Exploring Topological Worlds with Minimal Information

Hui Wang, Michael Jenkin and Patrick Dymond

Technical Report EECS-2016-05

December 9 2016

Department of Electrical Engineering and Computer Science
4700 Keele Street, Toronto, Ontario M3J 1P3 Canada

Exploring Topological Worlds with Minimal Information

Hui Wang, Michael Jenkin and Patrick Dymond

Abstract

What does it take to solve the Simultaneous Localization and Mapping (SLAM) problem deterministically? Dudek et al. [1] demonstrated that a single unique movable marker was sufficient. But is this necessary? Here we show that while a unique marker providing only position or only orientation is not sufficient to enable a deterministic solution, a marker that provides both position and orientation is. Such "directional lighthouse" information can be provided in a number of ways including through the addition of a single directional immovable marker in the environment.

I. Introduction

Although there is a wide range of existing algorithms for Simultaneous Localization and Mapping (SLAM), two main classes of algorithms have emerged: those that treat the problem as constructing a representation embedded in a geometric or metric space and those that treat the problem as constructing a topological or graph-based representation [2]. In a topological approach, the world and its corresponding representation are modeled as a graph embedded within some space in order to provide local orientation information to the edges incident on a given vertex. Each location in the world is modeled as a vertex of the graph, and locations are connected by undirected edges. Within a topological formalism, the goal of a SLAM algorithm is to generate a graph-like map representation that is isomorphic to the underlying world being explored by the agent (robot) and to determine the agent's state within this graph. Robot(s) explore the world/graph by starting at some vertex and following unexplored edges to 'new' locations. Each new location encountered must be added to the map, but before this is done, the robot must ensure that the supposedly new location is truly distinct from locations encountered before. This is perhaps the key problem in SLAM and is typically referred to as the 'loop closing' or 'have I been here before' problem.

It is straightforward to show that lacking any additional information it is not possible for an agent to solve SLAM deterministically in an embedded graph (see [1]) but that the problem can be solved deterministically if the robot is equipped with an appropriate marking aid. Using marker(s) to help deal with uncertainty in exploration has been investigated for many years including early work described in

[3] and [4]. Various types of markers are examined in the literature. Previous work including [1], [5] and [6] examined the power of *movable* marker(s) in exploring undirected topological worlds. It was shown in [1] that a single undirected movable marker is sufficient to solve the SLAM problem deterministically. That work solves the loop closing problem using a single undirected marker that is dropped and picked up at vertices. This marker is used to provide both *place validation*, which determines if two locations are distinct, and *back-link validation* that determines the relative embedding (orientation) of a given vertex. In order to determine if two map estimates correspond to the same real world location, the marker is dropped in one of the map locations and then the other is visited. If the marker is found then the two map estimates correspond to the same real world location, otherwise they do not. If they correspond to the same physical location then the relative orientation of the two maps must be solved, and in [1] this solution again involves moving and placing the marker. Is there a need for the marker to provide both position and orientation information? Here we show that while an immovable marker providing only position or orientation information in a vertex is not sufficient, mapping is solvable if unique position and orientation information exist in one vertex of the world. Such information can be established, for example, through the use of one *directional* immovable marker. This work expands on preliminary results appearing in [7], [8]. The preliminary results are generalized here. A proof of correctness and complexity analysis are presented. In addition, different mechanisms for establishing a directional lighthouse are described.

II. WORLD AND ROBOT MODEL

This work adopts the topological world and robot model introduced in [1] and used in [5], [6], [7], [8], [9].

A. The World Model

The world is modeled as an embedding of an undirected graph $G = (V, E)$ with a set of vertices $V = \{v_0, ..., v_{n-1}\}$ and a set of edges $E = \{(v_i, v_j)\}$. G is an unlabelled graph in which the vertices and edges of G are not necessarily uniquely distinguishable to the robot by using just its sensors. The graph is embedded within some space in order to permit relative directions to be defined on the edges incident upon a vertex. The definition of an edge is extended to allow for the explicit specification of the order of edges incident upon each vertex of the graph embedding.

B. Robot motion and edge-related perception

Assume that the robot can leave a vertex by a given door (edge) of the vertex, can move between vertices by traversing edges, and can identify when it is on an edge or when it has arrived at a vertex. At a vertex, the robot can enumerate the edges by following the pre-defined ordering convention, thus determining the relative ordering among the edges that are incident on the current vertex in a consistent manner. The robot can identify the edge through which it entered a vertex and assign a relative label (index) to each edge in the vertex, representing the current local edge ordering at the vertex. Note that this local edge ordering is not, in general, equal to the ordering specified by the embedding (which is not accessible to the robot), but rather is a permutation of it. For ease of exposition, in this work a planar embedding and a simple clock-wise enumeration rule are assumed in two-dimensional examples. Under this assumption, two edge orderings at a vertex are cyclic shifts of each other.

