2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Topological Exploration of Unknown and Partially Known Environments

Soonkyum Kim[1], Subhrajit Bhattacharya[2], Robert Ghrist[3] and Vijay Kumar[4]

***Abstract*— We present the mathematical framework and algorithms for multi-robot topological exploration of unknown environments in which the main goal is to identify the different topological classes of trajectories and thus efficiently distribute the task of exploration among different groups of robots. We consider two-dimensional configuration spaces. At any point in time, the robots' map consists of known, partially-mapped obstacles. The unknown, yet-to-be-explored area is mapped to a single point, thus giving us a *quotient space*. The topological classes on the quotient space allows us to define topological classes of trajectories connecting a robot pose to the unknown region in the original configuration space. Robots explore this configuration space choosing different homology classes when confronted by obstacles or walls. We illustrate the basic idea with simulations of small teams of robots. Experiments with a single robot illustrate the applicability of the method to robots that have small sensor footprints and limited computational resources. We also provide comparisons with a standard frontier-based algorithm.**

I. Introduction

Exploration and mapping have been treated quite extensively in the robotics literature. The general problem can be formulated as finding the next best view or pose [12] to acquire information required to build a map of the environment [15]. In most settings, the spatial representation of the map is based on metric information. Indeed approaches like metric-based multi-robot coordinated exploration have been studied widely in the past [1], [4], [14]. In decision-theoretic approaches to exploration, mutual information and entropy are often used [14], [16], [13], [17] to guide robots to perform efficient exploration. Simpler approaches involving the identification of frontiers and segmentation representing the boundaries between unexplored and explored regions have also been widely used for deployment of robots in exploration and mapping of unknown or partially known environments [20], [6], [18], [9]. Most of these fundamental techniques work in conjunction with coordination strategies for the multiple robots or team of robots.

In contrast, we are interested in building a topological representation of the environment that might serve as a coarse map for coverage or for search and rescue by a single robot or allow for coordination and planning of multiple robots engaged in cooperative tasks. In this paper, we explicitly compute the topological classes of trajectories (we will consider only *non-looping* classes [2] – classes in which the optimal trajectory is *embedded*) connecting the explored region to the unexplored region and use this to guide deployment of robots for efficient exploration of the environment. We also provide a detailed coordination algorithm for achieving the later. Methods involving segmentation of the known region and construction of a *Voronoi graph* for efficient deployment [19] is akin to a topological approach as ours. However, in our approach, we avoid complex segmentation algorithm by direct computation of optimal trajectories in different topological classes leading to the unknown region.

In the following discussion we assume that the reader is familiar with the notion of topological equivalences of trajectories, in particular that of *homology* [2], [11].

To motivate the approach in this paper, consider the simple scenario in Figure 1(a) in which there is a group of robots at locations close to $\mathbf{p}$ equipped with sensors with a limited field of view mapping an unknown environment. In the figure, the current map consists of the three obstacles (marked in black) and the free space colored in pale blue. The region, L, in pale yellow is not visible to any of the sensors and hence is unknown. An information gain maximization based approach as in [13] or [16] will essentially give an unique gradient descent direction at the location of the robots (if all the robots are roughly the same location, the control inputs will also be very similar) and make the robots move together. However, clearly there are three distinct topological classes in this environment that can lead the robots to the unknown region (indicated by the blue dashed arrows in the figure). We are interested in methods that will maximize the collection of *information* by naturally assigning robots to different topological classes of trajectories.

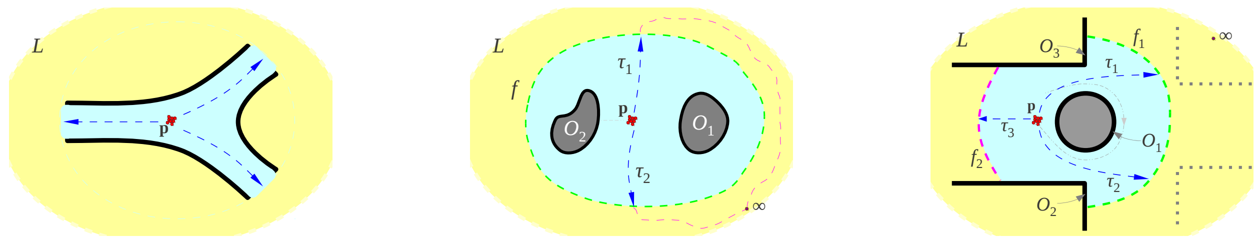
A frontier-based exploration as in [6] would find three distinct trajectories or assignments in two steps [18]: *i.* Identify the boundary between the known and the unknown regions and segment it to obtain its connected components (using an edge detection algorithm for example), followed by *ii.* finding optimal trajectories to each of the connected components (using Dijkstra's search). While this method would perform satisfactorily in the example of Figure 1(a), remarkable out-performance of a topology-based method as ours is observed in scenarios like that in Figures 1(b) or 1(c) – when the known region is not simply-connected. Moreover, in a frontier-based algorithm like in [18], the additional step of identification of the connected components of the frontier may be expensive. When there are multiple robots, the standard frontier-based approach makes robot-frontier assignment by taking into consideration the *size* of

[1]Soonkyum Kim is a graduate student in Department of Mechanical Engineering and Applied Mechanics. soonkyum@seas.upenn.edu

[2]Subhrajit Bhattacharya is a post-doctoral researcher in Department of Mathematics. subhrabh@math.upenn.edu

[3]Robert Ghrist is a professor with Department of Mathematics and Department of Electrical and Systems Engineering. ghrist@math.upenn.edu

[4]Vijay Kumar is a professor with Department of Mechanical Engineering and Applied Mechanics. kumar@seas.upenn.edu



(a) A group of robots, using their laser range sensors, finds 3 topological classes of paths leading to the unknown region, L. There are also 3 frontiers in this scenario.

(b) In this partially known environment there is a single frontier (green dashed curve), but 2 topological classes connecting the location of the robots and frontier, f. Such scenarios are natural when the known free region, $\mathbb{R}^2 - \mathcal{O} - L$, is not *simply-connected*.

(c) In this scenario the known region is not simply-connected and there are 2 frontiers. But on $\mathbb{R}^2 - \mathcal{O}$ as well as on $(\mathbb{R}^2 - \mathcal{O})/L$ there are 3 non-looping topological classes.

