Sensor Based Navigation for Car-Like Mobile Robots Based on Generalized Voronoi Graph

Keiji Nagatani[1] **Yosuke Iwai**[2] **Yutaka Tanaka**[1]

[1] The Graduate School of Natural Science and Technology, Okayama University
3-1-1 Tsushima-naka
Okayama 700-8530, JAPAN
keiji@ieee.org

[2] Open Control Business Development Dept., OMRON Corporation,
Gatecity Osaki West tower 15F, 1-11-1 Osaki
Shinagawa-ku, Tokyo 141-0032, JAPAN

Abstract

Our research objective is to realize sensor based navigation for car-like mobile robots. We adopt Generalized Voronoi Graph (GVG) for robot's local path and a map representation. It 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 robots directly, because limitation of minimum turning radius for car-like robot may prevent from following GVG exactly.

To solve above problem, we propose local smooth path planning algorithm for car-like mobile robots. Basically, an initial local path is generated by conventional path planning algorithm using GVG theory, and it is modified smoothly by Bezier Curve to enable car-like robots' following by maximizing our evaluation function.

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

Keywords: Mobile robot exploration; Generalized Voronoi Graph; Non-holonomic constraint; Bezier Curve

1 Introduction

Sensor based navigation enables a robot to explore an unknown environment and build a map of it, using its sensor information. In this research, we focus on sensor based navigation using Generalized Voronoi Graph (GVG). The GVG structure has an advantage to express the paths of mobile robots 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 objects, and **"Completeness"** means that GVG is guaranteed as the unique structure in a static environment.

To apply the exploration algorithm to a real environment, we adopt our steering type car-like mobile robot as research platform. A car-like robot has following advantages.

1. It is easy to build by remodelling a conventional car-like vehicle (such as a wheelchair).
2. It is stable, and load capacity can be very large.
3. Developed algorithm can be applied to a conventional AGV or a future automobile.

However, it has a limitation of turning radius, and there is no guarantee for a car-like robot to follow GVG exactly, because the shape of GVG may not smooth. Therefore, we can not apply a conventional exploration algorithm to this type of robots. To solve the problem, we developed local smooth path planner using the evaluation function proposed in this paper to follow GVG as much as possible.

In this paper, we propose a novel "sensor based planning algorithm" for car-like mobile robots based on GVG theory. In this algorithm, the robot acquires a local environment model by its range sensor, and generates a local GVG. Then, it generates many candidates of smooth path, and chooses the suitable one using our evaluation function. Finally, the robot traces the suitable path. By repeating this procedure, the robot explores in an unknown environment.

In the above procedure, the robot's path does not correspond to GVG. However, this procedure generates the unique GVG structure, and it is equal to knowing the environment information.

A problem to construct a local path is an occlusion. A conventional method based on GVG theory relies on only two closest range data to generate local GVG, so there is no occlusion problems. However, generating GVG by range sensor's data in 360 degrees direction includes occlusion problems, and it may generate wrong GVG caused by virtual free spaces. In this paper, we also discuss a guaranteed GVG from the point of view of occulusion problems to generate stable GVG.

2 Prior Works

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

2.1 Sensor Based Planning

Sensor based planning enables a robot to explore in 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 [1]. In this algorithm, only single touch sensor enables robust exploration, and it is mathematically complete. However, in a real environment and real robots, we believe that a robot should have powerful sensors to complete its exploration mission, so that it could generate a better path than Bug Algorithms.

The other approach is the incremental procedure to construct GVG [2] that requires only line of sight information (obtainable from robot's range sensors) and 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, it can not be applied to car-like mobile robots because GVG is not smooth.

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

2.2 Nonholonomic Robots' Navigation

To generate a path for Nonholonomic robots (for example, a path for an autonomous car's parking), many heuristic's algorithms were applied to actual robots in a real environment. One experimental approach is to use a state transition of trailer type robots for a corner curve, thus the robot's path is generated by a combination of state patterns [4]. It is successfully implemented on a real robot, however a target environment is restricted.

To consider non-heuristic algorithms for any types of an environment, one of the 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 paths when radii of all circle are bigger than minimum steering radius. However, optimization (or safety)

is not discussed. For optimization, the shortest path for car-like robots in manifold is proposed in [6], and a nonholonomic distance is discussed in [7]. Both researches aim to plan the shortest path for car-like robots. Instead, a feature of our research is that the robot basically tries to follow GVG to obtain the environmental information.

To consider shapes of mobile robots, configuration space (C-space) is useful for motion planning [8]. We tried to apply a C-space explanation to represent a local environment, constructed by laser range sensor's information [9]. However, in our case, it took too long for a motion planning to succeed a realtime sensor based navigation. Therefore, at first, we considered a robot motion on an x-y plane, and checked a collision for each path.

3 Incremental Construction of GVG

RGVG A base of our algorithm is an incremental procedure to construct GVG. This section reviews the procedure for holonomic mobile robots that has been proposed in [2].

3.1 Definition of Generalized Voronoi Graph

In planar case, a target environment is categorized by objects and free spaces, and each concave object is divided into several imaginary convex objects. Then at a location x in free space, distance function $d_i(x)$ from a convex object C_i is defined as $d_i(x) = \min_{c \in C_i} \|x - c\|$.

Using above distance function, a planar GVG edge is defined by two objects C_i and C_j, as follows.

$$F_{ij} = \{x \in F_s : d_i(x) = d_j(x) \leq d_h(x) such\ that \nabla d_i(x) \neq \nabla d_j(x)\} \tag{1}$$

where $\nabla d_i(x)$ is a gradient function that means a gradient vector from the closest point of object C_i.

GVG is a set of GVG edges that satisfy above equation for a combination of all objects. It is unique in a static environment, and it can be constructed incrementally by sensor based navigation for a mobile robot with range sensors by following the procedure.

3.2 Accessing GVG

First, the robot accesses to the closest GVG edge. This motion can be executed using gradient vector (The robot leaves from the closest object until it arrives the middle point between the two objects). Values of distance function and gradient function can be calculated by its range sensor.

3.3 Tracing GVG

A motion to follow GVG edge is realized by maintaining equidistance between two closest convex objects. An orientation of tangent vector of local GVG edge (that the robot should follow) is perpendicular to a segment between the location of two closest objects' points. Fig.1 shows an example of tangent vector to trace. A mathematical detail of a control law to trace GVG is described in [10].

3.4 Detecting Meet Points

A meet point is, as its name suggests, a point where GVG edges meet. A meet point is (at least) triple equidistance to closest objects in planar case.

$$G(x) = \begin{bmatrix} d_i - d_j \\ d_i - d_k \end{bmatrix}(x) = 0 \tag{2}$$

It also can be detected by range sensors. When the robot arrives at a meet point, it stores its location and branches' directions of the GVG. Then it continues tracing motion along unexplored GVG edges.

