Mobile Robot Navigation: Issues in Implementating the Generalized Voronoi Graph in the Plane

Howie Choset* and Ilhan Konukseven** and Joel Burdick***

Abstract *This paper describes 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, and we describe how to incrementally construct this roadmap using only range information in an unknown environment. The GVG may then be used to guide future excursions into the explored environment. Experimental results validate the utility of this work.*

1 Introduction

Sensor based planning integrates sensor information into the planning process as opposed to *classical planning*, which requires that full knowledge of the world be available to the robot prior to the planning event. Sensor based planning is a necessary component for the realistic deployment of mobile robots. In this paper, we show how a mobile robot effects sensor based planning using raw sensor readings from sonar sensors.

We describe the experimental implementation of a sensor based planner that is based on the *Generalized Voronoi Graph* (GVG) whose definitions and properties are found in [4], [5] and [6] (and are reviewed in Section 5). The computational aspects required to implement the GVG-based planning scheme was described in [6] (and is reviewed in Section 6) , where it was assumed that only information within line of sight of the robot can be detected. We term this style of planning *Incremental Path Planning*. For the incremental construction of the GVG [6], the robot need only know the distance and direction to the m closest objects in m dimensions (in the planar case, $m = 2$). The implementation of the incremental technique using realistic sensors is described in this work. Experiments show that raw sensor readings from realistic sensors can be used to implement the GVG-based motion planning scheme without performing complicated obstacle segmentation.

2 Relation to Previous Work

There have been many works in sensor based planning in two dimensions. Some are based on heuristic algorithms. That is, they work very well under a variety of conditions, but these methods do not have any proofs of correctness that guarantee a path can be found. Moreover, there does not exist well established thresholds for when these heuristic algorithms fail. One class of heuristic algorithms is a behavioral based approach in which the robot is armed with simple behaviors such as following a wall [2]. A hierarchy of cooperating behaviors forms more complicated behaviors such as exploration. An extension of this type of approach is called sequencing [7]. Since there are strong experimental results indicating the utility of these approaches (such as [7]), some of these algorithms may provide a future basis for provably correct sensor based planners.

Another type of heuristic approach involves discretizing the planar world into pixels of some resolution. Typically, this is used when approximating errors in sonar sensing readings, where each pixel is a assigned a value indicating the likelihood that it overlaps an obstacle [1]. This method lends itself very nicely to implementation with real sensors, but discretizing the world may require a large amount of computer memory and may lead to an inaccurate representation of the world.

There are many non-heuristic algorithms for which provably correct solutions exist in the plane (see [11] for an overview). For example, Lumelsky's "bug" algorithm [9] 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 [11]) 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.

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, a one-dimensional subset of a robot's free space which captures all of its important topological properties. A roadmap (sometimes called a retract-like structure) 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.

An example of a complete roadmap scheme is Canny

*Carnegie Mellon University **CMU, METU (NATO Science Fellowship Program by TUBITAK) ***Caltech

and Lin's Opportunistic Path Planner [3]. Rimon adapted this motion planning scheme for sensor based use [12]. Unfortunately, connectivity of the roadmap in [12] cannot be guaranteed without active perception. Furthermore, from a practical point of view, there are two detractions to Rimon's method. First, to construct the roadmap, the robot must contain "interesting critical point" and "interesting saddle point" sensors, whose implementation is not well described. Second, a robust and detailed procedure for constructing the roadmap fragments from sensor data is not presented.

Finally, 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 [13]. 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.

3 Contributions

The *generalized Voronoi graph* (GVG) is a new type of roadmap-like structure that is a generalization of the *generalized Voronoi diagram* (GVD) [10] into higher dimensions. Recall that the GVD is the $(m-1)$-dimensional locus of points equidistant to two obstacles in m-dimensions, whereas the GVG is the one-dimensional locus of points equidistant to m obstacles in m dimensions.

Since the GVG is one-dimensional in an m-dimensional space, it is a concise representation of its workspace or multi-dimensional configuration space and thus allows for motion planning of robots with many degrees of articulation. The GVG is also concise because it bypasses the need for the robot to store a detailed pixel representation of the environment.

