RandomBug: Novel Path Planning Algorithm in Unknown Environment

Qi-lei Xu*

College of Automation and Electronic Engineering, Qingdao University of Science and Technology, Qingdao, 266042, China

Abstract: This paper presents a novel real-time path planning algorithm for an autonomous mobile agent in completely unknown environment. In this algorithm, all the planned paths are described and stored in the form of vectors in the algorithm. Only the rotation angle and the movement distance in a single direction are considered when the autonomous moves along the planned paths. The algorithm combines range sensor data with a safety radius, which determines the blocking obstacles and calculates a shorter path by choosing the random intermediate points. These random intermediate points are be generated when blocking obstacles exist in the current path. Then the optimal intermediate points are selected and inserted into the current path to regenerate a new planned path. Simulation results are shown that the proposed algorithm is effective.

Keywords: Autonomous mobile agents, bug algorithm, complete unknown environment, path planning.

1. INTRODUCTION

Path planning for autonomous mobile agents in unknown environment needs to satisfy some requirements, such as accessibility, obstacle avoidance, optimal performance etc [1].

In terms of the path planning in completely unknown environment, common methods includes: Artificial potential field method [2-4], fuzzy logic algorithm [5, 6], and the Bug approach. Lumelsk, Mukhopadhyay, *et al.* proposed the Bug [7, 8] and the VisBug [9] algorithm. In their researches, the robot moved towards the goal position with a contact sensor or a range sensor detecting the boundary of obstacles. When the robot encountered the obstacles, it circumnavigated the obstacles until the motion towards the goal position was accessible again. This algorithm was simple, yet when the robot circumnavigated the obstacles it was unable to select an optimal circulatory direction. Like the VisBug algorithm, DistBug [10] and TangentBug [11] algorithm iterated two behaviors, motion-to-goal and boundary-following. They calculated a shorter path to the goal position by selecting an optimal circulatory direction and leaving obstacles' boundary as soon as possible. The path length generated by the TangentBug algorithm was shorter than the one obtained by DistBug algorithm, but the computational burden of the TangentBug algorithm was heavy caused by the real-time local tangent graph (LTG) calculation of the obstacles in its sensor range. In all of the Bug approaches, the robots moved along obstacles' boundary without the consideration of the safe radius of the moving robots, hence their security needs to be enhanced. Rapidly-exploring random tree (RRT) algorithm is suited for searching high-dimensional configuration space. Many variants of RRT were presented to solve the path planning in dynamic environment with unknown moving obstacles. In order to decrease the calculation burden, DRRTs [12] and MP-RRT [13] tried to effectively use the information from the previous planning result. Multi-stage probabilistic strategy was presented in [14] to further reduce the complexity of path planning in complex and dynamic environment; however its accessibility was lower than DRRTs and MP-RRT's.

In this paper, we present a novel path planning and obstacle avoidance strategy called RandomBug for an omnidirectional autonomous mobile agent in completely unknown environment. The agent computes its own path and regenerates the path by using its own sensor data to achieve the goal position without collision. In this algorithm, all the planned paths are described and stored in the form of vectors. All the vectors compose the vector path set. In an initial path vector set, there is only one vector that is generated by connecting the start position and the goal position. When the agent moves along the path vector and an obstacle blocks the planned path, several intermediate points are randomly generated. The algorithm will select the optimal point and regenerate the path vector set by inserting this point. Finally, a collision-free and feasible path is generated by constantly iterating the actions: moving along the path, detecting the obstacles, calculating the intermediate points and regenerating the path. The paper is organized as follows. The problem that we would like to solve is described in Section 2. In Section 3, our solution approach and the outline of the overall structure of the algorithm are described. In Section 4, we analyze the experimental results which show the satisfactory performance of our approach in completely unknown environment. Finally, the conclusion is presented in Section 5.

