Real-Time Multi-obstacle Detection and Tracking Using a Vision Sensor for Autonomous Vehicle

Sobers Francis, Sreenatha G. Anavatti, Matthew Garratt, and Hussein A. Abbass

Abstract In this paper, an effective approach for real-time multi-obstacle detection and tracking in the navigation module is discussed. To calculate a feasible path for an autonomous ground vehicle (AGV) from the start position to goal position, efficient Dstar lite global planner is added and adhered to ROS nav_core package. Later, the clustering of points based on distance from a laser scanner data is carried out to perform multi-obstacle detection and followed by tracking. Then, the clusters are categorised as static and dynamic obstacles from their location, orientation, speed and size of an individual cluster. Using this approach, dynamic obstacles' paths are estimated from their respective past positions. To predict the dynamic obstacle for the next five-time steps, linear extrapolation and line fitting are employed. The estimated obstacles' path data are published as a *PointCloud* ROS message, then it is subscribed by the *costmap* node of the ROS navigation package. The *costmap* automatically updates the obstacle map layer and rebuilds the 2D occupancy grid map with new information about obstacles. Then, the path planner replans the path using updated *costmap* to avoid obstacles in the dynamic environment. Finally, real-time experiments are conducted to validate the efficacy of this intelligent system.

Keywords Multi-obstacle tracking · Collision avoidance · Intelligent vehicles · ROS navigation

1 Introduction

AGVs are widely utilised for scientific, commercial, industrial, and military applications for different tasks such as exploration in hazardous environments and unknown areas, military surveillance and reconnaissance, search and rescue missions and industrial automation.

An AGV must be able to adequately sense its surroundings to operate in unknown environments and execute autonomous tasks in which vision sensors provide the

S. Francis (✉) · S. G. Anavatti · M. Garratt · H. A. Abbass
School of Engineering and IT, University of New South Wales, Canberra, Australia
e-mail: s.anavatti@adfa.edu.au

H. Sharma et al. (eds.), *Communication and Intelligent Systems*, Lecture Notes in Networks and Systems 204, https://doi.org/10.1007/978-981-16-1089-9_67

necessary information required for it to perceive and avoid any obstacles to accomplish autonomous path planning. Hence, the vision (perception) sensor becomes the key sensory device for perceiving the environment in intelligent mobile robots, and the perception objective depends on three basic system qualities, namely rapidity, compactness and robustness [1].

The basic operation of mobile robot navigation encompasses the robot's ability to act and react based on partial knowledge about its operating environment and vision sensor readings to reach its goal positions efficiently and reliably as possible [2]. Path planning in a cluttered environment involves identifying a trajectory in a partially known map and with a target location that will cause the mobile robot to reach the location when executed. Obstacle avoidance means modifying the mobile robot's path to avoid collisions using real-time sensor data. The problem in a dynamic environment is that in most real applications as future motions of moving objects are a priori unknown, and it is necessary to predict them based on observations of the obstacles' past and present states, so that the AGVs path can be re-planned in advance to avoid a collision in critical conditions. So, an approach is required to estimate future positions and orientations of moving obstacles [3].

Until recently, most motion prediction techniques [4] have been based on kinematic models that describe how the state (e.g. position and velocity) of an object evolves. The models that are used to improve prediction results are the hidden Markov stochastic models, the grey prediction, least-mean square-error, and Kalman filter. In this paper, numerical prediction approaches, such as linear extrapolation and linear fitting are used to predict future positions and orientations of a moving object in dynamic environments because they are simple and convenient. Light detection and ranging (LIDAR) [5] is widely utilised as a mobile robot's vision sensor in the detection and tracking of obstacles. The supervised machine learning technique is used to classify the detect objects with the help of a clustering algorithm for robust detection and tracking from the lidar data [6]. It requires more computation time than the conventional method during the training phase.

The robotic operating system (ROS) is an open-source, meta-operating robot framework, which was originally developed in 2007 by the Standford Artificial Intelligence Institute and is a set of software frameworks for robotics engineering [7]. In our work, ROS is adopted for implementing the navigation planner, tracking obstacles from laser data, and controlling the AGV. Our main contributions of this paper can be summarised as follows:

1. A simple but effective way of detecting and estimating the dynamic obstacles as clusters with the help of range information from a single laser sensor is developed, by which a mobile robot can navigate the large-scale cluttered map with ease.
2. The estimated obstacles' path data are projected into the *costmap* of the ROS navigation package automatically which helps to update the 2D occupancy grid map regularly. Using this, the path planner replans the path, if needed, without any delay for each cycle.
3. Finally, the effectiveness of the proposed obstacle detection and avoidance approach has been demonstrated by real-time experiments.

