Where to Look Next: Learning Viewpoint Recommendations for Informative Trajectory Planning

Max Lodel, Bruno Brito, Álvaro Serra-Gómez, Laura Ferranti, Robert Babuška, Javier Alonso-Mora

***Abstract*— Search missions require motion planning and navigation methods for information gathering that continuously replan based on new observations of the robot's surroundings. Current methods for information gathering, such as Monte Carlo Tree Search, are capable of reasoning over long horizons, but they are computationally expensive. An alternative for fast online execution is to train, offline, an information gathering policy, which indirectly reasons about the information value of new observations. However, these policies lack safety guarantees and do not account for the robot dynamics. To overcome these limitations we train an information-aware policy via deep reinforcement learning, that guides a receding-horizon trajectory optimization planner. In particular, the policy continuously recommends a reference viewpoint to the local planner, such that the resulting dynamically feasible and collision-free trajectories lead to observations that maximize the information gain and reduce the uncertainty about the environment. In simulation tests in previously unseen environments, our method consistently outperforms greedy next-best-view policies and achieves competitive performance compared to Monte Carlo Tree Search, in terms of information gains and coverage time, with a reduction in execution time by three orders of magnitude.**

I. Introduction

Autonomous robots can play a fundamental role in gathering information in critical and dynamic scenarios, such as search and rescue [1], [2] or environmental monitoring [3], [4]. For example, robots can support human emergency responders to locate victims in challenging or dangerous terrain. In such scenarios, environments are often unknown, and autonomous navigation methods must continuously replan actions that maximize the information gathered over long horizons. Moreover, these trajectories must be efficient with respect to time or energy costs.

Long horizon, or *non-myopic*, path planning methods for information gathering and map exploration [3], [5]–[12] suffer from high computational cost and thus long planning times, particularly in complex, obstacle-rich environments. To enable fast online execution, recent works have approached information gathering using deep reinforcement learning (DRL) [2], [13]–[18]. In these methods, a policy learns in offline training to select an action that maximizes the expected information gain of future observations. The policy is usually modeled as a deep neural network that reasons about high-dimensional observations of the agent's surroundings (e.g., obstacles), or its current belief about the environment. However, DRL-based information gathering methods do not explicitly consider constraints for collision avoidance and do not account for the robot's dynamics.

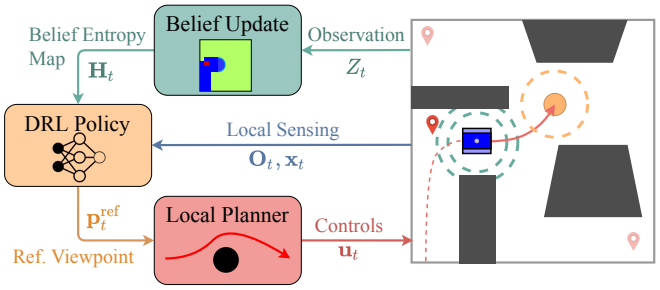


Fig. 1: Conceptual overview of the proposed informative trajectory planning framework. A DRL policy recommends a viewpoint reference to a local planner, based on the robot's current belief about the environment, and local sensory information. The local planner generates a feasible trajectory and executes control commands, leading the robot (depicted in blue on the right-hand side) to reduce the uncertainty about the environment.

In uncertain or dynamic scenarios, it is advantageous to employ an optimization-based local motion planner such as model predictive control (MPC), to generate dynamically feasible and collision-free trajectories and thus safe robot motion [19], [20]. The recent work of [21] combined MPC with a learned subgoal policy for navigation among interacting agents. In this paper, we propose a hierarchical framework for exploring unknown, obstacle-rich environments. Building on the idea of [21], we enhance a local motion planner with a guidance policy trained using DRL. By training in different simulated environments, the DRL agent learns a guidance policy that maximizes information gains from future sensor observations. In particular, the policy is trained to combine its belief about the environment with local observations of obstacles and the robot state for guiding an MPC-based motion planner by recommending a subgoal reference. This *viewpoint reference*, i.e., a subgoal leading towards informative observations, is then used by the MPC planner to generate low-level control commands while ensuring collision avoidance and kinodynamical feasibility of the trajectory.

A. Related Work

1) Planning for Information Gathering: Informative path planning (IPP) methods plan future observation poses that are expected to reduce uncertainty about the environment as efficiently as possible, generally at the expense of computation

This work was supported by the National Police of the Netherlands. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors. L. Ferranti received support from the Dutch Science Foundation NWO-TTW within the Veni project HARMONIA (nr. 18165).

The authors are with the Department of Cognitive Robotics (CoR), Delft University of Technology, 2628CD Delft, The Netherlands, {m.lodel; bruno.debrito; a.serragomez; l.ferranti; r.babuska; j.alonsomora}@tudelft.nl. R. Babuška is also with CIIRC, Czech Technical University in Prague, Czech Republic.

Video: https://youtu.be/qxabfC9I66k

time. Generally, *myopic* and *non-myopic* IPP methods can be distinguished. Myopic methods capitalize on the submodularity property of common IPP objectives such as maximizing mutual information [22]. These methods select their actions greedily either by considering the next best viewpoint at the current time step [23]–[25], or by finding a trajectory leading to the best reachable next viewpoint [26]. While their computation times are generally low, these methods sacrifice efficiency in terms of time or energy required to gather information about the environment due to their short planning horizon.

