Biomimetic Intelligence and Robotics 2 (2022) 100035

Contents lists available at ScienceDirect

Biomimetic Intelligence and Robotics

journal homepage: www.elsevier.com/locate/birob

Experimental evaluation of autonomous map-based Spot navigation in confined environments

Anton Koval [*], Samuel Karlsson, George Nikolakopoulos

Robotics & AI Team, Department of Computer, Electrical and Space Engineering, Luleå University of Technology, Luleå SE-97187, Sweden

ARTICLE INFO

Keywords:
Spot
SubT
Map-based navigation
3D lidar
Autonomy package

ABSTRACT

In this article, we address the task of experimental evaluation of autonomous map-based navigation of a Boston Dynamics Spot quadruped robot in confined environments equipped with the developed autonomy package that incorporates a 3D lidar, an IMU, and an onboard computer. For that, we propose an integrated software and hardware system that is considered as an enabler for robot localization and risk-aware path planning, based on the known map. The system design itself is modular and incorporates the perception capabilities required for autonomous navigation. The Spot robot first is utilized to build the offline map of the environment by utilizing the Google Cartographer simultaneous localization and mapping (SLAM) package. During the next step, the online environmental information from the autonomy package sensors and the offline map are provided to the onboard computer to localize the robot on the known map by utilizing means provided by Cartographer. Finally, the occupancy information is provided to the online grid-based path planner that generates risk-aware paths. The extensive experimental evaluation of the proposed system is performed in corridors and SubT environments.

1. Introduction

Robot navigation in GPS-denied environments has gained a lot of attention over the last few years. Presently, robotics encompasses applications in exploration, inspection and search and rescue missions [1,2]. In such missions, robots are commonly facing harsh, poorly illuminated, unstructured environments, as depicted in Fig. 1, which in most of the cases, are not friendly for human workers to enter [3]. The fundamental requirement for safe and successful robot navigation in these environments is the simultaneous localization and mapping (SLAM) approach that allows to estimate the current robot's state and to build a map of the surrounding environment.

To fulfil this requirement, multiple robot platforms like aerial, wheeled, or legged robots can be utilized together with an embedded autonomy package that commonly incorporates payloads for vision-based or lidar-based SLAM methods [4]. The first group of methods is having good capabilities of place recognition, however, on contrary, they are prone to poor performance under insufficient illumination conditions. The latter methods are based on a lidar and are determining ranges with a laser, which allows them to operate in poorly illuminated environments. Thus, in this study, as the perception sensor for navigation, will be considered a 3D lidar. Generally, building a map of a previously unknown environment is commonly referred as an exploration task, which includes application areas like planetary exploration [1], 3D reconstruction [5], search and rescue [6], etc. The accuracy and completeness of the map are the crucial factors required for the successful accomplishing of these applications. Another factor is to build a map with a sufficient number of features that will allow to re-use it for autonomous navigation of robots that for a first time enter in this area. This task is commonly referred to as a global localization problem [7] in which a robot tries to estimate its state on the known map based on its sensor data. Among the existing state-of-the-art localization algorithms one can mention Adaptive Monte Carlo Localization (AMCL) [8] and Google Cartographer [9], and among recent works, it could be mentioned [10–14], however, none of them addresses this task together with an online path planning. The authors in [15] used a static map for the localization and to continuously update another map for navigation in order to keep valid previous localization/navigation waypoints and also to allow the robot to perform path planning for longer travels (considering that some parts of the map can be impassable or new pathways might have become available). However, in their work the path planning method is not presented, the environment is full of features and is relatively small comparing to a typical confined environment.

In order to have an autonomous system operating in such challenging exploration scenarios, it is necessary to integrate the robot localization with a path planner. The general task of planning a safe path has been approached in different ways in the past, while as an example, one possible way is to use a kinematic model of the robot [16–18], where the size and manoeuvre capability of the robot could also

[*] Corresponding author.
E-mail address: antkov@ltu.se (A. Koval).

https://doi.org/10.1016/j.birob.2022.100035
Received 16 November 2021; Received in revised form 11 January 2022; Accepted 13 January 2022
Available online 20 January 2022

Fig. 1. An example of a tunnelling environment where the experimental autonomous map-based mission was carried out (Luleå Sweden).

be considered when planning a path that it is a sufficient marginal to any obstacles. Instead of using a specific robotic model, to generate paths with sufficient safety marginal, a safety distance could be added to the map. Probably the simplest way to achieve a safety distance in a map is to inflate the obstacles so that the planner thinks the obstacles are bigger than they are, as in [19]. The binary approach of inflating an obstacle could lead to unnecessary blockage, which often can be acceptable to go closer to obstacles when there are no alternatives, compared to when the space exists to have a larger marginal. Thus, instead of inflating obstacles, a risk map could be created [20,21] where in this case, the challenge is to create risk layers that are relevant and at the same time promote the desired behaviour. With a risk map, it is still needed to have a path planner and for that, as an example, the Rapidly exploring Random Tree [22] or A* style planners [23–25] could be utilized. In this article specifically the D* lite [26] expanded path planner called D$^*_+$ [27] was chosen, because of the incorporated risk layer for obstacle avoidance and treatment of unknown areas.

Thus, in this article, the main contributions are: (a) the design of the autonomy package that incorporates a 3D lidar and an IMU for the Spot (Fig. 2) robot and it can be used for its autonomous navigation; (b) the introduction of a risk-aware path planning framework for a quadruped robot, that uses the risk aware path planner D$^*_+$ [27] the and occupancy map from Google Cartographer [28], which is also used to solve the global localization task; (c) the developed Graphical User Interface (GUI) for mission control, and (d) the proposed overall system architecture for Spot autonomous map-based navigation is experimentally evaluated in a corridor like confined spaces and SubT environments.

The rest of the article is structured as follows. Section 2 describes the overall system architecture, while Section 3 introduces the risk-aware path planning framework for autonomous navigation of the quadruped robot. Finally, Section 4 provides results for the efficacy of the proposed scheme through an experimental evaluation.

2. System architecture

This Section will introduce the quadruped robot and the related sensors embedded in the autonomy package that are required for the overall autonomous operation. The autonomy package incorporates a 3D lidar Velodyne Puck Lite, an IMU Vectornav VN-100 that are placed in the front deck, an onboard Intel NUC computer, and batteries for powering all the electronics and are respectively placed in the back of the robot. To provide an unobstructed view, the 3D lidar was mounted as a column structure and the IMU was mounted at its bottom. The sensor placement on Spot is depicted in Fig. 3.

