Boundary Following and Globally Convergent Path Planning Using Instant Goals

Shuzhi Sam Ge, *Senior Member, IEEE,* Xue-Cheng Lai, Abdullah Al Mamun, *Senior Member, IEEE*

***Abstract*— In this paper, an Instant Goal approach is proposed for collision-free boundary following of obstacles of arbitrary shape and globally convergent path planning in unknown environments. Firstly, for effective knowledge representation and manipulation, a vector representation is presented, which not only saves much space but also conforms to the physical properties of range sensors. Secondly, the concept of instant goals is introduced enabling the robot to perform boundary following in a "natural" human-like manner, with additional measures taken to ensure that the robot is moving "forward" along the boundary, even if the obstacle is of arbitrary shape and disturbing obstacles are present. Collision checking is performed simultaneously and, when needed, collision avoidance is efficiently incorporated in. Based on the approach of boundary following, a realistic sensor-based path planner with global convergence property is designed for the robot capable of acquiring discrete, and noisy range data. Realistic simulation experiments validate the effectiveness of the proposed approaches.**

***Index Terms*— Mobile robots, boundary following, Instant Goal, sensor-based path planning, global convergence**

I. Introduction

In reality, motion planning of a robot often cannot be based on complete *a priori* knowledge of the environment. Sensors are employed to sense its surrounding environment and most of them are subject to range limitations. As such, there is a need for the robot to continuously sense using its onboard sensors in order to navigate through its environment. Much attention has been paid on sensor-based path planning of mobile robots in unknown, complicated environments. Some researches [1][2] assumed that the robots were able to follow the obstacle boundaries. In order to avoid collision with obstacles, many navigation approaches [3][4] choose the strategy of bypassing the blocking obstacle, which is similar to boundary following.

Wall following, as a popular and useful technique for mobile robot navigation in structured or known environments, has been intensively studied [5]. Providing a good setting for pose prediction and sensor fusion, Kalman filtering is commonly used to produce a good approximation of distance and angle from a known wall. Several constraints were introduced such as velocity and maximum angle deviation from the wall [6]. Considering the fact that any smooth boundary can be partitioned as piecewise linear, based on previous work on wall following, the related problems of boundary following can be solved without any difficulty by traveling in a parallel direction to the curve tangent. However, robots are usually only able to have a discrete approximation of their surroundings, which makes the problem harder. Furthermore, there is no ready solution for a robot to follow arbitrary obstacles with complicated shapes. Moreover, with the presence of nearby obstacles which may disturb the moving direction of the robot, the problem is not just following a curve while keeping a constant distance from it, but also involves making decisions as to which obstacle to follow.

This paper presents an Instant Goal approach which allows a physical robot equipped with a rangefinder to follow an obstacle of arbitrary shape. The "Instant Goal" is a local target specially designed for action planning based on sensory input. The Instant Goal Driven method was first presented in [7] where instant goals are generated at each time instant for behavior-based navigation. In this paper, the proposed approach allows the robot to move *forward* along the boundary of an obstacle without collision in obstacle cluttered environments. To facilitate realtime planning, a vector representation, which saves greatly on memory space, is used to represent the local environment sensed by a rangefinder. The search range for each Instant Goal is restricted to a certain scope determined by previous status and current range data input such that the boundary following is performed in the desired direction. The concepts of "safe path" and "passage" are used to distinguish the "disturbing" obstacles from the desired one such that Instant Goals are able to be determined under any complex obstacle condition.

Sensor-based path planning approaches can be categorized into either global planning or local planning. The global sensor-based planning approaches [8][9][10] build a global world model based on sensory information and use it for path planning. By mapping the entire (accessible) environment, the global approaches can somewhat solve the basic path planning problem[1] which implies the property of "global convergence" of path planning: i) if the goal is reachable, the path planner generates a continuous collision-free path from the start location to the goal; ii) otherwise, the path planner reports the unreachability of the goal and stops the navigation. However, the construction and maintenance of a global map imposes a heavy computational burden on the robot.

In contrast, the local sensor-based path planning approaches use local sensory information in a purely reactive fashion. These methods include the potential field methods [11][3][12][4][13], behavior-based systems [14][15] and fuzzy

S. S. Ge, X. C. Lai and A. A. Mamun are with the Electrical and Computer Engineering Department of the National University of Singapore.

†Corresponding author. Tel.: +65 6874 6821; Fax: +65 6779 1103; Email: elegesz@nus.edu.sg.

[1]Given an initial position and orientation of a robot and the goal position, a path planner will generate a continuous free path starting at the initial position and orientation and terminating at the goal position if such a path exists and report failure otherwise.

logic approaches [16]. The classical potential field methods involve an artificial force acting upon the robot, derived from the vector summation of an attractive force representing the goal and a number of repulsive forces associated with the individual known obstacles. Behavior-based methods decompose the problem of autonomous navigation into independent local behaviors and handles uncertainty and unpredictable changes well, by giving up the idea of modeling and reasoning about the environment and the future consequences of actions. Though they are usually much simpler to implement than the global ones, the local planners may get trapped in a local minimum and subsequently follow a diverging path or a loop while attempting to escape from the local minimum. This makes systems relying solely on them somewhat unreliable.

Bug algorithms, which combine local planning with global information, were proposed to solve the problem of a point mobile automaton moving amidst unknown obstacles of arbitrary shape [1]. Range sensing was subsequently incorporated in without causing the loss of their global convergence property [17]. These algorithms use two reactive modes of motion: moving towards the goal and following an obstacle boundary. In order to guarantee convergence to the goal, the transition between the two modes is governed by a criterion to ensure that the distance to the goal decreases monotonously. To improve navigation performance such as shorter paths, DistBug [2] and TangentBug [18] algorithms adapt the tangent graph to obtain the locally shortest path, define several different transition conditions for switching between the two motion modes, and choose the locally optimal direction for obstacle following. Similar to the Bug algorithms, the subgoal selection algorithm [19] generates a sequence of subgoals near the polygonal vertices, until the obstacle does not block the line-of-sight to the final goal.

Different switching criteria, which lead to the forming of the family of Bug algorithms, have been proposed for transiting between the two motion modes. However, several ideal assumptions are required, such as the capability of obstacle boundary following. We define *globally convergent path planning* as sensor-based path planning which can achieve the property of global convergence without first building the global map of the environment. In this paper, a realistic globally convergent path planner is presented for the navigation of a mobile robot with a rangefinder in an unstructured, complex environment. It uses a Bug-like strategy to switch between the two motion modes in order to ensure the global convergence of the path planner. Rather than assuming the robot is able to follow an obstacle, the path planner is based on the Instant Goal approach which does not require the range sensor to have a perfect sensing ability. Different from subgoals which are simply set to be near the polygon vertices in a polygon environment, the Instant Goals are generated such that the robot is able to follow an obstacle in the desired direction in a complex, obstacle cluttered environment.

The main contributions of this paper are: i) an Instant Goal approach is proposed for collision-free boundary following along an obstacle of arbitrary shape even in the presence of other disturbing obstacles; ii) based on the Instant Goal approach, rather than assuming the capability of boundary following, a realistic globally convergent path planner is presented for navigation in unknown environments; iii) a vector representation for the local environment is presented, which saves much memory space and is suitable for behavior generation and obstacle avoidance; and iv) there is no need for perfect sensing of a range sensor.

The rest of this paper is organized as follows. Section II introduces the representation and modeling of the sensed local environment. In Section III, an Instant Goal method is presented for the generation of local targets which ensures proper obstacle boundary following in cluttered environments. In Section IV, collision checking is introduced which uses potential field approach to guarantee no collision with obstacles. In Section V, based on the Instant Goal approach of boundary following, a realistic globally convergent path planner is presented. In Section VI, we present the simulation results. Finally, the conclusions are presented in Section VII.

II. Representation and Modeling of the Local Environment

Many approaches of modeling have been suggested in the literature for localization and path planning for robotic systems. In reactive motion planning, representations of the local environment should aid the motion determination or the generation of behaviors and satisfy the real-time requirement. The robot has limited memory that only allows it to store a limited number of important points, which makes it often insufficient, for example, for storing incremental maps which may be very large, given that obstacles can be of arbitrary shape. In this section, we propose the use of a simple vector to represent the local environment and an efficient method to model the sensed environment based on range data.

The following provides a list of notations which will be used throughout this paper:

$\overrightarrow{AB}$	straight-line segment starting from point A to point B;
$\overline{AB}$	non-directional straight-line segment with end points A and B;
$\|AB\|$	length of line segment $\overrightarrow{AB}$ or $\overline{AB}$;
$arc\ AB$	arc length of arc AB;
$\mathbf{n}_{AB}$	unit vector pointing from A to B
S, G	initial position of the robot, and the goal;
θ	direction with respect to the body frame attached to the robot;
P, ϑ	position and orientation with respect to the global frame;
Δt	period between the two successive moving actions planned;
t	the current time instant;
P_t, ϑ_t and $\mathbf{V}_t$	position, orientation and velocity of the robot at the time instant t.

A. New Representation of the Local Environment

For convenience, we assume that the mobile robot, denoted as $\mathcal{R}$, moves in a 2D indoor environment:

Assumption 1: The robot navigates on a horizontal, even surface and each obstacle is a simple closed curve of an arbitrary shape and of a finite length.

On-line sensors are utilized by the robot to sense the unknown environment. For an indoor environment, range

finders such as ultrasonics and laser scanner are often used because they can directly measure the ranges of the obstacles. Compared with ultrasonics, laser scanners can provide much better angular resolution and there are less spurious readings. Therefore, a laser scanner is used, and is able to provide N_s equally angular spaced readings for a rotation of 360°. As shown in Fig. 1, the origin of the body coordinate frame is located at the center of the robot and the x_b axis coincides with the orientation of the robot. The detectable area of the sensor is denoted by the dashed circle of radius R_d centered at O_b. The sensed free-space around the robot is denoted by the region encircled by solid dashed lines. The effective size of a robot is defined as the diameter (denoted by $2R_{\text{rob}}$) of the robot body which can be the actual diameter for a circular robot or the diagonal length for a car-like rectangular robot. The N_s beams of range readings are indexed in a counterclockwise manner with the direction of the 1st beam coinciding with the x_b axis.

