Using a Marker to Map an Unknown Environment

Gregory Dudek*, Michael Jenkin+, Evangelos Milios*, David Wilkes*

*Department of Computer Science, University of Toronto
Toronto, Ontario, Canada
+Department of Computer Science, York University
Downsview, Ontario, Canada
(authors listed alphabetically)

Abstract

A fundamental problem in robotics is the exploration of an unknown environment. Current approaches to exploration make use of a global distance metric that is used to relate past sensory experiences to local measurements. Rather than rely on such an assumption we consider the more general problem of exploration without a distance metric: we propose robot exploration as graph building. We demonstrate that it is not possible for a robot to successfully explore an arbitrary graph without additional aids. By augmenting the robot with a distinct marker which can be put down or picked up at will, we show that a robot can successfully map its environment. An algorithm is developed for map building and is applied to a complex environment.

Keywords: Autonomous Navigation, Robot Exploration, Map, Graph

I. Introduction

Many robotic systems exist in a completely known and well specified environment. This is particularly true for industrial robots such as welders. The parts to be welded are always located in exactly the same place, and very little work is required to ensure that the positioning is exact. More complicated systems allow parts to be located within a small window, but again the environment is well specified and rigidly controlled [20]. The situation is similar in current autonomous vehicle research. In many applications it is assumed that a metric map exists for the environment, and that the position of the robot with respect to the map can be maintained [11]. Such maps are often unavailable in practice, however, and accurately establishing the robot's position within one is very difficult. Leaving aside the problems involved in having a robot accurately locate itself within a map, many environments exist which cannot be well-mapped in advance. In addition, many systems rely on the use of a metric to establish spatial relationships between objects in the environment, the robot itself, and the map. This is not easily accomplished in practice, and a number of approaches, e.g. [5], have been suggested in order to more accurately track the motion of a robot in its environment so as to reduce positional errors.

Rather than have a "canned" map of the environment, another approach is to have the robot explore its own environment, and for the robot to build its own internal representation. As the robot moves it builds up a model of its environment by integrating information obtained at different times with the known motion of the robot. Unfortunately, errors in various sensor readings, coupled with errors in estimating the distance that the robot has moved between measurements, can quickly lead to horrendous errors in the robot's estimation of its location. This highlights a fundamental problem with these approaches: in most environments there is no reliable distance metric that can be extracted by a robot. This problem is manifested by errors in knowing how far the robot has moved, by range data errors, and by errors in several other parameter estimation tasks. In this paper we examine how a robot could go about exploring an environment without relying on a distance metric.

Motivated by the need for spatial representations other than ones based on metric information, [15] proposes a four-level spatial semantic hierarchy. The four levels, starting with the lowest, are the sensorimotor (robot sensations and primitive actions), procedural (robot actions to accomplish place-finding and route-following tasks), topological (places and paths and their topological relations), and metric (places and paths and their metric relations). In this paper, we provide precise definitions of what correspond to the sensorimotor, procedural and topological levels associated with the learning and navigation in a graph-like world, assuming that metric information is neither sensed nor stored.

The problem which we address in this paper is this: Given an unknown environment, formulate a series of plans for the robot so that after carrying out those actions, the robot will have an understanding of its environment sufficient for solving navigation tasks with the use of sensors (i.e. it builds a map). We begin by informally discussing the world in which the robot will exist, the robot itself, and the actions and sensations with which it will be equipped. In a later section we define these terms, and the problem we wish to solve, more formally. To solve the problem of determining when the robot has returned to a previously visited location (the "am I there yet" problem) during map build-

ing, we use a single portable marker (an idealized pebble that can be picked up and put down), as the only definitive approach in the lack of precise metric information and distinct landmark features (the extension of this technique to multiple markers is discussed in [6]).
The abstraction used here presents a particularly impoverished world. Only the number of exits from each location and one's own beacons can actually be sensed. In practice, most real robotic systems can be expected to have richer perceptual inputs than these. The simple model used here, however, serves as a lowest common denominator for the capabilities of robotic systems. By showing that the map acquisition problem can be solved with acceptible complexity bounds under these circumstances we demonstrate that such tasks are solvable within at least these bounds by more sophisticated systems. Furthermore, although more sophisticated perceptual mechanisms may be available, they are rarely completely dependable (visual tokens may be ambiguous as well as unstable). Hence, even robots with powerful sensing systems may occasionally find themselves reduced to the level of the model described here.