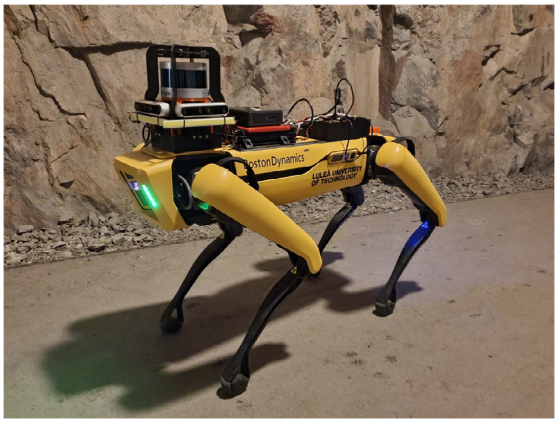


Fig. 2. The quadruped robot, Boston Dynamics Spot, that was used in experiments.

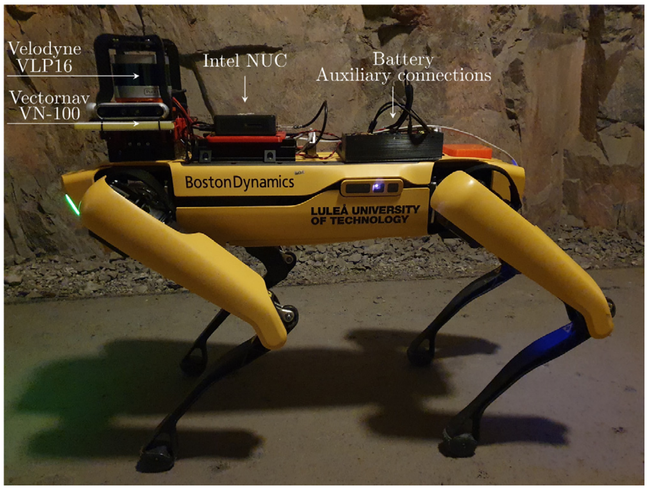


Fig. 3. Spot equipped with the proposed autonomy package. The Vectornav IMU is placed under the Velodyne lidar and is not visible on the figure.

The proposed software architecture is based on the Robot Operation System (ROS) framework for establishing a data flow between Spot and the autonomy package. Its operation can be divided into two stages. During the first stage, the offline map (**M**) is built. This is achieved

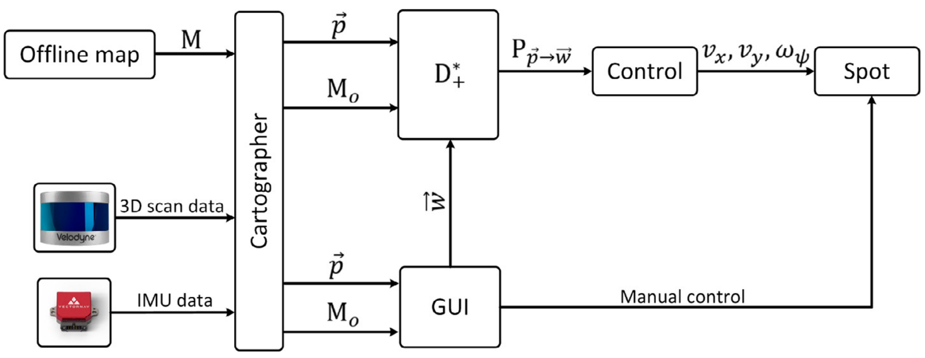


Fig. 4. Software architecture.

by registering 3D lidar and IMU sensor data at 10 Hz and 200 Hz respectively and feeding them to the Cartographer ROS package. Based on these data an offline SLAM algorithm computes the robot state that includes the position and the map information. On completion, this information is stored in a *.pbstream* file, which can be used as a **M** for localization and path planning. The second stage represents the architecture required for autonomous map-based navigation and is shown in Fig. 4. During this operation, **M** together with online data streams from the 3D lidar and IMU are coming to the Cartographer node that is operating in a localization mode and does not perform any mapping. Instead, the created submaps are matched with the ones stored in the offline map, while the IMU is used for initial estimates and to estimate the rotation between the scan matches. The estimated pose ($\vec{p}$) and the occupancy map ($\mathbf{M}_o$) are given to D^*_+ path planner and to a Graphical User Interface (GUI) as depicted in Fig. 10. In the GUI and based on $\mathbf{M}_o$ the waypoints ($\vec{w}$) are set and provided to D^*_+. After that, the D^*_+ generates a risk-aware path ($P_{\vec{p}\to\vec{w}}$) and provides it to a position PID controller that generates velocity commands (v_x, v_y, ω_ψ) for Spot.

3. Risk-aware path planning

3.1. Cartographer

To be able to navigate to a specific point $\vec{w}$, primarily three things are needed, the robot's current position $\vec{p}$, point $\vec{w}$'s position relative to $\vec{p}$, and a map $\mathbf{M}_o$ in the same reference frame as $\vec{p}$ and $\vec{w}$. Alike the aerial platforms that have 6 degrees of freedom, the ground robots have only 3 degrees of freedom, x, y and ψ and to use the 3D occupancy information for path planning is excessive in general. Considering this, our implementation is utilizing a 2D occupancy map ($\mathbf{M}_o$). For that, we integrated the Google Cartographer [9] Simultaneous Localization and Mapping (SLAM) ROS package that provides $\mathbf{M}_o$ and localization (i.e. provides $\vec{p}$). The Cartographer performs the local scan matching and the global loop closures in parallel. To achieve local scan matches and global loop closures, it introduced a concept called submaps used by Cartographer to internally accumulate the 3D scan data at best estimated position. Each submap contains a small section of the map with time consecutive 3D scans. When a new 3D scan at an estimated position is received it is matched into the latest submap, and upon submap finish, so no new scans can be inserted, it is considered for loop closure. The local scan matching is using the IMU data to generate an initial guess for $\vec{p}$, that are utilized to match the scans to the submap with interpolation of the submap and sub-pixel alignment. This process is slowly accumulating errors and the global loop closure is correcting the trajectory so the submaps form a coherent map. Global loop closure is matching finished submaps to each other, instead of operating on 3D scans, to a submap as local scan matching does. In addition to the loop closures and submaps, the Cartographer is also providing a 2D occupancy map $\mathbf{M}_o$ from the 3D data. Moreover, it can save the state of the generated map so it can be loaded as a pre-computed map at a later time and used only for localization. The global loop closures can find the robot's position $\vec{p}$ without any other information than the **M**, 3D scans and IMU data. This means that $\vec{p}$ can be found on $\mathbf{M}_o$ and thus allowing for an operation without creating a map before each mission.

Thus, by determining the point $\vec{p}$ in the $\mathbf{M}_o$'s reference frame and given the way-point $\vec{w}$, the path planner can generate a path $P_{\vec{p}\to\vec{w}}$ as can be seen in Fig. 5.

