Journal of Industrial Engineering 5 (2010) 1-15

Mobile Robot Online Motion Planning Using Generalized Voronoi Graphs

Ellips Masehian*, Amin Naseri

Tarbiat Modares University, Industrial Engineering Department, Tehran, Iran

Received 25 Oct., 2009; Revised 7 Nov., 2009; Accepted 20 Nov., 2009

Abstract

In this paper, a new online robot motion planner is developed for systematically exploring unknown environments by intelligent mobile robots in real-time applications. The algorithm takes advantage of sensory data to find an obstacle-free start-to-goal path. It does so by online calculation of the Generalized Voronoi Graph (GVG) of the free space, and utilizing a combination of depth-first and breadth-first searches on the GVG. The planner is equipped with components such as step generation and correction, backtracking, and loop handling. It is fast, simple, complete, and extendable to higher spaces.

Keywords: Robot Motion planning; Voronoi Diagrams; Medial Axis, Sensor-based Navigation.

1. Introduction

For a robot to perform its tasks correctly and safely, planning its actions and motions is indispensable. A perfect robot is no longer considered just a mechanism, but a software-supported hardware. The software must process the robot's knowledge of surroundings and take appropriate measures to guarantee the robot's collision avoidance and goal accomplishment.

The robot's knowledge of surroundings is either collected locally from robot's input devices such as sensors and cameras, or globally accessed via an environment map. Based on the scale of data acquisition, the robot's approach to planning will differ.

When no prior representation of the surrounding is available, a map of the environment has to be built incrementally. Thus, the mobile robot faces three fundamental questions as "Where am I?", "Where am I going?" and "How can I get there?" (Leonard and Durrant-Whyte [17]). The first question, which is on position estimation, is commonly referred to as *localization*. Related to localization is the concept of *navigation*. A system that can help a robot to establish its location or by some means Help it to find its way through its workspace correctly is called a Navigation System.

Navigation Systems are roughly classified into two groups: *relative* and *absolute position measurements*.

The relative position measurement approach, also known as dead reckoning, is further divided into two subgroups: *Odometry*, which uses encoders to measure wheel rotation and/or steering orientation; and *inertial navigation*, which uses gyroscopes and sometimes accelerometers to measure rate of rotation and acceleration. These approaches are subject to errors due to external factors beyond the robot's control such as wheel slippage or collisions. More importantly, dead reckoning errors increase unless the robot employs sensor feedback in order to recalibrate its position estimation.

The absolute position measurement approach contains a few techniques including Guidepath, active beacons, artificial landmark recognition, natural landmark recognition, and model matching. *Guidepath* is a static path (e.g. a wire that transmits audio or radio signals, or a magnetic stripe) which a robot can follow. Guidepaths are not suitable in applications where mobile robots should move freely. Being one of the simplest forms of robot

* Corresponding author E-mail: masehian@modares.ac.ir

navigation, Guidepath is used mostly for Automatic Guided Vehicles (AGVs) (Johansson [14]).

In *Active Beacons*, a set of light or radio signal transmitters (beacons) are used whose locations in the workspace are known in advance, and at least three of them must be "visible" (detectable) for a robot at all workspace locations. In the *artificial landmark recognition* method, objects or images with a distinctive shape are placed in the workspace for easy recognition. The positions of the objects are known, and if three or more objects are detectable at a certain position, an estimation of the position can be calculated.

In the *natural landmark recognition* method, the used objects already exist in the workspace, rather than being placed for robotic applications. Mostly they are easily distinguishable man-made structures like curbs, wall-floor edges, etc. In *model matching,* the information acquired from the robot's onboard sensors is compared to a map or world model of the environment. Map-based positioning often includes improving global maps based on new sensory observations in a dynamic environment and integrating local maps into the global map to cover previously unexplored areas.

Online navigation algorithms are those which plan the robot's motions in unknown environments using sensory data, and are generally categorized into two major groups: navigation using *touch sensing*, and navigation using *range finding* (laser/sonar sensors or camera vision).

Using one of the first methods for touch-based navigation, Lumelsky and Stepanov [18] presented *bug* algorithms for a point robot to move from a source point to a destination point, using touch sensing in a planar terrain populated with arbitrary shaped obstacles. Also, Cox and Yap [10] developed algorithms to navigate a rod to a destination position in planar polygonal terrains. A good review of early works on online path planning is provided in Rao et al. [23].

Recently, sensor-based motion planning has been done for real-time applications and unknown environments as in Brooks et al. [4] for kinodynamic constraints, and in Sam Ge et al. [25] for dynamic constraints.

During the past decade, technological advancements in sensor equipments caused the evolution and sophistication of navigational techniques. The long-range sensors obviated the need for robot-obstacle touch which is an unfavorable case in most motion planning environments and leads to physical damages to the robot. There are numerous studies done about sensor-based or vision-based navigation methods (e.g. Choset et al. [9]). Recent robotic manipulators are equipped with a camera for accomplishing fine motion planning tasks. An informative work dealing with sensors and sensor-based planning methods is Borenstein et al. [3].

1.1. Generalized Voronoi Graphs

The concept of Generalized Voronoi Graphs (GVG) or *Medial Axis* (MA) first appeared in the literature in 1967 when Blum [2] introduced the notion of a skeleton discussing Medial Axis Transform (MAT). He compared the symmetric or medial axis transform with a grass fire phenomenon where the fire on the borders of a grass field broke out toward the center. The fire fronts met and quenched in some points which formed the medial axis. Blum showed that these points are the centers of Maximal Inscribed Discs (MID). To mathematically express the MID, we need to define some terms:

Definition 1. Let W stand for the workspace and C its configuration space. Then C_{free} represents the free configuration space and C_{obs} denotes the C-space occupied by obstacles. Let the set of all possible distance values between any two elements in the C_{free} be called D:

$$D: = \{\|x-y\| \mid x, y \in C\}.$$

The *Distance Transform* DT: $C_{free} \rightarrow D$ assigns to every C_{free} element the minimal distance to the C_{obs}:

$$DT\,(x \in C_{free}) := \min\,\{\|x-y\| \mid y \in C_{obs}\},$$

where $\|\bullet\|$ is some arbitrary metric like Euclidean distance. The Distance Map is the set of all C_{free} elements along with their associated distance values:

$$DM\,(C_{free}) := \{\mathrm{x}, DT(x) \mid x \in C_{free}\}.$$

Definition 2. Since no boundary point is closer to x than $DT(x)$, every element $(x, DT(x))$ of the distance map defines the Locally Maximal Disc centered around x:

$$LMD(x) := \{\, y \mid \|x-y\| < DT(x)\},$$

describing the disc with maximal radius from among the values in D and centered around x which is completely contained in C_{free}.

Definition 3. A *Maximal Inscribed Disc* (MID) is a locally maximal disc which is not completely contained in any other disc. The set of maximal inscribed discs in C_{free} is therefore:

$$MID(C_{free}): = \{\, LMD(x \in C_{free}) \mid \forall\, y \in N(x),\; LMD(x) \not\subset LMD(y)\,\},$$

where $N(x)$ denotes the neighborhood of x. A MID touches at least two boundary points of C_{free}.

Definition 4. The loci of the centers of maximal inscribed discs comprise the *Medial Axis*, and the transformation of an object to its medial axis is called *Medial Axis Transform*.

Fig. 1 shows the Maximal Inscribed Discs and the Medial Axis of an L-shaped environment.

In Chin et al. [5], the medial axis of a simple polygon is calculated in linear time complexity, which is better than the previously known $O(n\log n)$. The authors decompose the polygon into pseudo-normal histograms, influence histograms and *xy* monotone histograms. A normal histogram is a simple polygon whose boundary consists of a base edge and a chain that is monotone with respect to the base, and a pseudo-normal histogram is a normal histogram with a missing corner. The medial axis for *xy* monotone histograms is then computed and merged to obtain the medial axis for the polygon.

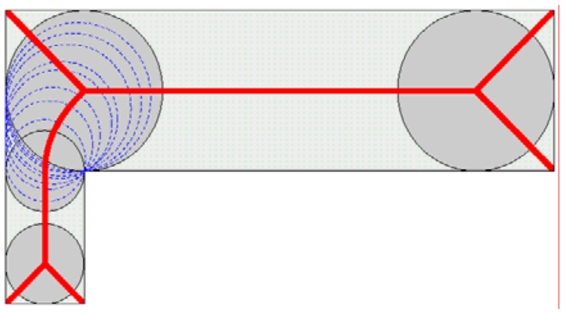


Fig. 1. Medial Axis (middle line) and some Maximal Inscribed Discs for a simple 2D L-shape. The axis is a piecewise quadratic curve representing the local symmetry axes.

In Datta [12], a constant-time O (1) algorithm for computing the medial axis transform of an $n \times n$ binary image on a reconfigurable mesh size of $n \times n \times n$ is presented. The mathematical properties of the medial axis are well studied in Choi et al. [6].

In Dardenne et al. [11] the MA for completely discrete objects in the form of pixels or voxels is approximated based on the Voronoi graph computed from a set of nodes distributed across the boundary. The approximations are robust to noise and suitable for mesh generation.