C. Markers and marker-related perception

We consider immovable markers. Such a marker attaches itself to the location where it is dropped. The marker can be directional. A directional marker in a vertex points to one of the incident edges of the vertex. Upon entering a vertex, the robot can sense the presence of the marker if the marker is present at the current vertex. The robot can also sense the direction of a directional marker, that is, the robot can identify the edge that is pointed to by the marker. By enumerating the edges and identifying the marked one, the robot can distinguish between the different edges at the vertex, and thus obtain relative orientation information at the vertex.

III. Is deterministic SLAM possible with a single unoriented marker or with only absolute orientation?

Dudek et al. showed in [1] that a single unoriented (undirected) movable marker was sufficient to map an embedded topological world. Is this mapping possible with a single unoriented *immovable* marker? First note that as shown in [1], given the world and robot model as described above, a robot lacking any position and orientation information is *not* able to map its environment deterministically. Consider a robot operating either in a cycle of length three or four. In each of the cycle graphs all vertices have the same degree. Moreover, the degree information of vertices connected to any vertex are all the same. All of the vertices thus appear identical to the robot even if the degree information (or signature [10]) of arbitrarily

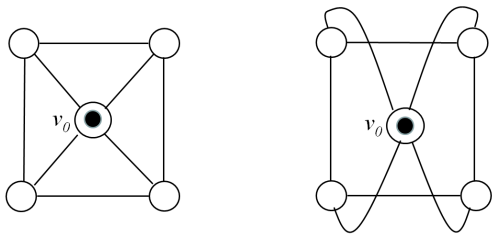


Fig. 1. Two different embedded graphs that are not distinguishable with unique position information. Assume a clockwise enumeration convention, and that the robot starts from the marked vertex facing one of the edges. Same motion sequences always result in same perceptions in both graphs. E.g., motion sequence $\mathcal{M} = \{1, 1, 2, 2, 1\}$, which denotes 'take the 1st edge on the left of the current orientation, upon arrival take the 1st next edge on the left of the entry edge .....') results in perceptions $\mathcal{P} = \{$[3,A][3,A][3,A][3,A][4,P]$\}$ on both the graphs, where [3,A] denotes that the degree of the current vertex is 3 and the marker is not present, [4,P] means the degree is 4 and the marker is present.

large neighborhoods is considered. In exploring the graphs, the robot always observes a non-terminating sequence of '2-door rooms'.

Can a robot map an arbitrary graph-like world deterministically with unique position information only? Suppose that the environment contains a uniquely marked vertex where position information exists due to the undirected immovable marker in it, but no orientation information is available. Under this assumption, while the robot can easily distinguish graphs such as different sized cycles, there exist embedded graphs that the robot cannot distinguish deterministically. Consider the two different embedded graphs shown in Fig. 1, which are not isomorphic to each other under the extended definition of isomorphism given in [1]. Assume that the robot is initially at the marked place and faces one of the edges. It can be shown that if the robot executes the *same* motion sequence on the two graphs, the perceptions it acquires on the different graphs are exactly the *same* [11]. Thus the robot cannot tell the two graphs apart.

Can a robot map an arbitrary graph-like world deterministically with some orientation information but no mechanism providing absolute position information? Suppose on a cycle graph the robot has global orientation information but no position information so it can determine its orientation (entry edge) at each vertex but not its location. Clearly the robot still cannot distinguish between different sized cycles.

IV. Mapping with both position and orientation information

Given that a single undirected movable marker is sufficient to map a topological environment deterministically, but that neither a single unique position information (e.g., in a location marked with an undirected immovable marker) nor absolute orientation is sufficient, what is the minimal marker necessary to map the world? Here we show that if the world contains a vertex that provides a unique signature that

also provides absolute orientation, then the world can be mapped deterministically. More specifically, we demonstrate that a unique *directional lighthouse* vertex is sufficient to enable deterministic SLAM. Here, a directional lighthouse vertex is a vertex that provides both a unique marking (this is a unique location in the environment) and that at the same time provides absolute orientation information at the vertex.

