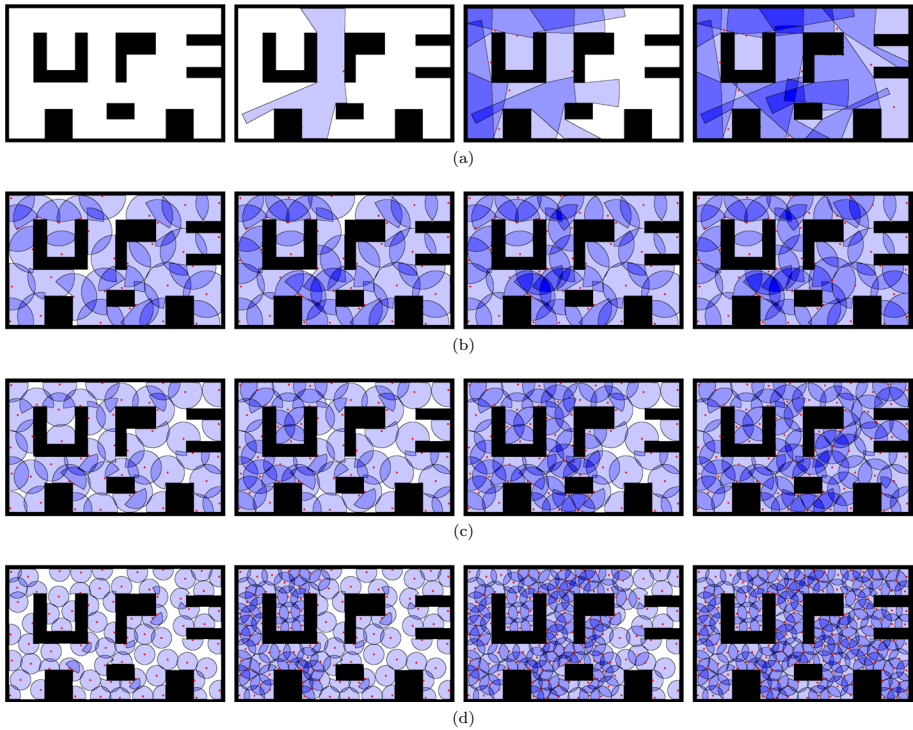
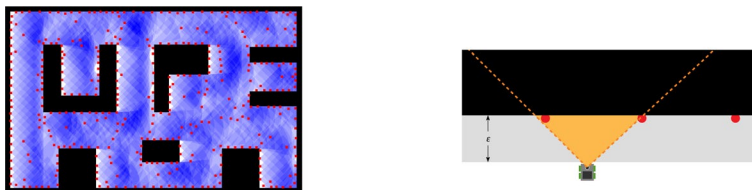


Fig. 7 An example of LPA in \mathbb{R}^2 with omni-directional sensor footprints (i.e., $\mathcal{C} \subseteq \mathbb{R}^2$). Each row demonstrates populating the environment with landmarks for a fixed sensor footprint (from a chosen filtration over the sensor footprints) at a particular iteration. In the first row sensor footprint is S^1 and in the last row it is the target footprint S^* . The final number of landmarks for this environment is 161



(a) Result of LPA with directional sensor footprint (sector of a disk) showing landmarks (red) and the domains of visibility (overlapping blue sectors) when $\theta = 0$. Note the higher density of landmarks along the obstacle and workspace boundaries. **(b)** A robot with directional sensor (orange footprint) facing a workspace boundary (black). In order for it to be able to see at least one landmark, even when it is not allowed to get closer than a distance of ϵ from the boundary (i.e., the grey region), there needs to be a denser placement of landmarks (red) at the boundary.

Fig. 8 a Landmark placement algorithm final result for a simple environment with directional sensors (i.e., $\mathcal{C} \subseteq SE(2)$). The number of landmarks placed is 378. However, note that most of these landmarks are *boundary landmarks* (about 270 landmarks) placed more densely along obstacle/workspace boundaries. **b** This is because, with directional sensors (footprints shaped like sectors of disks), there are pathological robot configurations—configurations in which a robot is very close to a boundary and facing the boundary—in which no landmarks will be visible unless there is a dense set of landmarks at the boundary

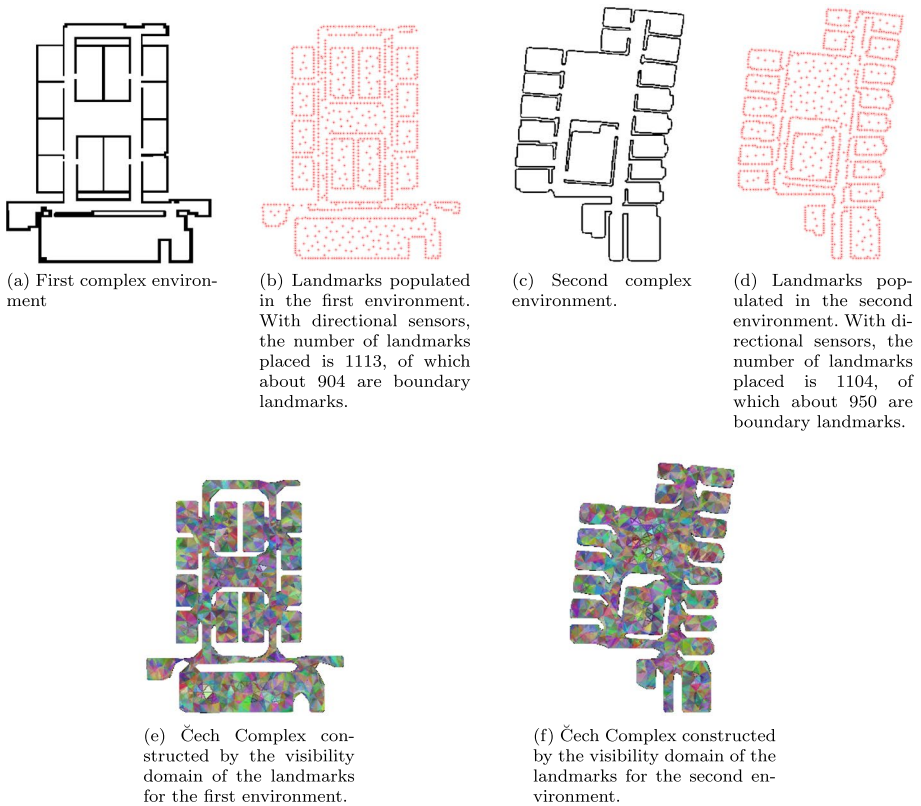


Fig. 9 Landmark placement algorithm results in complex indoor environments with directional sensors. **e**, **f** Visual representation of the corresponding Čech Complexes of the domains of visibility when overlaid/embedded on the map of the environment (note that the Čech Complex itself is an abstract simplicial complex and does not have a natural embedding—the figures show the embedding for visualization)

obstacle boundaries. This is because, with directional sensors (footprints shaped like sectors of disks), there are pathological robot configurations—configurations in which a robot is very close to a boundary and directly facing the boundary—in which no landmarks will be visible unless there is a dense set of landmarks at the boundary (Fig. 8b).

Figure 9e, f depict the visual representation of the complete Čech complex of the domains of visibility of all the landmarks, constructed by considering every (x, y, θ) robot positions in the configurations space $\mathcal{C} \subseteq SE(2)$.

Guarantees of the LPA: The landmark placement algorithm that we have presented is particularly meant to handle non-convex domains (with no visibility through obstacles), with directional sensor footprints that are not necessarily disk-shaped, and the sensing is a binary measurement rather than real valued (unlike, for example, range measurements). Hence, we cannot pose this problem as a practical optimization problem, especially since the environment is complex & highly non-convex, and even if it was possible to pose it as an optimization problem, such a formulation would involve a large number of integer variables (due to non-convexity of the environment, and the binary nature of the sensing). Many of the existing landmark placement algorithms in literature (a large number of which has been reviewed under Sect. 1.2.3) rely on convexity of the environment and/or real-valued

measurements (such as range) to pose the problem as an optimization problem. This naturally provides guarantees in terms of minimization of the optimization objective function. Our algorithm, on the other hand, is not optimization based. We instead use a filtration approach for landmark placement in complex nonconvex environments, with directional sensor footprints and binary measurement. Due to the complex nature of the environment and the highly restrictive sensor model, it is not possible to provide any meaningful closed-form, theoretical guarantee on the optimality of the landmark placement algorithm.

However, by construction, Algorithm 1 provides completeness guarantee:

Proposition 2 *Algorithm 1 terminates only when the necessary condition on landmark density (as described in Lemma 1) is satisfied. This termination happens in a finite time if the configuration space, \mathcal{C} , is finite-sized.*

Proof Algorithm 1, by construction, keeps placing landmarks until the condition for Lemma 1 is satisfied. At every placement of a landmark, the complement of the covered space (i.e., the uncovered space), $\mathcal{U} \subseteq \mathcal{C}$ is reduced by a finite amount (line 11 of Algorithm 1) since the new sensor is placed in a way such that its domain of visibility covers some previously uncovered parts of \mathcal{C} (procedure *Place_Landmark* in line 8 of Algorithm 1). Since \mathcal{C} is finite, the algorithm thus terminates in a finite number of iterations. \square

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. In our setup, individual robots make observations and communicate those observations to a centralized server that maintains and updates the landmark complex, \mathcal{K} . Direct computations involving the Landmark complex thus happen on the centralized server. All other computations (including making observation, navigation to a local landmark, and stochastic control) happens on the individual robots.

