The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

An improved Bug-type navigation algorithm for mobile robots

Yi Zhu[1], Tao Zhang[1], Jingyan Song[1], Xiaqin Li[1], Xuedong Chen[2], Masatoshi Nakamura[3]

[1]Department of Automation, School of Information Science and Technology,
Tsinghua University, Beijing 100084, China
(Email: zhu-y07@mails.tsinghua.edu.cn)
[2]School of Mechanical Science and Engineering,
Huazhong University of Science and Technology, Wuhan 430074, China
[3]Research Institute of Systems Control, Saga University, Saga 840-0047, Japan

Abstract: An improved Bug-type navigation algorithm that ensures convergence is proposed in this paper by integrating more heuristic information abstracted from the range data in the authors' previous work. While some similar concepts have been proposed before, the improved algorithm has fully considered many implementation issues that are ignored in the related works and therefore it is more practical than these works. Simulations show that compared with the authors' previous work, the improved algorithm can generate shorter average path length. Experiments on a real robot further verified its practicability.

Keywords: mobile robot; navigation; Bug algorithm; range sensor

1 INTRODUCTION

Autonomous navigation is an important research topic in mobile robotics. In many applications, the workspace is previously unknown and the robot has to utilize the sensory data to decide its motion, which significantly increases the difficulty of navigation. A common problem of many sensor-based navigation algorithms is that the robot may be trapped before reaching the goal when it encounters obstacles with complex boundaries, which is usually termed the local minima problem [1], [2]. Many efforts have been made to solve this problem and a well-known concept is the Bug model [3] which is focused on in this paper.

In the Bug model, the robot is assumed as a point moving in a 2D plane and it has only two motion modes: moving toward the goal and boundary following. It has been proved that if the switching criteria between the two modes are properly designed, the robot can converge to the goal as long as the goal is reachable (otherwise it will report failure). Due to its simplicity and convergence proof, the Bug model has attracted much attention. A series of algorithms have then been proposed based on this model, and they are often termed the Bug algorithms [4]. However, most existing Bug algorithms mainly focus on designing the switch criteria to ensure convergence and optimize the path, but ignore the implementation issues. As a result, the applicability of the algorithm may be affected. Focusing on this problem, we have proposed a new Bug-type algorithm in [5]. This algorithm presents not only a group of new switch criteria, but also a control strategy to implement it on real robots. The control strategy is also valid for many other Bug algorithms. Furthermore, it can generate shorter path than some previous Bug algorithms.

In this paper, the aforementioned algorithm (it is termed the original algorithm below) is further improved by integrating more heuristic information abstracted from the range data to optimize the generated path. The remainder of this paper is arranged as follows: related works are briefly reviewed in Section 2; our work is presented in Section 3 and the experimental studies are presented in Section 4.

2 RELATED WORKS

Since most Bug algorithms can ensure convergence, their performances mainly differ in the path length. Hence how to shorten the generated path is the main problem discussed in most previous works. The most common strategy is to design new switch criteria and our previous work [5] followed this way. Another method is to bypass the obstacles in advance by exploiting the sensory data. This idea firstly appeared in the VisBug algorithm [6], which improves the classical Bug2 algorithm [3] by using range data to find shortcuts of the original path. Based on a similar concept, TangentBug [7] constructs a "local tangent graph" based on the range data in every control cycle to guide the robot. Undoubtedly, exploiting the sensory data is beneficial to navigation. However, these two algorithms assume that any obstacle boundary can be continuously detected. Actually, due to the limited angular resolution, the practical range sensors can only detect the nearest obstacle point in a sector as shown in Fig. 1. Hence it is difficult to well realize these two algorithms since identification errors may occur [4], [8]. In this paper, a concept similar to

VisBug and TangentBug is proposed and integrated in our previous works to further improve its performance. A significant difference is that the angular resolution problem and many other implementation issues have been fully considered in the improved algorithm.