2. PROBLEM DESCRIPTION

We consider the planning path for an omnidirectional autonomous mobile agent in completely unknown environ-

*Address correspondence to this author at the College of Automation and Electronic Engineering, Qingdao University of Science and Technology, No.53, Zhengzhou Road, Qingdao, 266042, China; Tel: 13153268302; E-mail: candy_hxu@sina.com

ment. The agent is assumed as an operating point in a plane with a range sensor detecting obstacles. Assume that the range sensor could provide perfect readings of the distance between the mobile agent and the obstacles within the maximum detecting radius. The similar problem was solved in the configuration spaces [14]. The start and the goal are labeled q_{start} and q_{goal}.

The planned path is described by a vector set called path vector set. In this set, the former vector's terminal point is the latter's starting point, and every vector represents a path section which the agent should move along with. When the agent moves to the next section, it only need to steer to the next vector's direction, and then moves forward until it arrives at the next section or the goal position.

$\mathbf{P_i}$ denotes any i^{th} planned path vector set and in this set, the former vector's terminal point is its starting point.

$$\mathbf{P_i} = \mathbf{\acute{a}_{i1}}, \mathbf{\acute{a}_{i2}}, \cdots, \mathbf{\acute{a}_{in}} \tag{1}$$

where, $\mathbf{a_{ik}}(k = 1,2,L,n)$ is the k^{th} vector in the path vector set $\mathbf{P_i}$, and the terminal point of vector $\mathbf{\acute{a}_{i(k-1)}}$ is the start point of vector $\mathbf{\acute{a}_{ik}}$. $\mathbf{\acute{a}_{ik}} = (\theta_{ik}, \|\mathbf{\acute{a}_{ik}}\|)$, where θ_{ik} is the angle between the vector $\mathbf{\acute{a}_{ik}}$ and $\mathbf{\acute{a}_{i(k-1)}}$, and $-\pi \le \theta_{ik} \le \pi$. The clockwise direction is positive, and the counterclockwise direction is negative. $\|\mathbf{\acute{a}_{ik}}\|$ is the vector module, which is the distance the agent should move after it turns θ_{ik} angle.

In this algorithm, the planned for the agent is obtained by three steps. Firstly, according to the start and goal position, the initial path vector set is built. Secondly, it is determine whether there is an obstacle blocking the current path vector. When there is an obstacle blocking, the intermediate points are randomly generated. And then the path vector set is regenerated by inserting the selected intermediate point into the current path vector set.

3. PLANNING ALGORITHM

3.1. Obstacle Detection

In order to get a collision-free planned path during the process, the agent detects the obstacle in its sensor range with constant intervals. When the agent moves to a turning point of the path, it detects the obstacle immediately. Unlike in the past, the safety radius of the agent was considered to improve the safety of the algorithm.

In Fig. (**1**), the solid black line, from q_{start} to q_{goal}, represents the current path vector, q is the current position of the agent. The larger circle with hashed lines represents the sensor range and its radius is denoted by R_{mnge}. The smaller one represents the safety radius and is denoted by R_{safe}. φ is the angle between two vectors: vector (1) starting at agent's current position q_1 and terminating at the obstacle point p_o and vector (2), the current path vector. The module of vector (1), denoted by d, represents the distance between the agent and detected obstacle. When the obstacle holds two conditions: $d \cdot \sin|\varphi| < R_{safe}$ and d is not bigger than the difference between current path remnant distance and R_{safe}, we believe that an obstacle is in the current path. The first condition ensures that the agent keeps a safe distance from the obstacle.

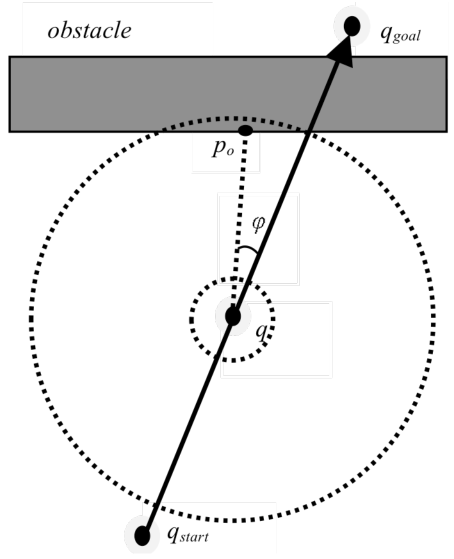


Fig. (1). Obstacle detection of a mobile agent.

