Artificial Life and Robotics (2021) 26:10–20
https://doi.org/10.1007/s10015-020-00617-3

ORIGINAL ARTICLE

Navigation of a mobile robot in a dynamic environment using a point cloud map

Xixun Wang[1] · Yoshiki Mizukami[2] · Makoto Tada[1] · Fumitoshi Matsuno[1]

Received: 17 December 2019 / Accepted: 9 June 2020 / Published online: 26 July 2020

Abstract
In this paper, we consider autonomous navigation of a wheeled mobile robot in a dynamic environment using a 3D point cloud map. We consider four kinds of 2D maps: static global map, dynamic global map, global cost map, and local cost map; to plan a feasible path of the robot to adapt to a dynamic environment. We consider a mobile robot for plant patrolling in a 3D environment with plane slopes but not rough terrain for which a 2D environment map suffices. We propose 2D static global map for robot navigation by projecting prior measured 3D point cloud map data on a horizontal plane with considering the climbing ability of the robot. We also build a 2D dynamic global map by projecting a real-time 3D point cloud on the 2D static global map by SLAM. Accumulated errors of SLAM can be canceled using some landmarks placed in the environment. A global planner calculates an optimal global path that minimizes the distance from an initial robot pose (position and orientation) to a goal pose (position and orientation) by A* algorithm based on the global cost map which is built from the dynamic global map. However, this process should take much time. To avoid moving obstacles, the TEB (Timed Elastic Band) local planner is used to calculate an optimal local path based on a local cost map which is given by a real-time local 3D point cloud. To demonstrate the effectiveness of the proposed system, experiments were carried out. In the experiment, we use an AR card as a landmark for simplification of implementation. We prove that the robot can navigate in a dynamic environment and accumulated errors can be canceled by the AR cards placed in the environment as landmarks.

Keywords Mobile robot · Point cloud map · SLAM · Path planning · Dynamic environment

1 Introduction

Autonomous mobile robots are expected to be utilized for daily monitoring/inspection of plants and emergency response to an accident of plants to collect informations. Currently, 3D environment point cloud maps of public space and an engineering plant have been gaining much attention. Decommissioning of a nuclear power plant reactor at Fukushima is one of the most challenging applications of the robots. A 3D environment map is useful to an autonomous mobile robot in an indoor environment where GPS is not available. In this paper, motivated by the above applications, we consider a navigation system of an autonomous mobile robot for patrolling plants. Unlike the robots which are customized for usual living rooms or office environments[1], robots for plant patrolling move in a 3D environment with plane slopes but not rough terrain. Path planning using a 3D point cloud map[2] is not suitable for real-time implementations, because a long calculation time is needed for real-time re-planning in large-scale dynamic environments. For the application of navigation of an autonomous mobile robot to the patrol of the plant environment with slopes, we produce a 2D environment map by two-dimensionalizing a given 3D environment map.

This work was presented in part at the 24th International Symposium on Artificial Life and Robotics (Beppu, Oita, January 23-25, 2019).

✉ Xixun Wang
ojijin93@gmail.com

Yoshiki Mizukami
mizu@yamaguchi-u.ac.jp

Makoto Tada
tada@wbps.jp

Fumitoshi Matsuno
matsuno@me.kyoto-u.ac.jp

1 Kyoto University, Nishikyo Kyoto, Japan

2 Yamaguchi University, Yamaguchi, Japan

Several approaches for a ground robot to navigate in non-flat environments have been reported. Normally, a prior map or a real-time built map is used to localize the robot itself. And a dynamic map is built to avoid the prior unknown obstacles[3–5]. For an example, BigDog[5] is navigated autonomously in an unstructured forest environment by localizing itself using VO (visual odometry), building a local cost map that includes the obstacle's position information using LiDAR and calculating a path to follow with obstacle avoidance. However, in these studies, there are two limitations. First, it is hard to apply the navigation method to a large-scale environment because of the accumulated errors of the real-time built map. Second, it is hard to avoid moving obstacles, because a long calculation time is needed for updating the dynamic map.

To solve these limitations, we introduce four maps: static global map, dynamic global map, global cost map (GCM), and local cost map (LCM). In[6], the initialization method of the position of a robot by matching a local map from SLAM and a static map is proposed to solve the first limitation by navigating the robot based on the static map. The localization with the static map alleviates the accumulated errors. However, it is difficult to apply this method to a dynamic environment. In our approach, we first obtain a 2D static global map from a prior given 3D point cloud map, and then, we update the static global map to a dynamic global map using SLAM to reflect the changes in the dynamic environment[7]. However, to avoid increasing the calculation time of Pose Graph SLAM[8] (used in[7]), we select Fast-SLAM[10] for patrolling in the large and dynamic environments. To solve the second limitation, we utilized the global and local path planning[9]. The global planner generates the shortest path on a global cost map which is updated from the dynamic global map. The local planner generates a local path based on a local cost map to avoid the moving obstacles and consider urgent changes in a dynamic environment.

Localization and mapping are important technologies for an autonomous mobile robot. Since 3D SLAM methods spend too much time to update the map and we consider the flat environment with slopes, we employ 2D SLAM methods. At the start of processing, we build the 2D static global map from a prior measured 3D point cloud map. During the processing, we build 2D point cloud and retain as much information as possible from a real-time measured 3D point cloud, and then localize the robot and build a 2D map. There are three popular 2D SLAM methods: AMCL + mapping, Gmapping, and Hector-SLAM in ROS[11]. AMCL can localize on a static map, but it cannot edit the map or build a new map. Gmapping and Hector-SLAM can perform localization and build a new map, as well, but a prior given static global map cannot be used as an initial map. For our task, although a static global map is given, it should be modified, because the environment is dynamically changing

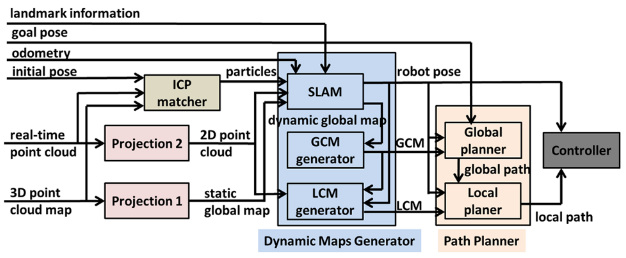


Fig. 1 Structure of the navigation system

due to moving obstacles. We use a 2D probability grid map SLAM[12] to build a dynamic grid map to compensate for the current difference from the static map. We also modified the Rao–Blackwellized Particle Filters[13] in[12] to fuse other positioning information from landmark and IMU. The robot pose (described by particles) is sampled using the odometry. The particles are updated when the robot finds a landmark using[13]. Besides, we also update the importance weight of particles using IMU information which is not used in[12, 13]. The detail of using IMU information is explained in Sect. 4.1.

