OperativeCriticalPointBug algorithm-local path planning of mobile robot avoiding obstacles

Subir Kumar Das[1], Ajoy Kumar Dutta[2], Subir Kumar Debnath[3]
[1]Department of Computer Application, Asansol Engineering College, India
[2,3]Department of Production Engineering, Jadavpur University, India

Article Info

Article history:

Received Sep 7, 2019
Revised Nov 9, 2019
Accepted Nov 23, 2019

Keywords:

Autonomous mobile robot
Bug algorithm
Critical point
Obstacle avoidance
Open point
Path planning

ABSTRACT

For Autonomous Mobile Robot one of the biggest and interesting issues is path planning. An autonomous mobile robot should be able determine its own path to reach destination. This paper offers a new algorithm for mobile robot to plan a path in local environments with stationary as well as moving obstacles. For movable robots' path planning OperativeCriticalPointBug (OCPB) algorithm, is a new Bug algorithm. This algorithm is carried out by the robot throughout the movement from source to goal, hence allowing the robot to rectify its way if a new obstacle comes into the route or any existing obstacle changes its route. According as, not only the robot tries to avoid clash with other obstacle but also tries a series of run time adjustment in its way to produce roughly a best possible path. During journey the robot is believed to be capable to act in an unknown location by acquiring information perceived locally. Using this algorithm the robot can avoid obstacle by considering its own as well as the obstacle's dimension. The obstacle may be static or dynamic. The algorithm belongs to bug family.

Corresponding Author:

Subir Kumar Das,
Department of Computer Application,
Asansol Engineering College,
Asansol, West Bengal, India.
Emails: subirkrdas@gmail.com

1. INTRODUCTION

Autonomous Mobile Robot is a machine which has the capability to navigate in a complex environment without the requirement any kind of supervision from any device or human being. Robot path planning is the task of determining collision-free path during a robot travels from an initial position to a destination position in an unknown or partially known locality to accomplish its objective. This task involves a series of computation to generate a non colliding path. To accomplish fully independent navigating path planning is the one of the major criteria for mobile robots. It includes searching a geometric path from the origin location of the robot to the destination.

While discussing about path planning the property of environment or the locality where the robot will work is an important issue. Depending on the environment path planning can be categories in two type: 1) Static environment – where the obstacles are static or what we can say the whole environment is static in nature and 2) Dynamic environment – the environment is dynamic, means the object in the environment may be static or may be dynamic. In this dynamic environment path planning comes in two form a) Global path planning – where the information about the environment is known to robot before it starts moving and b) Local path planning – where the robot doesn't have any information regarding the environment. This is sensory based path planning where the robot uses sensors to get information about the obstacles to avoid is and reach destination.

Moving obstacle avoidance at any time comes under dynamic path planning [1-3]. Artificial Potential Field is one of the widely used techniques to avoid obstacle in dynamic environment [4, 5]. There are many techniques given by many researchers to avoid dynamic obstacle and plan a new path. Md. Arafat Hossain and Israt Ferdous introduced bacterial foraging technique for path planning [6]. Dongsheng Zhou, Lan Wang and Qiang Zhang gave an idea of obstacle avoidance using ant colony algorithm [7]. In many cases of path planning the ultimate trajectory may not remain same as planned. The final path deforms while generating the trajectory [8]. This happens due to many reasons such as error in calculation of dimensions of robot itself and obstacles. During navigation it is necessary to consider the dimensions of the robot as well the obstacles, particularly when the functioning domain of the robot is complex. Robert L. Williams II and JianhuaWu gave an idea of Dynamic Obstacle Avoidance for an Omni directional Mobile Robot considering the dimensions [9]. Instead of obstacle avoidance Muhannad Mujahed et. al. uses collision avoidance approach using gap navigation [10]. How to plan motion for dynamic obstacles with Uncertain Motion Patterns is approached by Georges S. Aoude et. al. [11]. A mobile robot may have single or multiple trails with the main robot body (Snake Like). In those cases path planning are little bit complex [12-14]. Apart from these there are few other techniques for obstacle avoidance. Few of them are classical techniques or a variation of existing classical techniques like Equilateral Space Oriented Visibility Graph (ESOVG) [15] etc and few innovative like TDOA (Time Difference of Arrival) algorithm [16] etc. The bug algorithms [17-25] are bio-inspired path planning algorithm.

"Critical-PointBug Algorithm" [22] is used to avoid static obstacle in dynamic environment. Furthermore this is a path planning algorithm for a point robot. But in real world a robot used to have dimension. This paper attempts to develop more the "Critical-PointBug Algorithm" [22] in the said area.

In this paper, the proposed scheme tries to keep the robot away from both fixed and dynamic obstacles. The bug algorithm is a modification of existing "Critical-PointBug Algorithm" for path planning in dynamic situation for a robot having dimension. The projected algorithm is named as OperativeCriticalPointBug (OCPB). Using this algorithm a robot can reach to a particular destination point avoiding any moving and stationary obstacles considering its own dimension as a constraint with an aim to lessen the distance travel and time to arrive at target.

