2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Learning-based Path Planning for Autonomous Exploration of Subterranean Environments

Russell Reinhart, Tung Dang, Emily Hand, Christos Papachristos, and Kostas Alexis

***Abstract*— In this work we present a new methodology on learning-based path planning for autonomous exploration of subterranean environments using aerial robots. Utilizing a recently proposed graph-based path planner as a "training expert" and following an approach relying on the concepts of imitation learning, we derive a trained policy capable of guiding the robot to autonomously explore underground mine drifts and tunnels. The algorithm utilizes only a short window of range data sampled from the onboard LiDAR and achieves an exploratory behavior similar to that of the training expert with a more than an order of magnitude reduction in computational cost, while simultaneously relaxing the need to maintain a consistent and online reconstructed map of the environment. The trained path planning policy is extensively evaluated both in simulation and experimentally within field tests relating to the autonomous exploration of underground mines.**

I. INTRODUCTION

Robotic systems are paving their way to be utilized in an ever-increasing set of applications in both the civilian and military domains alike. Aerial robots, for example, are currently integrated in a multitude of surveillance [1], inspection [2–4], search and rescue [5, 6], and commercial applications [7]. However, not every environment is currently rendered possible for robotic entry and autonomous exploration. In this work, we specifically focus on the problem of autonomous exploration of subterranean environments which can often be dull, dirty, and dangerous thus calling for robotic access. Subterranean environments present certain characteristics that challenge the problem of exploration autonomy, namely that they are often a) sensor-degraded, as well as b) long and large-scale, narrow, and multi-branching. Nevertheless, the benefits of robotic autonomy underground can be major. Robots for mine rescue, monitoring of subterranean metropolitan infrastructure, or cave exploration, are all indicative examples of relevant applications.

With the goal of addressing the problem of autonomous exploration in the large-scale, often narrow, and multi-branching underground environments, we recently proposed a Graph-based exploration planner (GBPlanner) [8]. It provides paths that ensure high exploration gain, as evaluated in terms of uncovering previously unexplored volume given a depth sensor, while simultaneously avoiding obstacles and exploiting key observations with respect to the topology of subterranean settings such as mines and tunnels. However,

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111820045. The presented content and ideas are solely those of the authors.

The authors are with the Autonomous Robots Lab, University of Nevada, Reno, 1664 N. Virginia, 89557, Reno, NV, USA rreinhart@nevada.unr.edu

Fig. 1. An instance of the underground mine field deployments on the basis of which learning-based path planning policy results are presented.

GBPlanner - and other similar methods [9–11] - achieve such results on the basis of dense sampling, exhaustive collision checking and volumetric gain calculations on an occupancy map representation of the map, thus requiring significant computation time and resources.

Motivated by the above, the potential use-cases of subterranean robotics and the limitations of methods such as GBPlanner, in this work we present a Learning-Based exploration Planner (LBPlanner) policy that aims to provide planning performance analogous to that of GBPlanner (and similar methods [9–11]) but at a fraction of the computational cost, while simultaneously relaxing the need to maintain a consistent online reconstructed map of the environment. The latter is equally important as underground environments can span across multiple kilometers, thus challenging the state-of-the-art in GPS-denied localization and mapping. To achieve this goal, LBPlanner builds upon the principles of imitation learning, employs a network design that encodes solution multi-modality and utilizes the GBPlanner as an "expert" to provide training data. The proposed network architecture accounts for the topological characteristics of typical subterranean settings involving multiple branches, long and possibly inclined paths, alongside a large variety in terms of size. Furthermore, the LBPlanner departs from the assumption of an always-available volumetric representation of the environment (e.g., through Octomap [12]) and only employs a sliding window of direct LiDAR range observations for its training and inference steps. As evaluated through experimental data including field tests inside underground mines (Figure 1), the trained policy provides an exploratory behavior similar to that of its expert (GBPlanner) but with an order of magnitude reduction in computational cost, a fact particularly useful for its deployment in increasingly smaller, and more agile flying robots operating in

subterranean domains. The training and test data from this work are openly released together with this submission [13].

The remainder of this manuscript is structured as follows: Section II presents related work, Section III outlines the problem, followed by the proposed approach in Section IV. Evaluation studies are presented in Sections V. Finally, conclusions are drawn in Section VI.

