Sensor Based Planning: Using a Honing Strategy and Local Map Method to Implement the Generalized Voronoi Graph

Howie Choset, Keiji Nagatani and Alfred Rizzi
Mechanical Engineering and Robotics
Carnegie Mellon University

Abstract

This work prescribes the procedures that are required to implement, on a conventional mobile robot, a sensor based motion planning algorithm based on the generalized Voronoi graph (GVG). The GVG is a roadmap of a static environment; recall that a roadmap is a one-dimensional representation of an environment which the robot can use to plan a path between any two points in that environment. Once the robot has constructed the roadmap, it has in essence explored the environment. This work describes some issues in incrementally constructing the GVG with a mobile robot with a ring of sonar sensors. Specifically, we consider some issues in specularity and dead-reckoning error reduction.

Keywords: sensor based planning, exploration, Voronoi diagrams, mapping, mobile robot, control law, honing.

1 Introduction

Sensor based planning integrates sensor information into the planning process, in contrast to *classical planning*, which requires that full knowledge of the world be available to the robot prior to the planning event. The realistic deployment of robots into unknown environments requires sensor based planning. However, when full knowledge of the environment is available, but is too difficult to input into the robot, sensor based planning bypasses the need to enter the environmental model: the robot simply explores the environment and builds up its own representation.

This work uses a sensor based planning approach that relies on a *roadmap,* a representation which captures all of the salient geometric features found in a robot's free space. Once a robot constructs a roadmap, it has in effect, explored an environment. Previous work[5] includes the prescription of an incremental construction procedure for a particular class of roadmaps. This procedure requires only line of sight information, which is an upper bound to what sensors provide, and is thus amenable to sensor based implementation.

Unfortunately, this approach does not consider issues, such as sensor quantization, the physics of sonar sensors, and dead-reckoning error. The long-term goal of the authors' research program is to adapt the incremental construction procedure for realistic sensor based use. This paper introduces two techniques, meet point honing and sensing with a local map, both of which help adapt the incremental approaches[5] to reliable use on a Nomad 200 Mobile base with a ring of ultrasonic sensors. Both approaches exploit the geometry of the roadmap to make use of noisy sonar data. The meet point honing strategy will serve as a tool to help reduce dead-reckoning error and is general to any Euclidean multi-dimensional space.

2 Relation to Previous Work

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).[2] 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.[1] This approach may use lots of computer memory to store the environment, but 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 because they would require an enormous amount of memory and processing speed to handle billions upon billions of m-voxels that represent an m-dimensional space.

There are many non-heuristic algorithms for which provably correct solutions exist in the plane (see[13] for an overview). For example, Lumelsky's "bug" algorithm[11] is an example of one of the first provably correct sensor based schemes to work in the plane. However, this algorithm (like many described in[13]) requires knowledge of the goal's location during the planning process. Furthermore, this algorithm simply returns a path from the start to the goal. The resulting path does not reflect the topology of the free space and thus, it cannot be used to guide future robot excursions. The goal of this work is not only to plan a path from start to goal, but also to incrementally build a concise representation of the robot's environment.

An example of a method that has a provably correct solution, uses realistic sensor assumptions, and need not require prior knowledge of the goal's location is described in.[15] In this method, the robot forms a graph of a bounded freespace by circumnavigating each of the obstacles, and then creating an adjacency relationship between obstacles within line of sight of each other. However, this method requires landmarks in constructing its map and is limited to the planar case. Another example of exploration with realistic sensor assumptions can be found in.[10] In this case, a map called a "topological map" of the world is formed. However, the algorithm assumes that overlapping convex obstacles meet at nearly ninety degrees.

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,[3] 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). These methods are useful in higher dimensions because the bulk of the motion planning is done in a one-dimensional space.

We chose roadmaps because of their concise representation and their upward compatibility into higher dimensions. Roadmaps are useful in m-dimensional spaces because a bulk of motion planning occurs on the one-dimensional roadmap. Roadmaps are also concise in that they do not require that the entire environment be discretized into a fine resolution of pixels.

The roadmap, used in this work, can trace its roots to the *generalized Voronoi diagram* (GVD) in the plane (i.e., when $m = 2$). Ó'Dúnlaing and Yap[12] first applied the GVD, which is the locus of points equidistant to two or more obstacles, to motion planning for a disk in the plane. However, the method in[12] requires full knowledge of the world's geometry prior to the planning event. In[14] an incremental approach to create a Voronoi Diagram-like structure, which is limited to the case of a plane, was introduced.

The GVG may not have nay clear advantages over other roadmap methods, but its 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.[6] Unfortunately, the incremental construction procedure does not take into consideration the properties of the sonar sensors such as specularities, sensor range, and quantization. For example, sonar sensors are not good at detecting sharp corners that may appear in typical environments. A sharp corner is "invisible" to the sensor that is facing it. Recently, a new method,[9] where the robot backs up each time an obstacle "disappears" exploits the geometry of the GVG to trace it out even though the robot loses sight of an obstacle, was introduced. In this paper, the robot creates a local map, similar to a certainty grid, and thus no longer needs the "back up" approach.