II. Previous Work

Past work on spatial representations for robotic exploration can be broadly classified into three categories.

Metric Representations The first category uses a metric representation of space, where features are explicitly associated with their Cartesian coordinates. Often the metric representation has the form of a graph, the vertices of which are annotated with their Cartesian coordinates. After acquiring the metric representation, the robot is then assumed to be able to plan navigation paths and execute them with minimal dependence on external sensors. Representative papers in this category are the papers by Crowley [5], Iyengar et al. [12], Turchan and Wong [23], and Rao et al. [19, 18].
Crowley [5] describes a robot that acquires a metric model of its environment using sonar range sensor data. The model is identical to a metric floor plan, overlaid with a network of places connected by legal highways running through convex regions. Iyengar et al. [12] propose a system whose map is a spatial graph (a graph whose vertices have Cartesian coordinates and whose edges are straight-line segments) and its associated Voronoi diagram. Navigation is performed along the "safe" routes indicated on the spatial graph, which is updated to account for previously unseen stationary obstacles. The graph representation of Iyengar [12] also contains a significant amount of metric information. Turchan and Wong [23] use an attributed graph representation of the world, where all obstacles are assumed to be polygonal. Vertices of the graph correspond to walls, whereas edges correspond to their connectivity relations. Rao et al.. [19] adopt a model of the world as a Voronoi diagram and present an algorithm for acquiring it based on sonar scan data. The edges are either straight lines, or parabolic arcs. In another paper [18], they adopt a visibility-graph model of the world and its acquisition from sonar scan data. The vertices of the visibility graph are vertices of the polygonal obstacles, whereas edges are either obstacle edges or line segments connecting vertices that can be seen from each other. The robot moves from vertex to vertex and at each step it acquires all edges incident upon the current vertex. Both the Voronoi diagram and the visibility graph are spatial graphs, the vertices of which have Cartesian coordinates associated with them. Neither of the two papers addresses issues of inaccurate sensing: Both assume no error in robot motion, and suggest that the resulting graph can be used for navigation without requiring sensors.

Probabilistic Representations A second category of spatial representation uses concepts from probability theory in order to explicitly represent and manipulate spatial uncertainty, in the form of probability distributions associated with spatial coordinates. Key work in this category has been carried out by Brooks [4], Moravec, Elfes, Matthies, and Shafer [17, 16, 8], Smith et al. [22], and Durrant-Whyte [7].
Brooks [4] proposes a graph representation of free space, where elongated stretches of free space are edges, and convex and compact regions (informally) of free space are vertices. Vertices and edges are further described with metric and relative position and orientation properties. Brooks poses the problem of recognizing a familiar or previously visited location (the "am I there yet" problem), but allows the graph to contain multiple vertices for the same physical location. Moravec [17] and Elfes [8] use certainty grids, where space is uniformly subdivided into rectangular cells. Each cell is assigned a probability of being occupied based on sensory information, therefore spatial uncertainty becomes part of the metric map. Associations between certainty grids from two different robot positions are established by numeric correlation. A Bayesian approach is used to update cell probabilities as more information, possibly from different sensor types, becomes available. Smith et al. [21, 22] assume a stochastic approach to the representation of uncertain spatial relationships, which are represented by probability distributions over their spatial variables. Matthies and Shafer [16] use 3D Gaussian distributions to model the triangulation error for robot self-localization. Similar ideas are explored by Durrant-Whyte [7] in the context of manipulation of uncertain geometric information, and [9, 2] in the context of 3D object recognition.

Graph-based Representations A third category of spatial representation uses topological models for space and approaches map learning as as a graph theoretic problem. Graph theoreticians have examined the complexity required of a machine to traverse a 2- or 3-dimensional space with obstacles [1]. Although different constraints of the world structure and topology are used, the underlying formalism has similarities to the one discussed here.

Rather than attempting to construct a model of the environment, they have designed their automaton simply to visit every accessible cell.

Related is the work on random walks of graphs [3]. A main result there is that the expected number of edge traversals a random walk requires to visit all vertices of a connected undirected graph, beginning at any vertex, is at most n^3, where n is the number of nodes in the graph. A major difference between this work and ours is that we propose systematic ways for visiting vertices of the graph, as opposed to stochastic ones.