II. RELATED WORK

Robotic exploration of unknown environments remains a key area of interest in the robotics community and over the years a number of methods focusing on the exploration path planning problem have been proposed [14–17]. Volumetric exploration of unknown spaces has been historically addressed by frontier-based methods [15], where the objective of the path-planning algorithm is to actively guide the robot towards the frontiers of its sensor perception range. Similarly, sampling based methods have also been employed where the objective is to sample the "next–best–view" [14] which maximizes the volumetric observation of the unknown space. More recent efforts in this domain have focused on multi-objective planning [10, 11], alongside multi-layered architectures [8] combining efficient local exploration with a global re-planning step to reposition the robot towards frontiers previously discovered along large-scale topologies. Contributing into the same domain, a set of methods have been proposed to utilize machine learning for the problem of exploration planning [18–20], while a great amount of work has focused on learning for navigation [21]. Motivated by the progress in the literature, but also cognizant of the specific challenges of subterranean environments, the lack of extensively field-verified learning-based path planning methods, and the fact that algorithms shown to provide good behavior in such challenging conditions typically are both computationally expensive (e.g., the work in [8]) and require a consistent real-time reconstruction of the map of the environment, this contribution aims to provide a meaningful learning-based alternative. As such, the LBPlanner is both very efficient, with more than an order of magnitude reduction of computational time compared to its "training expert", and simultaneously a method that relaxes the need for consistent online 3D mapping.

III. PROBLEM STATEMENT

The exploration planning problem, as considered in this work, is that of autonomously exploring an initially unknown subterranean environment in the sense of identifying the robot paths that ensure that an onboard depth sensor incrementally unveils and uncovers previously unexplored volume. Let $\mathbb{M}$ be the map of the environment consisting of the explored subset $\mathbb{M}_E$ and the unexplored regions $\mathbb{M}_U = \mathbb{M} \setminus \mathbb{M}_E$. Furthermore, let $d_{\max}$ be the maximum effective range of the depth sensor $\mathbb{S}$, and F_H, V_H its horizontal and vertical field of view. In addition, let the robot's configuration at time t be defined as the flat state of robot position and heading $\xi_t = [x_t, y_t, z_t, \psi_t]$. Furthermore, since for most range sensors their perception stops at surfaces, hollow spaces or narrow pockets may not be fully explored thus leading to a residual map $\mathbb{M}_{*,res} \subset \mathbb{M}_U$ which is infeasible to explore given the robot's constraints. With the overall goal to explore the feasible volume and thus make $\mathbb{M}_U \rightarrow \mathbb{M}_{*,res}$, it must be noted that the learning-based exploration path planning problem is defined *per iteration*, as the trained policy does not assume the availability of the map $\mathbb{M}_E$ but only a sliding window of N_L point cloud observations coming from the depth sensor $\mathbb{S}$ (typically a LiDAR, an RGBD sensor, or a stereo camera).

Definition 1 (Iterative Exploration) Given an environment represented by map $\mathbb{M}$ from which a sequence of N_L observations from a depth sensor $\mathbb{S}$ have been sampled, derive a locally optimized path σ_{opt} that maximizes the volume of $\mathbb{M}_U$ expected to be uncovered within a given time duration ΔT_P. Note that map $\mathbb{M}$ is not assumed to be estimated online.

IV. PROPOSED APPROACH

The proposed learning-based approach on exploration path planning is based on the principles of imitation learning and utilizes "expert" paths from the previously developed GBPlanner - a graph based exploration planner tailored to subterranean environments [8]. The utility of imitation learning has been demonstrated within the framework of the problem of autonomous planning, for example in the context of navigating through forest trails [22], and for the purposes of autonomous driving [23]. In this work, which in turn focuses on exploration planning which further requires considerations with respect to the anticipated information gain, a fully-convolutional neural network is trained to imitate the short term behavior of the sampling-based GBPlanner.

A. Imitation Learning from Expert Planner

We begin with some datasets $D_e = \{(o_n, a_e)\}$ of range image observations o_n and subsequent actions (paths) a_e generated by an expert policy π_e (in this case, the GBPlanner) during exploration of an unknown environment. A deep neural network π_θ with parameters θ is trained to minimize a loss function $\mathcal{L}(a_e, \pi_\theta(o_n))$ via stochastic gradient descent. Once this "novice" policy has been trained on expert data, it is deployed in a simulation environment and proposes actions $a_n = \pi_\theta(o_n)$ based on N_L recent sensor observations. If this novice's action a_n is viable (in this case, the action does not lead to collision with the environment), then the action a_n is executed. At each time the planner is triggered, the expert policy also proposes an action $a_e = \pi_e(\mathbb{M}_E)$, which in conjunction with the novice's observation at this time, o_n, becomes a new training data point (o_n, a_e). Note that the observations for the novice and expert policies are different; the novice receives concatenated range images calculated from recent point cloud measurements, while the sampling-based expert depends on the current volumetric map reconstruction of $\mathbb{M}_E$. This method, known as imitation learning with dataset aggregation [24], or DAGGER, enables the generation of training data which would not be generated by an expert policy which is less likely to take sub-optimal

or dangerous actions and then have to recover. Algorithm 1 outlines this method performing n_{epoch} data collection and training epochs.

