

# Coping With Uncertainty in Map Learning

Kenneth Basye<sup>1</sup>   Thomas Dean<sup>2\*</sup>   Jeffrey Scott Vitter<sup>3†</sup>

<sup>1</sup>Department of Mathematics and Computer Science  
Clark University, Worcester, MA 01610  
kbasye@black.clarku.edu

<sup>2</sup>Department of Computer Science  
Brown University, Providence, RI 02912  
tld@cs.brown.edu

<sup>3</sup>Department of Computer Science  
Duke University, Durham, NC, 27708  
jsv@cs.duke.edu

**Running head:** Map Learning

**Keywords:** Inference Learning Maps Graphs Uncertainty Noise

---

\*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 with matching funds from IBM, and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

†This work was supported in part by a National Science Foundation Presidential Young Investigator Award CCR-8846714 with matching funds from IBM, by National Science Foundation research grant CCR-8403613, and by ONR grant N00014-83-K-0146, ARPA Order No. 4786.

## Abstract

In many applications in mobile robotics, it is important for a robot to explore its environment in order to construct a representation of space useful for guiding movement. We refer to such a representation as a *map*, and the process of constructing a map from a set of measurements as *map learning*. In this paper, we develop a framework for describing map-learning problems in which the measurements taken by the robot are subject to known errors. We investigate approaches to learning maps under such conditions based on Valiant's *probably approximately correct* learning model. We focus on the problem of coping with accumulated error in combining local measurements to make global inferences. In one approach, the effects of accumulated error are eliminated by the use of local sensing methods that never mislead but occasionally fail to produce an answer. In another approach, the effects of accumulated error are reduced to acceptable levels by repeated exploration of the area to be learned. We also suggest some insights into why certain existing techniques for map learning perform as well as they do. The learning problems explored in this paper are quite different from most of the classification and boolean-function learning problems appearing in the literature. The methods described, while specific to map learning, suggest directions to take in tackling other learning problems.

# 1 Introduction

Many of the problems faced by robots navigating in the environment can be facilitated by using expectations in the form of explicit models of objects and the spaces that they occupy. We use the term *map* to refer to any model of large-scale space used for purposes of navigation. The construction of useful maps is complicated by the fact that observations involving the position, orientation, and identification of spatially remote objects are invariably error prone. In this paper, we explore a number of problems involved in constructing useful maps from measurements taken with sensors subject to known errors.

In previous work [Dean, 1988], we have looked at various optimization problems related to constructing maps (*e.g.*, construct the most accurate map consistent with a set of measurements). Even in cases involving only a single dimension, such optimization problems can turn out to be NP-hard [Yemini, 1979]. In this paper, rather than look at problems that involve doing the best with what you have, we consider problems that involve going out and getting what you need to generate useful representations.

Map learning, as it is developed here, is different from most other learning tasks that have been examined. In particular, we assume that the robot eventually will see all of the environment it is trying to learn, rather than a small sample. Thus the problem is not so much one of generalization or rule formation as of remembering and sifting through noisy data. We have found, however, that some existing models for learning are applicable to map learning. In particular, we consider forms of probabilistic learning such as *probably approximately correct* learning [Valiant, 1984] and *reliable and probably almost always useful* learning [Rivest and Sloan, 1994] in which the robot gathers information to ensure that it nearly always (with probability  $1 - \delta$ ) can provide a probably good or guaranteed perfect path from one location to another. A prerequisite to this latter sort of learning is that the robot in moving around in its environment can discern the local properties of space with absolute certainty with high probability having expended an amount of effort polynomial in  $\frac{1}{\delta}$  and  $n$ , where  $n$  is some measure of the size of the environment. It would seem that, without this guarantee of being able to eliminate local uncertainty, errors will propagate throughout the map rendering global queries unacceptably inaccurate. To illustrate, we consider the problem of building an accurate map of a simple graph.

In Figure 1, an explorer is constructing a map of the graph shown. Suppose that the explorer begins at A, and explores east to B and C, noting the direction in which he

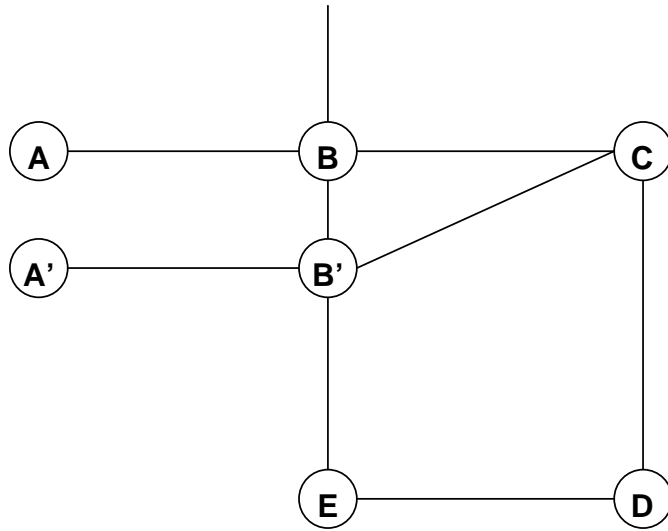


Figure 1: Exploring a graph

is travelling, but not the distance travelled. After turning south and finding D, moving west to E and turning North, he encounters B', a location which looks to him very much like B. Wishing to return to A, but not sure whether he is really at B, he decides to perform an experiment by retracing his steps. Moving east, he finds C, D and, E as he expects, and returns to B' confident that he really is at B. This example illustrates several aspects of the problem of map construction. First, the explorer may have some ability to recognize locations locally. For example, the explorer in the graph above may have the ability to determine the degree of vertices, and might have some general information about the direction of the edges. Thus the explorer in the example can distinguish E from C, but not B from B'. Second, when this is not sufficient, the explorer may use non-local evidence about the identity of the location. For example, the explorer may measure his movements so as to be able to determine that he is now at some previously visited location. In practice, however, these measurements will be error-prone. The explorer can also gather evidence about the identity of a location by moving away from the location in the direction of some known location, or executing some sequence of moves which serves to identify the location. Third, in some cases, simple identification strategies may fail, as in the example given. Here, neither local identification nor a simple retracing strategy allow disambiguation of B and B'.

In this paper, we are concerned with strategies which, with high probability, provide local certainty. Most existing map-learning schemes exploit this sort of certainty in one way or another (see Section 4). The rehearsal strategies of Kuipers [Kuipers and Byun, 1988] are one example of how a robot might plan to eliminate uncertainty. Once we have a method for eliminating uncertainty, the problem then reduces to one of planning out and executing the necessary experiments to extract certain information about the environment. What happens if complete elimination of uncertainty is impossible? In general, the problem is hard, as we have seen above. If, however, there are a number of landmarks distributed about the environment, and these landmarks can be reliably identified by the robot in exploring that environment, then the robot only has to overcome propagation of uncertainty between these landmarks.

In the next section we introduce a model that facilitates the analysis of map learning, and discuss several types of uncertainty and their representations in this model. In Section 3, we begin by analyzing a particular type of map learning that exploits knowledge of the structure of the environment to eliminate local uncertainty. We then go on to show how local uncertainty can be reduced to sufficiently low levels to allow global learning in certain cases where complete elimination is impossible, but there are some landmarks. Finally, in Section 4, we examine several other approaches to the problem of map learning.

## 2 Spatial Representation

We model the world, for the purposes of studying map learning, as a graph with labels on the edges at each vertex. In practice, a graph will be induced from a set of measurements by identifying a set of distinctive locations in the world, and by noting their connectivity. For example, we might model a city by considering intersections of streets to be distinguished locations, and this will induce a grid-like graph. Kuipers [1988] develops a mapping based on locations distinguished by sensed features like those found in buildings (see Figure 2).

Figure 3 shows a portion of a building and the graph that might be induced from it using such a mapping. Levitt [1987] develops a mapping based on locations in the world distinguished by the visibility of landmarks at a distance. In general, different mappings result in graphs with different characteristics, but there are some properties common to most mappings. For example, if the mapping is built for the purpose of navigating on a



surface, the graph induced will almost certainly be planar and cyclic. Other properties may include regularity or bounded degree. In what follows, we will always assume that the graphs induced are connected and undirected; any other properties will be explicitly noted.