The medial axis has found applications in extracting skeletons from 3D neuron images (Petřek1 et al. [20]), Binary Data Compression (Pujar et al. [21]), solid modeling, motion planning, etc. In Wilmarth et al. [26], the medial axis transform is combined with a probabilistic roadmap planner (PRM) where randomly generated points in configuration space are connected by local methods (Kavraki and Latombe, [15]). The drawback of this planner emerges in narrow passages, where the probability of locating a random point in those regions is quite small, leading to a failure in connecting the two ends of the corridor. In Wilmarth et al. [26], a method for sampling the C-space is proposed, where randomly generated configurations, free or not, are retracted onto the medial axis of the free space without computing the medial axis explicitly.

1.2. Incremental Construction of Voronoi Graphs

Sensor based planning by incremental construction of the GVG includes three phases: (1) connecting the start point to GVG, (2) navigating through GVG roadmap, and (3) constructing a path to the vicinity of the goal. The properties associated with these phases are called *accessibility*, *connectivity* and *departability*, respectively, and are essential for a roadmap regarded for path planning.

There were some attempts to take advantage of the maximum clearance property of the Voronoi diagram, and to build the Voronoi diagram iteratively (e.g. Masehian et al. [19]). In Rao et al. [24], some proofs for four basic properties of Voronoi diagrams including (1) finiteness, (2) connectivity, (3) local constructability, and (4) terrain visibility are stated. Then they suggest an algorithm for the navigation of a circular robot in unknown terrains by iteratively visiting the Voronoi vertices. Rao [22] then extended these results to generalized polygons in plane.

A well-known work dealing with incremental construction of Hierarchical Generalized Voronoi Diagrams belongs to Choset in Choset and Burdick [7] and Choset et al. [9]. The importance of these studies lies in their completeness and applicability to higher dimensions than planar. Since the Voronoi Diagram is disconnected in three and more dimensional spaces, some "bridge" edges (called GVG^2) are used to maintain the connectivity of the GVG in high dimensions. The structure of Choset's algorithm depends extensively on the mechanism of the robot in hand (as in the Nomad® 200 robot with a sonar sensors ring), and is tailored for discrete information acquired from sensor readings.

In incremental construction of Hierarchical Generalized Voronoi Diagrams, the main procedure for incrementally building the GVG edge involves mathematical computations, that is, numerical continuation techniques as well as a need for implementing a correction step through Newton's recursive correction function, a calculation tailored to meet points considered m+1-equidistant faces. Although this approach is precise, the usually limited number of sensors and their incomplete perception of the world overshadow this advantage and force the robot to 'guess' its next direction.

The meet points (i.e. Voronoi vertices) are perceived by a comparative analysis of different sensor readings; that is, by watching for an abrupt change in the direction of the (negated) gradients to the m closest obstacles (Choset et al. [8]). A meet point is attributed as an m+1-equidistant point in $\Re^m$ space (e.g. 3 in planar cases) and calculations on this non-generic assumption are established. However, this is not the case with many situations where more than m+1 Voronoi edges conjoin at a single meet point, or there are inaccuracies in defining the borders of obstacles.

This approach renders numerous problems like dead reckoning error, localization error, sharp corners problem, weak meet points and problems due to hyper-symmetrical environments which are addressed in Choset et al. [8].

In the present paper, we propose a new motion planner which systematically and intelligently searches the workspace by incrementally building the Medial Axis or the Generalized Voronoi Graph of the free workspace. The paper is organized in this way: the following section describes the algorithm's components in detail, and section

3 provides all steps of the algorithm. Section 4 illustrates some simulations and deals with important issues such as completeness and time complexity. Also, a Potential Fields variant of the algorithm is proposed. Finally, section 5 presents the conclusion.

2. Algorithm Components

The online motion planner developed in this paper takes advantage of a number of basic geometric components, which together with a few control matrices allow the robot to perform motion planning tasks. These components are:

1. Distance checking component,
2. Departure component,
3. Projection component,
4. Correction component,
5. Backtracking component,
6. Loop handling component.

A high-level explanation of the online motion planner, which describes how the above components are integrated and incorporated in a single model, is presented below:

The algorithm begins with positioning the robot on the Start point and collecting information about its distance from surrounding obstacles. If the Goal point is in the robot's line of sight, then the robot's current location is connected to the Goal via a straight line, and the planning task terminates accordingly. Otherwise, depending on the robot's location (which can be either on MA edge, MA vertex, or not on MA at all), its next action is planned as follows:

If the robot is not on the MA, then it must correct its position by moving directly toward the MA. Afterwards, it must repeat the distance checking step.

If it is already on the MA, the robot should take a step along the MA in a promising direction. Nevertheless, this projection could 'derail' the robot from the MA (as in the previous case); so a correction step might be necessary to maintain the robot on maximum clearance from obstacles. This step guarantees that no collision will occur through the course of planning.

If during the navigation, the robot comes too close to an obstacle, it does a backtracking behavior, and traces back the trajectory points until a vertex (meet point) is reached. In that case, the robot starts exploring another edge stemmed from that vertex. It should be noted that all edge-traversing tasks (except for the backtracking operation) are accomplished through gradual construction of the MA via successive projection-correction steps.

Before we present the details of each component, a mathematical notation is introduced to establish a framework for the algorithm:

Let $P(x) = \{\rho_1, \rho_2, \ldots, \rho_r\}$ be the set of radial visibility (or sonar) rays emanated from the robot at location x. Each ρ_i has a magnitude $|\rho_i|$, and an angle θ_i.

Let $P_M(x) = \{\rho_m^1, \rho_m^2, \ldots, \rho_m^s\}$ be the set of rays having minimum distances from *different* convex obstacles sorted in an ascending order. So the first element, ρ_m^1, is the radius of the Maximal Inscribed Disc, MID(x) (see Fig. 5). Obviously, $P_M(x) \subset P(x)$. Note that $P_M(x)$ is not just a set of relatively small elements of $P(x)$; the fact that the minimum distances from different convex obstacles are calculated determines the elements of $PM(x)$ to be local minimums in the rays histogram.

The following subsections explain the components of the algorithm in more detail.

2.1. Distance Checking Component

This component provides information about the robot's environment. Consider Fig. 2 for instance, which illustrates a point robot among obstacles. It emits sonar or laser rays to measure its distance from the surrounding obstacles. Here the number of sensors r is 360, i.e. one sensor per degree. In real experiments this number is usually less, e.g. 18 or 36 sensors. The rays represent the set $P(x)$.

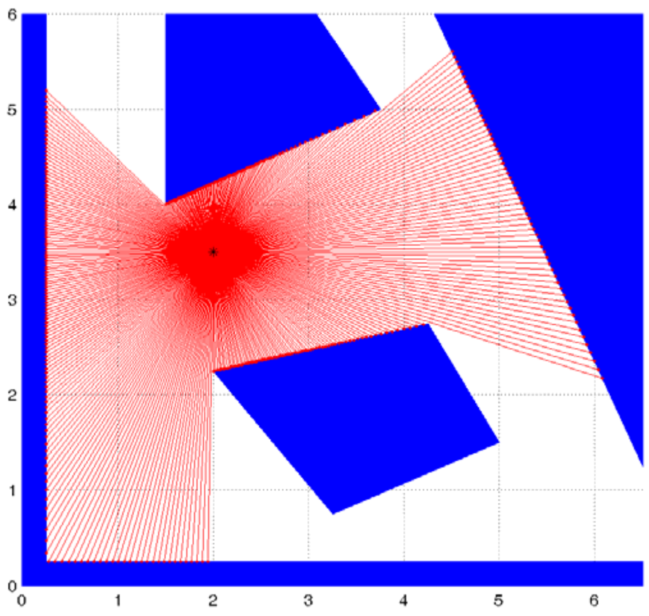


Fig. 2. Sonar rays emanated from a point robot amid obstacles.

To calculate the minimum distances from all visible obstacles, the robot must perform a simple arithmetic manipulation on magnitudes and angles of the rays. Fig. 3 depicts a polar histogram of ray magnitudes.

Fig. 4 is the Cartesian plot of the ray magnitudes. The local minimums in this histogram represent the rays with shortest distances from distinct obstacles. Fig. 5 shows these rays, which are elements of the set $P_M(x) = \{\rho_m^1, \rho_m^2, \rho_m^3, \rho_m^4\}$. ρ_m^1 is the minimal ray and is the radius of the Maximal Inscribed Disc.

Depending on the distances from obstacles, three cases, depicted in Fig. 6, can express the robot's position relative to the Medial Axis:

Case 1: $\rho_m^{\ 1} << \rho_m^{\ 2}, \ldots, \rho_m^{\ s}$:

This condition indicates that the robot is very close to one obstacle and is likely to collide with it (Fig. 6(a)). Therefore, this is an unstable and unfavorable case, and must be corrected and resolved to Case 2 or 3. To make this premise more pragmatic and verifiable, it can be rephrased as $|\rho_m^{\ 1} - \rho_m^{\ 2}| > \varepsilon$, where ε is a small tolerance value.