3 Background Work

The work presented in this paper is based on the GVG, which has been described earlier in the literature.[4–7] A review of the GVG and its incremental construction procedure is included below for the sake of completeness, but it could be omitted by a reader already familiar with this work.

3.1 Distance Function

Assume the robot is a point operating in a work space, $\mathcal{W}$, which is a subset of an m-dimensional Euclidean space, $\mathbb{R}^m$. $\mathcal{W}$ is populated by convex obstacles $C_1, \dots, C_n$. Non-convex obstacles are modeled as the union of convex shapes. The distance between a point and an obstacle is the shortest distance between the point and all points in the obstacle. The distance function, and its "gradient," respectively are

$$d_i(x) = \min_{c_0 \in C_i} \|x - c_0\| \quad \text{and} \quad \nabla d_i(x) = \frac{x - c_0}{\|x - c_0\|},$$

where (1) d_i is the distance to obstacle C_i from a point x, and (2) the vector $\nabla d_i(x)$ is a unit vector in the direction from x to c_0, where c_0 is the nearest point to x in C_i. Typically, the environment contains multiple obstacles, and thus distance is measured to multiple obstacles with the multi-object distance function, $D(x) = \min_i d_i(x)$

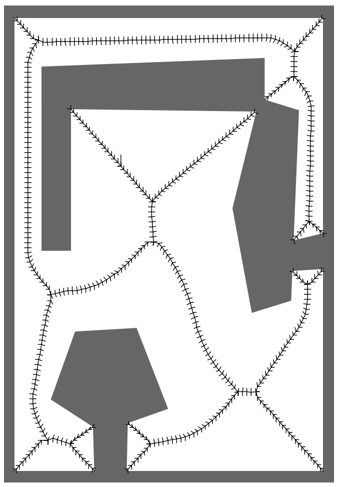

Fig. 1. The ticked line segments are the planar GVG for the bounded environment. The ticks point at the nearest point on an obstacle, and are thus the negated gradients.

3.2 The Generalized Voronoi Graph

The basic building block of the GVG is the set of points equidistant to two sets C_i and C_j, such that each point in this set is closer to the objects C_i and C_j than any other object. We term this structure the *two-equidistant face*,

$$\mathcal{F}_{ij} = \{x \in \mathbb{R}^m : 0 \leq d_i(x) = d_j(x) \leq d_h(x) \quad \forall h \neq i, j \quad \text{and } \nabla d_i(x) \neq \nabla d_j(x)\}.$$

A two-equidistant face has co-dimension one in the ambient space, and thus in the plane, a two-equidistant face is one dimensional.[4]

The Pre-image Theorem asserts that the union of the two-equidistant faces, i.e., the GVD, is $(m-1)$-dimensional.[4] The GVD does reduce the motion planning problem by a dimension, but a one-dimensional roadmap is required. Observe that the two-equidistant faces, $\mathcal{F}_{ij}$, $\mathcal{F}_{ik}$, and $\mathcal{F}_{jk}$ intersect to form an $(m-2)$-dimensional manifold that is equidistant to three obstacles. Such a structure is termed a *three-equidistant face* and is denoted $\mathcal{F}_{ijk}$. That is,

$$\mathcal{F}_{ijk} = \mathcal{F}_{ij} \bigcap \mathcal{F}_{ik} \bigcap \mathcal{F}_{jk}$$

This intersection procedure is repeated until a one-dimensional structure is formed; such a structure is an m-equidistant face, $\mathcal{F}_{i_1 \ldots i_m}$ and is a one-dimensional set of points equidistant to m objects in m dimensions. (Also note, an $m+1$-equidistant face is formed in a similar way and is always a point.)[4]

The *generalized Voronoi graph* (GVG) is the collection of m-equidistant faces and $m+1$-equidistant faces. Later, the m-equidistant faces are termed *generalized Voronoi edges* and $m+1$-equidistant faces are termed *meet points.* Note that the GVD is $m-1$-dimensional whereas the GVG one-dimensional. Also, the GVD is the locus of points equidistant to *two* obstacles whereas the GVG is the locus of points equidistant to m obstacles. In the planar case, the GVG and GVD coincide (See Fig. 1).

3.3 Incremental Construction of the GVG

A key feature of the GVG is that it can be incrementally constructed using line of sight range information. In the scenario in which the robot has no a priori information about the environment, the robot must construct a roadmap in an incremental manner because most environments do not contain one vantage point from which a robot can "see" the entire world, and thereby allow a robot to construct a roadmap from such a single vantage point. The incremental construction techniques described in this section provide a rigorous approach to constructing the GVG using only line of sight sensory information.

Incremental Accessibility. The robot accesses the GVG by increasing its distance to the nearest obstacle. Then, while maintaining double equidistance, the robot increases its distance from the two closest obstacles until it is three-way equidistant. This procedure is repeated, until the robot is m-wise equidistant. In the planar case, the robot simply moves in a direction opposite to which the nearest sensor is facing.