Following [Aleliunas *et al.*, 1979], a graph model consists of a graph,  $G = (V, E)$ , a set  $L$  of labels, and a labeling,  $\phi : \{V \times E\} \rightarrow L$ , where we may assume that  $L$  has a null element  $\perp$  which is the label of any pair  $(v \in V, e \in E)$  where  $e$  is not an edge from  $v$ . We will frequently use the word *direction* to refer to an edge and its associated label from a given vertex. With this notation, we can describe a path in the graph as a sequence of labels indicating the edges to be taken at each vertex. We can describe a procedure to follow as a function from  $V \rightarrow L$  indicating the preferred direction to follow from each vertex.

If the graph is a regular tessellation, we may assume that the labeling of the edges at each vertex is consistent, *i.e.*, there is a global scheme for labeling the edges and the labels conform to this scheme at every vertex. For example, in a grid tessellation, it is natural to label the edges at each vertex as North, South, East, and West. In general, we do not require a labeling scheme that is globally consistent. You can think of the labels on edges emanating from a given vertex as local directions. Such local directions might correspond to the robot having a compass that is locally consistent but globally inaccurate, or local directions might correspond to locally distinctive features visible from intersections in learning the map of a city.

The robot's activities are moving about in the world and sensing its environment. To model these activities we introduce functions. A *movement* function is a function from  $\{V \times L\} \rightarrow V$ . The intuition behind this function is that for any location, one may specify a desired edge to traverse, and the function gives the location reached when the move is executed. A *sensor* function is a function from  $V$  to some range of interest. One important sensor function maps vertices to the number of out edges, that is, the degree of the vertex. Another useful function maps vertices to the power set of labels,  $2^L$ , giving the possible directions to take from that vertex. More generally, we may partition the set of vertices into some number of equivalence classes and use a function that maps vertices into these classes. We refer to this as a recognition sensor, since it allows the robot to recognize locations. To introduce uncertainty, we may introduce probabilistic forms of these functions or alter the partitioning. We now develop and explore three kinds of uncertainty that arise in map learning.

## 2.1 Movement Uncertainty

There may be uncertainty in the movement of the robot. In particular, the robot may occasionally move in an unintended direction. We refer to this as *directional* uncertainty, and we model this type of uncertainty by introducing a probabilistic movement function. For example, if  $G$  is a grid with the labeling given above, and we associate the vertices of  $G$  with points  $(i, j)$  in the plane, we might define a movement function as follows:

$$\psi((i, j), l) = \begin{cases} (i, j + 1) & 70\% \text{ of the time if } (l = North) \\ (i + 1, j) & 10\% \text{ of the time if } (l = North) \\ (i - 1, j) & 10\% \text{ of the time if } (l = North) \\ (i, j - 1) & 10\% \text{ of the time if } (l = North) \\ \dots & \dots \end{cases}$$

where the “...” indicate the distribution governing movement in the other three directions.

The probabilities associated with each direction sum to 1. If all directions are equally likely regardless of the intended direction, then the movement function is said to be *random*. Our goal has been to make as few assumptions as possible about the distribution of error in the movement function. Throughout, we assume that the distribution governing movement is static. Other assumptions include the ability to generate an uniform random walk or a lower bound on the probability that movement is in the intended direction; these will be made explicit where they are used.

## 2.2 Recognition Uncertainty

A second source of uncertainty involves sensors, and in particular recognizing locations that have been seen before. The robot’s sensors have some error, and this can cause error in the recognition of places previously visited; the robot might either fail to recognize some previously visited location, or it might err by mistaking some new location for one seen in the past. We refer to this type of uncertainty as *recognition* uncertainty, and we may model it in two different ways. First, we may introduce a new range for the recognition function by partitioning the set of vertices into equivalence classes. We then assume that the function returns the same name for each class, *i.e.*, that the robot is unable to distinguish between elements of a given class using only its sensors. In this case the recognition function maps

vertices to subsets that are the elements of the partition of the set of vertices. For example, a robot that explores the interior of buildings might use sonar as its primary sensor and use hallway junctions as its distinguished locations. In this case, the robot might be able to distinguish an L junction from a T junction, but might be unable to distinguish between two T junctions. In general, expanding the sensor capabilities of the robot will result in better discrimination of locations, *i.e.*, more equivalence classes, but perfect discrimination will likely be either impractical or impossible.

Some locations may be sufficiently distinct that they are distinguishable from all others even with fairly simple sensors. In the model, these locations appear as singleton sets in the partition. We refer to these locations as *landmarks*. We use the term “landmark” advisedly; our landmarks have only some of the usual properties. Specifically, our landmarks are locations that we occupy, not things seen at a distance. They are landmarks because the “view” from them (as opposed to the view of them) is unique. In the following, we make the rather strong assumption that, not only can the robot name the equivalence classes, but it can also determine if a given location is a member of an equivalence class that contains exactly one member (*i.e.*, the robot can identify landmarks). We might also model this kind of uncertainty by having a probabilistic recognition sensor, that is, one that simply gave the wrong name for a vertex with some probability.

### 2.3 Continuity Uncertainty

A third source of error involves another manifestation of sensor error. In representing the world using a graph, some mapping must be established from a set of distinguished locations in the world to  $V$ . Error in the sensors could cause the robot to fail to notice a distinguished location some of the time. For example, a robot taxi might use intersections as distinguished locations, leading to a grid-like graph. But if sensor error causes the robot not to notice that it is passing through an intersection, its map will become flawed. In exploring an office environment, the point in a hallway in front of a door may correspond to a vertex in the induced graph. If the door is closed, there is some chance that the robot will not recognize the vertex in traversing the hall. We model this type of uncertainty by introducing a probabilistic movement function that can skip over vertices. We refer to this type of movement function as *discontinuous* and to the type of uncertainty modeled as *continuity* uncertainty.

Apparently, the three types of uncertainty described above are orthogonal in the sense that none implies or precludes the others. The issues involved in modeling and reasoning about continuity uncertainty are complex and will not be treated further in this paper. In the following, we are concerned with directional and recognition uncertainty.

### 3 Map Learning

For our purposes, a map is a data structure that facilitates queries concerning connectivity, both local and global. Answers to queries involving global connectivity will generally rely on information concerning local connectivity, and hence we regard the fundamental unit of information to be a connection between two nearby locations (*i.e.*, an edge between two vertices in the induced undirected graph). We say that a graph has been *learned completely* if for every location we know all of its neighbors and the directions in which they lie (*i.e.*, we know every triple of the form  $\langle u, l, v \rangle$  where  $u$  and  $v$  are vertices and  $l$  is the label at  $u$  of an edge in  $G$  from  $u$  to  $v$ ). We assume that the information used to construct the map will come from exploring the environment, and we identify two different procedures involved in learning maps: *exploration* and *assimilation*. Exploration involves moving about in the world gathering information, and assimilation involves using that information to construct a useful representation of space. Exploration and assimilation are generally handled in parallel, with assimilation performed incrementally as new information becomes available during exploration. In this section, we are concerned with the conditions under which a graph can be completely learned, and how much time is required for the exploration and assimilation.

#### 3.1 Tessellation Graphs

It's not hard to see that any connected, undirected graph can be completely learned easily if there is no uncertainty. Kuipers and Byun [1988] describe a way of doing this by building up an agenda consisting of unexplored paths leading out of locations and then moving about so as to eventually explore all such paths. Nothing about the graph need be known before the exploration begins. Introducing the kinds of uncertainty described in Section 2 complicates things considerably. If, however, the graph has additional structure, then that structure can often be exploited to eliminate uncertainty. In the following, we assume that

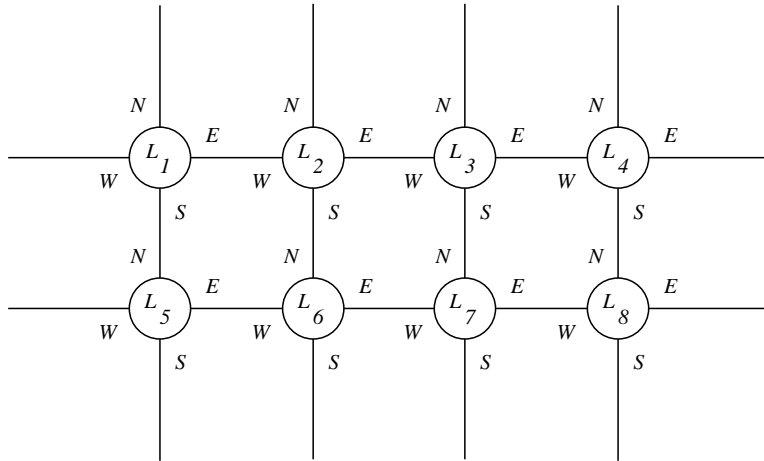


Figure 4: A tessellation of order four

the robot has certain sensing and movement capabilities, modeled by functions. The robot’s movement function is imperfect, that is, the robot has only partial control over which way it moves. Our only assumption concerning movement is that the robot is able to generate a uniform random walk. The robot has a perfect recognition function which allows it to identify any vertex in the graph uniquely when that vertex is reached. The robot also knows the degree of the graph. However, the robot’s information about how it has moved is very limited. Once it moves, it has no way of knowing which direction it went, and no way of knowing which direction it has come from when it arrives at its new location.