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

This sensor model is representative of sensors detecting passive RFID-based landmarks in an environment. RFIDs are low-cost, do not need external power and can be easily installed in an environment. The state-of-the-art RFID readers can achieve footprints of several meters [6]. The left and right information can be obtained using two partially

shielded RFID readers on each side of each robot that can read RFID tags on the left and right hemispheres around the robot respectively.

4.2 Landmark observation

Assuming $\mathcal{C} \subset SE(2)$ (for directional sensors) and $\mathcal{W} \subset \mathbb{R}^2$, at time t the location of i th robot is denoted by $r_i^t = (x_i, y_i, \theta_i)$ and $\mathcal{R}^t = \{r_i^t \in \mathcal{C} \mid 1 \leq i \leq N\}$ 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 an 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 $C_i, i < n - 1$). In Algorithm 2, given the i th robot's instantaneous location, $r_i = (x_i, y_i, \theta_i)$, the robot detects landmarks falling into its sensor footprint and stores them in the simplex set \mathcal{S} , which is computed on-board the robot based on the observations. \mathcal{S} is then communicated to the centralized server maintaining and updating \mathcal{K} . The subroutine *Update_Landmark_Complex*(r_i), running on the centralized server, inserts not only the observed simplex \mathcal{S} , but also all of the faces and sub faces of \mathcal{S} recursively into the landmark complex \mathcal{K} , maintained as a global variable stored on the centralized server.

Algorithm 2 *Make_Observation*(r_i) // Physical process of taking an observation. Centralized update of Landmark Complex, \mathcal{K} , using observations.

Require:

- a. Robot's location $r_i = (x_i, y_i, \theta_i)$ // Not used by robot. Only for performing the physical process of observation.
- b. Landmark Complex \mathcal{K} (Global Variable)

- 1: Set $\mathcal{S} = \text{Detect_Landmarks}(r_i)$ // **Physical Process.** Landmarks observed by i^{th} robot using onboard sensors.
 - 2: $\mathcal{K} = \text{Update_Landmark_Complex}(\mathcal{S})$ // **Centralized.** Recursively insert simplex \mathcal{S} and its faces into \mathcal{K} .
-

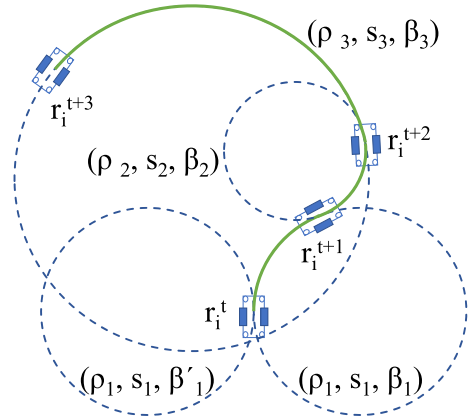
In Algorithm 2 it is important to note that the pose of the robot is not known to the robot. The pose of the robot determines the landmarks observed by the robot, and the procedure *Detect_Landmarks* represents that physical process, which uses the pose of the robot.

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 Sect. 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 [17] 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 ρ and arc length s , tangent to the robot's current orientation (See Fig. 10). Given these two parameters, there would be two choices of Dubins curves. One makes the robot turn towards its left and the other one to its right. This binary choice is described by a two-state variable β which can either be +1 (right turn) or -1 (left turn). Consequently, the short-term-trajectories are described by three variables, (ρ, s, β) .

Fig. 10 Robot short-term trajectory modeling: this example, shows a robot following the trajectories modeled as Dubins curves to reach a goal point



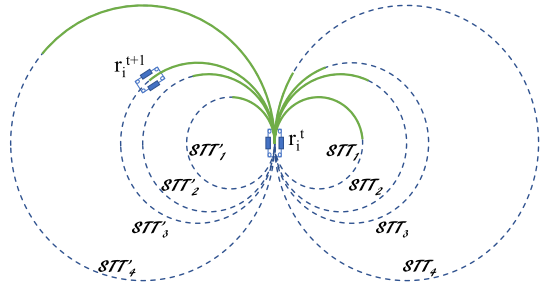
It is notable that ρ and s are limited to upper bounds and lower bounds. ρ can assume value in range of $[0, \infty)$ where $\rho = 0$ allowing rotations at the robot's position and $\rho = \infty$ corresponding to a straight line. However for computational purposes, we assume a large finite ρ_{max} as the upper bound on ρ . Moreover, s can assume value in range of $[0, \min\{s_{max}, \pi\rho\})$ where s_{max} is a constant value. Choosing the upper bound from the minimum of s_{max} and $\pi\rho$ will avoid arc lengths greater than half a circle. Hence, ρ will have a value from $[0, \rho_{max})$ and s from $[0, \min\{s_{max}, \pi\rho\})$.

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_i^t . At each time step, the 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 $\mathcal{T} = \{(\rho_1, s_1, \beta_1), (\rho_2, s_2, \beta_2), (\rho_3, s_3, \beta_3)\}$. In this figure the path (ρ_1, s_1, β_1) also can be found which implies that with a same ρ_1 and s_1 there exists two paths. This shows the importance of β .

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*.

Fig. 11 An illustrative example of Random Walk algorithm. In this figure, different short-term-trajectories (*STT*) have been shown where STT_i is (ρ_i, s_i, β_i) and STT'_i is (ρ_i, s'_i, β'_i)



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 (ρ, s, β) , are randomly chosen with ρ and s being selected from a uniform probability distribution. Therefore, ρ will be sampled from $[0, \rho_{max})$ and s from $[0, \min\{s_{max}, \pi\rho\})$. This will enable the robots to choose between a vast variety of paths at each time step. For instance Fig. 11 shows some of these possible paths. In this figure, in time t , the i th robot is in r_i^t and randomly picks one path which in here is (ρ_3, s_3, β_3) and moves to r_i^{t+1} . We refer to a single step of Random Walk as the execution of single short-term-trajectories explained above.

Algorithm 3 $r_i^* := RW_Observe(r_i)$ // **Decentralized**, on-board each robot.

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: $[\rho, s, \beta] := UniRand_Sample(\rho_{max}, s_{max})$

2: Set $\mathcal{J} := Generate_Path(\rho, s, \beta)$

3: **for** $k \in (1, 2, \dots, |\mathcal{J}|)$ **do**

4: $r_i \leftarrow \mathcal{J}[k]$ // **Physical process** of executing the planned short-term trajectory. $\mathcal{J}[k] = (x_i^k, y_i^k, \theta_i^k) \in \mathcal{C}$

5: $Make_Observation(r_i)$ // Observe landmarks and communicate observation to central server for updating \mathcal{K} .

6: **end for**

7: **return** r_i^*

The pseudo code in Algorithm 3, describes the *Random Walk and Observe* subroutine $RW_Observe(r_i)$. 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 (ρ, s, β) are sampled randomly from a uniform probability distribution. The function $Generate_Path$ constructs the corresponding short-term-trajectory, discretizes it, and returns a set of points, \mathcal{J} , that make up the path in the

configuration space $\mathcal{C} \subset SE(2)$ (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 3–6). The random sampling of the path parameters, generation of path, execution of path, and landmark observation happen onboard each robot. The observed landmarks are then communicated to the centralized server for adding the new simplices to the globally maintained Landmark Complex, \mathcal{K} (Algorithm 2).

4.4.2 Informed systematic walk

Once the robots have created a partial Landmark complex, they can exploit that information to perform a more systematic form of walk which results 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. Unlike [44], where a frontier in the landmark complex was detected using complex geometric inferences based on the bearing measurements to landmarks, with even more limited sensing & computation capabilities and with no bearing measurements, we use observation count as the means for identifying of potentially unexplored frontiers in the complex for guiding the exploration.

We construct the graph, G , from the partial Landmark complex, \mathcal{K} , built by robots during the exploration, such that the vertex set $\mathcal{V}(G)$ is the set of 0-simplices and the edge set $\mathcal{E}(G)$ is the set of 1-simplices of the constructed landmark complex. This graph is referred to as the 1-skeleton of \mathcal{K} . We use a Dijkstra type wave front propagation algorithm to construct Voronoi Partitioning on the graph G [3]. 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 chosen to be equal to 1 unit, since, without any metric/distance information, this is the simplest and most natural choice, and measures the number of “*landmark hops*” from the current location of the robot.

