Expert Systems With Applications 215 (2023) 119408

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Review

D^*_+: A risk aware platform agnostic heterogeneous path planner ☆

Samuel Karlsson *, Anton Koval, Christoforos Kanellakis, George Nikolakopoulos

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

ARTICLE INFO

Dataset link: https://github.com/LTU-RAI/Dsp

Keywords:
DSP
Path planing
Risk aware
Platform agnostic

ABSTRACT

This article establishes the novel D^*_+, a risk-aware and platform-agnostic heterogeneous global path planner for robotic navigation in complex environments. The proposed planner addresses a fundamental bottleneck of occupancy-based path planners related to their dependency on accurate and dense maps. More specifically, their performance is highly affected by poorly reconstructed or sparse areas (e.g. holes in the walls or ceilings) leading to faulty generated paths outside the physical boundaries of the 3-dimensional space. As it will be presented, D^*_+ addresses this challenge with three novel contributions, integrated into one solution, namely: (a) the proximity risk, (b) the modeling of the unknown space, and (c) the map updates. By adding a risk layer to spaces that are closer to the occupied ones, some holes are filled, and thus the problematic short-cutting through them to the final goal is prevented. The novel established D^*_+ also provides safety marginals to the walls and other obstacles, a property that results in paths that do not cut the corners that could potentially disrupt the platform operation. D^*_+ has also the capability to model the unknown space as risk-free areas that should keep the paths inside, e.g in a tunnel environment, and thus heavily reducing the risk of larger shortcuts through openings in the walls. D^*_+ is also introducing a dynamic map handling capability that continuously updates with the latest information acquired during the map building process, allowing the planner to use constant map growth and resolve cases of planning over outdated sparser map reconstructions. The proposed path planner is also capable to plan 2D and 3D paths by only changing the input map to a 2D or 3D map and it is independent of the dynamics of the robotic platform. The efficiency of the proposed scheme is experimentally evaluated in multiple real-life experiments where D^*_+ is producing successfully proper planned paths, either in 2D in the use case of the Boston dynamics Spot robot or 3D paths in the case of an unmanned areal vehicle in varying and challenging scenarios.

1. Introduction

Robots are becoming more and more common for a large variety of tasks in real-life challenging environments, including but not limited to search and rescue (Agha et al., 2021; Hayat, Yanmaz, Bettstetter, & Brown, 2020; Mishra, Garg, Narang, & Mishra, 2020; San Juan, Santos, & Andújar, 2018; Schedl, Kurmi, & Bimber, 2021), inspection (Zhang, Zhang, & Low, 2021), and delivery (She & Ouyang, 2021). In general, robotic competitions and similar events like the recent DARPA's Sub-Terranean Challenge (DARPA, 2020) have increased the popularity of robotic platforms, their interaction, and their collaborative operation. With the utilization of multiple robotic platforms, large areas can be covered collaboratively, or multiple tasks can be executed simultaneously with increased performance. However, with the increased number of robots, the complexity increases as well, while the overall usability reduces for every robot, a reality that is caused by the utilization of unique robot-specific software and corresponding settings. One way to simplify the challenge of operating multiple robots is to unify parts of the software independently of the type of platform. Path planning is such an area where a platform-agnostic algorithm could be beneficial and thus in this article we propose a novel D^*_+ platform-agnostic global path planner and experimentally evaluate it with the Boston Dynamics (BD) Spot quadruped robot and a quadrotor.

Some path-planning algorithms base their operation on the utilization of environmental characteristics to generate paths in the operational environment. As an example, one can mention the COMPRA framework (Lindqvist, Kanellakis, Mansouri, akbar Agha-mohammadi & Nikolakopoulos, 2021) where the narrow nature of tunnels is utilized to fly towards the deepest point of the tunnel and thus explore it in an efficient way. However, this method works well in tunnels and similar environments, but it is not able to handle junctions and open spaces in

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

* Corresponding author.
E-mail addresses: samkar@ltu.se (S. Karlsson), antkov@ltu.se (A. Koval), christoforos.kanellakis@ltu.se (C. Kanellakis), geonik@ltu.se (G. Nikolakopoulos).

https://doi.org/10.1016/j.eswa.2022.119408
Received 8 June 2022; Received in revised form 2 December 2022; Accepted 2 December 2022
Available online 6 December 2022

a good way. For similar reasons, using a heterogeneous path planner, like the herein novel proposed D^*_+ that is able to work efficiently in many environments, is a most desired need.

Many path planners utilize an occupancy map (Hornung, Wurm, Bennewitz, Stachniss, & Burgard, 2013; Wang et al., 2021; Xiong, Gao, Wang, Li, & Lin, 2021; Zhang et al., 2021) to plan upon, as for example, RRT*, A*, theta*, D*, and the jump point search (Wu, Xu, Zhen, & Wu, 2019; Zammit & Kampen). However, these planners do rely on a dense occupancy map in order to avoid the problematic situations of generating invalid paths or shortcuts that are planned through obstacles and wall openings. Multiple ways of plugging such holes have been suggested, like filtering (Atapour-Abarghouei & Breckon, 2018) or inflation (Li, Zlatanova, Koopman, Bai, & Diakité, 2018) for example. Filtering can solve many imperfections of a map, however, it can also filter out smaller features, such as small actual openings in the operational environment. Furthermore, inflation of the occupied spaces is a simple solution that creates a safety marginal that forces the path to be planned inside the safe region and thus has some marginal for errors. However, inflation is a hard artificially created boundary that can block a path, where a robot would have been able to go, even though it would be a higher risk passage. Alike these two approaches, risk-aware-based path planners can accept some risks to reach the goal without taking unnecessary risks (Laconte et al., 2021).

Utilizing a risk assessment to plan for a better path is something that has been addressed previously in the scientific literature as in Cuevas, Ramirez, Shames, and Manzic (2021), Huang et al. (2020). However, one of the biggest challenges, when working with risk maps, is how to generate the risk and what to consider as a risk. In many realistic application scenarios, a type of risk that can be considered is the consequences if a crash or failure occurs (Hu, Pang, Dai, & Low, 2020; Primatesta et al., 2017), and then minimizing the consequences of a crash. An alternative method is to utilize offline and online risk maps to plan crash-safe paths as in da Silva Arantes, Toledo, Williams, and Ono (2019) and Primatesta, Guglieri, and Rizzo (2019), while this approach is considering a 2D risk map in open areas to plan their paths. Considering the consequences of a crash is a very good approach to enable good paths, however, it does not aid in actually avoiding a true collision, while for the case of path planning for Unmanned Aerial Vehicles (UAVs) that can move in 3D, the risk assessment and path planning algorithms should utilize all three dimensions when planning a path (Hakobyan, Kim, & Yang, 2019). Risk considerations for safe path planning in a confined environment and by using 3D maneuvering is barely done in the literature. Thus, one of the few works as in Zhou, Pan, Gao, and Shen (2021) utilizes a prediction risk of unknown areas and controls the yaw angle to identify and calculate potential dangers as early as possible. For the case of ground robots, rough and uneven surfaces can be considered as a realistic risk (Ono, Fuchs, Steffy, Maimone, & Yen, 2015; Puck et al., 2020), however, sharp rocks and loose sand that are true risks for ground robots do not constitute real relevant risks for UAVs unless it is so loose that the UAV creates dust out of it. As such, depending on the environment and the utilized robotic platform, different things can be identified as a risk and with a different estimation of their severity impact, while one of the few things that are always dangerous for the robotic operation is the existence of obstacles and the corresponding proximity to the obstacles.

Many path planners try to plan the shortest possible path, either by ensuring that it as a grid search method like the D* style planners or as sample-based planners, like the RRT* that performs pseudo-random guessing of a path and settles for one that is good enough. In this approach, the shortest path around a corner is tangent to the inside corner, but although this is a free space, it is not necessarily safe for a robot to be that close to an obstacle, mainly due to inaccuracies in mapping, state estimation, and path tracking. Thus, one option to generate safe paths is to use model constraints, such in the methods in Tordesillas and How (2021), Tordesillas, Lopez, and How (2019) and Yan F. (2013) when planning. These path planners are able to generate paths with the knowledge of robot kinematics and ensure that the path is possible to traverse. One of the main disadvantages is the complexity of these planners is the need to know the kinematic model of the robot, which makes it hard to allow for a generalized application of these algorithms between different robots.

Among the platform agnostic planners, one can highlight the Graph-based Exploration Planner 2.0 (Gbplanner) Dang, Tranzatto, Khattak, Mascarich, Alexis, and Hutter (2020), Kulkarni, Dharmadhikari, Tranzatto, Zimmermann, Reijgwart, De Petris, Nguyen, Khedekar, Papachristos, Ott, Siegwart, Hutter, and Alexis (2022), which was developed and utilized by team CERBERUS who won the DARPA subterranean challenge (DARPA, 2020). Gbplanner is a combined exploration and path planning solution that builds a graph through a RRT local planning step. After the graph is built, Dijkstra's algorithm is used to rapidly plan global paths on it. In our work, we will evaluate our planner against Gbplanner, which is a state-of-the-art path planning solution.