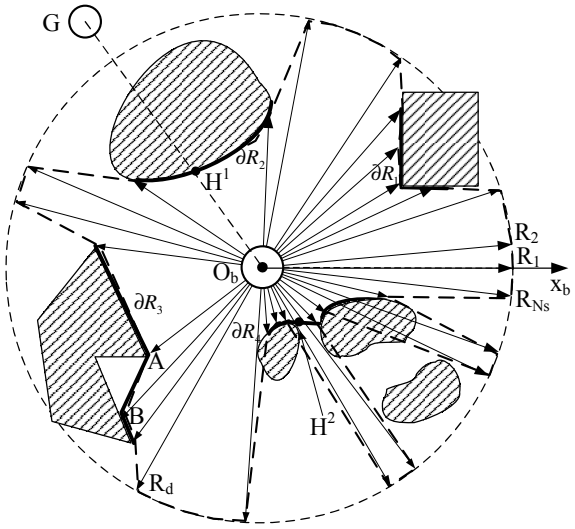


Fig. 1. The obstacles sensed by a rangefinder on the robot.

The 2D rangefinder information obtained in the jth beam direction is represented as pairs

$$p_j = (\rho_j, \theta_j)$$

in polar coordinates with respect to the coordinate frame attached to the laser scanner, where ρ_j is the measured distance of the detected object, and θ_j is the measured azimuth angle between the direction of the beam and the x axis of the reference. If the relationship between the index and beam angle is known for each beam, the measured point can be represented in a more convenient form:

$$p_j = (\rho_j, j) \tag{1}$$

A point on an object (an obstacle or the goal) is *detectable* if and only if it is within the sensor range. A point is *visible* if and only if it is detectable and its viewing line does not intersect any other object(s), where the viewing line is the straight line segment from P to the point. Furthermore, it is known that, if a point is visible from point P, it is also visible from any point between it and P.

Grids are often used to represent the navigable space around the robot due to their simplicity [20]. Grid representations are arbitrarily tessellated regions surrounding the robot and can vary in resolution, shape, and uniformity. The simplest version involves a two-dimensional array of cells, where each cell in the array corresponds to a square of fixed size in the region being mapped. The radial sector grid representations [21][22] have also been used for motion planning. One drawback of the grid representations is that if the sensed area is large, they may consume a large amount of memory resources and require long computation time to generate appropriate motion behaviors. Another disadvantage is that the distances from the robot to the obstacles, which are often measurements obtained directly from onboard sensors and the input to the motion generator, are hidden in the grid representations and need to be inferred from the indices of the grids. Moreover, at each instant, the information of the obstacle(s) hidden behind the detectable ones is not needed for the purpose of local navigation.

Taking these into account, a new representation of the local environment is proposed for the robot employing a rangefinder(sonar array, laser scanner, infrared, etc.). A simple vector[7] is constructed directly from the scanned data:

$$\mathbf{R} = [R_1, ...R_j, ...R_{N_s}]^T \in \mathbb{R}^{\mathrm{N_s}} \tag{2}$$

where R_j ($j = 1, 2, \ldots, N_s$) is the measured distance between the robot and the obstacle detected in the jth direction. When no obstacle is detected, R_j is set as R_d, the maximum detectable range.

The beams of a rangefinder can be unequally distributed if there is information on how the index of each beam is related to the beam angle. The vector representation is natural in the sense that it resembles the distribution of sensory data. The vector representation is suitable for behavior generation and obstacle avoidance of a robot employing a rangefinder. In the potential field methods, for example, the repulsive forces associated with the individual known obstacles can be directly derived from the vector representation.

As shown in Fig. 2, the map saves much space compared to a grid map or radial sector grid map. If the beam number is eight, the map can be described by a vector $\mathbf{R} = [R_1, R_2, R_d, R_4, R_5, R_d, R_7, R_8]^T$. To represent the same local space, the radial sector grid map will use a $8 \times N_d$ matrix, where N_d is the grid dimension along the range. To represent the same local environment in the figure, the radial grid map will be a $N_s \times N_d$ matrix of the form

$$M^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \times & 0 & 1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \times & 1 & 1 & \times & 1 & \times & \times \end{bmatrix}$$

where a zero and a non-zero value indicate the corresponding space is free space and occupied by an object respectively, and"$\times$" denotes that the corresponding cell is undetermined.

In actual implementation, the laser sensor is often mounted at the center of the robot. In that case, the sensor reference coincides with the body coordinate frame and the local coordinates of each obstacle point can be easily converted from the

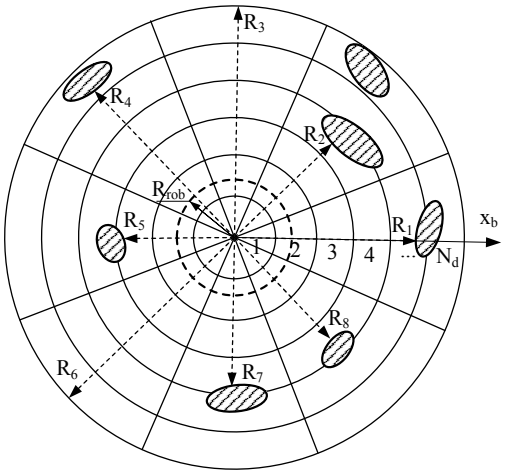


Fig. 2. The local environment sensed by a rangefinder.

expression of Eq. (2). Let $R_{\rm G}$ denote the distance of the goal from the robot and $j_{\rm G}$ denote the index of the beam nearest to $\theta_{\rm G}$, the direction of the goal. When the goal is detected by the sensor, the area behind the goal will be of no interest to the robot. Taken this into account, a modified version of the vector representation, $\hat{\mathbf{R}} \in \mathbb{R}^{\rm N_s}$, is used to describe the local environment and the jth element of $\hat{\mathbf{R}}$ is given by

$$\hat{R}_j = \begin{cases} R_j, & j \in (1, 2, \ldots, N_s),\ j \neq j_{\rm G} \\ \min(R_j, R_{\rm G}), & j = j_{\rm G} \end{cases} \tag{3}$$

Remark 1: The new representation of the local environment has the following advantages: i) it can be directly set up based on the range measurements from range sensors such as sonar or laser; ii) compared with the grid representations, the vector representation saves greatly on memory space, considering that it is a vector of dimension N_s rather than a $N_s \times N_d$ matrix of the same element size; and iii) $\hat{\mathbf{R}}$ is able to represent the local environment sensed by a rangefinder and is suitable for behavior generation and obstacle avoidance of a robot employing a rangefinder.

B. Modeling of the Sensed Environment

1) Grouping of Object Points into Obstacles: The above representations of scanned data have not taken uncertainties of measurements into account. However, these uncertainties should be properly incorporated in for safe navigation. The distance uncertainty of range sensors is often a function of range. In [23], a fit for the range error curve of a laser scanner is obtained from actual measurements: $\sigma_\rho = 9.6 \times 10^{-7}\rho^2 - 5.6 \times 10^{-4}\rho + 21.213 \quad (mm)$, where the distance error is identified by the *range error variance* σ_ρ^2. The maximum range error is denoted by σ_{Rd} as it occurs when the range measurement is around R_d.

A set of object positions can be obtained from one scan. If a range value is less than the maximum (taking the range uncertainty into account, $R_d - \sigma_{R_d}$ is used as the threshold), it implies that an obstacle has been detected. The scanned sequence of range, $\mathcal{S} = \{p_j | j = 1, \ldots, N_s\}$, can be divided into *obstacle point set* and *non-obstacle point set* as follows:

$$\bigcup_{k=1}^{n_\mathcal{O}} \mathcal{O}_k = \{p_j | p_j \in \mathcal{S}, \rho_j < R_d - \sigma_{R_d}\} \tag{4}$$

$$\bar{\mathcal{O}} = \{p_j | p_j \in \mathcal{S}, \rho_j \geq R_d - \sigma_{R_d}\} \tag{5}$$

where $n_\mathcal{O}$ is the number of obstacles. With this partition method of measured points in mind, the sensed free-space around $\mathcal{R}$ in Fig. 1 can be denoted by the region encircled by solid dashed lines.

As noticed from Fig. 1, these obstacle points can be grouped into several obstacles at a glance. Using the discontinuity property between adjacent observed points, the obstacle points may be divided into several different obstacles using a simple criterion that is based on constant thresholds such as:

$$\|p_j - p_{j+1}\| < \varepsilon$$

For example, an obstacle can be regarded as connected by the criterion that the distance between the associated n adjacent obstacle points is smaller than the effective size of the robot, i.e.,

$$\begin{aligned} \partial R = \{p_j | & \sqrt{\hat{R}_j^2 + R_{j+1}^2 - 2\hat{R}_j\hat{R}_{j+1}\cos(\tfrac{2\pi}{N_s})} \\ & < 2R_{\rm rob}, j = j_1, \ldots, j_1 + n - 1\} \end{aligned} \tag{6}$$

However, such approaches may be unable to cope with range data, because the distance between two measured points depends on both the angle formed by the beams and the normal vector of the surface being scanned. For example, although the distance of two adjacent obstacle points A and B in Fig. 1 is greater than $\mathcal{R}$'s effective size, for navigation purposes, they should be regarded as two points belonging to the same obstacle.

2) Range-based Straight Path: The term *path* refers to the shape of a motion, that is, the shape of the curve in the robot's configuration space. The term *trajectory* refers to the time history of positions and orientations along a path, that is, a curve through the robot's state space. In this paper, "straight path segment" is employed to determine i) if it is safe for $\mathcal{R}$ to move from its current position in a certain direction; and ii) if there is a "passage" (to be discussed later in this subsection), with which the obstacle point set is able to be distinguished into different obstacles.

Straight path segment $\gamma_{AB}(r)$ is defined as the segment consisting of a straight line segment $\overrightarrow{AB}$ with a thickness radius r and the semi-disk centered at B with a radius r which is not overlapped with the "thick" line segment $\overrightarrow{AB}$. As shown in Fig. 3, $\gamma_{AB}(r)$ is the space inside the thick solid lines $\overline{b_2a_2}$, $\overline{a_2a_1}$, $\overline{a_1b_1}$ and arc $\overline{b_1b_2}$ and it does not equal to $\gamma_{BA}(r)$. The *length* of the straight path segment $\gamma_{AB}(r)$, denoted as $\ell_{\gamma_{AB}(r)}$, is measured by $|AB|$. In the context of discrete range data, let $\gamma_{j,\ell}(r)$ denote the straight path of length ℓ in the jth beam direction.

Definition 2.1: The straight path segment $\gamma_{AB}(r)$ is said to be *safe* for the robot with an effective radius $R_{\rm rob}$ to move from its current position A to its destination B, if and only if $r \geq R_{\rm rob}$ holds and all the obstacle points are not in the path.

