Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization

Howie Choset, *Member, IEEE*, and Keiji Nagatani, *Member, IEEE*

***Abstract*—One of the critical components of mapping an unknown environment is the robot's ability to locate itself on a partially explored map. This becomes challenging when the robot experiences positioning error, does not have an external positioning device, nor the luxury of engineered landmarks placed in its free space. This paper presents a new method for simultaneous localization and mapping that exploits the topology of the robot's free space to localize the robot on a partially constructed map. The topology of the environment is encoded in a topological map; the particular topological map used in this paper is the generalized Voronoi graph (GVG), which also encodes some metric information about the robot's environment, as well. In this paper, we present the low-level control laws that generate the GVG edges and nodes, thereby allowing for exploration of an unknown space. With these prescribed control laws, the GVG (or other topological map) can be viewed as an arbitrator for a hybrid control system that determines when to invoke a particular low-level controller from a set of controllers all working toward the high-level capability of mobile robot exploration. The main contribution, however, is using the graph structure of the GVG, via a graph matching process, to localize the robot. Experimental results verify the described work.**

***Index Terms*—Exploration, localization, mapping, mobile robots, motion planning, tologoical maps, Voronoi diagrams.**

I. Introduction

SENSOR-BASED exploration enables a robot to explore an unknown environment and build a map of that environment. A critical component of this task is the robot's ability to ascertain its location in the partially explored map or to determine that it has entered new territory. Naively, one can determine the (x, y) coordinates of the robot using dead-reckoning, a process that determines the robot's location by integrating data from wheel encoders that count the number of wheel rotations (or fractional rotations). Dead-reckoning fails to accurately position the robot for many reasons, including wheel slippage. If the robot slips, the wheel rotation does not correspond to the robot's motion and thus encoder data, which reflect the state of the wheel rotation, does not reflect the robot's net motion, thereby causing positioning error. A global positioning systems (GPS) and inertial systems offer an alternative to dead-reckoning, but have their drawbacks as well. Finally, landmarks with known locations can be deployed in the environment, but the task described in this paper considers environments where their geometry is completely unknown *a priori*. We do assume, however, that the unknown environment is static, planar, and that our range sensors have sufficient range.

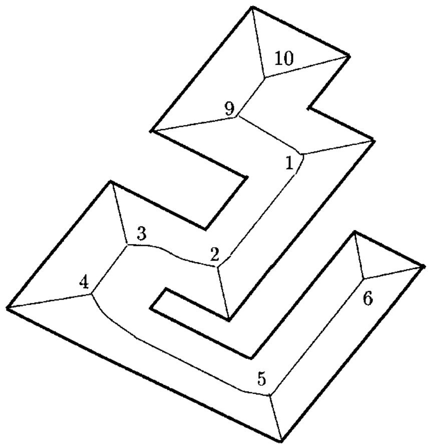

Fig. 1. The GVG where the symbols (nodes) are labeled 1–10.

All robots that do not use preplaced landmarks or GPS must employ a localization algorithm while mapping an unknown space, hence the term simultaneous localization and mapping, first coined by Leonard and Durrant-Whyte [16], [25]. This paper takes a topological approach to SLAM. It is our belief that the topological and geometric structure of free space induce a natural hierarchy of symbols and connections among these symbols that represent free space. At a high level, a topological map [14] serves as an example of symbols and connections between them. For Kuipers, the symbols are *distinct places*, which are local maxima of the distance to nearby obstacles, and the connections are the graph edges that link distinct places. For an indoor office-like environments, junctions and termination points of hallways represent symbols while the hallways themselves are the connections. For the generalized Voronoi graph (GVG) [21]; [7] (defined in Section II-A), the Voronoi vertices (we call them meet points) are the symbols while the edges form connections (Fig. 1).

Manuscript received January 4, 2000; revised October 23, 2000. This paper was recommended for publication by Associate Editor L. Kavraki and Editor A. De Luca upon evaluation of the reviewers' comments. This work was supported by T. McMullen at ONR under Grant 97PR06977 and H. Moraff, J. Xiao, L. Reeker, and E. P. Glinert at NSF under Grant IRI-9702768. This paper was presented in part at the International Conference on Robotics and Automation, Leuven, Belgium, 1998.

H. Choset is with the Department of Mechanical Engineering and Robotics, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: choset@cs.cmu.edu).

K. Nagatani was with the Department of Mechanical Engineering and Robotics, Carnegie Mellon University, Pittsburgh, PA 15213 USA. He is now with the Man–Machine System Laboratory, Department of System Engineering, Okayama University, Okayama 700-8530, Japan.

Publisher Item Identifier S 1042-296X(01)05570-7.

The connections of a high level representation can be discretized into a sequence of symbols with their corresponding set of connections. Leonard and Durrant-Whyte's approach to SLAM is an excellent example [16], [25]. Their method uses sensor information to define "features" and the relation among them to direct a robot experiencing positioning error. Using a Kalman filter approach to determine the "best" correlation of features, the robot navigates similarly to a sailor using stars to navigate a ship at night. Through the proper understanding of the robot's relationship to the features, the robot can maintain an accurate estimate of its position while moving through the environment using the features as low-level symbols that guide the robot from one high-level symbol to the next.

At a low level, the robot's environment can be modeled by a local map such as a fine array of pixels [20]; [12]. Thrun *et al.* [27] have been incredibly successful in demonstrating their probabilistic approach in museum environments using a grid-map representation. It is the belief of the authors that Thrun's main contribution is how to process a local map, whether it be topological or a grid, not the local map itself. Our contribution presented in this paper is to reduce the SLAM problem to a graph matching problem at the topological scale, not the graph matching itself.

Finally, defining the symbols and their connections is not enough. Stable well-defined control laws must ensure that the robot can identify (and converge onto if necessary) a symbol location while at the same time move from symbol to symbol, i.e., traverse an edge. Although the philosophy of this work is general to other topological maps, the map used in this paper is the GVG, which is a map embedded in robot's free space and captures the topologically salient features of the free space. With the GVG the robot can plan a path between any two points in a static environment by first planning a path onto the GVG, then along the GVG, and finally from the GVG to the goal. Thus, knowing the GVG is equivalent to knowing the free space and constructing the GVG is akin to exploring the free space.

The GVG can be defined in an arbitrarily dimensioned Euclidean space, but this paper stresses the planar version. We present in this paper well-defined control laws that generate GVG edges using line-of-sight range data. In deriving these control laws, we assume that the range and azimuth resolution of the sensors are adequate to capture the structure of the robot's environment. Finally, we are assuming that the obstacles in our environment are planar extrusions into three-dimensions, i.e., the obstacles are at the correct height for the sensors. With the control laws, the GVG is not only a representation of free space but it also forms a basis for exploring free space.

This paper presents some experimental results on how the robot can incrementally construct the GVG in an unknown space. Critical to this task is the robot's ability to locate itself on a partially constructed GVG. Although the robot locates itself (topologically) on the GVG, it never needs to determine its (x, y) coordinates (hence, the title of this paper). The robot can propagate the coordinates of each point on the GVG from the known location of one point, such as the start point, which can be specified to be $(0, 0)$.

In this paper, we first refer to prior work which has influenced the authors' thinking. Then, we define the GVG and prescribe the control laws to construct it. Incrementally constructing the GVG is not enough because of dead-reckoning error, as will be shown by example. We then introduce a the following three-tiered approach to localization: zero dimensional (using sensor signatures of the nodes); one dimensional (intentionally following a sequence of edges to a desired node); and two dimensional (handling the situation where the robot accidentally re-encounters an already-visited node). Finally, we discuss the tradeoffs of this approach and future work.

II. Relation to Prior Work

This work draws from two areas: sensor-based planning and localization. Although both of these fields are vast, we only discuss papers that have influenced our thinking.

A. Sensor-Based Planning and Roadmaps

Much of the previous work in sensor-based planning is heuristic and is limited to the plane. One class of heuristic algorithms employs a behavior-based approach in which the robot is armed with a simple set of behaviors (e.g., following a wall) [4]. Another heuristic approach involves discretizing a planar world into pixels of some resolution. Typically, this approach handles errors in sonar sensing readings quite well by assigning each pixel a value indicating the likelihood that it overlaps an obstacle [20]; [2]. Many heuristics exist for planning with the discretized map. Many behavior-based heuristics are sometimes termed as *reactive control* in that low level inputs directly affect the high level behavioral outcomes for the robot. Strong experimental results indicate the utility of these approaches, and thus these algorithms may provide a future basis for complete sensor-based planners. Unfortunately, these approaches currently neither afford proofs of correctness that guarantee a path can be found, nor offer well-established thresholds for when these heuristic algorithms fail. One of the goals of the work presented here is to demonstrate how low level inputs can direct a robot to construct a high-level representation: a topological map.