Thus, the main contribution of this article is the establishment of the novel D^*_+ occupancy-based risk-aware, platform-agnostic, heterogeneous global path planner. The novelty of D^*_+ stems from the evolution of D*-lite by (1) treating proximity to occupied spaces as risky areas, (2) modeling explicitly unknown areas as a risk, and (3) allowing dynamic 3D map updates for planning. Moreover, D^*_+ is a field-hardened global planner that has been extensively tested and evaluated on a quadruped Spot robot and an UAV. D^*_+ stands, to the best of the author's knowledge, one of the few global path planners that are able to avoid obstacles by maneuvering in full 3D and considering risks while doing so. At the same time, D^*_+ is the first algorithm where these types of risks and path planning methods are used in combination. Finally, the code has also been made open source for the robotics community and is available on GitHub.[1]

1.1. Outline

The rest of the article is structured as follows. In Section 2 a complete presentation of the architecture and core components of D^*_+ is provided, while in Section 3 the experimental setups for the algorithmic evaluation are explained. In Section 4 the extended experimental evaluation of the suggested scheme is presented, while future work and conclusions are drawn in Section 5.

2. D^*_+

The D*-lite (Koenig & Likhachev, 2002) algorithm is able to plan a path on a grid ($\mathbb{G}$) where it finds the shortest path $P_{\mathbf{A}\to\mathbf{B}}$ from the point **A** to the point **B**, based on the traversal cost (C) of the voxels (v) in $\mathbb{G}$ so that the $\sum \forall C \in P_{\mathbf{A}\to\mathbf{B}}$ is the smallest possible cost, while another benefit of D*-lite is the way it saves previous path calculations to speed up the re-planning of P.

The main difference between D*-lite and D^*_+ is how $\mathbb{G}$ is constructed from an input map $\mathbb{M}$. In the proposed scheme, D^*_+ is creating $\mathbb{G}$ with the consideration of free, occupied and unknown voxels (v_f, v_o, v_u) and a risk layer. Since the dimensions of $\mathbb{G}$ are changeable, the D^*_+ is also able to plan paths in both 2D and 3D. Thus, by creating $\mathbb{G}$ with only one layer in the z-axis it will enable D^*_+ to plan a 2D path, while if $\mathbb{G}$ is created with multiple layers in the z-axis it will enable for a 3D path. Because D^*_+ is planning P within $\mathbb{G}$, any path in Fig. 1(a) will be a 2D path and any path in Fig. 1(b) will be a 3D path. Thus, is it possible to use the same algorithm to plan both 2D and 3D paths, where the difference is the map $\mathbb{M}$ from which $\mathbb{G}$ is created.

The proposed D^*_+ path planer utilizes three main differences from D*-lite (as presented in Koenig & Likhachev, 2002) namely: (1) the

[1] Link to the code on GitHub: https://github.com/LTU-RAI/Dsp.

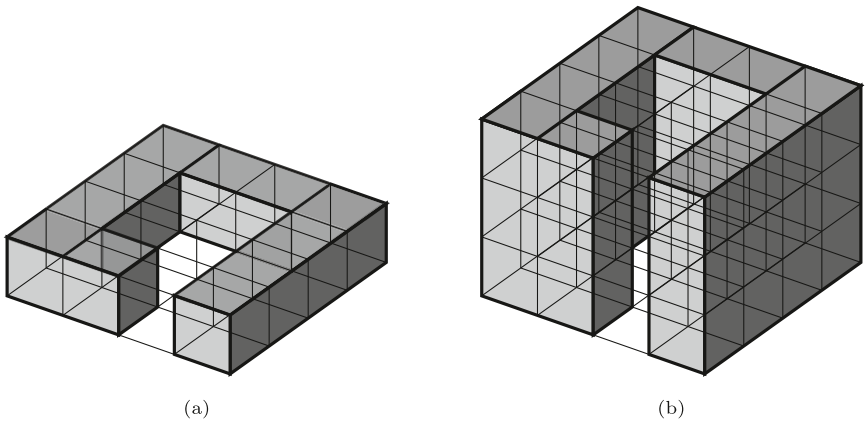


Fig. 1. $\mathbb{G}$ with one z-layer from a $\mathbb{M}_2$ and $\mathbb{G}$ with multiple z-layers from a $\mathbb{M}_3$. Both $\mathbb{M}_2$ and $\mathbb{M}_3$ are created from the same environment.

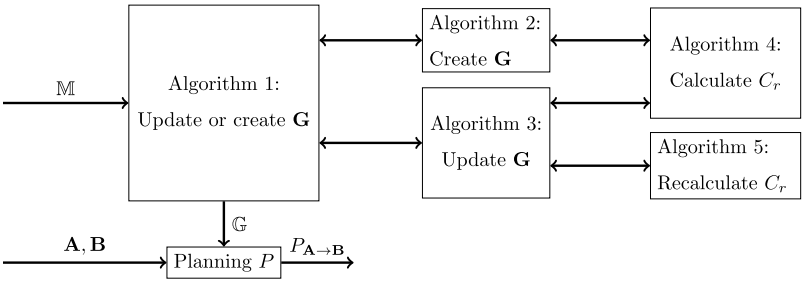


Fig. 2. A block diagram illustrating how the algorithmic blocks in D$^*_+$ are connected. Algorithm 1 receives updated $\mathbb{M}$ and decides if $\mathbb{G}$ should be created from a scratch (Algorithm 2) or updated (Algorithm 3). Algorithms 4–5 are used as help functions for Algorithms 2–3 to generate risk for $\mathbb{G}$.

treatment of unknown voxels, (2) proximity risk and (3) map updates. The rest of this Section describes the proposed algorithm in detail, which can be also summarized in Algorithms 1–5. In Fig. 2 a block diagram illustrating how the algorithms in D$^*_+$ are connected with each other is presented, while it should be mentioned that all the implementations are made in C++ within the Robotic Operating System Stanford Artificial Intelligence Laboratory et al. and are open sourced as indicated before.

2.1. Modeling of unknown space

Imperfections in $\mathbb{M}$ will lead to imperfections in $\mathbb{G}$ that may lead to path short-cutting through walls like the blue path in Fig. 3. In D$^*_+$ this problem is solved by having v_u represented in $\mathbb{G}$, as well as v_f and v_o. Let C_f, C_u, C_o costs assigned to v_f, v_u, v_o respectively so that $C_f < C_u < C_o$, where in such case P will be planned through v_f instead of v_u even though the resulting P length is longer. This results from the core of D$^*_+$ that plans the path where $\sum \forall C \in P_{\mathbf{A}\to\mathbf{B}}$ is the smallest possible. With a higher C_u value the D$^*_+$ will plan the P in v_f at a cost of longer P's than it would make with a smaller C_u.

2.2. Path planning with a proximity risk

D* style path planners are created to plan the shortest path from point **A** to **B**, and the shortest path around a bend, or corner is to hug, meaning traverse as close as possible inside of the corner of interest. As such, the utilization of such planners introduces the issue that the planned path leaves no room for errors in path following and is even setting limitations to the robot's physical size. To deal with this problematic situation, in D$^*_+$ a risk layer is introduced, where v's in the proximity of a v_o are given an extra traversal cost C_r and as such the traversal cost of a v in the proximity of a v_o can be defined as:

$$C_r = \begin{cases} \frac{C_u}{d+1} & \text{if } d < r \\ 0 & \text{else} \end{cases} \tag{1}$$

Fig. 3. A comparison of path planning with (red line) and without (blue line) consideration of v_u under the same input $\mathbb{M}$, starting and endpoint. This comparison does also show the difference between D$^*_+$ and D*-lite (adjusted for online usage with an octomap). With the utilization of v_u the planned path remains in the caving environment, while without the path it becomes invalid while passing through the solid rock wall.

$$C = \begin{cases} C_f + C_r & \text{if } v \text{ is } v_f \\ C_u + C_r & \text{if } v \text{ is } v_u \end{cases} \tag{2}$$

where d is the distance in voxels to the closest v_o and r is the range that is considered as the proximity to a v_o. This novel approach creates a gradient risk that has the highest value next to v_o and decreases with distance as depicted in Fig. 4 where the v with a C_r is depicted.

A direct effect of the proposed D$^*_+$ approach is that in places with sufficient space, this risk layer will enable the planning of a path that is outside of the risk area. However, in the cases where a passage is narrow, the gradient of the risk will cause the path to be planned in

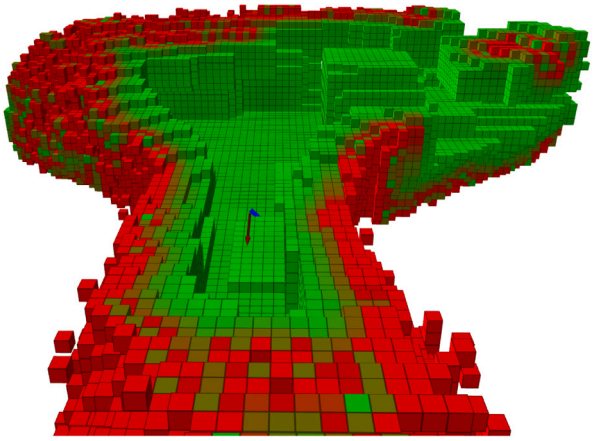


Fig. 4. A visualization of C_r, where red is a high risk and green is a low risk. It should be noted that V_f and v with $C \geq C_u$ are not visualized.

the center of the passage, as illustrated in Fig. 5. As such, this novel addition allows D^*_+ to plan paths in narrow passages and have a larger safety margin in more open areas, especially when compared with similar path planners.