Algorithm 4 Voronoi Partitioning Algorithm $\tau := \text{Voronoi}(\mathcal{R}, G)$ // **Centralized.**

Require:

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

Ensure:

- a. The tessellation map $\tau: \mathcal{V}(G) \rightarrow \{1, 2, \dots, N\}$

```

1: Initiate  $g$ : Set  $g(v) := \infty, \forall v \in \mathcal{V}(G)$  //Shortest distances to vertices
2: Initiate  $\tau$ : Set  $\tau(v) := -1, \forall v \in \mathcal{V}(G)$  //Tessellation
3: Detect landmarks for  $i^{th}$  robot,  $\{p_{i,1}, \dots, p_{i,M_i}\}$ ,  $i = 1, 2, \dots, N$ . //  $N$  is the number of robots and  $M_i$  is the number
of landmarks observed by each robot
4: for  $i \in \{1, \dots, N\}$  do
5:   for  $k \in \{1, \dots, M_i\}$  do
6:     Set  $g(p_{i,k}) := 0$ 
7:     Set  $\tau(p_{i,k}) := i$ 
8:   end for
9: end for
10: Set  $Q := \mathcal{V}(G)$  //Set of unexpanded vertices
11: while  $Q \neq \emptyset$  do
12:    $q := \text{argmin}_{q' \in Q} g(q')$  //Maintained by a heap data structure
13:   Set  $Q = Q - q$  //Remove  $q$  from  $Q$ 
14:   for each  $\{w \in \mathcal{N}_G(q)\}$  do //For each neighbor of  $q$ 
15:     Set  $g' := g(q) + C_G([q, w])$  // Cost function,  $C_G([q, w])$ , is chosen to be uniform, 1, for all edges
16:     if  $g' < g(w)$  then
17:       Set  $g(w) := g'$ 
18:       Set  $\tau(w) := \tau(q)$ 
19:     end if
20:   end for
21: end while
22: return  $\tau$ 

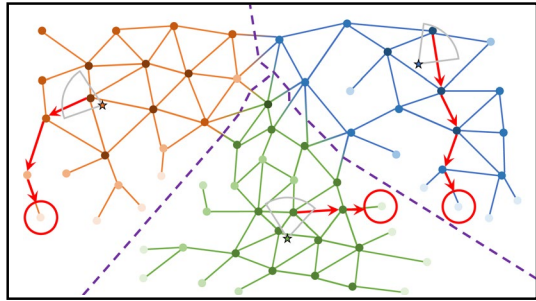
```

The pseudo-code given in Algorithm 4, describes our Voronoi partitioning algorithm which returns the tessellation map, τ , of landmarks to the robots and is reminiscent of the Dijkstra's search algorithm [3, 11]. In line 6 the *open list* is initiated with the landmarks observed by the i th robot at current time step for every $i = 1, 2, \dots, N$ 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 [11] 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 14–20). 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). This computation happens on the centralized server that maintains the Landmark Complex, \mathcal{K} , and hence the 1-skeleton of the complex, G .

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 τ is computed, each robot finds the landmark within its own partition with least observation count, but is no more than g_thresh landmark hops away from the current location in G (line 3 of Algorithm 5). 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 $L_i = \{S_i, l_1, l_2, \dots, G_i\}$. Each robot will use the obtained *path landmarks* to reach its assigned *goal landmark* in its own partition.

Fig. 12 Voronoi Partitioning around each robot (depicted as stars). Landmarks with darker colors imply that they have been observed more than landmarks with lighter colors. Landmarks that are observed fewer number of times form the boundary are the *ISW* target



Algorithm 5 Informed Systematic Walk Algorithm $i^* := ISW(\mathcal{R}, i, g_thresh)$ // **Centralized.**

Require:

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

- 1: Graph $G := Construct_Graph(\mathcal{K})$
- 2: The tessellation map $\tau := Voronoi(\mathcal{R}, G)$
- 3: $i^* := Least_Observed_Landmark(\tau, i, g_thresh)$
- 4: **return** [i^*, g_score]

Algorithm 5 describes our *ISW* subroutine that returns the identity of a *goal landmark* for navigating to. Inputs to this function are the set of robots location \mathcal{R} , the identity of the robot, i , running this function, and a parameter g_thresh . It also uses the global variable \mathcal{K} to construct the graph G (Line 1). In line 2, the algorithm computes the tessellation map τ by using the *Voronoi*(\mathcal{R}, G) subroutine, described in Algorithm 4. Subsequently, the least observed landmark (goal landmark), that is not farther than g_thresh landmark hops from the currently observed landmark, in the i th robot partition is identified as i^* (Line 3). Later on, this will be used as an input to the *Navigate* subroutine (Algorithm 7) to navigate the robot to the goal landmark when performing an instance of *ISW*. Since all these computations require the information about the graph G , which is the 1-skeleton of \mathcal{K} , this takes place on the centralized server. 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 β to create the appropriate short-term-trajectory to 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 ρ and s randomly as before.

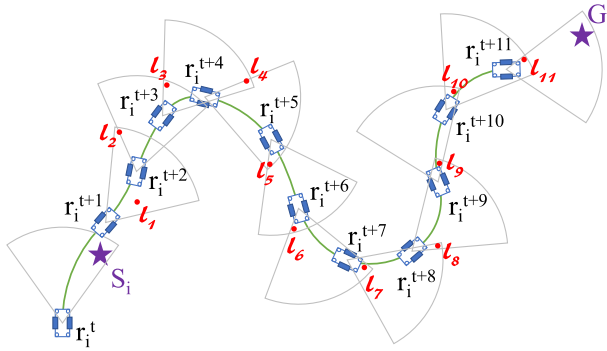


Fig. 13 Navigation algorithm: the i th robot uses the path landmarks L_i to navigate from S_i to G_i

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 how the navigation algorithm works. In this example, the i th robot is depicted at different time steps. Moreover, $L_i = \{S_i, l_1, l_2, l_3, l_4, \dots, l_{11}, G_i\}$ 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 β according to the orientation of the nearest path landmark.

Algorithm 6 $r_i^* := ESTT_Observe(r_i, l_k)$ // **Decentralized**, on-board each robot.

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:  $[\rho, s] := UniRand\_Sample(\rho_{max}, s_{max})$ 
2:  $\beta := Left\_Right(r_i, l_k)$ 
3: Set  $\mathcal{J} := Generate\_Path(\rho, s, \beta)$ 
4: for  $k \in (1, 2, \dots, |\mathcal{J}|)$  do
5:    $r_i \leftarrow \mathcal{J}[k]$  // Physical process of executing the planned short-term trajectory.  $\mathcal{J}[k] = (x_i^k, y_i^k, \theta_i^k) \in \mathcal{C}$ 
6:    $Make\_Observation(r_i)$  // Observe landmarks and communicate observation to central server for updating  $\mathcal{K}$ .
7: end for
8: return  $r_i^*$ 

```

In Algorithm 6, an outline of Execute Short-Term Trajectory and Observation sub-routine $ESTT_Observe(r_i, l_k)$ is presented. Similar to $RW_Observe(r_i)$ 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, $RW_Observe(r_i)$ generates the short-term-trajectories completely randomly by sampling all (ρ, s, β) variables from uniform probability distribution. However, in line 2 of $ESTT_Observe(r_i, l_k)$ 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, ρ and s are still randomly chosen (Line 1). Afterwards, similar to

$RW_Observe(r_i)$, the set of points \mathcal{J} in the configuration space $\mathcal{C} \subset SE(2)$ is generated by taking the (ρ, s, β) 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 4–7). Like in Algorithm 3, the computation of the path parameters, generation of path, execution of path, and landmark observation happen onboard each robot. The observed landmarks are then communicated to the centralized server for adding the new simplices to the globally maintained Landmark Complex, \mathcal{K} (Algorithm 2).

Algorithm 7 $r_i^* := Navigate(r_i, i^*)$ // **Mostly decentralized**, on-board i th robot.

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 $r_i^* = (x_i^*, y_i^*, \theta_i^*)$
-

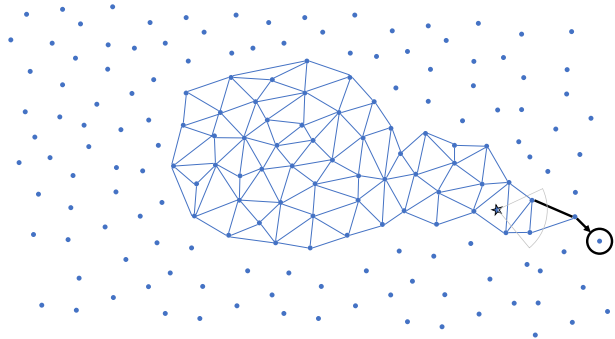
```

