International Journal of Robotics and Automation (IJRA)
Vol. 5, No. 3, September 2016, pp. 182~189
ISSN: 2089-4856

Local Path Planning of Mobile Robot Using Critical-PointBug Algorithm Avoiding Static Obstacles

Ajoy Kumar Dutta, Subir Kumar Debnath, Subir Kumar Das
Department of Production Engineering, Jadavpur University, India

Article Info

Article history:

Received May 27, 2016
Revised Aug 18, 2016
Accepted Aug 31, 2016

Keyword:

Autonomous mobile robot
Bug algorithm
Path planning
Static obstacle avoidance
Sub-goal point

ABSTRACT

Path planning is an essential task for the navigation of autonomous mobile robot. This is one of the basic problems in robotics. Path planning algorithms are classified as global or local, depending on the knowledge of surrounding environment. In local path planning, the environment is unknown to the robot, and sensors are used to detect the obstacles and to avoid collision. Bug algorithms are one of the frequently used path planning algorithms where a mobile robot moves to the target by detecting the nearest obstacle and avoiding it with limited information about the environment. This proposed Critical-PointBug algorithm, is a new Bug algorithm for path planning of mobile robots. This algorithm tries to minimize traversal of obstacle border by searching few important points on the boundary of obstacle area as a rotation point to goal and end with a complete path from source to goal.

Corresponding Author:

Subir Kumar Das,
Department of Computer Application,
Asansol Engineering College,
Vivekananda Sarani, Asansol, Burdwan, West Bengal – 713 305, India
Email: subirkrdas@gmail.com

1. INTRODUCTION

Autonomous robots currently used in the real world have been one of the major research areas in robotics and artificial intelligence. One of the main problems in robotics, called path planning of robot, is to find a free path clearing obstacles for a robot from its starting position to its destination. Obstacle avoidance is the primary requirement for any autonomous mobile robot which requires integration and coordination of many sensors and actuators [1]. The robot acquires information about its surrounding through various sensors mounted on the robot. The research in this field can be classified into two major areas: the global path planning and the local motion planning. In global path planning, coordinates of the starting point, destination point, and the obstacles are given to the robot in advance. The robot path in such applications can be calculated using this information. Local motion planning methods require less prior knowledge about the environment. The robot is dynamically guided on the basis of information about the locally sensed obstacles. Therefore this approach is more practical for mobile robots [1]. In local planning, the position of target point from its current position must be known to robot to ensure that robot can reach the destination accurately.

Several Algorithms as given by researchers include Bug Algorithms [2-4], Evolutionary Algorithms like Artificial Bee Colony Optimization [5], Ant Colony Optimization and Scout Ant Algorithm [6,7], Particle Swarm Optimization [8,9], Potential Functions [10-14], etc.

In this paper, we propose a path planning algorithm called Critical-Point Bug, to make the robot reach a specified goal point from a given start location with a target to minimize the use of outer perimeter of an obstacle. This algorithm tries to minimize the traversal time and path of a robot than "PointBug Algorithm" [4].

2. LOCAL PATH PLANNING AND BUG ALGORITHM

In local Path Planning, the area nearby the robot is unknown, or only moderately known. Sensors are used to detect the obstacles and a collision avoidance system must be integrated into the robot to avoid the obstacles. The goal where the robot should reach is known, but the shape and the position of the obstacles are indefinite. The directional angle to the goal or destination point of robot $\theta(t)$ $(0 \leq \theta(t) \leq 2\pi)$ is determined. There may be many obstacles on the plane and the only objective is to navigate the autonomous mobile robot to the destination avoiding those obstacles. To find out the best possible path the following navigation law is used [1-15]. $\dot{\theta}(t) = -\eta[\theta(t) - \theta^{*}(t)]$ here, $\theta(t)$ is current directional angle of robot, $\theta^{*}(t)$ is desirable direction angle, η is a positive constant.