A function SafeSP($\gamma_{AB}(r)$) is defined to check if the straight path $\gamma_{AB}(r)$ is "safe" (according to Definition 2.1)

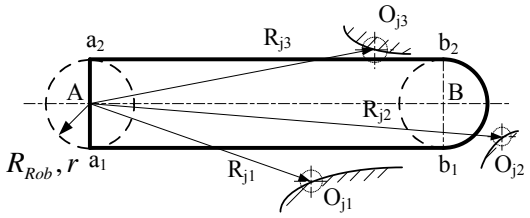


Fig. 3. Straight path: the space inside thick solid lines.

for the robot. In Fig. 3, the measured position value of each obstacle point is denoted by the center of a small dashed circle and, due to range sensing uncertainty, the actual position of the point is confined to the circle. Suppose that the robot of an effective radius $R_{\text{rob}} = r$ moves from A to B, $\gamma_{AB}(r)$ is safe for obstacle points O_{j_1} and O_{j_2}. However, when the uncertainty of the range sensing is considered, $\gamma_{AB}(r)$ is not safe for obstacle point O_{j_3}.

There exists a safe straight path from point A to point B for the robot, if and only if the straight path segment $\gamma_{AB}(R_{\text{rob}} + \sigma_{R_d})$ is safe for the robot to move from point A to point B. If any of the beams on the right-hand side of $\overline{a_1a_2}$ does not intersect line segments $\overline{b_2a_2}$, $\overline{a_1b_1}$ or arc b_1b_2, there exists no safe path from point A to point B for the robot.

3) Obstacle Boundary and Obstacle Range: A *passage in the jth beam direction*, denoted as Γ_j, is defined as the continuous safe straight path segment with a length of $\ell = R_d - (R_{\text{rob}} + \sigma_{R_d})$ in the jth beam direction for a physical robot to pass through from its current position. In the jth beam direction, there exists a passage

$$\Gamma_j = \gamma_{j,\ell}(R_{\text{rob}} + \sigma_{R_d}) \tag{7}$$

if and only if $\text{SafeSP}(\gamma_{j,\ell}(R_{\text{rob}} + \sigma_{R_d})) = \text{True}$.

The concept of passage helps to distinguish obstacle points belonging to different obstacles. No passage exists between any two successive obstacle points if they belong to the same obstacle. Then, for a measured sequence $\mathcal{S}_{j_1}^{j_2}$ of adjacent points, indexed from j_1 to j_2, no passage should exist between any two successive points in order for all obstacle points of the measured sequence to belong to the boundary ∂R, i.e.,

$$\{p_j \in \mathcal{S}_{j_1}^{j_2} | \rho_j < R_d - \sigma_{R_d}\} \subseteq \partial R \Rightarrow \forall\, j \in [j_1, j_2] : \nexists \Gamma_j \tag{8}$$

Note that the overall partition threshold is variable since the distance uncertainties have been taken into account and the partition is not simply based on the continual change of two adjacent obstacle points but on the length of the perpendicular line. In this way, the partition of obstacle boundaries becomes much more feasible and robust against sensing uncertainties.

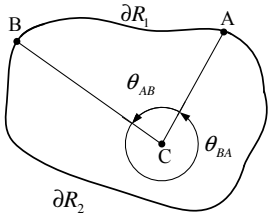


Fig. 4. Illustration of directional obstacle range and angle range.

As shown in Fig. 4, *left obstacle range* $\partial R_C^L(A, B)$ is defined as the continuous obstacle boundary (denoted as ∂R_1) from the start visible point A to the end visible point B on the left (counterclockwise) direction with respect to the start line (here is $\overrightarrow{CA}$) direction. *Left angle range* $\partial\theta_C^L(A, B)$ is the corresponding angle range. *Right obstacle range* $\partial R_C^R(A, B)$ and *right angle range* $\partial\theta_C^R(A, B)$ can be similarly defined. Note that $\partial R_C^L(A, B) \neq \partial R_C^R(A, B)$ and $\partial\theta_C^L(A, B) \neq \partial\theta_C^R(A, B)$.

As such, if $\overrightarrow{CA}$ and $\overrightarrow{CB}$ correspond to the j_1th and j_2th range readings respectively, left obstacle range $\partial R_C^L(A, B)$ can be denoted as $\partial R_C^L(j_1, j_2)$ and left angle range can be denoted as $\partial\theta_C^L(j_1, j_2)$.

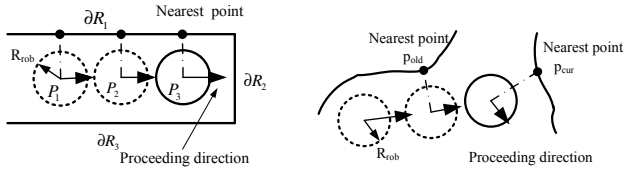
III. Boundary Following through Instant Goals

In this section, a range-based method is proposed to determine a series of Instant Goals which guide the robot to properly follow the boundary of an obstacle.

A. New Strategy of Boundary Following

The basic idea of conventional wall following algorithms is to maintain a set of (fixed) distance to a wall using a series of range measurements. The surface is followed by moving in a perpendicular direction to the nearest point on an obstacle. Using this strategy, a convex corner, concave corner or even a curve surface can be followed [5]. It assumed a sufficiently fast measurement (to obtain the nearest reflecting point) to the velocity of the robot and a capability of accurately measuring the direction to the reflecting point.

However, the reliance of these approaches on measurement of the nearest reflecting point to determine the motion direction not only requires a very good signal processing, but may also bring about improper behavior when the obstacle consists of some special shapes, or if there are some other obstacles detected very close to the one currently followed. In Fig. 5(a), ideally the robot's proceeding direction should be kept as right when following boundaries ∂R_1 and ∂R_2, and left when following ∂R_3. However, it can be seen that, due to uncertainties, the direction obtained has a 50% probability of turning out to be incorrect. In an obstacle cluttered environment as in Fig. 5(b), the robot may unexpectedly turn to follow other obstacle rather than the original one.



(a) A U-shaped obstacle. (b) Cluttered obstacles.

Fig. 5. The robot may exhibit an improper behavior during wall following.

Therefore, an Instant Goal approach is used for the robot to follow an obstacle in complex, obstacle cluttered environments. At each step, an *Instant Goal* (IG for short), a point serving as a temporary goal for the robot to reach, is computed. The robot will move *forward* in the sense of following the

boundary, which is achieved by restricting the search range of each Instant Goal to a special scope of the boundary. Let the Instant Goal be in the j_{IG}th beam direction and have a distance r_{IG} (not necessarily equals to $\hat{R}_{j_{\text{IG}}}$) from the rangefinder. If the sensor is mounted at the robot center, the Instant Goal can be expressed in the polar coordinate form with respect to the robot as

$$(r_{\text{IG}}, \theta_{\text{IG}}) = (r_{\text{IG}}, \frac{2\pi}{N_s}(j_{\text{IG}} - 1)) \tag{9}$$

In the boundary following mode, until the IG is reached or need to be revised, $\mathcal{R}$ will keep moving towards it with an orientation as follows:

$$\vartheta = \angle\overrightarrow{P_t P_{\text{IG}}} \tag{10}$$

When $\mathcal{R}$ is at a location to decide either to follow an obstacle or to move directly towards the goal, the location is defined as a *Start*, S_i, where $i = 1, 2, \cdots$ is the index of obstacles met in sequence. The position of $\mathcal{R}$ where an Instant Goal is determined is defined as *Instant Start*, $S_{i,k}$ where $k = 1, 2, \cdots$ is the index of *Instant Starts* during the following of the ith obstacle. $G_{i,k}$ denotes an Instant Goal in a similar way. Algorithm 1 describes the procedure of following an obstacle through the use of Instant Goals, the determination of which will be discussed in the following subsections.

Algorithm 1 Following An Obstacle.

```
S_i ← P_t, k ← 1
Assign a following direction
while the whole boundary is not covered do
    S_{i,k} ← P_t                          ▷ Set current Instant Start
    Compute search range
    Determine Instant Goal G_{i,k}
    while |P_t G_{i,k}| > ε_IG do
        Move towards G_{i,k} in the direction of the vector P_t G_{i,k}
        if Instant Goal need to be revised then
            Exit While
    k ← k + 1
```

When $\mathcal{R}$ is moving towards an Instant Goal, only the range data in "front" of $\mathcal{R}$ are "concerned" for the purpose of obstacle avoidance (to be described in the next section). The "concerned" beams are of an absolute angle not more than $(\pi/2)$ from the robot orientation (10). *Nearest concerned range*, $R_{\min}(\vartheta)$, is defined as the range value of the shortest one among the concerned beams.

For the safety of navigation, the velocity magnitude of $\mathcal{R}$ is determined depending on this nearest concerned range. It will be higher when $\mathcal{R}$ is far from the "concerned" obstacles; and vice versa. Let V_{opt} denote the reasonable speed that $\mathcal{R}$ can drive stably in an obstacle-free environment. With ϑ, the velocity magnitude of $\mathcal{R}$ is determined as follows:

$$\|\mathbf{V}_t\| = \begin{cases} \frac{|P_t P_{\text{IG}}|}{\Delta t}, & \text{if } \frac{|P_t P_{\text{IG}}|}{\Delta t} < \frac{R_{\min}(\vartheta)}{R_{\min}(\vartheta)+R_{\text{OB}}} V_{\text{opt}} \\ \frac{R_{\min}(\vartheta)}{R_{\min}(\vartheta)+R_{\text{OB}}} V_{\text{opt}}, & \text{otherwise} \end{cases} \tag{11}$$

where the constant R_{OB} is the obstacle influence range to be given as Eq. (19).

In Eq. (11), by comparing the velocity magnitude with $(|P_t P_{\text{IG}}|/\Delta t)$, it ensures that the robot will not pass beyond the IG after driving at the obtained speed for a duration of Δt. In addition, even if $R_{\min}(\vartheta) \gg R_{\text{OB}}$ or $R_{\min}(\vartheta) \ll R_{\text{OB}}$, due to the robot's physical dimension and the limit of range measurements, $\|\mathbf{V}_t\|$ will be bounded by

$$\frac{R_{\text{rob}}}{R_{\text{rob}} + R_{\text{OB}}} V_{\text{opt}} \le \|\mathbf{V}_t\| \le \frac{R_d}{R_d + R_{\text{OB}}} V_{\text{opt}}$$

B. Search Range for Instant Goal Determination

Three parameters critical for searching for Instant Goals are first defined. With respect to the body frame, *SchFrom* and *SchEnd* are the directions from which the searching process starts and ends respectively. *SchDir* is the left- or right-hand side of the SchFrom direction where the searching process is conducted. The *search range* determined by the quaternion $(P_t, SchDir, SchFrom, SchEnd)$ is expressed as

$$\Theta = \partial\theta_{P_t}^{SchDir}(SchFrom, SchEnd) \tag{12}$$

The direction of following $SchDir$ is determined at each *Start*, after which $\mathcal{R}$ will keep following the boundary in that direction. For the purpose of boundary following only, $SchDir$ can be arbitrarily assigned as either the left- or right-hand side direction. However, it can be chosen as the preferred following direction, which will be discussed in Section V-B. When determining the preferred $SchDir$ or $SchFrom$, if there is no actual goal, G will be selected as an imaginary goal for the robot to reach only requiring that the straight line between it and the robot intersects with the obstacle.

