Towards Exact Localization without Explicit Localization with the Generalized Voronoi Graph

Keiji Nagatani Howie Choset Sebastian Thrun
Carnegie Mellon University

Abstract. *Sensor based exploration is a task which enables a robot to explore and map an unknown environment, using sensor information. The map used in this paper is the generalized Voronoi graph (GVG). The robot explores an unknown environment using an already developed incremental construction procedure to generate the GVG using sensor information. This paper presents some initial results which uses the GVG for robot localization, while mitigating the need to update encoder values. Experimental results verify the described work.*

1 Introduction

Sensor based exploration enables a robot to explore an unknown environment, and using its sensor information, build a map of that environment. A critical component to 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. Many conventional methods attempt to make this determination via a localization scheme which updates the (x, y) coordinates of the robot. Most robots update their location by integrating data from their wheel encoders which count the number of wheel rotations (or fractional rotations). If the robot slips, the wheels do not rotate and thus this motion cannot be integrated by the robot's encoder, thereby causing error. GPS systems may offer an alternative, but commercially available systems do not work inside buildings nor provide the necessary resolution. Finally, landmarks with known locations can be deployed in the environment, but the task described in this paper considers environments that are completely unknown a priori.

In our approach, the robot exploits the geometries of the map it is building where it can either (i) locate itself on the partially explored map or (ii) conclude that it is exploring a new area. In this paper, we never explicitly update the (x, y) coordinates of the robot, but rather we locate the robot in partially explored map, as the map is being generated. The map used in this work is the the *generalized Voronoi graph* (GVG), which is a map embedded in robot's free space that captures the topologically salient features of the free space. With the GVG the robot can plan a path between any two points in the environment by first planning a path to 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 map is a kin to exploring the free space. A key feature of the GVG is that it can be constructed using line of sight range data. Therefore, the GVG is not only a representation of free space but also supplies a procedure to explore it.

This paper presents some initial results on how the robot can locate itself on a partially constructed GVG, or conclude new edge fragments are being generated. Although the robot locates itself on the GVG, it never needs to determine its (x, y, θ) coordinates (and 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 $(0, 0, 0)$.

2 Relation to Prior Work

This work draws from two areas: sensor based planning and localization. Although both of these fields are vast, only included in this paper are the works which have influenced the authors' thinking.

2.1 Sensor Based Planning and Roadmaps

Much of the previous work in sensor based planning is not complete and is limited to the plane. One class of heuristic algorithms employs a behavioral based approach in which the robot is armed with a simple set of behaviors (e.g., following a wall) [3]. 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 [2]. 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 neither afford proofs of correctness that guarantee a path can be found, nor offer well established thresholds for when these heuristic algorithms fail. Finally, these approaches do not typically generalize into higher dimensions.

There are many non-heuristic sensor based algorithms for which provably correct solutions exist in the plane (see [16] for an overview). Our approach is to adapt the structure of a provably correct classical motion planning scheme to a sensor based implementation. One such approach is based on a roadmap [4], 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).

The roadmap used in this work is the generalized Voronoi graph (GVG), which is the set of points equidistant to m obstacles in m dimensions. In the plane, the GVG is simply the *generalized Voronoi diagram* [15], the set of points equidistant to two obstacles. In ^{3}, the GVG is the one-dimensional set of points equidistant to three obstacles. The GVG method is useful in higher dimensions because the bulk of the motion planning is done in a one-dimensional space.

The GVG's incremental construction procedure [5] gives the GVG its primary strength. This incremental construction procedure only requires line of sight information and this procedure places no restrictions on the type of obstacles; obstacles need not be polygonal, polyhedral, nor convex, which are assumptions most motion planners require.

For some environments, this algorithm has been successfully implemented on a mobile robot with a ring of sonar sensors [7]. Unfortunately, the incremental construction procedure does not take into consideration errors in encoder readings and thus the original procedure is not suitable to deploy a robot into any environment of a meaningful size, say 100 square meters or more.

2.2 Localization

Localization is a major area of mobile robotics which 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 mention. See Borenstein et. al. [1] for a complete overview of current localization techniques.

Lu, Shatkay and Kaelbling, and Thrun et. al. have supplied some localization techniques which do not make overly restrictive assumptions, such all walls are orthogonal and there are no cycles in the environment. Lu, Milios and Gutmann [13], [14], [10] use gradient ascent to update various location estimates forwards and backwards in time. As a result, this approach has led to significantly larger maps that are more accurate, but is still limited to situations with bounded odometric error. Shatkay and Kaelbling [17] 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 [18], in that they both employ probabilistic representations and both use the Baum-Welch algorithm. However, the method in [17] 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 [18]. 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 in [18], 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, nor guides the robot to specific locations to reduce dead-reckoning error.