Case 2: $\rho_m^{\ 1} = \rho_m^{\ 2} << \rho_m^{\ 3}, \ldots, \rho_m^{\ s}$:

This case implies that the robot is located on the Medial Axis and maintains the maximum clearance from two obstacles (Fig. 6(b)). In the real world applications, the exact equality of $\rho_m^{\ 1} = \rho_m^{\ 2}$ may not be attained, and so it can be relaxed to $|\rho_m^{\ 1} - \rho_m^{\ 2}| \le \varepsilon$.

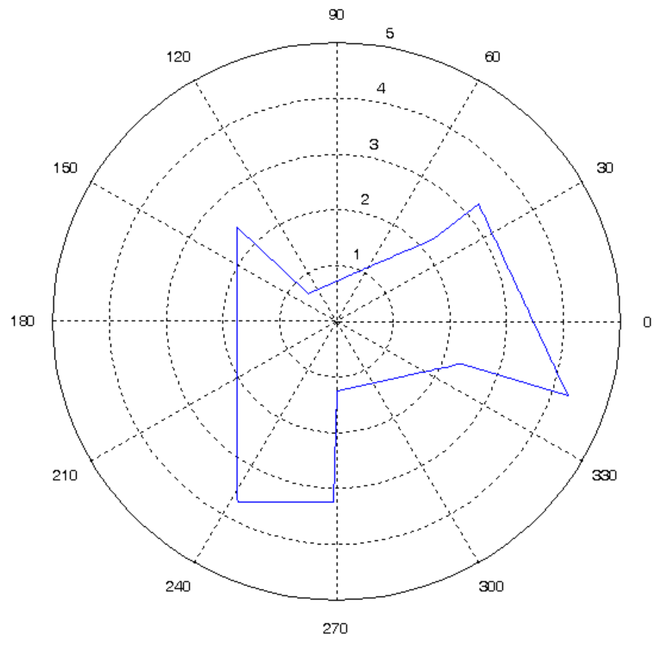


Fig. 3. Radial rays magnitudes plotted in polar coordinates.

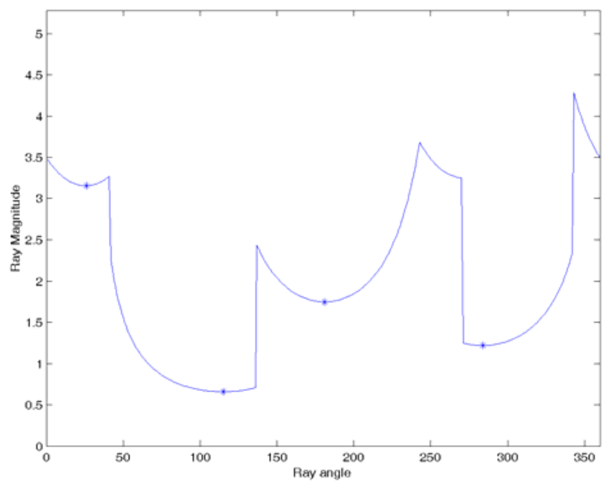


Fig. 4. Ray magnitudes plotted in Cartesian coordinates in which the local minima correspond to shortest distances from visible obstacle edges.

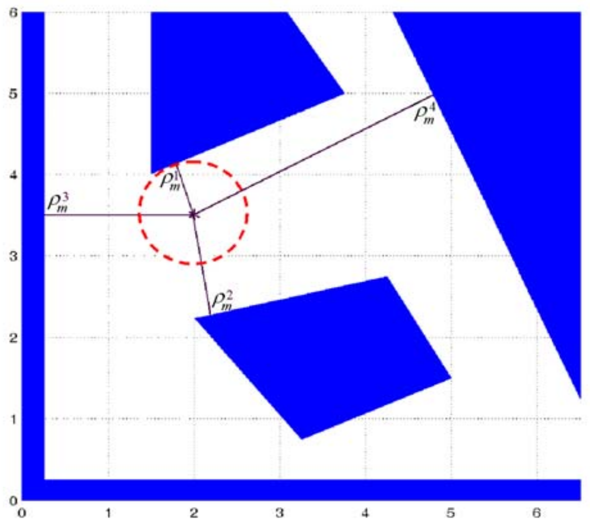
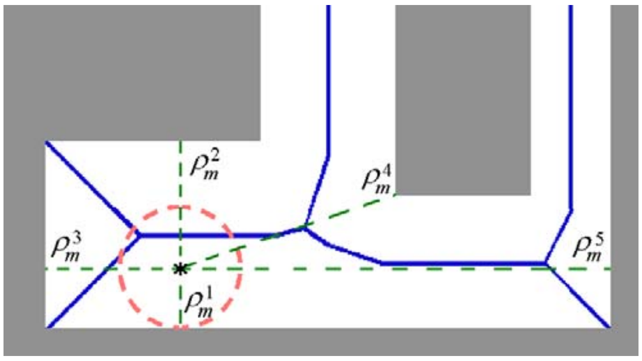


Fig. 5. Shortest distances from different convex obstacles. The minimal ray represents the radius of the Maximal Inscribed Disc (MID).

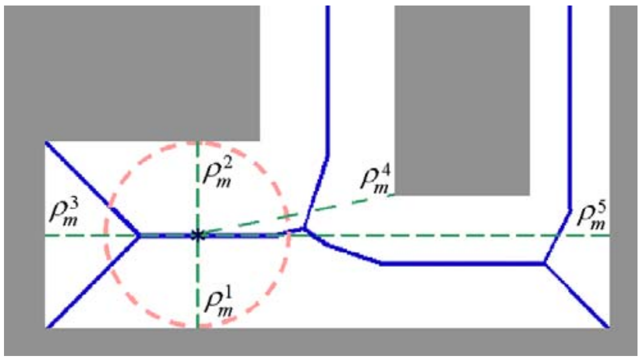
Case 3: $\rho_m^{\ 1} = \rho_m^{\ 2} = \rho_m^{\ 3} << \rho_m^{\ 4}, \ldots, \rho_m^{\ s}$:

This case indicates that the robot is located on a Medial Axis vertex (i.e. meet point) as in Fig. 6(c). Again we can define a tolerance value for real-world applications as $|\rho_m^{\ 1} - \rho_m^{\ 2}| \le \varepsilon$ and $|\rho_m^{\ 1} - \rho_m^{\ 3}| \le \varepsilon$.

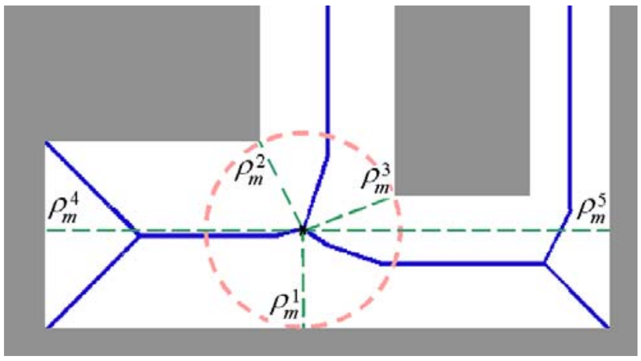
The above conditions determine the location of the robot in relation to the Medial Axis roadmap and the surrounding obstacles, based on which the next step to be taken by the robot is determined.



(a)



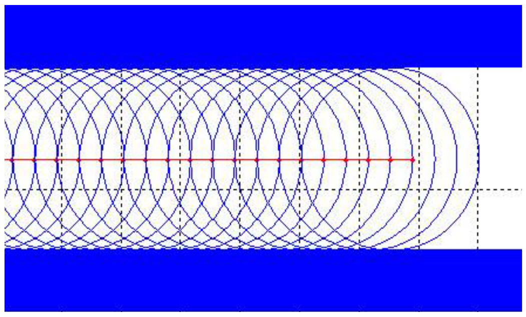
(b)



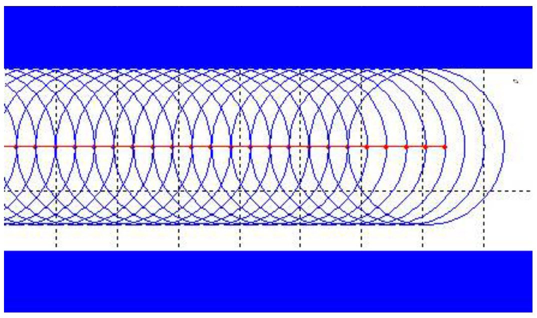
(c)

Fig. 6. The robot performing a distance check: Medial Axis, Maximal Inscribed Disc, and shortest rays are shown. (a) Case 1: $\rho_m^1 << \rho_m^2, \ldots, \rho_m^5$. (b) Case 2: $\rho_m^1 = \rho_m^2 << \rho_m^3, \ldots, \rho_m^5$. (c) Case 3: $\rho_m^1 = \rho_m^2 = \rho_m^3 << \rho_m^4, \rho_m^5$.

Setting the tolerance value ε properly is of great importance since determining the occurred cases depends on it directly. Generally, the smaller is the ε, the more precise would be the robot's motions. This means that the robot's path matches with the actual Medial Axis more precisely, but at the cost of more computations due to smaller step size and thus a slower motion (Fig. 7(a)). Contrarily, a large ε would result in generation of a 'thicker' and roughly-approximated medial axis; that is, a large set of points would be falsely considered as Medial Axis points although the computation cost is lower (Fig. 7(b)).



