2013 International Conference on Computer and Robot Vision

Enhancing Exploration in Topological Worlds with a Directional Immovable Marker

Hui Wang, Michael Jenkin and Patrick Dymond
Department of Computer Science and Engineering, York University
4700 Keele Street, Toronto, ON, Canada
Email: {huiwang, jenkin, dymond}@cse.yorku.ca

***Abstract*—One approach to answering the 'loop-closing' problem in embedded topological worlds is through the use of an external marking aid that can help the robot disambiguate otherwise identical locations. It has been shown in [1], [2], [3] that a single immovable directional marker is sufficient to solve loop closing, although at significant exploration cost ($O(m^2n)$ worst bound). Here we provide optimizations to the earlier algorithm and demonstrate improved exploration cost on a range of different topological worlds.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) [4], [5], [6] is considered to be a fundamental problem in mobile robotics. Although a wide range of SLAM algorithms exist, two main classes of algorithms have emerged: those that treat the problem as being embedded in a geometric or metric space and those that treat the problem as being topological or graph-based. In a topological approach, the world and its corresponding representation are modeled as a graph. Each location in the world is a vertex of the graph, and locations are connected by undirected edges. Agents (robots) 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 graph, 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 topological SLAM and is referred to as the 'loop closing' or 'have I been here before' problem. It has been shown that this problem is not solvable in general and that solving loop closing deterministically requires the use of a marking aid to help disambiguate different locations being explored [7]. Our previous work [1], [2], [3], [8] explored the expressive power of various forms of markers in exploring topological environments. The simplest type of marker that enables deterministic topological SLAM is a single directional immovable marker. (Note that a single immovable marker that is undirected is not sufficiently powerful to solve topological SLAM deterministically [1], [3].) We have developed algorithms that exploit a single directional immovable marker to map unknown topological embedded environment ([1], [2], [3]). Unfortunately, the basic single directional immovable marker algorithm is computationally expensive – given a world with n nodes and m edges, the worst case running time is $O(m^2n)$. (Here cost is measured in terms of number of edge traversals required by the robot.) Given this high cost we explore various optimizations that enhance the performance of the basic algorithm. Results show that for some environments up to a 60% cost reduction can be achieved.

Using marker(s) to help deal with uncertainty in exploration has been investigated for many years including early work described in [9] and [10]. Various types of markers are examined in the literature. Previous work including [7], [11], [12], [13], [14] examined the power of *movable* marker(s) in exploring undirected topological (graph-like) worlds. More recent work including [1] and [2] considered the increased power of more complex marker-based aids such as directional markers. A recent survey of marker-based exploration approaches can be found in [1]. In [1], [2], [3] and [8] we examined the power of different forms of *immovable* markers, and demonstrated that a robot with a single directional immovable marker can map the world deterministically. Unfortunately, the cost of the algorithm is significantly above the theoretical lower bound $O(m)$ for mapping and exploration. Motivated by the discrepancy, this paper explores a number of refinements to the single directional immovable marker algorithm. (This work expands on preliminary results presented at 3rd World Conference on Information Technology.)

II. Background

This paper adopts the world and robot model and evaluation metrics introduced in [7].

A. The World and Robot Model

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.

Robot motion and perception. Assume that the robot can leave a vertex by a given exit (edge) of the vertex, and can move between vertices by traversing edges. Assume the robot can identify when it is on an edge or when it has

978-0-7695-4983-5/13 $26.00 © 2013 IEEE
DOI 10.1109/CRV.2013.44

arrived at a vertex. At a vertex, the robot can enumerate the edges, and thus determine the relative ordering among the edges that are incident on the current vertex.

Marker related operation and perception. The robot has use of a directional immovable marker that can be left at one vertex, with the marker pointing to one of the incident edges at the vertex. The robot can sense the presence of the marker if the marker is present at the current vertex. In this case the robot can also sense the direction of the marker, that is, the robot can identify the edge that is pointed by the marker head. By enumerating the edges and identifying the marked one, the robot can distinguish between the different edges at the vertex. (In the algorithms presented here it is normally assumed that the marker is pre-placed in the environment. The question of selecting an optimal location for the marker is an interesting question for future research.)

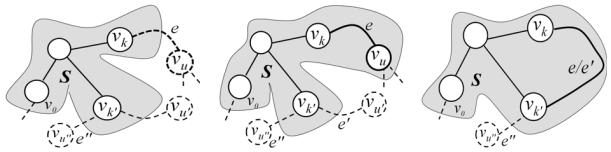
B. Metrics and lower exploration bound

As in [7], [12], [13], [2], [3], we consider physical steps moved in the environment (i.e., number of edge traversals by the robot) as the cost of the exploration algorithm. Given this, a trivial lower bound $O(m)$ exists for the cost of exploring an unknown graph-like world.

