\textbf{D}^*_{+} : A Generic Platform-Agnostic and Risk-Aware Path Planning Framework with an Expandable Grid.

Samuel Karlsson\textsuperscript{1}, Anton Koval\textsuperscript{1}, Christoforos Kanellakis\textsuperscript{1}, Ali-akbar Agha-mohammadi\textsuperscript{2}, George Nikolakopoulos\textsuperscript{1}

\textbf{Abstract}—This article establishes a novel generic and platform-agnostic risk-aware path planning framework that is based on the classical $D^*$ lite planner with a path design focus on safety and efficiency. The planner generates a grid map where the occupied/free/unknown spaces are represented with different traversal costs. As it will presented, in this case, a traversal cost is added to the unknown voxels that are close to an occupied one. The algorithmic implementation is also enhanced with a dynamic grid map that has the novel ability to update and expand during the robotic operation and thus increase the overall safety of the mission and it is suitable for exploration and search and rescue missions. On the generated grid map, the $D^*$ lite is able to plan a safer path that has a minimum traversal cost. The proposed path planning framework is suitable for generating 2D and 3D paths, for ground and aerial robots respectively and thus in the 3D case, the grid is created with one voxel height to plan for a 2D path, which is the main factor that differentiates between 2D and 3D path planning. The efficacy of the proposed novel path planning scheme is extensively evaluated in multiple simulation and real-world field experiments on both a quadcopter platform and the Boston Dynamics Spot legged robot.

I. INTRODUCTION

With the continuous development of the aerial and ground robotic technologies, the number of the related application areas is constantly increasing. Many of these applications require the robots to navigate in reactive navigation schemes and based on longer term of online and sequential waypoint generation configurations in a safe and efficient way. In these cases, the safety can be induced from staying away from structures and obstacles, while the efficiency can be indicated through the selection of interpolated waypoints that demonstrate a shorter distance, a low energy consumption, etc. Within the DARPA Sub-T challenge \cite{1} online sequential waypoint navigation is required for large-scale subterranean environments, while this work is part of the development efforts within the COSTAR team \cite{2}, \cite{3}. Other application areas are related to outdoor open space environments for the case of the last-mile delivery \cite{4}.

For a specific area of interest for the robotic deployment, it is always beneficial for the path planners and the overall navigation scheme, to be able to utilize the area’s specific characteristics, e.g. the utilization of darkness in cave environments \cite{5} or the deepest point contours as in \cite{6}. In this approach, the utilization of area-specific characteristics may result in significant improvements in the overall mission performance, but cannot be generalised to a greater spectrum of different area morphologies.

Towards the creation of safe paths, in a collision free with the environment consideration, it is very common in the robotics community to utilize robotic kinematic model constraints during the task of path planning as in \cite{7}, \cite{8}, \cite{9}. However, in this case, the disadvantage with such planers is that the kinematic model has to be a priori known and be very accurate to the specific robot utilized. An alternative option is to use heuristic path planers, as the one in \cite{10} to generate safe inherent paths. In this case, the heuristic path planers are generally not optimal but they are designed to work with limited information from the available knowledge of the surroundings and in general could be a very good option for ensuring a safe navigation. In the quest of developing a more generic-purpose path planers, multiple navigation approaches are incorporating in their path planning information from a priori global known map of the area in operation, or a smaller pre-build online local map that is generated during the operation of the robot \cite{11}, while such maps could be utilized in combination with a model-based or a heuristic planers.

In this article, we propose a novel grid-based mapping approach, the $D^*_+ \text{ planner}$, that has the merit to enable different variations of $D^*$ \cite{12}, \cite{13} or theta$^*$ \cite{14} in a generic platform agnostic approach, while incorporating a risk aware map and expandable grids of variable size for large scale path planning realisations. The proposed $D^*_+$ path planner is structured around a grid with occupied and free voxels that could be very well explored, fast and reliably, based on the algorithmic foundations on the $D^*$ lite \cite{15}. One challenge with this type of grid-based planners, is the generation of the grid itself, the need for constant grid update during the robot operation could potentially suffer from blind spots and sensor imperfections and eventually lead to mission failures. To overcome such bottlenecks, the classical approaches are forcing the path planning algorithms to update the a priori known map according to newly obtained sensorial information. In case of a grid utilization, for these updates to be feasible, it is needed to consider and retain a dynamic map, a feature that is missing from the implementation of $D^*$ lite.
One of the most popular grid mapping algorithms is OctoMap [16], an octree [17] based mapping algorithm that incorporates a tree representation of a 3D occupancy grid that is space-efficient and fast. One benefit with map updates is that the related holes/gaps and the map inaccuracies can be fixed/filled during the algorithmic operation, and as such, an initially bad path can be re-planned, which in the different case of non-updating can also result in an issue and overall mission failure. Especially for the case of exploration or search and rescue robotic missions, where the robots are constantly exploring an area and creating a growing map continuously, such map expansions are essential for allowing the overall proper functionality of the path planning frameworks. To enable a 2D path planing, an occupancy grid is needed [18] (alternative this is called framework. To enable a 2D path planing, an occupancy grid is needed [18] (alternative this is called x-ray map in the cartographer ROS framework [19]. In general cartographer is providing the ability for performing a relocalization and a 3D mapping that are useful in some scenarios, but it does not provide a useful 3D map for grid creation but only a useful 2D map.