Tracability: Numerical Technique. Once the robot accesses the GVG, it must trace it out. In an incremental context, the property of connectivity is interpreted as *tracability.* The GVG edges are traced in an incremental

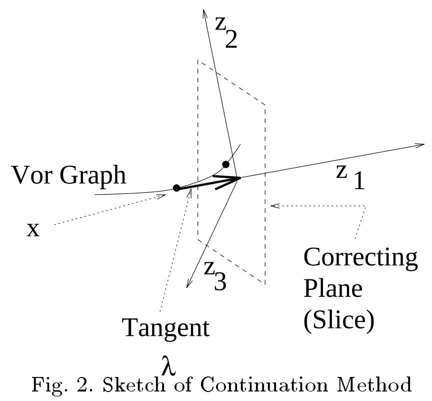

Fig. 2. Sketch of Continuation Method

manner using an adaptation of numerical continuation techniques.[8] These techniques usually have two phases: (1) a prediction step, and a correction procedure. Typcially, the prediction is along the tangent to the GVG, and the correction procedure occurs on a hyperplane orthogonal to the GVG. In the planar case, the tangent space is the line which is the perpendicular bisector of the line segment which connects the two closest points on the two closest obstacles.

More generally, these techniques trace the roots of the expression $G(y, \lambda) = 0$ as the parameter λ varies. Let x be the coordinates of a point on a GVG edge. At $x \in \mathbb{R}^m$, assign a local coordinate system (y, λ) such that λ points along the tangent of the GVG edge and the y coordinates spans Y, the hyperplane orthogonal to the GVG edge. Let Y be termed the "normal plane" (Figure 2). At a point $x \in \mathbb{R}^m$, $G(y, \lambda)$ has the form

$$G(y,\lambda) = \begin{bmatrix} d_1(y,\lambda) - d_2(y,\lambda) \\ \vdots \\ d_1(y,\lambda) - d_m(y,\lambda) \end{bmatrix} \tag{1}$$

where d_i is the single object distance function to the m closest obstacles. Since G is a function of distance, it can be computed from sensors.

Note that G maps $Y \times \mathbb{R}$ to Y where $\mathbb{R}^m = \mathbb{R} \times Y$. The function $G(y, \lambda)$ assumes a zero value only on the GVG. Hence, if the Jacobian of G is surjective,[5] then the implicit function theorem implies that the roots of $G(y, \lambda)$ locally define a GVG edge as λ is varied. A robot can locally construct a GVG edge by numerically tracing the roots of this function.

Just like the planar case, in an m-dimensional space the prediction step moves the robot a small step, $\Delta\lambda$, along the tangent to the GVG edge. The tangent is the vector orthogonal to the hyperplane which contains the m closest points on the m closest obstacles.[5] This is the null space of the Jacobian of G, denoted ∇G.

Typically, the prediction step takes the robot off the GVG and a correction method is used to bring the robot back to the GVG. In our method, the correction occurs on a hyperplane orthogonal to the tangent (i.e., along the y coordinates). Since $\nabla_Y G(y, \lambda)$ is full rank at $x = (y, \lambda)$,[5] it is possible to use an iterative Newton's Method to implement the correction procedure. If y^k is the estimate of y at the kth iteration, the $(k + 1)$st iteration of the correction procedure is defined as

$$y^{k+1} = y^k - \left(\nabla_Y G\right)^{-1} G(y^k, \lambda), \tag{2}$$

where $\nabla_Y G$ is evaluated at (y^k, λ).

To construct $G(y, \lambda)$ and $\nabla_Y G(y, \lambda)$, one only requires the distance and direction to the two closest objects. This information is within line of sight of the robot, thus making the incremental constructive method amenable to sensor based implementation.

Tracability: Control Law. The numerical continuation methods produce paths for the robot that are jagged. For example in the planar case, the robot steps in the tangent direction during its prediction phase, and then rotates ninety degrees to enter its correction phase. After the correction phase, the robot must rotate again to re-orient itself on the GVG. These rotations take time and cause additional wheel slippage.

Instead, the robot should use a control law which continuously controls the heading of the robot allowing for smooth paths. The numerical continuation methods of Section motivates the below described control law. See

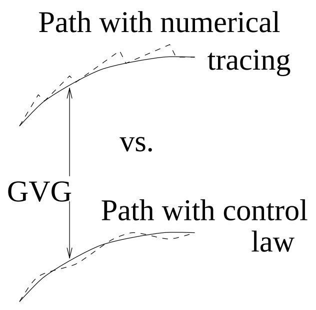

Fig. 3. Dotted lines represent path traced out by either the numerical continuation methods or the control law.

Figure 3.

In essence, the control law merges the prediction and correction phases. At a point x in the neighborhood of the interior of a GVG edge, the robot steps in the direction

$$\dot{x} = \alpha\,\mathrm{Null}(\nabla G(x)) + \beta(\nabla G(x))^{\dagger} G(x), \tag{3}$$

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 $\dot{x} = \alpha\,\mathrm{Null}(\nabla G(x))$. To make notation easier to follow, let

- $G = G(x)$,
- $\nabla G = \nabla G(x)$,
- $\nabla G^{T} = (\nabla G(x))^{T}$,
- $\nabla G^{\dagger} = (\nabla G(x))^{\dagger}$, and
- $\nabla G^{\perp} = (\nabla G(x))^{\perp}$.