C. Sketch of the basic mapping approach

The goal of the mapping algorithms is to generate a map representation of the underlying graph-like world G that is being explored. Following [7] and [11], the marker-based algorithms proceed by building incrementally a known map S out of the explored subgraph of the unknown world G. As new locations (vertices and edges) in G are encountered during exploration, they are added (represented) in S, and unexplored incident edges of new vertices are added to U which is a set of unexplored edges. Initially map $S = \{v_0\}$ where v_0 corresponds the initial location of the robot and the marker. Incident edges at the initial location are the initial elements of U.

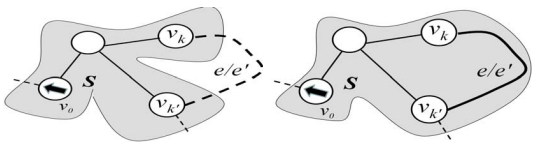
Each step of the algorithm begins with selecting (and removing) an unexplored edge $e = (v_k, v_u)$ from U, and having the robot traverse S to the known end v_k of e and then following e to the unknown end v_u of e (Figure 1(a)). Upon arrival at v_u, the robot needs to answer the question 'have I been here before?', which involves validating v_u and e ensuring that v_u is distinct from all the previously encountered places. If validation shows that v_u is a new place, then no loop is formed and both v_u and e are added to S (Figure 1(b), non-loop augmentation). If validation shows that v_u is a known place $v_{k'}$ on S and e corresponds to unexplored edge e' at $v_{k'}$, then a loop is formed (the robot has entered $v_{k'}$ via e') and thus (only) edge $e/e' = (v_k, v_{k'})$ is added to S (Figure 1(c), loop augmentation). Now edge e' no longer represents an unexplored edge and is removed from U. This process repeats with newly selected edges from the unexplored edge set U, until U becomes empty. It has been shown that when U is empty, the map S is isomorphic to the underlying world G [7] (provided that the algorithm correctly validated each newly visited place v_u).



(a) Traverse to v_u via e (b) Non-loop augmentation (c) Loop augmentation

Figure 1. Basic exploration step. The robot must determine if $e = \{v_k, v_u\}$ corresponds to a loop or not. S is augmented in (b) and (c). Dotted lines represent the unexplored portions of the graph-like world G, and solid lines represent the explored portion of G (i.e., S).

The key challenge is the validation process for each newly explored edge e and its unknown end v_u. With a single directional immovable marker, which is dropped at the robot's starting place v_0 and thereafter remains there, validations are conducted by disambiguating each currently explored edge e (and its unknown end v_u) against unexplored edges currently in U. Each unexplored edge $e' = (v_{k'}, v_{u'})$ currently in U is considered as a potential loop-closing hypothesis. 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' (Figure 2(a)). For each hypothesis $e' = (v_{k'}, v_{u'})$, based on the hypothesized location ($v_{k'}$) and orientation (e') of the robot, a simple path $v_{k'}, ..., v_0$ on S is computed. The path is represented as a motion sequence, which is a sequence of relative edge orderings (relative to the entry edge) at each vertex visited along the path. An example motion sequence is (1,3,...4), which, assuming a clockwise enumeration rule for a planar embedding, specifies that 'start with the 1st next edge (at $v_{k'}$) on the left of e', upon arrival take the 3rd next edge on the left of the entry edge, ..., finally (at v_0) the marker-pointed edge is the 4th edge on the left of the entry edge'. The expected perception that the robot should obtain along the path of e', denoted $P^E_{e'}$, is also computed. $P^E_{e'}$ consists of a sequence of node 'signatures' along the path, which are perceived characteristics of nodes including their degrees and the absence (A) or presence (P) of the marker at the node. An example perception sequence is ([2,A][4,A]...[3,P]) where [2,A] indicates that the current vertex has degree 2 and the marker is absent. The robot then validates the hypothesis by attempting to execute the motion sequence, which, if the hypothesis holds, would start from $v_{k'}$ and lead the robot to v_0 where the marker will be sensed with expected direction. During execution, the sensed perception $P_{e'}$ is compared against the expected perception $P^E_{e'}$. If $P_{e'}$ and $P^E_{e'}$ match – both during and at the end of execution – then the hypothesis is confirmed and validation for e is completed (i.e., $e = (v_k, v_u)$ and $e' = (v_{k'}, v_{u'})$ correspond to the same edge, and v_u corresponds to known



(a) A loop hypothesis (b) Hypothesis is confirmed

Figure 2. Single directional immovable marker algorithm.

place $v_{k'}$, and S is loop augmented with e/e' (Figure 2(b)). If a mismatch between $P_{e'}$ and $P^E_{e'}$ is detected at any time during traversal, hypothesis e' is rejected immediately. In this case the robot stops and returns back to v_u by executing the motion sequence in reverse, and then repeats the process for another hypothesis of e (if any). If all the hypotheses of e are examined but none of them are confirmed, then v_u must be a new place and thus S is non-loop augmented. The algorithm is sketched in Algorithm 1.