1: Initiate Bool successful := false
2: while (!successful) do
3:   Initiate Bool executed_rw := false
4:   Graph  $G := Construct\_Graph(\mathcal{K})$  // Centralized storage of  $\mathcal{K}$  and computation of  $G$ .
5:   Set  $L_i := Shortest\_Path(i^*, G)$  // Centralized computation of path to landmark  $i^*$ .
6:   while  $L_i \neq \emptyset$  do // Execution of path – decentralized, onboard the robot.
7:     Initiate Bool visible := false
8:     for  $j \in \{L_i, \dots, 1\}$  do //
9:       if  $L_i[j]$  is visible to  $r_i$  then
10:         $r_i \leftarrow ESTT\_Observe(r_i, L_i[j])$ 
11:        for  $k \in \{1, 2, \dots, j\}$  do
12:          Set  $L_i := L_i - L_i[k]$ 
13:        end for
14:        visible := true
15:        break
16:      end if
17:    end for
18:    if (!visible) then
19:      if (executed_rw) then
20:        break
21:      else
22:        for  $q \in \{1, 2, \dots, \sigma\}$  do
23:           $r_i \leftarrow RW\_Observe(r_i)$ 
24:        end for
25:        Bool executed_rw := true
26:      end if
27:    end if
28:    end while
29:    if  $L_i = \emptyset$  then
30:      Bool successful := true
31:    end if
32: end while
33: return  $r_i^*$ 

```

In Algorithm 7, the pseudo-code for the $Navigate(r_i, i^*)$ subroutine is presented. Inputs to this function are the position of the i th robot and the identity of the goal landmark i^* . In line 5, the function $Shortest_Path$ 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 17 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 9). 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 $ESTT_Observe(r_i, L_i[j])$ 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

Fig. 14 Too many consecutive *ISW* will lead the boundary of the simplicial complex to grow in horn-like shape as depicted



landmark complex \mathcal{K} . Afterwards, all the path landmarks until $L_i[j]$ will be removed from L_i (Lines 11–13). However, if none of the path landmarks in L_i are visible to the i th robot, the algorithm will perform few steps (σ) of Random Walk (Lines 22–25) and checks for the visibility of the landmarks in L_i one more time. Even if after the execution of Random Walk none 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 . Most of these computations happen on-board the i th robot, except for the computation of the path (sequence of landmarks), which happens in lines 4 and 5.

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. The g_thresh parameter in *ISW* (Algorithm 5) mitigates this by setting a limit on how far the next goal landmark can be. Even then, 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.

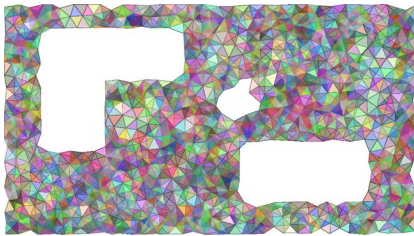
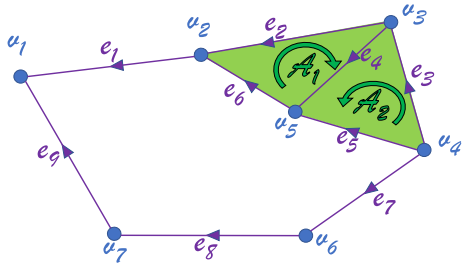
Another issue caused by having consecutive *ISW* is that, the frontier of the simplicial complex could grow non-uniformly in one particular direction. Essentially, *ISW* tries to grow the boundaries and it always do that with the closest landmark. Consequently, after too many consecutive steps of *ISW* the boundary will have a horn-like shape as depicted in Fig. 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*. This approach is implemented later in our complete Landmark Complex Construction algorithm (Algorithm 9).

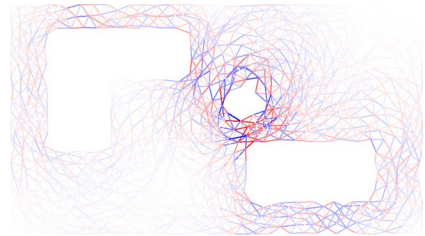
4.4.3 Homology informed walk

Once the robots have performed *RW* and *ISW*, there might still remain some holes in the constructed landmark complex coverage \mathcal{K} due to the insufficient number of observations.

Fig. 15 A directed simplicial complex of dimension 2



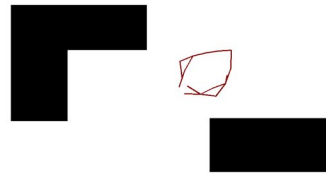
(a) As shown, there is a false hole in the simplicial complex



(b) The corresponding result after solving the Laplacian Dynamics



(c) Result after ℓ_1 -Norm Minimization



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

Fig. 16 The Laplacian Dynamic and ℓ_1 -norm minimization can localize the holes in the simplicial complex

We call them *false holes* (holes in the Landmark complex due to insufficient exploration—See Fig. 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 Fig. 16d). In the following subsections (i.) and (ii.), we briefly describe how existing tools from homology theory can be used to identify these 1-simplices and navigate robots to them. For more details on these methods the reader can refer to [14, 25]. In subsection (iii.) we describe our *Homology-Informed Walk* algorithm.

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.

We start with a simple example of how these matrices are constructed. Consider the simplicial complex in Fig. 15. To construct B_1 and B_2 we assign arbitrary orientation to the 1-simplices and 2-simplices as shown in the figure, and index the 1-simplices, 2-simplices and 0-simplices with natural numbers. The $(i, j)^{th}$ element of B_1 matrix, where i is the index of the vertex and j is the index of the 1-simplex, is 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 from the i th vertex. For instance, in the given example, the first column of B_1 , which corresponds to e_1 , has +1 for the $(1, 1)^{th}$ element, -1 for the $(2, 1)^{th}$ element, and the rest are zero. The matrix B_1 is equivalent to the *incidence matrix* of the graph consisting of the 0-simplices as its vertices and 1-simplices as its edges.

Similarly, the $(j, k)^{th}$ element of B_2 , where j is the index of the 1-simplex and k is the index of the 2-simplex, is 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, in the example of Fig. 15, the $(2, 1)^{th}$ element of B_2 is -1 , whereas $(4, 1)^{th}$ and $(6, 1)^{th}$ elements of B_2 are +1, while the rest are zero. The complete B_1 and B_2 matrices of the simplicial complex in Fig. 15 are shown below:

$$B_1 = \begin{matrix} & \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{matrix} \\ \begin{matrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \end{matrix}, \quad B_2 = \begin{matrix} & \begin{matrix} A_1 & A_2 \end{matrix} \\ \begin{matrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{matrix} & \begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & -1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

These matrices, viewed as linear maps $B_1 : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $B_2 : \mathbb{R}^p \rightarrow \mathbb{R}^m$, compute *boundaries* of simplices (in case of a 1-simplex this is the difference of the 0 simplices at its ends, in case of a 2-simplex this is the linear combination of 1-simplices that forms its boundary) and linear combinations of simplices (referred to as *chains*). Because of the way these matrices are defined, the following identity holds: $B_2 B_1 = 0$ – i.e., the boundary of the boundary of a 2-chain is always zero.

In order to compute and reason about 1-cycles (i.e., 1-chains with empty boundaries—these corresponds to closed loops), the 1st order Laplacian matrix \mathcal{L}_1 is defined as the following $m \times m$ matrix,

$$\mathcal{L}_1 = B_1^T B_1 + B_2 B_2^T \tag{1}$$

It is easy to observe that this matrix is positive semi-definite. It can be shown that the kernel of \mathcal{L}_1 is spanned by 1-cycles. We are interested in specific elements from the kernel of \mathcal{L}_1 which correspond to 1-cycles (a linear combination of 1-simplices that represent a closed loop) forming tight cycles around the *holes* in the complex. We use the method outlined in [14, 46] for computing such a 1-cycle.

ii. Laplacian dynamics and ℓ_1 -norm minimization: In order to identify the 1-cycles that bound the *false holes* tightly, we used the method from [14]. The method starts with the computation of an element $x \in \ker \mathcal{L}_1$ using the combinatorial Laplacian flow,

$$\dot{x}(t) = -\mathcal{L}_1 x(t), \quad x(0) \in \mathbb{R}^m \quad (2)$$

Starting with a randomly generated $x(0) \in \mathbb{R}^m$, where m is the number of 1-simplices in the landmark complex \mathcal{K} , the Laplacian flow converges to a $x \in \ker \mathcal{L}_1$ which corresponds to a linear combination of 1-cycles that bound the 1-dimensional holes as well as 1-cycles that are trivial (i.e., boundaries of 2-chains)—See Fig. 16b. This is because, in the eigenbasis of \mathcal{L}_1 , the components of x corresponding to the non-zero (positive) eigenvalues of \mathcal{L}_1 decay to zero exponentially fast under the dynamics of (2), while only the component along the null-space of \mathcal{L}_1 survive.

In order to get the tightest 1-cycle around a hole we solve the following ℓ_1 -norm optimization problem:

$$\min_{z \in \mathbb{R}^p} \|x + B_2 z\|_1 \quad (3)$$

where x is the converged solution from the combinatorial Laplacian flow, (2). The ℓ_1 norm minimization is a LP-relaxation for an original ℓ_0 norm minimization problem [46], $\min_{z \in \mathbb{R}^p} \|x + B_2 z\|_0$. The intuition behind minimization of the the ℓ_0 norm of the vector (the vector $x + B_2 z$ in this case, which represents a 1-cycle in the same homology class as x , since adding a boundary of a 2-chain keeps its homology class unchanged) is that it minimizes the number of non-zero elements in the vector. This corresponds to the 1-cycle with the least number of 1-simplices (non-zero elements)—in other words, the “tightest” 1-cycle.

To solve this ℓ_1 norm minimization problem, a subgradient method is employed [46].

$$z^{k+1} = z^k - \alpha_k B_2^T \text{sgn}(B_2 z^k + x) \quad (4)$$

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 \mathcal{K} . Moreover, α_k is the step size, and by picking a small enough α_k , the converged z gets close to an optimal solution (See Fig. 16c).

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

Algorithm 8 *Homology_Informed_Walk*(\mathcal{R}) // **Centralized** identification of “holes” in Landmark Complex, \mathcal{K} .

Require:

```

a. Landmark Complex  $\mathcal{K}$  (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:  $\mathcal{R} = \{r_i \in \mathcal{C} \subseteq SE(2) \mid 1 \leq i \leq N\}$ 

```