Another way to decrease the issue of invalid shortcuts in robotic navigation, due to map imperfection, is to take the unknown area and discourage planned paths through unknown areas, with a higher traversability cost. In this case, an additional challenge is that these algorithms are not taking under consideration the size of the robot and can therefore plan paths that are very close to walls and other artifacts that indicate a rather weird and in most of the cases dangerous for a collision overall robotic navigation. Solutions to this problem have been suggested by performing multiple and online collision checks in the proximity of the path [20], under the requirement that Hoppasthis approach requires some knowledge of the robot’s size. An additional option, is to inflate the obstacle in the 2D or 3D space, so that the path planner perceives the obstacle as larger, and thus creates an inherent safety marginal [21] during the path planning.

However, by only adding a safety margin in the path planning is not always a desired method and this could result in local minima trapping situations and sub-optimal suggested paths or not complete paths. In many operational scenarios, there are situations where it could be desirable for the robot to take a moderate risk to get through a narrow passageway, but in general when more free space exists, larger marginals are desired. In this case, one way to achieve this slightly dynamic behavior, is to assign a different traversal cost to different areas of the map and thus create a risk-aware planner [22], [23]. These risk-aware planners are however model dependent and thus suffers from the non ability to be transferred to another robot. Traversal costs have been also used in combination with other planners as well, for example the Rapidly exploring Random Tree [24], while in general, with any traversal cost-based planner it comes also the challenge of creating/assigning the traversal cost to a map. If the traversal cost is based on the risk of traversal, then this traversal cost is a measurement of the related risk and thus the risk maps are generally constructed, as the works in [25], [26] that can produce good cost maps. In this case, the drawback is that different robots have different capabilities to tolerate different hazards and generalisations are hard to be achieved. As an example, ground and aerial robots have vastly different risks, friction, while surface traction is a concern for ground robots where aerial robots are totally independent of that, however for both cases, obstacles and the related proximity to them are a common risk factor and are commonly treated as risky areas. A cost map that is enhanced with constraints on the proximity to the obstacles can be more sufficient for ensuring a safe and optimal navigation and it can work for both ground and aerial robots, something that the classical path planning methods are not capable of [25], [26]. Thus, applying the overall concept of a risk-aware path planner that can with a minimal amount of work be implemented in a platform agnostic approach and in the most common environments is the basic idea and the area of contribution of this article.

In this article the main contribution stems from establishing the novel $D^*_\rightarrow$ architecture for a generic, platform-agnostic, and risk-aware path planning based on $D^* \text{ lite}$. Towards this contribution, with the proposed $D^*_\rightarrow$ framework we are introducing a novel grid planning approach with an additional risk layer that increases the traversal cost ($C_r$) for voxels in the proximity of occupied voxels ($v_o$). We are also introducing a novel grid representation that it is created with three distinct states, free, occupied, and unknown-voxels ($v_f, v_o, v_u \in \mathbb{R}^3$) that are represented with different traversal costs ($C_f, C_o, C_u$ combined with $C_r$) in the grid. Finally, we are proposing a novel approach where the grid creation is performed from the dynamic maps (occupancy grid [18] and octotree [16] for 2D and 3D planning respectively) that enables continuous and stable seamless map updates and expansion, specifically oriented towards large scale robotic exploration missions. In Figure 1 the overall concept of the $D^*_\rightarrow$ is depicted with the utilized costs and overall planning philosophy. The rest of the article is structured as it follows.