Algorithm 1: Single directional immovable marker algorithm described in [1], [2], [3].

Input: the starting place v_0 in underlying world G
Output: a graph representation that is isomorphic to G

the robot drops the marker at v_0, pointing to an edge;
$S \leftarrow \{v_0\}$; $U \leftarrow$ 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 follows e to v_u;
  $H \leftarrow$ set of hypotheses (edges) in U whose known end vertices have the same signatures as v_u;
  while *H is not empty* **do**
    $e' = (v_{k'}, v_{u'}) \leftarrow$ a (removed) hypothesis in H;
    compute motion sequence for a path $v_{k'}, ..., v_0$, and the expected perception $P^E_{e'}$ along the path;
    the robot tries to traverse path $v_{k'}, ..., v_0$;
    based on actual perception $P_{e'}$ in traversal **do**
      case *$P_{e'}$ and $P^E_{e'}$ match exactly*
        confirm e', exit inner 'while' loop;
      case *$P_{e'}$ and $P^E_{e'}$ do not match*
        the robot retraces to v_u and continue;
  if *a hypothesis is confirmed* **then**
    do 'loop augmentation' on S;
  else `// all hypotheses are rejected`
    do 'non-loop augmentation' on S and U;
return S;

III. Enhancements

The above algorithm is a provably correct solution to topological SLAM but has a high cost. As an example, for a 10 × 10 lattice with 20% holes (removed nodes), the algorithm requires more than 10,000 edge traversals, as shown in Figure 3. The high exploration cost comes from the traversal-based validation of hypotheses, in which each hypothesis is validated *independently* – for a hypothesis e' of edge e, once it is rejected due to a mismatch between the sensed perception $P_{e'}$ and the expected perception $P^E_{e'}$, the robot returns back to v_u and then computes and executes the motion sequence for another yet-to-be validated hypothesis.

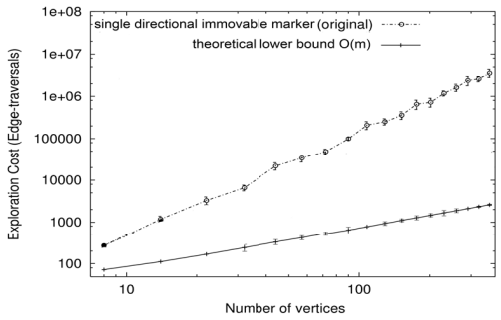


Figure 3. Performance of original single directional immovable marker algorithm (log scale). On lattice with 20% holes of vary sizes. Optimal cost bound is also plotted. Each data point represents an average over 30 random lattice graphs.

The high cost motivates the enhancements described below. Each of the following enhancements seeks to reduce the overall number of edge traversals by exploiting information obtained from earlier validations in later hypothesis validation steps. Note that these enhancements do not improve the theoretical exploration cost $O(m^2n)$, rather they are optimizations that find substantive improvements in practice.

A. Enhancement-I

When a hypothesis is rejected, Enhancement-I uses the executed traversals associated with the rejected hypothesis to examine other yet to be validated hypotheses, in an attempt to reject some hypotheses without motion, or to reduce the amount of motion needed in validating other hypotheses.

The key observation is that for a newly explored edge e, motion sequences for the different hypotheses may define embedded paths that partially or completely *overlap*. Here we capture one kind of path overlap: the paths overlap from v_u. While we don't know where each motion visits until e and v_u are finally validated, we do know that beginning from v_u, the same motion sequence (relative doors) visits the same edges and vertices in the environment. For example, motion sequences (2,1,3) and (2,1,4) for two hypotheses define paths that partially overlap at the first two edges and vertices (excluding the starting vertex v_u). Suppose a hypothesis e' has motion sequence (2,1,3,1,2,1) and the robot, starting from v_u, executes this motion sequence and stops after it has executed (2,1,3,1,2) when a mismatch is found. Suppose now another unvalidated hypothesis e'' has motion sequence (2,1,3,3,2,4), which defines a path that

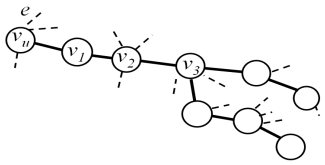


Figure 4. Overlap of motion sequences (2,1,3,1,2) and (2,1,3,3,2,4) – overlap at the first three edges and vertices (v_1 v_2 and v_3, excluding v_u).

