ORIGINAL ARTICLE
UDC 004.93
https://doi.org/10.26907/2541-7746.2025.1.38-53

ROS-based navigation in unknown environment using the InsertBug algorithm: Issues of practical usage

I.A. Nekerov[1], R.N. Safin[2], T.G. Tsoy[2], S. Sulaiman[3,2], E.A. Martinez-Garcia[4], E.A. Magid[2,1] ✉

[1]*HSE University, Moscow, Russia*

[2]*Kazan Federal University, Kazan, Russia*

[3]*University of Naples Federico II, Napoli, Italy*

[4]*Universidad Autónoma de Ciudad Juárez, Zona PRONAF, Chihuahua, Ciudad Juárez, Mexico*

✉ *magid@it.kfu.ru*

Abstract

BUG algorithms are effective strategies for local path planning in unknown environments. This article presents a practical implementation of the InsertBug algorithm using the Robot Operating System (ROS) and highlights its challenges. The algorithm relies on laser sensor and odometry data to construct a locally optimal path in an unknown terrain. Its evaluation was performed in the Gazebo 3D virtual environment, employing the TurtleBot 3 Burger robot. The evaluation spanned three types of environments: mazes, settings with simple convex and concave obstacles, and office spaces. The algorithm was assessed based on the robot's overall traveled distance and accumulated turns in yaw rotations measured in radians. The findings demonstrate the effectiveness of the algorithm in diverse layouts. The implementation serves as a valuable resource to further advance autonomous navigation systems.

Keywords: robotics, control, algorithm, path planning, obstacle avoidance, robot navigation, InsertBug, TangentBug

Acknowledgments. This study was supported by the Kazan Federal University Strategic Academic Leadership Program (PRIORITY-2030).

For citation: Nekerov I.A., Safin R.N., Tsoy T.G., Sulaiman S., Martinez-Garcia E.A., Magid E.A. ROS-based navigation in unknown environment using the InsertBug algorithm: Issues of practical usage. *Uchenye Zapiski Kazanskogo Universiteta. Seriya Fiziko-Matematicheskie Nauki*, 2025, vol. 167, no. 1, pp. 38–53. https://doi.org/10.26907/2541-7746.2025.1.38-53.

ОРИГИНАЛЬНАЯ СТАТЬЯ
УДК 004.93
https://doi.org/10.26907/2541-7746.2025.1.38-53

ROS-навигация в неизвестной среде на базе алгоритма InsertBug: проблемы практического применения

Я.А. Некеров[1], Р.Н. Сафин[2], Т.Г. Цой[2], Ш. Сулайман[3,2], Э.А. Мартинез-Гарсия[4], Е.А. Магид[2,1] ✉

[1]*Национальный исследовательский университет «Высшая школа экономики», г. Москва, Россия*

[2]*Казанский (Приволжский) федеральный университет, г. Казань, Россия*

[3]*Неаполитанский университет имени Фридриха II, г. Неаполь, Италия*

[4]*Автономный университет Сьюдад-Хуарес, зона PRONAF, Чиуауа, г. Сьюдад-Хуарес, Мексика*

✉ *magid@it.kfu.ru*

Аннотация

BUG-алгоритмы являются эффективным решением для локальной навигации роботов в неизвестных средах. В статье рассмотрены особенности и сложности практической реализации алгоритма InsertBug на базе робототехнической операционной системы ROS, который использует данные лазерного дальномера и одометрии для построения локально оптимального пути в неизвестной среде. Апробация разработанного алгоритма проводилась на роботе TurtleBot 3 Burger в виртуальной среде Gazebo. Эффективность алгоритма оценивалась в лабиринтах, средах с простыми выпуклыми и вогнутыми препятствиями, а также в условиях, имитирующих офисные помещения. Критериями оценки служили пройденное роботом расстояние и сумма угловых вращений. Полученные результаты подтверждают высокую эффективность алгоритма в различных средах и демонстрируют существенный вклад, который вносит представленная реализация в дальнейшее развитие и совершенствование систем автономной навигации роботов.

Ключевые слова: робототехника, управление, алгоритм, планирование пути, избегание препятствий, навигация робота, InsertBug, TangentBug

Благодарности. Работа выполнена за счет средств Программы стратегического академического лидерства Казанского (Приволжского) федерального университета («ПРИОРИТЕТ-2030»).

Для цитирования: *Некеров Я.А., Сафин Р.Н., Цой Т.Г., Сулайман Ш., Мартинез-Гарсия Э.А., Магид Е.А.* ROS-навигация в неизвестной среде на базе алгоритма InsertBug: проблемы практического применения // Учен. зап. Казан. ун-та. Сер. Физ.-матем. науки. 2025. Т. 167, кн. 1. С. 38–53. https://doi.org/10.26907/2541-7746.2025.1.38-53.

Introduction

Optimal path planning is a fundamental task in robotics, which can be solved using two different approaches: global and local path planning. Global path planning is based on a priori known parameters and obstacles of an environment and requires a robot to navigate through a predetermined space. In local path planning, a robot moves in an unknown environment and adjusts its path relying solely on sensory data [1].

The BUG family of algorithms employs local path planning and the power of laser rangefinders (LRF) and odometry data for robot localization and navigation. Since a robot has no prior knowledge of the environment, it decides its path in real time. BUG algorithms enable a robot to plan the path in two modes: following the boundary of an obstacle and moving towards a goal.