We seek to adapt the structure of a provably correct classical motion planning scheme to a sensor-based implementation. For example, Lumelsky's "bug" algorithm [19] proves that a robot using on-line information can plan a path to the goal. This method, however, does not yield a map of an unknown space. Our approach is based on a roadmap [5], a one-dimensional subset of a robot's free space, which captures all of its important topological properties. A roadmap has the following properties: *accessibility*; *connectivity*; and *departability*. These properties imply that the planner can construct a path between any two points in a connected component of the robot's free space by first finding a path onto the roadmap (accessibility), traversing the roadmap to the vicinity of the goal (connectivity), and then constructing a path from the roadmap to the goal (departability). If the robot can incrementally construct the roadmap, then it has in essence explored its free space because the robot can always use the roadmap to plan future excursions into the free space.

Already, the motion planning field has produced many roadmaps: silhouette methods [5], Voronoi diagrams [21], visibility graphs [15], etc. (see [15] for a survey of roadmaps). The roadmap used in this work is the GVG [7], which is the one-dimensional set of points equidistant to m obstacles in m dimensions. In the plane, the GVG is simply the *generalized Voronoi diagram*, which is the set of points equidistant to two obstacles [21]. In a three-dimensional Euclidean space $\mathbb{R}^3$, the GVG is the one-dimensional set of points equidistant to three obstacles.

Rao [22] achieves exploration by incrementally constructing the planar GVG in a two-dimensional static polygonal environment. Choset and Burdick [6] also describe a method to construct the GVG in a static m-dimensional environment, but their approach does not require obstacles to be polygonal, polyhedral, nor convex, which are assumptions most motion planners require. Both methods assume that the robot only uses line-of-sight information.

Both Rao's and Choset and Burdick's methods also assume that the robot is equipped with a 360° field-of-view high azimuth infinite range sensor that detects nearby obstacles. They do not consider how real range readings can be integrated into a control law to direct a mobile robot. In this paper, we present control laws that take low level sensor information and then directs the robot to follow the GVG edges without knowing the GVG ahead of time. This has the affect of incrementally constructing the GVG. Here, low level reactive-style control has a direct affect on high level behavior of the robot which *guarantees* that the robot explores an unknown space. This control law works well in small environments with a mobile robot equipped with a ring of sonar sensors [9]. Unfortunately the control laws, by themselves, are not enough for deploying a robot in large environments because of errors in encoder readings. The main contribution of this paper is to use the topology encoded in the GVG to overcome these errors and to localize the robot.

B. Localization

Localization is a major area of mobile robotics that has received increased attention over the past decade. Again, the literature in this field is vast, so only work which has influenced this paper's results are mentioned. See [3] for a complete overview of current localization techniques. Lu *et al.* [17], [18], [13] use gradient ascent to update various location estimates forward and backward in time. As a result, this approach has led to significantly larger maps that are more accurate than previous approaches, but is still limited to situations with bounded odometric error. Shatkay and Kaelbling [24] proposed an approach that uses probabilistic representations, along with the well-known Baum–Welch algorithm for efficient estimation. Their approach is similar in nature to the one described by Thrun [27], in that they both employ probabilistic representations and both use the Baum–Welch algorithm. However, the method in [24] does not consider orientation dead-reckoning error.

Thrun has recently completed a localization approach that has been successfully verified in very large environments on a mobile robot where a map is known *a priori* or the robot is driven (currently by an external agent) to acquire environmental information [27]. This approach poses a maximum-likelihood estimation problem, where both the location of landmarks and the robot's position have to be estimated. Likelihood is maximized under probabilistic constraints that arise from the physics of robot motion and perception. Just like [27], they use a Baum–Welch (or alpha–beta) algorithm. Unfortunately, this approach requires a user to specify the landmarks, as opposed to the robot self-selecting them. Also, their approach does not include an exploration strategy. In other words, there is nothing in their approach that directs the robot to explore new areas, or that guides the robot to specific locations to reduce dead-reckoning error.

C. Localization with Topological Maps

The above localization techniques took the approach of constantly trying to update the robot's (x, y) coordinates relative to a global frame. In this paper, we localize the robot on a topological map without ever having to do so explicitly. Others such as Dudek [11] and Kuipers [14] have reported localization results with the same philosophy. In [11], an agent locates itself on a graph by matching nodes and the adjacency relationships between them. This approach assumes that the agent can label each node by depositing a marker at the nodes. The approach in this paper and in [14] has the robot automatically identify nodes in the topological graph from geometric environmental information.

The basic thrust of this paper's work is quite similar to Kuiper's. In [14], the robot essentially traces points that are equidistant from two portions of the environment until it reaches a point that is three-way equidistant or until it reaches a point where a distance threshold is met, at which point the robot follows the obstacle boundaries. The nodes in this graph are termed *distinct places*, which are local maxima of the distance to nearby obstacles. The robot can easily self-determine distinct places from sensor data. Distinct places are a subset of the nodes of the GVG, which are the set of points equidistant to three obstacles (in other words, there exist examples of triple equidistance that are not local maxima). Localization is achieved again by matching distinct places of the graph.

D. Simultaneous Localization and Mapping (SLAM)

Leonard and Durrant-Whyte coined the term SLAM, which as its name suggests, enables a robot to construct a map of an unknown environment while using the same map to bound or nullify positioning errors [16]. With SLAM, the robot must automatically determine landmarks while constructing the map. Smith and Leonard [25] developed a feature-based approach to address SLAM by generating multiple hypothesis and selecting among them while mapping an unknown region. The work presented here builds upon Leonard and Durrant-Whyte's work by determining a rigorously well-posed method for identifying features and the topological relations among them.

Schultz *et al.* [23] present a more conventional approach to SLAM that uses certainty grids. This approach is more of an outgrowth of the algorithms presented in Section II-B. Although this work does not directly impact the SLAM algorithm presented in this paper, Schultz and our approaches share some common ideas and problems.

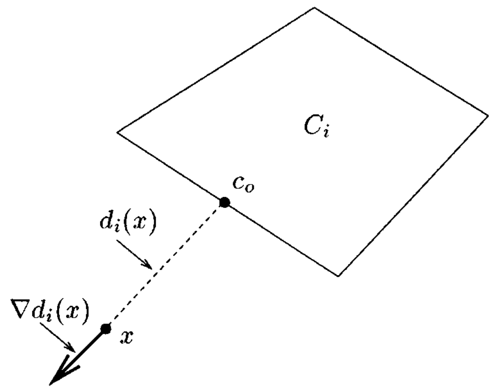

Fig. 2. Distance between x and C_i is the distance to the closest point on C_i. The gradient is a unit vector pointing away from the nearest point.

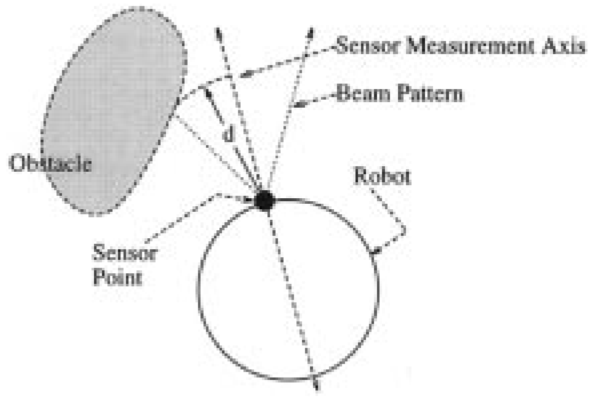

Fig. 3. Centerline model.