An important feature of the GVG is its incremental construction procedure that relies on line of sight information. This procedure is guaranteed to be complete, i.e., it has been rigorously shown that given only line of sight distance data, the incremental construction procedure will generate the GVG, thereby creating a representation that captures every detail of the environment. This incremental construction procedure does not rely on the existence of landmarks, nor does it need to perform any obstacle segmentation while constructing the GVG. Furthermore, this procedure does not use any abstract sensors, but instead uses raw sensor readings to construct its roadmap.

4 Distance Function

Model the robot as 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 $C_1, \ldots, C_n$. Non-convex obstacles are modeled as the union of convex shapes. The GVG and its properties are based on the following workspace distance function definitions:

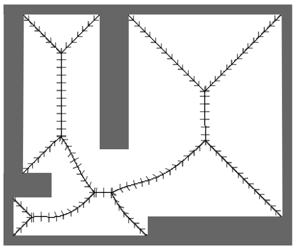

Fig. 1. The ticked line segments are the planar GVG for the bounded environment. The ticks point at the nearest point on an obstacle, and are thus the negated gradients.

$$d_i(x) = \min_{c_0 \in C_i} \|x - c_0\| \quad \text{and} \quad \nabla d_i(x) = \frac{x - c_0}{\|x - c_0\|}, \tag{1}$$

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 c_0 to x, where c_0 is the nearest point to x in C_i.

Typically, the environment is populated with multiple obstacles, and thus we define a multi-object distance function as $D(x) = \min_i d_i(x)$. *An important characteristic of $d_i(x)$, $\nabla d_i(x)$, and $D(x)$ is that they can be computed from sensor data.*

5 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*,

$$\mathcal{F}_{ij} = \{x \in \mathbb{R}^m : 0 \le d_i(x) = d_j(x) \le d_h(x) \quad \forall h \ne i, j \text{ and } \nabla d_i(x) \ne \nabla d_j(x)\}. \tag{2}$$

A two-equidistant face has co-dimension one in the ambient space, and thus in the plane, a two-equidistant face is one dimensional [5].

The Pre-image Theorem asserts that the union of the two-equidistant faces, i.e., the GVD, is $(m-1)$-dimensional [5]. 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.)[5]

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.

In [5], it was shown that the GVG possesses the properties of accessibility and departability. Nevertheless, connectivity is only guaranteed in planar case. In higher dimensions, higher order generalized Voronoi graphs must be constructed to connect the GVG components [5]. The results of this paper generalize to the higher order generalized Voronoi graphs, but for the purposes of explanation, we focus discussion in this paper to the GVG (which applies to all mobile robots).

6 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. As described in [5], [10], gradient ascent applied to the distance function to the nearest object can be used to trace a path from any point in the free space to the planar GVG. See Fig. 2.

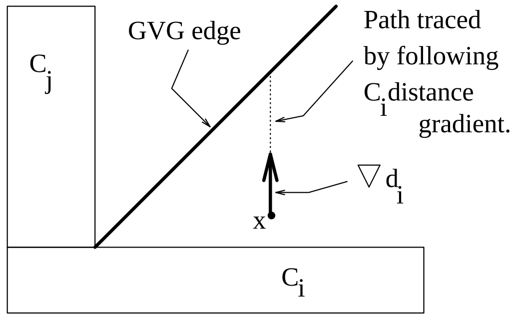

Fig. 2. Accessibility is achieved by following the gradient of the distance function to the nearest obstacle.

Traceability. In an incremental context, the property of connectivity is interpreted as *traceability.* More specifically, traceability implies that using only local data, the robot can: (1) "trace" the GVG edges; (2) determine when to terminate the edge tracing process, and (3) determine when to start new edge tracing procedures.

