

Topological Mapping with Sensing-Limited Robots

Wesley H. Huang and Kristopher R. Beevers

Rensselaer Polytechnic Institute
110 8th Street, Troy, New York 12180, U.S.A.
{whuang, beevk}@cs.rpi.edu

Abstract. Most mobile robot mapping and exploration research makes use of long-range, powerful sensors such as laser rangefinders to create maps. In this paper we take a different approach, creating maps using robots with limited sensing capabilities, most notably in their sensor range. Our prototype robots use only five infrared range sensors with a maximum range of 80 cm; in general, these robots cannot “see” both sides of a hallway. We present an algorithm for such a robot to build a topological map in the presence of sensing and odometry error. In doing so, we develop a paradigm to extend topological mapping to open spaces, long considered a deficiency of topological mapping; we also introduce an evidential approach to the problem of “closing the loop.”

1 Introduction

Imagine you are navigating through your house in the dark — you’ve woken up in the middle of the night, hungry for a snack. You get out of bed, but leave the lights off so you don’t wake everyone else. You can’t see anything, so you run one hand along the wall as you walk down the hallway. There are small discontinuities in the wall, like the closet door, but you keep walking straight. You feel the wall “end,” so you turn the corner and keep going. Perhaps at some point, you leave the wall and walk across a (known obstacle-free) room, holding your hand out in front to detect when you reach the opposite wall. You eventually reach the kitchen without much trouble.

This blind navigation is an apt analogy for how a sensing-limited robot “sees” the world. By “sensing-limited,” we mean that the robot has sparse and very limited-range information about the world. These limited sensing capabilities result in a restricted view of the world — features of the environment can only be detected by observing a time series of sensor readings.

Small robots with limited sensing capabilities are inexpensive, and are therefore ideal in applications where many potentially disposable robots are required to map and explore a building in parallel, e.g. in search and rescue or urban reconnaissance scenarios. In this paper however, we focus solely on the problems of exploration, mapping, and navigation for a single robot.

Our problem is for a sensing-limited robot to explore an unknown environment, creating a topological map that it can later use to navigate between

any two previously-visited locations. The resulting paths should be efficient to the extent that safety, limited-range sensing, and errors in sensing, robot motion and odometry allow.

In this paper, we demonstrate that such a robot can complete this task, though there are some limitations based on the size and geometric complexity of the world. We should not expect the same performance from a sensing-limited robot as we would from a robot with richer sensing modalities (e.g. a laser scanner). While we permit lesser performance in terms of time and scope, we still expect reasonable accuracy.

This paper presents a mapping strategy that constructs a topological map in which vertices are places where a robot behavior terminates, and edges represent a sequence of behaviors that move the robot from one place to another. We develop a new approach to the problem of “closing the loop” — recognizing that the robot has returned to a place in the world that it has already visited. Since sensing-limited robots cannot uniquely recognize a place based on instantaneous sensor readings, we create a hypothesis that the robot has closed the loop based on its geometric position estimate, and attempt to verify this hypothesis using an evidence-based approach that compares characteristics of subsequent edge traversals.

Our behavior-based map representation also allows us to introduce techniques that enable our maps to transcend the typical weakness of topological maps in representing open spaces. First, we introduce the concept of *portals*, which link open spaces (mapped by wall-following) to narrow corridors (mapped by hall-following). Second, we introduce the idea of *forays*, which cross open spaces to produce more direct links between vertices in the map.

1.1 Assumptions

We consider a robot that has a small number of short-range sensors. These sensors return the distance to the nearest obstacle along a straight line; they are “short-range” in that their maximum range is small compared to the dimensions of the environment. There must be enough sensors to robustly execute the behaviors described in Sec. 3.1. As an example, our robot has five infrared range sensors: one forward sensor and two sensors on each side.

There is known error in the robot’s movement, odometry, and sensing. We assume these errors are random and zero-mean, and that we can merge and compound measurements and compute confidence bounds for a given confidence level.

We assume an enclosed, static, rectilinear environment. Rectilinearity is a strong assumption — it removes uncertainty in the robot’s orientation — but it allows us to focus on the fundamental mapping problems. However, the environment may be “non-smooth:” there may be small discontinuities in the walls of the environment, such as rectilinear protrusions and recesses in a hallway due to door frames, structural columns, etc.

2 Related work

There are two traditional paradigms for robotic mapping: metric maps and topological maps. In metric maps, the geometry of the world is explicitly represented, either through exact or approximate representations. In topological maps [8], “places” in the world (typically hallway junctions) are represented by vertices in a graph, and paths between places are represented by edges. Metric maps provide a detailed world representation but require more storage and are sensitive to measurement errors. Topological maps offer a concise representation but (as traditionally implemented) cannot represent open spaces.