3.2. Path planner

D^*_+ is a 2D or 3D grid based global path planner that is an extension of D* lite [26]. In this case, it is the 2D grid map ($\mathbf{M}_o$) from Google Cartographer [9] used to perform the path planning. In general, D^*_+ plans the shortest path $P_{\vec{p}\to\vec{w}}$ between points $\vec{p}$ and $\vec{w}$ in a grid map $\mathbb{G}$, based on the traversal costs ζ so that $P_{\vec{p}\to\vec{w}}$ is the path in $\mathbb{G}$, where the overall traversal cost $\sum \forall\zeta \in P_{\vec{p}\to\vec{w}}$ is the smallest possible cost. D* lite finds P with a breadth-first search [23] executed from $\vec{w}$ to $\vec{p}$. The benefit of building a path from $\vec{w}$ to $\vec{p}$, is that changes in $\mathbb{G}$ and changes in $\vec{p}$ are likely to happen close to $\vec{p}$. Therefore it is likely that most of the path remains the same even after changes, it is there for possibly to utilize the previously performed calculations to speed up future planning of P. The reason to compare D^*_+ against D* lite is that D^*_+ is based on D* lite package. The original D* algorithm has support of dynamic path replanning based on the occupancy information updates. D* lite package takes as an input the offline map in a mesh file format that does not have occupancy information updates, while the developed D^*_+ takes as an input occupancy map that is dynamically updated and allows navigation in non static environments. So, the replanning rate is tailored to the occupancy map updates and thus it cannot be used for active obstacle avoidance. Thus D* lite was chosen to be used as the base algorithm for D^*_+ planner. The D^*_+ path planner utilizes three main differences from D* lite namely: (1) the treatment of unknown cells, (2) proximity risk and (3) map updates. The rest of this Section describes these modifications.

Traditionally, a D* lite grid is built by using a binary map that includes either the occupied cells c_o or the free cells c_f. To avoid invalid P's short-cutting through walls, due to imperfection in sensors or map sparsity, a third main type of cells is introduced, the unknown cells c_u. Each c is assigned with a traversal cost ζ, where c_o should have the max value ζ_o and c_f should have a minimum ζ_f. The traversal cost ζ_u for c_u can be set in between ζ_f and ζ_o, where the value depends on the navigation scenario, by assigning ζ_u to a low value for exploration behaviour or to a higher value for safely following known predefined paths.

The benefit of using c_u can be seen in Fig. 6 where D* lite plans an invalid path through what actually is a solid rock. By using D^*_+, the path is planned alongside the tunnel and with sufficient safety marginal when both planners used the same input map $\mathbf{M}_o$ and the same start and goal positions.

Originally, D* lite plans paths close to c_o to get the shorter path without considering safety aspects that can cause issues in real-life

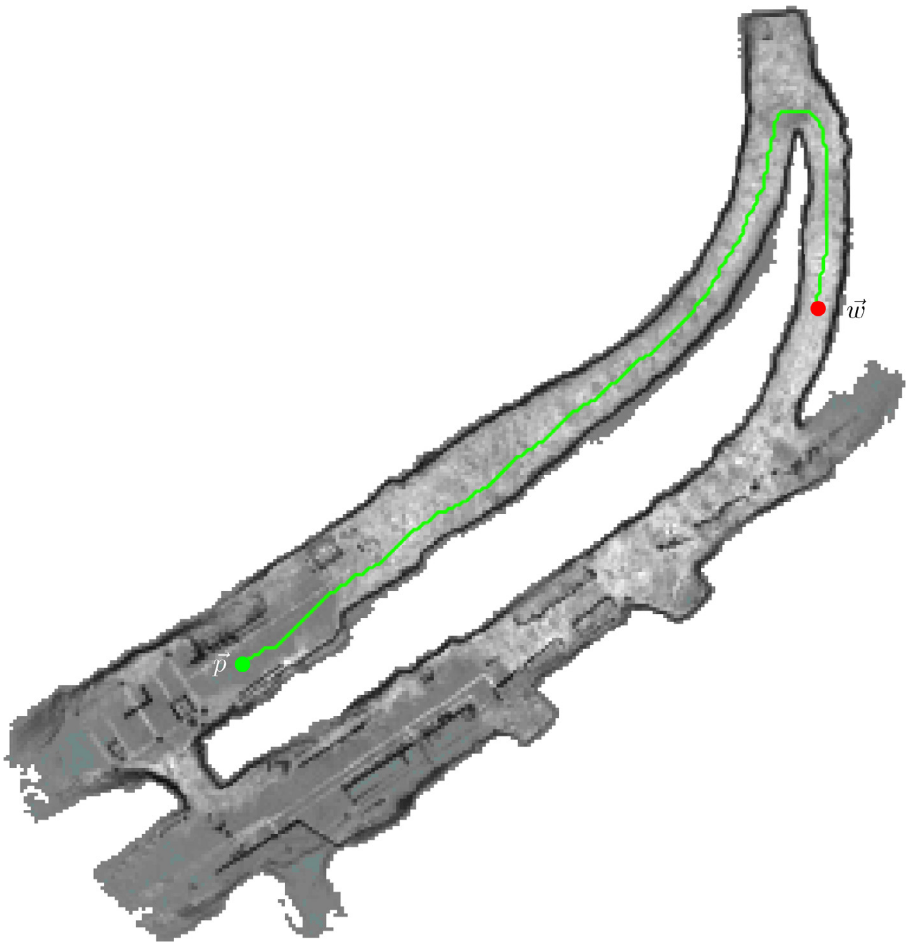


Fig. 5. A 2D map from Cartographer, where D^*_+ have planned a path (green line) from $\vec{p}$ (green dot) to $\vec{w}$ (red dot). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

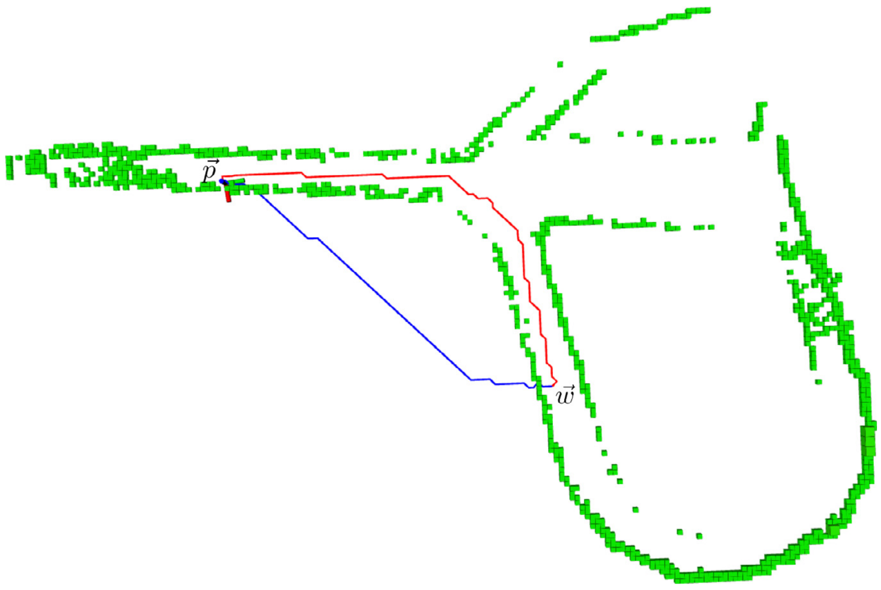


Fig. 6. A comparison between the path generated by D^* lite (blue line) and D^*_+ (red line). D^* lite is creating a shortcut through the solid rock, while D^*_+ generates a path inside the tunnel that is possible to follow. The path is planned from $\vec{p}$ to $\vec{w}$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