3.5 Detecting Boundary Points

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

3.6 Backtracking GVG

At a boundary point, the robot can not continue to trace unexplored edges, then it must return by tracing explored GVG edges (backtracking GVG). In the other case, if there are no unexplored edges at a meet point, the robot must backtrack GVG edges to arrive at a meet point that includes unexplored GVG edges.

3.7 Terminate Condition

Incremental construction of GVG is done by the above procedure, "tracing GVG edge", "storing meet/boundary points" and "backtracking GVG". Once the robot recognizes that there is no meet point that connects to unexplored GVG edge, the exploration is done.

3.8 Reduced GVG

Using above procedure, the robot may generate useless GVG edges caused by small uneven objects in an actual environment. Also, weak meet points (it appears and disappears depending on sensing noise and threshold) are troublesome. Fig.2-(A) shows an example of the weak meet point problem. Two meet points may not be detected because of sensing noise in this figure.

Usually, useless GVG edges are connected to boundary points, so Reduced GVG (RGVG) is defined by cutting such useless GVG edges from an original GVG structure[11]. Once the robot detects a GVG edge connected to a boundary point, it removes the edges. Fig.2-(B) is an example of RGVG structure reconstructed from normal GVG structure of Fig.2-(A). This simple rule can save costs for entire exploration in an unknown environment.

In this research, an objective is to construct RGVG in an actual environment using car-like mobile robots (Fig.5 is one example of a part of RGVG). To avoid troublesome expressions, we describe RGVG as GVG in the following sections.

4 Exploration Procedure for Car-like Robot

In the conventional approach shown in section 3, a robot must locate on GVG edge always to keep equidistance from two or more objects. A car-like robot can not perform such motion because of constraint of minimum turning radius. To realize exploration by car-like robots, we designed the following procedure.

1. Acquiring a local environment information by range sensors
2. Constructing GVG based on the local environment information
3. Generating candidates of smooth path based on the GVG
4. Evaluating these candidates to choose the suitable one
5. Executing the planned path

6. Repeating from 1 to 5, until whole GVG structure is acquired

In the following section, we explain details of "how to generate a local GVG (in section 4.1 – 4.4)", "how to generate candidates of smooth paths (in section 4.5 – 4.7)", and "a definition of our evaluation function (in section 4.8)".

4.1 Detection Area

A problem for general range sensors in exploration tasks is that an accuracy of range data becomes worse according to detection distance. Therefore, we exclude an outer range of fixed distance (L_{max}) for generating GVG.

Fig.3 shows an example of a range sensor's data in an actual corridor environment (next to our laboratory), generated by a laser range finder produced by Hamamatsu Photonics (located at "x" position). It gives 500 points of distance data in 360 degrees, and the maximum range is 50 meters. Locations of objects' surface calculated by distance information and orientation are expressed by dots. (We call the $(x - y)$ locations of dots "range data".) Also, we set L_{max} to 5 meters in this example. Applying proposed conditions, a gray area is the region that local GVG should detect.

4.2 Detecting GVG point

Theoretically, GVG can be generated by dividing objects into several pieces of convex objects. Practically, the division is very difficult without pre-knowledge. Furthermore, a fine range sensor returns small bumpy data even if a surface of the object is flat. (We use a laser range finder for a range sensor). It generates a number of small GVG edges theoretically. To avoid these problems, we use the following procedure to detect points on GVG edges (we call such a point "GVG point").

1. Choosing an arbitrary point in free space, named "reference point".
2. Drawing a circle (named "search circle") with reference point as a center of it, which pass the closest point of range data, named "object point".
3. Expanding the search circle a little to the direction of gradient vector from the object point.
4. If a number of groups of range data inside the circle is only one, return to 3. Definition of the "group" is that a distance between two range data locations is less than robot's size. It means that the robot never passes through the opening.
5. If the number of groups inside the circle is two, the center of the search circle is the GVG point.

Fig.4 shows an example of the above procedure. The reference point is moving away from the object point until the other group (dots of the bottom of the right) touches the circle.

4.3 Detecting GVG edge

If the above procedure is done at every point inside GVG detection area, acquisition of local GVG has to be completed. However redundant detections may happen, and a calculation cost is too big. Also, the constructed GVG edge is expressed by a set of GVG points, which is difficult to use for robot's navigation. Therefore, we perform the following procedure to detect local GVG edges.

1. Finding bigger range data gaps than robot's size

2. Picking up both ends of each open space
3. Connecting one of the ends and sensor's position to make "reference segment"
4. Choosing a reference point at the end of a "reference segment"
5. Applying "GVG point detection procedure" (in Section 4.2) to detect a GVG point
6. Sifting a reference point along the "reference segment", and repeating 5
7. Repeating 4-6 for all "reference segments"

Using this procedure, the GVG is expressed by a series of GVG points.

Fig.5 shows an example of detected GVG edges. Original environment is shown in Fig.3, and detected GVG is expressed with bold lines. Reference segments are expressed with dotted lines.

Using the above procedure, local GVG edges can be constructed with low calculation costs.

4.4 Reliable local GVG

Range sensor can not detect distances to the back of objects (occlusion areas). If there are occlusions in detected area, the constructed local GVG may not be completed because of an ambiguity of existence. Fig.6 shows an example of an incomplete local GVG. Upper figure shows an example of environment data from range sensor that includes occlusion. **R** is the location of the robot, dots along the walls are visible range data, and the dark gray area is an occlusion area. If there is another space to the left, GVG becomes the bold line in Fig.6-(A). However, if it is a narrower path, GVG becomes the bold line in Fig.6-(B). This example clearly shows that an occlusion area prevents to fix the shape of local GVG. We call the GVG point "unreliable GVG point".

On the other hand, if there is no occlusion area in a search circle, GVG point is settled. Therefore, we classify such GVG point as a "reliable GVG point". The bold line in Fig.6-(C) is reliable GVG edges, and the gray lines are "reference segments". The center of the smaller circle is an example of "unreliable GVG point" because it includes occlusion areas. The center of the bigger circle is an example of "reliable GVG point".

We choose sub-goals in local GVG as the farthest "reliable GVG points" from an initial position.

Fig.7 is an example of reliable GVG edges, which is constructed in section 4.3 (Fig.5).

4.5 Initial Position and Sub-goal

Next step is to generate candidates of traceable paths for the car-like robot. GVG structure is not smooth (particularly in the vicinity of meet points), then the car-like robot can not trace it because of the minimum turning radius. Therefore, we locally generate candidates of smooth paths.

To construct the candidates, it is reasonable that initial positions and an orientation of the robot be set as the current robot's position. We also set a sub-goal in local GVG as the farthest point of connected local reliable GVG. Also, orientation at the sub-goal is set as a tangent direction of the GVG point. If there is two or more possible sub-goals, one of it is chosen based on right-handed priority.

4.6 Expression of Path by Bezier Curve

To connect the initial position and one of the sub-goals smoothly, we use the third order of Bezier curve.