Several approaches have been proposed in the literature in the past to solve the path planning problem in an unknown region. One of the widely used schemes that extensively discussed in the literature is 'Bug algorithms', the sensor-based path planning approach. Two algorithms namely Bug1 and Bug2 were proposed by Lumelsky et al [3]. This algorithm operates switching between two simple behaviors: (i) the movement towards the goal and (ii) the movement around an obstacle. Several versions of Bug algorithms have been proposed since then. The most commonly used and referred in mobile robot path planning are Bug1 and Bug2, VisBug, DistBug and TangentBug. Others bug algorithms are Alg1 and Alg2 Class, Rev and Rev2, OneBug and LeaveBug [4].

Lumelsky and Skewis proposed an improvement in the Bug2 with the VisBug incorporating a range sensor, which is an enhancement to the condition that the robot uses to stop contouring an obstacle and resume the movement to the goal, the so called leaving condition. Such improvement generates short cuts in the path [2].

Kamon and Rivlin created the DistBug which is characterized by another alteration in the leaving condition. Under certain special conditions the convergence of the DistBug can be proved. The DistBug algorithm incorporates, basically, two contributions in relation to the earlier algorithms: (i) a routine that keeps the computation cost in range but offers more aggressive leaving condition and (ii) a method to determine which side of the obstacle should be con-toured. It requires its own position by applying odometry, destination and sensor data. To ensure convergence to the goal, the DistBug algorithm needs a little amount of global information for modifying d_{min} (distance from robot to destination) and for determining that the robot finished a loop around an obstacle. The value of d_{min} can be extracted directly from the visual information. This convergence using updating d_{min} value makes problem in determining accuracy because the value of d_{min} is taken directly from user.

The TangentBug improves the DistBug and Bug2 algorithm by integrating range sensors from zero to infinity to detect obstacles. Robot will start moving around the obstacle on detection of an obstacle and as soon as it clears the obstacle will continue motion toward target point. During following boundary, it records the minimal distance to target d_{min} which determines obstacle leaving and reaching condition. The robot constructs a local tangent graph (LTG) based on its sensors' immediate readings. To decide the next motion robot continuously modified LTG and use it. The disadvantage of this algorithm is a complete 360^{0} scan is required by robot in making decision to move to the next target.

Another variation of Bug algorithm is PointBug algorithm [4] which improves the TangentBug algorithm. This algorithm tries to minimize moving around of an obstacle (obstacle border) by considering points on the outer perimeter of obstacle area as a rotating position to goal and finally create an entire path from source to objective. The main idea is fewer use of outer perimeter of obstacle area minimizes total path length taken by a mobile robot. As robot considers here the right most sudden point first, so this algorithm may take few extra times if more than one sudden point exists in an obstacle. Fig. 1 shows the different trajectories generated by Bug2, VisBug, DistBug, TangentBug and Point Bug.

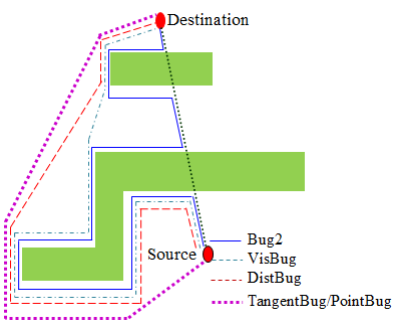

Figure 1. Trajectories Formed by different Bug Algorithm

3. CRITICAL-POINTBUG ALGORITHM

This algorithm helps to navigate a robot in a plane filled with static obstacles of unknown shape, size and location. The robot uses range sensor to detect abrupt change in distance to detect obstacle positions. Depending on the obstacle positions it calculates and determines the next point to move from current point to reach the target. We consider a possibly unbounded space $Q \subset R^2$ which is occupied by a set of bounded static obstacles $O = \{O_1, O_2, \ldots, O_K\}$. We consider a wheeled robot which is equipped with sensors to detect obstacles. The robot has its initial coordinates with reference to a global frame of reference. We solve the problem in the configuration space where the robot is represented as a point.

Before proceeding to the description of the algorithms, we make some necessary and useful assumptions and definitions for this algorithm.