2.3. Continuous map updates and expansions during mission

While the robotic platform is traversing, the mapping software is continuously updating the known map or $\mathbb{M}$, while at the same time it is expanding it to include the newly discovered areas and adding the previously unknown sections, while adjusting the incorrectly registered voxels as indicated in Figs. 6(a) and 6(b).

To accommodate for these changes in $\mathbb{M}$, the $\mathbb{G}$ is created dynamically so that it is capable of expanding and adjusting according to $\mathbb{M}$. In case D^*_+ is deployed in an area where dynamic obstacles exist, the $\mathbb{M}$ and subsequently the $\mathbb{G}$ will be updated and as such, P will be updated accordingly, thus attempting to keep it safe and collision-free. The dynamic expansion of $\mathbb{G}$ in D^*_+ is to guarantee that P will be updated in time to avoid a collision if obstacles are moving or the environment changes. Nevertheless, D^*_+ cannot replace a low-level local reactive obstacle avoidance module, in situations where the expansion time is longer. Even if it would be fast enough is it still recommended to have a reactive safety component in the autonomy stack to ensure safety in the overall robotic mission.

Algorithm 1: Decide if $\mathbb{G}$ will be updated or recreated

```
Input: New map ⁿM
Old map ˡM
Output: grid 𝔾
if size of ⁿM == size of ˡM then
    Algorithm 3 (ⁿM)        ▷ If the same size update existing 𝔾
else
    Algorithm 2 (ⁿM)        ▷ Else create a new 𝔾
end
```

3. Experiments

Three sets of experiments were executed: (a) with the Boston Dynamics Spot robot, used for 2D path planning, (b) with a UAV for 3D path planning (c) comparison of 3D planning using the UAV. The duration of the experiments was varying among the performed experiments between approximately 45s and 680s, where the robots had an average moving velocity of approximately 0.7m/s. To execute the experiments that evaluated the D^*_+'s capability to plan for a safe path

Algorithm 2: Create grid $\mathbb{G}$

```
Input: Map M
Output: Grid 𝔾
g ← new array[size of M : C_u]
for each v in M do
    if v == v_o then
        ▷ If occupied set occupied and add traversal cost to v in
          the proximity
        g[v] ← C_o
        Algorithm 4(v)
    else
        g[v] ← g[v] − C_u
                ▷ If free remove cost for C_u but keep any C_r
OctoGrid ← from g
                                ▷ Connect and search the OctoGrid
𝔾 ← from OctoGrid
```

Algorithm 3: Update occupancy grid $\mathbb{G}$

```
Input: New map ⁿM
Old map ˡM
Output: Updated occupancy grid 𝔾
for each ⁿv in ⁿM do
    if C_{ⁿv} == C_o and C_{ˡv} ≠ C_o then
        𝔾[ⁿv] ← C_{ⁿv}
        Algorithm 4 (ⁿv)
                ▷ Set proximity cost for surrounding voxels
    else if C_{ⁿv} == C_f and C_{ˡv} ≥ C_u then
        if C_{ˡv} == C_o then
            𝔾[ⁿv] ← C_f
            Algorithm 5 (ⁿv)        ▷ Adding C_r if needed
        else
            𝔾[ⁿv] ← C_{ˡv} − C_u
                ▷ Removing C_u without changing C_r
```

Algorithm 4: Calculate C_r for voxels within r from v_o

```
Input: v
Output: Updates in 𝔾
for each i in [−r, r] do
    for each j in [−r, r] do
        for each k in [−r, r] do
            v_r ← 𝔾[v_x + i, v_y + j, v_z + k]
            C_r ← C_u/(i² + j² + k² + 1)
            if C_{vr} == C_o and C_r > C_v then
                𝔾[v] ← C_r
            else if C_{vr} == C_o and C_v ≥ C_u and C_r + C_u > C_v
              then
                𝔾[v] ← C_r + C_u
```

that a robot can traverse, multiple additional components are needed for enabling an overall autonomous robotic mission that indirectly affects the overall performance mission. As such, in the rest of this Section, we will provide a comprehensive overview of the robotic configurations and the software components that were involved in the experimental evaluations.

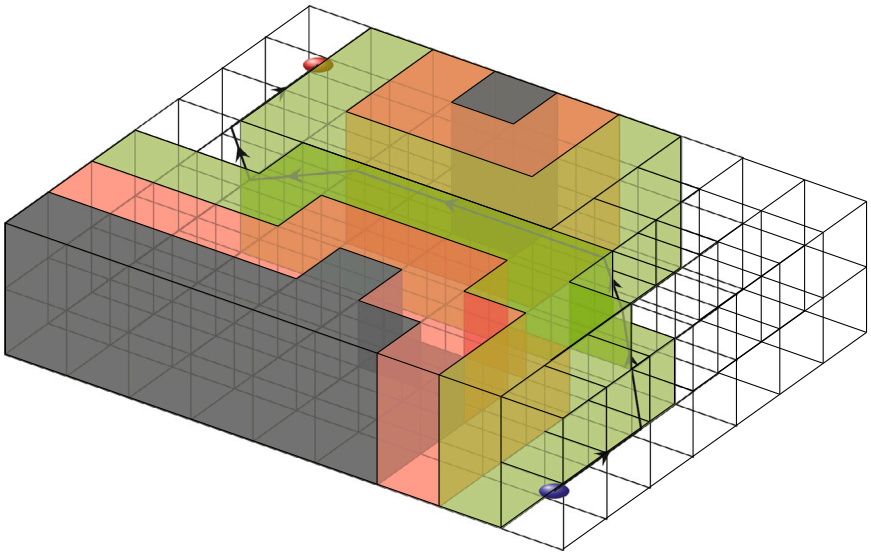


Fig. 5. An illustration of how D^*_+ plans a path from the blue ball to the red ball with regards to the risk layer, red voxels are higher risk, green voxels are lower risk, and black voxels are occupied.

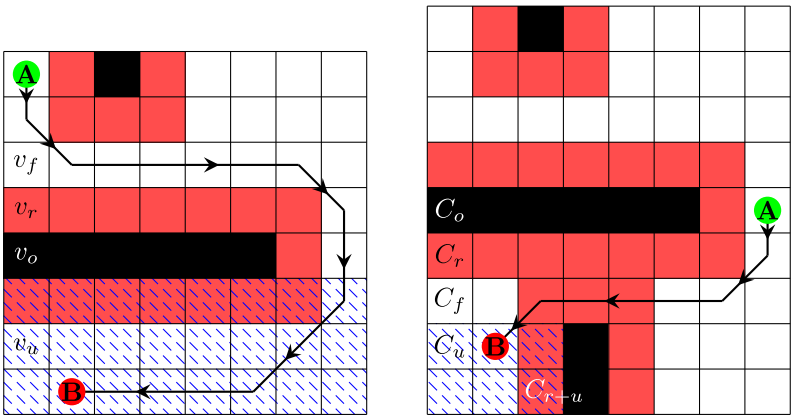


Fig. 6. D^*_+ outcome in planning a path from **A** to **B** in a partially known area where the unknown part is denoted as blue striped. As the robot traverse in the initially planned path, the robot discovers and register more known area and as such the overall path to be executed is online updated.

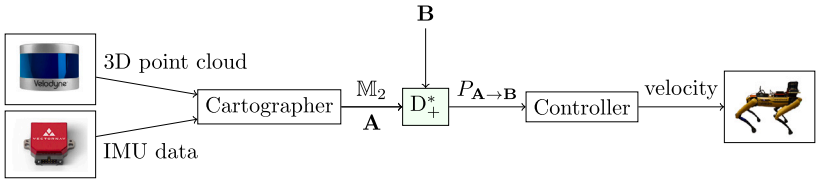


Fig. 7. Software architecture on Spot.

3.1. 2D path planning: The Boston dynamics spot use case

In this work, the 2D path planning capabilities of D^*_+ are tested with the BD Spot robot (see Fig. 8), which is equipped with a 3D lidar, an Inertial Measurement Unit (IMU), and an onboard computer. The software architecture used is depicted in Fig. 7. In this software configuration, the Google Cartographer (Hess, Kohler, Rapp, & Andor, 2016) was utilized is a Simultaneous Localisation and Mapping (SLAM) algorithm being able to create a 2D grid map $\mathbb{M}_2$ and the corresponding state estimation information. The information from the state estimation uses the current position of the robot, as the starting position **A** that together with $\mathbb{M}_2$, and the goal position **B** are used by D^*_+ to plan a path P. Moreover, a position controller takes P and gives corresponding velocity commands to Spot's built-in low-level motion controller that makes Spot move. In the experiments with Spot, a prebuild $\mathcal{M}_2$ of the area was provided and thus allowing for a longer, than the sensor range, P to be planned from the beginning of the mission.