Algorithm 1 Imitation Learning with Onboard Expert DAGGER

$\pi_\theta \leftarrow rand$ ▷ initialize novice policy parameters
$D \leftarrow D_e = \{(o_n, a_e)\}$
for $i = 0, 1, ..., n_{\text{epoch}}$ **do**
  $\theta \leftarrow \arg\min_\theta \sum_{(o_n, a_e) \in D} \mathcal{L}(a_e, \pi_\theta(o_n))$
  for $i = 0, 1, ..., n_{max}$ **do**
    $a_n \leftarrow \pi_\theta(o_n)$ ▷ action from novice policy
    $a_e \leftarrow \pi_e(\mathbb{M}_E)$ ▷ action from expert policy
    $D \leftarrow D \cup \{(o_n, a_e)\}$ ▷ add new training data
    if a_n is valid **then**
      Execute a_n ▷ explore with novice policy
    else
      Execute a_e ▷ collision-free expert action
return π_θ

B. Model Design

We require the model to be able to identify multiple potential exploration directions. For example in a tunnel, the model should always identify the dominant directions of the tunnel as potential paths, and must be able to select between different branches of an intersection. We achieve this behavior by dividing the space around the robot into octants and constructing a network with 8-dimensional output where each dimension is responsible for proposing paths in a specific octant around the robot.

The input to the deep neural network model is a stack of $N_L = 3$ $[16 \times 90]$ range images calculated from recently-received point clouds. Features are extracted from this image, and the features are then input into three subnetworks with outputs:

- Predicted scores for trajectories in each octant, an 8×1 vector s.
- Angular offsets from each direction, an 8×2 matrix $\mathbf{D}$ where the i-th row, $[\Delta\psi_i, \Delta\theta_i]$, contains angular offsets from the i-th octant around the robot. $\Delta\psi$ is a pitch angle offset, and $\Delta\theta$ is a roll angle offset.
- A $16 \times 90 \times 3$ matrix $\mathbf{o}$, output by a decoder network, which is the reconstructed input range image stack. This is used for model training and not during navigation.

Note that the decision for the network to represent an 8-modal distribution is an engineering one. It was found that this level of discretization enabled the model to learn to effectively navigate narrow, branching, tunnel-like environments. The selected resolution can be increased, at the cost of increasing the dimensionality of the underlying network. Figure 2 presents a diagram of the network architecture.

These eight directions are further divided into "forward" and "backward" directions. Forward directions are those in $[-\pi/2, \pi/2]$ of the most recent direction chosen and backward directions lie in $(\pi/2, 3\pi/2)$. Define i_f as indices corresponding to forward directions and i_b as indices for backward directions. At inference time, the maximum forward predicted score is $s_{f,\max} = \max s[i_f]$ corresponding to direction $d = \arg\max s[i_f]$. If $s_{f,\max}$ is less than some threshold (i.e., there is low predicted score for forward paths) then the direction d becomes $d = \arg\max s[i_b]$. Once the direction is determined, the d-th row of the angular offset matrix $\mathbf{D}$ is added to the d-th direction's angular coordinates and the robot proceeds along a 2m path in this direction.

This division into forward and backward directions enables the robot to reverse exploration directions if forward directions have low predicted score as detailed in Section V. This heuristic also prevents the robot from oscillating between forward and backward motion in a tunnel where each direction may look equally promising (similar predicted score) to a model with no notion of time history or access to a reconstructed map.

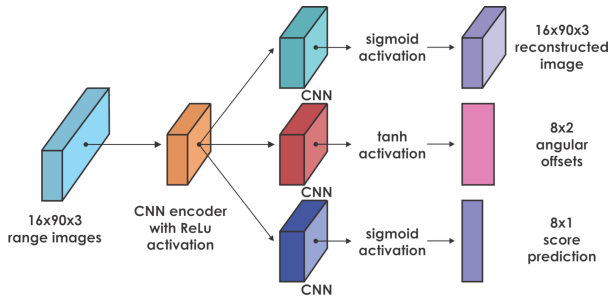

Fig. 2. Network architecture of the learning-based exploration planner.