overlaps with the executed path at the first three edges and vertices, as shown in Figure 4. We can exploit this overlap to reduce the number of edge traversals that are required. This enhancement exploits the overlap in two stages: First, the overlap due to the first three motions means that if the robot executes motion sequence of e'' from v_u and finishes (as in the original algorithm), it will visit exactly the same first three vertices (and edges) as the first three vertices (and edges) it has just visited (v_1, v_2 and v_3). This implies that if the robot executes motion for e'', the observed perception $P_{e''}$ at the first three vertices (v_1, v_2 and v_3) must be the same as the observed perception $P_{e'}$ for e' in the first three vertices, which in turn implies that for e'' to be accepted, the observed perception $P_{e'}$ for e' at the first three vertices must match the expected perception $P^E_{e''}$ of e'' at the first three vertices, otherwise e'' should be rejected. Thus in this example, when the robot stops its motions for e' due to a signature mismatch, it remains at the vertex where it stopped (instead of coming back to v_u as in the original algorithm), and compares the observed perceptions $P_{e'}$ the robot just obtained at the overlapped three vertices against the expected perception $P^E_{e''}$ of e'' at the first three vertices. If $P_{e'}$ and $P^E_{e''}$ do not match exactly at the three overlapped nodes, hypothesis e'' is (also) rejected immediately without executing its motion sequence.

What if the perceptions $P_{e'}$ and $P^E_{e''}$ match at all of the overlapped places? In such a case, the second stage of Enhancement-I starts. For the example given above, suppose that expected perception $P^E_{e''}$ for e'' matches the obtained perception $P_{e'}$ for e' at the first three vertices. Now the robot needs to validate e'' with motions but only needs to validate (traverse) the *non-overlapped* part of the motion for e''. To do this, the robot retraces its motions but only until it comes back to the *first* overlapped vertex it encounters on the way back (v_3 in the example). Then the robot executes the non-overlapped part of the motion sequence of e''. In this example, the robot executes (0, -2, -1) with 0 and -2 leading the robot back to v_3 and -1 re-orienting the robot to the original entry edge at v_3, and then executes (3, 2, 4). Now further suppose that the robot finishes traversing the non-overlapping part of the path for e'' and detects a mismatch. It then repeats the above process by staying at the current location and examining the possible path overlap with another unvalidated hypothesis e''' (if any). Note that in calculating the possible path overlap of e'' and e''', the robot uses its *original* motion sequence (2,1,3,3,2,4) for e'' as the traversed motion sequence for e'', and in comparing the perceptions, uses the observed perception for e' at the overlapped vertices *plus* the observed perception for e'' at the non-overlapped part as the observed perception for the whole sequence of e'', i.e., $P_{e''} = P^{overlap}_{e'} + P^{non-overlap}_{e''}$. That is, the robot proceeds as if it has traversed from v_u as in the original algorithm. By returning to the first overlapped vertex and continuing on the non-overlapping part, two traversals on each overlapped edge are avoided.

In order to keep the robot 'hanging around' as much as possible before it has to return to v_u, we adopt a *greedy* approach to exploiting the possible overlaps of the paths. When the robot enters an unknown vertex v_u via e, we pre-process all the hypotheses for e, computing the motion sequence and expected perception P^E for each hypothesis. One of the hypotheses is chosen as the first hypothesis which is validated with traversals (as in the original algorithm). From then on, whenever the robot stops executing the motion sequence of a hypothesis due to perception mismatch, it stops there and computes possible overlaps between the just traversed path and the path defined by the motion sequence of each remaining hypothesis. We first compare the perceptions based on the observed perception at the overlapped vertices and filter out (reject) all the hypotheses whose expected perceptions do not match the obeserved perception at the overlapped nodes (stage 1). For the remaining hypotheses, whose motion sequences need to be executed, the robot selects the one with the *longest* overlapped part (i.e., highest number of overlapped vertices), and executes the non-overlapped motion sequence of this selected hypothesis (by coming back to the nearest overlapped vertex and then continuing on the non-overlapped part (stage 2)). Note that in the case that the path of an unvalidated hypothesis is part of a traversed path of the validated hypothesis from the beginning (e.g., the executed motion sequence is (3,2,1,4) and the to-be-executed motion is (3,2,1)), then the robot has already traversed the path for the unvalidated hypothesis and thus no movements are required for the unvalidated hypothesis. The algorithm is sketched in Algorithm 2.

B. Enhancement-II

The above enhancement uses the traversals of one rejected hypothesis to reduce the exploration cost associated with the other hypotheses that have not yet been validated, in the special case of overlapping of paths from the beginning. This raises an interesting and general question: can we exploit the executed traversals of a rejected hypothesis to examine hypotheses that have not been validated yet, even if the paths do not overlap? Suppose the robot has traversed motion sequence (3,2,1,4) and a yet to be validated hypothesis has motion sequence (1,2,3,4). The paths defined by the motion sequences do not overlap from the beginning, and

