# Implementation of Bug1 and Bug2 Basic Path-Planning Algorithms for a TurtleBot 3 Robot in ROS Noetic

**Ilya Spektor**
*HSE University, Moscow, Russian Federation*

**Aidar Zagirov**
*Intelligent Robotics Department, Kazan Federal University, 420008, Kazan, Russian Federation*

**Ramil Safin**
*Intelligent Robotics Department, Kazan Federal University, 420008, Kazan, Russian Federation*

**Evgeni Magid**
*Intelligent Robotics Department, Kazan Federal University, 420008, Kazan, Russian Federation*
*HSE University, Moscow, Russian Federation*
*E-mail: ivspektor@edu.hse.ru, ai.zag@it.kfu.ru, safin.ramil@it.kfu.ru, magid@it.kfu.ru*

**Abstract**

Mobile robots typically operate in a dynamically changing unknown environments. The Bug family algorithms were developed to address path planning challenges for ground vehicles within 2D configuration spaces of unknown environments. These algorithms utilize local sensory data about obstacles encountered during navigation. This paper introduces the implementation of Bug1 and Bug2 local path planning algorithms in ROS Noetic. Traditional 2D Bug algorithms are extended into 3D Gazebo simulation environment. Performance evaluation of Bug1 and Bug2 were conducted using the TurtleBot3 Burger model in both simple convex and maze environments.

*Keywords:* BUG Algorithms, Path Planning, Navigation, Mobile Robots, ROS, Gazebo

## 1. Introduction

There are two main path planning models distinguished by the type of information they utilize. The first model, known as path planning with complete information or global navigation, assumes that a robot possesses comprehensive information about its surroundings. In contrast, the second model, known as path planning with incomplete information, introduces uncertainty about an environment [1]. It is essential to integrate both global and local path planning strategies to ensure secure robotic navigation. In this context, a primary objective of local planners is to adhere to an initial plan provided by global planners [2]. Thus, an implementation of a safe and reliable local navigation algorithm, tested in conditions close to real-world scenarios, holds a significant importance.

The Boundary-following and Ultimate Goal (BUG) family of algorithms is among the most renowned in the field of local path-planning. The BUG algorithms follow a simple cycle consisting of two states. In the first state, a robot pursues a directly or intricately calculated path to a target point until encountering an obstacle. The second step involves circumnavigating the obstacle until an exit condition is met.

This paper outlines challenges and potential solutions for implementing local path-planning algorithms within 3D environments, utilizing Bug1 and Bug2 as reference models. It marks an initial step toward an implementation of more complex BUG algorithms [3], [4]. The algorithms were implemented in ROS and evaluated in Gazebo simulator using TurtleBot3 Burger [5]. Custom-designed environments were used for testing while as a part of our ongoing work the algorithms are being validated employing a broad set of environments generated by an automatic tool for Gazebo simulator [6] and a number of standard warehouses' models [7].

## 2. Algorithms Overview

The Bug1 and Bug2 algorithms are a foundation for the entire BUG family, encompassing algorithms such as Class 1, 2 and 3 [8], Alg1 and Alg2 [9], [10], VisBug21 and VisBug22 [11], Wedgebug [12], Rev1 and Rev2 [13],

Distbug [14], TangentBug [15], CautiousBug [16] and others. Both algorithms operate by transitioning between two states: move-to-target and boundary-following (Fig. 1). These algorithms require a robot to be equipped with a touch sensor, rendering them useful for compact machinery. Next, we offer a brief overview of Bug1 and Bug2 algorithms. For a more in-depth analysis refer to the original work of Lumelsky and Stepanov [1].

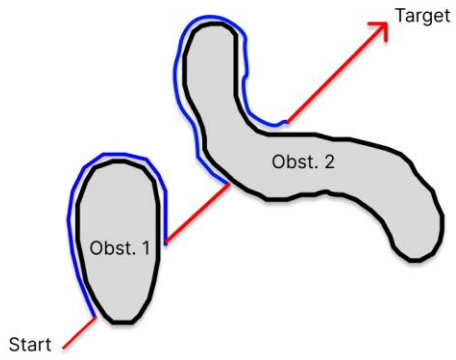


Fig. 1. Trajectory of Bug2 algorithm. Red lines depict the move-to-target state, while blue lines illustrate the boundary-following state.

We use the following notations for both algorithms:

- *Target* – a destination point that the robot should eventually reach.
- $L_i$ (leave point) – a point defined when departing from an *i*-th obstacle to resume moving toward *Target*.
- $H_i$ (hit point) – a point defined when an *i*-th obstacle is encountered.