In the case of a dynamic environment with moving and suddenly appeared objects, it takes much time of generating a global map and a global path. From the point of view of computational time, we carry out a general idea of a fusion of global and local path plannings[9]. Based on the dynamic global map, we build the GCM. The LCM is built frequently based on GCM and the 2D point cloud in the local region. Similarly, we build a rough global path at a low frequency based on GCM to get a path to the final goal and then build a local path at a high frequency based on a local goal. We use A* algorithm[14] for the global planner and TEB[15] for the local planner. For local motion planning, both behavior-based algorithm[16] (such as fuzzy logic[17] or neural network[18]) and the model-based algorithm have a possibility that the robot falls into a deadlock. The deadlock means that the global planner cannot find a feasible solution to go to the goal, initial position, or any exploration goals from the current robot pose. To solve it, we design a recovery behavior of a backtracking path in which the robot backtracks until the robot finds another path for leaving the deadlock area.

2 System structure

A block diagram of the whole system is shown in Fig. 1. At the core of the system, four 2D maps are built to describe the dynamical environment as explained in Sect. 3.1. In the beginning, the static global map is built from a prior measured 3D point cloud map, which is explained in Sect. 3.2. The projection of real-time 3D points on the local ground as

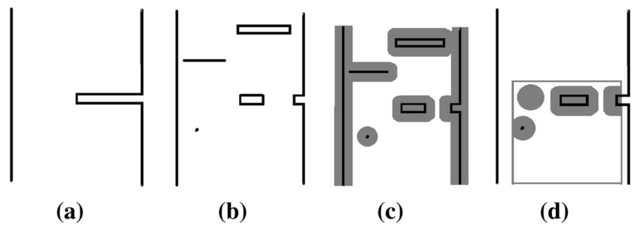


Fig. 2 Four grid maps **a** static global map, **b** dynamic global map, **c** global cost map, and **d** local cost map: the black line represents the object. The gray area represents the area where the robot can not move in

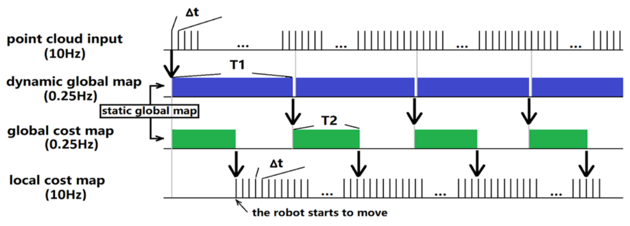


Fig. 3 Time-scale of updating maps: $T_1, T_2, \Delta t$ are parameters of time which satisfy $T_1 > T_2 >> \Delta t$

2D point cloud data is explained in Sect. 3.3. For initialization, a rough estimate of the initial pose (position and orientation) of the robot in a form of probabilistic distribution is given. The robot then obtains its precise pose by matching a real-time 3D point cloud and the prior measured 3D point cloud using ICP (Iterative Closest Point). The details are explained in Sect. 4.1.

In the dynamic map generator, SLAM is used to calculate the current pose of the robot and the dynamic global map is built based on a static global map, odometry, and 2D point cloud data. The details are explained in Sect. 3.4. The dynamic maps generator also builds a global cost map (GCM) and local cost map (LCM) explained in Sect. 3.5. Then, the path planner calculates the optimal path from the current robot pose to the desired goal pose based on GCM and LCM, the details of which are explained in Sects. 4.2 4.3 and 4.4. Finally, the path following controller generates a velocity command to control the robot as explained in Sect. 4.5.

3 Maps for dynamic environment

3.1 Four maps

We assume that a mobile robot moves on an environment consists of slopes and is almost flat. We consider that a 2D map is sufficient for the robot navigation in such an environment. SLAM, path planning, and control are carried out based on the 2D map. Four examples of four maps are shown in Fig. 2. Figure 2a shows a prior given static global map. Figure 2b shows a dynamic global map. It is updated based on current obstacle information. Some new obstacles are added and parts of the disappeared obstacles are erased. Figure 2c is a global cost map which is built based on the dynamic global map with consideration of the robot's size and exclusion of points corresponding to the bar whose height is bigger than the robot height. Figure 2d is a local cost map which is cut out from the global cost map and also considers the current changes of the dynamic environment nearby the robot's current pose.

The involved time-scale for generating the maps is shown in Fig. 3. 2D point cloud data are built from a real-time 3D point cloud every Δt s. It takes T_1 s to create a dynamic global map based on the 2D static global map and the current 2D point cloud data. The initial dynamic global map is set as a static global map. By cutting out a part of the built global cost map around the current robot position which is updated every T_2 s and adding a real-time 2D point cloud data with a sampling time of $1/\Delta t$ Hz, a local cost map is built every Δt s. Δt, T_1, and T_2 are the parameters, which satisfy $T_1 > T_2 >> \Delta t$, depending on the map settings and PC performance.

3.2 Conversion of 3D point cloud map into 2D static global map

We build the static global map from a 3D point cloud map off-line with considering the robot climbing ability. We assume that the ground planes are flat surfaces and a prior measured 3D point cloud map is given. We detect the points of ground planes and extract the points of obstacles included slopes whose elevation angles are bigger than the climbable angle of the robot. The extracted points corresponding to the obstacles are projected on a plane of a grid map. Then, we build the static global map.