The curve is defined by two pairs of the anchor points and the control points. Once the pairs of anchor and control points are defined as $\{(x_0, y_0), (x_1, y_1)\}$ and $\{(x_3, y_3), (x_2, y_2)\}$, a point of the curve is defined as the following equation, where u is a parameter from 0 to 1.

$$x = x_0(1-u)^3 + 3x_1(1-u)^2u + 3x_2(1-u)u^2 + x_3u^3 \quad (3)$$

$$y = y_0(1-u)^3 + 3y_1(1-u)^2u + 3y_2(1-u)u^2 + y_3u^3 \quad (4)$$

To apply this curve to our path, we choose an initial position and a sub-goal position as anchor points. An example of the path is shown in Fig.8. Orientation of the robot is fixed at each anchor point, so two unknown parameters are the distances between the anchor points and the control points (L_1 and L_2). In other words, we can generate candidates of smooth curve by changing these two parameters. A feature of this path is that it is differentiable at any point of the path.

4.7 Restrict Condition of Candidate Paths

Parameters of L_1 and L_2 are restricted by two conditions, **traceability** and **non-collision**. We check the above conditions for each path candidate to remove unsuitable paths.

First is **traceability**. The robot has no capability to trace a smaller circle than the minimum turning radius. Thus the maximum curvature at any point of Bezier curve $\kappa(u)\,(0 \leq u \leq 1)$ is restricted by the radius of curvature ρ, where

$$\kappa(u) = \frac{\left(\frac{d^2}{du^2}y(u) \cdot \frac{d}{du}x(u)\right) - \left(\frac{d^2}{du^2}x(u) \cdot \frac{d}{du}y(u)\right)}{\left(\frac{d}{du}x(u)^2\right) + \left(\frac{d}{du}y(u)^2\right) \cdot leng} \quad (5)$$

$$\rho = \frac{1}{R} \quad (6)$$

R is the minimum turning radius (fixed value), and $leng$ is the length of Bezier curve $(0 \leq u \leq 1)$.

Second is **non-collision**. In an actual environment, the shape of the robot should be considered to avoid a collision. A tangent direction of a Bezier curve is calculated by the following equation.

$$\theta(u) = \tan^{-1}\left(\frac{\delta y(u)}{\delta x(u)}\right) \quad (0 \leq u \leq 1) \quad (7)$$

In this research, we regarded the robot's orientation as the tangent direction of the Bezier curve, and the shape of the target robot as a rectangle. Then, the robot's position and orientation are specified at any point of the Bezier curve, and the collision is detected by checking it's position and orientation along the curve $(0 \leq u \leq 1)$.

Before applying our evaluation function to path candidates, we narrow parameters of generating Bezier curve to reduce calculation time using the above condition.

4.8 Evaluation Function

To consider an optimal path for car-like robots, we use a distance function for our evaluation function that is similar to GVG's distance function shown in the equation (1).

Our distance function $d(x)$ is the closest distance from objects to the robot model (rectangle). An example of $d(x)$ is shown in Fig.9.

Totally, the bigger $d(x)$ is the better for robot's path. Therefore, we define the evaluation function as an integral of $d(x)$, as follows.

$$J_L = \frac{1}{\|L\|}\int_L d(x)dx \quad (8)$$

We change L_1 and L_2 to maximize J_L, and choose the biggest one to determine the optimal path.

5 Experiments

5.1 Platform and Sensor

Our research platform is a mobile robot based on a commercial electric wheelchair. Steering angle of it is determined by rotational speed of two driving wheels, automatically, and the minimum turning radius is 80 centimeters. Maximum velocity is 160 centimeters par second, however we use less than 30 centimeters par second because of sensor ability and to ensure our safety. Each driving wheel has a rotary encoder to realize odometry system. In this research, an odometry is only used to display sensor data. This robot can be controlled by setting analog voltage for a joystick port where the controller is onboard PC with CPU (Celeron 450MHZ).

We use a laser range finder (C8074) shown in Section 4 for a range sensor that detects 500 range data in 360 degrees in every 200 milliseconds. It is mounted on the center of two driving wheels (the height of sensor detection face is 73 centimeters from the ground).

A photograph of the robot is shown in Fig.10.

5.2 Experimental Environment

Our target environment includes corridor and laboratory rooms in Okayama University. A schematic of the environment is shown in Fig.11. It is one of the standard indoor environments, and there is no object that can not be detected by a laser range finder. A surface of the ground is flat, and a wheeled robot can navigate smoothly. Steps are blocked by panels, as shown in Fig.11-(A) and (B). An intricate area is also blocked by panels, as shown in Fig.11-(D) and (E).

There is one loop that makes the robot confuse for exploration. It is a big problem to localize the robot's position topologically in such a loop environment, because of a positioning error of the mobile base. (When the robot comes back to the first meet point through the loop, it is very difficult to distinguish whether it is a new meet point or already visited meet point.) Several approaches to solve this problem were proposed (i.e. [9]). However, in this research, we just set a panel as a wall to cut off the loop, shown in Fig.11-(C).

5.3 Exploration Method

Globally, an exploration is performed as the conventional procedure, shown in Section 3. Locally, a smooth path is determined by the proposed method shown in Section 4. The robot traces the path based on it's odometry information only, and the next planning is done when the robot arrives at the sub-goal.

When the robot arrives at a meet point, it just picks up a right-hand path in case that there are several possible branches. When the robot arrives at a boundary of a GVG edge, it should backtrack. In fact, it is very difficult to generate a switch back motion in narrow area for car-like mobile robots, so at every dead end, the robot switches back.

5.4 Experimental Results

Fig.12 shows one of experimental results. Each sensing data (gray dots) and planned path (bold line) is overlapped based on odometry information. The robot started at the lower right point of this figure, and followed the path to the upper right. Then it started backward navigation, and continued following GVG. Sensing points are represented by gray circles, and the robot planned local paths at these points. Finally, it arrived at the goal location (In this case, the goal means the last position of exploration). It took about 30 minutes to explore this environment.

This result seems a wrong map. However this procedure generates GVG (topological map), so there is no problem about the odometry error inside the robot.

Fig.13 shows three typical examples of generated local GVG. The alphabets in Fig.13 correspond to the location in Fig.12. These are expressed by robot's coordinates in Fig.13, and the center of each figure is the base position.

6 Summary and Future Works

In this paper, we introduced a sensor based navigation method based on GVG theory for a car-like robot. We solved a nonholonomic problem by generating local smooth paths at each sensing point. We also took care of an occlusion problem that generated a wrong GVG. Experimental results supported the validity of this approach.

In our current implementation, the robot stops when the path curvature is bigger than robot's capability. If the robot has an ability of back and forth motion, it may overcome this situation. Another problem is reflective objects. It is difficult for a laser range finder to detect reflective objects (such as glasses), and this prevents constructing precise GVG. In this case, other kinds of sensors are required to recognize reflective objects.

Our final goal is to solve the above problems, and to perform a robust exploration motion in an indoor environment.

References

[1] 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.

