New Real Time (M-Bug) Algorithm for Path Planning and Obstacle Avoidance In 2D Unknown Environment

Ahmed Mohamed Mohsen
Electronics & Communications Engineering
Arab Academy for Science, Technology & Maritime Transport
Abukir, Alexandria, Egypt
a.mohsen101@gmail.com

Maha Ahmed Sharkas
Electronics & Communications Engineering
Arab Academy for Science, Technology & Maritime Transport
Abukir, Alexandria, Egypt
msharkas@aast.edu

Mohamed Saad Zaghlol
Electronics & Communications Engineering
Arab Academy for Science, Technology & Maritime Transport
Abukir, Alexandria, Egypt
dr_mszaghloul@yahoo.com

Abstract- **This paper proposes a new sensor-based path planning and collision avoidance algorithm for non-holonomic robot travelling in unknown 2D environment named as (M-Bug) algorithm in order to reduce the overall of traveled path. It uses the difference between slope lines to find and choose the smallest angle located between slope of straight line from mobile robot to goal point and slope from mobile robot and two endpoints of edges for obstacle, by choosing the smallest one which refers to nearest leave point to reach goal point and repeat this action with other obstacles until mobile robot reach to goal point. this algorithm based on using on-board sensors to generate information about the environment and obstacle located in front of mobile robot. This algorithm creates better performance when compared to the other traditional algorithms. It provides real time obstacle avoidance, compatible with LIDAR sensor and fast enough to be implemented on embedded microcontrollers. The used algorithm is based on a modification for existing path planning and obstacle voidance algorithms (Bug1, Bug2, Tangent-Bug, K-Bug, Dist-Bug …...etc.), and The Vector Field Histogram (VFH) Algorithm in unknown environment. The main advantage of this algorithm is that it is not limited to the origin-destiny line and changes the main line every leave point. MATLAB 2018a is used in simulation to validate the effectiveness of the proposed algorithm, and it is implemented on multiple scenarios to show the ability of the mobile robot to follow a path, detect obstacles, and navigate around them to avoid collision.**

Keywords- Path planning, Collision avoidance, Bug algorithm, Autonomous mobile robots, LIDAR, non-holonomic.

I. INTRODUCTION

Over the recent decades, the field of robotics and especially mobile robotics has great progress in technological advancement and degree of autonomy. Aautonomous mobile robots are used in a wide range of fields; from factories production lines, Research and rescue missions fields, surveillance and exploration tasks such as forest fire monitoring, ... etc.

Autonomous mobile robots can be simply defined as intelligent machines that are capable of performing certain tasks in their environment without explicit human interaction, decrease the operating costs and complexity[1]. For the purpose of this research, only the algorithms that allow for conditions of navigation in an unknown environment using range on-board sensors and are suitable for real-time, embedded applications with an acceptable performance are considered.

One of the fundamental fields of research, is path planning [2], Path planning in robotics is defined as navigation that will be collision free and most optimum for the autonomous vehicle to maneuver from a start point to goal point. The purpose of a path-planning method for a mobile robot is to find a continuous collision-free path from start point to goal point. The path planning problem is divided into two main categories: known-environment and unknown-environment. In the known-environment, the information about the location and size of obstacles is sufficiently provided and the opposite for the unknown-environment [3].

There were many algorithms which are used in the unknown environment. one of the most simple and effective algorithms is Bug family algorithms which it will be discussed. Bug algorithms, a kind of autonomous navigation algorithm for unknown environment and is proposed by Lumelsky and Stepanov[4]. The method tries to find and specify a route with the least path points. the advantages of this principle are simple and easy to realize. The assumption considered that on-board sensors can get complete obstacle information. the robot is a particle with perfect self-positioning capability, and according

to this concept, the robot moves directly to the goal point firstly, and updates data form on-board sensors online for current path to avoid the obstacles, until the goal point is reached.

The aim of this paper is proposing a new algorithm for autonomous navigation for mobile robot from the start point to the goal point. This paper is organized as follows. Section II reviews for related work. Section III discusses a proposed algorithm for path planning and collision avoidance. Section IV provides the simulation results for the proposed algorithm and finally the paper is concluded in section V.

II. RELATED WORK

Path planning and Collision avoidance in an unknown environment has been a subject of interest for decades and several algorithms have been proposed to tackle such problem[5]. This section surveysthe most commonly used collision avoidance algorithms.