Fig. 1. Partially explored environments. The group of robots (red dots) need to be split and deployed for exploration of the unknown regions (pale yellow region marked as L). The figures illustrate the distinction between frontier-based and topology-based deployments.

the frontier [18]. In an indoor environment with lots of corridors and passages, possibly leading up to large open areas, the size of the frontier may not be the best indicator of information gain. Furthermore, when there are multiple groups of robots in different locations, performing distributed cooperative exploration, it is unclear how the groups can have a consistent way of referring to a particular frontier when communicating (without communicating the complete description of the frontier). Our topological approach, on the other hand, is completely free from the task of frontier identification or representation, and instead uses a single pass of search/planning to discover trajectories in different topological classes, each of which is represented by a topological invariant (H-signature) that is consistent over the different groups (Figure 3(b)).

Consider the scenario illustrated in Figure 1(b), where a group of robots are provided with a partial map of the environment. There is a single frontier, f. A frontier-based approach would find a single shortest path to the frontier (either τ_1 or τ_2). However, there are two topological classes of trajectories in the plane punctured by the obstacle, O, that connect $\mathbf{p}$ to all points in L. As shown, τ_1 and τ_2 are two trajectories in different topological classes. Thus clearly, the number of frontiers do not correspond to the number of distinct non-looping topological classes when the explored/known free region is not simply connected. A similar example is shown in Figure 1(c), where $\mathcal{O}$ is the set of all obstacles that have been discovered. In particular, the group of robots have explored the perimeter of obstacle O_1, thus resulting in a map that is not simply connected. In this case although there are two connected components of the frontier, there are three topological classes of trajectories and therefore three directions for exploration. Clearly, deployment of groups of robots in each of the distinct topological classes will result in more efficient exploration (*i.e.* discovery of the unseen obstacles marked in thick dotted lines).

Our focus here is on topology-based exploration. In our recent work [2] we developed a method to find least cost paths in different topological classes connecting a start and a goal coordinate. We can choose a *candidate point* in the unknown region (for example, the points at ∞ shown in Figure 1) and directly use the method in [2] to determine the lowest cost trajectories leading up to the frontiers, but one in each topological class in $\mathbb{R}^2 - \mathcal{O}$. Since we are interested in the topological classes in the known region for deciding deployment strategy, and do not want any possible non-trivial topology of the unknown region to influence that, we will *collapse* the unknown region, L, to a single point. We do not want multiplicity in the topological classes due to topological features in L (*e.g.* trajectories in different homology classes in $\mathbb{R}^2 - \mathcal{O}$ that agree on the explored free space, $\mathbb{R}^2 - \mathcal{O} - L$, but are in different topological classes in L). Thus, we use the notion of *quotient spaces* [11], and describe a *homology class invariant* in the quotient space, $(\mathbb{R}^2 - \mathcal{O})/L$, to characterize the topological classes of trajectories.

Because our interest is in topological mapping, we will not concern ourselves with questions of localization, detection or mapping of obstacles or control. We will assume each robot is able to localize either using lasers and cameras or by using GPS, detect obstacles with a laser scanner, communicate with other robots, and avoid collisions with the environment. We implement our algorithm with nonholonomic robots in ROS, and demonstrate the multi robot exploration in simulation, along with comparisons with a frontier-based exploration algorithm. We also present experimental results with a single robot with a small field-of-view laser and odometry to illustrate the basic ideas in a real world setting.

II. Homology Classes

A. Homotopy and Homology Class Invariants of Trajectories

Two trajectories connecting the same start and goal coordinates are said to be in the same homotopy class if one can be continuously deformed into the other without intersecting any obstacle and with endpoints always fixed. Such homotopy classes possess algebraic manifestations (in terms of the fundamental group) which are clean but extremely difficult to compute. A simpler analog can be found in homology, which is built on linear-algebraic constructs. Homology classes are similar to, but subtly different from, homotopy classes (see see Fig. 1 of [2]). In [2] we developed an invariant for homology classes of trajectories, called the H-signature, to find trajectories in different homology classes and for planning with topological constraints. In this paper we will exploit this idea for finding trajectories in multiple homology classes in the explored region of the environment.

For completeness of this paper we include below some of the basic definitions that appear in [2]. A few of these definitions are slightly narrower than the general form (as in [8], [11]) for simplicity and to accommodate the present applications.

Definition 1 (Homologous trajectories): Two trajectories τ_1 and τ_2 connecting the same start and end coordinates, $\mathbf{x}_s$ and $\mathbf{x}_g$ respectively, are *homologous* iff $\tau_1 - \tau_2$ (i.e., τ_1 union an orientation-reversed τ_2) forms the complete boundary of a 2-dimensional chain (i.e., compact sub-domain) in the free configuration space, $\mathcal{C}$, not containing/intersecting any of the obstacles.

We represent $\mathbb{R}^2$ by the complex plane, $\mathbb{C}$.

Definition 2 (Representative points [2]): Given a set of connected planar obstacles, $\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_N$, a *system of representative points*, denoted $\zeta_l \in Int(\mathcal{O}_l)$, $\forall l = 1, \cdots, N$, is a selection of one point in the interior of each obstacle. The exact location of the representative points is not of particular significance as long as each lies inside the respective obstacle interiors.

Note that in general it is possible that two (or more) of the obstacles might not be mutually disjoint. That is, $\mathcal{O}_i$ and $\mathcal{O}_j$ can be parts of a same connected obstacle, but each of these parts need to be connected. This amounts to assigning multiple representative points for the same connected component of an obstacle, which changes the H-signature but not the basic method.