Algorithm 2: Enhancement-I.

```
the robot drops the marker at v_0, pointing to an edge;
S ← {v_0}; U ← 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 follows e to v_u;
    H ← set of hypotheses (edges) in U whose known
    end vertices have the same signatures as v_u;
    e' = (v_k', v_u') ← a (removed) hypothesis in H;
    compute motions for a path v_k', ..., v_0, and P^E_e';
    the robot traverses path v_k', ..., v_0;
    while True do
        based on observed perception P_e' during
        traversal do
            case P_e' and P^E_e' match exactly
                confirm e', exit inner 'while' loop;
            case P_e' and P^E_e' do not match
                the robot stops and stays at the location;
                for each e'' in H do
                    compute the overlap of executed
                    path of e' v.s. computed path of e'';
                    if perception P_e' and P^E_e'' do not
                    match at overlapped parts then
                        remove e'' from H
                if H is empty then
                    exit the inner 'while' loop
                e' ← one of the remaining edges in H
                whose overlap path is the longest;
                the robot retraces to the nearest
                overlapping node v_x and continues on
                the non-overlapped part v_x,...,v_0 of e'.
    if a hypothesis is confirmed then
        do 'loop augmentation' on S;
    else // all hypotheses are rejected
        do 'non-loop augmentation' on S and U;
return S;
```

may or may not overlap at some points later (but before e and v_u are validated, we do not know). Here we present another enhancement that extends Enhancement-I, exploring techniques that exploit executed motions even when no overlaps are captured.

Suppose that when validating a newly explored edge e, we keep track of all of the executed motion sequences for the rejected hypotheses of e, and their corresponding observed perceptions. Then for a hypothesis $e'' = (v_{k''}, v_{u''})$ of e, before validating it by executing its motion sequence, we can assume for a moment that e'' is true (and thus the robot has entered $v_{k''}$ via e''). Then based on the hypothesized position ($v_{k''}$) and orientation (e'') of the robot, trace (on map S) all the previously executed motion sequences of the rejected hypotheses of e. Some of the executed motions may map into known locations (on map S), and some other executed motions may map into unknown locations. The key observation is that *if* hypothesis e'' is true, then for the previously executed motions that map into known vertices (based on e''), the previously observed perception for these motions must match the signatures of these known vertices. Thus if any of the perception at a known location does not match the signature of the location, the hypothesis e'' can be rejected without executing its motion sequence. Suppose the robot has executed motion sequence (3,2,1,4) for a hypothesis e' and rejected e' due to perception mismatch, and the corresponding observed perceptions $P_{e'}$, excluding that of v_u, are ([4,A][2,A][3,A],[5,A]). Now in validating another hypothesis $e'' = (v_{k''}, v_{u''})$, we first assume for a moment that e'' is true (and thus the robot has entered $v_{k''}$ via edge e'') and then trace the previous motion sequence (3,2,1,4) on S. Suppose that relative edge 3 with respect to edge e'' at $v_{k''}$ is an explored edge in $v_{k''}$ leading to another known vertex $v_{k'''}$ on S, relative edge 2 with respect to this explored (entry) edge at $v_{k'''}$ is an explored edge in $v_{k'''}$ leading to a known vertex $v_{k''''}$, and relative edge 1 with respect to this explored (entry) edge at $v_{k''''}$ is an unexplored edge in $v_{k''''}$ (Figure 5). Then we retrieve the signatures of $v_{k'''}$ and $v_{k''''}$ from the map and compare them against the corresponding observed perceptions [4,A] and [2,A] when executing motion sequence 3, and 2 for e'. If any signature mismatch is detected then hypothesis e'' is rejected immediately without executing its motion sequences. That is, the traversal for hypothesis e' is used to validate e'' as well. In the example, e'' should be rejected as the signature of $v_{k''''}$ is [3,A] but the observed perception is [2,A]. So during exploration, for a newly explored edge e, we record all the motions executed in validating its hypotheses. We also record their corresponding perceptions. Then for each hypothesis of edge e (except the first hypothesis), we assume that the hypothesis is true and then trace all the previously executed motions on the map. For the motions mapped into known locations on the map, we compare the observed perceptions against the signatures retrieved from the map. If any mismatch is detected, the hypothesis is rejected immediately. If all the perceptions match, then we validate the

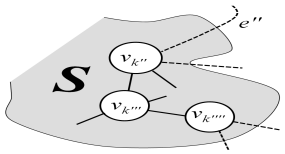


Figure 5. Trace the previous motions on S. Assuming e'' is true, trace motion (3,2,1,4) from $v_{k''}$. Map into $v_{k'''}$ and $v_{k''''}$. Signature of $v_{k'''}$ is [4,A]. Signature of $v_{k''''}$ is [3,A].

hypothesis as in the original algorithm or use Enhancement-I (stage 2). Note that, as the example shows, in tracing motion sequences, we have to stop tracing when a motion traverses an unexplored edge and thus leads to an unknown location as we don't have information about unknown locations. This constraint is addressed in Enhancement-III.