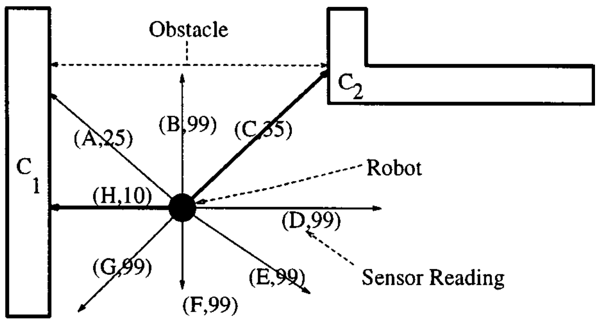

Fig. 4. A robot with eight sensors and their measurements is drawn. Sensor H has the smallest value, 10, and is thus pointing at the nearest obstacle. Sensor A has the second smallest value, but is not associated with the second closest obstacle because Sensor A is not a local minimum. Sensor C is associated with the second closest obstacle because its value is the second smallest local minimum in the sensor array. When two adjacent sensors have the same value and together form a local minimum, then assume the closest obstacle lies between the two sensors.

III. Sensor-Based Navigation and Map Building

In this section, we review the definition of the GVG graph structure and prescribe the control laws that construct its edges and nodes. In the next section (Section IV), we give an example of GVG exploration that highlights the problem of localization using dead reckoning, a problem that is independent of the GVG representation itself. The next section (Section V) then presents our contribution of topological localization.

A. A Topological Map: The GVG

The GVG is a one-dimensional set of curves that captures the salient topology of the robot's environment. Just as people use roadway systems, the planner uses the GVG to plan a path from a start to a goal by first planning a path from the start to the GVG, then along the GVG to the vicinity of the goal, and then from the GVG to the goal. Our approach to exploration involves constructing the GVG.

The GVG lends itself nicely to sensor-based implementation because it is defined in terms of a distance function d_i, which measures the distance to the closest point on object C_i, i.e., $d_i(x) = \min_{c \in C_i} \|x - c\|$ (Fig. 2). Distance can be determined from sonar sensors that are rigidly attached to the perimeter of mobile robots, pointing radially outward from the robot. Sonar is emitted from the sonar sensors, bounces off an obstacle, and returns to the robot. The time of flight is proportional to the distance between the object and the sensor. The object can lie anywhere along an arc (Fig. 3). In our implementation, for simplicity, we assume the echo originates from the center of the arc. Distance to an obstacle is then approximated by looking for local minima in the circular sonar array (Fig. 4). A laser range finder can be used to determine distance to the nearest obstacles in a similar fashion.

In the planar case, GVG edges are simply the set of points equidistant to two obstacles. More specifically, the planar GVG edge defined by obstacles C_i and C_j is the set

$$\begin{aligned}\mathcal{F}_{ij} = \{x \in \mathcal{F}s\colon\ & d_i(x) = d_j(x) \le d_h(x)\\ \text{such that}\quad & \nabla d_i(x) \ne \nabla d_j(x)\}.\end{aligned}$$

By definition, the end points or "nodes" of the GVG edges are *boundary points* $(d_i(x) = d_j(x) = 0)$ and *meet points* $[d_i(x) = d_j(x) = d_h(x)$ for at least one $h]$ [7]. In the Voronoi diagram literature [1], meet points are called Voronoi vertices, but we use the term meet points because GVG edges terminate (and meet) at them. The planar GVG is $\bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} \mathcal{F}_{ij}$. Fig. 5 has examples of planar GVGs.

Prior work [21], [7] has proven that the GVG can be used for path planning because the GVG is a *roadmap* of the robot's free space. A roadmap is a geometric structure that possesses the properties of accessibility, connectivity, and departability. In other words, there exists a path between two points q_s and q_g in the free space if and only if there exists a path from q_s to a point q_s' in the GVG (accessibility), a path from a point q_g' in the GVG to q_g (departability), and a path solely contained in the GVG between q_s' and q_g' (connectivity).

First, lets assume the GVG exists. Accessibility for an already constructed GVG is achieved by moving away from the closest object until a point on the GVG is reached. In other words, the planner uses gradient ascent to determine the accessibility path c. Assuming that C_i is the closest obstacle, the robot accesses the GVG following

$$\dot{c}(t) = \nabla d_i(c(t))$$

until it reaches q_s'.

The planner then uses a standard graph search algorithm to find a point q_g' in the GVG such that $\|q_g' - q_g\| \le d_h(q_g')$ for all h, which is a point in the GVG that is closer to the goal than nearby

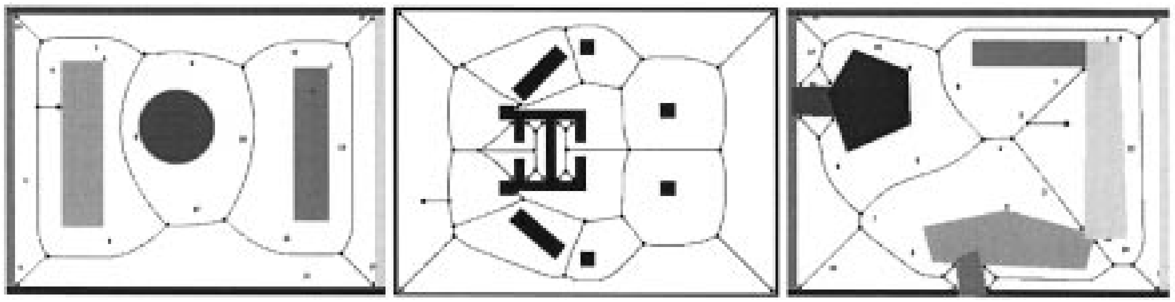

Fig. 5. The solid curve segments are the edges of GVG, the set of points equidistant to two obstacles.

obstacles. Connectivity of the GVG ensures that q_g' exists. Since q_g' is closer to the goal than all other objects, departability is achieved by following a straight line directly to the goal.

Effectively, if the robot knows the GVG, then it knows the environment because the planner can use the GVG to determine a path between any two points in a connected component of free space. Likewise, if the robot can construct the GVG using sensor data as it moves through the environment, then it has in essence explored the space because it could use the GVG to plan paths in the free space once exploration is complete.

B. Control Laws to Construct the GVG in an Unknown Space

One of the key features of the GVG is that a robot can incrementally construct it using only line-of-sight information. Exploration of free space via incremental construction of the GVG has the following five key components: 1) access the GVG; 2) explicitly "trace" the GVG edges; 3) determine the location of the meet points (GVG vertices); 4) explore the branches emanating from the meet points; and 5) determine when to terminate the tracing procedure.

The robot accesses the GVG in an unknown environment using the same procedure as in a known environment: the robot simply moves away from the nearest obstacle until it is equidistant to two obstacles. The distances from and directions to the closest obstacles can be computed using range sensors such as lasers scanners and ultrasonic sensors. For the moment, imagine rays emanating from the center of the robot and terminating when they encounter an obstacle. The length of the shortest ray corresponds to the distance to the closest obstacle and the direction of the ray is the direction to this obstacle. The robot simply moves in a direction opposite to that of the shortest ray (see Fig. 6).

Once the robot accesses the GVG, it must incrementally trace GVG edges. We derive a control law that effectively traces the roots of the expression

$$G(x) = \begin{bmatrix} d_1 - d_2 \\ d_1 - d_3 \\ \vdots \\ d_1 - d_m \end{bmatrix} (x) = 0$$

where d_i is the distance to an object C_i, and thus if $(d_1 - d_2)(x) = (d_1 - d_3)(x) = \cdots = (d_1 - d_m)(x) = 0$, the robot is equidistant to m obstacles.[1] In the planar case $G(x) = (d_1 - d_2)(x)$, which is zero when the robot is equidistant to two obstacles.

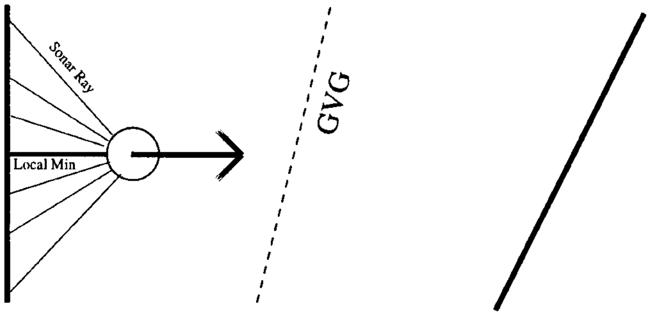

Fig. 6. The circular disk represents a mobile robot with some of its sonar sensor readings displayed as rays emanating from the mobile robot. The dark ray corresponds to the smallest sensor reading, hence the robot will move in the direction indicated by the dark arrow to access the GVG. The GVG is denoted by a dotted line between two nonparallel walls.