Kuipers [14, 15] proposes the use of a topological model of the world. [14] assumes that all nodes in his network correspond to places with distinctive signatures, assuming a very rich perceptual mechanism and the use of a compass for absolute orientation. If two places have the same signature, they are distinguished by exploring some of the surrounding area using a rehearsal procedure.

Levitt [15] addresses representation of spatial information in cross-country environments, which are characterized by distinct landmarks. His graph notation resembles the dual of the complete graph defined by the straight line segments connecting all possible pairs of landmarks and their intersections. That system is geared primarily towards navigation, as opposed to map learning.

III. Informal Description

In order to domain and algorithm dependent problems encountered in making primitive measurements of an environment, and in general with the low level tasks associated with vision and robotics, we assume a more abstract representation of the environment and the robot than is often considered. We assume that the world can be modelled as a collection of isolated locations of interest (rooms) that are connected by featureless pathways (corridors). This simplification of the environment will work well for office-like environments which consist of long sections of featureless hallways which are only distinguished by connections to rooms or other hallways. An essential characteristic of the representation is that there are a limited set of locations at which major control decisions must be made. We assume that low-level processes are available to map local feature measurements (such as edges and parallel lines) made by the robot into this higher level description. The robot should always be able to determine if it is in a room or a hallway, and if it is in a room it should be able to determine the possible exits from that room. We will not assume that the robot has inertial guidance, nor a compass, nor any other mechanism for determining absolute orientation. Thus when a robot enters a room it has no way of determining, for example, which exit is north.

We model the world as a graph embedding consisting of a set of vertices, a set of edges between them and an ordering defined on all edges incident upon each vertex. The ordering captures the relative direction in which edges, viewed as exits, leave a vertex. Full specification of an edge consists of the vertices upon which the edge is incident, as well as the two indices of the edge defined by the edge ordering for each of the two vertices. Note that this description does not restrict us to modelling 2-dimensional environments or planar graph embeddings, since all that is required is some method of ordering the possible exits from a vertex that the robot can apply using its perceptual capability.

The vertices in the world are featureless except for the exits to other vertices. A marker (also referred to as a pebble in the literature) is a unique item that can be picked up or put down at different vertices. We will also assume that there are no cycles of length ≤ 2 in the graph that represents the world, i.e. there are no degenerate or redundant edges in the graph (this simplification is not required, but it simplifies the exposition).

This representation is very minimalist. It is quite likely that many locations will have other, more complex information associated with them. Since the basic problem any environment exploration algorithm has to contend with is the "have I been here before" problem, the existence of additional information can only help to make the problem easier. Hence, the techniques presented here solve what amounts to a worst-case scenario.

Much in the same way that we have assumed that the world can be represented as a graph, we will make simplifying assumptions concerning the robot. In this paper we will only be concerned with its ability to perform certain high-level actions on its environment, and its ability to sense certain things in the environment.

The robot can carry out the following actions:

- move between vertices by traversing edges.
- leave a vertex by a given exit. Exits will be given a local labelling defined by the order of the exits in some standard enumeration scheme starting with the exit by which the robot entered the vertex. Thus, the local labelling will only be the same on occasions when the robot re-enters this vertex along the same edge.
- manipulate a marker located at its current vertex or on the robot.

Note that all of the actions are local. The robot is only able to act on items at the current vertex. Sensation of the robot is local: it can sense the local ordering of the edges incident upon the current vertex. It can sense the presence or absence of the marker. The exploring robot remembers all of its past actions, thereby enabling it to retrace its steps.

Under this model a robotic explorer is capable of carrying out certain complex actions that affect either its position in the environment or the status of the marker. We can specify a particular movement by providing the outgoing edge of the current vertex that is to be traversed. The outgoing edge is specified relative to the edge that the robot used to enter the vertex. It is assumed that the robot can navigate autonomously along the edge from one vertex to another (eg. down a hallway).