Definition 3 (H-signature in $\mathbb{R}^2$ punctured by obstacles): For the given system of N obstacles and their representative points, we define the *obstacle marker function* $\mathcal{F} : \mathbb{C} \to \mathbb{C}^N$ as $\mathcal{F}(z) = \left[\frac{1}{z-\zeta_1}, \frac{1}{z-\zeta_2}, \cdots, \frac{1}{z-\zeta_N}\right]^T$. We thus define the H-signature of a trajectory τ using the complex integral

$$\mathcal{H}(\tau) = \int_\tau \mathcal{F}(z) \mathrm{d}z$$

Lemma 1 (H-signature as homology class invariant [2]): Two trajectories τ_1 and τ_2 connecting the same points in the plane punctured by obstacles are homologous if and only if $\mathcal{H}(\tau_1) = \mathcal{H}(\tau_2)$

Remark: The above lemma gives a way of computing homology invariant in $\mathbb{R}^2 - \mathcal{O}$. Note that from the definition of H-signature and using the *residue theorem* from complex analysis, it is easy to see that when τ is a closed loop, $\mathcal{H}(\tau)$ is a vector of the form $[a_1, a_2, \cdots, a_N]^T 2\pi i$, where $a_l \in \mathbb{Z}$. Moreover, according to the terminology of [2], we call a trajectory *non-looping* (which is an *embedded* curve) if the imaginary part of each element of its H-signature is between $-2\pi i$ and $2\pi i$.

Equipped with these basic tools, we define the homology class invariant for quotient spaces

B. The Quotient Space and H-signature

In [2] the H-signature of trajectories was used for finding different homology classes of trajectories connecting two points. However, for exploration we are interested in the topological classes of trajectories that emanate from a start coordinate, with the goal being not a single point but rather a set L. To adapt to this situation, we collapse the set L to a single point via the construction of a *quotient space* [11].

To adapt the definition of the H-signature in this context, we will collapse the entire unknown region to a single abstract point via a *quotient map*, q, so that the space under

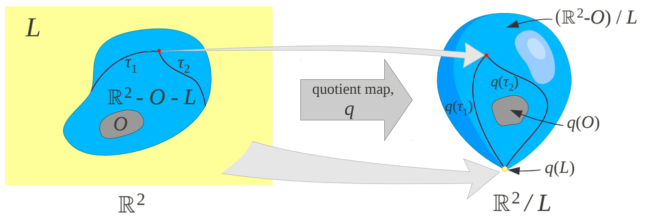


Fig. 2. A simple illustration of a quotient map. The set L is collapsed to a point, $q(L)$. Here we consider the Euclidean plane, $\mathbb{R}^2$, with its subset L being the entire region outside a small disk on the plane. Collapsing L to a single point gives us the topological 2-sphere. All non-trivial 1-cycles (or closed loops) that completely lie in L become trivial in the quotient space under the quotient map, q.

consideration becomes $(\mathbb{R}^2 - \mathcal{O})/L$ (where $\mathcal{O}$ is the set of obstacles in the known region). The image of L under the quotient map, q, thus being a single point lets us use the notion of homology classes of trajectories connecting to this point from the image of the start coordinate on the quotient space (Figure 2). For a formal definition of quotient map see [11], [8].

The following proposition extends the result of Lemma 1 to the quotient space $(\mathbb{R}^2 - \mathcal{O})/L$.

Proposition 1 (Homology invariant in quotient space [3]): Let $\mathcal{O}$ be the collection of obstacles in $\mathbb{R}^2$ with respect to which we compute the H-signature as described in [2], and let $L \subset \mathbb{R}^2 - \mathcal{O}$. Let Q be the set of H-signatures of all closed loops (1-cycles) in $(\mathbb{R}^2 - \mathcal{O})$ contained entirely in L. Let τ_1 and τ_2 be two trajectories connecting two points, $\mathbf{s}, \mathbf{g} \in (\mathbb{R}^2 - \mathcal{O})$. Now consider the quotient map $q : (\mathbb{R}^2 - \mathcal{O}) \to (\mathbb{R}^2 - \mathcal{O})/L$. The images of the trajectories τ_1 and τ_2 under the action of q are homologous in $(\mathbb{R}^2 - \mathcal{O})/L$ iff $\mathcal{H}(\tau_1) - \mathcal{H}(\tau_2) \in Q$.

Sketch of proof. First, note that the set Q is a countable set. Each element of Q corresponds to an element of the homology group $H_1(L; \mathbb{Z})$ [8]. The proof follows from the observation that by identifying L to a point under the quotient map, we essentially *trivialize* every closed loop (1-cycle) in L. This implies that the loops that were non-trivial in L before applying the quotient map (*i.e.* whose H-signatures were not zero), need to be set to zero when we compute and compare the H-signatures in the quotient space. Thus, before applying the quotient map we would say that $\tau_1 \approx \tau_2$ (*i.e.* belong to same homology class) iff $\mathcal{H}(\tau_1) - \mathcal{H}(\tau_2) = \mathbf{0}$. However, after applying the quotient map, each element of Q, containing the H-signatures of non-trivial loops in L, are to be considered equivalent to $\mathbf{0}$. Thus the new criteria becomes $q(\tau_1) \approx q(\tau_2)$ (*i.e.* the images of the trajectories belong to same homology class in the quotient space) iff $\mathcal{H}(\tau_1) - \mathcal{H}(\tau_2) \in Q$.

For a more formal algebraic proof and an illustration demonstrating the concept behind the proof, see Section 7 of [3]. $\square$

III. The Algorithm

In this section we provide a complete description of the algorithm for topological exploration with multiple robots while respecting the present constraints on the available space. We assume that the reader is familiar with the construction of the H-augmented graph [2] and the process of performing search (using Dijkstra's or A* algorithm) in it [5].

A. Representation

We discretize the environment (the subset of $\mathbb{R}^2$ that is of interest) into a uniform square grid and create a graph, $\mathcal{G}$, by placing a vertex in each square cell and connecting a cell with its neighbors using directed edges. More complex forms of discretization (triangulation, unstructured or adaptive discretization) can also be used. But to focus on the main contribution of the paper, we choose the simplest discretization scheme. We maintain a probability map by associating an occupancy probability with each cell. The initial probability for each cell in a completely unknown environment is set to 0.5, and the *state* of each cell is designated as 'unknown'. As the laser sensor data are received, the probability map is updated. If the probability of a cell goes above a high threshold, T_{obs}, we designate the cell as an 'obstacle'. Otherwise if it goes below T_{free}, we designate it as a 'free' cell. This, at any instant of time, gives us an obstacle map (see Line 3 of Algorithm 1: *ToplogicalExplore*).

A *candidate point* (an arbitrarily chosen point) is placed inside each connected component of the unknown region (a point is chosen near the boundary of the region, and shifted, if possible, to create a padding). Like *representative point*, the exact location of a candidate point is not of significance as long as it falls inside the desired region.