(a)



(b)

Fig. 7. Moving in a passageway with (a) small tolerance value ε, and (b) large tolerance value ε. Note the loss of precision and noncompliance of the robot's motion to the real Medial Axis in (b), which theoretically lies exactly in the middle of the channel.

2.2. Departure Component

This component is activated when after performing a visibility check, the robot finds out that the Goal point lies inside its visibility envelope. In that case, the robot's current position is connected to the Goal point via a straight line, as in Fig. 8. New trajectory points are then created by an interpolation of the connecting line using a step size of λ. The component's name, i.e. *departure* is adopted from the fact that the robot generally departs from the Medial Axis to reach the Goal.

2.3. Projection Component

The *projection component* is the propelling element of the motion planning algorithm. Each time activated, this routine expands the robot's trajectory by a small step. The size and direction of this step is governed by a vector projected from the current point.

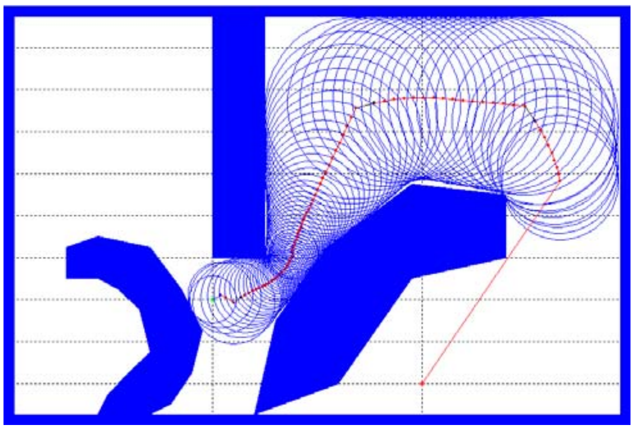


Fig. 8. Once the goal is visible to the robot, the Departure component is activated and the robot's position is connected to the goal point via a straight line. The circles show the Maximal Inscribed Discs centered at points on the Medial Axis.

This component is activated when it is verified that the robot is placed on the Medial Axis. Depending on that the robot is on a MA *edge* or *vertex*, two methods are utilized:

(*i*) If the robot is located on a Medial Axis *edge* (e.g. on the point x in Fig. 9), it has exactly two obstacles nearest to it.

Recalling the mathematical notation presented earlier in this section, suppose that the rays corresponding to the shortest distances from different obstacles are $\{\rho_m{}^1, \rho_m{}^2\}$ which intersect with the obstacle borders in points j_1 and j_2. Then the *minimum vectors* $\{\tilde{\rho}_m{}^1, \tilde{\rho}_m{}^2\}$ are vectors having their start point on $x \in MA$ and endpoints $\{j_1, j_2\}$ on obstacle borders, respectively (Fig. 9).

We define two *projection vectors* as follows:

$$\tilde{\mathbf{v}}_1 = \tilde{\rho}_m^1 + \tilde{\rho}_m^2,\ \tilde{\mathbf{v}}_2 = -(\tilde{\rho}_m^1 + \tilde{\rho}_m^2),$$
$$\|\tilde{\mathbf{v}}_1\| = \|\tilde{\mathbf{v}}_2\| = \lambda$$

in which λ is a scalar indicating the magnitude of the vectors. The size of λ must be selected in a way that it neither decelerates the robot's navigation improperly nor causes missing a meet point.

These two projection vectors point to opposite directions. However, the more promising direction is of course the one which will make a forward move, rather than a backward one. The robot then takes a step along the promising direction, with a size of λ.

(*ii*) If the robot is located on a Medial Axis *vertex* (e.g. on point x in Fig. 10), then it has more than two obstacles nearest to it. By using the notation of the previous case, we define the projection vectors as follows:

$$\tilde{\mathbf{v}}_1 = \tilde{\rho}_m^1 + \tilde{\rho}_m^2,\quad \tilde{\mathbf{v}}_2 = \tilde{\rho}_m^2 + \tilde{\rho}_m^3,\quad \tilde{\mathbf{v}}_3 = \tilde{\rho}_m^3 + \tilde{\rho}_m^1,$$
$$\|\tilde{\mathbf{v}}_1\| = \|\tilde{\mathbf{v}}_2\| = \|\tilde{\mathbf{v}}_3\| = \lambda.$$

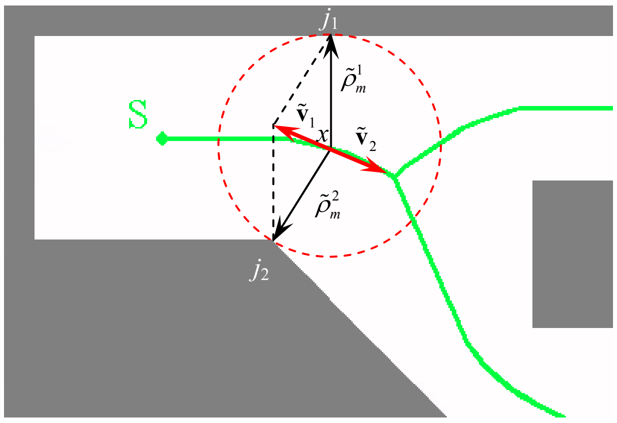


Fig. 9. The projection component provides the robot's movements by computing the most promising vector. Here the robot is on the *edge* of the Medial Axis.

This concept can straightforwardly be generalized to vertices conjoining more than three edges. Among these projection vectors, a vector which does not produce a backward movement, is along a previously unexplored edge, and has an endpoint nearer to the Goal point is considered as 'more promising'. If the location of the Goal is unknown, the vector is selected randomly from among unexplored, non-backward vectors. The robot then moves in the direction of the selected vector with a step size of λ.

The step taken by the robot in the projection component may lead the robot away from the Medial Axis. For example, in Fig. 10, if the robot decides to move along $\tilde{\mathbf{v}}_3$, it will be off the roadmap. The online motion planner responds to this situation by employing the Correction Component.

Figures 11(a) and 11(b) show the Projection Step during some simulations.

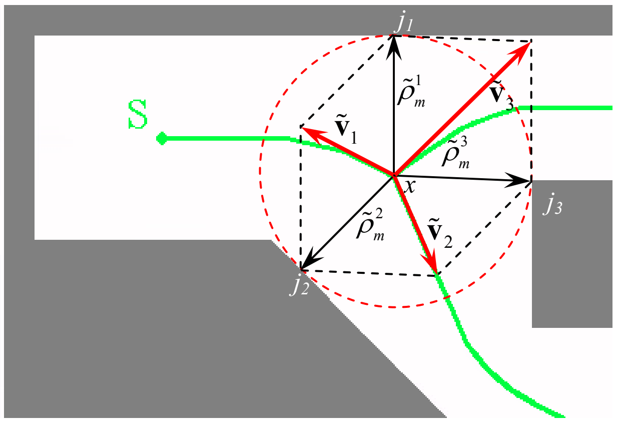
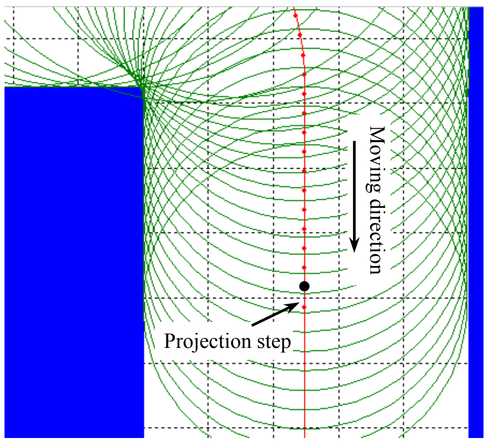
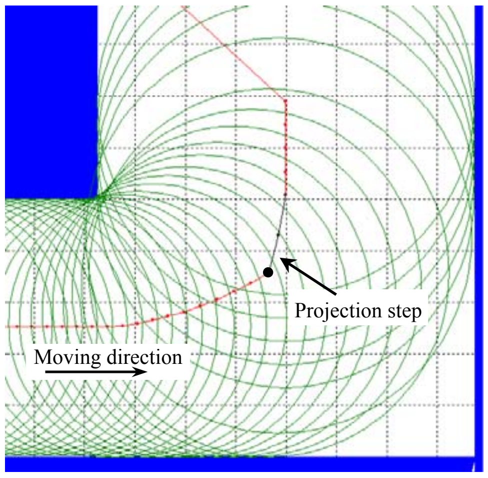


Fig. 10. The projection vectors are the sums of minimum vectors. Here the robot is on the vertex of the Medial Axis, and the most promising vector is the nearest to the goal.



(a)



(b)

Fig. 11. Simulations of the projection step: (a) the robot is on the edge of the Medial Axis, and (b) the robot is on the vertex of the Medial Axis.

2.4. Correction Component

This component guarantees that the robot remains on the Medial Axis roadmap as long as it has not reached the Goal. As mentioned earlier, the projection component may produce steps which despite having been taken along promising directions lead the robot off the main roadmap. This component may also be activated at the very beginning if the robot is not located on the Medial Axis initially.

