Exploration-RRT: A multi-objective Path Planning and Exploration Framework for Unknown and Unstructured Environments.

Björn Lindqvist[1], Ali-akbar Agha-mohammadi[2] and George Nikolakopoulos[1].

Abstract— This article establishes the Exploration-RRT algorithm: A novel general-purpose combined exploration and path planning algorithm, based on a multi-goal Rapidly-Exploring Random Trees (RRT) framework. Exploration-RRT (ERRT) has been specifically designed for utilization in 3D exploration missions, with partially or completely unknown and unstructured environments. The novel proposed ERRT is based on a multi-objective optimization framework and it is able to take under consideration the potential information gain, the distance travelled, and the actuation costs, along trajectories to pseudo-random goals, generated from considering the on-board sensor model and the non-linear model of the utilized platform. In this article, the algorithmic pipeline of the ERRT will be established and the overall applicability and efficiency of the proposed scheme will be presented on an application with an Unmanned Aerial Vehicle (UAV) model, equipped with a 3D lidar, in a simulated operating environment, with the goal of exploring a completely unknown area as efficiently and quickly as possible.
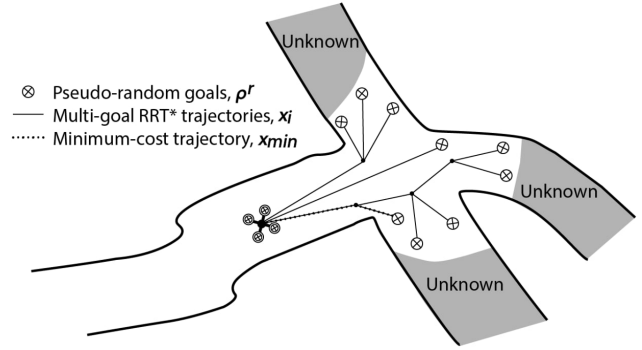
Fig. 1: The ERRT concept. Multiple trajectories are calculated to pseudo-random goals, and the lowest cost trajectory is chosen.

I. INTRODUCTION AND BACKGROUND

The exploration and mapping of unknown and complex structures has been always one of the main application areas of autonomous robots [1], including the inspection of large-scale infrastructures[2], search-and-rescue missions[3], and subterranean exploration [4]. Specifically, the area of subterranean environments, and in general harsh and GPS-denied ones, has been in focus for autonomous exploration missions with the utilization of autonomous robots. This area has gained a lot of attention in the latest years, especially due to the DARPA subterranean challenge[5], where teams of robots are tasked to explore and identify artifacts in complex and unstructured environments. Together with Simultaneous Localization and Mapping (SLAM), the problem of path planning and exploration behavior[6], [7] is one of the most fundamental problems in such applications, which was the main inspiration for this work as part of the NEBULA autonomy developments[8].

Both path planning and exploration have been cornerstone research areas in the field of robotics and have been massively studied as for example in [9], [10], [11], [12]. At the same time, it should be mentioned that there is a very large range of solutions to the problem of total exploration

of unknown areas, including pattern-based solutions[13], frontier[14] or entropy[15] approaches that have been also extensively demonstrated. Occupancy-based path planning (e.g. path planning that relies on a map of occupied and free space) are fundamentally based either on applications of Djikstra's Algorithm[16], with modern examples including improved versions of A*[17] or Jump-point-search[18], or on the Rapidly-Exploring Random Tree (RRT) algorithm [19], that also has numerous moderns improved versions [20][21] and is the core component of the proposed path planner as well. The advantage of RRT is its computational efficiency and the ease of adding functionalities[22] on top of the central planning problem, while its downside is that for limited iterations there is no guarantee of finding the shortest path to the goal.

In many applications, the two problems of exploration and planning are solved separately. In [23] a Deep Reinforcement model selects optimal frontiers and an A* algorithm is applied to find the shortest path. In [24] stochastic differential equations are used for identifying optimal frontiers, while a RRT* path planner computes the path.

Incorporating an exploration behavior into the central planning problem is a research area that gained significant attention lately, with multiple proposed solutions[25][26], where the Next-Best-View planners[27] established the fundamental concept. The flagship modern Next-Best-View planners have seen multiple application scenarios as local path planners [28][29] to name a few. The core concepts of Exploration-RRT (ERRT) are in the same direction as Next-Best-View planners, while its novelty comes from the difference in the algorithmic implementation where ERRT is explicitly solving and evaluating paths to pseudo-random goals versus

[1]The authors are with the Robotics and Artificial Intelligence Team, Department of Computer, Electrical and Space Engineering, Luleå University of Technology, Luleå SE-97187, Sweden. Corresponding Author's email: bjolin@ltu.se.

[2]The author is with the Jet Propulsion Laboratory California Institute of Technology Pasadena, CA, 91109.

an iterative evaluation of RRT-branches until a sufficiently good branch is found. ERRT novelty is also supported from the ability to compute the optimal actuation required to track the generated trajectory, which is done via solving a receding horizon NMPC problem. In the proposed method the NMPC problem is solved by the Optimization Engine[30] an open-source and very fast Rust-based optimization software.