[2] 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.

[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.

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

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

[9] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 2, pp. 125–137, 2001.

[10] 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.

[11] K. Nagatani and H. Choset, "Toward robust sensor based exploration by constructing reduced generalized voronoi graph," in *Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp. 1687–1692, 1999.

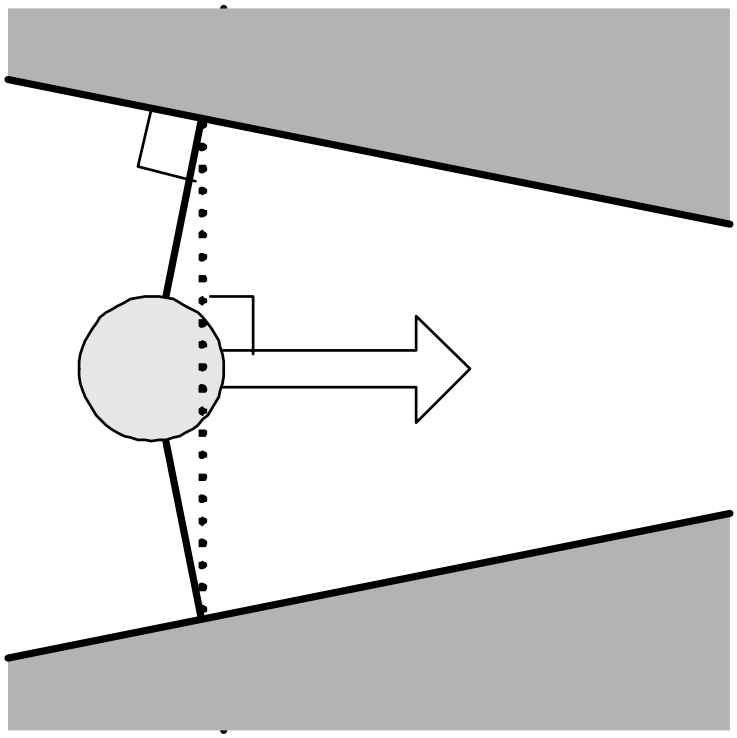