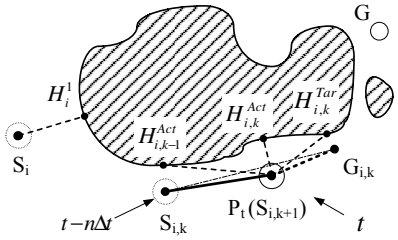


Fig. 6. Determination of the search range during the robot's moving from $S_{i,k}$ towards $G_{i,k}$.

As shown in Fig. 6, *Near Hit point* at the ith *Start*, H_i^1, is defined as the visible obstacle point on $\overrightarrow{S_i G}$. *Target Hit point*, $H_{i,k}^{\text{Tar}}$, is defined as the visible obstacle point that the robot is expected to encounter upon reaching $G_{i,k}$.

$SchFrom$ is determined according to what situation the robot is in. If $\mathcal{R}$ is currently at some *Start*, it is simply set as the goal direction; otherwise, that is, the robot is moving from $S_{i,k}$ towards $G_{i,k}$, it will be set as the direction of the *Actual Hit point*, $H_{i,k}^{\text{Act}}$, which is defined as the obstacle point (on the boundary) nearest to $\mathcal{R}$.

The Actual Hit point $H_{i,k}^{\text{Act}}$ is determined according to the following three different cases. Note that no matter the current IG is reached or not, there may exist a situation where a new Instant Goal needs to be determined. For example, the current IG is not reachable, or approaching it any further will cause the robot to follow an obstacle that is not desired.

i) The IG has been reached:
When $|P_tG_{i,k}| \le \varepsilon_{\text{IG}}$ or $|P_tG_{i,k}| \ll |S_{i,k}G_{i,k}|$, it is taken that the IG has been reached. Reasonably, take current Target Hit point obtained as the Actual Hit point:

$$H_{i,k}^{\text{Act}} = H_{i,k}^{\text{Tar}}$$

ii) Otherwise:
The Actual Hit point will be approximated by the point that $\mathcal{R}$ encounters first by imagining that its dimensions are big enough for it to contact the boundary. $H_{i,k}^{\text{Act}}$ is determined in a way that $\overrightarrow{P_tH_{i,k}^{\text{Act}}}$ is perpendicular to $\overrightarrow{S_{i,k}G_{i,k}}$ such that:

$$\mathbf{n}_{P_tH_{i,k}^{\text{Act}}} = \begin{bmatrix} \cos\theta_\perp & -\sin\theta_\perp \\ \sin\theta_\perp & \cos\theta_\perp \end{bmatrix} \mathbf{n}_{S_{i,k}G_{i,k}}$$

where $\theta_\perp$ is $(\pi/2)$ if $SchDir$ is $RIGHT$ or $-(\pi/2)$ otherwise.

iii) In case (ii), if $\overrightarrow{P_tH_{i,k}^{\text{Act}}}$ is not within $\partial\theta_{P_t}^{SchDir}(H_{i,k-1}^{\text{Act}}, H_{i,k}^{\text{Tar}})$:
$\mathcal{R}$'s current position, P_t, may not lie exactly on line segment $\overrightarrow{S_{i,k}G_{i,k}}$. Furthermore, since $\mathcal{R}$ is required to follow the boundary forward in the direction of $SchDir$, point $H_{i,k}^{\text{Act}}$ should be "between" points $H_{i,k-1}^{\text{Act}}$ and $H_{i,k}^{\text{Tar}}$ on the boundary. Considering these two factors, in this case, $H_{i,k}^{\text{Act}}$ will not be computed as case ii) but just be set conservatively as $H_{i,k-1}^{\text{Act}}$, i.e.,

$$H_{i,k}^{\text{Act}} = H_{i,k-1}^{\text{Act}}$$

C. Algorithm to Determine Instant Goals

After the search range $\Theta = \partial\theta_{S_{i,k}}^{SchDir}(SchFrom, SchEnd)$ is determined, Instant Goal $G_{i,k}$ will be obtained by checking each beam within it using the following procedure. Let variables j_{cur} and j_{old} record the current and last "searched" beams respectively. Initially, set $j_{\text{cur}} = j_{\text{old}} = SchFrom$.

Step 1: Check whether there is an IG candidate in the direction of the current or last "searched" beam:

Set j_{cur} as the index of the next beam in the $SchDir$ within the search range.

There will be an IG candidate (denoted as IG*) in the direction of current or last "searched" beam if either i) the distance between the two successive measured points exceeds a certain threshold $\varepsilon_r > 2\sigma_{Rd}$, i.e.,

$$\|p_{j_{\text{cur}}} - p_{j_{\text{old}}}\| = \sqrt{\hat{R}_{j_{\text{cur}}}^2 + \hat{R}_{j_{\text{old}}}^2 - 2\hat{R}_{j_{\text{cur}}}\hat{R}_{j_{\text{old}}}\cos\frac{2\pi}{N_s}} \ge \varepsilon_r \tag{13}$$

or ii) after checking a certain number (for example five) of beams, Eq. (13) does not hold for any of them.

If there is an IG candidate, go to Step 2, otherwise, repeat Step 1.

Step 2: Determine the parameters of IG*:

The direction of IG* is set to be that of the longer of the j_{cur}th and j_{old}th beams:

$$j_{\text{IG}^*} = \begin{cases} j_{\text{cur}}, & \text{if } \hat{R}_{j_{\text{cur}}} > \hat{R}_{j_{\text{old}}} \\ j_{\text{old}}, & \text{otherwise} \end{cases} \tag{14}$$

If $\hat{R}_{j_{\text{IG}^*}} \ge R_d - \sigma_{R_d}$, i.e., no obstacle is detected in the direction of IG*, $\mathcal{R}$ can move along the direction as far as $(\hat{R}_{j_{\text{IG}^*}} - R_{\text{rob}})$ (there is little knowledge of the environment beyond this distance). Otherwise, $\mathcal{R}$ can go as far as $(\hat{R}_{j_{\text{IG}^*}} - 2R_{\text{rob}})$ rather than $(\hat{R}_{j_{\text{IG}^*}} - R_{\text{rob}})$ so as not to collide with the obstacles nearby. That is, the distance of IG* is given by

$$r_{\text{IG}^*} = \begin{cases} \hat{R}_{j_{\text{IG}^*}} - R_{\text{rob}}, \text{ if } \hat{R}_{j_{\text{IG}^*}} \ge R_d - \sigma_{R_d} \\ \max(\hat{R}_{j_{\text{IG}^*}} - 2R_{\text{rob}}, r_0), \text{ otherwise} \end{cases} \tag{15}$$

where r_0 is a constant to ensure that r_{IG^*} is not less than a minimum value.

Step 3: IG* will be taken as IG in either of the following two cases:

i) The distance between two successive measured points $p_{j_{\text{old}}}$ and $p_{j\text{cur}}$ (as shown in Fig. 7) is greater than the robot size, i.e.,

$$\|p_{j_{\text{cur}}} - p_{j_{\text{old}}}\| > 2R_{\text{rob}} \tag{16}$$

If Eq. (16) is satisfied, there is possibly a passage between the two points and thus check if an IG exists near them. As shown in Fig. 7, point M is on line segment $\overrightarrow{p_{j_{\text{old}}}p_{j\text{cur}}}$ and is of a distance $\min(\|p_{j_{\text{cur}}} - p_{j_{\text{old}}}\|/2, R_{\text{OB}})$ from point $p_{j\text{cur}}$. Point C is on the same side as the robot with respect to the line $\overrightarrow{p_{j_{\text{old}}}p_{j\text{cur}}}$ and is perpendicular to the line with a distance of $(R_{\text{OB}} + R_{\text{rob}})/2$ from point M. Then the neighboring area NEIGH to choose IG*s is given by a disc of radius $(R_{\text{OB}} - R_{\text{rob}})/2$ centered at point C. There will be a point in the area that $\mathcal{R}$ can reach safely if

$$\{j|\text{SafeSP}(\gamma_{j,r_j}(R_{\text{rob}} + \sigma_{R_d})) = \text{True}, \quad (r_j, j) \in \text{NEIGH}, j \in \Theta\} \ne \emptyset \tag{17}$$

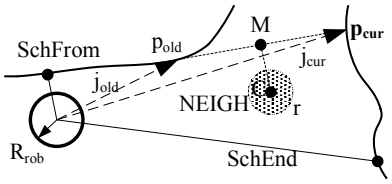


Fig. 7. Determination of neighboring area NEIGH.

ii) There exists a safe path with length r_{IG^*} in the direction of the j_{IG^*}th beam, that is,

$$\text{SafeSP}(\gamma_{j_{\text{IG}^*}, r_{\text{IG}^*}}(R_{\text{rob}} + \sigma_{R_d})) = \text{True} \tag{18}$$

In the first case, let j_{near} denote the nearest one in the set $\mathcal{J}$ with respect to the j_{old}th beam and set $j_{\text{IG}^*} := j_{near}$ and $r_{\text{IG}^*} := r_{j_{near}}$. In either case, the Instant Goal $G_{i,k}$ is found. If neither case is met, continue the IG search cycle.

Remark 2: During boundary following of the obstructive obstacle $\mathcal{O}$, if there is an obstacle, say $\mathcal{O}_{\text{dis}}$, very close to obstacle $\mathcal{O}_i$ and the robot, the approach will determine the obstacle to follow in the following manner: if there is a passage between the two obstacles, an Instant Goal will be generated between the two obstacles and the robot will continue to follow obstacle $\mathcal{O}_i$; otherwise, (there is no passage), obstacle $\mathcal{O}_{\text{dis}}$ is

regarded as part of obstacle $\mathcal{O}_i$ and the robot continues to follow the "expanded" obstacle.