C. Training Data Generation

Within this work, training data are generated using the GBPlanner [8] as follows. First, a set of simulated underground environments are considered and the GBPlanner is set to enable their autonomous exploration based on a simulated $1.4\text{m} \times 1.4\text{m} \times 0.5\text{m}$ multirotor equipped with a LiDAR sensor with horizontal and vertical field-of-view of $[F_H, F_V] = [360, 30]^\circ$ and a range of $d_{\max} = 100\text{m}$. At roughly 2m intervals in the simulation environment, the sampling-based GBPlanner is triggered and generates a set of scored exploration paths $\{\sigma_{exp}\}$. Sampled paths that are still viable after some cut-off distance $\delta^{\max}$ from the robot, and which lie within $\pm\pi/8$ of one of the octant axes of the robot are used to generate three regression targets, $\hat{s}$ (8×1), $\hat{D}$ (8×2), and $\hat{\mathbf{o}}$ $(16 \times 90 \times 3)$ for the model whose outputs are described in the preceding section. A schematic diagram of training data being generated is shown in Figure 3.

The path score is calculated such that directions (corresponding to indices in the score vector $\hat{s}$, and rows in the angular offset matrix $\hat{\mathbf{D}}$) which have no viable paths get 0 score. Scores for directions with viable paths are normalized so that the path with the highest expected volumetric exploration gain gets score 1, while the path with lowest exploration gain (often a backtracking path) gets score 0.5.

Initial training data were generated by running the GBPlanner in a number of simple simulation environments with geometries relevant to the subterranean exploration task.

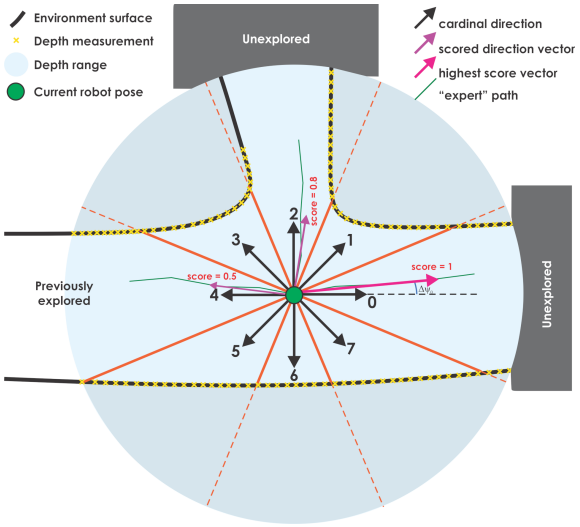

Fig. 3. Training data generation by organizing the expert solutions in octants and within those deriving the directions of proposed exploration.

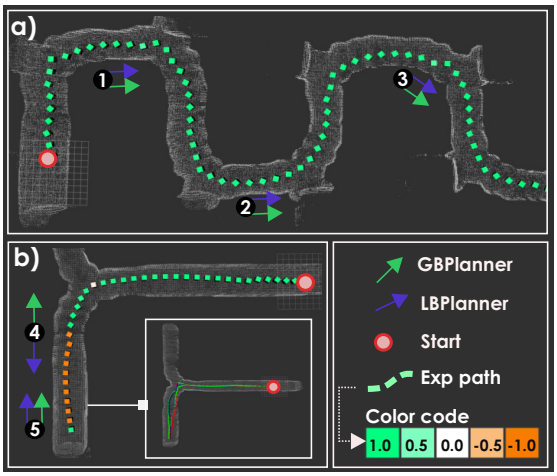

Fig. 4. Comparison of exploration directions proposed by LBPlanner and GBPlanner in test simulation environments. Both planners were triggered at each point shown, but the LBPlanner paths were executed. The color code for each planner iteration is determined by the dot product between the exploration directions proposed by each planner. Subfigure a) shows high correlation between the proposed exploration directions at all points (average 19° absolute angle difference). Subfigure b) shows the GBPlanner identifying the dead-end (4) before the LBPlanner (5), as well as the LBPlanner's failure to explore the other branch.

This includes straight tunnels, tunnels with branches, turns, inclines, declines, intersections, and obstructions. Once the LBPlanner agent has been trained on this data, the agent is put into the same simulation environments. Additional training data are then generated as the agent takes trajectories that would have been ignored by the GBPlanner due to their lack of exploration gain, inefficiency, or close proximity to occupied space (thus leading to a possible impending collision). Meanwhile, the GBPlanner concurrently labels observations with more optimal actions. All training datasets were duplicated by shifting pixels in the range image observations and rotating the corresponding path directions accordingly over an angular range of $\pm 80°$ in $4°$ increments to reduce directional bias in training data. A total 97k training samples were collected across 14 simulation environments, 5% of which were set aside as a validation set. The complete training dataset is released online at [13].