In the available studies on this topic, BUG algorithms have primarily been described mathematically, often with no details on how to implement them. For example, in [2], the motion of a robot viewed as a material point was analyzed, and the path planning simulation took place in a 2D world. Most BUG algorithms have been implemented in a similar manner, i.e., using conceptual tools such as Matlab or custom tools developed exclusively for individual algorithms, thus making it difficult to replicate virtual experiments or adapt such algorithms to fit other algorithms of the family. In [3], the authors compared eleven BUG algorithm variations against each other on the EyeSim simulation platform. In [4–6], the effectiveness of BUG algorithms for navigation in simulated environments was examined, considering scenarios with multi-robot path planning. In [7], the CautiousBug algorithm was proposed for safe and reliable sensory-based navigation, and its performance was evaluated through the theoretical analysis supported by mathematical proofs and virtual testing of a custom Linux application developed for the TangentBug [6] and CautiousBug algorithms. In contrast, this article presents an implementation of the InsertBug algorithm in a 3D virtual environment using the open-source Gazebo simulator, which has become a de facto standard in mobile robotics simulation. The applicability of our implementation outside of Gazebo in real-world scenarios, when integrated into a real robot control system, is shown.

1. Previous Research Overview

1.1. Comparison with other algorithms. Previous research highlights that, despite sharing many similarities, BUG algorithms differ in the conditions and/or sensors they require for the path planning process. For example, the Bug1 and Bug2 algorithms use tactile sensors rather than LRF. Tactile sensors help a robot identify the presence (or absence) of an occluding obstacle in its path [1]. Yet, they are often considered a last-resort option due to their limited detection range and lower resolution compared to other types of sensors.

More advanced algorithms, such as Alg1 and Alg2, analyze the traveled path and enable a robot to visit the waypoints more than once [8]. The VisBug21 and VisBug22 algorithms [9], among the first to incorporate an LRF, extend the Bug2 algorithm. A robot equipped with LRF makes more informed and safe decisions while navigating, as it can identify obstacles from a distance and plan its path accordingly. In [10], the path lengths generated by the Boundary Following Fast Matching Method (BF-FMM) were compared with those created using the Bug1 and Bug2 algorithms to quantitatively determine the impact of environmental uncertainty on path length.

The DistBug algorithm [11] uses refined concepts that enhance its performance in complex environments with convex obstacles and mazes. The TangentBug algorithm [6] and all its derivatives (including CautiousBug [7], WedgeBug [12], and InsertBug [13]) utilize a model of the surrounding environment derived by processing data about obstacles and representing them as points. In [14], a virtual planning technique inspired by the BUG algorithm was proposed to quickly calculate paths in an obstacle-rich environment, facilitating the identification of the shortest path to the goal.

The above navigation algorithms optimize a robot's movements towards a target position, making the task of maneuvering around obstacles less critical. Comparative studies of various algorithms, such as the one conducted by Ng and Bräunl [3], provide valuable insights into their effectiveness in different environments. However, in [3], the conceptual implementation of the algorithm was presented without practical details, with the resulting code being more theoretical and having limited applicability. This comparative analysis shows that Bug1-type algorithms [1] have longer paths than TangentBug-type algorithms. Additionally, in [13], it was demonstrated that InsertBug is faster than TangentBug in route construction. Sensor fusion in robots can further improve the robustness and accuracy of BUG algorithms [15].

1.2. Application of BUG algorithms. The application of BUG algorithms has been widely studied in various contexts of autonomous navigation. In [16], an improved obstacle avoidance and navigation algorithm was introduced for autonomous mobile robots in indoor environments based on the Bug-2 algorithm. The results obtained confirm the effectiveness of applying these algorithms in real-world confined spaces.

Another study, [17], proposes a method of autonomous navigation that takes into account the kinematic constraints of a robot and uses a combination of the Dubins path and the BUG algorithm for safe and efficient path planning. The simulation based on the developed method shows the flexibility of BUG algorithms when adapted to specific robot constraints.

In [18], the use of BUG algorithms for robot motion planning in an unknown environment with danger zones was discussed, and the capability of BUG algorithms to handle navigation tasks in complex and hazardous conditions was validated.

In [19], the performance of path planning algorithms, including BUG algorithms, in dynamic environments for omnidirectional robots was explored. It was found that BUG algorithms can be effectively used even in conditions that require high maneuverability and adaptability.

Furthermore, in [20], a broad analysis of various path planning algorithms, such as BUG algorithms, and their application in complex environments was carried out. This review helps to broaden the context of BUG algorithm applications, highlighting their significance and capabilities in various challenging scenarios.

2. InsertBug Overview

The algorithms of the BUG family were developed for navigating a material point in an unknown space filled with polygonal obstacles. More advanced algorithms, such as TangentBug and InsertBug, utilize LRF to avoid obstacles and construct routes. In the InsertBug algorithm, the sensor enables a robot to identify points in space that are close to the target point and free of obstacles. The robot uses these points as temporary targets or waypoints. Upon reaching each waypoint, it verifies its proximity to the final, user-defined target point. Selecting optimal points during the path planning is crucial to minimize a travel distance, find a collision-free path, and ensure the algorithm's completeness. We use the following notations:

(a) Target *(T)* – the final target point, which the robot is tasked to reach.

(b) Locally optimal direction – the direction of the shortest path to the target.