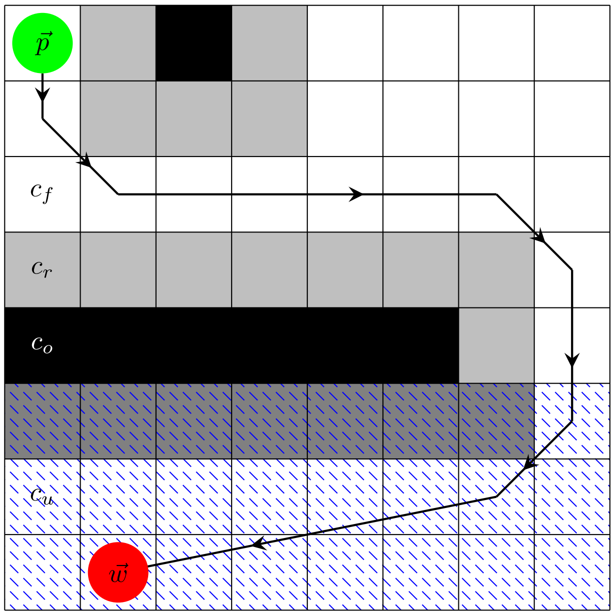


Fig. 7. A 2D grid for D_+^* planning. The grid contains free cells c_f (white), occupied cells c_o (black) and unknown cells c_u (blue dotted). The grey cells c_r represent the added traversal cost for proximity ζ_r to c_o, darker grey is a higher traversal cost $\zeta_r + \zeta_u$ due to the combination of unknown and proximity ones. In this case, D_+^* is planning a path from $\vec{p}$ to $\vec{w}$ along the arrow path. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

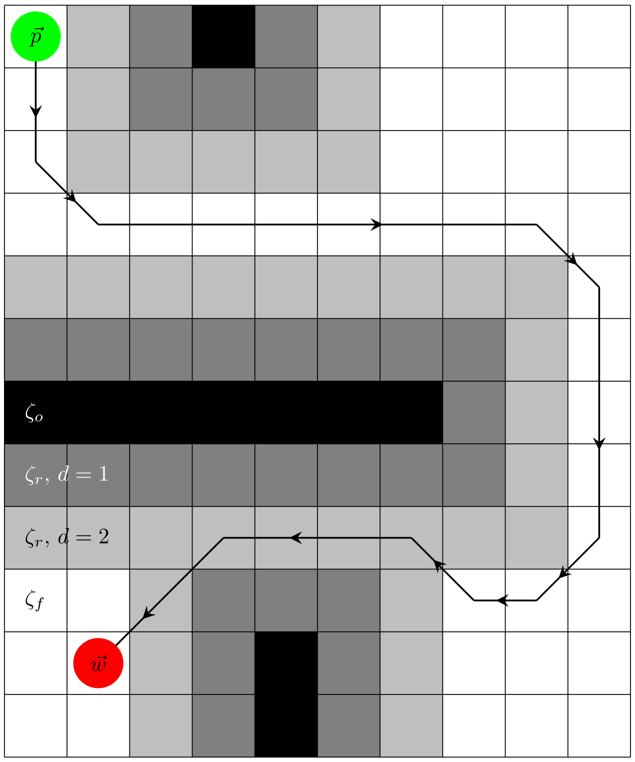


Fig. 8. An example path planned from $\vec{p}$ to $\vec{w}$, with proximity cost added to cells with two cells distance ($r = 2$) to c_o. ζ_r are represented with grey, light grey for lower cost and darker grey for higher cost. Close to the $\vec{w}$ (visualized in red circle) the path traverses through the lower proximity cost (light grey grid blocks), which is the lowest possible cost from the entire path due to the lack of a path through ζ_f. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

robot deployment e.g the physical robot size can potentially collide with the surrounding environment for such paths. To address this issue, D_+^* incorporates information for the cells close to c_o, within range r (measured in cells), by giving a traversal cost $\zeta_r = \zeta_u/(d+1)$ added to its ζ, where d is the distance to the c_o counted in cells. If a cell is within r of multiple c_o, then ζ_r will be added for the closest c_o.

In this case, the scaling behaviour for ζ_r creates a gradient risk area that will make D_+^* plan a path in the middle of a narrow tunnel-like area and with the desired safety marginal in more open spaces. By using ζ_r, the planned path resembles the ones seen in Fig. 8, where the path is planned inside c_f where it is possible. In situations (e.g. extremely narrow corridor-like environments), where there is no c_f, the proposed planner prioritizes the generation of paths inside c_r.

While the robot moves within an area, the mapping tool continuously expands the information of the visited area, updates the surroundings, fills the missing parts of the scene, while exploring new areas, thus $\mathbb{G}$ is created in a dynamic and updating way, by using the occupancy map as an input instead of the mesh map used in D* lite implementation. More specifically, this dynamic map update is captured in Figs. 7 and 9, where as long as the robot traverses the planned path, more and more knowledge of the environment is revealed and saved in $\mathbb{G}$. If the current path is discovered to be blocked with an obstacle then D_+^* will replan the path avoiding it.