2. PROPOSED METHOD

The Bug algorithms are very well known, absolute and simple algorithms used in local path planning and mobile robot navigation with minimum sensors. To accomplish its goal the robot uses as little global information as possible. The main logic is very simple: 1. move towards goal until an obstruction is revealed. 2. If there is obstruction then contour the obstruction until goal is once more achievable. 3. If goal point is visible move towards it again. In Bug family more than twelve different bug algorithms exist and each algorithm of which carries its own conncluding process. Names of Few bugs are Bug1, Bug2, Alg1, Alg2, DistBug, VisBug, TangentBug, Class1, Rev, Rev2, OneBug, LeaveBug, PointBug, K-Bug, Critical-PointBug etc.

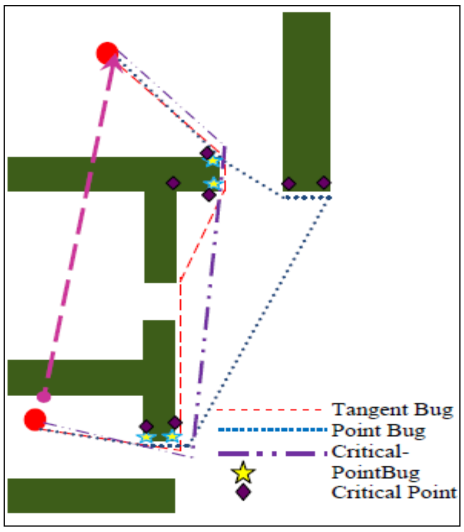

Figure 1. Trajectories formed by different Bug Algorithm with reference to literature [22]

The CriticalPointBug algorithm always avoids obstacles by finding and moving to a point which is closer to goal point than others and thus gives us an optimal path. The trajectories of CriticalPointBug and other bug algorithm is shown in Figure 1. It is clear from the figure that how this algorithm performs better than other existing bug algorithms. Though this is an efficient algorithm but consists of some drawback also. The algorithm assumes robot as point robot. But in real world a robot may be 2D or 3D but must have dimension. It is local path planning in presence of obstacles which are static in nature. Currently proposed attempts to develop the algorithm and considers a robot with dimension. The algorithm is on local path planning avoiding static as well as moving obstacles. The algorithm is also developed from ModifiedCriticalPointBug algorithm which can be used to avoid dynamic obstacle but the robot is point robot. The OperativeCriticalPointBus algorithm uses reference from [22] to find the sub goal points and critical point for avoiding obstacles and move to the next point to reach at goal.

2.1. OperativeCritical-PointBug Algorithm

The algorithm is supposed to carry out on a robot named Activity Bot. The Activity-Bot is a compact, zippy robot which consists of a multi-core Propeller microcontroller along with great hardware. Figure 2. shows the robot and Figure 3. Shows the how a range sensor detects an obstacle. The robot scans the locality to notice the presence of any obstacle.

Here we are considering the robot in a 2D plane for simulation purpose. Let us assume a unbounded space $Q \subset R^2$ that contains a set of bounded stationary and moving obstacles $O = \{O_1, O_2, . . ., O_K\}$. We consider the robot in rectangular shape with wheel and other equipment as shown in Figure 4. As per global frame of reference the robot knows its initial coordinates. Before going in to the full discussion of the algorithms, we consider few essential and helpful assumption and description for this algorithm which are referenced from [22]. Figure 5 shows how obstacles are detected by sensor. The obstacles are O_1, O_2, O_3, O_4, O_5. The obstacles identified by the sensor are shown using black lines. The open points are the finishing points of each black line.

