2023 21st International Conference on Advanced Robotics (ICAR)
5-8 December 2023. Abu Dhabi, UAE

Learning Heuristics for Efficient Environment Exploration using Graph Neural Networks

Edwin P. Herrera-Alarcón[1,2], Gabriele Baris[2], Massimo Satler[2], Carlo A. Avizzano[2], and Giuseppe Loianno[1]

***Abstract*— The robot exploration problem focuses on maximizing the volumetric map of a previously unknown environment. This is a relevant problem in several applications, such as search and rescue and monitoring, which require autonomous robots to examine the surroundings efficiently. Graph-based planning approaches embed the exploration information into a graph describing the global map while the robot incrementally builds it. Nevertheless, even if graph-based representations are computational and memory-efficient, the exploration decision-making problem complexity increases according to the graph size that grows at each iteration. In this paper, we propose a novel Graph Neural Network (GNN) approach trained with Reinforcement Learning (RL) that solves the decision-making problem for autonomous exploration. The learned policy represents the exploration expansion criterion, solving the decision-making problem efficiently and generalizing to different graph topologies and, consequently, environments. We validate the proposed approach with an aerial robot equipped with a depth camera in a benchmark exploration scenario using a high-performance physics engine for environment rendering. We compare the results against a state-of-the-art planning exploration algorithm, showing that the proposed approach matches its performance in terms of explored mapped volume. Additionally, our approach consistently maintains its performance regardless of the objective function used to explore the environment.**

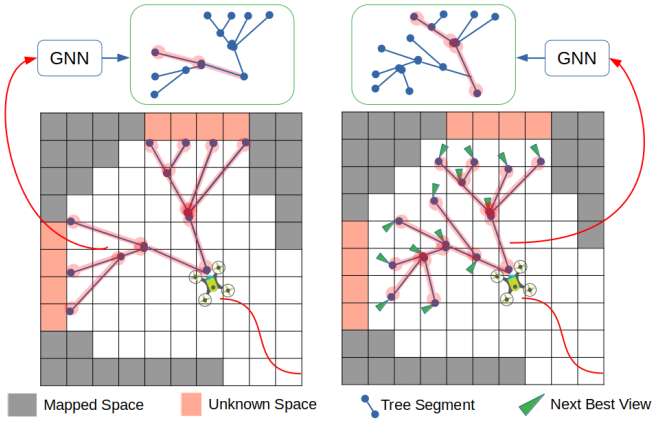


Fig. 1: Functional concept of graph-based exploration planners with a frontier-based (left) and a sampling-based (right) method. A tree (graph) is expanded following specific growing criteria and later evaluated in terms of a defined information gain that characterizes the next target to visit. Given a graph, our GNN module (top) can identify the best next segment to continue the exploration.

I. Introduction

Recent years have shown an increasing interest in raising the autonomy of mobile robots, especially aerial and ground solutions. These machines must be capable of autonomously exploring and scanning partially or fully unknown environments to resolve search and rescue or monitoring tasks. However, obtaining this level of autonomy requires high robustness and efficiency in the high-level decision-making process. Autonomous exploration is an intensely studied problem in robotics that has been addressed with numerous methods. Several aerial robot solutions [1], [2], [3] have shown to be particularly suited to solve complex or dangerous exploration tasks due to their low cost, ease of use, flexibility to navigate in 3D space, and efficient information-gathering ability.

More specifically, when stating the exploration problem, graph-based planners have improved the efficiency of the exploration algorithms, where even small-scale robots with SWaP (Size, Weight, and Power) constraints can explore and reconstruct environments [4], [5]. Exploration approaches can be classified into two main categories, as depicted in Fig.1,

- *Frontier-based methods* continuously explore the boundaries between known and unknown space, maximizing the exploration scenario.
- *Sampling-based methods* explore new paths in the currently explored map and select the most informative ones based on an objective function.

In both cases, route graphs are employed to abstract and represent the exploration problem using this simple, but effective formalism [6]. Nevertheless, as pointed out by the authors in [7], the exploration task in frontier-based methods takes more time to explore an unknown environment because their criterion chooses low-cost solutions that are usually near the current position. Conversely, sampling-based methods as *Next-Best-View* (NBV) [8] exploit an objective

[1]The authors are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA. email: {eph6953, loiannog}@nyu.edu.

[2]The authors are with the Perceptual Robotics Laboratory at the IIM Institute, Department of Excellence in Robotics and A.I., Scuola Superiore Sant'Anna, 56100 Pisa, Italy. email: {e.herreraalarcon, g.baris, m.satler, c.avizzano}@santannapisa.it.

This work was supported in part by the NSF CAREER Award 2145277, the DARPA YFA Grant D22AP00156-00, the Technology Innovation Institute, Qualcomm Research, Nokia, and NYU Wireless. Giuseppe Loianno serves as consultant for the Technology Innovation Institute. This arrangement has been reviewed and approved by the New York University in accordance with its policy on objectivity in research.

2023 21st International Conference on Advanced Robotics (ICAR) | 979-8-3503-4229-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICAR58858.2023.10406720

function formulation to search for robot poses with the *best viewpoint*. The continuous evaluation of the viewpoint with the highest value is an intensive computation process that may also postpone the exploration of small areas to later stages. Therefore, there is no preferred option between those exploration approaches. The most effective one generally relies on the expected behavior within the task and the environment characteristics (i.e., the environment's structure and topology).