In Section II, the risk aware and expandable $D^*_\rightarrow$ framework is being established, including a detailed description of the grid creation, the occupancy grid update, the risk map generation, the re-planning and overall expansion. In Section III, multiple simulation and experimental results are presented that prove the efficacy of the proposed scheme, while in Section IV, the conclusions are drawn. Finally, we should state that the proposed package in this article has been thoroughly tested both in simulation and real-life experiments, while it is publicly available and open-source:

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

II. RISK-AWARE AND REPLANNING EXPANDABLE $D^*_\rightarrow$ FRAMEWORK

In a robot navigation scenario, when internally representing the input map ($M$) in a grid representation ($G$, where $\forall v \in G$), the planning in either 2D or 3D is related to the number of layers in the $z$-axis. By using only one layer in the
includes either occupied voxels $v_f$ (white), occupied voxels $v_o$ (black) and unknown voxels $v_u$ (blue dotted). The gray voxels $v_r$ represent the added traversal cost for proximity $C_r$ to $v_u$, darker gray is a higher traversal cost $C_f + C_u$ due to the combination of unknown and proximity. In this case is $D^*_v$ planning a path from the green circle to the red circle along the arrow path.

$z$-axis to represent the free, unknown, and occupied space, a path in 2D will be planned, when using multiple layers in the $z$-axis a 3D path will be planned. The proposed $D^*_v$ is structured around this concept and is capable to provide either 2D and 3D paths by only changing the input $M$, the 2D occupancy grid ($M_2$) for 2D planning and the octree ($M_3$) for the 3D planning.

$D^*_v$ plans the shortest path $P_{A \rightarrow B}$ between voxel ‘A’ and ‘B’ in $G$ based on the traversal costs $C$ so that $P_{A \rightarrow B}$ is the path in $G$ where the overall traversal cost $\sum C \in P_{A \rightarrow B}$ is the smallest possibly. The proposed $D^*_v$ path planner utilizes three main differences from $D^*$ lite (as presented in [12]) namely: 1) the treatment of unknown voxels, 2) proximity risk, and 3) map updates. The rest of this section describes the proposed novelties in detail, while summarized in algorithms [1] - [5]. All implementation is made in C++ within the Robotic Operating System [27].

### Algorithm 1: Decide if $G$ will be updated or recreated

1. **Input:** New map $^nM$
2. **Old map $^lM$**
3. **Output:** Grid $G$
4. **if size of $^nM$ == size of $^lM$ then**
   5. Algorithm 3\left(\(^nM\)\) // If the same size update existing $G$
5. else
6. Algorithm 2\left(\(^nM\)\) // Else create a new $G$

### A. The Unknown Voxel

Traditionally, a $D^*$ grid is build using a binary map that includes either occupied voxels $v_o$ or free voxels $v_f$. To avoid invalid $P$’s short-cutting through walls, due to imperfection

in sensors or map sparsity, a third main type of voxel is introduced, the unknown voxels $v_u$. Each $v$ is assigned with a traversal cost $C$, where $v_o$ should have the max value $C_o$ and $v_f$ should have a minimum $C_f$. The traversal cost $C_u$ for $v_u$ can be set in between $C_f$ and $C_o$, where the value depends on the navigation scenario, by assigning $C_u$ to a low value for exploration behavior or to a higher value for safely following known predefined paths.

The benefit of using $v_u$ can be seen in Figure 2 where $D^*$ (Figure 2a) lite plans an invalid path through what actually is a solid rock. By using $D^*_v$ the path is planned alongside the tunnel (Figure 2b) and with sufficient safety marginal, when both planners used the same input map $M$.

### B. The Proximity risk

Originally, $D^*$ lite plans paths close to $v_o$ to get the shorter path without considering safety aspects that can cause issues in real-life robot deployment e.g. the physical robot size can potentially collide with the surrounding environment for such
Algorithm 4: Calculate $C_r$ for voxels within $r$ from $v$

1. **Input:** $v$
2. **Output:** Updates in $\mathcal{G}$
3. **for each** $i$ in $[-r, r]$ **do**
4.   **for each** $j$ in $[-r, r]$ **do**
5.     **for each** $k$ in $[-r, r]$ **do**
6.     $v_r \leftarrow \mathcal{G}[v_r + i, v_y + j, v_z + k]$
7.     $C_r \leftarrow c_u/(i^2 + j^2 + k^2 + 1)$
8.     **if** $v_r = v_0$ **and** $C_r > C_v$ **then**
9.     $\mathcal{G}[v] \leftarrow C_r$
10. **else if** $v_r = v_0$ **and** $C_r \geq C_u$ **and** $C_r + C_v > C_u$ **then**
11. $\mathcal{G}[v] \leftarrow C_r + C_u$