Non-myopic planning methods, in contrast, attempt to find long-horizon plans that maximize an information-related objective quantifying the cumulative information gain. These methods often rely on tree search algorithms [5]–[9], such as Monte Carlo Tree Search (MCTS) [5]–[7], or global optimization [3], [12]. While being able to find near-optimal paths over long horizons, they suffer from high computational costs due to repeated predictions of possible future observations. This is particularly exacerbated by computationally expensive visibility checks in obstacle-rich environments. The resulting long re-planning times make them unpractical for time-constrained and fast-paced dynamic scenarios. These computational issues are partially addressed by [9], [10], but as the aforementioned methods, they simplify or do not consider the kinodynamic constraints of the robot.

Local motion planners can ensure dynamic feasibility and collision avoidance, but this might result in a trajectory deviating from the planned informative path. Maximizing an information-theoretic objective directly in local trajectory optimization has been proposed for SLAM [27], [28] and grid mapping [11]. This approach requires a differentiable information gain model and is computationally expensive for long planning horizons.

Our proposed framework, in contrast, combines fast online execution times with non-myopic reasoning and explicit dynamic feasibility and is flexible with respect to observation and information model design choices. This is enabled by combining local motion planning with DRL.

2) DRL for Information Gathering: Thanks to their fast execution times and ability to choose actions conditioned on the recent history of observations, DRL-based approaches have the potential to find a suitable trade-off between quickly reacting to new observations and efficient information gathering. A common component of previous DRL-based information gathering approaches [2], [13]–[18], [29] is that the agent's current incomplete knowledge about the environment is formulated as an observation to reason about where informative sensor measurements can be taken. The methods differ in the type of actions being selected, and thus the way the policy interacts with the robot. A common approach is to select from a discrete set of motion primitives [13], [14], [17] for a simplified first-order dynamic model, and directly apply them to the robot. In other works, the learned policy makes higher-level decisions (e.g., by choosing the next frontier [2], subgoal [18], subregion [15] or graph vertex [16], [29] to observe). In [2], [15], [18], that action is executed by a lower-level path planner.

However, none of the mentioned works does explicitly account for how the actions chosen by the DRL agent can result in dynamically feasible, collision-free trajectories. Our method trains a policy to give a high-level local subgoal, or *viewpoint* reference, to a lower-level MPC trajectory planner that can ensure the satisfaction of the robot's constraints. In contrast to the global subgoal policy in [18], the subgoals in our method are restricted to the robot's local surroundings and continuously guide a dynamics-aware trajectory planner. Similar to [14], we use the current knowledge of the global map and local observations as inputs to our policy, but also include the robot's dynamical state to allow for reasoning about the behavior of the underlying MPC.

B. Contribution

The main contributions of this work are the following:

- An informative trajectory planning hierarchical framework combining a viewpoint recommendation policy with receding-horizon trajectory optimization. Our method plans safe and dynamically feasible trajectories while navigating the robot to informative observations.
- A method for training a DRL agent together with a local motion planner, such that the policy learns to guide the motion planner in an obstacle-rich environment and to maximize the cumulative information-theoretic reward.

We present simulation results comparing our method with an MCTS planner and a greedy policy, in terms of the execution time and information gathering performance. We aim at significantly faster execution than MCTS, with little loss of performance, and substantially better performance than with the greedy policy. Additionally, we present qualitative results demonstrating the exploration behavior of our method.

II. Preliminaries

A. Problem Formulation

Consider a robot that has to explore an unknown 2D environment $\mathcal{W} \subset \mathbb{R}^2$ in order to find an unknown number of targets in this environment. The dynamics of the robot are described by a discrete-time model $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ where $\mathbf{x}_t$ is the state of the robot, and $\mathbf{u}_t$ is the control input applied at time step t. We assume that $\mathbf{x}_t$ is observed, e.g., using onboard sensing. We denote the position of the robot in $\mathcal{W}$ at time t by $\mathbf{p}_t = [x_t, y_t]^T$, $\mathbf{p}_t \in \mathcal{W}$. The area covered by the robot at time t is denoted by $\mathcal{O}_t$. The robot must avoid collisions with static obstacles $\mathcal{O}_{\text{obst}} \subset \mathcal{W}$.

When moving in the environment, the robot builds, from sensor observations, a map about possible target locations in the environment. We model this target map as a probabilistic occupancy grid map [30], represented by the random variable $\mathbf{M}$ (see Section II-B). The observation vector is modeled as a random variable Z_t, with a realization denoted by z_t. At each time step t the robot makes an observation Z_t about nearby targets at its current position $\mathbf{p}_t$, and updates its belief about the target map $\mathbf{M}$. Subsequently, the control inputs are computed that move the robot to its next observation pose $\mathbf{p}_{t+1}$.