II. CONTRIBUTIONS

This article proposes a novel solution to the combined path planning and exploration problem, based on a minimum-cost formulation problem with pseudo-random generated goals combined with multi-path planning and evaluation. The proposed method evaluates the model-based actuation based on a nonlinear system model along each generated trajectory, which, to the authors best knowledge, has never been included in such a scheme before, together with the information gain and the total distance. Additionally, ERRT considers the full coverage of the unknown map, as long as feasible frontiers or points-of-interest exist. We evaluate the scheme specifically for an UAV platform with an added 3D lidar sensor model, analysing the consistency and efficiency of exploring a completely unknown subterranean-like area. The algorithm is designed around a very general input-output model that allows for the user to on-the-fly change the desired goal and behavior of the planner.

III. METHODOLOGY

A. The Problem

The overarching goal of coupling the path planning and exploration problem is considered in the proposed ERRT framework as the minimization of three quantities namely: The total distance of the 3D trajectory $\boldsymbol{x}$, the actuation required to track the trajectory, where $\boldsymbol{u}$ denotes a series of control actions, and increasing the known space, here considered as a negative cost associated with the information gain $\nu \in \mathbb{R}$ along trajectory $\boldsymbol{x}$. These quantities result in the following minimization:

$$\text{Minimize } J_a(\boldsymbol{u}) + J_d(\boldsymbol{x}) + J_e(\nu) \qquad (1)$$
$$\text{subj. to: } \boldsymbol{x} \in V_{free}$$
$$J_e(\nu) \neq 0$$

with $J_a(\boldsymbol{u})$ denoting the actuation cost, $J_d(\boldsymbol{x})$ the distance cost, $J_e(\nu)$ the exploration, or information-gain cost, which should be non-zero to expand the known space, and $V_{free} \in V_{map}$ denoting the obstacle-free space, with $V_{map} \in \mathbb{R}^3$ as the 3D position-space encompassed by the current map configuration. If solved completely, this would result in the optimal trajectory for the exploration task, in a compromise between quickly discovering more space, limiting actuation based on a dynamic system model, and being *lazy* e.g. moving as little as possible.

B. Proposed Solution

Towards this goal, ERRT proposes a solution composed of four components: pseudo-random goal generation based on a sensor model, a multi-goal RRT* planner, a receding horizon optimization problem (Nonlinear MPC) to solve for the optimal actuation along the trajectory, and finally computing the total costs associated with each trajectory and choosing the minimal-cost solution. An overview of this process is found in Figure 2. Each component will be explained in more detail in III-C. The process of generating many goals, solving the path to each of them, and then computing the total costs associated with each trajectory, turns (1) from a true optimization problem into:

$$\arg\min(J_a(\boldsymbol{u}_i) + J_d(\boldsymbol{x}_i) + J_e(\nu_i))_i, i = 1, \ldots, n_{goals} \quad (2)$$
$$\text{where } \boldsymbol{x}_i \in V_{free}$$

that considers the finding of $\boldsymbol{x}_{\min} \in V_{free}$ that is the $\boldsymbol{x}_i$ trajectory having the lowest cost associated to it , and $n_{goal} \in \mathbb{N}$ is the overall number of path planner goals. This process will converge towards approximating the complete problem in (1), as we increase n_{goal} and optimize the trajectory-generating algorithm (or simply by increasing the number of iterations of the RRT*), as more and more possible solutions are being investigated.

C. The Algorithm

For the sake of notation, let us directly define desired points to explore, or unknown voxels, as $\{U\}$, and a binary 3D grid map $\boldsymbol{G}$ where occupied voxels, $\{O\}$ and unknown voxels $\{U\}$ are set to 1, and the resulting V_{free} is set to 0. Let us also define the measured vehicle state as $\hat{x}$.

1) Goal Generation: The fundamental part of ERRT that allows for the exploration behavior, is the generation of pseudo-random goals ρ^r within V_{map}. At each call to the algorithm, n_{goal} goals are generated, under the conditions of being inside the sensor range of at least one unknown voxel $U \in \{U\}$. Other conditions are $\rho^r \in V_{free}$, and the straight path from ρ^r to the center of U being obstacle-free (including being blocked by other unknown voxels). While many works focus on onboard cameras [28] ERRT considers an onboard 3D lidar, with the advantage in terms of mapping being a 360 ° vision in the *x-y* plane and a long sensor range, and with the drawback of a narrow field-of-view in *z* close to the platform, thus making narrow 3D spaces a challenge. A simplified sensor model of a 3D lidar is shown in Figure 3 assuming the lidar is placed flat on the platform, considering only three parameters: the sensor range $r_s \in \mathbb{R}$, the field-of-view $\theta_s \in \mathbb{R}$ and $l_s \in \mathbb{R}$, describing the size of the sensor array.