The GVG incremental approach to edge construction borrows ideas from numerical continuation methods [8]. Continuation methods trace the roots of the expression $G(y, \lambda) = 0$ as the parameter λ is varied. For the case of the GVG, the tracing function $G\colon \mathbb{R}^{m-1} \times \mathbb{R} \to \mathbb{R}^{m-1}$ is

$$G(y, \lambda) = \begin{bmatrix} (d_1 - d_2)(y, \lambda) \\ (d_1 - d_3)(y, \lambda) \\ \vdots \\ (d_1 - d_m)(y, \lambda) \end{bmatrix} \tag{3}$$

The explicit edge construction procedure has two steps: a predictor step and a corrector step. The predictor step moves the robot for a small distance along the tangent of the GVG. This tangent is the vector orthogonal to the m closest points in the m closest obstacles [6]; this can be readily computed with line of sight information. Typically, the prediction step takes the robot off of a GVG edge, so a correction procedure is required to bring the robot back to the GVG. If step size along the tangent is "small," then the graph intersects a "correcting plane" (Figure 3), which is a plane orthogonal to the tangent. The correction step finds the location where the GVG intersects the correcting plane (Figure 3) and is achieved via a iterative Newton's Method. If y^k and λ^k are the kth estimates of y and λ, the $k+1$st iteration is defined as

$$y^{k+1} = y^k - \left(\nabla_y G\right)^{-1} G(y^k, \lambda^k) \tag{4}$$

where $\nabla_y G$ is evaluated at (y^k, λ^k). It is shown in [6] that $\nabla_y G$ is invertible and thus Equation (4) is well posed. Practically speaking, this result states that the numerical procedure defined by Equation (4) will be robust for reasonable errors in robot position, sensor errors, and numerical round off.

Terminating Conditions. The explicit terminating conditions for edge tracing are described in [6], 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

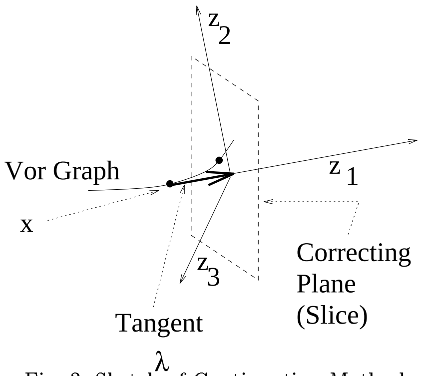

Fig. 3. Sketch of Continuation Method

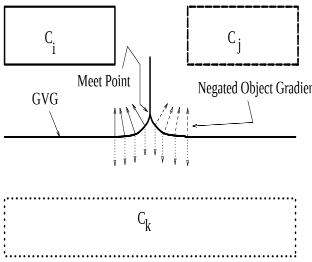

Fig. 4. Meet Point Detection.

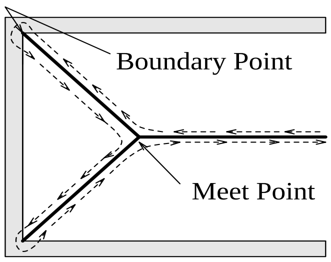

Fig. 5. Boundary Point. The arrows delineate the path a robot would follow while construction the GVG.

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.

Recall that at a meet point, the robot is equidistant to $m+1$ objects. Furthermore, at a meet point, $m+1$ GVG edges join (three in the planar case), and thus after the robot reaches a meet point, m new tangents, each corresponding to a new GVG edge, need be computed. Each new tangent is determined from an m-wise combination of the $m+1$ obstacles that define the meet point [6].

At a boundary point, the robot simply back tracks to a previous meet point that has unexplored GVG edges associated with it. See Figure 5.

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." Once the robot has accessed a point on the GVG, it begins tracing an edge. If the robot encounters a meet point, it marks off the direction from where it came as explored, and then explores one of the other m edges that emanate from the meet point. It also marks off that direction as being explored. If the robot reaches another unvisited meet point, the above procedure is recursively repeated. When the robot hits a boundary point, it simply turns around and retraces its path to some previous meet point with unexplored directions. The robot terminates exploration of the GVG fragment when there are no more unexplored directions associated with any meet point. 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.