Both algorithms are prone to continuously growing computation times proportional to the scale of the exploration environment. Consequently, robots will struggle to meet real-time requirements, especially when exploring large areas. This work proposes a novel decision-making module for exploration algorithms designed using a GNN trained using RL where the learned policy efficiently chooses the exploration tree expansion. The advances in machine learning, particularly GNNs, have opened up new ways to exploit graph structures for addressing several problems, from multi-robot perception [9] to robot planning [10], [11]. Moreover, these methods have proved to be effective for classification [12], [13], link prediction [14], and state relational association [15], [16]. GNNs are usually trained employing a supervised method, but such an approach bounds the learning process capability to the performance of the ground truth used in the training phase. Conversely, RL can be leveraged to train GNNs [17], [18], increasing its generalization capabilities and hence improving its performance.

More in detail, RL used in conjunction with GNNs shows promising results in solving decision-making problems without explicitly generating a dataset with the expected behavior to be learned. GNNs exhibit relational reasoning and combinatorial generalization capabilities. GNNs are naturally suited to represent directional and un-directional graphs' spatial structures. They can be generalized to different graph sizes and topologies, making them suitable for solving complex exploration graphs. The proposed work aims to solve the underlying policy expansion problem in sampling-based planners exploiting modern GNN methods.

The contributions of this work are the following:

- A novel GNN approach able to choose the exploration tree expansion problem generalizing over arbitrary graph topologies.
- A modular training approach where the decision-making module learns from heuristics on combinatorial problems, which is subsequently embedded in a sampling-based planner.
- A comparison against a state-of-the-art exploration algorithm using a conventional decision-making module shows how the proposed approach matches the performance of this algorithm in terms of spatial explored volume while being more efficient and building more accurate maps.

The work is organized as follows: Section II overviews different planning solutions used to explore unknown environments. The methodology is presented in Section III, whereas simulated experiments are shown and analyzed in Section IV. Finally, Section V summarizes the results and future project developments.

II. Related Works

The fundamental goal of an autonomous exploration task is mapping an unknown environment and classifying the space as occupied or free. Depending on the specific application, the exploration task can focus on speed [19], [20], object search [21], [22], 3D reconstruction and inspection [23], [24].

Sampling-based planners use cost-utility formulations to determine the robot's exploration behavior, maximizing exploration-related objective functions associated with the aforementioned objectives. The authors in [1] propose Cauchy-Schwartz quadratic mutual information to evaluate trajectories during exploration, whereas in [25], a Shannon mutual information formulation in Truncated Signed Distance Fields (TSDF) for active exploration is presented. Also, [26] proposed a 3D Reconstruction Gain based on the TSDF that was later implemented in an online informative path planning algorithm based on RRT*. In this case, the algorithm continuously expands an exploration tree, keeping non-executed parts of newly generated sampled viewpoints alive, proving through experiments to overcome costly RRT computation and addressing local minimum problems.

Other global planning approaches have been proposed to escape local minima problems presented at dead-ends during exploration. In [27], the authors propose the *Autonomous Exploration Planner* (AEP), a hybrid strategy mixing frontier-based and sampling-based algorithms adopted for global and local exploration, respectively. In addition, also [28] proposes a hybrid strategy, using the fast marching method to choose the next exploration goal among frontier points. Conversely, the authors in [29] present *GLocal*, a two-stage planning approach that leverages a submap structure to compute efficiently global frontiers in a 3D changing volumetric map. Instead, the planning structure presented in [20] consists of a local receding horizon component and a global sparse topological graph computed online to enable high-speed aerial flight in large-scale unknown environments. Finally, the approach proposed in [30] demonstrates a better scaling performance than other planning approaches as the environment increases by choosing the closest node with an information gain above a minimum threshold.

In recent years, learning-based solutions have also become popular in addressing robot planning and navigation problems [31], [32], [33], [34], [35] due to their ability to generalize to environments of different sizes while being computationally efficient. Related to the exploration task, the approach in [33] builds a topological map using observations obtained from previously collected data for visual goal navigation, whereas in [36], the authors proposed the use of an unsupervised clustering algorithm for obtaining a topological graph of the environment to be used for efficient run-time path planning. The work in [37] tackles the problem of learning sampling-based local exploration planning. The method is based on *Conditional Variational AutoEncoders*

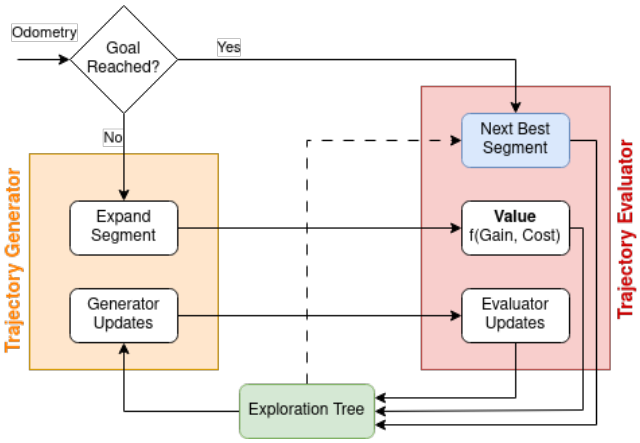


Fig. 2: Planner structure with its submodules. The blue submodule is where the key contribution of this work is located. The dotted line is used for clarity due to the lines crossing.