3.3. GUI and interface

Boston Dynamics provides an API for interfacing with the Spot robot, that makes it possible to develop a ROS driver for Spot as Clearpath robotics have done [29]. In our study, we are utilizing Clearpath robotics Spot ROS driver to control Spot in the experiments. To make it possible to send $\vec{w}$ on M_o on a larger scale and to simplify the overall usability, a GUI as depicted in Fig. 10 has been developed that has the basic control functions needed to operate Spot. The GUI itself has the following structural elements, including the Spot services, the Manual control blocks, and the Mission control block with the map. In the GUI is implemented a visualization of M_o, on which it is possible

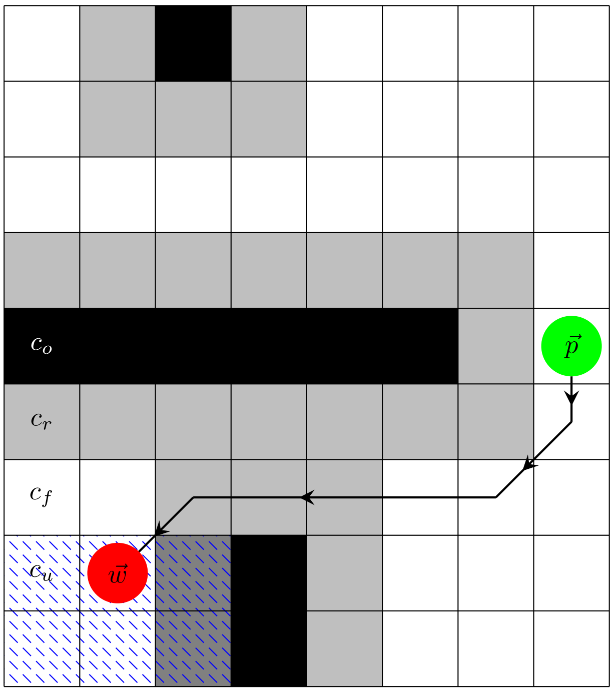


Fig. 9. As the robot traverses the path planned in Fig. 7, the unknown cells (c_u) will become known and D_+^* will replan the path. The map is expanded as well.

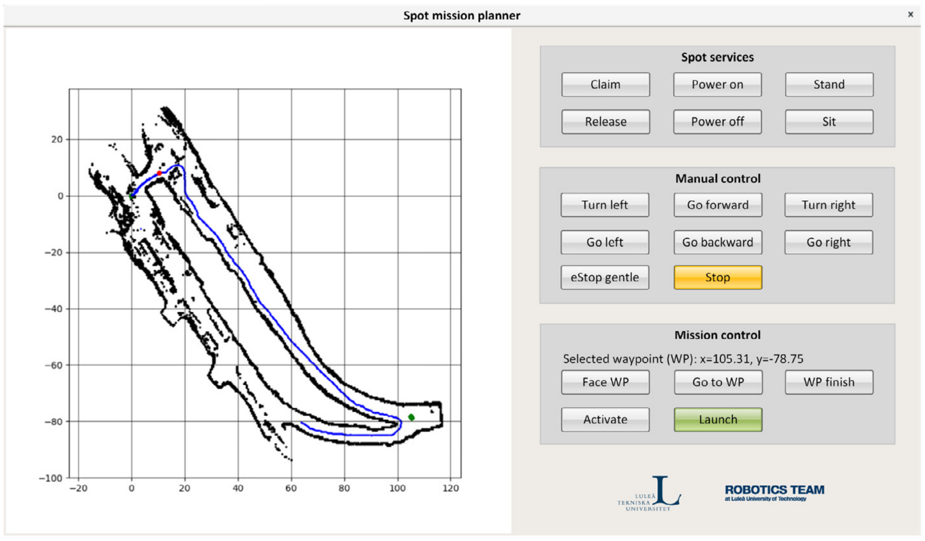


Fig. 10. The developed GUI for map based navigation, that incorporates map, mission control, manual control, and Spot services.

to set $\vec{w}$ by clicking. A waypoint mission is also possible to set up using the GUI, by using the "Finish WP" button where the current clicked point is set as a second $\vec{w}$ and thus allowing an intermediate $\vec{w}$ to be set before the mission is launched. While a mission is executed, its progress can be followed in the GUI from where it can be aborted, if needed. Additionally, it is also possible to manually control Spot through the GUI, while it should be stated that this is not an autonomous operation but it supports the initial M creation as well. Once the waypoints are set and D^*_+ has planned $P_{\vec{p}\rightarrow\vec{w}}$, the path is sent to the position PID controller that calculates the necessary velocities to follow the path. For added usability, the controller is capable to hold positions, adjusting the heading towards the desired point, and for safety, it has an electronic off switch, all of which is controlled through the GUI.

4. Experimental evaluation

In this Section it will be introduced the experimental evaluation of Spot map-based navigation. The main goal of the conducted experiments was to evaluate the relocalization and path planner performance in terms of their update rates. For that, the system performance was tested in three experimental areas, with all computations running on an onboard computer Intel NUC with an i7 CPU, 8 GB of RAM and SSD storage. Moreover, the computer was running Ubuntu 18.04 along with ROS Melodic. The areas in which the experiments took place include a corridor environment at Luleå University of Technology (a map of the area is in Fig. 11(a)), a small underground tunnel (a map of the area is in Fig. 11(b)) with the 3–5 m width, and in a large underground tunnel (a map of the area is in Fig. 11(c)) with width of 5–15 m, where in the area were placed some large mining machines on the sides. Both underground tunnels had uneven bare rock as walls for most of the parts, as depicted in Fig. 1.

To build maps (Fig. 11) of the targeted environments, the IMU and 3D lidar data were recorded into rosbags and afterwards the Cartographer was tuned and used to generate the maps for relocalization.

The first set of experiments was carried out in the corridor environment depicted in Fig. 11(a) at the LuleåUniversity that was relatively small and a featureless area with straight walls. The planned and travelled paths are depicted in Fig. 12. During the experiment, the Spot robot travelled 12 m, while keeping the position update rate at 17 Hz on average and the path planning and replanning average update rate of 8 Hz.

Table 1
Summary of results from the experimental evaluation.

	Corridor	Small tunnel	Large tunnel
Position update rate, [Hz]	17.077	24.3980	19.6963
Path update rate, [Hz]	8.096	9.1327	6.4733
Travelled distance, [metres]	12.1827	42.6708	334.6075

The second set of experiments was carried out in the small tunnel environment (Fig. 11(b)) with the corridor width of 3.5 m. In this case, the waypoint was set beyond the line of sight. As it can be seen from Fig. 13, the initially loaded map provided the occupancy information that produced the path going through the obstacles. However, along with the robot navigation towards the set waypoints, the occupancy information got updated and the initially created path was dynamically re-planned with respect to the robot's current position. In this case, the robot reached successfully the initially set waypoint. During the experiment, the Spot robot travelled 43 m, while keeping the relocalization/position update rate at 24 Hz on average and the path planning and replanning average update rate of 9 Hz. The overall travelled and planned paths from $\vec{p}$ to $\vec{w}$ is shown in Fig. 14.

The third set of experiments was conducted in the large underground tunnel area as depicted in Fig. 15. During this test, multiple waypoints were set. The first waypoint $\vec{w}_1$ was set in order to evaluate the continuous system performance in the environment that downscales from 15 metres in width to 5 m. The next waypoint $\vec{w}_2$ was set as shown in Fig. 16(b) and while the robot was navigating it was changed to $\vec{w}_3$. The follow-up waypoints $\vec{w}_{4-7}$ were set sequentially until the robot reached the final goal position. During the experiment, the Spot robot travelled 335 m, while keeping the relocalization/position update rate at 19 Hz in average and the path planning and replanning average update rate of 6 Hz. The overall travelled path is shown in Fig. 15.