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

1. **Input:** $v$
2. **Output:** Updates in $\mathcal{G}$
3. $C_r \leftarrow 0$
4. **for each** $i$ in $[-r, r]$ **do**
5.   **for each** $j$ in $[-r, r]$ **do**
6.     **for each** $k$ in $[-r, r]$ **do**
7.     $v_r \leftarrow \mathcal{G}[v_r + i, v_y + j, v_z + k]$
8.     **if** $v_r = v_0$ **then**
9.     $C \leftarrow c_u/(i^2 + j^2 + k^2 + 1)$
10. **if** $C > C_r$ **then**
11. $C_r \leftarrow C$
12. $\mathcal{G}[v] \leftarrow C_r$

paths. To address the issue, $D^*_r$ incorporates information from the voxels close to $v_0$, within range $r$ (measured in voxels), giving a traversal cost $C_r = C_u/(d + 1)$ added to its $C_r$, where $d$ is the distance to the $v_0$ counted in voxels. If a $v$ is within $r$, the $C_r$ from multiple $v_0$ will be added for the closest $v_0$.

In this case, the scaling behavior for $C_r$ creates a gradient risk area that will make $D^*_r$ plan a path in the middle of a narrow tunnel-like area and with the desired safety marginal in more open spaces. By using $C_r$ the planned path resembles the ones seen in Figure 3 where the path is planned inside $v_f$ where it is possible. In situations (e.g. extremely narrow corridor-like environments) where there is no $v_f$, the proposed planner prioritizes the generation of paths inside $v_r$.

C. Updates and expansion

While the robot moves within an area, the mapping tool continuously expands the information of the visited area, updates the surroundings, fills missing parts of the scene while exploring new areas, thus $\mathcal{G}$ is created in a dynamic and updating way. More specifically, this dynamic map update is captured in Figures 1 and 4 where as the robot traverses the desired path, more and more knowledge of the environment is revealed and saved in $\mathcal{G}$.

III. RESULTS

A. Evaluation in Simulations

In this Section we provide simulation results from a cave environment [28], comparing $D^*_r$ with a Rapidly exploring Random Tree (RRT) voxelbox planner [29], [30], [31]. All
as the robot traverses the path, planned in Figure 1, the unknown cells (\(v_u\)) will become known and \(D^*_+\) will re-plan the path. The map is expanded as well.

![Image](image1)

**Fig. 4:** As the robot traverses the path, planned in Figure 1, the unknown cells (\(v_u\)) will become known and \(D^*_+\) will re-plan the path. The map is expanded as well.

![Image](image2)

**Fig. 5:** The part of a SubTerranean cave that was used for simulations. The blue line depicts the drones’ path generated from \(D^*_+\), while the red line visualizes the drones’ path generated from the RRT planner. The test mission was a waypoint navigation from ‘A’ to ‘C’ and then back to ‘A’. The point ‘B’ is how far RRT managed to plan a path until a failure in the planner.

Components (aerial platform, control, mission definition) utilised in the simulation runs were identical, with the only difference the path planners. The simulated quadcopter was equipped with the RGB-D Intel Realsense D435 to observe the surroundings and capture the map. The waypoint commands were converted to roll, pitch, yawrate and thrust commands for the drone, using a Nonlinear Model Predictive controller [32] to follow the paths that were generated by the different path planners. The defined mission for the aerial platform, considered the navigation from location ‘A’ to location ‘C’ and then back to ‘A’ as shown in Figure 5. An internal map for the path planning was built during the operation, where the path planners were compared in a simulated exploration mission.

From the obtained simulation results, the \(D^*_+\) method allowed the aerial platform to reach up to location ‘B’ and hover, without providing new paths. Apart from the quantitative comparison of the generated path, we also provide qualitative comparison regarding the average computation load, memory usage at the end, total distance traveled, and execution time for the traversal locations ‘A’ to ‘B’, summarized in Table I.

TABLE I: Numeric comparisons between \(D^*_+\) and RRT in the from the considered simulation results.

| Planner | CPU (%) | Memory (%) | Distance (m) | Time (s) |
|---------|---------|------------|--------------|----------|
| \(D^*_+\) | 88.9 | 13.1 | 55.9 | 506 |
| RRT | 32.9 | 1.3 | 51.3 | 1020 |

![Image](image3)

**Fig. 6:** The aerial platform and the Spot robot utilized in the field trials for evaluating the efficacy of \(D^*_+\).