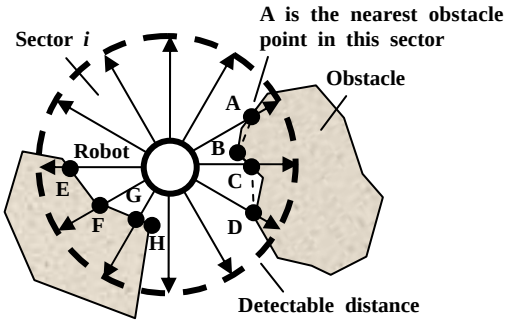


Fig. 1. The angular resolution of the practical range sensors: the robot can only detect eight obstacle points but not two continuous boundaries.

3 THE IMPROVED ALGORITHM

The improved algorithm consists of two reactive motion modes: moving toward the goal (Mode 1) and boundary following (Mode 2). They are respectively described below.

3.1 Motion toward the Goal

In Mode 1, the motion direction is selected by two steps. First, a rough direction θ_R that may lead to an optimal path is selected based on heuristic information abstracted from the range data. Then an exact direction θ_E for obstacle avoidance is calculated based on θ_R. Note that all the angles in this paper refer to the robot-fixed coordinates where the current robot direction is 0° and the angle increases in the anticlockwise direction.

In the first step, the key issue is how to abstract useful heuristic information from the range data. Assumed that the detectable view of the robot is divided into N sectors due to the angular resolution of range sensors as shown in Fig. 1 and each sector is labeled by an integer (a specific sector can be labeled as Sector 1 and the integer increases in the anticlockwise direction), an index set I that contains useful heuristic information can be defined as

$$\left\{ i \mid d_i \neq L \wedge (\left\| \mathbf{O}_i - \mathbf{O}_j \right\| \geq 2(r+D) \vee d_j = L) \wedge d_i < d_j \right\}$$

$$1 \leq i \leq N,\ j = \begin{cases} (i-1) \vee (i+1) & 1 < i < N \\ N \vee (i+1) & i = 1 \\ (i-1) \vee 1 & i = N \end{cases} \quad (1)$$

where "$\vee$", "$\wedge$" are "logic or" and "logic and", $\mathbf{O}_i$ and d_i are respectively the position of the nearest obstacle point in the i^{th} sector and the distance to this point, r is the robot radius, D is the predefined safe distance to obstacles, L is the detectable distance (if there is no obstacle in the i^{th} sector, $d_i = L$). I contains the indexes of the key obstacle points which are the endpoints of the gaps between different obstacles, e.g., the point A, D, E, H in Fig. 1. A, H are termed "left jump point" and D, E are termed "right jump point". As shown in Fig. 1, if the goal is located in the gaps, the robot can approach it directly. If the goal is located behind any obstacle, the directions of the key obstacle points correspond to the shortest path for bypassing the obstacle.

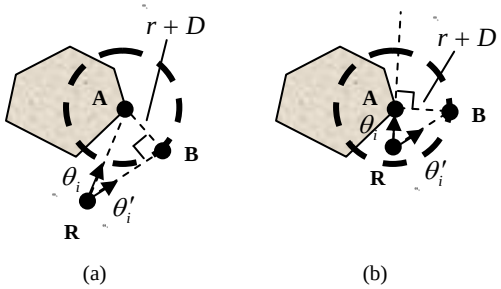
Based on the above concept, only the goal direction θ_G and the directions indexed by (1) are considered as candidates for θ_R. However, three practical issues should also be considered. First, it is not safe to directly moving to a key obstacle point. Second, it is not wise to select a direction whose corresponding gap is not wide enough to go through. Third, the possible errors caused by the angular resolution and sensor inaccuracy should be considered.