The summary of the performed experiments is shown in Table 1, where it can be highlighted that the Cartographer's relocalization performance depends on the number of features and the size of the area, seeing that the highest position update rate was achieved in the small tunnel with relatively rich with features walls, comparatively to the corridor environment. And having also a higher update rate than in a larger tunnel, which had more distinct features placed inside the tunnel. The path planner was showing a solid performance in planning and dynamically updating the path with respect to current robot position.

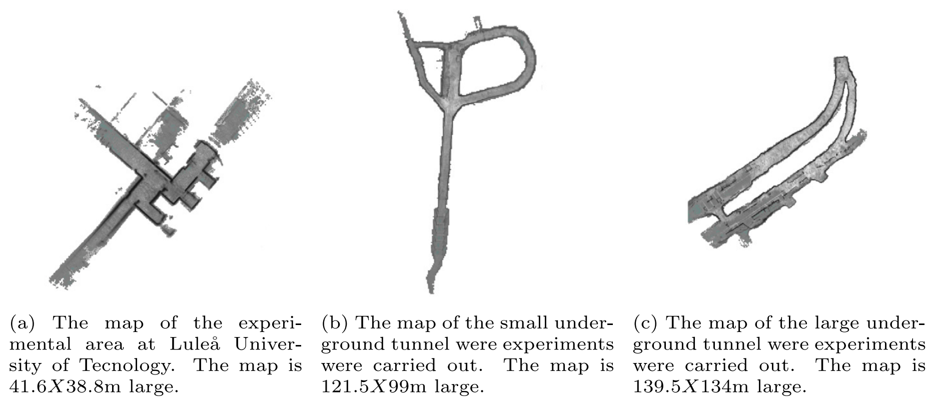
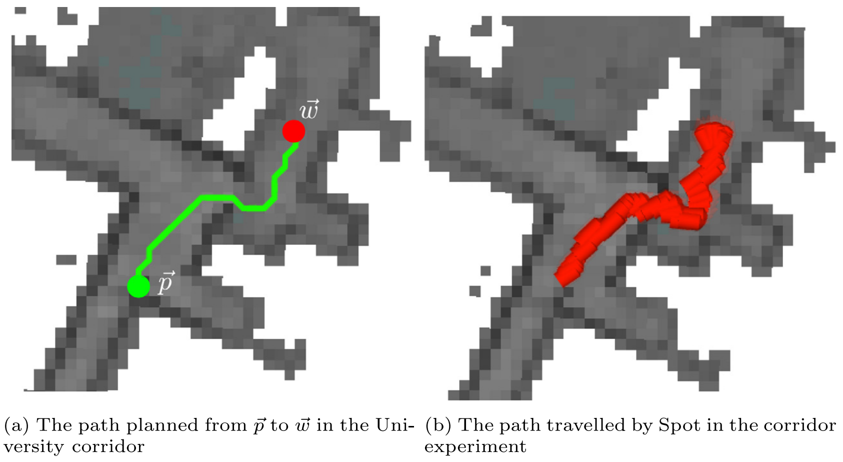

(a) The map of the experimental area at Luleå University of Tecnology. The map is $41.6X38.8$m large. (b) The map of the small underground tunnel were experiments were carried out. The map is $121.5X99$m large. (c) The map of the large underground tunnel were experiments were carried out. The map is $139.5X134$m large.

Fig. 11. Maps off the areas were the experiments took place.



(a) The path planned from $\vec{p}$ to $\vec{w}$ in the University corridor (b) The path travelled by Spot in the corridor experiment

Fig. 12. An experiment in corridor environment at the university where the traversed path (b) can be compared to the planned path (a).

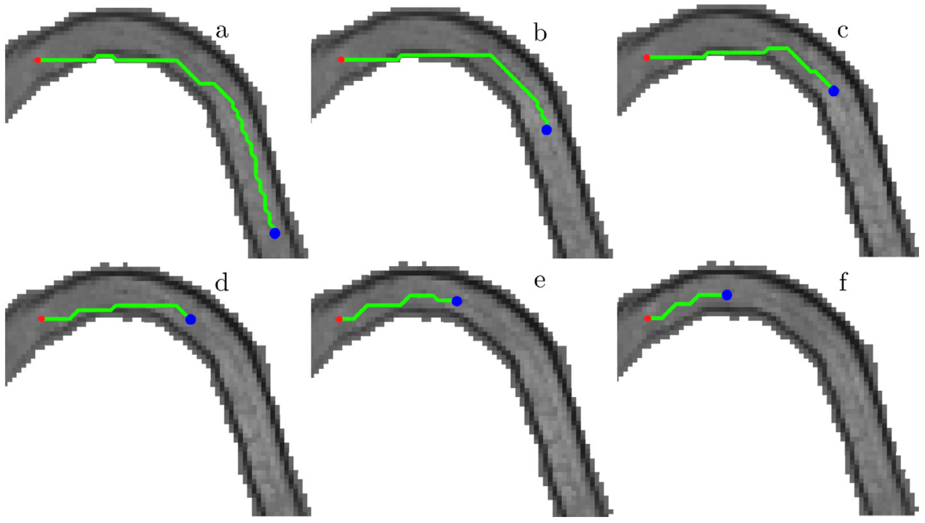


Fig. 13. A procreation of how $P_{\vec{p}\to\vec{w}}$ updates as $\vec{p}$ (blue circle) changes in a–f while $\vec{w}$ (red circle) stays the same. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5. Conclusions

In this article, we presented a novel system for the autonomous map-based risk-aware navigation that was experimentally evaluated in the corridor and SubT environments. The designed system showed solid performance, while being able to produce at an average a 17 Hz position update rate for the autonomous path planning, while the path planner was generating feasible and safe paths in all the cases. Through

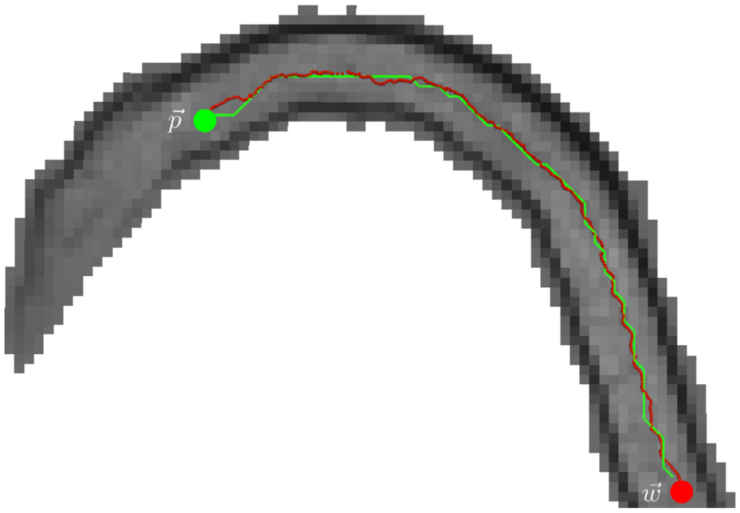


Fig. 14. Representation of travelled and planned paths. The current position $\vec{p}$ is represented with a green circle, the waypoint $\vec{w}$ is represented with red circle. The travelled path $\vec{w}$ to $\vec{p}$ (that was planned and followed in Fig. 13) is represented with red line, while the planned path to return from $\vec{p}$ to $\vec{w}$ is shown in green. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