Fig. 1 Jackal with laser range finder [autonomous lab @ UNSW Canberra]

The paper is organised as follows: Section 2 discusses the custom ROS navigation stack configuration. The multi-obstacle detection and tracking approach is explained in Sect. 3 along with the obstacle avoidance strategy. In Sect. 4, real-time experiments are demonstrated to validate the performance of the approach. Lastly, Sect. 5 concludes the paper with some recommendations for future work.

1.1 System Overview

The Jackal AGV is a small, fast and entry-level field robotics research platform, which has an on-board computer, GPS and IMU fully integrated with ROS for out-of-the-box autonomous capability [8]. Figure 1 shows one of our SICK LMS 111 laser-rangefinder equipped Jackal AGVs from the UNSW Canberra autonomous system lab. The AGV has the following specifications:

- Processor: Intel i5-4570TE Dual Core, 2.7 GHz Processor with 8GB RAM.
- Sensors: LMS 111 Laser finder, in-built Odometer fused with IMU.
- OS: ROS Kinetic Kame in Ubuntu 16.04.

2 ROS Path Planner

Path planning of mobile robots relies greatly on known information about the immediate environment and the motion constraints of robot kinematics and dynamics. ROS provides a package to navigate from a start position to goal position while avoiding obstacles. The navigation stack [9] is a ROS package that subscribes to

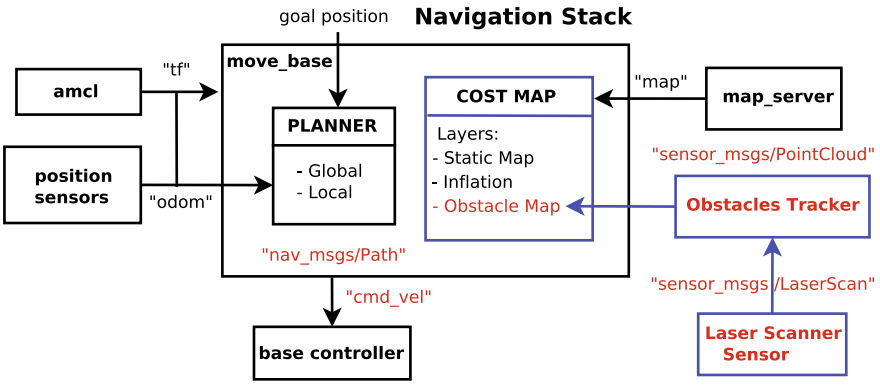


Fig. 2 ROS navigation stack with obstacle tracker

different ROS topics from odometry (robot position and orientation), sensor sources (like Laser scanner ("scan")) and a goal position ("*move_base_simple/goal*") and publishes safe velocity commands ("*cmd_vel*") to a mobile base controller. Then, the mobile base controller converts the "cmd_vel" topic into motor commands to send to a mobile base.

Adaptive Monte Carlo localization (amcl) is a package that deals with mobile robot localisation in which the position and orientation data representing the robot's pose [10]. Figure 2 illustrates an overview of the navigation stack configuration used for our approach. The occupancy grid path planning method in ROS divides the operating area into cells/map pixels and assigns them as occupied or empty. The path planner finds a feasible path that avoids any of the occupied cells.

An efficient Dstar lite path planner [11] is added and adhered to the *nav_core* :: *BaseGlobalPlanner* C++ interface defined in *nav_core* package [12]. A new path planner is added as a plugin to ROS, so that the AGV determines path sequence to move from starting position to destination in the navigation *move_base* package. The navigation stack uses two different *costmap* (local and global *costmap*) to store information about the obstacles from the static map and obstacles. Global *costmap* is for planning over the entire map, whereas local *costmap* is for local planning and obstacle avoidance. Each ROS topic can be subscribed by the navigation stack cycles at the rate specified by the parameter which determines the frequency, in Hz, at which the topic will run its update loop.