B. Multi-robot Exploration Algorithm

Suppose we start with N robots at a location, say $\mathbf{p}_0$, in the environment. At the beginning we have a single group of robots. The basic idea behind our algorithm is to split the group of robots based on the number of homology classes of trajectories discovered and deploy each newly-formed smaller group along those trajectories, and repeat this process for each subsequently formed group (Figure 3).

Discrete time is represented by t. The re-planning for trajectories does not happen in every time step, and instead happens at time steps $t_0, t_1, \cdots$. The values at the subscript of these time steps are the *planning cycle* numbers, and are denoted by the variable, $pl = 0, 1, 2, \cdots$.

At any instant, the groups formed by the robots are represented by a partition of the set of robot indices, $\{1, 2, 3, \cdots, N\}$. We represent that partition (created after *planning cycle*, *pl*) by the ordered set $G_{pl} = \left\{\{r_{pl}^{1,1}, r_{pl}^{1,2}, \cdots\}, \{r_{pl}^{2,1}, \cdots\}, \cdots\right\}$. A group, g, is simply a partition element $g \in G_{pl}$, and variables giving attributes to the groups are indexed by g (*e.g.*, τ_{pl}^g). $|G_{pl}|$ denotes the number of groups.

The planning cycle, *pl*, creates a set of trajectories, $\tau_{pl}^g,\ g \in G_{pl}$ (with H-signature, h_{pl}^g, w.r.t. base-point $\mathbf{p}_0$ – see Section III-B.1), that the groups need to follow. We will unambiguously (and without going into implementational details) refer to two obvious components of each such trajectory: *traversed part* and the *un-traversed part*.

Each group of robots, during their coordinated travel together as a group, has a representative location (a point in configuration space), with respect to which all computations of trajectories are performed. This point, representing the position of the group $g \in G_{pl}$ at time t (with $t_{pl} \leq t < t_{pl+1}$), is denoted as $\mathbf{P}_t^g$. On the contrary, the positions of individual robots are denoted by $\mathbf{p}_t^r,\ r \in \{1, 2, \cdots, N\}$ (and thus at the individual level of robot r, the control objective will be to reach $\mathbf{P}_t^g$, where $r \in g$). We represent the trajectory history of the g^{th} group at the time instant t by $\mathbf{P}_{0:t}^g$.

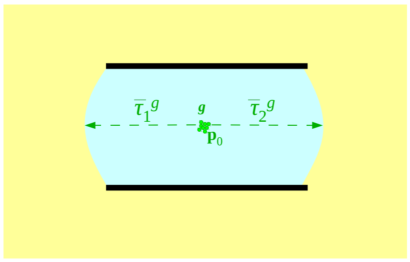
At $t = 0, pl = 0$, we start with a single group, $G_0 = \{\{1, 2, \cdots, N\}\}$. After obtaining the first few sets of laser sensor data and building the occupancy map in the neighborhood of the robot group, the algorithm *ToplogicalExplore* (Algorithm 1) is used to direct the exploration task. Figure 3 illustrates the working of the algorithm.

We use '$*$' in place of a index of a variable to denote the entire set of variables over all the possible indices (*e.g.*, $\tau_{pl}^* = \{\tau_{pl}^g \mid g \in G_{pl}\}$). An *overline* over a variable is used to emphasize that it is a temporary variable.

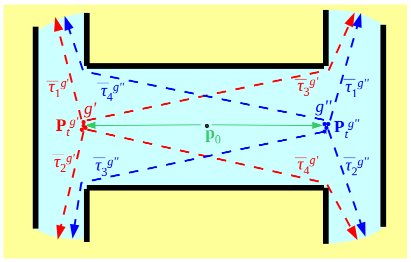
Algorithm 1: Pseudocode for *ToplogicalExplore*:

```
1.  $t = 0$; $pl = 0$; $t_{pl} = 0$; $G_{pl} = \{\{1, 2, \cdots, N\}\}$
2.  while TRUE
3.      i. Update probability map based on laser sensor data.
        ii. Threshold probability map to generate obstacle map.
4.      if $t == 0$  OR  map has changed significantly
                     OR  a group has reached its immediate goal
5.          i. Place representative points on newly discovered obstacles,
            ii. Place candidate points in connected components of
                   unexplored regions.
6.          for each $g \in G_{pl}$    // Plan new trajectories
7.              $\overline{\gamma}^g = \{\{\overline{\tau}_1^g, \overline{h}_1^g\}, \{\overline{\tau}_2^g, \overline{h}_2^g\}, \cdots\} =$
                                  $FindTrajectories(\mathbf{P}_t^g, \mathbf{P}_{0:t}^g)$
8.          end for each
9.          if $\overline{\gamma}^g = \emptyset,\ \forall g \in G_{pl}$    // No trajectory found. All explored!
10.             break while loop
11.         end if
12.         Set $G_{pl+1} = G_{pl}$    // Copy groupings from previous plan cycle.
13.         $\{\overline{H}^g \mid g \in G_{pl+1}\} =$
                $AssignHomologyClassesToGroups(c(\overline{\tau}_*^*), \overline{h}_*^*, h_{pl}^*)$
14.         $\{G_{pl+1}, \overline{\gamma}^*\} =$
                $CheckNearbyGroupsForRedistribution(\mathbf{P}_t^*, G_{pl+1}, \overline{H}^*)$

15.         for each $g \in G_{pl+1}$
16.             if $|\overline{H}^g| == 0$      // Group not assigned any homology class.
17.                 $\{G_{pl+1}, \overline{H}^*, \overline{\gamma}^*\} = RejoinWithClosestGroup(g)$
18.             end if
19.         end for each
20.         for each $g \in G_{pl+1}$
21.             if $|\overline{H}^g| > 0$      // Group assigned multiple homology classes.
22.                 $\{G_{pl+1}, \overline{H}^*, \overline{\gamma}^*\} = SplitGroup(g)$
23.             end if
24.         end for each
            // At this point each $\overline{H}^g,\ g \in G_{pl+1}$ contains one $H$-signature.
            // The new group structure is present in $G_{pl+1}$.
25.         for each $g \in G_{pl+1}$
26.             $\tau_{pl+1}^g = \overline{\tau}_k^g,\ h_{pl+1}^g = \overline{h}_k^g$,   $k$ is such that $\overline{h}_k^g \in \overline{H}^g$
27.         end for each
28.         $t_{pl+1} = t;\ pl++$
29.     end if
30.     for each $g \in G_{pl}$
31.         Choose the next point ($(t - t_{pl})^{th}$ point in $\tau_{pl}^g$), $\mathbf{P}_{t+1}^g \in \tau_{pl}^g$.
32.         $\mathbf{P}_{0:t+1}^g = \mathbf{P}_{0:t}^g \sqcup \mathbf{P}_{t+1}^g$
33.         for each $r \in g$
34.             Move robot $r$ towards $\mathbf{P}_{t+1}^g$ via the shortest path in the map.
                      // Controller for making robot follow planned trajectory.
35.         end for each
36.     end for each
37.     $t++$
38. end while
```