Assuming a randomly generated point $\rho^r = [\rho_x^r, \rho_y^r, \rho_z^r] \in \mathbb{R}^3$ and a center-position of an unknown voxel U as $p^u = [p_x^u, p_y^u, p_z^u] \in \mathbb{R}^3$, we can evaluate, by assuming small roll/pitch angles in the UAV case, if ρ^r is inside the space seen by the sensor model if $r_s \geq \sqrt{(\rho_x^r - p_x^u)^2 + (\rho_y^r - p_y^u)^2}$ and $| \rho_z^r - p_z^u | \leq \sqrt{(\rho_x^r - p_x^u)^2 + (\rho_y^r - p_y^u)^2} \tan(\frac{\theta_s}{2}) + \frac{l_s}{2}$.
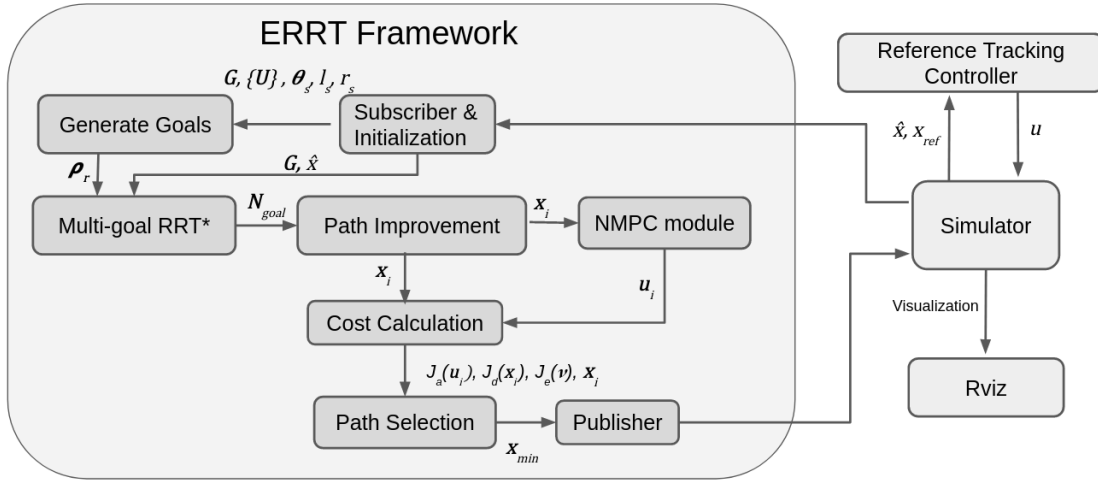
Fig. 2: Exploration-RRT framework and pipeline (left) and simulation architecture (right).
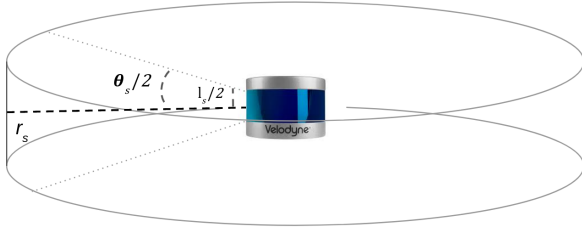

Fig. 3: Simplified sensor model of a 3D lidar. r_s denotes the sensor range and θ_s the field-of-view. l_s denotes the size of the inner scanner array.

Algorithm 1: Goal generation

Inputs: $\boldsymbol{G}, \{U\}, n_{goal}, r_s, \theta_s, l_s$
Result: Generate random goals within sensor range
while $i \leq n_{goal}$ **do**
    ρ^r = random_point
    if *not is_occupied* **then**
        for *j = 0, n_u* **do**
            if *in_sensor_view* **then**
                if *not collision_check* **then**
                    $\boldsymbol{\rho}^r \leftarrow [\boldsymbol{\rho}^r, \rho^r]$
                    i++
Output: ρ_r

This results in Algorithm 1 where $n_u \in \mathbb{N}$ denotes the number of unknown voxels, and $\boldsymbol{\rho}^r \in \mathbb{R}^{n_{goal} \times 3}$ is the list of random goals. The result of this selection process is a list of candidate goals that the multi-goal RRT* can be tasked to plan potential paths to. In the evaluation of the algorithm we will set r_s to 6 m, a very conservative choice as high-end 3D lidar can have ranges of up to 100 m, but it is generally good to under-value sensor ranges in mapping missions to guarantee detection hits. Moreover, θ_s and l_s are set to 32 ° and 0.1 m approximating a Velodyne VLP16 Puck LITE.

2) Multi-goal RRT:* While there are numerous RRT implementations, ERRT uses a 3D RRT* structure. Random points p^r are generated within V_{map} and added as an end point N_{end} to extend the closest graph vertex $N_{closest}$ in a graph

network $\boldsymbol{N}$, if $p^r \in V_{free}$ and there is no collision/obstacles between the p_r and $N_{closest}$. In RRT, this process continues until the goal ρ is reached, generally as $\| \rho - N_{end} \| \leq d$, with d denoting some specified distance. In RRT* the process is instead run for a pre-defined number of iterations, and the shortest generated graph that reached the goal (by the goal condition being satisfied), $N_{goal} \in \boldsymbol{N}$ is selected. The ERRT addition to this baseline framework of RRT algorithms, is the consideration of multiple goals, $\boldsymbol{\rho}^r$ provided by Algorithm 1. This is quite effective in the RRT* framework, as the graphs can be built as before, while only adding the evaluation of the goal conditions for each $\rho^r \in \boldsymbol{\rho}^r$, and after a specified number of iterations extracting the shortest graphs to each goal, which we can denote as $\boldsymbol{N}_{goal} \in \boldsymbol{N}$. Of course, for a limited number of iterations, there is no guarantee that a path can be found to each goal, but running the algorithm with a sufficiently high number of goals, the effect of discarding some goals is reduced. ERRT also adds logical operations to remove some unnecessary vertices in each selected graph, such as for each N_{goal} connecting ρ^r to the first collision-free vertex in N_{goal} and discarding the rest, and removing redundant vertices in the graph. The resulting trajectories are interpolated to have a specified distance between points. The output of the multi-goal RRT* with some trajectory improvements is thus the shortest computed trajectory to each discovered goal, $\boldsymbol{x}_i$, with the non-discovered goals resulting in empty entries. As the 3D lidar field-of-view is not orientation-specific, the RRT* implementation considers the vehicle as a point-like object, and includes only position states. However, in the ERRT demonstration in Section IV, where an UAV model is used in III-C.3, we consider the UAV as $x = [p, v, \phi, \theta]$ where ϕ and θ are the roll/pitch angles of the UAV, let us directly assume trajectory $\boldsymbol{x}$ consists of entries as $x = [p_x, p_y, p_z, 0, 0, 0, 0, 0]$ e.g. only the position-states of the trajectory are set.