(c) $d(\omega, T)$ – the Euclidean distance from a point ω, located within a free space, to T.

(d) $d_{\min}(T)$ – the shortest Euclidean distance for the robot moving along the boundary of an obstacle to the target.

(e) Local minimum – the basin of attraction of a local minimum of $d(\omega, T)$.

The robot starts from a user-defined initial point and moves to its target (goal point), sensing the environment and navigating through the surroundings. The algorithm's outline is given in Fig. 1:

(a) **Build an initial set of path vectors**: scan the entire space to identify circular obstacles. Use sensor data to detect these obstacles and represent them as vectors from the robot's position to the centers of the identified obstacles.

(b) **Obstacle detection and path adjustment**: determine if there are obstacles blocking the current path vector. If an obstacle is detected, calculate the intermediate waypoint to bypass it and insert the coordinates of the point into the current set of path vectors.

(c) **Path recalculation**: adjust the path vectors to navigate around the detected obstacles. Update the set of path vectors by inserting intermediate waypoints where necessary to avoid obstacles.

(d) **Navigation steps**:

Step 1. Move towards the goal T.

- If the goal is reached, stop.
- If a local minimum is detected, proceed to **Step 2.**

Step 2. Navigate around obstacles. Choose the direction to follow along the boundary of the obstacle. Insert an intermediate waypoint to divert the path around the obstacle. Move towards the intermediate waypoint while maintaining $d_{\min}(T)$:

- If the goal is reached, stop.
- If an intermediate point outside the obstacle is found and $d(\omega, T) < d_{\min}(T)$, proceed to **Step 3.**
- If the robot completes its detour around the obstacle and determines the goal is unreachable, stop.

Step 3. Transition phase. Move directly towards ω until the point Z: $d(Z, T) < d_{\min}(T)$ is reached. Then proceed to **Step 1**.