```

1: Initiate  $\mathcal{Q} = \{\}$  //Set of 1-simplices identities
2: Initiate  $C_c = \{\}$  //Global variable of connected components' identity
3:  $x^* :=$  Converged solution of  $\dot{x}(t) = -\mathcal{L}_1 x(t)$ 
4:  $z^* := \operatorname{argmin}_{z \in \mathbb{R}^m} \|x + B_2 z\|_1$ 
5: Set  $y = x^* + B_2 z^*$ 
6: for  $i \in \{1, 2, \dots, m\}$  do //  $m$  is the number of 1-simplices in  $\mathcal{K}$ 
7:   if  $|y[i]| > \zeta$  then //The  $i^{\text{th}}$  element of vector  $y$ 
8:     Set  $\mathcal{Q} \leftarrow \mathcal{Q} \cup i$ 
9:   end if
10: end for
11: Graph  $G' := \text{Construct\_Graph}(\mathcal{Q})$ 
12: Set  $C_c := \text{Identify\_Connected\_Components}(G')$  //By using Dijkstra algorithm on graph  $G'$ 
13: for all  $i \in \{1, 2, \dots, N\}$  on individual threads do // De-centralized on individual robots.
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  $k \in \{1, 2, \dots, w\}$  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

```

In line 3 and 4 of this pseudo-code, x^* and z^* respectively are the converged solutions of Eqs. (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 (ζ), the identity of that 1-simplex will be inserted to \mathcal{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 ζ is computed using the standard deviation of $|y[i]|$, $i = 1, 2, \dots, m$. Since a large number of elements in vector y have values close to zero and only a small fraction has absolute values greater than zero (See Fig. 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 $|y[i]|$, $i = 1, 2, \dots, m$ to compute ζ , 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 \mathcal{K} (See Fig. 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 $\mathcal{V}(G')$ and the edge set $\mathcal{E}(G')$ are the 0-simplices and 1-simplices in \mathcal{Q} (Line 11). Afterwards in line 12, the function *Identify_Connected_Components* takes the graph G' as an input and by using Dijkstra algorithm on G' , is able to identify these connected components. The output of this function is the set C_c where the i th element in the set is itself a set of 0-simplices corresponding to the i th connected component.

In lines 13–20, 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 cubic complexity [31]. To

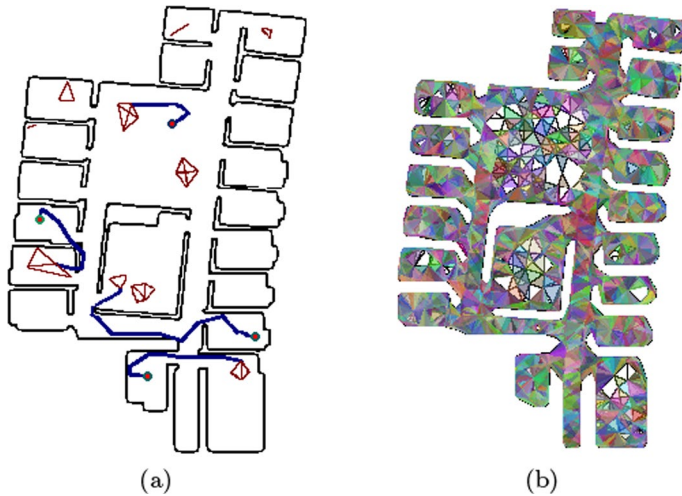


Fig. 17 Hungarian assignments with corresponding shortest paths to the false holes in the simplicial complex. In **a** some of the false holes and the assigned robots with the shortest paths to each assignment are shown, and **b** the corresponding Landmark complex (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in \mathbb{R}^2 just for the purpose of visualization)

use Hungarian Algorithm, we compute the cost matrix C_t (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 C_t 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 \mathcal{K} . Furthermore, in line 16, the *Hungarian_Assignment* function, takes the cost matrix C_t and the identity of the robot running this thread as inputs, and returns the identity of the assigned connected component (j^*). Lines 17–19 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. Since computation of the 0-simplices adjacent to holes require information about the complete simplicial complex, \mathcal{K} , lines 1–12 need to be implemented on the centralized server. However, the Hungarian assignment computation is done by individual robots separately (on their own individual threads), and each robot uses its own computed assignment to navigate to its goal landmark using 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 Fig. 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 Fig. 18. In Fig. 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 Sect. 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.

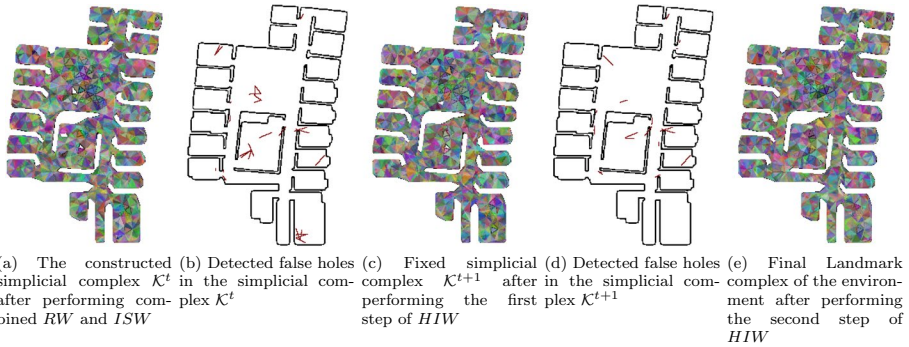


Fig. 18 Detection and exploration of *false holes* using HIW (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in \mathbb{R}^2 just for the purpose of visualization)

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 \mathcal{K} . Moreover, this algorithm runs for each robot on a separate thread (Line 1 through 17).

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 \mathcal{K}