3.1. Assumptions

A1. Here, the Robot is considered as Point Robot
A2. World co-ordinate system is used
A3. All points (including source and destination) are in first quadrant
A4. The velocity and angular velocity is constant in every movement and rotation respectively
A5. Surface is smooth and in same altitude
A6. All the obstacles are stationary and of any shape and size
A7. The mobile robot moves in a two-dimensional space

3.2. Sub Goal Point and Critical Point

The massive change of distance reading from range sensor output either in increasing or decreasing mode is considered for finding Sub goal point. It can be from infinity to a definite value or a definite value to infinity or definite value to a definite value where the difference value, Δd is defined. Any reading from range sensor from interval time, t_n to t_{n+1} which detects this massive change in range is considered as Sub goal Point.

The robot may scan the surroundings by range sensor from 0^0 to 360^0. The initially the robot faced straight towards goal point and then it starts scanning for sub goal point.

A sub goal point chosen by the robot for next point to move is Critical point. Generally this point has the lowest distance from destination within the set of sub goal and is not traversed yet.

We consider,

$T= \{(x_1,y_1),(x_2,y_2),\ldots,(x_i,y_i)\}$ as a set of points traversed by the robot where (x_i,y_i) represents the coordinate values

$SG= \{(\alpha_a,d_a), (\alpha_b,d_b),\ldots,(\alpha_k,d_k)\}$ as a set of next sub goal points detected by the sensor where α and d represents the angles & distances of sub goals from the robot respectively

$D= \{((x_i,y_i),\delta_i),\ldots,((x_j,y_j),\delta_j)\}$ as a set of sub goal points and distance from destination of that point

Here d_{min} is the distance from the robot to target point and ϕ is the direction of the same.

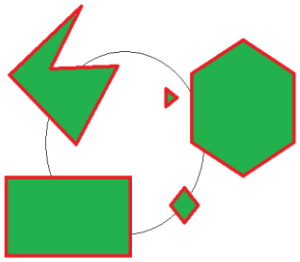

Figure 2. Obstacles detected by Range sensor(R)

The algorithm is as follows:

1. Robot Start
2. Take input of the position co-ordinates of source and destination
3. Calculate the distance and direction from source to destination d_{min} and ϕ respectively
4. WHILE not Destination
5. IF obstacle in direction
6. Find out the sub goal points using distance and angle of rotation required

7. Calculate the coordinates of sub goals from SG and save it in set D
8. Calculate distance of each sub goal and save it in set D
9. Select the coordinate having the lowest distance
10. IF the point exists in Traverse point set T
11. Discard the point
12. Select the next lowest distance from D
13. Follow step 7
14. ELSE Save the coordinate in traverse point set T
15. Calculate angle of rotation
16. Move towards sub goal
17. Get direction ϕ
18. ELSE
19. Calculate the coordinate at radius towards the direction ϕ
20. Save the coordinate in T
21. Move up to radius of vision towards direction ϕ
22. END IF
23. END WHILE
24. Robot Stop

Figure 2. Shows how a range sensor scanning obstacles with its maximum radius. The circle is the scanned area at any point of time. O_1, O_2, O_3, O_4, O_5 are the obstacles. Thin black line shows the existence of obstacles detected by the sensor within its radius. The end points of the each black line can be treated as sub goal points with distance and sensor angle.

3.3. Critical-Pointbug Algorithm Analysis

Let us consider a mobile robot as shown in Figure 5, with its starting position (x_0,y_0). The formulation considers evaluation of next obstacle free co-ordinate position of the robot. The robot knows its goal position. During its motion at any instance of time:

Let,

(x_i,y_i) – The current position of the robot

(x_{i+1},y_{i+1}) – The next possible position to move by the robot

α –Angle where sub goal point is detected by sensor

β – Robot rotation angle with respect to the line parallel to x-axis and passing through (x_i,y_i) before movement

θ – Angle generated by β with respect to the line parallel to x-axis for sub goal point coordinates calculation

d_k – Distance of a sub goal point from current location

v – Velocity of the robot

ω – Angular velocity of the robot

Four kinds of movement are possible for the robot. These are: 1. Left UP, 2. Right UP, 3. Left Down, 4 Right Down. Depending upon sub goal each four movement can make rotation of robot. This rotation for next possible movement can be classified into four sub kind which can be described pictorially

Figure 3 represents the various types of movement and Figure 4 shows how robot is generating possible next position from current position. Next point will be decided from sub goal point calculation.