The procedure to obtain the 2D static global map from the 3D point cloud off-line is as follows. First, using robust segmentation[19], the 3D point cloud is segmented into several point groups and a plane is extracted from each point group. The normal vector of each plane is calculated and if the angle difference between the normal vector of the plane and the gravity vector is smaller than the maximum slope angle of robot climbing ability γ, the points of the planes are regarded as the ground plane points and are not projected on the plane of a temporary 2D static global map, as shown in Fig. 4a. Second, points which are between a ground plane and the height of the robot are considered as obstacle points and projected on the plane of the temporary 2D static global map, as shown in Fig. 4b. Finally, the 2D position (x, y) of all obstacle points is added to the temporary 2D static global

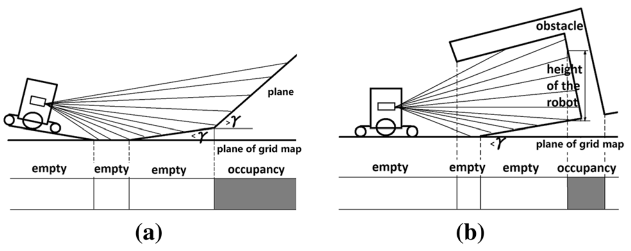


Fig. 4 Conversion of 3D point cloud into 2D static global map

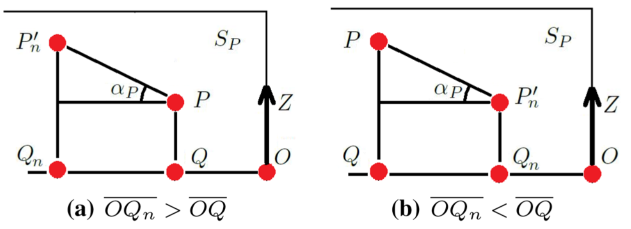


Fig. 6 Definition of angle of point

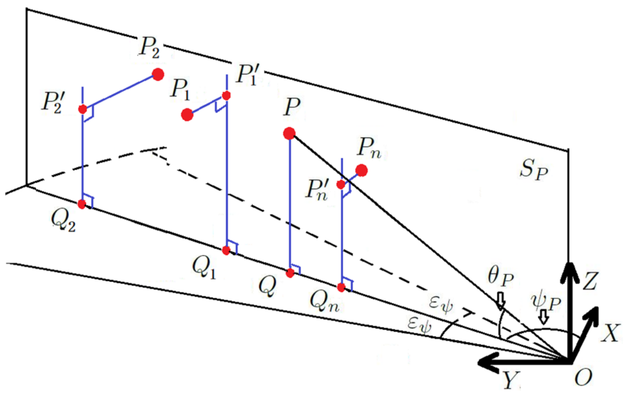


Fig. 5 Definitions of neighbor points

map and this processing is executed for all points, and then, we obtain the 2D static global map.

3.3 Conversion of real-time 3D point cloud into 2D point cloud to generate dynamic global map

To generate the dynamic global map, a 2D point cloud is extracted from a real-time measured 3D point cloud with consideration of the robot's climbing ability.

After receiving a 3D point cloud measured by a sensor (3D LiDAR or stereo camera), a local ground plane is extracted from the points[1]. The points on the local ground plane are regarded as ground points. Then, we calculate the angle α_P of the point P, as shown in Figs. 5 and 6. If the angle α_P is bigger than the maximum slope angle γ corresponding to the robot's climbing ability, we remove the point P from a set of the ground points. Then, we project the points, which do not belong to the set of the ground points and exist between the local ground plane and the height of the robot, into a plane of grid map to obtain 2D point cloud.

We now explain how to calculate the angle α_P of point P. Let us define a sensor coordinate frame $O-XYZ$, as shown in Fig. 5. The origin O is set as a sensor origin. The yaw angle ψ_P and pitch angle θ_P are calculated using the position data of the point P from the sensor. P is a function of ψ_P and θ_P, and let us donate $P(\psi_P, \theta_P)$. We consider a vertical plane S_P that the line OP is included. We define a neighbor point $P_i(\psi_{P_i}, \theta_{P_i})(i = 1, \dots, N)$, which satisfies the condition:

$$\begin{cases} \varepsilon'_\psi < |\psi_P - \psi_{P_i}| < \varepsilon_\psi \\ \varepsilon'_\theta < |\theta_P - \theta_{P_i}| < \varepsilon_\theta, \end{cases} \tag{1}$$

where ε'_ψ and ε'_θ are minimum optical resolutions of the sensor. ε_ψ and ε_θ are region limitations. Let us define the projection point P'_i on the plane S_P corresponding to P_i. And let Q and Q_i be the foot point of perpendicular line corresponding to P and P'_i, respectively. We define the nearest neighbor point P_n corresponding to P which attains minimum distance of $\overline{QQ_i}(i = 1, \dots, N)$. These definitions are shown in Fig. 5. The definition of the angle α_P of the point P is shown in Fig. 6. The angle α_P is defined as:

$$\alpha_P = \tan^{-1} \frac{|\overline{PQ} - \overline{P'_n Q_n}|}{\overline{QQ_n}}. \tag{2}$$

Using this angle α_P of the point P and the maximum slope angle γ, we can select a 2D point which is included on a dynamic global map.

3.4 Dynamic global map

The dynamic global map is built based on the static global map explained in Sect. 3.2, and 2D point cloud explained in Sect. 3.3 and odometry using SLAM[12], while considering the blind area of the sensor.

We explain the procedure to obtain the dynamic global map. In the beginning, the occupancy grid map of SLAM is built based on the static global map. The occupancy probability of a grid cell is set as 1 if the area is occupied by an obstacle on the static global map. The occupancy probability

[1] The method[19] can be also applicable to project the points into a local ground. It can segment multiple planes. However, it requires a higher density 3D point cloud and takes much time for processing. In this phase, we consider a local area and can assume that there is only one ground plane around the robot roughly. We can apply faster algorithm to segment one plane such as[20, 21].

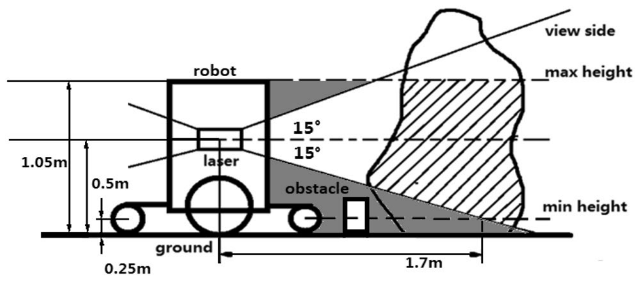


Fig. 7 Blind area of the robot

is set as 0 for empty areas of the static global map. The element of a grid cell in the unknown area is set as NaN (Not a Number). During the movement of the robot, the current 2D point cloud gives information of the new obstacles measured by the sensor in the dynamic environment. The occupancy grid map is renewed by updating the occupancy probability of each cell when it receives a 2D point cloud data. In[12], if a cell has a prior occupancy probability between 0 to 1, the cell updates its occupancy probability using Bayes' theorem. If the sensor measures data corresponding to the cell with an NaN element, the cells update its occupancy probability to its measured value. The dynamic global map is built from the occupancy grid map. Normally, a grid whose probability is larger than 0.5 is regarded as an obstacle area and a probability smaller than 0.5 as an empty area. Some obstacles might not be marked on the dynamic global map, because they have a probability lower than 0.5. Although the global planner might not be able to avoid those uncharted obstacles on the dynamic global map, the global planner explained in Sect. 4.2 is more likely to find a feasible global path. And the obstacles can still be marked on the local cost map and be further avoided using a local planner explained in Sect. 4.3.