- Bug Algorithms

The Bug algorithms family is one of the simplest obstacle avoidance algorithms ever described. Moreover, they are fundamental and complete algorithms with provable guarantees Like all other obstacle avoidance algorithms [5]. The goal of the Bug algorithm is to generate a collision free path from a start point to goal point. The Bug algorithms do not assume knowledge of the global environment or shapes or sizes of obstacles but depend on local knowledge perceived from sensors.

A. Bug1

The Bug1 algorithm is one of the earliest obstacle avoidance algorithms [4]. In this algorithm, the robot circles the obstacle one complete time, calculating the distance from the goal at each position to find the shortest distance. Once a complete rotation along the borders of the obstacle is achieved, the robot would move to that point and leave towards the goal in a straight-line manner. This is calculated based upon the leaving point and goal point using the slope 'm' and y-intercept 'c' as follows.

$$y = mx + c \qquad (1)$$

Bug1 algorithm shown in Fig.1 suffers some drawbacks, when the robot is following the edge of one obstacle it may collide with a neighboring one if both obstacles are very close or the gap between them is less than the width of the robot and it takes more time to reach the goal point.

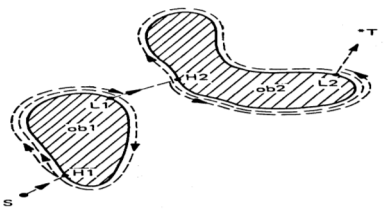


Fig.1.Bug1 algorithm[4]

B. Bug2

The Bug2 algorithm shown in Fig.2 is one of the improved Bug algorithms. Using this algorithm, the robot would move along the initial path towards a goal until an obstacle is encountered. Once an obstacle is encountered, the robot would keep a fixed distance and start following the borders of the obstacle, calculating the slope of the straight line joining its current position to the final goal. When the slope becomes equal to the slope of the initial path, the robot leaves the border and continues towards the goal[4].

Even though this algorithm improves the efficiency of Bug1 by not repeating the path around the obstacle, it is still limited to the origin-destiny line and does not address the problem of generating the shortest path.

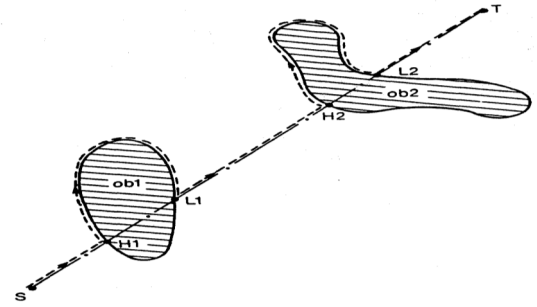


Fig.2.Bug2 algorithm[4]

C. Dist-Bug

The Dist-Bug shown in Fig.3 is another improvement to the Bug algorithm family. It was invented by Kamon and Rivlin[6], and originates from Sankaranarayanan's algorithms Alg1 and Alg2 [7]. Dist-Bug allows the robot to reach the destination in a comparatively shorter time. The algorithm considers the maximum detection range of the robot to be **R**. Similar to the other Bug algorithms, first the mobile robot starts by moving towards the goal point **G** until it faces an obstacle. Once an obstacle is encountered, the mobile robot starts to follow the borders of the obstacle in a predefined direction (default) either counter clockwise or clockwise. when the mobile robot follows the borders of the obstacle at the preset distance, it calculates its distance from the goal $\boldsymbol{dmin(G)}$ in a continuous way and keeps record of the minimum distance since last hit point. Furthermore, the mobile robot also uses the on-board sensors to detect the free space **F**, which is a distance between obstacle and the mobile robot current location **X** in the direction of the predefined goal point. For the case when no obstacle is detected, **F** is set to **R** and the robot leaves the border of the obstacle once the following condition is satisfied.

$$d(X, G) - F \leq dmin(G) - step \qquad (2)$$

Where *d(X, G)* is the distance between current location and goal, and where step is a predefined constant which is one of the algorithm's limitations and drawbacks.