3.2. Generation of Intermediate Points

Once there is an obstacle blocking the current path, several intermediate points are generated randomly.

Fig. (**2**). shows the generation of the random intermediate points. When the agent moves to the position q, it can detect the obstacle point p_o which blocks the current path. Then the random intermediate points p_1 to p_{10} are generated with the angle ranging from φ to 2π and the module ranging from R_{safe} to R_{range}. Although p_4 is in the range, p_4 should be removed from those points because it is on the obstacle. When there are more than half of the points on the obstacle, the points are regenerated randomly.

Then through calculation, we insert the selected points into the path vector set. p_i is any i^{th} random intermediate point, $\mathbf{\hat{a}_{i1}}$ and $\mathbf{\hat{a}_{i2}}$ are two vectors built by q, p_i and q_{goal}, S_i is the sum of the two vectors' modules. Take the intermediate point p_2 for instance, $S_2 = \|\mathbf{\hat{a}_{21}}\| + \|\mathbf{\hat{a}_{22}}\|$. Then calculate the minimum S_i of these points. The point with the minimum sum is selected as an intermediate point. In Fig. (**2**), p_1 is the selected as the intermediate point.

3.3. Path Regeneration

Insert the selected point into the path vector set and regenerate a new path vector set.

As shown in Fig. (**3a**), in the initial path vector set $\mathbf{P}_0$, there is only one vector formed by the line connecting the start position q_{start} with the goal position q_{goal}. The angle of the vector is zero. When the agent does not face the goal

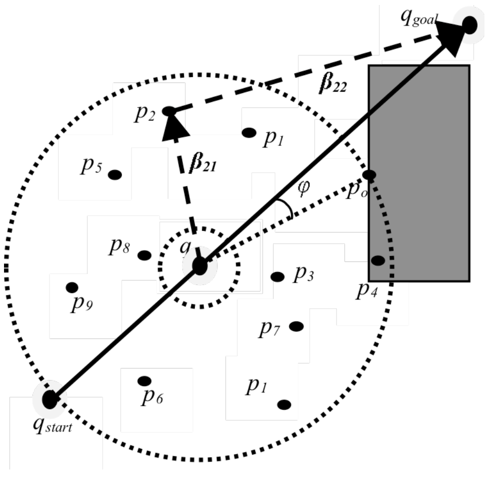


Fig. (2). Generation of intermediate points.

position, the agent is steered to the goal position in its original place.

$$\mathbf{P}_0 = \mathbf{\acute{a}}_{01} \tag{2}$$

$$\mathbf{\acute{a}}_{01} = (0, \|\mathbf{\acute{a}}_{01}\|) \tag{3}$$

Assuming an obstacle blocked the path, we insert intermediate point q_1 into initial path vector set $\mathbf{P}_0$ to get the regenerated path vector set $\mathbf{P}_1$,

$$\mathbf{P}_1 = \mathbf{\acute{a}}_{11}, \mathbf{\acute{a}}_{12}, \mathbf{\acute{a}}_{13} \tag{4}$$

$$\mathbf{\acute{a}}_{11} = (\theta_{11}, \|\mathbf{\acute{a}}_{11}\|) \tag{5}$$

$$\mathbf{\acute{a}}_{12} = (\theta_{12}, \|\mathbf{\acute{a}}_{12}\|) \tag{6}$$

$$\mathbf{\acute{a}}_{13} = (\theta_{13}, \|\mathbf{\acute{a}}_{13}\|) \tag{7}$$

where, $\|\mathbf{\acute{a}}_{11}\|$, $\|\mathbf{\acute{a}}_{01}\|$, $\|\mathbf{\acute{a}}_{12}\|$ and θ_{12} are known. $\|\mathbf{\acute{a}}_{13}\|$ and θ_{13} can be calculated based on the Cosine theorem,

$$\|\mathbf{\acute{a}}_{13}\| = \sqrt{\left(\|\mathbf{\acute{a}}_{01}\| - \|\mathbf{\acute{a}}_{11}\|\right)^2 + \|\mathbf{\acute{a}}_{12}\|^2 - 2\left(\|\mathbf{\acute{a}}_{01}\| - \|\mathbf{\acute{a}}_{11}\|\right) \cdot \|\mathbf{\acute{a}}_{12}\| \cdot \cos\theta_{12}} \tag{8}$$