There are some work on navigation through local goals, for example, the subgoal selection algorithm [19] where subgoals are chosen to be points at a certain distance ϵ with respect to associated obstacle vertices. It sets a certain distance ϵ to ensure that the subgoal for an edge is not on another obstacle ("which must be at least a distance $\geq 2\epsilon$ from the edge"), and thus guarantees that further subgoals will be generated. However, for the purpose of following an obstacle, the generation of Instant Goals has several advantages over that of subgoals:

(i) The Instant Goal approach does not require the geometric properties of obstacles. The subgoal method requires a point robot and the environment consisting of convex polygonal obstacles. It is therefore not directly applicable to the navigation of a physical robot in a real environment.

(ii) The subgoal method sets a certain distance to ensure that the subgoal for an edge is not on another obstacle. The Instant Goal approach uses the concept of "safe path" and "passage" and thus is more appropriate to distinguish the disturbing obstacles from the desired one.

(iii) Just setting a certain distance may be not able to guarantee that further subgoals will be generated, for example if the next edge is currently invisible. In contrast, Instant Goals are able to be determined in any environment containing complex obstacles due to the following reasons: 1) it does not require the geometric properties about obstacles; 2) Instant Goals are dynamically determined based on the new sensory information, unlike subgoals which are all determined at one time; and 3) as in Step 1 of Sec 3.4, the condition to determine an IG* ensures that there is always an IG candidate that will be obtained within the searching scope.

IV. Collision Checking

The generation of Instant Goals has taken the locations of obstacles into account as well as the robot dimensions and the range uncertainty (σ_{R_d}). Boundary following through Instant Goals can solve obstacle avoidance problem to some extent. However, due to the unpredictability of the environment, sensor noise and imperfectness of control, the robot may still face the threat of collisions. In order to fully guarantee the safety of navigation, a special obstacle avoidance is considered here when some obstacles are near enough in the moving direction of the robot.

A. Critical Obstacle Avoidance

For a robot with velocity V_t in the duration of Δt, a collision with one or more obstacle(s) is possible only when it is within a certain range of the obstacles. If $\mathcal{R}$ is out of this range, it can move safely with the velocity in any direction. This *obstacle influence range*, denoted as R_{OB}, can be represented with respect to $\mathcal{R}$ as a circle centered at $\mathcal{R}$ with radius

$$R_{\text{OB}} = R_{\text{rob}} + \|V_t\|\Delta t + \sigma_{R_d} \tag{19}$$

However, even if an obstacle is within this influence range, it will not directly affect the motion of $\mathcal{R}$ to its Instant Goal if $\mathcal{R}$ is *behind* the IG. This *IG influence range* is given by

$$R_{\text{IG}} = |P_t P_{\text{IG}}| + R_{\text{rob}} + \sigma_{Rd} \tag{20}$$

where σ_{Rd} is added in order to take the uncertainty of range data into account.

Our strategy is that obstacle avoidance is triggered only when the nearest concerned range $R_{\min}(\vartheta)$ is within the minimum value of R_{OB} and R_{IG}, that is,

$$R_{\min}(\vartheta) \leq R_{\text{OI}} = \min(R_{\text{OB}}, R_{\text{IG}}) \tag{21}$$

At each time instant, every detected obstacle within the influence circle R_{OI} exerts on the robot a repulsive force which opposes the direction from the center of the robot to the obstacle and the total repulsive force exerted on the robot by all obstacles is the sum of the repulsive force in each beam direction. The repulsive potential U_j generated by the obstacle in the jth beam direction is constructed similar to [7]:

$$U_j = \begin{cases} \frac{1}{2}\left(\frac{1}{\hat{R}_j} - \frac{1}{R_{\text{OI}}}\right)^2, & \text{if } \hat{R}_j \leq R_{\text{OI}} \\ 0, & \text{if } \hat{R}_j > R_{\text{OI}} \end{cases}$$

Only the concerned range data need to be considered for avoiding collision with obstacles and those of a smaller angle relative to the velocity affect motion more significantly. To reflect these factors, a coefficient function is introduced:

$$\lambda(j) = \frac{1 + \text{sgn}(\frac{\pi}{2} - \angle_{j,j_{\text{IG}}})}{2}(C_0 + \frac{\frac{\pi}{2} - \angle_{j,j_{\text{IG}}}}{\frac{\pi}{2}}) \tag{22}$$

where $\angle_{j,j_{\text{IG}}} \in [0, \pi]$ denotes the absolute angle between the jth and j_{IG}th beam directions; the parameter C_0 is a positive constant, for instance 0.1, to make sure the obstacles detected in the direction perpendicular to $\mathbf{V}_t$ may also have some effect (though relatively small) on collision avoidance.

With the information of both obstacle ranges and distance from IG, *obstacle influence map* $\tilde{\mathbf{R}}$ is used to reveal in which direction there exist obstacles that may collide with the robot:

$$\tilde{\mathbf{R}} = \left[\frac{1 - \text{sgn}(\hat{R}_j - R_{\text{OI}})}{2}\lambda(j)\right] \in \mathbb{R}^{\text{N}_\text{s}} \tag{23}$$

which implies that $\tilde{\mathbf{R}}$ will be nonzero only if $\hat{R}_j < R_{\text{OI}}$.

A pre-calculated unit vector $\mathbf{B}$ is defined such that its jth element, B_j ($j = 1, 2, \ldots, N_s$), is the direction opposite to the jth beam direction. The repulsive force required to avoid the obstacles in the direction of the jth beam is given by

$$\mathbf{f}_j = \tilde{R}_j \left(\frac{1}{\hat{R}_j} - \frac{1}{R_{\text{OI}}}\right)\frac{1}{\hat{R}_j^2} B_j \tag{24}$$

where only for $\tilde{R}_j \neq 0$ does $\mathbf{f}_j$ need to be calculated.

B. Integration of Collision Avoidance with Boundary Following

During boundary following, collision checking is made at each time instant. Normally $\mathcal{R}$ moves directly towards its Instant Goal except when there are obstacles detected as near as in Eq. (21), in which case $\mathcal{R}$ will move in a different direction temporarily (for a period of Δt). There

are two basic behaviors: the motion towards the Instant Goal (*Instant Goal driven* or $\mathcal{IG}$ behavior), and the motion due to the repulsive force generated by close obstacles (*Obstacle Avoidance* or $\mathcal{OA}$ behavior). The final motion is determined through the combination of the two behaviors based on the vector summation method due to the simplicity of this method. Obstacle avoidance is integrated into the whole navigation in a way not to affect the nature of boundary following. After that, if obstacle avoidance is no longer necessary, $\mathcal{R}$ will switch back to move directly towards the Instant Goal.

After the IG is determined, $\mathcal{IG}$ behavior can be expressed in the body coordinate frame by a unit vector originating from the center of the robot and pointing to the IG:

$$\mathbf{F}_{\mathcal{IG}} = [\cos\theta_{\text{IG}} \ \ \sin\theta_{\text{IG}}]^T \in \mathbb{R}^2 \tag{25}$$

Compared with vector $\mathbf{F}_{\mathcal{IG}}$ which is of a unit magnitude, the magnitude of the total repulsive force summed by Eq. (24) is inversely proportional to a cubic distance. As in [7], a scalar $\xi > 0$ is used to scale the repulsive force to a level comparable to $\mathbf{F}_{\mathcal{IG}}$:

$$\mathbf{F}_{\mathcal{OA}} = \xi \sum_{j=1}^{N_s} \mathbf{f}_j \tag{26}$$

In order to determine the value of ξ, considering that the range value of the j_{IG} beam is as short as the minimum collision-free distance R_{rob}, the magnitude of the repulsive force in this beam direction should be much greater than $\mathbf{F}_{\mathcal{IG}}$ in order to make sure that avoidance behavior dominates in the behavior combination. Note that $\lambda(j_{\text{IG}}) = 1$, ξ can be calculated by setting

$$\xi \left(\frac{1}{R_{\text{rob}}} - \frac{1}{R_{\text{OI}}} \right) \frac{1}{{R_{\text{rob}}}^2} = C_f \tag{27}$$

where C_f is a predefined constant much greater than 1.

The orientation of $\mathcal{R}$, stated initially as in Eq. (10), is now re-determined by the summation of vectors $\mathbf{F}_{\mathcal{IG}}$ and $\mathbf{F}_{\mathcal{OA}}$:

$$\vartheta = \angle(\mathbf{F}_{\mathcal{IG}} + \mathbf{F}_{\mathcal{OA}}) \tag{28}$$

V. Instant Goal Based Globally Convergent Path Planning

In this section, based on the Instant Goal approach of boundary following rather than assuming a perfect capability of boundary following, a realistic globally convergent path planner is presented for the navigation of a mobile robot. The sequence of generated Instant Goals incrementally guide the robot to the goal and a Bug-like strategy is used to switch between the two motion modes. In contrast to the previous sections on "lower level" navigation algorithm (boundary following), the design of a high level path planner is presented here which includes the leave condition, test of reachability and choice of following direction.

A. Design of the Path Planner

1) Leave Condition: There are two basic modes of robot motion: i) moving towards the goal; ii) following the obstacle that is obstructing the robot. Initially, the robot moves to the goal until it encounters an obstacle in its direct path. At each *Start*, if no obstructing obstacle is detected, $\mathcal{R}$ moves directly towards the goal until one is detected; otherwise, it will enter the boundary following mode and move along current obstructing obstacle using the Instant Goal approach.

During the boundary following mode, upon the receipt of each laser scan, if a certain condition is satisfied, $\mathcal{R}$ will leave from current obstructing obstacle $\mathcal{O}_i'$, and either move towards the goal or start following another obstacle. The robot's current position is denoted as "leave point" L_i and in the meantime it will be set as the next *Start*, S_{i+1}.

The basic leave condition [1] can be described as follows: $\mathcal{R}$ moves along $\overrightarrow{SG}$ until an obstacle, $\mathcal{O}_i^{'}$, is encountered where a hit point H_i^1 is defined; then it follows the obstacle boundary until $\overrightarrow{SG}$ is met at a distance d from G such that $d < |H_i^1 G|$, where a new hit point H_{i+1}^1 is defined. After leaving the obstacle, $\mathcal{R}$ continues to move along $\overrightarrow{SG}$.

We propose a *leave condition* upon which the robot will leave the obstacle:

Case (i): no obstructing obstacle is detected and the distance from the goal is less than $|S_i^1 G|$; or

Case (ii): a new obstacle, rather the original obstructing one, is detected to be obstructing, and the distance from current Near Hit point H_{i+1}^1 to the goal is less than $|H_i^1 G|$.

2) Test of Goal Reachability: Goal reachability test is dynamically performed during boundary following. It is natural to think the goal is unreachable if the robot has returned to a previous position after having traversed the whole boundary of an obstacle. But in the presence of sensing uncertainties, it is not easy to accurately determine whether a location has been accessed for a second time. Furthermore, returning to a previous position is only a special case that the goal is unreachable.