3) Trajectory Actuation via NMPC: Many exploration-planning frameworks are lacking a consideration of the predicted actuation a vehicle will use, while following a generated path. Minimizing actuation is a key component

in limiting the energy utilization, while providing easier-to-follow (or more feasible) trajectories. Especially when the ERRT is considered for an UAV case a lower-actuation trajectory implies more stable flying behavior, which stresses localization, mapping, or detection software less as there are less rapid movements. Also, rovers or legged robots often traverse slowly or need extra maneuvering to make tight turns that is not at all included in the path selection if the predicted model-based actuation along the path is not considered. In short: only evaluating information gain and the length of the trajectory (ex. [28]) in trajectory selection misses key aspects that are more properly considered by also evaluating the predicted actuation along the trajectory.

In general, there will be differences between the predicted actuation and the real live actuation of the vehicle, but for comparison any computation of information gain is similarly just a gauge for the approximate information gain as there is no way to know, for example, if more unknown voxels are right behind the frontier and would also be discovered. Similarly, solving a receding horizon NMPC problem where the reference set-points along the prediction horizon are the points in the generated trajectory, will result in the *optimal* actuation based on the provided nonlinear dynamic model of the system with added constraints, and provides an approximation of the required actuation of the real vehicle (or rather the minimal required actuation). The flexibility in defining the NMPC cost function and constraints also allow platform-specific penalties when computing the actuation cost $J_a(\boldsymbol{u})$.

The NMPC approach used in ERRT follows previous works in the literature closely, while the same nonlinear UAV model is used[31], [32], [30], with inputs as $u = [T, \theta_{ref}, \phi_{ref}]$ being references in thrust, roll, and pitch angles, but with a different application in this case. Thus, instead of solving the NMPC problem, with a very short sampling time to compute real-time control signals to a platform, we use a longer sampling time, combined with the interpolation length of trajectory $\boldsymbol{x}_i$, to match a desired predicted behavior and velocity of the vehicle. Let us denote $n_p \in \mathbb{N}^+$ as the prediction horizon, and $k+j|k$ the predicted time step $k+j$ produced at time step $k \in \mathbb{N}^+$. Similarly let $j \in \mathbb{N}^+$ also index the first n_p entries in the path $\boldsymbol{x}$. As an example and based on the UAV case, the cost function, to make each predicted state reach the desired reference set as the entries in trajectory $\boldsymbol{x}$, and at each predicted time step, is:

$$
\begin{aligned}
J_{nmpc}(\boldsymbol{x}_{p,k}, \boldsymbol{u}_k, u_{k-1|k}) = \sum_{j=0}^{N-1} \big(\underbrace{\|x_j - x_{p,k+j|k}\|_{Q_x}^2}_{\text{State penalty}} \\
+ \underbrace{\|u_{\text{ref}} - u_{k+j|k}\|_{Q_u}^2}_{\text{Input penalty}} + \underbrace{\|u_{k+j|k} - u_{k+j-1|k}\|_{Q_{\Delta u}}^2}_{\text{Input change penalty}} \big)
\end{aligned}
\tag{3}
$$

where $Q_x, Q_t \in \mathbb{R}^{8 \times 8}, Q_u, Q_{\Delta u} \in \mathbb{R}^{3 \times 3}$ are positive definite weight matrices for the states, inputs and input change respectively, $x_{p,k+j|k}$ are the predicted states and u_{ref} is the reference input, commonly set as a steady-state input of

the platform. Let us also define input constraints as $u_{\min} \leq u_{k+j|k} \leq u_{\max}$. This leads to the following optimization problem:

$$
\underset{\boldsymbol{u}_k, \boldsymbol{x}_{p,k}}{\text{Minimize}} \ J_{nmpc}(\boldsymbol{x}_{p,k}, \boldsymbol{u}_k, u_{k-1|k}) \tag{4a}
$$
$$
\text{subj. to: } x_{p,k+j+1|k} = \zeta(x_{p,k+j|k}, u_{k+j|k}), j \in \mathbb{N}_{[0, n_p-1]},
$$
$$
u_{\min} \leq u_{k+j|k} \leq u_{\max}, j \in \mathbb{N}_{[0, n_p-1]},
$$
$$
x_{p,k|k} = \hat{x}_k,
$$