```

1: for  $i \in \{1, 2, 3, \dots, N\}$  on individual thread do // De-centralized, on individual robots.
2:   for  $j \in \{1, 2, \dots, \gamma\}$  do
3:      $r_i \leftarrow RW\_Observe(r_i)$  // Involves the physical process of taking an observation and moving robot.
4:   end for
5:   while rate of growth  $\mathcal{K} \leq \varepsilon_1$  do
6:      $i^* := ISW(\mathcal{R}, i, \xi)$ 
7:     if  $ISW\_counter \geq \eta$  then
8:       for  $k \in \{1, \dots, \delta\}$  do
9:          $r_i \leftarrow RW\_Observe(r_i)$  // Involves the physical process of taking an observation and moving robot.
10:      end for
11:       $ISW\_counter = 0$ 
12:    else
13:       $r_i \leftarrow Navigate(r_i, i^*)$  // Involves the physical process of taking an observation and moving robot.
14:       $ISW\_counter++$ 
15:    end if
16:  end while
17: end for
18: while rate of growth  $\mathcal{K} \leq \varepsilon_2$  do
19:    $Homology\_Informed\_Walk(\mathcal{R})$ 
20: end while
21: return  $\mathcal{K}$ 

```

Since at the beginning of the exploration, the location of landmarks and robots are unknown, robots will perform a fixed number of steps (γ) of RW , to construct a partial

landmark complex in order to localize 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 5 through 16) until the growth rate of landmark complex \mathcal{K} become less than ε_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 \mathcal{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.

During *LCCA*, a threshold of ξ is used on the distance (in terms of the number of landmark hops in G) to the goal landmark to prevent navigating the robots to the far frontier landmarks back and forth. In addition, *LCCA* interleaves between *ISW* and *RW* to avoid growing the landmark complex non-uniformly as described in *Informed Systematic Walk* subsection. To this end, in line 7, the algorithm checks the number of *ISW*s that have been performed consecutively. If the counter on consecutive steps of *ISW* is greater than a fixed number η , *LCCA* will execute δ number of *RW* (Lines 8 through 10).

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 \mathcal{K} and to cover the false holes much faster than *ISW* and *RW*. When the growth rate of the landmark complex \mathcal{K} becomes less than ε_2 , where $\varepsilon_2 < \varepsilon_1$, the exploration will be stopped.

The individual threads in this complete algorithm run on the individual robots. However, in the subroutines *IWS*, *Navigate* and *Homology_Informed_Walk*, since computations need to be performed in the centrally-maintained Landmark Complex, \mathcal{K} , the individual threads communicate with the centralized server for the necessary computations in order to received the next target landmark (or sequence of landmarks) that it needs to visit.

5 Results

5.1 Landmark placement algorithm

Our proposed Landmark Placement Algorithm (*LPA*) described in Sect. 3 was implemented in C++ and is available open-source at <https://github.com/subh83/Topological-Mapping-and-Navigation>. The algorithm allows us to design strategic placement of landmarks in an environment during its design/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 Figs. 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, Fig. 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.

5.2 Landmark complex construction algorithm

We also implemented the Landmark Complex Construction Algorithm (*LCCA*) described in Sect. 4 using C++, and is available open-source at <https://github.com/subh83>

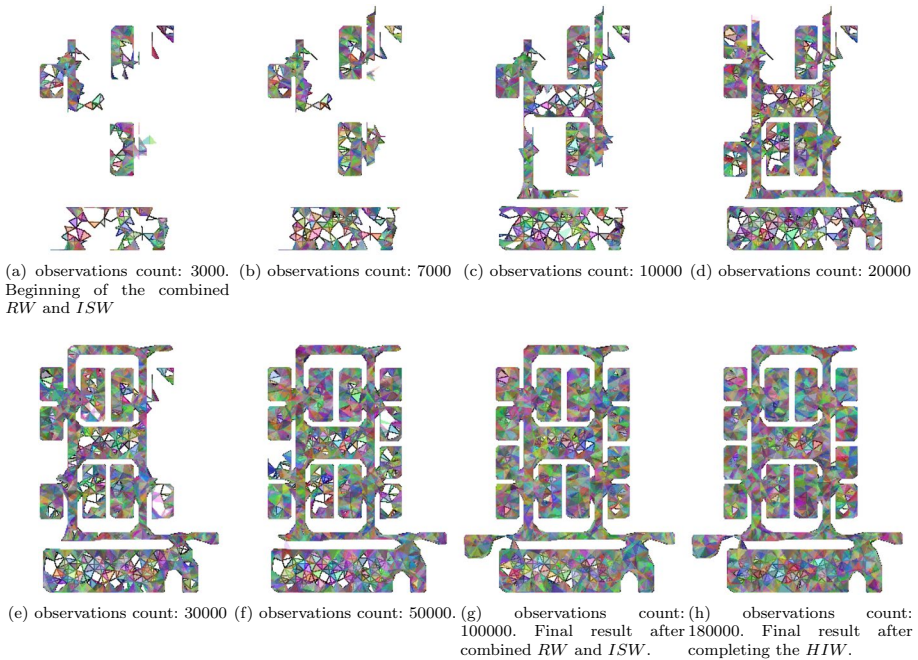


Fig. 19 LCCA in first complex environment (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in \mathbb{R}^2 and superimpose that on top of a map of the actual environment just for the purpose of visualization)

/Topological – Mapping – and – Navigation. 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). In all *LCCA* simulations, unless otherwise specified (such as in Sect. 5.4), we have use 4 robots for easy comparison and consistency. We evaluated the proposed algorithm using two different complex environments refereed to as the *first complex environment* and the *second complex environment*, using the landmark count and distribution shown in Fig. 9 as computed by *LPA*. Results of the Landmark Complex Construction Algorithm is presented in Figs. 19 and 20 for the first and second complex environments respectively.

5.3 Effectiveness of *ISW* Over *RW*

In order to demonstrate that the Informed Systematic walk (*ISW*) is effective in achieving faster exploration compared to a pure Random Walk (*RW*) approach during the combined *RW* and *ISW* phase of the algorithm (i.e., before *HIW* is initiated), we executed the *LCCA* with different amount of initial *RW* steps, γ (see Lines 2–4 of Algorithm 9). As is clear from Algorithm 9, $\gamma = 0$ implies that there is no initial random walk performed, while an increasing value of γ will imply an increasing proportion of the time spent in the combined *RW* and *ISW* is on performing pure *RW*. Figure 21a shows the number of iterations required to complete 85% of the simplicial complex using the combined *RW* and *ISW* with different value of γ . For each value of γ we ran 10 experiments, and each data point shows the

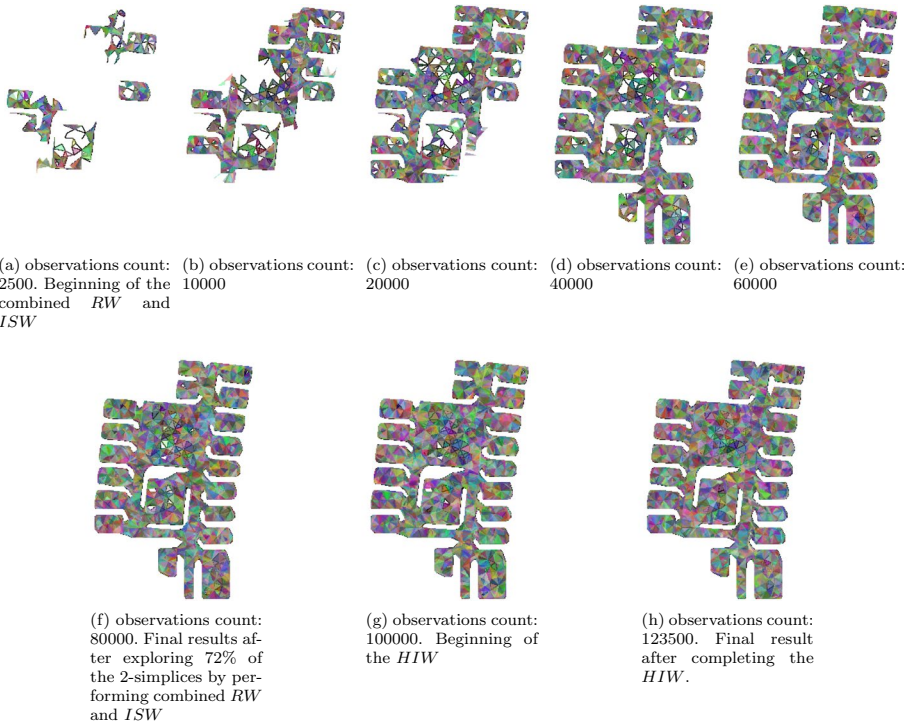
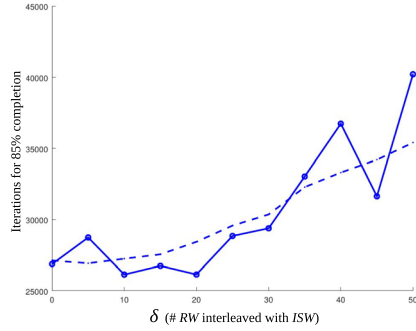
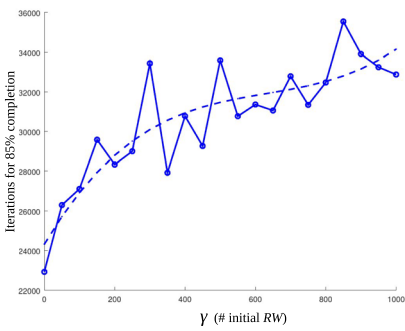


Fig. 20 LCCA in second complex environment (note that the Landmark complex itself is an abstract simplicial complex. We immerse it in \mathbb{R}^2 and superimpose that on top of a map of the actual environment just for the purpose of visualization)

average number of iterations required for completing 85% of the simplicial complex for that value of γ . For these experiments we fixed $\xi = 10$, $\eta = 10$ and $\delta = 5$.

Likewise, we varied the amount of interleaved *RW* as measured by δ (see Lines 8–10 of Algorithm 9) and measured the number of iterations required to complete 85% of the simplicial complex using the combined *RW* and *ISW* with different values of δ . The results are shown in Fig. 21b. For each value of δ we ran 5 experiments, and each data point shows the average number of iterations required for completing 85% of the simplicial complex for that value of δ .

As clearly demonstrated by the plots of Fig. 21, despite the fluctuations, the overall trend in the data shows an increasing number of required iterations with an increasing amount of random walk, indicating that a higher proportion of *ISW* helps with faster exploration. However, in general, for the other experiments, we do not set $\gamma = 0$ though, since, at a small time-scale, with no initial simplicial complex to work with, *ISW* alone tend to have difficulty in constructing an initial simplicial complex. For all the previous experiments we chose $\gamma = 10$. Likewise, for the reasons described in Sect. 4.4.2.iv., we choose a non-zero $\delta = 5$, to ensure that the *ISW* is interleaved with *RW* to avoid robots creating a *horn-like shape* of the explored domain.

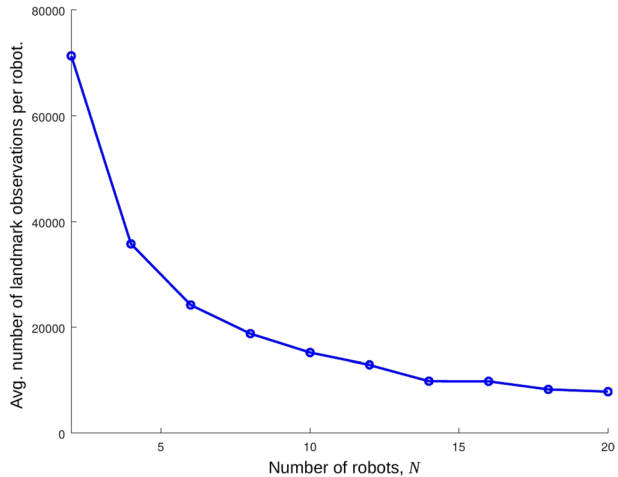


(a) The number of iterations required increases with the amount of pure initial *RW* used for the exploration. With increasing γ , more iterations are spent on the initial *RW* than on the *ISW*. Solid curve shows actual data points, while the dashed curve shows a cubic polynomial fit minimizing the least square error.

(b) The number of iterations required increases with the amount of interleaved *RW* used for the exploration. Solid curve shows actual data points, while the dashed curve shows a windowed average.

Fig. 21 The number of iterations taken to construct 85% of the simplicial complex in the second complex environment using combined *RW* and *ISW*

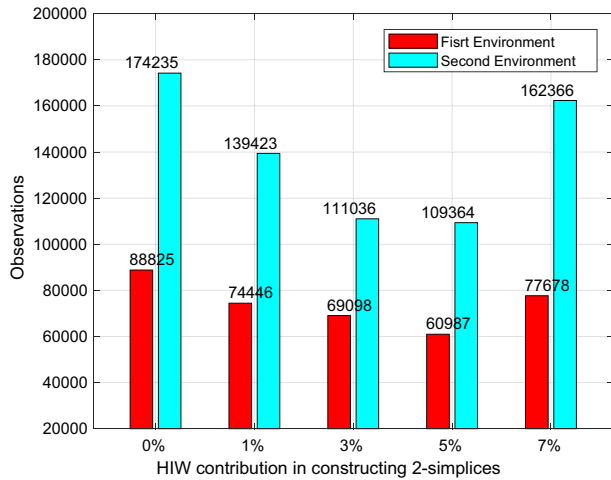
Fig. 22 The average number of observations per robot decreases with the number of robots used in constructing 85% of the landmark complex in the second complex environment using the combined *RW* and *ISW*. Each data point shows the average value over 10 different simulations with the same number of robots



5.4 Effect of changing the number of robots

In order to demonstrate that our *LCCA* scales with the number of robots, we evaluated the performance of the algorithm with an increasing number of robots, N (note that for all other *LCCA* simulations we have used $N = 4$). Figure 22 unsurprisingly shows that with increasing number of robots the number of observations per robot required for exploration decreases. We used the number of observations per robot (note that the total observation count will be similar for constructing the 85% of the landmark complex irrespective of the number of robots) as the metric for comparison because, with our current multi-threaded implementation of the simulations running on a single computer processor, the total simulation time (either in terms of seconds or the number of iterations in the central server thread) does not decrease monotonically with the number of

