Sensor Based Navigation for car-like mobile robots using Generalized Voronoi Graph

K.Nagatani   Y.Iwai   Y.Tanaka

Graduate School of Natural Science and Technology, Okayama University
3-1-1 Tsushima-naka
Okayama 700-8530, JAPAN
Keiji@sys.okayama-u.ac.jp

Abstract

Our research objective is to realize sensor based navigation by car-like mobile robots. Generalized Voronoi Graph (GVG) [1] has an advantage to describe mobile robot's path for sensor based navigation from the point of view of completeness and safety. However, it is impossible to apply the path to car-like mobile robot directly, because limitation of minimum turning radius prevents following non-smooth GVG.

To solve the problem, we propose local smooth path planning algorithm for car-like mobile robots. Basically, an initial path is generated by conventional path planning algorithm using GVG theory, and it is deformed smoothly to enable car-like robots' following by maximizing an evaluation function proposed in this paper. The key topics are (A) definition of our evaluation function and (B) how to modify the GVG.

In this paper, we introduce a local smooth path planning algorithm based on GVG, and explain a detail of the evaluation function. Simulation results support validity of the algorithm.

1 Introduction

Sensor based navigation enables a robot to explore an unknown environment and build a map of it, using its sensor information. This is one of the traditional research subject for mobile robots' intelligence and many algorithms have been proposed.

We focus on sensor based navigation using Generalized Voronoi Graph (GVG). The GVG structure has an advantage to express the path of mobile robot for sensor based navigation from the point of view of safety and completeness. Safety means that every point on GVG is the farthest from the closest obstacles, and completeness means that the GVG is a unique structure in static environment.

To apply the algorithm to real environment, we adopt steering type car-like mobile robot as research platform. It has advantages that it is easy to build by remodelling a conventional vehicle, and usually load capacity is very large. However, it has a disadvantage for steering ability, and we can not apply non-smooth GVG to car-like mobile robots' path directly.

To solve the problem, we made local smooth path planner using an evaluation function to follow GVG as much as possible.

In this paper, we introduce (A) a detail of the evaluation function and (B) how to generate smooth paths. Simulation results support a validity of the algorithm.

2 Prior Works

This work contains two research areas: sensor based planning and nonholonomic robots' navigation. Although both of these fields are vast, included works in here are the works that have influenced the authors' thinking.

2.1 Sensor Based Planning

Sensor based navigation enables a robot to explore an unknown environment, with an assumption of simple and weak sensors. One of the famous and robust algorithm is named "Bug algorithm" proposed by Prof. Lumelsky [2]. In this algorithm, only single touch sensor enables robust exploration, and it is mathematically complete. However, in actual exploration case, we believe that the robot should have more powerful sensors, and it can generate better path than Bug Algorithms.

The other approach is an incremental procedure to construct GVG [1]. It requires only line of sight information (that is obtainable from robot's range sensors) and the procedure has no restrictions on the type of obstacles. The algorithm has been successfully implemented on an actual mobile robot with a ring of sonar sensors [3]. Unfortunately, the algorithm can not be applied to car-like mobile robot because GVG is not smooth.

Our sensor based navigation is mainly based on incremental approach of constructing GVG. However, the robot does not follow exact GVG, instead the robot deforms GVG locally to follow it.

2.2 Nonholonomic Robots' Navigation

Research area of nonholonomic robots' navigation is also vast. Many heuristics algorithms were applied to actual robots in real environment. One experimental approach is to use a state transition of trailer type robot for corner curve, and the robot's path is generated by combination of state pattern [4]. It is successfully implemented on a real robot. However, target environment is restricted.

To consider non-heuristic algorithm for any shape environment, one of simple algorithm for car-like robots is to use a set of line segments and arc of circles for its path [5]. It is guaranteed that the robot follows the path when all circle radiuses are bigger than minimum steering radius. However, optimality (or safety) is not discussed in this case. For optimization, the shortest path for car-like robots in manifold is proposed in [6], and a nonholonomic distance is discussed in [7].

To consider shape of mobile robots, configuration space (C-space) is very useful for motion planning [8]. However, it defines only robot's configuration, and steering restriction for car-like robots in configuration space is another problem.

In our approach, traceability of the path is guaranteed by checking a maximum curvature for each path. To check a collision, we construct a C-space to check an intersection of a point robot to configuration obstacles. Also, optimality and safety are discussed using our evaluation function.