D. Training

The loss function minimized is a weighted sum of mean-squared errors between predicted scores, predicted angular offsets, and reconstructed images. For each sample, this is:

$$\mathcal{L} = \alpha \sum_{j=1}^{8} (\hat{s}_j - s_j)^2 + \beta \sum_{j=1}^{8} \sum_{k=1}^{2} (\hat{\mathbf{D}}_{jk} - \mathbf{D}_{jk})^2 \quad (1)$$
$$+\gamma \sum_{m=1}^{16} \sum_{n=1}^{90} \sum_{p=1}^{3} (\hat{o}_{mnp} - o_{mnp})^2$$

where α, β, γ are hyperparameters controlling the relative weight of each regression target. Training was done on batches of 600 samples with a learning rate 5×10^{-6} and using an ADAM optimizer.

V. EVALUATION STUDIES

In order to evaluate the learning-based exploration planner, we performed a series of both simulation and experimental studies. Within the experimental field results, an autonomous aerial robot equipped with an onboard 3D LiDAR was utilized and deployed inside an underground mine in Northern Nevada. The data from this environment were not utilized in any training phase.

A. Simulation Studies

Models trained using the methods described above were tested extensively in simulation environments from which no training data were acquired. Figure 5 shows trajectories taken by the LBPlanner trained model in a square circuit, a T intersection, and a mesh reconstruction of an underground mine. Exploration trajectories taken by the model trainer (GBPlanner) in the same environments are also shown. The T and square environments were built from different model components than those used in the training environments, thus indicating that the model which was trained in tubular tunnels seems to generalize to more rectangular tunnel environments. All LBPlanner trajectories shown were collision-free despite not explicitly checking for collision prior to executing any proposed paths (e.g., through using a map). The exploration rates e_r, the computation times t_c per iteration (using an Intel i7-7820HQ CPU at 2.90GHz), and the number of paths planned N_c of both planners for these simulations and the field experiments are listed in Table I.

As discussed, the primary goal of this work is to approximate the local exploration behavior of the sampling-based expert (GBPlanner), without assuming the availability of an online reconstructed map or other way to store or access temporal information regarding the robot mission. This fact can in turn lead to some undesired behavior when faced with multiple viable directions. The trajectory of the LBPLanner while exploring a "T" shaped environment exemplifies this (see Figure 4b). After exploring one branch of the T and returning to the intersection, the predicted highest-score

TABLE I

	LBPlanner			GBPlanner		
Env.	$e_r\ [m^3/s]$	$t_c\ [s]$	N_c	$e_r\ [m^3/s]$	$t_c\ [s]$	N_c
T	3.48	0.046	54	4.31	3.13	17
Square	4.66	0.041	73	6.09	2.82	21
Mine sim	4.69	0.028	45	4.27	3.69	25
Mine real	8.37	0.031	60	8.43	3.0	12

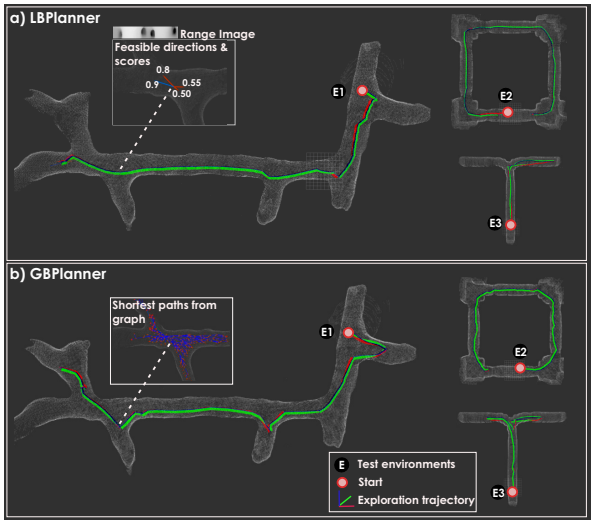

Fig. 5. Trajectories taken by LBPlanner a) and GBPlanner b) corresponding to the exploration rate results in Table I. As visually observed, the paths are highly similar which indicates the ability of LBPlanner to mimic the performance of its training expert.

trajectory lies toward the starting position and the other branch of the environment remains unexplored. This behavior is an inevitable consequence of the network data input used, as the inference of waypoints only depends on very recent sensor measurements, and the network architecture which encodes no notion of time-history or map.