We now provide a proof that it is possible, given these capabilities, to efficiently learn maps that correspond to regular tessellations without boundaries.<sup>1</sup> Figure 4 shows a portion of a grid tessellation. It turns out that the exploration component of learning regular tessellations is quite simple; random walks suffice for polynomial-time performance. We first describe an efficient incremental assimilation procedure that is called whenever the robot encounters a new location during exploration, and then prove that this procedure can be used together with a random walk to learn regular tessellations.

The assimilation algorithm uses two data structures: *vertices* and *edges*. Vertices have two fields: `NEIGHBORS` (a list of known adjacent vertices) and `ARCS` (a list of known incident

---

<sup>1</sup>These graphs are closed and finite. A similar technique works for regular tessellations with boundaries, but the proof is more complicated.

edges). Edges have four fields, an `ORIENTATION` (a pair of labels, one at each vertex), `VERTEX1` (one incident vertex), `VERTEX2` (the other incident vertex), and `DEPENDENCY_LIST` (list of edges used during assimilation). All fields not known at the time a data structure is created are initialized to `NIL`.

We say that an edge is *established* when its `ORIENTATION` is fixed (*i.e.*, assigned a particular direction from each vertex). We say that a vertex is established when all of its incident edges are established. Absolute directional information is difficult to come by without the use of an accurate compass. The assimilation algorithm presented here simply picks a random direction the first time it establishes an edge; all subsequent edges are established with respect to this initial assignment. The resulting map is guaranteed correct up to a reflection and rotation. One edge is said to be *dependent* on another when establishing the second will enable us to establish the first. The following procedure takes two locations: the location the robot started in and the location it ended up in as a result of moving. If  $l$  is a location that the robot has already seen, then it has a vertex data structure denoted  $\mathbf{v}(l)$ .

**Procedure:** `assimilate( $l_1, l_2$ )`

1. If  $l_2$  has never been seen before, then create  $\mathbf{v}(l_2)$ .
2. If there already is an edge from  $\mathbf{v}(l_1)$  to  $\mathbf{v}(l_2)$ , then exit, else
  - (a) Add  $\mathbf{v}(l_2)$  to the `NEIGHBORS` of  $\mathbf{v}(l_1)$ .
  - (b) Add  $\mathbf{v}(l_1)$  to the `NEIGHBORS` of  $\mathbf{v}(l_2)$ .
  - (c) Create an edge from  $\mathbf{v}(l_1)$  to  $\mathbf{v}(l_2)$ , and set  $e$  to be that edge.
3. Check the two vertices of  $e$  to determine if either one can be established. A vertex  $v$  can be established if all of the edges out of  $v$  are known and all but one of the edges out of  $v$  are established. If either of  $e$ 's vertices can be established, then  $e$  is established using the `ORIENTATIONS` of adjacent edges and the set `NEW_EDGES` is initialized to `{}` in preparation for propagating the consequences of establishing  $e$ . If neither of  $e$ 's vertices can be established and at least one `ORIENTATION` has already been assigned, then exit as there are no other consequences to be realized.
4. Search the set of edges for other edges that complete a shortest possible cycle involving

$e$ . (A shortest possible cycle for a grid is a cycle of length 4.) Two such cycles are possible. If either or both cycles are found, they are analyzed as follows.

- (a) If this is the first cycle discovered, then an `ORIENTATION` is randomly assigned to  $e$  and the other edges in the cycle are assigned `ORIENTATIONS` in accord with this initial assignment. These edges are now established, and they are added to `NEW_EDGES`, and their `DEPENDENCY_LISTS` are set to `{}`. From now on, all established edges get their `ORIENTATIONS` directly or indirectly from this initial assignment.
  - (b) If two or more adjacent edges in the cycle are established, then all the edges in the cycle can be established. Add all of the newly established edges to `NEW_EDGES` and set their `DEPENDENCY_LISTS` to `{}`.
  - (c) If the cycle contains one established edge, or more than one non-adjacent established edge, then each non-established edge in the cycle is put on the `DEPENDENCY_LIST` of each non-established edge adjacent to an established edge.
5. Establish each edge in the dependency list of  $e$ , add them to `NEW_EDGES`, and set their `DEPENDENCY_LISTS` to `{}`.
  6. If `NEW_EDGES = {}`, then exit, else remove some edge from `NEW_EDGES`, set it equal to  $e$  and return to (4).

We can run `assimilate` over a trace of locations or incrementally. There is no need to keep explicit track of where the robot has been in a chain of locations. If edges are kept in a table indexed by endpoints, then searching for a shortest possible cycle including a vertex  $v$  can be done in constant time:  $c^d$  table lookups, where  $c$  is the length of the shortest cycle, and  $d$  is the order of the graph (*i.e.*, the degree of  $v$  in this case). The worst-case running time of `assimilate` is  $O(|E|)$ , and if `assimilate` is used at each step on a tour of length  $m$  then the overall cost is just  $O(m)$ .

**Lemma 1** *The assimilation algorithm provided will learn a finite tessellation completely if the exploration tour traverses every edge in the graph. The overall cost of assimilation is  $O(m)$  where  $m$  is the length of the tour.*

**Proof:** Traversing every edge in the graph ensures that Step 2 will fill in the NEIGHBORS field for every vertex in the graph. By selecting one shortest cycle in the graph and assigning an orientation (*i.e.*, labeling) to each edge in this cycle, we can propagate outward from that cycle and assign orientations to the remaining cycles. Since all cycles are found in Step 4 if all edges are traversed, every edge in the graph will be labeled. The algorithm is complicated somewhat by the fact that it operates in a greedy fashion, *i.e.*, it propagates orientation as soon as possible.  $\square$

We now have to ensure that during exploration the robot traverses each edge in the graph at least once with high probability. The following two lemmas establish that, for any connected, regular, undirected graph  $G$  and any  $\delta > 0$ , a random walk of length polynomial in  $\frac{1}{\delta}$  and the size of  $G$  is sufficient for traversing every edge in  $G$  with probability  $1 - \delta$ .

**Lemma 2** *For any  $d > 1$ , there exists a polynomial  $p(d, \frac{1}{\delta})$  of order  $O(d \log d \log \frac{1}{\delta})$  such that with probability  $1 - \delta$ ,  $p$  visits to a vertex of order  $d$  result in traversing all edges out of the vertex at least once.*

**Proof:** This is a variation on the Coupon Collector's problem (see [Graham *et al.*, 1994]). We consider only the traversals resulting from leaving the vertex, thus each visit to the vertex results in the traversal of one edge chosen at random. The expected number of visits required for traversing all edges at least once is shown in [Graham *et al.*, 1994] to be  $d\mathbf{H}_d < d \ln d + d$ , where  $\mathbf{H}_d$  is the value of the harmonic function at  $d$ . Let  $k = 2d \ln d + d$ , and let  $P_r$  be the probability that every edge out of the vertex has been traversed at least once given  $r$  visits. By Markov's Inequality,  $P_k \geq \frac{1}{2}$ . Further,  $P_{nk} \geq 1 - \frac{1}{2^n}$  and this will be at least  $1 - \delta$  if  $n \geq \log \frac{1}{\delta}$ . Thus  $nk = O(d \log d \log \frac{1}{\delta})$  visits will suffice.  $\square$

**Lemma 3** *For any connected, regular, undirected graph  $G = (V, E)$  with order  $d$ , any  $\delta > 0$ , and any  $m \geq 1$ , there exists a polynomial  $p(|E|, m, \frac{1}{\delta})$  such that with probability  $1 - \delta$ , a random tour on  $G$  of length  $p$  visits every vertex in  $V$  at least  $m$  times.*

**Proof:** We rely on a result due to Aleliunas, *et al.* [Aleliunas *et al.*, 1979] that establishes that the expected number of steps for an unbiased random walk to traverse every undirected edge in  $E$  is less than or equal to  $2d|V|(|V| - 1)$ . This result combined with Markov's Inequality assures that a tour of length  $4d|V|(|V| - 1)$  traverses every edge in  $E$

with probability greater than  $\frac{1}{2}$ , and, hence, a path of this length visits every node with probability greater than  $\frac{1}{2}$ . If we let  $k$  be this length, then we may consider each tour of length  $k$  as an individual, independent trial, and the probability that every point is hit in  $r$  of these trials is greater than  $1 - (\frac{1}{2})^r$ . If we now consider each of these  $rk$  length tours as independent trials, the probability that  $m$  of these trials will result in every point being hit  $m$  or more times is at least  $(1 - (\frac{1}{2})^r)^m$ . It is this value that we must ensure is larger than  $1 - \delta$ . Expanding and solving for  $r$  shows that  $r = \log_2(\frac{m}{\delta})$  will suffice, making the entire tour of length  $kmr = 4dmn(n - 1) \log_2(\frac{m}{\delta})$ .  $\square$

In most cases, we can do better than random exploration. If the robot moves in the intended direction with probability greater than  $\frac{1}{2}$  then the robot can traverse every edge in the graph with high probability in time linear in the size of the graph. Using the above three lemmas it is easy to prove the following.