Fig. 1: Steering direction to trace GVG

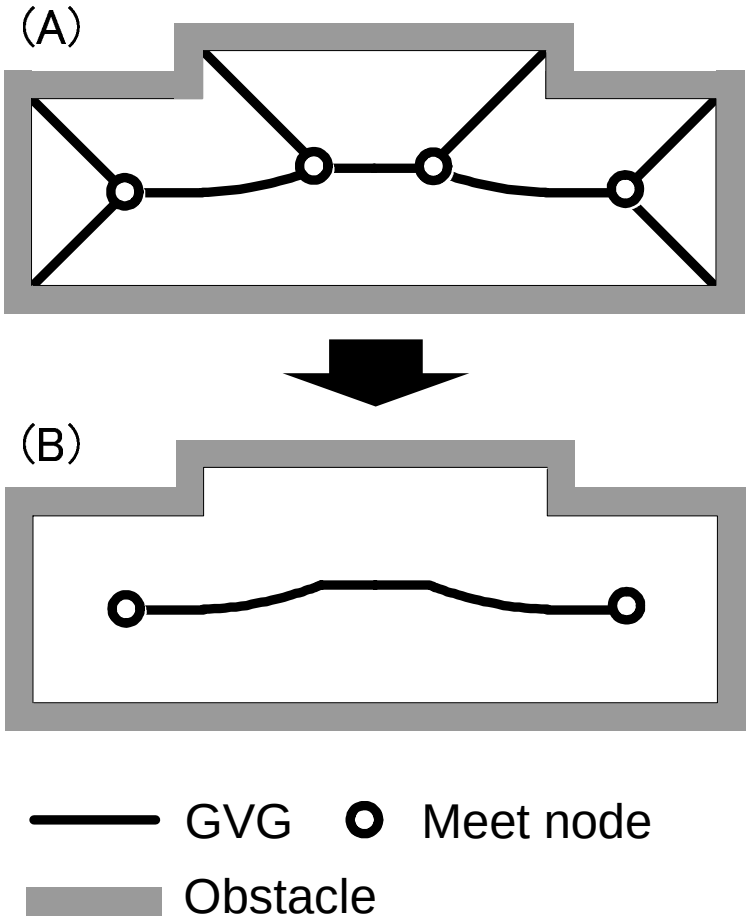

Fig. 2: Reduced GVG

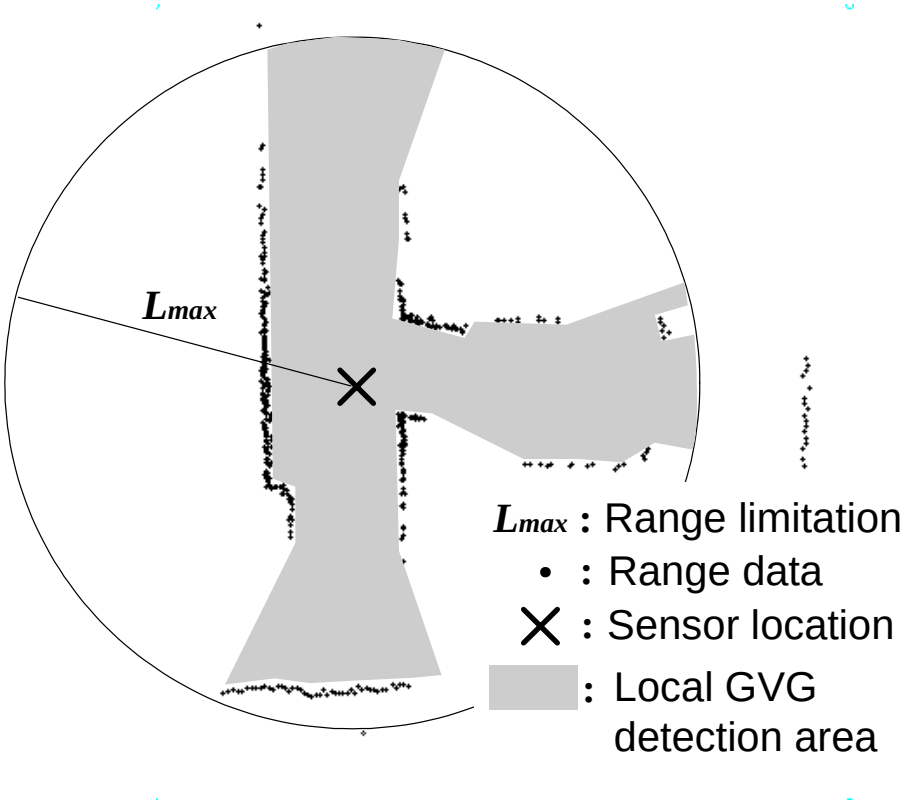

Fig. 3: Example of range sensor data

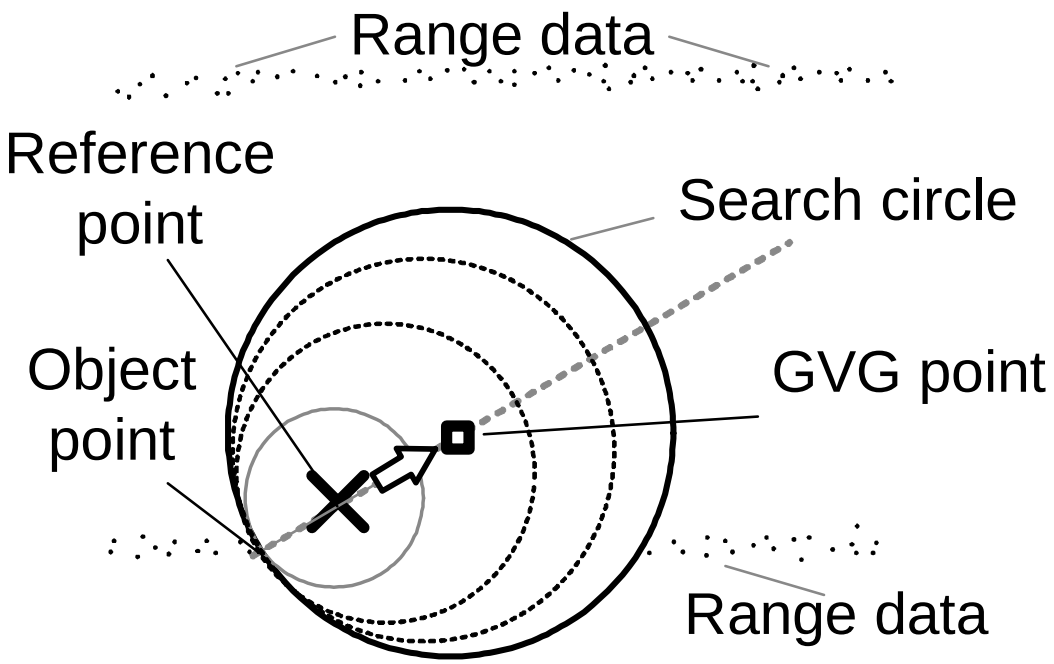

Fig. 4: Searching circle

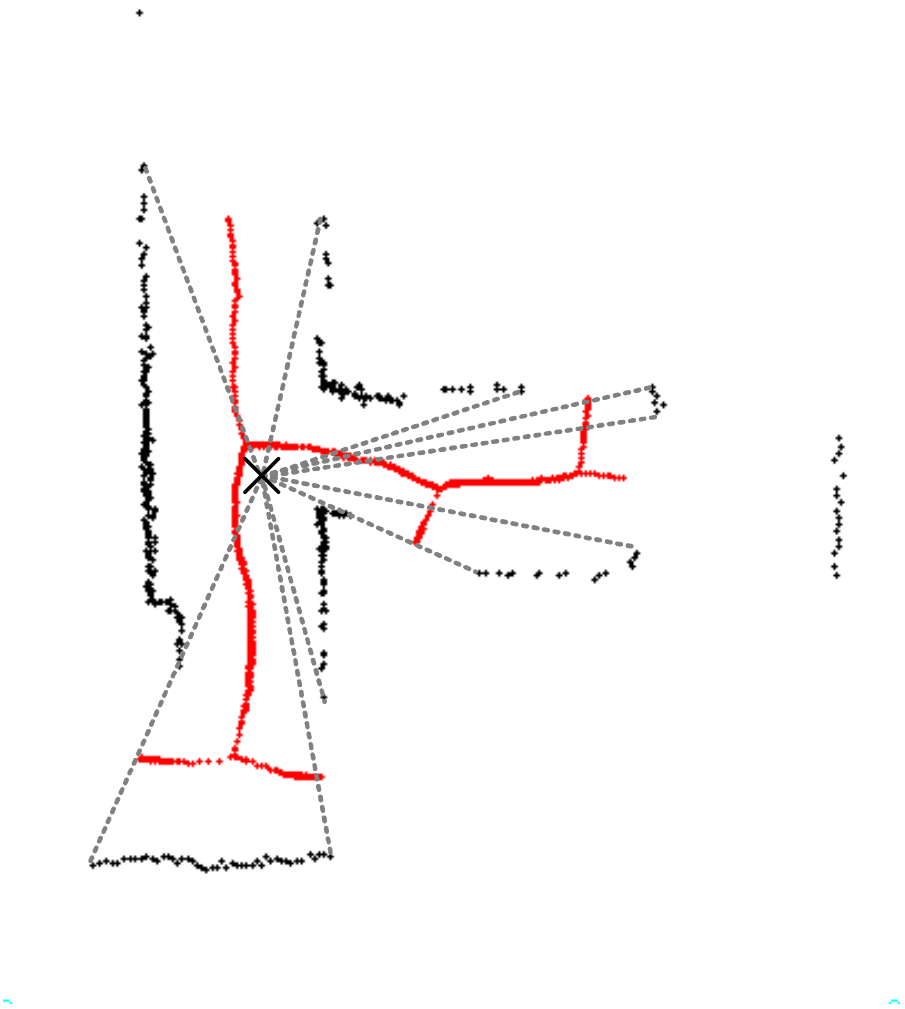

Fig. 5: GVG edge detection