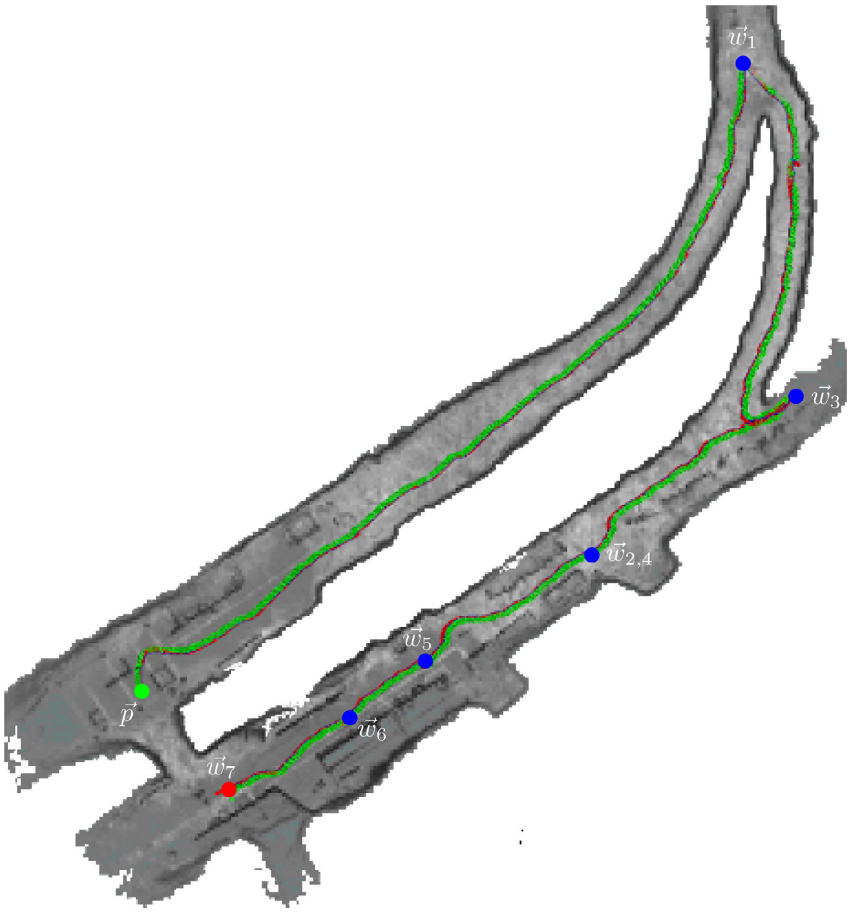


Fig. 15. Sequential $\vec{w}$ were set in a large tunnel. Spot navigates to each of $\vec{w}$ and traverses about 335m. A section wise presentation can be seen in Fig. 16.

the operation, Cartographer was able to perform relocalization in the known maps and thus allowing to get a global pose estimate required for map-based waypoint navigation. Future work will include 3D scene analysis and building of a traversability map, which will allow to identify slopes and uneven areas and to design an adaptive control architecture for the Spot robot.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

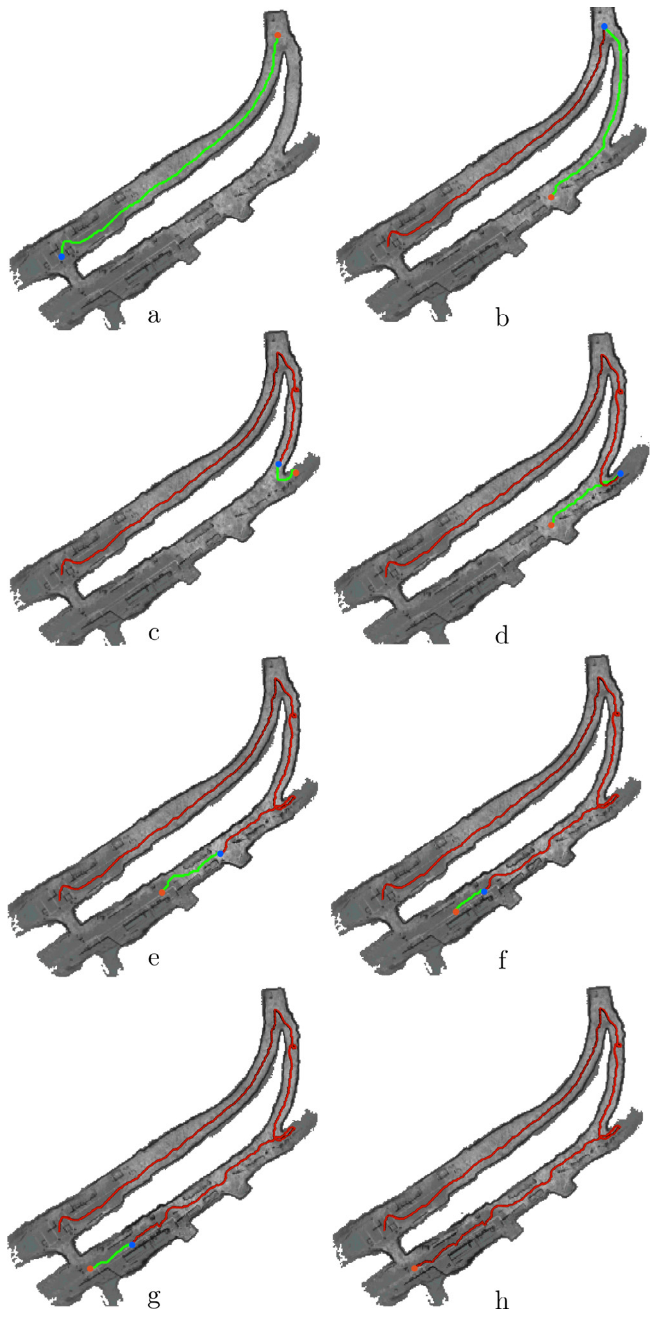


Fig. 16. Long waypoint mission where each waypoint section is shown in a–h. For each section of the mission a path is planned (green line) from the blue circle ($\vec{p}$) to the red circle ($\vec{w}$). The path that Spot actually traversed is shown as the red line for the previous section. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Acknowledgment

This work has been partially funded by the European Unions Horizon 2020 Research and Innovation Programme under the Grant Agreement No. 869379 illuMINEation.