A more complete and exhaustive presentation of utilized Spot's autonomous capability is presented in Koval, Karlsson, and Nikolakopoulos (2022) (see Fig. 8).

Fig. 8. Spot equipped with the attached sensors for the autonomy navigation and computational power. In this view perspective, the Vectornav IMU is placed under the Velodyne lidar and is not visible in the Figure.

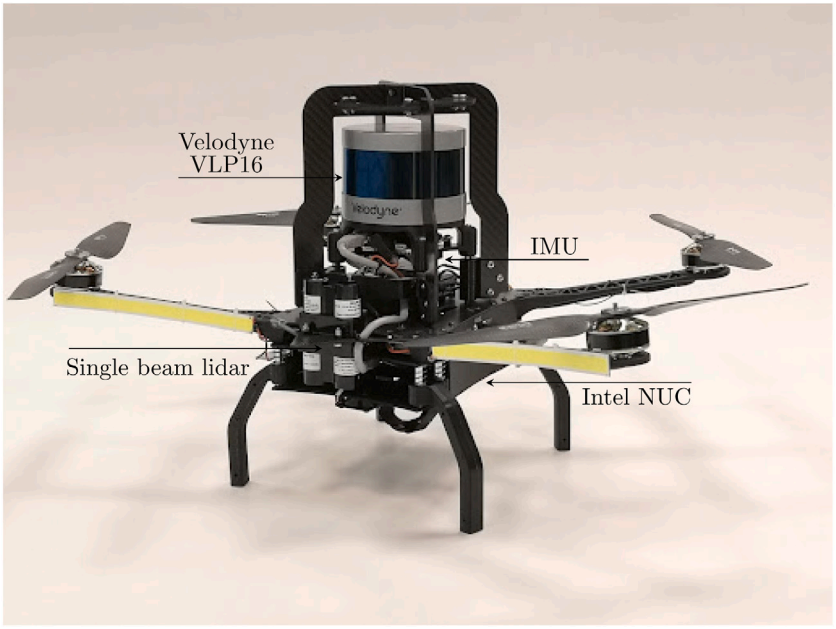


Fig. 9. The UAV used for the 3D path planning experiments.

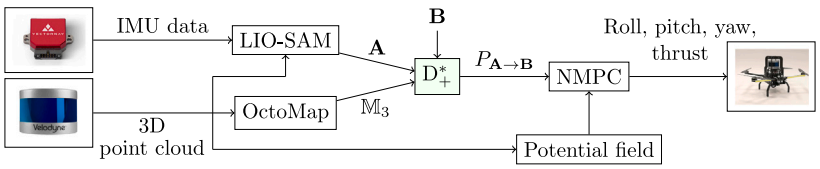


Fig. 10. Software architecture on the UAV.

3.2. 3D path planning: the UAV use case

For the 3D path planning experiments, a custom-built UAV was used as depicted in Fig. 9. In this case, the D^*_+ architecture operates in the 3-dimensional space and requires as input the current position **A**, a goal position **B** and an occupancy map $\mathbb{M}_3$, while it provides the path $P_{\mathbf{A}\to\mathbf{B}}$ as an output. The autonomous navigation of the robot is based on the path generated from D^*_+, which essentially coordinates the motion of the robot, while other modules, such as the state estimation, and a Non-Linear Model Predictive Controller (NMPC) (Lindqvist, Mansouri,

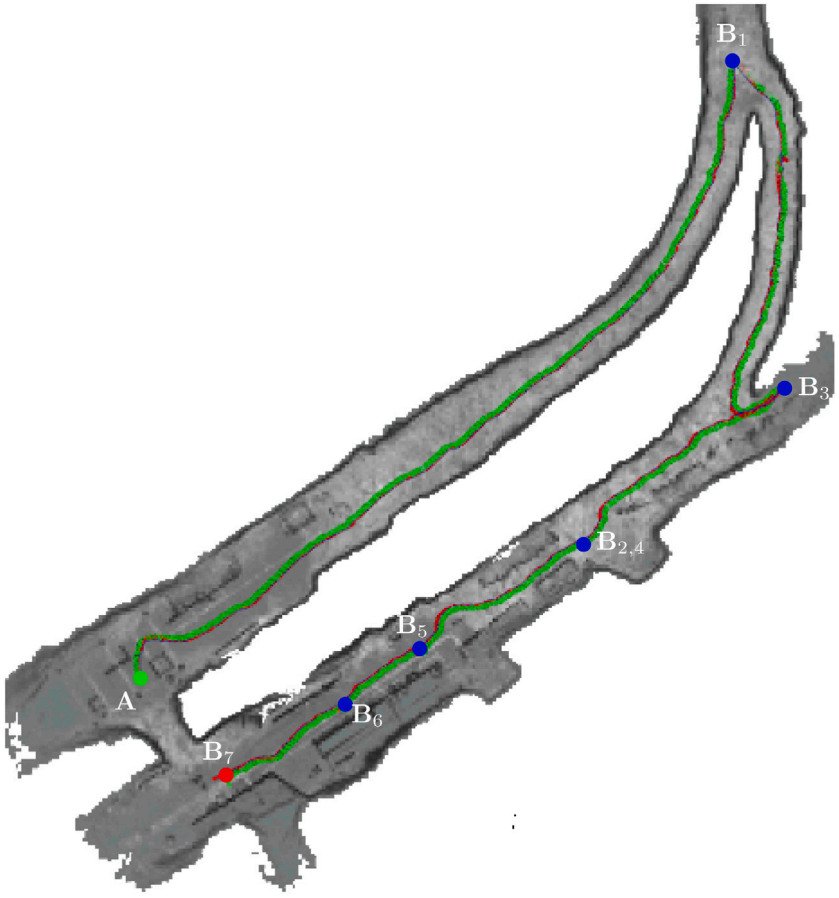


Fig. 11. Sequential **B** were set in a large subterranean tunnel. Spot navigates to each **B** and traverses about 335 m in total. In this experiment, where the map is known a priori, D^*_+ shows an increased capability to plan longer paths and excellent replanning capabilities. In this case, $\mathbf{B}_3$ was set before Spot reached $\mathbf{B}_2$, thus forcing a replanning. When $\mathbf{B}_3$ was reached where a new point $\mathbf{B}_4$ was set at approximately the same point as $\mathbf{B}_2$.

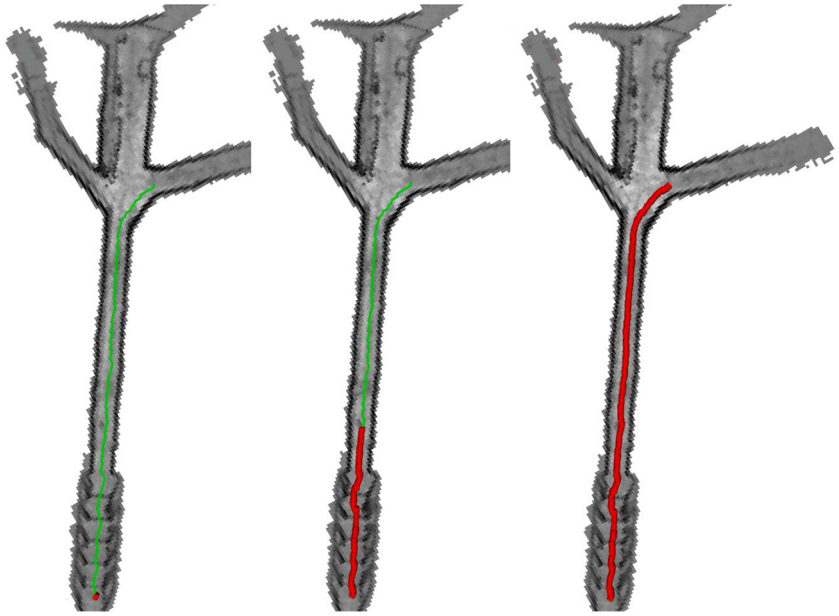


Fig. 12. Spot navigates along the planned path shown in green, and the traversed path shown in red.