**B. Experimental Evaluation**

For the real experimental trials, the Velodyne VLP-16 was used as the sensor to capture the surrounding area. The experiment took place in a local mine-like environment in Northen Sweden.

1) Aerial robot: For the experiments with a 3D path, a UAV was used and the map was built by octomap [16]. In the experiment the \(D^*_+\) was tuned to plan a path with \(r = 2\) voxels safety distance to any obstacle. \(D^*_+\) was able to plan a path that the drone could follow autonomously.

A test of \(D^*_+\)’s capability to plan a 3D path over obstacles was also performed with offline data captured from previous real-life field trials. The generated path was obstacle free, while keeping safe distance from the surrounding walls, as depicted in Figure 7.

2) Ground robot: For the ground robot experiments the Boston Dynamics Spot legged robot was used as the robotic platform. The utilized map was crated with Cartographer SLAM [19] and the goal waypoint were manually selected. Longer paths were tested where Spot navigated a distance of approximately 83m, following a path generated by \(D^*_+\) (see

![Image](image4)

**Fig. 7:** Field evaluation of the \(D^*_+\) during the task of planning a path over an obstacle.
accommodate for imperfect maps or dynamic obstacles and it can also incorporate a safety distance $r$ to any obstacle. $D^*_+$ is agnostic to the robot and thus in its planning the robotic size and related maneuverability, are not considered. It can also plan path for both aerial and ground robots with the only difference to be the initial $M$.

The two path planners, $D^*_+$ and RRT, relied on different mapping methods and that plays a part in the results. From the numeric comparison, RRT is better in all categories except execution time, but those categories do not reflect reliability and safety, were $D^*_+$ was superior in our testing. A large portion of the time difference in execution comes form RRT’s struggle to find a path. RRT outperforms $D^*_+$ in terms of CPU load, which is expected because $D^*_+$ finds the optimal path, while RRT settles for a path quicker. The difference in memory usages has to do with how the map is internally stored and RRT is more efficient at it. Apart from the measurable differences (seen in table I), the RRT planner was struggling more to find a path, while $D^*_+$ was able to complete the route. $D^*_+$ prioritized the safest path to the cost of some distance, at the same time was able to pass through narrow passages and complete the route without collisions.

C. Future work

From the obtained simulation and experimental results it was indicated that the expansion of $D^*_+$’s internal map is too slow to work on a larger scale. This is a weakness that will be addressed in future work by segmentation of the global map. A strict safety limit that invalidates any path that passes through voxels with too high traversal costs, should also be added to avoid any invalid path to be generated in the cases where no safe path exists. Finally, $D^*_+$ is currently not able to explore to find a path that leads outside the current map, that is a feature that could be considered for future work.

IV. Conclusion

This article established a novel generic and platform-agnostic risk-aware path planning algorithm based on the $D^+$ framework. The planner was able to generate a grid map where a traversal cost was added to the unknown voxels that are close to an occupied one. The algorithmic implementation was also enhanced with a dynamic grid map that had the novel ability to update and expand during the robotic operation and thus increased the overall safety of the mission, while being ideal for exploration and search and rescue missions of large areas. Finally, the efficiency of the proposed scheme has been evaluated based on multiple simulations and field experimental results with an aerial robot and Spot.

References

[1] DARPA. (2020) DARPA Subterranean (SubT) challenge. Accessed: February 2021. [Online]. Available: https://www.darpa.mil/program/darpa-subterranean-challenge

[2] A. Agha, K. Otsu, B. Morell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, et al., “Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge,” arXiv preprint arXiv:2103.11470, 2021.
[3] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick, et al., “Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 2518–2525.

[4] M. Eichley, E. Evans, K. Stankevitz, and C. Parker, “Using the unmanned aerial vehicle delivery decision tool to consider transporting medical supplies via drone,” Global Health: Science and Practice, 2019.

[5] S. S. Mansouri, C. Kanelakis, D. Kominiak, and G. Nikolakopoulos, “Deploying MAVs for autonomous navigation in dark underground mine environments,” Robotics and Autonomous Systems, 2020.

[6] B. Lindqvist, C. Kanelakis, S. S. Mansouri, A. akbar Aghamohammadi, and G. Nikolakopoulos, “Compa: A compact reactive autonomy framework for subterranean mav based search-and-rescue operations,” 2021.