**Theorem 1** *It is possible to completely learn any finite regular tessellation  $G = (V, E)$  with probability  $1 - \delta$  in time polynomial in  $\frac{1}{\delta}$  and the size of  $G$ .*

**Proof:** Let  $B_i^m$  denote the event of making  $m$  visits to vertex  $i$ , and  $\mathbf{B}^m$  denote the event of making  $m$  visits to every vertex in the graph. Let  $A_i$  denote the event of traversing all edges incident with vertex  $i$ , and  $\mathbf{A}$  denote the event of traversing all edges in the graph. If  $n = |V|$ , then we have

$$\begin{aligned} \Pr(\mathbf{A}|\mathbf{B}^m) &= \Pr(A_1, A_2, \dots, A_n|\mathbf{B}^m) \\ &= \Pr(A_1|\mathbf{B}^m) \Pr(A_2|A_1, \mathbf{B}^m) \dots \Pr(A_n|A_1, A_2, \dots, A_{n-1}, \mathbf{B}^m) \\ &= \prod_{i=1}^n \Pr(A_i|B_1^m, B_2^m, \dots, B_n^m) \\ &= \prod_{i=1}^n \Pr(A_i|B_i^m). \end{aligned}$$

given that  $\Pr(A_i|B_j^m) = \Pr(A_i)$  and  $\Pr(A_i|A_j) = \Pr(A_i)$  for  $i \neq j$ . In a regular graph, the probability of traversing all edges incident with the  $i$ th vertex given  $m$  visits to the vertex is the same for all vertices, so let  $\Pr(A_i|B_i^m) = \pi$  for all  $i$ , and we have

$$\Pr(\mathbf{A}) \geq \pi^n \Pr(\mathbf{B}^m).$$

Our objective is to find, for a given  $G$  and  $\delta$ , a polynomial  $p$  such that a random walk of length  $p$  on  $G$  allows  $G$  to be learned completely with probability  $1 - \delta$ . Clearly this condition will be met if we can satisfy the following two inequalities:

$$\Pr(\mathbf{B}^m) \geq 1 - \delta/2; \tag{1}$$

$$\pi^n \geq 1 - \delta/2. \tag{2}$$

By Lemma 3, (1) is satisfied by a polynomial number of steps if the number of visits,  $m$ , is polynomial in the relevant variables, whereas (2) will be satisfied if

$$\pi \geq 1 - \frac{\delta}{2n}, \tag{3}$$

and, by Lemma 2, the number of visits  $m$  to satisfy (3) is polynomial in  $n$  and  $\delta$ .  $\square$

The lemmas and form of the proof described above provide a framework for proving that other kinds of graphs can be reliably probably almost always usefully learned in a polynomial number of steps. In general, all we require is that a polynomial number of visits to every vertex provides enough information to learn the graph. Perhaps, the most important lesson to extract from this exercise is that the effects of multiplicative error in learning maps of large-scale space can be eliminated if there is a reliable method for eliminating local uncertainty that works with high probability. The above approach to map learning was inspired by Rivest's model of learning [Rivest and Sloan, 1994], in which complex problems are broken down into simple subproblems that can be learned independently. In order to learn a useful representation of the global structure of its environment, it is sufficient that a robot have reliable and usually effective methods for sensing the local structure of its environment and a method for composing the local structure to generate an accurate global structure. The sensing methods need not always provide useful answers; they need only guarantee that the answer returned is not wrong. The problem then becomes largely one of determining a sequence of sensing and movement tasks that will provide useful answers with high probability. There are situations, however, in which reliable sensing methods are not available, and it is still possible to learn useful maps of large-scale space.

### 3.2 General Graphs

The next problem we look at involves both recognition and directional uncertainty with general undirected graphs. We show that a form of Valiant's probably approximately correct

learning is possible when applied to learning maps. In this section, we consider the case in which movement in the intended direction takes place with probability greater than  $\frac{1}{2}$ , and that, upon entering a vertex, the robot knows with certainty the local name of the edge upon which it entered. We call the latter requirement *reverse movement certainty*. Results for related models are summarized in the next section.

At any point in time, the robot is facing in a direction defined by the label of a particular edge/vertex pair—the vertex being the location of the robot and the edge being one of the edges emanating from that vertex. We assume that the robot can turn to face in the direction of any of the edges emanating from the robot’s location. We also assume that upon entering a vertex the robot can determine with certainty the direction in which it entered. Directional uncertainty arises when the robot attempts to move in the direction it is pointing. Let  $\alpha > 0.5$  be the probability that the robot moves in the direction it is currently pointing. More than 50% of the time, the robot ends up at the other end of the edge defining its current direction, but some percentage of the time it ends up at the other end of some other edge emanating from its starting vertex. While the robot won’t know that it has ended up at some unintended location, it will know the direction to follow in trying to return to its previous location. No further assumptions are made concerning the distribution of the error probability, in particular, it need not be evenly distributed over the remaining edges.

With regard to recognition uncertainty, we assume that the locations in the world are of two kinds, those that can be distinguished, and all others. That is, there is some set of landmarks, in the sense explained above, and all other locations are indistinguishable. We model this situation using a partitioning  $W$  of  $V$  and assuming that we have a sensor function which maps  $V$  to  $W$ . Here  $W$  consists of some number of singleton sets plus the set of all indistinguishable elements. We further assume that a second sensor function allows us to determine whether the current location is or is not a landmark. For convenience, we define  $D$  to be the subset of  $V$  consisting of the landmark vertices and  $I$  to be the subset of  $V$  consisting of the non-landmark vertices. We refer to this kind of graph as a *landmark graph*. We define the *landmark distribution parameter*,  $r$ , to be the maximum distance from any vertex in  $I$  to its nearest landmark (if  $r = 0$ , then  $I$  is empty and all vertices are landmarks). We say that a procedure learns the *local connectivity within radius  $b$*  of some  $v \in D$  if it can provide the shortest path between  $v$  and any other vertex in  $D$  within a radius  $b$  of  $v$ . We say that a procedure learns the *global connectivity of a graph  $G$  within*

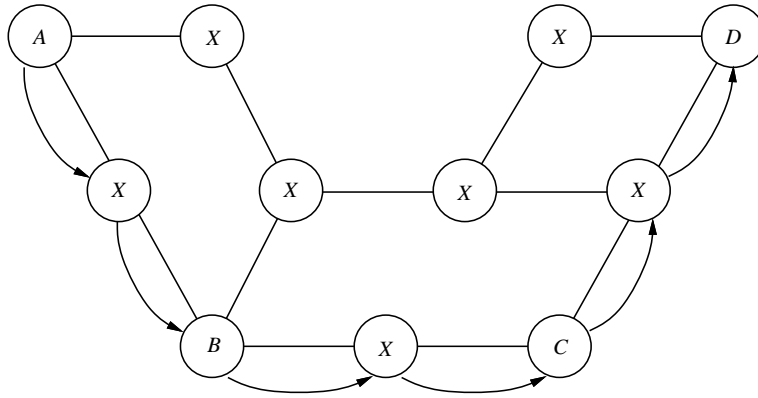


Figure 5: A path found between landmarks  $A$  and  $D$

a *constant factor* if, for any two vertices  $u$  and  $v$  in  $D$ , it can provide a path between  $u$  and  $v$  whose length is within a constant factor of the length of the shortest path between  $u$  and  $v$  in  $G$ . Our procedures will construct such a path from paths found between locally connected landmarks (see Figure 5).