(a) At $t = t_0 = 0$ a planning cycle starts with a single group of N robots at $\mathbf{p}_0$. Thus $G_{t_0} = \{\{1, 2, \cdots, N\}\} =: \{g\}$. The group finds 2 topological classes of trajectories: $\{\{\overline{\tau}_1^g, \overline{h}_1^g\}, \{\overline{\tau}_2^g, \overline{h}_2^g\}\} = \mathit{FindTrajectories}(\mathbf{p}_0, \emptyset)$. Thus the group splits into two sub-groups, each containing $\sim N/2$ robots. The new groups are $g' = \{1, 2, \cdots, \lfloor N/2 \rfloor\}$ and $g'' = \{\lfloor N/2 \rfloor + 1, \cdots, N\}$ (see figure on the right), and they follow trajectories $\tau_1^{g'} := \overline{\tau}_1^g$ and $\tau_1^{g''} := \overline{\tau}_2^g$.



(b) At the beginning of the next planning cycle (at $t = t_1$) there are two groups: $G_{t_1} = \{g', g''\}$, when the condition in Line 4 of Algorithm *ToplogicalExplore* returns true. Thus, in this cycle of planning the groups obtain the following trajectories respectively:
$\mathit{FindTrajectories}(\mathbf{P}_{t_1}^{g'}, \tau_1^{g'}) = \{\{\overline{\tau}_1^{g'}, h_a\}, \{\overline{\tau}_2^{g'}, h_b\}, \{\overline{\tau}_3^{g'}, h_c\}, \{\overline{\tau}_4^{g'}, h_d\}\}$;
$\mathit{FindTrajectories}(\mathbf{P}_{t_1}^{g''}, \tau_1^{g''}) = \{\{\overline{\tau}_1^{g''}, h_c\}, \{\overline{\tau}_2^{g''}, h_d\}, \{\overline{\tau}_3^{g''}, h_b\}, \{\overline{\tau}_4^{g''}, h_a\}\}$.
Note the correspondence between the values of the H-signatures.

Fig. 3. Illustration of algorithm *ToplogicalExplore*.

In Line 4 of the above algorithm, the condition for checking whether the '*map has changed significantly*' consists of two checks: *i.* We first check if any of the most recently planned trajectories (*i.e.*, $\tau_{pl}^i,\ i \in G_t$) has become invalid (blocked by newly discovered obstacles), and *ii.* the number of cells in the environment that have changed state (*i.e.* from 'unknown' to 'free' or 'obstacle') is greater than a threshold.

Below are brief descriptions of each of the remaining subroutines used in the algorithm.

1) ***FindTrajectories***$(\mathbf{P}, \tau)$*:* (Refer to Figure 3) This subroutine is used to find all trajectories emanating from $\mathbf{P}$ in the different topological classes. The subroutine also returns the H-signature of the planned trajectory appended with the already traversed path, τ. This requires searching in the H-augmented graph, $\mathcal{G}_H$, as described in [2]. However, in the search algorithm we *initiate* the *open set* with the vertex $\{\mathbf{P}, \mathcal{H}(\tau)\}$ (*i.e.*, instead of using $\mathbf{0}$ as the H-signature of the start vertex, we use $\mathcal{H}(\tau)$ – the H-signature of the traversed path, τ). Consequently we expand the vertices in $\mathcal{G}_H$ as usual. This ensures that we consider $\mathbf{p}_0$ as the *base point* of the space so that the value of the H-signature remains consistent over the different groups and over time (see Figure 3(b)). Vertices that lie in the explored region are expanded, and a path is stored every time a vertex connected to the unknown region is reached via a new homology class (identified by the sum of the H-signature of the expanded vertex and the H-signature of a trajectory connecting that vertex with the candidate point in unknown region).

Note that according to Proposition 1, the way we determine whether H-signatures h and h' represent the same homology class in the quotient space is to check if the elements of the difference, $h - h'$ (which, recall from the definition of H-signature, is a vector of complex numbers), are either ***i.*** all equal when the unknown region is not simply connected (*i.e.*, the unknown region that extends to the boundary of the environment), or, ***ii.*** all zero when the unknown region is simply connected (for all other unknown regions). If none of these is true, they represent different homology classes. Using a method similar to [2], we do not allow trajectories that loop around obstacles. Moreover we do not place *representative points* on obstacles smaller than a threshold radius, thus avoiding multiplicity of topological classes merely due to sensor noise. Arguably, this subroutine is the most computationally intensive since it involves searching in the H-augmented graph. The worst complexity is that of a Dijkstra's algorithm (for a graph of almost-constant *degree*): $O(v \log(v))$, where v is number of vertices in the graph created by discretization of the environment.

2) ***AssignHomologyClassesToGroups****:* The number of trajectories returned by the '*FindTrajectories*' procedure will be the same for each of the groups $g \in G_{pl}$ (see Figure 3(b)). Since we used the same base-point, $\mathbf{P}_0$, for the searches for each group, we will obtain the same set of H-signatures for each group from the search in Line 7, although the trajectories will of course be different.

The purpose of this subroutine is to make the assignment of each of the homology classes to the different groups of robots based on the cost of the planned trajectories, $c(\overline{\tau}_*^*)$, their H-signatures, $\overline{h}_*^*$, and the H-signature of the trajectories assigned in the last plan cycle, h_{pl}^*. The basic strategy for doing this is as follows: *i.* If, for a group g, the H-signature of the last planned trajectory, h_{pl}^g, that it has been following, is found in the result returned by *FindTrajectories*, that homology class is assigned to the group g (the H-signature comparison being made with respect to obstacles that are common to the time instants when the last plan was made and the current time). This ensures that a group (or one of its subgroups) keep following the homology class that it has been following. *ii.* Whichever homology class remain unassigned after this is assigned to group for which the trajectory corresponding to the class is shortest. The H-signatures of the homology classes assigned to group $g \in G_{pl+1}$ is fixed in $\overline{H}^g$ (*i.e.*, it is a set of H-signatures, $\overline{H}^g = \{\overline{\eta}_1^g, \overline{\eta}_2^g, \cdots\}$).