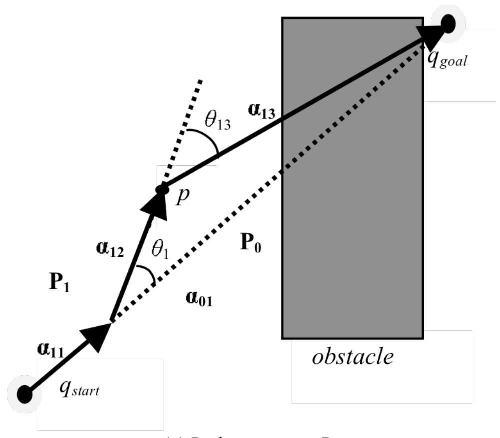
$$\theta_{12} = \cos^{-1}\left(\left(\left(\|\mathbf{\acute{a}}_{01}\| - \|\mathbf{\acute{a}}_{11}\|\right) \cdot \cos\theta_{12} - \|\mathbf{\acute{a}}_{12}\|\right) / \|\mathbf{\acute{a}}_{13}\|\right) \tag{9}$$

As shown in Fig. (**3b**), the agent keeps moving in accordance with $\mathbf{P}_1$. When the agent moves to the turning point p_1, it detects the obstacles immediately, and finds obstacle in the path. We insert intermediate point p_2 into $\mathbf{P}_1$ to get the regenerated path vector set $\mathbf{P}_2$,

$$\mathbf{P}_2 = \mathbf{\acute{a}}_{21}, \mathbf{\acute{a}}_{22}, \mathbf{\acute{a}}_{23}, \mathbf{\acute{a}}_{24} \tag{10}$$

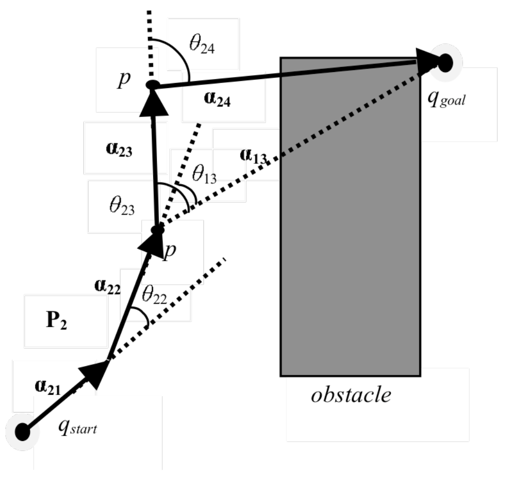
$$\mathbf{\acute{a}}_{2k} = \left(\theta_{2k}, \|\mathbf{\acute{a}}_{2k}\|\right), k = 1,2,3,4 \tag{11}$$

where, $\theta_{22} = \theta_{12}$, $\|\mathbf{\acute{a}}_{23}\|$, θ_{23} and $\|\mathbf{\acute{a}}_{13}\|$ are known. $\|\mathbf{\acute{a}}_{24}\|$ and θ_{24} can be calculated based on the Cosines theorem,



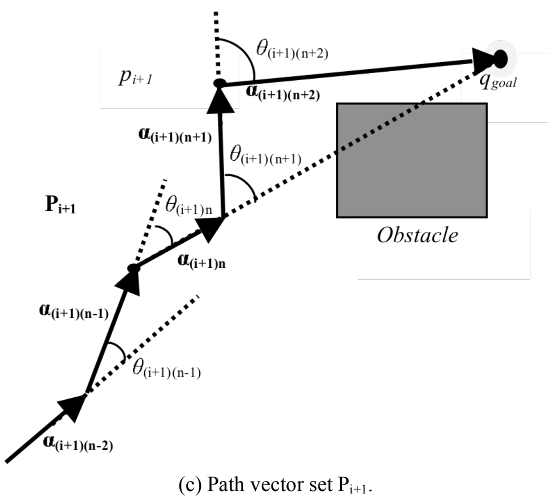
q_{goal}
θ_{13}
$\boldsymbol{\alpha}_{13}$
p
$\boldsymbol{\alpha}_{12}$
θ_1
$\mathbf{P}_0$
$\mathbf{P}_1$
$\boldsymbol{\alpha}_{01}$
$\boldsymbol{\alpha}_{11}$
obstacle
q_{start}