### 2.1. Bug1 Algorithm

For the Bug1 algorithm we introduce variable $Q_m$ that stores an encountered point with a minimal distance between an obstacle boundary and *Target*. Additionally, we will introduce three registers: $R_1$ to store coordinates of the point $Q_m$, $R_2$ for a length of the obstacle boundary from $H_i$ to $Q_m$, and $R_3$ for a length of the obstacle boundary from $Q_m$ to $H_i$. The following steps are executed at every point of a continuous path:

1) From point $L_{i-1}$ move toward *Target* along a straight line until one of the following events occurs:
   (a) *Target* is reached: stop the algorithm.
   (b) An obstacle is encountered: define hit point $H_i$ and proceed to step 2.
2) Follow the obstacle boundary using a local direction (local information about the obstacle's boundary):
   (a) If *Target* is reached: stop the algorithm.
   (b) Otherwise, traverse the boundary and return to $H_i$. Define new leave point $L_i = Q_m$ and proceed to step 3.
3) Apply a test for target reachability. If *Target* is not reachable, stop the algorithm. Otherwise, using content of registers $R_2$ and $R_3$, determine a shorter way along the boundary to $L_i$, use it to get to $L_i$. Increment counter *i* and go back to step 1.

### 2.2. Bug2 Algorithm

Steps of Bug2 are executed at any point of a continuous path as follows:

1) From point $L_{j-1}$, move along a straight line (*Start*, *Target*) until one of the following events occurs:
   (a) If *Target* is reached: stop the algorithm.
   (b) An obstacle is encountered: define hit point $H_j$ and proceed to step 2.
2) Follow an obstacle boundary:
   (a) If *Target* is reached: stop the algorithm.
   (b) If line (*Start*, *Target*) is met at point *Q* such that distance $d(Q) < d(H_j)$, and line (*Q*, *Target*) does not cross the current obstacle at point *Q*. Define leave point $L_j = Q$. Increment counter *j* and go back to step 1.
   (c) If the robot returns to $H_j$ and, hence, completes a closed curve (the obstacle boundary) without defining next hit point $H_{j+1}$, it means that *Target* cannot be reached, stop the algorithm.

## 3. Implementation

Several challenges were encountered during Bug1 and Bug2 algorithms' implementation in ROS/Gazebo. The first problem was a simulation of a touch sensor for the TurtleBot3 Burger robot. It is equipped with a 180 degrees field of view range sensor for safety and integrity reasons [5]. To simulate a touch sensor, we restricted the sensor's vision range to $\rho_v = 1.5R_b$, where $R_b$ is a radius of a robot's body circumcircle. Touch sensors usually provide boolean values: *true* means that a robot is touching an obstacle, *false* otherwise. In our implementation we divided a field of view into five quadrants (of 36 degrees each) and register collision events within each quadrant independently. Another problem is that Bug1 and Bug2 do not limit robot's forward and angular velocities [1]. The most straightforward solution is to set a constant velocity. Potential further enhancements may incorporate a PID controller into the algorithm's pipeline.

Our implementation and validation were conducted using Ubuntu OS, employing robust capabilities of Robot Operating System (ROS) Noetic Ninjemys and the Gazebo simulator [17], [18]. The system configuration for the rigorous testing and validation is outlined in Table 1. For testing purposes, we selected the TurtleBot3 Burger (see Fig. 2) as the robot model [19]. The algorithms were initially implemented in Python, leveraging its popularity and the ease of prototyping and development. However, recognizing the importance of computational efficiency, we plan to reimplement the algorithms in C++.

Table 1. System configuration

| *Component* | *Specification* |
|---|---|
| OS | Ubuntu 20.04 LTS |
| ROS | Noetic Ninjemys |
| Gazebo | 11.11.0 |
| Processor | Intel Core i5-11300H @ 3.10GHz |
| RAM | 24 GB |

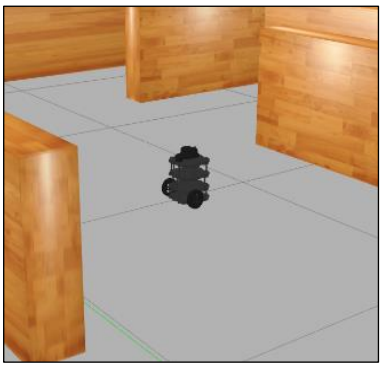

Fig. 2. TurtleBot3 Burger in the Gazebo environment.