At a point x in the neighborhood of the interior of a GVG edge, the robot steps in the direction

$$\dot{x} = \alpha \text{Null}(\nabla G(x)) + \beta(\nabla G(x))^{\dagger} G(x) \qquad (1)$$

where

- α and β are scalar gains;
- Null$(\nabla G(x))$ is the null space of $\nabla G(x)$;
- $(\nabla G(x))^{\dagger}$ is the Penrose pseudo inverse of $\nabla G(x)$, i.e.,

$$(\nabla G(x))^{\dagger} = (\nabla G(x))^T (\nabla G(x)(\nabla G(x))^T)^{-1}.$$

Note that when x is on the GVG, $G(x) = 0$ and thus $\dot{x} = \alpha\text{Null}(\nabla G(x))$, which is simply the tangent direction of the GVG. The stability of this control law has been derived [9]. Stability requires that $\beta/\alpha > 1$; the α determines how quickly the robot moves along the GVG and the β represents how aggressively the robot moves back to the GVG.

Since G and ∇G are functions of distance and distance gradient, respectively, they can be easily computed from sensors, i.e., a mobile robot can poll its sonar array for local minima to determine d_i and ∇d_i (and hence G and ∇G). Therefore, using minimally processed raw sensor data, the robot can generate and traverse an edge from node to node of the GVG. Hence,

[1]Note that there are many equivalent choices of G, such as $G(x) = [d_1(x) - d_2(x),\ d_2(x) - d_3(x),\ \ldots,\ d_{m-1}(x) - d_m(x)]^T$, all of which trace the same GVG edge.

we have established a method by which a mobile robot using low level sensory information can determine the connections between topological symbols, which are the GVG meet points.

C. Control Laws to Determine GVG Meet Point Homing

The robot traces an edge until it detects a meet point or a boundary point. A meet point is, as its name suggests, a point where GVG edges meet. Meet points are one example of a topological symbol. At a meet point, the robot must determine the directions of the other GVG edges that emanate from it. In the planar case, the (at least) three nearest obstacles are equidistant to a meet point.

While generating the GVG, it is significant that the robot precisely locates itself on the meet points. Thus a meet point homing algorithm was introduced to trace a path that stably converges onto the meet point location [10]. The control law for homing onto a meet point is similar to the one for generating GVG edges, except G and its Jacobian are[2]

$$G(x) = \begin{bmatrix} d_1(x) - d_2(x) \\ d_1(x) - d_3(x) \end{bmatrix}$$

and

$$\nabla G(x) = \begin{bmatrix} (\nabla d_1(x) - \nabla d_2(x))^T \\ (\nabla d_1(x) - \nabla d_3(x))^T \end{bmatrix}.$$

Therefore, $G(x) = 0$ at a meet point, i.e., $d_1(x) = d_2(x) = d_3(x)$, and the robot makes the following correction step to home in on the meet point according to:

$$\dot{x} = \beta \begin{bmatrix} \nabla d_1(x) - \nabla d_2(x) \\ \nabla d_1(x) - \nabla d_3(x) \end{bmatrix}^{\dagger} \begin{bmatrix} d_1(x) - d_2(x) \\ d_1(x) - d_3(x) \end{bmatrix}$$

which can be shown to be stable using the previous analysis [9]. (Note that $\text{Null}(\nabla G(x)) = 0$.)

Geometrically, what is going on is that when the robot is in the vicinity of the meet point, it draws a circle through the three closest points on the three closest obstacles. It then determines the center of that circle and move a differential step toward the center. After taking this small step, the robot repeats this procedure. The stability of the resulting system allows us to conclude that the robot will converge to the location of the actual meet point.

D. Exploration

As mentioned above, the robot terminates edge tracing at a meet point or a boundary point. When the robot encounters a *new* meet point, it marks off the direction from where it came as explored, and then identifies all new GVG edges that emanate from it. From the meet point, the robot explores a new GVG edge until it detects either another meet point or a boundary point. In the case that it detects another *new* meet point, the above branching process is recursively repeated.

When there is a cycle in the environment, the robot will encounter a meet point that it already has discovered. It will have found an *old* meet point. In this case, the robot will search for the nearest meet point with unexplored emanating edges, from which it will continue the edge tracing process. Making this distinction between old and new meet points in large environments (in the presence of odometry error) is the ultimate challenge that we address in this paper.

Finally, when a robot reaches a boundary, it simply turns around and returns to a meet point with unexplored GVG edges. When all meet points have no unexplored edges associated with them, then exploration is complete. Exploring with the GVG is akin to simultaneously generating and exploring a graph that is embedded in the free space.

[2]Note that G could have many forms, including $[d_1(x) - d_2(x),\ d_2(x) - d_3(x)]^T$.

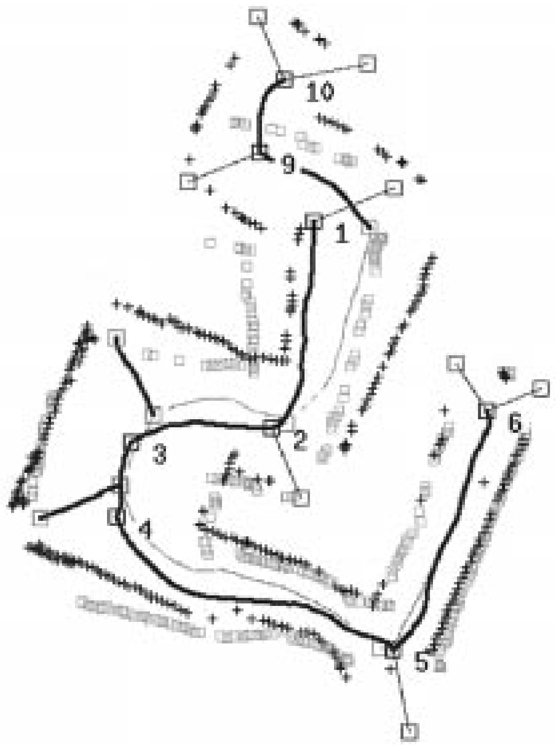

Fig. 7. The sensor data collected from a mobile robot after running through the environment depicted in Fig. 1. The walls of this environment have seemingly rotated as a result of dead-reckoning error. The robot starts at meet point 1, and then traces a path to meet point 6, at which point is backtracks to 1. When the robot returns to meet point 1, it thinks it is located at the light gray square, when it is actually located at the black square.

IV. Dead-Reckoning Error Problem

Critical to the above stated exploration procedure is the robot's ability to determine if it has encountered a new meet point or revisited an old one. When the robot reaches a meet point, a *naive* approach would compare the coordinates of the current meet point with the coordinates of all discovered meet points. If there is a match, then the robot can locate itself on the partially explored GVG. Otherwise, the robot can conclude it has reached a new meet point.

Unfortunately, dead reckoning error interferes with this decision, as depicted in Fig. 7, which contains data collected from running a Nomad 200 Mobile Robot in the environment depicted in Fig. 1. This experiment occurred in 7×14 square meter floor space covered with linoleum tiles. The Nomad 200 can translate and rotate in the plane. The robot also has 16 radially pointing outward sonar sensors to measure distance to nearby obstacles.

In this experiment, the robot starts near the square which denotes meet point 1. The robot heads toward meet point 2 tracing a GVG edge, which is denoted as a thick solid curve. The thick solid curves are the (x, y) coordinates of the GVG edge, based on encoder readings. The gray squares represent the sensor readings used to generate the GVG edge (Fig. 7). Note how they cluster together to form two parallel walls and the thick solid curve lies in the middle of them. These sensor readings are not stored by the robot, but are displayed here for visualization purposes.

The robot then continues from meet point 2, to 3, and then all the way to meet point 6. Now, the robot must back-track to a meet point with unexplored directions. The back-tracked (retraced) GVG is represented by a thin gray line and the plus marks denote the sensor returns from the back-tracking procedure. Again, the thin gray line represents the coordinates of the GVG, based on encoder readings.