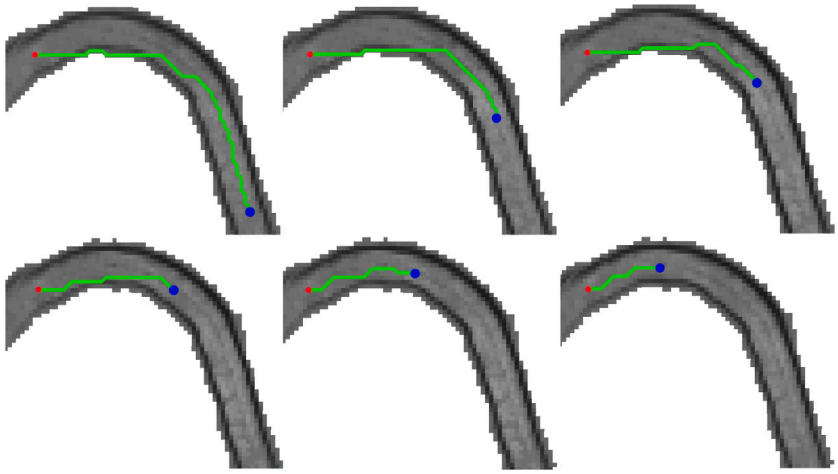


Fig. 13. As Spot traverse the D^*_+ planned path from the blue dot to the red dot, more information on the surrounding environment is provided. Thus the planner updates the current $\mathbb{M}$ causing D^*_+ to reactively re-plan and adjust the overall path.

Algorithm 5: Calculate C_r when a v is discovered to be free

```
Input: v
Output: Updates in 𝔾
C_r ← 0
for each i in [−r, r] do
    for each j in [−r, r] do
        for each k in [−r, r] do
            v_r ← 𝔾[v_x + i, v_y + j, v_z + k]
            if C_vr == C_o then
                C ← c_u/(i² + j² + k² + 1)
                if C > C_r then
                    C_r ← C
𝔾[v] ← C_r
```

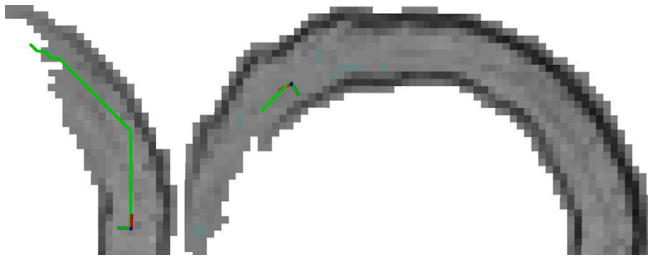


Fig. 14. On the left path planned until the edge of the known map and on the right new path planned towards the goal after a map expansion of the same area. In both cases, the path is denoted by the solid green line.

Haluška & Nikolakopoulos, 2021) that uses an artificial potential field algorithm (Lindqvist, Kanellakis et al., 2021) to reactively avoid obstacles, are part of the overall mission execution. The UAV's primary sensor in the autonomy stack is a Velodyne Puck Lite lidar, which is used alongside an IMU to provide state estimation information on the current position **A**. In this work, the **A** is calculated based on LIO-SAM (Shan et al., 2020), while the 3D occupancy map $\mathbb{M}_3$ is generated based on the Octomap (Hornung et al., 2013) framework.

In these experiments, the artificial potential fields are only used as an extra safety layer to avoid collisions when flying close to obstacles (e.g. less than 50 cm relative distance to an obstacle), where the designed behavior is to prioritize the influence of the potential fields to directly affect the movement since the path, in that case, is considered to be unsafe. Furthermore, the software architecture for the UAV is visualized in Fig. 10. In all the 3D tests, no information was given a-priori, meaning that the UAV explored $\mathcal{M}$ during the mission, thus utilizing both related map updates and expansions. During the experiments, the waypoints were manually set as the mission was evolving and the desired waypoints were reached, while after some time a waypoint was commanded to set a return to the starting location.

3.3. Graph-based Exploration Planner 2.0

The Graph-based Exploration Planner 2.0 was selected to do a comparison with the proposed method because of its proven capabilities in the DARPA subterranean challenge and the possibility to run it in a waypoint navigation mode, which is similar to how the D^*_+ is used. In general, there are considerable differences between Gbplanner and D^*_+, which may bias the evaluation results, depending on the experimental setup. In order to achieve a representative comparison in Gbplanner, the collision model was adjusted to match the UAV size.

4. Experimental results

4.1. 2D path planning

Among the platforms that were used in our experiments, Spot robot has a significantly longer battery life than the UAV and can explore larger subterranean areas. Thus, in this article, the Spot robot was used to evaluate the D^*_+ 2D performance. This allowed us to evaluate path planning in larger maps and over longer distances in comparison with the UAV. Worth noting that the computational load for 2D path planning was significantly lower than for 3D. In this case, the experiments with Spot showed that D^*_+ is able to plan long paths that were approximately 80m at a time and without collisions. One such experiment, where a multiple waypoint mission was executed, is presented in Fig. 11, where a total of 7 waypoints, were reached progressively. in this case, Spot was capable of navigating to each of the waypoints without collisions.

Another experiment was focusing on the use case where Spot had to traverse approximately 80 m in a more narrow tunnel. Fig. 12 shows how Spot can follow the path planned by D^*_+ as it has suitable distances to both walls.

When $\mathbb{M}$ updates it will generate new conditions for D^*_+ to plan paths, therefore D^*_+ will constantly perform an update of the path to keep it safe. In Fig. 13 it is presented an example of how the path may be re-planed as new information about the environment is acquired. As such, the starting path is planned initially inside a wall but as it is

(a) Low obstacle. (b) Obstacle to the side. (c) Pillar in the void.

Fig. 15. Snapshots from three of experiments executed with the UAV.

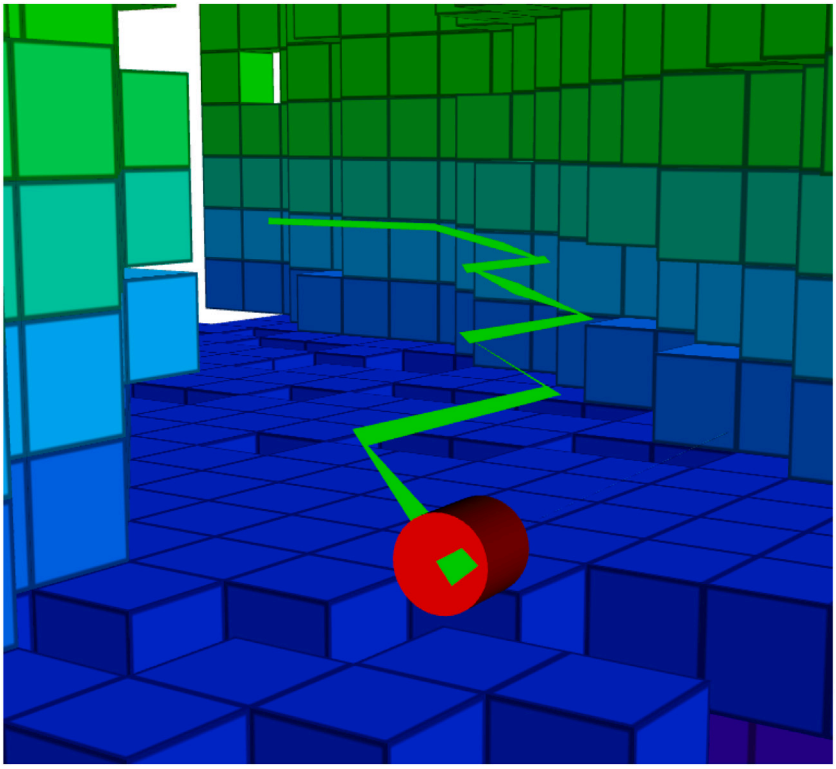


Fig. 16. D^*_+ plans a path around a bend while keeping a suitable distance to the inside wall.

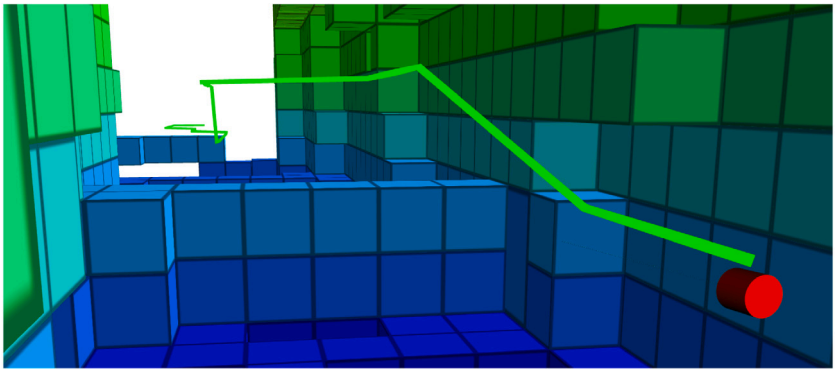


Fig. 17. D^*_+ path planning with a low blockage so that the UAV needs to fly higher to keep a suitable safety marginal to the blockage.