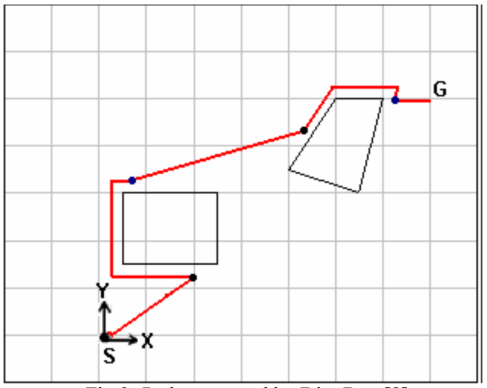


Fig.3. Path generated by Dist-Bug [8]

D. The Vector Field Histogram (VFH) Algorithm

The Vector Field Histogram (VFH) Algorithm is a sensor based algorithm in an unknown environment described for the first time by J. Borenstein and Y. Koren [9], and was later improved by I. Ulrich and J. Borenstein [10] under name VFH+ then improved to VFH* by I. Ulrich and J. Borenstein [11]. The VFH method is manly based on the two principles. The first one is a two-dimensional Cartesian histogram grid which is continuously updated in real-time with range data sampled by the on-board range sensors. In the second one, the histogram grid is reduced to a one-dimensional polar histogram shown in Fig.4, that is constructed around the momentary location of the robot. The polar histogram is the most significant in Fig.4, Angles reding by on-board sensor(k)and probability of finding obstacle (H(k)) in that direction presented by x-axis and y-axis Sequentially.

By creating a local occupancy grid map of the environment around the mobile robot all probabilities for existence obstacles computed. Only paths which is greater than the width of mobile robot is allowed by using the polar histogram.

The mobile robot selects a certain path based on cost function which defined for each path. The mobile robot takes the path with the minimum cost. This action depends on the alignment of the mobile robot's path with the goal point, and on the difference between new path direction and the orientation for current wheel.

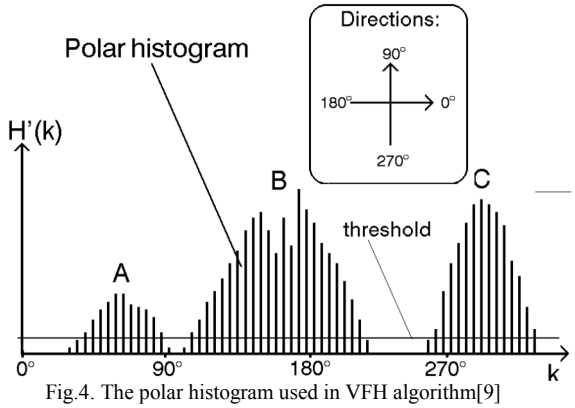


Fig.4. The polar histogram used in VFH algorithm[9]

VFH algorithm offers better efficiency by using a threshold point to overcome the sensor noise, and takes into consideration the kinematics of the mobile robot.

III. THE PROPOSED ALGORITHM

This section proposes a novel modified real time path planning and obstacle avoidance algorithm named as (M-Bug) algorithm. This method based in using on-board sensors to generate information about the environment for performing fast and a dynamic smooth path planning for non-holonomic mobile robot using on-board LIDAR sensor to map the surrounding environment to detect the obstacles to avoid collision [12], and also choosing the shortest path to reach the goal point. This algorithm creates better paths when compared to the other traditional algorithms.

For practical autonomous mobile robot, path planning and obstacle avoidance in real time is important issue. Therefore, many algorithms have been proposed to achieve balance between low cost and effectiveness[13].

This study presents a new algorithm for path planning and obstacle avoidance depend on low cost LIDAR sensor, and involving a reasonable level of calculations, so that it can be easily used in real time control applications with microcontrollers [14]. This algorithm is simulated in MATLAB2018a and the performance is introduced in the coming sections.

The suggested algorithm is based on the fact that the shortest path between two points is a straight line between them. Therefore, the efforts to find the path for a mobile robot by dividing overall path to small straight line paths with beginning point in mobile robot and end with leave point which will be a new start point for next step, and repeat this step until reach to goal point.

The main concept of this algorithm is

1) straight line is the shortest path between two points.

2) hit point and leave point in this work is determined by LIDAR, before the robot hit the obstacle with acceptable distance (can be adjustable depending on mobile robot size and the environment scale.

Algorithm Description

To implement this algorithm, first the 2-D Cartesian co-ordinate of goal point and start point putted in algorithm (or from GPS). There are three main steps for the algorithm in order to implement the autonomous navigation from the start point to the goal point as shown in the flow chart in Fig.5. The three points are summarized as follows

1) motion towards target.
2) scan obstacle surface or boundary follower until find leaving point depending on the relation between the main slope and the edge slope.