3 Exploration Procedure

To realize sensor based navigation based on GVG for car-like robots, we designed following procedure.

1. Acquiring local environment information by range sensors
2. Calculating local GVG
3. Constructing Configuration Space (C-space)
4. Generating candidates of smooth path
5. Applying an evaluation function to each path in C-space, and choosing the best one
6. Executing the planned path in local area
7. Repeating from 1 to 6, until whole GVG structure is completed.

Basically, the local GVG and C-space are constructed by conventional algorithm. Therefore, we focus on the evaluation function and how to generate candidates of smooth path.

4 Incremental Construction of GVG

In this section, we introduce a sensor based navigation method by incrementally constructing GVG.

4.1 Generalized Voronoi Graph

In our assumption, target environment is expressed by free space and a set of obstacles in planar area. Then the planar GVG is a set of points expressed by following equation.

$$G(x) = \left[d_i - d_j\right](x) = 0 \qquad (1)$$

where $d_i(x)$ is a distance to the closest point within an obstacle i from robot's position x. The equation generates several equidistant edges between two obstacles (called GVG edges), and Voronoi graph is expressed by a set of GVG edges.

4.2 Tracing GVG

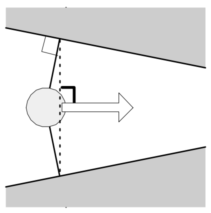
Fig. 1: Tracing

One of features of this approach is that the GVG can be calculated incrementally by using robot's range sensor. Then, the exploring motion is realized by maintaining equidistance between two closest obstacles. A orientation of tangent vector of local GVG edge that the robot should follow is perpendicular to a segment between the location of two closest points in the closest obstacles. Fig.1 shows an example of tangent vector. A mathematical detail of the control law for tracing GVG is described in [9].

4.3 Meet Points and Boundary Points

A meet point is, as its name suggests, a point where GVG edges meet. A meet point is (at least) triple equidistance to the closest obstacles.

$$G(x) = \left[\begin{array}{c} d_i - d_j \\ d_i - d_k \end{array} \right] (x) = 0$$

At a meet point, the robot has two or more candidates of path to trace, so it is very important point in GVG structure.

A boundary point is a point where two or more convex obstacles are contacted. Of course, the robot must stop tracing GVG before it arrives at boundary point, not to collide obstacles.

4.4 Conventional Exploration Method

Usually, the robot follows GVG (Section 4.2). When it encounters a meet point, the robot remembers location of it, and choose one of unexplored GVG edge. When it encounters a boundary point, the robot returns the traced GVG edge. Once there is no unexplored edge, the exploration task is done.

In above conventional approach, the robot follows GVG edge exactly to keep equidistance from obstacles, which a car-like robot can not.

5 Evaluation Function

In this section, we introduce our evaluation function for choosing optimal smooth path. To evaluate it, we define two distance functions in configuration space.

5.1 Configuration Space

We can not discuss a distance from robot to obstacles without its shape. Therefore, we construct configuration space in three dimensions, robot's position (x_r, y_r) and orientation θ_r.

Once C-space in local environment is constructed by conventional procedure[8] using laser range finder, a robot is regarded as a point in it. If a target path is differentiable, we can calculate an orientation of the robot at any point on the path. In this case, the path in C-space is also smooth curve. On the other hand, the GVG is expressed by surfaces in C-space perpendicular to x-y plane, because the GVG is not differentiable. Fig.2 shows C-space of L-shape environment and GVG. In this example, we assume a target robot as a rectangle.

5.2 Distance Functions

It is impossible for car-like robot to trace GVG, because the path is not smooth. However, the path is significant from the point of view of completeness and safety. Therefore, we consider following heuristics for definition of our evaluation function.

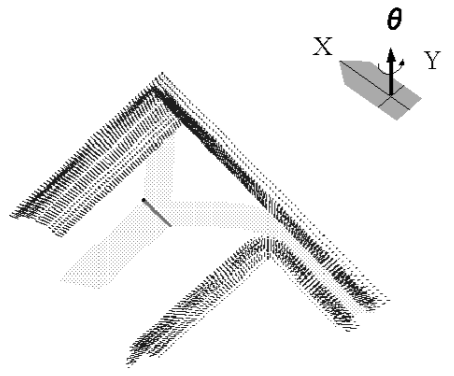

Fig. 2: Configuration Space and GVG

1. The path similar to GVG is better.
2. The path far from the obstacles is better.

According to above, we define following functions.