To solve the first problem, the directions indexed by (1) are slightly revised by

$$f(\theta_i) = \begin{cases} \theta_i + \Delta\theta_i & \left\| \mathbf{O}_i - \mathbf{O}_j \right\| \geq 2(r+D) \vee d_j = L \\ \theta_i - \Delta\theta_i & \text{otherwise} \end{cases}$$

$$j = \begin{cases} i+1 & i < N \\ 1 & i = N \end{cases}, \Delta\theta_i = \begin{cases} \arcsin(\dfrac{r+D}{d_i}) & d_i > r+D \\ \arctan(\dfrac{r+D}{d_i}) & d_i \leq r+D \end{cases} \quad (2)$$

where θ_i is the original direction indexed by (1). An example is shown in Fig. 2. Such a revision can guide the robot to smoothly bypass an obstacle.



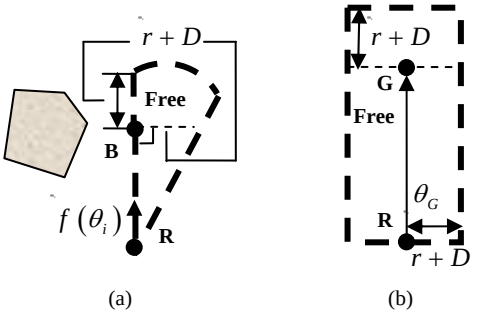
(a) (b)

Fig. 2. An example of revising the directions (R is the robot center; A is a right jump point and therefore the direction should be decreased): (a) $d_i > r + D$; (b) $d_i \leq r + D$.

To solve the second problem, a precise method requires expensive computations. Hence a rough method is proposed: if there is no obstacle in a zone near the selected direction, the gap corresponding to this direction is considered as wide enough to go through. An example is shown in Fig. 3.

As to the third problem, due to the angular resolution, the relative position between the robot and the obstacle will affect the judgment of whether an obstacle point is a jump point indexed by (1). As an example shown in Fig. 4 (θ_0 is the minimal detective angle), the robot is initially at R_1 and

it judges that B is not a jump point since D_1 is short. However, assumed that the robot moves to the jump point A, when it reaches R_2, B may be misjudged as a jump point since D_2 is long enough. Then the robot may move to B, which will cause a zigzag path. To solve this problem, it is required that if the robot selected a left (right) jump point as θ_R in the last cycle, the jump point will be recorded and in the current cycle, the robot can only select θ_R in the left (right) half-side of the vector from the current robot position to this jump point, i.e., the "admissible half-side" shown in Fig. 4. In Addition, the sensor inaccuracy sometimes also affects the judgment. Hence another constraint is added: if the robot selected a left (right) jump point in the last cycle, it cannot select a right (left) jump point in the current cycle unless it have attempted to select right (left) jump point in five successive cycles (one cycle is about 100ms in our experiments).



(a) (b)

Fig. 3. An example of roughly examining whether a gap is wide enough to go through (B is the same point as shown in Fig. 2; R is the robot center; G is the goal): (a) if the direction is indexed by (1), the examined zone is a sector on the opposite side of the corresponding obstacle; (b) if the direction is the goal direction, the zone is a rectangle.

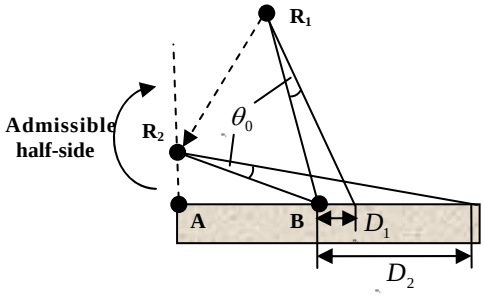


Fig. 4. An example of the misjudgment caused by the angular resolution.