2.3 Localization with Topological Maps

There is one philosophical difference between the previous approaches and the one in this paper: previous approaches are constantly trying to update the robot's (x, y) coordinates, relative to a global frame, whereas the approach in this paper locates the robot on a map and never updates the robot's (x, y) location. Others such as Dudek [9] and Kuipers et. al [12] have reported localizations results with the same philosophy. In [9], an agent locates itself on a graph by matching up nodes and the adjacency relationship between them. This approach assumes the agent can label each node by depositing a marker at the nodes. The approach in this paper and in [12] has the robot automatically identify nodes in the topological graph from geometric environmental information.

The basic thrust of the this paper's work is quite similar to Kuiper's. In [12], the robot essentially traces double equidistance until a sensor 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 exists examples of triple equidistance that are not local maxima). Localization is achieved again by matching distinct places of the graph.

3 Sensor Based Navigation and Map Building

One of the key features of the GVG is that a robot can incrementally construct it using only line of sight information. Incremental construction of the GVG has four key components: (1) explicitly "trace" the GVG edges; (2) determine the location of the meet points (GVG vertices); (3) explore the branches emanating from the meet points; and (4) determine when to terminate the tracing procedure. This section reviews the edge tracing procedure for the GVG which has been reported in [5], [6].

Unfortunately, robots have dead-reckoning error, and the incremental construction procedure must accommodate this error, as well.

3.1 Edge Tracing

The GVG edges are traced in an incremental manner using an adaptation of numerical continuation techniques [11]. Practically speaking, these techniques trace 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 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. In the planar case $G(x) = (d_1 - d_2)(x)$, which is zero when the robot is equidistant to two obstacles.

Since G is a function of distance, it can be computed from sensors. At a point x in the neighborhood of the interior of a GVG edge, the robot steps in the direction

$$\dot{x} = \alpha \, \mathrm{Null}(\nabla G(x)) + \beta (\nabla G(x))^{\dagger} G(x), \tag{1}$$

where

- α and β are scalar gains,
- $\mathrm{Null}(\nabla G(x))$ 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 $\dot{x} = \alpha \, \mathrm{Null}(\nabla G(x))$, which is simply the tangent direction of the GVG. The stability of this control law was derived [7].

Edge tracing has been implemented on a mobile robot with a ring of sonar sensors radially pointing outward from the center. A local minimum in the sensor ring corresponds to the distance to the nearest obstacle. So, when the robot is generating a GVG edge, it is maintaining equidistance between its two smallest local minima.

3.2 Meet Point Honing

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. Sometimes meet points are called generalized Voronoi vertices. At a meet point, the robot must determine the directions of the other GVG edges that emanate from it. In the planar case, a meet point is (at least) triply equidistant to the nearest obstacles.

While generating the GVG, it is significant that the robot determine a precise location of the meet point and thus a meet point honing algorithm was introduced to trace a path that stably converges onto a meet point location [8]. The control law for honing on a meet point is similar to the one for generating new GVG edges, except the G matrix and its Jacobian are

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

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

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

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

which can be shown to be stable using the previous analysis [7]. (Note that $\mathrm{Null}(\nabla G(x)) = 0$.)

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 towards it. After taking this small step, it 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.

3.3 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 which 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 determination between old and new meet points in large environments (non-simulator) is the primary contribution of this paper, to be described in the following sections.

Finally, when a robot reaches a boundary, it simply turns around and returns to a meet point with unexplored GVG edges. Therefore, exploring the GVG in the workspace is akin to exploring a graph, where the GVG edges are the graph edges, and the GVG vertices and boundary points are the graph nodes.

4 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 re-visited 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 Figure 1. In this experiment, we deployed a Nomad 200 Mobile base into a 7×14 square meter floor space covered with linoleum tiles. The Nomad 200 can translate and orient in the plane. The robot also has sixteen radially pointing outward sonar sensors to measure distance to nearby obstacles. Since the GVG is defined in terms of distance, its incremental construction is well suited to sensor based navigation.

In this experiment, the robot starts near the hatch mark which denotes meet point 1. The robot heads towards 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 reading used to generate the GVG edge. See Figure 1. 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 (re-traced) 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 clock-wise. When the robot returns to meet point 1, the robot cannot conclude from its encoder readings that it is truly at meet point 1. This is the problem we address in this paper.