So in short-hand notation, the robot takes the following step

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

For the mobile robot, the G matrix and its Jacobian are

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

Therefore, the robot takes the following step

$$\dot{x} = \alpha(\nabla d_1(x) - \nabla d_2(x))^{\perp} + \beta(\nabla d_1(x) - \nabla d_2(x))^{\dagger}(d_1(x) - d_2(x)).$$

Note that $\alpha(\nabla d_1(x) - \nabla d_2(x))^{\perp}$ is the tangent space to the GVG and the $\beta(\nabla d_1(x) - \nabla d_2(x))^{\dagger}(d_1(x) - d_2(x))$ is the correcting factor. This control law was shown to be stable[7] in some neighborhood of the GVG. Experiments have shown that this neighborhood can be as big as thirteen centimeters around the GVG. When the robot "falls" off of the GVG far enough, it simply re-invokes the continuation methods which have a larger neighborhood of convergence. Determing the exact size of these neighborhoods still has to be determined. Also, this control law is stable when $\frac{|\beta|}{|\alpha|} > 1$.

Terminating Conditions. The explicit terminating conditions for edge tracing are described in,[5] but in the planar case there are two terminating conditions: a meet point, where three GVG edges join, and a boundary point, where a GVG edge intersects the boundary of the environment.

Finding the meet points is essential to proper construction of the graph. While a meet point occurs when the robot is equidistant to $m + 1$ objects, it is unreasonable to expect that a robot can exactly detect such points because of sensor error. Furthermore, since the robot is taking finite sized steps while tracing an edge, it is unlikely that the robot will pass exactly through an $(m+1)$-equidistant point. However, as shown in Figure 4, meet points can be robustly detected by watching for an abrupt change in the direction of the (negated) gradients to the m closest obstacles. Such a change will occur in the vicinity of a meet point.

After reaching a meet point, the robot explores an unexplored edges emanating from the meet point. At a boundary point, the robot simply back tracks to a previous meet point that has unexplored GVG edges associated

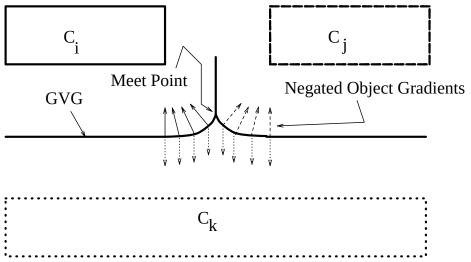

Fig. 4. Meet Point Detection.

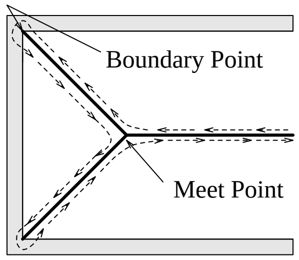

Fig. 5. Boundary Point. The arrows delineate the path a robot would follow while construction the GVG.

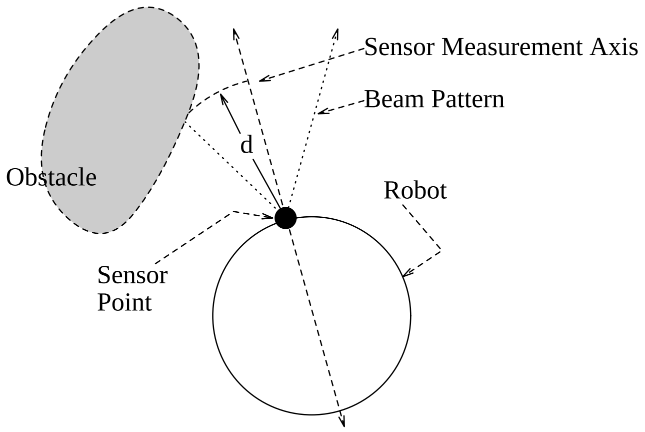

Fig. 6. Simplified Distance Measurement Sensor Model

with it. See Figure 5. Once all meets points have no unexplored edges emanating from them, the GVG has been fully explored. Note that incremental construction of the GVG is akin to a graph search where GVG edges are the "edges" and the meet points and boundary points are the "nodes." If the robot is looking for a particular destination whose coordinates are known, then the robot can invoke graph searching techniques, such as the A-star algorithm, to control the tracing procedure.

4 Previous Implementation

The GVG scheme was implemented on a mobile robot with a ring of ultrasonic sensors radially distributed around the perimeter of the robot. These sensors determine distance by measuring the time of flight of the ultrasound pulses that reflect off an object and return to the sensor. Although these sensors provide accurate distance measurements, their readings are not precise in the azimuth.

The GVG incremental construction procedure has been implemented using a simplistic sensor model.[6] This model assumes that the sensors are rigidly attached, pointing radially outward from the robot. The sensors measure distance to nearby obstacles, along a fixed direction termed the *sensor measurement axis*. The sensor measurement axis is a function of the robot's position and orientation (See Fig. 6). Local minima of the distance readings correspond to distance to the nearby obstacles.[6] Once the distance to nearby obstacles is obtained, then the robot can employ the edge tracing techniques described in Section .