3) make a new start point and go forward to goal point.

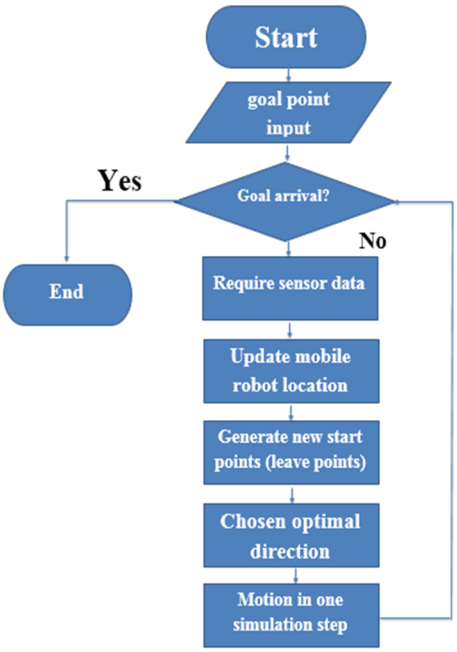


Fig.5. Flow chart for proposed (M-Bug) algorithm

The mobile robot starts to moving from start point to goal point in straight line until the LIDAR sensor rays hit the first obstacle in this path. the algorithm scans the first obstacle defining the two-boundary points Fig.6, and calculated the slope angles $\boldsymbol{\theta}, \boldsymbol{\theta obs1}$ and $\boldsymbol{\theta obs2}$ from equations 3, 4 and 5.

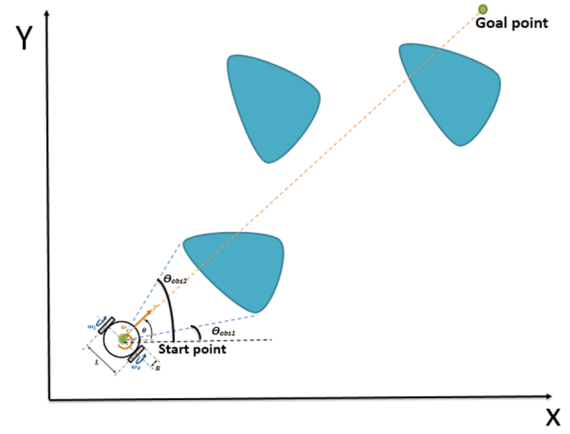


Fig.6. slope angles θ, $\theta obs1$ and $\theta obs2$

$$\theta = tan^{-1}m \quad (3)$$

$$\theta obs1 = tan^{-1}m1 \quad (4)$$

$$\theta obs2 = tan^{-1}m2 \quad (5)$$

where

$\boldsymbol{\theta}$ .... is angle of slope line between center of mobile robot (LIDAR sensor) and goal point.

$\boldsymbol{\theta obs1}$ .... Is angle of slope line between center of mobile robot (LIDAR sensor) and lower edge of obstacle.

$\boldsymbol{\theta obs2}$…. Is angle of slope line between center of mobile robot (LIDAR sensor) and upper edge of obstacle.

Then calculated two angles $\boldsymbol{\theta 1}$and $\boldsymbol{\theta 2}$ shown in Fig.7. located between mobile robot and two boundaries with respect to origin-destiny line (dashed line) from equations 6 and 7.

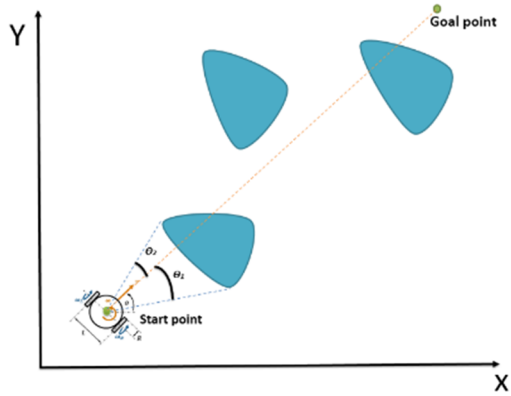


Fig.7. Angles $\theta 1$and $\theta 2$ located between mobile robot and two boundaries with respect to origin-destiny line (dashed line)

$$\theta 1 = |\theta obs1 - \theta| \quad (6)$$

$$\theta 2 = |\theta obs2 - \theta| \quad (7)$$

where

$\boldsymbol{\theta 1}$…. Is the angle between upper edge and density line.