(a) Path vector set P_1



θ_{24}
p
$\boldsymbol{\alpha}_{24}$
q_{goal}
$\boldsymbol{\alpha}_{23}$
$\boldsymbol{\alpha}_{13}$
θ_{13}
θ_{23}
p
$\boldsymbol{\alpha}_{22}$
θ_{22}
$\mathbf{P}_2$
$\boldsymbol{\alpha}_{21}$
obstacle
q_{start}

(b) Path vector set P_2



$\theta_{(i+1)(n+2)}$
p_{i+1}
$\boldsymbol{\alpha}_{(i+1)(n+2)}$
q_{goal}
$\boldsymbol{\alpha}_{(i+1)(n+1)}$
$\theta_{(i+1)(n+1)}$
$\mathbf{P}_{i+1}$
$\theta_{(i+1)n}$
Obstacle
$\boldsymbol{\alpha}_{(i+1)n}$
$\boldsymbol{\alpha}_{(i+1)(n-1)}$
$\theta_{(i+1)(n-1)}$
$\boldsymbol{\alpha}_{(i+1)(n-2)}$

(c) Path vector set P_{i+1}.

Fig. (3). Path regeneration.

$$\|\mathbf{\acute{a}}_{24}\| = \sqrt{\|\mathbf{\acute{a}}_{13}\|^2 + \|\mathbf{\acute{a}}_{23}\|^2 - 2\|\mathbf{\acute{a}}_{13}\| \cdot \|\mathbf{\acute{a}}_{23}\| \cdot \cos\theta_{23}} \tag{12}$$

$$\theta_{24} = \cos^{-1}\left(\left(\|\mathbf{\acute{a}}_{13}\| \cdot \cos\theta_{23} - \|\mathbf{\acute{a}}_{23}\|\right) / \|\mathbf{\acute{a}}_{24}\|\right) \tag{13}$$

Similarly, we insert intermediate point p_{i+1} into $\mathbf{P_i}$ to get regenerated path vector set $\mathbf{P_{i+1}}$, and Fig. (**3c**) shows the path vectors in path vector set $\mathbf{P_{i+1}}$.

$$\mathbf{P}_{i+1} = \mathbf{á}_{(i+1)1}, \mathbf{á}_{(i+1)2}, \cdots, \mathbf{á}_{(i+1)m} \tag{14}$$

$$\mathbf{á}_{(i+1)l} = \left(\theta_{(i+1)l}, \left\|\mathbf{á}_{(i+1)l}\right\|\right),\ l = 1,2,\ldots,m \tag{15}$$

where, $\theta_{(i+1)n} = \theta_{in}$, $\left\|\mathbf{á}_{(i+1)n}\right\|$, $\left\|\mathbf{á}_{in}\right\| - \left\|\mathbf{á}_{(i+1)n}\right\|$, $\left\|\mathbf{á}_{(i+1)(n+1)}\right\|$ and $\theta_{(i+1)(n+1)}$ are known. $\left\|\mathbf{á}_{(i+1)(n+2)}\right\|$ and $\theta_{(i+1)(n+2)}$ can be calculated based on the Cosines theorem,

$$\left\|\mathbf{á}_{(i+1)(i+2)}\right\| = \Big[\left(\left\|\mathbf{á}_{in}\right\| - \left\|\mathbf{á}_{(i+1)n}\right\|\right)^2 + \left\|\mathbf{á}_{(i+1)(n+2)}\right\|^2 - 2\left(\left\|\mathbf{á}_{in}\right\| - \left\|\mathbf{á}_{(i+1)n}\right\|\right)\left\|\mathbf{á}_{(i+1)(n+2)}\right\| \cdot \cos\theta_{(i+1)(n+1)}\Big]^{\frac{1}{2}} \tag{16}$$

$$\theta_{(i+1)(n+2)} = \cos^{-1}\left(\frac{\left(\left\|\mathbf{á}_{in}\right\| - \left\|\mathbf{á}_{(i+1)n}\right\|\right)\cos\theta_{(i+1)(n+1)} - \left\|\mathbf{á}_{(i+1)(n+2)}\right\|}{\left\|\mathbf{á}_{(i+1)(n+2)}\right\|}\right) \tag{17}$$