1. Distance to GVG

   Let an objective smooth path in C-space be L, and a point on L be x_c. Then, distance function to GVG is defined as,

   $$d_v(x_c) = min||x_c - v_i|| \qquad (2)$$

   where v_i is a point on GVG. Note that the distance $||x_c - v_i||$ is calculated on a $x - y$ plane which x_c belongs to. It means that θ_r is constant.

2. Distance to C-obstacles

   Distance function to C-obstacles (obstacles in C-space) is defined as follows.

   $$d_o(x) = min||x - C_i|| \qquad (3)$$

   where C_i is C-Obstacle on a $x - y$ plane which x_c belongs to.

Above calculation should be done on the plane of θ_r-constant in C-space. Fig.3 shows an example of two distance functions.

5.3 Evaluation Function

Based on above distance functions, we defined evaluation function $j(x_c)$ at each point x_c on L by following equation.

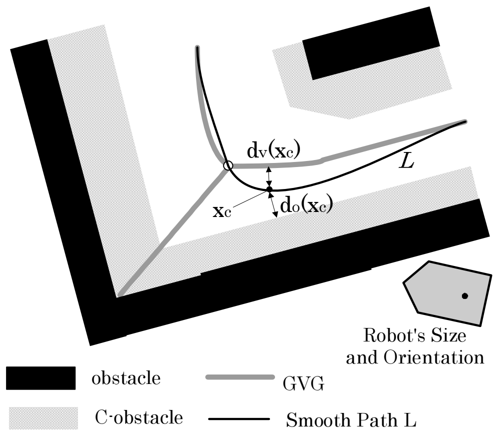

Fig. 3: Distance Functions

$$j(x_c) = \alpha \cdot \frac{1}{d_o(x_c)} + (1-\alpha) \cdot d_v(x_c) \tag{4}$$

The value of $\alpha(0 \leq \alpha \leq 1)$ is adjusting weight of two distance functions. In case that α is equal to 0, this function considers only traceability of GVG. On the other hand, in case that α is equal to 1, the function just considers the distance to obstacles.

To evaluate the target smooth path, we calculate integral of $j(x)$ along the path L, as follows.

$$J_L = \frac{1}{\|L\|} \int_L j(x_c) dx_c \tag{5}$$

Once the candidates of path are determined, we can calculate an evaluation value for each path by equation (5). Therefore, next problem is how we can generate candidates of traceable path for car-like robot locally.

6 Candidates of Smooth Path

In this section, we introduce a method to generate candidates of smooth path, which are expressed by Bezier curves.

6.1 Assumption

The number of candidates of path is immense even if small environment. Therefore, we assume following condition to narrow candidates.

1. The paths include (1) a point of current robot's location, (2) local goal point and (3) meet points. We call these points as anchor points. The reason we should include meet points is that the robot can adjust its estimated location easily at the meet point by sensor information.

2. Anchor points are connected by a cubic Bezier Curve. Therefore, the path is expressed by a set of Bezier Curves. The feature of it is that the line is smooth and differentiable.

3. The paths are smooth enough to be followed by car-like robots. It is discussed in section 6.3. Also, the robot should not corride to objects along the paths.

6.2 Bezier Curve

Bezier curve is famous for creating postscript drawing. It is smooth and differentiable, so it can be applied to car-like robot's path. The cubic Bezier curve is expressed by following equation.

$$x = x_1(1-u)^3 + 3x_2(1-u)^2u + 3x_3(1-u)u^2 + x_4u^3$$

$$y = y_1(1-u)^3 + 3y_2(1-u)^2u + 3y_3(1-u)u^2 + y_4u^3$$

where the curve tips are points of (x_1, y_1) and (x_4, y_4). Points of (x_2, y_2) and (x_3, y_3) are called control points to determine curvature of the path. u is a mediation variable $(0 \leq u \leq 1)$.

According to above equation, Bezier curve is expressed by four points' location. Two tips of the curve overlaps to anchor points. Therefore the curve is determined by two control vectors. The control vector is defined from an anchor point to the neighbour of control point.

Additionally, we assume that (A) the lengths of control vectors are the same in one Bezier curve, (B) one control vector in Bezier curve i should be the inverse orientation of the other control vector in neighbour Bezier curve $(i+1)$ for smooth connection between two Bezier curves. An example of above definition is shown in Fig.4.

