DistBug path planning algorithm package for ROS Noetic

Alexander Pak
Higher School of Economics University, Moscow, Russian Federation

Alexander Eremin
Intelligent Robotics Department, Kazan Federal University, Kazan, Russian Federation

Tatyana Tsoy
Intelligent Robotics Department, Kazan Federal University, Kazan, Russian Federation
E-mail: adpak_1@edu.hse.ru, aneremin@it.kfu.ru, tt@it.kfu.ru

Abstract

Algorithms of path-planning in an unknown environment play an important role in robotics. They do not require a prior information about obstacles' locations around a robot and allow calculating a path in a real time. This article presents an implementation of a sensory-based DistBug algorithm, which operates reactively using range data for immediate decision-making without constructing a world model. The algorithm was programmed in Python using robot operating system (ROS) and validated in the Gazebo simulator. For virtual experiments Turtlebot 3 Burger mobile robot was employed. The experiments were conducted in two types of environment: an environment with convex obstacles and a maze. The paper demonstrates analysis of experiments using several standard criteria of a path quality estimation.

Keywords: BUG algorithm, path planning, sensor-based navigation.

1. Introduction

A basic motion planning issue is to compute a path from a given starting point to a specified destination point [1]. In robotics, there are two approaches for path planning: global and local [2]. Global planning typically relies on preliminarily known data about an environment [3]. Local path planning uses measurements from sensors and knowledge of a target point [4] while the robot moves in an unknown environment avoiding obstacles [5].

The algorithms of the BUG family use only local sensory information and odometry data to control movement and address a challenge of local navigation without a need to create a comprehensive map of an environment [6]. BUG algorithms have two motion modes: moving toward a target and following a boundary of an obstacle [7]. The robot moves in a straight line towards the target until it reaches a minimum distance to an obstacle [8]. Further, the robot follows a boundary of the obstacle [9]. The algorithm uses *leaving conditions* to determine when to exit the obstacle boundary and head directly toward the target again [10]. The robot receives measurements from its sensors to detect nearby obstacles and plan a subsequent trajectory [11].

In this article, we present an implementation of the DistBug [12] algorithm using robot operating system (ROS) [13]. The algorithm was tested in the Gazebo simulator [14]. The navigation was performed in a 3D environment for the TurtleBot 3 Burger robot model [15]. The robot measures a distance to an obstacle with a 2D LiDAR of 3.5 m range and 360 degrees field of view.

2. A brief description of the DistBug algorithm

The sensory-based DistBug algorithm was designed to reach a goal in an unknown environment or indicate that the goal is unreachable [12]. The robot does not construct a model of the world and relies on sensory data to make decisions [16]. The method consists of two behavior models, represented as movement modes: a straight-line motion towards the goal and an obstacle boundary following. A condition for leaving an obstacle is based on a free range in a goal direction when the target is on a line of sight or a distance to it gradually decreases. The algorithm utilizes sensory data with a limited range of view, and the robot moves in steps of a given size. The core concept of the algorithm is the following:

1) Moving from starting point S to target point T in a straight line:

- If T is reached, the navigation is successfully completed.

- If an obstacle is encountered, a collision point with the obstacle is determined, and the algorithm proceeds to step 2.

2) The robot follows a boundary of the obstacle, taking steps of a specified size. If the robot reaches *T*, the navigation is successfully completed. Otherwise, the robot leaves the obstacle and heads towards *T* from a point of exit, referred as a *leaving point*. The selection of the leaving point depends on achieving the following conditions:

- The robot can move towards *T* without colliding with the obstacle.
- $Free_{dist} > 0$, and ($Curr_{dist} - Free_{dist} \leq 0$ or $Curr_{dist} - Free_{dist} \leq Best_{dist}$), where $Free_{dist}$ is a distance within the line of sight from a current location to a nearest obstacle in a direction of *T*, $Curr_{dist}$ is a distance from the current location to *T*, and $Best_{dist}$ is a difference between the distance from the collision point with the obstacle to *T* and the step size.

3) If the robot cannot find a leaving point or it returns to a position it was at a previous step, then *T* is unreachable.

3. System setup

The algorithm was implemented in Noetic Ninjemys version of ROS. The algorithm uses ROS topics to obtain measurements from sensors and wheel odometry [17], and to control the robot. Messages are transmitted using standard types predefined in ROS: *sensor_msgs::PointCloud*, *geometry_msgs::Twist*, *nav_msgs::Odometry*, and *geometry_msgs::Point*.