$\boldsymbol{\theta 2}$…. Is the angle between lower edge and density line.

After the small angle chosen by the algorithm, the mobile robot moves to the direction for corresponding boundary edge which will be leave point.

Under condition of non-holonomic mobile robot which have front width of ($\boldsymbol{L}$). adding to the offset distance ($\boldsymbol{L/2}$) to avoid hit obstacle and for smooth rotation and will be the start point for the next step Fig.8.

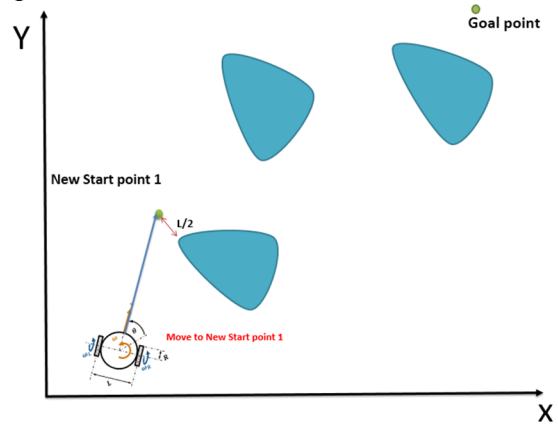


Fig.8.Move to new start point with offset $L/2$

The new point will be considered as new start point1 (x_n,Y_n) and repeats above steps until reach to goal point Fig.9.

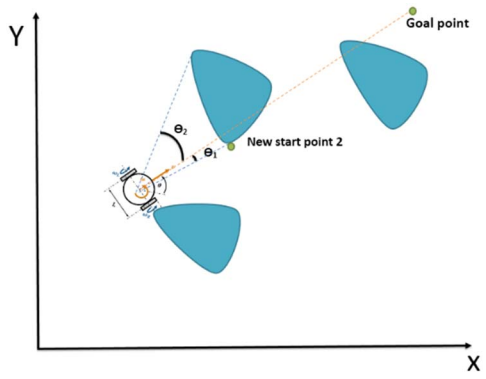


Fig.9.Repeting algorithm steps to reach to goal point

The main advantage of this algorithm is that it is not limited to the origin-destiny line and changes the main line every hit point until reach to the goal point throw trajectory created by algorithm Fig.10.

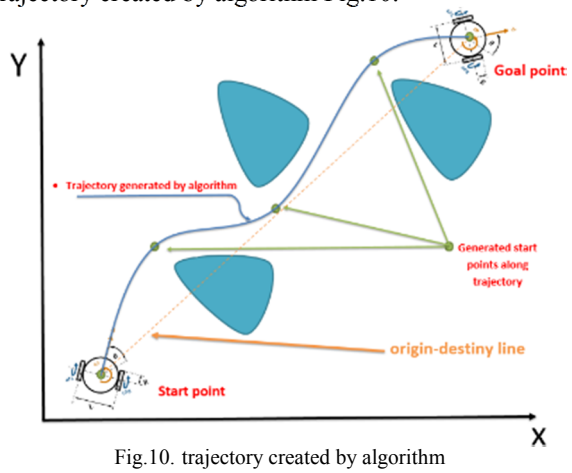


Fig.10. trajectory created by algorithm

Special cases

1) In the case of a gap between two obstacles ≤ the width of mobile robot front with (*L*) Fig.11, the algorithm will neglect two mid points and consider the two obstacle as one obstacle with two boundaries and the decision depends on the measured $\theta_1$and θ_2 which refers to a new line path slope (m).

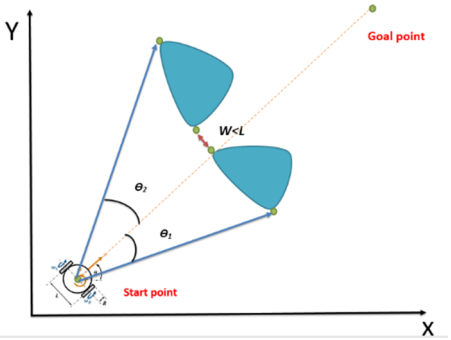


Fig.11.Gap between two obstacles ≤ the width of mobile robot front with (L)

2)in case of obstacle with U shape, local minima effect the LIDER will detect two boundary edges, mobile robot will not enter inside to reduce time, distance and energy cost Fig.12.