Recall the robot terminates the GVG edge tracing procedure at a meet point, at which time it begins tracing a new GVG edge. In order to do this, the robot first must determine when it encounters a meet point. When there is an "abrupt" change in a sensor associated with one of the two closest obstacles, then the robot has passed by a meet point. Since the robot has few sensors and thus a low resolution, an "abrupt" change is indicated by a shift of the local minimum by more than one sensor location (Fig. 7). (Note Section contains more details on detecting meet points.)

Our experiments indicate that this sensor model and meet point detection scheme work well when the two closest features on the two closest obstacles are flat faces, curved faces, and gentle corners. This is so, because the sensors with the two smallest local minima (i.e., those that correspond to the two closest obstacles) have a sensor centerline axis that is nearly parallel to the obstacle normals.

Unfortunately, the previous sensor model does not consider specularities associated with sonar sensors. When

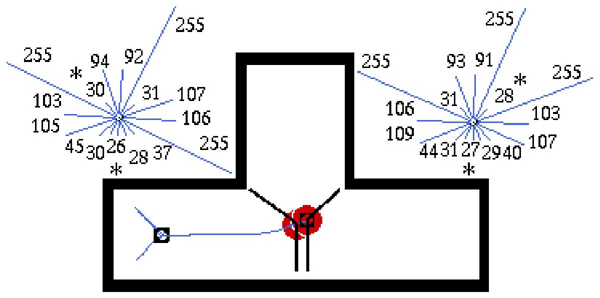

Fig. 7. Naive Meet Point Detection: The robot determines it has passed a meet point when the direction to one of the two closest obstacles changes.

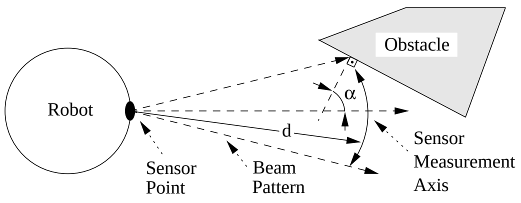

Fig. 8. New Sensor Model.

the sensor faces a sharp corner, there may or may not be a return echo because the ultrasound did not reflect back to the sensor. To demonstrate this could be a problem, consider the following example which was implemented in simulation with a sensor model which assumes an echo returns when the angle between the normal to the surface and the sensor measurement axis is less than α. Therefore, with this model, all obstacles in the sensor cone are not necessarily detected. See Figure 8.

Example 4.1 (Sharp Corner) Note the differences between the environments in Figs. 9 and 10. In Fig. 9, C_3 has a blunt protrusion (not sharp) in contrast to C_3 in Fig. 10. Now, compare the GVG traces in Figs. 9 and 10. The GVG trace in Fig. 9 is correct, whereas the one in Fig. 10 is not because there is an incorrect meet point.

In both cases, a meet point is detected when there is an abrupt change in one of the two closest obstacles, as perceived by the sensors. However in Fig. 10, there is an abrupt change in one of the two closest obstacles because the distance to C_3 became infinite as a result of the sonar sensor's inability to see C_3 because of its sharp corner. In effect, C_3 became "invisible" to the robot and the two perceived closest obstacles become C_1 and C_2. See Figs. 11 and 12. Note that the naive sensor model would consider C_3 to be visible.

Finally, note that this example used data which simulated the *worst case* scenario of when the robot is facing a sharp corner and cannot "see" it. In actual practice, the robot may or may not see it.

5 Local Map

In the previous implementation, the sixteen sonar sensor readings are stored in an array and then the local minima are determined. The sensors associated with the two smallest local minima are the distances to the two

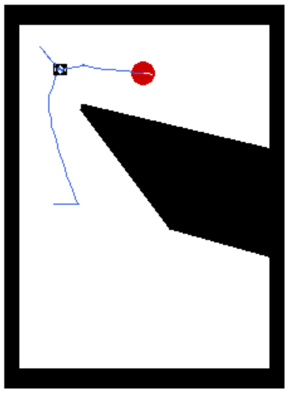

Fig. 9. Working.

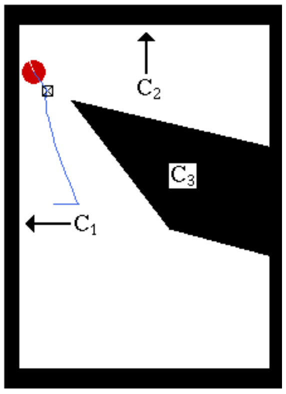

Fig. 10. Not Working

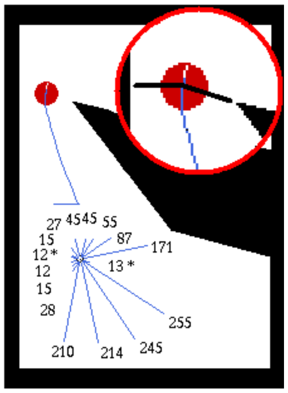

Fig. 11. Robot has traced out GVG edge fragment. The lower left portion displays the sonar sensor values and the upper-right portion contains a closeup of the robot and its two local minima.