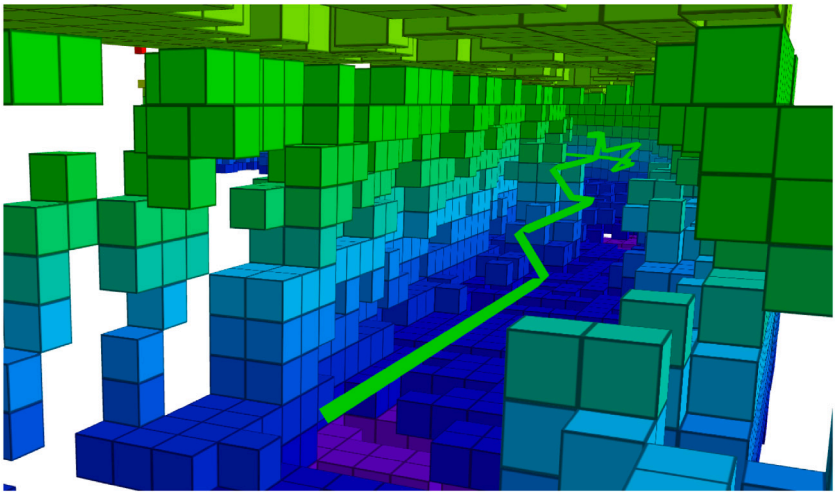


Fig. 18. D^*_+ is capable of planning a path from beyond the line of sight to the end of the green line while having the desired safety margin throughout the whole path.

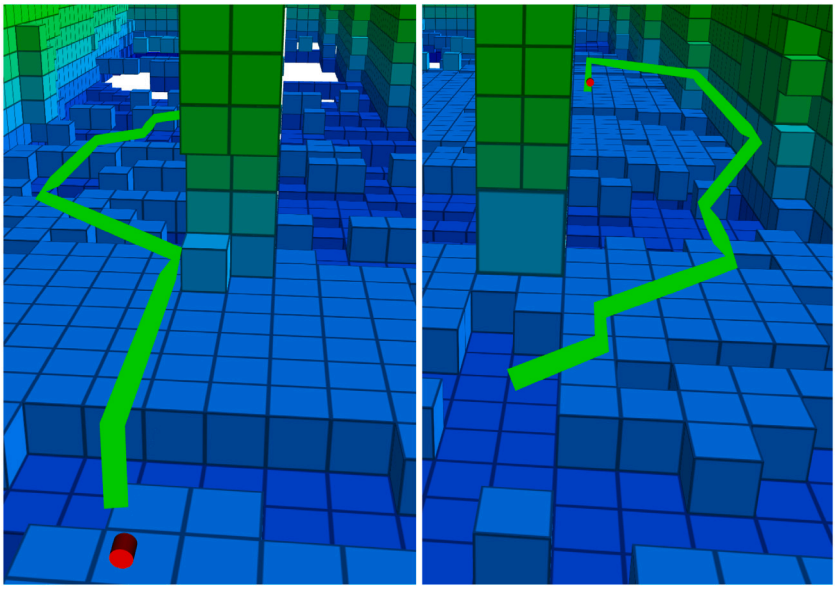


Fig. 19. D^*_+ is tasked to plan a path from the red dot to the end of the green line. The resulting path keeps proper safety margins that are relative to the pillar so that the UAV can traverse it safely.

discovered in the corresponding map updates, the path is interfering with a wall and thus reactively D^*_+ performs a re-planning of the path to stay away from it and in a safe operation.

Another merit of the proposed D^*_+ is the sequential map and graph update capability, which allows planning or re-plan paths towards $\mathbb{B}$ once the robot discovers more parts of the area that navigates along. Fig. 14 depicts the map expansion capability with Spot traversing in the tunnel, wherein the first case the path is planned until the edge of the known map and then a new path is planned in the expanded map of the same area to reach another goal during the mission.

4.2. 3D path planning

D^*_+ 3D planning capabilities were also tested in three experiments in a cave tunnel where it was challenged to plan P along (a) a low blockage as depicted in Fig. 15(a), (b) a tall obstacle to the side making a narrow passage as in Fig. 15(b), and (c) around a pillar as depicted in Fig. 15(c).

In the tunnel corner experiment, the D^*_+ planned P to follow the bend with a suitable safe marginal to the walls. As shown in Fig. 16 D^*_+ is planning a safe path following the tunnel where only the center point of voxels is selected as waypoints in the path, resulting in a zig-zag path.

D^*_+ was also challenged with a low-height obstacle that could potentially risk a collision with the UAV's landing gear if it keeps its current altitude. In this case, P is planned with an altitude change in order to keep the path safe, as depicted in Fig. 17.

At this point, it should be also mentioned that when D^*_+ is tasked to plan a path to a point beyond the line of sight, the algorithm is still capable of providing a path that is inherently safe and short as possible. This performance is depicted in Fig. 18 whereas it is shown, the free space gap between the obstacle to the left and the right wall, was very narrow for the UAV to pass through them, and thus a higher path has been selected.

In the last experimental test, the D^*_+ is tasked to plan a path from one side of a pillar to the other and then back again. This experiment shows that D^*_+ takes the shortest path outside the risky area from the pillar, as also depicted in Fig. 19.

During the experiment, with an obstacle at the side of the tunnel, D^*_+ planned P that was in the center of the opening as shown in Fig. 20.

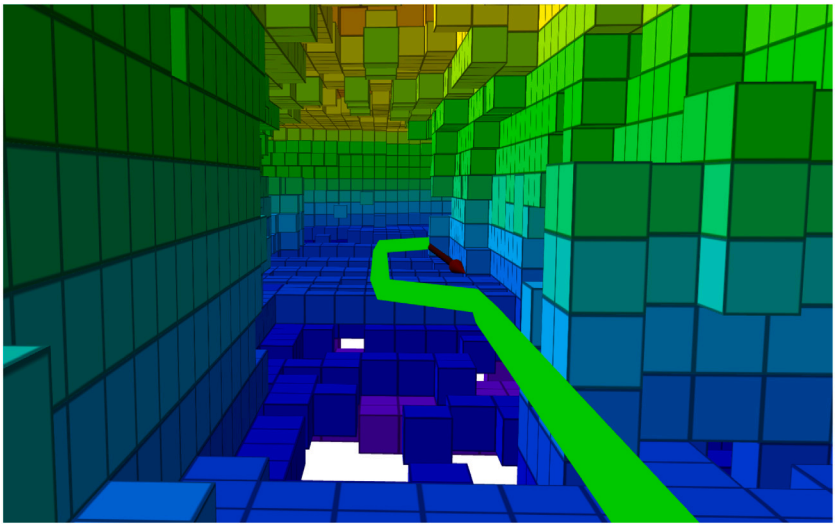


Fig. 20. A UAV avoids colliding with an obstacle on the side by planning P (the green line) in the center of the opening.

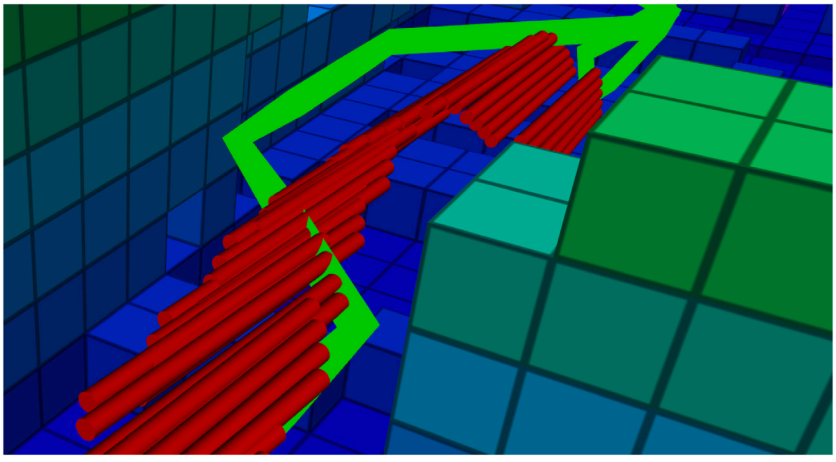
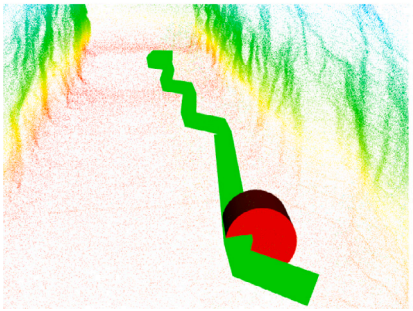
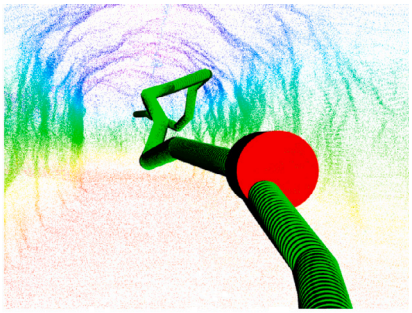


Fig. 21. An instance from the autonomous mission and corresponding path planning when the UAV took a too-tight turn around an obstacle and caused the artificial potential field to avoid a collision. The green line is P and the red bars are the path followed by the UAV.



(a) The path D^*_+ planned (green) when doing the comparison experiments. The path is as straight as voxel-based planning can be.



(b) The path Gbplanner planned (green) when doing comparison experiments. Note how the path is curving up and around for no apparent reason.

Fig. 22. Differences in planned paths between D^*_+ and Gbplanner are visible when both planners plan in the same area under the same conditions.