The basic approach is outlined in Algorithm 1. Assume that the robot starts the exploration from the directional lighthouse vertex (call it v_0). This can be accomplished by marking this location with a single marker that provides directional information, or in a number of other ways. But for the purpose of exposition assume that a unique *directional* immovable marker exists at this location. Following the same basic approach as [1], the algorithm constructs a map of the environment deterministically, maintaining S, the set of mapped verticies and edges, and U, the set of known but not yet mapped edges extending from verticies in S. Initially the robot enumerates incident edges at v_0 and sets edge labels (on map S) based on the enumerated edge ordering. v_0 is the initial definition of S, and incident edges at v_0 are the initial elements of the unexplored edge set U. After this initialization, each step of the algorithm involves selecting (and removing) an unexplored edge e from U, having the robot traverse S to the known end v_k of e and then following e to the unknown end v_u (Fig. 2(a)). We need to determine if v_u corresponds to a vertex already present in the map (place validation), and if so by which edge it entered the vertex (back-link validation). Each known vertex $v_{k'}$ could potentially correspond to v_u if 1) it has the same signature (degree and presence/absence of the marker) as v_u, 2) it has unexplored edge(s). Each unexplored edge incident on such a vertex could potentially correspond to e. In [1], a single undirected movable marker is used to perform place validation first and if a loop was formed then the marker is used again for back-link validation. With only an immovable marker, here place and back-link validations are conducted *simultaneously* by disambiguating e and v_u against other unexplored edges currently in U and their known ends. Specifically, each unexplored edge $e' = (v_{k'}, v_{u'})$ in U along with its known end vertex $v_{k'}$ is considered as a potential loop-closing hypothesis if $v_{k'}$ has the same signature as v_u. That is, it is hypothesized that $e = (v_k, v_u)$ and $e' = (v_{k'}, v_{u'})$ correspond to the same edge and thus the robot has entered $v_{k'}$ from v_k via e/e'. If no such hypothesis exists, the algorithm moves on to the augmentation stage for e as explained later. Otherwise, the hypothesis validation process for e starts. For each hypothesis (edge) $e' = (v_{k'}, v_{u'})$, a simple path (i.e., path with no repeated vertices) $v_{k'}, ..., v_0$ in S is computed. The path is represented as a motion sequence $\mathcal{M}_{e'}$ consisting of a sequence of relative edge orderings (relative to the entry edge) at each vertex along the path. Hypothesizing that the robot is in

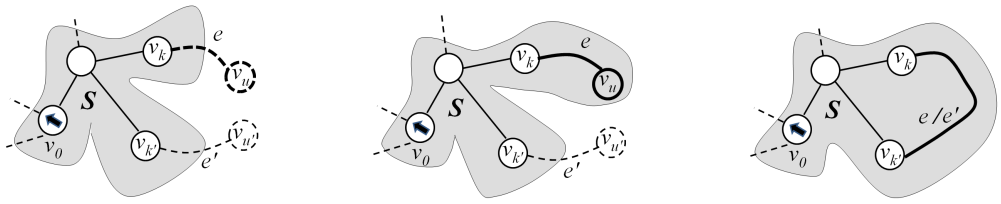
Algorithm 1: Mapping with a unique position & orientation info

Input: location v_0 in G marked with a directional marker
Output: a map representation S that is isomorphic to world G

$S \leftarrow \{v_0\}$; $U \leftarrow$ incident edges in v_0; `// initial S & U`;
while *U is not empty* **do**
  remove an unexplored edge $e = (v_k, v_u)$ from U;
  the robot traverses S to v_k and then follows e to v_u;
  the robot senses the signature (degree, marker presence and direction) at v_u;
  $H \leftarrow$ a set of loop closing hypotheses (edges) in U whose known end vertices have the same signature as v_u;
  while *H is not empty* **do**
    $e' = (v_{k'}, v_{u'}) \leftarrow$ a (removed) hypothesis in H;
    compute motions $\mathcal{M}_{e'}$ for a simple path $v_{k'}, ..., v_0$;
    compute the expected perception $\mathcal{P}^E_{e'}$ along the path;
    the robot tries to traverse the path by executing $\mathcal{M}_{e'}$;
    based on perceptions $\mathcal{P}_{e'}$ obtained in executing $\mathcal{M}_{e'}$ **do**
      case *(1) or (2) – $\mathcal{P}_{e'}$ and $\mathcal{P}^E_{e'}$ does not match*
        reject the hypothesis;
        the robot retraces $\mathcal{M}_{e'}$, coming back to v_u and resuming original orientation at v_u;
      case *(3) – $\mathcal{P}_{e'}$ and $\mathcal{P}^E_{e'}$ match throughout*
        confirm the hypothesis and exit inner **while** loop;
  `// now do augmentations on S`;
  if *a hypothesis $e' = (v_{k'}, v_{u'})$ is confirmed* **then**
    `// loop augmentation`;
    add edge $e/e' = (v_k, v_{k'})$ to S; remove e' from U;
  else `// no hypothesis, or all were rejected;`
    `// non-loop augmentation`;
    add e and v_u to S; add other edges in v_u to U;
return S;