However, one limitation of 2D SLAM compared to 3D mapping is that low height obstacles are removed by 2D SLAM. The sensor cannot detect, such as the low height obstacles that lie in the sensor's blind area, as shown in Fig. 7. To solve it, an obstacle in a part of the past dynamic global map corresponding to the sensor's blind area is not erased.

3.5 Cost maps

The global cost map is built based on the dynamic global map to plan a global optimal path to the goal. Around an obstacle on the dynamic global map, a prohibited area is built with respect to robot footprint size when the global cost map is created from the global map. The 3D shape of the robot is simplified into an inscribed circle of R_I radius and a circumscribed circle R_C radius in 2D projection for the purpose of computational efficiency. Then, we determine the cost of the cells of the global cost map using the two circles

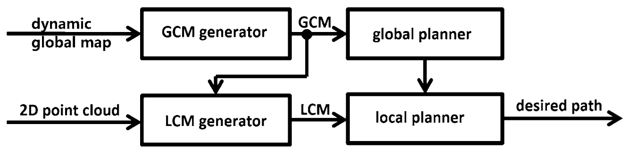


Fig. 8 The structure of path planning[9]

and the dynamic global map with a value between 0 and 1 or NaN. We define a set D as a set of all cells with value 1, which means the cells of obstacles, on the dynamic global map. Let us define D_i as a cell with its center P_i in the set D and S_j as a cell in the global cost map. For every position P_i of D_i, we draw a circle C_I of R_I radius and a circle C_C of R_C radius whose centers are P_i on the global cost map. If S_j is overlapped with the circle C_I, the cost of S_j is set as a maximum value 255 (the cost is defined as an integer value from 0 to 255). If S_j is outside the circle C_C, the cost of S_j is set to the minimum value 0. If there are some cells between the two circles, the cost of the cells increases linearly between the inscribed radius R_I and the circumscribed radius R_C. If S_j has several costs corresponding to different cells in D, the highest cost is set as the cost of the cell on the global cost map.

The local cost map in this study is a small size grid map around the robot. As shown in Fig. 2, the position of a moving obstacle is updated every $T1$ s on the dynamic global map from the input 2D point cloud data, and it also takes another $T2$ s to build the global cost map from the dynamic global map. It is hard to include the moving obstacle in the dynamic global map in real-time. Therefore, the robot uses the local cost map that is built every Δt s in a local area.

4 Navigation

In this section, we describe the localization of the robot pose, the global and local path planner algorithm shown in Fig. 8, and the path following control to generate velocity commands to lead the robot to the given local path.

4.1 Calculation of the robot pose

We have two ways, ICP matching and landmark matching, to generate the robot pose on the static global map.

For ICP matching, the real-time 3D point cloud is matched with the prior measured 3D point cloud map by ICP method. To avoid the incorrect result of ICP matching, we randomize the process by performing the ICP matching for each sampled particles, and then, the best ICP matching result is selected by updating the particle weight. The initial pose for the iteration is taken as position/orientation with a probability distribution of Gaussian. In the beginning, we initialize N particles by sampling the pose randomly using the given distribution. Then, after the robot moves and receives the first point cloud data, a local 3D point cloud is built. Based on a pose of each particle and odometry of the robot, the ICP searches a robot for each particle pose by minimizing the difference between the local 3D point cloud and the prior measured 3D point cloud map. We obtain an error value as a result of ICP matching, it is regarded as the particle weight. The current robot pose is given by the best particle which has the lowest weight. Finally, the 2D poses of particles are copied to SLAM and the inner map of SLAM is refreshed as the static global map.

For the landmark matching, a robot saves the extracted landmarks' features and poses in the database. In real-time processing, the robot uses a sensor to detect a landmark and carries out the matching of those landmarks' features. When the robot detects a landmark around itself, a relative pose between the landmark and the robot is calculated. Then, a current robot pose is refreshed based on the relative pose from the landmark absolute pose.

We use IMU to correct the estimation errors of the robot direction by updating the importance weight w of particles. The Rao–Blackwellized Particle Filter has N particles. Each particle stores a different gaussian distribution of robot pose (mean vector $s = [x_s, y_s, \psi_s]^T$ and a variance matrix Σ_s) and an importance weight w. The data of IMU are described as ψ_{imu}. Then, we update the importance weight as $w = w * N(\psi_{\mathrm{imu}}|\psi_s, \Sigma_s[3,3])$, where the function $N(\psi_{\mathrm{imu}}|\psi_s\ \Sigma_s[3,3])$ represents the probability density at the orientation ψ_{imu} using orientation distribution ψ_s and $\Sigma_s[3,3]$ ($\Sigma_s[3,3]$ represents the element at 3rd row and 3rd column of the matrix Σ_s).

4.2 Global planner

The global planner generates a global path based on a global cost map. If we do not have a feasible path by considering only empty grids with 0 value, an unknown grid with NaN element is considered as an empty grid and an optimal path is calculated. The shortest path from the current robot pose to the goal pose is calculated by A* algorithm without considering the robot's mobility. The global cost map is built from the dynamic global map explained in Sects. 3.4 and 3.5. When the dynamic global map is updated, the global planner recalculates a new desired path. Although A* has a concern of efficiency, it is applicable to our system, because the calculation of the global path by A* algorithm is much faster than the generation of the global cost map.

4.3 Local planner

The local planner is necessary to avoid moving obstacles and to compensate for the difference between a calculated dynamic global map at the last sampling time and the current environment. Because the size of the global map is much bigger than that of the local map, there is a big delay in constructing the global map (see Fig. 3). On the other hand, the local map is easily modified based on the real-time information. It is suitable to plan a local path to avoid moving obstacles based on the current local map. Besides, the local planner can consider the limitation of the mobility of the robot by introducing the corresponding cost function.