3 Multi-obstacles Detection and Tracking

This section provides a detailed explanation about multi-obstacles tracking of dynamic obstacles. The present navigation stack can only subscribe sensor data published using ROS message type, *sensor_msgs/LaserScan* and *sensor_msgs/Point*

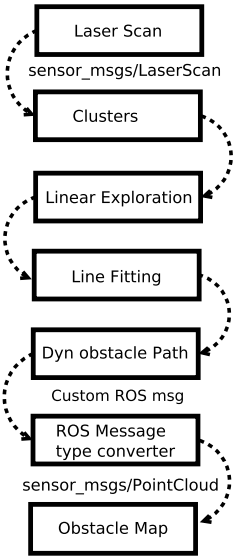


Fig. 3 Flowchart: raw laser data to multi-obstacle information

Cloud. The laser sensor is used to provide information to the navigation stack about the environment. Flowchart (Fig. 3) depicts the step-by-step approach in sequential order to detect and track the obstacles.

The location of the obstacles with respect to the scanner on the robot can be plotted in *xy* coordinates Eq. (1).

$$\begin{aligned} x\_obs &= \text{range}[i] * \cos(\text{angle\_min} + i * \text{angle\_increment}) \\ y\_obs &= \text{range}[i] * \sin(\text{angle\_min} + i * \text{angle\_increment}) \end{aligned} \tag{1}$$

where i is the total number of measurement steps varies from zero to maximum values and range[i] is the distance measurement corresponding to measurement steps.

Initially, laser scan data is subscribed, and their successive time step data are compared to filter the obstacles if any. Followed by the clustering of points based on distance is carried out to perform multi-obstacle detection. Then, the clusters are grouped as static and dynamic obstacles from their location, orientation, speed and size of an individual cluster. Later, dynamic obstacles' paths are estimated from their respective past positions. Linear extrapolation is employed to estimate future values by observing the relationship between the available data values. The range values of a cluster at different time steps are sampled periodically, and these data are used to approximate the future values. Then, the series of cluster position points have fitted a line using line fitting. These estimated obstacles path data are published as a *PointCloud* ROS message, and then it is subscribed by the *costmap* node of the ROS navigation package. The *costmap* automatically updates the obstacle map layer and rebuilds the 2D occupancy grid map with new information about obstacles. Then, the path planner replans the path using updated *costmap* to avoid obstacles in the dynamic environment.

The *costmap* ROS package [13] offers a configurable structure that keeps the information about where the AGV should navigate in the form of an occupancy grid. Each cell in this costmap structure can be either free, occupied or unknown and has assigned to special cost values (one of 255 different cost values). The occupied cells in the *costmap* are allotted a *costmap_2d* :: *LETHAL_OBSTACLE* with cost value = 254, the unknown cells are assigned a *costmap_2d* :: *NO_INFORMATION* cost, and others are allotted a *costmap_2d* :: *FREE_SPACE* with cost value = 0. For each cycle, the *costmap* updates the map information at the rate of update_frequency (5 Hz).

3.1 Obstacle Avoidance Strategy

There is a need for a strategy to avoid obstacles, while the AGV is travelling to the goal position in a cluttered environment. In this paper, an obstacle avoidance strategy is followed as explained below. Laser sensor (LMS111) can provide data up to a maximum range of 20 m. In *LaserScan.msg*, LIDAR scans from the start angle which commences along the positive x-axis to the end angle in a counter-clockwise direction. To discard the unnecessary range data, laser scan data is filtered with a start angle of +90° and an end angle of −90° as illustrated in Fig. 4. Our strategy procures obstacle data once their range is *leq*10 m. When the obstacles enter a 10 m region, the algorithm starts to cluster the obstacles as static and dynamic obstacles and predicts their next trajectory points if it is a dynamic obstacle. Further, when obstacle arrives within a 5 m region, obstacle map layer of *costmap* updates the map about the new obstacles information. Finally, the planner replans the mobile robot's path if there is any chance of collision.

Figure 4 shows the laser scan image with two obstacles *Obs*1 and *Obs*2. Both *Obs*1 and *Obs*2 are clustered and tracked as their distances are below 10 m away from the laser. But the *costmap* updates only *Obs*1 information into the map layer that helps the planner to perform re-planning only if necessary. The estimated Obs_1 paths are updated in the map with the cost value assigned to 255 and projected into a *costmap* structure.