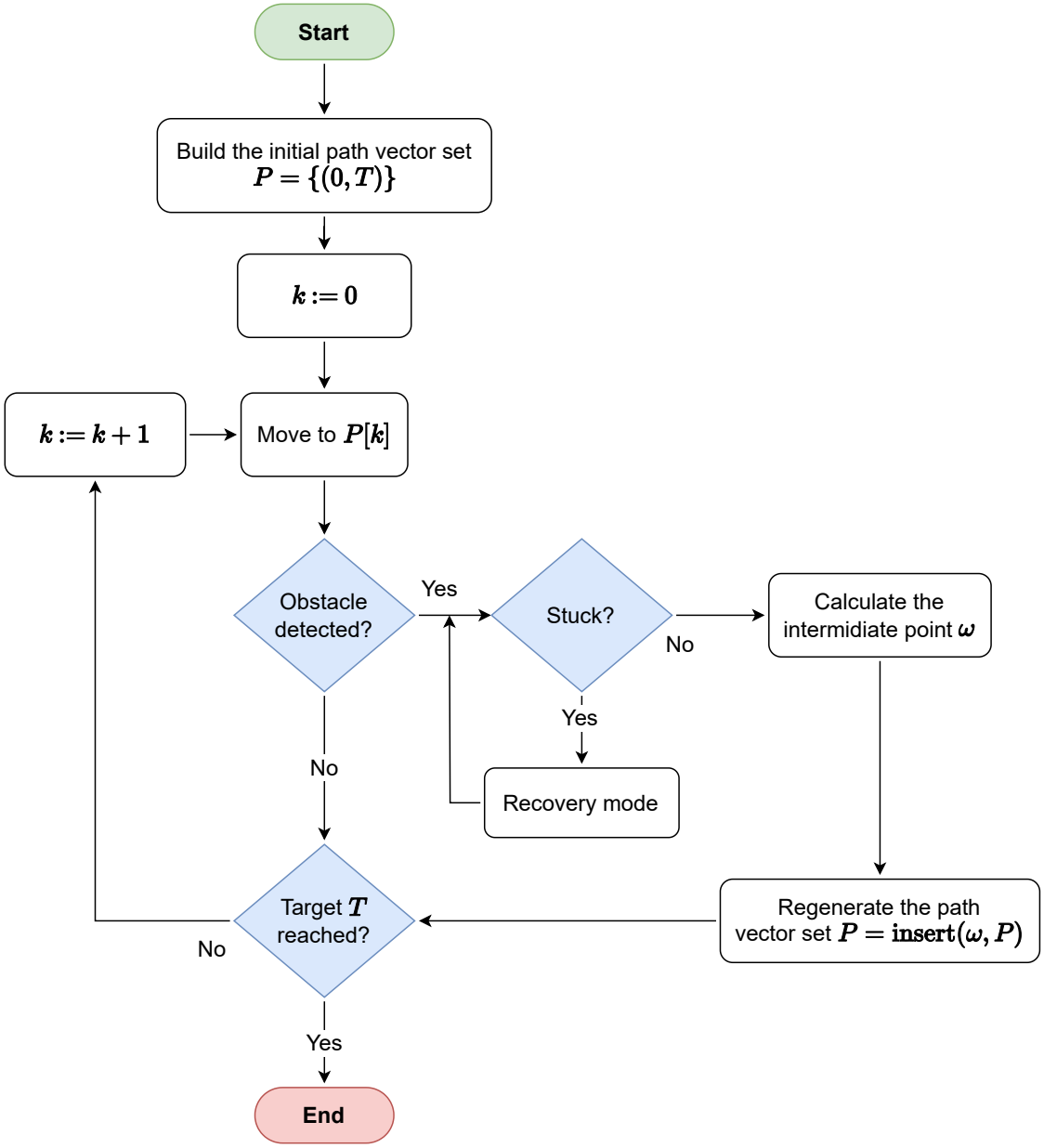

Fig. 1. Flowchart of the InsertBug algorithm

3. System Setup

The algorithm was implemented in ROS Noetic Ninjemys using C++ [21]. Interaction with the robot occurred via the ROS topics. The algorithm's implementation was tested in the Gazebo simulation environment, with 30 different maps generated to simulate conditions and scenarios that the robot might encounter. Gazebo made it possible to detect and correct errors that appeared while implementing the algorithm, as well as evaluate its accuracy under conditions that mimic real-world scenarios. This approach aligns with other ROS-based simulations in complex environments, which provide accurate modeling and visualization of a robot's behavior before deployment in real-world settings [22, 23]. The tests were conducted using the LRF-equipped Turtlebot 3 Burger robot (Fig. 2) that detects obstacles and plans its path in accordance with the InsertBug algorithm.

Fig. 2. TurtleBot 3 Burger robot in the Gazebo environment

4. Validation

The proposed algorithm was tested and validated experimentally using the Gazebo simulator. The tests were performed with three types of maps:

- **Concave and convex obstacles (C&CO):** environments with basic geometric obstacles (Fig. 3).
- **Mazes:** complex configurations designed to challenge the robot's path-finding capabilities (Fig. 4).
- **Office spaces:** simulated indoor settings resembling real-world environments (Fig. 5).

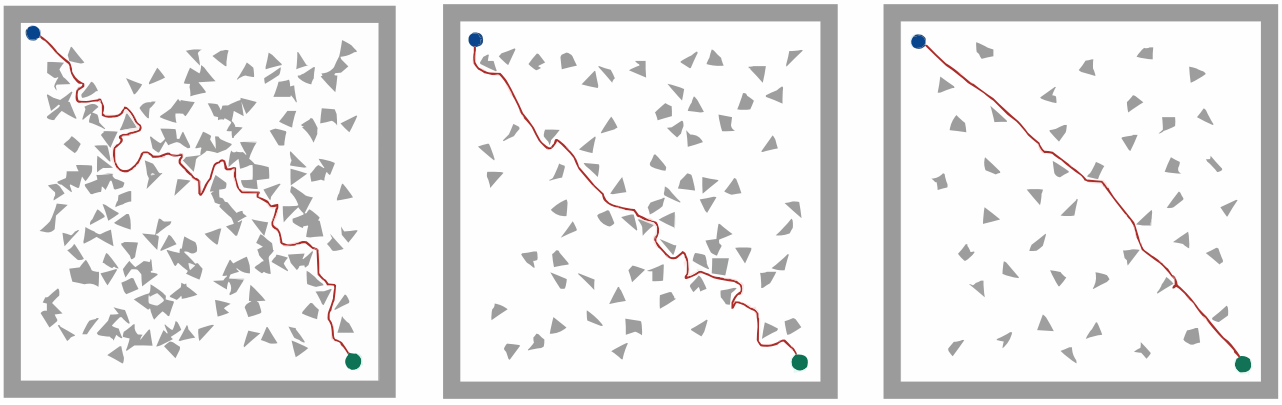

Fig. 3. Traversed path in a C&CO environment with different obstacle density levels: (a) close (CCO1), (b) medium (CCO2), and (c) distant (CCO3). The green dot shows the starting position of the robot on the map, while the blue dot is the target position

The maps (20 × 20 m) were generated using the Gazebo Worlds Construction Tool [24]. For each map, 10 pairs of starting and target points were randomly generated, with a minimum distance of 1 m and not exceeding the diagonal length of the square formed by these points. The points were placed so that they did not overlap with the obstacles. To ensure the algorithm's reliability, each map was tested in two trials.

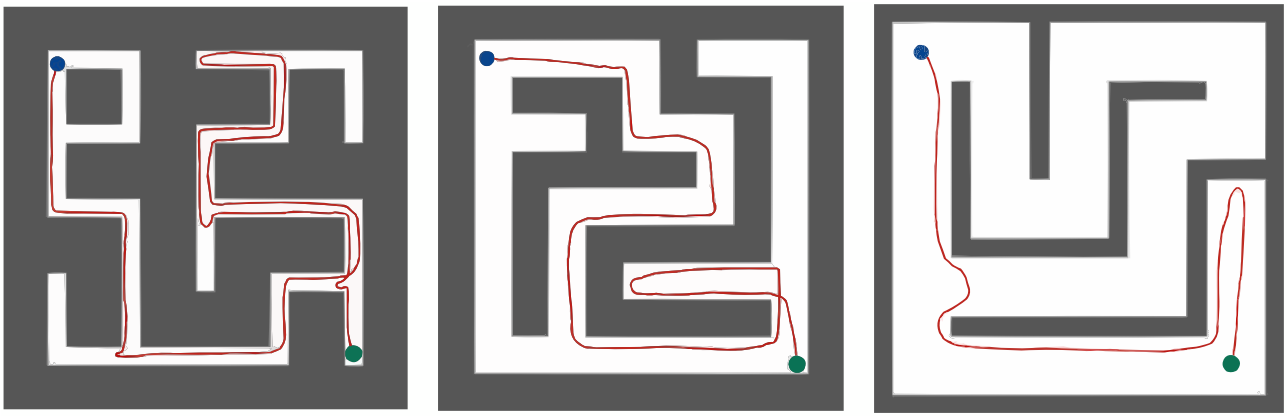

Fig. 4. Traversed path in a maze environment with different wall thickness levels: (a) thick (Maze1), (b) medium (Maze2), and (c) thin (Maze3). The green dot shows the starting position of the robot on the map, while the blue dot is the target position