Many researchers have combined these approaches by using metric maps at nodes in topological maps. For example, Tomatis *et al.* [17] use a topological map to represent a network of hallways and use metric maps to represent rooms. In contrast, our work extends the topological mapping paradigm to represent open spaces. Our use of a wall-following behavior to generate an initial map of an open area is related to the idea of “coastal navigation” [13] which recognizes that areas near walls and obstacles have “high information content” due to the features they produce. Once the robot has built a map of the world boundaries, it can safely explore the interior.

A fundamental problem in topological mapping is recognizing an already-visited place (“closing the loop”). With sufficiently rich sensing, this problem is easily solved by recording a unique “sensing signature” [9] for each place. Without such sensing capabilities, the robot must continue exploring to deduce whether a loop has been closed. Kuipers’ “rehearsal procedure” [10] encapsulates the general idea of using the map topology to make this decision. Choset and Nagatani [3] describe an approach where structural characteristics of the map (e.g., the degree of vertices and the order of incident edges) are the primary criteria for verification. Tomatis *et al.* [17] embed this comparison in a POMDP that should show a single peak upon loop-closing.

Our approach to this problem is based on accumulating evidence to verify or reject loop-closing hypotheses. The evidence is based on the odometry error model and is combined using Dempster-Shafer theory [14]. This approach is related to that of Cox and Leonard [4], who maintain multiple hypotheses about the state of a dynamic world. The probability of each hypothesis is assigned and updated using a Bayesian framework. The main advantage of a Dempster-Shafer based method is the ability to represent “ignorance.”

Our work is related to the simultaneous localization and mapping (SLAM) problem. However, SLAM is typically solved using rich sensing (a scanning laser range-finder) and produces metric maps. See Thrun’s recent survey [16] for an overview of work on SLAM.

Others have addressed mapping and exploration with limited sensing. Butler *et al.* [2] describe coverage (equivalent to metric mapping) using robots with only contact sensors but with near-perfect odometry. Doty and Seed [5] have shown preliminary results in creating a “landmark map” using a robot with four short-range infrared sensors and one long-range SONAR sensor.

3 Basic mapping algorithm

The “basic map” created by our robots is a topological map created by circumnavigating open spaces with a wall-following behavior and traversing narrow corridors with a hall-following behavior. In this paper, we focus primarily on mapping of open spaces. First, we describe the required robot behaviors and then detail construction of the map.

3.1 Behaviors

In order to create the basic map, we require that our robot be capable of executing the following simple behaviors:

- The “turn” behavior causes the robot to rotate a specified angle.
- The “move-to-wall” behavior causes the robot to move forward until one of its sensors detects a wall. The robot then moves so that its “wall-following sensors” (side sensors) are aligned with this wall.
- The “wall-following” behavior (Fig. 1) causes the robot to move forward, maintaining a range r_0 to the wall with its wall-following sensors; any small discontinuities in the wall cause the robot to adjust and continue wall-following. The behavior terminates when the robot detects a *well-defined* corner. A well-defined *exterior* corner occurs when the wall “falls away” beyond a range of r_{\max} ; likewise, a well-defined *interior* corner occurs when the wall juts inward within r_{\min} . Any discontinuity that remains within the range $[r_{\min}, r_{\max}]$ of the robot’s wall-following sensors is an *ill-defined* corner (currently unused in our maps).

In the case of an exterior corner, wall-following terminates at a distance of r_0 past the discontinuity; for an interior corner, wall-following stops at a distance of r_0 *before* the discontinuity. Note that exterior corners will typically be detected by the wall-following sensors, whereas interior corners will be detected by front, side or bump sensors. The wall-following behavior must be capable of terminating in a reasonably repeatable location, regardless of the direction of travel. Because of these conditions, no two well-defined corners of the same type can be within a distance

$$r_* = \sqrt{r_0^2 + r_{\min}^2} \quad (1)$$

- The “hall-following” behavior is similar to the wall-following behavior, except that it is only used when the robot is traversing a “corridor” of width $\leq 2r_0$. In this case, both of the robot’s side sensors can “see” walls, and the robot moves through the center of the corridor.

Discussion The wall-following and hall-following behaviors are difficult to design because they are responsible for properly navigating through the environment. One requirement is that they must terminate in the same location,

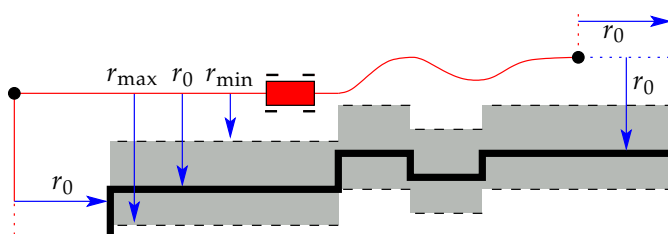


Fig. 1. Wall-following and feature detection. The “safe” wall-following range $[r_{\min}, r_{\max}]$ is indicated by the highlighted area. Vertices are placed at a distance of r_0 from both incident walls of a well-defined corner. The leftmost corner is exterior; the rightmost is interior. The rest of the discontinuities are ill-defined.

regardless of the direction of travel. This is done by maintaining the proper r_0 offset during wall-following and by staying centered while hall-following.