$v_{k'}$ and is oriented against e', the sequence includes the ordering at $v_{k'}$ (relative to the known ordering of e' at $v_{k'}$). The expected perception $\mathcal{P}^E_{e'}$ that the robot should obtain along the execution of $\mathcal{M}_{e'}$ is also computed. $\mathcal{P}^E_{e'}$ consists of a sequence of vertex signatures along the path, including the expected marker presence and direction (in terms of the relative ordering between the marked edge and the entry edge) at the end vertex v_0. The key to the correctness of the validation process is the fact that for each hypothesis e', $\mathcal{M}_{e'}$ along with $\mathcal{P}^E_{e'}$ define an embedded path $v_0, ..., v_{k'}, v_{u'}$ in S, which *uniquely* identifies the hypothesized location and orientation (hypothesis e' and its known end $v_{k'}$). Other hypotheses may have the same motion sequence as e' or may have the same expected perception as e', but not both. Formal justification for this is given in Section V.

Hypothesized to have entered $v_{k'}$ via e', the robot then validates the hypothesis by attempting to execute



(a) Traverse to v_u via e

(b) No hypothesis is confirmed. v_u is new

(c) Hypothesis e' is confirmed. v_u is $v_{k'}$

Fig. 2. Basic exploration step of the directional immovable marker algorithm. Must determine if $e = \{v_k, v_u\}$ corresponds to a loop or if v_u is an unvisited vertex. e' is a loop closing hypothesis. S is augmented in (b) and (c).

$\mathcal{M}_{e'}$, which, if the hypothesis holds, would allow the robot to complete the traversal and obtain a perception $\mathcal{P}_{e'}$ that matches $\mathcal{P}^E_{e'}$. That is, the robot starts from $v_{k'}$ and arrives at v_0 via the expected entry edge. Based on $\mathcal{P}_{e'}$, the algorithm distinguishes three cases:

(1) A mismatch between $\mathcal{P}_{e'}$ and $\mathcal{P}^E_{e'}$ is observed during execution of $\mathcal{M}_{e'}$. In this case the marker is encountered at some point along the execution of the path prior to completion, or, the path cannot be followed completely due to mismatch between the sensed degree of the physical vertex and the expected degree of the vertex on the planned path.

(2) A mismatch between $\mathcal{P}_{e'}$ and $\mathcal{P}^E_{e'}$ is observed at the end of $\mathcal{M}_{e'}$. The path is completed but upon completion, the marker is not present, or it is present but the marker-pointed edge does not have the expected orientation (ordering) relative to the entry edge.

(3) $\mathcal{P}_{e'}$ and $\mathcal{P}^E_{e'}$ matches throughout execution of $\mathcal{M}_{e'}$. That is, the path is completed and upon completion of path execution, the marker is present and the marker-pointed edge has the expected orientation relative to the entry edge.

In case (1) the hypothesis is rejected. If the hypothesis holds then the robot should be able to follow the planned path $v_{k'}, ..., v_0$, and as the path is a simple path to v_0, the robot should not encounter the marker prior to v_0. The hypothesis is also rejected in case (2). In this case the robot did not arrive at v_0 or did not arrive from the correct entry edge. Once a hypothesis e' is rejected, the robot executes the reverse of $\mathcal{M}_{e'}$, coming back to v_u and resumes the original orientation at v_u. It then validates the next hypothesis for e (if any). In case (3) the hypothesis is confirmed and no other hypotheses of e will be tested. The validation process for e thus terminates either when a hypothesis is confirmed, or when all the hypotheses have been tested and rejected. Then the algorithm moves on to the augmentation stage.