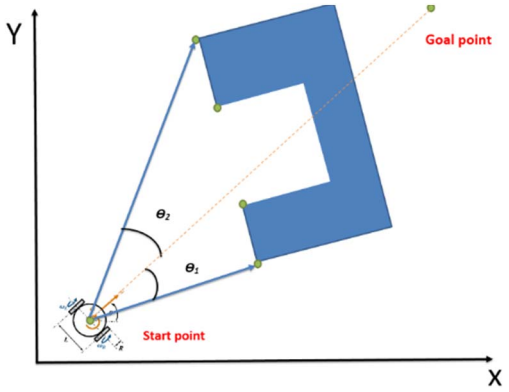


Fig.12.Obstacle with U shape (local minima effect)

3)The worst case if the LIDAR sensor with 360 degree of scan and there is no visible path with width> W. The vehicle will enter in sleep mood until the conditions changed.

IV. SIMULATION AND RESULTS ANALYSIS

In this work, two differential wheel non-holonomic mobile robot model was used [15] as shown in Fig.13 with specifications, Look ahead Distance for LIDAR = 0.5m, Linear Velocity = 0.75 m/s, angular Velocity = 1.5 rad/s, Wheel radius R = 0.1 m, Wheelbase L = 0.5m with controller created in MATLAB2018a for this experiment as shown in the block diagram in Fig.14, and the environment grade is (10X10)m .

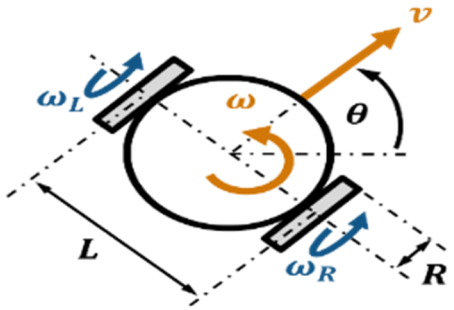


Fig.13. geometry of used two differential wheel non-holonomic robot.

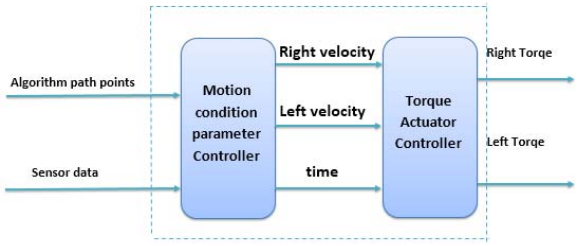


Fig.14. controller block diagram.

some simulations were shown to illustrate the proposed algorithm. All simulations were programmed using MATLAB2018a shown in Fig15, 16, 17, 18 and 19. First, the robot starts sensing the environment with lidar sensor and follows the steps of the algorithm to reach the goal point, to avoid collision based on the

proposed algorithm stated above. When the robot detects an object with lidar sensor, it navigates around based on the smallest angle between the slope of the major line and the slopes of boundary lines which correspond to short distance and therefore short time. The robot will follow the same steps each time it detects an obstacle and cerates its path again.

1)Environment #1

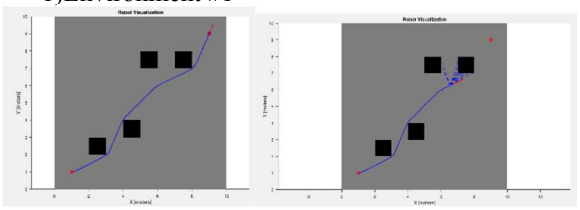


Fig.15. Simulation for environment #1

2) Environment #2

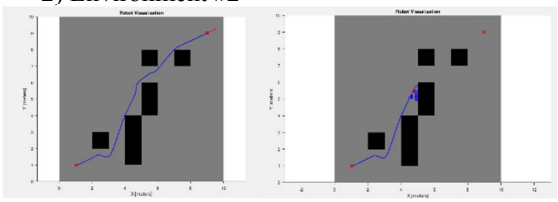


Fig.16. Simulation for environment #2

3) Environment #3

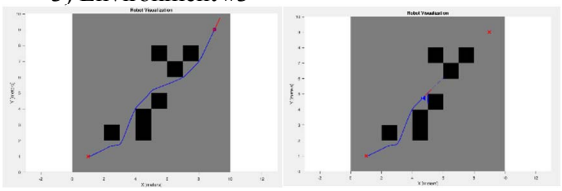


Fig.17. Simulation for environment #3

4) Environment #4(local minima)