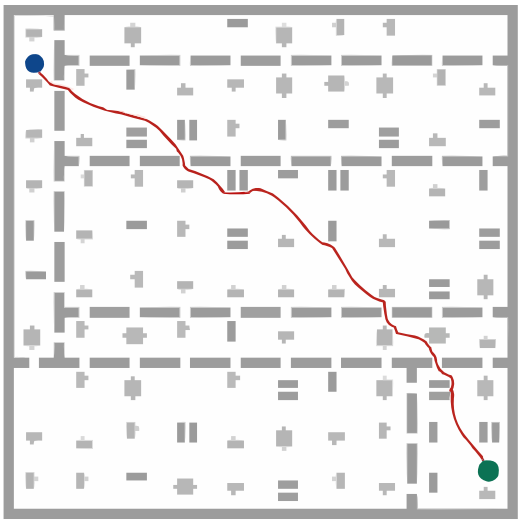

Fig. 5. Traversed path in an office space environment (OFFICE1). The green dot shows the starting position of the robot on the map, while the blue dot is the target position

4.1. Concave and convex obstacles. The concave and convex settings in the simulation represent various obstacle shapes. The convex obstacles appeared as bumps, while the concave obstacles were like dents or hollows. Both obstacle types are shown in black. White areas indicate spaces without obstacles. These settings were used to test the robot's ability to navigate around bumps and hollows in order to reach its target points effectively. The density of obstacles on the map could be adjusted by choosing one of the following three options (Fig. 3):

- **Close**: high obstacle density.
- **Medium**: medium obstacle density.
- **Distant**: low obstacle density.

The obstacles were evenly distributed and adequately spaced across the map. The total number of obstacles is proportional to the map area. In the figures, the green and blue dots on the map indicate the initial and target positions of the robot, respectively.

4.2. Mazes. The maze settings in the simulation represent confined spaces designed to challenge the algorithm's navigation capabilities. Obstacles are shown as black regions, while white areas indicate free spaces. The intricate layouts of the mazes were used to simulate complex environments where the robot must navigate through narrow passages and around obstacles to reach its target points. The maze walls had the following thickness (Fig. 4):

- **Thick**: 300 px.
- **Medium**: 200 px.
- **Thin**: 100 px.

These parameters define how wall thickness in the maze influenced the complexity of navigation. Accessibility between all free space points was ensured using an algorithm similar to depth-first graph traversal. The figures show a map where the green and blue dots indicate the initial and target positions of the robot, respectively.

4.3. Office space maps. The office space settings in the simulation represent realistic indoor environments. Obstacles such as walls, furniture, and other fixtures are shown as black regions, while white regions are free spaces. The layout mimics typical office designs, providing a mix of open areas and confined spaces that challenge the robot's ability to maneuver around common indoor obstacles to reach its target points. In (Fig. 5), office desks (40×80 px) are evenly distributed, ensuring adequate spacing between obstacles.

4.4. Robot parameters. The following parameters were set for the robot's optimal performance across the experiments:

- **Maximum translational velocity**: the robot's maximum speed when moving in a straight line is set to 0.22 m/s, determined empirically to ensure smooth linear motion.
- **Maximum rotational velocity**: the robot's maximum speed when rotating is set to 2.75 rad/s, determined empirically to ensure smooth angular motion.
- **Goal tolerance parameters**:
  - **Cartesian goal tolerance**: the tolerance for the robot's position in the XY plane is set to 0.05 m.
  - **Yaw goal tolerance**: the tolerance for the robot's orientation (yaw) is determined experimentally and set to 0.17 rad.

Limiting the translational velocity prevents the robot from moving too quickly, reducing the risk of overshooting the target points or losing control, especially in environments with tight spaces or numerous obstacles. Controlling the rotational velocity is crucial for maintaining stability and precision when changing directions. High rotational speeds could cause the robot to become unstable or make inaccurate turns, negatively affecting its ability to navigate in complex environments. Goal tolerance parameters define how close the robot needs to get to the target position and orientation before considering the goal to be reached. Tight tolerance ensures precise navigation, reducing errors in positioning that could accumulate over long distances.

4.5. Experimental results. The experiments addressed two main problems. First, inaccuracies in the robot's odometry may affect obstacle traversal assessments due to the robot's physical dimensions. Second, the robot may get stuck at an obstacle's boundaries during its mission, which requires recovery strategies.

Tables 1, 2, and 3 present the test results for various map settings. Each table has eight columns: (1) test number, (2) map name, (3) total distance traveled by the robot (in meters), (4) sum of the angles turned by the robot (including the angles turned during the recovery mode when the robot was stuck), (5) absolute difference in distance traveled between the first and second trials (in meters), (6) absolute difference in angles turned between the first and second trials (in radians), (7) normalized difference in distance traveled relative to the first trial, calculated by dividing the difference by the total distance traveled by the robot in the first trial, and (8) normalized difference in angles turned, calculated similarly to the normalized difference in distance traveled.