The mechanism of the correction step is again based on vectors. Suppose that the robot has taken a step in Fig. 12, calculated in the projection component. After performing a distance checking, the robot realizes that it is off-the-roadmap (i.e. Case 1). The correction component is then invoked to correct this situation and transfer the robot to a more stable status (i.e. either Case 2 or Case 3).

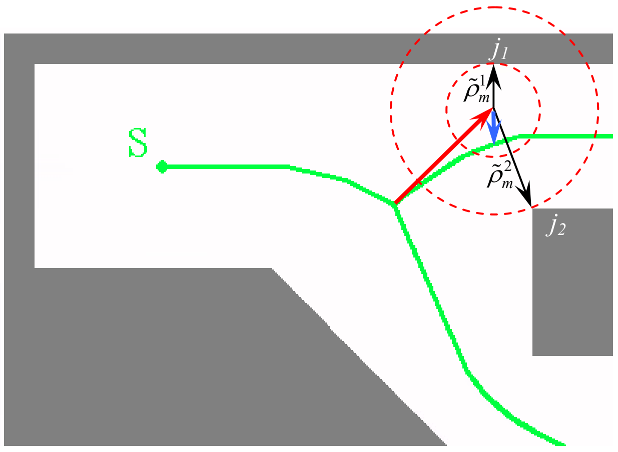


Fig. 12. The Correction step (downward arrow) adjusts the robot's off-the-roadmap status and relocates it on the Medial Axis.

The correction step is taken along a vector with the direction of $-\tilde{\rho}_m{}^1$. In order to calculate the vector's size, suppose that the robot is located on a point near an obstacle border as x in Fig. 13. The required step size λ_c for situating the robot on the Medial Axis is computed as follows:

$$\lambda_c = \left\|\tilde{\rho}_m^2\right\| - \frac{\left\|\tilde{\rho}_m^2\right\| + \left\|\tilde{\rho}_m^1\right\|}{2} = \frac{\left\|\tilde{\rho}_m^2\right\| - \left\|\tilde{\rho}_m^1\right\|}{2}.$$

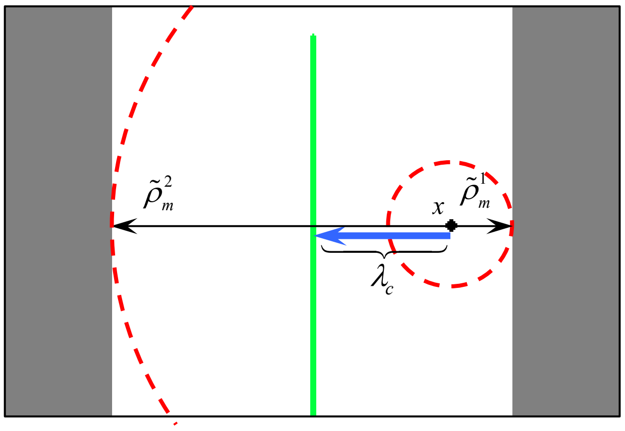


Fig. 13. The size of the correction step λ_c is calculated such that the robot's position will transferred to lie on the Medial Axis roadmap.

For general cases, where the obstacle walls are not parallel, this step size also provides a good approximation for the deviation error. The correction vector is shown in Fig. 12 as a downward arrow. Another instance of the Correction step in a simulation is illustrated in Fig. 14.

If after this correction the robot still does not lie on the roadmap, the Case 1 is not resolved yet. Consequently, correction must be repeated until the situation is transferred to Case 2 or 3.

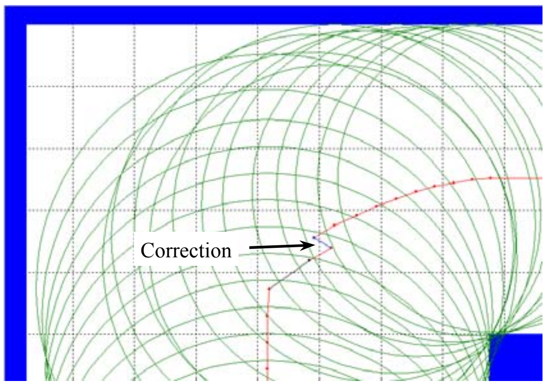


Fig. 14. A correction step activated during a simulation.

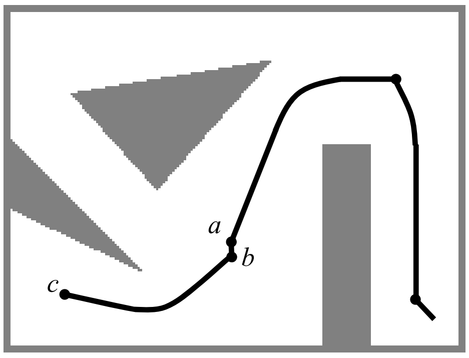
2.5. Backtracking Component

This component is triggered when the robot reaches an obstacle boundary or a dead-end area. In order to prevent the robot from colliding with obstacles, we define the safety radius r_s as the least permissible distance to obstacles boundaries. If during edge exploration, the condition $\rho_m^{\ 1} \leq r_s$ is satisfied, the robot collides with an obstacle. As a result, the Backtracking component is activated to trace back the non-promising edge by selecting a new edge emanated from the last visited vertex, and extending that edge. A sample demonstration of the process is depicted in Fig. 15.

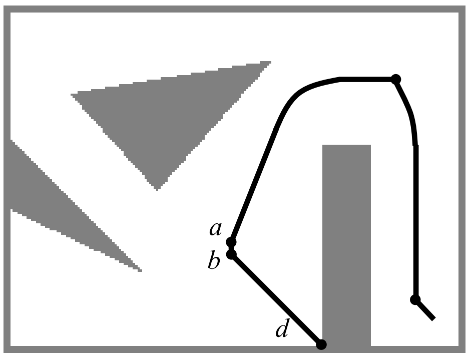
The Backtracking step does not apply projection-correction steps for excluding 'wrong' points of trajectory; rather, it simply removes those points from the existing trajectory points list. A simulation of the Backtracking movement is illustrated in Fig. 16.

2.6. Loop Handling Component

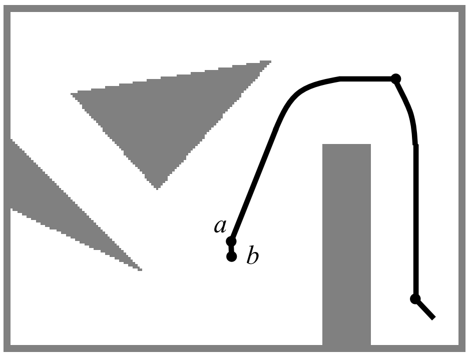
Although the projection vectors introduced in Section 2.3 stipulate that the robot must avoid traveling along already explored edges, it is possible that during the course of navigation the robot encounters a previously-visited vertex of the roadmap. The robot becomes aware of such an event by means of integrating its prior information on that point with the Odometry information and the elements of the shortest rays' matrix, $PM(x)$, through sensor readings. Accordingly, the robot can infer that a loop exists in its trajectory.



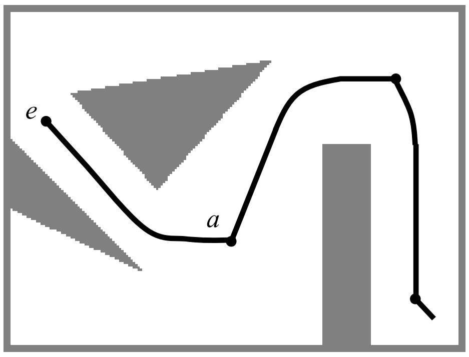
(a)



(b)



(c)



(d)

Fig. 15. The Backtracking component traces back the collision-leading edges of the traversed trajectory. (a) The point *c* leads to a dead-end area, so the robot traces back the edge *cb* until it reaches the vertex *b*. (b) Another edge originated from vertex *b* is explored until point *d*, which is near to the obstacle boundary. (c) The edge *db* is backtracked, and since all edges of the vertex *b* are explored, the robot moves back to the last vertex reached before *b* (i.e. vertex *a*) by removing the edge *ba*. (d) A new edge *ae* is extended from the vertex *a*.

Recall that the robot employs a depth-first search method to explore the Medial Axis roadmap. The backtracking component also traces back the trajectory in a reverse depth-first manner. Therefore, in order to make the robot to investigate 'unexplored' areas of the workspace, as well as to resolve the generated loop, it is rational to adopt a breadth-first strategy upon reaching a previously-visited vertex.