3) ***CheckNearbyGroupsForRedistribution****:* If a group has been assigned homology classes more than the number of robots available in that group (*i.e.*, $|\overline{H}^g| > |g|$), then it is checked if there is another *nearby* group, g', such that $c|\overline{H}^{g'}| < |g'|$, $\|\mathbf{P}_t^{g'} - \mathbf{P}_t^g\| < R$ ($c > 1, R > 0$ are parameters). If so, a re-shuffling of the groups is performed (with dome robots from g' being transferred to g) and the new group arrangement is returned to G_{pl+1}. Since the content of each group gets changed, the indices of $\overline{\gamma}^*$ are updated accordingly.

4) ***RejoinWithClosestGroup:*** This subroutine gets triggered when a group is not assigned any homology class. The reason for this is typically two-fold: *i.* Sometimes a spurious homology class may be observed because of incorrect laser readings, which would soon turn out to be blocked as new sensor data arrives, thus resulting in some of the recently created group to be assigned no trajectories, or, *ii.* a group can reach a dead-end in the environment (*e.g.*, end of a corridor). This requires that we rejoin those groups with other groups so that they don't remain idle. We first look for closest "cousin" groups (groups having common distant parent – group at an earlier plan cycle from which the current groups originated – see $SplitGroup$ next) that are not more than D generations apart. This requires a traversal of D levels of the family tree (the sets $G_{pl}, G_{pl-1}, G_{pl-2}, \cdots, G_{pl-D}$ contain all the information required for this) and identification of the closest *cousin*. If such a cousin cannot be found, the group is joined with the distance-wise closest group in the environment. The subroutine returns the new grouping (*i.e.* partition of the set $\{1, 2, \cdots, N\}$) and the corresponding reordering that is required in $\overline{H}^*$. Since $\overline{\gamma}^*$ and $\overline{H}^*$ are indexed by the groups, an update of their indices is also required (and removal of the elements corresponding to the joined, hence no-more existing, group).

5) ***SplitGroup:*** If $\overline{H}^g$ contains more than one element (*i.e.* multiple homology classes assigned to a single group of robots), the group will be split into sub-groups of almost-equal sizes and at most one homology class will be assigned to each of the sub-groups. Thus, if there aren't enough robots in the group (*i.e.* $|\overline{H}^g| > |g|$), clearly a choice has to be made and some of the homology classes has to be left unattended for future exploration. Under such situations the unattended homology classes are removed from $\overline{H}^g$. As before, the indices of $\overline{\gamma}^*$ and $\overline{H}^*$ are updated.

C. Distributed Implementation

It is to be noted that the algorithm *ToplogicalExplore* can be implemented in a distributed manner where the i^{th} group performs its own computation for the robots in the group. In a distributed implementation the 'for each' loops starting at Lines 6, 15, 20, 25 and 30 would be replaced by computation for the respective group only in their respective threads. Each group would maintain its own probability map and update it based on the laser sensor readings. Each group also broadcasts the changes in its own map so that the other groups in the environment can update their maps (a communication protocol similar to that in [4]). Moreover, when one group decides that a re-planning of trajectory is required (condition in Line 4 becomes true), all the groups are communicated the decision and they come to a consensus to re-plan. Since the procedure $AssignHomologyClassesToGroups$ requires a consensus, the groups communicate the cost of their respective planned trajectories as well.

IV. Results

We implemented the *ToplogicalExplore* algorithm on ROS (Robot Operating System), that lets us accurately simulate robot dynamics, actuator noise and sensor noise. Although our current implementation is mostly centralized and runs on a single processor, the overall structure of the algorithm is perfectly suited for distributed implementation on multiple parallel processors as described in Section III-C.

We also provide extensive comparison with the frontier-based algorithm described in [18] (the implementation of which was also made in ROS, with identical models of robot dynamics, sensor and actuator). Section IV-A illustrates, using a simple environment, why our algorithm logically outperforms a frontier-based algorithm. Section IV-B demonstrates similar performance comparison for a more complex indoor environment.

All simulations were run on a dual core machine with processor clock speed of 2.6GHz and 4GB memory. Note that the run times reported involve the complete dynamic simulation of the non-holonomic robots.

A. Partially Known Environment

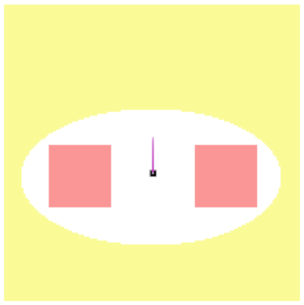
We consider a simple partially known environment that is 30m $\times$ 30m in size, discretized by 0.1m $\times$ 0.1m cells, with 4 robots exploring it. The environment has 3 rectangular obstacles, of which two fall inside the initially known elliptical region as shown in Figures 4(a) and 4(c). The initial known region, as clearly seen, is not simply-connected. Consequently, the number of topological classes do not correspond to the number of frontiers. Thus, using a frontier-based algorithm (as described in [18]) the entire group of 4 robots are driven towards the single frontier as shown in Figure 4(a). However, using our topological exploration, the initial group of robots discover two topological classes of trajectories and hence split up into two sub-groups as seen in Figure 4(c). This, without surprise, results in more efficient exploration of the environment. Our $TopologicalExplore$ algorithm explores the entire environment in 1045 iterations (and actual run time of $\sim$ 35mins), while the frontier-based algorithm took 2359 iterations (and run time of $\sim$ 78mins).

This example illustrates how a topological approach to exploration, as ours, visibly and structurally outperforms standard frontier-based approaches in cases when the known environment is not simply-connected.