Sub goal Point Coordinate Calculation:

$\beta_i = (\alpha + \beta_{i-1})\%360$ where '%' is a modulo operator

Case 1: $\beta <= 90$

$\theta = \beta$

$x_s=1, y_s=1$

Case 2: $90 \leq \beta \leq 180$

$\theta = 180^0 - \beta$

$x_s=-1, y_s=1$

Case 3: $180 \leq \beta \leq 270$

$\theta = \beta - 180^0$

$x_s=-1, y_s=-1$

Case 4: $270 \leq \beta \leq 360$

$\theta = 360^0 - \beta$

x_s=1,y_s=-1

$x_{i+1} = x_i + d_k \cos\theta(x_s)$

$y_{i+1} = y_i + d_k \sin\theta(y_s)$

x_s and y_s are the sign factors used in determination of the coordinates

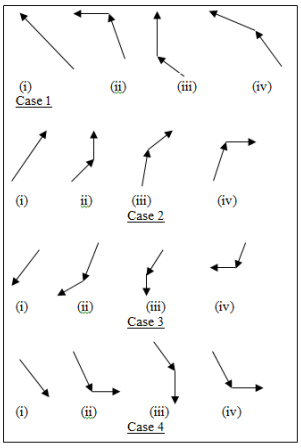

Figure 3. Different kinds of robot movements and rotation

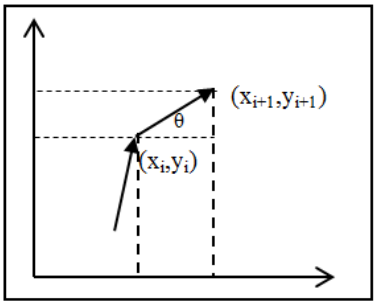

Figure 4. Current and next position of the robot

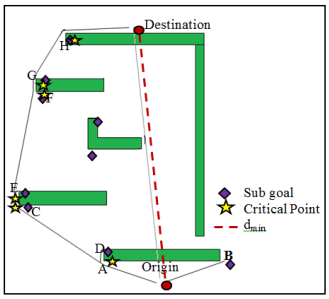

Figure 5. Trajectory of a robot using Critical-PointBug Algorithm

Figure 5 shows how a robot can reach to its destination. From source point it gets two sub goal point A & B. It selects A as Critical Point (As the distance from A is lower than B). Next it selects C from next sub goals C & D for the same reason. In this way it reaches E, F, G, H and finally destination.

Path: source→A→C→E→F→G→H→destination

Figure 6 shows Critical-PointBug algorithm can be used to handle local minima problem. The rhombus marks shows sub goal points the robot scanned. It chooses the point i as the distance is less than the distance of j from destination. Then it follows the process as described before.

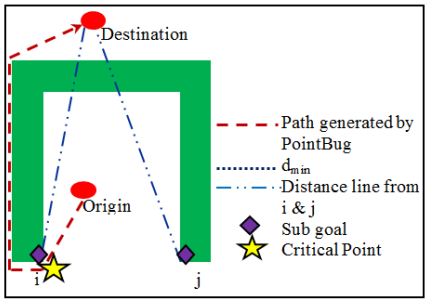

Figure 6. Critical-PointBug Algorithm solving the Local Minima Problem

3.4. Total Time And Path Length Calculation

During each movement Euclidian distance traversed by the robot is from (x_i,y_i) to (x_{i+1},y_{i+1}) is

$$d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

If the robot takes n intervals to reach its destination total path, P covered in n intervals is:

$$P = \sum_{i=0}^{n} d_i$$

In this whole path planning process other than range sensor scanning, coordinate calculation and taking decision where to move, robot mainly takes for two purposes. 1. Time taken to move from current point to next point and 2.Time taken to rotate the robot for proper alignment before leaving for next point.