Note that this enhancement incorporates the idea (stage 1) in Enhancement-I that if a hypothesis e'' has a motion sequence that overlaps with the executed motion sequence of a previous hypothesis e' from the beginning then the expected perceptions for e'' on the overlapped part are compared against the observed perceptions on the overlapped part. In this enhancement, by assuming e'' is true, the executed motions of e' that overlap (from the beginning) with the computed motions of e'' must map into known vertices, because these (overlapped) executed motions visited vertices that are on the computed path for e'', which is on the map and thus is composed of known vertices. So the observed perceptions of e' at the overlapped parts are compared against the expected perception of e'' on the overlapped parts, as in stage 1 of Enhancement-I.

C. Enhancement-III

In Enhancement-II, when validating a hypothesis we first assume it is true and then trace all the previous traversals looking for inconsistencies. When a motion traverses an unexplored edge that leads to an unknown place, we have to stop tracing as we don't have signature information of unknown places for comparison. All the observed perceptions thereafter are discarded in the current comparison. In the example above, we had to stop when motion 1 in (3,2,1,4) leads to an unexplored edge of $v_{k''''}$ and discard subsequent perceptions [3,A] and [5,A]. Can we also exploit the perceptions at the unknown places? This is explored here. When the explored edge e and the unknown end vertex v_u are finally validated – either as a new place or a known place – and thus the map is augmented (either by a new edge and a new vertex or by a loop edge), we trace all the previous traversals executed in validating v_u (again), based on the validated robot position and orientation. For a motion that leads to an unexplored edge of a known vertex, we record the corresponding perception at the unknown end as the signature of the unknown end. For the example given in Enhancement-II, suppose after all the validations, v_u turns out to be a new vertex, denoted v_x and assign e to be the 0'th edge of v_x. Then by tracing a previously executed motion sequence (3,2,1,4) and its corresponding observed perception ([4,A][3,A][2,A][5,A]), we can infer that the robot has traversed edge 3 of the new vertex v_x. So we can record that edge 3 of vertex v_x leads to an unknown neighbor vertex having signature [4,A], denoted v_x–3–[4,A]. Then during validation in later exploration, for a hypothesis $e'' = (v_{k''}, v_{u''})$ of a newly explored edge e, as in Enhancement-II, we assume e'' is true and trace previous motions executed in validating edge e. Now suppose that a previously traversed motion sequence for validating hypothesis e' of e is (5,3,2,1), and that motion 5 (from $v_{k''}$) maps into v_x and motion 3 maps into the 3rd edge of v_x. Now we can retrieve the correct signature [4,A] for the other end of edge 3 in v_x, even if edge 3 of v_x is (still) an unexplored edge. (If edge 3 of v_x already becomes an explored edge, then as usual, the signature can be retrieved from map S directly.) Suppose signature of v_x is [6,A], then if the observed perception in executing (5,3,2,1) is not ([6,A],[4,A],....), hypothesis e'' can be rejected immediately. Note that in Enhancement-II, only [6, A] – the signature of v_x – can be used for comparison (v_x is a known place). Now with extra efforts, we have more information [4,A] to examine a hypothesis before executing its motion sequence.

Above we record the observed perception (signature) of unknown vertices that are 'neighbors' of known vertices (call this technique 'one-level Enhancement-III'). We can extend this idea, at the cost of increased memory, to record *all* the observed perceptions (signatures) on unknown places (call it 'all-level Enhancement-III'). As above, when the explored edge e and the unknown end vertex v_u are finally validated we trace the previous traversals for v_u based on the validated v_u and e. In addition to unknown places that are neighbors of known vertices, we also record the perception information of the unknown places that are adjacent to other unknown places. (We expand the local signature.) We index the unknown places using the relative door orderings from a known place. Consider again the above example, where v_u is validated as a new vertex v_x, and e is the 0'th edge of v_x. For the previous motion sequence (3,2,1,4) whose observed perception is ([4,A][3,A][2,A][5,A]), as above, we record that edge 3 of v_x leads to an unknown neighbor of signature [4,A]. Further, we record that motion (relative edge) 2 of this neighbor leads to an unknown place of signature [3,A], and then relative door 1 leads to an unknown vertex of signature [2,A], and then relative 4 lead to a vertex of signature [5,A], denoted v_x–3-[4,A]–2-[3,A]–1-[2,A]–4-[5,A]. We maintain a searchable list of such records. During validation in later exploration, when validating e'' of e, we assume e'' is true and trace previous motions in validating e. Suppose as before that a moiton for e' of e is (5,3,2,1), in which motion 5 (from $v_{k''}$) maps into v_x and 3 maps into the 3rd edge of v_x. Now in addition to retrieve [4,A] for edge 3 of v_x, we can further retrieve from the maintained list of records that for next motion 2 and 1, the corresponding signatures are [3,A], [2,A], even if the edges are (still) unexplored edges. The retrieved signatures are used for comparison with the corresponding perceptions. The 'all-level' version of Enhancement-III is sketched in Algorithm 3.