The local planner generates a local path based on a local cost map, explained in Sect. 3.5, that the robot should follow. We apply TEB local planner[15]. It simulates the robot motion and finds several feasible paths on the local map based on a cost function considering the current robot pose and the given global path. It computes the distance to the nearest obstacle, the error of the local path from the global path, error of the attitude of the robot from the desired one, velocity limit, and acceleration limit of the robot[15]. Figure 9 shows the simulation results for the generation of the global path on the local cost map in (a) and the local path in (b) in a dynamic environment. We can see that the local planner generates a feasible path to avoid both static and moving obstacles.

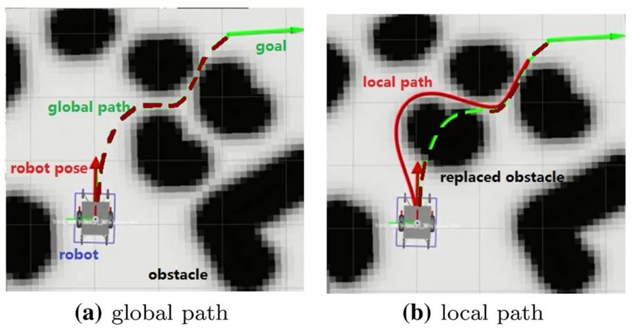


Fig. 9 Simulation results of generation of a feasible path to avoid moving obstacles by the local planner: an arrow represents an attitude of the robot. A broken line and a solid line are global path and local path, respectively. A black area represents the area where the robot cannot move in

4.4 Backtracking planner

The backtracking planner provides a recovery behavior when the planner cannot calculate any feasible path. When the environment is changed drastically, moving obstacles sometimes block the local path to the current goal. In this case, the local planner cannot find a feasible solution using the local map, then a backtracking path is built to move the robot backward. When the local planner falls into failure, the robot goes backward on its trajectory as the backtracking action. The recovery behavior is finished when the planner can find

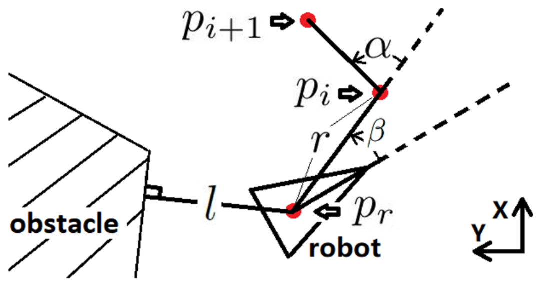


Fig. 10 Generation of a velocity command using a desired path: dots are the points on the desired path for a robot. Triangle is a footprint of the robot. A hatching area is an obstacle

a new feasible path due to the change of the local map of the dynamic environment.

4.5 Path following control

A path following control is designed as a point-to-point control in 2D map. In Fig. 10, the control purpose is to make the robot reach the current goal point p_i from the current robot point p_r. The logic to decide the current goal is as follows. The desired path based on the local map is given as a set of points. In the beginning, the nearest point on the desired path from the current robot position p_r is set as the current goal p_i. The next nearest point of p_i on the path is set as the next goal p_{i+1}. If the distance from p_r to p_i is smaller than 0.1 m, the next goal p_{i+1} is set as the current goal p_i in our experiment unless the p_i is the final point on the path.

In Fig. 10, r is the distance from the current robot position p_r to the current goal p_i, l is the distance from the robot to the nearest point of the obstacle, β is the angle between the robot moving direction and the direction from the robot to the current goal, α is the angle between $\overrightarrow{p_rp_i}$ (the direction from the robot to the current goal), and $\overrightarrow{p_ip_{i+1}}$ (the direction from current goal to the next goal). Let ω be the current angular velocity of the robot. The reference of the translational velocity v_{ref} and that of the angular velocity ω_{ref} for the robot are given as follows:

$$\begin{cases} v_{ref} = k_v(1 - 0.9\dfrac{k_p}{l + k_p})r - k_{\omega_t}\mathrm{abs}(\omega), & \text{if } v_{\mathrm{ref}} > 0 \\ v_{ref} = 0, & \text{if } v_{\mathrm{ref}} < 0 \end{cases} \quad (3)$$

$$\omega_{\mathrm{ref}} = k_\omega\beta + \frac{k_\alpha}{r + k_\alpha}\alpha, \quad (4)$$

where k_v, k_p, k_{ω_t}, k_ω, and k_α are positive coefficients.

The first term on the right side of the top of (3) has an effect to slow down the robot when it is close to the current goal or the obstacle. The second term has an effect to slow down the robot when it is rotating. The first term on the right-hand side of (4) leads the robot to the direction of the current goal p_i. The second term has an effect that the robot faces to the next goal p_{i+1} when it comes close to the current goal p_i.

If the next desired direction $\overrightarrow{p_ip_{i+1}}$ is opposite to the current desired direction $\overrightarrow{p_rp_i}$, α and r are modified as follows:

$$\begin{cases} r = -r \\ \beta = \beta + \pi, & \text{if } \overrightarrow{p_rp_i} \cdot \overrightarrow{p_ip_{i+1}} < 0 \\ \alpha = \alpha + \pi \end{cases} \quad (5)$$

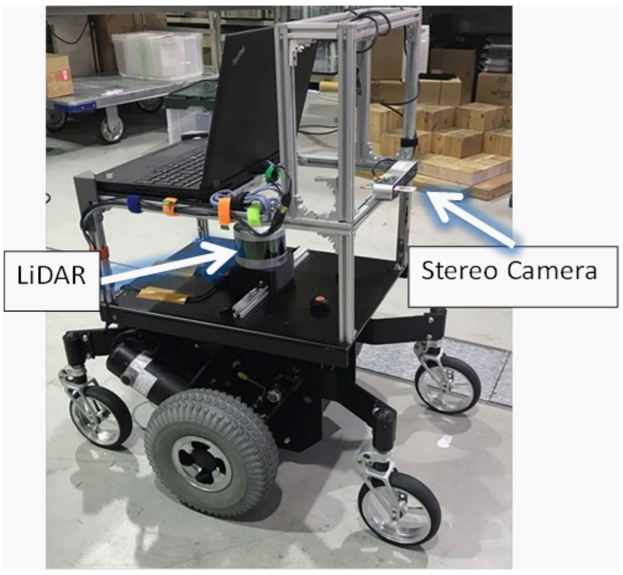


Fig. 11 A mobile robot used in our experiments

5 Experiment

5.1 Platform

Experiments in this paper are performed with a mobile wheeled robot, as shown in Fig. 11. It is a double-wheeled robot with a 70 kg payload and its maximum speed is 3 (m/s). The size of the robot is W600×D500×H1050 mm and the total weight is 40 kg. It can go up and down a hill whose slope angle is less than 15 (°) ($\beta = 15$ (°)). Each wheel has an encoder and we can measure the velocity. A 360 (°) LiDAR sensor (VLP-16) is mounted at the center of the mobile base. A stereo camera ZED is attached in front of the mobile base to detect landmarks. An IMU sensor (LPMS-CU2) is mounted on the robot to measure the attitude of the robot. The PC (Thinkpad P50 (Intel I7 2.7 GHz), 32 G)