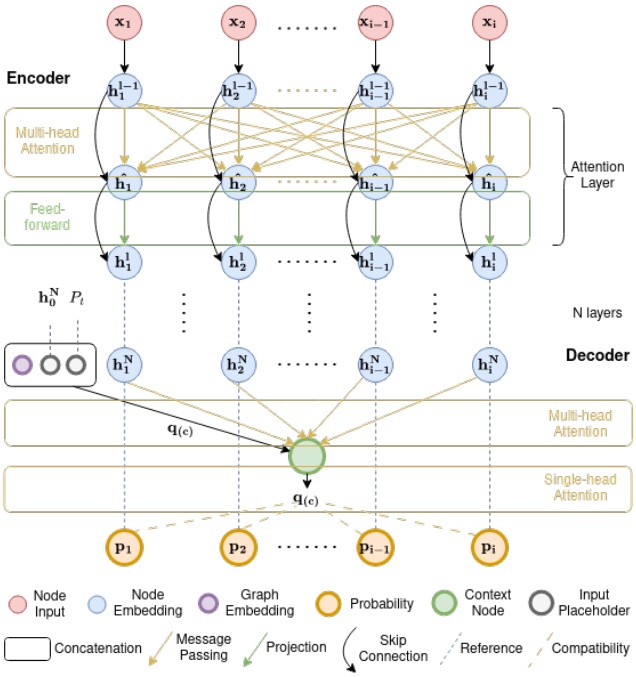


Fig. 3: Architecture of the Attention-based GNN. The Encoder embeds the i inputs in N sequential layers, each composed of a Multi-head attention layer and a Feed-forward sublayer. The Decoder uses the graph and node embeddings, with a Multi-head attention layer and a Single-head attention layer, to compute the output probabilities of each node.

(CVAE) to learn informed multi-modal sampling distributions from occupancy maps, whose performance manages to match and overcome conventional methods while reducing the computational load.

More recently, GNNs have been introduced in robot exploration tasks such as learning sampling distributions to generate the trajectory tree [38], reducing collision checking [39], coupling a frontier-based exploration problem under localization uncertainty [40], [41] with virtual landmarks to predict the robot's optimal sensing action in belief space, or for centralized learning and decentralized decision-making in multi-robot exploration [42].

This work uses an attention-based GNN trained using a policy-based method that returns a stochastic policy about where to go next. Sampling-based policies maximize an objective function determining the exploration's behavior. Instead, the proposed policy is learned from small-scaled, randomly generated problems to be later implemented directly in a growing state-action space as the exploration tree.

III. Methodology

We train a GNN using RL to learn how to solve a routing problem. As depicted in Fig 2, with a blue submodule, the learned policy is used as the expansion criterion for the exploration tree in a sample-based planner. This approach generalizes to unknown topologies while training the policy modularly and without an expert algorithm. In the following subsections, we describe in detail our approach.

A. Overview and Problem Description

The exploration problem discussed in this work involves exploring a bounded 3D unknown space. The objective is to find an efficient and collision-free sequence of robot poses ξ_n that classifies the initial unknown space as either free or occupied. The classified space is described using a discretized map M_d composed of voxels, representing the continuous space with cubical volumes.

The sampling-based planning structure depicted in Fig. 2 shows the components that compose the exploration algorithm. On the one hand, the trajectory generator is responsible for expanding the trajectory tree while guaranteeing mission constraints (i.e., the drone's maximum velocity and distance from obstacles). On the other hand, the trajectory evaluator computes expected gains, costs, and values for the expanded trajectory segments. Specifically, the expansion exploration policy has to validate the best branch according to node value criteria until the environment is entirely classified as free or occupied. Each sample node, robot configuration, has a corresponding value, V, depending on its gain and cost, $V_n = f(G_n, C_n)$. Whereas the cost is usually defined as a numerical description of the trajectory segment (either length or time), the gain is defined due to the occupancy map M_d. Actually, the gain quantifies the set of visible and unmapped voxels from the candidate configuration ξ.

The selection complexity increases with the duration of a mission as the exploration graph continues to grow due to the additionally collected information by the robot. The essential idea of our approach is to improve the robot's decision-making process to choose the best node sequence in the graph efficiently. Learning graph heuristics will save significant time at each decision-making step by finding a path from the robot's current position within every accessible part of the map. Additionally, a decision-making module such as the proposed GNN can better adapt to the different

situations that can arise during the exploration mission. For instance, classic decision-making modules maximize objective functions, meaning the same criterion is going to be used for all the decisions made during the exploration task. Several state-of-the-art algorithms have successfully implemented strategies such as the closest distance or the highest gain node; however, stochastically speaking, there is no unique criterion to approach the exploration of unknown environments better.

B. Decision-making Module

In Fig. 3, the overall architecture of the decision-making module is shown. From the input features $\mathbf{x_n}$, for each of the inputs $n = \{1, ..., i\}$, to the output probability vector $\mathbf{p}$, which decides where to expand the current tree and add adjacent trajectory segments to the target segment. Our approach is based on a neural network architecture that uses a Transformer-style attention mechanism (similar to Graph Attention Networks) but without positional encoding, so the resulting node embeddings are invariant to the input order. This approach has been proven to be successful in solving combinatorial optimization problems [18].

1) Encoder: Composed with N sequential layers consisting of Multi-Head Attention (MHA), weighted message passing between nodes, and node-wise Feed-Forward (FF) sublayer. The attention mechanism during neighborhood aggregation allows each node to weigh its neighbors based on their relative importance. In addition, each sublayer adds a skip-connection and batch normalization (BN) layer. For the following notation, vectors are written in bold font.