Lemma 1: While following an obstacle boundary from *Start* S_i, if $\mathcal{R}$ traverses line segment $\overrightarrow{H_i^1 H_i^2}$ in the same direction twice, then $\mathcal{R}$ completes a loop around the obstacle and thus the goal is unreachable.

Proof: If the current moving mode is determined to be boundary following, subsequently $\mathcal{R}$ keeps following the obstructing obstacle. When $\overrightarrow{H_i^1 H_i^2}$ is traversed by $\mathcal{R}$ in the same direction for a second time, it means that $\mathcal{R}$ has traveled at least 360^o around all the obstacle points on boundary $\partial R_i^{'}$ (note that boundary $\partial R_i^{'}$ may be inside obstacle $\mathcal{O}_i^{'}$). This implies that either the robot or the goal is trapped, in which case the goal is unreachable.

The using of Lemma 1 greatly relaxes the requirement that the robot need to return to the hit point H_i^1 before it identifies the next hit point. In addition, it is not required for $\mathcal{R}$ to return exactly to a previously covered position to determine the goal reachability. The opposite situation is that the robot rather than the goal is trapped, where the goal is also not reachable.

3) Convergence Analysis: While $\mathcal{R}$ moves towards the goal, it may meet a series of obstacles which are indexed in sequence. Suppose that the current blocking obstacle has the index i. In Fig. 8, S_i denotes the ith *Start* and accordingly H_i^1 denotes the Near Hit point at *Start* S_i.

i) if $\mathcal{R}$ leaves the boundary according to Case (ii) of the proposed leave condition, it will ensure that

$$|H_i^1 G| > |H_{i+1}^1 G|$$

For example, in Fig. 8, $|H_{i+1}^1 G| > |H_{i+2}^1 G|$ and $|H_{i+2}^1 G| > |H_{i+3}^1 G|$ hold.

ii) if $\mathcal{R}$ leaves the boundary according to Case (i) of the proposed leave condition, then

$$\begin{cases} |S_i G| & > |S_{i+1} G| \\ |S_i G| & = |S_i H_i^1| + |H_i^1 G| \\ |S_{i+1} G| & = |S_{i+1} H_{i+1}^1| + |H_{i+1}^1 G| \\ |S_i H_i^1| & \le R_d < |S_{i+1} H_{i+1}^1| \end{cases} \Rightarrow |H_i^1 G| > |H_{i+1}^1 G|$$

For example, in Fig. 8, $|H_{i+3}^1 G| > |H_{i+4}^1 G|$ holds.

iii) if current motion mode is *direct moving*, it is known that $|H_k^1 G| > |H_{k+1}^1 G|$, since $\mathcal{R}$ moves along $\overrightarrow{S_k G}$ before an obstacle is detected. For example, in Fig. 8, $|H_i^1 G| > |H_{i+1}^1 G|$ holds.

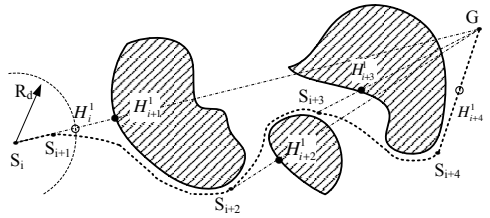


Fig. 8. A path generated by the path planner using the proposed leave condition.

As the above has included all the three possible cases, it can be concluded that the distance between each Near Hit point and the goal decreases monotonically. Moreover, from the assumption that all obstacle boundaries are of a finite length, we know that the path length from a *Start* to the leave point is finite. If the goal is reachable, convergence to the goal is guaranteed by the *leave condition*. Let H_n^1 denote the last Hit point before the robot reaches the goal, the finite sequence of the distance function between the *Starts* and the goal can be given by

$$|H_1^1 G| > |H_2^1 G| > \cdots > |H_i^1 G| > \cdots > |H_n^1 G|$$

B. Direction of Boundary Following

From the global performance standpoint, neither local direction can be judged better than the other as long as no complete information is available. The local direction decided will directly affect the performance of the subsequent boundary following. It can be deduced that the likelihood of obtaining a better global performance will increase by choosing a locally preferable direction at each *Start*. In [18], the initial boundary following direction was chosen based on the orientation of the boundary at the hit point (similar to H_i^1 in this paper) and expected that following the direction would take the robot closer to the goal. However, this approach utilizes only partial information, that is, the information of the hit point.

In this paper, the determination of the following direction, $SchDir$, will take into account as much information of the obstacle as possible. It is conducted by searching the Instant Goal at the *Start* in a way similar to the "Algorithm to Determine Instant Goals" in Section III-C, except that i) the search scope is the concerned beams; ii) checking is conducted simultaneously on the left- and right-hand sides of the j_{G}th beam direction; and iii) the search will be stopped only after checking all the beams within the search scope.

Let $\mathscr{P}(j)$ denote the probability of being the IG direction for the jth beam. When Eq. (13) is satisfied, $\mathscr{P}(j_{\mathrm{IG}^*})$ is given by

$$\mathscr{P}(j_{\mathrm{IG}^*}) = \min(1, \frac{\|p_{j_{\mathrm{cur}}} - p_{j_{\mathrm{old}}}\|}{2R_{\mathrm{rob}}}) + \frac{|\hat{R}_{j_{\mathrm{cur}}} - \hat{R}_{j_{\mathrm{old}}}|}{R_d} - \frac{\angle_{j_{\mathrm{IG}^*}, j_{\mathrm{G}}}}{\pi/2} \quad (29)$$

where, on the right-hand side, the first term represents the possibility of a passage for $\mathcal{R}$ to go towards somewhere near the two successive obstacle points, the second term relates to how far the obstacles are in this direction, and the third term is about how big the angle is between this direction and the goal direction.

If in the meantime, Case (i) of Step 3 in Section III-C is satisfied, which means that there is a passage for $\mathcal{R}$ to go towards somewhere near the two successive obstacle points, the probability of the beam to be the final IG direction is greatly increased:

$$\mathscr{P}(j_{\mathrm{IG}^*}) = \mathscr{P}(j_{\mathrm{IG}^*}) + 1 \quad (30)$$

If the obtained probability is greater than the maximum probability, $\mathscr{P}\mathrm{max}$, ever recorded, update the maximum probability accordingly. Continue checking the next beams in the two directions.

Finally, the local direction of boundary following is determined according to the direction of the beam where $\mathscr{P}\mathrm{max}$ is found. It will be assigned the value of LEFT if this beam is to the left of the goal direction, or RIGHT otherwise.

VI. Simulation Studies

Extensive simulations are carried out to study the effect of the Instant Goal approach and the globally convergent path planner. The "sensor data" are simulated with noise information based on the sensor's characteristics and behavior, which enables the test of the algorithms to be conducted more realistically.

A. Simulation Setup

In practice there is always some uncertainty associated with the sensor readings. A *sensor model* describes how a sensor interacts with the environment and gives some information about the properties the sensor reports. For the time-of-flight range sensors like sonar or laser, the measured value represents reflection from the nearest surface (for sonar, specular reflection of the wave is not negligible). Moravec and Elfes [24] and many other researchers interpret the sonar sensor data in a probabilistic approach using the Gaussian distribution.

In the simulations, the probability density of the laser sensor is given by a constant $\mathscr{P}_c$ plus a Gaussian function which is

multiplied by $\alpha(\rho)$, an attenuation of detection with distance:

$$\mathscr{P}(\rho|z,\theta) = \mathscr{P}_c + \frac{\alpha(\rho)}{2\pi\sigma_\rho\sigma_\theta}\exp\left[-\frac{1}{2}\left(\frac{(\rho-z)^2}{\sigma_\rho^2}+\frac{\theta^2}{\sigma_\theta^2}\right)\right] \tag{31}$$

where ρ is the range reading, θ is the angle with the optical axis of the sensor, and z is the true space range value. In Fig. 9, the vertical axis plots the likelihood of the detected object being at a given range and angle while the horizontal axes plot the distance from the range reading and the angle respectively.

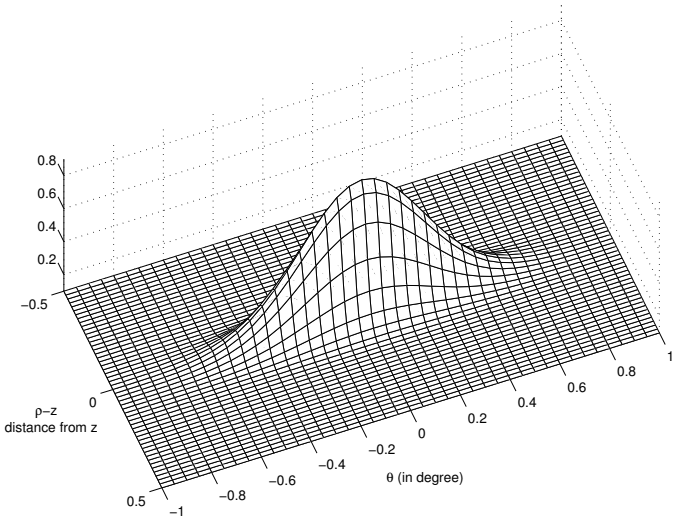


Fig. 9. The sensor model of a laser scanner ($\sigma_\rho = 0.05$m, $\sigma_\theta = 0.25$ degree, and $\mathscr{P}_c$=0.1).

As previously discussed, σ_ρ^2 is a measure of the range error. *Azimuth error variance* σ_θ^2 denotes the angular error. It can be represented by a normally distributed random variable whose 95% error bound is given by the angular resolution of the laser sensor. Then for the angular resolution $\Delta\alpha_s$, we have

$$2\sigma_\theta = \frac{\Delta\alpha_s}{2} \quad \Rightarrow \quad \sigma_\theta = \frac{\Delta\alpha_s}{4} \tag{32}$$

Two other kinds of errors are introduced to the perception of range values. Sometimes (denote the probability as $\mathscr{P}$randmax) in some directions the laser scanner erroneously obtains maximum range measurements due to laser reflection of some objects, which causes "holes" in the perception of the obstacles. The laser may also erroneously give some random measurements in some directions. This (denote the probability as $\mathscr{P}$randuni) can be modeled as getting a random range reading with a uniform distribution between 0 and the maximum range value. The jth element of the vector representation (2) is generated as Algorithm 2[2].

The following is the list of the basic settings of the simulations:

- An omni-directional robot is assumed to be used and its optimal velocity is set as $v_{\text{opt}} = 0.40$ m/s.