IV. Empirical evaluations

We first conduct experiments on various sized lattice graphs with random holes (removed nodes), which rep-

Algorithm 3: Enhancement-III (all-level)

```
the robot drops the marker at v_0, pointing to an edge;
S ← {v_0}; U ← edges in v_0;    // initial S & U
T ← {};  // container (e.g., hash table) to store perceptions by unexplored edges
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 follows e to v_u;
    H ← set of hypotheses (edges) in U whose known end vertices have the same signatures as v_u;
    while H is not empty do
        e' = (v_k', v_u') ← a (removed) hypothesis in H;
        assume e' is true, based on hypothesized v_k' and e', trace all previous motions in validate e;
        if a motion mapped into a known place then
            retrieve signature from S;
        else // mapped into unexplored edge
            try to retrieve signature by searching T;
        if a signature is retrievable and/but does not match a motion's observed perception then
            reject e' and continue;
        else
            compute motion for v_k', ..., v_0, and P^E_e';
            the robot traverses path v_k', ..., v_0;
            based on observed perception P_e' during traversal do
                case P_e' and P^E_e' match exactly
                    confirm e', exit inner 'while' loop;
                case P_e' and P^E_e' do not match
                    the robot retraces to v_u, continue;
                    // or use Enhancement-I
    if a hypothesis is confirmed then
        do 'loop augmentation' on S;
    else // all hypotheses are rejected
        do 'non-loop augmentation' on S and U;
    based on validated position v_u and orientation e, trace all executed motions in validating e, and add to T the perceptions on unkown places;
return S;
```

resent typical indoor environments. The robot explores this environment with Enhancement-I. Results show that for the heterogeneous environment, Enhancement-I gives significant cost reductions (Figure 6(a)). The same set of graphs was explored using Enhancement-II. For this environment, Enhancement-II provides further cost reduction (Figure 6(b)). The same set of graphs was also explored using Enhancement-III. We first used the 'one-level' Enhancement-III which keeps track of perceptions at unknown places that are adjacent to known places. Results show that further reductions are achieved (Figure 6(c)). Exploring with the 'all-level' Enhancement-III which maintains the perceptions at all visited but yet unknown places shows that with extra work, further reductions are achieved (Figure 6(d)). For a lattice of 10 by 10 with 20% holes, the cost is reduced from 14000 edge traversals in the original algorithm to about 4500. We also conducted experiments on other graphs including densely-connected graphs, which are regular graphs with random holes. The results for Enhancement-I and Enhancement-II are shown in Figure 6(e)), and results for Enhancement-III, both one-level and all-levels, are shown in Figure 6(f)). Results showed that, similar to the case of lattice graphs, all-level Enhancement-III provides cost reduction over one-level Enhancement-III. Both versions of Enhancement-III provide cost reduction over Enhancement-II. Enhancement-II gives cost reduction over Enhancement-I, which in turn gives cost reduction over the original algorithm.

V. Conclusion

This paper investigates robotic exploration and mapping with marker-based aids. We present several techniques to improve earlier results for mapping with a single directional immovable marker. The techniques require no extra motions by the robot. So if computation cost is negligible compared to the cost of actual traversals, they can be incorporated into the original algorithm with considerable cost reduction.

References

[1] H. Wang, "Exploring topological environments," Department of Computer Science and Engineering, York University, Tech. Rep. CSE-2010-05, 2010.

[2] H. Wang, M. Jenkin, and P. Dymond, "Using a string to map the world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 561–566.

[3] ——, "The relative power of immovable markers in topological mapping," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1050–1057.

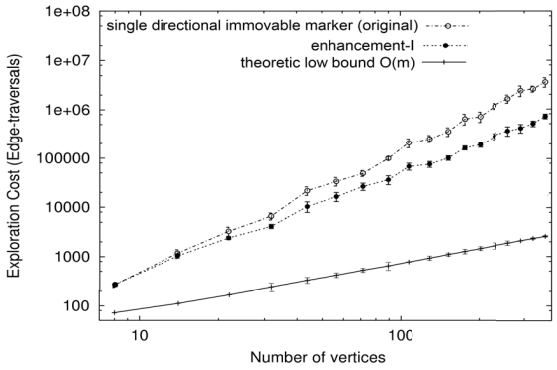
[4] S. Thrun, "Robotic mapping: a survey," in *Exploring Artificial Intelligence in the New Millennium*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1–35.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. USA: MIT Press, 2005.

