Sensor Based Planning: A Control Law for Generating the Generalized Voronoi Graph

Howie Choset* and Ilhan Konukseven** and Alfred Rizzi*

Abstract *We introduce a new control law for robots that explore unknown environments and configuration spaces. Unlike numerical continuation methods, which produce a jagged path because of the predictor-corrector approach, the control law, introduced in this paper, performs sensor based planning by directing the head of a robot to follow continuously a roadmap. Recall that a roadmap is a one-dimensional representation of a robot's environment. Once the robot exhaustively traces out the roadmap using this control law, it has in essence explored an environment. Experiments on a mobile robot validate the control law.*

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. We believe that the realistic deployment of robots into unknown environments requires sensor based planning. Similarly, 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.

The sensor based planning approach, introduced in this paper, makes use of a *roadmap,* a representation which captures all of the salient geometric features found in a robot's free space. A roadmap is typically a one-dimensional structure embedded in a robot's m-dimensional configuration space. We developed a new "control law" which directs the heading of a robot to incrementally trace out the roadmap, using only line of sight range data. Once a robot constructs a roadmap using this control law, it has in essence explored an environment.

This control law relies on distance information and thus is well suited to sensor based planners because many sensors provide distance information. This law also possesses the property of completeness, which means for the appropriate gains, the control law guarantees that the robot will explore the entire environment. Finally, the control law functions in multi-dimensional spaces which is useful for robots which have many degrees of freedom.

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

2 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]. A hierarchy of cooperating behaviors forms more complicated actions, such as exploration. An extension of this approach is called sequencing [10]. 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]. 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.

One complete motion planning algorithm that functions in higher dimensions, but requires full a priori knowledge of the world be available to the robot, is based on a *roadmap* (Canny, [3]), which is a collection of one-dimensional curves that capture the topological and geometric properties of a robot's environment. Roadmaps are analogous to highway systems and have the following properties: *accessibility*, *connectivity*, and *departability.* Using a roadmap, a planner can construct a path between any two points in a connected component of the robot's free space by first finding a collision free path onto the roadmap (accessibility), traversing the roadmap to the vicinity of the goal (connectivity), and then constructing a collision free path from a point on the roadmap to the goal (departability). An example of a complete roadmap scheme is Canny and Lin's Opportunistic Path Planner (OPP) [4]. Rimon made some initial steps towards adapting the OPP motion planning scheme for sensor based use [14].

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. 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 [13], an incremental approach to create a Voronoi Diagram-like structure, which is limited to the case of a plane, was introduced.

Previous sensor based planners cannot handle completeness in higher dimensions, so the challenge is to develop a roadmap in higher dimensions that can be constructed using line of sight information. The GVD is only a roadmap for point robots in planar environments. Consequently, the first step in the long-term research program produced the *generalized Voronoi graph* (GVG), which is a natural extension of the GVD into higher dimensions; it is the one-dimensional set of points in m dimensions equidistant to m obstacles. However, unlike the GVD, the GVG is not necessarily connected in dimensions greater than two, and thus, in general, is not a roadmap. Additional structures, termed *higher order generalized Voronoi graphs*, connect GVG components, and together with the GVG form the *the hierarchical generalized Voronoi graph* (HGVG) (Choset and Burdick, [6], [8]). The HGVG is well suited to motion planning in multi-dimensional spaces (such as configuration spaces) because a motion planner can perform a bulk of its path search on the one-dimensional HGVG.

The HGVG may have no clear advantage over other methods when full knowledge of the world is available, but its incremental construction procedure [7], [8] gives the HGVG its primary strength. This incremental construction procedure only requires line of sight information to configuration space obstacles. Additionally, 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. Since the incremental construction procedure relies solely on line of sight information in configuration space, the HGVG construction procedure is amenable to sensor based planning.

Unfortunately, the previous incremental construction procedure produces jagged paths for the robot to follow. Such paths take a long time to traverse as the robot spends significant amount of time making ninety degree turns. The control law, introduced in this paper, produces a smooth path by continuously controlling the heading of the robot, thus improving overall performance.

3 Related Work

The work presented in this paper is based on the GVG, which is described in [6], [7], [8], [9]. 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 $C_1, \ldots, C_n$. 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

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

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 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, $D(x) = \min_i d_i(x)$

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*,

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

A two-equidistant face has co-dimension one in the ambient space, and thus in the plane, a two-equidistant face is one dimensional [6].

The Pre-image Theorem asserts that the union of the two-equidistant faces, i.e., the GVD, is $(m-1)$-dimensional [6]. 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.)[6]

The *generalized Voronoi graph* (GVG) is the collection of m-equidistant faces and $m+1$-equidistant faces.