There are three ROS nodes created to simulate a cycle of switching the states:

1) **Main**: operates continuously, responsible for calculating an optimal time to switch between the states based on predefined conditions.
2) **Go-to-Target**: orients and moves the robot toward the target.
3) **Follow-Wall**: facilitates the circumnavigation of obstacles. In the case of Bug1, there are two instances of this node – one for clockwise rotations and another for counterclockwise rotations.

## 4. Validation

To ensure the robustness and effectiveness of the Bug1 and Bug2 algorithm implementations, a comprehensive validation process was undertaken in Gazebo simulator using ROS navigation stack [19], [20]. The validation encompassed a diverse set of tests conducted in 3D simulation world featuring individual complex obstacles and a maze [6]. To ascertain the correctness of the target point reachability determination, a testing scenario was implemented for both Bug1 and Bug2 algorithms. The validation process involved 30 tests for each algorithm on maps with convex obstacles. Half of these tests featured reachable targets, while the remaining half featured unreachable targets.

Given the Bug1's requirement to fully circumnavigate obstacles, testing in maze environments was time-consuming. Thus, only 10 tests for each algorithm were conducted on the maze map, with an equal distribution of reachable and unreachable target scenarios. All target points were accurately identified for reachability in every test scenario. When a point was deemed reachable, the robot successfully navigated its path to the target, affirming the efficacy of the Bug1 and Bug2 algorithms.

The outcomes of the Bug1 and Bug2 algorithms were visually demonstrated through simulations in Gazebo. Fig. 3 illustrates a resulting path in an environment with convex obstacles, achieved by the Bug1 algorithm. Fig. 4 presents a resulting path in the same environment by Bug2. Notably, a difference was significant, with Bug2 path displaying a more straightforward trajectory, which is observed in a substantial portion of the conducted tests.

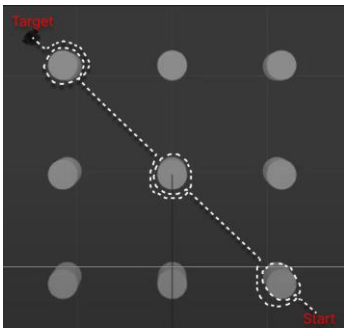


Fig. 3. Path of Bug1 in the environment with convex obstacles (white dashed line).

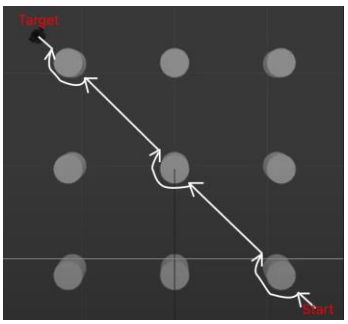


Fig. 4. Path of Bug2 in the environment with convex obstacles (white line).

## 5. Conclusion

In this work we presented the implementation of the Bug1 and Bug2 algorithms, delving into the challenges encountered while integrating local navigation algorithms into realistic ROS/Gazebo simulations. This article lays a ground for future implementations of more complex BUG family algorithms.

## Acknowledgements

This paper was supported by the Kazan Federal University Strategic Academic Leadership Program (“PRIORITY-2030”).