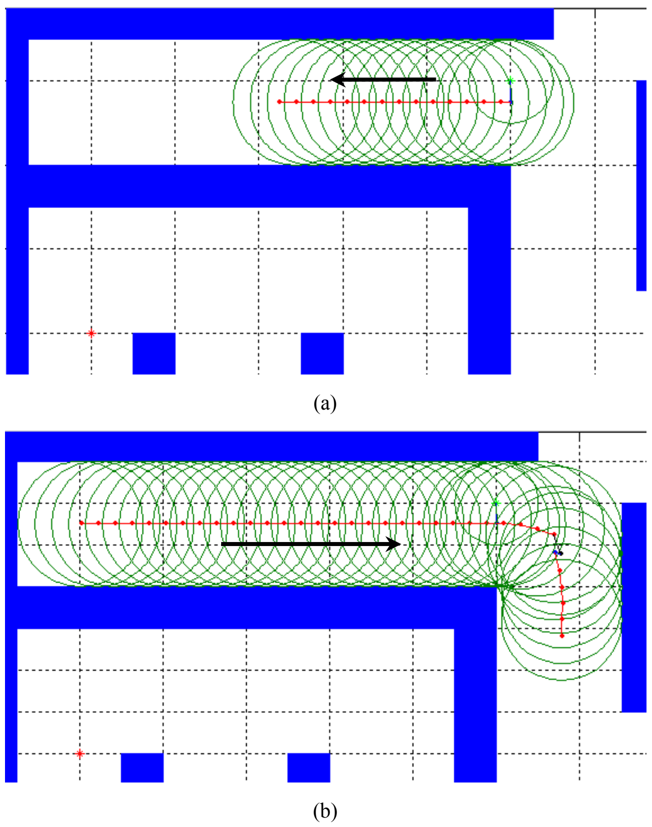


Fig. 16. Simulation of the Backtracking behavior. (a) The robot moves to the left from its start position. (b) After encountering a dead-end, backtracking to the right helps the robot to exit from the corridor.

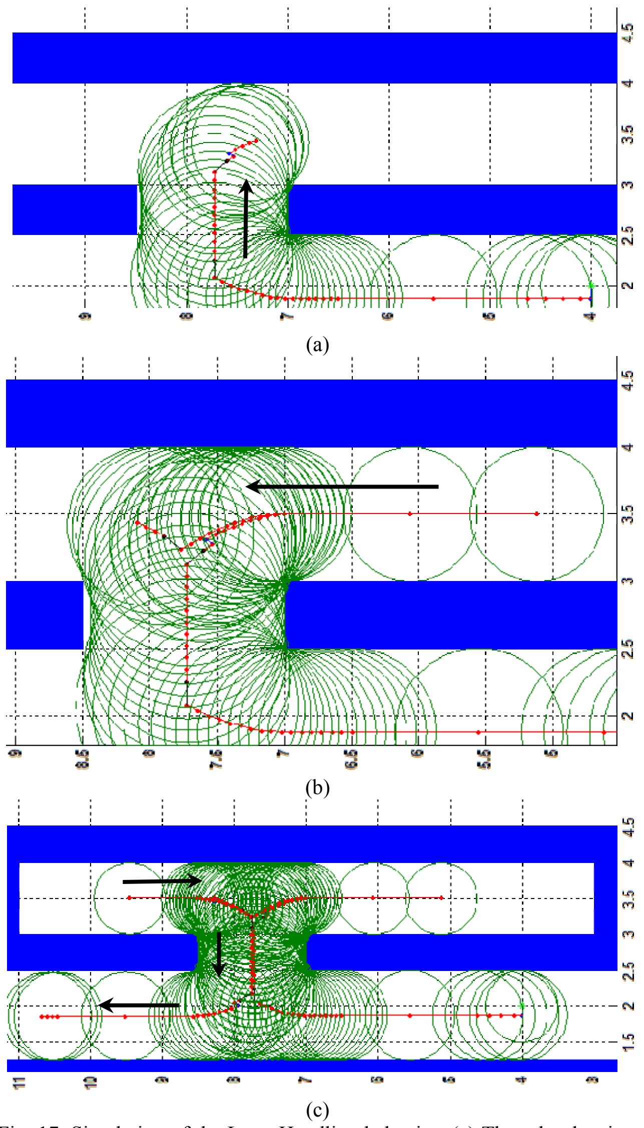


Fig. 17. Simulation of the Loop Handling behavior: (a) The robot has just passed a meet point and turns to the right. (b) After realizing that the right corridor is dead-end, the robot backtracks and after passing the meet point, selects the leftward edge to traverse. (c) The left corridor also proved to be dead-end, and so the robot backtracks toward the first meet point it encountered (at the bottom) and continues with an untraversed edge. The robot's movement is similar to a reverse depth-first search.

To apply the breadth-first search approach, the robot tries to move along an unexplored edge of the revisited vertex. This action guides the robot towards previously unexplored areas of the environment. If the current vertex does not have such edges, the robot will move towards the neighboring vertex it visited last time after the current vertex (i.e. according to its last itinerary). This operation is repeated for the new vertex (or vertices) until the robot moves along an edge outside the loop.

The reason for naming this procedure as 'breadth-first search' is that the robot tries to expand all potential edges emanated from a revisited vertex (node) but not to probe deeply into successors of the node.

As soon as a breadth-first movement is completed, the search strategy shifts to the depth-first approach, mainly to maintain the algorithm's convergence toward the Goal point. Fig. 17 illustrates a simulation of this component.

The Loop handling component guarantees a thorough coverage of the Medial Axis roadmap and prevents the robot from rambling along endless loops.

3. Algorithm Steps

The main steps of the online motion planning algorithm are now presented based on the notation introduced earlier:

STEP 1. Locate the robot on the Start point, s. Initialize the trajectory points set as:

$$Traj \leftarrow \{s\}.$$

Also set $T = \varnothing$, $T_V = \varnothing$, and $DE = \varnothing$. These sets are updated later.

STEP 2. Perform the Distance Checking component according to Section 2.1 to determine the sets $P(x)$ and $P_M(x)$ (x denotes the current location of the robot). If the

Goal point is visible (i.e. is in the robot's line of sight), then go to Step 8. Otherwise, continue with the next step.

STEP 3. Compare the values of the elements of $P_M(x) = \{\rho_m^1, \rho_m^2, \ldots, \rho_m^s\}$:

3.1 If $\rho_m^1 << \rho_m^2, \ldots, \rho_m^s$ (i.e. Case 1 holds), the robot is off the Medial Axis roadmap. This situation must be corrected by taking Step 7.

3.2 If $\rho_m^1 = \rho_m^2 << \rho_m^3, \ldots, \rho_m^s$ (i.e. Case 2 holds), the robot is located on the Medial Axis edge.

If $\rho_m^1 \leq r_s$ (the safety radius), go to Step 4. Unless, update the trajectory matrix by appending the robot's current position (x) by

$$Traj \leftarrow Traj \cup \{x\}.$$

And continue the navigation along the roadmap according to Step 6.

3.3 If $\rho_m^1 = \rho_m^2 = \rho_m^3 << \rho_m^4, \ldots, \rho_m^s$ (i.e. Case 3 holds), the robot is located on a Medial Axis vertex (meet point).

If the current meet point is recorded in the matrix T as a previously visited point, a loop has been detected in the course of navigation. So go to step 5.

If not, append the robot's current position (x) to the set of trajectory points already traversed by the robot, *Traj*:

$$Traj \leftarrow Traj \cup \{x\}.$$

And update the following matrices:

$$T \leftarrow T \cup \{(x, |J(x)|)\}, J(x) = \{\mathrm{MID}(x) \cap OB\}.$$

$J(x)$ is the set of points on Obstacle Boundary (OB) which are also located on the Maximal Inscribed Disc centered on x. In other words, $J(x)$ is the set of endpoints of the rays $\rho_m^1, \rho_m^2, \ldots, \rho_m^s$ (See Figures 9 and 10). T is the set of ordered pairs of meet points and the number of elements in their respective $J(x)$. The symbol $|\bullet|$ shows the cardinality of the set. For instance, $T = \{(x_1, 3), (x_2, 3), (x_3, 4)\}$. The next point to move towards is determined in Step 6.

STEP 4. *Backtracking step*: This step removes the trajectory points extended toward a dead-end or obstacle boundary (e.g. a corner). It is performed also when there are no extendable edges from a vertex. From the last point of the matrix *Traj* eliminate all points preceding it until the last visited vertex is reached. Add all the eliminated points to the Dead Ends matrix, *DE*. In fact, New(*Traj*) = Old(*Traj*) \ *DE*. Mark the removed edge so that it will not be explored again. Then go to Step 3.3.

STEP 5. *Loop Handling step*: This step imposes a change in search strategy from depth-first to breadth-first search. Append the revisited meet point to the matrix of revisited meet points, T_V, and try to explore new edges emanated from it. If it is not possible, move to the very vertex you visited last time after the current meet point, and select a new edge to explore. Go to Step 6 for extending this new edge.

STEP 6. *Projection step*: Select a vector and take a step in its direction with considering the forbidden directions determined by the facts that the robot's next move:

- should not be backward (except in the Backtracking step),
- should not be along an already explored edge (except in the Loop Handling step), and
- should be in the most promising direction.

If the robot is placed on a Medial Axis edge, follow the procedure in Section 2.2(*i*). But, if it is on a vertex, perform the instructions in Section 2.2(*ii*), and then go to the next step.

STEP 7. *Correction step*: This step is taken when it is identified that the robot does not lie on the roadmap. Perform the correction according to Section 2.4, and then update the matrix $T \leftarrow T \cup \{(y, |J(y)|)\}$. Go to Step 2.

STEP 8. *Departure step*: This step connects the current point to the visible Goal point. Through a linear interpolation, decompose the connecting straight line into several intervals (by interpolation) and append the endpoints of those pieces to the Trajectory. Upon reaching the Goal, report the *Traj* as the solution path, and terminate the algorithm.