Note that the thick solid and the thin gray curves do not line up. This is a result of dead-reckoning error. So, from the robot's view of the world, through its encoders, the environment is starting to rotate clockwise. When the robot returns to meet point 1, the robot cannot conclude from its encoder readings that it is truly at meet point 1.

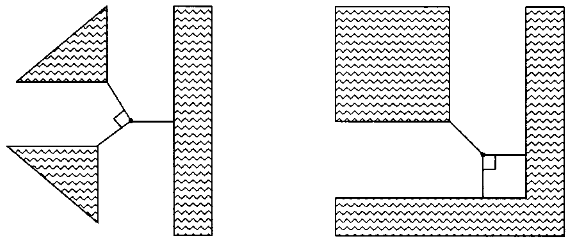
Fig. 8. The relative location of the three smallest local minima (vectors to the closest points on the closest obstacles) are similar for strikingly disparate meet points.

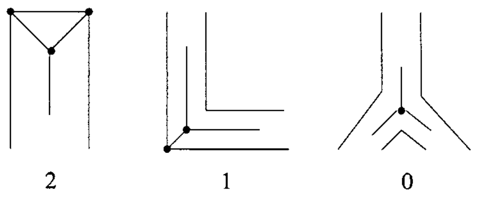

Fig. 9. Varying neighboring boundary nodes.

V. Topological Localization

With the control laws to generate edges and nodes of a topological graph in hand, we can now present a three-tiered system to topological localization: zero dimensional, one dimensional, and two dimensional. The zero-dimensional method assumes that all meet points have a unique signature. Simply returning to a meet point immediately localizes the robot. Unfortunately, this method is naive because many meet points may look the same to the robot, especially in man-made environments like office buildings. Therefore, we can use the "history" of the robot, i.e., we recall the (one-dimensional) sequence of edges and nodes that the robot traversed in order to determine the robot's position. This method works quite well when the robot intentionally visits an already explored meet point. When the robot unintentionally encounters an already visited meet point, it must then rely on the topology (two dimensions) of the graph structure to determine that the robot is visiting old territory.

In this section, we introduce some methods by which a robot may detect a meet point and then show why we do not want to rely upon these methods. In other words, we show why we prefer not to use a purely zero-dimensional approach. Next, we show how a robot can use its knowledge of a partially explored map to follow a sequence of edges to return to an already encountered meet point. Finally, we describe the full two-dimensional case which is invoked when the robot accidentally encounters an already visited meeting.

A. Meet Point Identification

The meet points, or any other topological node, serve as landmarks that the robot determines on-line. Here, we describe some robust, albeit naive, methods for identifying meet points. If each node in a topological graph "looked" different to the robot, then we could stop here. Simply invoking the control law to return to a meet point would be enough to determine the robot's position. However, this is not going to be the case, because many environments have meet points that look the same. Take, for example, a long corridor with lots of T-intersections. All of the intersections look the same.

Initially, we tried to derive a "sensor signature" for each meet point based on the robot's 16 sonar sensor readings, but this proved to be ineffective. Using all 16 sensors was not useful because many of the readings were inaccurate due to specularities and false echoes. Then, we considered the three smallest local minima of the circular sonar array. This was not useful because local minima with "similar" signatures correspond to meet points of significantly different geometry (Fig. 8).

Instead of using a complicated sensor signature to identify a meet point, we look for a *stable feature*, one which will not change in the presence of sensor noise and slight changes in robot location. A stable feature can be viewed as a landmark that the robot determines reliably on-the-fly. The first distinguishing and stable feature is the distance to the closest obstacle(s) at the meet point; distance is a stable feature because the homing algorithm described in the previous section has been proven to be stable. Obviously, this distance measurement will not distinguish different meet points very well, but it can be used to quickly eliminate any candidate meet points.

We can also exploit the topology of the GVG to reliably disambiguate meet points by looking at the neighboring nodes of a particular meet point. For example, a meet point with one neighboring boundary node is significantly different from a meet point that has no boundary nodes. This criterion readily distinguishes between the two meet points in Fig. 8, where the sensor signature was virtually useless. For a triply equidistant meet point, the varying combinations of meet points with neighboring boundary points is depicted in Fig. 9.

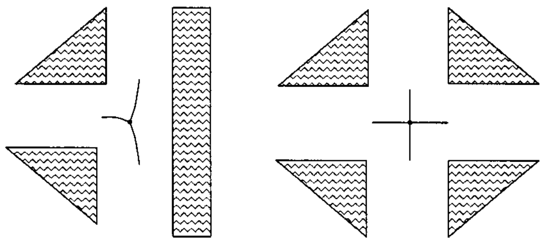
Fig. 10. Three edges is different from four edges.

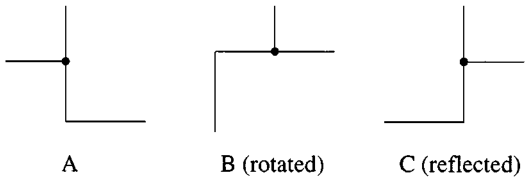

Fig. 11. Departure angle criterion does not distinguish between (A) and (B), but does discriminate between (A) and (C).

Another stable criterion looks at the number of edges emanating from a meet point. Again, while using the homing algorithm, sensor noise and position uncertainty will not affect the number of edges emanating from a meet point (Fig. 10).

The final criterion matches the relative departure angles of GVG edges emanating from the meet points. Meet point (A) in Fig. 11 has the same ordered set of departure angles as meet point (B), whereas meet point (C) may have the same angles as meet point (A), but the ordering is different. Encoder error cannot affect the ordering of these angles, therefore meet point (A) is *definitely* different from meet point (C).

By combining the distance and neighboring nodes criteria, the robot now has a significantly better chance for determining at which meet point it is located, or for concluding that it is at a new meet point. Unfortunately in real environments, these criteria will produce a candidate set of meet points at which the robot is located. The robot must further exploit the topology of the constructed roadmap.

B. Intentionally Revisiting a Meet Point

Our goal is to minimally rely upon sensor signatures to determine the robots position. Instead, we can exploit the topology (adjacency relationships) of the GVG to determine the robot's position and reliably direct it to a destination. The control laws guarantee that the robot can follow an edge and home onto a meet point. When the robot uses its partially explored map to intentionally return to a meet point, all it needs to do is follow a path, i.e., a sequence of edges and nodes, in the GVG by invoking a sequence of edge following and meet point homing control laws. The planner uses a one-dimensional history.

This method is best explained by example. In Fig. 7, the robot started at meet point 1 and worked its way to meet point 6. Now, the robot must retraverse the GVG back to meet point 1 to explore the unvisited edge emanating from meet point 1. The robot first *intentionally* returns to meet point 5. In order to return to meet point 5, the robot simply invokes an edge tracing control law until it detects a meet point and then it invokes the meet point homing routine to converge onto a precise location of the meet point. In Fig. 7, recall that the gray box is the robot's perceived location based on encoder coordinates, but in actuality the robot was located at the solid box.

The robot repeated this procedure four more times until it reached meet point 1. Even though the encoders indicated that the robot was located at the gray square, in actuality, it was located at the black square. The robot knew its accurate position because it intentionally sought meet point 1 and its meet point identifier confirmed that it had reached meet point 1. Therefore, the robot knew exactly where it was in the GVG, without ever relying on global encoder information.

This approach may not work well when meet points are "close" to each other. Sensor error can cause the robot to temporarily escape the current meet point's basin of attraction and fall into the "next" meet point's basin of attraction. When this happens, the robot homes onto the "next" meet point in its sequence. To address this problem, when the robot homes onto a meet point, it can use the sensor signature, described above, to confirm that it has arrived at the correct meet point. If the robot accidentally arrives at the wrong meet point, it can then use its partially explored map to determine if there was another a meet point "closer" to the desired meet point. If so, the robot can then conclude where it is located on the graph and then continue along in the sequence of edge following and node homing procedures.

C. Accidentally Revisiting a Meet Point

In the previous section, the robot intentionally directed itself to a meet point it expected to encounter by undergoing a sequence of edge following and meet point homing control laws. In many situations, the robot can unexpectedly find an already visited meet point. The challenge is to determine if the "next" meet point that the robot encounters is a new point or an already visited one. The robot must distinguish between new and old meet points in order to successfully explore an unknown environment. The robot does this by combining the zero-dimensional and the one-dimensional localization methods of the previous two subsections to form a truly two-dimensional localization procedure.

