Implementation of VisBug-21 and VisBug-22 Path Planning Algorithms Using ROS Noetic

Viktoriia Mirzoian
HSE University, Moscow, Russian Federation

Maxim Mustafin
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: vimirzoyan@edu.hse.ru, mustafin@it.kfu.ru, magid@it.kfu.ru

Abstract

Local navigation algorithms are crucial for autonomous robots operating in unknown environments where a presence of obstacles and dynamic changes pose significant challenges. A focus of these algorithms is to enable a real-time path calculation, allowing a robot to adapt its states dynamically based on corresponding environmental conditions, despite an absence of prior knowledge about surroundings. This paper presents an implementation of the VisBug-21 and VisBug-22 algorithms, designed to address challenges of a local navigation. The algorithms were implemented for a differential drive robot Turtlebot3 Burger using Robot Operation System (ROS). Virtual experiments were performed in the Gazebo simulator employing a simple 3D environment that contained only convex obstacles and a small 3D maze.

Keywords: Path-Planning, Mobile Robots, Local Navigation, VisBug-21, VisBug-22, Gazebo, ROS Noetic

1. Introduction

Robotics researchers face a significant challenge when it comes to efficient navigation [1]. A current focus in robot motion planning concentrates around two different models, each based on distinct assumptions about available data for planning [2]. The first model, known as a path planning with complete information or the *piano movers problem*, assumes perfect knowledge about a robot and obstacles [3]. This paper is concerned with the second model, which is a path planning with incomplete information [4]. In this model, some degree of uncertainty exists, and missing data are obtained from local sources, e.g., laser range finders or cameras. One advantage of this model is the ability to incorporate sensory feedback, transforming a motion planning into a continuous dynamic process. It also eliminates a need for an analytic representation of obstacle boundaries [5]. A key issue in the motion planning with incomplete information is how to integrate sensory data into a planning function [6], [7].

This paper describes how to implement two mathematically described algorithms that can guide a mobile robot through complex environments with various obstacles, using basic distance data provided by a range finder or stereo vision. To achieve this goal, a simplified model of a vision sensor [8] is proposed, which functions similarly to a rangefinder, providing the robot with coordinates of obstacles' boundaries within a limited radius of its visibility [9].

2. Algorithms' overview

BUG family of algorithms provides efficient and effective ways for robots to navigate through populated with obstacles environments and reach their target locations [10]. These algorithms are widely used in robotics research and are valuable tools for path planning and navigation applications. VisBug-21 and VisBug-22 algorithms are an enhanced version of the basic BUG algorithm [8], which allow the robot making more informed decisions by introducing a vision control.

2.1. *VisBUG-21*

Incorporating vision into Bug2 algorithm is to mentally reconstruct a segment of Bug2 path visible to the robot at each step [11]. Using this reconstructed path, the robot identifies a farthest point and moves directly towards it. It is important to note that a segment continuity is a crucial factor in this process, rather than remembering the entire segment itself [12], [13]. The algorithm is comprised of two key components: a main body that plans a motion along a path responsible for generating the

next intermediate target T_{i+1}, and a procedure "Compute T_{i+1}" that tests its reachability [8]. Typically, the main body only operates in step S1, but step S2 is executed when the robot is navigating a (locally) convex boundary of an obstacle, preventing it from utilizing its vision to define next intermediate target T_{i+1}.

Main body:

1. Move towards current target T_i, execute "Compute T_i-21" and check:
   a. If start point C = target point T, stop the procedure.
   b. If T is unreachable, stop the procedure.
   c. If C = T_i, go to step 2.
2. Move along the obstacle boundary, execute "Compute T_i-21" and check:
   a. If C = T, stop the procedure.
   b. If T is unreachable, stop the procedure.
   c. If C ≠ T_i, go to step 1.

Procedure Compute Ti-21 builds a path that BUG2 algorithm would build, and then selects a farthest point on a section of a path that it sees. This point becomes a next current target.