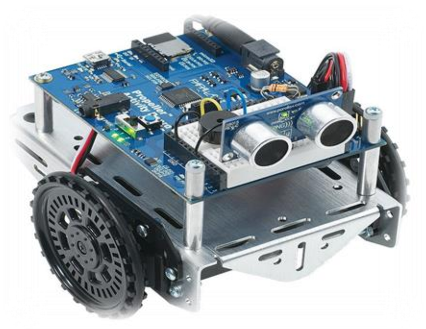
Figure 2. Parallax activity bot

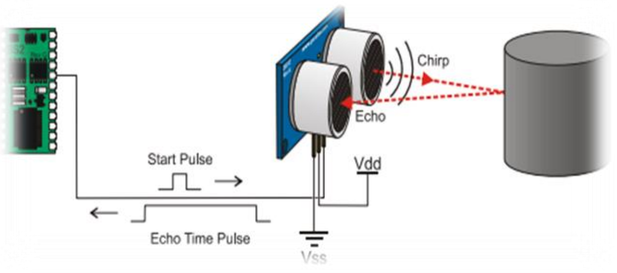

Figure 3. Working principle of ultrasonic range sensor

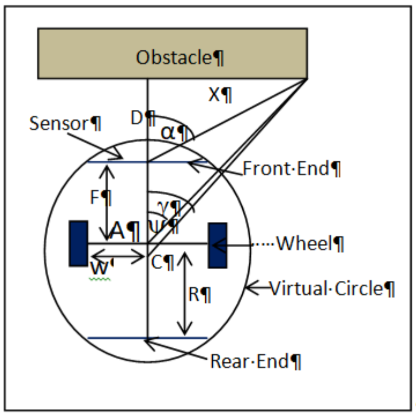

Figure 4. Display of robot and obstacles in 2D plot

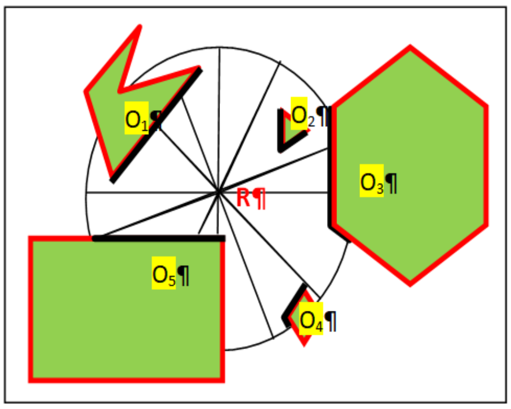

Figure 5. Obstacles detected by Range sensor (R)

2.1.1 Assumptions
A1. World co-ordinate system is used
A2. All points (including source and destination) are at first quadrant
A3. The velocity and angular velocity of robot is constant in every movement and rotation respectively
A4. Surface is smooth and in same altitude
A5. The mobile robot travel in a two-dimensional area and rotates without slipping
A6. Both the robot and dynamic obstacles are run in constant speed and in straight line. If anyone wants to change direction it has to stop then turn then again start moving

2.1.2 Essential Definitions
We consider on bounded or unbounded space O and Q at some stage in any reading from range sensor within a time period, t_n to t_{n+1} if it detects the difference Δd in range at any point then that point is considered as Open Point where Δd is defined. The robot scans the surroundings 0^0 to 360^0 by range sensor. The primarily the robot facing directly to goal point and afterward it begins scanning for open point. Depending on the coordinate of open point sub goal point is calculated. The sub goal point situates at a specific distance from the open point and possibly makes a perpendicular at open point to the line joining the sensor point and open point. This sub goal point is intermediary target of the robot to reach at final target. From a set of sub goal points the robot chooses one point for next move which is called critical point. The robot tries to move its center of virtual circle to critical point. This point is chosen on the basis of minimum distance from the destination and yet not visited.

We plot the robot and the obstacle in 2D plane and assume a virtual circle surrounding the robot. Centre of the virtual circle is located at the midpoint of the two farthest points of the robot and the radius is half between the two farthest points of the robot+λ where λ is a safety constant. The minimum value of λ can be obtained from few number of experiments. Figure 4 shows the plotting and other required geometrical details.

D – Perpendicular distance of obstacle from the sensor
A – Location of robot wheel-'axle' (imaginary line) center
X – Distance from sensor to open point
w – Distance between wheel-'axle' center and wheel center
F – Distance between the rear end and wheel 'axle' center
R – Distance between the wheel 'axle' center and the rear end of the robot
C – Center of the virtual circle
α – Open point detection angle by the sensor
λ - Safety constant for the robot
r – Radious of the virtual circle
β – Sensor direction angle at open point 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
ψ– Angle created on a line parallel to x axis and passing through sensor by a line from sensor to open point
γ – Angle between line joining open point & subgoal and the line parallel to x-axis and passing through open point
d_k – Distance of a sub goal point from current location
x_s and y_s – are the sign factors used in determination of the coordinates of open points
s_x and s_y – are the sign factors used to determine the coordinates of sub goal points
We consider,
T= $\{(x_1,y_1),(x_2,y_2),...,(x_i,y_i)\}$ as a set points navigated by the robot where (x_i,y_i) symbolize the coordinate values
OP=$\{((\alpha_a,d_a),(\alpha_b,d_b)),....,((\alpha_k,d_k),(\alpha_j,d_j))\}$ as a set of subsequent open points of obstacles identified by the sensor where α and d symbolize the angles & distances of open points respectively from the robot and each $((\alpha_a,d_a),(\alpha_b,d_b))$ represents open points connected to each obstacle, if only one point is detected then $\alpha_a=\alpha_b$ and $d_a=d_b$
SG= $\{(a_i,b_i),.....,(x_j,y_j)\}$ as a set of sub goal points of obstacles identified by the robot and each (a_i,b_i) represents sub goal point which the can choose for next move
T_{obs}= $\{((a_i,b_i),...(a_j,b_j)),.....,((x_i,y_i),..,(x_j,y_j))\}$ as a set of provisionaly identified obstacles where each $((a_i,b_i),...(a_j,b_j))$ are the set of points of each obstacle.
S_{obs}= $\{((a_i,b_i),...(a_j,b_j)),.....,((x_i,y_i),..,(x_j,y_j))\}$ as a set of stationary obstacles where each $((a_i,b_i),...(a_j,b_j))$ are the set of points of every stationary obstacle.
TD_{obs}= $\{((a_i,b_i),...(a_j,b_j)),.....,((x_i,y_i),..,(x_j,y_j))\}$ as a set of provisionaly dynamic obstacles where each $((a_i,b_i),...(a_j,b_j))$ are the set of points of each temporary dynamic obstacle.