One way for the robot to make lateral adjustments while wall- or hall-following is to make only rectilinear motions, mimicking every discontinuity of the wall. This would be time consuming and unnecessary: our primary interest is in capturing the overall structure of the environment. Our current implementation instead makes gradual corrections while continuing forward. The tradeoff for this approach is that distance estimates may be inaccurate, and there are situations where small spacing between discontinuities may cause feature misidentification since the robot is not at offset r_0 and not parallel to the wall. Adjusting the responsiveness of the behaviors trades off the risk of feature misidentification for accuracy in the distance measurements.

The “perceptual aliasing” problem — the inability to detect small obstacles due to low sensor density — is another issue when creating maps with sensing-limited robots: the robot may encounter an obstacle without first sensing it. For example, the robot may bump into a chair leg that is not seen by its sparsely-placed range sensors. For such situations, bump sensors are necessary and can be thought of as extremely short-range sensors to be incorporated into the mapping behaviors accordingly.

3.2 Mapping process

We first consider an environment consisting only of open spaces. (We remove this assumption in Sec. 3.4.) Our basic map-making process is as follows:

- From an arbitrary starting location, the robot initiates the move-to-wall behavior, which aligns the robot with some wall in the environment.
- The wall-following behavior is activated, and its termination point becomes the start vertex v_0 — the first vertex in our topological map.
- After finding v_0 , the robot turns as appropriate to follow the next wall (incident to v_0). Vertices are added for each well-defined feature that

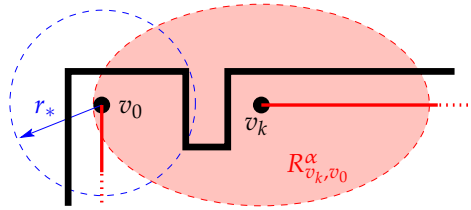


Fig. 2. A situation in which we hypothesize that $v_k = v_0$. The region R_{v_k, v_0}^α represents the positional uncertainty in the location of v_k with respect to v_0 . Since this region contains v_0 , it is possible that the two vertices are the same.

terminates the wall-following behavior. Each new vertex is connected, with an undirected edge, to the previously discovered feature (i.e. $v_0 \leftrightarrow v_1$, $v_1 \leftrightarrow v_2$, and so on).

- This process repeats until the robot returns to v_0 . This is guaranteed to happen, since the environment is enclosed. Recognizing when it occurs is the problem of “closing the loop.”

3.3 Closing the loop

For each vertex v_i in the map, we use the model of the robot’s odometry error to maintain a probability density function U_{v_i, v_0} describing the robot’s location with respect to v_0 . The uncertainty in this distribution grows monotonically as the map expands. For a confidence level α specified to our mapping algorithm, let R_{v_i, v_0}^α represent confidence bounds of U_{v_i, v_0} .

If the robot reaches a vertex v_k that is the same type (interior or exterior corner) and R_{v_k, v_0}^α contains v_0 , then we create a hypothesis that vertex v_k closes the loop. See Fig. 2 for an example of such a situation. Note that if the entire region R_{v_k, v_0}^α falls completely within r_* of v_0 (generally not the case), we can immediately declare the hypothesis correct with at least the specified confidence since no other vertex can be within that radius of v_0 .

Once a loop-closing hypothesis is made, we attempt to verify it. One way to do this is to *backtrack* — travel back and forth between v_k and v_0 , taking additional measurements of all the edges. Under the assumption of random zero-mean errors, this will reduce the size of R_{v_k, v_0}^α until it either lies entirely within r_* of v_0 or until v_0 does not lie in R_{v_k, v_0}^α . This is the only certain way to verify or reject a hypothesis, but it is impractical as many trips back and forth between v_k and v_0 are required.

Instead, we have the robot continue traversing edges in the “forward” direction. If it comes upon a structural difference (e.g. vertices of different types or incident edges at different orientations), the hypothesis can be rejected. In order to verify the hypothesis, however, we compare lengths of edges that should match under the hypothesis. The degree to which the measurements match (or don’t match) provides evidence in support of (or against)

the correctness of the hypothesis. The robot continues traversing edges until sufficient evidence accumulates to accept or reject the hypothesis.