In some cases, the robot changed the direction multiple times when it detected it was moving away from an obstacle. A significant deviation from the planned trajectory was observed in test no. 7 (Maze 4), where the robot got stuck near the obstacle corners due to odometry drift. The recovery mode was activated, causing the robot to spin in order to escape the obstacle boundary. As a result, the absolute normalized yaw difference in this case was 18.5 %, compared to an average of 4.24 % for this map type.

In contrast, the C&CO maps yielded better results. The distances traveled were close to the shortest possible straight-line paths, as the robot relied less on boundary following and more on optimized local routing. These results confirm the algorithm's ability to build shorter paths on maps with convex and concave obstacles.

The tests in the office environments produced varying results. On maps such as OFFICE1, OFFICE2, and OFFICE5, the robot followed relatively efficient trajectories. However, on maps like OFFICE3, OFFICE4, and OFFICE7, the robot got stuck more frequently and activated the recovery mode, leading to higher absolute normalized yaw difference values. In particular, OFFICE7 showed a significant absolute normalized deviation from the course (45.01 %) due to the accumulated errors triggering the recovery mode. The absolute normalized difference in the path length generally remained within acceptable limits, varying with the complexity of the map layout.

The average travel time in the mazes was twice as long as in other setting types, with the robot making nearly four times as many turns. Additionally, although the total distances traveled by the robot in the C&CO and office settings were similar, the average number of turns in the office spaces was about 1.5 times higher.

Conclusions

The InsertBug algorithm implemented in the ROS framework was studied using the TurtleBot3 Burger platform tested in the Gazebo simulator. It addresses practical challenges, which are often overlooked in theoretical models. Key issues include: the robot's odometry inaccuracies affecting the obstacle traversal assessments due to the robot's physical dimensions; recovery strategies used by the robot when it gets stuck at the boundaries of an obstacle.

The algorithm proved to be effective for real-world scenarios. Its performance was evaluated based on three distinct map types: convex and concave obstacles, mazes, and office spaces. The results of the tests varied across the environments and demonstrate that the algorithm's

capacity to navigate is better in simpler settings than in complex scenarios, such as maze navigation and maneuvering within office layouts.

Table 1. Comparison of path lengths, yaw, and differences across various mazes

No	**Map**	**Path length (m)**	**Total yaw (rad)**	**Abs. length diff. (m)**	**Abs. yaw diff. (rad)**	**Abs. norm. length diff. (%)**	**Abs. norm. yaw diff. (%)**
1	Maze1	95.55	344.68	2.1	7.86	2.2	2.28
2		93.45	336.82				
3	Maze2	17.66	16.71	1.05	2.45	5.95	14.66
4		18.71	19.16				
5	Maze3	120.77	378.41	0.24	1.62	0.2	0.43
6		121.01	376.79				
7	Maze4	87.91	220.09	0.33	0.58	0.36	0.26
8		88.24	219.51				
9	Maze5	21.34	36.22	1.07	1.71	5.0	4.72
10		20.27	34.51				
11	Maze6	80.12	291.03	0.7	0.53	0.87	0.18
12		79.42	290.50				
13	Maze7	129.32	550.75	3.51	101.72	2.71	18.5
14		125.81	449.03				
15	Maze8	45.51	100.21	1.39	0.76	3.05	0.76
16		44.12	99.45				
17	Maze9	55.67	87.64	0.0	0.42	0.0	0.48
18		55.67	88.06				
19	Maze10	102.73	305.12	0.7	0.45	0.68	0.15
20		102.03	304.67				

Table 2. Comparison of path lengths, yaw, and differences across various C&CO environments

No	Map	Path length (m)	Total yaw (radians)	Abs. length diff. (m)	Abs. yaw diff. (radians)	Abs. norm. length diff. (%)	Abs. norm. yaw diff. (%)
1	CCO1	40.01	59.32	0.23	0.72	0.57	1.21
2		40.24	60.04				
3	CCO2	25.67	48.05	0.78	0.11	3.04	0.23
4		24.89	47.94				
5	CCO3	38.34	43.54	0.44	0.56	1.14	1.29
6		37.90	42.98				
7	CCO4	31.67	55.23	0.38	0.47	1.19	0.85
8		32.05	54.76				
9	CCO5	29.98	80.53	0.56	27.52	1.86	34.17
10		30.54	53.01				
11	CCO6	23.09	48.67	0.28	0.37	1.22	0.76
12		22.81	49.04				
13	CCO7	34.65	56.08	0.76	0.09	2.2	0.16
14		33.89	55.99				
15	CCO8	33.29	39.49	0	0.53	0	1.34
16		33.29	40.02				
17	CCO9	20.01	32.04	1.02	9.99	5.1	32.2
18		18.99	42.03				
19	CCO10	35.67	55.12	0.33	0.23	0.92	0.42
20		36	54.89				