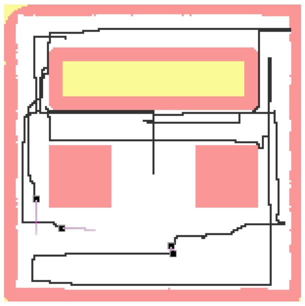
B. Simulations of Multi-robot Topological Exploration

Figure 6 shows an example with eight robots. The environment used is a part of the 4^{th} floor of the Levine hall at the University of Pennsylvania (a 21.3m $\times$ 34.2m environment, discretized by 0.1m $\times$ 0.1m cells). In Figure 6(a) the single group of robots discovers two topological classes, and hence splits into two groups, each consisting of four robots (Figure 6(b)). In Figure 6(c) each of those groups get assigned two topological classes to discover, thus each splitting further into groups of two robots (Figure 6(d)). Further splitting of three of those groups happen in Figure 6(e). Following which, as some of the groups end up exploring the homology classes assigned to them, they rejoin the other groups to help explore whatever remains.

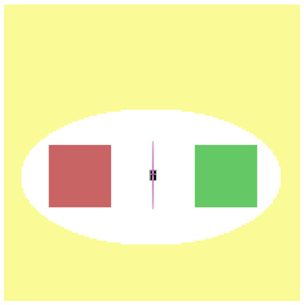
Figure 6(i) shows comparison with the final result obtained using the frontier-based approach of [18]. Even in this case not only the number of iterations required using the



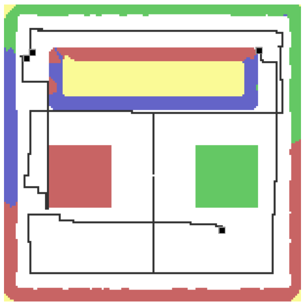
(a) $t = 3$: A *frontier-based exploration* algorithm initially finds a single frontier and plans a trajectory to drive all the robots towards it.



(b) Using the *frontier-based algorithm* the robots explore the entire environment in $t = 2359$ iterations.



(c) $t=3$: Our $TopologicalExplore$ algorithm finds 2 topological classes of trajectories and hence splits the group of robots into two.



(d) Using $TopologicalExplore$ algorithm the robots explore the entire environment in $t = 1045$ iterations.

Fig. 4. Comparison between the frontier-based exploration algorithm (top row) of [18] and our $TopologicalExplore$ algorithm (bottom row) in a partially-known environment using 4 robots. The purple curves show parts of the planned trajectories, while black represents traversed trajectories. White is known/explored, while light yellow is the unknown region.

frontier-based approach is higher, the actual time required for computation was also higher in case of the frontier-based exploration. The frontier-based approach took ~ 100mins, while our algorithm took ~ 69mins to completely explore this particular environment.

C. Experiment with a Single Robot

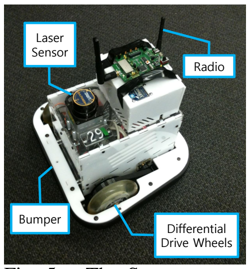


Fig. 5. The SCARAB mobile robot platform [10]

To demonstrate practical applicability we implemented our algorithm on a mobile robot platform developed in the GRASP laboratory and known as the SCARAB [10]. Figure 5 illustrates the various components of the experimental platform and a snapshot of the robot in action. To localize the robot we currently use an adaptive Monte Carlo localization [7] module that relies on laser sensor data. Having multiple robots would not only require an additional local collision check layer, but also an additional complexity for localization.

The overall *ToplogicalExplore* algorithm, even when there is a single robot, remains the same. The key feature during the execution, however, is that we always have a single group of robots consisting of a single robot (*i.e.*, $G_{pl} = \{\{1\}\}$), and whenever the $SplitGroup$ subroutine is called, the group/robot has to choose one of the trajectories.

We performed the single-robot experiment in the same indoor environment (the blue-print of which we used to perform the multi-robot simulations). However, we sealed the two entrances at lower left and lower right leading to the larger room at the bottom. Figure 7 shows the result. In Figure 7(a), the robot starts from the bottom left corner and explores the environment to initially find three topological classes. The robot follows one of these trajectories that lead to the frontier 2 in Figure 7(a) (In the figures we number the frontiers for convenience of referencing. It should however be noted that at no point in our algorithm do we need to compute or identify the frontiers or its connected components). When the robot finds further branches, it keeps on following the trajectory with the current H-signature (thus, for example, reaching frontier 2 in Figure 7(b)). When there are no more feasible trajectories with the current H-signature (*e.g.*, the frontier 2 disappears in Figure 7(c)), the robot starts following the shortest trajectory with a new H-signature to a new frontier (*e.g.*, frontier 5 in Figure 7(c)). This process continues until there are no frontiers left, hence completing the process of building the map (Figure 7(e)).

V. Conclusions and Future Direction

In this paper we have presented an algorithm to explore an unknown or partially known environment by gradually building a topological description of the environment. Using the notion of quotient spaces, optimal trajectories in different topological classes leading up to the unknown region were found by searching in the H-augmented graph. Groups of robots are split into subgroups with each subgroup being assigned to a different homology class to enable efficient exploration of the environment. In contrast to previous work, the exploration is guided by topological and not metric information about the world and is ideally suited to obtaining a coarse topological map without detailed metric information. We demonstrated the performance of our algorithm in simulation using multiple robots, and in experiment using a single robot. We also provided a comparison of performance between our algorithm and a frontier-based approach. We are in the process of creating a distributed implementation of the algorithm and in near future plan to conduct experiments with more than one robot.

Acknowledgment

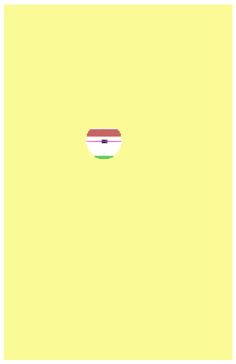
We gratefully acknowledge support from the ONR Antidote MURI project, grant no. N00014-09-1-1031. We would also like to thank Mr. Benjamin Charrow for his help with setting up and resolving technical issues with the SCARAB robots and the Monte Carlo localization module.

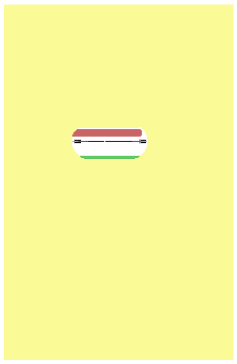
References

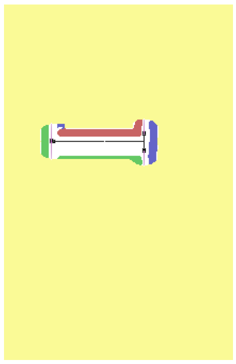
[1] Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Multi-robot coverage and exploration in non-euclidean metric spaces. In *Proceedings of The Tenth International Workshop on the Algorithmic Foundations of Robotics*, 13-15 June 2012.