1. Time taken in moving:
   If d_i and v_i are the distance covered and velocity at ith interval then the time taken during ith movement is d_i/v_i Total time taken in moving is:

$$T_M = \sum_{i=0}^{n} d_i/v_i$$

2. Time taken in rotating the robot at each interval:
   If α_i is the detection angle of critical point and ω_i is the angular velocity then time taken for rotation at ith interval is α_i/ω_i Total time taken in rotation is:

$$T_R = \sum_{i=0}^{n} \alpha_i/\omega_i$$

   Therefore the cost function will be sum of time taken in both cases i.e. moving time and rotation time:

$$C_T = \sum_{i=0}^{n} (d_i/v_i) + \sum_{i=0}^{n} (\alpha_i/\omega_i)$$

$$\therefore C_T = \sum_{i=0}^{n} (d_i/v_i + \alpha_i/\omega_i)$$

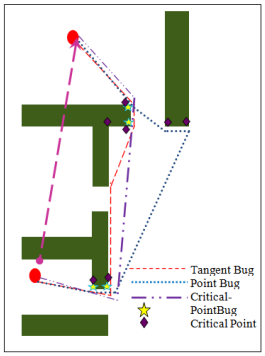

Figure 7. Path Generated by TabgentBug, PointBug
and Critical-PointBug algorithm with reference to literature [4]

4. SIMULATION AND RESULTS

The simulation of Critical-Point bug algorithm is carried out using Adobe Flash. The algorithm is simulated on environment with local minima, office like environment A and office like environment B. Figures presented here makes comparisons between the paths generated by the Critical-PointBug algorithms and other well known Bug algorithms, with the intention to analyze their similarity.

The Trajectories produced Critical-Point and other algorithms are plotted in different colors. The sub goal and critical points are also plotted using specific symbols and colors. Fig. 7 shows office like environment A with different trajectories applying Critical-Point and other bug algorithms. To complete this simulation reference is taken from literature [4]

Figure 8 presents an office like environment B. All the obstacles in the room, such as chairs, desks and walls, were represented in either rectangle or square shape.

This algorithm some time may generate long path compared to other existing path. In Fig. 5 robot chooses the sub goal A & generates the trajectory as shown in the figure. Choosing sub goal B as critical point may generate a path of comparatively shorter length. But due to lack of full information about the environment it chooses A.

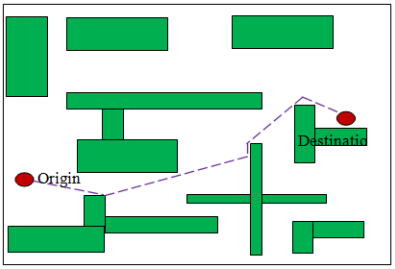

Figure 8. Trajectory generated by Critical-PointBug algorithm in office like environment B

5. CONCLUSIONS AND FUTURE WORK

We presented a simple sensor-based path planning algorithm to make a robot reach a specified destination from a given start location, in a region occupied by unknown obstacles. In this paper other path planning algorithms are studied and the Critical-PointBug algorithm is presented. This sensor based path planning algorithm works without any global data.

There may not be difference in time if environment with less complexity, precisely, with few obstacles and not many bifurcations. The algorithm considers only those obstacles' vertices that generate collisions.

Amongst the advantages of the Critical-PointBug algorithms when compared to the other methods are: (i) little iteration required to find the goal. (ii) There is no need to have knowledge about the environment. (iii) Only those obstacles will be processed for calculating sub goal and critical point, which may produce collision. (iv) The coordinate points can easily be calculated.

The algorithm is not designed to operate in dynamical environments, where the obstacles change its position during the robot movement. Future work includes both theoretical studies and practical work in this particular area.

REFERENCES

[1] R. Abiyev, D. Ibrahim, B. Erin. *Navigation of mobile robots in the presence of obstacles*. *Advances in Engineering Software*. 2010; 41:1179–1186.

[2] Ricardo A. Langer, Leandro S. Coelho and Gustavo H. C. Oliveira *K-Bug, A new bug approach for mobile robot's path planning. 16th IEEE International Conference on Control Applications Part of IEEE Multi-conference on Systems and Control Singapore.* 2007;MoC03.2:403-408

[3] K.R. Guruprasad *EgressBug: A Real Time Path Planning Algorithm for a Mobile Robot in an Unknown Environment.* P.S. Thilagam et al. (Eds.): *ADCONS 2011*. 2012; LNCS 7135:228–236.