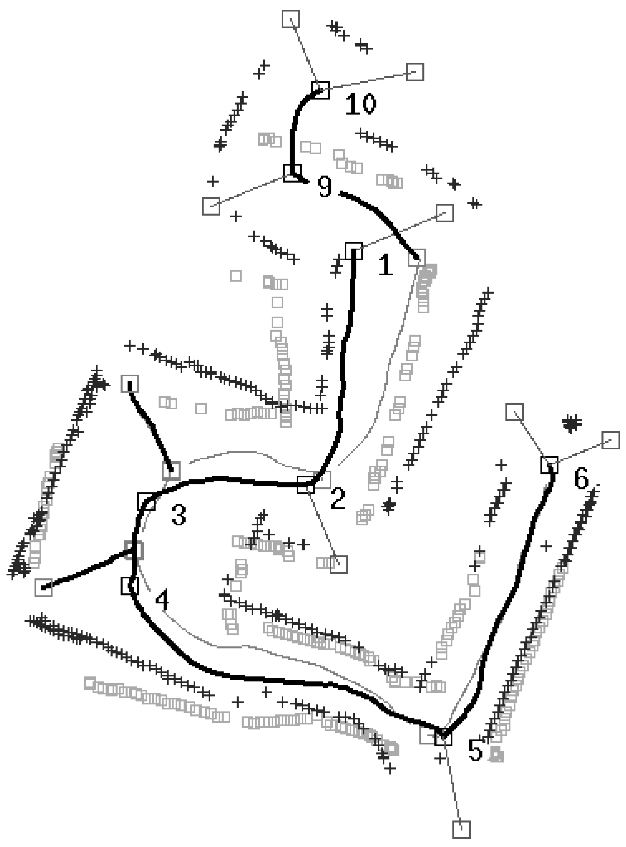

Fig. 1. Dead-reckoning Error.

5 Meet Point Identification

When the robot explores an environment, it can use meet points as a landmark to reduce dead-reckoning error. Therefore, the robot must be able to identify at which meet point it is located or determine it has found a new one.

Initially, we tried to derive a "sensor signature" for each meet point based on the robot's sixteen sonar sensor readings, but this proved to be ineffective. Using all sixteen sensors was not useful because many of the readings were in accurate 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. See Figure 2.

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 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 honing algorithm

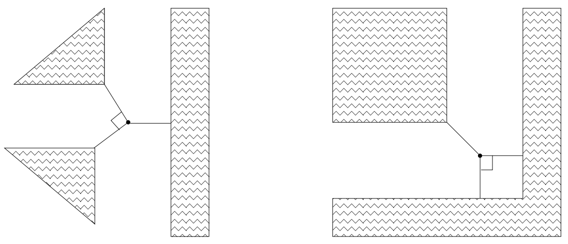

Fig. 2. The relative location of the three smallest local minima (vectors to the closest points on the closest obstacles) are similar for strikingly disparate meet points.

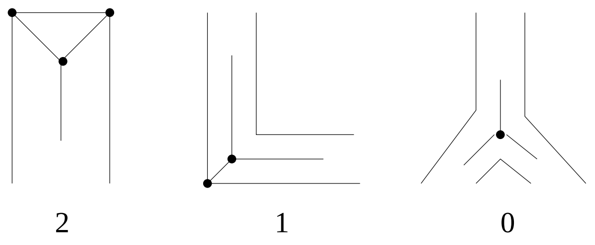

Fig. 3. Varying neighboring boundary nodes.

described in the previous section has been proven to be stable. Obviously, this distance measurement will not distinguish very well meet points among each other, but it can used to quickly eliminate any candidate matches.

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 one that has two. This criterion readily distinguishes between the two meet points in Figure 2, 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 Figure 3.

Another stable criterion looks at the number of edges emanating from a meet point. Again, while using the honing algorithm, sensor noise and position uncertainty will not affect the number of edges emanating from a meet point. See Figure 4.

The final criterion matches the relative departure angles of GVG edges emanating from the meet points. Meet point (A) in Figure 5 has the same ordered set of departure angles and meet point (B), whereas meet

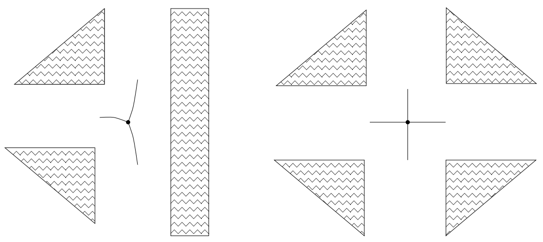

Fig. 4. Three edges is different from four edges.