[7] J. Tordesillas, B. T. Lopez, and 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.

[8] J. Tordesillas and J. P. How, “FASTER: Fast and safe trajectory planner for navigation in unknown environments,” IEEE Transactions on Robotics, 2021.

[9] X. J. Yan F., Liu YS., “Path planning in complex 3d environments using a probabilistic roadmap method,” International Journal of Automation and Computing, vol. 10, pp. 525–533, December 2013.

[10] M. N. A. Wahab, S. Nefti-Meziani, and A. Atyabi, “A comparative review on mobile robot path planning: Classical or meta-heuristic methods?” Annual Reviews in Control, vol. 50, pp. 233–252, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367588820300675

[11] N. C. Rowe and R. S. Alexander, “Finding optimal-path maps for path planning across weighted regions,” The International Journal of Robotics Research, vol. 19, no. 2, pp. 83–95, 2000. [Online]. Available: https://doi.org/10.1177/02783640022066761

[12] S. Koenig and M. Likhachev, “D* lite,” Annals of AI, vol. 15, 2002.

[13] D. Ferguson and A. Stentz, “Using interpolation to improve path planning: The field D* algorithm,” Journal of Field Robotics, vol. 23, no. 2, pp. 79–101, 2006.

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

[15] M. Sheckells. (2018) DSL Gridsearch: D* lite on a uniformly spaced 3D or 2D grid. Accessed: February 2021. [Online]. Available: https://github.com/jhu-asco/dsl_gridsearch

[16] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3D mapping framework based on octrees,” Autonomous robots, vol. 34, no. 3, pp. 189–206, 2013.

[17] P. Payeur, P. Hébert, D. Laurendeau, and C. M. Gosselin, “Probabilistic octree modeling of a 3d dynamic environment,” in Proceedings of International Conference on Robotics and Automation, vol. 2. IEEE, 1997, pp. 1289–1296.

[18] H. Moravec and A. Elfe, “High resolution maps from wide angle sonar,” in Proceedings. 1985 IEEE International Conference on Robotics and Automation, vol. 2, 1985, pp. 116–121.

[19] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1271–1278.

[20] J. Carsten, D. Ferguson, and A. Stentz, “3d field d: Improved path planning and replanning in three dimensions;” in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 3381–3386.

[21] F. Li, S. Zlatanova, M. Koopman, X. Bai, and A. Diakité, “Universal path planning for an indoor drone,” Automation in Construction, vol. 95, pp. 275–283, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580517311184

[22] N. Jaklin, A. Cook IV, and R. Geraerts, “Real-time path planning in heterogeneous environments,” Computer Animation and Virtual Worlds, vol. 24, no. 3-4, pp. 285–295, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1511

[23] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A. Aghamohammadi, “STEP: stochastic traversability evaluation and planning for safe off-road navigation,” CoRR, vol. abs/2103.02828, 2021. [Online]. Available: https://arxiv.org/abs/2103.02828

[24] L. Jaille, J. Cortés, and T. Siméon, “Sampling-based path planning on configuration-space costmaps,” IEEE Transactions on Robotics, vol. 26, no. 4, pp. 635–646, 2010.

[25] B. Hoover, S. Yaw, and R. Middleton, “Costmap: an open-source software package for developing cost surfaces using a multi-scale search kernel,” International Journal of Geographical Information Science, vol. 34, no. 3, pp. 520–538, 2020. [Online]. Available: https://doi.org/10.1080/13658816.2019.1675885

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

[27] Stanford Artificial Intelligence Laboratory et al., “Robotic operating system.” [Online]. Available: https://www.ros.org

[28] A. Koval, C. Kanelakis, E. Vidmark, J. Hafusa, and G. Nikolakopoulos, “A subterranean virtual cave world for gazebobased on the darpa subt challenge,” http://arxiv.org/abs/2004.08452, 2020.

[29] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, “Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[30] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, “Continuous-time trajectory optimization for online uav replanning,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016.

[31] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, “Safe local exploration for replanning in cluttered unknown environments for micro-aerial vehicles,” IEEE Robotics and Automation Letters, 2018.

[32] S. S. Mansouri, C. Kanelakis, B. Lindqvist, F. Pourkamali-Anaraki, A. a. Aghamohammadi, J. Burdick, and G. Nikolakopoulos, “A Unified NMPC Scheme for MAVs Navigation With 3D Collision Avoidance Under Position Uncertainty,” IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 5740–5747, 2020.