Considering all the issues discussed above, θ_R is finally calculated by

$$\theta_R = \begin{cases} f(\theta_{i_0}) & I_S \neq \varnothing \wedge \mathrm{C}_1 \\ f(\angle\overrightarrow{RO}) & I_S \neq \varnothing \wedge \overline{\mathrm{C}}_1 \end{cases}, i_0 = \arg\min_{i \in I_S} \{|\theta_i - \theta_G|\}$$
$$I_S = \{ i \mid W_i \subset W_{free} \wedge |f(\theta_i) - \theta_G| < 90^\circ, i \in I \cup \{N+1\} \}$$
$$f(\theta_{N+1}) = \theta_G, W_{N+1} = W_G \tag{3}$$

where $\angle\overrightarrow{RO}$ is the angle of the vector from the current robot position to the last jump point that has been used as θ_R, f is the map function described in (2), W_i, W_G are respectively the corresponding zone of θ_i and θ_G as shown in Fig. 3, W_{free} is the free workspace, C_1 is satisfied if any following event occurs: i_0 in the current or last cycle indexes the goal direction; both i_0 in the current and last cycle index left or right jump point; no above event has occurred in five successive cycles including the current cycle. In addition, if $I_S = \varnothing$, which implies that there is no collision-free direction for approaching the goal (usually caused by a dead corner and sometimes by the sensor inaccuracy), then $\theta_R \equiv \theta_G$ until the robot switches to the mode of boundary following.

After selecting θ_R, θ_E can be calculated by

$$\theta_E = \arg\min_{\theta \in \Theta} \{|\theta - \theta_R|\}$$
$$\Theta = \{ \theta \mid |\theta - \theta_G| < 90^\circ \wedge \theta \in \Theta_S \} \tag{4}$$

where Θ_S is the current set of the directions that the robot can keep a safe distance D from obstacles. Note that if $\theta_R = \theta_G$, (4) is degraded to the criterion in the original algorithm [5]. In another word, $\theta_R \equiv \theta_G$ in the original algorithm, but there are more candidates in the improved algorithm and the one that may generate the optimal path according to the heuristic information will be selected.

If $\Theta = \varnothing$, which implies that there is no direction that can both shorten the goal distance and maintain safe for collision avoidance, the robot will switch to Mode 2 after selecting a boundary following direction dir. Such a position is termed a hit-point and each hit-point and the selected dir will be recorded in a list termed Hit-list.

3.2 Boundary Following

Mode 2 is the same as the original algorithm. The concept of finding shortcuts as VisBug and TangentBug are not adopted since it is difficult to always correctly recognize the points in the followed boundary from the range data (we have tried many methods) and it may cause navigation failure. In Mode 2, the robot will record the minimal goal distance d_{Min} and resume Mode 1 once

$$\begin{cases} 0^\circ < \theta_G < 180^\circ \wedge d < d_{\mathrm{Min}} & \text{if } dir = \mathrm{R} \\ -180^\circ < \theta_G < 0^\circ \wedge d < d_{\mathrm{Min}} & \text{if } dir = \mathrm{L} \end{cases} \tag{5}$$

where "L" and "R" represent "left" and "right" to the obstacle, d is the current goal distance. Such a position is termed a leave-point. Additionally, the boundary following direction will be reversed if a previous hit-point (not the current hit-point, i.e., the start point of the current boundary following motion) is met with the same dir as the record in Hit-list (this record is concurrently deleted). Furthermore, if the above event has not occurred after the last time that the robot passed the current hit-point but the current hit-

point is met again, the robot can stop and report that the goal is unreachable [5].

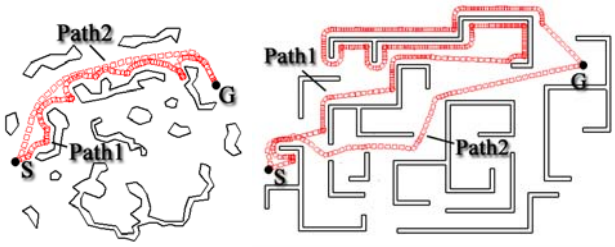
3.3 Convergence Analysis

The improved algorithm can be proved to be convergent. A proof sketch is presented below (the detail is omitted due to the limited space).