The objective of the robot is to reduce the uncertainty in the target map $\mathbf{M}$ by making informative observations Z_t. We formalize this objective as maximizing the cumulative mutual information (MI) between the robot's prior about $\mathbf{M}$ at time step t, and the latest measurement Z_t, given the history of measurements until the last time step, $z_{0:t-1}$. The MI quantifies the reduction in uncertainty by making observation Z_t, and it is denoted by $I(\mathbf{M}; Z_t|z_{0:t-1})$ [31].

The informative trajectory planning problem then is to maximize the cumulative MI while ensuring a collision-free, kinodynamically feasible trajectory over a horizon L (the total time-budget for the mission) and given an initial state $\mathbf{x}_0$ and an initial observation z_0:

$$\max_{\mathbf{u}_{0:L-1}} \quad \sum_{t=1}^{L} I(\mathbf{M}; Z_t|z_{0:t-1}) \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{1b}$$

$$\mathcal{O}_t \cap \mathcal{O}_{\text{obst}} = \emptyset \tag{1c}$$

$$Z_t = h(\mathbf{x}_t), \tag{1d}$$

$$\mathbf{x}_t \in \mathcal{X}, \mathbf{u}_{t-1} \in \mathcal{U}, \tag{1e}$$

$$t = 1, ..., L$$

where (1b) is the constraint on the robot dynamical model (Section II-C), (1c) is the collision avoidance constraints, and $\mathcal{X}$,$\mathcal{U}$ are the admissible sets of robot states and control inputs, respectively. Eq. (1d) is the observation model, described in Section II-B.2.

B. Belief Map and Observation Model

1) Belief Map: The target map $\mathbf{M}$ is a discretization of $\mathcal{W}$ in n grid cells, associated with independent Bernoulli random variables $M_i \in \{0, 1\}$, $\forall i \in \{1, ..., n\}$, with 1 indicating target occupancy, and 0 otherwise. The robot's *belief* about the map $\mathbf{M}$ is described by probabilities of target occupancy in each cell i, denoted by $P_{t,i} \coloneqq \mathbb{P}(M_i = 1|z_{0:t})$, and initialized with a uniform prior of $\mathbb{P}(M_i = 1) = 0.5$. Given a new observation z_t, the Bayesian update of $P_{t,i}$ using log odds [30] is:

$$l(M_i|z_{0:t}) = l(M_i|z_{0:t-1}) + l(M_i|z_t), \tag{2}$$

where $l(M_i|z_t)$ is an inverse sensor model [30].

2) Observation Model: To make observations Z_t, the robot is equipped with a sensor that can detect targets up to a distance $d_{\max}$ from the robot and within a field-of-view of 360° and associate it with a cell in the map $\mathbf{M}$. The set of cells visible from position $\mathbf{p}_t$ is denoted by $\mathcal{I}_t$. It only includes cells for which the visibility of its center point is not occluded by obstacles $\mathcal{O}_{obst}$. The observation made at each time step is a vector composed of the individual cell observations of target occupancy, namely $Z_t \in \{0, 1\}^{|\mathcal{I}_t|}$. Eq. (2) is only applied to the cells in $\mathcal{I}_t$ after each observation.

The mutual information between the prior about the target map $\mathbf{M}$ and an observation Z_t is equal to the reduction of the conditional entropy in $\mathbf{M}$ by observation Z_t [31], [32]:

$$I(\mathbf{M}; Z_t|z_{0:t-1}) = H(\mathbf{M}|z_{0:t-1}) - H(\mathbf{M}|Z_t, z_{0:t-1}) \tag{3}$$

where $H(\mathbf{M}) = \sum_{i=1}^{n} H(M_i)$ is the entropy of the target map and $H(M_i)$ are the respective cell entropies.

C. Robot Dynamics

We consider the robot to be modeled by a second-order unicycle model [33]

$$\begin{aligned} \dot{x} &= v\cos\psi \quad \dot{v} = u_a \quad \dot{\psi} = \omega \\ \dot{y} &= v\sin\psi \quad \dot{\omega} = u_\alpha. \end{aligned} \tag{4}$$

which is discretized with sampling time T_S. Thus the state of the robot is described by $\mathbf{x} = [x, y, \psi, v, \omega]^T$, where ψ the heading angle in a global frame, v denotes the robot's longitudinal velocity, and $\omega = \dot{\psi}$ the angular velocity. The control input $\mathbf{u} = [u_a, u_\alpha]^T$ consists of the robot's linear and angular acceleration, respectively.

III. Method

We hierarchically solve the problem in (1) by separating it into a high-level sequential decision-making problem and a local trajectory planning problem. The first aims at determining a reference viewpoint, such that future information gains are maximized based on the current belief following from past observations (Section III-A). The local trajectory planning problem aims at moving the robot towards the recommended viewpoint while ensuring kinodynamic feasibility and collision avoidance (Section III-B). The concept of the proposed framework is depicted in Fig. 1. Our proposed approach builds on [21], extending its task and environment scope for information gathering in obstacle-rich environments.

A. Reinforcement Learning of Viewpoint Recommendations

Our method learns, via reinforcement learning, a policy π that recommends every N_a timesteps a reference position $\mathbf{p}_t^{\text{ref}}$ in the robot's neighborhood (the reference viewpoint) to an MPC motion planner, such that the resulting trajectories of the robot lead to observations that maximize rewards, and eventually result in near-complete coverage of the available information in the environment.