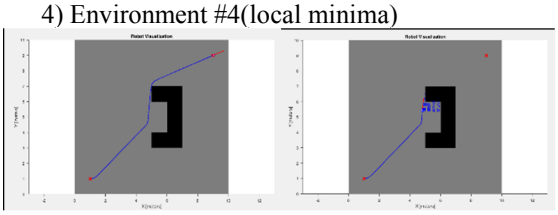


Fig.18. Simulation for environment #4

5) Environment #5(local minima and obstacles)

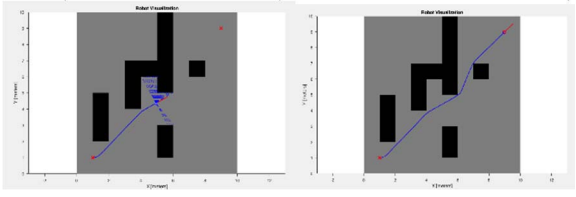
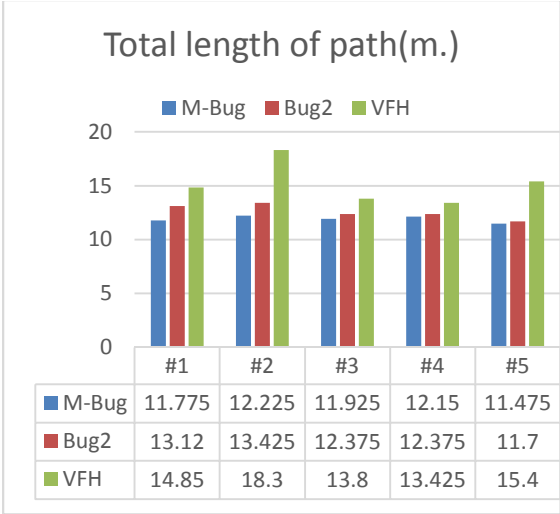


Fig.19. Simulation for environment #5

In order to verify the algorithm, Fig.20 represent the distance taken by mobile robot to reach the goal point in M-Bug, Bug2 and VFH algorithms and Fig.21 represent the time taken by mobile robot to reach the goal point in M-Bug, Bug2 and VFH algorithms .

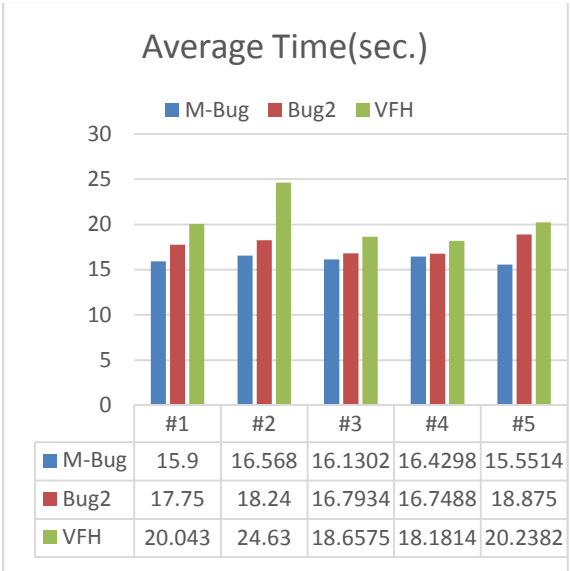
Table.1 show the Rate of distance and time decrease between M-Bug and two aerial algorithm Bug2 and vector field histogram (VFH).

The start and goal points are represented by (1,1) and (9,9), the robot can move from start point to goal point independently, safely and without collision, and the robot can cross the long and narrow channels, complex environment and avoid local minima.