B. Experimental Studies

In order to field demonstrate and evaluate the LBPlanner we further conducted an experimental study in an actual underground mine. The presented field evaluation study took place utilizing an autonomous aerial robot developed around a DJI Matrice M100 and integrating a NUCi7 and combined LiDAR Odometry And Mapping, as well as visual-inertial localization. The system relies on Model Predictive Control (MPC) for its automated operation. The LBPlanner subscribes to the LiDAR sensor stream, calculates the reference path and this is in turn tracked by the onboard MPC. Details for the overall system can be found in [25–28], while an overview is illustrated in Figure 6. The integrated depth sensor $\mathbb{S}$ is a Velodyne PuckLITE with $[F_H, F_V] = [360, 30]^\circ$, and a maximum range of 100m.

This robotic system, integrating the LBPlanner, was deployed at the "Lucerne" underground mine in Northern Nevada. This is a portal mine which allowed the robot to be deployed from outside and be tasked to explore the complete mine drift. During its operation, paths commanded by LBPlanner were set to be followed with an average

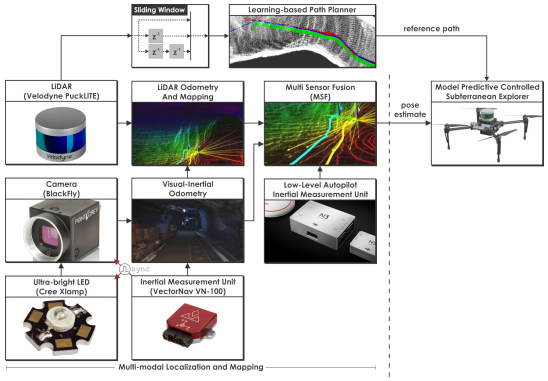

Fig. 6. System overview of our aerial subterranean robotic scout.

velocity of 0.5m/s, while the path length for the LBPlanner solutions was set to 2.0m. Figure 7 presents the respective experimental results with a robot traversing more than 170m inside this underground mine involving multiple intersections and in completely autonomous exploration mode on the basis of LBPlanner. It is noted that the side-drifts in the mine are identified as feasible directions by the LBPlanner, however the predicted scores for these directions are less than that of the forward direction, so the robot never chooses to enter these. This behavior is desirable, as these are very short and offer little exploration gain. Furthermore, Table I offers a comparison of the average exploration rate e_r and planning time per iteration between LBPlanner and GBPlanner, alongside the total number of planning steps required to explore the Lucerne mine. The respective result for GBPlanner deployed to explore the same environment was presented in [29]. As shown LBPlanner achieves similar exploration rate (Figure 8) at a fraction of the associated computational cost. Finally, a second experimental study took place within the same underground mine with the robot now being deployed from inside one of the muckbays (Figure 7c). With the left side of the mine being blocked by introduced structure we observe that the LBPlanner correctly navigates and explores the mine by taking the appropriate right turn and then progressing to explore the remaining drift.

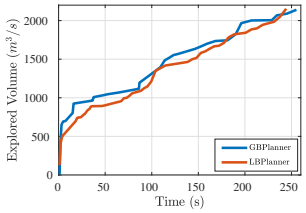

Fig. 8. Exploration rate of the two planners in the Lucerne mine.

C. Discussion on the Computational Cost

The design and training of the LBPlanner delivers a performance similar to that of its expert (GBPlanner), but at a fraction of the computational cost as detailed in Table I. This "compression" operation corresponds to a key feature of the method. To allow further understanding, we relate the computational analysis of this new method with GBPlanner.