1) Observation: The goal is to learn a policy that uses the robot's own belief about $\mathbf{M}$, and local observations about nearby obstacles $\mathbf{O}_t \in \mathcal{O}_{\text{obst}}$. Both inputs are visualized in Fig. 2. The local obstacle observation $\mathbf{O}_t$ is a binary grid map of obstacles around the robot, given as an $m \times m$ image, centered at the robot's position and aligned with its orientation [14], [34]. Such an egocentric observation improves generalization across different environments. The robot's belief about $\mathbf{M}$ is represented by a map of cell entropies $H(M_i)$, denoted by $\mathbf{H}_t$, that informs the agent about uncertainties in different map regions. An indicator function map $\mathbf{X}_t$ for the agent's position in the map is appended as a second channel to $\mathbf{H}_t$ [2], [15]. Hence, at time step t the RL observation vector $\mathbf{s}_t$ is

$$\mathbf{s}_t = [\mathbf{H}_t, \mathbf{X}_t, \mathbf{O}_t, \mathbf{x}_t]^{\mathrm{T}}, \tag{5}$$

where $\mathbf{x}_t$ is the robot's state defined in Section II-C.

2) Action: The RL action $\mathbf{a}_t \in \mathcal{A} \subset \mathbb{R}^2$ is defined as the relative position δ_t of the viewpoint reference with respect to the robot's current position,

$$\begin{aligned} \mathbf{a}_t &= \delta_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t) \\ \mathbf{p}_t^{\text{ref}} &= \mathbf{p}_t + \delta_t \end{aligned} \tag{6}$$

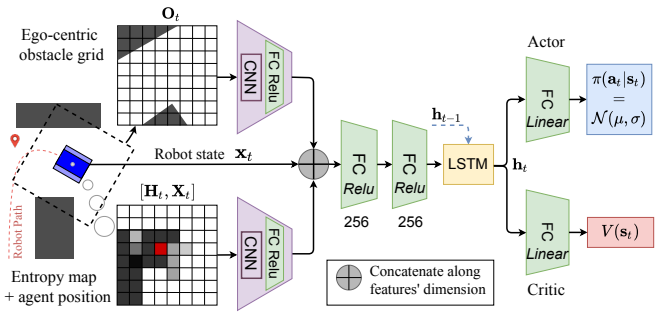


Fig. 2: Proposed policy and value function network, with two encoders processing an ego-centric obstacle grid map $\mathbf{O}_t$ and a two-channel image representing the belief $[\mathbf{H}_t, \mathbf{X}_t]$, respectively. The entropy map $\mathbf{H}_t$ is depicted as a grayscale image, with darker shades corresponding to lower uncertainties. The second channel $\mathbf{X}_t$ is visualized by a red grid cell marking the current agent position. The encoder structure and hyperparameters are as in [34], $\mathbf{h}$ is the LSTM hidden state, and FC refers to a fully-connected layer.

The position increment is constrained inside a square around the robot, such that the continuous action space of our RL method is $\mathcal{A} = \{\ \delta_t \in \mathbb{R}^2 \mid \|\delta_t\|_\infty \leq \delta_{\max}\}$.

3) Reward Function: The main objective of the informative trajectory planning problem (1) is to maximize the information gains. Hence, we define an information-theoretic reward function using the mutual information gained through the observation Z_{t+N_a}, made N_a steps after the last action $\mathbf{a}_t$. Moreover, we add a term r_{pen} penalizing each time step, incentivizing the agent to achieve the coverage goal, terminating the episode, as soon as possible. The reward function is defined as

$$r(\mathbf{s}_t, \mathbf{a}_t) = I(\mathbf{M}; Z_{t+N_a}|z_{0:t}) + r_{\text{pen}}. \tag{7}$$

4) Policy Network Architecture: Fig. 2 depicts our proposed policy network architecture. We employ two CNN models using the architecture and hyperparameters proposed in [34] to encode spatial information in the two image inputs $[\mathbf{H}_t, \mathbf{X}_t]$ and $\mathbf{O}_t$. These encoder networks are each trained by gradients coming from both the policy update loss (Section III-C) and a reconstruction loss generated using a decoder network reversing the encoder operations [34]. Thus, compressed latent representations of spatial features in the local obstacle grid and the entropy map are learned, which the policy can exploit to learn actions that guide the robot around nearby obstacles and to map regions with high uncertainties. The two latent feature vectors are concatenated with the state $\mathbf{x}_t$ of the robot's dynamical model, so that the policy can learn how to guide the MPC planner using viewpoint references with respect to its closed-loop dynamical behavior.

After feeding the full feature vector into two fully-connected (FC) layers, an LSTM layer [35] models the time-dependencies between previous states and the current state. The hidden state of the LSTM is fed to the final actor and critic heads modeled as FC linear layers. We model the policy π as a diagonal Gaussian distribution, i.e. $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = \mathcal{N}(\mu, \sigma^2)$, such that $\delta_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$. The distribution's mean μ and log-standard-deviation $\log\sigma$ are learned by the actor head. The critic head estimates the state-value function $V_\theta^\pi(\mathbf{s}_t) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$ of the current policy, where γ is the discount factor. The subscript θ denotes the current network parameters.