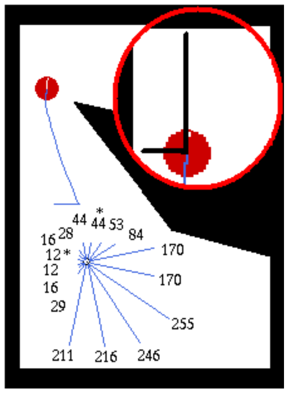

Fig. 12. Robot's sensors cannot see C_3 and thus C_1 and C_2 become to two closest obstacles.

closest obstacles.

In order to deal with sharp corners and other sensor anomolies, the robot generates a local map which is used to enrich the robot's current sensor readings. The robot only stores the sensor readings associated with the local minima of the sensor ring because these tend to be lined up (or close to it) with the normals of the obstacle surfaces in the environment. Local minima which are above a maximum threshold are ignored because long range sensor data, even if it is a local minimum, tends not to be accurate.

Essentially, the local map is list of $x-y$ coordinates stored in a ring buffer, so old data is thrown away as robot the progesses through the environment. Each time the robot takes a sonar scan, it stores all sixteen sonar readings in a *current sensor* array. The robot determines all of the local minima and then infers the $x-y$ locations from where these sonar echos came. The angle these $x-y$ locations make, with respect to the robot, are compared to angles of each record in the ring buffer. If the angle of a local minimum does not correspond to an angle in the ring buffer, then the new $x-y$ coordinate is added to the ring buffer. If the angle is the same and the distance to the new sonar reading is less than the distance to the $x-y$ coordinate in the ring buffer, then the $x-y$ coordinate of the sonar echo replaces the $x-y$ coordinate in the ring buffer. Otherwise, the distance to the $x-y$ coordinate in the ring buffer replaces the value of the sonar sensor in the current sensor array. With the updates current sensor array, the robot determines the two smallest local minima and proceeds with the incremental construction routine. Figure 13 demonstrates this process; the dots are the perceived locations of echoes.

It should be noted that the $x-y$ coordinates are relative to a global coordinate frame, but with dead-reckoning error. That is, the coordinates are not with respect to the exact inertial coordinate frame, but rather to the perceived world coordinate frame according to the robot's encoders.

6 Meet Point Honing

The incremental construction procedure described in Section does not pay any respect to sensor and dead-reckoning error. Our initial experiments indicate that the robot does not have to trace an exact GVG edge; that is, there is some built-in robustness to the graph edges which allows the GVG edges to contain error.

However, the robot must accurately locate the meet points to capture an accurate topological model of the environment. The robot does not find the exact location of the meet point during the edge tracing process because it is taking finite steps and thus passes by the meet point. Also, sensor noise prevents the robot from detecting an exact $m+1$ equidistance location.

Therefore, we have introduced a new meet point honing strategy to bring the robot closer to the exact location of the meet point. This procedure only works when the robot is in the neighborhood of a meet point, which is the case in our implementation. 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

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

Therefore, $G(x) = 0$ at a meet point, i.e., $d_1(x) = d_2(x) = d_3(x)$. Note that $\text{Null}(\nabla G(x)) = 0$ because, whereas before with the GVG the null space was a one-dimensional line, now the null space is a point. This point

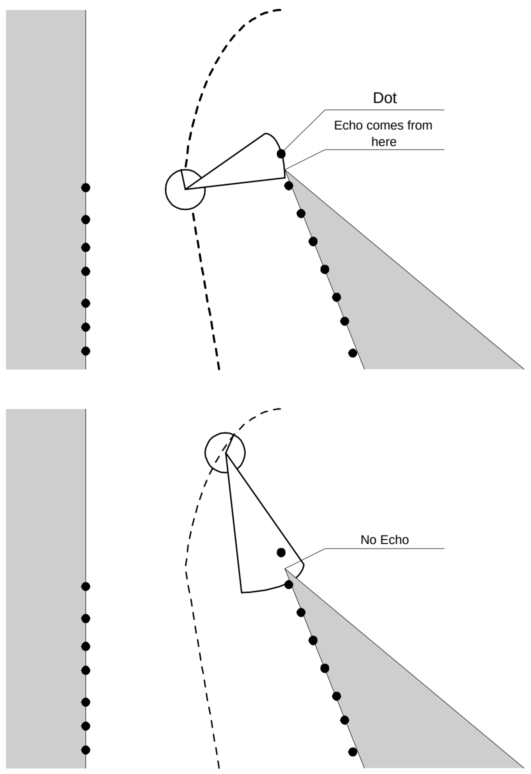

Fig. 13. (top) Data points are stored in the ring buffer. (bottom) Data point from ring buffer is used when an echo does not return.