7 Sensor Based Implementation

Incremental construction of the GVG is based on the distance function, d_i, the distance to the nearest point on object C_i. Sensors provide the distance to the nearest point from the sensor, without knowing from which obstacle the nearest point came. Therefore, the incremental construction of the GVG described in Section 6 and in [6] has to be adapted for sensor based implementation on actual robots.

7.1 The Robot

Experiments were performed on Nomadic Technologies mobile robot base, which is a circular platform that has a ring of sixteen sonar sensors radially pointing outward. (See Fig. 16) Dead reckoning for both systems is accomplished by integrating the number of encoder counts on robot's wheels, for translation, and base, for rotation. This procedure does not take into consideration slippage of the robot's wheels.

7.2 Sensor Model

This mobile robot uses ultrasonic sensors to infer environmental information. 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. For this, we develop a simple sonar sensor model that is compatible with the incremental construction procedure of the GVG. We assume 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). Finally, the distance gradient associated with a particular sensor is assumed to be a unit vector pointing along the sensor measurement axis away from the robot. Since the closest point may occur anywhere within the sensor beam pattern, and it is assumed that the distance gradient points along the beam pattern centerline; this can induce errors in the direction of the gradient. However, the accumulated error decreases with increasing number

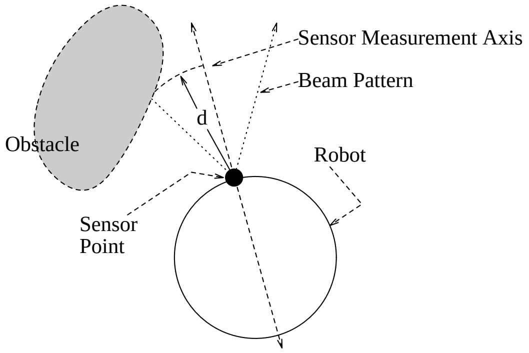

Fig. 6. Simplified Distance Measurement Sensor Model

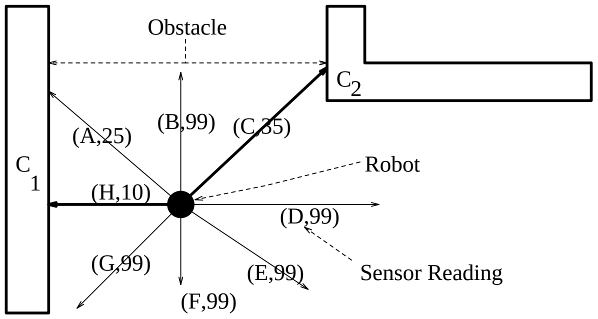

Fig. 7. Sensor Readings

of sensors.

7.3 Distance Function

The robot must be able to convert raw sensor readings into distance function readings while ideally avoiding a costly obstacle segmentation procedure in order to effectively perform the incremental construction procedure.

It is not sufficient to sample the two smallest sensor readings to determine the distance to the two closest obstacles because multiple sensors may detect the same obstacle. The minimum distance to each of the obstacles can be approximated from the local minima in the circular array sensor readings. An example is depicted in Fig. 7 where 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 C is associated with the second closest obstacles because its value is the second smallest local minimum in the sensor array. Note, Sensor A should not be associated with the second closest obstacle because it detects the same object as Sensor H. Nevertheless, the value of Sensor A is not a local minima and thus should not be considered. The distance gradients are the unit vectors pointing along the respective sensor centerlines. This method bypasses a costly obstacle segmentation procedure and enables construction of the GVG directly from range sensor data.

The above claim that the distance to obstacles is the local minima of the sensor array is proven in the Appendix.

7.4 Incremental Accessibility

Inremental accessibility is simply gradient ascent applied to the distance to the nearest obstacle. Since the nearest obstacle is associated with the sensor reading with the smallest value, simply moving in a direction opposite to which the sensor with the smallest value is facing is gradient ascent of the distance to the nearest obstacle.

7.5 Incremental Traceability

Once the robot has found a GVG edge, it must incrementally trace the edge and store an internal representation of it. The distance measurement method, described in Section 7.3 determines the distance and direction of the two closest obstacles.