Finally, one path that includes n of anchor points is expressed by $(n-1)$ of Bezier curves, and $(n-2)$ of variable orientation at connecting anchor points. Therefore, $(2n-3)$ of parameters is required to define the path. We can make candidates of smooth path by changing parameters of control vector length and orientation for each Bezier curve.

In unreal environment, it is difficult for only cubic Bezier curve to describe path between two anchor points (e.g. waving corridor). However, to consider usual environment and car-like robot, we do not assume a very complicated environment.

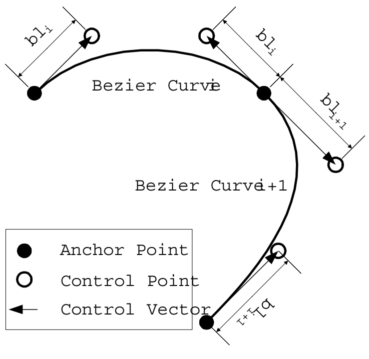

Fig. 4: The Path expressed by Bezier Curves

6.3 Traceability

Before applying our evaluation function to a candidate path, the traceability should be checked. It is done by calculation of a maximum curvature in target path, and checking corrision to objects. Usually, we had better discuss about the traceability with velocity. However in this research, we assume that the robot's speed is very slow, and we ignore the effect.

In two dimension case, curvature $\kappa(u)$ of general curve at the location of u is calculated by equation (6), where curve is expressed as $(f_x(u), f_y(u))$ and u is a mediate variable.

$$\kappa(u) = \frac{(\frac{\partial^2}{\partial u^2}\,\mathrm{fy}(u))\,(\frac{\partial}{\partial u}\,\mathrm{fx}(u)) - (\frac{\partial}{\partial u}\,\mathrm{fy}(u))\,(\frac{\partial^2}{\partial u^2}\,\mathrm{fx}(u))}{leng\,((\frac{\partial}{\partial u}\,\mathrm{fx}(u))^2 + (\frac{\partial}{\partial u}\,\mathrm{fy}(u))^2)} \tag{6}$$

The parameter *leng* is the length of the target curve. Bezier Curve is second-order differentiable equation, so curvature is guaranteed to be calculated at any point along the curve.

Using above equation, we can check the traceability of each Bezier curve by comparing maximum curvature of car-like robot κ_{max}. It is calculated by

$$\kappa_{max} = \frac{1}{p_{min}} \tag{7}$$

where p_{min} is a minimum turning radius of car-like robot, which is a peculiar value for each robot.

Finally, the generated curves are mapped into C-space to check a corrision, and corrision-free paths are defined as candidate paths.

7 Path Planning Method

Once the robot generates C-space and GVG locally, candidates of smooth path are calculated (Section 6.2) by changing parameters of control vectors. Then the evaluation function (Section 5.3) is applied to pick up the best one.

If the path consists of many Bezier curves, a number of candidates of smooth path explode and path planning is computationally impossible. However, we apply the algorithm in local path planning, and then the number of combination must be small.

8 Simulation

We implemented local path planning algorithm to simple L-shape environment. We introduce the simulation results in this section.

8.1 Target Robot and Environment

The size of the car-like robot is almost the same as actual robot that we possess, shown in Fig.5-(A). In simulation, we regard the robot as rectangle, shown in Fig.5-(B).

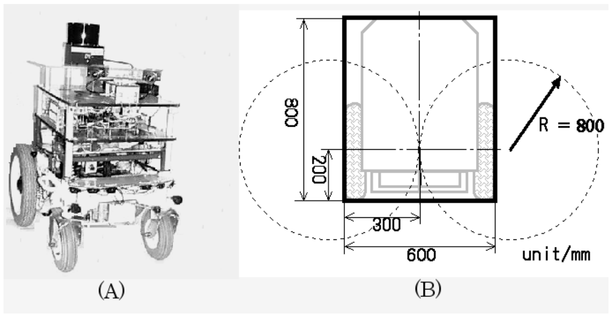

Fig. 5: Target Robot

A target environment is L-shape corner, which size is about 5 meters square. Configuration space and GVG in the environment are calculated in advance, shown in Fig.2.

8.2 Candidates of Smooth Path

We assume that the robot locates at the middle of narrow corridor, and goal location is middle of wide corridor. Both points are the tip of GVG. Note that the robot obtains the environment information from range sensors, so it skips exploring to boundary point.