Fig. 18 shows the flowchart of the online motion planner.

4. Experimentation and Discussion

Fig. 19 illustrates some simulations of the developed online motion planner. Fig. 19(a) clearly indicates the algorithm's completeness, and shows that it explores every corridor of the workspace. Fig. 19(b) shows the effectiveness of the loop handling component as it helps the robot to move out of the circular corridor, and Fig. 19(c) depicts the process of finding a start-to-goal path in 24.3 seconds. All simulations were programmed and run in Matlab software.

Below, some discussions on the algorithm's properties including its completeness and complexity are provided.

4.1. Completeness

An interesting property of our online planning method is its *completeness*, that is, it has the ability to find the path to goal if it is reachable, or report that no such a path exists.

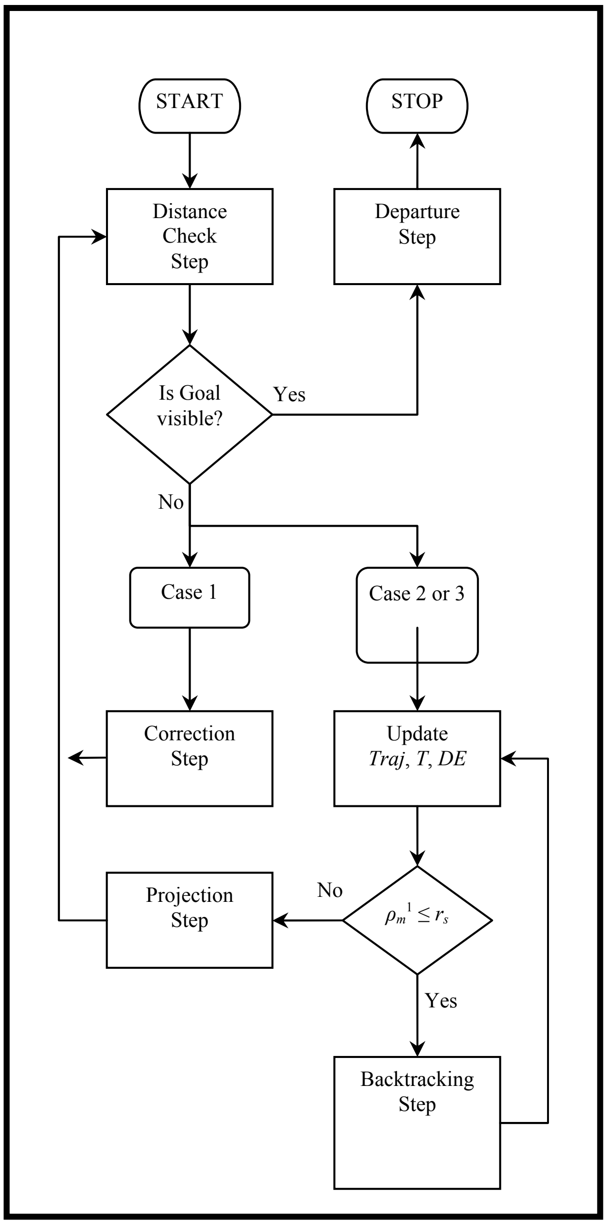


Fig. 18. Flowchart of the online motion planner

Since the method utilizes a *retraction* roadmap of the environment (Medial Axis) which is connected (as proved in (Rao et al. [24]), any path that contains the MA roadmap and links to start and goal points is also connected. Therefore, the robot will reach the goal if it lies in C_{free}. This guarantees that the algorithm is *exact* (Goldberg, [13]).

In order to provide the condition for *completeness*, the following question is to be answered: "will the algorithm show that a path to the goal does not exist if it is the case?" To answer, recall the Step 5 of the algorithm in Section 3 where a matrix T_V keeps the record of revisited meet points. If this matrix includes all meet points before the goal is reached, the robot can infer that there is no valid path toward the goal since no unexplored meet point remains and the workspace is searched completely. If this happens, the algorithm will terminate and report that there is no path from start to goal.

4.2. Localization

The robot's localization is done by a combination of information of odometry (wheel accelerometers, etc.) and sensor readings information about the Maximal Inscribed Discs. Since all the visited meet points and the number of unexplored edges emanating from them are stored in the matrix *T*, the meet point locations are constantly being compared and matched with previous data. Therefore, the localization errors are dynamically corrected and do not accumulate.

4.3. Extension to Higher Dimensions

The online planner can be generalized to higher spaces by taking advantage of the properties of GVG roadmap in higher dimensional spaces, and extending the MID to higher spaces which turn into Maximal Inscribed Balls (MIBs).

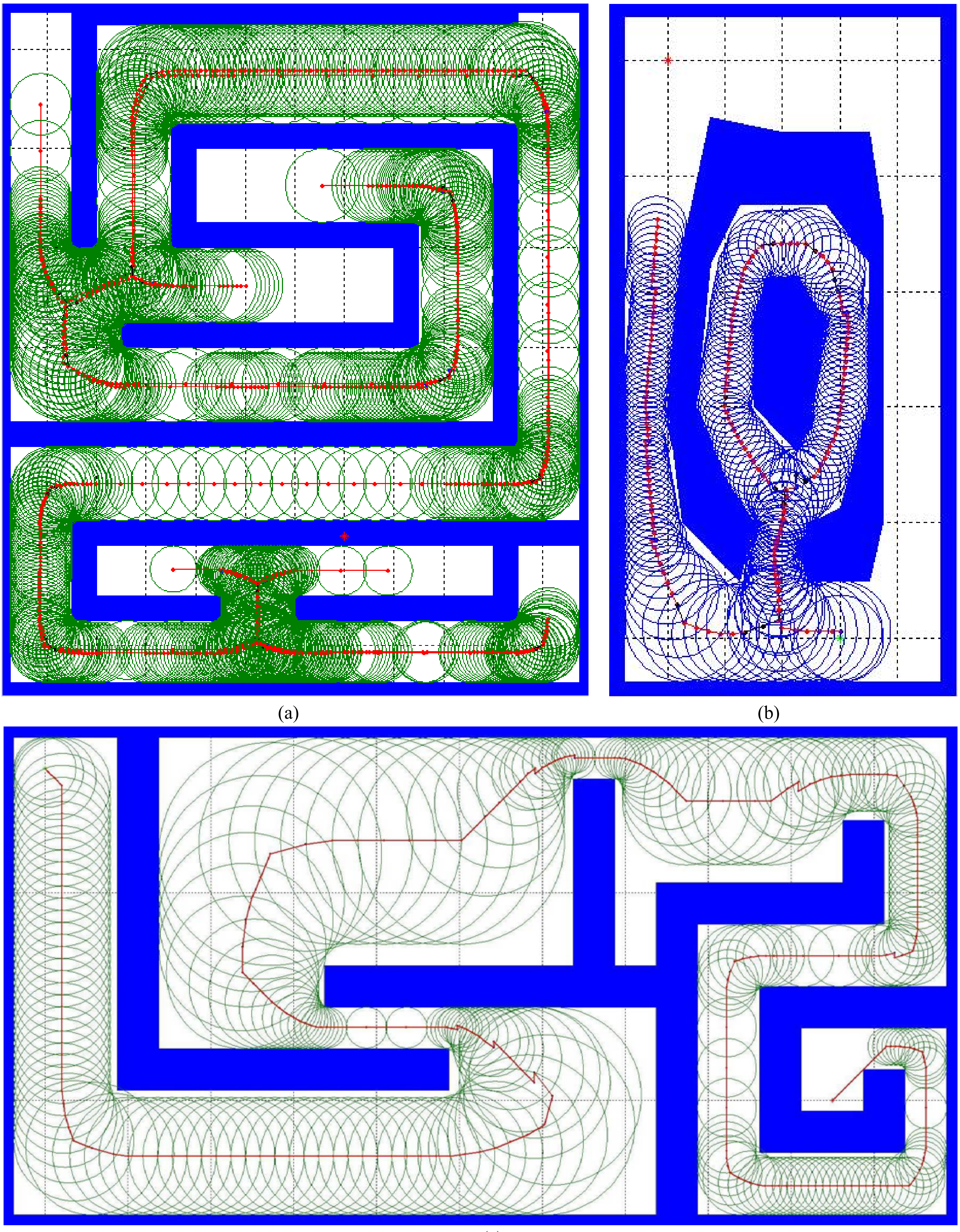
4.4. Time Complexity

The time complexity of the planner is directly related to the number of edges it should traverse and is equal to $O(n)$, n being the number of obstacle vertices (Aurenhammer and Klein [1]). The time required to navigate an edge is proportional to the length of the edge and is independent of the number of vertices. Considering that the algorithm is complete and in the worst case would traverse all the Voronoi graph's edges twice (one for traversing a new edge and one for backtracking it), the planner's overall time complexity is in $O(n)$.

The space required to store control matrices (e.g. *T*) is related to the number of vertices which is in $O(n)$.

4.5. A Potential Fields Variation

When knowledge of the location of Goal point is available at the outset, the proposed motion planner can be made more efficient by integrating the Potential Fields approach into it.