Table 3. Comparison of path lengths, yaw, and differences across various offices

No	Map	Path length (m)	Total yaw (rad)	Abs. length diff. (m)	Abs. yaw diff. (rad)	Abs. norm. length diff. (%)	Abs. norm. yaw diff. (%)
1 2	OFFICE1	29.03 28.68	65.88 64.03	0.35	1.85	1.21	2.81
3 4	OFFICE2	50.06 51.04	89.32 90.04	0.98	0.72	1.96	0.81
5 6	OFFICE3	43.09 42.56	83.55 84.79	0.53	1.24	1.23	1.48
7 8	OFFICE4	21.01 22.09	39.02 41.01	1.08	2.00	5.14	5.12
9 10	OFFICE5	32.04 33.09	79.03 83.14	1.05	4.11	3.28	5.21
11 12	OFFICE6	45.68 47.03	98.32 96.56	1.35	1.76	2.95	1.78
13 14	OFFICE7	19.03 19.56	34.43 50.04	0.53	15.61	2.78	45.01
15 16	OFFICE8	32.54 33.05	69.03 68.39	0.51	0.64	1.57	0.92
17 18	OFFICE9	60.21 58.90	72.34 74.03	1.31	1.69	2.17	2.33
19 20	OFFICE10	47.93 49.03	98.32 99.07	1.1	0.75	2.3	0.76

Conflicts of Interest. The authors declare no conflicts of interest.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

References

1. Lumelsky V.J., Stepanov A.A. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. In: Cox I.J., Wilfong G.T. (Eds.) *Autonomous Robot Vehicles*. New York, NY, Springer, 1990, pp. 363–390. https://doi.org/10.1007/978-1-4613-8997-2_27.

2. Dwaracherla V., Thakar S., Vachhani L., Gupta A., Yadav A., Modi S. Motion planning for point-to-point navigation of spherical robot using position feedback. *IEEE/ASME Trans. Mechatron.*, 2019, vol. 24, no. 5, pp. 2416–2426. https://doi.org/10.1109/TMECH.2019.2934789.

3. Ng J., Bräunl T. Performance comparison of bug navigation algorithms. *J. Intell. Rob. Syst.*, 2007, vol. 50, no. 1, pp. 73–84. https://doi.org/10.1007/s10846-007-9157-6.

4. McGuire K.N., de Croon G.C.H.E., Tuyls K. A comparative study of bug algorithms for robot navigation. *Rob. Auton. Syst.*, 2019, vol. 121, art. 103261. https://doi.org/10.1016/j.robot.2019.103261.

5. Mohamed E.F., El-Metwally K., Hanafy A.R. An improved Tangent Bug method integrated with artificial potential field for multi-robot path planning. *Proc. 2011 Int. Symp. on Innovations in Intelligent Systems and Applications*. IEEE, 2011, pp. 555–559. https://doi.org/10.1109/INISTA.2011.5946136.

6. Kamon I., Rimon E., Rivlin E. TangentBug: A range-sensor-based navigation algorithm. *Int. J. Rob. Res.*, 1998, vol. 17, no. 9, pp. 934–953. https://doi.org/10.1177/027836499801700903.

7. Magid E., Rivlin E. CautiousBug: A competitive algorithm for sensory-based robot navigation. *Proc. 2004 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2004, pp. 2757–2762. https://doi.org/10.1109/IROS.2004.1389826.

8. Sankaranarayanan A., Vidyasagar M. A new path planning algorithm for moving a point object amidst unknown obstacles in a plane. *Proc. IEEE Int. Conf. on Robotics and Automation*. Vol. 3. IEEE, 1990, pp. 1930–1936. https://doi.org/10.1109/ROBOT.1990.126290.

9. Lumelsky V.J., Skewis T. Incorporating range sensing in the robot navigation function. *IEEE Trans. Syst., Man, Cybern.*, 1990, vol. 20, no. 5, pp. 1058–1069. https://doi.org/10.1109/21.59969.

10. Chiang C.H., Liu J.-S., Chou Y.-S. Comparing path length by boundary following fast matching method and Bug algorithms for path planning. In: Chien B.-C., Hong T.-P. (Eds.) *Opportunities and Challenges for Next-Generation Applied Intelligence*. Ser.: Studies in Computational Intelligence. Vol. 214. Berlin, Heidelberg, Springer, 2009, pp. 303–309. https://doi.org/10.1007/978-3-540-92814-0_47.

11. Kamon I., Rivlin E. Sensory-based motion planning with global proofs. *IEEE Trans. Rob. Autom.*, 1997, vol. 13, no. 6, pp. 814–822. https://doi.org/10.1109/70.650160.

12. Laubach S.L., Burdick J.W. An autonomous sensor-based path-planner for planetary microrovers. *Proc. 1999 IEEE Int. Conf. on Robotics and Automation*. Vol. 1. IEEE, 1999, pp. 347–354. https://doi.org/10.1109/ROBOT.1999.770003.

13. Xu Q.-L., Tang G.-Y. Vectorization path planning for autonomous mobile agent in unknown environment. *Neural Comput. Appl.*, 2013, vol. 23, no. 7, pp. 2129–2135. https://doi.org/10.1007/s00521-012-1163-3.

14. Sharma N., Thukral S., Aine S., Sujit P.B. A virtual bug planning technique for 2D robot path planning. *Proc. 2018 Annu. Am. Control Conf. (ACC)*. IEEE, 2018, pp. 5062–5069. https://doi.org/10.23919/ACC.2018.8430678.

15. Eryomin A., Safin R., Tsoy T., Lavrenov R., Magid E. Optical sensors fusion approaches for map construction: A review of recent studies. *J. Rob., Networking Artif. Life*, 2023, vol. 10, no. 2, pp. 127–130. https://doi.org/10.57417/jrnal.10.2_127.