The robot can manipulate the marker by either picking it up at the current vertex (if it is lying there), or by dropping it (if it currently holds it).
Sensations of the robot are of two kinds: edge-related and marker-related. The robot can enumerate edges consistently at the current vertex. Thus it can determine the ordering of any adjacent edge with respect to the edge that the robot used to enter the current vertex. As a byproduct of the ability, the robot can also sense the degree of the current vertex. The enumeration of edges is specific to the path by which the robot entered the current vertex. A different enumeration will be obtained if the robot enters the vertex from a different edge. We will, however, assume that enumerations are consistent and unique. This will allow paths through the graph to be invertible.
The robot can also sense whether the marker is present at the current vertex. It can also determine if it carries the marker. Details of how a marker is used will be dealt with later in the paper. Note that the sensory ability of the robot is rather limited. It cannot, for example, measure angles or absolute distances, and it cannot tell anything about the possible embeddings of the graph. Real robots can typically measure a richer set of world properties, which should only make the exploration task easier.
The robot's "purpose in life" is to use its ability to act and to perceive in the graph domain in order to build a graph embedding which is isomorphic [10] to the finite world it has been assigned to explore (finite being equivalent to the environment having a finite number of vertices and edges). The robot's inputs are its sensations and it can interact with the world only through its actions.

IV. Formal Description

In this section we formally describe the environment in which our model robot explorer exists, its actions, "sensations", and its goals. It is important to remember that this representation is deliberately minimalist in nature. In reality much more information is likely to be available at a given location than we assume here.

The World The world is defined as an embedding of an undirected graph G:

$$G = (V, E) \tag{1}$$

with set of vertices V and set of edges E. The vertices are denoted by:

$$V = \{v_0, v_1, ..., v_N\} \tag{2}$$

We will restrict the world model to graphs G that contain no cycles of length ≤ 2, i.e. the graph contains no degenerate or redundant paths. This restriction prohibits the world from having multiple edges between two vertices or an edge incident twice at the same vertex.
The definition of an edge is extended slightly to allow for the explicit specification of the order of edges incident upon each vertex of the graph embedding. This ordering is obtained by enumerating the edges in a systematic (e.g. clockwise) manner from some standard starting direction. An edge $E_{i,j}$ incident upon v_i and v_j is assigned labels n and m, one for each of v_i and v_j respectively. n represents the ordering of the edge $E_{i,j}$ with respect to the consistent enumeration of edges at v_i, m represents the ordering of the edge $E_{i,j}$ with respect to the consistent enumeration of the edges at v_j. The labels m and n can be considered as general directions, e.g. from vertex v_i the n'th exit takes edge $E_{i,j}$ to vertex v_j.

Assuming that the robot started at v_l, then the transition function δ identifies exit r from vertex v_i as being an edge to v_j.

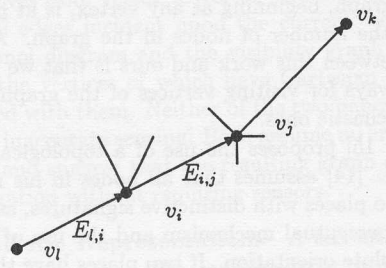


Figure 1: Transition function

Movement and action The robot can move from one vertex to another by traversing an edge (a *move*), it can pick up a marker that is located at the current vertex, and it can put down a marker it holds at the current vertex (a *marker operation*).
Assume the robot is at a single vertex, v_i, having entered the vertex through edge $E_{i,l}$. In a single move, it leaves vertex v_i for vertex v_j by traversing the edge $E_{j,i}$, which is r edges after $E_{i,l}$ according to the edge order at vertex v_i (see Figure 1). This is given by the transition function:

$$\delta(v_i, E_{i,l}, r) = v_j \tag{3}$$

Note that if $\delta(v_i, E_{i,l}, r) = v_j$ and $\delta(v_j, E_{i,j}, s) = v_k$, then the consistent enumeration allows the robot to compute a $-s$ such that $\delta(v_j, E_{j,k}, -s) = v_i$. This implies that a sequence of moves is invertible, and can be retraced. This also allows the robot to move throughout the explored portion of the graph.
A marker operation is fully specified by indicating whether it is being picked up, put down, or not operated upon. This is specified by a $\Omega = (op)$, where op has a value from the set $\{pickup, putdown, null\}$, according to the operation performed on the marker.

A simple action a is defined as a marker operation accompanied by a move. Let $b \in \Omega$ then $a = (b, \delta)$. The robot performs some action on the marker in the current room and then moves to a new location. A path $A \in a^+$ is a nonempty sequence of actions.

Perception The robot's perception can be divided into two portions; a marker based perception, and an edge based perception.