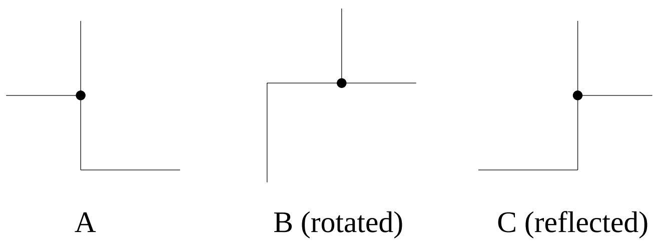

Fig. 5. Departure angle criterion does not distinguish between (A) and (B), but does discriminate between (A) and (C).

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).

6 Topological Matching: Intensionally Re-visiting a Meet Point

In this procedure we assume that 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 Figure 1 in Section 4. In Figure 1, the robot started at meet point 1 and worked its way to meet point 6. Now, the robot must re-traverse the GVG back to meet point 1 to explore the unvisited edge emanating from meet point 1. The robot first *intentionally* returned to meet point 5. Recall, the gray box is the robot's perceived location based on encoder coordinates, but in actuality the robot was located at the solid box. Since the robot knew it was going to meet point 5 and we are assuming translational dead-reckoning error is minimal, the robot re-traversed the edge to meet point 5 using its sensors and then honed onto the meet point. Since the distance traveled was the same and the robot identified this meet point as meet point 5, the robot was able to conclude where it was in the GVG without ever worrying about its encoder values.

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 this 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 (just the distance traversed from the last meet point).

7 Topological Matching: Accidentally Re-visiting a Meet Point

In the previous section, the robot intentionally directed itself to a meet point it expected to encounter. However, the robot can unexpectedly find a meet point when it encounters a new meet point or accidentally finds an already visited one. The robot must distinguish between new and old meet point in order to succesfully explore an unknown environment. The robot does this by using the previous meet point indentification schemes and by exploiting the adjacency relationships of the meet points.

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 re-traces an already explored edge emanating from the current meet point 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 traversed. This procedure is repeated once more to further reduces the set of candidate edges. This is the next criterion for localizing the robot in the GVG.

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 distances of each edge. The robot also stores the departure angle of the GVG edges.

8 Experimental Result

Figure 6 demonstrates an experiment in a real environment using a Nomad 200 mobile base where localization was performed using criteria of the previous sections. Initially the robot starts at meet point 1, then travels to 2, 3, and 4. From meet point 4, the robot heads towards meet point 1, but when it encounters meet point 1 it temporarily labels it meet point 5, but 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.

Now, the robot explores meet points 8 and 9. When the robot re-encounters meet point 2, it is fairly apparent that encoder error has significantly accrued. Although the encoders deceive the robot into thinking it

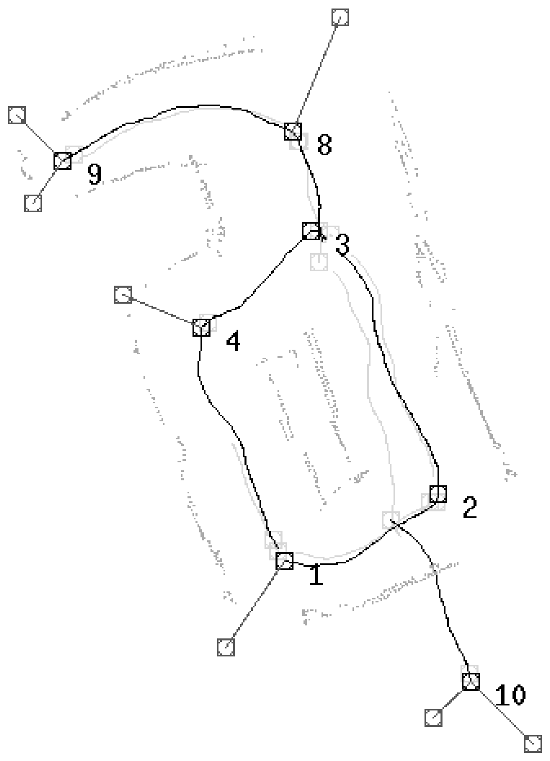

Fig. 6. Topological Matching. Dark lines correspond to the first pass and shaded lines delineate re-traces.