Thus, we may summarize the robot's capabilities as follows. The robot's movement function is not perfect, but serves to move the robot in the intended direction more than half the time. At each vertex, the robot knows how many out-edges there are, and what their labels are. In addition, the robot knows which of these it has just arrived from, whether the vertex is a landmark, and if so, what its unique name is.

We begin by showing that the multiplicative error incurred in trying to answer global path queries can be kept low if the local error can be kept low, that the transition from a local uncertainty measure to a global uncertainty measure does not increase the complexity by more than a polynomial factor, and that it is possible to build a procedure that directs exploration and map building so as to answer global path queries that are accurate and within a small constant factor of optimal with high probability.

**Lemma 4** *Let  $G$  be a landmark graph with distribution parameter  $r$ , and let  $c$  be any integer  $> 2$ . Given a procedure that, for any  $\delta_l > 0$ , learns the local connectivity within  $cr$  of every landmark in  $G$  in time polynomial in  $\frac{1}{\delta_l}$  with probability  $1 - \delta_l$ , it is possible to learn the global connectivity of  $G$  with probability  $1 - \delta_g$  for any  $\delta_g > 0$  in time polynomial*

in  $\frac{1}{\delta_g}$  and the size of the graph. Any global path returned as a result will be at most  $\frac{c}{c-2}$  times the length of the optimal path.

**Proof:** Let  $m$  be the number of local paths required to construct the longest answer we might have to provide to a global query. Then the probability of correctness for any global answer obeys

$$\Pr(\text{correct answer}) \geq (1 - \delta_l)^m. \quad (4)$$

A simple expansion of 4 gives

$$(1 - \delta_l)^m = 1 - m\delta_l + E \geq 1 - m\delta_l,$$

because  $E \geq 0$ . Thus, ensuring that every  $\delta_l \leq \delta_g/m$  will ensure that

$$\Pr(\text{correct answer}) \geq 1 - \delta_g.$$

We use the given procedure to find the local connectivity of every distinguishable vertex in the graph and the resulting representation is sufficient to provide a path between any two distinguishable vertices. Note that we do not have to know  $|V|$  in order to calculate  $\delta_l$ , only the length of the longest answer expected. Any path between two landmarks will use the optimal paths in and out of the intervening landmarks, and we will show that this path will be within a factor  $\frac{c}{c-2}$  of the optimal path.

We now show that the lengths of the paths returned are within the stated limit. Let  $v = v_0, v_1, v_2, \dots, v_j = v'$  be a shortest path in  $G$  from landmark  $v$  to landmark  $v'$ . (It might be that none of the intermediate vertices in the path are landmarks.) We shall show that there is a sequence of landmarks  $v = w_0, w_1, \dots, w_k = v'$ , such that each pair of consecutive landmarks is within distance  $cr$  of one another and that there is a path from  $v$  to  $v'$  of length  $\leq (\frac{c}{c-2})j$  that visits each landmark. This will imply that the global path returned by the above procedure has length at most  $(\frac{c}{c-2})j$ , which will prove the lemma.

Let  $k = \lceil j/(c-2)r \rceil$ . We define the “detour points”  $d_0 = v$ ,  $d_i = v_{i(c-2)r}$ , for  $1 \leq i \leq k-1$ , and  $d_k = v'$ . The definition of the landmark distribution parameter  $r$  tells us that each  $d_i$  is within distance  $r$  of some landmark, call it  $w_i$ . We require that  $w_0 = v$  and  $w_k = v'$ . By transitivity, the distance between  $w_{i-1}$  and  $w_i$ , for  $1 \leq i \leq k$ , is at most  $cr$ , and thus  $w_i$  is “visible” from  $w_{i-1}$  using local connectivity. The distance of the modified

path from  $v$  to  $v'$  that makes a detour at each  $d_i$  to go to landmark  $w_i$  and back is bounded by

$$j + (k - 1)2r \leq j + \frac{j}{(c - 2)r} 2r = \frac{c}{c - 2} j.$$

□

Briefly, the most important stage of the procedure referred to in the proof of Lemma 4 searches outward from a vertex  $v \in D$  to a distance  $cr$ , and then uses the edges found while entering vertices on the outward path to attempt to return to  $v$ . The directions used on the way out form an expectation for the labels observed on the way back. When these expectations are not met, the traversal is said to have failed, and the procedure tries again. The procedure keeps track of the edge/vertex labels associated with vertices visited during exploration in order to ensure that it explores all paths of length  $cr$  or less emanating from each vertex in  $D$  with high probability.

There is a possibility that some combination of movement errors could result in false positive or false negative tests. But we show by exploiting reverse certainty that we can statistically distinguish between the true and false test results. By attempting enough traversals, the procedure can ensure with high probability that the most frequently occurring sets of directions corresponding to perceived traversals actually correspond to paths in  $G$ . What is required, then, is for the learning procedure to do enough exploration to identify all paths of length  $cr$  or less in  $G$  with high probability.

**Lemma 5** *There is a procedure that, for any  $\delta_l > 0$ , learns the local connectivity within  $cr$  of each vertex in any landmark graph with probability  $1 - \delta_l$  in time polynomial in  $\frac{1}{\delta_l}$ ,  $\frac{1}{1 - 2\alpha}$ , and the size of  $G$ , and exponential in  $cr$ .*

**Proof:** The learning algorithm can be broken down into three steps: a landmark identification step in which the robot finds and identifies a set of landmarks, a candidate selection step in which the robot finds a set of candidate paths in  $G$  connecting landmarks, and a candidate filtering step in which the robot determines which of those candidate paths correspond to actual paths in  $G$ . In order to prove the lemma, landmark identification has to succeed in identifying all landmarks in  $G$  with high probability, candidate selection has to find all paths (or at least all of the shortest paths) between landmarks with high probability, and candidate filtering has to determine which of the candidates correspond to

actual paths in  $G$  with high probability. We define  $1 - \delta_i$ ,  $1 - \delta_s$ , and  $1 - \delta_f$ , respectively, to be the desired lower bounds on the probabilities that the three steps succeed in performing their associated tasks. It suffices to set  $\delta_i = \delta_s = \delta_f = \frac{1}{3}\delta_l$ . We will consider each of the three steps in turn.

The first step is easy. The robot identifies all the landmarks in  $G$  with probability  $1 - \delta_i$  by making a random walk whose length is polynomial in  $\frac{1}{\delta_i}$  and the size of  $G$ . A more sophisticated exploration might be possible, but a random walk suffices for polynomial-time performance.

Having identified a set of landmarks, the robot enters the candidate selection step, in which it tries all paths of length  $cr$  starting from each identified landmark  $A$ . If  $d$  is the maximum degree of any vertex in  $G$ , then there can be as many as  $d^{cr}$  paths of length  $cr$  starting from  $A$ . Since we expect that  $cr$  will generally be small, this “local” exponential factor should not be critical. The robot systematically tries all paths of length  $cr$  trying to connect other landmarks within a radius  $r$ . For each landmark (other than  $A$ ) encountered in a particular traversal, the robot records as a *candidate* path the subpath that it thinks it traversed from  $A$  to that landmark. For a given starting landmark  $A$ , the resulting candidates have the following form:

$$A_{out_0}, in_1 X_{out_1}, \dots, in_{k-1} X_{out_{k-1}}, in_k B, \quad (5)$$

where  $B$  is the ending landmark, and  $in X_{out}$  indicates that the robot *observed* itself entering a vertex of type  $X$  on the arc labeled *in* and it *observed* itself attempting to leave on the arc labeled *out*.

The robot has to make enough attempts  $n$  to traverse each possible path of length  $cr$  so that with high probability it records all the actual paths of length  $\leq cr$ . For each path of length  $cr$  starting from  $A$  that the robot attempts to traverse, the probability that it correctly traverses that path during a given attempt is  $\alpha^{cr}$ . Thus, the probability that the robot will correctly traverse that path at least once in  $n$  attempts is

$$1 - (1 - \alpha^{cr})^n.$$

In order to ensure that we record all such paths with probability  $1 - \delta_s$  we have to ensure that all possible paths of length  $cr$  are traversed at least once, which happens with probability at least

$$(1 - (1 - \alpha^{cr})^n)^{d^{cr}}.$$

This probability will be greater than  $1 - \delta_s$  if

$$n \geq \frac{1}{\alpha^{cr}} \left( 2r \log d + \log \left( \frac{1}{\delta_s} \right) \right).$$

The goal of the next step, candidate filtering, is to determine for each candidate path  $P$ , of the form (5), whether  $P$  corresponds to an actual path in the graph. The robot systematically tries  $n$  times to traverse each candidate  $P$ . That is, starting from  $A$  it attempts to take the following sequence of arcs:

$$out_0, out_1, \dots, out_{k-1}.$$

During each attempt, the robot checks whether its reverse observations along the way are consistent with the desired bidirectional path  $P$  in (5); that is, at the  $j$ th vertex on the path, for  $1 \leq j \leq k$ , the robot checks whether the incoming arc is labeled  $in_j$ . The robot also checks that it ended up at landmark  $B$ . If all the observations are consistent during a particular attempted traversal of  $P$ , we say that the traversal of  $P$  is *successful*. We denote by  $count(P)$  the number of attempted traversals of  $P$  (out of  $n$ ) that are successful.