4 Experimental Results

To validate the performance of our approach, various experiments are carried out on Jackal's model in Gazebo and mainly in Jackal UGV with LMS111 laser finder as a vision sensor. Though the Jackal is mounted with other vision sensors such as 3D Velodyne, Bumblebee camera and IP camera, only the LMS111 sensor is utilised for obstacle avoidance and estimation. A map is created with gmapping ROS package using a laser camera and is used by the map server. The map of the operating environment is of 4000 × 4000 pixels with 0.05 m/cell in Fig. 5. During

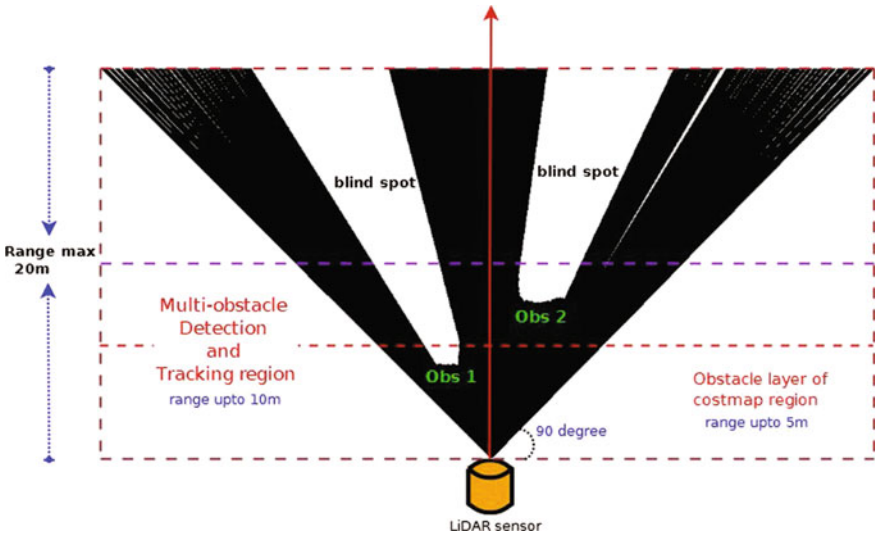
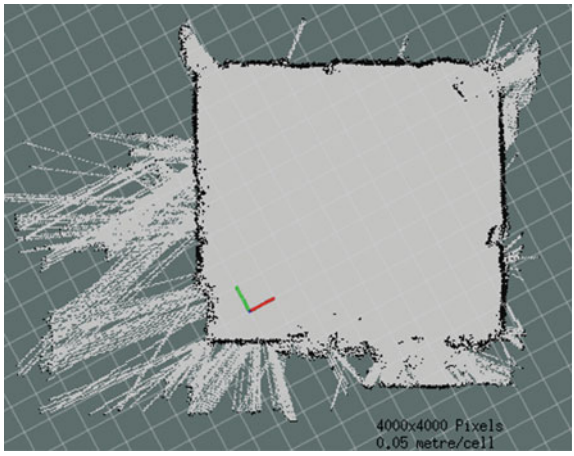


Fig. 4 Top view of laser scan with the obstacles

Fig. 5 Operating map in ROS rviz



the experiments, other Jackal robots, which can be controlled manually through the PS4 gaming controller, are used as the dynamic obstacles as shown in Fig. 7.

The laser range values from LMS 111 are utilised to detect the moving obstacles, and later, their position and orientation are estimated. These estimated trajectories are incorporated with the path planning algorithm, which helps to handle dynamic obstacles, so that AGV can calculate its feasible path without any collision. LIDAR LMS111 sensor has a maximum aperture angle of 270° with a scanning frequency of 25/50 Hz. The angular resolution of the scan measurement is of 0.25 or 0.50. In this work, LaserScan Message holds information about a single scan of range data as in Table 1.

Algorithm 1 Costmap updates map with the tracked obstacles

Require: LaserScan.msg provides start angle *angle_min*, end angle of the Laser scan, angle increment *angle_incre*, *range_min* and *range_max*. The *Off _angle* is the angle ignored at the sides. The *ns* is the number of samples per degree. The *range_count* is the number of range data per scan