Since there are a finite number of sensors, the robot relies upon a lookup table to determine the tangent space of the GVG. The orientation of the tangent spaces corresponding to each closest sensor pair is stored in the lookup table, indexed by the two closest sensor locations. Since there are only sixteen sensors, a lookup table is a good trade-off between speed and memory storage. The robot orients itself into the tangent space of the GVG edge and then takes a fixed step along the GVG edge's tangent direction.

In the current implementation, the correction step is an adaptation of the procedure described by Equation 4, which prescribes the direction and magnitude of the robot's correction course. Upon completion of the prediction step, the robot makes a ninety degree turn which points the robot in the direction prescribed by Equation 4. Instead of moving by the amount specified in Equation 4, the robot rolls in a straight line until its two smallest sensor readings are equal or within a threshold distance of eachother. After reaching the GVG, the robot repeats the step-correct procedure until it encounters a meet point or a boundary point. The robot stores a GVG edge as a doubly-linked list of points.

It is worth noting that the sensor associated with the nearest reading to an obstacle will change in a "continuous" fashion as the robot explores a GVG edge. Since we have very few sensors and thus a very low resolution, a change of the location of a local minima by one sensor location constitutes a "continuous" change.

7.6 Meet Point Detection

Recall, at a meet point, $\mathcal{F}_{ijk}$, the robot terminates the edge tracing procedure of a GVG edge $\mathcal{F}_{ij}$, and begins tracing edge $\mathcal{F}_{ik}$. 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

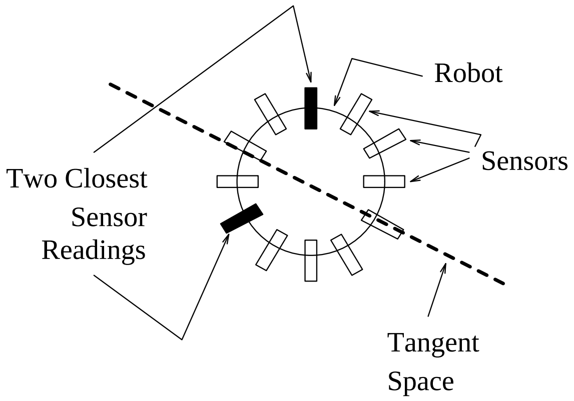

Fig. 8. The tangent space associated is defined by the closest sensor readings.

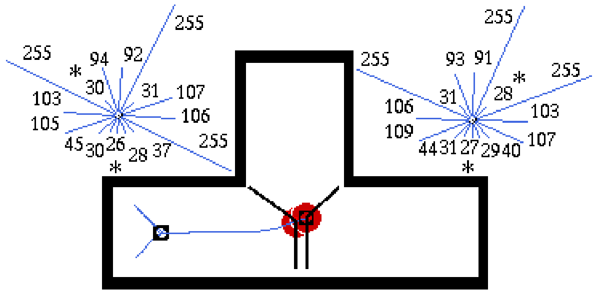

Fig. 9. Naive Meet Point Detection

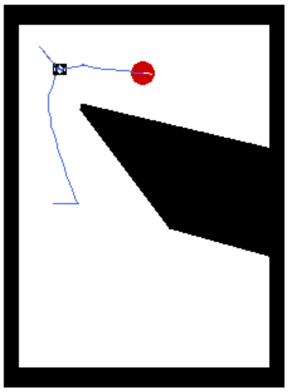

Fig. 10. Working.

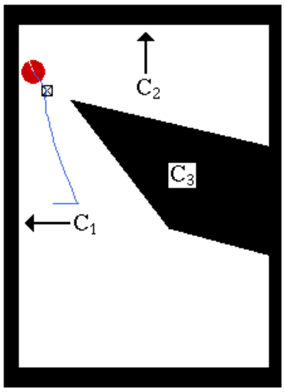

Fig. 11. Not Working

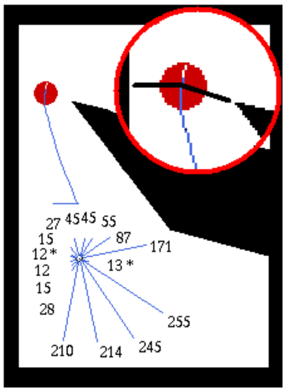

Fig. 12. Robot has traced out GVG edge fragment. The lower left portion displays the sonar sensor values and the upper-right portion contains a closeup of the robot and its two local minima.