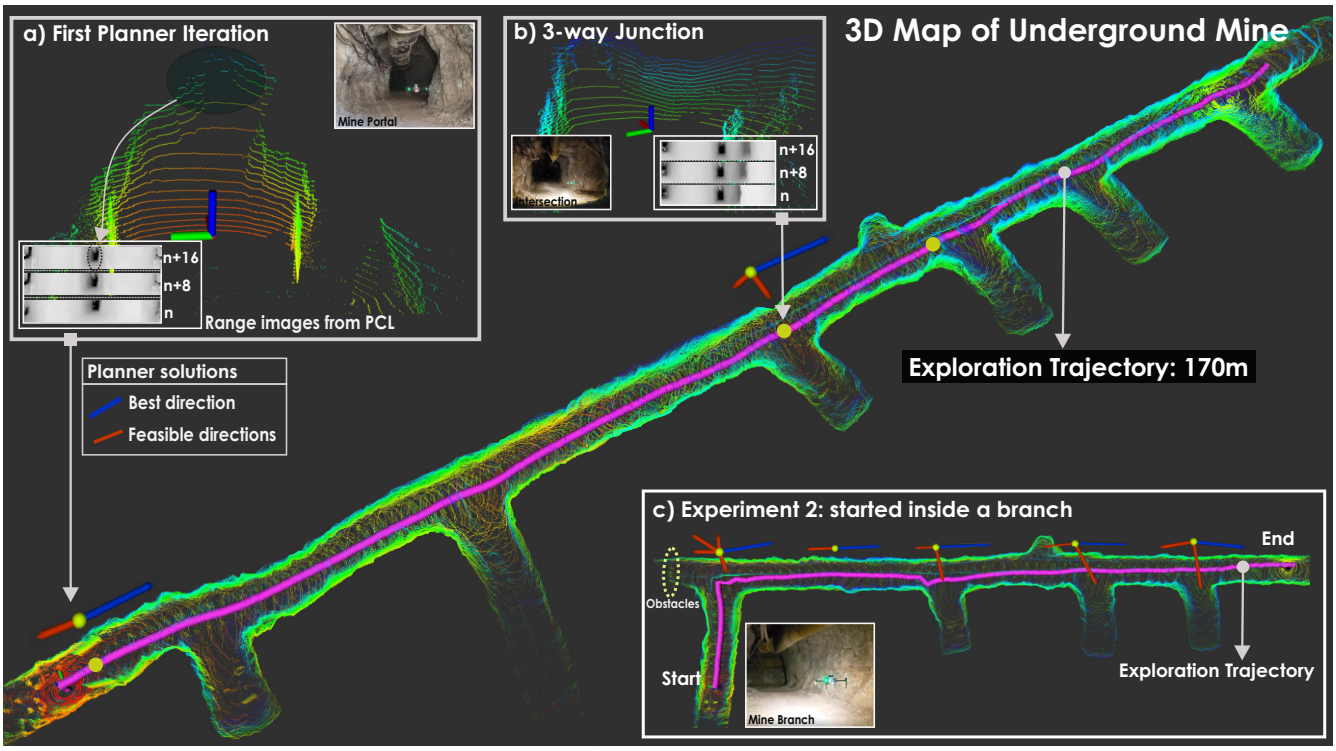

Fig. 7. Learning-based exploration planner navigating a 170m drift of an underground mine starting from the main portal. Feasible directions identified by the LBPlanner (arrow lengths proportional to predicted score) at the start of the tunnel and an intersection are shown along with corresponding local point clouds, range images, and images captured by the onboard camera (sub-figures a-b). The proposed planner was further evaluated in another scenario where the robot started at the base of a T-junction; hence, the planner had to safely guide the robot to enter the intersection then keep exploring towards the left or the right branch. As shown in sub-figure c), the planner correctly proposed safe and high exploration gain paths towards the right branch.

As detailed in [8], the local exploration step of the GBPlanner (which is "imitated" by LBPlanner) has a computational growth rate function that depends, among others, on the amount of vertices and edges sampled, and the ratio between the volume evaluated in terms of volumetric gain or the length of the edges evaluated for collision against the resolution of the volumetric representation. Indicatively, only the process of evaluating the volumetric gain for each vertex of the sampled graph has a growth rate that takes the form $\mathcal{O}(N_{\mathbb{V}} F_H F_V d_{\max}/(r_H r_V r) \times \log(V_{D_G}/r^3))$, where $N_{\mathbb{V}}$ the number of vertices, r_H, r_V the sensor resolution, V_{D_G} the size of the volume of the environment and r the resolution of the underlying occupancy map. Similarly, the collision checking step has a growth rate of $\mathcal{O}(V_{D_R}/r^3 \times d_{avg}/r \times \log(V_{D_G}/r^3))$, where V_{D_R} is the considered volume for the robot, and d_{avg} the average length of the graph edge. As this brief analysis indicates, the GBPlanner operation can become computationally very demanding for sensors with long measurement distance (e.g., a Velodyne LiDAR), for planning over large subsets of maps, or for planning with high resolution of the occupancy map (which may be required to ensure collision avoidance in narrow settings).