B. Local Planner

We rely on receding-horizon trajectory optimization to generate control commands for the robot satisfying dynamic and collision constraints. For dynamic constraints we employ a second-order unicycle model as defined in Section II-C.

For the collision constraints, we assume the robot's space, $\mathcal{O}_t$, to be a circle with radius r, and each obstacle's space is defined as a polygon. To ensure collision-free motions, first, we compute a linear constraint to ensure that the robot's space does not overlap with static obstacle's space, i.e., $\mathcal{O}_t(\mathbf{x}_t) \cap \mathcal{O}_{\text{obst}} = \emptyset$, at planning step k defined as

$$c_k^{o_j} = {\mathbf{n}_k^{o_j}}^{\mathrm{T}} \mathbf{p}_k \leq b_j - r, \tag{8}$$

where $\mathbf{n}_k^{o_j}$ is the normal vector at the closest point $\mathbf{p}_k^{o_j}$ on the surface of the j-th obstacle and $b_j = -{\mathbf{n}_k^{o_j}}^{\mathrm{T}} \mathbf{p}_k^{o_j}$. To limit the complexity of the optimization problem, we only consider a set of n_{obs} constraints for the closest obstacles. The distance between the robot's position and the j-th linear constraint is computed as:

$$\left\|\mathbf{p}_k, c_k^{o_j}\right\| = \frac{\left|{\mathbf{n}_k^{o_j}}^{\mathrm{T}} \mathbf{p}_k + b_j\right|}{\left\|\mathbf{n}_k^{o_j}\right\|} \tag{9}$$

The DRL policy provides a reference position $\mathbf{p}_t^{\text{ref}}$ (viewpoint) guiding the robot to maximize future rewards. Similarly to [21], we define a terminal cost enabling the robot to reach the provided viewpoint reference:

$$J_N(\mathbf{x}_{t+N}, \mathbf{p}_t^{\text{ref}}) = \left\| \frac{\mathbf{p}_{t+N} - \mathbf{p}_t^{\text{ref}}}{\mathbf{p}_t - \mathbf{p}_t^{\text{ref}}} \right\|_{Q_N}, \tag{10}$$

where $\mathbf{p}_{t+N}$ is the robot's terminal position (at planning step N) and $Q_N = \text{diag}(q_N, q_N)$ is the terminal cost matrix. To generate smooth trajectories, we employ a quadratic penalty on the control commands as a stage cost for planning step k:

$$J_k^{\mathbf{u}}(\mathbf{u}_k) = \|\mathbf{u}_k\|_{Q_u} \tag{11}$$

where $Q_u = \text{diag}(q_a, q_\alpha)$ is the stage cost matrix.

At every time step t, a non-convex optimization problem is solved with planning horizon N under the kinodynamic and collision constraints, given the initial state $\mathbf{x}_t \in \mathcal{X}$:

$$\begin{aligned} \min_{\mathbf{u}_{t:t+N-1}} \quad & \sum_{k=t}^{t+N-1} J_k^u(\mathbf{u}_k) + J_N(\mathbf{x}_{t+N}, \mathbf{p}_t^{\text{ref}}) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \\ & c_{k+1}^{o_j} \leq b_j - r,\ \forall j \in \{1, \ldots, n_{\text{obs}}\}, \\ & \mathbf{u}_k \in \mathcal{U},\ \mathbf{x}_{k+1} \in \mathcal{X},\ \forall k \in \{t, ..., t+N-1\}. \end{aligned} \tag{12}$$

The equality constraint is the discrete-time model of the continuous dynamics model presented in (4).

C. Training Procedure

First, we warm-start the policy training with behavior cloning updates, using the one-step greedy policy outlined in Section IV-B.1, which outputs $\mathbf{p}_t^{\text{ref}}$, in combination with MPC (12) as the expert policy. We define the expert reference viewpoint as $\mathbf{a}_t^* = \mathbf{p}_N^* - \mathbf{p}_t$, where $\mathbf{p}_N^*$ is the last position in the MPC-generated trajectory. For the first N_{SL} policy

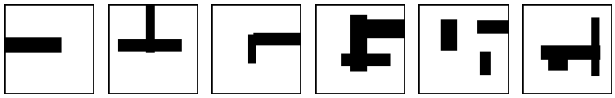


Fig. 3: Examples of the random environments used in training.

steps of the training, we apply $\mathbf{a}_t^*$ as the agent's action and use it as a label to perform supervised learning of the policy π_θ, as described in [21]. Subsequently, the policy is trained with DRL using Proximal Policy Optimization (PPO) [36] until reaching N_{train} policy steps. One policy step yielding a new viewpoint corresponds to N_a timesteps, and MPC is executed at every time step t with the last sampled viewpoint reference. The PPO horizon is $S = L/N_a$ policy steps.

Training and testing are performed in randomly generated environments as depicted in Fig. 3, with the agent initialized at a random position. Random rectangular obstacles are generated and environments, where obstacles block the agent from reaching the entire free space, are omitted. We employ curriculum learning during training [37], increasing the number of obstacles from one to three.