The algorithm was tested in the ROS-compatible Gazebo 11 simulator [18]. The Gazebo simulator creates realistic environments that allow conducting experiments with virtual robot models. The robot can be equipped with any of the standard sensors, and Gazebo has a support for third-party plug-ins. The Robot Visualization tool (RViz, [19]) is used to visualize a robot's position, measurements from sensors, and a path trajectory. The tool allows setting a target point as a 2D Nav Goal. A message is sent to an appropriate topic and accepted by a local planner.

The TurtleBot3 robot model (Fig. 1) with an open-source software was selected. The TurtleBot3 Burger [20] model was used, which has necessary sensors for navigating an environment under DistBug protocol. The robot's design allows making a transition from a 3D configuration space to a 2D one, which simplifies a path calculation [21].

4. Implementation details

DistBug algorithm was implemented in Python programming language, version 3.7. The implementation relies on ROS libraries and modules, including *rospy* (a Python client library for ROS), *geometry_msgs* (contains a type of messages for exchanging a position of the robot), *sensor_msgs* (contains types of laser measurement messages and motion commands), *gazebo_msgs* (contains types that determine a robot state), *tf* (manages transformations between coordinate systems), *nav_msgs* (contains the odometry data type), *std_srv* (contains packages for creating services in ROS) [22].

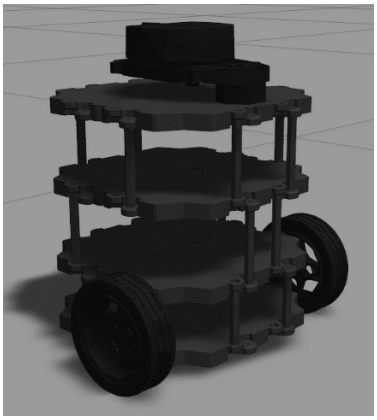

Fig. 1. Turtlebot3 Burger model in the Gazebo.

The algorithm's implementation contains three nodes. The first and second nodes regulate a robot motion directly to the target and the clockwise wall following motion, respectively. Each node implements a ROS server to activate and deactivate the mode. The third node launches a certain mode via a request to a corresponding server. The server choice depends on measurements received from the laser rangefinder. If the rangefinder indicates an occupied space ahead of the robot while the robot is moving towards the target, the robot changes the motion mode. Similarly, if a space in a direction of the target is free while the robot is moving along an obstacle boundary, the robot switches the mode. When the stop condition occurs, the algorithm turns off all nodes. To evaluate a performance the algorithm logs its operation time, all trajectory points (to calculate a path length), and a number of encountered obstacles during a path search.

To configure the algorithm, the following parameters were introduced: a timeout, a stopping threshold, a range of the rangefinder (i.e., radius of the robot's vision), and a robot's step size. The stopping threshold was empirically defined to determine a point where the robot stops when a certain coordinate value is reached. The timeout allows avoiding robot's stuck cases: despite a static environment, in some cases the TurtleBot 3 robot can get stuck at an obstacle boundary if one of its wheel touches an outer corner of the obstacle. This occurs due

to a rangefinder's placement that may cause a misregistration of occupied cells of the environment as free cells. If the robot gets stuck, a timer is activated and after the timeout expires, the algorithm stops with an error message. The rangefinder range and the step size are user provided parameters. The range should be less than the step size. A small step size causes a better precision of a computed path with regard to a theoretical path of an infinitely smell step and affects how often the obstacle detour check is be performed. With a large step size a robot may miss the target. The rangefinder measurements were divided into sectors by their direction. The value of a sector was determined by a minimum distance to an obstacle in the range.

5. Validation

To evaluate the implementation, tests were conducted in two types of environment: an environment with convex 3D obstacles of a cylindrical shape and a maze. The tests included reachable and unreachable goals. In the latter case the algorithm returned an error after reaching the stop state. The test results were recorded into a file that contained tracked metrics for each (start *S*, target *T*) pair.