2.2. *VisBug-22*

It is possible to create a different method, similar to BUG2 mechanism but with a more opportunistic approach compared to VisBug-21. Instead of strictly following a BUG2 path, the robot can deviate from it if it spots more promising opportunities, while still ensuring a convergence. This alternative process is referred to as VisBug-22. A design of this algorithm bears a strong resemblance to VisBug-21 algorithm, albeit with one notable difference - the robot no longer prioritizes making sure that all intermediate targets T_j are located on the BUG2 path [11]. Instead, it strives to select intermediate targets that are as close as possible to the target T while still being on the same line. This leads to a distinct convergence mechanism and a behavior that sets it apart from VisBug-21 algorithm. The algorithm is made up of two parts: a main body, which is the same as the main body of VisBug-21, and a procedure named «Compute T_i-22». This procedure calculates a next intermediate target T based on a current position C of the robot and also checks if the target can be reached.

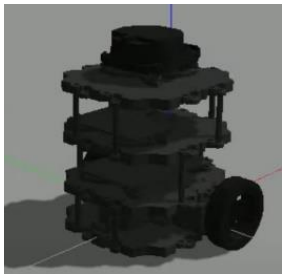

Fig.1. Turtlebot3 Burger in the Gazebo environment

Procedure Compute T_i-22 keeps identifying points along the BUG2 path or a quasi- BUG2 path segment, until a better point (in terms of its proximity to T), S', is identified on the line Start-Target. Then S' becomes the starting point of another quasi- BUG2 path segment, and the process repeats. As a result, unlike algorithms Bug2 and VisBug-21, where each defined hit point has its matching leave point, in VisBug-22 no such matching necessarily occurs.

3. Configuration

An environment configuration consists of Robot Operation System (ROS), Gazebo, and the TurtleBot. ROS was chosen due to its modular framework that allows developers breaking down complex algorithms into smaller, reusable software components known as nodes. This modularity enables easier development, testing, and maintenance of robot motion algorithms [14]. Additionally, ROS provides a wide range of sensor drivers and libraries that make it easier to integrate different sensors into a robot's system. This allows robots gathering data from various sources, such as cameras, LiDAR, or depth sensors, which is crucial for navigating in unknown environments [15].

The benefit of using Gazebo for programming algorithms for robot movement in unknown conditions is a realistic physics simulation environment for robots [16], [17] allowing developers to test and refine their algorithms in various unknown conditions without a need for physical prototypes and without damage risks. This enables faster development and iteration cycles. Gazebo provides a highly customizable and extensible platform where developers can create virtual environments with different terrain types, lighting conditions, or weather effects to simulate distinct unknown conditions [18]. This flexibility allows for comprehensive testing and robustness evaluation of algorithms in various scenarios.

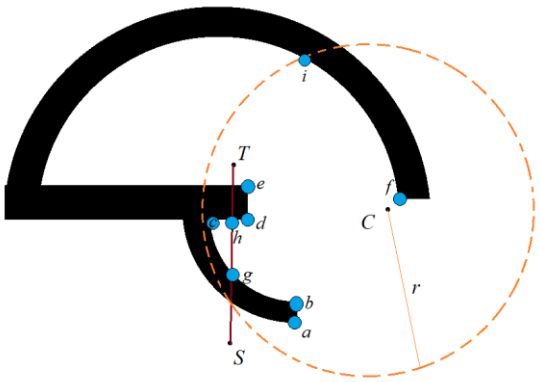

Fig. 2. Schematic representation of the search for "interesting" points

TurtleBot 3 Burger virtual model of a robot was employed for experiments in Gazebo simulator (Fig. 1) due to its simplicity that makes it accessible for users of

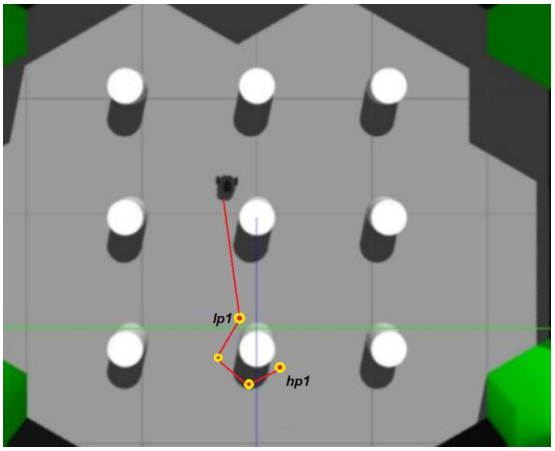

Fig. 3. VisBug-21 path in the map with convex pillars

all levels of expertise, including beginners [19], round shape and differential drive chassis that ease local navigation. The robot's small size allows navigating through tight spaces, making it suitable for various environments, including confined ones.