D_{obs}= {((a_i,b_i),…(a_j,b_j)),…..,((x_i,y_i),..,(x_j,y_j))} as a set of moving obstacles where each ((a_i,b_i),…(a_j,b_j)) are the set of points of each moving obstacle.
D= {((x_a,y_a),δ_a),…,((x_j,y_j),δ_j)} as a set of sub goal points and distance from target of that point where each set ((x_a,y_a),δ_a) represents the set of sub goal point and their distance from target
Here d_{min} is the distance from the robot to goal point and ϕ is the direction of the same.
P_C is the position where the robot may collide with the obstacle and the distance from current position of robot to the point of collision is D_C.

In Cartesian space we can represent the robot state as $q=[x, y, \omega, v, t]^T$, where (x, y) are the coordinate of centre, robot's angular velocity, the speed of the mobile robot is v and t is the time.

2.1.3 Algorithm

MAIN Procedure

```
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 $\phi$ respectively
4.  WHILE not Destination
5.      Start OBSTACLE_DETECTION procedure in FOV
6.      IF obstacle or virtual obstacle in direction
7.          Calculate the coordinates of sub goals from OP, don't calculate same set from OP twice and
            save it in set SG and D
8.          Calculate distance of each sub goal from destination and save it in set D
9.          Select the coordinate point P having the lowest distance in D
10.         IF the point exists in Traverse point set T
11.             Discard the point
12.             Select the next lowest distance point P from D
13.             Follow step 10
14.         ELSE
15.             IF P is SAFE_POINT
16.                 Save the coordinate in traverse point set T
17.                 Calculate angle of rotation and rotate
18.                 Move towards the critical point
19.                 Calculate direction $\phi$ and distance $d_{min}$
20.             ENDIF
21.         ENDIF
22.     ELSE
23.         Calculate the next point coordinate P towards the direction $\phi$
24.         IF P is SAFE_POINT
25.             Save the coordinate in T
26.             Move to P
27.             Calculate new $d_{min}$
28.         END IF
29.     END IF
30. END WHILE
31. Robot Stop
```

OBSTACLE_DETECTION Procedure

```
1.  Identify obstacles in FOV
2.  Calculate the coordinates of open points and save those points in set $T_{obs}$
3.  FOR any coordinate in $T_{obs}$
4.      IF coordinate set of $T_{obs}$ exists in $S_{obs}$
5.          Discard the coordinate from $T_{obs}$
6.      ELSE
7.          IF there exists any nearest point in $S_{obs}$ with small change in x or y value
8.              The obstacle may be dynamic
9.              Save coordinate in $TD_{obs}$
10.         ELSE
11.             IF there exists any nearest point in $TD_{obs}$ with small change in x or y value
12.                 It is part of moving obstacle
```

```
13.             Save the points of $TD_{obs}$ and $T_{obs}$ to the robot registry and all the open points of the
    obstacle    in $D_{obs}$
14.           ELSE
15.             Save the points in $S_{obs}$
16.             RETURN obstacle
17.           END IF
18.         END IF
19.       END IF
20. END FOR
21. IF obstacle dynamic
22.     Calculate direction $\xi$ and speed $v_{obs}$ of the obstacle   from the points exists in robot registry
23.     IF the obstacle direction $\xi$ intersecting robot direction $\phi$
24.         Calculate probable position $P_C$ of collision
25.         Calculate probable distance $D_C$ from the  robot to $P_C$
26.           IF $D_C<d_{min}$
27.             Calculate the time $R_{TC}$ and $O_{TC}$ both the robot and obstacle will take to reach at $D_C$
                respectively
28.             IF $R_{TC}=O_{TC}$
29.                 Identify the point $P_C$ and all the  adjacent points of the obstacle as virtual Obstacle
30.                 Calculate the open points and save in OP
31.                 RETURN virtual obstacle
32.             ENDIF
33.           END IF
34.     END IF
35. END IF
```

```
SAFE_POINT PROCEDURE
1.  IF P corresponds to a sub sub set of OP
2.      Select the sub set from OP that corresponds to P
3.      IF it is first sub set
4.        Select the second sub set from previous set from OP
5.      ELSE
6.        Select the first sub set from next set from OP
7.      END IF
8.      Calculate the distance Sd between two sub sets
9.  ELSE
10.     Select the two points from OP having minimum distance from P
11.     Calculate the distance Sd between two points
12. END IF
13. IF Sd > 2r
14.     RETURN safe
15. ELSE
16.     RETURN not safe
17. ENDIF
```

3. DISCUSSION AND RESULT

The foremost purpose of the algorithm is to produce an uninterrupted path from initial to end points and these points remain fixed for any particular case. When the mobile robot starts to traverse through the environment, it will start bearing in mind unidentified stationary obstacles. The general observation of avoiding an unidentified stationary obstacle as per [22] is shown in Figure 1.