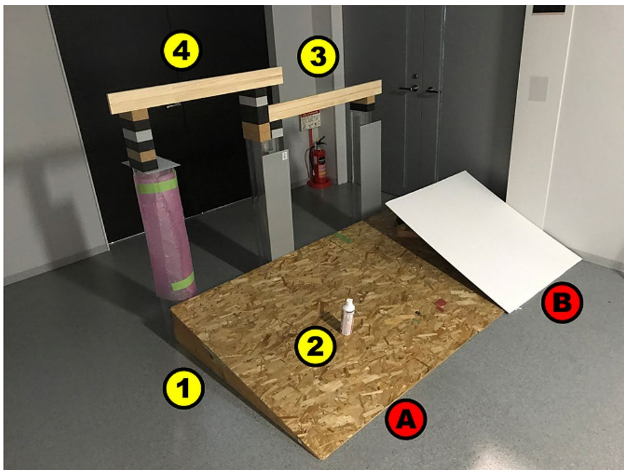


Fig. 12 Experiment field: Ⓐ Ⓑ are slopes. ①–④ are objects

is mounted on the robot. In Eqs.(3) and (4), we set $k_v = 3$, $k_p = 0.6$, $k_{\omega_t} = 0.4$, $k_\omega = 1.5$, and $k_\alpha = 0.25$.

5.2 Conversion of real-time 3D data into 2D

The purpose of the experiment is to demonstrate that the method introduced in Sect. 3.3 can recognize whether the points belong to obstacles or not in a 3D environment with two slopes.

The experiment field, shown in Fig. 12, consists of two slopes Ⓐ, Ⓑ on the ground and four objects ①–④. The robot can climb the slope Ⓐ with a slope angle of 11°. On the other hand, the robot cannot climb up on the steep slope Ⓑ whose angle is 23°, because the robot's climbing ability is 15°. The object ① is the side edge of the slope Ⓐ. The object ② is a low height obstacle on the gentle slope Ⓐ. The object ③ is a lower gate whose height is 85 cm. The object ④ is a higher gate whose height is 110 cm. The objects ①②③ are obstacle, while the bar of the object ④ is not an obstacle for the robot whose height is 105 cm.

The result is shown in Fig. 13. By comparing Fig. 13a and b, the ground and the gentle slope Ⓐ were removed, and the steep slope Ⓑ and objects ①–④ were preserved. In Fig. 13c, it can be observed that the objects ①-③ were correctly preserved, and the bar of object ④ was removed as an obstacle considering the height of the robot.

5.3 Positioning by landmarks

The purpose of the experiment is to demonstrate that the localization accuracy can be increased using landmark matching. We use AR cards as landmarks to initialize the robot pose and update the robot pose.

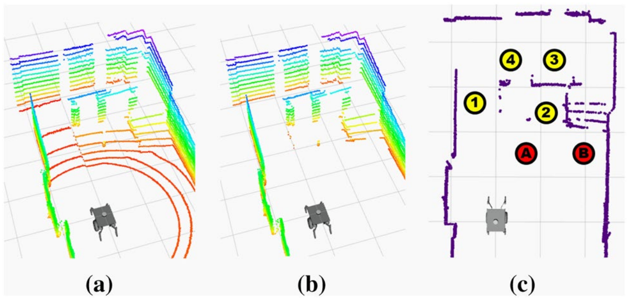


Fig. 13 Real-time 3D and 2D point cloud data: **a** 3D point cloud from a LiDAR, **b** 3D point cloud after removing the points corresponding to ground planes and slopes, and **c** 2D point cloud data by projecting the remaining 3D point cloud onto the local ground: Ⓐ Ⓑ are slopes. ①–④ are objects

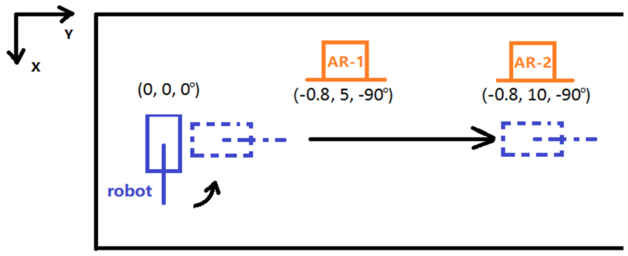


Fig. 14 Experimental setup for canceling of accumulated errors by detecting AR cards

Table 1 Experimental result of maximum error of each case

Case	X (cm)	Y (cm)	Yaw (°)
(1) No map, no AR cards	6	12	3.5
(2) No map, AR cards	2.5	2.7	2.8
(3) Map, no AR cards	2.5	3.1	2.7
(4) Map, AR cards	1.5	0.9	2.6

In the experiment, we compare localizing accuracy for four cases: (1) SLAM without a static global map and AR card; (2) SLAM without a static global map but with AR card; (3) SLAM with a static global map but without AR card; (4) SLAM with both static global map and AR card. In the beginning, we place the robot at the assigned position, facing to the wall, as shown in Fig. 14. The robot rotates 90 °, and then, it drives forward 10 m. We have two AR cards with an unique number in the environment as landmarks. The size of each AR card is 24×24 cm.

Experiments for the above four cases are carried out three times. The recorded maximum position errors are shown in Table 1. Using the landmark information, the errors for both with and without a static global map are decreased.

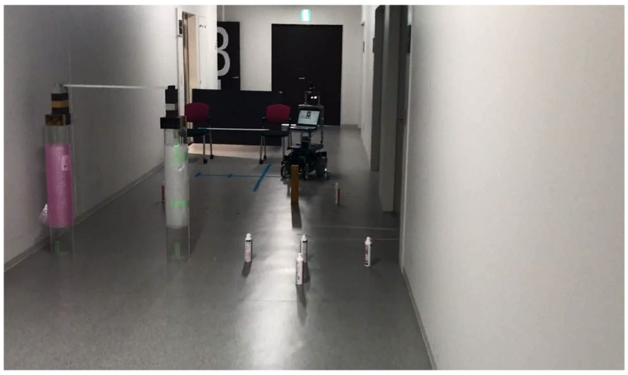


Fig. 15 Indoor environment field: multiple objects are prior unknown

5.4 Indoor experiment

The purpose of the experiments is to demonstrate that the proposed navigation method can avoid different kinds of obstacles, the SLAM can update the dynamic global map based on the static global map, and the robot can accomplish self-localization in a dynamic environment.