Episodes are terminated after the completion of the information gathering task, or if a maximum number of time steps t_{max} is reached (*failure*). The task is completed when, at time t, the conditional entropy of the belief about the map cells in the free space $\mathcal{W} \setminus \mathcal{O}_{\text{obst}}$, denoted by $\mathbf{M}^{\text{free}}$, is smaller than a predefined ratio of the entropy of the initial belief prior. That is, when $H(\mathbf{M}^{\text{free}}|z_{0:t}) \leq (1-\beta)H(\mathbf{M}^{\text{free}})$, where $\beta \in [0, 1)$ is the share of information that should be gathered by the robot, defining the coverage goal.

IV. Results

In this section, we present quantitative and qualitative simulation results of the proposed method. We compare the performance metrics of our method against two baseline approaches (Section IV-C) and analyze the behavior of the informative trajectory planning method (Section IV-D). The baselines are introduced in Section IV-B, and the simulation setup for training and testing is outlined in Section IV-A.

A. Simulation Setup

The training procedure described in Section III-C builds upon the open-source PPO2 implementation provided by the Stable Baselines framework [38]. The nonlinear optimization problem (12) is solved using the Forces Pro solver [39]. Simulations are run in the environment provided by the open-source "gym-collision-avoidance" package [40]. We train the policy π_θ with N_{proc} processes for rollouts on a desktop computer with an AMD Ryzen 9 CPU and 64 GB of RAM. Table I presents the hyperparameters used.

B. Baselines

We compare the performance against two baseline approaches: one myopic and one non-myopic informative path planning method. Similar to our approach, we use both baselines to compute a reference viewpoint $\mathbf{p}_t^{\text{ref}}$ for the MPC.

1) Myopic Greedy Viewpoint Selection: As a myopic baseline, we use a one-step next-best-view planner similar to [24]. At each time step, we uniformly sample $N_{nbv} = 30$ points $\tilde{\mathbf{p}}_i, \forall i = 1, ..., N_{nbv}$ in the policy's action space $\mathcal{A}$, and evaluate the objective $I(\mathbf{M}; Z(\tilde{\mathbf{p}}_i)|z_{0:t})$ for expected observations $Z(\tilde{\mathbf{p}}_i)$ at these viewpoints. The point with the highest reward is chosen and passed as $\mathbf{p}_t^{\text{ref}}$ to the MPC. This greedy method is also used for warm-starting the training, as explained in Section III-C.

TABLE I: Hyperparameters.

MPC					
Horizon N	15	T_S	0.1 s	q_N	5.0
q_a	0.003	q_α	0.003	–	–
Training					
Horizon S	128	N_a	5	N_{proc}	16
Learning rate	10^{-4}	γ	0.99	N_{epochs}	2
Clip range	0.2	N_{train}	$2 \cdot 10^7$	N_{SL}	10^6
δ_{max}	4 m	t_{max}	640	r_{pen}	-0.1
MCTS Baseline [6]					
N_{tree}	100	N_{sim}	10	H_{MCTS}	4
T_{mp}	1.2 s	u_v	$\{0, 1, 3\}$	C_{UCB}	2.0
u_ω	$\{-\pi/4, -\pi/10, 0, \pi/10, \pi/4\}$			–	–

2) Non-Myopic Monte Carlo Tree Search (MCTS): We use an MCTS planner [5]–[7] as a baseline to find finite-horizon sequences of viewpoints that maximize cumulative information rewards. We build on an open-source Python implementation of Dec-MCTS [6], and use it for single-robot planning. The planner uses a simplified first-order kinematic model of the robot dynamics and a small set of motion primitives as discrete action space. The motion primitives are combinations from different velocity and angular velocity inputs, u_v and u_ω, as given in Table I, with a length of N_a timesteps. The planning horizon is H_{MCTS}, for each replanning, N_{tree} MCTS iterations are performed, and each new leaf node is evaluated with N_{sim} rollouts. The first position in the best plan is passed as $\mathbf{p}_t^{\text{ref}}$ to the MPC. We replan at every time step t, but the planning time does not affect the simulated time due to the sequential implementation. Thus the robot does not have to stop for replanning. Furthermore, our MCTS implementation has access to the global obstacle map for computing rewards of possible future positions.

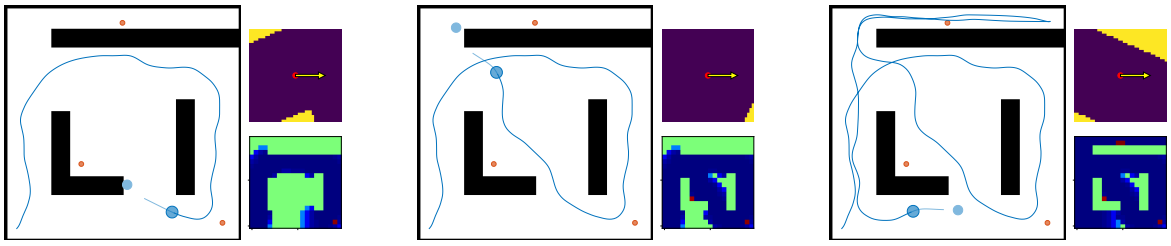
C. Performance Results

This section presents the quantitative performance results of our method and the two baselines. The results, summarized in Table II, are aggregated over a set of random environments for three map complexity levels defined by the number of sampled obstacles. For each number of obstacles, 100 random scenarios are simulated for each of the methods. In each episode, the agent has a maximum of t_{max} to reach the coverage goal of $\beta = 0.9$ before it is considered a *failure*. We quantify the performance by the average cumulated reward over an episode, the percentage of failure episodes, the average travel time, and the average runtime (excluding the MPC) of the three viewpoint recommending methods.