This algorithm tries to avoid dynamic obstacle also. Not only that unlike algorithm of [22] and its next other algorithm this algorithm considers the robot as a whole body not a point robot and then avoid the obstacles using safety precaution. Figure 6 shows how a robot can be considered under a virtual circle for obstacle avoidance. Figure 7 shows the open points and its associate sub goal points. Figure 8 reflects how the robot is calculating value of a sob goal point using formula. All the sub goals situate at a distance of virtual circles radius from the open point and at perpendicularly opposite direction of the line joining the sensor point and associated open point.

3.1. Analysis of Algorithm

Every obstacle, it may be dynamic or static, but is static at a particular point of time. After small time Δt, if the position of an obstacle remains same then it is static otherwise dynamic. Anticipating the probable position of contact with dynamic obstacle from its direction and speed and then using proper action for avoidance is necessary in dynamic environment. Figure 9 and Figure 10 explains how we can avoid dynamic obstacles using this algorithm by estimating the virtual obstacle position.

Let an open point is detected by sensor at angle α. As per [22]

$$\therefore \beta_i = (\alpha + \beta_{i-1})\%360 \quad (1)$$

$$\left.\begin{aligned} x_{i+1,j} &= x_i + d_k\cos\theta(x_s) \\ y_{i+1,j} &= y_i + d_k\sin\theta(y_s) \end{aligned}\right\} \quad (2)$$

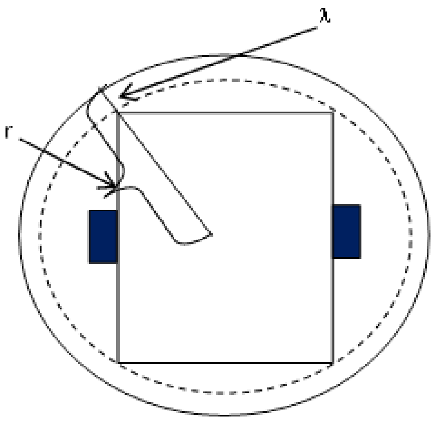

Figure 6. Whole robot inside the virtual circle with safety measurement λ

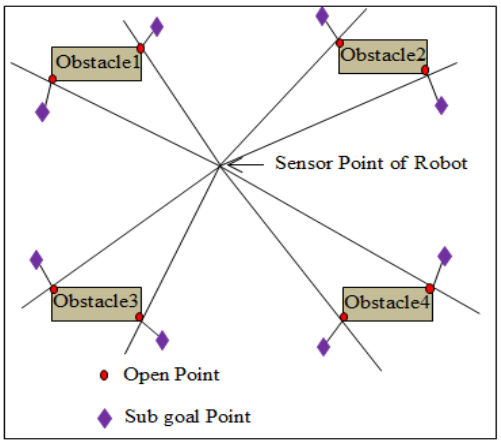

Figure 7. Obstacles' Open point and Sub goal Point detected by robot

Here (x_i,y_i) is the coordinate of sensor and $(x_{i+1,j},y_{i+1,j})$ is the coordinate of jth detected, one of next open points at (i+1)th iteration and according to worlds coordinate Calculation of open point and its sub goal point system. The open point comes in two ways. 1. If previous value of α, the sensor angle α -1 detects an obstacle and 2. Next value of α, the sensor angle α+1 detects obstacle or α-1 is not an obstacle. So the cases are obstacle to open point and open point to obstacle. Each of the cases also have two sub cases and those are: a) $\psi > 90^0$ and b) $\psi <= 90^0$

$$\text{where } \psi = \beta\%180° \quad (3)$$

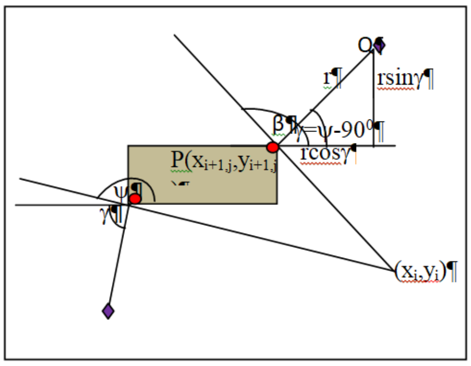

Figure 8. Open point to Sub goal point calculation

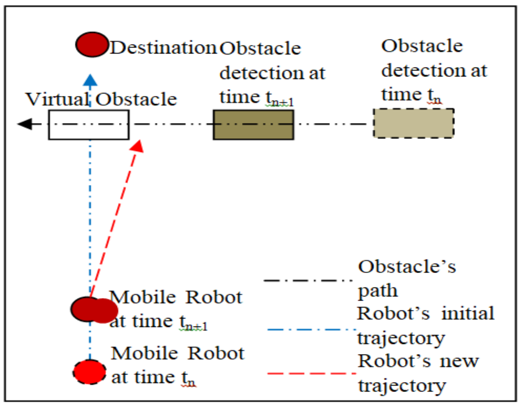
Destination
Obstacle detection at time t_{n+1}
Obstacle detection at time t_n
Virtual Obstacle
Mobile Robot at time t_{n+1}
Mobile Robot at time t_n
Obstacle's path
Robot's initial trajectory
Robot's new trajectory