The input dimensional features $\mathbf{x_n}$ are fit into dimensional node embeddings $\mathbf{h_n^0}$ through linear projection.

$$\mathbf{h_n^0} = W^x \mathbf{x_n} + \mathbf{b^x}, \quad W^x \in \mathbb{R}^{d_h \times d_x}. \tag{1}$$

The message value passing is weighted through attention weights a_{nj} that are computed from compatibility values u_{nj}. The compatibility, u, between nodes depends on the key $\mathbf{k_n}$ of each neighbor node related to the own attention query $\mathbf{q_n}$ of each node embedding

$$\mathbf{k_n} = W^k \mathbf{h_n}. \qquad W^k \in \mathbb{R}^{d_k \times d_h},$$

$$\begin{cases} \mathbf{q_n} = W^q \mathbf{h_n} & W^q \in \mathbb{R}^{d_k \times d_h}, \\ u_{nj} = \begin{cases} \frac{\mathbf{q_n}^\top \mathbf{k_j}}{\sqrt{d_k}} & \text{if } n \text{ adjacent to } j \\ -\infty & \text{otherwise} \end{cases}, \\ a_{nj} = \frac{e^{u_{nj}}}{\sum_j^i e^{u_{nj}}} \end{cases} \tag{2}$$

$$\mathbf{v_n} = W^v \mathbf{h_n} \quad W^v \in \mathbb{R}^{d_v \times d_h},$$

Then, a combination of value messages v_j is computed for each attention head $\mathbf{h_n}' = \sum_j a_{nj} \mathbf{v_j}$, where each head has different parameters. Multiple attention heads M allow nodes to communicate different messages between neighbors.

$$MHA_n^l(h_1^{l-1}, ..., h_i^{l-1}) = \sum_{m=1}^{M} W_m^O \mathbf{h_{nm}}',$$

$$BN(\mathbf{h_n}) = \omega^{\mathbf{BN}} \odot \overline{BN}(\mathbf{h_n}) + \mathbf{b^{BN}},$$

$$\hat{\mathbf{h}}_\mathbf{n} = BN(\mathbf{h_n^{l-1}} + MHA_n^l(h_1^{l-1}, \cdots h_i^{l-1})), \tag{3}$$

where the FF layer calculates node-wise projections, whereas BN uses element-wise product together with a BN without affine transformation ($\overline{BN}$)

$$FF^l(\hat{\mathbf{h}_\mathbf{n}}) = W^{ff,1} ReLU(W^{ff,0} \hat{\mathbf{h}_\mathbf{n}} + \mathbf{b^{ff,0}}) + \mathbf{b^{ff,1}},$$

$$\mathbf{h_n^l} = \omega^{\mathbf{bn}} \odot \overline{BN}(\hat{\mathbf{h}_\mathbf{n}} + FF^l(\hat{\mathbf{h}_\mathbf{n}})) + \mathbf{b^{bn}}. \tag{4}$$

The input dimensional feature vector $\mathbf{x_n}$ is defined by a position in the 3D euclidean space, the value of the node, and a cost related to its distance. Also, the node embeddings dimension is $d_h = 128$, while the dimensions of the keys and values depend on the number of heads in the attention layer, $d_k = d_v = \frac{d_h}{M}$. In addition, the MHA layer uses 8 heads, whereas the dimension of the FF layer is 128.

2) Decoder: Structured with an architecture similar to the Encoder but simpler, it uses an MHA sublayer to compute the information of a context embedding but without skip-connections, FF, and BN. Subsequently, a single-head attention layer is used to obtain the probabilities related to each input node. The decoder inputs are all singular node embeddings and a temporary context embedding, $\mathbf{h_{(c)}^{(N)}}$. The temporary context embedding is a horizontal concatenation of the mean of the last layer of node embeddings $\frac{1}{i}\sum_{n=1}^{i} \mathbf{h_n^N}$ (graph embedding), the embedding of the current pose $\mathbf{h_0^N}$ and a minimum price constraint P_t.

The temporary context embedding is generated to obtain an attention context query $q_{(c)}$, from the overall problem. Then, a new context embedding is computed using a MHA with messages from each node embedding $\mathbf{h_n^N}$ of the last layer for the keys and value messages, while the query comes from $\mathbf{h_{(c)}^{(N)}}$. Therefore, we employ the same paradigm as in eq. (2), but referenced with the subindex (c) as shown in Fig. 3 and without skip-connections, FF, and BN.

From the resultant context embedding, $\mathbf{h_{(c)}^{(N+1)}}$, we obtain a new context query that is used to calculate the compatibility of each node. However, the compatibility value is slightly modified by clipping its values with a $\tanh$ function,

$$u_{(c)j} = C \cdot \tanh\left(\frac{\mathbf{q_{(c)}}^\top \mathbf{k_j}}{\sqrt{d_k}}\right).$$

Finally, the probabilities are calculated from the mentioned compatibility values, $p_n = \frac{e^{u_{(c)}n}}{\sum_j e^{u_{(c)}j}}$ for each node, from which the resultant probability vector is obtained $\mathbf{p}$. Referenced eqs. (1), (3), (4) are depicted in Fig. 3, as the variation of the attention query using only the context embedding.