## References

1. V. J. Lumelsky and A. A. Stepanov, Path-planning strategies for a point mobile automaton moving amidst

unknown obstacles of arbitrary shape, In Autonomous robot vehicles, Vol. 2, 1990, pp. 403-430.
2. A. Apurin, B. Abbyasov, E. A. Martínez-García and E. Magid, Comparison of ROS Local Planners for a Holonomic Robot in Gazebo Simulator, In International Conference on Interactive Collaborative Robotics, Cham: Springer Nature Switzerland, 2023, pp. 116-126.
3. J. Ng and T. Bräunl, A Practical Comparison of Robot Path Planning Algorithms given only Local Information, Centre for Intelligent Information Processing Systems School of Electrical and Electronic Engineering The University of Western Australia, 2005.
4. J. Ng and T. Bräunl, Performance comparison of bug navigation algorithms, Journal of Intelligent and Robotic Systems, Vol. 50(1), 2007, pp. 73-84.
5. R. Safin, T. Tsoy, R. Lavrenov, I. Afanasyev and E. Magid, Modern Methods of Map Construction Using Optical Sensors Fusion, International Conference on Artificial Life and Robotics (ICAROB 2023), 2023, pp. 166-169.
6. A. Iskhakova, B. Abbyasov, T. Tsoy, T. Mironchuk, M. Svinin and E. Magid, LIRS-Mazegen: An Easy-to-Use Blender Extension for Modeling Maze-Like Environments for Gazebo Simulator, Frontiers in Robotics and Electromechanics, Vol. 329, 2023, pp. 147-161.
7. A. Khazetdinov, A. Zakiev, T. Tsoy, R. Lavrenov and K.-H. Hsia, Standard-complaint Gazebo warehouse modelling and validation, Proceedings of 13th International Conference on Developments in eSystems Engineering (DeSE), 2020, p. 218-221.
8. H. Noborio, Several path-planning algorithms of a mobile robot for an uncertain workspace and their evaluation, Proceedings of the IEEE International Workshop on Intelligent Motion Control, Istanbul, Turkey, 1990, pp.289-294.
9. A. Sankaranarayanan and M. Vidyasagar, A New Path Planning Algorithm For Moving A Point Object Amidst Unknown Obstacles In A Plane, IEEE Conference on Robotics and Automation, 1990, pp. 1930-1936.
10. A. Sankaranarayanan and M. Vidyasagar, Path Planning For Moving A Point Object Amidst Unknown Obstacles In A Plane: A New Algorithm And A General Theory For Algorithm Development, Proceedings of the 29th Conference on Decision and Control, 1991, pp. 1111-1119.
11. V. Lumelsky and T. Skewis, Incorporating range sensing in the robot navigation function, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 20(5), 1990, pp. 1058-1069.
12. S. L. Laubach and J. W. Burdick, An autonomous sensor-based path-planner for planetary microrovers, In Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Vol. 1, 1999, pp. 347-354.
13. Y. Horiuchi and H. Noborio, 2001, May. Evaluation of path length made in sensor-based path-planning with the alternative following, In Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164), Vol. 2, 2001, pp. 1728-1735.
14. I. Kamon and E. Rivlin, Sensory-based motion planning with global proofs, IEEE Transactions on Robotics and Automation, Vol. 13(6), 1997, pp. 814-822.
15. I. Kamon, E. Rimon and E. Rivlin, TangentBug: A Range-Sensor-Based Navigation Algorithm. The International Journal of Robotics Research, Vol. 17(9), 1998, pp. 934-953.
16. E. Magid and E. Rivlin, CAUTIOUSBUG: A Competitive Algorithm for Sensory-Based Robot Navigation. Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, pp. 2757-2762.
17. R. Sultanov, S. Sulaiman, T. Tsoy and E. Chebotareva, Virtual Collaborative Cells Modelling for UR3 and UR5 Robots in Gazebo Simulator, International Conference on Artificial Life and Robotics (ICAROB 2023), Vol. 28, 2023, pp. 149-152.
18. A. Apurin, A. Dobrokvashina, B. Abbyasov, T. Tsoy, E. A. Martinez-Garcia and E. Magid, LIRS-ArtBul: Design, modelling and construction of an omnidirectional chassis for a modular multipurpose robotic platform, Lecture Notes in Computer Science, Vol. 13719, 2022, p. 70-80.
19. F. U. Pereira, P. M. de A. Brasil, Marco Antonio De Souza Leite Cuadros and Anselmo Rafael Cukla, Analysis of Local Trajectory Planners for Mobile Robot with Robot Operating System, IEEE Latin America Transactions, Vol. 20(1), 2022, pp. 92-99.
20. A. Dobrokvashina, R. Lavrenov, T. Tsoy, E. A. Martinez-Garcia, Y. Bai, Navigation stack for the crawler robot Servosila Engineer, Proceedings of the IEEE 16th Conference on Industrial Electronics and Applications (ICIEA), 2021, p. 1907-1912.

## Authors Introduction

Mr. Spektor Ilya

Spektor Ilya is a fourth-year bachelor student in Applied Mathematics' at Tikhonov Moscow Institute of Electronics and Mathematics, HSE University, Russian Federation.

Mr. Aidar Zagirov

Aidar Zagirov is a master student in intelligent robotics at the Institute of Information Technology and Intelligent Systems, Kazan Federal University (KFU), Russia. He works as a research assistant at the Laboratory of Intelligent Robotic Systems at KFU.

Mr. Ramil Safin

Ramil Safin is a Ph.D. student and a research assistant at the Laboratory of Intelligent Robotic Systems, Institute of Information Technology and Intelligent Systems, Kazan Federal University, Russia. He teaches «Fundamentals of technical vision», «Robotic systems sensors».

Professor Evgeni Magid

A Professor, a Head of Intelligent Robotics Department and a Head of Laboratory of Intelligent Robotic Systems (LIRS) at Kazan Federal University, Russia. Professor at HSE University, Russia. Senior IEEE member. He earned his Ph.D. degree from University of Tsukuba, Japan. He authors over 200 publications.