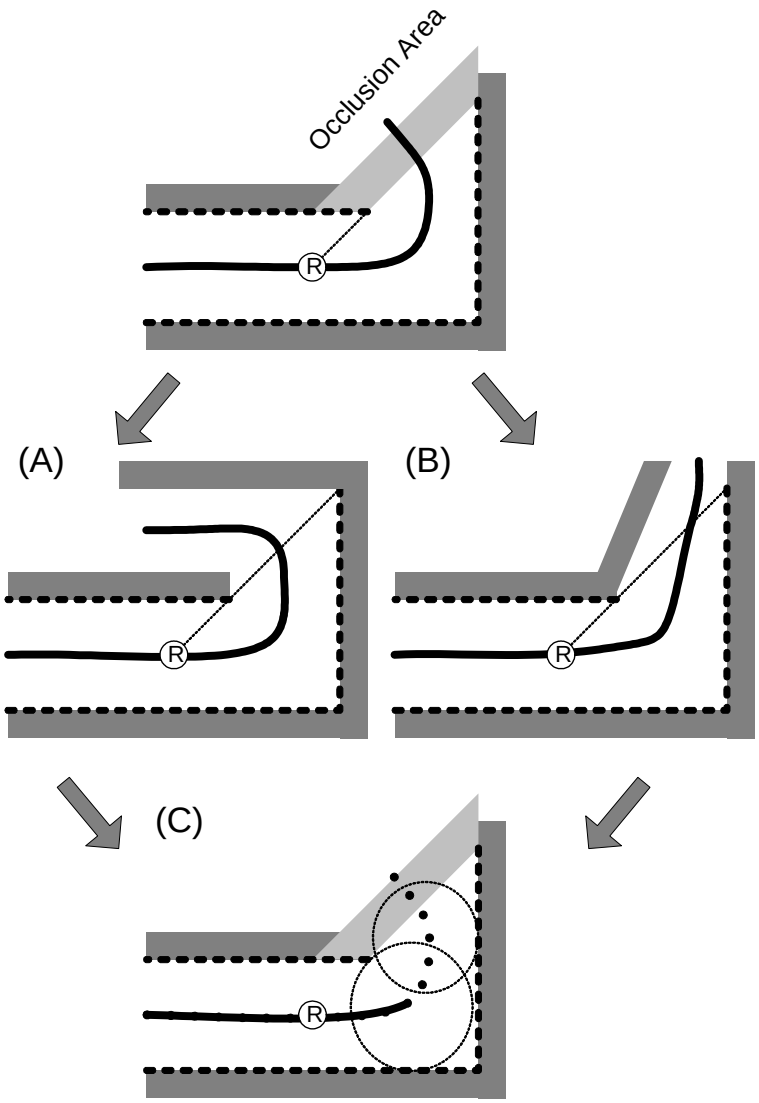

Fig. 6: Incomplete local GVG because of occlusions

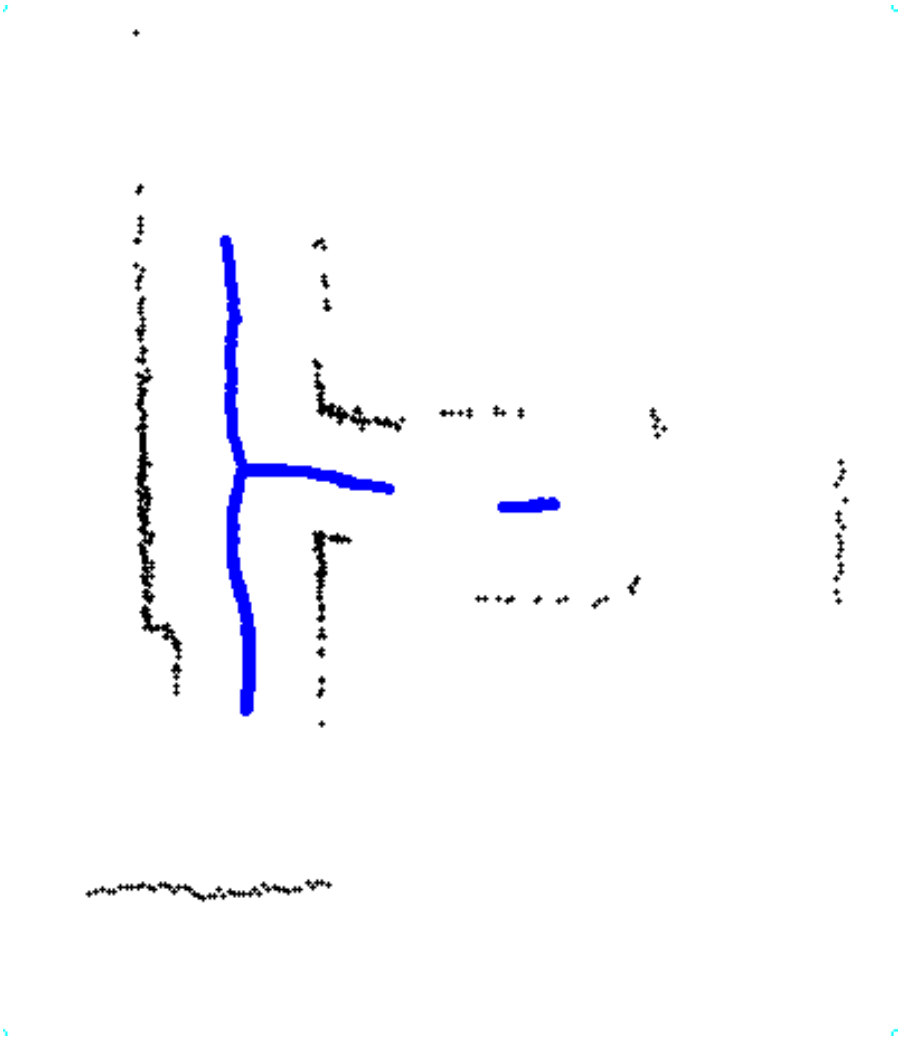

Fig. 7: Example of reliable GVG

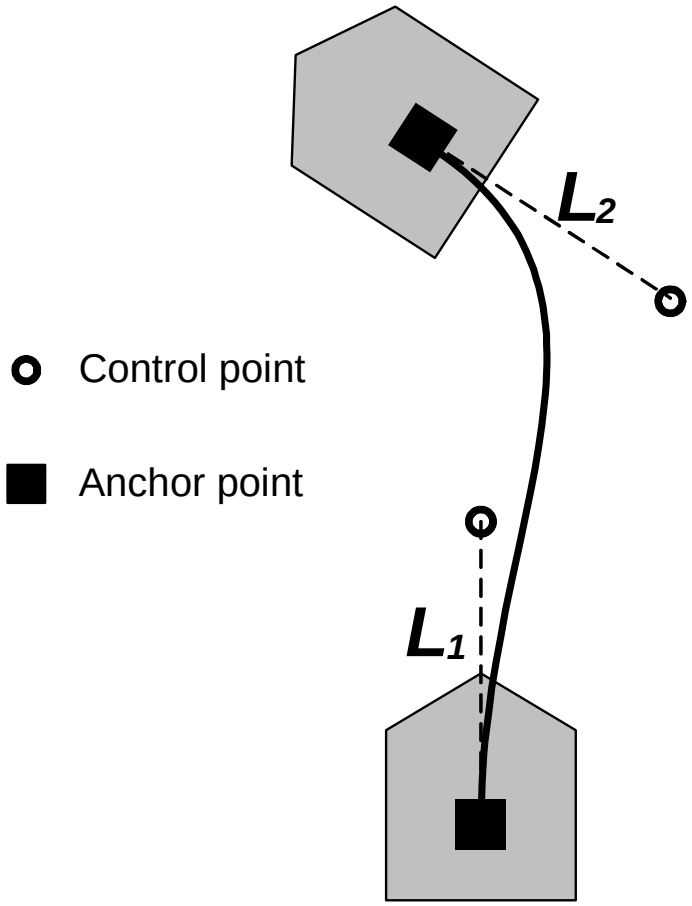

Fig. 8: Expression of smooth path by Bezier curve

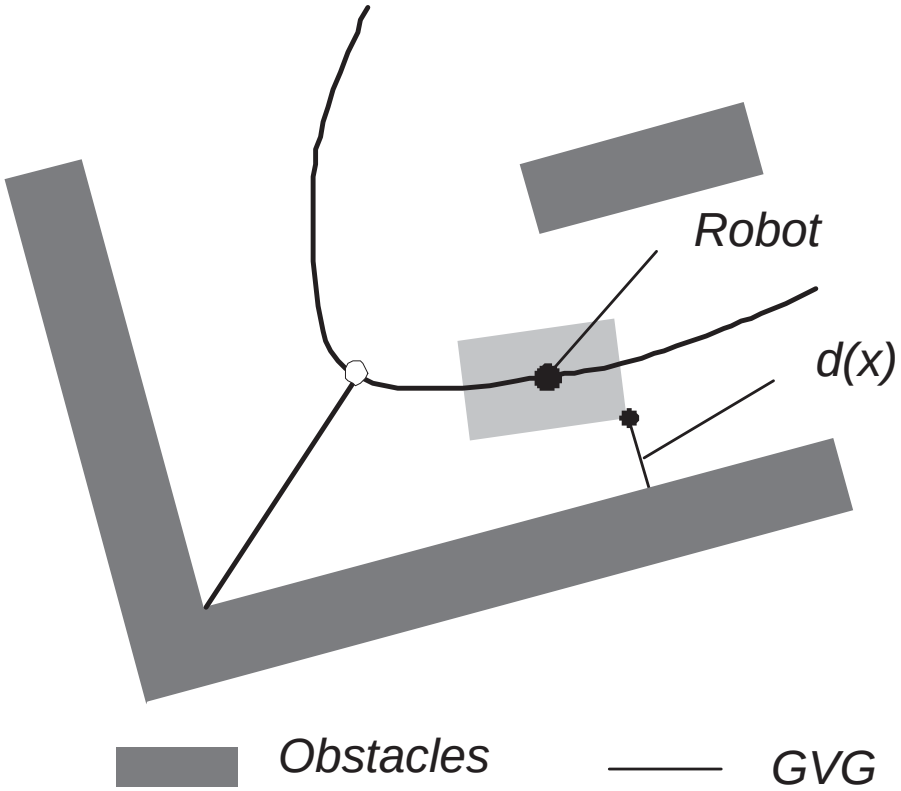