with $\zeta(x_{p,k+j|k}, u_{k+j|k})$ defining the discrete state model of the platform. The optimization problem is solved, for each generated trajectory from the multi-goal RRT*, to compute $\boldsymbol{u}_i$ with the Optimization Engine [30], following a *single-shooting* approach. In the following evaluation we solve the problem with $n_p = 50$. Since the actuation is solved as a receding horizon problem, only n_p predicted time steps can be considered and as such, we are solving for the actuation $\boldsymbol{u}$ only for the n_p first entries in $\boldsymbol{x}$. In the case where the trajectory length is lower, the last entry is simply repeated. How far into the trajectory this limit of only considering n_p entries is, depends on the interpolation of $\boldsymbol{x}_i$ and the sampling time T_s of the NMPC. For the following evaluation case we use a sampling time of $0.5\,\mathrm{s}$ and an interpolation length $0.75\,\mathrm{m}$, meaning that the desired platform's velocity to predict and optimize actuation for is $1.5\,\frac{m}{s}$. We should also note that based on the optimal actuation vector $\boldsymbol{u}$ and the initial measured state vector $\hat{x}$, we can compute the full-state dynamic trajectory $\boldsymbol{x}_p$ up until the N:th entry, although despite the NMPC trajectory reference tracking, there is no guarantee that $\boldsymbol{x}_p$ is obstacle-free until the obstacle avoidance, based on the known map, is integrated into the NMPC framework. As such, in the following evaluation, we shall stick with a position-trajectory $\boldsymbol{x}$ to guarantee completely obstacle-free paths and only use the NMPC module solution to calculate the predicted actuation cost $J_a(\boldsymbol{u})$ along the trajectories, defined by the last two terms in 3, the input cost and the input rate cost.

4) Cost Calculation: The final step of the ERRT algorithm is to evaluate each computed path in accordance with (2). By denoting the number of entries in $\boldsymbol{x}_i$ as n_i, the distance cost of each trajectory is easily computed as the sum:

$$
J_d(\boldsymbol{x}_i)_i = K_d \sum_{j=1}^{n_i} \| p_{i,j} - p_{i,j-1} \| . \tag{5}
$$

Based on the predicted actuation $\boldsymbol{u}_i$ along the trajectory, the actuation cost can be computed by feeding the actuation vector back into the relevant parts of the NMPC cost function. For the presented case it is

$$
J_a(\boldsymbol{u}_i)_i = \sum_{j=1}^{n_p} \|u_{\text{ref}} - u_{i,j}\|_{Q_u}^2 + \|u_{i,j} - u_{i,j-1}\|_{Q_{\Delta u}}^2 \tag{6}
$$

with $u_{\text{ref}} = [9.81, 0, 0]$, the input that describes no movement for the UAV, with the thrust value of 9.81 compensating for gravity. The total information gain ν is calculated by evaluating the information gain, at each point in the trajectory,

as each unknown voxel within sensor view, and without obstructed line-of-sight and removing any duplicates (voxels seen at multiple points in the trajectory). For clarity the process is seen in Algorithm 2.

Algorithm 2: Information Gain

Inputs: $\boldsymbol{G}, \{U\}, \boldsymbol{x}_i, n_{goal}, r_s, \theta_s, l_s$
Result: Information gain along the trajectory $\boldsymbol{x}_i$
for $j = 0,\ n_i$ **do**
    for $k = 0,\ n_u$ **do**
        if *in_sensor_view* **then**
            if not *collision_check* **then**
                seen_unknown $\leftarrow$ [seen_unknown, U_k]
ν = length(remove_duplicates(seen_unknown))
Output: ν

Once ν is computed, as the total number of unknowns that will be in sensor range by following $\boldsymbol{x}_i$, the exploration cost is then computed as:

$$J_e(\nu) = -K_\nu \nu \qquad (7)$$

with K_ν denoting a gain representing the relative emphasis on maximizing the information gain. With $J_a(\boldsymbol{u_i})_i, J_e(\nu_i)_i, J_d(\boldsymbol{x}_i)_i$ calculated for $i = 0, \ldots n_{goal}$ (2) can be evaluated and the $\boldsymbol{x}_i$ related to the minimum-cost solution and denoted by $\boldsymbol{x}_{\min}$ is selected as the final result of the algorithm.

5) ERRT input structure: The overall ERRT framework has been implemented in ROS [33], with a custom message including all input parameters to the algorithm, which are summarized in Table I.

Occupied voxels	$\{O\}$
Unknown voxels	$\{U\}$
Vehicle state	$\hat{x}$
Number of goals	n_{goal}
Sensor model parameters	r_s, θ_s, l_s
Cost parameters	$K_d, K_\nu, Q_u, Q_{\Delta u}$
RRT* iterations	$iter$
Grid resolution	g_{res}

TABLE I: ERRT Initialization Parameters.

At every call to the algorithm the grid map **G** is initialized based on point clouds of occupied and unknown voxels, and the user is free to change resolution, size of the map, or any other parameter at every call to the algorithm to meet the desired mission specifications. After calculation is complete the node publishes the computed trajectory. The very general input model was one of the motivations for developing ERRT, allowing for example high-resolution exploration of the local area, and low-resolution computation of what area to explore next, by the same planner with different inputs. The process described in Section III can be summarized in Algorithm 3.

IV. RESULTS

The proposed method is evaluated in a simulated environment separated from other mapping or frontier-generating

---

Algorithm 3: The ERRT algorithm.