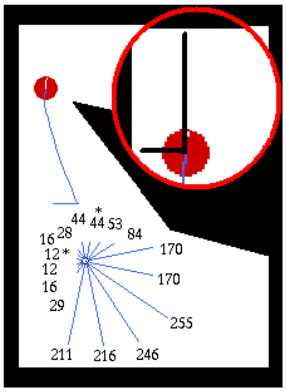

Fig. 13. Robot has two local mimina whose values are not close to each other.

with one of the two closest obstacles, then the robot has passed by a meet point. Just like a continuous change, an "abrupt" change is a function of the resolution of the sensor system and robot's step size. 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. 9). Term this procedure the *naive meet point* detection scheme.

The naive meet point detection scheme works well in environments whose objects can be bounded by polygons with non-acute angles. For example, in Fig. 10, the robot detects the meet point associated with the left-most obstacle (C_1), top-most obstacle (C_2), and central-protruding obstacle (C_3). Note that C_3 has blunt (not sharp) protrusion which forms the meet points.

Unfortunately, when C_3 has a sharp angle (Fig. 11), a false meet point is detected because of the limited resolution and the reflective properties of the sonar sensors. Figures 12 and 13 delineate why a false meet point is detected in Fig. 11. In Fig. 12, the two smallest local minima are associated with C_1 and C_3, and thus the robot infers C_1 and C_3 are the two closest obstacles. In Fig. 13, the robot takes one more step, and is not able to infer the existence of C_3 from its sensor readings because C_3 has sharp angle (that is, C_3 is now "invisible" to the robot). The perceived absense of C_3 results in an abrupt change in one of the two smallest local minima causing a false meet point to be detected. (now, the two smallest local minima are associated with C_1 and C_2).

Note that in Fig. 13 the values of the two smallest local minima are not the same, nor close to each other. This indicates that the two smallest local minima do not define a GVG edge. In other words, the robot should not have stop tracing $\mathcal{F}_{13}$ and start tracing $\mathcal{F}_{12}$, i.e., it should not infer that there is a meet point Fig. 11. Therefore, the naive meet point detection scheme is just a necessary and not sufficient condition.

The correct meet point detection scheme has two parts. First, the robot looks for an abrupt change in one of the locations of the two smallest local minima. When this happens, the robot checks the values of the two smallest local minima. If they are not the same nor close to each other, the robot assumes no meet point is nearby and continues tracing the current GVG edge. The robot uses the smallest local minima and the local minima associated with the "invisible" obstacle from the previous step to continue tracing the GVG edge. Con-

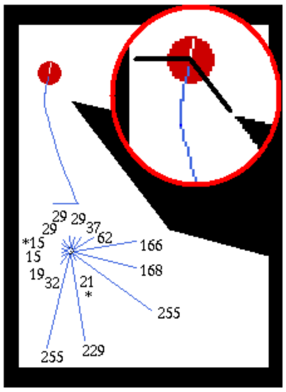

Fig. 14. Robot uses location of previous local minima.

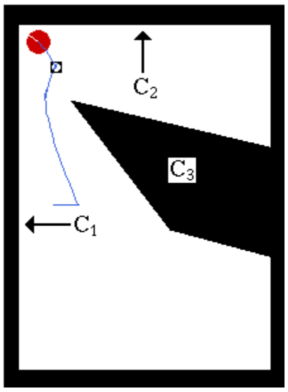

Fig. 15. Robot detects the correct meet point.