Fig. 9: Evaluation

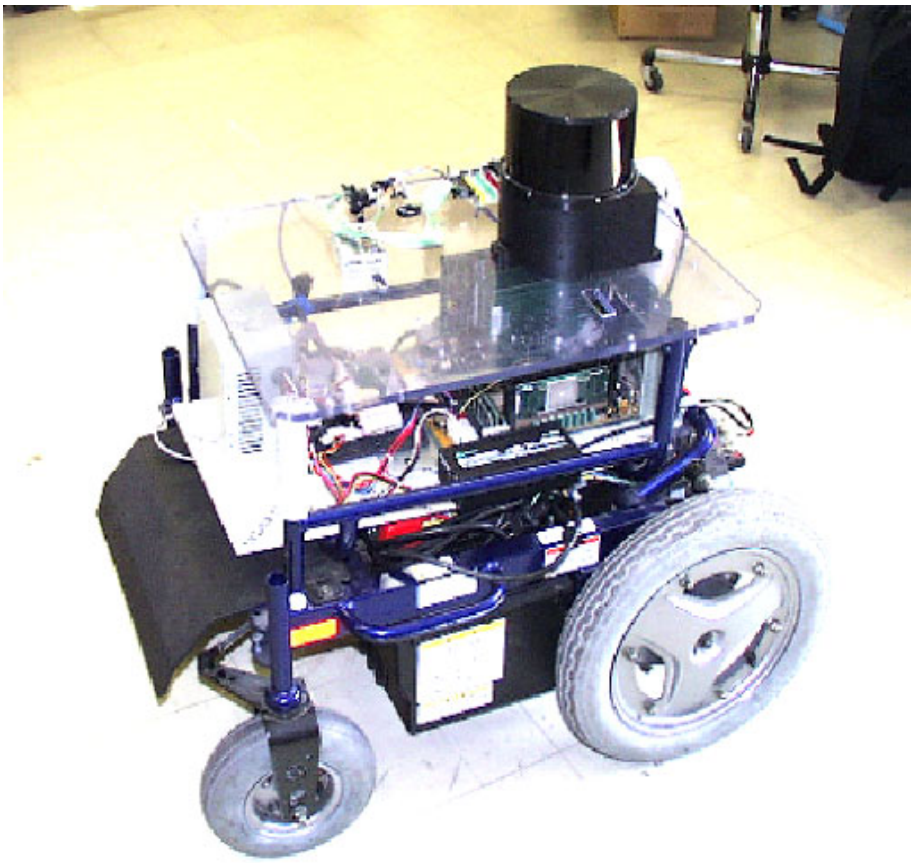

Fig. 10: Robot platform

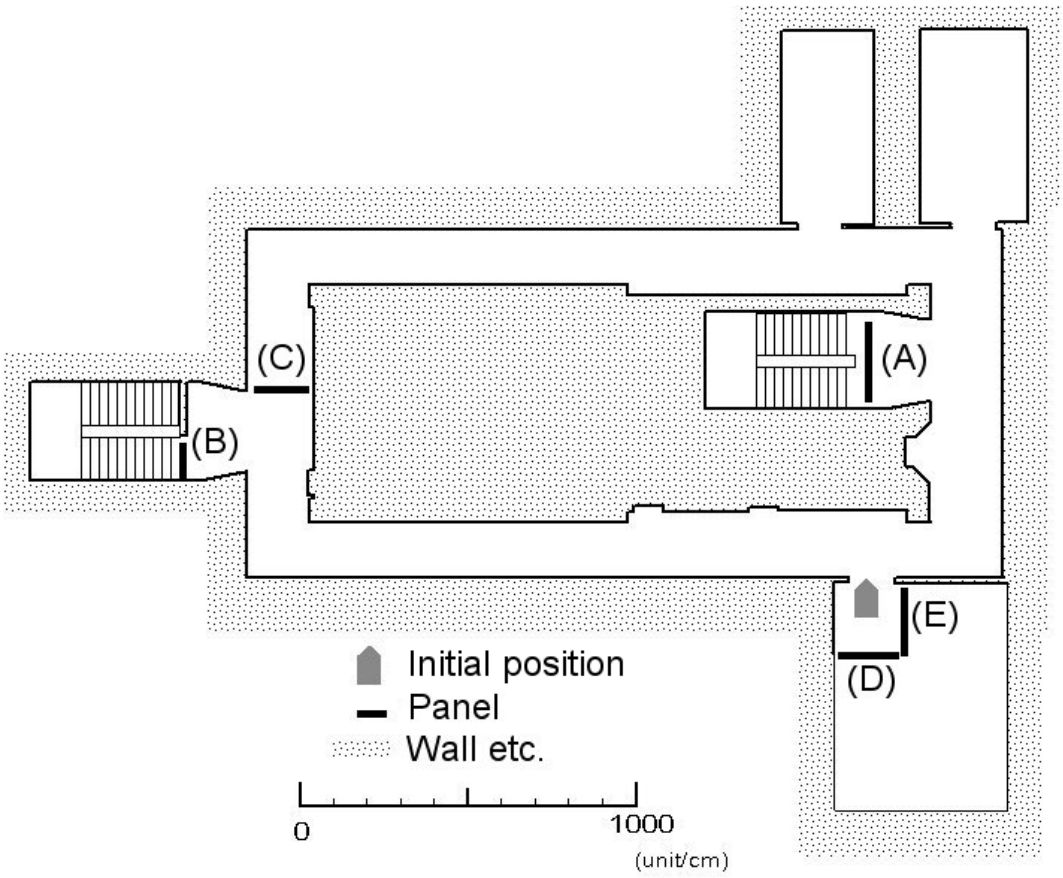

Fig. 11: Target environment

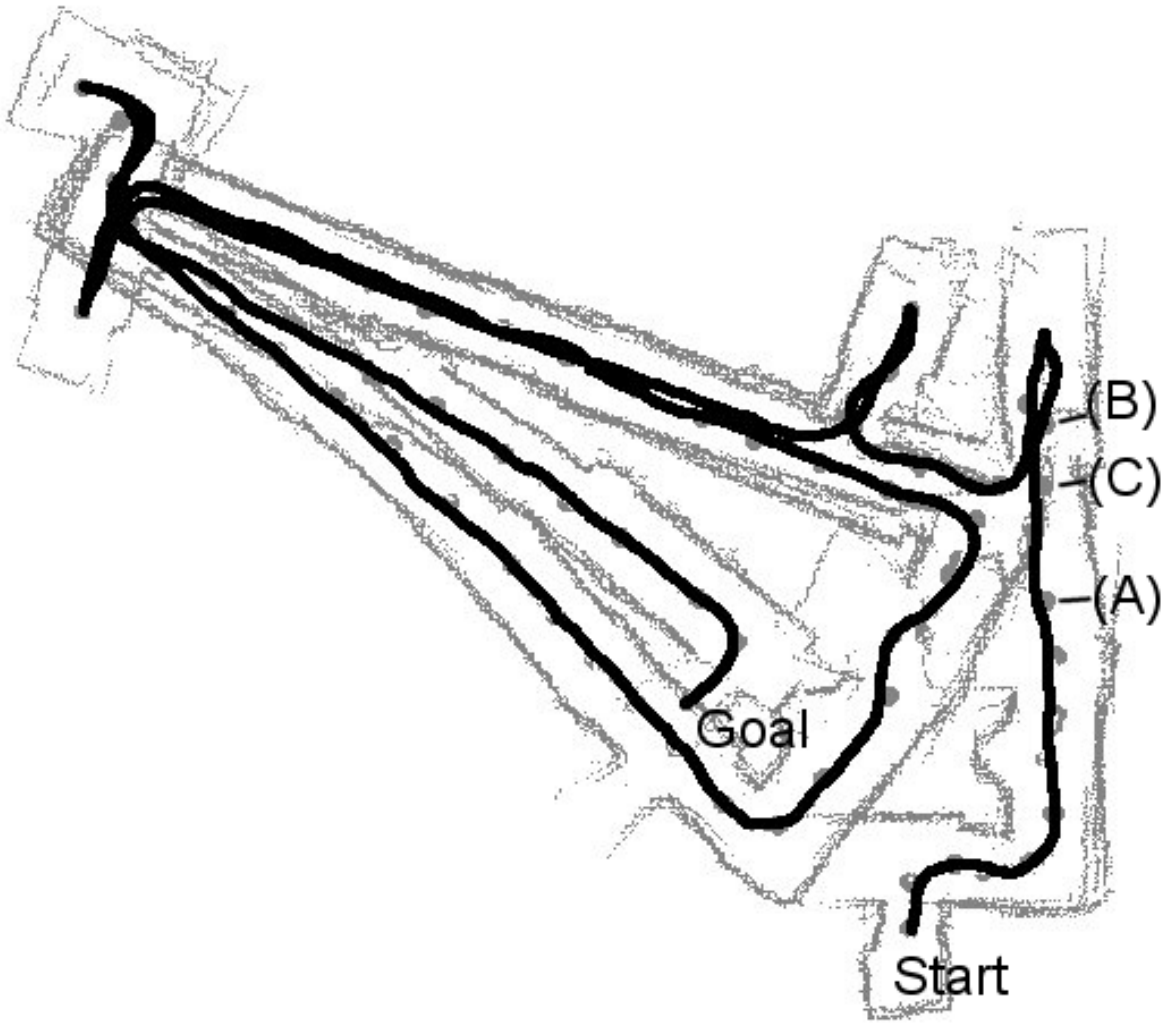

Fig. 12: Experimental result

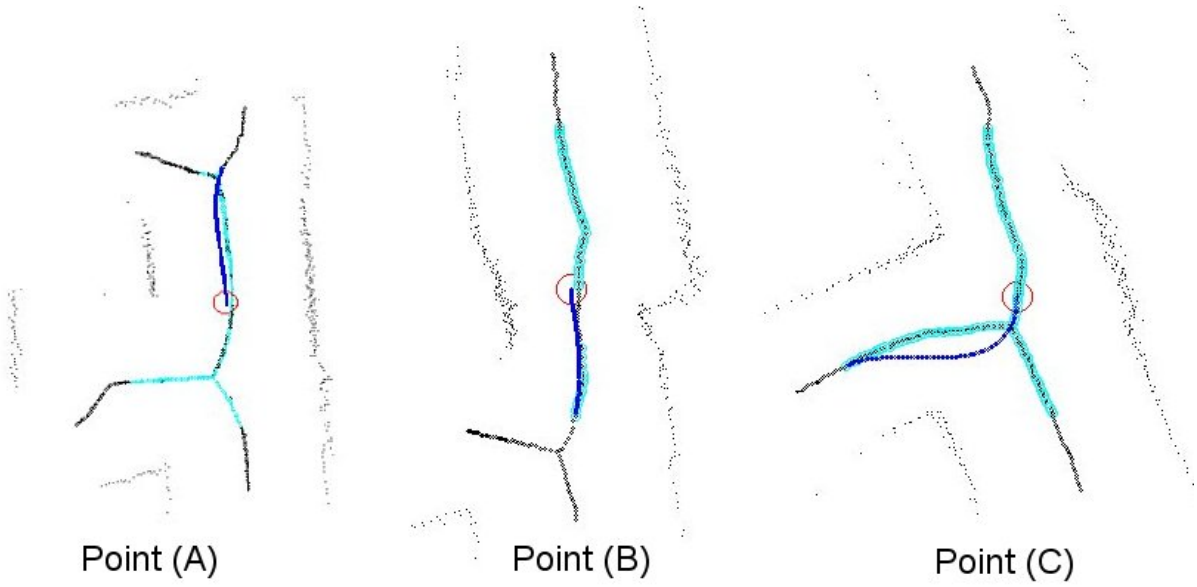

Fig. 13: Details of Experimental Result