	#1	#2	#3	#4	#5
M-Bug	11.775	12.225	11.925	12.15	11.475
Bug2	13.12	13.425	12.375	12.375	11.7
VFH	14.85	18.3	13.8	13.425	15.4

Fig.20. Distance taken by mobile robot to reach the goal point in M-Bug, Bug2 and VFH algorithms



	#1	#2	#3	#4	#5
M-Bug	15.9	16.568	16.1302	16.4298	15.5514
Bug2	17.75	18.24	16.7934	16.7488	18.875
VFH	20.043	24.63	18.6575	18.1814	20.2382

Fig.21. Time taken by mobile robot to reach the goal point in M-Bug, Bug2 and VFH algorithms

Table .1. Rate of distance and time decrease

		Rate of distance decrease		**Rate of time decrease**	
Enviro-nment Number	Operations Number	M-Bug with Bug2	M-Bug with VFH	M-Bug with Bug2	M-Bug with VFH
#1	10	10.3%	20.7%	10.4%	20.7%
#2	10	8.9%	33.2%	9.2%	32.7%
#3	10	3.6%	13.6%	3.9%	13.5%
#4	10	1.8%	9.5%	1.9%	9.6%
#5	10	1.9%	25.5%	17.6%	23.2%

V. CONCLUSION.

In this paper a new algorithm is introduced based on sensor-based algorithms in 2D unknown environment to resolve the autonomous navigation in the unknown 2D static environment considering the kinematical characteristics of the mobile robot. It depends on the use of LIDAR sensor for generating information about the environment and obstacle.

The proposed algorism was compared with Bug2 and VFH algorithms. The simulation results proved that the proposed (M-Bug) algorithm decrease distance and time taken by mobile robot and the algorithm outperforms other methods in the length and time.

This research could be further developed in 3D unknown environment with static obstacle and /or dynamic environment.

REFERENCES

[1] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles," pp. 1–27, 2016.

[2] Q. M. Nguyen, L. N. M. Tran, and T. C. Phung, "A Study on Building Optimal Path Planning Algorithms for Mobile Robot," *Proc. 2018 4th Int. Conf. Green Technol. Sustain. Dev. GTSD 2018*, pp. 341–346, 2018.

[3] J. Giesbrecht, B. Touchton, T. Galluzzo, D. Kent, C. D. Crane, and J. Giesbrecht, "Global Path Planning for Unmanned Ground Vehicles," *From OAIster, Provid. by OCLC Coop.*, no. December, pp. 1–41, 2004.

[4] V. L. and A. Stepanov, "Dynamic path planning for a mobile automation with limited information on the environment," *IEEE Trans. Automat. Contr.*, vol. 31, no. 11, pp. 1058–1063, 1986.

[5] J. Ng and T. Bräunl, "Performance comparison of Bug navigation algorithms," *J. Intell. Robot. Syst. Theory Appl.*, vol. 50, no. 1, pp. 73–84, 2007.

[6] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE Trans. Robot. Autom.*, vol. 13, no. 6, pp. 814–822, 1997.

[7] N. Buniyamin, W. A. J. Wan Ngah, and Z. Mohamad, "PointsBug versus TangentBug algorithm, a performance comparison in unknown static environment," *2014 IEEE Sensors Appl. Symp. SAS 2014 - Proc.*, pp. 278–282, 2014.

[8] Yufka, Alpaslan & Parlaktuna, Osman. (2009). Performance Comparison of BUG Algorithms for Mobile Robots. 10.13140/RG.2.1.2043.7920.

[9] J. Borenstein, Y. Koren, S. Member, A. Arbor, A. T. Laboratories, and A. Arbor, "THE VECTOR FIELD HISTOGRAM -," vol. 7, no. 3, pp. 278–288, 1991.

[10] I. Ulrich and J. Borenstein, "Reliable Obstacle Avoidance for Fast Mobile Robots," no. May, pp. 1572–1577, 1998.

[11] I. Ulrich and J. Borenstein, "VFH *: Local Obstacle Avoidance with Look-Ahead Verification," no. April, pp. 2505–2511, 2000.

[12] A. Pfrunder, P. V. K. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2017-Septe, pp. 2601–2608, 2017.

[13] S. Ricardo, D. Bein, and A. Panagadan, "Low-cost, real-time obstacle avoidance for mobile robots," *2017 IEEE 7th Annu. Comput. Commun. Work. Conf. CCWC 2017*, 2017.

[14] M. M. Almasri, A. M. Alajlan, and K. M. Elleithy, "Trajectory Planning and Collision Avoidance Algorithm for Mobile Robotics System," *IEEE Sens. J.*, vol. 16, no. 12, pp. 5021–5028, 2016.

[15] M. A. A. Hemmat, Z. Liu, and Y. Zhang, "Real-Time path planning and following for nonholonomic unmanned ground vehicles," *Int. Conf. Adv. Mechatron. Syst. ICAMechS*, vol. 2017-Decem, pp. 202–207, 2018.