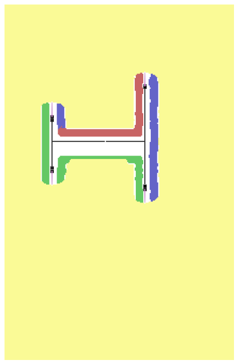
[2] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, pages 1–18, June 2012. DOI: 10.1007/s10514-012-9304-1.

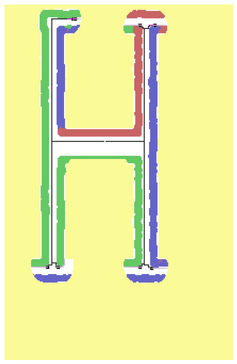
[3] Subhrajit Bhattacharya, David Lipsky, Robert Ghrist, and Vijay Kumar. Invariants for homology classes with application to optimal search and planning problem in robotics. *Annals of Mathematics and Artificial Intelligence*, 67(3-4):251–281, March 2013.

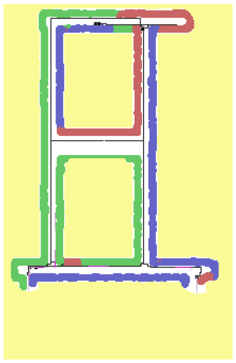

(a) $t = 4$.

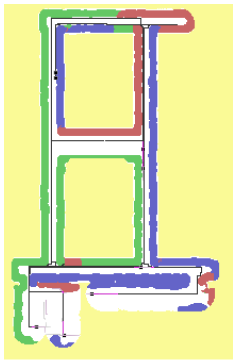

(b) $t = 136$.

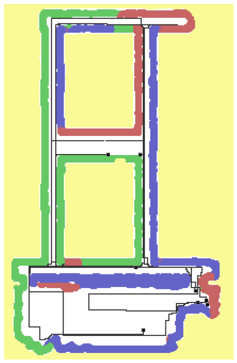

(c) $t = 186$.


(d) $t = 275$.


(e) $t = 580$.


(f) $t = 891$.


(g) $t = 1180$.


(h) $t = 1790$ (exploration complete).

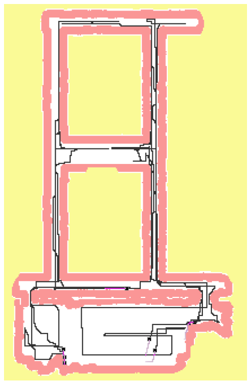
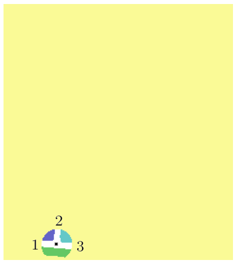
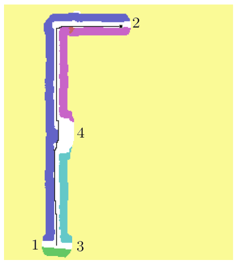

(i) *Comparison:* Final result ($t = 2638$ iterations) using frontier-based approach of [18].

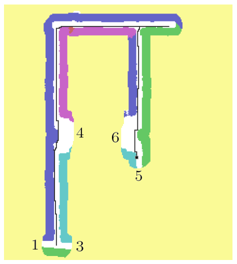
Fig. 6. (a)-(h): Simulation result with 8 robots exploring an indoor office-like environment. (i): Comparison of performance with frontier-based algorithm of [18] (in the same environment, with same number of robots and same initial configurations).



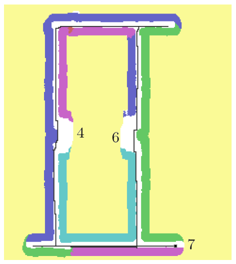
(a) $t = 1$.



(b) $t = 310$.



(c) $t = 540$.



(d) $t = 900$.

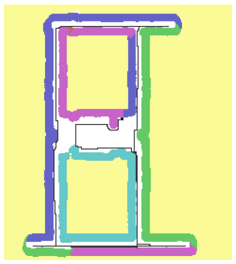

(e) $t = 1226$.

Fig. 7. Experiment result with a single robot exploring an indoor office-like environment.

[4] Subhrajit Bhattacharya, Nathan Michael, and Vijay Kumar. Distributed coverage and exploration in unknown non-convex environments. In *Proceedings of 10th International Symposium on Distributed Autonomous Robotics Systems*. Springer, 1-3 Nov 2010.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2nd edition, 2001.

[6] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325 –1339, july 2006.

[7] Brian P. Gerkey. amcl ros package. http://www.ros.org/wiki/amcl.

[8] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

[9] John G. Rogers III, Carlos Nieto-Granda, and Henrik I. Christensen. Coordination strategies for multi-robot exploration and mapping. *Experimental Robotics*, 88:231–243, 2013.

[10] N. Michael, J. Fink, and V. Kumar. Experimental testbed for large multi-robot teams: Verification and validation. *Robotics and Automation Magazine*, 15(1):53–61, Mar 2008.

[11] James Munkres. *Topology*. Prentice Hall, 1999.

[12] Richard Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(10):1016–1030, October 1999.

[13] M. Schwager, P. Dames, D. Rus, and V. Kumar. A multi-robot control policy for information gathering in the presence of unknown hazards. In *International Symposium on Robotics Research*, Aug. 2011.

[14] C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Freiburg, Germany, April 2006.

[15] C. Stachniss. *Robotic Mapping and Exploration*. Springer Tracts in Advanced Robotics. Springer, 2009.

[16] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robot.: Sci. and Syst.*, pages 65–72, Cambridge, MA, June 2005.

[17] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[18] Arnoud Visser and Bayu Slamet. Balancing the information gain against the movement cost for multi-robot frontier exploration. In *Second European Robotics Symposium 2008, EUROS 2008, Prague, Czech Republic*, pages 43–52, 2008.

[19] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1160–1165, 2008.

[20] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146 –151, jul 1997.