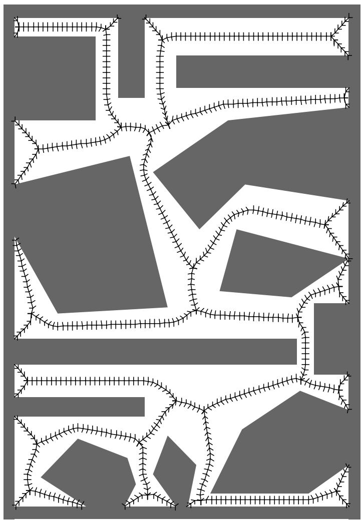

Fig. 1. The ticked line segments are the planar GVG for the bounded environment. The ticks point at the nearest point on an obstacle, and are thus the negated gradients.

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 [6], it was shown that the GVG possesses the property of accessibility, but connectivity and departability are only guaranteed in planar case. In higher dimensions, higher order generalized Voronoi graphs must be constructed to connect the GVG components [6]. The resulting connected network is termed the *higher order generalized Voronoi graph* (HGVG). For the purposes of explanation, we will describe the incremental construction procedure for the GVG first. However, it should be noted that the incremental construction of the GVG is sufficient for mobile robot exploration.

3.3 Incremental Construction

One of the key features of the GVG is that a robot can incrementally construct it using only line of sight information. See [7] for details. This section summarizes how the robot traces out a GVG edge. After the robot has arrived at a point on a GVG edge, it must incrementally trace the one-dimensional branches of the GVG. 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.

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 $G(y, \lambda) = 0$ 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." At a point $x \in \mathbb{R}^m$, $G(y, \lambda)$ has the form

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

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 (which is proven in [7]), 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.

The GVG edge tracing method relies upon two iterative stages: (1) a prediction step and (2) a correction procedure. In the prediction step, the robot takes 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 [7]. 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 $x = (y, \lambda)$ [7], 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

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

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.

The robot continues edge tracing until it detects a meet point or a boundary. At a meet point, the robot must determine the directions of the other GVG edges that emanate from the meet point. In the planar case,

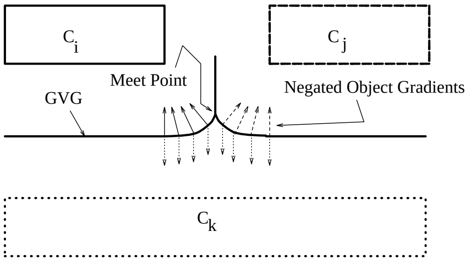

Fig. 2. Meet Point Detection

the meet point is triply equidistant to the nearest obstacles. But due to sensor noise and small inaccuracies in the curve tracing technique, it is unreasonable for the robot to detect exact triple equidistance. Fortunately, the robot can robustly infer the location of a meet point by looking for an abrupt change in the closest obstacle set [5], [7]. For example, in Fig. 2 as the robot travels from left to right, initially the two closest obstacles are C_i and C_k. As the robot passes by the meet point, the two closest obstacles become C_j and C_k. This change is detected by an abrupt change in the gradients to the nearby obstacles.

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 a meet point, the above branching process is repeated. Otherwise, when a robot reaches a boundary, it simply turns around and returns to a meet point with unexplored GVG edges. 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 Control Law

The numerical continuation methods produces 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 3.3 motivates the below described control law. See 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

$$\dot{x} = \alpha \, \mathrm{Null}(\nabla G(x)) + \beta (\nabla G(x))^{\dagger} G(x), \qquad (5)$$

where

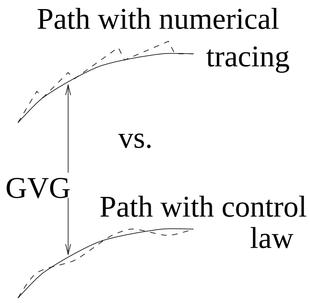

Fig. 3. Dotted lines represent path traced out by either the numerical continuation methods or the control law.

- 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

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

Let $\Gamma = \frac{1}{2} G^T G$ measure the distance a point x is away from the GVG. Therefore,

$$\begin{aligned} \dot{\Gamma} &= G^T \dot{G} \\ &= G^T \nabla G \dot{x} \\ &= G^T \nabla G (\alpha \, \mathrm{Null}(\nabla G) + \beta \nabla G^{\dagger} G) \\ &= \beta G^T \nabla G \nabla G^{\dagger} G \\ &= \beta G^T \nabla G \nabla G^T (\nabla G \nabla G^T)^{-1} G \\ &= \beta G^T G. \end{aligned}$$

So, the control law produces a direction that converges onto the GVG, if $\beta < 0$ and if $\nabla G \nabla G^T$ is invertible in a neighborhood of the GVG, which is guaranteed by the following Lemma.