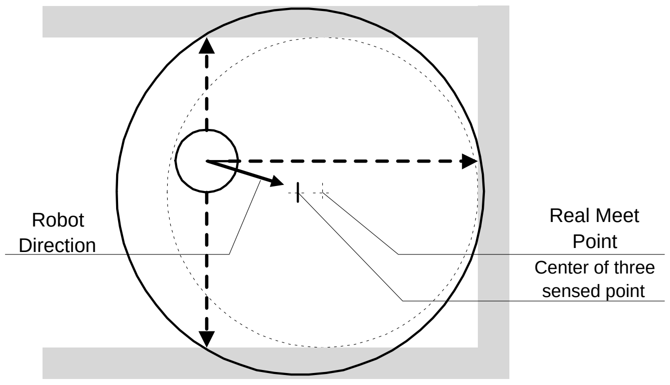

Fig. 14. Meet Point Honing

corresponds to the origin of the tangent space at 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]

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. See Figure 14

Current work considers having the robot use this honing strategy to reduce dead reckoning error when the robot *intentionally* visits an already encountered meet point. When the robot re-encounters such a meet point,

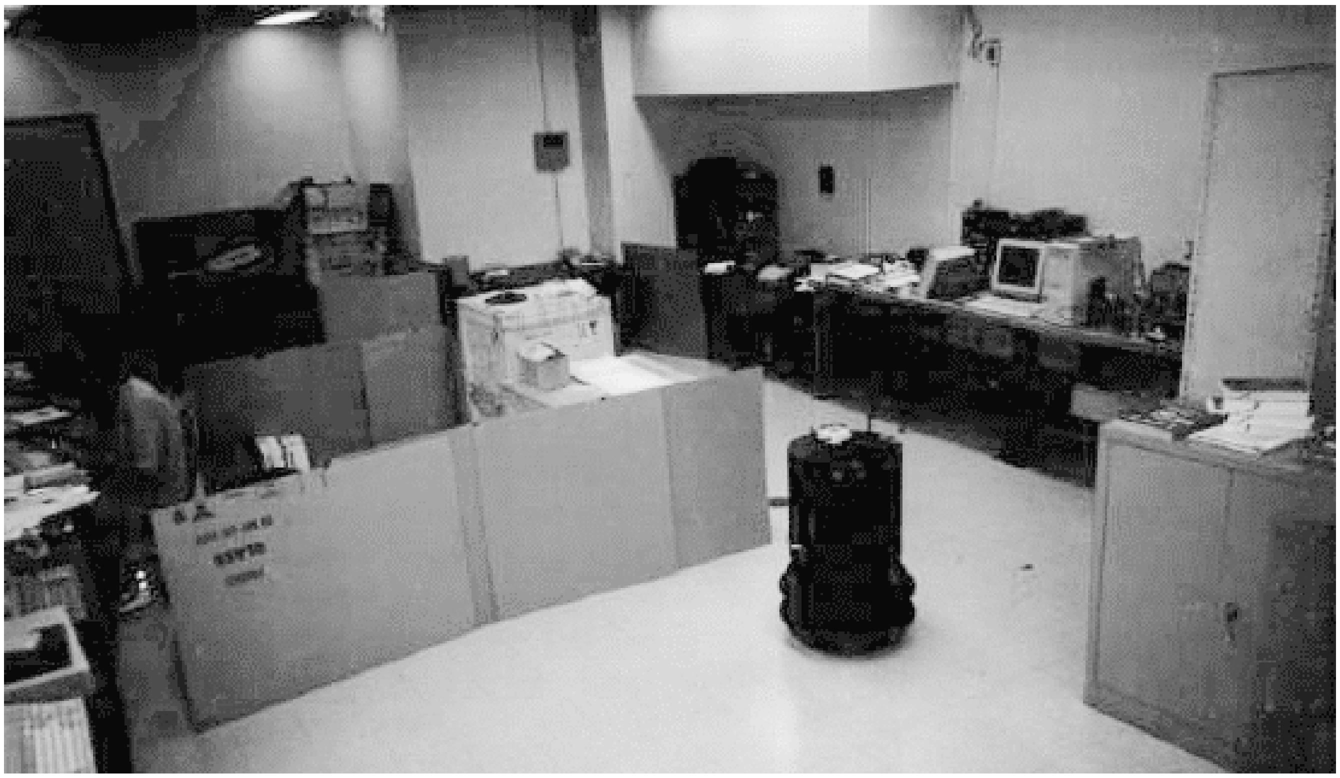

Fig. 15. Laboratory Setup.

it can hone in on the meet point's exact location, and then compare the meet point's previous coordinates with the meet point's current coordinates, based on the robot's current encoder values. The difference is the amount of encoder error accrued since the robot visited the meet point. The robot can then use this information to reduce dead-reckoning error.

The case of unintentially revisiting meet points and statistically determining new meet points is also a current topic of research. A robot unintentionally revisits a meet point when there are cycles in the GVG. Therefore, the above mentioned procedure is only used in backtracking.

7 Experimental Results

By adding the above local map process and honing algorithm to the previous implementation, we have demonstrated improved exploration performance on a mobile robot in an indoor environment, such as the one displayed in Figure 15. This environment is a maze of cardboard walls which can meet at any angle (we are not restricted to ninety degree junctions). Since the ultrasonic range sensors are located at one meter in height, lower objects, such as small chairs, are removed from this environment.