MARKER BASED PERCEPTION

Assume that the robot is at vertex v_i, having arrived via edge $E_{i,j}$. The marker-related perception of the robot is $B_s = (bs)$, where bs has a value from the set $\{present, not{-}present\}$, according to whether the marker is present at vertex v_i.

EDGE BASED PERCEPTION

The robot can determine the relative positions of edges incident on the vertex v_i in a consistent manner, e.g. by a clockwise enumeration starting with $E_{i,j}$. As a result, it can assign an integer to each edge incident with v_i, representing the order of that edge with respect to an enumeration starting at v_i. The label 0 is assigned arbitrarily to the edge $E_{i,j}$, through which the robot entered vertex v_i. Thus for each vertex v_i the robot defines a mapping l_{v_i} (its perception), that maps edge $E_{i,k}$ onto an integer $l_{v_i}(E_{i,k})$, where $0 \leq l_{v_i}(E_{i,k}) < degree(v_i)$. By definition, $l_{v_i}(E_{i,j}) = 0$. The mapping clearly depends on the edge $E_{i,j}$. Entering the same vertex from two different edges will lead to two mappings, one of which is a permutation of the other. Note that if the graph is planar and a spatially consistent (e.g. clockwise) enumeration of edges is used, then two permutations will be simple circular translations of each other. But this will not hold in general, and in this paper we will only assume that the edges are labelled consistently.

The sensory information that the robot acquires while at vertex v_i is the pair consisting of the marker-related perception at that vertex and the order of edges incident on that vertex, i.e. the mapping l_{v_i}.

If the robot visits the same vertex twice, it must relate the two different mappings produced and unify them into a single mapping, i.e. find the label of the 0-th edge of the second mapping with respect to the first mapping. Determining when the same vertex has been visited twice and unifying the two mappings into one is part of the exploration algorithms of the next section.

Robot Memory The robot remembers all raw sensory information that it has acquired so far and all of its actions. Formally, if the robot has performed steps $0, 1, ..., t$, the raw memory of the robot contains the sequence $S_1 S_2 ... S_t$, where $S_i = (a_i, Bs_i, l_{v_i})$, where a_i is the action at the i-th step, Bs_i is the marker sensing at the the i-th step, and l_{v_i} is the order of edges incident on vertex v_i, the robot's position at the i-th step.

Based on this information the robot constructs a model of the world it has explored so far. The details of how the model is constructed depend on the specific exploration algorithm being used. Some exploration algorithms are presented in the following sections. Note that any vertex visited p steps ago can always be found again by simply taking p steps backwards, since individual moves can always be inverted.

These two graphs are indistinguishable without a marker. Each vertex appears identical to every other vertex. Note that there are an infinite number of other graphs that are indistinguishable from these two examples here.

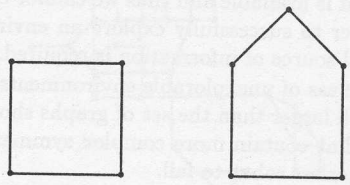


Figure 2: Simple indistinguishable graphs

V. The markerless case

Can an autonomous robot (as described above) explore an arbitrary finite environment without a marker? The answer is no. To see this, consider the two graphs shown in Figure 2. Remember that the only sensations that the robot can make are local. Each vertex is identical in every way; they all have the same degree, and the vertices at the end of the edges are indistinguishable, and so are the vertices at the end of these edges, and so on. Thus no two vertices are distinguishable. The robot is unable to determine the number of vertices in the environment. If the robot was to explore the two environments shown in Figure 2, it would not be able to tell them apart. Since the two environments are different (they have differing numbers of locations), our robot is unable to map these environments. Note that as we have assumed that there are no cycles in the graph of length ≤ 2, the smallest regular graph of degree 2 consists of three vertices. Had we permitted cycles of arbitrary degree, all regular graphs of order 2 would be indistinguishable from the graph consisting of a single node with one edge to itself.

To consider the problem more formally, we extend the notion of signatures proposed by Kuipers [14]. The signature of a given location is a description that encodes all that is known about that location. Two locations must be different if their signatures are different, but they are not necessarily the same if their signatures agree. Consider the possible signatures that such a technique can assign to vertices in the graphs in Figure 2. The only information

that it can obtain (as there are no markers available) is the degree of the current vertex, and the signatures of vertices connected to the current vertex. Note that this signature definition is recursive. As all vertices are identical in every way no two vertices will have different signatures, even if arbitrarily large neighborhoods are taken into consideration. No information disambiguating different locations in the environment is available and thus we cannot construct a map. In order to successfully explore an environment, some additional source of information is required.