Inputs: $\boldsymbol{G}, \{U\}, n_{goal}, r_s, \theta_s, l_s, \hat{x}$
Result: Minimum-cost trajectory $\boldsymbol{x}_{min}$
$\boldsymbol{\rho}^r$ = generate_goals($\boldsymbol{G}, \{U\}, n_{goal}, r_s, \theta_s, l_s$)
$\boldsymbol{N}_{goal}$ = multigoal_rrt($\boldsymbol{\rho}^r, \boldsymbol{G}$)
$\boldsymbol{x}_i$ = trajectory_improvement($\boldsymbol{\rho}^r, \boldsymbol{N}_{goal}, \boldsymbol{G}$)
$\boldsymbol{u}_i$ = NMPC_module($\boldsymbol{x}_i, \hat{x}$)
$(J_a, J_d, J_e)_i$ =
   cost_calc($\{U\}, \boldsymbol{G}, \boldsymbol{x}_i, \boldsymbol{u}_i, K_d, K_\nu, Q_u, Q_{\Delta u}$)
$\boldsymbol{x}_{min}$ = path_selection($(J_a, J_d, J_e)_i, \boldsymbol{x}_i$)
Output: $\boldsymbol{x}_{min}$

---

software, that considers a nonlinear dynamic UAV model with added small magnitude localization noise, and a sensor model that mimics the model described in III-C.1 with a 1 m longer sensor range for a voxel to be considered discovered than what is considered in the ERRT. The map is initialized with the full space as undiscovered, except a small area around the starting location, and as the center of a voxel comes into sensor range, it is set either as free space or as occupied. As the planner is ROS-integrated, the exploration process can be easily visualized in Rviz[34]. It should be noted that for the sake of visualization, the occupied voxels are always depicted. A full-state reference tracking controller, tuned to approximately match the desired predicted behavior as stated in III-C.3, follows the generated path, until the goal is reached and the path is recalculated.

The considered evaluation environment has been generated to mimic a subterranean cave area with multiple rooms, a larger void area, and several small hard-to-reach nooks and passages, that encompasses a total of 27x27x4 m^3, with a grid size of 1 m. The ERRT algorithm was executed for a total of twenty times in the simulated area, ten times with a greedy tuning prioritizing information gain over minimizing distance and actuation and a more conservative tuning with higher costs on distance and actuation as compared to information-gain. Figures 7 and 8 present time sampled images through the evolution of the mission from one of the greedy runs, displaying all the generated paths at critical times during the simulation. We evaluate the simulations in terms of the total distance travelled, and the time required to complete the task, considering both at 90% of voxels discovered as a gauge for the effective volumetric gain, and at full completion.

The results for the greedy tuning can be found in Figure 4 and for the conservative in Figure 5. As it can be observed, a majority of the exploration time is devoted to cleaning up voxels that were ignored along the way due to maximizing the information gain, and exploring the final hard to reach areas to achieve complete coverage of the area. In Figure 6 the volumetric gain (or information gain) from one of the runs is depicted, which also represents the behavior of maximizing the total explored volume as quickly as possible. Both configurations show very consistent results over multiple executions, despite the very different behavior for each tuning, with the greedy tuning over-all
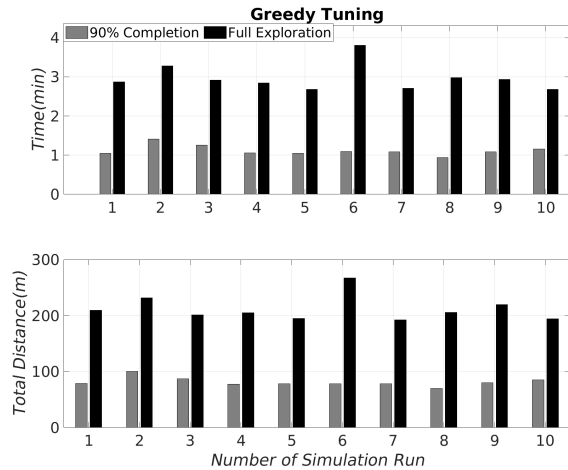
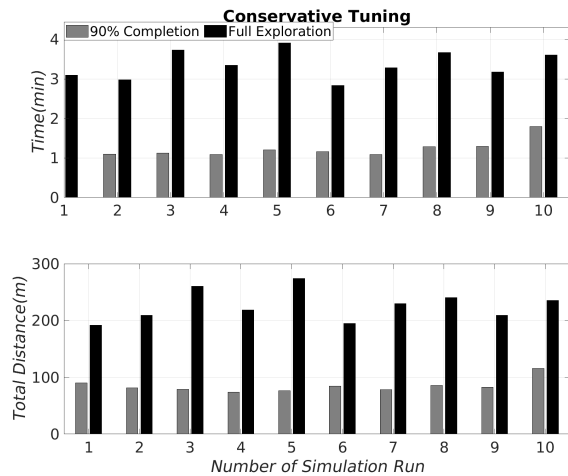Fig. 4: Exploration time and total distance travelled for the greedy tuning of Exploration-RRT.

Fig. 5: Exploration time and total distance travelled for the conservative tuning of Exploration-RRT.