Dempster-Shafer approach Dempster-Shafer theory gives us a natural way to assign “belief” to a set of possibilities — in this case, that the hypothesis is correct, which we denote as C , or incorrect, denoted I . It also provides a way to represent “ignorance” about these possibilities. Ignorance can be thought of as uncertainty about which possibility is supported by evidence, and is denoted as $\{C, I\}$.

Suppose that we have made a hypothesis H_0 that two vertices in our map are the same. Continuing our forward traversal of the walls of the environment, we take a series of new wall length measurements. Under H_0 , each revisited edge in the map has an associated set of measurements $L = \{\ell_1, \ell_2, \dots, \ell_n\}$, with $n \geq 2$ which should all be of similar length. Dempster-Shafer theory is used to merge the evidence provided by multiple sets of such measurements (each corresponding to a different edge in the map).

We compare edge measurements directly instead of comparing vertex locations because errors in each edge measurement are independent. Using vertex locations for evidence would reuse early edge measurements when comparing subsequent vertices, giving them undue weight in providing evidence.

Belief function Given a set L of measurements, and their associated uncertainty distributions, we must determine the degree of support they provide for our hypothesis. To do this, we define a *belief function*, denoted m_L , where $m_L(C)$, $m_L(I)$ and $m_L(\{C, I\})$ are our belief, given L , that H_0 is correct, our belief that H_0 is incorrect, and our “ignorance” (the degree to which L provides evidence neither for nor against H_0), respectively. These are basic probabilities meeting the requirements of Dempster-Shafer theory such that:

$$m_L(C) + m_L(I) + m_L(\{C, I\}) = 1 \quad (2)$$

Our particular belief function draws on methods of statistical inference. It essentially computes probabilities that the measurements in L were taken from the same wall, given the robot’s error model. The “probability information content” [15] of these probabilities is used to measure our degree of ignorance. Specifically, the belief function we use is:

$$m_L(C) = \Phi(L)\eta_L \quad (3)$$

$$m_L(I) = (1 - \Phi(L))\eta_L \quad (4)$$

Here, $\Phi: L \rightarrow [0, 1]$ returns the probability of obtaining L , assuming L contains measurements from the same wall. The particular Φ -function we use is a goodness-of-fit test based on the z -score of the measurements in L . (The z -score represents the deviation of a sample from the expected mean, expressed in units of the standard deviation of the underlying distribution.) We use the

square of the z -score, computed using a distribution with standard deviation $\hat{\sigma}$ generated according to the robot's error model for the current "best estimate" $\hat{\ell}$ (based on L) of the length of the wall:

$$z^2 = \sum_{\ell_i \in L} (\ell_i - \hat{\ell})^2 / \hat{\sigma}^2 \quad (5)$$

The value of z^2 follows a χ^2 distribution with $n = |L|$ degrees of freedom. So, the degree of support provided by L for H_0 is computed as:

$$\Phi(L) = \int_{z^2}^{\infty} \frac{y^{n/2-1} e^{-y/2}}{2^{n/2} \Gamma(n/2)} dy \quad (6)$$

The value $\eta_L \in [0, 1]$ in Eqns. 3 and 4 represents the "probability information content" [15] of $\Phi(L)$, computed as follows:

$$\eta_L = 1 + \Phi(L) \log_2(\Phi(L)) + (1 - \Phi(L)) \log_2(1 - \Phi(L)) \quad (7)$$

The probability information content of $\Phi(L)$ essentially measures the "distance" of $\Phi(L)$ from the uniform distribution. Thus, given a $\Phi(L)$ with high entropy (less information), we profess more ignorance, and for a $\Phi(L)$ with low entropy (more information), we profess less ignorance.

Evidence accumulation Upon first making a hypothesis, we must initialize the overall belief in the hypothesis. Recall that geometrically, no two features of the same type and orientation can be within a radius r_* (see Sec. 3.1). This suggests the metric

$$m(C) = \iint_{B_{v_0}^{r_*}} U_{v_0, v_h}(x, y) d\mathbf{A} \quad (8)$$

where v_0 is the start vertex, $B_{v_0}^{r_*}$ is the ball of radius r_* about v_0 , and $H_0 \equiv v_0 = v_h$. In other words, the initial value of $m(C)$ is the probability, according to our error model, that $|v_0 - v_h| \leq r_*$ (in which case they must represent the same feature). We initialize $m(I)$ to zero: at the moment we make H_0 , we have no basis for assigning belief mass to the possibility that H_0 is incorrect.