Figure 9. Avoiding a moving obstacle perpendicular direction to the mobile robot

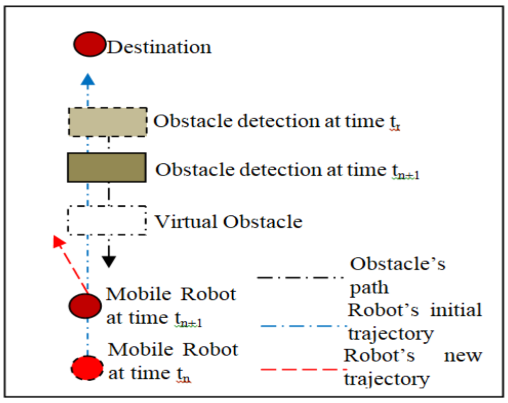
Destination
Obstacle detection at time t_t
Obstacle detection at time t_{n+1}
Virtual Obstacle
Mobile Robot at time t_{n+1}
Mobile Robot at time t_n
Obstacle's path
Robot's initial trajectory
Robot's new trajectory

Figure 10. Avoiding a moving obstacle in opposite direction to the mobile robot

So we can define γ as follows,

$$\gamma = \begin{cases} \psi - 90°, Sx = 1, Sy = 1 & \text{if } \psi > 90° \;\&\; \alpha - 1 \text{ not obstacle} \\ \psi - 90°, Sx = -1, Sy = -1 & \text{if } \psi > 90° \;\&\; \alpha - 1 \text{ is obstacle} \\ 90° - \psi, Sx = 1, Sy = -1 & \text{if } \psi \le 90° \;\&\; \alpha - 1 \text{ not obstacle} \\ 90° - \psi, Sx = -1, Sy = 1 & \text{if } \psi \le 90° \& \alpha - 1 \text{ is obstacle} \end{cases} \tag{4}$$

According to Figure 8 if ($x_{i+1,j}$,$y_{i+1,j}$) is open point P then, associate sub goal point Q is:

$$\left.\begin{aligned} Qx &= Px + Sx.rcos\gamma \\ Qy &= Py + Sy.rcos\gamma \end{aligned}\right\} \tag{5}$$

After generating the next coordinate point to move the virtual circle's centre, the wheel axel centre of the robot will turn and start moving towards that point. According to Figure 4 A is the wheel axel centre. Q should be the next virtual circle centre. If (Ax,Ay) be the next coordinate of wheel axel centre A and AC is the distance between wheel axel centre and virtual circle centre then,

$$\left.\begin{aligned} Ax &= Qx + Sx.ACcos\gamma \\ Ay &= Qy + Sy.ACsin\gamma \end{aligned}\right\} \tag{6}$$

Now the robot will calculate the angle to rotate to the next coordinate from current coordinate of wheel axel centre and then move directly. Using the stated method the robot will safely avoid static obstacles. But in case of dynamic obstacles is has determine the speed of the running obstacle by capturing the location of two different time t_{n+1} and t_n where $\Delta t = t_{n+1} - t_n$. Therefore,

$$v_{obs} = \frac{(O_{t+1} - O_t)}{\Delta t} \tag{7}$$

where O_{t+1},O_t and v_{obs} are Obstacle locations at moment t_{n+1} and t_n and speed of the running obstacle respectively. Since the direction of obstacle and robot fixed (as per A7) so the calculating the conflicting point (P_C) is possible. The algorithm will consider the area of probable collision as a virtual obstacle. If $P_R(rx_{n+1},ry_{n+1})$ be present location of the robot at t_{n+1}. Then,

$$D_C = \sqrt{(Cx - rx_{n+1})^2 + (Cy - ry_{n+1})^2} \tag{8}$$

Where (Cx,Cy) is the coordinate value of P_C. Using value of D_C then robot then determines,

$$R_{TC} = {}^{D_C}/_{V} \tag{9}$$

$$O_{TC} = {}^{D_C}/_{V_{obs}} \tag{10}$$

The velocity of robot is v. R_{TC} and O_{TC} are the time the robot and running obstacle will take to arrive at colliding point respectively. R_{TC} and O_{TC} equal means they will take same time to reach at same point P_C in near future. So the point will be denoted as virtual obstacle and changed trajectory will be formed with safety precaution as shown in Figure 11 by the robot from current position to the target as per algorithm, through the deviation point as shown in Figure 5. Snapshots of trajectory using OperativeCriticalPointBug algorithm of a counterpart of activity bot on 2D plane as shown in Figure 12.