In order for a traversal of  $P$  by the robot to be successful, all the reverse observations along the path, which are made with complete accuracy, must be consistent with (5). This means that the reverse observations must be  $in_1, in_2, \dots, in_k$ . When taken in reverse order these observations form a path  $Q$  from  $B$  to  $A$  given by the arcs labeled

$$in_k, in_{k-1}, \dots, in_1.$$

We let  $\overline{Q}$  denote the reversal of path  $Q$ ; note that  $\overline{Q}$  is a path from  $A$  to  $B$ . This means that the only possible way that an attempted traversal of  $P$  can be successful is if  $\overline{Q}$  is actually traversed.

If  $P$  is an actual path in the graph, then  $\overline{Q}$  is the same as the forward traversal of  $P$ . Thus, the probability that the traversal is successful (which is a “true positive” result) is the probability that the robot correctly traverses each forward arc in  $P$ , which happens with probability  $\gamma_{true} = \alpha^k$ .

The other case is when  $P$  is not an actual path in the graph. The traversal will appear to be successful (which we call a “false positive” result) when the robot actually traverses  $\overline{Q}$  when it tries to traverse  $P$ . But  $\overline{Q}$  is different from a forward traversal of  $P$

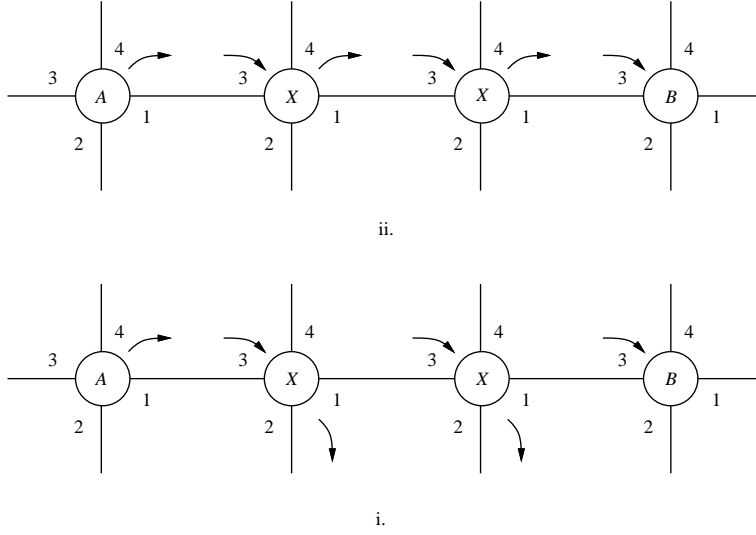


Figure 6: Path traversals with (i) and without (ii) errors

since  $P$  is not an actual path in the graph. Let  $m \geq 1$  be the number of arcs in which  $\bar{Q}$  differs from a forward traversal of  $P$ . The probability of a false positive is thus at most  $\gamma_{false} = \alpha^{k-m}(1 - \alpha)^m \leq \alpha^{k-1}(1 - \alpha)$ .

Figure 6.i illustrates the case in which the robot thinks it traversed  $A_1, {}_3X_2, {}_3X_2, {}_3B$  when in fact it traversed  $A_1, {}_3X_1, {}_3X_1, {}_3B$ . In order to get a false positive, the robot has to make exactly the same mistakes that it made when it originally tried to traverse  $A_1, {}_3X_2, {}_3X_2, {}_3B$  during candidate selection. Figure 6.ii illustrates the case in which the robot thinks it traversed  $A_1, {}_3X_1, {}_3X_1, {}_3B$  and is not mistaken.

We now show that the false positives can be statistically distinguished from true positives in a reasonable amount of time. After  $n$  attempted traversals of a path  $P$ , if the path  $P$  is an actual path in  $G$  the expected value of  $count(P)$  is  $n\gamma_{true}$ , whereas if the path  $P$  is not an actual path in  $G$  the expected value of  $count(P)$  is at most  $n\gamma_{false}$ . We set  $\tau = \frac{1}{2}(\gamma_{true} + \gamma_{false})$  and using  $n\tau$  as a threshold, we include in our constructed representation only those candidate paths  $P$  for which  $count(P) > n\tau$ . By making  $n$  sufficiently large, we can assure that this filtering accepts all and only real paths with the desired probability,  $1 - \delta_f$ .

We now show that the number of times  $n$  that the robot must attempt to traverse each

$P$  during the candidate traversal step is polynomial in  $\frac{1}{\delta_f}$ ,  $\frac{1}{(2\alpha-1)}$ ,  $|D|$ , and is exponential in  $r$ . If the path  $P$  is actually in  $G$ , then  $P$  is wrongfully excluded from the constructed representation if the number  $\text{count}(P)$  of successful traversals of  $P$  is at most  $n\tau$ . The quantity  $\text{count}(P)$  is a Bernoulli distributed random variable corresponding to  $n$  trials, each with probability  $\gamma_{\text{true}}$  of success; it has mean  $n\gamma_{\text{true}}$  and standard deviation  $(n\gamma_{\text{true}}(1-\gamma_{\text{true}}))^{1/2}$ . By Hoeffding's inequality [Hoeffding, 1963] we have

$$\Pr(\text{error on } P) = \Pr(\text{count}(P) - n\gamma_{\text{true}} \leq -n(\gamma_{\text{true}} - \tau)) \leq e^{-2n(\gamma_{\text{true}} - \tau)^2}.$$

Substituting  $\gamma_{\text{true}} - \tau \geq \alpha^{k-1}(\alpha - \frac{1}{2}) \geq \alpha^{cr-1}(\alpha - \frac{1}{2})$ , we find that

$$\Pr(\text{error on } P) \leq e^{-2n(\alpha^{cr-1}(\alpha - \frac{1}{2}))^2}.$$

When  $P$  is not an actual path in the graph, the probability  $\Pr(\text{error on } P)$  of wrongly including a path in the constructed representation can be bounded in the same fashion.

Thus the probability of correctly including in the constructed representation all and only the actual candidate paths in  $G$  is at least

$$1 - |D|d^{2cr} \Pr(\text{error on } P) \geq 1 - |D|d^{2cr} e^{-2n(\alpha^{cr-1}(\alpha - \frac{1}{2}))^2},$$

which is greater than  $1 - \delta_f$  when

$$n > \frac{1}{2(\alpha^{cr-1}(\alpha - \frac{1}{2}))^2} \left( \ln \frac{1}{\delta_f} + 2cr \ln d + \ln |D| \right). \quad \square$$

**Theorem 2** *It is possible to learn the global connectivity of any landmark graph with probability  $1 - \delta$  in time polynomial in  $\frac{1}{\delta}$ ,  $\frac{1}{1-2\alpha}$ , and the size of  $G$ , and exponential in  $r$ .*

Theorem 2 is a simple consequence of Lemma 4 and 5. It has an immediate application to the problem of learning the global connectivity of a graph where all the vertices are landmarks. In this case, the parameter  $r = 0$ , and we need only explore paths of length 1 in order to establish the global connectivity of the graph. Because each candidate path has length one, this process works even if there is no reverse certainty.

**Corollary 1** *It is possible to learn the connectivity of a graph  $G$  with only distinguishable locations with probability  $1 - \delta$  in time polynomial in  $\frac{1}{\delta}$ ,  $\frac{1}{1-2\alpha}$ , and the size of  $G$ , even if there is reverse uncertainty.*

The notion of global connectivity defined above does not require that the graph be *completely learned*, (i.e., to recover the structure of the entire graph). It is assumed that the indistinguishable vertices are of interest only in so far as they provide directions necessary to traverse a direct path between two landmarks. But it is easy to imagine situations where the indistinguishable vertices and the paths between them are of interest. For instance, the indistinguishable vertices might be partitioned further into equivalence classes so that one could uniquely designate a vertex by specifying its equivalence class and some radius from a particular global landmark (e.g., the bookstore just across the street from the Chrysler building).