(a) (b)

(c)

Fig. 19. Some simulations of the new motion planner.

The Potential Fields concept was first introduced by Oussama Khatib [16], and has shown a good performance especially for higher dimensions. A robot in the Potential Fields method is represented in the configuration space as a particle *q* (with a positive charge) under the influence of an artificial potential field *U* whose local variations reflect the 'structure' of the free space. The total potential function (Fig. 20(c)) can be defined over the free space as the sum of an attractive (negative) potential (as in equation (1) and Fig. 20(a)) which pulls the robot toward the goal configuration, and repulsive (positive) potentials (as in equation (2) and Fig. 20(b)) which push the robot away from the obstacles.

The attractive potential applied to the goal point is generally in the form of a paraboloid (as in (1)) and facilitates the departure phase.

$$U_{att}(q) = \tfrac{1}{2}\xi\,\rho^{2}_{goal}(q) \tag{1}$$

$$U_{rep}(q) = \begin{cases} \dfrac{1}{2}\eta\left(\dfrac{1}{\rho(q)} - \dfrac{1}{\rho_0}\right)^2 & \text{if } \rho(q) \le \rho_0, \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases} \tag{2}$$

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{3}$$

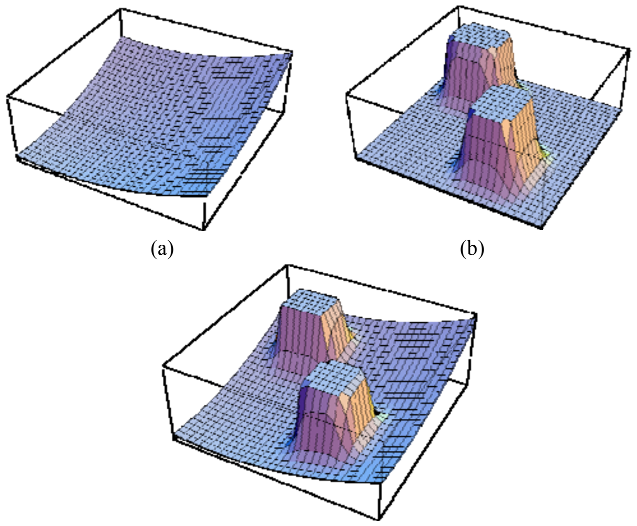

(a) (b) (c)

Fig. 20. Potential Fields. (a) The goal potential has a unique minimum at the goal point. (b) The obstacle potential has a high value inside obstacles. (c) A path to the goal can be found from the start point by moving in the direction of the negative gradient of the combined total potential field.

The proposed new algorithm can be integrated with the potential Fields concept readily, that is, the principal steps for this variation remain similar to the one described in Section 3 except for the following modifications:

In the Projection component, among the alternative vectors, the one with the least potential is selected as the next and promising edge to be explored. When a new vertex is reached, its potential value is compared with the potential value of its preceding vertex. If a higher (worse) potential is recorded, instead of further exploration of the new vertex (i.e. a depth-first approach), the robot backtracks the last traversed edge and investigates a new edge associated with the preceding low-potential vertex (i.e. a breadth-first approach).

This variation has the advantage that the repulsive potential of obstacles discourages the robot to move towards the vicinity of their boundary, and in turn leads it toward the goal point.

5. Conclusions

In this paper, a new online robot motion planner for systematically exploring unknown environments in real-time applications based on the Generalized Voronoi Graph (or Medial Axis) of the environment is proposed. The algorithm takes advantage of sensory data to find an obstacle-free start-to-goal path. It does so by utilizing a combination of depth-first and breadth-first searches.

The planner is equipped with several components such as step generation and correction, backtracking, and loop handling. It is fast, simple, complete, and extendable to higher spaces. Besides, a variation to the online planner which incorporates the Potential Fields approach in the principal algorithm is discussed. This method has some advantages when the Goal position is known beforehand.

Acknowledgements

The authors wish to thank Mr. Behrooz Bakhtiari for his efforts to program the method in Matlab software.

References

[1] F. Aurenhammer, R. Klein, Voronoi diagrams, in Handbook of Computational Geometry, J. Sack, & G. Urrutia, (Eds.), Elsevier, pp. 201-290, 2000.

[2] H. Blum, A Transformation for Extracting New Descriptors of Shape, In W. Waltendunn, (Eds.), Models for the Perception of Speech and Visual Form, MIT Press, 1967.

[3] J. Borenstein, H. R. Everett, and L. Feng, Where am I? Sensors and Methods for Mobile Robot Positioning, University of Michigan, 1996.

[4] A. Brooks, T. Kaupp, and A. Makarenko, Randomised MPC-based motion-planning for mobile robot obstacle avoidance Source. Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 397-402, 2009.

[5] F. Chin, J. Snoeyink, and C. Wang, Finding the Medial Axis of a Simple Polygon in Linear Time. Proceedings of the 6th Annual International Symposium on Algorithms and Computation (ISAAC '95), Lecture Notes in Computer Science 1004, 382-391, 1995.

[6] H. Choi, S. Choi, and H. Moon, Mathematical Theory of Medial Axis Transform. Pacific Journal of Mathematics, 181(1), 57-88, 1997.

[7] H. Choset, J. Burdick, Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph. International Journal of Robotics Research, 19(2), 96-125, 2000.

[8] H. Choset, I. Konukseven, and A. Rizzi, Sensor Based Planning: A Control Law for Generating the Generalized Voronoi Graph. Proceedings of the 8th International Conference on Advanced Robotics (ICAR), 333-338, 1997.

[9] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, Sensor-Based Exploration: Incremental Construction of the Hierarchical Generalized Voronoi Graph. International Journal of Robotics Research, 19(2), 126-148, 2000.

[10] J. Cox, C. K. Yap, On-line motion planning: Moving a planar arm by probing an unknown environment. Technical report, Courant Institute of Mathematical Sciences, New York University, New York, July 1988.

[11] J. Dardenne, S. Valette, N. Siauve, and R. Prost, Medial Axis Approximation with Constrained Centroidal Voronoi Diagrams On Discrete Data. Proceedings of the 26th Computer Graphics International Conference (CGI 2008), Istanbul, Turkey, June 2008.

[12] A. Datta, Constant-Time Algorithm for Medial Axis Transform on the Reconfigurable Mesh. Proceedings of the 13th International. Parallel Processing Symposium and the 10th Symposium on Parallel and Distributed Processing, Puerto Rico, IEEE Computer Society, 431-435, 1999.

[13] K. Goldberg, Completeness in Robot Motion Planning", in Algorithmic Foundations of Robotics, K. Goldberg, D. Halperin, J-C. Latombe, and R. Willson (Eds.), A. K. Peters Pub., pp. 419-429, 1995.

[14] R. Johansson, Intelligent Motion Planning for a Multi-Robot System. MS Thesis, School of Computer Science and Engineering, Royal Institute of Technology, Stockholm, Sweden, 2000.

[15] L. E. Kavraki, J-C. Latombe, "Probabilistic Roadmaps for Robot Path Planning", in Practical Motion Planning in Robotics: Current Approaches and Future Challenges, K. Gupta and A.P. del Pobil, (Eds.), John Wiley, West Sussex, England, pp. 33-53, 1998.

[16] O. Khatib, Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. International Journal of Robotics Research, 5(1), 90-98, 1986.

[17] J. Leonard, H. F. Durrant-Whyte, Application of Multi-Target Tracking to Sonar-Based Mobile Robot Navigation. Proceedings of the IEEE 29th International Conference on Decision and Control, Hawaii, USA, 3118-3123, 1990.

[18] V. J. Lumelsky, A. A. Stepanov, Dynamic path planning for a mobile automaton with limited information on the environment. IEEE Transactions on Automatic Control, 31(11), 1058-1063, 1986.

[19] E. Masehian, M. R. Amin-Naseri, and S. Esmaeilzadeh Khadem, Online Motion Planning Using Incremental Construction of Medial Axis. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Taiwan, 2928-2933, 2003.

[20] M. Petřek1, P. Košinová, J. Koča1, and M. Otyepka, MOLE: A Voronoi Diagram-Based Explorer of Molecular Channels, Pores, and Tunnels. Structure, 15(11), 1357-1363, 2007.

[21] J. H. Pujar, P. S. Gurjal, Binary Data Compression Using Medial Axis Transform Algorithm. Proceedings of International Conference on Recent Trends in Business Administration and Information Processing, Kerala, India, March 2010.

[22] N. S. V. Rao, Robot navigation in unknown generalized polygonal terrains using vision sensors. IEEE Transactions on Systems, Man and Cybernetics, 25(6), 947-962, 1995.

[23] 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, July 1993.

[24] N. S. V. Rao, N. Stolzfus, and S. S. Iyengar, A 'Retraction' method for Learned Navigation in Unknown Terrains for a Circular Robot. IEEE Transaction on Robotics and Automation, 7(5), 699-707, 1991.

[25] S. Sam Ge, X. C. Laia, and A. Al Mamuna, Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints. Robotics and Autonomous Systems, 55(7), 513-526, 2007.

[26] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, MAPRM: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space. Proceedings of the IEEE International Conference on Robotics and Automation, 2, 1024-1031, 1999.