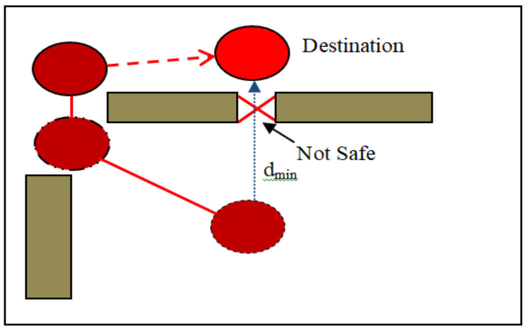

Figure 11. Obstacle avoidance with safety measurement

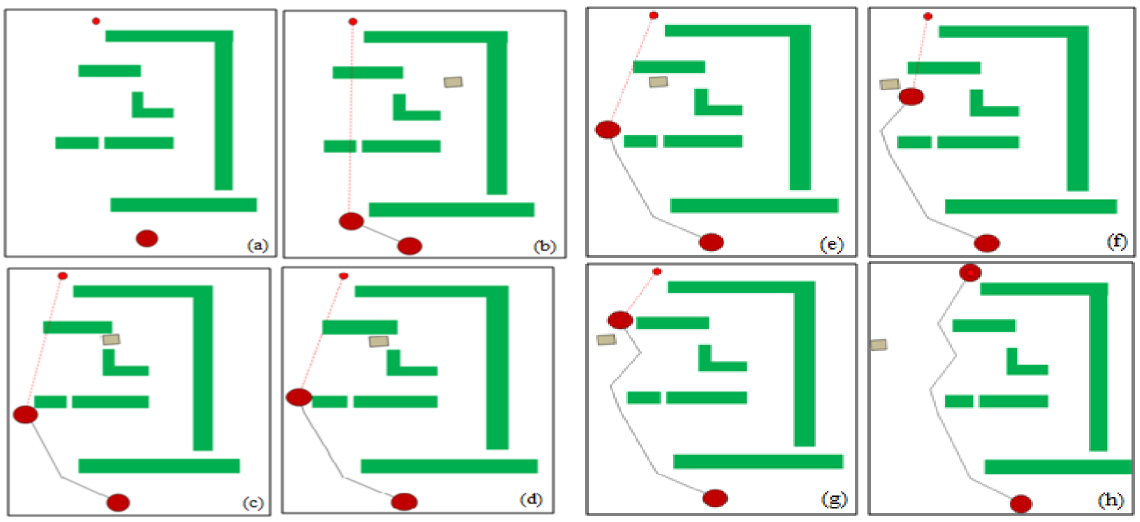

Figure 12. Snapshots of trajectory using OperativeCriticalPointBug algorithm of a counterpart of activity bot on 2D plane

3.2. Total Time And Path Length Calculation

If v_i and ω_i are the speed and angular velocity at ith iteration then, time taken in moving and rotating [20]:

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

So the cost function in terms of time is:

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

Where DOT is the time taken to detect moving obstacle and its direction of movement and speed

3.3. Simulation Results

The Figure 12 shows the simulation model of the OperativeCriticalPointbug algorithm and how it avoids static as moving obstacle. There are eight different snap shots of different moments. The rectangular shaped gray coloured object is moving obstacle and green rectangular objects are static obstacles. The algorithm is modelled using Python 3.5 on windows 7 platform. Intel core i3-2350@2.30 Ghz Laptop with 2 GB RAM is used. The orange circle at the top and red circle at the foot are the target and source points respectively. The d_{min} is shown by red line. Black line is the path the robot traversed through.

4. CONCLUSIONS AND FUTURE WORK

This paper is a continuation of its two previous works. The main objective is building a simple sensor based algorithm on local path planning. The algorithm tries to consider the constraints of robots' dimension. Using this bug algorithm very little prior information is required for the robot to complete task. The safety point calculation makes the algorithm more efficient. It helps the robot to pass through small gap as well also to avoid the gap which is too small to pass through. If the environment is not much complex or the number of obstacle is very few, then this algorithm and other existing algorithm may take almost same time. Few favorable points of this algorithm are: (a) other than those which may cause collision, it never considers the entire obstacles for avoidance. (b) Uses efficient technique to calculate coordinates of points. (c) The algorithm requires only source and destination coordinates to complete task. (d) Number of iteration is less to reach the goal. In the future, the algorithm will be used for real test. Further it will be studied on path planning on snake like robot and path planning of multiple robots in one environment.

ACKNOWLEDGEMENT

Thank you to Production Engineering Department, Jadavpur University and Asansol Engineering College for giving us the opportunity to use their resources.

REFERENCES

[1] Cruz, V. Muiioz, A. Garcia.Cerezo, A. Ollero, "Moving Obstacle A Voidance Algorithm For Mobile Robots Under Speed Restrictions", *IFAC Intelligent Components for Vehicles*, 1998, pp. 205 – 210

[2] Haojie Zhang, Guangming Xiong, Bo Su, Jianwei Gong,Yan Jiang, Huiyan Chen and Wei Lan, "Anytime Path Planning in Graduated State Space", *IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 358 – 362

[3] Junghee Park, Jeong S. Choi, Jimin Kin, and Beom H. Lee, "*Moving Obstacle Avoidance for a Mobile Robot*", IEEE International Conference on Control and Automation, 2009, 367 – 372

[4] Zhihao Xu, Robin Hess, Klaus Schilling, "*Constraints of Potential Field for Obstacle Avoidance on Car-like Mobile Robots*", IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control – CESCIT, 2012, 169 – 175