In this procedure, we assume that effect of orientation errors are more pronounced than translational errors, i.e., orientation dead-reckoning error dominates translational error accumulation. This is a reasonable assumption, at least with the Nomad 200 from Nomadic Technologies and can be readily seen in Fig. 7 in Section IV. In Fig. 7, the environment has seemingly rotated because of dead-reckoning error, but note how the lengths between meet points remains fixed. With this assumption in place, we can now determine the lengths of all GVG edges using encoder data. Note that we are only using "recent" encoder data when determining edge lengths.

When the robot encounters a meet point, it enumerates a set of candidate meet points that it could have reached, based on the criteria of the previous section. The robot then retraces an already explored edge emanating from the current meet point

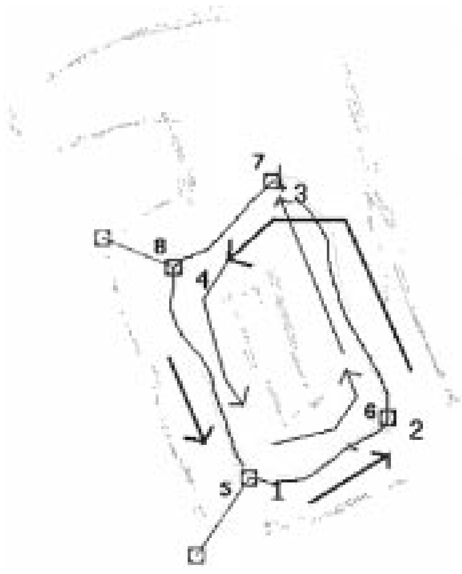

Fig. 12. The robot has partially explored its free space. The GVG for this environment has a cycle encircling the obstacle in the middle of the room. The GVG has driven around this cycle twice noting that meet point 1 and 5, 2 and 6, and 3 and 7, respectively, all look the same.

to an adjacent meet point. Again, a set of candidate meet points corresponding to the second meet point is enumerated. If the distance between a meet point in the second set to *any* meet point in the first set is not the same as the distance the robot traveled from the first meet point to the second, then the appropriate meet point is eliminated from the second set. Therefore, the second set of points only contains meet points which *could* be adjacent to at least one meet point in the first set. That is, we have in essence identified a set of candidate edges that the robot just has traversed. This procedure is repeated until only one possible meet point remains. Essentially, we have described a graph matching procedure where the robot partially reconstructs a fragment of the GVG and matches it with the already constructed GVG. This is why we call this a full two-dimensional localization approach.

It is worth noting that the robot does not store the GVG edges because it retraces them during the backtracking operation. Therefore, the robot only stores the meet points, their adjacency relationships (as edges), and the lengths of each edge. The robot also stores the departure angle of the GVG edges, i.e., the angle the GVG edge makes with the other GVG edges emanating from the meet point.

VI. Experimental Result

As can be seen in Fig. 7, the robot can intentionally return to an already discovered meet point without much trouble by following the already explored GVG. It is worth noting that the robot does this by *not* passing through a sequence of way-points specified by (x, y) coordinates along the GVG. Instead, the robot determines a path in the GVG (a path in the graph, not a path in the robot's free space), and then determines the appropriate sequence of edge following and node homing control laws from this graph-path. Essentially, the control laws prescribe a set of sensory experiences that the robot must undertake to arrive at the meet point, or pass through a sequence of meet points to arrive at the target meet point.

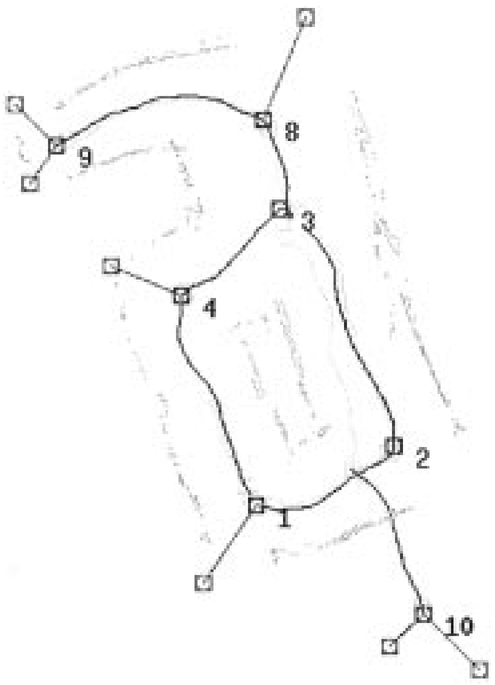

Fig. 13. Topological matching in a 750 ft^2 environment. Dark lines correspond to the first pass and shaded lines delineate retraces.

The challenge occurs when the robot unintentionally visits an already discovered meet point. Figs. 12 and 13 demonstrate an experiment in a real environment using a Nomad 200 mobile base where localization was performed using criteria of the previous sections. In Fig. 12, the robot starts at meet point 1, then travels to 2, 3, and 4. From meet point 4, the robot heads toward meet point 1, but when it encounters meet point 1 it temporarily labels it meet point 5, but based on the sensor signature described above, the robot notes that it could be meet point 1. The robot then moves to meet point 2 and labels it meet point 6. Since meet point 6 "looks like" meet point 2, and the distance between meet points 5 and 6 is the same as meet points 1 and 2, meet point 2 is a candidate for meet point 6. Now, the robot moves to meet point 3, temporarily labels it meet point 7 and makes the appropriate matches. At this point, the robot concludes that 5 is 1, 6 is 2, and 7 is 3. See Fig. 13.

Now, the robot explores meet points 8 and 9 and then backtracks to meet points 3 and 2, in order (Fig. 13). When the robot re-encounters meet point 2, it is fairly apparent that encoder error has significantly accrued. Note the robot thinks it is located on the light gray box, when in actuality, it is located on the dark gray box labeled meet point 2. Although the encoders deceive the robot into thinking it is a foot away from its actual location, by matching GVG edges and nodes, the robot can conclude it is at meet point 2. From there, the robot explores meet point 10. The final edge is drawn emanating from the shaded box to emphasize that dead-reckoning error has accumulated, but the robot knows meet point 2 anchors this edge. So, the robot has computed the entire GVG without ever resorting to dead-reckoning, nor having to update its encoders.

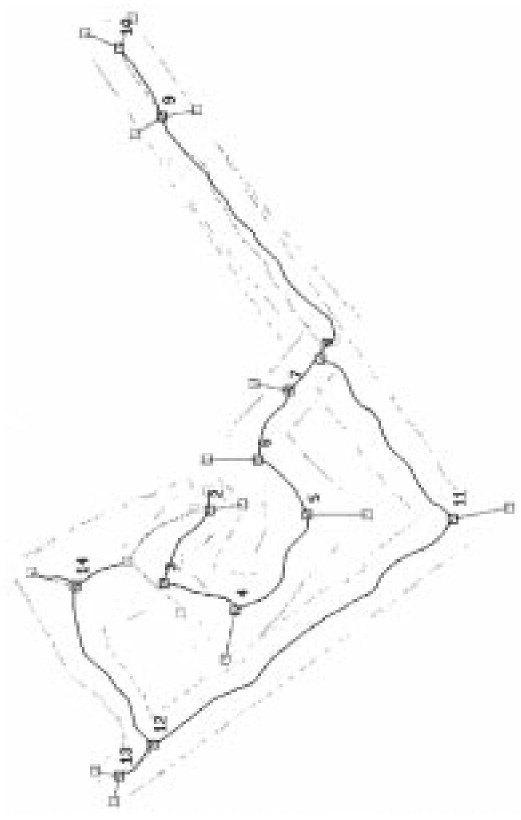

Fig. 14. Topological matching in a 1800 ft^2 environment.

The environment in Fig. 14 is a superset of the one in Fig. 13. In Fig. 13, meet points 9 and 10 are near closed doors of the Sensor-Based Planning (SBP) Laboratory at Carnegie Mellon University (CMU), Pittsburgh, PA. For the experiment depicted in Fig. 14, we opened the doors and let the robot explore the hall ways surround the SBP Lab at CMU. Note there is no correspondence between the meet point labels in Figs. 13 and 14.