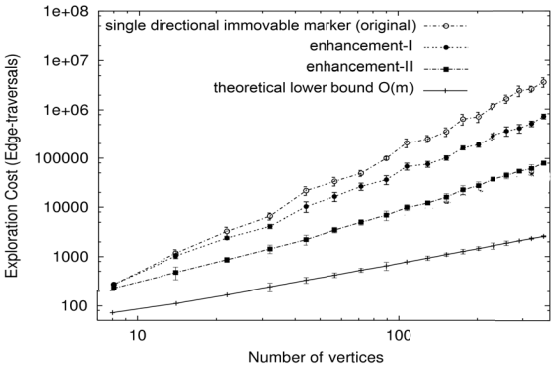
[6] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping (SLAM): Part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–100, 2006.

[7] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," Dept. of Computer Science, University of Toronto, Tech. Rep. RBCV-TR-88-23, 1988.

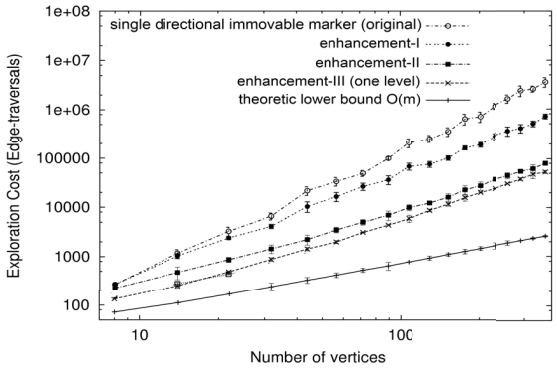
[8] H. Wang, M. Jenkin, and P. Dymond, "Enhancing topological exploration with multiple immovable markers," in *Canadian Conference on Computer and Robot Vision (CRV)*, 2012, pp. 322–329.



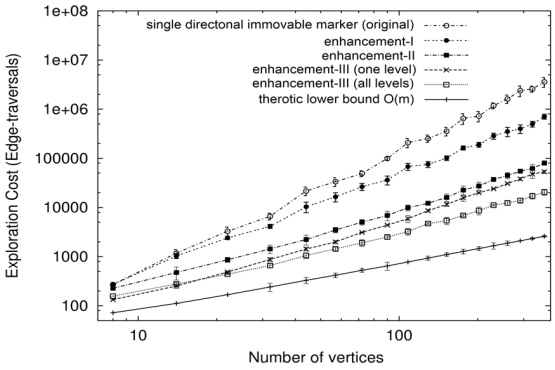
(a) Enhacement-I. Lattice graphs.



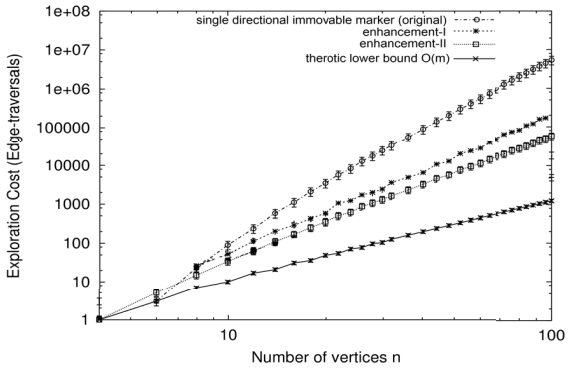
(b) Enhancement-II. Lattice graphs.



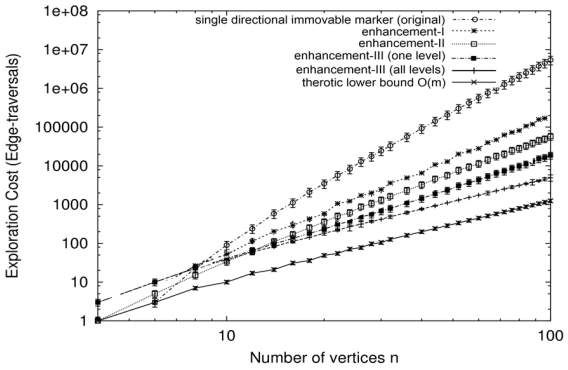
(c) Enhancement-III (one level). Lattice graphs.



(d) Enhancement-III (all level). Lattice graphs



(e) Enhancement-I and II. Densely-connected graphs



(f) Enhancement-III (one and all level). Densely-connected graphs.

Figure 6. Performance of enhancements on varies sized lattice hole ((a)-(d)) and densely connected graphs ((e)-(f)). Results are shown in log scale and averaged over 30 graphs, each has randomly located holes. Error bars show standard errors. Cost of original algorithm and theoretic bound are also plotted.

[9] 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*, 1978.

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

[11] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 7, pp. 859–865, 1991.

[12] ——, "Map validation and self-location in a graph-like world," in *13th International Conference on Artificial Intelligence*, Chambery, France, 1993, pp. 1648–1653.

[13] X. Deng and A. Mirzaian, "Competitive robot mapping with homogeneous markers," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 532–542, 1996.

[14] H. Wang, M. Jenkin, and P. Dymond, "Enhancing exploration in graph-like worlds," in *Canadian Conference on Computer and Robot Vision (CRV)*, 2008, pp. 53–60.