As we take new measurements, we must combine the evidence they provide with previous evidence we've acquired, in order to update our global belief in the correctness of the hypothesis. To combine evidence, we use Dempster's rule of combination. In our particular application, the combination rules are as follows: given global belief m and belief m_L attributed to a specific set of measurements L ,

$$m \oplus m_L(C) = \frac{m(C)m_L(C) + m(C)m_L(\{C, I\}) + m(\{C, I\})m_L(C)}{1 - m(C)m_L(I) - m(I)m_L(C)} \quad (9)$$

$$m \oplus m_L(I) = \frac{m(I)m_L(I) + m(I)m_L(\{C, I\}) + m(\{C, I\})m_L(I)}{1 - m(C)m_L(I) - m(I)m_L(C)} \quad (10)$$

We continue building evidence about a hypothesis until our overall belief in its correctness (or incorrectness) surpasses some threshold β . In general, we choose $\beta = \alpha$, the confidence bound used in generating the hypothesis. In [1], we provide additional discussion about making loop-closing decisions, and present extensions to the evidential loop-closing method for dealing with more complicated mapping scenarios.

Discussion The worlds in which it is most difficult to close the loop are those with structural self-similarity. Two ways in which a world might be self-similar occur when the world: (1) consists of walls that spiral inward or outward; or (2) consists entirely or partially of repeating similar sequences of walls. Our evidential approach fares well in “spiral” worlds (by their nature, these worlds yield comparisons between different-length edges). Worlds of “repeating similar sequences” are generally more difficult, though those consisting only partially of such sequences can often be properly mapped despite the presence of error, given a high enough evidence threshold.

Continuing the forward traversal of the environment after making a loop-closing hypothesis presents several issues. The key concern is that one can never be completely certain that a hypothesis is correct — a deficiency of nearly all hypothesis-based methods for closing loops. The backtracking method resolves this issue, but only after indefinitely many measurements. We believe that the idea of building evidence in support of and against the hypothesis is both intuitive and reasonable for use in practical situations.

A danger of the hypothesis-based approach is that incorrect hypotheses may be accepted as correct (or vice versa). One way to deal with this is to keep multiple hypotheses and never commit to any one of them (i.e. retain the hypothesis that the loop has not been closed). If several hypotheses seem correct, the robot can direct its exploration toward differentiating features. When a single hypothesis has emerged as correct, the map associated with it can be used for navigation, etc. If it is later deemed incorrect, additions to the map depending on that hypothesis can be discarded.

We have made a few other observations about the loop-closing problem:

- When the robot has actually returned to v_0 , all inconsistent hypotheses will have been disproved by structural mismatches.
- The robot’s orientation must pass through a full 360° (net) before it can return to the start vertex.
- False positive or false negative loop-closings can be disproved or proved later, but it is preferable to find the right hypothesis during basic mapping. As such, evidence thresholds should be reasonably high given no prior knowledge of the environment.
- Prior knowledge of some aspects of the world (e.g. feature density, distribution of potential wall lengths, etc.) might lead to more efficient loop-closing by allowing us to design more informed belief functions and letting us be more aggressive in accepting or rejecting hypotheses.

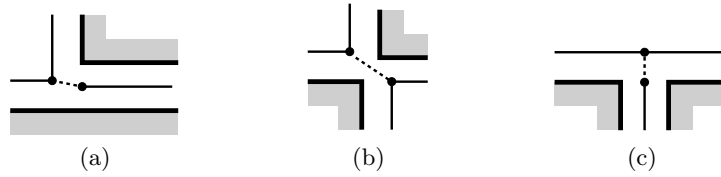


Fig. 3. Portals occur under special wall-following or hall-following terminating conditions. In each figure, the robot moves from left to right. In Fig. (a), the interior-corner portal is detected while following a wall. In Fig. (b), the portal, detected while turning an interior corner, leads directly to another open space. In Fig. (c), the portal is detected while turning an exterior corner.

3.4 Portals and narrow corridors

We have so far addressed only the case where open space lies to one side of the robot. When the area around the robot is narrow enough that the robot can see walls on both sides, the robot should switch from wall-following to hall-following. We define *portals* (Fig. 3) to be special undirected edges in the map that make the connection between these two different mapping modes.

In our basic mapping, we can use portals to skip over narrow corridors that may lead to other open areas, leaving them to be explored later. Often, this has the advantage of making the loops in the basic map smaller and therefore easier to close. Although doorways are prime candidates for portals, there is nothing special about them except that they constitute a narrow corridor.

Due to limited space, we omit a complete discussion of detecting portals and exploring narrow corridors. Though recognizing a portal condition is generally simple — side sensors on the non-wall-following side suddenly detect an object — a number of circumstances exist in which portals might occur, and each needs to be handled specifically. Our wall-following and hall-following behaviors use a “probing” strategy to find portals: upon encountering what appears to be a well-defined feature, they examine the area around the feature, attempting to determine if it constitutes a boundary between an open space and a narrow corridor. This method is also used when hall-following to differentiate between different types of hallway junctions.