16. Khan F., Alakberi A., Almaamari S., Beig A.R. Navigation algorithm for autonomous mobile robots in indoor environments. *Proc. 2018 Advances in Science and Engineering Technology Int. Conf. (ASET)*. IEEE, 2018, pp. 1–6. https://doi.org/10.1109/ICASET.2018.8376834.

17. Xu Q.L., Yu T., Bai J. The mobile robot path planning with motion constraints based on Bug algorithm. *Proc. 2017 Chinese Automation Congr. (CAC)*. IEEE, 2017, pp. 2348–2352. https://doi.org/10.1109/CAC.2017.8243168.

18. Jahanshahi H., Jafarzadeh M., Sari N.N., Pham V.-T., Huynh V.V., Nguyen X.Q. Robot motion planning in an unknown environment with danger space. *Electronics*, 2019, vol. 8, no. 2, art. 201. https://doi.org/10.3390/electronics8020201.

19. Haro F., Torres M. A comparison of path planning algorithms for omni-directional robots in dynamic environments. *Proc. 2006 IEEE 3rd Latin American Robotics Symp.* IEEE, 2006, pp. 18–25. https://doi.org/10.1109/LARS.2006.334319.

20. Janis A., Bade A. Path planning algorithm in complex environment: A survey. *Trans. Sci. Technol.*, 2016, vol. 3, no. 1, pp. 31–40.

21. Lentin J., Cacace J. *Mastering ROS for Robotics Programming: Design, Build, and Simulate Complex Robots Using the Robot Operating System*. Birmingham, Packt Publ., 2018. 580 p.

22. Marian M., Stîngă F., Georgescu M.-T., Roibu H., Popescu D., Manta F. A ROS-based control application for a robotic platform using the Gazebo 3D simulator. *Proc. 2020 21st Int. Carpathian Control Conf. (ICCC)*. IEEE, 2020, pp. 1–5. https://doi.org/10.1109/ICCC49264.2020.9257256.

23. Takaya K., Asai T., Kroumov V., Smarandache F. Simulation environment for mobile robots testing using ROS and Gazebo. *Proc. 2016 20th Int. Conf. on System Theory, Control and Computing (ICSTCC)*. IEEE, 2016, pp. 96–101. https://doi.org/10.1109/ICSTCC.2016.7790647.

24. Abbyasov B., Lavrenov R., Zakiev A., Yakovlev K., Svinin M., Magid E. Automatic tool for Gazebo world construction: From a grayscale image to a 3D solid model. *Proc. 2020 IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7226–7232. https://doi.org/10.1109/ICRA40945.2020.9196621.

Author Information

Iaroslav A. Nekerov, Undergraduate Student, HSE University
E-mail: *yaanekerov@edu.hse.ru*

Ramil N. Safin, Senior Lecturer, Kazan Federal University
E-mail: *safin.ramil@it.kfu.ru*
ORCID: https://orcid.org/0000-0003-1429-1876

Tatyana G. Tsoy, PhD, Senior Lecturer, Kazan Federal University
E-mail: *tt@it.kfu.ru*
ORCID: https://orcid.org/0000-0002-5715-7768

Shifa Sulaiman, PhD, Research Associate, University of Naples Federico II; Kazan Federal University
E-mail: *shifa1910@it.kfu.ru*
ORCID: https://orcid.org/0000-0002-5330-0053

Edgar A. Martinez-Garcia, PhD, Professor, Universidad Autónoma de Ciudad Juárez
E-mail: *edmartin@uacj.mx*
ORCID: https://orcid.org/0000-0001-9163-8285

Evgeni A. Magid, PhD, Professor, Head of Intelligent Robotics Department, Kazan Federal University; HSE University
E-mail: *magid@it.kfu.ru*
ORCID: https://orcid.org/0000-0001-7316-5664

Информация об авторах

Ярослав Александрович Некеров, студент бакалавриата, Национальный исследовательский университет «Высшая школа экономики»
E-mail: *yaanekerov@edu.hse.ru*

Рамиль Набиуллович Сафин, старший преподаватель, Казанский (Приволжский) федеральный университет
E-mail: *safin.ramil@it.kfu.ru*
ORCID: https://orcid.org/0000-0003-1429-1876

Татьяна Григорьевна Цой, кандидат технических наук, старший преподаватель, Казанский (Приволжский) федеральный университет
E-mail: *tt@it.kfu.ru*
ORCID: https://orcid.org/0000-0002-5715-7768

Шифа Сулайман, PhD, научный сотрудник, Неаполитанский университет имени Фридриха II; Казанский (Приволжский) федеральный университет
E-mail: *ssajmech@gmail.com*
ORCID: https://orcid.org/0000-0002-5330-0053

Эдгар Алонсо Мартинез-Гарсия, PhD, профессор, Автономный университет Сьюдад-Хуарес
E-mail: *edmartin@uacj.mx*
ORCID: https://orcid.org/0000-0001-9163-8285

Евгений Аркадьевич Магид, PhD, профессор, заведующий кафедрой интеллектуальной робототехники, Казанский (Приволжский) федеральный университет; Национальный исследовательский университет «Высшая школа экономики»
E-mail: *magid@it.kfu.ru*
ORCID: https://orcid.org/0000-0001-7316-5664

Received August 14, 2024
Accepted September 24, 2024

Поступила в редакцию 14.08.2024
Принята к публикации 24.09.2024