4. Implementation details

Python 3 was utilized to implement VisBug-21 and VisBug-22. The package includes two separate files containing the algorithms for VisBug-21 and VisBug-22, representing distinct processes. Additionally, the main service offers user services for point navigation, clockwise and counterclockwise wall following. To ensure the functionality of all services, the following libraries and modules were employed [20], [21]:

- *rospy*: is a Python client library specifically designed for ROS.
- *sensor_msgs* module: records laser measurements.
- *geometry_msgs* module: generates and transmitts messages pertaining to setting the robot's position and utilizing points.
- *gazebo_msgs* module: defines and configures the robot's current state.
- *nav_msgs* module: odometry use.
- *tf* module: allows angle transformations.
- *std_srv* module: involved in running ROS services.

A main difficulty in implementing the algorithm was a vision system design for the robot. The original articles [8], [11] described only a visual sensor range concept. It was decided to model the vision system using a rangefinder package of ROS. The algorithm contains a method that serves as a callback for processing laser scan messages. The purpose of this function is to extract specific ranges of data from the laser scan message and store them as global variables. It defines global variables for each range of 360 degrees around the robot, populates a range dictionary with minimum range values from specific ranges and assigns a maximal vision range value if no valid readings are available.

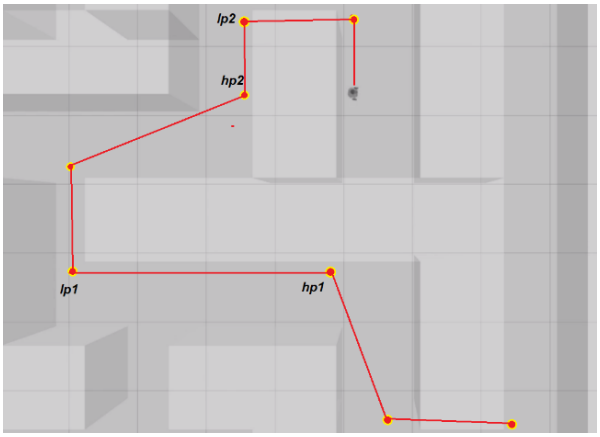

Fig. 4. VisBug-21 path in the maze

Another difficulty was related to a practical identify-cation of "interesting" points that were described in [1], [2]. At the current location *C* of the robot (Fig. 2), the algorithm processes a grid of its sensor's vision within its radius *r* (an area inside the orange circle). The robot rotates 360 degrees in place and searches for interesting points (marked with blue dots) that form three groups:

- corners of obstacles (indicated as *a*, *b*, *c*, *d*, *e*, *f*);
- visible points of obstacles' boundaries at distance *r* (indicated as *i*);
- visible intersections of M-line and obstacles (*g*, *h*).

5. Validation

The algorithms' implementation was validated in two types of environments: a world of identical pillars of a cylindric shape and a maze. Virtual tests in Gazebo were carried out for reachable and unreachable targets. 20 (Start, Target) pairs were selected in the first environment and 25 pairs in the second. Each test was run multiple times. While most tests were successful, 9.5% of the tests failed due to situations at the boundaries of the obstacles where the robot's wheels were not considered by the sensors and thus the robot got stuck.

Table 1 presents system configuration that was used for the validation.

Table 1. System configuration

Parameter	*Characteristics*
Operation System details	Ubuntu 20.0.4
Memory	16.00 GB
Processor	AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz

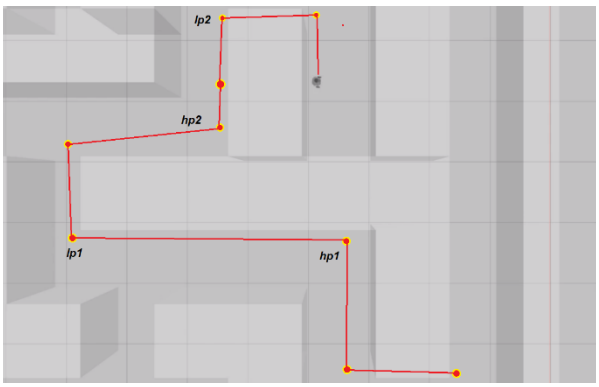

Fig. 5. VisBug-22 path in the maze

Fig. 3 present a path constructed by VisBug-21 algorithm in the convex environment. Fig. 4 and Fig. 5 demonstrate paths with the same S, T selection in the maze environment by VisBug-21 and VisBug-22 respectively.