Fig. 16. Nomadic Robot

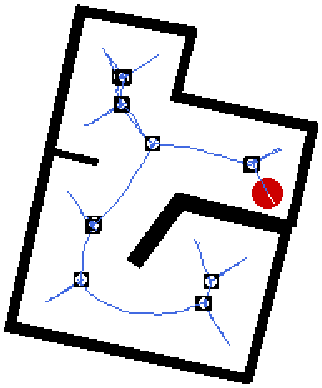

Fig. 17. Experiment.

tinuity of the distance function asserts that procedure generates an edge that is a good approximation of the actual GVG edge. Figs. 14 and 15 validate the effectiveness of the new meet point detection scheme.

The meet points are stored as a double linked list of meet point structures. Each meet point structure has three pointers, each pointing to an GVG edge list emanating from the meet point. The meet point structure also contains three flags, each indicating in which direction (forward or backward) its corresponding the GVG edge list must be traversed to get the robot to an adjacent meet point in the GVG. When the robot enters a corner (i.e., reaches a boundary point), it simply turns around and retraces its steps to a previous meet point with unexplored GVG edges. This produces a more concise and precise representation of the environment in contrast to the pixel methods.

8 Experimental Results

The results of one experiment can be found in Fig. 17. The robot was put into an unknown environment bounded by styrofoam, cardboard, and wooden walls. The GVG for this environment is depected as line segments in Fig.17. The squares denote the location of meet points in the environment.

9 Conclusion

This paper described the implementation of a general sensor based planning strategy, based on the generalized Voronoi graph, for the special case of a planar environment. We showed that using the algorithm of [5], [6], a robot equipped with only a ring of sonar sensors can explore an a priori unknown environment and produce a one-dimensional representation (the GVG) of that static environment. With this one-dimensional representation, the robot can plan future excursions into the environment.

An important feature of the GVG is that it produces a concise representation of the world. The GVG edge is stored as a list of points and the GVG vertices are stored as list of pointers, each pointing to a GVG edge with which the GVG vertex is associated. This representation is a significant savings in storage when compared to other world models such as a discrete pixel representation.

A key feature of the GVG is its incremental construction procedure which relies on line of sight information. This procedure is guaranteed to be complete; that is, it will produce a representation the captures the full geometry of the environment. Another advantage of this approach is that while tracing out the GVG edges, there is no need to fully identify each obstacle (i.e., perform complicated obstacle segmentation). Therefore, there is no need to store a representation of each obstacle.

In this work, we did not consider all of the implications of sensor noise, and limited sensor range on our algorithm. Also, we assume that the robot has a relatively accurate dead reckoning system. The next step in our research is to consider the issues of sensor noise, limited sensor range, and dead reckoning for the planar robot. The strength this work is that it generalizes into higher dimensions, and thus to a more general class of robots.

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 and M.C. Lin. An Opportunistic Global Path Planner. *Algorithmica*, 10:102–120, 1993.

[4] H. Choset and J.W. Burdick. Sensor Based Planning and Nonsmooth Analysis. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3034–3041, San Diego, CA, 1994.

[5] 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.

[6] 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.

[7] E. Gat and G. Dorais. Robot Navigation by Conditional Sequencing. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1293–1299, San Diego, CA, May 1994.

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

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

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

[11] 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.

[12] E. Rimon and J.F. Canny. Construction of C-space Roadmaps Using Local Sensory Data — What Should the Sensors Look For? In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 117–124, San Diego, CA, 1994.

[13] 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.

Appendix

A Raw Distance Function

The following definitions and lemma validate the claim that the distance to obstacles is associated with the local minima in the sensor array. The *raw distance function*, defined as

$$\rho(x, s) = \|x + \lambda\bar{s}\| \quad \text{where } \lambda = \min_{\Lambda\in[0,\infty)} D(x + \Lambda\bar{s}) = 0, \tag{5}$$

provides the distance to all the points on the boundary of the environment that are *within line of sight* of the robot. See Fig. 18 for an example of the raw distance function.