When passing the obstacle, the remaining planned path converges again to the center of the opening, while keeping a safe distance to all sides.

During all the experimental trials with the UAV, there have been only four times that the potential fields have been triggered to avoid a collision, while two of them were barely noticeable, only just clipping

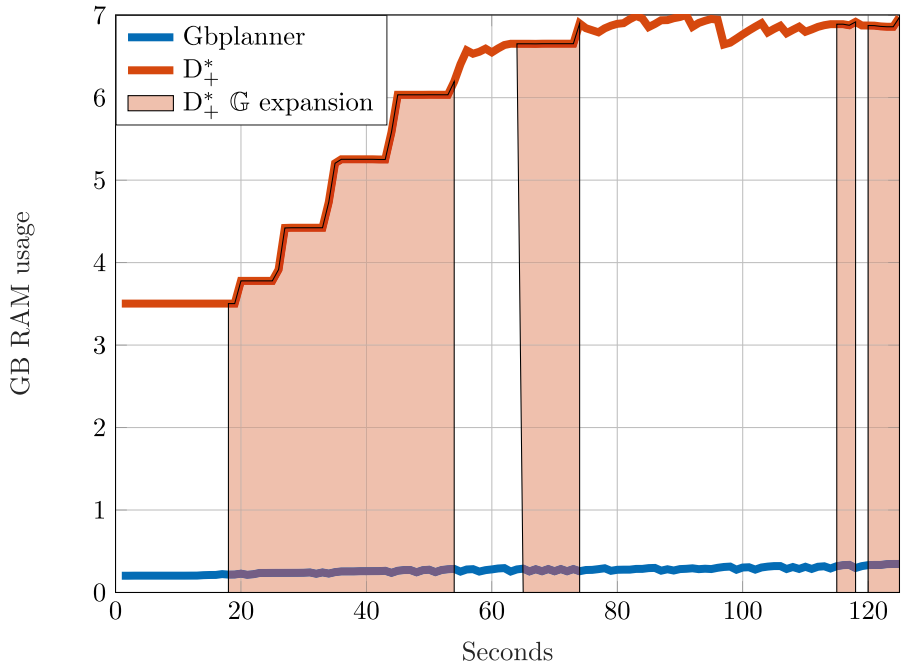


Fig. 23. The memory consumption of D^*_+ and Gbplanner in the comparison experiment. The shaded areas mark the time when D^*_+ performed expansion of $\mathbb{G}$.

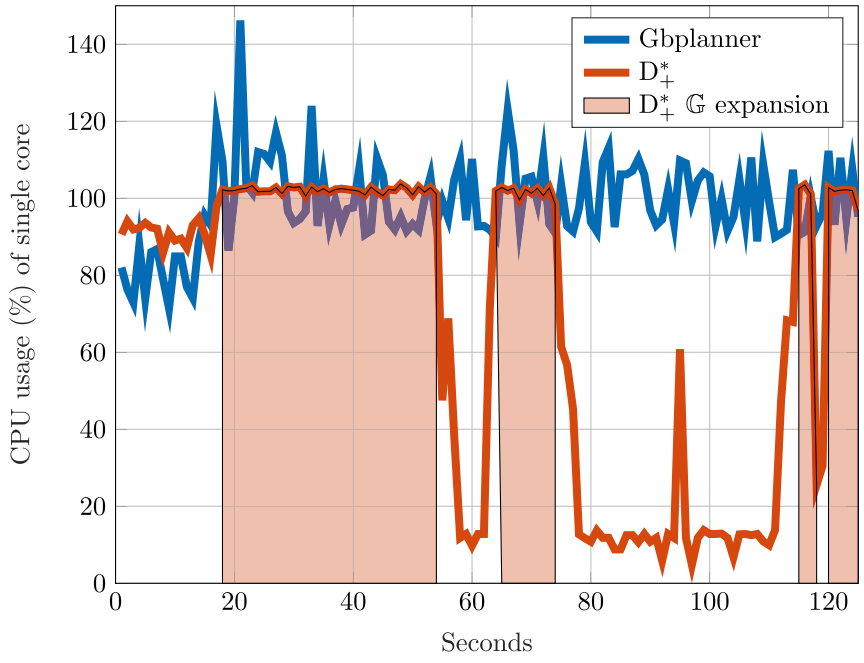


Fig. 24. The CPU load comparison of D^*_+ and Gbplanner in the comparison experiment. The shaded areas mark the time when D^*_+ performed expansion of $\mathbb{G}$.

a corner of the protected area, while it should be noted that: (a) the safety would have been guaranteed even without the potential fields, (b) one triggering was because **B** were set to close to a wall, and (c) the fourth time was during the return on the side obstacle, where P were planned tightly around that. In the final case, the path following had a slightly too long look ahead distance, thus causing it to shortcut the corner in a more intense approach, as it can be seen in Fig. 21. The latest was a dangerous passage close to the obstacle but it was not due to a fault in the D^*_+ planner, rather than a weakness in the path following the autonomy stack. In this case, the easiest solution would have been to increase r so P are planned with the proper safe margins for these types of corner-cutting as well.

4.3. Comparison with Graph-based Exploration Planner 2.0

The experiments used for path planners comparison were carried out in the curved subterranean tunnel in Luleå, Sweden. To evaluate their performance the following metrics were used: CPU and memory usage, as well as an assessment of the planned path.

Both planners have been successfully planning a path to traverse the tunnel but the path characteristics differ significantly (see Fig. 22). D^*_+ is planning the shortest path from voxel to voxel all the way to the end voxel, resulting in a strait path with no unnecessary detours. Gbplanner on the other hand is an RRT-based planner, which chooses a path leading left and right, up and down to reach a target that is

accessible through a straight line. This approach produces less safe paths, like for example when the path goes close to small objects or walls as it is shown in Fig. 22(b).

Based on our field experience, small structures are not guaranteed to be seen by the mapping algorithm so as a safety feature for the UAV we were using the artificial potential field. It is likely that Gbplanner would have succeeded in any way but the tendency to approach walls and small objects were increasing the risk of collisions. Moreover, these unnecessary movements will reduce UAV's battery life and shorten the maximum distance of the mission.

When we are considering the computational load, the proposed D^*_+ is mainly utilizing the CPU for building the graph, which is correlated with the memory usage increases as depicted in Fig. 23 and Fig. 24. During planning and the $\mathbb{G}$ updating, the CPU load is at approximately 10 % of a single core. When it comes to resource efficiency, D^*_+ is prone to high memory consumption. For example, in the evaluation test, D^*_+ used 3.5 – 7.0 GB of RAM, while Gbplanner used only 0.2 – 0.3 GB of RAM, as shown in Fig. 23. D^*_+ capability to replan a path on the fly cannot be compared to Gbplanner because it does not provide an equivalent feature. Some of the differences in performance can be explained by differences in those types of features that in some cases affects the measurable performance (see Fig. 24).

5. Conclusions

In this article, a risk-aware, platform-agnostic path planner called D^*_+ has been established. As it was presented, D^*_+ utilizes a proximity risk layer to generate a safety margin for any occupied space. The introduced combination of proximity risk and unknown space treatment is able to solve the common issue of shortcuts due to imperfections in the map. By allowing the map to update and expand, the proposed framework is capable of exploring and operating in a previously unknown environment. Finally, the efficiency of the D^*_+ has been tested in challenging real-world scenarios with both 2D and 3D planning and it has been proven to reliably plan safe paths in all the evaluated use cases independently of the utilized platform.

Considering large-scale and long-term missions, a potential bottleneck of D^*_+ is its computation time when $\mathbf{G}$ is large, and as such future work should aim to improve the computation time for large $\mathbf{G}$ by considering for example map segmentation approaches. Additionally, another point of D^*_+ that should be addressed is the severe memory usage. The comparison with Gbplanner showed that it is possible to do memory-efficient path planning that does not require more computation power. This part needs to be investigated and addressed for D^*_+ while maintaining the major components of the short straight paths and the dynamic re-planning. The above-mentioned shortcoming in general is related to the resources required to compute P. One direction for lowering memory usage and possibly saving some computation time is to change the data structure of the internal grid map $\mathbf{G}$. Currently, $\mathbf{G}$ is stored as an array in D^*_+ but if it is changed to an oct-tree it is possible that memory usage and computation time can be lowered. Another improvement that can be considered is more types of risks, such as wind or slippery surfaces.

CRediT authorship contribution statement

Samuel Karlsson: Conceptualization, Methodology Software, Experiments, Writing – original draft, Writing – review & editing. **Anton Koval:** Methodology, Experiments, Writing – original draft, Writing – review & editing. **Christoforos Kanellakis:** Experiments, Writing – original draft, Writing – review & editing, Supervision. **George Nikolakopoulos:** Resources, Experiments, Writing – original draft, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: George Nikolakopoulos reports financial support was provided by Horizon 2020.

Data availability

The source code is open source at https://github.com/LTU-RAI/Dsp.

References

Agha, A., Otsu, K., Morrell, B., Fan, D. D., Thakker, R., Santamaria-Navarro, A., et al. (2021). Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. arXiv preprint arXiv:2103.11470.