[2]The uniform random function RandUni$(\min, \max)$ generates a set of random variables x with a uniform distribution within the scope $[\min, \max]$. As opposed to a Gaussian function $\mathcal{N}(\bar{x}, \sigma)$, the Gaussian random function RandGau$(\bar{x}, \sigma)$ generates a set of random variables x with a Gaussian (or Normal) distribution.

Algorithm 2 Add Sensor Error to the jth Reading.

$\hat{\theta}_j \leftarrow \theta_j +$ RandGau$(0, \sigma_\theta)$ ▷ Add error to θ_j
Obtain $\rho_{\hat{\theta}_j}$, the distance of obstacles in the direction of $\hat{\theta}_j$
Begin ▷ Add noise to the jth "measured" range
  if RandUni$(0, 1.0) <$ $\mathscr{P}$randmax **then**
    $\hat{R}_j \leftarrow R_d$
  else if RandUni$(0, 1.0) <$ $\mathscr{P}$randuni **then**
    $\hat{R}_j \leftarrow$ RandUni$(0, R_d)$
  else
    $\hat{R}_j \leftarrow \rho_{\hat{\theta}_j} +$ RandGau$(0, \sigma_r)$
End

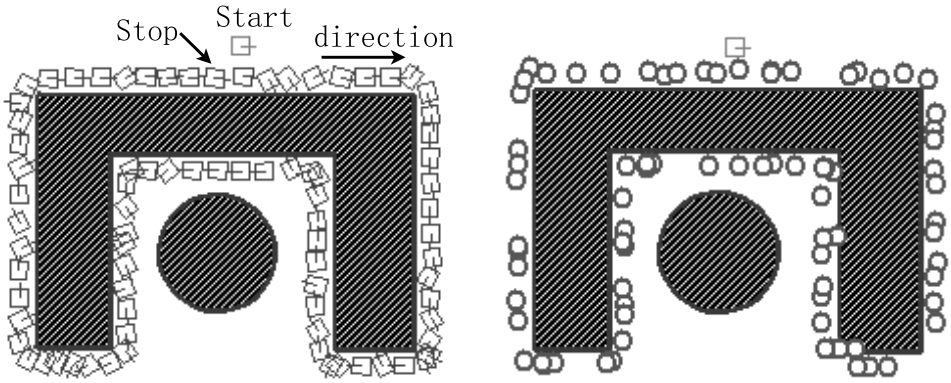
- The detectable range of the sensor is 15 m and the beam width is $\Delta\alpha_s = 1^\circ$, i.e., $N_s = 360$.
- The period between the two successive moving actions planned by the robot is set as $\Delta t = 0.5$ s and the trace of the robot motions is plotted at an interval of 10 periods.
- The obstacles can be configured to be of any shape and no model of the world was pre-created. The decisions were based directly on the range data obtained from the simulated sensor at each position.
- The parameters of sensor errors are $\sigma_\rho = 0.05$ m, $\sigma_\theta = 0.25^\circ$, $\mathscr{P}$randmax=0.01 and $\mathscr{P}$randuni=0.01.
- ε_r in Eq. (13), the threshold between the two successive measured points, is set as 0.12 m.

B. Boundary Following

In the simulations, the effective radius of $\mathcal{R}$ is set as $R_{\text{rob}} = 0.40$m. The algorithm was first tested in an environment as shown in Fig. 10. The navigation begins at the point labeled as "Start" in Fig. 10(a). The robot follows the obstacle for one full loop in a clockwise direction (labeled as a line with an arrow) and stops at the location labeled as "Stop". The final trajectories are plotted with a series of rectangles where the orientation of the robot is denoted by a straight line on each rectangle. It is shown that the robot is able to track around both convex and concave corners even with obstacles nearby. Note that the trajectories are not completely parallel to the straight lines of the obstacle because the robot does not use any reasoning about the obstacle shape it follows.

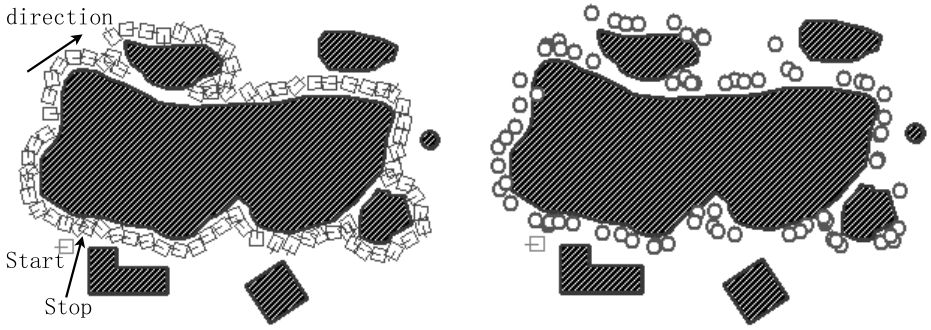
The algorithm was then tested in a densely cluttered environment as shown in Fig. 11. Initially, the robot is at the left bottom of the biggest obstacle. It then follows the obstacle for one full loop in a clockwise direction. It can be observed that the robot is able to track around the obstacle properly even with obstacles nearby. When the distance between the adjacent obstacle points is smaller than the effective size of the robot, it is reasonable to regard them as two points of the same obstacle even if they actually belong to two different obstacles. In addition, when there is a passage between two obstacles, the robot should continue following the original obstacle rather than the other one that had been detected.

The Instant Goals during the robot's motion in the two environments are represented as a series of small circles in Fig. 10(b) and 11(b) respectively. It is shown that, generally, the generated Instant Goals guide the robot move forward correctly. The Instant Goals are frequently generated or revised where they are densely plotted; sometimes they are located



(a) Trajectories. (b) Instant Goals.

Fig. 10. Following a large U-shaped obstacle consisting of convex and concave corners. A round obstacle is placed near the U-shaped obstacle.



(a) Trajectories. (b) Instant Goals.

Fig. 11. Following a complex curve with some disturbing obstacles.

very close to or even within the obstacles. This will not affect the correct motion of the robot, since collision checking is made at each time instant and the Instant Goals can be revised when they are found to be inappropriate.

C. Path Planner

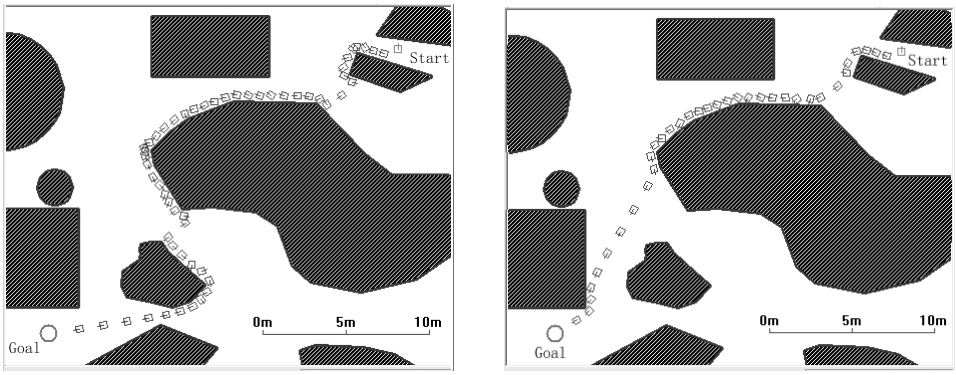
The effective radius of $\mathcal{R}$ is set as $R_{\text{rob}} = 0.30$m. The algorithm was tested in an environment with convex and concave obstacles. Figs. 12(a) and 12(b) show the trajectories of $\mathcal{R}$ under the basic leave condition and the proposed one respectively. The same approaches are used for the searching of Instant Goals and the determination of local direction. The performance of a path planner can be evaluated considering the path length and the time used. One can see that the proposed leave condition generated shorter paths (also in less time), which is generally the case for an environment of low obstacle density. However, this is not necessarily true in a highly cluttered environment, because in that case there are much less chances for $\mathcal{R}$ to detect a new obstructing obstacle rather the original one before meeting line $\overrightarrow{H_i^1 G}$.

Fig. 13 shows the result of the test in a highly cluttered environment, where diagrams (a) and (b) show the robot trajectories and the generated Instant Goals respectively. Fig. 14 shows the statistics of the errors introduced to the sensor data in this simulation test. A "measurement" is regarded as "erroneous" if the distance between the measured point detected in some direction and the true obstacle point is greater than 0.10 m ($2\sigma_\rho$). It will be regarded as "large error" if the distance is more than 0.30 m ($6\sigma_\rho$). As shown in Fig. 14(a), in each scan of 360 "measured" range values, the percentage of erroneous measurements and "large error" are around 9% and 2.5% respectively. Fig. 14(b) shows that the mean value of erroneous measurements and "large error" are around 2m and 10m respectively with the maximum range being 15m. Other simulations are also carried out using laser data which are similarly noisy.

All the tests are based on range data from the laser scanner and there is no requirement for an analytical expression of the obstacle boundaries. It is shown that during boundary following, the robot can detect a small gap and then enter into it to check if there is a passage there, which is important for the robot not to miss any passage during boundary following. It is also shown that the Instant Goal approach is suitable for discrete, noisy range data as the generation of Instant Goals does not solely rely on a one time scan and a wrong decision is able to be corrected in subsequent steps. However, continuous presence of misleading and noisy data with large errors may cause the approach to fail, e.g., moving back along the obstacle or, even worse, following a diverging path.