Fig. 23 Different *HIW* contributions on constructing the 2-simplices in the Cech complexes of the visibility domain of the landmarks, to reach the target amount for the first and second complex environments



robots. This is because with the increasing number of threads, the processing overhead for the processor also increases.

5.5 Role of homology informed walk in LCCA

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 Cech 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 remaining 0%, 1%, 3%, 5% and 7% respectively of the simplices to reach the target 98%. Figure 23 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 Fig. 23 suggests that the optimum level of *HIW* use is to construct the last 5% of total 2-simplices with *HIW*.

5.6 Growth rate based criterion for switching to *HIW*

Since the total number of 2-simplices is unknown to the robots, we need a strategy for switching to *HIW* without using the percentage completion of the simplicial complex. 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 24 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

Fig. 24 Growth rate of the simplicial complex (r) as a function of time (measured in terms of number of iterations in the central server thread)

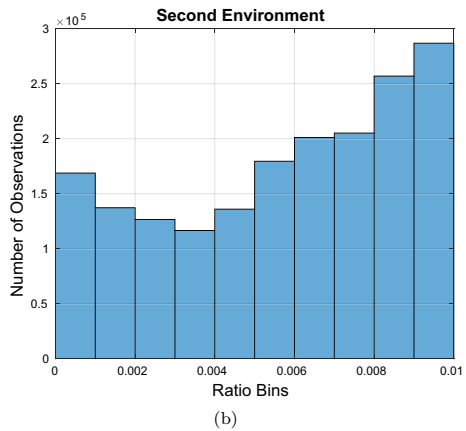
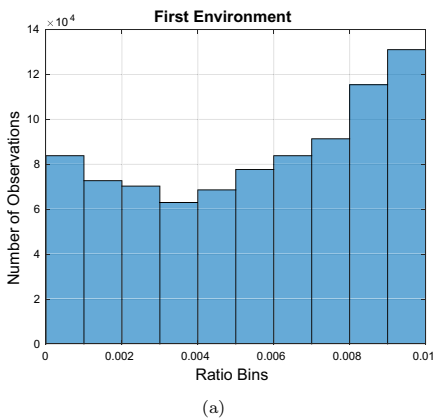
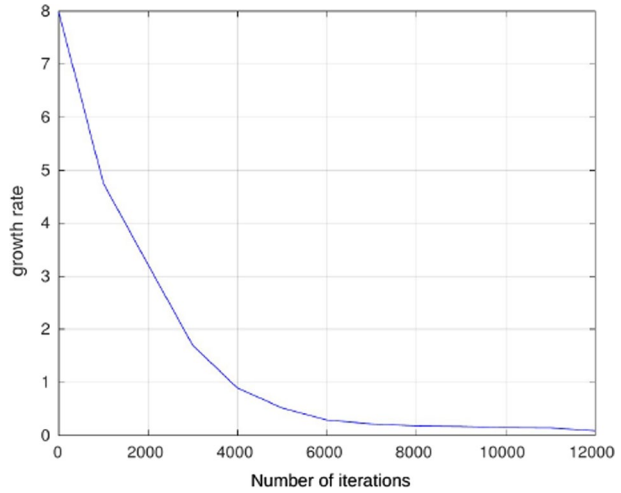
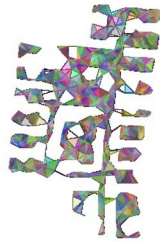


Fig. 25 The number of the observations required to complete the simplicial complex plotted against the growth rate value at which the switch to *HIW* is performed. As evident, a switch at the growth rate of about 0.004 gives an optimal performance in either of the environments

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 Fig. 25. 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 at 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



(a) Randomly placed landmarks in the second complex environment.



(b) Čech Complex constructed by the visibility domains of the randomly-placed landmarks.



(c) Landmark complex constructed using *LCCA*. Only about 86% of the complex shown in (b) was completed, at which point no further improvement was observed.

Fig. 26 Effect of randomly placed landmarks on the Čech Complex and the performance of *LCCA*

from the plots in Fig. 25, a growth rate of about 0.004 is the optimal value at which switching to *HIW* minimizes the total observation count.

5.7 Performance of *LCCA* with sparse and randomly placed landmarks

The performance of *LCCA* gives asymptotic guarantee on complete construction of the landmark complex only if the landmarks satisfy the necessary visibility condition (see Sect. 3.2). This is guaranteed by the Landmark Placement Algorithm (*LPA*) proposed in the paper. However, in an environment where the landmarks are sparse and randomly-placed, the Čech Complex of the visibility domains of the landmarks do not form a correct representation of the configuration space. This is illustrated in the example of Fig. 26a, where we placed about 50% of the landmarks as suggested by the *LPA*, but dispersed randomly throughout the second complex environment. As can be seen from Fig. 26b, the Čech Complex of the domains of visibility have two major issues:

- i. Due to random placement, the landmarks missed some narrow passages, and hence created multiple disconnected components in the complex, and,
- ii. It also created multiple *false holes* in the complex that do not correspond to or bound any obstacles in the environment

As a consequence, *LCCA* has difficulty in constructing the landmark complex because: *i.* In the combined *RW* and *ISW* phase of the algorithm, the robots are often unable to explore and discover components of the complex disconnected from (or weakly connected to) the component that it started in, and, *ii.* in the *HIW* phase of the algorithm the robots are repeatedly driven towards the false holes, which they are unable to *fill* because the absence of the necessary landmarks for constructing the the simplices that fill those hole make the holes intrinsic to the complex that cannot be filled. As a result, upon running *LCCA* with the landmarks of Fig. 26a, we were able to complete about 86% of the full landmark complex (i.e., the landmark complex had 86% of the 2-simplices in the Čech Complex shown in Fig. 26b) and the final result of *LCCA* is shown in Fig. 26c. At this point *HIW* was unable to make any further progress on the exploration, and as can be observed, the false holes remained un-filled and there are a few disconnected or weakly-connected components of the complex that remained undiscovered.

The landmark placement algorithm (Algorithm 1) and the landmark complex construction algorithm (Algorithm 9) are intrinsically linked to each other by the necessary landmark density criterion, which depends on the footprint of the sensors onboard the robots as described in Sect. 3.2. Having landmarks placed in the environment using a different approach, with less density and/or with fewer number, will unsurprisingly result in incomplete construction of the landmark complex, which in turn due to violation of Proposition 1, will not be a topologically-correct (homotopy equivalent) representation of the configuration space.

5.8 Comparison with other methods

Our proposed approach is unique because we assume that each robot has extremely limited sensing capabilities (only the binary information on whether or not a particular landmark is visible) and no odometry information. To the best of our knowledge, there exists no other work that addresses the problem with such limited capability assumptions. 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 (see Sect. 1.2.1 for a more detailed literature review). Any comparison of performance will most certainly prove the existing SLAM approaches to be more efficient and accurate than our proposed approach. So a fair comparison with existing state-of-the-art is not possible. However, 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.

Likewise, for a meaningful comparison with an alternative method for landmark placement there needs to be a meaningful metric for comparison. Existing algorithms for landmark placement, for example [22, 28], solve optimization problems and the landmark placement is achieved to minimize a real-valued objective/cost function that is intricately linked to the specific problem setup and sensing model that those works use. There is no doubt that the result of such landmark placement algorithms will thus minimize their respective objective functions and our proposed landmark placement algorithm will not minimize those same objective functions. However, our proposed landmark placement algorithm serves a completely different purpose altogether—our landmark placement algorithm (LPA) is meant to ensure that our proposed landmark complex construction algorithm (LCCA) can construct the correct Landmark Complex representation of the environment with the extremely limited sensing capabilities (binary information of whether or not a landmark is present in a directional sensor footprint) onboard each robot. The two algorithms are thus co-designed. In our setup, we cannot pose the landmark placement problem as a practical optimization problem, especially since the environment is complex and highly non-convex, and even if it was possible to pose it as an optimization problem, such a formulation would involve a large number of integer variables (due to non-convexity of the environment, and the binary nature of the sensing), and will be completely different from the optimization

problems in existing work. Instead, we use a filtration-based algorithm for landmark placement.

5.9 Conclusion

In this paper we consider the problem of multi-agent landmark-based mapping, exploration and navigation in a localization-free (GPS-denied) environment with extremely limited sensing capabilities and limited computational resources. Each robot can sense the binary information of whether or not a landmark is present in its directional sensor footprint, and for the purpose of navigation is able to sense if the landmark is to the left or right of the robot. For mapping with such limited sensing capability we use a topological representation of the environment called Landmark Complex. This metric-free representation constructed using the binary information of presence of landmarks in robots' sensing domains is robust to sensory noise, requires little onboard computational capability, and is amiable to easy distributed construction using multiple agents. The mapping and exploration algorithm thus developed is called Landmark Complex Construction Algorithm (LCCA). In order to correctly and effectively construct the said landmark complex, the presence of a necessary density/distribution of landmarks in an urban/indoor environment can be achieved during the construction of the environment. To that end, our proposed Landmark Placement Algorithm (LPA) uses a novel filtration-based approach to ensure that at least one landmark is visible from every sensor configuration in the environment.