Atapour-Abarghouei, A., & Breckon, T. P. (2018). A comparative review of plausible hole filling strategies in the context of scene depth image completion. *Computers & Graphics*, *72*, 39–58. http://dx.doi.org/10.1016/j.cag.2018.02.001, URL https://www.sciencedirect.com/science/article/pii/S0097849318300219.

Cuevas, L., Ramirez, M., Shames, I., & Manzic, C. (2021). Path planning under risk and uncertainty of the environment. In *2021 American control conference* (pp. 4231–4236). http://dx.doi.org/10.23919/ACC50511.2021.9483405.

Dang, T., Tranzatto, M., Khattak, S., Mascarich, F., Alexis, K., & Hutter, M. (2020). Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, *37*(8), 1363–1388, Wiley Online Library.

DARPA (2020). DARPA subterranean (SubT) challenge. URL https://www.subtchallenge.com/, (Accessed February 2021).

Hakobyan, A., Kim, G. C., & Yang, I. (2019). Risk-aware motion planning and control using CVaR-constrained optimization. *IEEE Robotics and Automation Letters*, *4*(4), 3924–3931.

Hayat, S., Yanmaz, E., Bettstetter, C., & Brown, T. X. (2020). Multi-objective drone path planning for search and rescue with quality-of-service requirements. *Autonomous Robots*, *44*(7), 1183–1198. http://dx.doi.org/10.1007/s10514-020-09926-9.

Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016). Real-time loop closure in 2D LIDAR SLAM. In *2016 IEEE International conference on robotics and automation* (pp. 1271–1278).

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, http://dx.doi.org/10.1007/s10514-012-9321-0, Software available at URL https://octomap.github.io.

Hu, X., Pang, B., Dai, F., & Low, K. H. (2020). Risk assessment model for UAV cost-effective path planning in urban environments. *IEEE Access*, *8*, 150162–150173. http://dx.doi.org/10.1109/ACCESS.2020.3016118.

Huang, Z., Schwarting, W., Pierson, A., Guo, H., Ang, M., & Rus, D. (2020). Safe path planning with multi-model risk level sets. In *2020 IEEE/RSJ International conference on intelligent robots and systems* (pp. 6268–6275). http://dx.doi.org/10.1109/IROS45743.2020.9341084.

Koenig, S., & Likhachev, M. (2002). D* lite. *Aaai/Iaai*, *15*.

Koval, A., Karlsson, S., & Nikolakopoulos, G. (2022). Experimental evaluation of autonomous map-based spot navigation in confined environments. *Biomimetic Intelligence and Robotics*, Article 100035.

Kulkarni, M., Dharmadhikari, M., Tranzatto, M., Zimmermann, S., Reijgwart, V., De Petris, P., et al. (2022). Autonomous teamed exploration of subterranean environments using legged and aerial robots. In *2022 International Conference on Robotics and Automation (ICRA)* (pp. 3306–3313). http://dx.doi.org/10.1109/ICRA46639.2022.9812401.

Laconte, J., Kasmi, A., Pomerleau, F., Chapuis, R., Malaterre, L., Debain, C., et al. (2021). A novel occupancy mapping framework for risk-aware path planning in unstructured environments. *Sensors*, *21*(22), 7562.

Li, F., Zlatanova, S., Koopman, M., Bai, X., & Diakité, A. (2018). Universal path planning for an indoor drone. *Automation in Construction*, *95*, 275–283. http://dx.doi.org/10.1016/j.autcon.2018.07.025, URL https://www.sciencedirect.com/science/article/pii/S0926580517311184.

Lindqvist, B., Kanellakis, C., Mansouri, S. S., akbar Agha-mohammadi, A., & Nikolakopoulos, G. (2021). COMPRA: A compact reactive autonomy framework for subterranean mav based search-and-rescue operations. arXiv:2108.13105.

Lindqvist, B., Mansouri, S. S., Haluška, J., & Nikolakopoulos, G. (2021). Reactive navigation of an unmanned aerial vehicle with perception-based obstacle avoidance constraints. *IEEE Transactions on Control Systems Technology*, 1–16. http://dx.doi.org/10.1109/TCST.2021.3124820.

Mishra, B., Garg, D., Narang, P., & Mishra, V. (2020). Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, *156*, 1–10. http://dx.doi.org/10.1016/j.comcom.2020.03.012, URL https://www.sciencedirect.com/science/article/pii/S0140366419318602.

Ono, M., Fuchs, T. J., Steffy, A., Maimone, M., & Yen, J. (2015). Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In *2015 IEEE Aerospace conference* (pp. 1–10). http://dx.doi.org/10.1109/AERO.2015.7119022.

Primatesta, S., Capello, E., Antonini, R., Gaspardone, M., Guglieri, G., & Rizzo, A. (2017). A cloud-based framework for risk-aware intelligent navigation in urban environments. In *2017 International Conference on Unmanned Aircraft Systems* (pp. 447–455). http://dx.doi.org/10.1109/ICUAS.2017.7991358.

Primatesta, S., Guglieri, G., & Rizzo, A. (2019). A risk-aware path planning strategy for UAVs in urban environments. *Journal of Intelligent & Robotic Systems*, *95*(2), 629–643. http://dx.doi.org/10.1007/s10846-018-0924-3.

Puck, L., Schnell, T., Plasberg, C., Büttner, T., Heppner, G., Rönnau, A., et al. (2020). Modular, risk-aware mapping and fusion of environmental hazards. In *2020 IEEE 23rd International conference on information fusion* (pp. 1–6). IEEE.

San Juan, V., Santos, M., & Andújar, J. M. (2018). Intelligent UAV map generation and discrete path planning for search and rescue operations. *Complexity*, *2018*, Article 6879419. http://dx.doi.org/10.1155/2018/6879419.

Schedl, D. C., Kurmi, I., & Bimber, O. (2021). An autonomous drone for search and rescue in forests using airborne optical sectioning. *Science Robotics*, *6*(55), eabg1188. http://dx.doi.org/10.1126/scirobotics.abg1188, arXiv:https://www.science.org/doi/pdf/10.1126/scirobotics.abg1188.

Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., & Daniela, R. (2020). LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International conference on intelligent robots and systems* (pp. 5135–5142). IEEE.

She, R., & Ouyang, Y. (2021). Efficiency of UAV-based last-mile delivery under congestion in low-altitude air. *Transportation Research Part C (Emerging Technologies)*, *122*, Article 102878. http://dx.doi.org/10.1016/j.trc.2020.102878, URL https://www.sciencedirect.com/science/article/pii/S0968090X20307786.

da Silva Arantes, M., Toledo, C. F. M., Williams, B. C., & Ono, M. (2019). Collision-free encoding for chance-constrained nonconvex path planning. *IEEE Transactions on Robotics*, *35*(2), 433–448.

Stanford Artificial Intelligence Laboratory, et al. 2018. Robotic Operating System URL https://www.ros.org.

Tordesillas, J., & How, J. P. (2021). FASTER: Fast and safe trajectory planner for navigation in unknown environments. *IEEE Transactions on Robotics*.

Tordesillas, J., Lopez, B. T., & How, J. P. (2019). FASTER: Fast and safe trajectory planner for flights in unknown environments. In *2019 IEEE/RSJ International conference on intelligent robots and systems*. IEEE.

Wang, L., Ye, H., Wang, Q., Gao, Y., Xu, C., & Gao, F. (2021). Learning-based 3D occupancy prediction for autonomous navigation in occluded environments. arXiv:2011.03981.

Wu, X., Xu, L., Zhen, R., & Wu, X. (2019). Biased sampling potentially guided intelligent bidirectional RRT Algorithm for UAV path planning in 3D environment. *Mathematical Problems in Engineering*, *2019*, Article 5157403. http://dx.doi.org/10.1155/2019/5157403.

Xiong, J., Gao, H., Wang, M., Li, H., & Lin, W. (2021). Occupancy map guided fast video-based dynamic point cloud coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 1. http://dx.doi.org/10.1109/TCSVT.2021.3063501.

Yan F., . X. J. (2013). Path planning in complex 3D environments using a probabilistic roadmap method. *International Journal of Automation and Computing*, *10*, 525–533. http://dx.doi.org/10.1007/s11633-013-0750-9.

Zammit, C., & Kampen, E.-J. V. Comparison between a* and RRT algorithms for UAV path planning. In *2018 AIAA Guidance, navigation, and control conference*. http://dx.doi.org/10.2514/6.2018-1846, arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2018-1846.

Zhang, N., Zhang, M., & Low, K. H. (2021). 3D path planning and real-time collision resolution of multirotor drone operations in complex urban low-altitude airspace. *Transportation Research Part C (Emerging Technologies)*, *129*, Article 103123. http://dx.doi.org/10.1016/j.trc.2021.103123, URL https://www.sciencedirect.com/science/article/pii/S0968090X2100142X.

Zhou, B., Pan, J., Gao, F., & Shen, S. (2021). Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, *37*(6), 1992–2009.