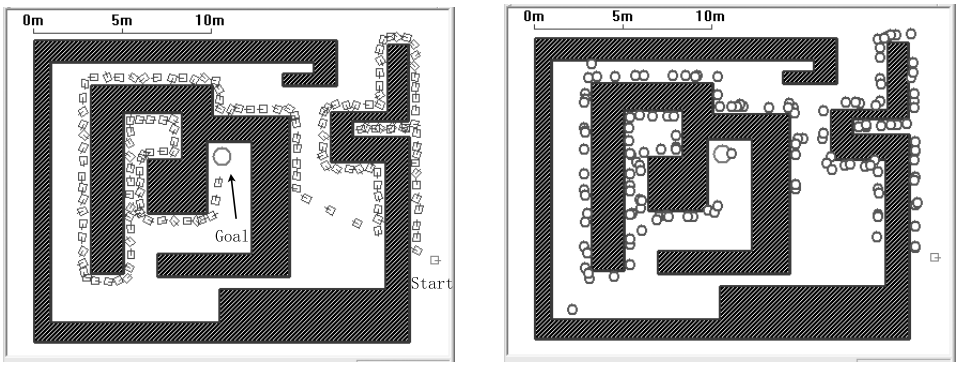
D. Comparison with Other Approaches

The sensor-based local path planners such as potential field methods and behavior-based systems use local sensory information in a purely reactive fashion and thus are usually much



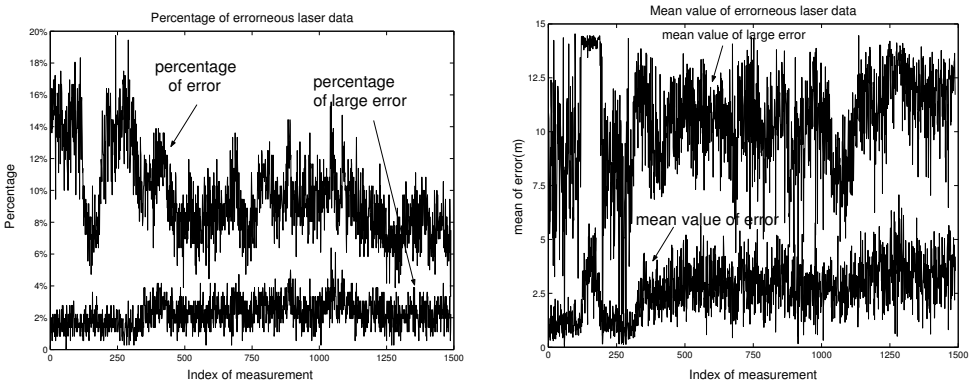
(a) Basic leave condition. (b) Proposed leave condition.

Fig. 12. The robot trajectories in a low obstacle density environment.



(a) Trajectories. (b) Instant Goals.

Fig. 13. Navigation in a high obstacle density environment.

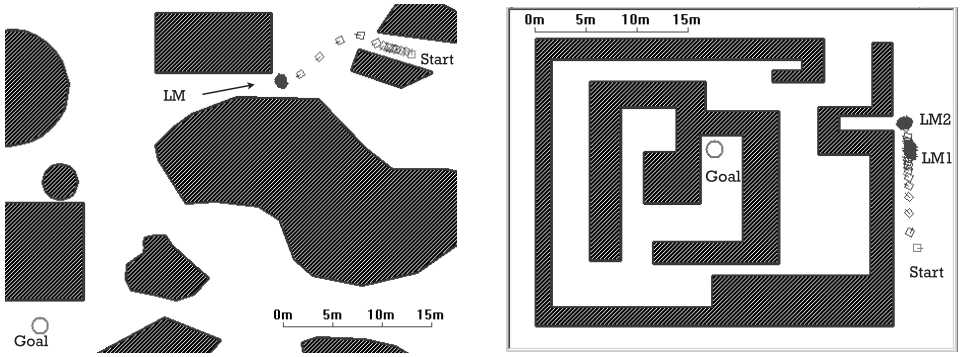


(a) Percentage of errors. (b) Mean of erroneous values.

Fig. 14. Statistics of the erroneous "simulated" measurements. The horizontal axes plot the index of laser scans. The vertical axes plot "percentage of errors" and "mean of erroneous values" in Figures (a) and (b) respectively.

simpler to implement than the global ones. However, they may get trapped in a local minimum and subsequently follow a diverging path or a loop while attempting to escape from the local minimum. Fig. 15 shows the results of navigation of the mobile robot using classical potential field methods, in the same environments as in Figs. 12 and 13. Though the algorithm itself is much simpler than the Instant Goal approach, the robot may be easily trapped at local minimums. In Fig. 15(a), the robot is stuck around $LM1$. In Fig. 15(b), at first, the robot is stuck around $LM1$ for some time and then finally trapped at $LM2$ without being able to move out of the local minimum. In the simulations, sensor errors have been added as previously but such noise still does not help the robot to go out from local minimums.

Like the Bug algorithms, the Instant-Goal-Based approach combines the global information and local sensor data and is able to achieve the convergence to the goal without facing the problem of local minimum. In addition, they do not need to maintain a global map of the overall navigation environment as the global sensor-based path planners and thus they share



(a) Low obstacle density. (b) High obstacle density.

Fig. 15. Navigation using potential field methods.

the property of real-time planning like reactive systems. The Instant-Goal-Based approach is easier to implement in the real world considering the following aspects:

(i) It does not assume a perfect capability of boundary following and an Instant Goal approach is used instead: by reaching a series of Instant Goals the robot moves "forward" along the boundary even if the obstacle is of arbitrary shape and "disturbing" obstacles are present.

(ii) The proposed approach is fully based on realistic (discrete, noisy) range data received from a rangefinder. In contrast, a range sensor was assumed to be able to present a disc of radius r_v [1][17] or to provide perfect readings of the obstacles [2][18].

(iii) The approach makes no assumption about the geometric properties about obstacles or the treatment of the robot as a point. The subgoal selection algorithm [19] requires the environment to consist of only convex polygonal obstacles, which is obviously not applicable in a real environment. Some algorithms [1][19] assume and represent the robot as a point, which results in the need to extract boundary expression from range information, and transform a physical robot into a point using the $\mathcal{C}$-space approach.

VII. Conclusion

Several issues with regards to range-based boundary following and globally convergent path planning have been resolved in this paper. Firstly, a vector representation of the local environment has been presented, which saves much memory space and is suitable for behavior generation and obstacle avoidance. Then, the Instant Goal methodology has been proposed which is able to keep the robot moving forward along an obstacle of arbitrary shape even in the presence of other disturbing obstacles. Based on this approach of boundary following, a realistic globally convergent path planner has been presented for robot navigation in an unstructured, complex environment, where the decision of the local direction utilizes all the local information available. The effectiveness of the algorithms has been validated through realistic simulations with noisy range data.

References

[1] V. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, no. 2, pp. 403–430, 1987.

[2] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 6, pp. 814–822, 1997.

[3] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[4] J. C. Latombe, *Robot Motion Planning*. London: Kluwer Academic Publishers, 1991.

[5] T. Yata, L. Kleeman, and S. Yuta, "Wall following using angle information measured by a single ultrasonic transducer," *Proceedings of 1998 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1590 – 1596, May 1998.

[6] A. Bemporad, M. D. Marco, and A. Tesi, "Sonar-based wall-following control of mobile robots," *ASME J. Dynamic Systems, Measurement and Control*, vol. 122, pp. 226–230, 2000.

[7] S. S. Ge, Y. J. Cui, and C. Zhang, "Instant-goal-driven methods for behavior-based mobile robot navigation," *Proceedings of 2003 IEEE International Symposium on Intelligent Control*, pp. 269–274, Oct 2003.

[8] A. Stentz, "Optimal and efficient path planning for partially-known environments," *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 3310–3317, 1994.

[9] H. Choset and J. Burdick, "Sensor based planning, part ii: Incremental construction of the generalized voronoi graph," *Proceedings IEEE International Conference on Robotics and Automation*, 1995.

[10] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[12] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1398–1404, Apr 1991.

[13] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[14] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[15] R. C. Arkin, "Motor-schema-based mobile robot navigation," *The International Journal of Robotics Research*, vol. 8, no. 4, pp. 92–112, 1989.

[16] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 3, pp. 464 –477, 1995.

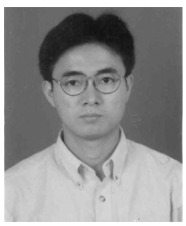
[17] V. Lumelsky and T. Skewis, "Incorporating range sensing in the robot navigation function," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 5, pp. 1058–1069, 1990.

[18] I. Kamon, E. Rimon, and E. Rivlin, "Tangentbug: A range-sensor-based navigation algorithm," *The International Journal of Robotics Research*, vol. 17, no. 9, pp. 934–953, 1998.

[19] B. H. Krogh and D. Feng, "Dynamic generation of subgoals for autonomous mobile robots using local feedback information," *IEEE Transation on Automatic Control*, vol. 34, no. 5, pp. 483–493, 1989.
[20] Y. K. Hwang and N. Ahuja, "Gross motion planning — a survey," *ACM Computing Surveys*, vol. 24, no. 3, pp. 219–291, 1992.
[21] P. Malkin and S. Addanki, "Lognets: A hybrid graph spatial representation for robot navigation," *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 1045–1050, 1990.
[22] M. Piaggio and A. Sgorbissa, "Ai-cart: an algorithm to incrementally calculate artificial potential fields in real-time," *1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 238–243, 1999.
[23] J. A. Castellanos and J. D. Tardós, *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Norwell, Massachusetts: Kluwer Academics Publishers, 1999.
[24] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 116–121, Mar 1985.

Shuzhi Sam Ge received the B.Sc. degree from Beijing University of Aeronautics and Astronautics (BUAA), in 1986, and the Ph.D. degree and the Diploma of Imperial College (DIC) from Imperial College of Science, Technology and Medicine, University of London, in 1993. He has been with the Department of Electrical & Computer Engineering, the National University of Singapore since 1993, and is currently as an Associate Professor. He has authored and co-authored over 200 international journal and conference papers, two monographs and co-invented three patents. He has been serving as Editor of International Journal of Control, Automation, and Systems since 2003. He was the recipient of the 1999 National Technology Award, 2001 University Young Research Award, and 2002 Temasek Young Investigator Award, Singapore. He serves as a technical consultant local industry. His current research interests are control of nonlinear systems, neural/fuzzy systems, robotics, hybrid systems, sensor fusion, and system development.

He has being serving as a member of the Technical Committee on Intelligent Control since 2000, is a member of Board of Governors (BOGs), IEEE Control Systems Society, in 2004. He served as an Associate Editor on the Conference Editorial Board of the IEEE Control Systems Society in 1998 and 1999. He has been serving as Associate Editor, IEEE Transactions on Control Systems Technology since 1999, Associate Editor, IEEE Transactions on Automatic Control, since 2004, Corresponding Editor for Asia and Australia, IEEE Control Systems Magazine, since 2004, and Associate Editor, IEEE Transactions on Neural Networks, since 2004.



Xuecheng Lai received the B.Eng and the M.Eng degrees from Zhejiang University, China in 1998 and 2002, respectively. He is currently a research scholar toward a Ph.D. degree in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include sensor-based robot navigation, constrained path planning, and map building for autonomous mobile robots.

Abdullah Al Mamun received B.Tech. (Hons.) degree from the Indian Institute of Technology, Kharagpur, India in 1985, and the Ph.D. degree from the National University of Singapore in 1997.

In his professional career, he worked as Research Engineer at the Data Storage Institute, Singapore and as Staff Engineer at Maxtor Peripherals prior to joining the faculty of the department of Electrical and Computer Engineering, National University of Singapore. His research interests include precision servomechanism, mechatronics, intelligent control, and autonomous mobile robots.