In the Fig. 14 experiment, the robot starts near meet point 1, explores a boundary edge terminating at boundary point 2, and then goes on to meet points 3, 4, 5, 6, 7, and 8. After meet point 8, the robot leaves the lab, makes a left turn and explores corridor until encountering meet point 10. The robot back tracks to meet point 8 to continue exploring the other side of the corridor. Note the light gray lines indicate the amount accrued dead-reckoning error. The robot re-enters the lab at meet point 12, visits meet point 14, and then finally returns to meet point 3. However, the robot cannot conclude that it has returned to meet point 3, so it temporarily labels it as meet point 15. The robot then travels to meet point 2, labeling it 16 and then to meet point 4, labeling it 17. At this point, the robot has enough edges and nodes to match 2 to 16, 3 to 15, and 4 to 17, a which point the graph is matched. Here, the robot has accrued over 10 ft in dead-reckoning error.

VII. Discussion and Relation to Future Work

A. Richness of Topological Information

The work presented in this paper is only the first step toward the long-term goal of localization. Our immediate problem deals with environments with repeated symmetries. One could increase the number of edges and nodes to be matched, but there will always be an environment which will require one more matching. Furthermore, there are environments that are symmetric and thus this procedure theoretically should not be able determine the robot's location in the GVG. Consider the free space in Fig. 15. From the robot's perspective, meet points 1, 2, 3, and 4 all look the same, using any sensor signature. Meet point 1's neighbors, meet points 2 and 4, look like meet point 2's neighbors, meet points 1 and 3, etc. Therefore, simply moving to a neighboring meet point to determine at which meet point the robot is currently located will not work. Likewise, moving to a neighbor's neighbor will not work either.

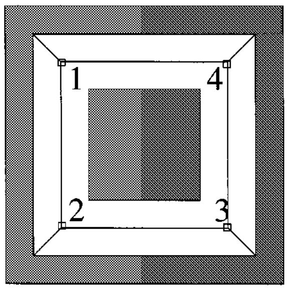

Fig. 15. A hypersymmetric environment in which all meet points look the same, their neighbors look the same, their neighbors' neighbors look the same, etc.

We believe that this problem is not a fault with the GVG method, but rather, results from the lack of richness of information that topology provides. Imagine being in a building whose hallways all look the same, as they do in Fig. 15. It is very easy to get lost because the hallways and their adjacency relationships do not determine any specific location. To achieve localization, a nontopological feature, such as the numbers on the office doors, would have to be used. In mobile robotics, this means that the robot would have to rely on a feature-based approach [16], [25] to localize itself. If all of the features, themselves, also "look" the same, then the robot would have to rely on a model of how dead-reckoning error accrues and use a lower level representation [27]. Incorporating this hierarchical approach to SLAM is a current topic of research.

Topological-based approaches do not work well when the robot is in the middle of large open spaces with obstacles beyond the range of the robot's sensors. Again, the GVG is not a good choice for a map here, not because of a fault with the GVG, but rather because of a lack of richness of topological information in large open spaces. It is hard to locate oneself in the middle of the desert. Current work in robotic coverage [8] addresses this problem; here the robot must plan a path to pass the sensor-range of the robot over all points of the free space.

B. Weak Meet Points

Another problem we have encountered is the emergence of sporadic meet points from "weak" features in the environment. Sometimes the robot "sees" a third obstacle and sometimes it does not. For the GVG, the problematic landmarks are called *weak meet points*. GVGs that have weak meet points lie on the boundary of GVG equivalence classes: a set of GVGs that can be continuously deformed into one another. GVGs in the *interior* of each class are stable because a small perturbation in the arrangement of obstacles causes a small continuous change in the GVG

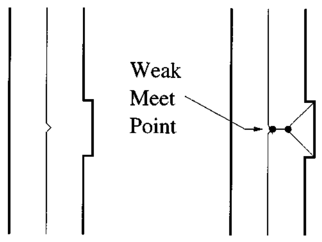

Fig. 16. Weak meet point.

that preserves the neighboring relationships of all meet points in the GVG. The GVGs on the boundary are unstable because a small perturbation in the arrangement of objects can cause a drastic change in the GVG.

Slight changes in sensor readings can have the same effect as small changes in the arrangement of obstacles. In other words, small changes in sensor readings can change GVG classes which destroys the neighboring relationships of all meet points. Consider a portion of the free space in Fig. 16. In one pass, the robot starts from the top and drives down the corridor; the range sensors on the robot give the robot the impression that the indent on the right is not deep enough to warrant a meet point. In a second pass, the robot moves from the top and drives down the corridor; this time, a slight change in the range sensors gives the robot the impression that the indent on the right is deep enough to warrant a meet point. This meet point is a weak meet point. Although the change in the GVG is not topologically significant, performing topological matching on a varying GVG becomes quite difficult because the meet points are our landmarks.

This work addresses this problem by defining a new structure that has significantly fewer meet points called the *reduced GVG*. Essentially, this new structure is the GVG with many of the weak meet points removed from it. We observed that many weak meet points, such as those in Fig. 16, have a boundary node adjacent to it. So, we simply deleted all meet points that have boundary nodes adjacent to them. This naturally deletes additional nonweak meet points. With the problem meet points removed, localization becomes more reliable. The drawback is that we have fewer meet points to use for matching, as can be seen in Fig. 17

Unfortunately, the reduced GVG does not entirely solve the weak meet point problem. As mentioned before, another problem lies in points that are close to each other. In one pass, the robot may perceive them as separate meet points, but in another, it may merge them into one meet point. Again, the map will have to be updated to reflect the robot's perception of the world. Future work will incorporate the probabilistic method of Thrun *et al.* [27] to allow for meet points that appear and disappear. This will help us implement our approach in dynamic environments.

C. Dynamic Environments

Future work will consider dynamic environments after the GVG is constructed. In this scenario, the robot goes to a location by invoking an interleaved sequence of edge tracing and meet point homing control laws, identical to the ones used in back-tracking to an already visited meet point. While tracing an edge to an already visited or *expected* meet point, if a transient

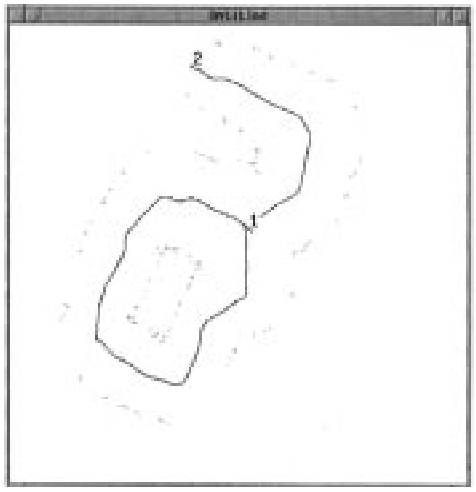

Fig. 17. Reduced GVG for similar environment as depicted in Fig. 13.

obstacle enters the robot's environment, this will cause an *unscheduled*[3] meet point to appear.

The challenge for the robot is to disambiguate between expected and unscheduled meet points. If this meet point "looks" different from the expected meet point, then the robot can conclude the meet point was unscheduled. However, if the unscheduled and expected meet points look the same, the robot will have a false understanding of its location. To handle this situation, we will update the one-dimensional localization procedure of intentionally revisiting meet points to use more geometric information as we did in Section V-C. We will use encoder data to infer an approximate length of the GVG edges. Just as in Section V-C, we will assume that linear dead-reckoning error is negligible, so the robot can accurately, within an error box, measure its distance along an edge. With this information, we can, to within a tolerance, disambiguate between unscheduled and expected meet points. When the robot does indeed find an unscheduled meet point, it can find a new path using an optimal graph strategy like D^* [26] or it can generate temporary edges until it re-accesses the original GVG.

VIII. Conclusion

This paper formulates a new approach to SLAM of unknown regions using a topological map annotated with some geometric features. We term this procedure T-SLAM. The specific map used in this work is the GVG, the locus of points equidistant to two obstacles in the plane, but this work is general to other topological maps. Already, prior work has rigorously shown that the GVG is sufficient for motion planning. Therefore, constructing the GVG is akin to provably complete exploration because once the robot knows the GVG it can plan a path between any two locations.