For complete specifications of the wall-following and hall-following behaviors and discussion of portal detection, see [1].

4 Refinements

After we have completed the basic mapping process, navigation between any two points in the map is straightforward: we search the map (graph) for the shortest sequence of behaviors (based on time estimates) between the points and then execute this sequence. However, navigation through the basic map is inefficient in many cases: the robot may need to circumnavigate half of a building to reach a point across a wide hallway.

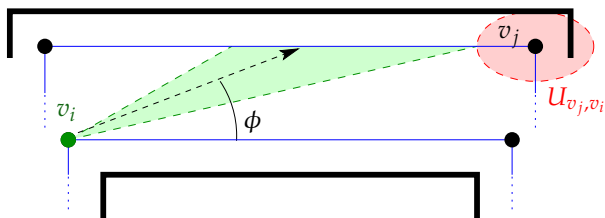


Fig. 4. A refinement from vertex v_i to vertex v_j . The robot turns to angle ϕ and “forays” toward the opposite wall. The robot’s angular uncertainty (highlighted) is small enough that it is confident it will land on the target wall.

The second phase of our algorithm introduces *refinements* into the map which improve its usefulness for navigating through open spaces (and for discovering “islands” in these spaces when they exist). Refinements to our basic map are special directed edges that connect disjoint places in the environment with paths that pass through open (featureless) regions. Refinements offer a tool for extending topological maps to such spaces in a natural way.

Our refinements are based on the geometry of the basic map. Because of measurement error during basic mapping, the map is not geometrically consistent. Prior to generating candidate refinements, we must embed the vertices of the map in a metric space. This problem has been addressed by several researchers, including Duckett *et al.* [6], Lu and Milios [12], and Golfarelli *et al.* [7]. Any of these methods suffice; the reference frame for the vertices can be placed arbitrarily.

4.1 Forays

Suppose we wish to add a path through free space to connect vertex v_i in our basic map with vertex v_j . If the robot experienced no movement error, we could do this by simply turning the robot towards v_j and driving straight. However, to allow for error in the basic map and in the robot’s motion, we direct the robot towards a wall adjacent to v_j . This *foray* across open space, then, consists of the sequence of behaviors: (1) turn toward the target wall (incident to the target vertex); (2) move-to-wall; (3) wall-follow to the target vertex.

We already have a means for estimating the relative uncertainty of vertex locations, using uncertainty distributions based on the robot’s error model. Confidence limits specify a region within which we believe (with the desired confidence) the vertex lies.

We assume that the uncertainty in the robot’s location estimate as it forays from v_i , due to the robot’s inability to drive in a straight line, can be bounded by an angular range originating at v_i . The robot’s positional uncertainty grows with the length of the foray.

There are two potential refinements from a vertex v_i to another vertex v_j , corresponding to the two walls incident to v_j . For each path, the robot forays in such a way that it lands on the desired wall. So, a valid refinement is possible only when the positional uncertainty accumulated by the robot as it forays is less than the total length of the target wall (between the confidence bounds in the locations of its endpoints). For an example foray, see Fig. 4.

We can make two types of refinements with this approach: *passive refinements*, which can be computed directly from the basic map, and *active refinements*, which require further exploration of the world.

Passive refinements Passive refinements use knowledge of the area swept out by the robot’s sensors as it performed its initial exploration. The union of all the area swept out by the robot’s sensors is the “known space.” A simple approximation can be obtained by storing the shortest range detected by the sensors opposite the wall-following sensors when traversing each wall.

Using the geometry of our basic map, we can immediately determine which candidate refinements pass only through the known space, and add these refinements to our map. Fig. 4 shows a simple example of such a refinement; though the robot could not see both sides of the hallway while building the basic map, the areas swept out by its non-wall-following sensors overlap, so we may infer the entire hallway is obstacle-free.

Active refinements & exploration Potential refinements passing partially or entirely through unknown space must be actively explored to verify that they can be traversed, since there may be obstacles in the unknown space.

Suppose that, when foraying, the robot encounters an obstacle. In this case the robot *must* be able to safely return to a known place in its basic map — otherwise, it is lost and will have to rebuild its basic map, or at least relocalize itself in the map. Prior to making forays, we discard those that are so long that we are unable to ensure the robot will be able to return to a known location in its original map. The identification of these “unsafe” forays is based on the robot’s odometry error model and the estimated distance between the origin and destination of each foray.

For the remaining exploration targets, we compute a sequence in which to explore them. One method for constructing this sequence might be to use a traveling-salesman like approach; unexpected obstacles might require the sequence to be re-planned. After planning a sequence, each foray is executed in turn.