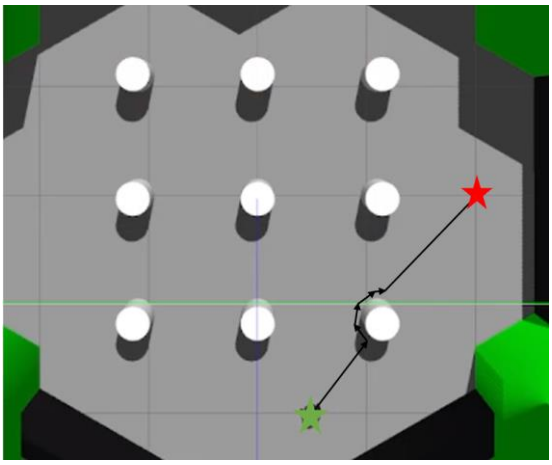

Fig. 2. A path of the DistBug in the simple map. The green star marks the start, the red star is the target.

We made 20 selection of (*S*,*T*) for the simple map (Fig. 2) and 15 pairs for the maze (Fig. 3). For each pair a number of runs was performed and they demonstrated approximately 97% success. The failures were cause by the problem mentioned in Section 4. Fig. 2 and Fig. 3 present the resulted paths of the robot using DistBug in the simple map and in the maze respectively. The virtual experiments were conducted using a PC with Ubuntu 20.0.4(LTS) operation system, 6.00 GB memory and AMD Ryzen 5 5500U processor with 2.10 GHz Radeon Graphics. While the environments for testing were constructed manually, a part of our ongoing work is to test the algorithm in automatically constructed environments for Gazebo, e.g., [23], [24], [25].

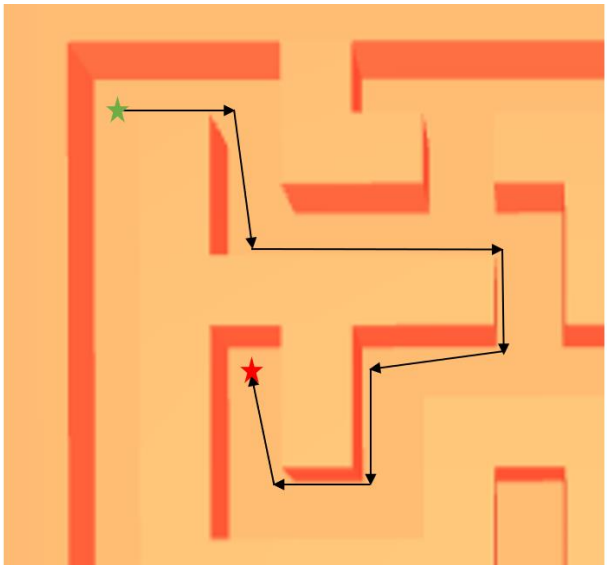

Fig. 3. A path of the DistBug in the maze

6. Conclusions

The paper presented an implementation of DistBug algorithm in the ROS Noetic environment. The evaluation was performed in two different types of environment and with a set of different target points. After reaching the stop condition, the control node logged the metrics, which included a flag of arrival to the target, a trajectory of the robot, a total time of the algorithm execution, and a number of encountered on the way obstacles. The tests demonstrated 97% of success. The implemented method can be adapted for other robot models and environments by empirically adjusting the timeout and the stopping threshold, while setting corresponding to the robot sensor range and the step size.

Acknowledgements

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

References

1. A. Zakiev, R. Lavrenov, E. Magid and V. Indelman, Path planning for Indoor Partially Unknown Environment Exploration and Mapping. ICAROB 2018: Proceedings of the 2018 International Conference on Artificial Life and Robotics, 2018, p. 399-402.
2. B. K. Patle et al, A review: On path planning strategies for navigation of mobile robot. Defence Technology, vol. 15, 2019, pp. 582-606.
3. E. Magid, R. Lavrenov and A. Khasianov, Modified spline-based path planning for autonomous ground vehicle. Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2017), vol. 2, 2017, p. 132-141.
4. H. Zhang, W. Lin and A. Chen, Path planning for the mobile robot: A review. Symmetry, vol. 10, 2018, pp. 450.