Our method outperforms the greedy next-best-view baseline in terms of average episode rewards, completion time, and failures for all map complexities. The greedy method exhibits a large number of failures because it cannot reason about unexplored areas outside the local surroundings. Thus the robot often revisits already explored areas multiple times

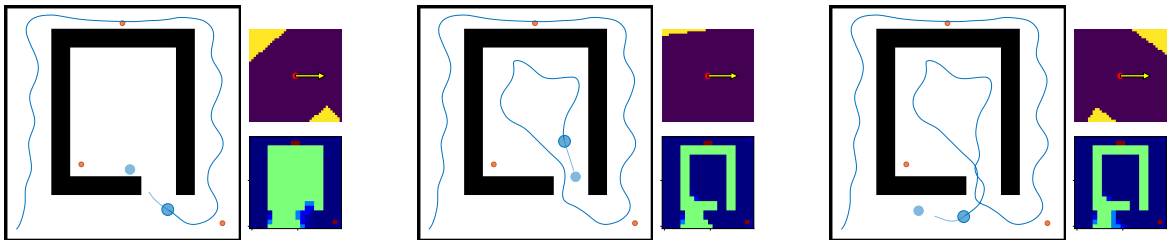
TABLE II: Performance results, aggregated over 100 random maps with $n \in \{1, 2, 3\}$ obstacles (see Section IV-C for details).

	Avg. episode rewards (mean ± std)			% failure			Time until completion [s]			Avg. Runtime [s]
# obstacles	1	2	3	1	2	3	1	2	3	-
MCTS	19.60 ± 1.99	18.24 ± 1.09	16.79 ± 2.00	1	2	2	46.8	50.7	51.7	2.486
Greedy	18.98 ± 2.25	17.50 ± 2.65	15.64 ± 3.75	6	2	8	56.7	61.6	65.7	0.046
Viewpoint Policy (ours)	19.41 ± 0.89	18.03 ± 1.17	16.49 ± 1.41	0	1	1	53.6	55.8	59.6	0.004



(a) $t = 22.5\,\mathrm{s}$ (b) $t = 30\,\mathrm{s}$ (c) $t = 71\,\mathrm{s}$

Fig. 4: Trained policy behavior in an *unstructured* environment of higher complexity than in training, with three timesteps of an episode displayed. The agent effectively explores all areas of the environment and manages to enter and leave the narrow dead-end corridor. The upper-right grid next to each map shows the ego-centric local obstacle observation $\mathbf{O}_t$ of the agent, and the lower-right grid the belief map of the probabilities $P_{t,i}$ of the current belief (Section II-B). The colors in the belief map range from dark blue $P_{t,i} = 0$ to dark red $P_{t,i} = 1$, with the the green areas indicating $P_{t,i} = 0.5$ (the initial value).



(a) $t = 30.5\,\mathrm{s}$ (b) $t = 48\,\mathrm{s}$ (c) $t = 53\,\mathrm{s}$

Fig. 5: Trained policy behavior in a *structured* environment of higher complexity than in training, with three timesteps of an episode displayed. The three snapshots show that the robot is effectively guided into and out of a room-like structure, without visiting areas twice. The setup of the figures is as described in Fig. 4.

instead of moving to unexplored areas to complete coverage. Moreover, our method achieves the lowest percentages of failure episodes and the lowest execution times. Failures of the MCTS planner occur when it determines viewpoint references that are unreachable for the MPC, which our method avoids by training with the MPC. The long runtimes of MCTS are caused by the expensive computation of the set of visible cells $\mathcal{I}_t$ (Section II-B) for a large number of viewpoint candidates during planning, necessary to evaluate their information gain. In contrast, our trained policy π can infer a promising viewpoint reference only from currently available observations. This comes at the cost of suboptimal average rewards and completion time compared to the MCTS planner. Note, however, the MCTS planner's advantageous assumptions (Section IV-B.2), as the long runtime does not affect performance and the global obstacle knowledge enable evaluating rewards for distant positions during planning.

D. Qualitative Analysis

This section analyzes the behavior of our proposed method in two scenarios not used during training and with higher complexity than the training scenarios, in terms of obstacle placement and an increased coverage goal of $\beta = 0.95$. Figures 4 and 5 show the agent path for three different time steps with the recommended viewpoint, the local observation, and belief map of the agent for each scenario. In Figures 4a through 4c, the viewpoint reference leads the agent into the most promising unobserved areas and enables it to enter and leave a narrow dead-end corridor at the top of the map. While not globally optimal, the behavior exhibits an efficient strategy of guiding the robot towards unobserved areas, maximizing information gains, and dealing with difficult environment structures. In Figures 5a through 5c, the robot is able to enter and leave a room-like structure. The policy guides the robot to observe inside the room when reaching the entrance, instead of moving further, and decides to leave the room as soon as almost all available information has been gathered. Subsequently, it guides the robot into the remaining unobserved areas.