If we encounter no obstacles until we reach the target wall, we keep the refinement; otherwise we discard it. If an obstacle is encountered, the robot begins the basic mapping process to map the obstacle. If its positional uncertainty accumulates too much before it encounters a well-defined feature, it gives up and returns to its original map. Otherwise, it creates a basic map of the obstacle, connecting the obstacle to its original map with refinements.

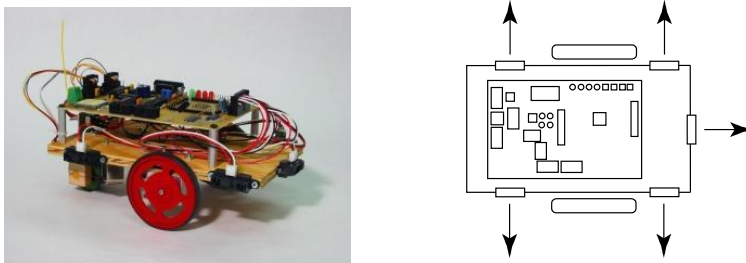


Fig. 5. Our prototype robot is a differential drive mobile robot approximately 20 cm long. It has five Sharp GP2D12 infrared range sensors, 256 CPR encoders on each wheel, and an Atmel ATMEGA64 microcontroller as the main processor.

4.2 Discussion

The refinements described here are reminiscent of the LMT [11] approach to compliant motion planning in which successive preimages of the goal are computed, taking into account uncertainty in the robot’s motion. Note that refinements serve to both explore unknown parts of the space as well as to create more efficient pathways through the world. The actions associated with a foray are a simple sequence of behaviors. More complex sequences could be planned that involve conditional branches, loops, or longer sequences of behaviors. For example, the robot could drive directly toward an interior corner vertex; it might land on either of the two supporting walls and must wall-follow in the appropriate direction to reach the goal vertex. In the extreme, we could form a completely connected graph, creating a “program” of behaviors to take the robot between any pair of vertices.

The computation required would be considerable and mostly unnecessary — most of these links would never be used. The refinements described earlier can serve as the building blocks for more sophisticated behavior sequences.

5 Experimental Results

We have implemented our mapping algorithm both in simulation and on a mobile robot (Fig. 5), with promising results. Simulated experiments were performed with a variety of simple worlds, and with several more complicated environments. Real-world experiments were performed in both specially-tailored and unmodified building environments.

Our simulations, which include the exploration of refinements, have been effective in mapping large worlds (Fig. 6), even under harsh error conditions. In over 200 trials in small worlds, with varying degrees of odometry error, 97% of the maps were structurally correct after initial loop-closing.

Our hardware experiments have focused primarily on verifying the evidential loop-closing method. Despite the robot’s sparse sensing and the often

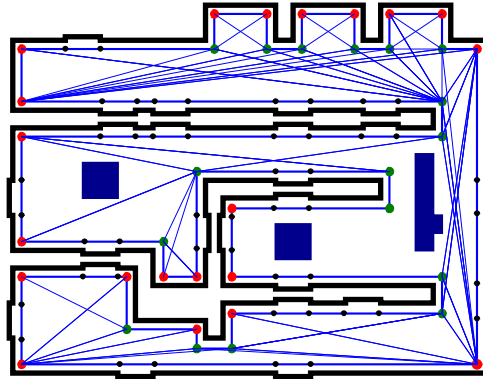


Fig. 6. Map created in simulation. Refinements to the basic map are included here; islands have not been mapped.

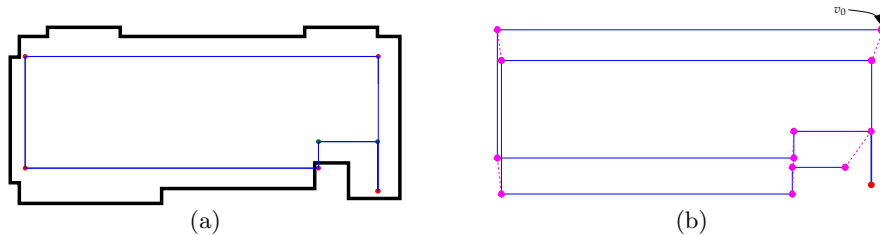


Fig. 7. Basic map created using the hardware from Fig. 5. In Fig. (a), the final map (after loop-closing and embedding) is overlaid on the floor plan. The environment is approximately $4.5 \text{ m} \times 2 \text{ m}$. In Fig. (b), the raw data for the map is displayed. Dotted lines are drawn between vertices that match under the correct hypothesis; the start vertex is the large dot at upper right. Here, the robot needed five measurements to accumulate 0.99 belief in the correct loop-closing hypothesis.

large errors it experienced, the algorithm closed loops correctly in most circumstances. Real-world maps were made of both small areas (Fig. 7) and large areas, including the $12 \text{ m} \times 30 \text{ m}$ first floor of an academic building.