Note that the class of unexplorable environments with no markers is much larger than the set of graphs shown here. Many graphs that contain more complex symmetries will cause the markerless robot to fail.

A slightly different problem *is* solvable without markers, however. If we demand of the robot only that the model it builds is indistinguishable from the world as *it* perceives it, then a simple, correct model consisting of three vertices of degree 2 can be built for the graph of Figure 2. Such a model would account for the fact that any movement brings the robot to a vertex indistinguishable from the vertex just left.

VI. The marker-based exploration algorithm

This section describes an algorithm for exploring a given environment. The description of the algorithm includes the robot's world model, the use of markers, the graph traversal strategy , and the representation of incompletely explored edges and vertices. The basis of the algorithm is the maintenance of an explored graph. As new vertices are encountered, they are added to the explored graph and their outgoing edges are added to the set of edges that lead to unknown places and must be explored.

The algorithm approaches the exploration problem by maintaining an explored subgraph of the world S, and a set of unexplored edges U, which emanate from vertices of the explored subgraph. A step of the algorithm consists of selecting an unexplored edge $e = (v_1, v_2)$ from U, and "validating" the vertex v_2 at the unexplored end of the edge. The other vertex v_1 incident with e is already in the subgraph S. Validating a vertex v_2 means making sure that it is not identical to any other vertex in the explored subgraph. Validation is carried out by placing the marker at v_2 and visiting all vertices of the known subgraph S, looking for the marker.

If the marker is found at vertex v_i of the explored subgraph S, then vertex v_2 (where the marker was dropped) is identical to the already known v_i (where the marker was found). In this case, edge $e = (v_1, v_2)$ must be assigned an index with respect to the edge ordering of vertex v_i. Note that the index of e with respect to the edge ordering of v_1 is known by construction. Edge e is then added to the subgraph S, and it is removed from the set of unexplored edges.

If the marker is not found at one of the vertices of S, then vertex v_2 is not in the subgraph S, and therefore must be added to it. The unexplored edge e is also added to S, which has now been augmented by one edge and one vertex. Adding the vertex v_2 to the subgraph causes all edges incident upon it to be assigned an index with respect to the edge e by which the robot entered the vertex (edge e is assigned index 0) and the new edges are added to the set of unexplored edges U. Note that no other edge of the new vertex v_2 has previously been part of the subgraph, because otherwise v_2 would have been in the explored subgraph as well. This index assignment establishes the edge ordering local to v_2.

The algorithm terminates when the set of unexplored edges U is empty. Before we present a formal statement of the algorithm, we will define more precisely the unexplored subgraph S and the set of unexplored edges U, and we will describe the method for ordering edges incident upon a vertex with respect to that vertex.

The explored subgraph S and unexplored edge set U. The explored subgraph consists of

- all vertices which have been validated against all other vertices of the subgraph, and thus guaranteed to be distinct from each other.
- all edges which have been assigned an ordering with respect to both vertices they are incident upon. Of course, both of these vertices belong to the explored subgraph.

The set of unexplored edges U contains edges which have been seen by the algorithm but which are incident with an unexplored vertex. Let $e \in U$. Then one vertex of e will be in the subgraph S, while the other vertex may be in S under a different name, or it may be a new vertex that does not exist in S. This partial edge is held in U to indicate that it requires further exploration. The list of unexplored edges thus consists of pairs (v, e) where v is a vertex in S that caused edge e to be added to U. By definition, e is ordered with respect to v, but not with respect to the other vertex v' it is incident upon. Vertex v' may or may not belong to S.

During exploration, elements are selected from the set of unexplored edges, to be explored. Exploring an element (v, e) means two things: (1) validating the other vertex v' upon which e is incident, and (2) assigning an ordering to e with respect to v'. If v' is not part of S yet, then the assignment is easy, because e becomes the reference edge for v'. If v' is already part of S, then the edge ordering is more difficult as the ordering must conform with any partial ordering already assigned to v'. The ordering method is described in the following section.