If no hypothesis exists or all the hypotheses of e are rejected, then the unknown place v_u does not correspond to any known vertex in S and is added to S as a new vertex. Edge e is also added to S as

an explored edge, augmenting S by one edge and one vertex (non-loop augmentation. Fig. 2(b)). Other edges incident on v_u are added to U. The algorithm is free to set the ordering (labels) of e and the other incident edges at this new vertex, representing the enumerated edge ordering at the new vertex (e.g., e is the 0'th edge and the others are ordered accordingly). If a hypothesis $e' = (v_{k'}, v_{u'})$ is confirmed (case 3), then v_u corresponds to the known vertex $v_{k'}$ (place validation) and e corresponds to the incident edge e' at $v_{k'}$ (back-link validation). In this case S is augmented with a new edge $e/e' = (v_k, v_{k'})$ as shown in Fig. 2(c), using the index of e at v_k and index of e' at $v_{k'}$ as the index of e/e' at v_k and $v_{k'}$ respectively. e' no longer represents an unexplored edge in the world so it is removed from U. The algorithm then proceeds with a newly selected edge from U, and terminates when U is empty. We justify in Section V that when U is empty, the map S is isomorphic to the world G. Note that if the orientation information is established in other manners, e.g., by having one neighbor of v_0 contain two undirected markers, then this algorithm only requires minor modification in order to retrieve the orientation information (by taking extra traversals to the marked neighbor). The validation process remains the same.

V. Correctness sketch of the algorithm

Following the proof sketch given in [1], here we prove that when the algorithm terminates, the maintained map S is isomorphic to the world model G. We use the extended definition of graph isomorphism, under which, map S and real world model G are said to be isomorphic if and only if they are isomorphic under the usual definition of graph isomorphism ([12]), and in addition, for each vertex v of S and each edge e leaving v, $index(e, v) = relabelling(index(\phi(e), \phi(v)))$ where ϕ denotes the mapping from map S to the world model G, i.e., $\phi(v)$ and $\phi(e)$ denote the vertex and edge in G to which v and e in S correspond (represent). $relabelling$, which states that the edges leaving v and $\phi(v)$ have the same labeling (follow the same pre-defined ordering convention) but may have different reference edges, is referred to below as the 'edge-index condition'.

We prove the algorithm correct by establishing an invariant **I** and showing that **I** is initially true, is maintained true throughout execution, and that the algorithm terminates. Then we show that the termination condition plus the invariant imply that map S is isomorphic to the world model G. We define **I** as follows:

I-1 S is isomorphic to G_s, which is the explored subgraph of the real world model G

I-2 U contains a set of edges that may be bijectively mapped to the set of unexplored edges in G that have at least one incident vertex in G_s, with edge indices with respect to the end vertices in S satisfying the edge-index condition.

We also define a bound function $t = |E_G| - |E_S|$ for the loop, where E_G and E_S are the set of edges in G and S respectively. $|E|$ denotes the cardinality of the set E.

Proof. **1. I is true before the loop is entered.** Before the loop starts, the explored subgraph G_s of G consists of the single (starting) vertex $\phi(v_0)$, and S consists of the single vertex v_0. Thus, S and G_s are isomorphic, maintaining **I-1**. U is initialized with the edges that correspond to the edges leaving $\phi(v_0)$. Both set of edges are indexed using the same ordering convention and so the edge-index condition holds. This maintains **I-2**.

2. I is maintained by the loop body. Each time through the loop body, an edge e is selected and removed from U, through which the robot explores to the unknown end v_u. Every (other) unexplored edges $e' = (v_{k'}, v_{u'})$ whose known vertex $v_{k'}$ has the same signature (degree and presence/absence of marker) as v_u is considered as a potential loop-closing hypotheses. For each hypothesis (edge) $e' = (v_{k'}, v_{u'})$, a simple motion sequence $\mathcal{M}_{e'}$ which should drive the robot from $v_{k'}$ to v_0 is computed, along with the expected perceptions $\mathcal{P}^E_{e'}$ that the robot should encounter while executing $\mathcal{M}_{e'}$. The key to the correctness of the algorithm is the fact that the motion sequence $\mathcal{M}_{e'}$ which encodes (at the beginning) the ordering of e' at $v_{k'}$, along with the expected perceptions $\mathcal{P}^E_{e'}$ which encodes (at the end) the ordering of the entry edge at v_0 define an embedded path $v_0, ..., v_{k'}, v_{u'}$ in S, which *uniquely* identifies edge e' and its known end $v_{k'}$. Other hypotheses cannot have the same embedded path: they may have the same motion sequence as e' or may have the same expected perception as e', but not both. Specifically, the relative ordering at the initial vertex $v_{k'}$ specified in $\mathcal{M}_{e'}$ disambiguates e' against other unexplored edges (hypotheses) that are also incident at $v_{k'}$. Such hypotheses have the same expected perception as e', but have different beginning motion. On the other hand, the marker direction information specified at the end of $\mathcal{P}^E_{e'}$, which is not available in the undirected marker case, disambiguates e' against an unexplored edges (hypothesis) whose motion sequence is the same as that of e' but leads the robot to enter v_0 from a different edge. The different expected entry edge at v_0 results in perception that is different from $\mathcal{P}^E_{e'}$. The uniqueness of the embedded path implies that when a hypothesis $e' = (v_{k'}, v_{u'})$ is accepted (according to case 3) the robot must have traversed an edge $\phi(e')$ which is incident in $\phi(k)$ in G_s and leads the robot to another vertex $\phi(v_{k'})$ that is also in G_s. On the other hand, if no hypothesis exists or all the hypotheses are rejected, then no unexplored edges out of G_s corresponds to $\phi(e)$ and the robot must have traversed an edge that is incident in $\phi(k)$ in G_s but leads it to a vertex $\phi(v_u)$ that is not in G_s.