4. SIMULATION RESULTS

In order to prove the feasibility and the optimality of the proposed algorithm, we simulate the algorithm which is programmed in C#. The simulations include the cases with different planning parameters for the same map. The simulation results are shown in Fig. (**4**). The map size is 1024×768 pixel with each pixel representing one unit length. The map is used as an unknown actual environment. The initial distance between the start position and goal position is 1250.61. In Fig. (**4**), ● represents the start position, ■ represents the goal position, and ○ represents the inserted intermediate point. The entities are obstacles, and the solid line is the moving path of the agent.

(Table **1**) shows the data of simulation results. It is assumed that the moving and obstacle sensing time is zero. Therefore the planning time is the total time cost during the path planning. It is indicated that the agent can move closer to obstacle's contour by reducing safety radius or increasing the number of random intermediate points, when the sensor range and the number of random points are constant. In a special case, with a narrow channel, the safety radius cannot be too large, otherwise the agent will not able to move through the narrow channel and the path length will increase. The path becomes smoother by increasing the number of random intermediate points, whereas the time cost and path length increases considerably. In order to avoid the local optimal, the number of random points cannot be too large. Because the modular range of random points is from safety radius to sensor range, the number of inserting points can be reduced to some extent by decreasing the sensor range. Simulation was performed in plenty of maps, and all of the agents can achieve the goal position without collision It is proved that the accessibility, obstacle avoidance, and real-time performance of RandomBug algorithm meet the requirements.

Taking $R_{range} = 200$ and $R_{safe} = 10$, ten random points, we simulated the proposed algorithm in several maps and compared with the TangentBug using sensor range as 200.