LEMMA 4.1 *The matrix $\nabla G \nabla G^T$ is invertible in a neighborhood of the GVG.*

This proof relies on the property that for any matrix A,

$$(\mathrm{Null}(A^T))^{\perp} = \mathrm{Im}(A). \qquad (6)$$

Proof: Assume there exists a $z \in \mathrm{Im}(\nabla G)$ such that $(\nabla G \nabla G^T) z = 0$. This means that $\nabla G^T z = 0$ or $\nabla G^T z \in \mathrm{Null}(\nabla G)$. If $\nabla G^T z = 0$, then $z \in \mathrm{Null}(\nabla G^T)$ which is $(\mathrm{Im}(\nabla G))^{\perp}$, and thus $z \notin \mathrm{Im}(\nabla G)$. This is a contradiction.

Fig. 4. Nomadic Robot

Recall that ∇G is full rank on Y, the normal plane. In fact, $Y = \mathrm{Im}(\nabla G)$. If $\nabla G^T z \in \mathrm{Null}(\nabla G)$, then $\mathrm{Im}(\nabla G^T) \subset \mathrm{Null}(\nabla G)$ which contradicts Equation(6). Therefore, there cannot exist a $z \in \mathrm{Im}(\nabla G)$ for which $(\nabla G \nabla G^T)z = 0$. That is, $(\nabla G \nabla G^T)$ is non-singular at a point on the GVG.

Since the set invertible matrices is an open set, there exists an open neighborhood around the GVG for which $(\nabla G \nabla G^T)$ is invertible. ■

Note the following.

- In a neighborhood of the GVG, $\mathrm{Null}(\nabla G)$ is locally parallel to the GVG by continuity of the distance function.
- The α determines how quickly the robot moves along the GVG and the β represents how aggressively the robot moves back to the GVG.
- The control law relies on first order information, and thus nothing need be assumed about curvature. However, in the discrete implementation of this system, the ratio of α to β must be considered to ensure tracking around tight corners.

5 Implementation

The control law was implemented on a Nomadic Technologies mobile robot base, which is a circular platform that has a ring of sixteen sonar sensors radially pointing outward. (See Fig. 4) The distance to nearby obstacles are the values of the local minima of the sonar array [9]. The sonar sensor that has the smallest local minima is the distance to the closest obstacle, C_1, and the sonar sensor with the second smallest local minima is the distance to the second closest obstacle, C_2.

In the planar case, the G matrix and its Jacobian are

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

$$\nabla G(x) = \begin{bmatrix} (\nabla d_1(x) - \nabla d_2(x))^T \end{bmatrix}. \quad (7)$$

Therefore, the robot takes the following step

$$\dot{x} = \alpha(\nabla d_1(x) - \nabla d_2(x))^{\perp} + \beta(\nabla d_1(x) - \nabla d_2(x))^{\dagger}(d_1(x) - d_2(x)). \quad (8)$$

The direction $\dot{x}$ is a vector in the plane which can be parameterized by a single parameter, the angle the vector makes with the horizontal. The above law will be used to determine this heading.

On the GVG, $(\nabla d_1(x) - \nabla d_2(x))^{\perp}$ is the tangent to the GVG, and in a neighborhood of the GVG, $(\nabla d_1(x) - \nabla d_2(x))^{\perp}$ is locally parallel to the tangent of the GVG. This vector in the plane can be parameterized by the angle it makes with the horizontal. Instead of performing the explicit vector subtraction $(\nabla d_1(x) - \nabla d_2(x))$, taking its orthogonal compliment, and determining the compliment's angle with respect to the horizontal, a lookup table is used to determine the tangent direction. This lookup table is indexed by the two smallest local minima sensor id's. Now, the tangent computation requires just a lookup operation.

Let Y be the vector orthogonal to the tangent direction, $(\nabla d_1(x) - \nabla d_2(x))^{\perp}$. Together, these two vectors form a coordinate system. In this coordinate system, the first column of $\nabla G(x)$ is zero when x is on the GVG [7]. Therefore,

$$[(\nabla d_1(x) - \nabla d_2(x))^T] = \begin{bmatrix} 0 & p \end{bmatrix},$$

and the pseudo inversion of $\nabla G(x)$ is

$$\begin{aligned} [(\nabla d_1(x) - \nabla d_2(x))^T]^{\dagger} &= \begin{bmatrix} 0 & p \end{bmatrix}^{\dagger} \\ &= \begin{bmatrix} 0 \\ p \end{bmatrix} \left(\begin{bmatrix} 0 & p \end{bmatrix} \begin{bmatrix} 0 \\ p \end{bmatrix} \right)^{-1} \\ &= \begin{bmatrix} 0 \\ p \end{bmatrix} [p^2]^{-1} \\ &= \begin{bmatrix} 0 \\ \frac{1}{p} \end{bmatrix} \end{aligned} \quad (9)$$