On the contrary LBPlanner has a computational cost per iteration that only depends on the designed complexity of its neural network and the associated calculations during an inference step. The presented network architecture in LBPlanner contains 160k weights and an inference step requires 311k float operations. As such, and despite the fact that for a single iteration of the GBPlanner it typically requires multiple LBPlanner iterations (~ 4) due to the shorter horizon of its commanded paths, it is revealed that the method achieves an order of magnitude in complexity reduction with average time for path calculation around ~ 35ms. In fact, it presents a constant and small complexity, whereas GBPlanner is both in general expensive and its complexity grows with the size of the map. Future research will focus on retaining this important achievement and address the limitation of LBPlanner with respect to not exploiting information from the full history of robot observations.

VI. CONCLUSIONS

An imitation learning-based exploration path planning policy capable to enable the autonomous exploration in subterranean environments was presented. By ensuring that the training data include challenging configurations of the state space, while explicitly accounting for planning results organized in a selected set of cardinal directions, the trained policy is able to negotiate challenging multi-branching underground environments. As shown both in simulation and in field experiments, the new planner presents efficient behavior at a fraction of the computational cost of the expert trainer, and without the need for a consistent map. Future work will focus on extending the network architecture of the planner to further relate to the previous history of observations of the robot but still without the need to build a map.

REFERENCES

[1] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.

[2] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel and R. Siegwart, "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," *Autonomous Robots*, pp. 1–25, 2015.

[3] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel and R. Siegwart, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6423–6430. [Online]. Available: https://github.com/ethz-asl/StructuralInspectionPlanner

[4] C. Papachristos, M. Kamel, M. Popović, S. Khattak, A. Bircher, H. Oleynikova, T. Dang, F. Mascarich, K. Alexis, and R. Siegwart, "Autonomous exploration and inspection path planning for aerial robots using the robot operating system," in *Robot Operating System (ROS)*. Springer, 2019, pp. 67–111.

[5] H. Balta, J. Bedkowski, S. Govindaraj, K. Majek, P. Musialik, D. Serrano, K. Alexis, R. Siegwart, and G. De Cubber, "Integrated data management for a fleet of search-and-rescue robots," *Journal of Field Robotics*, vol. 34, no. 3, pp. 539–582, 2017.

[6] T. Tomic, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[7] B. Rao, A. G. Gopi, and R. Maione, "The societal impact of commercial drones," *Technology in Society*, vol. 45, pp. 83–90, 2016.

[8] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2019.

[9] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016. [Online]. Available: https://github.com/ethz-asl/nbvplanner

[10] T. Dang, C. Papachristos, and K. Alexis, "Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[11] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty–aware receding horizon exploration and mapping using aerial robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[12] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[13] R. Reinhart, T. Dang, E. Hand, C. Papachristos, and K. Alexis, "LBPlanner Open Dataset." [Online]. Available: https://www.autonomousrobotslab.com/lbplanner-release.html

[21] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[14] C. Connolly *et al.*, "The determination of next best views," in *IEEE International Conference on Robotics and Automation 1985*.

[15] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *CIRA'97*. IEEE, 1997, pp. 146–151.

[16] M. Popovic, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran, "Online informative path planning for active classification using uavs," *arXiv preprint arXiv:1609.08446*, 2016.

[17] M. Nieuwenhuisen and S. Behnke, "Search-based 3d planning and trajectory optimization for safe micro aerial vehicle flight under sensor visibility constraints," *arXiv:1903.05165*, 2019.

[18] B. Chen, B. Dai, and L. Song, "Learning to plan via neural exploration-exploitation trees," *arXiv preprint arXiv:1903.00070*, 2019.

[19] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet, "A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot," *Autonomous Robots*, vol. 27, no. 2, pp. 93–103, 2009.

[20] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, "Active policy learning for robot planning and exploration under uncertainty." in *Robotics: Science and Systems*, vol. 3, 2007, pp. 321–328.

[22] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2015.

[23] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[24] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[25] C. Papachristos, S. Khattak, F. Mascarich, and K. Alexis, "Autonomous navigation and mapping in underground mines using aerial robots," in *2019 IEEE Aerospace Conference*, March 2019, p. *to appear*.

[26] C. Papachristos, and K. Alexis, "Thermal-inertial localization for autonomous navigation of aerial robots through obscurants," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018.

[27] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using ros," *Springer Book on Robot Operating System (ROS)*.

[28] S. Khattak, C. Papachristos, and K. Alexis, "Keyframe-based direct thermal–inertial odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.

[29] T. Dang, F. Mascarich, S. Khattak, H. Nguyen, N. Khedekar, C. Papachristos, and K. Alexis, "Field-hardened robotic autonomy for subterranean exploration," in *12th Conference on Field and Service Robotics (FSR)*, August 2019.