In the case that no hypothesis exists or otherwise all the hypotheses are rejected, which means that

the other end of vertex $\phi(v_u)$ of $\phi(e)$ must not in G_s, both $\phi(e)$ and $\phi(v_u)$ become new elements of the explored subgraph G_s. Correspondingly, e and v_u are added to S. We need to show that e is correctly labelled with respect to v_k and $v_{k'}$, and that U is updated correctly. By the hypothesis that **I** is true, $index(e, v_k)$ satisfies the edge-index condition. The algorithm is free to set $index(e, v_u)$ arbitrarily, since no indices of edges leaving v_u has been set yet. The algorithm sets the edge indices in v_u (including that of e) using the standard ordering convention to satisfy the edge-index condition, maintaining **I-1**. In this case, two updates occurred to the unexplored edges in G that have at least one incident vertex in G_s: 1) $\phi(e)$ is no longer such an edge, as it is explored now. 2) since $\phi(v_u)$ is added to G_s, all untraversed edges incident in $\phi(v_u)$ become such edges. Correspondingly, two updates are made to U: 1) e is removed from U. Note that since v_u was not in S before the loop on this occasion, e was the only entry in U that corresponds to $\phi(e)$. 2) edges incident in v_u are added into U. These edges satisfy the edge-index condition. These updates to U maintain **I-2**.

In the case that a hypothesis $e' = (v_{k'}, v_{u'})$ is accepted, which means that the other end vertex $\phi(v_u)$ of $\phi(e)$ is already present in G_s, $\phi(e/e')$ becomes a new edge in G_s but not the 'unknown' end vertex. Correspondingly, edge $e'/e = (v_k, v_{k'})$ is added to S. We show that e/e' is correctly labelled with respect to v_k and $v_{k'}$, and that U is updated correctly. The algorithm uses the index of e at v_k as the index of the new edge at v_k, which satisfies the edge-index condition, by the hypothesis that **I-1** is true. The algorithm uses the index of e' at $v_{k'}$ as the index of e/e' at $v_{k'}$, which again by the hypothesis **I-1** is true, satisfies the edge-index condition. So the new edge is indexed at its two end vertices correctly. This maintains **I-1**. Since no new vertex is explored, no new untraversed edges are generated for G_s. The only update concerning untraversed edges in G_s is that $\phi(e/e')$ is no longer an untraversed edge in G with at least one incident vertex in G_s. Before the execution of the loop body on this occasion, there must have been two unexplored edges in U (i.e., e and e') that correspond to $\phi(e/e')$. Both the two edges are removed from U by the loop body. So U is updated correctly, maintaining **I-2**.

3. The loop terminates. The loop invariant asserts that S and G_s are isomorphic, so $|E_S| = |E_{G_s}|$. Since G_s is the explored subgraph of G, it only contains edges that are also in G thus $|E_{G_s}| \leq |E_G|$, and therefore $|E_S| \leq |E_G|$. This implies that the bound function $t = |E_G| - |E_S|$ must be non-negative. In each iteration one edge is included into S. So in each iteration of the loop body, $|E_S|$ is increased and thus t is decreased ($|E_G|$ is fixed). So the loop must terminate eventually, as t can only remain non-negative for a finite number of iterations through the loop.