We can modify the above approach and try to completely learn the graph by first completely learning local neighborhoods of each landmark. Let us define  $G_d(v)$ , for any positive integer  $d$ , to be the subgraph of  $G$  consisting of all vertices and edges within radius  $d$  of  $v$ .

**Lemma 6** *Let  $G$  be a landmark graph with distribution parameter  $r$ . Given a procedure that, for any  $\delta_l > 0$ , completely learns  $G_{2r+1}(v)$  for each landmark  $v \in G$  in time polynomial in  $\frac{1}{\delta_l}$  with probability  $1 - \delta_l$ , it is possible to completely learn  $G$  with probability  $1 - \delta_g$  for any  $\delta_g > 0$  in time polynomial in  $\frac{1}{\delta_g}$  and the size of  $G$ .*

The proof for this lemma is a simple variation on the proof of lemma 4.

The above algorithms used for determining the local connectivity of landmarks can be thought of as building a search tree emanating from each landmark, in which each indistinguishable vertex in  $G$  may correspond to several vertices in the search tree. In order to completely learn  $G_{2r+1}(v)$  (as opposed to just learning the local connectivity), we must avoid this redundant representation. Again, the idea is to systematically explore each path away from each landmark. But in this case, each time the robot encounters a vertex that is not a landmark along a new path, it must check to see if that vertex is actually one that it has come upon previously by some other path.

It turns out that we can extend our methods to completely learn  $G_{2r+1}(v)$ . The algorithm builds  $G_{2r+1}(v)$  via an incremental breadth-first search in which each vertex encountered is tested via repeated walks from  $v$  to determine with high probability if it has already been added to  $G_{2r+1}(v)$ . The proof requires a careful examination of the probabilities of true and false test results.

**Lemma 7** *There is a procedure that, for any  $\delta_l > 0$  and  $c > 2$ , completely learns  $G_{cr}(v)$  for each landmark  $v$  in a landmark graph with probability  $1 - \delta_l$  in time polynomial in  $\frac{1}{\delta_l}$ ,  $\frac{1}{1-2\alpha}$  and the size of  $G_{cr}$ , and exponential in  $cr$ .*

**Proof:** The robot can identify all landmarks as in Lemma 5. Let us turn our attention to the incremental step in the breadth-first search. Suppose that the robot has already determined that the bidirectional path  $P$

$$A_{out_0}, in_1 X_1 out_1, \dots, in_{k-1} X_{k-1} out_{k-1}, in_k X_k \quad (6)$$

is actually in the graph, where  $A$  is a landmark and  $X_1, \dots, X_k$  are *distinct* vertices for  $k < cr$ . The robot is currently attempting to determine if the extended path  $P'$

$$A_{out_0}, in_1 X_1 out_1, \dots, in_{k-1} X_{k-1} out_{k-1}, in_k X_k out_k, in_{k+1} X_{k+1} \quad (7)$$

is actually in the graph, such that  $X_{k+1}$  is distinct from all vertices encountered previously in the breadth-first search.

This is done in two steps. In the first step, called *path filtering*, the robot determines if there is an outgoing arc from  $X_k$  labeled  $out_k$  that corresponds to the incoming arc  $in_{k+1}$  into  $X_{k+1}$ . The robot tries  $n$  times to traverse the path of length  $2(k+1)$  consisting of  $P'$  followed by its reversal. (The value of  $n$  will be determined later.) The outgoing arcs it attempts to take, starting from node  $A$ , are labeled

$$out_0, out_1, \dots, out_k, in_{k+1}, in_k, \dots, in_1. \quad (8)$$

During each round-trip traversal, the robot checks whether its reverse observations along the way are consistent with the desired bidirectional path  $P'$ , as follows: At the  $j$ th vertex on the path, for  $1 \leq j \leq k+1$ , the robot checks whether the incoming arc is labeled  $in_j$ , and for  $k+2 \leq j \leq 2(k+1)$ , the robot checks whether the the incoming arc is labeled  $out_{2(k+1)-j}$ . The robot also checks whether the path ends at landmark  $A$ . If all the observations are consistent during a particular round-trip traversal of  $P'$ , we say that the round-trip traversal of  $P'$  is *successful*. We denote by  $count(P')$  the number of attempted round-trip traversals of  $P'$  (out of  $n$ ) that are successful.

In order for a round-trip traversal of  $P'$  by the robot to be successful during the candidate traversal step, all the reverse observations along the path must be consistent with (7).

In particular, as pointed out above, the  $2(k+1)$ st incoming arc must be labeled  $out_0$ , the  $(2k+1)$ st incoming arc must be labeled  $out_1$ , and so on. Thus the sequence of arcs actually traversed by the robot must be the *reverse* of the one specified in (8); no other possible successful traversals are possible without some observation being inconsistent.

If  $P'$  is an actual path in the graph, the reversal of (8) is equal to itself. Thus, the probability that the round-trip traversal is successful (which is a “true positive” result) is the probability that the robot correctly traverses each arc it attempts to traverse, which happens with probability  $\gamma_{true} = \alpha^{2(k+1)}$ .

The other case is when  $P'$  is not an actual path in the graph. The reversal of (8) is now different from (8), since  $P'$  is not an actual path in the graph. Let  $m \geq 2$  be the number of arcs in which the reversal of path (8) differs from (8). The probability of a “false positive” result, in which the reverse of (8) is traversed, is at most  $\gamma_{false} = \alpha^{2(k+1)-m}(1-\alpha)^m \leq \alpha^{2k}(1-\alpha)^2$ .

A statistical argument similar to that in Lemma 5 shows that if

$$n > \frac{1}{2(\alpha^{2cr-1}(\alpha - \frac{1}{2}))^2} \left( \ln \frac{1}{\delta_p} + 2 \ln d + \ln |V| + \ln |D| \right),$$

then with probability  $1 - \delta_p$  path filtering will correctly determine for all extensions  $P'$  of the form (7) whether there is an outgoing arc from  $X_k$  labeled  $out_k$  that corresponds to the endpoint vertex’s arc  $in_{k+1}$ .

The next step, which we call *redundancy filtering*, determines whether  $X_{k+1}$  has already been processed during the breadth-first construction, and if so, which previous vertex it is. For each vertex  $Y$  previously encountered in the breadth-first search, let  $R$  be the path ending at  $Y$  that first reached  $Y$  in the breadth-first search. We shall denote that path by

$$R = A_{out_0}, in_1 Y_1 out_1, \dots, in_{k-1} Y_{j-1} out_{j-1}, in_j Y_j. \quad (9)$$

Here  $Y = Y_j$ . The robot attempts  $n$  times to traverse the path consisting of  $P'$  followed by the reversal of  $R$ . It checks its observations along the way to see if they’re consistent with (7) and (9) and if the traversal started and ended at  $A$ . If so the traversal is successful and it increments  $count(P', R)$ .

Arguments similar to those above show that all successful traversals must follow the reverse of the path consisting of  $R$  followed by the reverse of  $P'$ . This allows us to statistically

distinguish true positives (the case  $X_{k+1} = Y$ ) from false positives (the case  $X_{k+1} \neq Y$ ). In order for all the redundancy filterings to be correct with probability  $1 - \delta_d$ , it suffices for

$$n > \frac{1}{(2\alpha^{2cr-1}(\alpha - \frac{1}{2}))^2} \left( \ln \frac{1}{\delta_d} + \ln d + 2 \ln |V| + \ln |D| \right).$$

This completes the proof of Lemma 7.  $\square$

**Theorem 3** *It is possible to completely learn any landmark graph with probability  $1 - \delta$  in time polynomial in  $\frac{1}{\delta}$ ,  $\frac{1}{1-2\alpha}$ , and the size of  $G$ , and exponential in  $r$ .*

**Proof:** Theorem 3 is a simple consequence of Lemma 6 and 7.  $\square$

### 3.3 Extensions

We can get the same results as in the last section if we allow uncertainty in the reverse direction, but demand forward movement certainty. The algorithms are similar, the justifications different. In this case, the graph can be reliably navigated by the same agent that did the map learning.