The experiments are carried out in an indoor environment of a building of Kyoto Univ. The robot moves from the corridor into a room of our laboratory. The experiment field is about 20 m ×15 m. The sizes of the static global map, dynamic global map, and global cost map are set as 200 m×200 m each with a resolution of 0.05 m for the plant patrolling objective. Since the indoor environment for the experiment is small and the SLAM does not generate obvious accumulated errors, the localization using landmarks is not needed. The size of the local cost map is 5 m×5 m with a resolution of 0.05 m. Considering the size of the maps (200 m×200 m), the parameters of the time-scale are set as $T_1 = 4$ s, $T_2 = 2$ s, and $\Delta t = 0.1$ s. An initial pose, a static global map and a goal pose are given. In the corridor, we placed four kinds of obstacles: small bottles, chairs, slender pipes that block the robot, and the slender pipes whose height is more than the height of the robot, as shown in Fig. 15. Inside the laboratory, the environment is drastically changed in comparison with the given static global map by placing tables, chairs, and other things. Since free space inside the room is narrow, the robot has to avoid the suddenly observed obstacles in the blind area using the local planner.

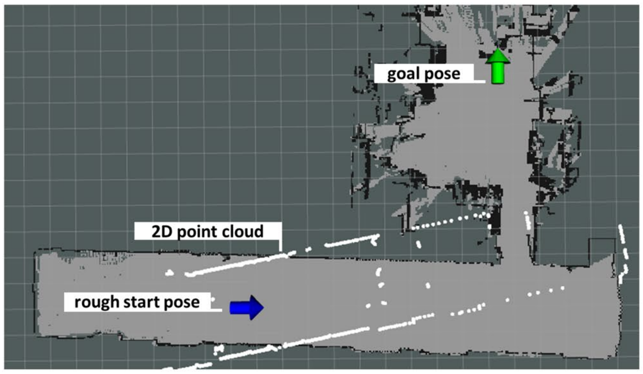
It is observed that the robot detected all obstacles in the dynamic environment and avoided all of them. In the experiments, the robot built a map by SLAM and could reach the goal successfully, as shown in Fig. 16. In Fig. 16, black spots mean obstacles. Figure 16a illustrates the prior given static global map, Fig. 16b shows a global path based on a dynamic global map corresponding to the initialization of the robot pose at the start point, and Fig. 16c shows the final dynamic global map built by SLAM when the robot reached the final goal. By comparing the prior given map shown in Fig. 16a and the final generated map shown in Fig. 16c, we can infer that the robot detected the changes of the environment, especially inside of the room and built a new map. Figure 16d shows a global path and a local path in the dramatically changed region. From this figure, we can find that the robot can avoid suddenly observed obstacles by the local planner in real time. Measured linear speed and travel length of the robot are shown in Fig. 17.

We measured the average calculation times. The calculation times of updating the global dynamic map, global cost map, and local cost map are 1.49 s ($< T_1$), 1.06 s ($< T_2$), and 0.061 s ($< \Delta t$), respectively. The calculation times of global path planner (A*) and local path planner (TEB) are 0.82 s and 0.056 s, respectively. It can be seen that the calculation times are smaller than the parameters of the time-scale (T_1, T_2, and Δt), which means that the parameters of the time-scale met the requests of the calculation time.

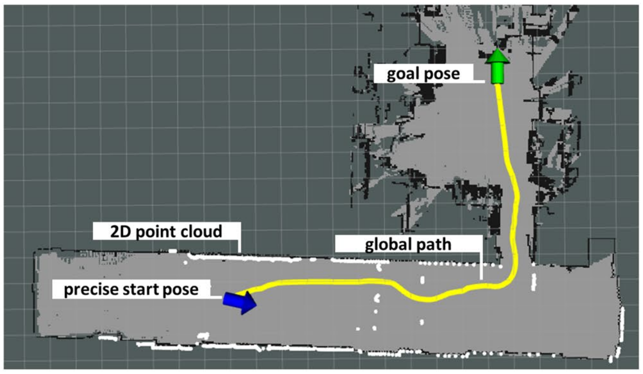
6 Conclusion

We developed a navigation system for an autonomous mobile robot based on static global 3D environment data. We proposed four grid maps: static global map, dynamic global map, global cost map, and local cost map. We built a 2D static global map from a prior given 3D point cloud map for the robot navigation. The global and local path planners are applied to obtain an optimal behavior of the robot in a dynamic environment based on the global and local cost maps. To demonstrate the effectiveness of the developed system, indoor experiments are carried out. The experimental results show that the robot detects the objects on the slope, builds a dynamic map based on a static map, and localizes itself even in the case that the environment is drastically changed. The error of localization decreases when the robot detects the landmarks.

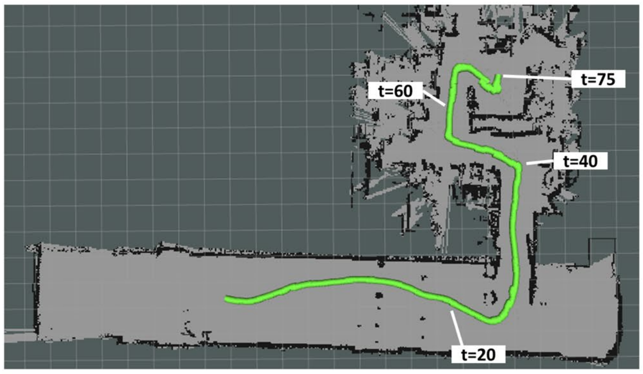
Although AR cards are used as the landmarks in our experiment, for application in the real environment, the landmarks can be replaced with other features which can be extracted from the prior 3D point cloud. And reconstruction of the real-time 3D environment using a real-time 3D point cloud from a sensor has not been accomplished yet. Finally, since we think that the difference of the travel distances between a slope of 15[deg] and flat ground is not so big, in our approach, we calculate the shortest path based on a 2D map. We do not consider the difference of the travel