Initially, the robot has no a priori information of the environment and is arbitrarily placed in it. First, the robot invokes the accessibility procedure to obtain double equidistance, i.e., reach the GVG. Once the robot has accessed the GVG it uses the control law until it reaches the vicinity of a meet point or a boundary point. While tracing the GVG edge, the robot may not necessarily trace the exact edge, but it must know the exact location of the meet point in order to capture an accurate topological map of the environment. Therefore, when the robot reaches the vicinity of a meet point, it invokes a "meet point honing" algorithm to converge onto the meet point. Normally, the robot reaches the meet point in two steps. When the robot encounters a boundary point, it simply turns around and retraces back to a meet point which has an unexplored edge emanating from it. When all meet points have no unexplored edge, the robot has generated the complete GVG and thus has explored the entire environment.

Figure 16 contains one experimental run. The equidistant lines are the GVG edges that the robot generated. The hatched marks are the meet points, i.e., the nodes of the GVG. The dots surrounding the GVG are sensor

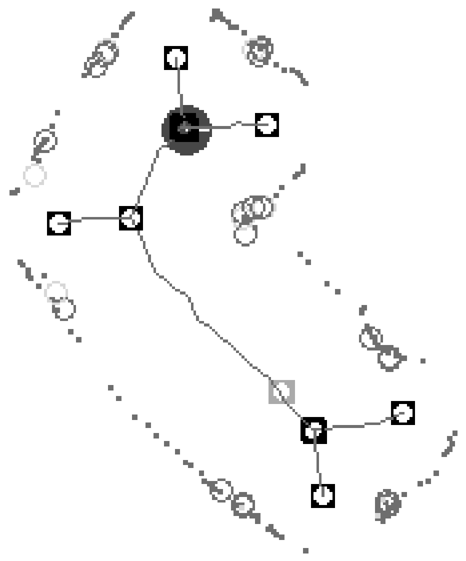

Fig. 16. Experimental Run.

data points, which are not stored, but are included for viewing purposes.

This work is only the beginning of a long-term research program. For example, the proposed methods do not consider obstacles which may be below the sonar sensor height. Furthermore, the exploration routine assumes the environment is static. These assumptions will be relaxed in future work.

8 Conclusion

This work uses a geometric structure termed a roadmap, which is a collection of one-dimensional curves that captures the salient topological and geometric properties of a robot's environment. Two key advantages of the roadmap in this work are: (1) it can be constructed using sensor data and (2) it is a one-dimensional subset of the robot's space, and thus searching in an m-dimensional robot configuration space is reduced to a one-dimensional search. Motion planning with a roadmap is achieved by (1) planning a path onto the roadmap, (2) traversing the roadmap, and (3) departing the roadmap. If the robot knows the roadmap, then it can generate a path between two points in the environment. Therefore, exploration of an environment is reduced to incrementally constructing a roadmap.

The roadmap used in this paper is the generalized Voronoi graph (GVG). Although the GVG may not have any clear advantages for motion planning, its is well suited to sensor based planning because the GVG can be constructed using sensor data. This paper described the next few steps in implementing the GVG for real sensor based use. Initially, the GVG incremental construction routine did not pay any respect to sensor noise and dead-reckoning error.

The robot can exploit some of the geometry of the GVG to help it zero out dead reckoning error. Specifically, the robot used the meet points (the nodes) of the GVG to zero out dead reckoning error. The robot zeroed out dead reckoning error by comparing its current encoder readings to those that were recorded the previous time the robot visited the meet point. This procedure only works when the robot was exactly (or close to) the meet point, so it used a new meet point honing procedure to locate precisely the meet point. This new procedure was described in this paper and was an extension of previous edge tracing techniques.

Finally, using a local certainty-grid type map, the robot was able to deal with sharp corners that suddenly disappear. Since the certainty grid approaches have shown great utility in previous experiments, it made sense to marry those approaches with the GVG exploration techniques.

References

[1] 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.

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

1986.

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

[4] H. Choset and J.W. Burdick. Sensor Based Planning, Part I: The Generalized Voronoi Graph. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, 1995.

[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, I. Konuksven, and J.W. Burdick. Sensor Based Planning for a Planar Rod Robot. In *Proc. IEEE/SICE/RSJ Int. Conf. on Multisensor Fusion on Multisensor Fusion and Integration for Intelligent Systems*, Washington, DC, 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.B. Keller. *Lectures on Numerical Methods in Bifurcation Problems.* Tata Institute of Fundamental Research, Bombay, India, 1987.

[9] I Konuksven and H Choset. Mobile Robot Navigation: Implementing the GVG in the Presence of Sharp Corners. In *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, Grenoble, France, 1997.

[10] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, July 1994.

[11] V. Lumelsky and A. Stepanov. Path Planning Strategies for Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape. *Algorithmica*, 2:403–430, 1987.

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

[13] 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.

[14] N.S.V. Rao, N. Stolzfus, and S.S. Iyengar. A Retraction Method for Learned Navigation in Unknown Terrains for a Circular Robot. *IEEE Transactions on Robotics and Automation*, 7:699–707, October 1991.

[15] C.J. Taylor and D.J. Kriegman. Vision-Basied Motion Planning and Exploration Algorithms for Mobile Robots. In *Proc. of Workshop on the Algorithmic Foundations of Robotics*, San Francisco, CA, February 1994.