Edge ordering. Internally the robot maintains an absolute ordering of the edges from a vertex. This ordering is mapped into the local ordering whenever it is necessary for the robot to traverse an edge. This ordering is also

required when merging a new edge into the explored subgraph S. The robot assigns the edge along which it visits a vertex for the first time as the reference edge for that vertex. The absolute ordering is performed in a consistent manner (i.e. clockwise) relative to the reference edge. Each edge has two indices associated with it, one for each of the two incident vertices. The index i of an edge e with respect to vertex v denotes the ordering of e with respect to the reference edge incident with v. The robot can only sense relative indices, i.e. it cannot sense which is the reference edge. However, the robot is able to reason about the reference edge by consulting the explored subgraph S and by remembering its position and moves. It can never determine the absolute numbering of edges in the graph, but by leaving a vertex and returning to it by another path, it may be able to find the relationship between a pair of edge indices. Using this process repeatedly allows the robot to build up a complete ordering of the edges at any vertex.
The indices of edges that the robot uses to leave vertices are always known. The index of an edge that a robot uses to enter a vertex is either 0, if the robot enters the vertex for the first time, or its value can be determined by the following rehearsal procedure, elaborating on a idea by Kuipers [13].
Assume that edge e is being explored, with v and v' being the two incident vertices, both of them now in the explored subgraph, with v' just validated. Therefore, the index of e with respect to v is known, and what has to be determined is the index with respect to v'. To determine that, the robot drops the marker at v and goes back to v' along the shortest path in the explored graph S. At v', it tries going out of the vertex along each of its incident edges. One of them will take the robot back to v, which the robot will immediately recognize by the existence of the marker. For a more formal treatment of the single-marker algorithm, including a proof of correctness, and computational bounds, the interested reader is directed to [6].

VII. Exploring a complex, unknown environment

In this section we present the actions of a robot using the one marker algorithm to explore a complex environment. We use a planar graph which is shown in Figure 3 to illustrate the robot's performance. The robot was placed in the upper left hand corner vertex, and was set exploring. Internally the robot has no knowledge of the physical position of locations in the environment. It can only measure the exits from a node, and the presence (or absence) of a marker. For display purposes only, we have attached to each node a physical location so that the resulting explored labyrinth can be compared with Figure 3. Two intermediate results (after 288, and 1336 steps) are shown in Figure 4. The robots perception of the maze after completely exploring it is shown in Figure 5. The robot had fully explored the graph after 5134 steps. Once the robot has fully explored the maze it can be instructed to find an optimal path from one location to another, based on simple graph traversal.

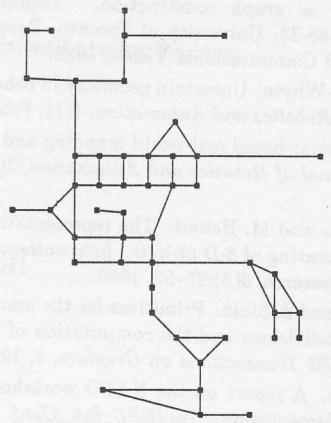


Figure 3: Sample Graph

VIII. Summary

In this paper we proposed an algorithm that a mobile robot can use in exploring unknown graph-like environments. Contrary to most current approaches, our algorithm does not make use of a distance metric, but rather uses a marker which can be perceived by the robot, and be put down or picked up at will. The required perceptual abilities of the robot are well-defined and they amount to the ability to traverse an edge and to enumerate the edges incident upon the current vertex. The success of the approach was demonstrated using a computer simulation of the robot exploring a variety of graphs.

References

[1] M. Blum and W. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In *FOCS Conference*, pages 147–161, 1977.

[2] R. Bolle and D. Cooper. On optimally combining pieces of information, with application to estimating 3-d complex object position from range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:619–638, September 1986.

[3] A. Borodin, S. Cook, P. Dymond, W. Ruzzo, and M. Tompa. Two applications of complementation via inductive counting. Technical Report 13179(no. 58972), IBM Research Division, 1987.

[4] R. Brooks. A robust layered control system for a mobile robot. Technical Report AIM-864, MIT AI Lab, 1985.

[5] J. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, 1(1):31–41, March 1985.

[6] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. Technical Report RBCV-TR-88-23, University of Toronto, Research in Biological and Computational Vision, 1988.

[7] H. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1), February 1988.

[8] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, June 1987.

[9] O. Faugeras and M. Hebert. The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[10] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4, 1984.