6. Conclusions

The paper presented the implementation of VisBug-21 and VisBug-22 algorithms in Python programming language for Turtlebot3 Burger robot model using ROS Noetic. The implementation was validated in two different environments of Gazebo simulator and demonstrated a success in 90.5% of the test cases.

Acknowledgment

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

References

1. G. Huang, Visual-inertial navigation: A concise review. In 2019 international conference on robotics and automation (ICRA), May 2019, pp. 9572-9582.
2. J. Kerr and K. Nickels, Robot operating systems: Bridging the gap between human and robot Proceedings. 44th IEEE Southeastern Symposium on System Theory (SSST), 2012.
3. G. E. Jan, K. Y. Chang and I. Parberry, Optimal path planning for mobile robot navigation. IEEE/ASME Transactions on mechatronics, 13(4), 2008, pp. 451-460.
4. 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, 2003, pp. 2757-2762.
5. I. Kamon and E. Rivlin, Sensory-based motion planning with global proofs. IEEE Transactions on Robotics and Automation, vol. 13, 1997, pp. 814-822.
6. E. Magid, Simulation Tools for Urban Search and Rescue Robotics. International Conference on Artificial Life and Robotics (ICAROB 2023), 2023, pp. 4-11.
7. I. Mavrin, T. Tsoy and E. Magid, Modified E3 exploration algorithm for unknown environments with obstacles. 13th Asian Control Conference (ASCC), May 2022, pp. 1413-1418.
8. V. Lumelsky and A. Stepanov, Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. In Autonomous robot vehicles, 1990, pp 363-390.
9. H. Noborio, Evaluation of Path Length Made in SensorBased Path-Planning with the Alternative Following. Proceedings of the 2001 IEEE International Conference on Robotics 8 Automation, 2001, pp. 1728-1735.
10. H. Noborio, Several Path-Planning Algorithms of a Mobile Robot for an Uncertain Workspace and their Evaluation. Proc. of the IEEE Intelligent Motion Control, 1990, pp. 289-294.
11. V. Lumelsky and T. Skewis, Incorporating range sensing in the robot navigation function. IEEE Transactions on Systems, Man, and Cybernetics, 1990, pp. 1058-1069.
12. A. Yufka and O. Parlaktuna, Performance comparison of bug algorithms for mobile robots. In Proceedings of the 5th international advanced technologies symposium, 2009, pp. 13-15.
13. 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. Proc. of the 29th Conference on Decision and Control, 1991, pp. 1111-1119.
14. N. James and T. Bräunl, Performance comparison of bug navigation algorithms. Journal of Intelligent and Robotic Systems 50, no. 1, 2007, pp. 73- 84.
15. 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.
16. 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.
17. 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.
18. R. Gabdrahmanov, T. Tsoy, Y. Bai, M. Svinin and E. Magid, Automatic Generation of Random Step Environment Models for Gazebo Simulator. Lecture Notes in Networks and Systems, №324, 2021, pp. 408-420.
19. E. Chebotareva, T. Tsoy, B. Abbyasov, J. Mustafina, E. A. Martinez-Garcia, Y. Bai and M. Svinin, On the Problems of SLAM Simulation for Mobile Robots in the Arctic Conditions. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12336, 2020, p. 34-44.
20. F. Duan, W. Li and Y. Tan, The Framework and Fundamental Use of ROS. In Intelligent Robot: Implementation and Applications, 2023, pp. 43-69.
21. A. Dobrokvashina, R. Lavrenov, T. Tsoy, E. A. Martinez-Garcia and 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

Ms.Viktoria Mirzoian

She is a master student of The Internet of Things and cyber physical systems program at HSE University, Russian Federation. She got a bachelor's degree in information systems and technologies at Saint Petersburg Mining University, Russia.

Mr. Maksim Mustafin

He is a master student of Intelligent Robotics program at Institute of Information Technology and Intelligent Systems (ITIS), Kazan Federal University (KFU), Russia. In 2023 he obtained a bachelor degree at ITIS, KFU.

Professor Evgeni Magid

He is 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. Previously he worked at University of Bristol, UK; Carnegie Mellon University, USA; University of Tsukuba, Japan; National Institute of Advanced Industrial Science and Technology, Japan. He earned his Ph.D. degree from University of Tsukuba, Japan. He authors over 200 publications.