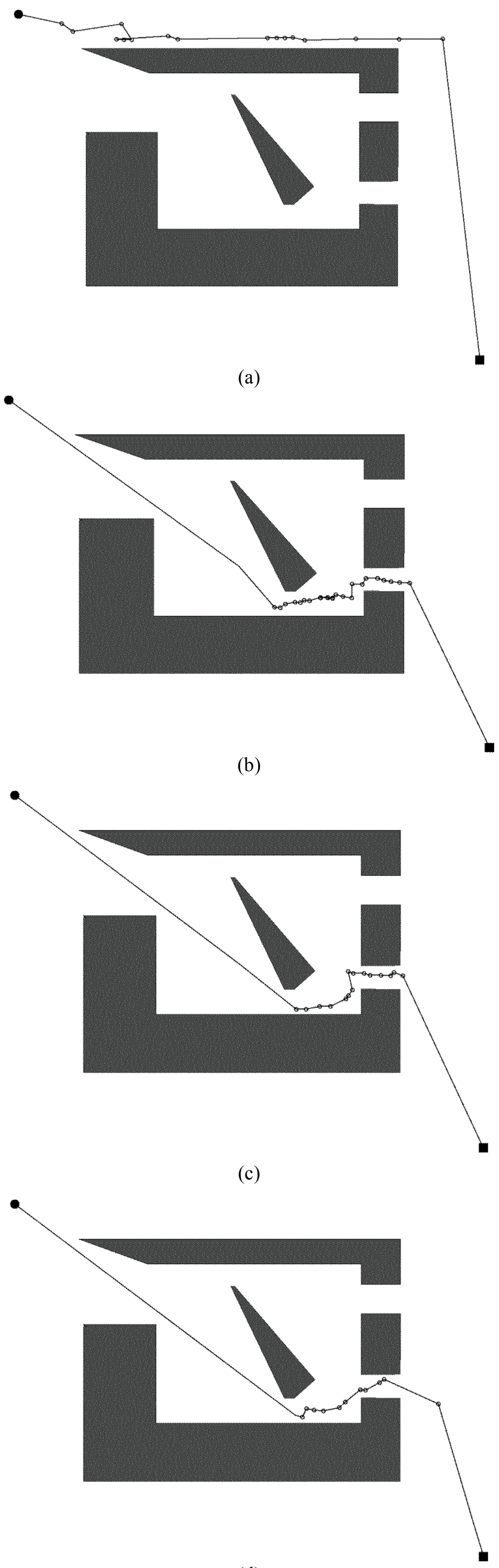


Fig. (4). Simulation results of different parameters in the same map.

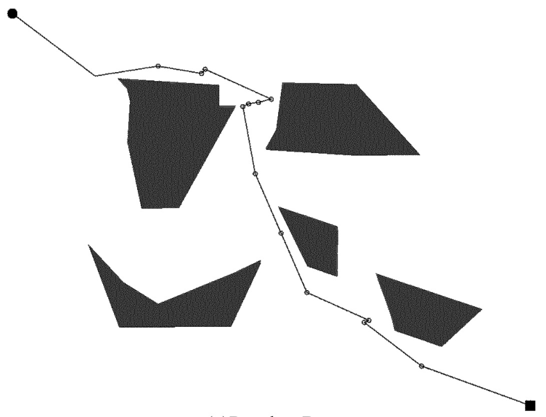
The planning results are shown in Figs. (**5**) and (**6**) with the result data listed in (Table **2**).

Table 1. **Result contrast of different parameters in the same map.**

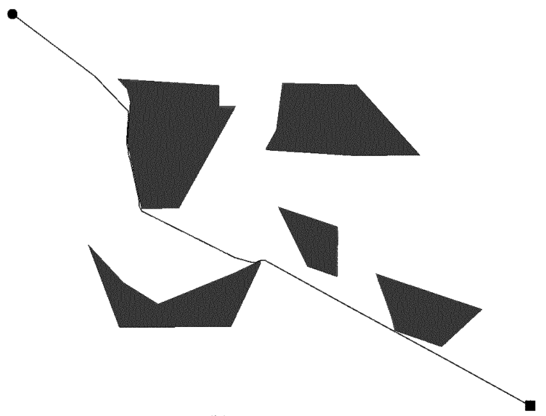
Map	R_{range}/Pixel	R_{safe}/Pixel	Random Point	Inserting Point	Path Length /Pixel	Time /ms
(a)	200	20	10	16	1716.10	19.35
(b)	300	10	10	24	1484.04	29.35
(c)	200	10	20	15	1450.83	17.83
(d)	200	10	10	11	1436.45	16.51

Table 2. **Planning results with different algorithms.**

Algorithm	Map	Path Length /Pixel	Time /ms
RandomBug	1	1497.09	11.34
TangentBug	1	1422.32	18.92
RandomBug	2	1456.74	18.46
TangentBug	2	1307.21	64.47



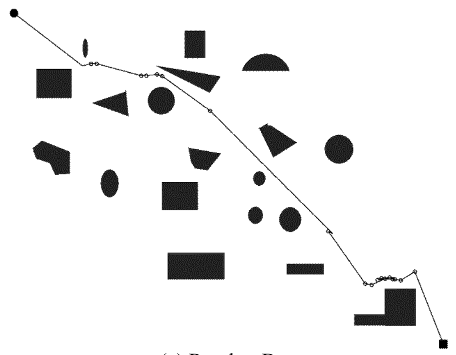
(a)RandomBug



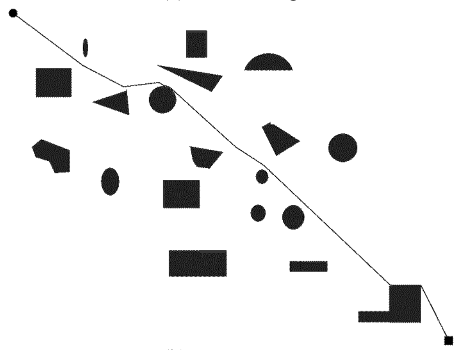
(b)TangentBug

Fig. (5). Contrast simulation results of map 1.