Acknowledgements This work was partially supported by the National Science Foundation under Grant No. CCF-2144246.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References


1. Angeli, A., Filliat, D., Doncieux, S., & Meyer, J.-A. (2008). Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5), 1027–1037.
2. Beinhofer, M., Müller, J., & Burgard, W. (2013). Effective landmark placement for accurate and reliable mobile robot navigation. *Robotics and Autonomous Systems*, 61(10), 1060–1069.
3. Bhattacharya, S., Ghrist, R., & Kumar, V. (2014). Multi-robot coverage and exploration on Riemannian manifolds with boundary. *International Journal of Robotics Research*, 33(1), 113–137. <https://doi.org/10.1177/0278364913507324>
4. Bhattacharya, S., Michael, N., & Kumar, V. (2010). Distributed coverage and exploration in unknown non-convex environments. In *Proceedings of 10th international symposium on distributed autonomous robotics systems* (pp. 1–3). Springer.
5. Carrillo, H., Dames, P., Kumar, V., & Castellanos, J. A. (2015). Autonomous robotic exploration using occupancy grid maps and graph slam based on Shannon and Renyi entropy. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 487–494).
6. Casati, G., Longhi, M., Latini, D., Carbone, F., Amendola, S., Del Frate, F., Schiavon, G., & Marrocco, G. (2017). The interrogation footprint of rfid-uav: Electromagnetic modeling and experimentations. *IEEE Journal of Radio Frequency Identification*, 1(2), 155–162.

7. Castellanos, J. A., Montiel, J. M., Neira, J., & Tardós, J. D. (1999). The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5), 948–952.
8. Chen, Y., Hafez, O. A., Pervan, B., & Spenko, M. (2020). Landmark augmentation for mobile robot localization safety. *IEEE Robotics and Automation Letters*, 6(1), 119–126.
9. Chen, Y., Francisco, J.A., Trappe, W. and Martin, R.P. (2006). A practical approach to landmark deployment for indoor localization. In *2006 3rd annual IEEE communications society on sensor and Ad Hoc communications and networks* (Vol. 1, pp. 365–373). IEEE.
10. Choset, H., & Nagatani, K. (2001). Topological simultaneous localization and mapping (slam): Toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2), 125–137.
11. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. Cambridge: MIT Press.
12. Dai, A., Papatheodorou, S., Funk, N., Tzoumanikas, D, & Leutenegger, S. (2020). Fast frontier-based information-driven autonomous exploration with an mav. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 9570–9576). IEEE.
13. Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 1052–1067.
14. Derenick, J., Kumar, V., & Jadbabaie, A. (2010). Towards simplicial coverage repair for mobile robot teams. In *2010 IEEE international conference on robotics and automation* (pp. 5472–5477).
15. Dirafzoon, A., & Lobaton, E. (2013). Topological mapping of unknown environments using an unlocalized robotic swarm. In *2013 IEEE/RSJ international conference on intelligent robots and systems (iros)* (pp. 5545–5451).
16. Dirafzoon, A., Betthausser, J., Schornick, J., Benavides, D., & Lobaton, E. (2014). Mapping of unknown environments using minimal sensing from a stochastic swarm. In *2014 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3842–3849).
17. Dubins, L. (1957). 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.
18. Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2), 99–110.
19. Engel, J., Schops, T., & Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision - ECCV 2014* (pp. 834–849). Springer.
20. Engel, J., Stücker, J., & Cremers, D. (2015). Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1935–1942). IEEE.
21. Erdos, P., & Taylor, S. J. (1960). Some intersection properties of random walk paths. *Acta Mathematica Academiae Scientiarum Hungaricae*, 11(231), 218.
22. Falque, R., Patel, M., & Biehl, J. (2018). Optimizing placement and number of rf beacons to achieve better indoor localization. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 2304–2311). IEEE.
23. Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)* (pp. 15–22). IEEE.
24. Ghrist, R., Lipsky, D., Derenick, J., & Speranzon, A. (2012). Topological landmark-based navigation and mapping. University of Pennsylvania, Department of Mathematics, Tech. Rep. 8.
25. Hatcher, A. (2001). *Algebraic topology*. Cambridge Univ. Press.
26. Howard, A. (2006). Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12), 1243–1256.
27. Ila, V., Porta, J. M., & Andrade-Cetto, J. (2009). Information-based compact pose slam. *IEEE Transactions on Robotics*, 26(1), 78–93.
28. Jourdan, D. B., & Roy, N. (2008). Optimal sensor placement for agent localization. *ACM Transactions on Sensor Networks (TOSN)*, 4(3), 1–40.
29. Junyan, H., Niu, H., Carrasco, J., Lennox, B., & Arvin, F. (2020). Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12), 14413–14423.
30. Knight, W. (2015). *The roomba now sees and maps a home*. MIT Technology Review.
31. Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2), 83–97.
32. Lategahn, H., Geiger, A., & Kitt, B. (2011). Visual slam for autonomous ground vehicles. In *2011 IEEE international conference on robotics and automation* (pp. 1732–1737).

33. Lei, Z., Chen, X., Tan, Y., Chen, X., & Chai, L. (2022). Optimization of directional landmark deployment for visual observer on se (3). *IEEE Transactions on Industrial Electronics*, 24, 5994–6003.
34. Liu, S., Guo, P., Feng, L., & Yang, A. (2019). Accurate and robust monocular slam with omnidirectional cameras. *Sensors*, 19(20), 4494.
35. Lluvia, I., Lazkano, E., & Ansuategi, A. (2021). Active mapping and robot exploration: A survey. *Sensors*, 21(7), 2445.
36. Magnago, V., Palopoli, L., Passerone, R., Fontanelli, D., & Macii, D. (2019). Effective landmark placement for robot indoor localization with position uncertainty constraints. *IEEE Transactions on Instrumentation and Measurement*, 68(11), 4443–4455.
37. Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI national conference on artificial intelligence* (pp. 593–598). AAAI.
38. Munguia, R. & Grau, A. (2007). Monocular slam for visual odometry. In *2007 IEEE international symposium on intelligent signal processing* (pp. 1–6). IEEE.
39. Mur-Artal, R., Montiel, J. M., & Tardos, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5), 1147–1163.
40. Pimentel, J. M., Alvim, M. S., Campos, M. F., & Macharet, D. G. (2018). Information-driven rapidly-exploring random tree for efficient environment exploration. *Journal of Intelligent & Robotic Systems*, 91(2), 313–331.
41. Quan, M., Piao, S., Tan, M., & Huang, S.-S. (2019). Tightly-coupled monocular visual-odometric slam using wheels and a mems gyroscope. *IEEE Access*, 7, 97374–97389.
42. Ramachandran, R. K., Wilson, S., & Berman, S. (2017). A probabilistic approach to automated construction of topological maps using a stochastic robotic swarm. *IEEE Robotics and Automation Letters*, 2(2), 616–623.
43. Ramaithitima, R. (2019). Sensor-based topological coverage and mapping algorithms for resource-constrained robot swarms. University of Pennsylvania. Ph.D. thesis.
44. Ramaithitima, R., Bhattacharya, S. (2018). Landmark-based exploration with swarm of resource constrained robots. In: *Proceedings of IEEE international conference on robotics and automation (ICRA)*.
45. Revesz, P. (2013). *Random walk in random and non-random environments*. World Scientific.
46. Tahbaz-Salehi, A., & Jadbabaie, A. (2010). Distributed coverage verification in sensor networks without location information. *IEEE Transactions on Automatic Control*, 55(8), 1837–1849.
47. Thrun, S., & Liu, Y. (2005). Multi-robot slam with sparse extended information filters. *Robotics Research* (pp. 254–266).
48. Valencia, R., & Andrade-Cetto, J. (2018). Active pose slam. In *Mapping, planning and exploration with pose SLAM* (pp. 89–108). Springer.
49. Vitus, M. P., & Tomlin, C. J. (2011). Sensor placement for improved robotic navigation. *Robotics Science and Systems*, 6, 217.
50. Zhao, J., Huang, Y., He, X., Zhang, S., Ye, C., Feng, T., & Xiong, L. (2019). Visual semantic landmark-based robust mapping and localization for autonomous indoor parking. *Sensors*, 19(1), 161.
51. Zhou, W., Miro, J. V., & Dissanayake, G. (2008). Information-driven 6d slam based on ranging vision. In *2008 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2072–2077). IEEE.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Mohammad Saleh Teymouri¹ · Subhrajit Bhattacharya¹ 

✉ Subhrajit Bhattacharya
sub216@lehigh.edu

Mohammad Saleh Teymouri
mot317@lehigh.edu

¹ Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18015, USA