5. V. J. Lumelsky and T. Skewis, Incorporating range sensing in the robot navigation function. IEEE transactions on Systems, Man, and Cybernetics, vol. 20, 1990, pp. 1058-1069.
6. A. Sankaranarayanan and M. Vidyasagar, A new path planning algorithm for moving a point object amidst unknown obstacles in a plane. IEEE International Conference on Robotics and Automation, IEEE, 1990, pp. 1930-1936.
7. V. J. Lumelsky and A. A. Stepanov, Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica, vol. 2, 1987, pp. 403-430.
8. E. Magid and E. Rivlin, CAUTIOUSBUG: A competitive algorithm for sensory-based robot navigation. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), IEEE, vol. 3, 2004, pp. 2757-2762.
9. H. Noborio, Several path-planning algorithms of a mobile robot for an uncertain workspace and their evaluation. Proc. of the IEEE Int. Work. Intel. Motion Control, Istanbul, Turkey, vol. 1, 1990, pp. 289-294.
10. A. Yufka and O. Parlaktuna, Performance comparison of bug algorithms for mobile robots. Proceedings of the 5th international advanced technologies symposium, Karabuk, Turkey, 2009, pp. 13-15.
11. Y. Horiuchi and H. Noborio, Evaluation of path length made in sensor-based path-planning with the alternative following. Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164), IEEE, vol. 2, 2001, pp. 1728-1735.
12. I. Kamon and E. Rivlin, Sensory-based motion planning with global proofs. IEEE transactions on Robotics and Automation, vol. 13, 1997, pp. 814-822.
13. A. Apurin, B. Abbyasov, E.A.Martínez-García and E. Magid, Comparison of ROS Local Planners for a Holonomic Robot in Gazebo Simulator. International Conference on Interactive Collaborative Robotics, Cham: Springer Nature Switzerland, 2023, pp. 116-126.
14. A. Dobrokvashina, R. Lavrenov, E. Magid, Y. Bai and M. Svinin, How to Create a New Model of a Mobile Robot in ROS/Gazebo Environment: An Extended Tutorial. International Journal of Mechanical Engineering and Robotics Research, vol. 12, 2023, pp. 192-199.
15. R. Safin, R. Lavrenov, K.-H. Hsia, E. Maslak, N. Schiefermeier-Mach and E. Magid, Modelling a TurtleBot3 Based Delivery System for a Smart Hospital in Gazebo. The 15th Siberian Conference on Control and Communications (SIBCON 2021), 2021, № 9438875.
16. 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. 167-170.
17. P. Marin-Plaza et al, Global and local path planning study in a ROS-based research platform for autonomous vehicles, Journal of Advanced Transportation, 2018, pp. 1-10.
18. B. Abbyasov, K. Kononov, T. Tsoy, E. A. Martinez-Garcia and E. Magid, Experience in Efficient Real Office Environment Modelling in Gazebo: a Tutorial. International Conference on Artificial Life and Robotics (ICAROB 2022), 2022, p. 673-677.
19. H. Maghfiroh and H. P. Santoso, Online Navigation of Self-Balancing Robot using Gazebo and RVIZ. Journal of Robotics and Control (JRC), 2(5), 2021.
20. Fabio Ugalde Pereira, Pedro Medeiros de Assis 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, 2022, pp. 92-99.
21. N. James and T. Bräunl, Performance comparison of bug navigation algorithms, Journal of Intelligent and Robotic Systems, vol. 50, 2007, pp. 73-84.
22. A. Dobrokvashina, S. Sulaiman, T. Gamberov, K.-H. Hsia and E. Magid, New Features Implementation for Servosilla Engineer Model in Gazebo Simulator for ROS Noetic. International Conference on Artificial Life and Robotics (ICAROB 2023), 2023, pp. 154-157.
23. R. Lavrenov, A. Zakiev and E. Magid, Automatic mapping and filtering tool: From a sensor-based occupancy grid to a 3D Gazebo octomap. International Conference on Mechanical, System and Control Engineering (ICMSC), 2017, p. 190-195.
24. 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.
25. B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin and E. Magid, Automatic Tool for Gazebo World Construction: From a Grayscale Image to a 3D Solid Model. International Conference on Robotics and Automation (ICRA), 2020, p. 7226-7232.

Authors Introduction

Mr. Alexander Pak

He is a third-year bachelor student of Computer Science and Engineering at HSE University, Russia.

Mr. Alexander Eryomin

He received a BSc degree in Software Engineering from the School of Space and Information Technology at the Siberian Federal University in 2022. Currently, he is a second-year student of Master program in Intelligent Robotics at the Institute of Information Technology and Intelligent Systems at the Kazan Federal University, Russia.

Ms. Tatyana Tsoy

In 2012 she graduated from the University of Tsukuba. Since 2018 she has been a PhD student in Robotics at the Institute of Information Technology and Intelligent Systems of Kazan Federal University, Russia.