In this case, a number of anchor points are three, (1) the point of robot's current location, (2) local goal point and (3) one meet point. Therefore, the number of variable parameters is $3 (= 2 \times 3 - 3)$ to determine a candidate of smooth path. We generated 20,000 paths by changing these parameters, checked maximum curvature (equation (6) and (7)), and confirmed not to collide to obstacles in C-space (Section 6.3). Once the path is traceable, we calculated evaluation function for each path (equation (4)), and pickup the best one.

8.3 Simulation Results

To compare an effectiveness of two distance functions, we changed a weight parameter α. Fig.6 and Fig.7 shows two cases of planned path ($\alpha = 0$ and $\alpha = 1$).

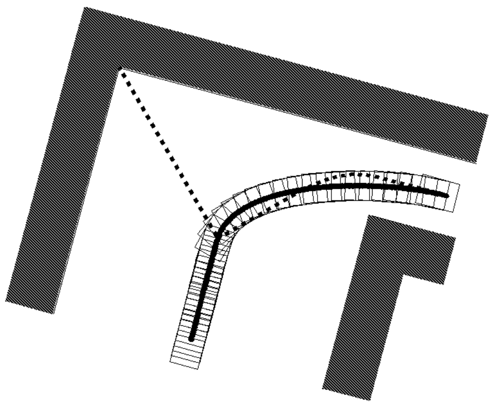

Fig. 6: Simulation Result ($\alpha = 0$)

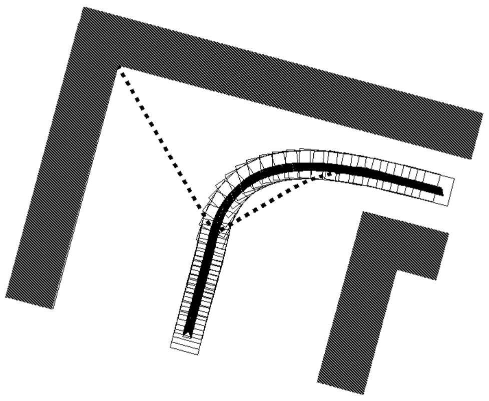

Fig. 7: Simulation Result ($\alpha = 1$)

In case of human driving a car, we may choose Fig.7 in intuition. The reason is that the maximum curvature of it is smaller than Fig.6-(A). However, from the point of view of similarity with GVG, (A) is better. It means that α depends on the importance of tracing GVG, and it is very difficult to find the best value.

9 Conclusion and Future Works

We proposed local path planning method for car-like mobile robot based on GVG. It is not simple solution and computational cost becomes much bigger than conventional approach. However, the simulation proves usefulness. One of the chosen path is almost same as car-driving path for people.

One of our future work is to integrate the local path planning algorithm into incremental construction of GVG for large scale exploration. The other is to take care of switch back motion for car-like robot in the same outline of our algorithm. Finally, our goal is to enable sensor based navigation for car-like mobile robot in real environment.

References

[1] H. Choset, I. Konukseven, and J. Burdick, "Mobile robot navigation: Issues in implementation the generalized voronoi graph in the plane," in *Proc. of IEEE/MFI*, (Washington DC), 1996.

[2] V. Lumelsky and A. Stepanov, "Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," in *Algorithmica, 2*, pp. 403–430, 1987.

[3] K. Nagatani, H. Choset, and S. Thrun, "Towards exact localization without explicit localization," in *Proc. of IEEE International Conf. on Robotics and Automation*, pp. 342–348, 1998.

[4] M. Viale, T. Tsubouchi, and S. Yuta, "A practical path and motion planner for a tractor-trailer robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)'97, vol.2*, (Grenoble, Canada), pp. 989–996, 1997.

[5] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics 145(2)*, pp. 367–393, 1990.

[6] P. Moutarlier, B. Mirtich, and J. Canny, "Shortest paths for a car-like robot to manifolds in configuration space," *Int. Journal of Robotics Research, 15(1):36-60, 1996.*, 1996.

[7] M. Vendittelli, J. Laumond, and C. Nissoux, "Obstacles distance for car-like robots," *IEEE Trans. on Robotics and Automation, 15 (4), 1999.*, 1999.

[8] J.-C. Latombe, *Robot Motion Planning.* Kluwer Academic Publishers, 1996.

[9] 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.