3) Training the model: Routing problems generally require sequential decision-making to minimize a problem-specific cost function (in our case, the value function defining the exploration behavior). We define the problem as a particular case of the prize-collecting traveling salesman problem (PCTSP), trading off the penalties of unvisited nodes with the collected price while choosing only one node to go. The previously described architecture is trained using RL. The reason for choosing RL rather than a supervised approach is to avoid limiting the learning performance to the teacher's

Algorithm 1 REINFORCE Algorithm with Rollout Baseline

Require: number of epochs E, steps per epoch T, batch size B, significance α

Initialize θ
$\theta^{BL} \leftarrow \theta$ ▷ Greedy Rollout Policy
for epoch = 1,...,E **do**
  for step = 1,...,T **do**
    $s_k \leftarrow RandomInstance()$ ▷ $\forall_k \in \{1, ..., B\}$
    $\pi_k \leftarrow SampleRollout(s_k, p_\theta)$ ▷ $\forall_k \in \{1, ..., B\}$
    $\pi_k^{BL} \leftarrow GreedyRollout(s_k, p_{\theta^{BL}})$ ▷ $\forall_k \in \{1, ..., B\}$
    $\nabla\mathcal{L} \leftarrow \sum_{k=1}^{B}(L(\pi_k) - L(\pi_k^{BL}))\nabla_\theta \log p_\theta(\pi_k)$
    $\theta \leftarrow Adam(\theta, \nabla\mathcal{L})$
  end for
  if OneSidedPairedTTest() $< \alpha$ **then**
    $\theta^{BL} \leftarrow \theta$
  end if
end for

capacity (i.e., imitating an optimal solver), allowing better generalization capabilities.

The GNN model is parametrized and described with θ, where given a problem instance s, the model returns a probability distribution vector $\mathbf{p} = \mathbf{p}_\theta(\pi|s)$ from which to pick the solution π. The model is trained by minimizing the loss $\mathcal{L}(\theta|s) = \mathbb{E}_{p_\theta(\pi|s)}\big[L(\theta)\big]$, defined as the expectation of the solution π given problem s, using gradient descent as shown in Algorithm 1. We used the gradient estimator defined in [43] with baseline $b(s)$, where $b(s)$ is defined as the cost of a solution from a deterministic greedy rollout of the policy given by the best model so far.

$$\nabla\mathcal{L}(\theta|s) = \mathbb{E}_{p_\theta(\pi|s)}[(L(\pi) - b(s))\nabla log p_\theta(\pi|s)]. \quad (5)$$

A proper selection of the baseline reduces gradient invariance, consequently increasing the learning speed. The baseline estimates the difficulty of a problem instance s, such that it can relate to the cost $L(\pi)$ to estimate the advantage of the solution selected by the model. In this work, we use the same baselines implemented in [18] for the PCTSP, where during the first epoch, an exponential baseline is used to stabilize the learning with the parameter β

$$b(s) = \beta \cdot b(s) + (1 - \beta) \cdot L(\pi)$$

The dataset on which the GNN is trained is generated using normalized randomly generated samples with positions, gains, and penalties with values between 0 and 1. The value of the minimum price constraint P_t is 1, considering all the values are normalized.

C. Planner Structure Embedding

At every planning time step t, the robot is given a local map of the environment where the task is identifying the next configuration ξ_n to reach. Each configuration possibility is depicted with a node that contains a trajectory associated with a gain, cost, and value. Once the current trajectory segment ends the execution, the learned policy chooses the next best adjacent segment based on the values related to each trajectory, and the trajectory tree is updated.

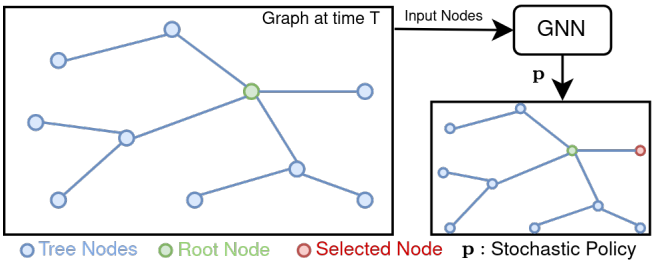


Fig. 4: Example of the proposed planning pipeline. The tree is expanded during the mission, and each node is embedded with exploration information. The nodes are processed by the GNN, which returns the next node to visit.

Parameter	Value	Parameter	Value
graph size	20	embedding dim. (d_h)	128
epoch size (T)	1280000	validation size	10000
batch size (B)	2048	batch size for baseline eval.	1024
FF layer dim.	128	encode layers (N)	3
tanh clipping (C)	10.0	learning rate model	0.0001
significance (α)	0.05	learning rate decay	1.0
seed	1234	number epochs (E)	100
		exp. avg. baseline decay (β)	0.8

TABLE I: Hyperparameters used for training.

The trajectory tree is expanded using a random sampling method such as RRT, which is different from other state-of-the-art planning methods that use RRT*. The RRT* algorithm reduces the number of candidates at each iteration because it already includes an optimization process that rewires the branches following a length reduction criterion. Therefore, we chose to use the classic RRT algorithm, which constitutes a more fair comparison with our approach and will allow to better appreciate the decision-making module's capabilities and usefulness.

The exploration tree information embedded into the input dimensional features $\mathbf{x_n}$ is the position of the nodes in the 3D space, their computed value $V_n = f(G_n, C_n)$ as the prize, and their cost as the penalty. All values are normalized to remain coherent with the dataset in which the GNN is trained. In addition, not the entire exploration tree is given to the GNN model at each iteration, but rather a local map from the current pose (root) to a graph depth of dimension 3, considering the continuous increase of random samples.

Then, after the node is chosen, following the receding horizon sampling-based planner fashion, the robot only executes the first edge of the tree towards the best viewpoint, after which the process is repeated.