Fig. 6: Volumetric Gain from one of the greedy runs.

performing better. This is naturally the case in a completely unknown area, as maximizing the information gain is of paramount importance and the exploration missions are often time-constrained by the platform that further promotes the greedy behavior. The average translation speed of the platform was 1.2 m/s, controlled by the interpolation of the trajectories, and the tuning of the reference tracking controller, while mimicking the slower pace of a fully autonomous UAV in a subterranean field application. A video compilation of different exploration executions in the evaluation environment can be found here: **https: //drive.google.com/file/d/1v3vg3Z9iB2DR-Oec3MxWUg_ 39d3lIj1F/view?usp=sharing**.

The ERRT algorithm was running 1500 RRT* iterations and the n_{goal} was set to 40, which resulted in solver times of 0.3-0.9 s, averaging at 0.6 s, of which around 0.1 s was consumed by the NMPC module (around 2.5 ms per goal). Although the framework is set up as to optimize over the full trajectory, it needs to be speed up to enable an online
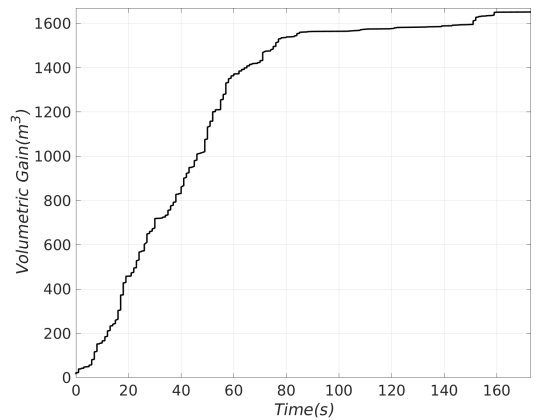
implementation where trajectories are recalculated at each execution step, mainly related to the multi-goal RRT*.

V. FUTURE WORK

From the algorithmic perspective, there are two main directions of future work: speeding up the core multi-goal RRT* framework which would expand the planner's application areas, and adding integrated collision avoidance that considers the size-radius of the platform both in terms of obstacle enlargement in the main RRT* planner and in the NMPC optimization problem to be able to guarantee the actuator-based paths x_p are completely obstacle free. Another interesting direction is adding negative costs related to moving through areas with well-defined features, which are commonly tracked by SLAM software [35], as doing so could improve localization accuracy. This would allow the planner to also consider the *localization cost*, and can be implemented into the framework in a very straight-forward way assuming that the locations of such features is provided by the SLAM software. From the implementation perspective we aim to release the framework as an open-source ROS package, and evaluate it with mapping, frontier-generation, and reactive obstacle avoidance software in the loop.

VI. CONCLUSIONS

This paper has presented a novel algorithm for combined exploration and path planning behavior towards the goal of considering the path planning and information-gain (exploration) as a coupled problem. In the selected initial evaluation environment, the algorithm quickly and efficiently explored all unknown areas of the map, showing a consistent behavior over multiple exploration runs and without failure to complete the task of complete coverage.

REFERENCES

[1] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai, "A survey on inspecting structures using robotic systems," *International Journal of Advanced Robotic Systems*, vol. 13, no. 6, p. 1729881416663664, 2016.

[2] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, and G. Niko-lakopoulos, "Cooperative coverage path planning for visual inspection," *Control Engineering Practice*, vol. 74, pp. 118–131, 2018.
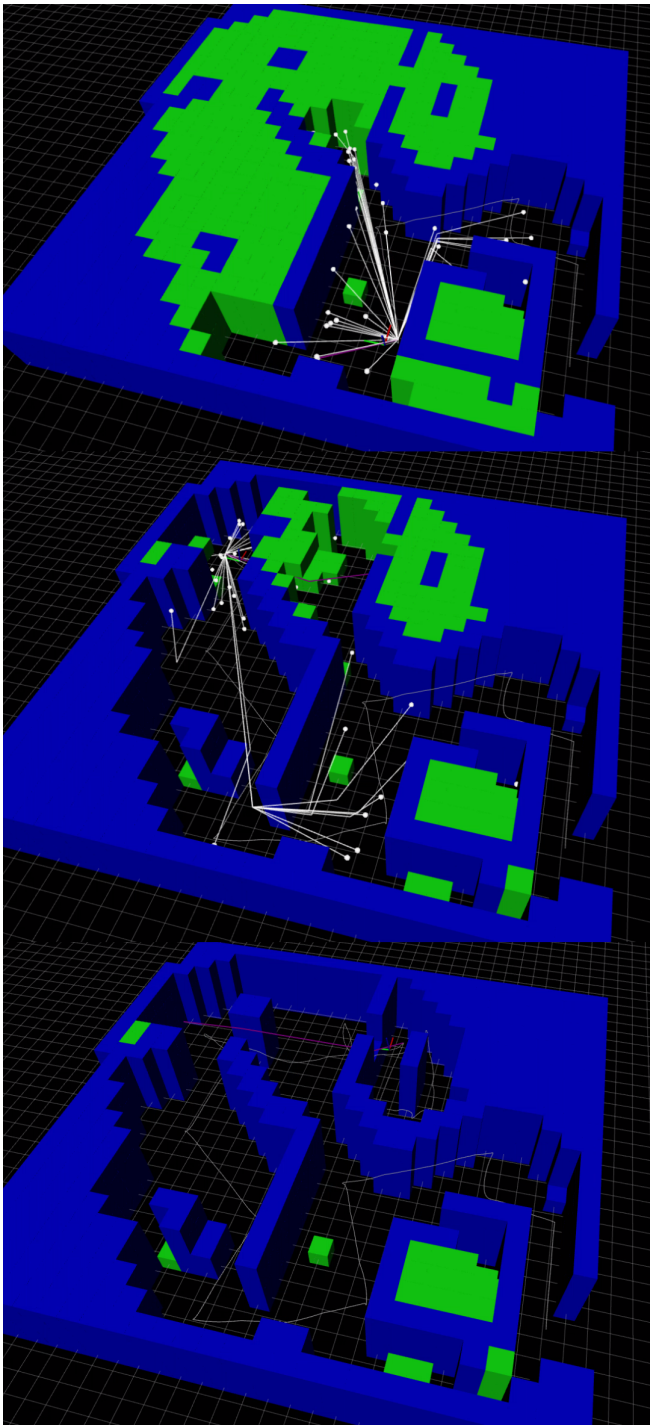
Fig. 8: Exploring the last unknown area.