It can be proved that there are only finite mode switches and the robot can always switch to Mode 1 from Mode 2 if the goal is reachable [5]. Hence the final mode is Mode 1 if the goal is reachable. According to (4), the goal distance always decreases in Mode 1 even if there are misjudgments for selecting θ_R since $|\theta_E - \theta_G| < 90^\circ$. Therefore, the robot can always reach the goal in Mode 1 as long as it is reachable. Otherwise, it will stop and report that the goal is unreachable in Mode 2.

4 EXPERIMENTAL RESULTS

Based on the MobileSim simulation platform and a model of Pioneer3-AT robot, the improved algorithm has been tested in two environments shown in Fig. 5 with 100 randomly selected start/goal points. The two motion modes are realized by the same control strategy proposed in [5]. In all the simulations, the robot has reached the goal, which verifies the convergence of the improved algorithm. For comparison, the original algorithm has also been simulated in the same conditions. As shown in Table 1, the average path length (APL, which is presented as the relative value to the globally shortest path) generated by the improved algorithm is shorter than the original algorithm in both the environments. The reason is that compared with the original algorithm, the improved algorithm can often shorten the path by bypassing the obstacles in advance with the help of the heuristic information abstracted from the range data.



(a) (b)

Fig. 5. Two simulation results (S is the start point; G is the goal; Path1, Path2 are the paths generated by the original and improved algorithm): (a) environments containing scattered obstacles; (b) a maze environment.

TABLE 1 AVERAGE PATH LENGTH (APL) OF THE SIMULATIONS

Environment	Original Algorithm	Improved Algorithm
Scattered Obstacles	1.983	1.409
Maze	2.233	2.108

The improved algorithm has also been implemented on a real Pioneer3-AT robot. An experimental result is presented in Fig. 6: the robot successfully bypasses the obstacles in advance and finally reaches the goal.

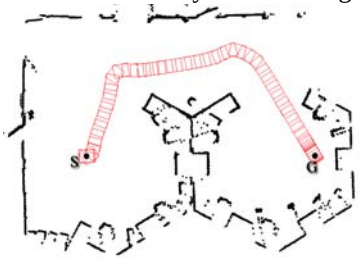


Fig. 6. An experiment on a real robot in an office environment.

5 CONCLUSION

An improved Bug-type algorithm that ensures convergence is proposed in this paper for robot navigation. It improves our previous work [5] by integrating more heuristic information abstracted from the range data to guide the robot. Compared with the similar concepts proposed in VisBug and TangentBug, the improved algorithm is more practical since it has fully considered many implementation issues that are ignored in the these works. Simulations show that it can generate shorter APL than the original algorithm. Experiments on a real robot have further verified its practicability.

REFERENCES

[1] Motlagh O R E, Hong T S, Ismail N (2008), Development of a new minimum avoidance system for a behavior-based mobile robot. Fuzzy Sets Syst. 160: 1929-1946.

[2] Zhang T, Zhu Y, Song J (2010), Real time motion planning for mobile robots by means of artificial potential field method in unknown environment. Ind. Robot 37(4): 384-400.

[3] Lumelsky V, Stepanov A (1987), Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica 2: 403-430.

[4] Ng J, Braunl T (2007), Performance comparison of bug navigation algorithms. J. Intell. Robot. Syst. 50: 73-84.

[5] Zhu Y, Zhang T, Song J, Li, X (2010), A new Bug-type navigation algorithm considering practical implementation issues for mobile robots. ROBIO: 531 - 536.

[6] Lumelsky V and Skewis T (1990), Incorporating range sensing in the robot navigation function, IEEE Trans. Syst. Man Cybern. 20(5): 1058-1069.

[7] Kamon I, Rimon E, Rivlin E (1998), TangentBug: a range-sensor-based navigation algorithm. Int. J. Robot. Res. 17(9): 934-953.

[8] Yun X, Tan K-C (1997), A wall-following method for escaping local minima in potential field based motion planning. ICAR: 421-426.