IV. Experiments

A. System Setup

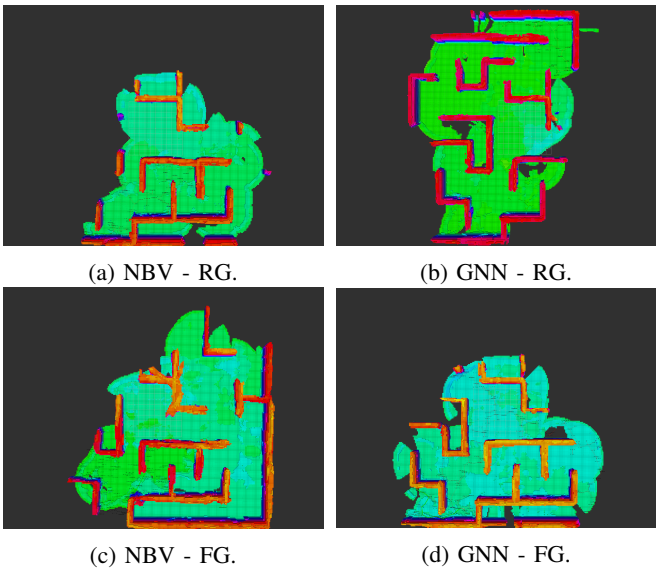
To evaluate the effectiveness of the proposed solution, we perform tests in a simulated environment, using PX4 Software in the Loop (PX4-SITL) and compare the proposed approach with a robust planner for Autonomous Exploration.

Parameter	Notation	Value
Camera FoV	$[a_h, a_v]$	$80°$, $50°$
Max. Acceleration		$0.75\,\mathrm{m\,s^{-2}}$
Max. Velocity		$0.75\,\mathrm{m\,s^{-1}}$
Max. Yaw Rate		$0.5\,\mathrm{rad\,s^{-1}}$
Camera Range	$d_{max}^{planner}$	$3.0\,\mathrm{m}$

TABLE II: Parameters used during the experiments.

Module	Parameter	Notation	Value
Mapping	Voxel size	r	$0.2\,\mathrm{m}$
	Ray length		$2.0\,\mathrm{m}$
	TSDF Integrator		Merged
Planning	Collision Radius	R	$0.5\,\mathrm{m}$
	Local Sampling Count	n_{local}	10
	Local Sampling Radius	r_{local}	$1.5\,\mathrm{m}$
	Edge Length	l_{max}	$1.5\,\mathrm{m}$
	Updating Radius	r_{update}	$3.0\,\mathrm{m}$
	Ray Sub-sampling Factor	f_{sub}	3.0

TABLE III: Parameters of the mapping and planning modules used during the experiments.



(a) NBV - RG. (b) GNN - RG.

(c) NBV - FG. (d) GNN - FG.

Fig. 5: Volummetric representation of the environment after 15 minutes of autonomous exploration. (Single Experiment)

PX4-SITL is based on RotorS simulator [44], a Gazebo-based simulation environment that provides reliable multi-rotor models. However, the advantage of using PX4-SITL is that it allows us to test the complete flight stack, which permits to quickly understand real-world operating conditions and therefore facilitates the transition to experiment with real robotic platforms. The drone is equipped with a stereo camera that provides real-time feedback on the simulated environment by building a map using depth images.

For the sampling-based path planner, we use an open-source modular framework from [26], adapted to PX4-SITL as in [22]. The framework's modularity allows us to test the variation in the exploration performance by changing the decision-making module with respect to different configurations and settings.

The GNN is trained on a 3D problem using the framework provided by [18]. We train with an Nvidia RTX 3090 using the hyperparameters in Table I. The overall training took approximately 3 hours, for an average epoch of 140 seconds. Our inference implementation consists of two components: a GNN *server* and a *proxy* module. The GNN server is a Python module that runs an instance of the trained GNN. The Python module receives the input from a socket, provides it to the GNN, parses the results, and returns the solution back to the caller. The caller is implemented as a C++ module, interacting with the other modules in the planning framework, preparing the input for the GNN, and managing the results. In addition, for a fair comparison, the inference used by our method is implemented on the CPU rather than the GPU.

B. Evaluation Environment

We test the proposed approach to the Receding Horizon NBV planner (RH-NBVP) [8]. The same system constraints are applied for all experiments to both algorithms for a fair comparison. The values of the constraints are summarized in Tables II and III. The decision-making module with whom we compare is the classic policy that chooses the segment with the highest value in its overall subtree to be the next node to reach (Maximum Gain).

We tested the algorithms using two different cost-utility formulations. The first formulation is based on the TSDF map of the reconstructed environment to estimate the impact a viewpoint has on already existing and unknown surfaces, Reconstruction Gain (RG), while the second formulation is a Frontier Gain (FG), which appraises unknown voxels which are next to occupied voxels (frontier voxels).

The selected synthetic environment consists of a 40×40 *m maze*-like environment with a maximum flight height of 2.5 *m*. This scenario is a testbed challenging well-known environment presented in the literature [26], [30]. In addition, under the premise that exploration algorithms will *eventually* explore the whole environment, the duration of the simulated experiments is limited to 15 min to resemble the average autonomy of a standard UAV system.

For each experiment, we save the map every 60 s to track the exploration progress evaluated as the ratio between the explored and the overall space. We perform 12 simulations for each exploration algorithm due to the stochastic nature of the planners. The metric used to evaluate the performance is the volume mapped and the traveled distance needed to map it. In addition, a comparison is made related to the decisions made by each decision-making module at each iteration.