From Figs. (**5**, **6**) and (Table **2**), it is found that the path planned by TangentBug is close to the obstacle contour, whereas the agent and obstacles can keep a certain safe distance away from each other in the proposed algorithm. Consequently, the path planned by the proposed algorithm is longer than TangentBug.



(a) RandomBug



(b) TangentBug

Fig. (6). Contrast simulation results of map 2.

In addition, TangentBug requires continuously computing the LTG of obstacles in its 360°sensor range, which makes its computation more complex than our algorithm. The proposed algorithm surpasses TangentBug in the planning, time and cost and but as the obstacles and environment becomes more complex its superiority becomes more apparent.

CONCLUSION

A novel path planning algorithm is presented in this paper. In this algorithm, the planned paths are described and stored in the form of vectors. The path planning result is easy to execute, and the requirement of storage space is greatly decreased especially in large area and complex environment. We fully considered the safety distance between the agent and obstacles in order to improve the practicality and safety of the algorithm. The presented strategy for selecting random intermediate points only requires simple calculation, which is beneficial for agents to regenerate their path in real-time. It is proved that the proposed algorithm can achieve the path planning for agents in completely unknown environment.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by Scientific Research Foundation of Qingdao University of Science and Technology.

REFERENCES

[1] D. Dolgoy, S. Thrum, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments", *Int. J. Rob. Res.*, vol. 29, pp. 485-501, 2010.

[2] A. Masoud, "Decentralized self-organizing potential field-based control for individually motivated mobile agents in a cluttered environment: A vector-harmonic potential field approach", *IEEE Transact. Syst. Man and Cybern.*, vol. 37, pp. 372-390, 2007.

[3] P. Vadakkepat, K. Tan, and M. Wang, "Evolutionary artificial potential fields and their application in real time robot path planning", In: *Proc. 2000 Congress Evolu. Comput.*: Academic, vol. 1, pp. 256-263, 1963.

[4] P. Kim, and D. Kurabayashi, "Forming an artificial pheromone potential field using mobile robot and RFID tags", In: *IEEE/SICE Int. Symp. Syst. Integrat.*, pp. 621-625, 2011.

[5] G. Antonelli, and S. Chiaverini, "A Fuzzy-logic-based approach for mobile robot path tracking", *IEEE Transact. Fuzzy Syst.*, vol. 15, no. 2, pp. 211-221, 2007.

[6] K. Sefer, C. Omer, and K. Okyay, "Fuzzy logic based approach to design of flight control and navigation tasks for autonomous unmanned aerial vehicles", *J. Intell. Robotic Syst.*, vol. 54, pp.229-244, 2009.

[7] I. Jacobs, and C. Bean, "Fine Particles, Thin Films and Exchange Anisotropy", In: *Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York*: Academic, 1963, pp. 271-350.

[8] V. Lumelsky, S. Mukhopadhyay, and K. Sun, "Dynamic path planning in sensor-based terrain acquisition", *IEEE Transact. Rob. Automat.*, vol. 6, pp. 462-472, 1990.

[9] V. Lumelsky, and T Skewis, "Incorporating range sensing in the robot navigation function", *IEEE Transact. Syst. Man Cybernet.*, vol. 20, pp. 1058-1068, 1990.

[10] I. Kamon, and E. Rivlin, "Sensory-based motion planning with global proofs", *IEEE Transact. Rob. Automat.*, vol. 13, pp. 814-822, 1997.

[11] I. Kamon, and E. Rivlin, "TangentBug: A range-sensor-based navigation algorithm", *Int. J. Rob. Res.*, vol. 17, pp. 934-953, 1998.

[12] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs", In: *IEEE Int. Conf. Rob. Automat.*, pp. 1243-1248, 2006.

[13] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments", In: *Int. Conf. Rob. Automat.*, pp. 1603-1609.

[14] H.Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion-theory: Algorithms, and Implementation*. MIT Press, Cambridge, 2005.

Received: June 09, 2014 Revised: September 22, 2014 Accepted: November 24, 2014