The main contribution of this work is using topology to localize a robot while mapping an unknown space. We use the nodes of the GVG as "landmarks." These landmarks are topologically meaningful events that the robot can determine on-line. Prior methods either use an *ad hoc* approach or human inputs to determine important features in the environment,

[3]We borrow this term from the land mine community that defines an unscheduled land mine to be one that was accidentally discovered in a region thought to be free of land mines.

whereas the GVG (or any geometrically annotated topological map) already has the features, i.e., the meet points, that the robot can use for localization.If each topological feature had a unique sensor signature, then a simple zero-dimensional approach of identifying the current meet point would be sufficient for localization. Unfortunately, with any method, several features may "look" the same, so this approach also considers the topological relationship among features while performing localization. In fact, our algorithm places more emphasis on how features relate to one another than on the actual features themselves, enabling us to store a concise data record of each individual feature and bypass the need to make detailed data comparisons among features.

The robot achieves SLAM by constructing a graph and comparing the "recently" constructed graph to subgraphs of the already constructed map. This map-and-compare method will cause the robot to "re-explore" subregions of the target environment. It is the conjecture of the authors that this re-exploration step is unavoidable and a feature: if the robot has multiple hypotheses of its true location, the algorithm can specifically direct the robot where to go to disambiguate among the possible locations of the robot in the partially constructed graph. This is a true two-dimensional matching process. The only assumption that we are making is that orientation positioning error dominates the linear.

We initially implemented this approach on a mobile robot with 16 sonar sensors and identified a problem with unstable features found in some environments. The intrinsic benefit that the algorithm uses the environment to define its landmarks has a problem: for some environments, a slight change can result in a different graph representation of the environment. This problem manifests itself when errors in sonar sensors cause the robot to oscillate between different environment representations.

Note that this method only works well when the robot's free space is rich with topological information, i.e., when there are lots of obstacles. If the robot is in the middle of a large open space where obstacles are beyond the sensor range of the robot, there is no topological information available to the robot and this procedure should not be invoked. Also, in hypersymmetric environments where every meet point looks the same, all of the meet points' neighbors (and neighbors' neighbors and so on) look the same, topological information will not help. The authors do not believe that method described here, by itself, is a magic solution to SLAM, but a high level component of it that rests on prior SLAM work.

Finally, the map used in this paper extends nicely into three dimensions. Future work will apply the methods described in this paper to three-dimensional localization. The target application is Aercam, a free-flying robot that will be used to inspect the future space station. Aercam will have an on-board GPS-like system, but occasionally this system may lose track of Aercam in which case the free-flyer must safely position itself.

ACKNOWLEDGMENT

The authors would like thank J. Burdick, A. Rizzi, M. Mason, S. Thrun, S. Hutchison, J. Ponce, B. Kuipers, T. Yata, and I. Rekleitis for their encouragement and insight into this work. The authors would also like to thank the anonymous reviewers for their careful and considerate comments toward this work.

REFERENCES

[1] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental geometric structure," *ACM Comput. Surv.*, vol. 23, pp. 345–405, 1991.
[2] J. Borenstein and J. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments," in *Proc. IEEE Conf. Robotics and Automation*, May 1990, pp. 572–577.
[3] J. Borenstein, B. Everett, and L. Feng, *Navigating Mobile Robots: Systems and Techniques*. Wellesley, MA: A. K. Peters, 1996.
[4] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, Mar. 1986.
[5] J. F. Canny, "Constructing roadmaps of semi-algebraic sets I: Completeness," *Artif. Intell.*, vol. 37, pp. 203–222, 1988.
[6] H. Choset and J. Burdick, "Sensor based motion planning: Incremental construction of the hierarchical generalized Voronoi graph," *Int. J. Robot. Res.*, vol. 19, no. 2, pp. 126–148, 2000.
[7] ——, "Sensor based motion planning: The hierarchical generalized Voronoi graph," *Int. J. Robot. Res.*, vol. 19, no. 2, pp. 96–125, 2000.
[8] H. Choset and P. Pignon, "Coverage path planning: The Boustrophedon decomposition," in *Proc. Int. Conf. Field and Service Robotics*, Dec. 1997.
[9] H. Choset, I. Konuksven, and A. Rizzi, "Sensor based planning: A control law for generating the generalized Voronoi graph," in *Proc. IEEE Int. Conf. Autonomous Robots*, 1997.
[10] H. Choset, K. Nagatani, and A. Rizzi, "Sensor based planning: Using a honing strategy and local map method to implement the generalized Voronoi graph," in *Proc. SPIE Conf. Systems and Manufacturing*, 1997.
[11] G. Dudeck, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 859–865, Dec. 1991.
[12] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE J. Robot. Automat.*, vol. RA-3, pp. 249–265, June 1987.
[13] J.-S. Gutmann, "Vergleich von Algorithmen zur Selbstlokalisierung eines mobil en Roboters," M.Phil. thesis (in German), Univ. Ulm, Ulm, Germany, 1996.
[14] B. Kuipers and Y. T. Byan, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *J. Robot. Autom. Syst.*, vol. 8, pp. 47–63, 1991.
[15] J. C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.
[16] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, May 1991, pp. 1442–1447.
[17] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autom. Robots*, vol. 4, pp. 333–349, 1997.
[18] ——, "Robot pose estimation in unknown environments by matching 2D range scans," *J. Intell. Robot. Syst.*, vol. 18, pp. 249–275, 1997.
[19] V. Lumelsky and A. Stepanov, "Path planning strategies for point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, pp. 403–430, 1987.
[20] H. Moravec and A. Elfes, "High resolution maps for wide angles sonar," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1985.
[21] C. Ó'Dúnlaing and C. K. Yap, "A 'retraction' method for planning the motion of a disc.," *Algorithmica*, vol. 6, pp. 104–111, 1985.
[22] N. S. V. Rao, N. Stolzfus, and S. S. Iyengar, "A retraction method for learned navigation in unknown terrains for a circular robot," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 699–707, Oct. 1991.
[23] A. Schultz, B. Yamauchi, and W. Adams, "Integrating map learning, localization and planning in a mobile robot," in *Proc. 1998 Conf. Computational Intelligence in Robotics and Automation*, Sept. 1998, pp. 331–336.
[24] H. Shatkay and L. Kaelbling, "Learning topological maps with weak local odometric information," in *Proc. IJCAI'97*, 1997.
[25] C. M. Smith and J. J. Leonard, "A multiple-hypothesis approach to concurrent mapping and localization for autonomous underwater vehicles," in *Proc. Int. Conf. Field and Service Robotics*, Dec. 1997.
[26] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," *Int. J. Robot. Automat.*, vol. 10, 1995.
[27] S. Thrun, D. D. Fox, and W. Burgard, "Probabilistic mobile robot localization and mapping," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 1998.

Howie Choset (S'93–M'96) is an Assistant Professor of Mechanical Engineering and Robotics at Carnegie Mellon University, Pittsburgh, PA, where he conducts research in motion planning and design of serpentine mechanisms, coverage path planning for de-mining and painting, mobile robot sensor-based exploration of unknown spaces, distributed manipulation with macroscopic arrays, and education with robotics. He directs the undergraduate robotics minor at Carnegie Mellon and teaches an overview course on robotics. Recently, he developed a series of Lego Labs to complement the course work and will be implementing these labs in the Fall.

Prof. Choset co-chairs the IEEE Technical Committee on Mobile Robots with X. Yun and co-chairs the SPIE Mobile Robots Conference each year with D. Gage. In 1999, he co-chaired with J. Bares the Field and Service Robotics Conference and co-organized with K. Bohringer a workshop on distributed manipulation; he edited a book on the subject, with K. Bohringer. He was awarded a Career Award, by the National Science Foundation in 1997, to continue the work in the underlying fundamentals of roadmaps for arbitrarily shaped objects; the long-term goal of this work is to define roadmaps for highly articulated robots. Recently, the Office of Naval Research started supporting him through its Young Investigator Program to develop strategies to search for land and sea mines and to construct a land-mine-search robot.

Keiji Nagatani (S'96–M'97) was born in Tokyo, Japan, in 1968. He received the Ph.D. degree from the University of Tsukuba in 1997. He was a Postdoctoral Scholar at Carnegie Mellon University, Pittsburgh, PA, for two years.

At present, he is a Lecturer in Okayama University. His current research interests include motion planning for mobile robot and control of an autonomous mobile manipulator.