Our mapping algorithm — especially the loop-closing method — relies on zero-mean error, particularly in the robot’s odometry. In reality, the error experienced by a robot is rarely zero-mean. For example, suppose the robot travels down a long hallway with a tile floor, and then returns down a heavily carpeted hallway. The difference in floor surface will yield biased (non-zero-mean) odometric error. So far in our experiments however, the negative effects of the zero-mean assumption have been minimal, even in large environments.

Overall, our simulated and real-world experiments indicate that the mapping algorithm closes loops correctly well over 95% of the time for non-self-similar worlds. For additional experiments and further discussion, see [1].

6 Conclusions and Future Work

In this paper, we have presented a topological mapping algorithm for robots capable of only sparse, limited-range sensing. We have introduced an evidential approach to the problem of “closing the loop” — recognizing when the robot has returned to a place it has already visited — based on the Dempster-Shafer theory of evidence. Finally, we have presented paradigms for incorporating open spaces into topological maps: “portals” that naturally connect narrow regions and open spaces and refinements that make “forays” through open spaces. Our experiments, both in simulation and with a real robot, verify the efficacy of our approach.

Our mapping approach is not without limitations. In particular, our robots are limited in the worlds they can map. These limitations are primarily dependent on the robot’s sensing deficiencies; for example, we should not expect a robot with large odometry error to be capable of efficiently making complete maps of vast open spaces. While creating precise maps of large spaces with a sensing-limited robot may be possible, our approach favors mapping and navigation efficiency over extensiveness.

Our approach is also limited by the capabilities of the behaviors it depends upon. Wall- and hall-following behaviors must be capable of terminating in repeatable locations and handling small spacing between discontinuities. While this is not strictly possible in a real-world implementation, our experiments have shown that reasonably capable behaviors can be developed.

6.1 Future work

Our approach to topological mapping should transfer directly to general polygonal worlds with a suitable wall-following behavior. We are also interested in more deeply exploring some of the fundamental questions brought forth by the topological mapping problem:

- How are the quality of the map and a robot’s mapping ability in general affected by sensing limitations? By the extent of sensor error? By the complexity of the world?
- What benefits are provided by specific assumptions about the world, such as rectilinearity or near-rectilinearity?
- How can we best use geometry to make inferences when loop-closing?
- What can we do in a curved world, with no “well-defined” features?

Acknowledgment

Thanks to Jonathan Fink and Alden Roberts for their help with the robot hardware. This work was supported by the NSF through award IIS-9983642.

References

1. K. R. Beevers. Topological mapping and map merging with sensing-limited robots. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, April 2004.
2. Z. J. Butler, A. A. Rizzi, and R. L. Hollis. Complete distributed coverage of rectilinear environments. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions (WAFR 2000)*, pages 51–61. A K Peters, 2001.
3. H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. *IEEE Transactions on Robotics & Automation*, 17(2), April 2001.
4. I. Cox and J. Leonard. Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, 66:311–344, 1994.
5. K. Doty and S. Seed. Autonomous agent map construction in unknown enclosed environments. In *Proc. MLC-COLT Workshop on Robot Learning*, pages 47–55, New Brunswick, N.J., July 1994.
6. T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proc. 2000 IEEE Intl. Conf. on Robotics & Automation*, volume 4, pages 3841–3846, 2000.
7. M. Golfarelli, D. Maio, and S. Rizzi. Elastic correction of dead-reckoning errors in map building. In *Proc. 1998 IEEE/RSJ Intl. Conf. on Intelligent Robots & Systems*, pages 905–911, B.C., Canada, 1998.
8. B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.
9. B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proc. 18th Natl. Conf. on Artificial Intelligence*, Edmonton, Canada, 2002.
10. B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.
11. T. Lozano-Pérez, M. Mason, and R. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984.
12. F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, October 1997.
13. N. Roy and S. Thrun. Coastal navigation with mobile robots. *Advances in Neural Processing Systems*, 12:1043–1049, 1999.
14. G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
15. J. Sudano. Pignistic probability transforms for mixes of low- and high-probability events. In *Proc. 4th International Conference on Information Fusion*, pages 23–27, Montreal, August 2001.
16. S. Thrun. Robotic mapping: A survey. Technical Report CMU-CS-02-111, Carnegie Mellon University, Pittsburgh, PA, February 2002.
17. N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: closing the loop with multi-hypothesis tracking. In *Proc. 2002 IEEE Intl. Conf. on Robotics & Automation*, May 2002.