We note that the purpose of either certainty requirement is to allow the robot to filter out paths reported by sensors that aren't really in the graph. In [Basye and Dean, 1989], we treat the problem of map learning given a probabilistic oracle which, for any given traversal, provides an answer to the question "was that traversal free of error" which is correct with probability greater than  $\frac{1}{2}$ . In this paper, the certainty requirements introduced provide such an oracle.

We are also investigating ways to remove the requirement of either reverse certainty or forward certainty. Reverse certainty is used in the last section to help distinguish probabilistically between true and false results in our testing procedures. We can show, for example, that if  $r(1 - \alpha)$  is bounded by a small constant, then efficient map learning is possible without either the reverse certainty or forward certainty requirement. Another way around this restriction is to allow the exploring agent to drop pebbles or beacons to remember where it has been [Dudek *et al.*, 1988].

We can make map learning somewhat easier and more realistic by assuming that the robot can do more work or take more time to get better measurements. This suggests an

interesting optimization problem to work on: given a probabilistic performance requirement, generate an exploration strategy that meets that requirement, minimizing some measure of cost. We are just beginning to work on this problem.

It should be noted that we make very few assumptions regarding the structure of the graphs that we are trying to learn. In most real applications, however, there is a great deal of additional information that can be exploited to expedite learning. Our intent here is to try to provide positive results about general classes of map-learning problems without relying on a great deal of domain-specific knowledge.

## 4 Related Work

There have been many approaches to dealing with uncertainty in spatial reasoning [Brooks, 1984, Davis, 1986, Kuipers, 1978, McDermott and Davis, 1982, Smith and Cheeseman, 1986, Durrant-Whyte, 1988, Moravec and Elfes, 1985], but most of these methods suffer from the effects of multiplicative error in estimating relative position and orientation. This paper is concerned with eliminating or limiting the effects of multiplicative error either by eliminating local uncertainty altogether, or by using a set of distributed landmarks to reduce the effects of local uncertainty. In this section, we consider three related approaches.

Kuipers defines the notion of “place” in terms of a set of related visual events [Kuipers, 1978]. This notion provides a basis for inducing graphs from measurements. In Kuipers’ framework [Kuipers and Byun, 1988], locations are arranged in an unrestricted planar graph. There is recognition uncertainty, but there is no directional uncertainty (if a robot tries to traverse a particular hall, then it will actually traverse that hall; it may not be able to measure exactly how long the hall is, but it will not mistakenly move down the wrong hall). Kuipers goes to some length to deal with recognition uncertainty. To ensure correctness, he has to assume that there is some reference location that is distinguishable from all other locations. Since there is no directional uncertainty, any two locations can be distinguished by traversing paths to the reference location. Given a procedure that is guaranteed to uniquely identify a location if it succeeds, and succeeds with high probability, we can show that a Kuipers-style map can be reliably probably almost always usefully learned using an analysis similar to that of Section 3.

Levitt *et al* [Levitt *et al.*, 1987] describe an approach to spatial reasoning that avoids

multiplicative error by introducing local coordinate systems based on landmarks. Landmarks correspond to environmental features that can be acquired and, more importantly, reacquired in exploring the environment. Given that landmarks can be uniquely identified, one can induce a graph whose vertices correspond to regions of space defined by the landmarks visible in that region. If the identification and reacquisition of landmarks is guaranteed, then the problem involves neither recognition nor movement uncertainty. (The robot will not cross the line between one pair of landmarks thinking that it has crossed the line between some other pair of landmarks.) Of course, in an outdoor environment, landmark identification and reacquisition are both situation and aspect dependent (*i.e.*, the current environmental factors and the direction from which the robot views a scene influence both recognition and reacquisition). In such cases, some amount of recognition uncertainty will certainly manifest itself.

Dudek *et al* [1988] consider the problem of learning a graph in which all vertices are indistinguishable and upon entering a vertex the robot can leave by any arc indexed from the one it entered on. The robot can always retrace its steps if it remembers the directions it took at each point during exploration. The authors show that the problem is unsolvable in general, but that by providing the robot with a number of distinct markers ( $k \geq 1$ ) the robot can learn the graph in time polynomial in the graph's size. In order to place a marker on a particular vertex, the robot must visit that vertex; in order to recover the marker at later time, the robot must return to the vertex. A vertex with a marker on it acts as a temporary landmark. No assumption is made regarding the planarity of the graph. The problem with a single marker that can be placed once but not recovered is also unsolvable, but, if you allow a compass in addition, the problem can be solved in polynomial time.

The sensor and movement functions presented in this paper are primarily useful for the analysis of map learning in which sensing is local and the environment restricts the choice of movement at any given location to a small number of options (*e.g.*, office-like environments). In Kuipers' work landmarks are few and may be difficult to identify, but movement errors are nonexistent. In Levitt's work, landmarks are plentiful and easy to identify, metric information is error prone, but the uncertainty is small. We are currently considering how to apply our methods to the type of problem discussed in Levitt's work, but there are a number of rather difficult barriers to be crossed.

We would like it to be the case that in exploring a bounded area, the robot generates a map whose size is a polynomial function of, say, the number of distinctive places in that

bounded area. Most map-learning algorithms can not guarantee this. We can provide such a guarantee for our algorithm, but only by making two rather restrictive assumptions: first, that the robot is able to correctly identify landmarks, and, second, that the robot knows the maximum out degree of any vertex in the induced graph. We believe that if the robot has information about the spatial distribution of locations satisfying certain perceptible properties and that distribution satisfies certain basic criteria, then it is possible to (probably approximately correctly) learn a map in polynomial time. The idea is that if a robot knows enough about the spatial distribution of the different types of locations, it should be able to identify and reacquire landmarks with a greater than  $\frac{1}{2}$  probability of success; this degree of competence should be sufficient for efficient map learning. Whether or not this is a reasonable assumption is primarily a function of the environment and the power of the robot's sensors. Our feeling is that there are interesting environments in which currently practical sensors are sufficient for competent map learning.

## References

- [Aleliunas *et al.*, 1979] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, Laszlo Lovasz, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the 20th Symposium on the Foundations of Computer Science*, pages 218–223, 1979.
- [Basye and Dean, 1989] Kenneth Basye and Thomas Dean. Map learning with indistinguishable locations. In *Proceedings of the 1989 Workshop on Uncertainty in Artificial Intelligence*, pages 7–13, 1989.
- [Brooks, 1984] Rodney A. Brooks. Aspects of mobile robot visual map making. In H. Hanafusa and H. Inoue, editors, *Second International Symposium on Robotics Research*, pages 325–331, Cambridge, Massachusetts, 1984. MIT Press.
- [Davis, 1986] Ernest Davis. *Representing and Acquiring Geographic Knowledge*. Morgan-Kaufmann, Los Altos, California, 1986.
- [Dean, 1988] Thomas Dean. On the complexity of integrating spatial measurements. In *Proceedings of the SPIE Conference on Advances in Intelligent Robotic Systems*. SPIE, 1988.

- [Dudek *et al.*, 1988] Gregory Dudek, Michael Jenkins, Evangelos Milios, and David Wilkes. Robotic exploration as graph construction. Technical Report RBCV-TR-88-23, University of Toronto, 1988.
- [Durrant-Whyte, 1988] Hugh F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer, Boston, Massachusetts, 1988.
- [Graham *et al.*, 1994] Ronald L. Graham, Donald E. Knuth, and Patashnik Oren. *Concrete Mathematics*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1994.
- [Hoeffding, 1963] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.
- [Kuipers and Byun, 1988] Benjamin J. Kuipers and Yung-Tai Byun. A robust, qualitative method for robot spatial reasoning. In *Proceedings AAAI-88*, pages 774–779. AAAI, 1988.
- [Kuipers, 1978] Benjamin Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.
- [Levitt *et al.*, 1987] Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, and Philip C. Nelson. Qualitative landmark-based path planning and following. In *Proceedings AAAI-87*, pages 689–694. AAAI, 1987.
- [McDermott and Davis, 1982] Drew V. McDermott and Ernest Davis. Planning routes through uncertain territory. *Artificial Intelligence*, 22:107–156, 1982.
- [Moravec and Elfes, 1985] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation*, pages 138–145, 1985.
- [Rivest and Sloan, 1994] Ronald L. Rivest and Robert Sloan. A formal model of hierarchical concept learning. *Information and Computation*, 114(1):88–114, 1994.
- [Smith and Cheeseman, 1986] Randall Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5:56–68, 1986.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[Yemini, 1979] Yechiam Yemini. Some theoretical aspects of position-location problems. In *Proceedings of the 20th Symposium on the Foundations of Computer Science*, pages 1–7, 1979.