Fig. 7: Selected point during exploration. Unknown areas (green voxels), all generated paths (white), selected (lowest cost) path (purple), total exploration path (grey). a) greedily maximizing information gain, b) selecting to ignore some unexplored voxels to go to the more information-dense area, c) cleaning up after exploring high-information open-areas.
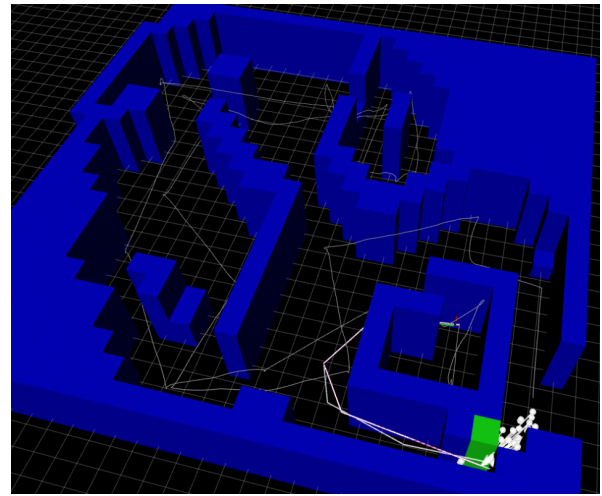
[3] Z. Beck, W. Teacy, N. Jennings, and A. Rogers, "Online planning for collaborative search and rescue by heterogeneous robot teams," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. Association of Computing Machinery, 2016.

[4] C. Kanellakis, S. S. Mansouri, G. Georgoulas, and G. Nikolakopoulos, "Towards autonomous surveying of underground mine using mavs," in *International Conference on Robotics in Alpe-Adria Danube Region*.

Springer, 2018, pp. 173–180.

[5] DARPA. Subterranean challenge (SubT). [Online]. Available: https://www.subtchallenge.com/

[6] M. Palieri, B. Morrell, A. Thakur, K. Ebadi, J. Nash, A. Chatterjee, C. Kanellakis, L. Carlone, C. Guaragnella, and A.-a. Agha-mohammadi, "Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 421–428, 2020.

[7] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Agha-mohammadi, "Plgrim: Hierarchical value learning for large-scale exploration in unknown environments," *arXiv preprint arXiv:2102.05633*, 2021.

[8] J. P. Laboratory. Team costar.

[9] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based uav path planning," *Knowledge-Based Systems*, vol. 158, pp. 54–64, 2018.

[10] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3d path planning algorithms," *Journal of Control Science and Engineering*, vol. 2016, 2016.

[11] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot exploration of communication-restricted environments: A survey," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 48–57, 2017.

[12] 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.

[13] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "Bsa: A complete coverage algorithm," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2040–2044.

[14] C. Zhu, R. Ding, M. Lin, and Y. Wu, "A 3d frontier-based exploration tool for mavs," in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2015, pp. 348–352.

[15] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, "Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 487–494.

[16] M. Noto and H. Sato, "A method for the shortest path search by extended dijkstra algorithm," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 3. IEEE, 2000, pp. 2316–2320.

[17] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, and H. Youssef, "Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments," *International Journal of Advanced Robotic Systems*, vol. 14, no. 2, p. 1729881416663663, 2017.

[18] F. DuchoÈ, A. Babineca, M. Kajana, P. BeÈoa, M. Floreka, T. Ficoa, and L. Jurišicaa, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[19] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[20] O. Adiyatov and H. A. Varol, "A novel rrt*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2017, pp. 1416–1421.

[21] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[22] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "Efficient sampling-based motion planning for on-road autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1961–1976, 2015.

[23] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[24] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3d exploration with a micro-aerial vehicle," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 9–15.

[25] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 179–185.

[26] Y. Wang, M. Ramezani, and M. Fallon, "Actively mapping industrial structures with information gain-based planning on a quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8609–8615.

[27] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[28] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[29] T. Dang, F. Mascarich, S. Khattak, H. Nguyen, H. Nguyen, S. Hirsh, R. Reinhart, C. Papachristos, and K. Alexis, "Autonomous search for underground mine rescue using aerial robots," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–8.

[30] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," *arXiv preprint arXiv:2003.00292*, 2020.

[31] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3556–3563.

[32] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001–6008, 2020.

[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5.

[34] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: a toolkit for real domain data visualization," *Telecommunication Systems*, vol. 60, no. 2, pp. 337–345, 2015.

[35] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.