C. Results

From a qualitative point of view, Fig. 5 shows the volumetric representations of the mapped environment at the end of an exploration mission for each of the given gain formulations and decision-making modules. Instead, quantitative data is presented in Fig. 6, depicting the evolution of the experiments where the average behavior is plotted

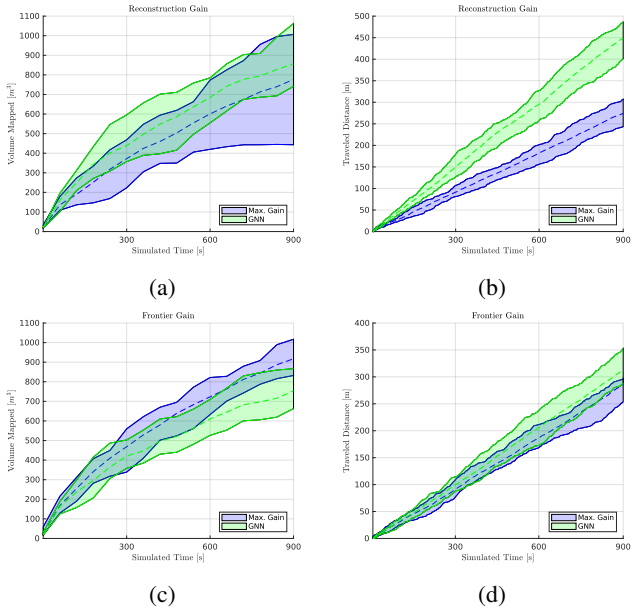


Fig. 6: Evolution of the exploration progress comparing a classic selector policy (Maximum Gain) and a learned policy (GNN). The experiments depict the Average Mapped Volume (left side) and the Traveled Distance (right side) using two definitions of gains: RG (first row) and FG (second row).

with a dashed line within a shade that depicts the variance performance of all the experiments.

Comparing Fig. 5a with 5b, and also Fig. 5c with 5d, it is possible to notice some misplaced voxels in the walls of the maze environment for the experiments with the classic policy. Analyzing the experiments related to the RG formulation in Fig. 6a and 6c, it can be seen that the learned policy manages to map a volumetric volume that matches the amount obtained by the maximum gain policy. However, the learned policy also covers more distance which can explain a better refinement in the resultant map, which explains the lower number of misplaced voxels. In addition, it is important to highlight the consistency (low variability) in the explored mapped volume using the proposed decision-making module, whose performance is more reliable independent of the gain formulation used to address the problem. This behavior can also be observed in the experiments related to the FG formulation, seen in Fig. 6b and 6d, noticing that the GNN makes decisions that cover a larger distance while having a smaller variance in the mapped volume.

The results of the experiments per iteration are summarized in Table IV. On average, in the RG formulation, the selected subsequent configuration by the GNN is 0.471 m longer than the maximum gain policy, while in the FG formulation is 0.083 m longer. Therefore, the learned policy often chooses nodes at a larger spatial distance compared to the classic policy. In addition, considering that the configurations chosen by the GNN are more distant, additional time is needed to reach the goal considering the imposed kinematic constraints. Therefore, the GNN makes an inferior number

	Feature	Max. Gain	GNN
RG	Number of decisions	442.42 ± 33.13	431.83 ± 16.28
	Distance per decision	0.737 ± 0.636 m	1.208 ± 0.804 m
FG	Number of decisions	298.33 ± 7.83	296 ± 13.40
	Distance per decision	1.112 ± 0.795 m	1.195 ± 0.811 m

TABLE IV: Comparison of the Exploration Results with respect to each formulation gain.

of decisions with respect to the Maximum Gain algorithm, while managing to map a similar volume of the environment and refine its surfaces in the same amount of time.

We also notice that in the same way as the classic algorithm, our decision-making module is prone to get stuck in local minima. This is mainly because its decision-making process corresponds to the information given by the local tree at the specific iteration and without considering the global spatial graph. Therefore, there is a missing notion of the overall exploration mission. In addition, the GNN has no memory of its previous decisions, which can be used in the future to improve its performance.

V. CONCLUSIONS

In this paper, we presented a novel strategy for decision-making on a sampling-based online path-planning algorithm. This work presents advancements toward learning strong heuristics in scenarios where no optimal strategies exist such as the exploration of unknown environments.

The exploration tree is a generalized topological data structure that represents the chronology of an exploration mission, embedding relevant information about the states of interest that have yet to be visited. Through our approach, exploiting GNN algorithms trained with RL, the autonomous robot exploration policy is shaped to make decisions at each planning step based on the tree sample available at that iteration. The policy learned is flexible in deciding among a variable number of possibilities generated at each step, different topologies, while preserving a consistent performance independent of the gain formulation with whom the environment is being explored.

The ability of the GNNs to generalize to multiple graph topologies and sizes makes them prone to extend the inference to the whole graph so that decisions can be constantly made continuously reviewing the entire exploration process since its starting. Future research will address the local minima and memory problems by embedding the whole exploration tree into the GNN. Also, we are interested in deploying the proposed approach on an aerial robot and testing it in other autonomous tasks, such as surface reconstruction.

REFERENCES

[1] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3d mapping." in *Robotics: Science and Systems*, vol. 11. Rome, 2015, pp. 3–12.

[2] P. Tripicchio, M. Satler, M. Unetti, and C. Avizzano, "Confined spaces industrial inspection with micro aerial vehicles and laser range finder localization," *International Journal of Micro Air Vehicles*, vol. 10, no. 2, pp. 207–224, 2018.