4. When the loop terminates, G_s = G. We show that when the loop terminates, i.e., when U=$\{\}$, there are no unexplored edges in G. Assume, to the contrary, that when the loop terminates there exists at least one untraversed edge in G. By invariant **I** and the termination condition U = $\{\}$, the edge must not have unexplored end(s) in G_s. Now assume v is one unexplored end vertex of an unexplored edge. Since G is connected, there must be a path from the starting vertex $\phi(v_0)$ to v. This requires that there is an edge on this path with one explored end and one unexplored end. By invariant **I**, there must be a corresponding edge in U, contradicting the termination condition that U = $\{\}$. So there are no unexplored edges in G. This also implies that there are no unexplored vertices in G. That is, the explored subgraph G_s = G. So the maintained map S, which is isomorphic to G_s, is now isomorphic to G. □

VI. Complexity analysis

As in [1] we consider physical steps moved in the environment (i.e., number of edge traversals by the robot) as the cost of the exploration algorithm. Denote the number of edges and vertices in the world by m and n respectively. Given this, a trivial lower bound $O(m)$ exists for the cost of exploring an unknown graph-like world as the robot must traverse every edge in the environment in order to know where all the edges go (e.g., DFS with $2m$ traversals).

Now consider the upper bound of the algorithm. We begin by bounding the number of edge traversals required in one pass through the loop body. Let n_s be the number of vertices in S during the current execution of the loop body. Each loop begins with the robot traversing S to v_k and then following edge e on v_k to v_u. This traversal requires at most n_s edge traversals. Then for each potential hypothesis of e, we validate it by having the robot traverse a simple path to v_0. Either all the hypotheses are rejected (v_u is a new place and e is a non-loop edge) or one of the hypotheses is accepted (v_u is one of the known places and e is a loop edge). There are $n-1$ iterations of loop executions where all the hypotheses are rejected and thus S grows by a non-loop edge and a new vertex, and in each of the remaining $m-n+1$ iterations one of the hypotheses is accepted and S grows by a loop edge. A worst case scenario would see S growing to its full number of vertices in the first $n-1$ iterations. In each of these iterations all the hypotheses are examined and rejected. In the worst case scenario all the current unexplored edges incident on non-marked places are potential hypotheses, and each hypothesis is rejected at the end of its path execution. Each path traversal is bounded by $2(n_s-1)$, and we can bound the number of hypothesis by $2m-2(n_s-1)$ where (n_s-1) represent the number of currently explored edges. Thus a bound on

the total number of edge traversals taken in the first $n-1$ iterations is

$$\sum_{n_s=1}^{n-1}[n_s + 2(n_s-1)(2m-2(n_s-1))] \tag{1}$$

In each of the remaining $m-n+1$ iterations, one of the hypotheses is accepted. In the worst case scenario the accepted hypothesis is the last hypothesis examined and each traversal comes to the end of the planned path, and all the unexplored (loop) edges are potential hypotheses. The length of each traversal path is bounded by $2(n-1)$, and we can bound the number of hypotheses in each of the remaining iterations i (for i in the range 1 to $m-n+1$) by $2m-2(n-1)-2(i-1)$ which represents all the unexplored loop edges in iteration i, and $i-1$ is the number of already explored loop edges in iteration i. Thus, a bound on the total number of edge traversals taken in the remaining $m-n+1$ iterations is

$$\sum_{i=1}^{m-n+1}[n + 2(n-1)(2m-2(n-1)-2(i-1))] \tag{2}$$

The total number of steps in the algorithm is bounded by (1) + (2), which simplifies to

$$2m^2n - 2mn^2 + \frac{2}{3}n^3 - 2m^2 - \frac{5}{2}n^2 + \text{lower order terms} \tag{3}$$

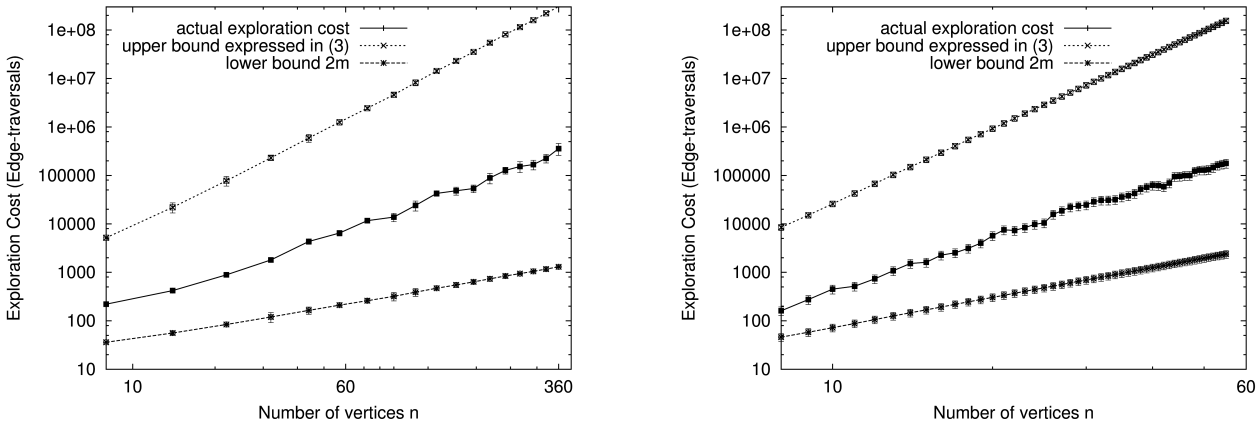
So the asymptotic complexity of the algorithm is $O(m^2n)$, where m and n are the number of edges and vertices in the world respectively.