Ensure: Clusters of Obstacles, Estimation of dynamic obstacles

$n \leftarrow 0$
for $i \leftarrow off\_ang * ns$ **to** $[range\_count - off\_ang * ns]$ **do**
  if $ranges[i] > range\_min$ **and** $ranges[i] < 10.0$ **then**
    $obstacle\_distances[n] = ranges[i]$
    $y \leftarrow ranges[i] * sin(angle\_min + i * angle\_incre)$
    $x \leftarrow ranges[i] * cos(angle\_min + i * angle\_incre)$
    $--> $ Cluster the obstacles
    $--> $ Estimate the obstacles position
    if $ranges[i] < 5.0$ **then**
      LETHAL_OBSTACLE: cost_lethal = 254
      $--> $ Insert obstacles information into costmap
      $--> $ Assign cell cost value = 254
      $--> $ Replan the planner if necessary
    end if
    $n++$
  end if
end for

return *Obstacles path*, *costmap updation*

Table 1 Single laser scan from LMS111

LaserScan.msg	
Message definition	Values
angle_min	−1.60570287704 rad
angle_max	1.60570287704 rad
angle_increment	0.00873
time_increment	2.77777780866e-05
scan_time	0.0199999
range_min	0.00999999 m
range_max	20.0 m

In Experiment I, the Jackal is stationary where a dynamic obstacle is moving towards the robot along with a few static obstacles. The consecutive laser scans are compared and then match each grid cells with cells from the previous iteration (based on distance). The location, size and speed of the obstacles as clusters are updated through observations by using a weighted average from cluster matches. ROS packages can also use a static map to help remove static obstacles and also to find out the moving obstacles. Once the obstacle enters the *costmap* region (<

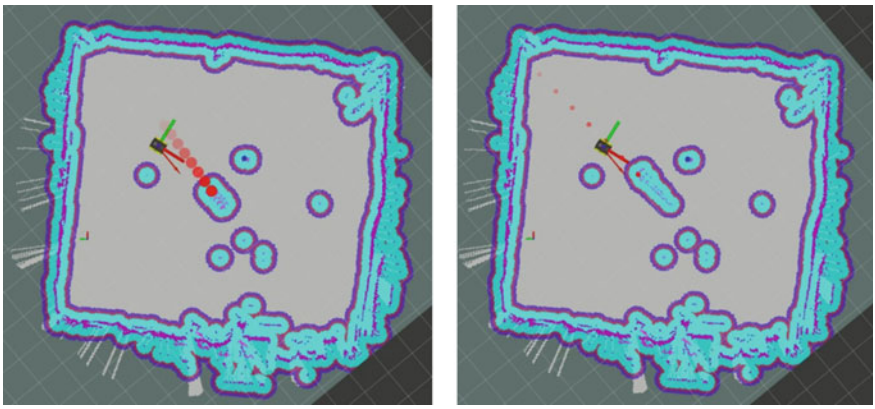


Fig. 6 Experiment I: detection and estimation of trajectories of the dynamic obstacle in ROS 3D visualiser

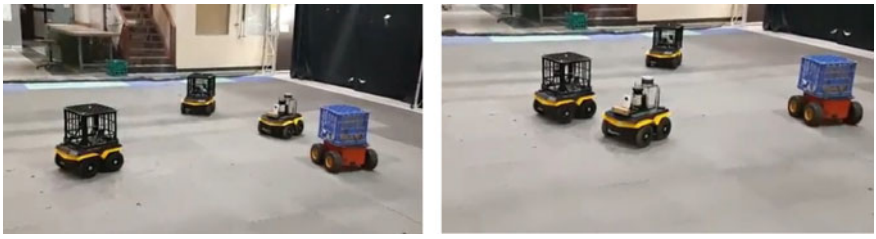


Fig. 7 Real-time path planning with obstacles

5 m) as proposed, its path is predicted and *costmap* updated. Figure 6 shows the static obstacles as blue circle clusters, dynamic obstacles with the predicted future trajectories as red circles (Fig. 7).

For Experiment II, Jackal has to perform a safe navigation task in a partially known map, and during its traverse towards the goal location, it simultaneously detect, estimate and avoid the obstacles and replan its path if necessary. Once the goal location is selected in the ROS rviz window inside the map, Jackal finds an initial feasible path to reach the goal. The readings from the laser scan on the Jackal help to locate the obstacles and provides information about the cluster whether static or dynamic. The dynamic cluster points are estimated from their past and present locations with the employed linear extrapolation and line fitting approach. Once the obstacles enter the 5-m region, the obstacles' trajectories are projected into the obstacle layer of the ROS *costmap* package. Eventually, Jackal has successfully reached the goal position by avoiding the obstacles, and the snapshots of the experiment are shown in Fig. 8.

The experimental results show that the proposed approach can help Jackal to detect and avoid both static and dynamic obstacles independently while performing navigation.