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 the edge. So, the robot has computed the entire TVG without ever resorting to dead-reckoning, nor having to update its encoders.

9 Relation to Future Work

The work presented in this paper is only the first step towards the long-term goal of localization. The first 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.

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. Although the GVG does not change that match, performing topological matching on a varying GVG is quite difficult. Another problem deals with meet point 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 [18] to allow for meet points that appear and disappear. This will lead to an implementation of this approach in dynamic environments.

10 Conclusion

Traditional localization procedures require detailed sensor processing and an a priori detailed map for robot localization. This paper presents some initial results for which a robot can explore, map, and localize itself in an unknown environment. A novel feature of the presented localization procedure is that it does not rely on encoder readings, nor other sensors, to explicitly determine its location. Instead, the robot exploits the topology of the map it is generating to determine its location, or conclude that it is visiting unexplored territory.

This work uses a map termed the generalized Voronoi graph (GVG), which is the locus of points equidistant to two obstacles in the plane. A robot using sonar sensors can reliably generate this structure using an incremental construction procedure which is summarized here, but detailed in other works. The incremental construction procedures are robust and have been proven to be stable, even in the presence of sensor noise. Experimental results also validate the GVG construction procedures.

Another feature of the GVG incremental construction routine is that it automatically instructs the robot where to go to explore new regions of the environment. This is important when exploring unknown environments. In this paper, the incremental construction routine is updated to direct the robot to reduce its position uncertainty in the GVG graph, whether the robot is exploring an unknown environment or it is re-traversing a known vicinity.

In this procedure, the robot never determines its (x, y, θ) coordinates. It only needs to know which edges and nodes correspond to each other. Therefore, the robot has localized itself, but never needed to explicitly localize itself, hence the title of this paper. Furthermore, we are not explicitly comparing detailed sensor signatures of the robot's current state to a previously stored state. Instead, we are using the topology of the GVG to localize. (The authors would also like to thank Jon Canny at Berkeley for his comments which help spark this research philosophy.)

References

[1] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques.* A. K. Peters, Ltd., Wellesley, MA, 1996.

[2] J. Borenstein and J. Koren. Real-time Onstacle Avoidance for Fast Mobile Robots in Cluttered Environments. In *IEEE Conference of Robotics and Automation*, pages 572–577, Cincinnati, Ohio, May 1990.

[3] R.A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal on Robotics and Automation*, RA-2, March 1986.

[4] J.F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[5] H. Choset and J.W. Burdick. Sensor Based Planning, Part II: Incremental Construction of the Generalized Voronoi Graph. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, 1995.

[6] H. Choset and J.W. Burdick. Sensor Based Planning: The Hierarhical Generalized Voronoi Graph. In *Proc. Workshop on Algorithmic Foundations of Robotics*, Toulouse, France, 1996.

[7] H Choset, I Konuksven, and A Rizzi. Sensor Based Planning: A Control Law for Generating the Generalized Voronoi Graph. In *Proc. of IEEE Int. Conf. on Autonomous Robots*, Monterey, CA, 1997.

[8] 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 *SPIE Conference on Systems and Manufacturing*, Pittsburgh, PA, 1997.

[9] G. Dudeck, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7:859–865, Dec. 1991.

[10] J.-S. Gutmann. Vergleich von algorithmen zur selbstlokalisierung eines mobil en roboters. Master's thesis, University of Ulm, Ulm, Germany, 1996. (in German).

[11] H.B. Keller. *Lectures on Numerical Methods in Bifurcation Problems.* Tata Institute of Fundamental Research, Bombay, India, 1987.

[12] B. Kuipers and Y.T. Byan. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[13] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapp ing. *Autonomous Robots*, 4:333–349, 1997.

[14] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, to appear.

[15] C. Ó'Dúnlaing and C.K. Yap. A "Retraction" Method for Planning the Motion of a Disc. *Algorithmica*, 6:104–111, 1985.

[16] N.S.V. Rao, S. Kareti, W. Shi, and S.S. Iyenagar. Robot Navigation in Unknown Terrains: Introductory Survey of Non-Heuristic Algorithms. *Oak Ridge National Laboratory Technical Report*, ORNL/TM-12410:1–58, July 1993.

[17] H Shatkay and L. Kaelbling. Learning topological maps with weak local odometric informati on. In *Proceedings of IJCAI-97.* IJCAI, Inc., 1997. 1997.

[18] S. Thrun, D. D. Fox, and W. Burgard. Probabilistic mobile robot localization and mapping. In **submitted to** *IEEE ICRA*, 1998.