[11] S. Harmon. A report on the NATO workshop on mobile robot implementation. In *IEEE Int. Conf. on Robotics and Automation*, pages 604–610, 1988.

[12] S. Iyengar, C. Jorgensen, S. Rao, and C. Weisbin. Learned navigation paths for a robot in unexplored terrain. In *2nd Conf. on Artificial Intelligence Applications*, pages 147–156, 1985.

[13] B. Kuipers. Modelling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[14] B. Kuipers and Y. Byun. A qualitative approach to robot exploration and map-learning. In *Workshop on Spatial Reasoning and Multisensor Fusion*, pages 390–404. Morgan Kaufmann, 1987.

[15] B. Kuipers and T. Levitt. Navigation and mapping in large-scale space. *AI Magazine*, pages 25–43, Summer 1988.

[16] L. Matthies and S. Shafer. Error modelling in stereo navigation. *IEEE Journal of Robotics and Automation*, 3(3):239–248, June 1987.

[17] H. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[18] N. Rao, S. Iyengar, and G. deSaussure. The visit problem: visibility graph-based solution. In *1988 Inf. Conf. on Robotics and Automation*, pages 1650–1655, 1988.

[19] N. Rao, N. Stoltzfus, and S. Iyengar. A retraction method for terrain model acquisition. In *IEEE Int. Conf. on Robotics and Automation*, pages 1224–1229, 1988.

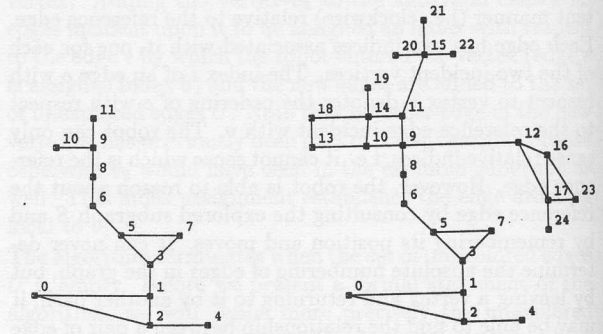
[20] Z. Roth, B. Mooring, and B. Ravani. An overview of robot calibration. *IEEE Journal of Robotics and Automation*, 3(5):377–385, October 1987.

[21] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, Winter 1986.

[22] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Workshop on Spatial Reasoning and Multisensor Fusion*, 1987.

[23] M. Turchan and A. Wong. Low-level learning for a mobile robot: Environment model acquisition. In *2nd Conf. on Artificial Intelligence Applications*, pages 156–161, 1985.

Acknowledgements. The support of the National Sciences and Engineering Research Council of Canada and of the Government of Ontario through the Information Technology Research Centre is gratefully acknowledged.



After 288 steps

After 1336 steps

Figure 4: Partial exploration results

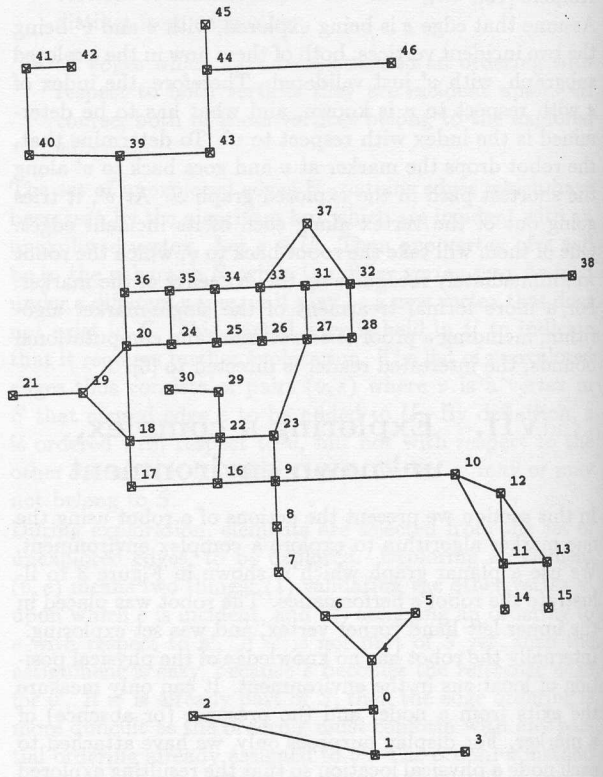


Figure 5: Complete exploration results