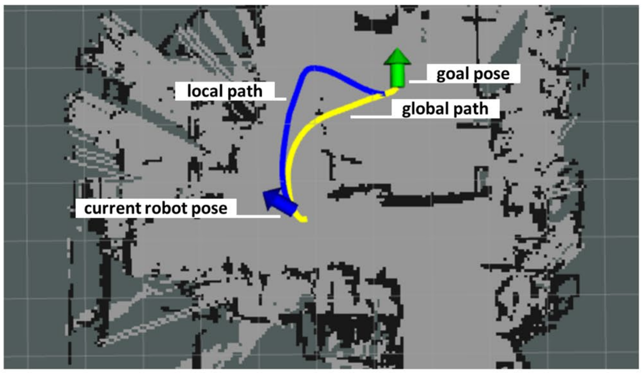
(a) Static global map and initial information



(b) Dynamic global map and global path



(c) Trajectory



(d) Global path and local path

Fig. 16 Result of navigating the robot and mapping prior unknown object on the dynamic global map. **a** The static global map and the given initial information: the blue arrow represents the rough initial robot pose given by an operator. The green arrow is the goal pose. The white line/dots represents the 2D point cloud. **b** The dynamic global map, the global path, and the precise initial robot pose using ICP in the beginning: the blue arrow represents the precise initial robot pose. The yellow line represents the global path. **c** The final dynamic global map and the robot trajectory that were generated by SLAM when the robot reached the goal: the green line represents the robot trajectory. **d** The global path and the local path: the yellow line represents the global path. The blue line represents the local path (color figure online)

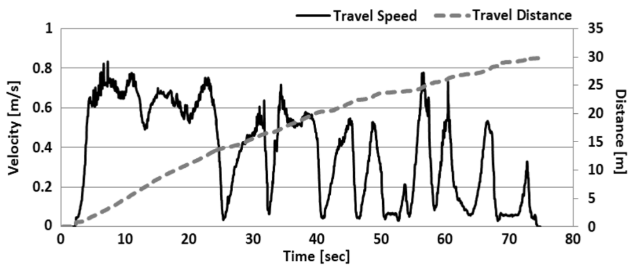


Fig. 17 The travel speed (measured by encoder) and travel distance (measured by SLAM) of the robot in the experiment

distance between 2D and 3D, and the calculated path might not be the shortest in 3D. One probable solution is to build another global grid map to describe the travel distance in 3D. These problems should be further investigated in the future.

Acknowledgements A part of this study was carried out as a collaborative research with BESTERRA Co., Ltd.

References

1. Eitan M, Eric B, Tully F, Brian G, Kurt K (2010) The office marathon: robust navigation in an indoor office environment. In: Proceedings in IEEE international conference on robotics and automation, pp 300–307
2. Ellips M, Golnaz H (2007) Robot path planning in 3D space using binary integer programming. J Comput Inf Eng 1(5):1255–1260
3. Andrew J. D, Nobuyuki K, (2001) 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In: Proceedings of IEEE international conference on computer vision and pattern recognition, pp 394–391
4. Konstantinos C, Ioannis K, Antonios G (2015) Thorough robot navigation based on SVM local planning. J Robot Auton Syst 70(1):166–180

5. David W, Matthew M, Kevin B, Andrew H, Alfred AR, Marc R (2010) Autonomous navigation for BigDog. In: Proceedings of IEEE international conference on robotics and automation, pp 4736–4741
6. Malcolm M, Martin M, Henrik A, Achim JL (2017) SLAM autocomplete: completing a robot map using an emergency map. In: Proceedings of IEEE international symposium on safety, security and rescue robotics, pp 35–40
7. John M, Michael K, Cesar C, Jose N, John JL (2013) Real-time 6-DOF multi-session visual SLAM over large-scale environments. J Robot Auton Syst 61(10):1144–1158
8. Aisha W.B, Michael K, Hordur J, John J.L (2012) Dynamic pose graph SLAM: long-term mapping in low dynamic environments. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems, pp 1871–1877
9. Pablo M, Ahmed H, David M, de la Arturo E (2018) Global and local path planning study in a ROS-based research platform for autonomous vehicles. J Adv Transp 2018:1–10
10. Michael M, Sebastian T, Daphne K, Ben W (2003) FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: Proceedings of the international conference on artificial intelligence, pp 1151–1156
11. Morgan Q, Brian G, Ken C, Josh F, Tully F, Jeremy L, Eric B, Rob W, Ng A (2009) ROS: an open-source robot operating system. In: Proceedings of IEEE international conference on robotics and automation, open-source software workshop
12. Giorgio G, Cyrill S, Wolfram B (2005) Improving grid-based slam with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In: Proceedings of IEEE international conference on robotics and automation, pp 2443–2448
13. Arnaud D, Nando de F, Kevin M, Stuart R (2000) Rao-Blackwellized partcile filtering for dynamic bayesian networks. In: Proceedongs of conference on uncertainty in artificial intelligence, pp 176–183
14. Kurt K (2000) A gradient method for realtime robot control. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems, pp 639–646
15. Rösmann C, Hoffmann F, Bertram T (2015) Planning of multiple robot trajectories in distinctive topologies. In: Proceedings of IEEE European conference on mobile robots, pp 1–6
16. Watanabe M, Onoguchi E, Kweon I, Kuno Y (1992) Architecture of behavior-based mobile robot in dynamic environment. In: Proceedings of IEEE international conference on robotics and automation, pp 2711–2718
17. Zhu A, Yang SX (2007) Neurofuzzy-based approach to mobile robot navigation in unknown environments. IEEE Trans Syst 37(4):610–621
18. Engedy I, Horvath G (2010) Artificial neural network based local motion planning of a wheeled mobile robot. In: Proceedings of IEEE international conference on computational intelligence and informatics, pp 213–218
19. Abdul N, David B, Geoff W (2016) Robust segmentation for large volumes of laser scanning three-dimensional point cloud data. IEEE Trans Geosci Remote Sens 54(8):4790–4805
20. Dimitris Z, Izzat I, Nikolaos P (2017) Fast segmentation of 3D point clouds: a paradigm on LiDAR data for autonomous vehicle applications. In: Proceedings of IEEE international conference on robotics and automation, pp 5067–5073
21. Ahmed H, Pablo M, David M, Arturo E, Jose M (2016) Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection. In: Proceedings of IEEE international conference on intelligent vehicles symposium, pp 104–109