[5] Nick Malone, Hao-Tien Chiang, Kendra Lesser, Meeko Oishi, Lydia Tapia, "Hybrid Dynamic Moving Obstacle Avoidance Using a Stochastic Reachable Set-Based Potential Field", *IEEE Transactions on Robotics*, vol. 33(5), 2017, 1124-1138

[6] Md. Arafat Hossain, Israt Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique", *Robotics and Autonomous Systems* 64, 2015, 137–141

[7] Dongsheng Zhou, Lan Wang, Qiang Zhang, "Obstacle avoidance planning of space manipulator end-effector based on improved ant colony algorithm", SpringerPlus (2016) 5, 509 – 521

[8] Hanna Kurniawati, Thierry Fraichard, "*From Path to Trajectory Deformation*", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, 159 – 164

[9] Robert L.Williams II, JianhuaWu, "Obstacle Avoidance for an Omnidirectional Mobile Robot", *Journal of Robotics*, Vol. 2010, Article ID 901365

[10] Muhannad Mujahed, Dirk Fischer, Bärbel Mertsching, "Admissible gap navigation: A new collision avoidance approach", *Robotics and Autonomous Systems* 103, 2018, 93 – 110

[11] Georges S. Aoude, Brandon D. Luders, Joshua M. Joseph, Nicholas Roy, Jonathan P. How, "Probabilistically Safe Motion Planning to Avoid Dynamic Obstacles with Uncertain Motion Patterns", Autonomous Robots, 2013, vol. 35(1), 51-76

[12] Jin Cheng, Yong Zhang, Zhonghua Wang, "Motion Planning Algorithm for Tractor-trailer Mobile Robot in Unknown Environment", *IEEE ICNC 2012*, 1050 – 1055

[13] Filippo Sanfilippo, Jon Azpiazu, Giancarlo Marafioti, Aksel A. Transeth, Øyvind Stavdahl and P˚al Liljeb¨ack, "*A Review on Perception-driven Obstacle-aided Locomotion for Snake Robots*", IEEE International Conference on Control, Automation, Robotics & Vision, 2016

[14] Ajoy Kumar Dutta, Subir Kumar Debnath. Subir Kumar Das "Path Planning of Snake-Like Robot in Presence of Static Obstacles Using Bug Algorithm", *Advances in Computer, Communication and Control*, Springer, 2018, 449 - 458

[15] Nor Badaria Abdul Latip, Rosli Omar, Sanjay Kumar Debnath, “Optimal Path Planning using Equilateral Spaces Oriented Visibility Graph Method”, *International Journal of Electrical and Computer Engineering(IJECE)*, 2017, vol. 7, no. 6, 3046-3051

[16] Jinho Kim, Jangmyung Lee, “An Active Virtual Impedance Control Algorithm For Collision Free Navigation of a Mobile a Robot”, *IAES International Journal of Robotics and Automation (IJRA)*, 2017, Vol. 6, No. 2, 101-113

[17] Ishay Kamon, Ehud Rivlin, “Sensory-Based Motion Planning with Global Proofs”, *IEEE Transactions On Robotics And Automation*, vol. 13(6), 1997, 814 - 822

[18] Nishant Sharma, Shivam Thukral, Sandip Aine, and P.B. Sujit, “*A virtual bug planning technique for 2D robot path planning*”, Annual American Control Conference, 2018, 5062 – 5069

[19] Bhanu Chander V, Asokan T, Ravindran B, “*A New Multi-Bug Path Planning Algorithm for Robot Navigation in Known Environments*”, 2016 IEEE Region 10 Conference (TENCON), 3363 – 3367

[20] Emam Fathy Mohamed, Khaled El-Metwally, A.R. Hanafy, “An Improved Tangent Bug Method Integrated with Artificial Potential Field for Multi-robot Path Planning”, International Symposium on Innovations in Intelligent Systems and Applications, *IEEE*, 2007, 555 - 559

[21] Tiago Pereira do Nascimento, Pedro Costa, Paulo G, Costa, António Paulo Moreira, André Gustavo Scolari Conceição, “A set of novel modifications to improve algorithms from the A star family applied in mobile robotics”, *Journal of Brazilian Computer Society* vol. 19, 2013, 167–179

[22] Ajoy Kumar Dutta, Subir Kumar Debnath, Subir Kumar Das, “Local Path Planning of Mobile Robot Using Critical-PointBug Algorithm Avoiding Static Obstacles”, *IAES International Journal of Robotics and Automation*, vol. 5(3), 182-187

[23] Kamilah Taylor, Steven M. LaValle, “*I-Bug: An intensity-based bug algorithm*”, IEEE International Conference on Robotics and Automation, 2009, 3981-3986

[24] T. Nayl, G. Nikolakopoulos, T. Gustafsson, "Real-time bug-like dynamic path planning for an articulated vehicle" in Informatics in Control Automation and Robotics, Springer, 2015, 201-215

[25] Y. Gabriely, E. Rimon, "Cbug: A quadratically competitive mobile robot navigation algorithm", *Robotics IEEE Transactions on*, vol. 24, no. 6, 2008, 1451-1457

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.