V. Conclusions and Future Work

In this paper, we introduced a navigation policy capable of guiding a local trajectory planner towards maximizing the information gathered in an unknown environment. We employed reinforcement learning to learn the information-gathering policy using only locally available observations and previously gathered information. The policy learns to maximize information-theoretic rewards by providing a viewpoint reference that an MPC-based local motion planner uses to generate trajectories respecting the robot's safety constraints. The results show that the learned policy is able to effectively guide the robot through unseen environments, and achieve quantitative performance comparable to an MCTS planner. Moreover, our method can be run at a rate three orders of magnitude faster than the MCTS planner, allowing for quick reactions in dynamic scenarios. Future work will consider a limited field of view and experiments on a real robotic platform.

REFERENCES

[1] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple UAVs," *Int. J. Rob. Res.*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[2] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: exploration in Unknown Cluttered Environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 610–617, apr 2019.

[3] M. Popovic, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran, "Online informative path planning for active classification using UAVs," in *2017 IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5753–5758.

[4] A. Vasilijević, D. Nad, F. Mandi, N. Mišković, and Z. Vukić, "Coordinated navigation of surface and underwater marine robotic vehicles for ocean sampling and environmental monitoring," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 3, pp. 1174–1184, 2017.

[5] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Rob. Auton. Syst.*, vol. 83, pp. 15–31, sep 2016.

[6] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *Int. J. Rob. Res.*, vol. 38, no. 2-3, pp. 316–337, 2019.

[7] T. Patten, W. Martens, and R. Fitch, "Monte Carlo planning for active object classification," *Auton. Robots*, vol. 42, no. 2, pp. 391–421, 2018.

[8] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in *2014 IEEE Int. Conf. Robot. Autom.*, 2014, pp. 6447–6454.

[9] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1025–1032, apr 2018.

[10] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments," in *Robot. Sci. Syst. XVII*, 2021.

[11] 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 *Robot. Sci. Syst. XI*, vol. 11, 2015.

[12] A. A. Meera, M. Popovic, A. Millane, and R. Siegwart, "Obstacle-aware adaptive informative path planning for UAV-based target search," in *2019 Int. Conf. Robot. Autom.*, 2019, pp. 718–724.

[13] H. Jeong, B. Schlotfeldt, H. Hassani, M. Morari, D. D. Lee, and G. J. Pappas, "Learning Q-network for active information acquisition," in *2019 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2019, pp. 6822–6827.

[14] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," in *7th Int. Conf. Learn. Represent. ICLR 2019*, 2019.

[15] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *2018 IEEE Int. Conf. Robot. Autom.* IEEE, 2018, pp. 7548–7555.

[16] A. Viseras and R. Garcia, "DeepIG: Multi-robot information gathering with deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 3059–3066, jul 2019.

[17] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," *J. Guid. Control. Dyn.*, vol. 42, no. 8, pp. 1768–1778, aug 2019.

[18] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *8th Int. Conf. Learn. Represent. ICLR 2020*, 2020.

[19] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic Environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4459–4466, 2019.

[20] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for MAVs in dynamic environments," *IEEE Robot. Autom. Lett.*, 2019.

[21] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4616–4623, 2021.

[22] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Intell. Res.*, vol. 34, pp. 707–755, 2009.

[23] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 1, 2002, pp. 540–545.

[24] H. H. González-Baños and J. C. Latombe, "Navigation strategies for exploring indoor environments," in *Int. J. Rob. Res.*, vol. 21, no. 10-11, 2002, pp. 829–848.

[25] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters," in *Robot. Sci. Syst. I*, vol. 1, 2005, pp. 65–72.

[26] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3D exploration and surface inspection," *Auton. Robots*, vol. 42, no. 2, pp. 291–306, 2018.

[27] C. Leung, S. Huang, N. Kwok, and G. Dissanayake, "Planning under uncertainty using model predictive control for information gathering," *Rob. Auton. Syst.*, vol. 54, no. 11, pp. 898–910, 2006.

[28] A. Ryan and J. K. Hedrick, "Particle filter based information-theoretic active sensing," *Rob. Auton. Syst.*, vol. 58, no. 5, pp. 574–584, 2010.

[29] Y. Wei and R. Zheng, "Informative path planning for mobile sensing with reinforcement learning," in *IEEE Conference on Computer Communications*, 2020, pp. 864–873.

[30] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[31] B. J. Julian, S. Karaman, and D. Rus, "On mutual information-based control of range sensing robots for mapping applications," *Int. J. Rob. Res.*, vol. 33, no. 10, pp. 1375–1392, sep 2014.

[32] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 2005.

[33] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[34] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *2018 IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–8.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint: 1707.06347*, 2017.

[37] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ACM Int. Conf. Proceeding Ser.*, vol. 382, 2009.

[38] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[39] A. Domahidi and J. Jerez, "FORCES professional," Embotech AG, https://embotech.com/FORCES-Pro.

[40] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 3052–3059, 2018.