[3] M. Popović, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran, "Online informative path planning for active classification using uavs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5753–5758.

[4] L. Campos-Macías, R. Aldana-López, R. de la Guardia, J. I. Parra-Vilchis, and D. Gómez-Gutiérrez, "Autonomous navigation of MAVs in unknown cluttered environments," *Journal of Field Robotics*, vol. 38, no. 2, pp. 307–326, 2021.

[5] Y. Kompis, L. Bartolomei, and M. Chli, "Fully autonomous live 3d reconstruction with an mav: Hardware- and software-setup," 2021-12-02, 9th International Conference on 3D Vision (3DV) 2021.

[6] S. Werner, B. Krieg-Brückner, and T. Herrmann, *Modelling Navigational Knowledge by Route Graphs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 295–316.

[7] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[8] 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)*, 2016, pp. 1462–1468.

[9] Y. Zhou, J. Xiao, Y. Zhou, and G. Loianno, "Multi-robot collaborative perception with graph neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2289–2296, 2022.

[10] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 785–11 792.

[11] E. Tolstaya, J. Paulos, V. Kumar, and A. Ribeiro, "Multi-robot coverage and exploration using spatial graph neural networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8944–8950.

[12] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations (ICLR)*, 2018.

[13] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *AAAI*, 2018, pp. 4438–4445.

[14] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[15] B. Kim and L. Shimanuki, "Learning value functions with relational state representations for guiding task-and-motion planning," in *Conference on Robot Learning (CoRL)*, 2019.

[16] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling, "Planning with learned object importance in large problem instances using graph neural networks," in *AAAI Conference on Artificial Intelligence*, 2020.

[17] H. Dai, E. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017, p. 6351–6361.

[18] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations (ICLR)*, 2019.

[19] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2135–2142.

[20] M. Collins and M. Nathan, "Efficient planning for high-speed mav flight in unknown environments using online sparse topological graphs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 11 450–11 456.

[21] T. Dang, C. Papachristos, and K. Alexis, "Autonomous exploration and simultaneous object search using aerial robots," in *IEEE Aerospace Conference*, 2018, pp. 1–7.

[22] E. Herrera-Alarcón, D. Bagheri-Ghavifekr, G. Baris, M. Mugnai, M. Satler, and C. Avizzano, "An efficient object-oriented exploration algorithm for unmanned aerial vehicles," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 330–337.

[23] G. Hardouin, J. Moras, F. Morbidi, J. Marzat, and E. Mouaddib, "Next-best-view planning for surface reconstruction of large-scale 3d environments with multiple uavs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 1567–1574.

[24] S. Song and S. Jo, "Surface-based exploration for autonomous 3d modeling," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4319–4326.

[25] K. Saulnier, N. Atanasov, G. Pappas, and V. Kumar, "Information theoretic active exploration in signed distance fields," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4080–4085.

[26] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.

[27] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-d environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1699–1706, 2019.

[28] P. Zhong, B. Chen, S. Lu, and X. Mengand Y. Liang, "Information-driven fast marching autonomous exploration with aerial robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 810–817, 2022.

[29] L. Schmid, V. Reijgwart, L. Ott, J. Nieto, R. Siegwart, and C. Cadena, "A unified approach for autonomous volumetric exploration of large scale environments under severe odometry drift," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4504–4511, 2021.

[30] D. Duberg and P. Jensfelt, "UFOExplorer: Fast and Scalable Sampling-Based Exploration With a Graph-Based Planning Structure," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2487–2494, 2022.

[31] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 11 2019.

[32] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Robotics: Science and Systems(RSS)*, 2017.

[33] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "Ving: Learning open-world navigation with visual goals," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 215–13 222.

[34] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectgoal navigation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 16 097–16 106.

[35] A. Devo, J. Mao, G. Costante, and G. Loianno, "Autonomous single-image drone exploration with deep reinforcement learning and mixed reality," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5031–5038, 2022.

[36] E. Herrera-Alarcón, M. Satler, M. Vannucci, and C. Avizzano, "Gn-graph: Self-organizing maps for autonomous aerial vehicle planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 721–10 728, 2022.

[37] L. Schmid, C. Ni, Y. Zhong, R. Siegwart, and O. Andersson, "Fast and compute-efficient sampling-based local exploration planning via distribution learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7810–7817, 2022.

[38] A. Khan, A. Ribeiro, V. Kumar, and A. G. Francis, "Graph neural networks for motion planning," 2020. [Online]. Available: https://arxiv.org/abs/2006.06248

[39] C. Yu and S. Gao, "Reducing collision checking for sampling-based motion planning using graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.

[40] F. Chen, J. D. Martin, Y. Huang, J. Wang, and B. Englot, "Autonomous exploration under uncertainty via deep reinforcement learning on graphs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 6140–6147.

[41] F. Chen, P. Szenher, Y. Huang, J. Wang, T. Shan, S. Bai, and B. Englot, "Zero-shot reinforcement learning on graphs for autonomous exploration under uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5193–5199.

[42] H. Zhang, J. Cheng, L. Zhang, Y. Li, and W. Zhang, "H2gnn: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3435–3442, 2022.

[43] R.J.Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, p. 229–256, may 1992.

[44] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Robot operating system (ros): The complete reference (volume 1)," A. Koubaa, Ed. Cham: Springer International Publishing, 2016, pp. 595–625.