It is shown in [7] that $\|\pi_Y \nabla d_i(x)\| = \|\pi_Y \nabla d_j(x)\|$ if x is on the GVG (and C_i and C_j are two of the m closest obstacles). By continuity of the distance function, $\|\pi_Y \nabla d_i(x)\| \simeq \|\pi_Y \nabla d_j(x)\|$ in a neighborhood of the GVG. From this, we can conclude that in the planar case, $\pi_Y \nabla d_i(x) \simeq -\pi_Y \nabla d_j(x)$ Therefore,

$$\begin{aligned} p &= \|\nabla d_i(x) - \nabla d_j(x)\| \\ &= \|\pi_Y(\nabla d_i(x) - \nabla d_j(x))\| \\ &= \|\pi_Y \nabla d_i(x) - \pi_Y \nabla d_j(x)\| \\ &= \|\pi_Y \nabla d_i(x) + \pi_Y \nabla d_i(x)\| \\ &= 2\|\pi_Y \nabla d_i(x)\|. \end{aligned}$$

Therefore, a look up table indexed by the sensor ID whose sensor reading is the smallest, can be used to determine $((\nabla d_1(x) - \nabla d_2(x))^T)^{\dagger}(d_1(x) - d_2(x))$. The controlled heading of the robot determined by two lookup operations, one sum, one difference, and a multiplication. With this implementation, $\alpha = \frac{1}{\|\nabla G(x)\|}$ and $\beta = -1$.

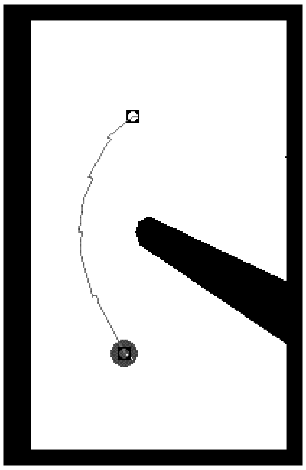

Fig. 5. Continuation Methods produce a jagged path.

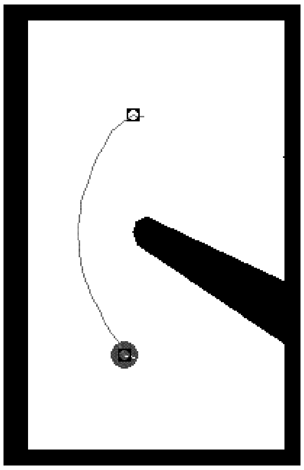

Fig. 6. Control Law produces a smooth path.

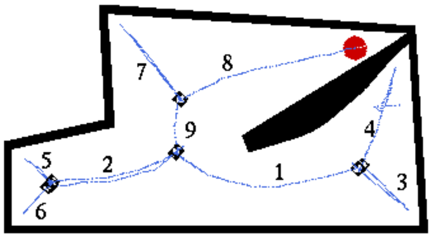

Fig. 7. Path traced by control law.

6 Experiment

The control law provides a smooth path that takes less time for the robot to follow. Figures 5 and 6 contain the same environment, but in Figure 5 the robot uses the numerical continuation methods to generate the GVG, whereas in Figure 6, the robot uses the control law. Note how the GVG path in Figure 6 is smoother than the one in Figure 5. Furthermore, and perhaps more importantly, it took the robot 78 seconds to traverse the path generated by the continuation methods and the robot required 48 seconds to navigate the path generated by the control law. This save nearly 40%.

Figure 7 contains the traced path of the robot while it generated a GVG using the control law. The control law improves edge tracing when generating edges 1 and 2 because of their relatively high curvature. The robot required 91 seconds to generate edge 1 with the control law whereas it require 126 second to generate the edge with the numerical continuation methods. Similarly, the robot required 59 seconds to generate edge 2 with the control law whereas it require 120 second to generate the edge with continuation methods.

7 Conclusion

This paper presents a new control law for generating roadmaps. The control law was developed to offset the draw back of continuation methods that produces jagged paths. A robot takes less time to follow and generate a smoother path, improving overall performance.

Although simulations and experiments were performed on the GVG, the control law, introduced in the paper, applies to the higher order GVG's in the HGVG and to other roadmaps, such as the OPP. By simply adjusting the G matrix to trace different roadmap structures, the control law can generate these different roadmap edges.

Finally, note that the control law is a local result. In practice, if the robot strays too far from the GVG (as a result of sensor noise), then it must use the numerical continuation methods to get back on the GVG.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[14] 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.