VII. Empirical Evaluation

Fig. 3 illustrates the performance of the algorithm on both lattice graphs and densely-connected graphs with some fraction of edges removed. Upper cost bound derived in expression (3) and the lower bound $2m$ are also plotted. Results show that the performance cost for each sized graph is substantially below its theoretical cost bound. This is to be expected given the conservative assumption in the complexity analysis above that the cost of transiting to a given vertex in the graph is n_s.

VIII. Summary

Given an embedded topological world, it is not, in general, possible to map the world deterministically without resorting to the use of sufficient position and orientation information to solve the 'have I been here before?' problem. Both position and orientation information is required. [1] solved this through the



(a) Lattice graphs with 10% holes of varying sizes

(b) Densely connected graphs with 10% holes

Fig. 3. Performance of the single immovable directional marker algorithm on different sized graphs with 10% of the edges removed to create 'holes'. Results are averaged over 30 graphs, each with randomly located holes. Error bars shown standard errors.

use of a movable marker. Here we have shown that given a single 'directional lighthouse' vertex that provides both information locally, a provably correct mapping strategy exists with cost bound $O(m^2n)$.

There are several ways in which a 'directional lighthouse' can be established. We observe that the environment itself may contain structures that can be exploited to establish such information. For example, a vertex that is of unique degree and is adjacent to another vertex also of unique degree provides such a directional lighthouse. Lacking such structures, a single directional immovable marker or clusters of undirected immovable markers can also be used to form a directional lighthouse.

ACKNOWLEDGMENTS

This work is part of the Ph.D. thesis of the first author. We thank Eric Ruppert for helpful suggestions.

This work was supported by NSERC Canada through the NSERC Canadian Field Robotics Network (NCFRN).

REFERENCES

[1] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Trans. Robot. Autom.*, vol. 6, pp. 859–865, 1991.

[2] S. Thrun, "Robotic mapping: a survey," in *Exploring Artificial Intelligence in the New Millennium*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1–35.

[3] M. Blum and D. Kozen, "On the power of the compass (or, why mazes are easier to search than graphs)," in *19th Annual Symposium on Foundations of Computer Science (FOCS)*, Ann Arbor, MI, USA, 1978, pp. 132–142.

[4] M. Rabin, "Maze threading automata," Presented at MIT and UC Berkley, 1967.

[5] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Map validation and self-location in a graph-like world," in *Proc. 13th Int. Conf. Artificial Intelligence*, Chambery, France, 1993, pp. 1648–1653.

[6] X. Deng and A. Mirzaian, "Competitive robot mapping with homogeneous markers," *IEEE Trans. Robot. Autom.*, vol. 12, pp. 532–542, 1996.

[7] H. Wang, M. Jenkin, and P. Dymond, "Using a string to map the world," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010, pp. 561–566.

[8] ——, "The relative power of immovable markers in topological mapping," in *Proc. IEEE/RSJ Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 1050–1057.

[9] X. Deng, E. Milios, and A. Mirzaian, "Robot map verification of a graph world," *Combinatorial Optimization*, vol. 5, pp. 383–395, 2001.

[10] B. Kuipers and Y. Byun, "A qualitative approach to robot exploration and map-learning," in *Workshop on Spatial Reasoning and Multi-Sensor Fusion*, St. Charles, IL, USA, 1987, pp. 390–404.

[11] H. Wang, "Exploring topological environments," Ph.D. dissertation, Department of Electrical Engineering and Computer Science, York University, 2014.

[12] L. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams," *ACM Trans. on Graphics*, vol. 4, pp. 74–123, 1985.