[4] Buniyamin N., Wan Ngah W.A.J., Sariff N., Mohamad Z. *A Simple Local Path Planning Algorithm for Autonomous Mobile Robots. International journal of systems applications, engineering & development*. 2011; 5(2): 151-159

[5] Preetha Bhattacharjee, Pratyusha Rakshit, Indrani Goswami (Chakraborty), Amit Konar, Atulya K. Nagar *Multi-Robot Path-Planning Using Artificial Bee Colony Optimization Algorithm*. *Third World Congress on Nature and Biologically Inspired Computing*. 2011;

[6] Qingbao Zhu, Jun Hu, Wenbin Cai, Larry Henschen. *A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm. Applied Soft Computing*. 2011;11: 4667–4676
[7] Alpa Reshamwala, Deepika P Vinchurkar. *Robot Path Planning using An Ant Colony Optimization Approach: A Survey. (IJARAI) International Journal of Advanced Research in Artificial Intelligence*. 2013;2(3): 65 – 71
[8] Narendra Singh Pal, Sanjeev Sharma, *Robot Path Planning using Swarm Intelligence: A Survey. International Journal of Computer Applications (0975 – 8887)*. 2013;83(12): 5 – 12
[9] Chengyu Hu, Xiangning Wu, Qingzhong Liang and Yongji Wang. *Autonomous Robot Path Planning Based on Swarm Intelligence and Stream Functions*. Springer *Verlag Berlin Heidelberg ICES 2007, LNCS 4684*. 2007; LNCS: 277–284.
[10] Wesley H. Huang, Brett R. Fajen, Jonathan R. Fink, William H. Warren. *Visual navigation and obstacle avoidance using a steering potential function. Robotics and Autonomous Systems*. 2006;54: 288–299
[11] S. S. Ge and Y. J. Cui. *New Potential Functions for Mobile Robot Path Planning. IEEE Transactions on Robotics and Automatio*, 2000; 16(5):615 – 620
[12] Dusan Glavaski, Mario Volf, Mirjana Bonkovic. *Mobile robot path planning using exact cell decomposition and potential field methods. WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS*. 2009;8(9):789-800
[13] S.S. Ge, Y.J. Cui. *Dynamic Motion Planning for Mobile Robots Using Potential Field Method*. *Autonomous Robots* 2002; 13: 207–222.
[14] M. Deng, A.Inoue, K.Sekiguchi, L. Jiang. *Two-wheeled mobile robot motion control in dynamic environments*. *Robotics and Computer-Integrated Manufacturing*. 2010; 26: 268–272.
[15] Atsushi Fujimori, Peter N. Nikiforuk, Madan M. Gupta. *Adaptive Navigation of Mobile Robots with Obstacle Avoidance*. *IEEE Transactions On Robotics And Automation*, 1997; 13(4): 596-602.

BIOGRAPHIES OF AUTHORS

Dr. Ajoy Kumar Dutta is currently a Professor in the Department of Production Engineering, Jadavpur University, INDIA. He received his B. E. & M. E. degrees in Electronics & Tele-communication Engg from Jadavpur University in 1983 & 1985 respectively, and Ph. D. (Engg) degree in the area of Robotics from Jadavpur University in 1991. His Field of Specialization and Research Area are Robotics, Sensors, Computer Vision, Microprocessor Applications, and Mechatronics. He has teaching & research experience of 31 years.

Mr. Subir Kumar Debnath is currently an Associate Professor in the Department of Production Engineering, Jadavpur University, INDIA. He received his B. E. degree in Mechanical Engg from Jadavpur University in 1982 & M. Tech in Mechanical Engg in 1984 from I.I.T.- Kharagpur, INDIA. His Field of Specialization and Research Area are Robotics, Sensors, Computer Vision, CNC Machines and Automation. He has teaching & research experience of 31 years.

Subir Kumar Das received M.Tech Operations Research in 2010 and M.Sc. Computer Science in 2007. He is currently pursuing a Ph.D. from Jadavpur University of India. His research interests include computer vision system, autonomous mobile robots, optimisation technique.