A key feature of the raw distance function is that it can be readily approximated by many realistic sensor configurations. The sensor measurement provides an approximate value of the distance function $\rho(x, s)$, and the direction to which the sensor is facing corresponds to the direction of measurement ($s \in S^{m-1}$). See Fig. ??. In Fig. 18, it can be seen that the raw distance function is not continuous.

Definition A.1 (Cone of Continuity) The *cone of continuity*, $\varrho(x)$ at a point x, is the closure of the set of directions for which $\rho(x, s)$ is continuous with respect to $s \in S^{m-1}$.

Note that $\varrho(x) \subset S^{m-1}$. Typically, at a point x there is more than one cone of continuity, so we attach an index, α, to it, and denote ϱ^α as the αth cones of continuity at x. Note that $S^{m-1} = \bigcup_\alpha \varrho^\alpha(x)$.

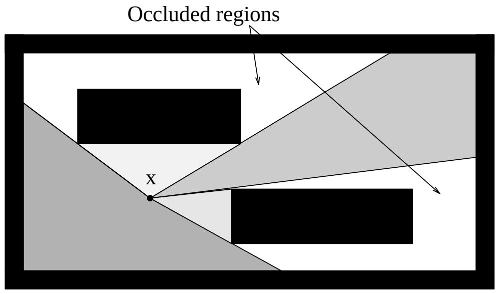

Fig. 18. The differently shaded regions each correspond to a cone of continuity. The unshaded region corresponds to the points that are not within line of sight of x.

In the planar case the cone of continuity is simply a closed interval. See Figure 18.

Finally, let $\tilde{C}_i(x)$ be the set of points on an object C_i that are within line of sight of x, i.e.,

$$\tilde{C}_i(x) = \{c \in \partial C_i : \forall t \in [0, 1],\ x(1 - t) + ct \in \mathcal{FS}\}.$$

With this we can define a *visible distance function*,

$$d_i(x) = \begin{cases} \min_{c\in\tilde{C}_i(x)} \|x - c\|, & \text{if } c \in \text{int}(\tilde{C}_i(x)), \\ \infty, & \text{if } c \notin \text{int}(\tilde{C}_i(x)), \end{cases} \tag{6}$$

which, in actuality, is the distance function used in the GVG definitions.

Lemma A.2 *The visible distance (V-distance) to an object at a point x is a local minima of the raw distance function in the interior of a cone of continuity.*

Proof: The αth cone of continuity, $\varrho^\alpha(x)$, can be broken down into sub-regions, each associated with a particular obstacle. The set directions associated with only the boundary of C_i (i.e., points in $\text{int}(\tilde{C}_i(x))$) be denoted $\tilde{\varrho}_i^\alpha(x)$. Clearly, $\bigcup_{i\in I(x)} \tilde{\varrho}_i^\alpha(x) = \varrho^\alpha(x)$ where the index set $I(x)$ corresponds to each obstacle associated with the cone of continuity.

Since $c = x + \rho(x, s)s$,

$$\tilde{C}_i(x) = \{x + \rho(x, s)s \in \partial C_i : \forall s \in \varrho_i^\alpha(x)\}.$$

And since $\rho(x, s) = \|x-c\|$, $c = x+\rho(x, s)s = x+\|x-c\|s$ which implies that $s = \frac{x-c}{\|x-c\|}$. Therefore, there is a one to one correspondence between each $s \in \varrho_i^\alpha(x)$ and $c \in \tilde{C}_i(x)$. Hence,

$$\begin{aligned} d_i(x) &= \min_{c\in\text{int}(\tilde{C}_i(x))} \|x - c\| \\ &= \min_{x+\rho(x,s)s\in\text{int}(\tilde{C}_i(x))} \|x - x - \rho(x, s)s\| \\ &= \min_{x+\rho(x,s)s\in\text{int}(\tilde{C}_i(x))} \|\rho(x, s)s\| \\ &= \min_{x+\rho(x,s)s\in\text{int}(\tilde{C}_i(x))} \rho(x, s) \\ &= \min_{s\in\text{int}(\tilde{\varrho}_i^\alpha(x))} \rho(x, s). \end{aligned}$$

■