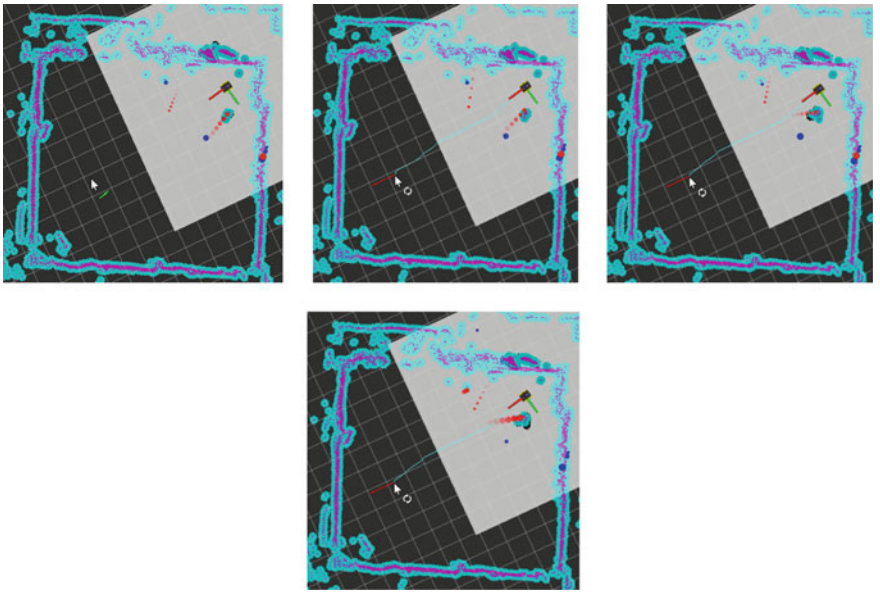


Fig. 8 Real-time path planning with static and dynamic obstacles

5 Conclusion

In this article, an approach for detecting and tracking obstacles is presented. The purpose of obstacle detection and tracking is to obtain the movement state of the moving obstacle, to predict the possible state and trajectory of the dynamic obstacle, which is of great significance to path planning of autonomous ground vehicles.

For future work, more experiments are planned with more complex scenarios to test and validate our approach. The performance of our approach needs to be verified with different scenarios such as when the AGV turns too fast when the surrounding has obstacles with various shapes and sizes and wider laser angles.

Acknowledgement This material is based upon work supported by the U.S. Army Ground Vehicle Systems Centre and International Technology Centre-Pacific under Contract No. FA5209-18-P-0140. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Army.

References

1. Patnaik S (2007) Robot cognition and navigation—an experiment with mobile robots. In: Cognitive technologies
2. Siegwart R, Nourbakhsh IR, Scaramuzza D (2011) Introduction to autonomous mobile robots, 2nd edn. The MIT Press, Cambridge

3. Volos CK, Jahanshahi H, Sari NN (2020) Recent advances in robot path planning algorithms: a review of theory and experiment. Robotics research and technology series. Nova Science Publishers, Incorporated
4. Jiang R, Tian X, Xie L, Chen Y (2008) A robot collision avoidance scheme based on the moving obstacle motion prediction. In: 2008 ISECS international colloquium on computing, communication, control, and management, vol 2, pp 341–345
5. Xie D, Xu Y, Wang R (2019) Obstacle detection and tracking method for autonomous vehicle based on three-dimensional LiDAR. Int J Adv Robot Syst 16:172988141983158
6. Karakaya S, Yasar Ocak H, Küçükyıldız G (2015) A bug-based local path planning method for static and dynamic environments*
7. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng A (2009) ROS: an open-source robot operating system, vol 3
8. Jackal: Unmanned ground vehicle. https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/
9. Fabro J, Guimarães R, Oliveira A, Becker T, Brenner V (2016) ROS navigation: concepts and tutorial 625:121–160
10. Zheng K (2017) ROS navigation tuning guide. 06
11. Francis S, Anavatti SG, Garratt M (2018) Real-time path planning module for autonomous vehicles in cluttered environment using a 3D camera. Int J Veh Auton Syst 14:40
12. Li Y, Shi C (2018) Localization and navigation for indoor mobile robot based on ROS. In: 2018 Chinese automation congress (CAC), pp 1135–1139
13. ROS: Costmap guide. http://wiki.ros.org/costmap_2d