References

[1] G. Nikolakopoulos, A. Agha, Pushing the limits of autonomy for enabling the next generation of space robotics exploration missions, Computer 54 (11) (2021) 100–103.

[2] S.S. Mansouri, C. Kanellakis, E. Fresk, B. Lindqvist, D. Kominiak, A. Koval, P. Sopasakis, G. Nikolakopoulos, Subterranean MAV navigation based on nonlinear MPC with collision avoidance constraints, IFAC-PapersOnLine 53 (2) (2020) 9650–9657.

[3] S.S. Mansouri, C. Kanellakis, D. Kominiak, G. Nikolakopoulos, Deploying MAVs for autonomous navigation in dark underground mine environments, Robot. Auton. Syst. (2020) 103472.

[4] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, R. Daniela, Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping, in: IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS), IEEE, 2020, pp. 5135–5142.

[5] J. Tachella, Y. Altmann, N. Mellado, A. McCarthy, R. Tobin, G.S. Buller, J.-Y. Tourneret, S. McLaughlin, Real-time 3D reconstruction from single-photon lidar data using plug-and-play point cloud denoisers, Nature Commun. 10 (1) (2019) 1–6.

[6] H.A. Lauterbach, C.B. Koch, R. Hess, D. Eck, K. Schilling, A. Nüchter, The Eins3D project—Instantaneous UAV-based 3D mapping for search and rescue applications, in: 2019 IEEE International Symposium On Safety, Security, And Rescue Robotics (SSRR), IEEE, 2019, pp. 1–6.

[7] S. Se, D. Lowe, J. Little, Global localization using distinctive visual features, in: IEEE/RSJ International Conference On Intelligent Robots And Systems, 1, IEEE, 2002, pp. 226–231.

[8] D. Fox, Adapting the sample size in particle filters through KLD-sampling, The Int. J. Robot Res 22 (12) (2003) 985–1003.

[9] W. Hess, D. Kohler, H. Rapp, D. Andor, Real-Time Loop Closure in 2D Lidar slam, in: 2016 IEEE International Conference On Robotics And Automation (ICRA), 2016, pp. 1271–1278.

[10] L. Feng, S. Bi, M. Dong, F. Hong, Y. Liang, Q. Lin, Y. Liu, A global localization system for mobile robot using LIDAR sensor, in: 2017 IEEE 7th Annual International Conference On CYBER Technology In Automation, Control, And Intelligent Systems (CYBER), IEEE, 2017, pp. 478–483.

[11] W. Xiaoyu, L. Caihong, S. Li, Z. Ning, F. Hao, On adaptive monte carlo localization algorithm for the mobile robot based on ROS, in: 2018 37th Chinese Control Conference (CCC), IEEE, 2018, pp. 5207–5212.

[12] Y. Chen, W. Chen, L. Zhu, Z. Su, X. Zhou, Y. Guan, G. Liu, A study of sensor-fusion mechanism for mobile robot global localization, Robotica 37 (11) (2019) 1835–1849.

[13] H.S. Hoj, S. Hansen, E. Svanebjerg, Probabilistic model-based global localization in an airport environment, in: 2021 IEEE 17th International Conference On Automation Science And Engineering (CASE), IEEE, 2021, pp. 1370–1375.

[14] Z. Wu, Y. Yue, M. Wen, J. Zhang, J. Yi, D. Wang, Infrastructure-free hierarchical mobile robot global localization in repetitive environments, IEEE Trans. Instrum Measur 70 (2021) 1–12.

[15] C.M. Costa, H.M. Sobreira, A.J. Sousa, G.M. Veiga, Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms, Robot. Auton. Syst 76 (2016) 113–140.

[16] J. Tordesillas, B.T. Lopez, J.P. How, FASTER: Fast and safe trajectory planner for flights in unknown environments, in: 2019 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS), IEEE, 2019.

[17] J. Tordesillas, J.P. How, FASTER: Fast and safe trajectory planner for navigation in unknown environments, IEEE Trans. Robot (2021) 1–17.

[18] F. Yan, Y.-S. Liu, J.-Z. Xiao, Path planning in complex 3D environments using a probabilistic roadmap method, Int. J. Autom Comput 10 (6) (2013) 525–533.

[19] F. Li, S. Zlatanova, M. Koopman, X. Bai, A. Diakité, Universal path planning for an indoor drone, Autom. Construct 95 (2018) 275–283.

[20] B. Hoover, S. Yaw, R. Middleton, CostMAP: an open-source software package for developing cost surfaces using a multi-scale search kernel, Int. J. Geograph Inform Sci 34 (3) (2020) 520–538.

[21] J. Faigl, M. Prágr, On unsupervised learning of traversal cost and terrain types identification using self-organizing maps, in: I.V. Tetko, V. Kůrková, P. Karpov, F. Theis (Eds.), Artificial Neural Networks And Machine Learning – ICANN 2019: Theoretical Neural Computation, Springer International Publishing, Cham, 2019, pp. 654–668.

[22] L. Jaillet, J. Cortés, T. Siméon, Sampling-based path planning on configuration-space costmaps, IEEE Trans. Robot 26 (4) (2010) 635–646, http://dx.doi.org/10.1109/TRO.2010.2049527.

[23] S. Koenig, M. Likhachev, D* lite, in: AAAI/IAAI, 15, 2002, pp. 476–483.

[24] D. Ferguson, A. Stentz, Using interpolation to improve path planning: The field D* algorithm, J. Field Robot 23 (2) (2006) 79–101.

[25] A. Nash, K. Daniel, S. Koenig, A. Felner, Theta*: Any-angle path planning on grids, in: AAAI, 7, 2007, pp. 1177–1183.

[26] M. Sheckells, DSL GridSearch: D*-lite on a uniformly spaced 3D or 2D grid, 2018, URL https://github.com/jhu-asco/dsl_gridsearch Accessed: February 2021.

[27] S. Karlsson, D^*_+: A Generic platform-agnostic and risk-aware path planing framework with an expandable grid, 2021, URL https://github.com/LTU-RAI/Dsp Accessed: January 2022.

[28] W. Hess, D. Kohler, H. Rapp, D. Andor, Real-time loop closure in 2D lidar slam, in: 2016 IEEE International Conference On Robotics And Automation (ICRA), IEEE, 2016, pp. 1271–1278.

[29] C. Robotics, Spot ROS (package) challenge, 2020, URL https://https://clearpathrobotics.com/spot-robot/ Accessed: November 2021.